Supervised Dimensionality Reduction for Different Learning Architectures

Nathan H. Parrish

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2012

Reading Committee:

Maya R. Gupta, Chair

Payman Arabshahi

Mari Ostendorf

Program Authorized to Offer Degree:
Department of Electrical Engineering

University of Washington

Abstract

Supervised Dimensionality Reduction for Different Learning Architectures

Nathan H. Parrish

Chair of the Supervisory Committee:
Associate Professor Maya R. Gupta
Electrical Engineering

As datasets become larger and larger, there is a need for algorithms that can efficiently extract the relevant information for a given task and represent it in a concise manner. Supervised dimensionality reduction is one approach to doing this, as it reduces the input space of the data while retaining the characteristics of the data that are useful for classification.

This dissertation motivates and analyzes a new supervised dimensionality reduction technique called local discriminative Gaussian (LDG) dimensionality reduction. Experiments show that LDG is fast and effective when compared to other state-of-the-art supervised linear dimensionality reduction methods. LDG is shown to also be effective in the small sample size problem, where few training examples are provided in relation to the input dimensionality of the data. LDG is then extended to the transfer learning problem, where the goal is to classify test examples drawn from a target domain distribution using training examples drawn from a source domain distribution that differs from the target domain.

Another contribution of this dissertation is an algorithm that reliably classifies incomplete test data. Incomplete test data classification is useful if one wishes to classify a test sample before all of the test data is gathered, for example, if one wishes to make an early decision on time-series data. Experiments show that the proposed algorithm can classify incomplete time-series data while maintaining accuracy that is comparable to that achieved using the complete test data. Furthermore, LDG dimensionality reduction is shown to greatly reduce the computational complexity of the incomplete test data classifier.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I could not have completed my Ph.D. without the help of numerous people who have provided me with support along the way. First and foremost, I would like to thank Theresa; her consistent encouragement, thoughtful advice, and unwavering trust in my abilities have buoyed me over the past five years.

My parents and brother have also motivated and inspired me. Their lifetime of support and encouragment as well as their willingness to be a sounding board for my frustrations and successes has been amazing. Theresa's parents, as well, have provided outstanding advice and support during my graduate career.

I am especially thankful for the mentorship and guidance that Maya Gupta has provided as my adviser. Anyone who works with Maya can tell that she thoroughly enjoys working with students and seeing them succeed. It has been an honor to be part of the Information Design Lab under her tutelage.

I would also like to thank the excellent faculty and staff as well as my friends and colleagues in the Electrical Engineering Department. I would particularly like to thank my committee members for their guidance and mentorship, and my friends in the Information Design Lab for their willingness to review my work, critique my presentations, and share their wisdom.

Finally, I would like to thank the National Defense Science and Engineering Graduate Fellowship for generously funding three years of my graduate school career.

Chapter 1

INTRODUCTION

Algorithms that can extract and represent data in a concise manner are a critical need in machine learning. This is evident in classification, where the goal is to map an input vector of classification attributes or features to one of several different classes. Recent advances in sensing and storage capabilities have resulted in a wealth of high-dimensional feature data. Indeed, the buzzword *big data* has become common vernacular in the field of machine learning as well as within popular society [35].

This high-dimensional data can cause a variety of challenges for classification algorithms. First, it is often the case that some features in high-dimensional data contain a low signal to noise ratio. Such noisy variables mask the useful structure of the data and make the training of an accurate classifier more difficult [17]. High-dimensional data also increases the classifier complexity and can lead to longer training and testing times. Increased data dimensionality also requires increased memory and storage capabilities.

Dimensionality reduction is one technique to address the above challenges. Dimensionality reduction algorithms reduce the input space of the feature data while retaining its useful attributes. The first contribution of this thesis is local discriminative Gaussian (LDG) dimensionality reduction, a supervised dimensionality reduction algorithm for classification. I show that this algorithm is comparable to other state-of-the-art algorithms for supervised dimensionality reduction in terms of both computational complexity and in the classification accuracy that a simple classifier can achieve using the dimensions that it finds.

The second contribution of this thesis is dimensionality reduction for transfer learning. Transfer learning addresses the problem of how to apply training data gathered under one set of conditions to test data that gathered under a different set of conditions but is still related to the training data. For example, I consider the use of labeled images from an Amazon database in classifying images taken with a personal webcam (see Figure 4.4 in

Section 4.5 for a clear example of this problem). I show that LDG dimensionality reduction can be used to find a concise data representation for transfer learning.

The final contribution of this thesis is an algorithm for the reliable classification of incomplete data. In classification algorithms, there is sometimes a benefit to being able to classify a test example before all of the test data is available. For example, in time-series classification, it may be beneficial to classify time-sensitive data before the entire time-series is available. In Chapter 5 this problem is approached with the intent of guaranteeing that a decision made with incomplete data is as good as that which would be made with complete data. Furthermore, I show that LDG dimensionality reduction can greatly reduce the computational complexity of incomplete data classification.

Organization of This Dissertation

This dissertation is organized as follows. Chapter 2 reviews background information and related work in classification and supervised dimensionality reduction. Chapter 3 motivates and analyzes LDG dimensionality reduction and provides experimental results comparing LDG to other state-of-the-art algorithms. Chapter 4 reviews the problem of transfer learning, and shows how the LDG framework can be adapted to this setting. Chapter 5 proposes an algorithm for classifying incomplete test data. Finally, Chapter 6 gives some extensions and future directions for the research in this dissertation.

