THESIS

COVARIANCE REGULARIZATION IN MIXTURE OF GAUSSIANS FOR HIGH-DIMENSIONAL IMAGE CLASSIFICATION

Submitted by

Daniel L Elliott

Department of Computer Science

In partial fulfillment of the requirements

for the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Spring 2009

COLORADO STATE UNIVERSITY

February, 2009

WE HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER OUR SUPERVISION BY DANIEL L ELLIOTT ENTITLED COVARIANCE REGULARIZATION IN MIXTURE OF GAUSSIANS FOR HIGH-DIMENSIONAL IMAGE CLASSIFICATION BE ACCEPTED AS FULFILLING IN PART REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE.

Committee on Graduate Work

Committee Member

Committee Member

Adviser

Department Head

ABSTRACT OF THESIS

COVARIANCE REGULARIZATION IN MIXTURE OF GAUSSIANS FOR HIGH-DIMENSIONAL IMAGE CLASSIFICATION

In high dimensions, it is rare to find a data set large enough to compute a non-singular covariance matrix. This problem is exacerbated when performing clustering using a mixture of Gaussians (MoG) because now each cluster's covariance matrix is computed from only a subset of the data set making the presence of a sufficient amount of data even less likely.

The objective of this Thesis is to study the effect of performing a MoG where the covariance matrix is regularized in high-dimensional spaces. Furthermore we present several long-standing algorithms as MoG with various forms of covariance regularization. From this new perspective of MoG with covariance regularization, we begin taking a look at the effect of covariance regularization on MoG classification accuracy. Our secondary objective is to use our results to begin a discussion about the merits of using increased amounts of covariance for MoG on high-dimensional data.

We experiment with three different covariance regularization methods What we find is that the simplest form of covariance regularization experimented with in this Thesis results in the best classification of our natural image data sets. Our experiments clearly show that, as the dimensionality of the data decreases, so does the desired or necessary level of covariance regularization.

Daniel L Elliott
Department of Computer Science
Colorado State University
Fort Collins, Colorado 80523
Spring 2009

TABLE OF CONTENTS

Chapter 1

Introduction

In high dimensions, it is rare to find a data set large enough to compute a non-singular covariance matrix. This problem is exacerbated when performing clustering using a mixture of Gaussians (MoG) because now each cluster's covariance matrix is computed from only a subset of the data set making the presence of a sufficient amount of data even less likely. Even when clustering a data set with many times more data points than dimensions it is possible that a particular cluster will be assigned fewer data points than dimensions causing that cluster to have a singular covariance matrix.

Computer vision is one such domain where there is rarely enough data to compute a full-rank covariance matrix. As a result the mixture of Gaussians model has been largely neglected in the literature in favor of methods that dramatically reduce the number of parameters to be learned and are estimates to a mixture of Gaussians with full covariance. When MoG is applied per se the covariance matrices are typically constrained to be diagonal.

This Thesis proposes a mixture of Gaussians model that uses covariance regularization methods to improve upon the covariance matrix computed using the expectation maximization (EM) update equations. The use of covariance regularization allows each Gaussian in the mixture to model its data with something more closely resembling the actual, underlying covariance matrix which is almost certainly not a diagonal matrix.

We experiment with our algorithm using three different covariance regularization methods. The simplest method combines the identity matrix with a fractional multiple of the empirical covariance matrix. The next simplest method combines a fractional multiple of the diagonal of the empirical covariance matrix with a fractional multiple of the empirical covariance matrix. The most complicated form of covariance regularization uses a method somewhat similar to principal component analysis to create a covariance matrix which, in experiments by the authors, has eigenvalues which are closer to those of the actual, underlying covariance matrix than those of the empirical covariance matrix. The two simpler methods are known as shrinkage methods and have been in existence for quite some time. The third, most complex, method is a new algorithm which uses something called the sparse matrix transform.

In our application, covariance regularization is applied after the M-step of the EM algorithm for a mixture of Gaussians [4, 5]. This is done by using the covariance matrix computed via the EM update equations as the empirical covariance matrix with the two shrinkage methods and the probabilistically weighted data with the sparse matrix transform method.

The objective of this Thesis is to formally present a new method of performing a MoG where the covariance matrix is regularized. Furthermore we present several long-standing algorithms as MoG with various forms of covariance regularization. From this new perspective of MoG with covariance regularization, we begin taking a look at the effect of covariance regularization on MoG classification accuracy using the three proposed covariance regularization methods.

As we discussed above, and shown in Section 2.4, several popular image classification techniques can be viewed in terms of a mixture of Gaussians with covariance regularization. Our secondary objective is to, through our versions of MoG, use our results to begin a discussion about the merits of using increased amounts of covariance for MoG on high-dimensional data. Even those works that do not consider their ties to MoG through covariance regularization provide very little in the way of comparison against MoG techniques. The shrinkage method of covariance regularization is a very direct estimate of the empirical covariance matrix and, therefore, is a good vehicle to investigate whether or not additional covariance parameters in the model is helpful in image classification.

What we find is that the simplest form of covariance regularization experimented with in this Thesis results in the best classification of our natural image data sets. Shrinkage involves a single configurable parameter that controls the amount of regularization applied to the covariance matrix. Our experiments clearly show that, as the dimensionality of the data decreases, so does the desired or necessary level of covariance regularization.

1.1 Thesis overview

In this Chapter we briefly described the approach proposed in this Thesis and took a peek at the primary result of this Thesis. In the next Chapter we cover the mathematical background of our proposed approach, mixture of Gaussians and expectation-maximization in particular, and we review related work including pointing out previously ignored connections between these related works through a mixture of Gaussians with covariance regularization. In Chapter 3, we cover the specific details of our proposed algorithm with pseudo code and the additional mathematical background to understand our particular addition to mixture of Gaussians. In Chapter 3 we also explain the experimental methodology used. In Chapter 4 we present the results from a series of experiments comparing our three MoG versions on data of varying dimensionality. Chapter 5 begins with a discussion of what we learned from these experiments. Finally, we provide a summary of the contributions of this Thesis, some of its deficiencies, and areas of future work.

Chapter 2

Background

In this Chapter we begin by reviewing the mathematical background behind the mixture of Gaussians – the backbone of our proposed algorithm. We begin with principal component analysis which is both a covariance regularization technique and a common pre-processing technique for image classification problems. We then move on to the mixture of Gaussians model and the concepts behind the derivation of the expectation maximization update equations. We then provide a high level description of covariance regularization. Finally we take a look at a brief history of the mixture of Gaussians model making novel connections between popular variations of mixture of Gaussians and their covariance regularization roots.

2.1 Principal component analysis

Principal component analysis (PCA) is a very widely used technique across many domains. PCA is a method of finding a set of vectors, U, such that $\frac{1}{N}\mathcal{X}\mathcal{X}^T U = \Lambda U$ where $\mathcal{X}$ is a matrix of mean-subtracted data points, Λ is a diagonal matrix, U is an orthonormal matrix, and N is the number of data points in $\mathcal{X}$. The column vectors of U, known as the principal components, are chosen to be those which maximize the variance of $\mathcal{X}$ captured in the subspace defined by the columns of U, also known as the PCA subspace. The elements of Λ correspond to the amount of variance captured by their respective principal components. Typically U is taken to have fewer columns than data points in $\mathcal{X}$; hence PCA is a data reduction technique.

The columns of U, the principal components, are the eigenvectors of $\frac{1}{N}\mathcal{X}\mathcal{X}^T$ and the diagonal elements of Λ are the corresponding eigenvalues. PCA is synonymous with the Karhunen-Loève (KL) expansion [27].

PCA can also be considered a covariance regularization technique. Moghaddam and Pentland were the first to point out that PCA can be used to compute a Gaussian or regularize one if reducing the dimensionality of the data by throwing away some of the principal components [31].

In domains like vision it is rare that one can compute enough principal components to prevent dimensionality reduction. There can only be at most $N-1$ non-zero eigenvalues for a data set [27]. Keeping all $N-1$ possible principal components results in the loss of no data: projecting the data back into the original space would result in the perfect reconstruction of the data. In our experiments we use PCA to reduce the dimensionality of the data which is a well established pre-processing technique in computer vision.

2.2 Mixture of Gaussians

Mixture of Gaussians (MoG) is historically the most used and researched mixture model. Its strength comes from their ability to approximate any density function. The data used in this Thesis are almost certainly not generated using a mixture of Gaussians, but we take advantage of this ability to represent a probability density function that classifies the data.

A mixture of Gaussians is just that: multiple Gaussians distributions combining to model a set of data points. A Gaussian, or multivariate normal, distribution models data using a mean vector and a covariance matrix. The probability of a data point is determined by the formula

$$p(x|\mu,\Sigma) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}. \tag{2.1}$$

Here x is a data point, μ is the mean, or location of the center of the Gaussian, and Σ is the covariance matrix which determines the shape of the Gaussian. A mixture of Gaussians has just one additional parameter: $\pi_c \geq 0$ which is the mixing parameter and $\sum_{c\in C}\pi_c = 1$. The mixing parameter, as we will see below, is simply a function of the number of data points assigned to that cluster. Each Gaussian in the mixture has its own μ and Σ so they are each located over different data points and have different shapes to fit that data.

Mixture of Gaussians is a mixture model and so we begin our look at MoG there. We then describe the most popular method of learning MoG parameters: the EM algorithm. Finally, we show some low dimensional examples of MoG performance.

2.2.1 Mixture models

A general solution to the problem of data modeling is Bayesian learning [34]. In this formulation $p(\mathcal{D})$, where $\mathcal{D}$ is not just the observed data but all data to be modeled, is estimated by performing a weighted averaging over all possible hypotheses:

$$\begin{aligned} p(\mathcal{D}|\mathcal{X}) &= \sum_{h_i \in \mathcal{H}} p(\mathcal{D}|\mathcal{X}, h_i) \\ &= \sum_{h_i \in \mathcal{H}} p(\mathcal{D}|h_i) p(h_i|\mathcal{X}) \end{aligned} \tag{2.2}$$

where $\mathcal{X}$ is the training data set, $\mathcal{H}$ is the space of all possible hypotheses, h_i is an individual hypothesis, and $p(\mathcal{D}|\mathcal{X})$ is the probability of the underlying distribution responsible for generating all the data given the training set. In this context a hypothesis corresponds to a model and its parameter values so the concept of a hypothesis is a very general one. This assumes that the underlying quantity, $\mathcal{D}$, is conditionally independent of $\mathcal{X}$ given h_i which is valid considering each h_i determines a probability distribution over $\mathcal{X}$, and $\mathcal{X}$ is our only glimpse at this underlying process. In all but the simplest problems, this probability is too computationally expensive to calculate or analytically intractable to derive [29].

Another approach to statistical learning is *maximum a posteriori* (MAP) [30]. MAP approximates $p(\mathcal{D}|\mathcal{X})$ by using only one hypothesis to make predictions:

$$p(\mathcal{D}|\mathcal{X}) = p(\mathcal{D}|h_{MAP}) \tag{2.3}$$

where

$$h_{MAP} = \underset{h_i}{\operatorname{argmax}}\, p(h_i|\mathcal{X}) \tag{2.4}$$

In general, MAP provides a good approximation to the fully Bayesian treatment discussed above. Usually, when applying Bayesian Learning to a large data set, the correct hypothesis will dominate the $p(\mathcal{D}|\mathcal{X})$ prediction [34]. Therefore, as the size of the data set increases, the fully Bayesian solution can be replaced by a single hypothesis. This is the justification behind MAP methods. MAP methods are preferable to fully Bayesian treatments because removing the summation or integral over the hypotheses in equation (2.2) converts the more general Bayesian solution into an optimization problem. In addition, MAP still makes use of prior probabilities over the hypotheses being considered. The presence of priors over hypotheses can help deter overfitting since more complicated hypotheses should be given a lower prior probability while simpler hypotheses will be given a higher prior probability [29], helping MAP algorithms choose simpler models.

Maximum Likelihood (ML) methods are less general still. Instead of picking a hypothesis based on its *a posteriori* probability, ML methods only consider one hypothesis, or set of parameter values, regardless of which hypothesis is most probable [11]. Instead, all hypotheses are considered to have equal probability *a priori*. Maximum Likelihood methods are highly sensitive to the observed data when the training data set is small, but usually converge to a solution nearly identical to the fully Bayesian treatment when the data set is sufficiently large. ML methods are attractive because there is no need for hypothesis priors which may be highly subjective. The algorithms discussed here fit into the ML framework because we often have no *a priori* knowledge about which hypotheses are most likely. Therefore, a uniform prior across all possible hypotheses is reasonable.

Figure 2.1 shows the solution surface of a ML problem with two parameters, one for each dimension of the Gaussian mean. The best ML solution is that where the quantity $p(\mathcal{X}|\Theta)$ is at a maximum. Here, Θ is all model parameters. In this example, $\Theta = \{\mu_1, \mu_2\}$ which results in:

$$\log p(\mathcal{X}|\Theta) = \sum_{x \in \mathcal{X}} \log \mathcal{N}(x| \left[\mu_1, \mu_2\right], I). \tag{2.5}$$

2.2.2 Expectation-Maximization

We are performing supervised classification in our experiments by fitting a separate mixture of Gaussians model to each class. The training of each class, however, is done using an unsupervised learning algorithm – we are modeling each class with a mixture of Gaussians but the cluster memberships for the class members is not predetermined. This section provides

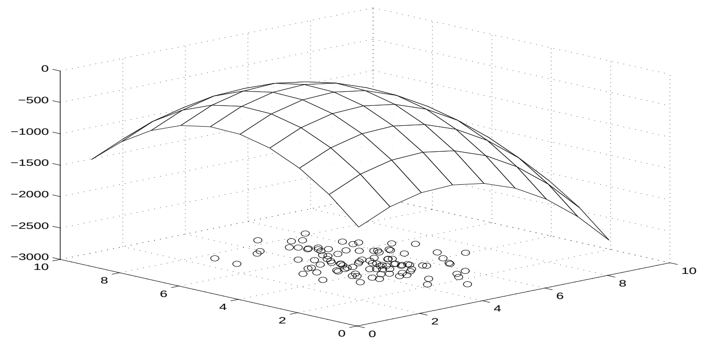


Figure 2.1: This plot shows the effect of varying the parameter for the mean on the log likelihood of the model producing the data set, $P(\mathcal{X}|\Theta)$.

a basic understanding of the basis for the expectation-maximization (EM) algorithm which is used to train a MoG for each class.

The expectation-maximization method [9] is a general procedure for finding the maximum likelihood estimate of the parameters of an underlying distribution from an incomplete data set [4]. The data set is assumed to have been generated by this underlying distribution. The data set may be incomplete due to inadequacies in the data gathering process, or it may be that the values of some variables are not observable. These unobserved variables are referred to as latent variables. In MoG the latent variables are the variables which specify which cluster a data point belongs to.

Letting $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ denote the observed data, $\mathcal{Y} = \{y_1, y_2, \ldots, y_n\}$ the latent data, and $\mathcal{Z} = \mathcal{X} \cup \mathcal{Y} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ the complete data, we want to maximize the probability of the model generating the observed data. This probability is known as the likelihood of the model given the data and is denoted by $\mathcal{L}(\Theta|\mathcal{Z}) = p(\mathcal{Z}|\Theta)$. There are many ways in which $p(\mathcal{Z}|\Theta)$ can be maximized. In fact, we can maximize any function as long as we can prove that the function we are maximizing has the same global optima as $\mathcal{L}(\Theta|\mathcal{Z})$. A popular choice for an alternate to $\mathcal{L}(\Theta|\mathcal{Z})$ is the log-likelihood of the data.

The most generally used function to be maximized is [4]:

$$Q(\Theta, \Theta^{(i-1)}) \quad = \quad E[\log P(\mathcal{Z}|\Theta)|\mathcal{X}, \Theta^{(i-1)}]$$

$$\begin{aligned} &= E[\log P(\mathcal{X},\mathcal{Y}|\Theta)|\mathcal{X},\Theta^{(i-1)}] \\ &= \int_{y\in\Gamma} p(y|\mathcal{X},\Theta^{(i-1)}) \log P(\mathcal{X},y|\Theta)dy \qquad (2.6)\end{aligned}$$

where Γ represents the set of all possible values for the latent data. The latent data, $\mathcal{Y}$, is the variable of interest in the expected value because the other half of the data, $\mathcal{X}$, is given and, therefore, constant. This is the expected value of the completed data set given the parameter values. This quantity is attempting to model the underlying process which generated the data, not just the observed data. Maximizing the expression in equation (2.6) consists of taking its derivative with respect to the model parameters which yields:

$$\frac{\partial}{\partial\Theta}Q(\Theta,\Theta^{(i-1)}) = \frac{\partial}{\partial\Theta}\int_{y\in\Gamma} P(\mathcal{Y}|\mathcal{X},\Theta^{(i-1)}) \log P(\mathcal{X},\mathcal{Y}|\Theta)dy \qquad (2.7)$$

To find the proper values for Θ, EM-style algorithms consist of two steps: the expectation (E) step and the maximization (M) step. The E-step involves the calculation of $Q(\Theta,\Theta^{(i-1)})$. The M step consists of computing updated values for the parameters of the model. These update equations are found by completing the derivative of $Q(\Theta,\Theta^{(i-1)})$ for each parameter. These two steps are repeated until $Q(\Theta,\Theta^{(i-1)})$ converges to a local maximum. Since computing $Q(\Theta,\Theta^{(i-1)})$ can be expensive in practice, many other termination conditions may be used, including running the algorithm for a set number of iterations. Each iteration is guaranteed to increase the log likelihood of the model generating the data [21].

2.2.3 Mixture of Gaussians via EM

Bilmes provides an in-depth tutorial showing how to derive the EM updates for a mixture of Gaussians in [4]. He replaces the two components of the function $Q(\Theta,\Theta^{i-1})$ which we then want to maximize:

$$\begin{aligned} P(\mathcal{Y}|\mathcal{X},\Theta^{(i-1)}) &= \prod_{n=1}^{N} p(y_n|x_n,\Theta^{(i-1)}) && (2.8) \\ \log P(\mathcal{X},\mathcal{Y}|\Theta) &= \sum_{n=1}^{N} \log P(x_n|y_n)P(y_n) = \sum_{n=1}^{N} \log(\pi_{y_n} p(x_n|\theta_{y_n})) && (2.9) \\ Q(\Theta,\Theta^{i-1}) &= \sum_{y\in\Gamma}\sum_{n=1}^{N} \log(\pi_{y_n} p(x_n|\theta_{y_n})) \prod_{n=1}^{N} p(y_n|x_n,\Theta^{(i-1)}) && (2.10)\end{aligned}$$

where y_n is the current assumption for the cluster which generated x_n and the integral in (2.7) is replaced with a sum due to the fact that y can only take a discrete number of values. The θ_{y_n} notation is dropped after this point and is assumed whenever θ is used. Still following [4], (2.10) is then algebraically manipulated using the EM assumption that we know which cluster generated the data and $\sum_{c=1}^{C} p(c|x_n, \Theta^{(i-1)}) = 1$:

$$Q(\Theta, \Theta^{i-1}) = \sum_{c=1}^{C}\sum_{n=1}^{N} \log(\pi_c) p(c|x_n, \Theta^{(i-1)}) + \sum_{c=1}^{C}\sum_{n=1}^{N} \log(p(x_n|\theta_c)) p(c|x_n, \Theta^{(i-1)}) \quad (2.11)$$

where

$$p(c|x_n, \Theta^{i-1}) = \frac{\pi_c p(x_n|\theta_c)}{\sum_{k=1}^{C} \pi_k p(x_n|\theta_k)}. \quad (2.12)$$

Then, taking the derivative of $Q(\Theta, \Theta^{i-1})$ with respect to the three model parameters, π, μ, and Σ reveals the update parameters for those parameters respectively. All of the parameters involved in modeling a given cluster (i.e. $\{\mu_c, \Sigma_c\}$) are denoted by θ_c and Θ^{i-1} is dropped altogether because it can be safely assumed when using values computed during the E-step such as $p(x|c)$. Equation 2.11 is separated into two parts, one with the π parameter and the other with the μ and Σ parameters given that $p(c|x_n)$ doesn't directly involve any of these model parameters. Therefore, the derivatives only need to be taken using the portion of (2.11) that is relevant to the parameter in question. Doing so yields:

$$\pi_c \leftarrow \frac{1}{N}\sum_{n=1}^{N} p(c|x_n) \quad (2.13)$$

$$\mu_c \leftarrow \frac{\sum_{n=1}^{N} x_n p(c|x_n)}{\sum_{n=1}^{N} p(c|x_n)} \quad (2.14)$$

$$\Sigma_c \leftarrow \frac{\sum_{n=1}^{N} p(c|x_n)(x_n - \mu_c)(x_n - \mu_c)^T}{\sum_{n=1}^{N} p(c|x_n)}. \quad (2.15)$$

EM is a two step, iterative algorithm. The update equations comprise the M-step. The E-step is completed by computing the probability $p(c|x_n)$ which is the same thing as (2.12). Learning should cease when (2.11) ceases to increase at a rate greater than some pre-specified value or after a certain number of iterations.