Some of the work in this thesis has appeared in previous publications. A condensed version of the work in Chapters 3 and 4 was published jointly with Maya Gupta in a 2012 ICML paper [45], and some of the figures and text are copied directly from that paper. The work in Chapter 5 was conducted jointly with Hyrum Anderson, Maya Gupta, Kristi Tsukida, and Dun Yu Hsaio. A preliminary version of the work in Chapter 5 appeared in a 2012 ICASSP conference paper [2], and a more complete version of this work has been submitted for journal publication as well [44]. Much of the text and figures in Chapter 5 were taken from the submitted journal paper.

Chapter 2

BACKGROUND AND RELATED WORK

The work in this thesis makes contributions in three areas of machine learning: supervised dimensionality reduction, dimensionality reduction for transfer learning, and incomplete data classification. In this section I first provide background material on the classifiers that will be used in this thesis. I then provide a discussion of the related work on dimensionality reduction. The transfer learning problem and associated related work will be discussed in Chapter 4, and the incomplete classification problem and related work will be discussed in Chapter 5.

2.1 Classifiers

In this thesis, I use several different standard classifiers that are briefly described here. For more information and background on supervised classification see the texts [28, 19].

2.1.1 Generative Classifiers: QDA

Generative classifiers use the labeled training data to learn the joint distribution of feature vectors and class labels $p(x, g)$ by factorizing it as $p(x|g)p(g)$ [7]. At test time, the trained classifier classifies test sample x according to the maximum a-posteriori rule:

$$\hat{g}(x) = \arg\max_{g} \hat{p}(x|g)\hat{p}(g), \tag{2.1}$$

where $\hat{p}(x|g)$ and $\hat{p}(g)$ are the estimated class-conditional and class-prior distributions.

Quadratic discriminant analysis (QDA) models the class-conditional distributions as Gaussian:

$$\hat{p}(x|g) = \mathcal{N}(\hat{\mu}_g, \hat{\Sigma}_g),$$

where $\hat{\mu}_g$ and $\hat{\Sigma}_g$ are the mean and covariance for class g. These parameters, as well as the estimated class priors $\hat{p}(g)$ are learned using the class g training data. Typical choices for for estimation of the Gaussian parameters are maximum likelihood or regularized maximum likelihood estimation [22].

QDA produces quadratic boundaries between the classes by taking the log of the MAP classifier function. Define $f_g(x) = -(x-\mu_g)^T\hat{\Sigma}_g^{-1}(x-\mu_g) - \log(\mid \Sigma_g \mid) + 2\log(\hat{p}(g))$ as the class g discriminant function for QDA. Then the MAP classifier (2.1) for QDA is equivalent to the following maximum discriminant rule:

$$\hat{g}(x) = \arg\max_g f_g(x).$$

2.1.2 Local Learning Algorithms: K nearest-neighbors and Local QDA

K nearest-neighbors (k-nn) classifiers assign a test sample x to the class that wins a majority vote among the k closest training points to x in feature space. K-nn requires no classifier training, and is thus sometimes referred to as a *lazy learner* [24]. However, in practice the number of neighbors k is often chosen by cross-validation as this number can impact classifier performance. K-nn has been shown to have the attractive theoretical property of achieving classification accuracy of no worse than twice the optimal Bayes error rate as the number of training examples approaches infinity [28]. However, in practice the performance of k-nn often suffers on high-dimensional data due to the *curse of dimensionality* which states that as the feature dimensionality grows the near-neighbors get farther and farther apart.

Another local learner is local QDA [24]. Sometimes the QDA assumption that the data is distributed globally Gaussian according to class is too restrictive. Therefore, local QDA models the data as Gaussian locally to the test point. Given a test point x, local QDA uses only the k nearest class g training points to x to estimate the Gaussian parameters for the class g Gaussian. As is the case for k-nn, the number of neighbors k for local QDA is typically chosen via cross-validation on the training data.

2.1.3 Discriminative Classifiers: Linear Support Vector Machine

Discriminative classifiers differ from generative classifiers in that they are not concerned with the probability model for the training data $p(x, g)$, but instead try to find the classifier function from some set of possible functions that minimizes the empirical error rate on the training data. In this manner, they attempt to directly minimize an estimate of the expected test error rate.

The linear support vector machine (SVM) is a discriminative classifer that is motivated by a two-class classification problem, where the set of class labels is $g_i \in \{-1, 1\}$. The SVM classifies a test vector x according to $\hat{g}(x) = \text{sign}(f(x))$, where $f(x)$ is a linear discriminant function of the form $f(x) = \beta^T x + b$. If the training data are linearly separable according to class, then the SVM finds the hyperplane in feature space that separates the data with maximum margin: $\frac{2}{\|\beta\|}$.

The SVM discriminant function is trained via the convex optimization problem [28, 56]:

$$\begin{aligned} \min_{\beta,b,\xi} \quad & \frac{1}{2}\beta^T\beta + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & \xi_i \geq 0,\ i = 1, \ldots, n \\ & g_i(x_i^T\beta + b) \geq 1 - \xi_i,\ i = 1, \ldots, n, \end{aligned} \tag{2.2}$$

where the ξ_i are slack variables that allow some training points to fall on the wrong side of the hyperplane and C is a regularization parameter that controls the model complexity.