2.3 Covariance Regularization

Covariance regularization is simply a method for reducing the number of free parameters in a model. In all domains, not just the exceptionally high dimensional, the effect of having many free parameters and not enough data to provide an accurate estimate of the parameters' true values is over-fitting. When over-fitting has occurred the learned model no longer generalizes well to unseen data. Reducing the number of parameters is one way to approach the problem. High dimensional applications suffer from an even more pressing problem than over-fitting when using methods that require the inverse of the covariance matrix: singularity. The rank of a matrix is a function of the number of columns (or rows) that are not linear combinations of the other columns (or rows) and a square matrix with rank less than the number of rows or columns of the matrix is not invertible and called singular [23]. The rank of a covariance matrix is less than or equal to the number of data points; therefore the covariance matrix of data with fewer data points than dimensions is going to be singular.

We view covariance regularization as having two types of techniques in use today: modifying the empirical covariance matrix or using a technique to create a covariance matrix that is full-rank from the data. The class of covariance regularization that modifies the empirical covariance matrix can be further broken down into a spectrum. At one end there are those that constrain the covariance matrix to be diagonal and at the other end there are those that modify the covariance matrix so that it is no longer singular. An interesting interpretation of the effect of those covariance regularization methods that modify the empirical covariance matrix is that they make the matrix less specific to the observed data so the learned model performs better on the unseen validation and testing data sets. Those methods that create a covariance matrix using a method that doesn't involve the empirical covariance matrix are intended to produce covariance matrices which are truer to the underlying data distribution than the empirical covariance matrix. In other words, covariance regularization gives a Gaussian model a more general shape with which to fit the data – a shape that is less tightly fit to the observed data and, therefore, more likely to capture any unseen data points. This is particularly crucial in high dimensions where the sparseness of the data leaves us with an unrepresentative sample of data points with which to learn our model.

In addition to proposing the use of covariance regularization as a way to improve MoG, we experiment with covariance regularization techniques from both groups. The two shrinkage methods represent the first type of covariance regularization where the empirical covariance matrix is modified. In fact, the two shrinkage methods can also represent both ends of the spectrum of covariance regularization. At the large extreme value of the shrinkage parameter these shrinkage methods do nothing more than constrain the empirical covariance matrix; one to be diagonal, the other to have only ones on the diagonal. The constraint to have only diagonal elements is a common one in the literature and the constraint to have only ones on the diagonal is equivalent to fuzzy kmeans which is a form of kmeans where the data has a fractional membership to each cluster which is determined by Euclidean distance[30]. The latter type of covariance regularization is represented by sparse matrix transform. PCA would also fall into the second category of covariance regularization. We attempted to experiment with MPPCA (an algorithm that uses PCA to perform covariance regularization and is described in Section 2.4) as implemented in [32] which was numerically unstable for data as high dimensional as ours because their code does not compute $\log p(x|c)$ but $p(x|c)$ which will always go to zero for high dimensional data.

2.4 Related Literature

Many variations on the mixture of Gaussians model have been proposed for use in computer vision – many of them presented under titles such as linear discriminant analysis or mixture of factor analyzers. Taken as a whole we see a body of literature advancing toward MoG models that attempt to exploit certain aspects of the data in order to achieve better classification. What is missing is a common connection between all these algorithms – they all perform covariance regularization.

Moghaddam and Pentland proposed the use of a mixture of Gaussians on PCA-projected data for object detection and tested it on human faces, human hands, and facial features such as eyes, nose, and mouth [31]. Their work is seminal in that they were the first to show how to use principal component analysis to estimate a Gaussian distribution using (2.16).

$$p(x|\mu,\Sigma) = \left[\frac{e^{-\frac{1}{2}\sum_{i=1}^{D}\frac{y_i^2}{\lambda_i}}}{(2\pi)^2\prod_{i=1}^{D}\lambda_i^{\frac{1}{2}}}\right] \times \left[\frac{e^{-\frac{\epsilon^2(x)}{2\sigma}}}{(2\pi\sigma)^{\frac{M-D}{2}}}\right] \tag{2.16}$$

where

$$\begin{aligned} \epsilon^2(x) &= \sum_{i=D+1}^{M} y_i^2 \\ &= ||\tilde{x}||^2 - \sum_{i=1}^{D} y_i^2 \qquad (2.17) \\ \sigma &= \frac{1}{M-D} \sum_{i=D+1}^{M} \lambda_i \qquad (2.18) \\ y &= U_D^T \tilde{x}. \qquad (2.19) \end{aligned}$$

The λ values are the eigenvectors sorted in decreasing order and columns of the matrix U are the corresponding eigenvectors calculated using PCA. The constant D corresponds to the number of PCA subspace dimensions kept so U_D is the matrix of the first D eigenvectors.

The first term of (2.16) is called the in-space probability and corresponds to the probability as calculated using the kept subspace dimensions. The second term is the out-of-space probability and is the probability as calculated using the discarded subspace dimensions. The second term is an estimate that assumes equal variance in all discarded dimensions and, therefore, the entire calculation is a form of covariance regularization. The $\epsilon^2(x)$ value is simply the mass of the data projected into the discarded dimensions and σ is the variance of the discarded dimensions.

Their method differs from ours in that detection is a single-class, classification problem. This is done by modeling the class of interest and determining membership by comparing the probability of membership, $p(c|x)$, to some threshold. In their work, the class of interest may be modeled using one or more Gaussians. When modeled using a single Gaussian, they use their PCA regularization of a Gaussian to compute $p(x|c)$ hence performing covariance regularization. They found that using PCA and one Gaussian to represent a class outperformed the use of mixture of Gaussians in the original, high-dimensional space.

Hinton et al. used a mixture of factor analyzers (MFA) [17] to model each class of handwritten digits in a digit classification problem [22]. Their hypothesis is that factor analysis, performed locally on each cluster, could model the manifold that each cluster's data points lie upon. These hypothetical manifolds are formed through the application of small image transformations to a class prototype image. Each digit class may have multiple, separated, continuous manifolds; therefore each class is represented using a mixture of factor analyzers. Hinton et al. cite work showing that image transformations found naturally in handwritten digits such as shearing, scale, and translation yield at least seven free dimensions so they keep seven or more factors in their experiments. Their model also has a probabilistic interpretation in that it uses factor analysis to represent a reduced-dimensionality Gaussian distribution. So, in effect, their model is an approximation to a true mixture of Gaussians via factor analysis. Variational Bayesian mixture of factor analyzers (VBMFA) is another recent, frequently cited mixture model [16]. VBMFA is essentially the same as MFA but has update equations that have been derived from the variational Bayes framework [26, 15] instead of EM. VBMFA represents a potential improvement over MFA because the Bayesian framework allows one to compare models using $p(\mathcal{X}|\Theta)$ which allows for modifications such as changing the number of clusters or factors during learning.

The use of factor analysis to estimate a Gaussian distribution is predicated upon (2.20) [20] and the addition of a mixture of factor analyzers involves the addition of a cluster mean (2.21):

$$\begin{aligned} p(x) &= \mathcal{N}(0, UU^T + \psi) && (2.20) \\ p(x|c) &= \mathcal{N}(\mu_c, U_c U_c^T + \psi) && (2.21) \end{aligned}$$

where U is the matrix of factor loadings and ψ is the variance unaccounted for by U. Even when used as a part of a mixture model, the ψ term does not have a cluster component but captures the variance unaccounted for by the mixture of factor analyzers. The aim of factor analysis is to find the U and ψ that best model the covariance structure of $\mathcal{X}$. The factor loadings play a role similar to that of the principal components of PCA in that they provide an informative projection of the data.

In [17, 16] (2.21) is never explicitly calculated. MFA is formulated as

$$p(x) = \sum_{c=1}^{C} \int p(x|y, c)p(y|c)\pi_c dy \tag{2.22}$$

where y is the factors for x or the projected value of x in the factor analysis subspace, $p(y|c) = \mathcal{N}(0, I)$, and π_c is the mixing proportion of cluster c or $p(c)$. The generative process for MFA begins with the selection of a cluster using π_c. Then, the factors are generated according to their zero-mean, unit variance distribution. Finally, the observed data is generated according to $p(x|y, c) = \mathcal{N}(\mu_c + U_c y, \psi)$. However, this is merely a computational decision and MFA can still be considered a mixture of Gaussians via covariance regularization according to (2.21).

Tipping and Bishop brought principal component analysis to the world of mixture of Gaussians and EM through their mixture of probabilistic principal component analyzers (MPPCA) [37]. In their work they create a probabilistic derivation of PCA based upon a minor change to factor analysis involving the way in which noise is modeled. Before their work, PCA had no probabilistic derivation. In MPPCA, PCA is performed locally for each cluster. Like the mixture of factor analyzers, the resulting principal components (or principal factors) are used to represent a Gaussian distribution. The mixture model is fit to the data using update equations derived via the EM framework.

The MPPCA formulation does not use Moghaddam and Pentland's method of performing covariance regularization using PCA although the concepts are very similar. Instead they replace the covariance matrix Σ from (2.1) with an estimated one which we will denote Σ^* for each cluster.

$$\Sigma_c^* = \sigma_c^2 I + U_c U_c^T. \tag{2.23}$$

U_c contains the eigenvectors of Σ_c as computed by the standard MoG model in (2.15) which they refer to as the responsibility weighted covariance matrix and σ_c is a term modeling the noise not accounted for by the local PCA for cluster c computed the same way as in (2.18). The motivation behind MPPCA is that the unique dimensionality reduction of each cluster will improve clustering by better fitting the data of that cluster.

Computer vision researchers have long focused on linear discriminant analysis (LDA) for classification problems. For many, like factor analysis and principal component analysis, LDA's appeal lies in its creation of a subspace which is more amenable to clustering. Unlike factor analysis and principal component analysis, LDA and LDA's cousin, quadratic discriminant analysis (QDA), are not designed to approximate the true subspace or manifold which contains a class of images. Instead, they create a subspace representation that makes clustering easier by increasing the distance between inter-class points while reducing the distance between intra-class points [11]. LDA and QDA have roots in a mixture of Gaussians [21]. LDA exists when each class is modeled using a single Gaussian and each Gaussian is constrained to have identical covariance. QDA exists when each class is modeled using a single Gaussian but each Gaussian is allowed to have unique covariance resulting in a non-linear decision boundary.

The use of discriminant functions for classifications using a Gaussian distribution are less concerned with computing probabilities of membership and more concerned with solving for a decision boundary for class memberships. The discriminant function for class c has the form [11]:

$$g_c(x) = \log p(x|c) + \log P(c). \tag{2.24}$$

where $P(c)$ is the prior probability of observing data from class c and is treated like a constant and is not computed as a part of updating the model parameters. In the case of normally distributed data we need to take the log of a Gaussian for the $\log p(x|c)$ term:

$$g_c(x) = -\frac{1}{2}(x-\mu_c)\Sigma_c^{-1}(x-\mu_c) - \frac{M}{2}\log 2\pi \log|\Sigma_c| + \log P(c). \tag{2.25}$$

In the case of linear discriminate analysis, all of the covariance matrices of all classes are constrained to be equal. This means that only

$$g_c(x) = -\frac{1}{2}(x-\mu_c)\Sigma_c^{-1}(x-\mu_c) + \log P(c) \tag{2.26}$$

have terms that concern the selection of the class c. Expanding $(x-\mu_c)\Sigma_c^{-1}(x-\mu_c)$ and removing the terms that have no dependence upon c results in:

$$g_c(x) = x^T\Sigma^{-1}\mu_c - \frac{1}{2}\mu_c^T\Sigma^{-1}\mu_c + \log P(c). \tag{2.27}$$

Equation (2.27), which defines the discriminant function for LDA, has a single x component so it defines a linear function.

In quadratic discriminant analysis, the covariance matrices are allowed to vary across classes. This changes the result of expanding $(x-\mu_c)\Sigma_c^{-1}(x-\mu_c)$ between (2.26) and (2.27) because the covariance matrices are now dependent upon c increasing the number terms kept:

$$g_c(x) = -\frac{1}{2}\log|\Sigma_c| - \frac{1}{2}(x-\mu_c)^T\Sigma_c^{-1}(x-\mu_c) + \log P(c). \tag{2.28}$$

This change makes the decision surface quadratic with respect to x.

Covariance regularization has been a part of the computer vision literature for at least two decades in the form of regularization of LDA for recognition tasks. The most popular method is PCA+LDA which uses PCA to project the data into a dimensionality where the covariance matrices used to compute LDA are no longer singular [3]. In recent years several variations on this theme have appeared where PCA null space is exploited or completely ignored for its lack of discriminatory information [39, 7, 41, 38]. Regularized discriminant analysis (RDA) is a popular technique in many domains including computer vision [38]. RDA is to LDA and QDA as our shrinkage MoG method is to fuzzy kmeans and MoG constrained to a diagonal covariance matrix.

Over time it has become clear that the invariance to small image transformations provided by discriminate analysis, FA, and PCA are not sufficient for all but small, sub-pixel transformations. Frey and Jojic take a more direct approach of bringing image transformation invariance to the mixture of Gaussians model (TMG) [14]. In their work, Frey and Jojic argue that explicitly accounting for image transformations is needed because methods like MFA only account for small, locally linear regions of their respective locations on its class's manifold [13]. By explicitly accounting for these image transforms, they now attempt to cover the entire manifold. Assuming the specified transforms (very nearly) fully define the differences between images within a class, using more than one Gaussian per class should no longer be necessary. In subsequent work, Frey and Jojic show their TMG model can be extended to any mixture model by presenting a transformed mixture of factor analyzers [14]. Adding factor analysis to TMG would presumably increase the coverage of the manifold from the discrete points of the manifold selected prior to training to a small region around each point on the manifold.

TMG is only a mixture of Gaussians, however. The probability of a data point given a cluster and transformation is defined as:

$$p(x|T,c) = \mathcal{N}(x|T\mu_c, T\Sigma_c T^T + \psi)\pi_{T,c} \tag{2.29}$$

where T is the selected transformation and ψ is a diagonal matrix defining some zero-mean noise added to the transformed data immediately prior to generating the observed data. It simply replicates each Gaussian in the mixture once for every transformation accounted for which is the reason for the additional T index on the mixing parameter π. Letting $\mathcal{T}$ be the set of transformations accounted for the model, each Gaussian in the mixture is actually $|\mathcal{T}|$ Gaussians constrained to have means and covariances that are transformed version of each other. Furthermore, when applying TMG to a high-dimensional data set, the covariance matrices, Σ_c, are constrained to be diagonal.

Mixture modeling is used in many computer vision (end-to-end) systems. Draper uses kmeans in his biologically motivated object recognition system but also compares the use of an algorithm similar to the MPPCA model [10]. In Draper's system, clustering is performed in an unsupervised context, not for each class, because he is primarily concerned with simply breaking the data into groups so a subspace can be learned for each cluster to better compare the objects.

Recently a comprehensive look at mixture of Gaussian model learning heuristics for the problem of classifying satellite images was published [40] where, like this Thesis, a mixture of Gaussians is learned for each class. However, this work goes well beyond asking whether MoG is a good technique for their problem domain to fine-tuning MoG for their specific application. They add sophisticated techniques for initializing the MoG models, determining the proper number of clusters for each class, and sharing the off-diagonal covariance matrix elements between clusters to allow them to move beyond their constraint that all covariance matrices be diagonal – all specially designed for their particular needs.

Various forms of mixtures of Gaussians were compared in the problem of classifying objects in video clips [19]. This comparison is similar to ours in that the various forms of mixtures of Gaussians were defined by how they represented each Gaussian's covariance matrix. Their variations included using spherical-only (identity matrix as covariance matrix) to using the full covariance matrix. What this work didn't cover was the possibility of using covariance regularization techniques like the ones investigated in this Thesis. Their covariance regularization techniques are of the kind that constrain the covariance matrix to be variations on the diagonal matrix which we will see are equivalent to specific parameter settings of two of the three covariance regularization methods experimented with in this Thesis.

Another study seeking best practices for using a mixture of Gaussians for image data (as well as other domains) proposed a heuristic for fixing singular covariance matrices during learning [33]. The covariance matrices are made non-singular by increasing the tiny elements from a Cholesky decomposition [23] by some heuristically determined value. This work did not identify this heuristic as a covariance regularization technique nor did they compare it to other alternative forms of covariance regularization including the frequently used constraint that only the diagonal elements of the empirical covariance matrix be kept.

The transformation methods such as MFA and MPPCA present their work, not as covariance regularization methods, but as manifold and subspace learning methods. These algorithms allowed for simultaneous classification and dimensionality reduction which was an improvement upon previous efforts which created subspaces after clustering was completed using another algorithm such as kmeans [27]. The benefit of these algorithms were considered to be their invariance to small image transformations and their relationship to covariance regularization was largely ignored.

All of the models discussed above do regularize a mixture of Gaussians using one method or another. The transformation-based methods (PPCA, MFA, and VBMFA) regularize a multivariate normal using factors or the closely related principal factors [20]. LDA and QDA estimate a mixture of Gaussians by restricting the mixture to one Gaussian per class and the covariance matrices to be identical in the case of LDA. The methods currently used in computer vision applications that are referred to as a mixture of Gaussians are mostly reduced parameter Gaussians where the covariance matrix is reduced to just a few parameters multiplied against a diagonal matrix or simply the diagonal of the covariance matrix.

In the computer vision literature, mixture models using Gaussians and LDA have been primarily focused on representing the manifolds that object images lie upon or the subspaces that those object images lie within. Modifying the mixture models to explicitly account for image transformations and, therefore, model the object's manifold have made progress toward capturing image transformation invariance. Regularizing LDA to work in the presence of insufficient amounts of data has also been popular and is a popular application of covariance regularization to computer vision data but the use of covariance regularization in computer vision has not moved much beyond this. In this Thesis the effect of these algorithms on the covariance matrix takes a position of primary importance. What has been lacking in the computer vision community is using covariance regularization, beyond simply reducing the number of covariance parameters, to improve the basic mixture of Gaussians model. Although we only apply our algorithm to supervised classification, the mixture of Gaussians model can be used to perform unsupervised classification unlike discriminant analysis which can only be used in supervised problems. This makes our algorithm a more widely applicable method of clustering data. A mixture of Gaussians using covariance regularization to better fit the data while avoiding issues of covariance matrix singularity could, in and of itself, yield a method that is superior in its classification power to the state of the art mixture of Gaussians models and linear and quadratic discriminant analysis. This will be the focus of this Thesis.

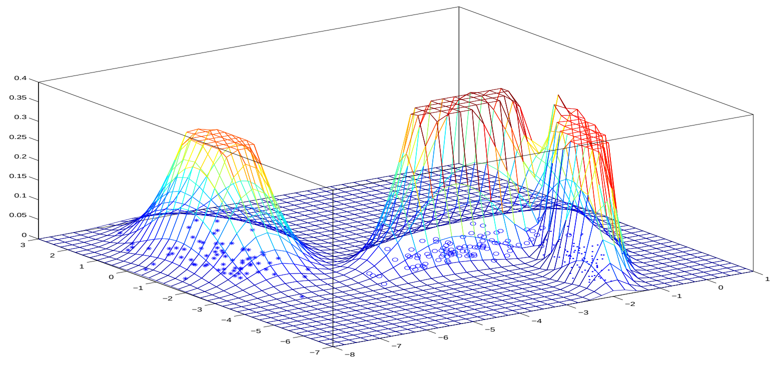
Chapter 3

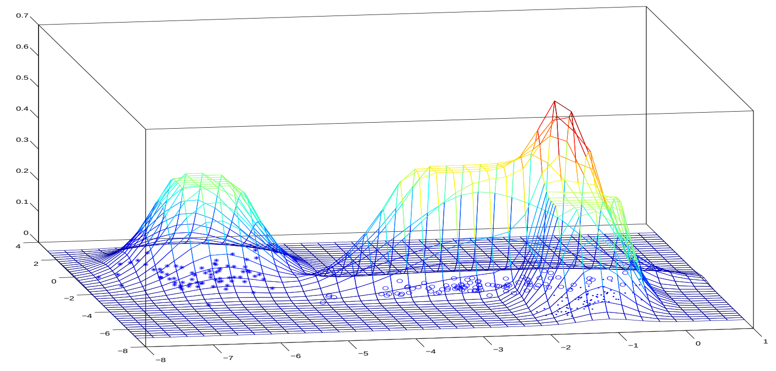
Approach

This chapter builds upon the mathematical preliminaries presented in the Background Chapter by describing theoretical background behind those aspects that are particular to our version of mixture of Gaussians. We then provide specific implementation details of the algorithms analyzed in this Thesis in addition to the experimental methods used in their comparisons. First, we begin with some low-dimensional examples of how covariance matrix modifications effect how the data is modeled using a mixture of Gaussians.