The SVM can be extended to multi-class classification scenarios by using the one-vs-all method [30]. In the one-vs-all method a different SVM discriminant function $f_g(x)$ is created for each class g. The class g discriminant function is trained by temporarily assigning all class g training examples a label of 1 and all other classes a label of -1. Then, $f_g(x)$ is trained using the two-class SVM objective function (2.2) on this data. At test time, a test sample x is classified according to the maximum discriminant rule:

$$\hat{g}(x) = \arg\max_{g}\ f_g(x).$$

One-vs-all SVM classification has been shown to produce class labels that are as accurate as other methods, such as all-vs-all, when the discriminant functions are well-tuned [50].

2.1.4 The Kernel Trick and Nonlinear SVMs

Suppose that instead of finding a linear decision boundary that separates the training x_i according to class, we instead wish to find a nonlinear boundary. This can be accomplished using the kernel trick, which allows inner products in feature space, $x_i^T x_j$, to be replaced with a kernel function $K(x_i, x_j)$ that computes inner products in some higher dimensional (possibly infinitely dimensional) reproducing kernel Hilbert space (RKHS). In order for the kernel function to be valid, it must satisfy Mercer's conditions, and thus must be positive semi-definite. Popular kernel functions include the Gaussian radial basis function with bandwidth parameter γ

$$K(x_i, x_j) = e^{\frac{-1}{\gamma}\|x_i - x_j\|^2},$$

and the polynomial kernel with degree parameter s

$$K(x_i, x_j) = (x_i^T x_j)^s.$$

The kernel trick can be used to turn linear objective functions into non-linear objective functions whenever the feature vectors x_i are expressed only as inner products.

Define the kernel matrix, K, as a matrix with entry (i, j) equal to $K(x_i, x_j)$. Using the kernel matrix, nonlinear SVMs can be found using the kernel trick via the optimization problem

$$\begin{aligned} \min_{\alpha,b,\xi} \quad & \frac{1}{2}\alpha^T K \alpha + C\sum_{i=1}^{n} \xi_i \\ \text{subject to} \quad & \xi_i \geq 0,\ i = 1, \ldots, n \\ & g_i(\sum_{j=1}^{n} \alpha_j K(x_i, x_j) + b) \geq 1 - \xi_i,\ i = 1, \ldots, n, \end{aligned} \tag{2.3}$$

where the i^{th} entry of $\alpha \in R^n$ is a weight associated with training point x_i. The resulting SVM discriminant function is $f(x) = \sum_{i=1}^{n} \alpha_i K(x, x_i) + b$.

2.2 Background and Related Work in Dimensionality Reduction

The basic goal of linear dimensionality reduction is to find a matrix $B \in \mathbb{R}^{d\times\ell},\ \ell < d$ that maps the training feature vectors $\{x_i\}_{i=1}^n \in \mathbb{R}^d$ to a lower dimensional space: $\{B^T x_i\}_{i=1}^n \in \mathbb{R}^\ell$. Since there are infinitely many choices for B, the algorithm must have some criteria for choosing between them. In supervised dimensionality reduction for classification, the B is commonly selected with the goal of finding a subspace in which the data can be distinguished according to class.

2.2.1 Principal Components Analysis

One of the most well-known dimensionality reduction methods is principal components analysis (PCA) [69]. PCA finds an orthonormal matrix B that maps the data into a variance-maximizing subspace. Let S be the covariance matrix of the training data defined as

$$S = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_x)(x_i - \mu_x)^T,$$

where $\mu_x = \sum_{i=1}^n x_i$ is the mean of the training data. PCA finds the dimensionality reduction matrix that solves the following optimization problem

$$\arg\max_{B\in\mathbb{R}^{d\times\ell}} \mathrm{Tr}(B^T S B) \text{ subject to } B^T B = I. \tag{2.4}$$

The solution to (2.4) is to set the columns of B to the ℓ leading eigenvectors of S; for completeness, I state this result formally in Proposition 1 in Chapter 3 and prove it in Appendix A.1. PCA is an unsupervised algorithm, meaning that it does not take into account the class labels of the training data. Nevertheless, PCA is still often used as a dimensionality reduction algorithm prior to classification.

2.2.2 Fisher Discriminant Analysis

Supervised dimensionality reduction for classification dates to 1936 with Fisher discriminant analysis (FDA) [21], which is sometimes also known in the literature as linear discriminant analysis (LDA). Whereas PCA tries to find a matrix B that maximizes the variance of the

data, FDA instead considers the contributions to S from variance between the classes and that due to variance within the individual classes. Define S_b as the between-class covariance and S_w as the within-class covariance:

$$S_b = \frac{1}{n}\sum_{c=1}^{G} n_c(\mu_c - \mu_x)(\mu_c - \mu_x)^T \text{ and} \tag{2.5}$$

$$S_w = \frac{1}{n}\sum_{c=1}^{G}\sum_{i:g_i=c}(x_i - \mu_c)(x_i - \mu_c)^T, \tag{2.6}$$

where μ_c is the mean of the class c training data and n_c is the number of class c training points. FDA finds the matrix B that maximizes the ratio of the between-class variance to the within-class variance:

$$\arg\max_{B\in\mathbb{R}^{d\times\ell}} \text{Tr}\left(\frac{B^T S_b B}{B^T S_w B}\right). \tag{2.7}$$

The solution is to choose the top eigenvectors of the generalized eigendecomposition $S_b\lambda = \nu S_w\lambda$.

FDA has several drawbacks. First, FDA uses only a single mean per class. Therefore, FDA performs poorly on data with multi-modal classes that are not well represented by a single mean. Second, the between-class covariance matrix is at most rank $G-1$, so the the reduced space B can contain no more than $G-1$ dimensions.

2.2.3 Local Fisher Discriminant Analysis

Local Fisher discriminant analysis (LFDA) attempts to alleviate the drawbacks of FDA [64]. LFDA redefines the between-class and within-class covariance matrices as:

$$\bar{S}_b = \frac{1}{2}\sum_{i,j=1}^{n} \bar{A}_b(i,j)(x_i - x_j)(x_i - x_j)^T \text{ and} \tag{2.8}$$

$$\bar{S}_w = \frac{1}{2}\sum_{i,j=1}^{n} \bar{A}_w(i,j)(x_i - x_j)(x_i - x_j)^T, \tag{2.9}$$

where $\bar{A}_b(i,j)$ and $\bar{A}_w(i,j)$ are between-class and within-class affinity matrices. The LFDA formulation reduces to FDA for certain choices of the affinity matrices. However, the authors

propose that the affinity matrices be designed so that far apart same-class training points have less weight. This choice enables LFDA to find embeddings that separate multi-modal data. Also, LFDA results in a linear projection that can have more than $G-1$ eigenvalues. LFDA is solved using the same generalized eigndecomposition as FDA.

2.2.4 Neighbourhood Components Analysis

Neighbourhood components analysis (NCA) is a dimensionality reduction technique that attempts finds an embedding matrix B that makes same-class samples close and different-class samples far [26]. For a candidate B and training samples x_i and x_j, define a weight based on the distance between x_i and x_j after the mapping:

$$w_{i,j}(B) = \begin{cases} \frac{e^{-\|B^T x_i - B^T x_j\|^2}}{\sum_{k \neq i} e^{-\|B^T x_i - B^T x_k\|^2}} & \text{if i} \neq \text{j} \\ 0 & \text{if i} = \text{j} \end{cases} \tag{2.10}$$

Let $\mathrm{I}(\cdot)$ be a function that is equals one if the argument is true and zero otherwise. The NCA objective is to choose the B that maximizes the weight among same-class neighbors:

$$\arg\max_{B \in R^{d \times \ell}} \sum_{i,j=1}^{n} \mathrm{I}(g_i = g_j) w_{i,j}(B).$$

This objective function is non-convex; the authors propose a gradient-based optimization.

The dimensionality reduction found by NCA was shown to provide good classification accuracy; however, it suffers from two key drawbacks. One, the gradient descent can be slow for datasets with a large number of features or training examples. Second, the NCA optimization must be re-run for any desired number of final dimensions. This is in contrast to PCA, FDA, and LFDA where B can be found once for the largest number of final dimensions, and then the top submatrices of B are the optimal solution for smaller dimensions.

2.2.5 Distance Metric Learning Methods and Information Theoretic Metric Learning

There is a large body of work in distance metric learning and feature selection that is related to linear dimensionality reduction. Distance metric learning addresses the problem

of how to best determine the distance between feature vectors in $\mathbb{R}^d$. A positive definite Mahalanobis distance metric M defines the distance between x_i and x_j as

$$d_M(x_i, x_j) = \sqrt{(x_i - x_j)^T M (x_i - x_j)}$$

If the matrix M is positive semi-definite, then M defines a Mahalanobis *pseudo-metric*. Therefore, linear dimensionality reduction can be thought of as finding a low-rank pseudo-metric M, such that $M = BB^T$.

The approaches given in [16, 25, 68] propose convex optimization problems for finding M. In the context of dimensionality reduction, these methods suffer from the drawback that rank constraints are non-convex, and thus the M that they find is typically not low rank. However, dimensionality reduction can be achieved by rewriting the Mahalanobis metric as $M = L\Lambda L^T$ and using a feature selection method on the resulting $z_i = \Lambda^{\frac{1}{2}} L^T x_i$ as proposed in [16, 25]. Feature selection methods for classification choose the best subset of features out of those available. The final dimensionality reduction matrix, $B \in \mathbb{R}^{d \times \ell}$, can then be defined as the columns of matrix $L\Lambda^{\frac{1}{2}}$ that are associated with the top ℓ features chosen by the feature selection method.

Information theoretic metric learning (ITML) is a popular metric learning algorithm for classification [16, 31]. The goal of ITML is to find a Mahalanobis metric that brings similar points close together while forcing dissimilar points far apart. Let $d_M(x_i, x_j)^2 = (x_i - x_j)^T M (x_i - x_j)$ be the squared distance between training points i and j under distance metric M. The goal of ITML is to minimize a log-determinant regularizer subject to constraints on the distances between same-class and different-class points:

$$\begin{aligned} &\arg\min_{M \succeq 0} \operatorname{Tr}(M) - \log\det(M) \\ &\text{subject to } d_M(x_i, x_j)^2 \leq u, \text{ if } g_i = g_j \\ &\qquad\qquad d_M(x_i, x_j)^2 \geq v, \text{ if } g_i \neq\ g_j, \end{aligned} \tag{2.11}$$

where $\sqrt{u}$ is a predefined maximum distance between same-class points, and $\sqrt{v}$ is a predefined minimum distance between different-class points. In practice, slack variables are incorporated into the objective function in order to guaranatee that the problem is feasible.