3.1 Low dimensional mixture of Gaussians examples

Now we include a few examples to illustrate the performance of a mixture of Gaussians in low dimensions shown in Figure 3.1. We generated three classes of data, shown by the three types of markers. The classes are generated using the MPPCA generative model [37] which gives the data non-axis aligned variance. Figure 3.1(a) shows the results of training a mixture of Gaussians with full covariance on the data. Notice how nicely the learned Gaussians model each cluster. Figure 3.1(b) shows the effect of constraining the covariance matrix to be diagonal by forcing all off-diagonal values to be zero. We see that the Gaussians are now axis aligned and, therefore, are unable to fit the data as well as the full covariance version and the two right-most clusters' Gaussians overlap causing a region of high probability that does not jive with the observed data or the underlying, generative model. Figure 3.1(c) shows the effect of constraining each Gaussian to be a unit sphere (ones along the diagonal, zeros everywhere else). Now we see a total breakdown. Only two


(a) Unconstrained MoG.


(b) MoG constrained to have axis-aligned variance.

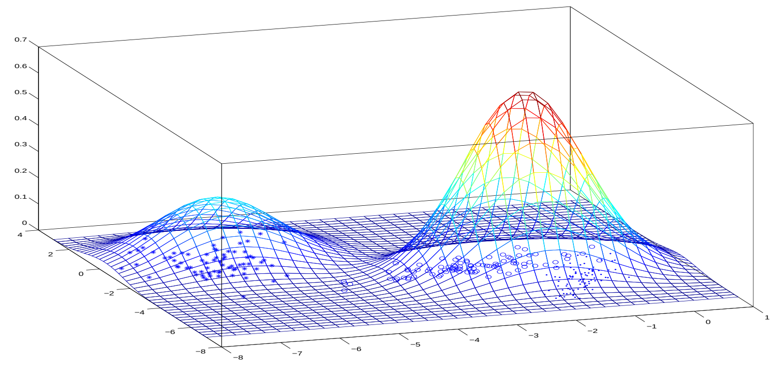

(c) MoG constrained to have spherical Gaussians only.

Figure 3.1: Examples of MoG learning in low dimensions for three versions of MoG. Each plot is rotated to best show the features of that version's estimate of the probability distribution function. Data was generated using a mixture of Gaussians where each Gaussian is randomly rotated so the data are not axis aligned.

clusters are visible because two of the three are nearly sitting on top of each other. A MoG with only spherical Gaussians is unable to detect three clusters.

3.2 Shrinkage covariance regularization

Shrinkage estimates usually involve a parameter that balances between the calculated, empirical covariance matrix and some pre-specified target matrix that is non-singular. Shrinkage methods can be written in a form resembling

$$\Sigma^* = \lambda T + (1 - \lambda)\Sigma \tag{3.1}$$

where Σ^* is the regularized covariance matrix, T is the target matrix, and λ is the shrinkage parameter. This shrinkage parameter can be determined in a number of ways including cross-validation and Bayesian inference.

Equation (3.1) is the form of shrinkage used in this Thesis. Six target matrices are compiled in [35]. From [35] we chose targets A and D for experimentation. In the MoG context, Σ is simply the covariance matrix computed via EM updates.

$$A_{ij} = \begin{cases} 1 & \text{if} \quad i = j \\ 0 & \text{if} \quad i \neq j \end{cases} \tag{3.2}$$

$$D_{ij} = \begin{cases} \sigma_{ii} & \text{if} \quad i = j \\ 0 & \text{if} \quad i \neq j \end{cases} \tag{3.3}$$

We chose versions A and D because they correspond to two popular MoG models when λ is set to one. When $\lambda = 1$, version A is equivalent to fuzzy kmeans [30]. When $\lambda = 1$, version D is equivalent to a mixture of Gaussians constrained to have only diagonal covariance matrices. As a result, MoG with shrinkage version A will be referred to as FKM and MoG with shrinkage version D will be referred to as DIAG for the remainder of this Thesis. It is worth noting that the authors of [35] prefer D over the other six methods because it represents a compromise between the versions presented in their paper in that it treats the diagonal elements differently than the covariance elements. Their preference was not based upon presented results.

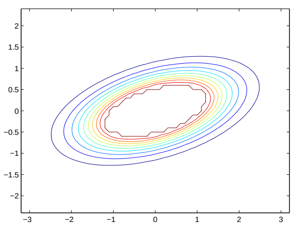
In its simplest, most common form, kmeans uses Euclidean distance to form hard classifications. Each iteration the cluster centers are updated according to the classifications and then new classifications are computed. Fuzzy kmeans is similar except it allows for fractional membership. In this way, fuzzy kmeans is similar to a mixture model. Furthermore, when you account for the fact that fuzzy kmeans uses Euclidean distance to compute classification, fuzzy kmeans becomes equivalent to the use of a perfectly spherical Gaussian because probability of membership is the same in all directions. The size of the sphere would not affect the classifications of the data points so a covariance matrix set to the identity matrix is just as good of a representation as any other constant along the diagonals. Target A uses (3.2) to compute the target matrix which will result in the identity matrix for $\lambda = 1$.

The diagonal elements of the covariance matrix are variance values. Equation (3.3) computes the target matrix as having these values on the diagonal and zeros elsewhere. This is simply a restriction of the covariance matrix. If all σ_{ii} values are non-zero, then T will be non-singular and so should Σ^* but not necessarily Σ.

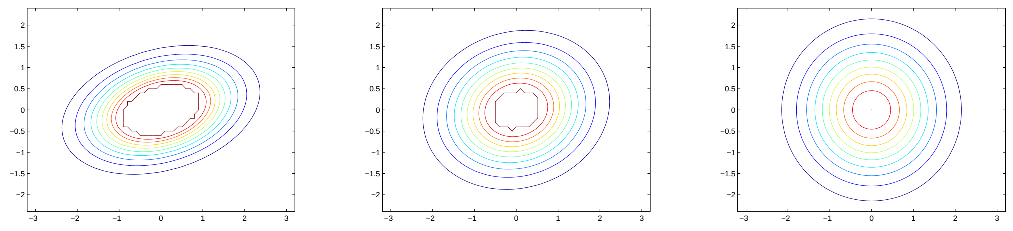
Figure 3.2 illustrates how the two shrinkage versions affect a Gaussian distribution given increasing values of λ. The unconstrained Gaussian has large amount of covariance which gives it its tilted, elongated shape. FKM converts this into a sphere, becoming increasingly spherical with each increase in λ. DIAG appears to rotate the Gaussian but it is actually just removing the covariance until it is axis-aligned.

3.3 Sparse matrix transform covariance regularization

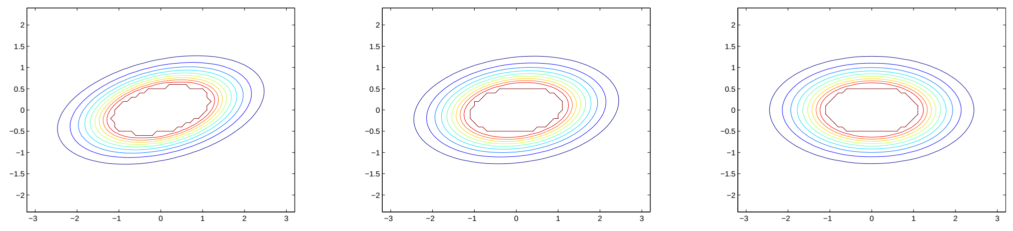
Using the sparse matrix transform (SMT) to estimate a covariance matrix is of our second type of covariance regularization. The authors of the SMT method claim that their method is truer to the underlying data than the covariance matrix that was computed from the observed data [6]. They back this up with experiments showing that the covariance matrix generated using SMT has eigenvalues much more similar to those of the true covariance matrix than those of the empirical covariance matrix. This is different from the shrinkage methods in that shrinkage methods are simply reducing the number of free parameters by constraining their values and making them more general by pushing their values away from



(a) The unconstrained Gaussian.



(b) FKM with increasing λ values of 0.2, 0.6, and 1.



(c) DIAG with increasing λ values of 0.2, 0.6, and 1.

Figure 3.2: Contour plots of the same Gaussian distribution with the two different version of shrinkage and λ values applied to it.

the learned values toward something more generic.

The rationale behind SMT is that the eigenvalues of a covariance matrix computed from the observed data when $N < M$ (where $N \ll M$ is typically the case in computer vision) are incorrectly zero because the covariance matrix is singular. When solving for eigenvalues using any of the most common methods (i.e. eigendecomposition, PCA, SVD) there can be at most $N - 1$ non-zero eigenvalues [27]. Of course if the data is created from a Gaussian process, an assumption one makes when classifying using a mixture of Gaussians, this cannot be the case. SMT is designed to find eigenvalues and eigenvectors that are a better approximation to the underlying Σ than that found using MLE.

The covariance regularization comes from their constraint of the possible rotations used as eigenvectors. In their work, they constrain the matrix of eigenvectors to be a combination, through multiplication, of a subset of many pre-specified Givens rotations. The rotations are chosen greedily from the set until K transformations have been selected by choosing the Givens rotation that removes the greatest amount of covariance whereas a non-greedy selection of rotations would look at all permutations (including the order in which they were removed) of allowed rotations to account for the maximum amount of covariance. The Givens rotations extract covariance from Σ two elements at a time which correspond to the covariance between two pixels and combines the rotations to create a matrix of eigenvectors (U) and eigenvalues (Λ) such that $\Sigma = U\Lambda U^T$.

The number of Givens rotations used to create U is determined by the number of rotations that maximizes the log probability of the data given the eigenvectors and eigenvalues returned by the sparse matrix transform. This probability is derived using the following steps in [6]:

$$\begin{aligned} p(x|U,\Lambda) &= \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma^*|^{\frac{1}{2}}} e^{-\frac{1}{2}\mathrm{tr}(\tilde{\mathrm{x}}^T\Sigma^{*-1}\tilde{\mathrm{x}})} \\ &= \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}\mathrm{tr}(\tilde{\mathrm{x}}^T U\Lambda^{-1}U^T\tilde{\mathrm{x}})} \\ &= \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{M}{2}\mathrm{tr}(U^T\Sigma U)\Lambda^{-1}} \\ &= \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{M}{2}\mathrm{tr}(\mathrm{diag}(U^T\Sigma U)\Lambda^{-1}} \end{aligned} \tag{3.4}$$

the log of which is

$$\log p(x|U,\Lambda) = -\frac{M}{2}\mathrm{tr}(\mathrm{diag}(U^T\Sigma U)\Lambda^{-1} - \frac{M}{2}\log|\Lambda| - \frac{NM}{2}\log(2\pi). \tag{3.5}$$

Starting at iteration $k = 0$ we begin with the empirical covariance matrix Σ_k at each iteration. The Givens rotation that removes the correlation between the two most correlated elements are chosen as U_k and the quality of covariance estimate is computed using (3.5). This process is repeated with the updated covariance matrix $\Sigma_{k+1} = U_k^T\Sigma_k U_k$.

The number of Givens rotations is determined via a cross-validation scheme using (3.5). The data is separated into three subsets and the Givens rotations are iteratively chosen as described above. At each iteration of selection of Givens rotations (3.5) is evaluated using

$\log p(\mathcal{X}_p|U_{p,k},\Lambda_{p,k})$ where p is one of the three partitions of the data. The k which provides, on average, the greatest probability of the data is selected:

$$D = \arg\max_k \sum_{p=1}^{3} \log p(\mathcal{X}_p|U_{p,k},\Lambda_{p,k}). \tag{3.6}$$

As Givens rotations are selected and combined, it is possible that a covariance matrix element that had previously been set to zero will no longer be zero after the application of further Givens rotations. Therefore it is possible that more Givens rotations than $\frac{N(N-1)}{2}$, which is the number of unique values in the covariance matrix, can be used to estimate the covariance matrix.

3.4 Implementations

Algorithm listing 1 shows the algorithm for the standard EM-based mixture of Gaussians model parameter updates. This pseudo-code also serves as the backbone of the MoG versions explored in this Thesis. The use of covariance regularization only varies in how Σ is computed.

The require statements are the algorithm inputs. Step 1 is the model initialization. As described in [8, 33, 19, 40] model initialization is very important for MoG via EM. We initialize the means to randomly chosen data points, the covariances to be the identity matrix, and the mixing parameters to be $\frac{1}{C}$. A more popular initialization scheme is to use kmeans to initialize the means and then compute the covariance matrices and mixing parameters accordingly. We chose not to use this because, as we mentioned earlier, fuzzy kmeans is equivalent to shrinkage version A with $\lambda = 1$. Step 2 is the termination condition. The parameter $maxIters$ determines the maximum number of iterations allowed before termination. The other portion of the termination condition is comparing the absolute differences between the newly computed $p(x|c)$ values (denoted by time t) and the $p(x|c)$ values from the previous iteration. This comparison is faster than the alternative $\log p(\mathcal{X})$ calculation. Given that $p(x|c)$ is the largest component of $p(\mathcal{X})$, a change in $p(x|c)$ would most likely result in a change in $p(\mathcal{X})$. Therefore this is a valid substitution. Step 5 is the $p(c|x)$ calculation. The purpose of the $p(x) = 0$ case is to prevent divide by zero errors.

Algorithm 1 High level description of general MoG EM learning algorithm

```
Require: $C$ {the number of clusters}.
Require: $\mathcal{X} \in \Re_{(M,N)}$ {$N$, $\Re^m$ data points}.
Require: $\epsilon$ {Min change in $p(c|x)$ to continue learning}.
Require: $maxIters$
 1: Initialize model.
 2: while $\max_{\forall x,c} |(p(x|c)^{(t)} - p(x|c)^{(t-1)})| > \epsilon$ and $t < maxIters$ do
 3:   Compute $p(x|c) \forall x \in \mathcal{X}, c \in C$. {Described in Algorithm 2}
 4:   $p(x) \leftarrow \sum_{c \in C} \pi_c p(x|c)$.
 5:   $p(c|x) \leftarrow \begin{cases} 0 & : p(x) = 0 \\ \frac{\pi_c p(x|c)}{p(x)} & : otherwise \end{cases} \forall x, c.$
 6:   $\pi_c \leftarrow \frac{1}{N} \sum_{x \in \mathcal{X}} p(c|x)$.
 7:   for all $c \in C$ do
 8:     if $\pi_c > 0$ then {Only update clusters with membership.}
 9:       $\mu_c \leftarrow \frac{\sum_{x \in \mathcal{X}} p(c|x) x}{\sum_{x \in \mathcal{X}} p(c|x)}$.
10:       $\tilde{\mathcal{X}} \leftarrow \mathcal{X} - [\mu_c, \mu_c, \cdots, \mu_c]$.
11:       $\tilde{\mathcal{X}}^{(c)} \leftarrow [p(c|x_1)\tilde{x}_1, p(c|x_2)\tilde{x}_2, \cdots, p(c|x_N)\tilde{x}_N]$.
12:       $\Sigma_c \leftarrow \tilde{\mathcal{X}}^{(c)} \tilde{\mathcal{X}}^T$.
13:     else
14:       Optionally reinitialize cluster.
15:     end if
16:   end for
17:   Increment $t$.
18: end while
```

Steps 9 – 12 are only performed when $\pi_c > 0$, otherwise that cluster's parameters are left alone for that iteration. $\tilde{\mathcal{X}}$ is notation for the mean-subtracted data set and $\tilde{\mathcal{X}}^{(c)}$ is the mean-subtracted data set weighted by $p(c|x)$. Mixture of Gaussians implementations vary in how they deal with clusters which have no membership (the $\pi_c = 0$ case); we chose to not touch the parameter values for all experiments.

Algorithm 2 High level description of $p(x|c)$ calculation.

```
1: for all $c \in C$ do
2:   $V, \Lambda \leftarrow \text{eig}(\Sigma_c)$.
3:   if $\min(\text{diag}(\Lambda)) < M * \epsilon$ then {Test for full rank $\Sigma_c$.}
4:     $\Sigma_c$ is not full rank. Quit.
5:   else
6:     $\Sigma_c^{-1} \leftarrow V\Lambda^{-1}V^T$.
7:     $\tilde{\mathcal{X}} \leftarrow \mathcal{X} - [\mu_c, \mu_c, \cdots, \mu_c]$.
8:     for all $\tilde{x} \in \tilde{\mathcal{X}}$ do
9:       $\log p(x|c) \leftarrow -\frac{1}{2}(\tilde{x}^T\Sigma_c^{-1}\tilde{x} + \sum_i \log \Lambda_{ii})$.
10:    end for
11:    $p(x|c) \leftarrow e^{\log p(x|c) - \max_{\forall x,c}(\log p(x|c))}$.
12:  end if
13: end for
```

Algorithm 2 describes the steps taken to compute $p(x|c)$. In many applications this calculation may not be worthy of its own block of pseudo-code but our implementation of MoG is general enough to operate on both high and low dimensional data sets. Step 2 is the eigenvalue solver – either Matlab [24] or Octave [12]. This function, given the covariance matrix, returns the eigenvalues (Λ) and eigenvectors (V) associated with that matrix. The eigenvectors are stored in a matrix where each column is an eigenvector and the eigenvalues are stored in a diagonal matrix. Step 3 is the test for covariance matrix singularity. Given that the covariance estimate methods should each produce a non-singular matrix, a singular matrix is treated as an error. This is done in a way similar to Matlab's QR decomposition function where the smallest eigenvalue is compared against the dimensionality of the data set (M) times some constant (ϵ) which, in this case, is the machine precision for type double values. In step 6 we take advantage of the fact that we have already computed the eigenvectors and eigenvalues of Σ_c and use them to invert Σ_c [23]. Step 9 is just the log of the multivariate normal with a few constants removed and using the sum of the log of the eigenvalues to compute the log determinant of Σ_c [23]. Finally, step 11 improves the numerical stability of the algorithm. Although only necessary in high dimensions, it is used for all experiments for consistency. The net effect of this calculation is that one of the $p(x|c)$ values will have a value of one after being taken out of log space and the other values will be scaled up accordingly. This is analogous to multiplying the $p(x|c)$ values by a constant.

In the basic mixture of Gaussians EM algorithm, the model covariance matrices are computed as shown in step 12 of Algorithm 1. This is the covariance computed by multiplying the entire data set, mean subtracted and weighted according to $p(c|x)$, times itself transposed.

Again, the algorithms explored in this Thesis are variants on this EM mixture of Gaussians theme in that they deviate from Algorithm 1 in how Σ is computed. Two shrinkage methods are used and they compute Σ as shown in Algorithm 3.

Algorithm 3 Steps used by shrinkage mixture of Gaussian algorithms to compute Σ, each clusters covariance matrix.

```
1: $R \leftarrow \tilde{\mathfrak{X}}^{(c)}\tilde{\mathfrak{X}}^T$.
2: Select $T$ according to shrinkage version.
3: $\Sigma \leftarrow \lambda T + (1-\lambda)R$.
```

$\tilde{\mathfrak{X}}^{(c)}$ and $\tilde{\mathfrak{X}}$ are computed in steps 10 and 11 of Algorithm 1 respectively. R assumes the role of Σ in the standard EM mixture of Gaussians algorithm which is the empirical covariance. In these shrinkage versions of mixture of Gaussians, λ would be another configurable parameter input to the algorithm. The target covariance matrix, T, is computed using either (3.2) or (3.3).