The ITML objective function is a convex optimization problem. The authors propose to solve the optimization problem by repeatedly picking different pairs of training points and then taking a step in the direction that minimizes the objective function for that pair until convergence. After finding M in this manner, the authors propose to use an information gain feature selection method, such as the max-relevance min-redundency algorithm [46], to perform dimensionality reduction.

Chapter 3

LOCAL DISCRIMINATIVE GAUSSIAN DIMENSIONALITY REDUCTION

LDG dimensionality reduction is a supervised dimensionality reduction technique for classification. As described earlier, the goal of supervised dimensionality reduction as a preprocessing step for classification is to find a mapping that reduces the size of the data and separates the data according to class. One way to achieve separation by class is to find a mapping such that the performance of a simple classifier is maximized. LDG, therefore, is based on the idea of finding a mapping that minimizes the leave-one-out training error of a quadratic discriminant analysis (QDA) classifier. Because the goal of separation by class may be difficult to achieve globally, the LDG criterion instead operates locally to each training point by minimizing the error of local QDA.

Since the leave-one-out QDA error is a non-differentiable function, I approximate it by using the discriminative Gaussian criterion, a smooth approximation. The discriminative Gaussian objective function is non-convex with no analytical solution; however, I present a final approximation that has an analytical solution that requires only a maximal eigenvalue decomposition. Furthermore, I show that this approximation can be interpreted as the mapping that maximizes the regularized class-conditional likelihood of the training data.

I also show that LDG dimensionality reduction can find non-linear mappings as well. Kernelized LDG dimensionality reduction uses the kernel trick to find non-linear mappings in Euclidean space.

I perform experiments to compare LDG dimensionality reduction to other state-of-the-art methods, and show that LDG dimensionality reduction is very competitive in terms of both classification performance and computational complexity. I also describe and perform experiments for small sample size dimensionality reduction. The experiments show that LDG works well in this setting as well.

Some of the material in this chapter was conducted jointly with Maya Gupta and has been previously published [45].

3.1 Problem Formulation and LDG Solution

I take as given a set of labeled training data $\{(x_i, g_i)\}_{i=1}^n$, with $x_i \in \mathbb{R}^d$ being the i^{th} feature vector and $g_i \in \{1, 2, ..., G\}$ being the associated class label drawn from a set of G total classes.

The goal of my dimensionality reduction algorithm is to find a matrix $B \in \mathbb{R}^{d \times l}, l < d$ that maps the data to a reduced dimensional space $\{B^T x_i\}$ that separates the data according to class. In order to find such a mapping, I measure the separability by the performance of a generative classifier. Let $p(x_i|g)$ be the likelihood of x_i given class g, estimated from the other $n-1$ training sample pairs, and let p_g be the class prior. Then the leave-one-out cross-validation error of a maximum a-posteriori (MAP) classifier acting on the mapped features measures the separation achieved by B:

$$\sum_{i=1}^{n} \mathrm{I}\left(p(B^T x_i|g_i)p_{g_i} < \max_j p(B^T x_i|j)p_j\right), \tag{3.1}$$

where the indicator function $\mathrm{I}(\cdot)$ is one if its argument is true and zero otherwise.

The discontinuity of the indicator function in (3.1) makes it difficult to minimize. In order to arrive at a smooth, differentiable objective function that approximates (3.1), I substitute a log for the indicator and a sum for the max:

$$J(B) = \sum_{i=1}^{n} \log\left(\frac{\sum_{j=1}^{G} p(B^T x_i|j)p_j}{p(B^T x_i|g_i)p_{g_i}}\right). \tag{3.2}$$

In related work, $p(x_i|j)$ was assumed to be a Gaussian mixture model (GMM), and the objective was to learn a parameter vector, Θ, of GMM weights, means, and variances that minimized $J(\Theta)$ with $p(x_i|j, \Theta)$ replacing $p(B^T x_i|j)$ in (3.2) [36]. The learned parameters were shown to improve the GMM classification performance over the parameters learned by maximum likelihood estimation. In that work, (3.2) is motivated as maximizing the mutual information between the class labels and the feature vectors.

I assume $p(x_i|j)$ is Gaussian, $\mathcal{N}(x_i; \mu_{i,j}, \Sigma_{i,j})$. However, to reduce the model bias of assuming one Gaussian per class, I model $p(x_i|j)$ as *locally* Gaussian [24]. That is, I estimate the parameters of the Gaussian for point x_i and class j by finding the k nearest class j neighbors, in Euclidean distance, to training point x_i and using these points to estimate the Gaussian's maximum likelihood mean and covariance. To reduce estimation variance, and to make the eventual solution tractable, I model each covariance matrix as $\Sigma_{i,j} = I$, where I is the properly sized identity matrix. Therefore, $p(B^T x_i|j) = \mathcal{N}(B^T x_i; B^T \mu_{i,j}, B^T B)$.