The sparse matrix transform method of performing covariance regularization was implemented by the authors of [6]. We use their implementation of SMT and, therefore, we will not go beyond the description of SMT given in Section 3.3. We incorporate the SMT into a mixture of Gaussians by passing $\tilde{\mathfrak{X}}$ in to the SMT covariance regularization function supplied by the SMT authors. The SMT method builds eigenvectors, U, and eigenvalues, Λ, which we use to create a covariance matrix using (3.7). The determinant of Σ in the denominator of (2.1), the multivariate normal pdf, is computed using (3.8):

$$\Sigma^{-1} = U\Lambda^{-1}U^T \tag{3.7}$$

$$\log|\Sigma| = \sum_{i=1}^{M} \log \lambda_i \tag{3.8}$$

where M is the dimensionality of the data. Prior to computing (3.7) and (3.8) the Λ values are are set to a minimum value to avoid singularities. In our work with this algorithm, we found that it sometimes returned zero, or nearly zero, eigenvalues.

3.4.1 Supervised classification

Although a mixture of Gaussians is fit to each class in an unsupervised way we are still performing supervised learning by fitting a model to each class and then computing the Bayes optimal classification [30]:

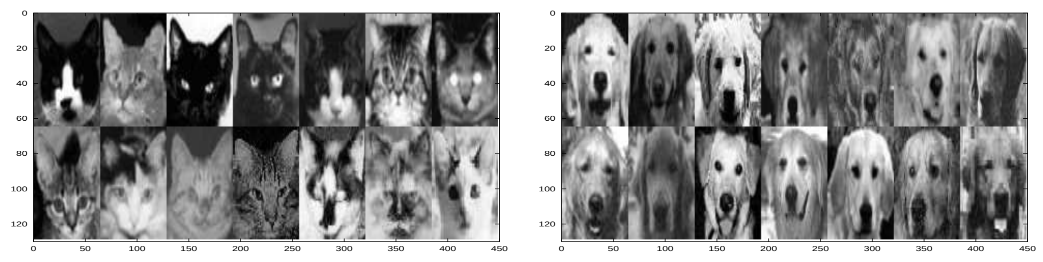
$$\underset{c \in C}{\operatorname{argmax}}\, P(c)P(x|c). \tag{3.9}$$

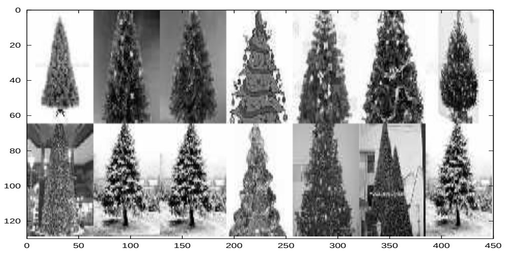
Here, the $P(c)$ value is the prior probability of a data point being from class c and is simply the fraction of the observed data that belong to class c. Unlike the π_c values used during learning and that correspond to the probability of a data point belonging to a particular cluster within a class, this value is unchanging and not learned.

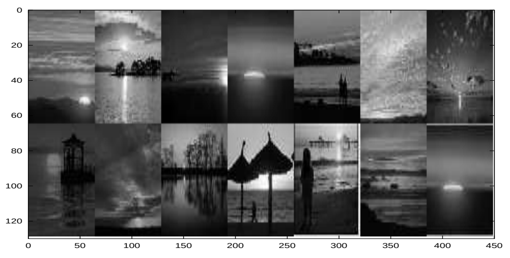
3.5 Data Sets

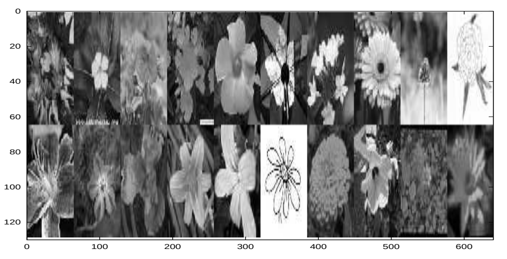
All experiments in this thesis are performed on images from five different classes, though each experiment uses only a subset of these classes. The five classes are: cat/dog, Christmas tree, sunset, flower, and car.

The cat/dog images are frequently used by CSU's computer vision group to test algorithms like the ones discussed in this Thesis and have been used in previous publications. There are 100 cats and 100 dogs. Each image is size 64x64 and are in grayscale with values in the range $[0 - 256]$. The images have been manipulated to include only the face of the animal. The cat/dog images have been hand registered using the eyes. We chose to keep the cat and dog sets together because we have experience with this data set and their overall


(a) The cat/dog class.


(b) Christmas tree class.


(c) Sunset class.


(d) Flower class.

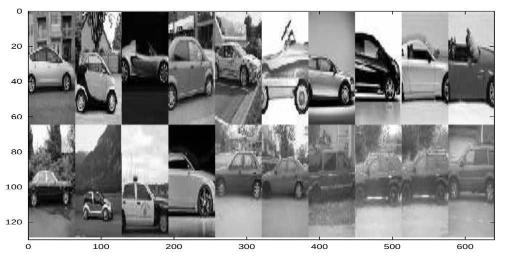

(e) Car side class.

Figure 3.3: Selected images from each of the five classes used in this Thesis.

structures are similar relative to the other classes and we want each data set to have as many data points as possible.

The 101 Christmas tree images were chosen to have a tree in the middle and we strongly favored images where the tree was darker than its surroundings. The 93 sunset images were chosen to have a bright middle region and a dark lower region. While searching for flower images, those with a flower in the center of the image were preferred. Like the sunset class, the flower class looks very challenging due the extreme variation in the number, size, and shape of the flowers. All but five of the car data set images were collected using a digital camera in the Omaha, Nebraska area. The sunset and flower classes, appear to be the most difficult of the three classes due to their relatively large variation in appearance. The car images were taken from the side and allowed to face either to the right or left of the image. Images collected from the Internet were found using Google's image search [1] and Picsearch [2]. After collection, all duplicates were programmatically removed which only removed those images which were exact matches at each pixel.

These five classes of images make for good data sets for testing these mixture of Gaussian variants because they each have a distinctive shape making their classification obvious but are real images so they are not trivial to classify for a computer. The classification is done using a clustering algorithm that has no preference toward imagery – rather, they are very general and used in many different domains. Furthermore, the performance of these methods are being evaluated independently of any end-to-end vision system which does some form of feature extraction prior to classification or further processing prior to, or after classification. Therefore it is important to use classes that are significantly different from one another: classes with gross, image-wide differences but still representative of real-world images.

The flower, car, sunset, and Christmas tree images are rectangularly shaped, color images so they must be converted to square, grayscale images to match the cat/dog class. First, the images are converted from their RGB representation to grayscale by averaging the color values to a single, floating point number between zero and 255. These images are then cropped along their largest dimension (row or column) to make them square. Image cropping is done with special attention to preserving the center of each image because each class is designed to have the object of interest in the center of the image. Finally, the images are resized to either 32x32 or 64x64 using bilinear interpolation [18]. When an experiment uses 32x32 images, the cat/dog images are also resized using bilinear interpolation.

The data is broken into two datasets. Data set one is comprised of the cat/dog, sunset, and Christmas tree classes while data set two is made up of the flower and car classes.

3.6 Experimental Methodology

Each of the three algorithms have experimental parameters. For data set one, all of them have three experimental parameters for the number of clusters used to model each class. For data set two there are two such experimental parameters. The two shrinkage methods have an additional experimental parameter which controls the λ value for each of the clusters in each of the classes. In other words, in these experiments, the λ values are the same for each cluster of each class which dramatically reduces the number of experimental parameters. The number of Givens rotations used to estimate the covariance was not initially treated as an experimental parameter because the authors solve for the optimal number. Starting in Section 4.2 we convert the number of Givens rotations to an experimental parameter because it provides better results when selected via cross-validation.

The performance of an algorithm is measured by classification accuracy. In a given experiment algorithm comparison is summarized by a single value: the classification accuracy averaged over 32 different, random partitions of the data set. This is done using the cross-validation method.

In cross-validation, the best set of experimental parameters for each data set partitioning is selected. This is done by randomly breaking the data set into a training, validation, and testing set. For each parameter setting and algorithm, one model is learned using the training set and then used to classify the validation set. For each algorithm, the best performing experimental parameter setting with respect to the validation set is selected as the representative for that algorithm. Then each algorithm's representative model is used to classify the testing set. It is the testing set classification accuracy that is considered the true measure of that algorithm's performance on a particular random selection of data points from the overall data set. This process is repeated for 32 total random selections of data points from the entire data set. The data points for each class are selected independently from the other classes maintaining the prior probabilities across classes. Those classification accuracies on their respective testing sets are taken together to get a statistically significant number of observations for each of the algorithms.

In individual experiments, the data is further resized to 32x32 or pre-processed using principal components analysis. When the images are resized to 32x32, the process outlined above is run from the beginning to create 32x32 images while the cat/dog images are simply resized to 32x32 using bilinear interpolation. In our initial experiments we reduce the dimensionality as little as possible and PCA is used primarily to decrease the run times of the algorithms. The maximum possible rank for a covariance matrix is $\min(M, N-1)$ where M is the dimensionality of the data and N is the number of data points. In our data sets N is always much smaller than M. At times we were forced to reduce the dimensionality to something less than $N-1$. The subspace returned by PCA are ordered decreasingly by the amount of variance captured. If we reduce the dimensionality to anything lower than $N-1$ it is because the removed principal components capture such little data that they had zero variance (to machine precision) which caused singularity in the covariance matrices. In subsequent experiments the performance of these algorithms with respect to dimensionality is of interest and PCA is used to change the dimensionality of the data.

Chapter 4

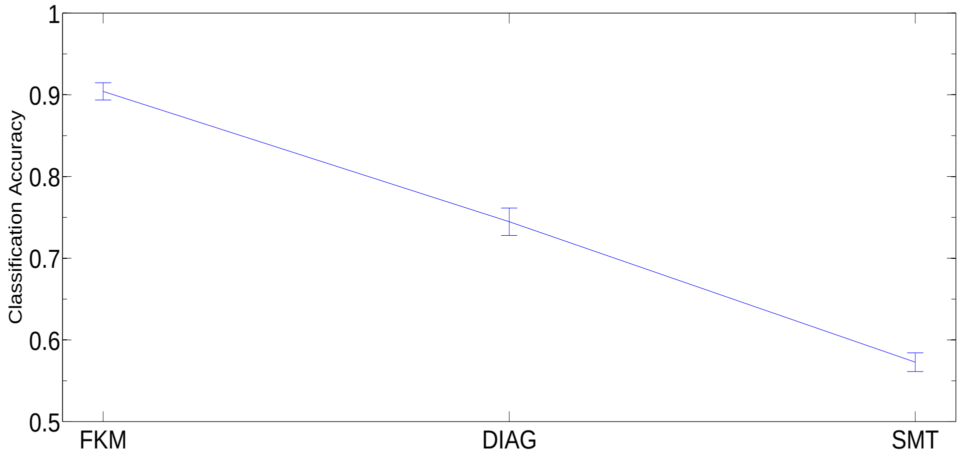
Experimental Results

In this Chapter, we take a look at the results of experiments designed to evaluate the performance of our supervised mixture of Gaussians algorithm with covariance regularization. We use two data sets comprised of the five classes described in Section 3.5. Data set one is comprised of the Christmas tree, sunset, and cat/dog classes while data set two is made up of the flower and car side classes.

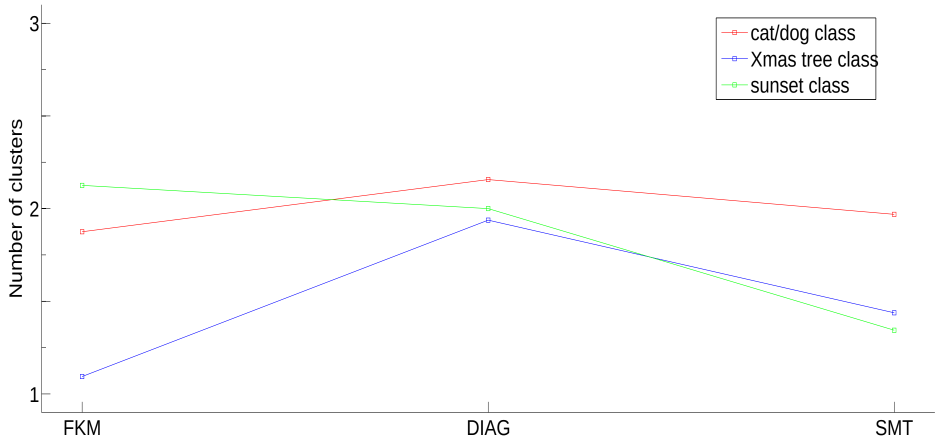
4.1 Initial Experimentation

We initially ran the three methods on data set one reduced to 383 dimensions. This is two fewer than the maximum number allowed theoretically [27]. As mentioned before, we used the maximum number of PCA dimensions while still getting variance in the smallest dimensions. The detailed results from this experiment are shown in Table A.1 and Figure 4.1 provide a graphical summary. Figure 4.1(a) shows the classification accuracies of the three methods' best performing experimental parameter combinations for each of the 32 partitions as selected via cross-validation. Figure 4.1(b) plots the mean C_1, C_2, and C_3 values as selected via cross-validation.

Looking at Figure 4.1 and the left-most column of Figure 4.2, we see that FKM is a clear winner in this preliminary experiment. Interestingly, according to Figure 4.2(b) and Table A.1, FKM with $\lambda = 0.8$ is ubiquitous in its superior classification accuracy for this experiment. DIAG also shows a preference toward a particular value of λ but its preferred values are at the opposite end of the range of λ values used in these experiments. DIAG



(a) Classification accuracies of the three versions of MoG. Error bars represent the 95% confidence interval.



(b) Mean number of clusters selected via cross-validation for each algorithm for each class.

Figure 4.1: Graphical summary of the preliminary experiment's results shown in Table A.1

favors $\lambda = 0.2$ or $\lambda = 0.4$ and, at this point, it is not clear why this is the case. Figure 4.6 shows the mean classification accuracy of DIAG across all λ values. Looking at the mean value for the 383-dimensional data we see $\lambda = 0.8$ and $\lambda = 0.4$ are ahead of the other λ values for DIAG. There does not appear to be a pattern relating λ and classification accuracy for DIAG here. SMT performed woefully compared to the other MoG versions in this experiment; its mean classification accuracy is not much better than if it had assigned every data point to the cat/dog class which would be 51.95% accuracy. However a glimpse at the classifications given by the SMT version show that this is not the case. FKM only required one cluster to represent the Christmas tree data. This is sensible in that it is the most uniform-looking class of the three in data set one. SMT appears to require fewer clusters per class except for the cat/dog class where there are arguably two classes. To ensure that these results were not specific to data set one, we ran a similar experiment on data set two and found very similar results (shown in the second measurement, 823 dimensions, from the left in Figure 4.2 and Table A.2). The rerun of this experiment using data set two is different from the original in that the data was projected to 823 dimensions because there were more data points available and, therefore, more principal components could be computed.

One possible explanation for these results is that the PCA pre-processing whitened the data such that very little covariance (or variance) was needed to classify the data. The λ value of 0.8 must have been sufficient because the Christmas tree class only required one cluster to be properly classified.

An argument against the hypothesis that PCA pre-processing is behind the dominance of FKM with $\lambda = 0.8$ is that each principal component is a linear combination of the data and, therefore, probably would not whiten each class equally or even whiten each class at all. It would require that the PCA pre-processing allowed each class to project sparsely onto the global PCA subspace. Such a projection is highly unlikely and is the purpose of the sparse PCA algorithm [42]. Looking at the eigenvectors shown in Figure 4.4, it appears that such a sparse projection did not take place.

In order to account for the possibility that PCA pre-processing is accountable for the results of Tables A.1 and A.2 we perform the same experiment on data set one but without the PCA pre-processing. However to keep processing time to a minimum, the data's dimensionality is still reduced. This time we use bilinear interpolation to decrease the image size from 64x64 to 32x32 or from 4096 to 1024 dimensions. SMT took too long to estimate a covariance matrix, mostly due to the steps taken by the algorithm to determine the optimal number of Givens rotations to use (discussed in Section 3.3), for this experiment so it was omitted.

Table A.3 shows the detailed results of this experiment and the mean classification accuracies in the leftmost column of Figure 4.2. The preferred λ value for FKM increased to the max value possible: 1.0. Also, FKM's average classification accuracy dropped by 10%. DIAG also had a drop in classification accuracy but it was more modest. The λ values selected by cross-validation also increased and included all experimental values of λ but it primarily favored $\lambda = 1$ and $\lambda = 0.2$. This can be seen by comparing the plots at the 383 and 1024 dimensionalities in Figure 4.2. Note that there is no variance of the λ value for 1024 dimensional data with FKM – it is always selected to be one.

This result does not favor the hypothesis that PCA pre-processing was responsible for the results of Tables A.1 and A.2. If data whitening is responsible for the prevalence of $\lambda = 0.8$, or simply a higher value of λ, you would expect the favored λ values to decrease, not increase when the whitening step is removed.

Thus far, the performances of DIAG and SMT are a bit puzzling. However, it appears that varying data dimensionality has a large influence on the performance of our FKM and the λ values selected via cross-validation. In the next section we continue along this path.

4.2 Examining λ with respect to varying dimensionality

As discussed in the previous Section, dimensionality of the data appears to play a significant role in the classification accuracy and the desired λ value. In this Section we perform experiments similar to those of the previous Section, but here we vary the dimensionality of the data using PCA.

The data dimensionality of the two data sets is further reduced from 383 and 823 respectively to 300, 150, 50, and 3. This is done by reducing the number of eigenvectors kept to form the subspace that the data is projected onto prior to fitting the MoG model.

In light of SMT's poor performance on the initial experiments we added an experimental parameter to SMT which controls the number of Givens rotations combined to create the matrix of eigenvectors. This value is selected from $\{5, 50, 150, 300\}$ using cross-validation in the same way the number of clusters are selected. This addition involved a small modification of the SMT code written by the authors of [6] where the computation of the optimal number of Givens rotations is replaced with a constant, K. Table 4.1 shows the difference in performance between the use of the calculated, optimal number of Givens rotations and the selection of this number using cross-validation. As can be seen, ignoring the computed, optimal number of Givens rotations and selecting that number using cross-validation as an experimental parameter gives superior results and, therefore, is used to evaluate SMT from this point forward.

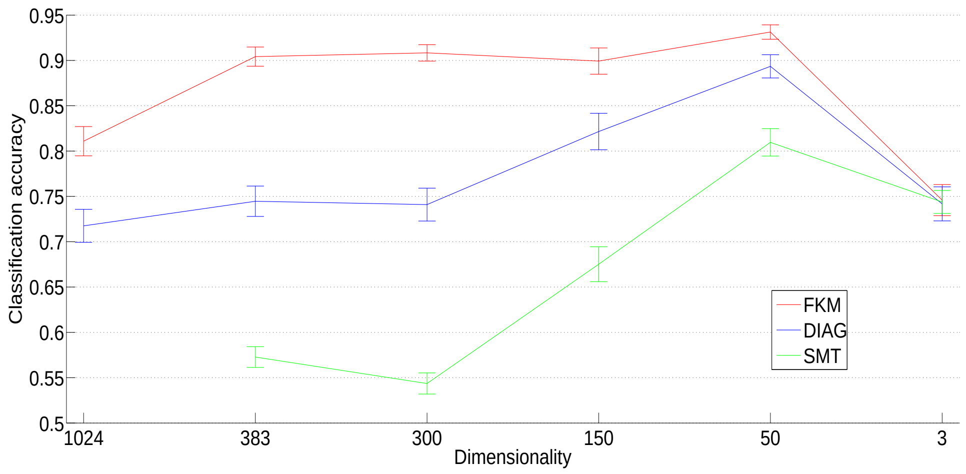
	accuracy
D=3	
optimal calculation	0.74383 ± 0.0013402
cross-validation	0.75123 ± 0.0024055
D=50	
optimal calculation	0.80962 ± 0.0019098
cross-validation	0.86554 ± 0.0016055
D=150	
optimal calculation	0.67516 ± 0.0030989
cross-validation	0.76069 ± 0.0032106
D=300	
optimal calculation	0.54359 ± 0.001133
cross-validation	0.69120 ± 0.0041773

Table 4.1: Comparison between use of calculated, optimal number of Givens rotations and the selection of this number using cross-validation on data set one for varying levels of dimensionality. The cross-validation results are a summary of the results shown in Tables A.4 – A.7.