3.1.1 Gradient Descent Solution

Objective (3.2) is non-convex with no analytical solution; however, a local minimum can be achieved by gradient-descent. To show this, I first expand the right hand side of (3.2) and substitute in the Gaussian distributions:

$$\sum_{i=1}^{n} \log \left(\sum_{j=1}^{G} p_j (2\pi)^{-\frac{\ell}{2}} (B^T B)^{-\frac{1}{2}} e^{-\frac{1}{2}(x_i - \mu_{i,j})^T B (B^T B)^{-1} B^T (x_i - \mu_{i,j})} \right) \tag{3.3}$$
$$- \log \left(p_{g_i} (2\pi)^{-\frac{\ell}{2}} (B^T B)^{-\frac{1}{2}} e^{-\frac{1}{2}(x_i - \mu_{i,g_i})^T B (B^T B)^{-1} B^T (x_i - \mu_{i,g_i})} \right).$$

To further simplify the above objective function, I add the constraint that $B^T B = I$ so that the covariance of the Gaussians is independent of the mapping. This constraint serves two other purposes as well. First, it makes the solution unique. Second, it makes the columns of B orthogonal, meaning that the information contained in the features after the mapping is non-redundant. Adding this constraint to (3.3) and getting rid of terms that do not effect the minimization results in the constrained optimization problem

$$B^* = \underset{B \in \mathbb{R}^{d \times \ell}}{\arg\min} \sum_{i=1}^{n} \log \left(\sum_{j=1}^{G} p_j e^{-\frac{1}{2}(x_i - \mu_{i,j})^T B B^T (x_i - \mu_{i,j})} \right) \tag{3.4}$$
$$+ \frac{1}{2} (x_i - \mu_{i,g_i})^T B B^T (x_i - \mu_{i,g_i})$$
$$\text{subject to } B^T B = I.$$

Define $\Delta_{i,j} = x_i - \mu_{i,j}$ and $c_{i,j} = p_j e^{\frac{-1}{2}\Delta_{i,j}^T BB^T \Delta_{i,j}}$, then the gradient of the objective function in (3.4) is

$$\sum_{i=1}^{n} \left(\Delta_{i,g_i} \Delta_{i,g_i}^T B - \frac{1}{\sum_{j=1}^{G} c_{i,j}} \left(\sum_{j=1}^{G} c_{i,j} \Delta_{i,j} \Delta_{i,j}^T B \right) \right). \tag{3.5}$$

A gradient descent approach can then be used to find a local minimum to (3.4) by computing the gradient of (3.4) at the current value of B using (3.5), taking a gradient step in this direction, and then projecting the resulting B back onto the constraint set. Projection of B onto the constraint set $B^T B$ can be performed using an orthonormalization procedure such as the Graham-Schmidt procedure [49] or QR decomposition [67]. Let $orth(B)$ be a function that orthonormalizes the columns of B, the gradient descent approach is summarized in Algorithm (1)

Input: $B \in \mathbb{R}^{d \times \ell}$, η
Output: B^*
while *!converged* **do**
  $\hat{B} \leftarrow B - \eta \sum_{i=1}^{n} \left(\Delta_{i,g_i} \Delta_{i,g_i}^T B - \frac{1}{\sum_{j=1}^{G} c_{i,j}} \left(\sum_{j=1}^{G} c_{i,j} \Delta_{i,j} \Delta_{i,j}^T B \right) \right)$
  $\hat{B} \leftarrow orth(\hat{B})$
  if $f(\hat{B}) \leq f(B)$ **then**
    $B \leftarrow \hat{B}$
  end
  if $f(\hat{B}) > f(B)$ **then**
    $\eta \leftarrow \frac{\eta}{2}$
  end
end

Algorithm 1: Gradient descent approach to finding the B that minimizes (3.4).

3.1.2 Maximal-eigenvalue Solution

In this section, I show that by making one further approximation to (3.2) I can arrive at an objective function whose solution requires only a maximal-eigenvalue decomposition. The resulting B^* can then be used as the final LDG dimensionality reduction matrix or can be used as an initializer for the gradient descent solution described in previous section.

First, rewrite (3.2) as

$$\sum_{i=1}^{n}\left(\log\left(\sum_{j=1}^{G}p(B^Tx_i|j)p_j\right)-\log\left(p(B^Tx_i|g_i)p_{g_i}\right)\right), \tag{3.6}$$

and bound (3.2) from below with Jensen's inequality by replacing the first log term in (3.6) with $\sum_{j=1}^{G}p_j\log(p(B^Tx_i|j))$:

$$\sum_{i=1}^{n}\left(\left(\sum_{j=1}^{G}p_j\log p(B^Tx_i|j)\right)-\log\left(p(B^Tx_i|g_i)\right)\right). \tag{3.7}$$

Figure 3.1 plots the original MAP error function (3.1), the log approximation (3.2), and the lower bound to the log approximation (3.7) for a two-class problem with a single x_i. The figure shows that (3.2) smoothly approximates the MAP error function by increasingly punishing solution matrices B that make the posterior of the incorrect class higher than that of the correct class. The figure also clearly shows that (3.7) lower bounds (3.2). The main difference in these two functions occurs along the left edge of the plots where the probability of the incorrect class approaches zero. It is straightforward to verify that (3.7) evaluates to $-\infty$ for any B that results in the posterior of the incorrect class evaluating to zero while the posterior of the correct class is greater than zero. This is in contrast to objective function (3.2) that evaluates to zero for the same value of B. Therefore, it may be possible for the lower bound approximation (3.7) to over-train to a few training examples by pushing them close to this edge. Later, I will show how adding a regularization parameter to (3.7) can alleviate this problem.

Returning to the problem of finding the B that minimizes (3.7), I again impose the constraint $B^TB=I$. Substituting the Gaussian distributions into (3.7) and assuming that $B^TB=I$ (the eventual solution will guarantee that this assumption holds) results in

$$\sum_{i=1}^{n}\left(\sum_{j=1}^{G}p_j\log\left((2\pi)^{-\frac{\ell}{2}}e^{-\frac{1}{2}\Delta_{i,j}^TBB^T\Delta_{i,j}}\right)\right)-\log\left(p_{g_i}(2\pi)^{-\frac{\ell}{2}}e^{-\frac{1}{2}\Delta_{i,g_i}^TBB^T\Delta_{i,g_i}}\right),$$

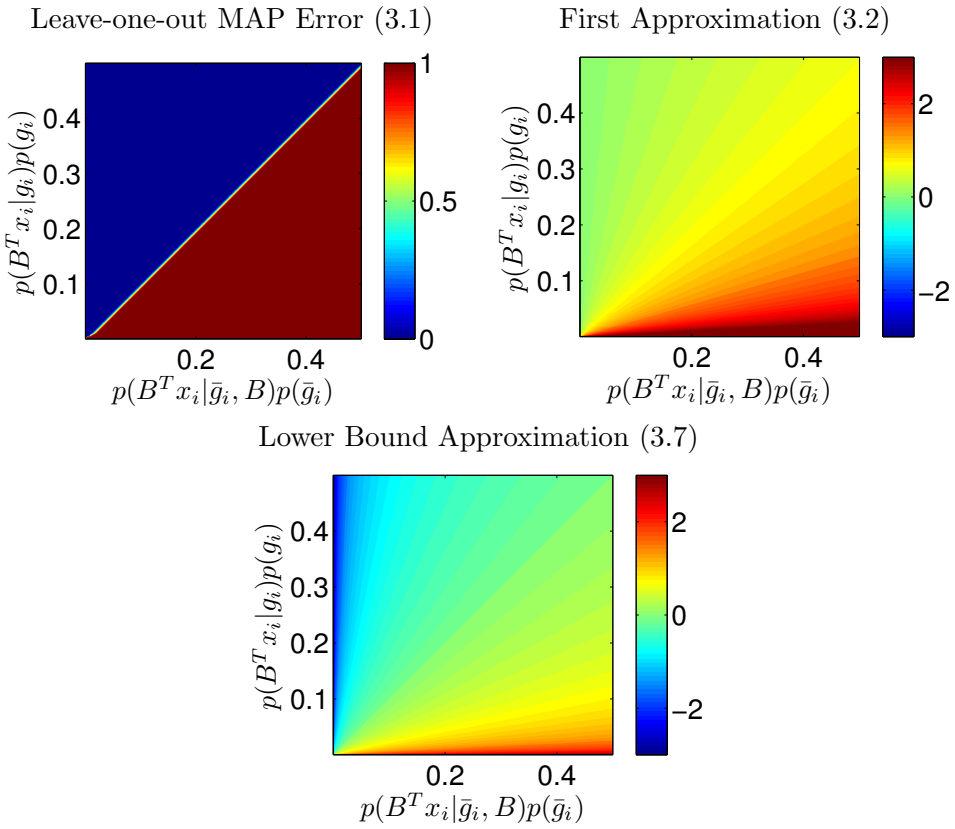

Figure 3.1: The MAP classifier error function (3.1), the approximation given in equation (3.2), and the lower bound approximation using Jensen's inequality (3.7) for a two-class problem plotted for a single x_i as a function of the posteriors resulting from B. The vertical axis for each plot is the posterior probability of the correct class after dimensionality reduction by B, and the horizontal axis is the posterior of the incorrect class.

Taking the log of the Gaussians and dropping the terms that are independent of B, we arrive at the LDG optimization problem

$$B^* = \underset{B\in\mathbb{R}^{d\times l}}{\arg\min} \sum_{i=1}^{n} \left(\frac{1-p_{g_i}}{2} \Delta_{i,g_i}^T B B^T \Delta_{i,g_i} - \sum_{j=1, j\neq g_i}^{G} \left(\frac{p_j}{2} \Delta_{i,j}^T B B^T \Delta_{i,j} \right) \right) \tag{3.8}$$

subject to $B^T B = I$.

Despite the approximations, (3.8) retains an intuitive meaning. The B that minimizes the first term in (3.8) is the maximum likelihood solution for the *correct-class* local Gaus-

sians. The second term is composed of $G-1$ different terms, each of which, if minimized individually, will give the maximum likelihood solution for an *incorrect-class* Gaussian, ie. a Gaussian distribution trained by the local neighbors of x_i coming from a different class. Therefore, (3.8) can be viewed as a regularized maximum likelihood estimate, where the regularization term attempts to minimize the likelihood of incorrect classes.