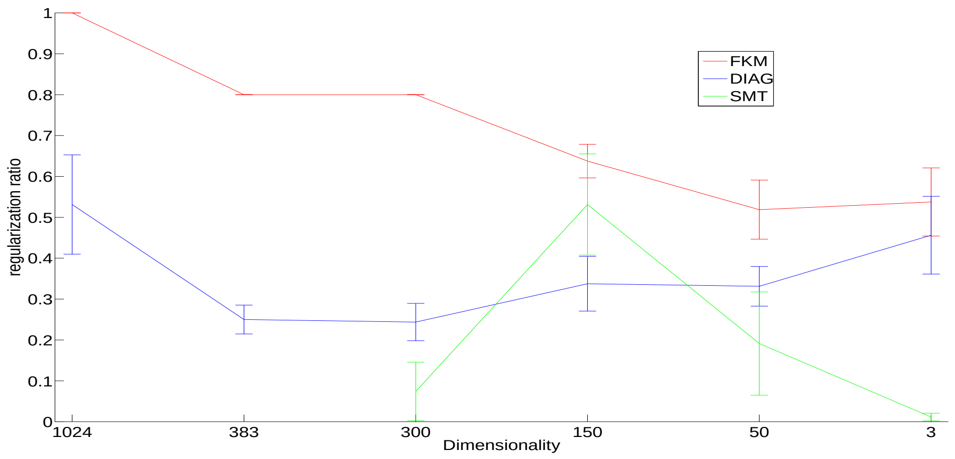
Tables A.4 – A.7 show the results for FKM, DIAG, and SMT for PCA are-processed images with decreasing numbers of principal components kept for creation of the subspace on data set one. Tables A.8 – A.11 show the detailed results of the same experiment for data set two but without SMT.

One interesting, although well known, observation is that there is an optimal value for D which is the number of principal components kept [25]. This point about optimal dimensionality reduction is regarding achieving maximal dimensionality reduction while not adversely affecting the ability to cluster the data. From these results, it is clear that this point exists somewhere between D=50 and D=3 for both data sets.

As the dimensionality decreases, the difference in the performance between FKM and DIAG decreases as well. For reasons not elucidated in these experiments DIAG fails to ever prefer a high λ value. However, as the λ value preferred by FKM decreases and becomes



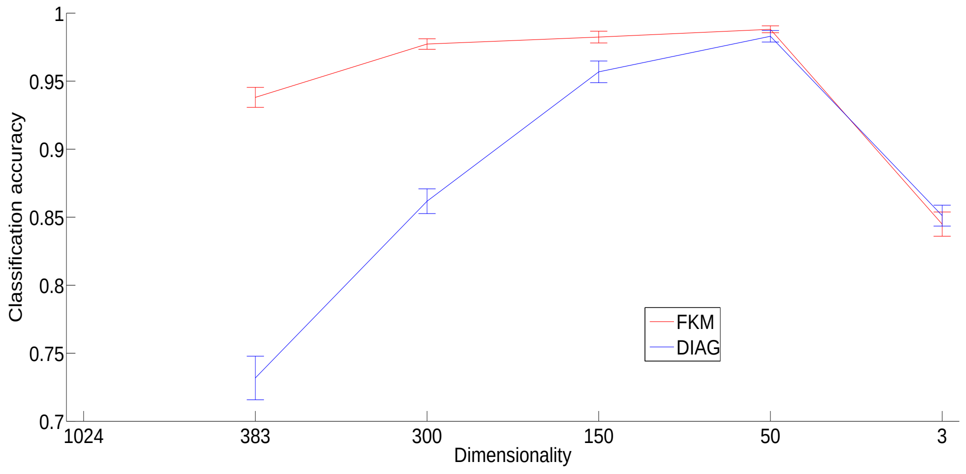
(a) Classification accuracies of the three versions of MoG accross all dimensionalities. Error bars represent the 95% confidence interval.



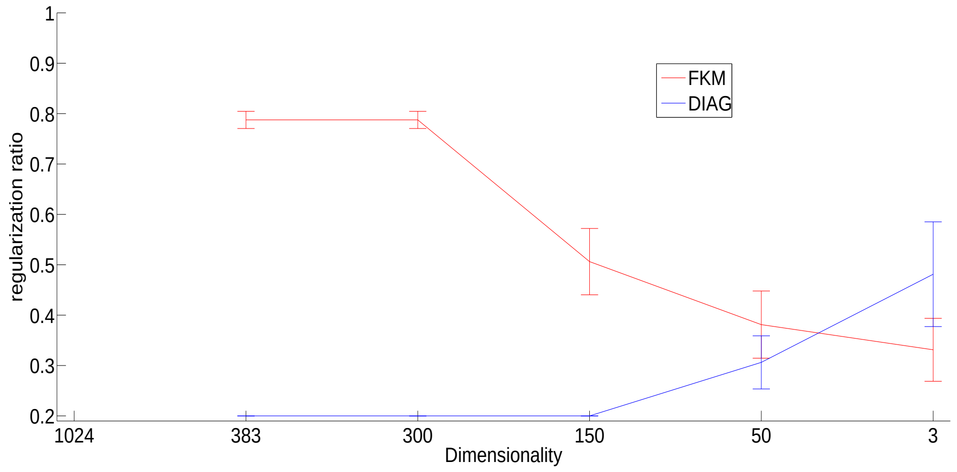
(b) Mean value of a ratio designed to capture the amount of covariance regularization is used as a fraction of the total possible covariance regularization allowed in the experimentation. For the shrinkage methods, FKM and DIAG, this value is simply equal to λ. For SMT we computed this ratio as $\frac{|K-D|}{\max(|D-K_{\min}|,|D-K_{\max}|)}$.

Figure 4.2: Graphical summary of all experiments performed on data set one.

closer to the value typically preferred by DIAG, their performance becomes very similar. Although DIAG adherence to low λ values is still a mystery, their comparative performance on lower dimensionality data is not. As λ decreases for FKM, the two methods become increasingly similar (although they are never identical in these experiments because we do not allow $\lambda = 0$). Because they become more similar as FKM's λ value drops, it is expected that their classification accuracy and desired λ values would become closer as well. This,



(a) Classification accuracies of the three versions of MoG accross all dimensionalities. Error bars represent the 95% confidence interval.



(b) Mean value of a ratio designed to capture the amount of covariance regularization is used as a fraction of the total possible covariance regularization allowed in the experimentation. For the shrinkage methods, FKM and DIAG, this value is simply equal to λ. For SMT we computed this ratio as $\frac{|K-D|}{\max(|D-K_{\min}|,|D-K_{\max}|)}$.

Figure 4.3: Graphical summary of all experiments performed on data set two.

in fact, is what is seen in Figures 4.2 and 4.3.

For SMT, the best level of covariance regularization generally follows the number of dimensions of the data set. The 150-dimensional data, however, has a great deal of variation from using 150 rotations. Because the amount of covariance estimation is a number between five and 300, we created a ratio: $\frac{|K-D|}{\max(|D-K_{\min}|,|D-K_{\max}|)}$ to bring it within the same range as the shrinkage parameter, λ. This ratio is intended to capture the how far the number of rotations used differs from the number of dimensions in the data.

Also of note is DIAG's slow increase in desired λ from a mean of nearly 0.3 to 0.5 between 300 and 3 dimensions. Given the disparity in the mean performance between FKM and DIAG until, at least, $D = 50$, we feel that DIAG's preferred λ value is not indicative of a true rise in the best amount of shrinkage for classification. Instead the convergence of the classification accuracy and chosen λ is the critical piece of information indicating that their convergence represents a continued reduction in covariance regularization. Furthermore, looking at the error bars, this change may not be statistically significant.

Nevertheless, it is appears that for the data used in this Thesis and when using shrinkage methods to model a Gaussian in a mixture of Gaussians, the best λ will decrease away from one as the dimensionality decreases.

Figure 4.5 shows two views of one of the training sets randomly selected from data set one projected onto the three largest principal components. We can see that the three classes, indicated by differing colors and shapes, are no longer separable hence the poor performance across algorithms. Furthermore, it is apparent that pre-processing the data set using a global PCA does not whiten the data. These data sets are clearly asymmetric and the green "x" class (the Christmas tree class) is multimodal. The multimodality of the Christmas tree class stands in stark contrast to its 383 dimensional representation which is best modeled using a single, nearly spherical Gaussian. The multimodality of the Christmas tree class shows the effect of data set one's first eigenvector shown in Figure 4.4. The first eigenvector is clearly a Christmas tree shape. Although the signs of the eigenvector elements are undefined [25], the darkness of the Christmas tree with respect to its background would greatly affect how that image projects onto the largest eigenvector. Christmas tree images with darker Christmas trees would most likely project on the negative side of the axis defined by the first eigenvector while images with a lighter Christmas tree would most likely project on the positive side of the axis.

Figure 4.6 shows the classification accuracy for both shrinkage algorithms for the five chosen λ values across the six dimensionality versions of data set one on the test set. As seen in Table A.3, $\lambda = 1.0$ is the best value for FKM on the 32x32 data. This is most likely because the many free parameters to learn for lower λ values and incredible distances between points in high dimension. Also $\lambda = 1.0$ has the most consistent performance for FKM across dimensionalities.

DIAG, by comparison, does not show consistent performance for $\lambda = 1$ or any value of λ. Instead $\lambda = 1$ varies as all the other λ values do peaking at 150 dimensions. This performance is shared by the $\lambda \neq 1.0$ FKM runs. This implies that $\lambda = 1$ with FKM is fundamentally different from all the other methods. FKM with $\lambda = 1$ is the minimum number of parameters possible in this experiment and has the most generic covariance matrix.

Figure 4.6 also shows that $\lambda = 0.8$ is superior to the other values with $\lambda = 0.6$ and $\lambda = 1.0$ coming in second and third respectively at 383 dimensions for FKM. Again, this is in agreement with what we see in Figure 4.2. The lower dimension projections are less difficult to see a clear benefit to any of the λ values. However, it is clear that high values of λ are no longer superior and that $\lambda = 1.0$ should not be used in lower dimensions.

Figure 4.6 shows us that DIAG, despite its inferior performance, follows a similar story to that of FKM in that the performance with respect to various λ values are more spread out at the $D = 1024$ and $D = 3$ values and are bunched up at the $D = 150$ data sets. Although to a lesser degree than FKM, DIAG with $\lambda = 1.0$ starts out toward the head of the pack and then falls to the rear of the pack as dimensionality decreases while lower values of λ assume the top spots at the lower dimensionality data sets.

Figure 4.7(a) shows an example of what was learned for the cluster means by FKM on data set one reduced to 50 dimensions using two clusters per class. Figure 4.7(b) shows the learned cluster mean parameter values for DIAG for data set one. The learned parameter values for the same experimental parameter settings are greatly varied due to the fact that EM is highly susceptible to local optima and the local optima a given model will be trapped in is determined by the initialization values. Class one, in many of the runs, is broken into cat and dog clusters. When it did not break into cat and dog clusters it typically clustered based upon background darkness. The Christmas tree class many times has very similar means but in some cases used the two clusters to represent light and dark trees. Unsurprisingly, the sunset class means are the most difficult to differentiate. This is probably due to the fact that this class is the most varied among data set one though a few times class three appears to cluster based upon the position of the sun in the images.

Figure 4.7(c) shows an example of learned parameter values for FKM on data set one reduced to 150 dimensions. This example is representative of what we typically see for this data set and nearly all experimental parameter settings for FKM. Notice that the cluster means within a class look similar to each other. In fact, these means are closer to each other than the closest data points from their respective classes.

4.3 Discussion

In this Chapter we have seen results showing that FKM provides the best classification accuracy across dimensionalities, across data sets when compared to an alternative version of shrinkage, DIAG, and the sparse matrix transformation covariance regularization methods with the only exception being when projected down to three dimensions which is a suboptimal amount of data reduction.

The suboptimal projection of the data down to three dimensions is the only instance where SMT outperforms either of the shrinkage methods for any of the 32 random partitions of the data. A closer look at SMT's behavior during clustering did not reveal anything which would indicate that the algorithm is suffering from numerical instability and our use of the algorithm is theoretically sound. We feel encouraged in our results for the SMT approach by the fact that the SMT implementation used in our experiments was developed by the SMT's authors [6].

Our experiments show that FKM's performance was not a product of the PCA preprocessing. What we do see is that the best combination of experimental parameters of these versions of the mixture of Gaussians algorithm are highly dependent upon the dimensionality of the data. This observation goes further than "shrinkage is a way to make a covariance matrix non-singular," which is the original intent of shrinkage. Our results show that there is a true preference toward a more spherical Gaussian as the dimensionality increases.

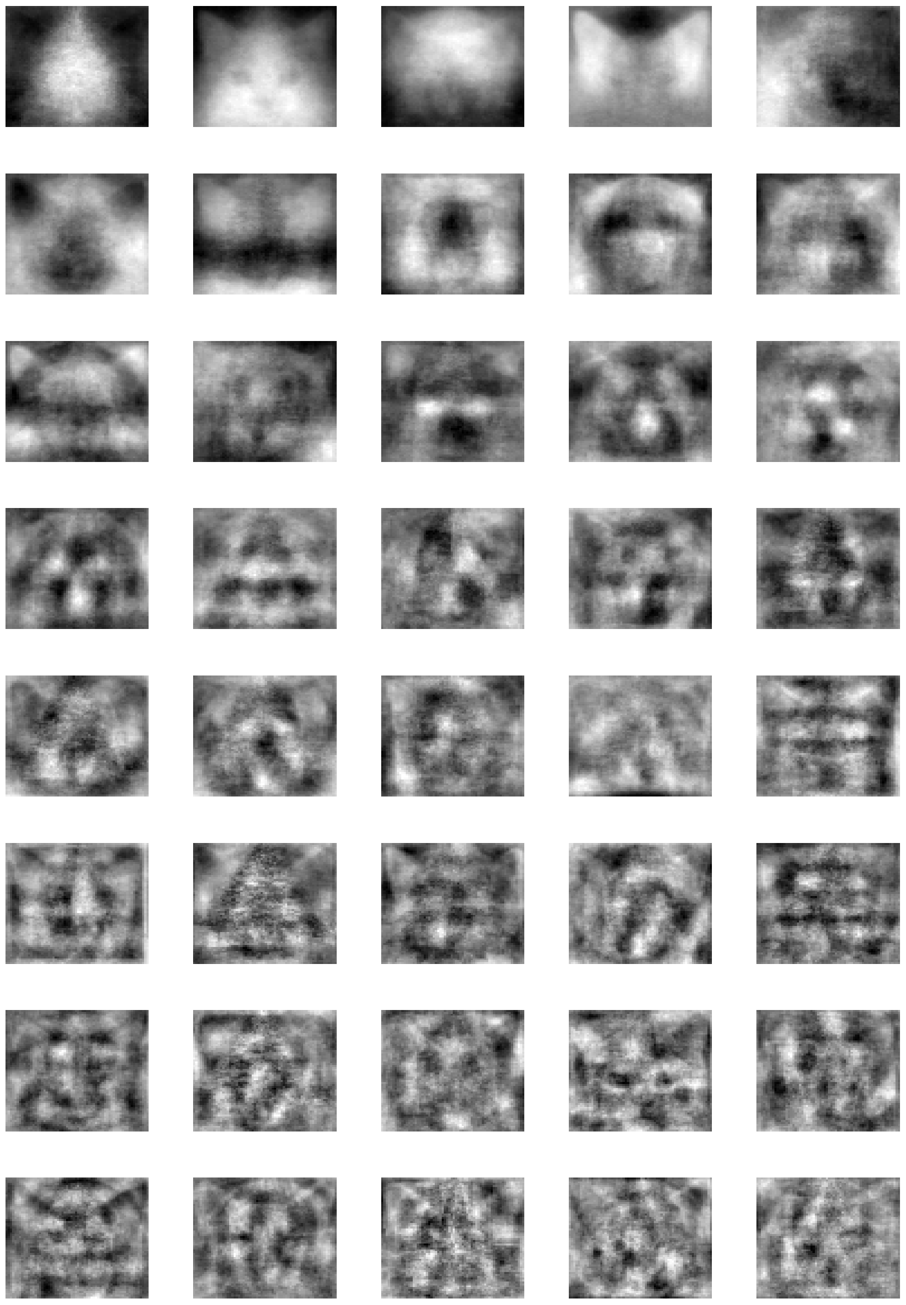
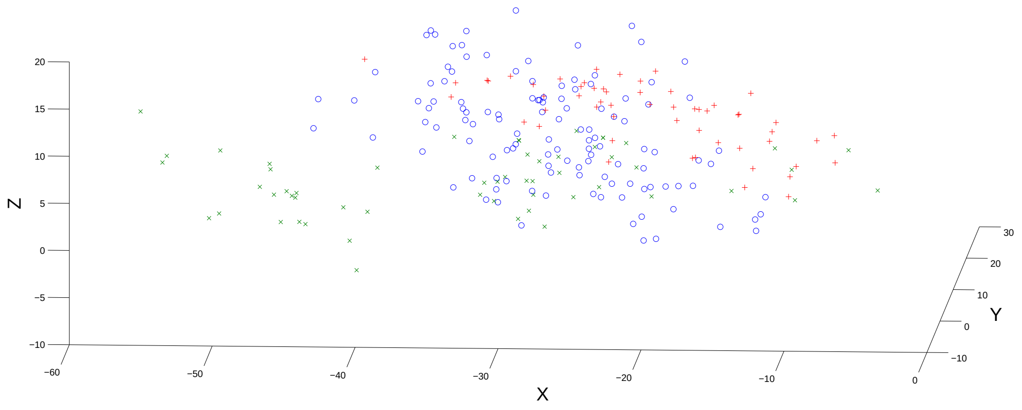


Figure 4.4: First 40 eigenvectors (or principal components) for data set one. The eigenvectors are sorted from left to right, top to bottom in order of their respective eigenvalues.



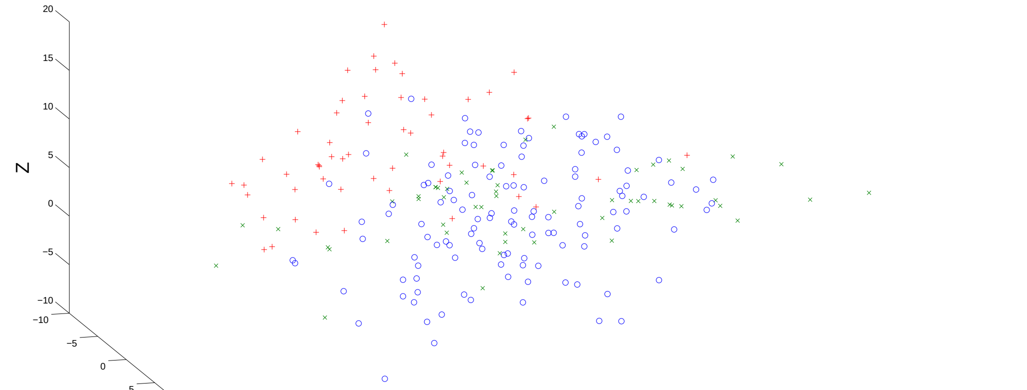
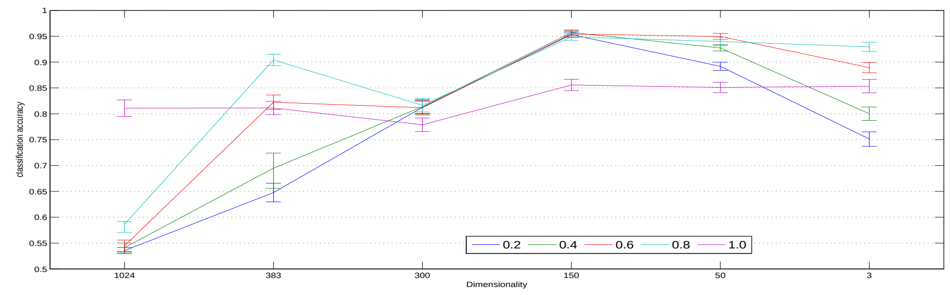
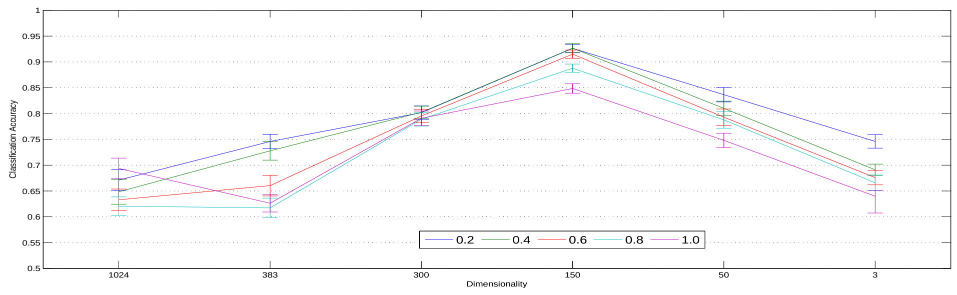


Figure 4.5: A couple perspectives of the third training set of data set one projected to three principal components.

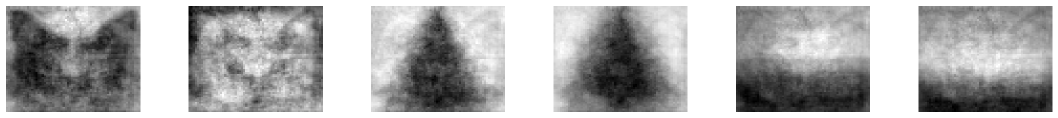


(a) FKM.

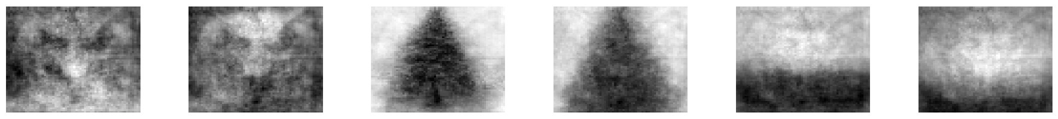


(b) DIAG.

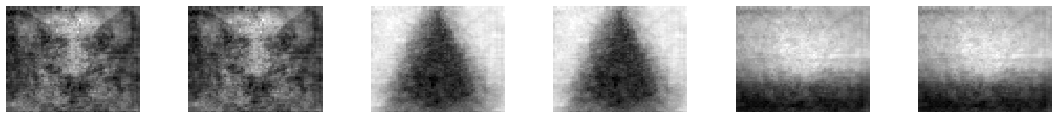
Figure 4.6: Classification accuracies of both shrinkage algorithm versions for the selected λ values across the six different dimensionalities for data set one. The lines represent the mean accuracies on the test sets for the best C_1, C_2, and C_3 values as selected by cross-validation across all 32 partitions of the data set and the error bars represent the 95% confidence interval. The legend specifies which color is associated with a particular λ value.



(a) FKM on 50 dimensional data



(b) DIAG on 50 dimensional data



(c) FKM on 150 dimensional data

Figure 4.7: Cluster mean parameter values. These images were created by projecting the cluster means out of the PCA subspace and back into the 4096-dimensional space associated with the original 64x64 data.

Chapter 5

Discussion

In this Chapter we begin with possible explanations for the results presented in Chapter 4. We then move on to a summary of the conclusions made in this Thesis. Finally, we provide some possible directions for future work.

5.1 Two potential explanations of results

There are two factors, discussed in detail in the forthcoming Sections, at work to give spherical Gaussians, as computed via FKM, an advantage in high dimensions. The first factor is that the layout of the class data points becomes less meaningful as the dimensionality increases. The second factor is that the spherical Gaussians have far fewer model parameters to fit to the data and that shrinkage effectively mitigates the nasty effects of over-fitting which occur when there are far too many parameters to fit given the amount of data available.

5.1.1 High dimensional data may not need anything more than a spherical Gaussian

As the dimensionality increases, data points become increasingly sparse and increasingly uniform. Steinbach, et al., provide a nice, simple example of how this increased sparseness comes about [36]. In their example they begin by asking the reader to consider 100, one-dimensional data points distributed uniformly on the interval $[0, 1]$ and then breaking that interval into 10 regions that each measure 0.1 across. In this scenario there would almost certainly be one data point in each cell. They then further ask the reader to consider 100 data points distributed uniformly over a unit square and break that area into 100 regions of 0.1 on each side. With only 100 data points it is highly likely that at least one of the cells will be empty. Increasing dimensionality to three we now have 1000 cells at least 900 of which must be empty. As we can see, the data is becoming increasingly sparse and it is doing so at an alarming rate.

Steinbach, et al., also discuss another aspect of high dimensional data: that the data becomes increasingly uniform as the dimensionality increases. This can be seen by comparing a ratio of the difference between the distance of the furthest data point and the distance of the nearest data point with the distance of the nearest data point (5.1).

$$\lim_{D\to\infty} \frac{\text{MaxDist} - \text{MinDist}}{\text{MinDist}} = 0. \tag{5.1}$$

What (5.1) implies is that, in high dimensions, if you formed a sphere centered at a given data point and set its radius to be equal to MinDist that expanding that radius just a little would capture many additional data points [36]. This further implies that the data distribution is losing its shape as dimensionality increases and therefore a mixture of Gaussians constrained to use only spherical Gaussian distributions (Euclidean distance) should be sufficient.

5.1.2 More parameters require more data

The more often discussed ugly side of clustering data in high dimensions is the exponential increase in parameters that must be fit – even in models like LDA which drastically reduce the number of parameters that must be fit by sharing a covariance matrix between classes.

Estimating a covariance matrix requires fitting $\frac{D(D+1)}{2}$ parameters where D is the dimensionality of the data. This is because there are D elements on the diagonal and $\frac{D(D-1)}{2}$ independent, off-diagonal elements [11]. In our application of the mixture of Gaussians, we have multiple clusters per class and each cluster with its own covariance matrix.

The SMT algorithm estimates the covariance matrix parameters using a method that looks for correlations in the data and uses those covariances to build an orthogonal matrix. The SMT algorithm uses a complex calculation to determine how many rotations (correlations) to use to estimate the orthonormal matrix and associated eigenvalues. Duda and Hart point out that a covariance matrix estimated from an insufficient number of data points may suffer from "accidental correlations" which were present in the data but were not a part of the underlying distribution generating the data and would not be present in the presence of a sufficient amount of data [11, pg 68]. The presence of these accidental correlations could be responsible for some of SMT's struggles.

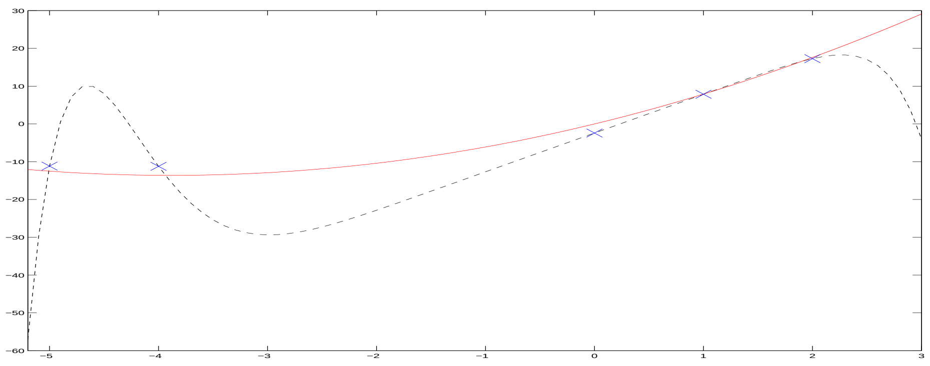


Figure 5.1: Example showing over-fitting. The data points were generated by the solid line, plus a little Normally distributed noise. The dotted line is a 10th degree polynomial fit to the data.

Duda and Hart also provide a nice description of how an insufficient amount of data will affect the learned model parameters or will make an incorrect, overly complex model overfit the training data. Following their example, notice the five data points in Figure 5.1. These data points were sampled from the solid, red parabola with a small amount of Normally distributed noise added to each point. The dashed line is a 10th order polynomial which was fit to the data. There are too few data points to fit a 10th order polynomial but there is an excellent fit nonetheless. The problem occurs when we try to generalize this learned, 10th degree polynomial to unseen data. We now see that, although it fit the training data very well, better than the generating curve, it has a poor fit to the unseen data which would follow the generative curve.

5.1.3 Putting it all together

The two major factors affecting classifier performance are the suitability of the classifiers' basic assumptions and constraints (e. g., diagonal covariance or all classes have the same covariance) which we refer to as the shape of the model and the ability to have enough data to learn model parameter values which will generalize well to unseen data. These two factors certainly appear to combine to favor FKM over the other two approaches experimented with in this Thesis as the preferred method for the high-dimensional, small sample size problem.

FKM is, without a doubt, the method with the lowest number of free parameters and SMT has the most. This also describes the opposite ends of our algorithm performance spectrum. DIAG always uses more parameters than FKM for like λ values because of its use of the covariance matrix diagonal in the target matrix.

Figure 5.2 shows three graphs comparing the average performance of FKM on three selected dimensionalities of data set one. Each line in the graph corresponds to a different value of λ. At 1024 dimensions, $\lambda = 1$ is the clear victor with all other values performing equally poorly. At 150 dimensions we see that $\lambda = 1$ has been overtaken by $\lambda = 0.8$ and is roughly equal to $\lambda = 0.6$ in terms of performance. Finally, when we get to 50 dimensions, $\lambda = 1$ is far below the other λ values which perform equally well. The omitted dimensionalities' plots continue this trend and show $\lambda = 1$ slowly sinking or the $\lambda \neq 1$ lines slowly rising above $\lambda = 1$ as the dimensionality decreases.

Also worth noting is the preference of $\lambda = 0.8$ toward $C_2 = 1$ at 300 and 150 dimensions. This reveals that $\lambda = 0.8$ is sufficient to model the tree class with a single cluster at those dimensionalities. The preference of $\lambda = 1$ toward $C_1 > 1$ indicates that $\lambda = 1$ is not capable of representing the cat/dog class with only one cluster at each level of dimensionality.

These plots are difficult to interpret with respect to determining which of the two mentioned competing high dimensional phenomena are more responsible for the shift in favors for λ values of FKM as the dimensionality decreases. Figure 5.3, however, does shed some light on the subject. Figure 5.3 plots the average class accuracy of FKM with $\lambda = 1$ across all experimental parameter values governing the number of clusters used for each class. Each line corresponds to a different dimensionality of the data. What we see here is that, with the exception of the data projected onto three dimensions which was deleterious to the behavior of all MoG versions and λ values, is that $\lambda = 1$ with FKM performed equally across dimensionalities. Therefore, you can look back at Figure 5.3 using $\lambda = 1$ as a landmark to show how the other λ values' performances changed with decreasing dimensionality. Furthermore, the steady performance of $\lambda = 1$ implies that the data is not changing considerably from 1024 to 50 dimensions with respect to clusterability. This would further imply that learning would benefit from the use of a Gaussian with covariance in high dimension as well but there are not enough data points to learn those parameters. This would also fit with an interpretation of the DIAG and SMT versions that they, even when using a reduced number of parameters or a shrinkage parameter that reduces the effects of overfitting, have too many parameters and are hopeless until the data dimensionality is reduced to a manageable number.

Figure 5.4 shows this in a single graphic for each shrinkage version. Figure 5.4(a) shows the average classification accuracy for FKM across dimensionality and λ values. The rightmost bars correspond to $\lambda = 1$ and are fairly constant across dimensionalities. What is difficult to see from this bar chart is that the entire plot peaks at $\lambda = 0.6$ with 50 dimensions. If this were drawn as a surface, a steep incline would be visible starting at 1024 dimensions with $\lambda = 0.2$ and peaking at around $\lambda = 0.6$ and $\lambda = 0.8$ at 50 dimensions.

Figure 5.4(b) shows the results for DIAG. Where FKM peaks at $\lambda = 0.6$ with 50 dimensions this peaks at $\lambda = 0.4$ and 50 dimensions and that the incline moves in the opposite way: with its lowest point at $\lambda = 1.0$ in 1024 dimensions. DIAG with $\lambda = 1$ behaves similarly to the other values of λ because it never has the extreme parsimony of parameters that FKM is capable of. As soon as the dimensionality is reduced to a point where the number of parameters used to represent the variance only is manageable (happens somewhere between 383 and 1024 dimensions) some off-diagonal covariance elements are desirable and $\lambda = 1$ is at an instant disadvantage.

5.2 Conclusions

The main idea of this Thesis was to present a novel form of the mixture of Gaussians model where we use covariance regularization techniques to improve the generality of the cluster covariance parameters. In the Background Chapter we revisited popular MoG models and see that our novel forms of MoG may not be so novel but three new methods of performing the types of modifications that have been applied to MoG for at least two decades. We view the various forms of MoG through the lens of covariance regularization - a method which can, when used on data with fewer data points than dimensions, bring MoG closer to the impossible, full covariance version of MoG by simultaneously making the covariance matrix invertible while improving the model's performance on unseen data.

Our experiments include three types of covariance regularization: two versions of the long-standing shrinkage method and the new sparse matrix transform method. What we find is that the covariance regularization method that is the simplest is the best for both data sets and for each level of dimensionality with one unimportant exception. In our experiments, the simplest form was a shrinkage method which augments the empirical covariance matrix with a weighted identity matrix.

Our experiments have shown this method of image classification to be highly effective and may be a positive alternative to the popular methods of linear and quadratic discriminant analysis. Using a MoG to model a class of data is more flexible than the single Gaussian LDA and QDA methods. A greater emphasis on covariance regularization may make MoG just as robust to the high dimensional small sample size problem – especially when using our simplest shrinkage method.

We also see that use of the optimal number of Givens rotations as computed by (3.5) is out performed by controlling this value as an experimental parameter and selecting it via cross-validation. The result that the best number of Givens rotations mostly matches the dimensionality of the data is not surprising. However this value has the opposite relationship with dimensionality than that of the shrinkage parameter – the higher the dimensionality, the more Givens rotations desired to accurately model the data.

Along the way we were able to use our model to begin thinking about why high dimensional data is difficult: is it simply a result of too many parameters and too little data or does high dimensional data require nothing more than a spherical Gaussian? While our experiments do not provide definitive evidence for either hypothesis, it does appear that the number of parameters was the primary factor in classification accuracy. The questions of which of these two factors (or both) makes the most difference and should we continue to attempt to invent further versions of mixture of Gaussians or just focus on covariance regularization are intriguing to us. In the Background Chapter we try to make the argument that using covariance regularization with MoG has been overlooked but is related to prior, seminal works dealing with MoG. In addition to ignoring the connections between these MoG-related models and covariance regularization, invariance to image transformations and methods like LDA took precedence over trying to make a simple MoG work in high dimensions. We hope that this Thesis will renew interest in MoG by researchers in computer vision.

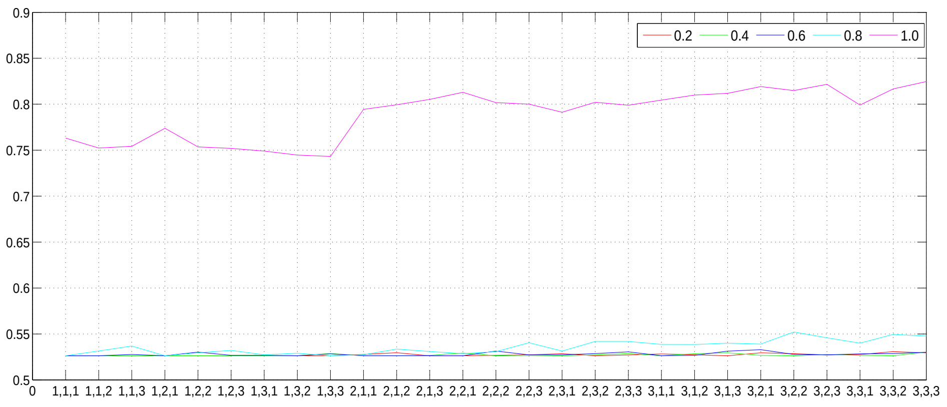
5.3 Future work

There are a few limitations to our work in this Thesis, however. In our experiments we select the shrinkage parameter value using cross-validation. As pointed out in [35], Ledoit and Wolf provide a proof of a computation to solve for the optimal shrinkage parameter value [28]. Furthermore, we do not allow the shrinkage parameter value to change between classes or between clusters within a class. We feel that there is a high possibility that our MoG approach would benefit from at least allowing the shrinkage parameter to vary between the classes.

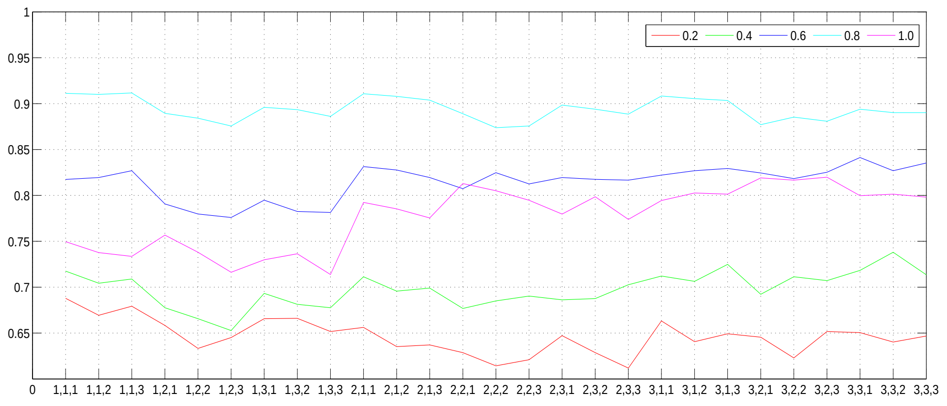
As discussed previously QDA is analogous to training a single Gaussian on each class. In our experiments we allow each class to be modeled by only a single Gaussian but do not allow for $\lambda = 0$ which would be equivalent to QDA. So, our experiments also represent a variant of QDA where covariance regularization is utilized – something that would be necessary to use QDA with this data when not projected into low dimensions anyway given that the covariance matrices would be singular. An interesting comparison to our work would be the state-of-the-art LDA and QDA algorithms which use other regularization methods. Since these are the predominant classification algorithms in the computer vision literature this comparison should be made.

Future work could definitely include fixing the shrinkage parameter value limitations of our experiments. Furthermore, we would like to experiment with the other covariance regularization methods mentioned in Section 2.4.

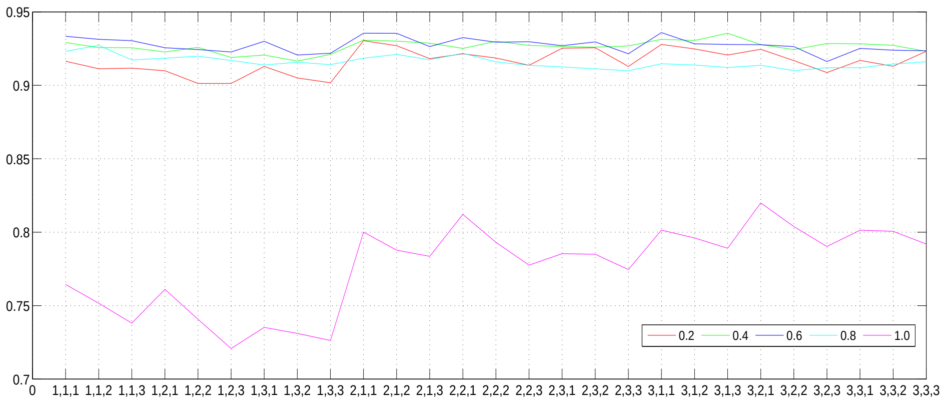
Our experiments primarily used data sets collected from the Internet and have not been used outside this Thesis with the exception of the cat/dog class. As a basis of comparison, it would be interesting to apply the MoG with shrinkage to well known handwritten digit and face recognition data sets. In addition, there is no reason to believe that our results are not applicable to other domains like bioinformatics or EEG classification. Comparison with the state of the art could prove helpful to those fields.



(a) 1024 dimensions



(b) 300 dimensions.



(c) 50 dimensions.

Figure 5.2: Plots illustrating the performance of FKM with all experimental values of λ and number of clusters on three levels of dimensionality on data set one. Each graph corresponds a different level of dimensionality. The x-axis corresponds to different numbers of clusters for each class (C_1, C_2, C_3) which are cat/dog, Christmas tree, and sunset respectively.

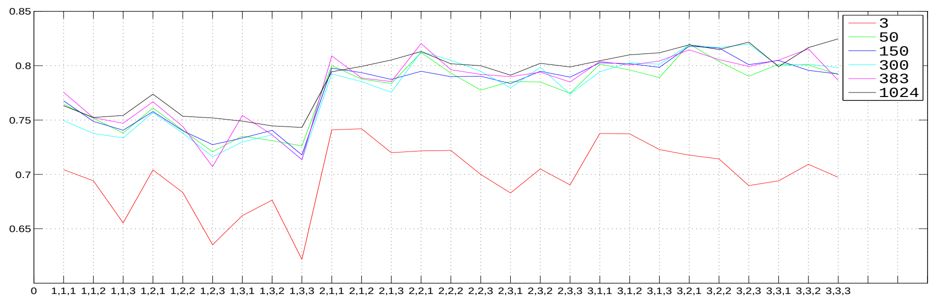
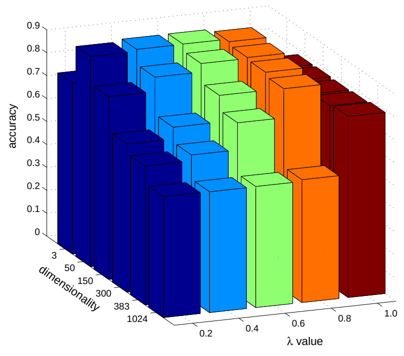
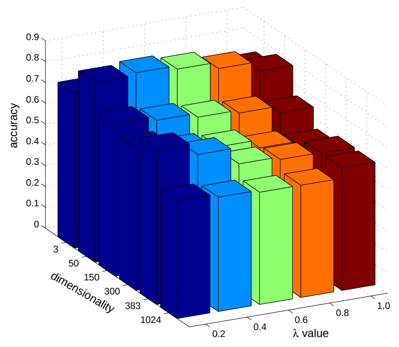


Figure 5.3: Plot of FKM with $\lambda = 1$ on data set one. Each line represents a different level of dimensionality. The x-axis corresponds to different numbers of clusters for each class (C_1, C_2, C_3) which are cat/dog, Christmas tree, and sunset respectively.



(a) FKM.



(b) DIAG.

Figure 5.4: Bar charts showing average classification accuracy for both shrinkage versions. The x-axis is the λ values, the y-axis is the data dimensionality, and the z-axis is classification accuracy.

REFERENCES

[1] Google image search, October 2008. `http://images.google.com/`.

[2] Picsearch – image search for pictures and images, October 2008. `http://www.picsearch.com/`.

[3] P.N. Belhumeur, J.P. Hespanha, and D.J. Kriegman. Eigenfaces vs. fisherfaces: recognition using class specific linear projection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):711–720, Jul 1997.

[4] J. Bilmes. A gentle tutorial on the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical report, ICSI, 1997.

[5] Christopher M. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2007.

[6] Guangzhi Cao and Charles A. Bouman. Covariance estimation for high dimensional data vectors using the sparse matrix transform. Technical report, Purdue University, 2008.

[7] Li-Fen Chen, Liao Hong-Yuan Mark, Ming-Tat Ko, Ja-Chen Lin, and Yu Gwo-Jong. A new lda-based face recognition system which can solve the small sample size problem. *Pattern Recognition*, 33(10):1713–1726, 2000.

[8] Sanjoy Dasgupta and Leonard Schulman. A probabilistic analysis of EM for mixtures of separated, spherical gaussians. *Journal of Machine Learning Research*, 8:203–226, 2007.

[9] A.P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Society*, B(39):1–38, 1977.

[10] B.A. Draper, K. Baek, and J. Boody. Implementing the expert object recognition pathway. In *International Conference on Vision Systems*, volume 16, pages 27–32, 2003.

[11] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* Wiley, New York, 1973.

[12] John W. Eaton. Octave documentation, October 2008. `http://www.gnu.org/software/octave/doc/interpreter/`.

[13] B.J. Frey and N. Jojic. Estimating mixture models of images and inferring spatial transformations using the EM algorithm. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 416–422, June 1999.

[14] B.J. Frey and N. Jojic. Transformation-invariant clustering using the EM algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(1):1–17, 2003.

[15] Z. Ghahramani and M.J. Beal. Graphical models and variational methods. In *Advanced Mean Field Methods*. MIT Press, 2000.

[16] Z. Ghahramani and M.J. Beal. Variational inference for Bayesian mixtures of factor analysers. In K. Muller S. A. Solla, T.K. Leen, editor, *Advances in Neural Information Processing Systems*, volume 12, pages 449–455. MIT Press, 2000.

[17] Zoubin Ghahramani and Geoffrey E. Hinton. The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, University of Toronto, 21 1996.

[18] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison-Wesley, 1993.

[19] Riad Hammoud and Roger Mohr. Mixture densities for video objects recognition. *International Conference on Pattern Recognition*, 02:2071, 2000.

[20] Harry H. Harman. *Modern Factor Analysis*. University Chicago Press, 1976.

[21] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, New York, 2001.

[22] G.E. Hinton, P. Dayan, and M. Revow. Modelling the manifolds of images of handwritten digits. *IEEE transactions on Neural Networks*, 8(1):65–74, 1997.

[23] R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge, Cambridge, UK, 1985.

[24] The Mathworks Inc. Matlab technical documentation, October 2008. `http://www.mathworks.com/access/helpdesk/help/techdoc/`.

[25] I.T. Jolliffe. *Principal Component Analysis*. Springer, New York, 2002.

[26] M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

[27] M. Kirby. *Geometric Data Analysis*. Wiley, New York, 2001.

[28] Olivier Ledoit and Michael Wolf. Improved estimation of the covariance matrix of stock returns with an application to portfolio selection. *Journal of Empirical Finance*, 10:603–621, 2003.

[29] James W. Miskin. *Ensemble Learning for Independent Componenet Analysis*. PhD thesis, University of Cambridge, 2000.

[30] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, Boston, 1997.

[31] Baback Moghaddam and Alex Pentland. Probabilistic visual learning for object representation. *PAMI*, 19(7):696–710, 1997.

[32] Ian T. Nabney. *NETLAB : algorithms for pattern recognitions.* Springer, London, 2002.

[33] Pekka Paalanen, Joni-Kristian Kamarainen, Jarmo Ilonen, and Heikki Kälviäinen. Feature representation and discrimination based on Gaussian mixture model probability densities-practices and algorithms. *Pattern Recognition*, 39(7):1346–1358, 2006.

[34] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, New Jersey, second edition, 2003.

[35] Juliane Schäfer and Korbinian Strimmer. A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. *Statistical Applications in Genetics and Molecular Biology*, 4(1), 2005.

[36] Michael Steinbach, Levent Ertöz, and Vipin Kumar. The challenges of clustering high dimensional data. In L.T. Wille, editor, *New Vistas in Statistical Physics Applications in Econophysics, Bioinformatics, and Pattern Recognition.* Springer-Verlag, 2003.

[37] M.E. Tipping and C.M. Bishop. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11(2):443–482, 1999.

[38] Jieping Ye and Tie Wang. Regularized discriminant analysis for high dimensional, low sample size data. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 454–463, New York, NY, USA, 2006. ACM.

[39] Hua Yu and Jie Yang. A direct LDA algorithm for high-dimensional data — with application to face recognition. *Pattern Recognition*, 34(10):2067–2070, 2001.

[40] X. Zhou and X. Wang. Optimisation of Gaussian mixture model for satellite image classification. *Vision, Image and Signal Processing*, 153(3):349–356, June 2006.

[41] Xiao-Sheng Zhuang and Dao-Qing Dai. Improved discriminate analysis for high-dimensional data and its application to face recognition. *Pattern Recognition*, 40:1570–1578, 2007.

[42] Hui Zou, Trevor Hastie, and Robert Tibshirani. Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15:262–286, 2004.

Appendix A

Detailed experimental results

	FKM					DIAG					SMT			
run	C_1	C_2	C_3	λ	Acc	C_1	C_2	C_3	λ	Acc	C_1	C_2	C_3	Acc
1	2	1	2	0.8	0.89474	3	3	2	0.4	0.84211	1	1	1	0.56579
2	2	1	1	0.8	0.94737	3	2	2	0.4	0.72368	1	1	1	0.59211
3	2	1	3	0.8	0.88158	3	3	3	0.6	0.64474	3	1	1	0.55263
4	1	1	2	0.8	0.89474	3	2	1	0.2	0.81579	1	1	1	0.56579
5	2	2	3	0.8	0.85526	1	2	2	0.2	0.73684	2	2	1	0.52632
6	2	1	2	0.8	0.86842	3	1	1	0.2	0.73684	2	1	1	0.52632
7	3	1	3	0.8	0.92105	2	3	3	0.2	0.73684	1	1	1	0.56579
8	2	1	3	0.8	0.88158	1	2	1	0.2	0.69737	3	3	2	0.56579
9	3	1	2	0.8	0.96053	2	1	3	0.2	0.71053	3	1	1	0.53947
10	2	1	2	0.8	0.89474	3	1	2	0.2	0.75	2	1	3	0.57895
11	3	3	2	0.8	0.86842	3	2	2	0.2	0.80263	2	3	2	0.61842
12	2	1	3	0.8	0.92105	2	3	3	0.2	0.80263	1	1	1	0.60526
13	2	1	2	0.8	0.92105	1	1	3	0.2	0.73684	1	1	1	0.59211
14	3	1	1	0.8	0.88158	3	2	1	0.4	0.69737	2	2	2	0.67105
15	1	1	3	0.8	0.88158	1	3	1	0.2	0.76316	2	2	2	0.53947
16	1	1	2	0.8	0.90789	1	1	1	0.2	0.75	3	3	2	0.61842
17	3	1	2	0.8	0.92105	2	3	3	0.2	0.71053	2	2	1	0.53947
18	1	1	3	0.8	0.94737	1	1	2	0.2	0.67105	3	2	1	0.57895
19	1	1	1	0.8	0.89474	3	1	1	0.2	0.78947	2	2	1	0.59211
20	3	1	2	0.8	0.86842	2	2	2	0.2	0.77632	3	2	2	0.57895
21	2	1	2	0.8	0.89474	2	1	2	0.2	0.67105	1	1	1	0.56579
22	1	1	1	0.8	0.92105	2	2	2	0.4	0.76316	2	1	1	0.60526
23	2	1	2	0.8	0.89474	2	3	2	0.2	0.75	1	1	1	0.60526
24	2	1	3	0.8	0.90789	2	1	2	0.4	0.68421	3	1	3	0.56579
25	1	1	3	0.8	0.94737	2	3	2	0.4	0.68421	2	1	1	0.52632
26	2	1	1	0.8	0.86842	3	2	1	0.2	0.71053	3	2	2	0.56579
27	1	1	2	0.8	0.93421	1	1	2	0.2	0.81579	1	1	1	0.60526
28	3	1	2	0.8	0.97368	3	3	3	0.2	0.78947	2	1	1	0.59211
29	1	1	1	0.8	0.88158	2	3	3	0.2	0.75	3	1	1	0.55263
30	2	1	1	0.8	0.93421	2	1	2	0.2	0.75	1	1	1	0.52632
31	1	1	3	0.8	0.89474	2	1	2	0.2	0.80263	3	1	1	0.53947
32	1	1	3	0.8	0.86842	3	2	2	0.2	0.76316	1	1	1	0.56579
mean					0.90419					0.74465				0.57278
var					0.0009297					0.0023441				0.0010944

Table A.1: Results of FKM and DIAG shrinkage methods and SMT on 32 test sets pre-processed using PCA to 383 dimensions.

	FKM				DIAG			
run	C_1	C_2	λ	Acc	C_1	C_2	λ	Acc
1	3	3	0.8	0.97576	2	3	0.2	0.75758
2	1	1	0.8	0.93333	1	3	0.2	0.67879
3	1	3	0.8	0.96364	2	3	0.2	0.78182
4	3	2	0.8	0.95758	1	3	0.2	0.75152
5	3	3	0.8	0.95152	2	3	0.2	0.64848
6	3	2	0.8	0.90909	3	2	0.2	0.70909
7	3	3	0.8	0.93939	3	3	0.2	0.64848
8	2	3	0.8	0.91515	1	3	0.2	0.67879
9	3	3	1	0.95152	1	3	0.2	0.76364
10	2	2	0.8	0.92727	1	2	0.2	0.72121
11	2	3	0.8	0.93333	2	3	0.2	0.73939
12	3	2	0.8	0.93939	1	2	0.2	0.7697
13	2	3	0.8	0.94545	3	2	0.2	0.73939
14	3	1	1	0.92121	3	3	0.2	0.79394
15	3	3	0.8	0.94545	1	2	0.2	0.75152
16	2	2	0.8	0.91515	2	3	0.2	0.79394
17	3	3	1	0.94545	1	1	0.2	0.76364
18	2	3	1	0.86061	2	1	0.2	0.62424
19	2	3	0.8	0.95758	3	3	0.2	0.72121
20	3	2	1	0.93939	1	2	0.2	0.72727
21	1	1	0.8	0.93939	3	2	0.2	0.70909
22	1	1	0.8	0.95758	2	2	0.2	0.75152
23	3	3	1	0.93939	1	2	0.2	0.78788
24	3	3	1	0.92121	3	3	0.2	0.71515
25	1	3	0.8	0.93333	3	2	0.2	0.73333
26	2	2	0.8	0.91515	1	3	0.2	0.72727
27	3	3	0.8	0.94545	1	2	0.2	0.69091
28	2	3	0.8	0.93939	1	3	0.2	0.78788
29	3	3	1	0.95152	1	3	0.2	0.67273
30	1	2	0.8	0.94545	1	2	0.2	0.70909
31	2	1	0.8	0.93939	1	3	0.2	0.8
32	1	3	0.8	0.96364	3	3	0.2	0.7697
mean				0.93807				0.73182
var				0.0004725				0.00214

Table A.2: Results of FKM and DIAG shrinkage methods on data set two broken randomly into 32 test sets using PCA to 823 dimensions.

	FKM					DIAG				
run	C_1	C_2	C_3	λ	Acc	C_1	C_2	C_3	λ	Acc
1	3	2	2	1	0.89474	3	3	3	0.2	0.60526
2	2	2	1	1	0.69737	3	3	3	0.4	0.76316
3	3	2	1	1	0.76316	3	2	3	0.6	0.69737
4	3	2	2	1	0.90789	2	3	2	0.2	0.75
5	2	1	3	1	0.84211	3	2	3	0.2	0.56579
6	2	1	3	1	0.72368	2	2	2	1	0.73684
7	2	3	3	1	0.80263	2	2	3	0.6	0.75
8	2	2	3	1	0.78947	3	2	2	0.2	0.69737
9	3	1	2	1	0.84211	3	2	2	0.2	0.76316
10	2	2	3	1	0.86842	3	2	2	1	0.75
11	1	1	3	1	0.76316	2	2	3	0.4	0.80263
12	3	2	3	1	0.84211	3	3	3	0.4	0.71053
13	3	3	3	1	0.77632	3	3	2	0.2	0.69737
14	2	2	1	1	0.85526	3	3	2	0.2	0.75
15	3	1	2	1	0.77632	3	3	2	1	0.73684
16	3	2	1	1	0.84211	3	2	3	1	0.65789
17	3	2	1	1	0.81579	3	2	2	0.2	0.76316
18	3	2	2	1	0.82895	3	3	2	1	0.68421
19	3	3	3	1	0.82895	3	3	3	0.2	0.67105
20	3	3	2	1	0.86842	3	3	3	0.8	0.76316
21	3	2	2	1	0.75	3	2	3	0.4	0.71053
22	2	2	1	1	0.81579	3	2	2	0.2	0.73684
23	3	2	2	1	0.81579	3	2	3	1	0.64474
24	2	1	3	1	0.78947	3	2	2	1	0.76316
25	2	3	3	1	0.80263	3	3	2	0.4	0.76316
26	2	1	1	1	0.77632	2	2	3	1	0.71053
27	3	2	1	1	0.81579	3	3	3	0.2	0.71053
28	3	1	3	1	0.76316	2	2	3	0.4	0.76316
29	3	3	3	1	0.82895	3	2	3	1	0.64474
30	2	2	1	1	0.84211	3	3	2	0.2	0.69737
31	2	1	1	1	0.78947	2	2	3	0.2	0.72368
32	3	2	2	1	0.82895	3	2	2	1	0.77632
mean					0.81086					0.71752
var					0.0021753					0.0027587

Table A.3: Results of FKM and DIAG shrinkage methods on data set one reduced to size 32x32 via bilinear interpolation and broken randomly into 32 test sets.

Dimensionality=3															
	FKM					DIAG					SMT				
run	C_1	C_2	C_3	λ	Acc	C_1	C_2	C_3	λ	Acc	C_1	C_2	C_3	K	Acc
1	1	2	2	0.8	0.80263	1	3	2	0.2	0.82895	1	2	2	5	0.72368
2	2	3	1	0.2	0.69737	1	2	2	0.6	0.77632	1	1	3	5	0.63158
3	2	2	3	0.2	0.81579	2	3	2	0.2	0.78947	3	3	1	5	0.69737
4	1	3	3	0.6	0.68421	3	2	3	0.2	0.73684	2	3	2	5	0.78947
5	3	3	1	0.6	0.80263	2	1	2	1	0.77632	1	1	1	5	0.73684
6	1	3	3	0.6	0.78947	2	2	2	0.6	0.75	3	3	3	5	0.78947
7	3	1	3	0.6	0.80263	3	2	3	0.4	0.84211	1	1	2	5	0.73684
8	1	2	1	0.6	0.77632	2	2	3	0.4	0.75	2	2	1	5	0.77632
9	1	2	2	0.8	0.73684	2	2	2	0.6	0.77632	3	3	3	50	0.78947
10	2	2	3	0.4	0.64474	2	3	1	1	0.65789	1	3	3	5	0.81579
11	2	3	3	0.4	0.76316	1	3	3	0.4	0.71053	1	1	1	5	0.75
12	1	3	2	0.6	0.75	2	3	2	0.6	0.77632	1	3	1	5	0.75
13	3	1	2	0.8	0.71053	3	1	1	0.2	0.68421	1	3	3	5	0.84211
14	1	3	2	0.6	0.69737	1	3	3	0.8	0.67105	1	2	2	5	0.69737
15	2	3	3	0.8	0.77632	3	2	3	0.6	0.81579	1	2	3	5	0.73684
16	2	3	1	0.8	0.77632	1	3	1	0.4	0.76316	1	2	2	5	0.68421
17	2	3	1	0.2	0.85526	1	2	3	0.4	0.80263	1	1	3	5	0.67105
18	1	3	2	0.2	0.80263	1	2	3	0.2	0.80263	1	1	1	5	0.80263
19	3	3	1	0.8	0.72368	3	3	3	0.4	0.81579	1	1	2	5	0.77632
20	2	2	1	0.6	0.75	1	3	1	0.2	0.72368	1	1	2	5	0.77632
21	1	2	1	0.8	0.77632	2	1	2	0.8	0.71053	3	2	2	5	0.77632
22	3	1	2	0.4	0.69737	3	2	3	0.2	0.73684	1	1	3	5	0.77632
23	1	3	1	1	0.67105	1	3	2	0.4	0.72368	2	3	3	5	0.72368
24	2	1	2	0.8	0.78947	2	2	2	0.2	0.7763	1	1	3	5	0.78947
25	2	3	1	0.2	0.69737	1	3	3	1	0.63158	3	3	3	5	0.75
26	1	2	2	0.2	0.76316	1	2	2	0.4	0.7368	2	2	2	5	0.81579
27	1	2	2	0.4	0.72368	2	3	2	0.4	0.6842	1	1	3	5	0.76316
28	1	3	3	0.4	0.68421	3	3	3	0.2	0.7105	1	2	3	5	0.76316
29	3	1	3	0.8	0.76316	1	2	3	0.2	0.6710	1	1	3	5	0.77632
30	1	2	1	0.4	0.71053	2	2	2	0.2	0.6842	3	3	1	5	0.69737
31	3	1	1	0.2	0.72368	3	1	2	0.2	0.7368	3	3	3	5	0.77632
32	3	3	1	0.4	0.71053	2	3	3	1	0.68421	2	3	3	5	0.65789
mean					0.74589					0.74178					0.75123
var					0.0024622					0.0029384					0.0024055

Table A.4: Classification accuracies for shrinkage versions A and D and SMT with data set 1 projected to 3 dimensions.

Dimensionality=50															
	FKM					DIAG					SMT				
run	C_1	C_2	C_3	λ	Acc	C_1	C_2	C_3	λ	Acc	C_1	C_2	C_3	K	Acc
1	1	1	1	0.8	0.92105	3	1	1	0.2	0.90789	3	2	1	300	0.86842
2	1	2	2	0.8	0.96053	2	1	1	0.2	0.90789	3	1	1	300	0.86842
3	3	1	3	0.4	0.89474	1	1	2	0.4	0.84211	2	1	1	300	0.93421
4	3	1	3	0.4	0.96053	1	3	2	0.4	0.82895	3	1	1	300	0.96053
5	2	1	2	0.4	0.97368	1	2	3	0.4	0.93421	3	2	2	50	0.88158
6	2	1	2	0.2	0.92105	3	1	2	0.4	0.85526	2	2	1	50	0.84211
7	2	1	1	0.4	0.96053	2	1	1	0.2	0.90789	1	1	1	50	0.88158
8	3	2	3	0.4	0.93421	2	1	1	0.4	0.85526	2	1	1	300	0.84211
9	1	1	3	0.6	0.92105	2	1	1	0.2	0.96053	1	1	1	50	0.86842
10	1	1	3	0.8	0.90789	1	2	1	0.2	0.88158	3	1	1	50	0.84211
11	1	1	3	0.6	0.93421	3	1	2	0.4	0.92105	3	1	1	50	0.88158
12	1	1	2	0.6	0.92105	2	1	2	0.4	0.82895	1	1	1	50	0.89474
13	1	2	2	0.6	0.93421	1	1	1	0.2	0.90789	1	2	1	5	0.84211
14	2	3	2	0.2	0.92105	3	1	1	0.2	0.93421	3	2	3	5	0.75
15	2	1	1	0.4	0.92105	2	2	1	0.6	0.89474	2	2	3	5	0.85526
16	1	2	3	0.8	0.90789	2	1	3	0.4	0.90789	1	1	1	50	0.82895
17	1	2	2	0.8	0.92105	2	2	3	0.4	0.89474	1	2	2	50	0.89474
18	1	1	1	0.6	0.93421	2	2	1	0.6	0.88158	2	2	1	50	0.86842
19	1	3	1	0.6	0.94737	3	1	1	0.2	0.92105	1	1	1	150	0.82895
20	3	1	1	0.2	0.92105	3	3	1	0.6	0.85526	1	1	1	50	0.84211
21	1	2	1	0.6	0.94737	3	1	1	0.4	0.93421	3	1	2	50	0.90789
22	3	2	3	0.2	0.94737	3	1	1	0.2	0.93421	2	2	3	50	0.90789
23	1	1	1	0.6	0.90789	2	1	2	0.6	0.84211	3	1	1	50	0.88158
24	3	3	1	0.6	0.89474	1	2	1	0.4	0.85526	2	1	3	50	0.85526
25	1	2	1	0.8	0.94737	3	1	1	0.2	0.92105	1	1	3	50	0.88158
26	3	2	3	0.2	0.93421	2	2	1	0.2	0.84211	1	2	1	50	0.92105
27	2	2	3	0.4	0.93421	3	1	1	0.2	0.93421	1	1	2	50	0.86842
28	2	1	2	0.6	0.92105	3	1	2	0.4	0.86842	2	1	2	50	0.85526
29	1	3	2	0.8	0.97368	3	2	2	0.2	0.92105	2	2	2	50	0.85526
30	1	1	1	0.4	0.90789	3	1	2	0.2	0.90789	2	2	1	50	0.86842
31	3	2	2	0.2	0.89474	1	1	1	0.4	0.86842	1	3	3	5	0.78947
32	1	1	3	0.6	0.97368	3	1	1	0.2	0.93421	3	3	3	50	0.82895
mean					0.93133					0.8935					0.86554
var					0.00052201					0.0013667					0.0016055

Table A.5: Classification accuracies for shrinkage versions A and D and SMT with data set 1 projected to 50 dimensions.

Dimensionality=150															
	FKM					DIAG					SMT				
run	C_1	C_2	C_3	λ	Acc	C_1	C_2	C_3	λ	Acc	C_1	C_2	C_3	K	Acc
1	2	3	2	0.6	0.86842	3	2	2	0.4	0.82895	2	2	1	300	0.84211
2	2	3	2	0.6	0.92105	2	3	1	0.2	0.86842	2	1	1	150	0.72368
3	1	2	1	0.8	0.88158	2	3	1	0.6	0.78947	2	2	1	50	0.78947
4	1	2	1	0.6	0.89474	2	2	1	0.4	0.78947	2	2	1	150	0.75
5	1	3	2	0.8	0.90789	3	3	1	0.2	0.80263	3	2	1	50	0.73684
6	2	3	3	0.6	0.92105	3	2	2	0.2	0.90789	3	1	1	50	0.77632
7	1	3	2	0.8	0.92105	3	3	2	0.2	0.81579	3	2	2	50	0.75
8	1	2	1	0.6	0.92105	3	3	1	0.2	0.85526	3	3	1	50	0.72368
9	1	2	1	0.6	0.82895	3	3	1	0.2	0.76316	3	2	1	50	0.80263
10	2	2	3	0.6	0.92105	3	3	2	0.2	0.78947	3	2	1	150	0.75
11	1	3	1	0.6	0.94737	3	3	1	0.2	0.90789	2	1	1	150	0.76316
12	3	2	1	0.8	0.86842	3	3	2	0.6	0.78947	3	2	1	50	0.78947
13	1	3	2	0.4	0.82895	3	3	1	0.2	0.86842	3	2	1	50	0.89474
14	1	1	1	0.6	0.92105	2	3	2	0.8	0.78947	3	2	2	300	0.65789
15	3	2	2	0.6	0.89474	3	3	1	0.4	0.85526	3	3	1	50	0.84211
16	2	3	2	0.4	0.82895	2	3	1	0.2	0.85526	3	3	1	150	0.80263
17	1	2	1	0.8	0.96053	3	3	1	0.4	0.89474	3	2	1	50	0.75
18	1	2	1	0.6	0.77632	2	1	1	0.2	0.69737	3	1	1	300	0.69737
19	2	3	1	0.6	0.92105	3	2	1	0.2	0.73684	2	1	1	300	0.75
20	1	2	2	0.6	0.89474	3	3	1	0.2	0.85526	1	1	1	50	0.67105
21	3	2	1	0.8	0.90789	2	2	1	0.4	0.78947	3	1	1	50	0.72368
22	2	2	1	0.6	0.96053	3	3	1	0.2	0.89474	1	1	3	50	0.68421
23	1	3	2	0.6	0.89474	3	3	1	0.2	0.88158	3	2	1	50	0.81579
24	1	3	2	0.8	0.92105	2	3	2	0.6	0.76316	2	2	1	150	0.72368
25	1	3	1	0.4	0.85526	2	3	2	0.6	0.67105	3	2	2	50	0.68421
26	3	2	1	0.6	0.90789	2	3	3	0.8	0.81579	3	3	1	50	0.68421
27	2	2	1	0.6	0.96053	3	3	2	0.4	0.86842	3	1	1	150	0.80263
28	3	2	2	0.8	0.89474	3	2	1	0.2	0.86842	3	1	1	150	0.73684
29	1	2	2	0.6	0.89474	3	2	1	0.6	0.81579	3	2	1	50	0.81579
30	2	2	1	0.8	0.90789	3	2	1	0.2	0.86842	3	2	1	300	0.77632
31	1	3	1	0.6	0.94737	2	2	1	0.2	0.80263	3	2	1	150	0.78947
32	3	3	1	0.6	0.89474	2	2	1	0.2	0.78947	2	2	1	50	0.84211
mean					0.89926					0.82155					0.76069
var					0.0017493					0.0033837					0.0032106

Table A.6: Classification accuracies for shrinkage versions A and D and SMT with data set 1 projected to 150 dimensions.

Dimensionality=300															
	FKM					DIAG					SMT				
run	C_1	C_2	C_3	λ	Acc	C_1	C_2	C_3	λ	Acc	C_1	C_2	C_3	K	Acc
1	1	1	1	0.8	0.86842	3	3	1	0.2	0.80263	3	2	1	300	0.67105
2	1	2	1	0.8	0.93421	3	3	1	0.2	0.68421	3	1	1	300	0.59211
3	1	1	2	0.8	0.88158	3	3	2	0.2	0.78947	3	3	1	300	0.76316
4	1	1	2	0.8	0.94737	3	3	1	0.2	0.78947	3	2	1	300	0.77632
5	1	1	2	0.8	0.85526	2	3	1	0.2	0.75	1	1	1	150	0.57895
6	2	1	2	0.8	0.94737	2	1	1	0.2	0.71053	2	1	1	300	0.68421
7	3	3	1	0.8	0.88158	3	3	1	0.2	0.68421	2	2	1	300	0.73684
8	2	1	1	0.8	0.92105	3	3	3	0.8	0.69737	3	1	1	300	0.76316
9	3	1	3	0.8	0.88158	3	3	2	0.2	0.67105	2	2	1	300	0.77632
10	3	3	3	0.8	0.92105	3	2	1	0.2	0.77632	1	1	1	300	0.63158
11	1	1	1	0.8	0.89474	3	3	1	0.2	0.72368	2	2	1	300	0.75
12	2	2	1	0.8	0.90789	3	2	1	0.2	0.80263	1	1	1	300	0.65789
13	1	2	1	0.8	0.92105	2	3	1	0.2	0.77632	2	2	1	300	0.75
14	1	1	3	0.8	0.89474	1	3	1	0.2	0.72368	2	1	1	300	0.65789
15	3	1	2	0.8	0.89474	1	2	1	0.2	0.71053	2	1	1	300	0.67105
16	3	1	1	0.8	0.90789	3	3	1	0.2	0.88158	3	2	1	150	0.71053
17	1	3	3	0.8	0.92105	3	3	2	0.2	0.72368	3	3	1	300	0.72368
18	2	1	3	0.8	0.93421	3	3	1	0.2	0.72368	3	2	1	300	0.61842
19	1	3	3	0.8	0.92105	3	2	1	0.2	0.73684	3	2	2	50	0.64474
20	2	1	3	0.8	0.86842	2	3	1	0.2	0.73684	1	1	1	300	0.60526
21	2	3	2	0.8	0.90789	2	3	2	0.2	0.69737	3	3	1	300	0.67105
22	1	1	1	0.8	0.93421	3	2	1	0.2	0.75	2	3	1	300	0.72368
23	2	1	1	0.8	0.88158	2	3	1	0.2	0.78947	3	2	1	300	0.86842
24	3	1	1	0.8	0.92105	3	3	1	0.4	0.75	2	1	1	300	0.60526
25	2	1	3	0.8	0.88158	3	2	1	0.2	0.76316	3	1	1	300	0.64474
26	1	1	2	0.8	0.90789	3	2	1	0.2	0.72368	2	3	1	300	0.71053
27	1	1	2	0.8	0.94737	3	2	1	0.2	0.72368	3	1	1	300	0.63158
28	3	1	2	0.8	0.92105	3	2	2	0.6	0.63158	3	1	1	300	0.67105
29	1	3	3	0.8	0.93421	3	3	1	0.2	0.81579	3	3	2	150	0.69737
30	1	3	2	0.8	0.92105	3	2	1	0.2	0.80263	2	1	1	300	0.69737
31	1	1	1	0.8	0.93421	3	3	2	0.2	0.69737	2	2	1	300	0.68421
32	1	2	2	0.8	0.86842	3	1	2	0.4	0.67105	3	3	1	300	0.75
mean					0.90831					0.74095					0.6912
var					0.00067559					0.0027303					0.0041773

Table A.7: Classification accuracies for shrinkage versions A and D and SMT with data set 1 projected to 300 dimensions.

Dimensionality=3								
	FKM				DIAG			
1	1	2	0.2	0.8303	2	1	0.6	0.86061
2	2	1	0.2	0.81818	3	2	1	0.82424
3	3	1	0.2	0.89091	3	3	0.2	0.89697
4	2	3	0.2	0.8	1	3	0.2	0.81212
5	1	1	0.4	0.83636	2	2	0.2	0.83636
6	1	3	0.6	0.83636	2	2	0.4	0.84242
7	1	1	0.4	0.85455	2	1	0.6	0.84848
8	1	1	0.2	0.86061	3	1	0.2	0.87273
9	2	1	0.2	0.81212	1	1	0.2	0.84848
10	1	2	0.2	0.84242	3	1	0.8	0.86061
11	1	1	0.2	0.79394	1	1	0.2	0.80606
12	1	1	0.6	0.89697	2	2	0.2	0.86667
13	1	3	0.4	0.80606	1	2	0.4	0.80606
14	2	2	0.2	0.83636	2	2	0.2	0.82424
15	2	1	0.2	0.85455	3	1	0.2	0.84848
16	1	3	0.4	0.87273	1	3	0.2	0.86061
17	1	3	0.4	0.83636	3	2	0.8	0.84242
18	3	3	0.2	0.82424	1	1	0.2	0.82424
19	2	1	0.6	0.87879	1	1	0.2	0.89091
20	1	2	0.2	0.84242	1	1	0.2	0.84848
21	2	1	0.6	0.86667	1	2	0.8	0.85455
22	2	1	0.2	0.86061	3	2	0.8	0.86061
23	1	3	0.2	0.87273	2	1	0.6	0.86061
24	2	2	0.2	0.85455	1	1	1	0.86061
25	2	3	0.6	0.87273	3	2	1	0.87879
26	1	1	0.2	0.84242	2	2	0.4	0.84242
27	3	2	0.2	0.85455	3	2	0.8	0.86061
28	2	1	0.8	0.80606	2	1	0.6	0.86667
29	3	1	0.6	0.8303	1	1	0.4	0.86667
30	3	2	0.2	0.85455	1	1	1	0.85455
31	1	3	0.4	0.84242	1	1	0.2	0.83636
32	2	3	0.2	0.85455	1	3	0.6	0.87273
mean				0.84489				0.85114
var				0.00065845				0.00049039

Table A.8: Classification accuracies for FKM and DIAG with data set two projected to 3 dimensions.

Dimensionality=50								
	FKM				DIAG			
1	1	3	0.8	0.98182	1	3	0.6	0.98788
2	1	1	0.6	1	1	1	0.4	0.98182
3	1	1	0.6	0.98788	1	2	0.4	0.95758
4	1	1	0.6	0.98182	1	2	0.2	0.97576
5	1	1	0.6	0.97576	3	2	0.4	0.95152
6	1	1	0.4	0.99394	1	2	0.2	0.98788
7	1	1	0.2	0.98182	2	2	0.2	0.98788
8	1	1	0.2	0.99394	1	1	0.4	0.98182
9	1	1	0.4	0.98788	1	2	0.2	0.98182
10	1	1	0.4	0.97576	1	2	0.6	0.9697
11	1	1	0.2	0.98788	1	1	0.2	0.98182
12	1	1	0.4	0.99394	1	1	0.2	0.97576
13	1	1	0.2	0.98182	1	2	0.2	0.98182
14	2	1	0.2	0.98788	1	2	0.4	0.98788
15	1	1	0.4	0.98182	1	2	0.2	0.98182
16	1	1	0.2	1	1	2	0.2	1
17	1	1	0.2	0.99394	1	2	0.4	1
18	1	2	0.4	0.97576	1	2	0.2	0.98182
19	1	1	0.4	1	2	2	0.4	0.99394
20	3	3	0.2	0.99394	1	1	0.4	0.99394
21	1	2	0.2	0.99394	2	2	0.8	0.95758
22	1	2	0.2	0.99394	1	1	0.2	0.99394
23	1	2	0.6	0.98788	3	3	0.2	0.98788
24	1	3	0.2	0.97576	1	3	0.4	0.97576
25	1	1	0.6	0.99394	2	1	0.2	1
26	1	1	0.2	0.99394	2	2	0.2	0.99394
27	1	2	0.6	0.98182	1	2	0.2	0.98182
28	1	1	0.4	0.98788	1	2	0.4	0.9697
29	1	2	0.8	0.98788	1	1	0.2	0.97576
30	2	3	0.2	0.98788	1	2	0.2	0.99394
31	1	1	0.4	0.99394	3	3	0.2	0.98788
32	3	3	0.2	0.98182	1	3	0.2	0.99394
mean				0.98807				0.98295
var				0.000053282				0.0014796

Table A.9: Classification accuracies for FKM and DIAG with data set two projected to 50 dimensions.

Dimensionality=150								
	FKM				DIAG			
1	1	1	0.4	0.98788	2	1	0.2	0.98182
2	1	1	0.6	0.99394	1	3	0.2	0.95758
3	1	2	0.4	0.97576	3	3	0.2	0.93333
4	1	1	0.6	0.95152	1	3	0.2	0.95152
5	1	2	0.6	0.99394	2	3	0.2	0.96364
6	1	1	0.6	0.99394	1	3	0.2	0.9697
7	1	2	0.8	0.98788	3	2	0.2	0.96364
8	1	1	0.4	0.99394	2	1	0.2	0.94545
9	1	1	0.8	0.98182	3	3	0.2	0.87879
10	1	1	0.6	0.98788	2	3	0.2	0.95758
11	1	3	0.6	0.99394	1	1	0.2	0.9697
12	1	1	0.4	0.98182	3	3	0.2	0.94545
13	3	3	0.2	0.98788	2	2	0.2	0.9697
14	1	2	0.2	0.95152	2	1	0.2	0.96364
15	1	1	0.2	0.97576	2	3	0.2	0.95152
16	1	1	0.6	0.98788	3	2	0.2	0.98182
17	1	1	0.6	0.99394	1	3	0.2	0.94545
18	3	3	0.8	0.96364	3	3	0.2	0.95758
19	1	1	0.4	0.99394	2	1	0.2	0.9697
20	1	1	0.2	0.96364	3	3	0.2	0.98182
21	2	1	0.2	0.97576	1	2	0.2	0.9697
22	1	2	0.6	0.99394	2	2	0.2	0.98788
23	1	1	0.4	0.98182	2	1	0.2	0.96364
24	1	1	0.4	0.9697	1	2	0.2	0.93333
25	1	1	0.6	0.98788	2	2	0.2	0.90909
26	1	1	0.8	0.96364	1	1	0.2	0.93939
27	2	1	0.6	0.98182	1	2	0.2	0.97576
28	3	2	0.6	0.98788	2	1	0.2	0.97576
29	2	1	0.6	0.97576	3	1	0.2	0.92727
30	1	1	0.6	0.99394	3	3	0.2	0.94545
31	1	1	0.6	0.98788	1	2	0.2	0.97576
32	1	2	0.2	0.99394	2	3	0.2	0.97576
mean				0.98239				0.95682
var				0.00015725				0.0005326

Table A.10: Classification accuracies for FKM and DIAG with data set two projected to 150 dimensions.

Dimensionality=300								
	FKM				DIAG			
1	1	2	0.8	0.98182	2	3	0.2	0.89697
2	2	1	0.8	0.98182	1	2	0.2	0.88485
3	2	1	0.8	0.98788	1	1	0.2	0.86061
4	1	1	0.8	0.97576	3	2	0.2	0.82424
5	1	2	0.8	0.98182	1	1	0.2	0.8303
6	1	1	0.8	0.9697	1	2	0.2	0.87879
7	1	1	0.8	0.9697	1	1	0.2	0.83636
8	1	1	0.8	0.97576	3	1	0.2	0.81212
9	1	3	0.8	0.97576	2	1	0.2	0.87879
10	3	1	0.8	0.97576	1	1	0.2	0.86061
11	1	2	0.8	0.9697	1	2	0.2	0.86667
12	3	3	0.8	0.9697	1	1	0.2	0.87879
13	1	1	0.8	0.99394	1	2	0.2	0.87273
14	2	1	0.8	1	2	1	0.2	0.90909
15	3	2	0.8	1	2	2	0.2	0.86667
16	1	1	0.8	0.98788	1	1	0.2	0.84848
17	1	1	0.8	0.9697	3	2	0.2	0.84242
18	1	1	0.8	0.98182	3	2	0.2	0.84848
19	1	1	0.8	0.9697	1	2	0.2	0.8303
20	1	1	0.6	0.95152	2	3	0.2	0.88485
21	1	1	0.8	0.98788	1	2	0.2	0.87273
22	2	1	0.6	0.96364	1	1	0.2	0.89697
23	1	3	0.8	0.97576	3	3	0.2	0.87273
24	1	1	0.8	0.96364	2	1	0.2	0.86667
25	1	3	0.8	0.97576	1	1	0.2	0.84242
26	1	1	0.8	0.99394	2	1	0.2	0.87273
27	2	1	0.8	0.97576	1	1	0.2	0.82424
28	1	3	0.8	0.97576	2	3	0.2	0.83636
29	1	2	0.8	0.98182	1	1	0.2	0.84242
30	1	1	0.8	0.98182	1	1	0.2	0.83636
31	2	1	0.8	0.96364	2	1	0.2	0.91515
32	1	2	0.8	0.96364	3	3	0.2	0.88485
mean				0.97727				0.86174
var				0.00012323				0.00068589

Table A.11: Classification accuracies for FKM and DIAG with data set 1 projected to 300 dimensions.