The B that optimizes (3.8) can be found with one eigendecomposition. Define

$$V = \sum_{i=1}^{n} \Delta_{i,g_i} \Delta_{i,g_i}^T \tag{3.9}$$

$$A = \sum_{i=1}^{n} \sum_{j=1}^{G} p_j \Delta_{i,j} \Delta_{i,j}^T. \tag{3.10}$$

Then (3.8) can be written as

$$\begin{aligned} B^* = \underset{B \in \mathbb{R}^{d \times l}}{\arg\min} \frac{1}{2} \mathrm{Tr}\left(B^T (V - A) B\right) \\ \text{subject to } B^T B = I. \end{aligned} \tag{3.11}$$

Proposition 1: *The solution to (3.11) is to set B^*'s columns to be the l smallest eigenvectors of matrix $(V - A)$.*
The proof is in Appendix A.1.

Additionally, I add a cross-validated regularization parameter, γ, to (3.11), which I have found in practice can produce a mapping that better separates the data:

$$\begin{aligned} B_\gamma^* = \underset{B \in \mathbb{R}^{d \times l}}{\arg\min} \frac{1}{2} \mathrm{Tr}\left(B^T (V - \gamma A) B\right) \\ \text{subject to } B^T B = I. \end{aligned} \tag{3.12}$$

As a result of Proposition 1, the solution to (3.12) is to set B_γ^*'s columns to be the l smallest eigenvectors of matrix $(V - \gamma A)$.

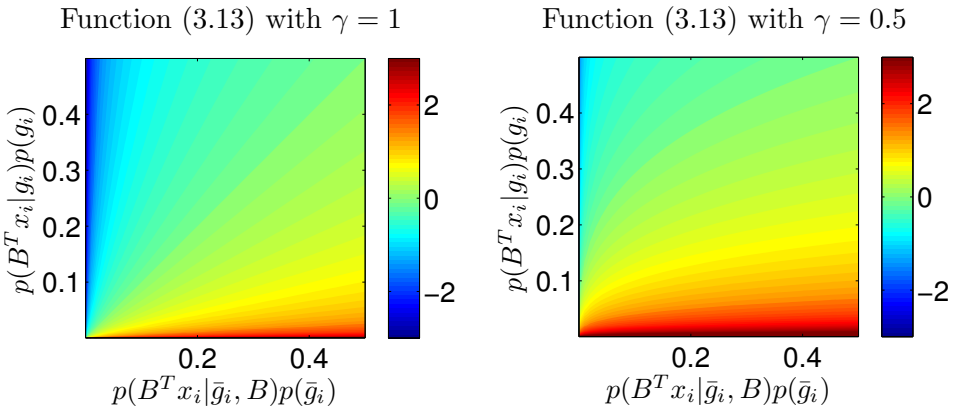

Figure 3.2: This figure shows the effect of varying γ in (3.13) as a function of the posterior probabilities resulting from B. The vertical axis for each plot is the posterior probability of the correct class after dimensionality reduction by B, and the horizontal axis is the posterior of the incorrect class.

In order to visualize the effect of this γ regularizer, it can be added to function (3.7) by multiplying the first sum by γ:

$$\sum_{i=1}^{n} \left(\left(\gamma \sum_{j=1}^{G} p_j \log p(B^T x_i|j) \right) - \log \left(p(B^T x_i|g_i) \right) \right). \tag{3.13}$$

I plot this function in Figure 3.2 for $\gamma = 1$ and $\gamma = 0.5$. The figure shows that reducing γ reduces the reward of making the correct class posterior large in relation to the incorrect class. At the same time, reducing γ makes the penalty more severe if the incorrect class posterior is large in relation to the correct class posterior.

As a result of (3.13), I choose to cross-validate γ over the range $(0, 1)$. I choose one as an upper bound for γ due to the fact that (3.13) no longer lower bounds (3.6) if $\gamma > 1$. Zero is chosen as the smallest value of γ because if $\gamma < 0$, then the objective function tries to bring x_i towards the incorrect-class mean, which is intuitively a bad objective.

3.2 Asymptotic Analysis of LDG

In this section, I present an asymptotic analysis of the LDG solution, B^*_γ in (3.12), as the number of training examples approaches infinity. In order to simplify the analysis, I

assume that I don't perform *local* LDG, but instead I perform *global* LDG, where I model $p(x_i|j) = \mathcal{N}(x_i; \hat{\mu}_j, I)$, (ie. there is a single Gaussian for each class).

I begin by rearranging the terms in (3.12), and dividing by n, which does not effect the minimization. Therefore, the function to be minimized in (3.12) can be written as

$$\frac{1}{n}\sum_{q=1}^{G}\sum_{i:g_i=q}\left((x_i-\hat{\mu}_q)^T BB^T(x_i-\hat{\mu}_q) - \gamma\sum_{j=1}^{G}\left(\hat{p}_j(x_i-\hat{\mu}_j)^T BB^T(x_i-\hat{\mu}_j)\right)\right). \tag{3.14}$$

Equation (3.14) can again be rewritten as

$$\mathrm{Tr}(B^T ZB), \tag{3.15}$$

where

$$Z = \sum_{q=1}^{G}\left(\frac{1}{n}\sum_{i:g_i=q}(x_i-\hat{\mu}_q)(x_i-\hat{\mu}_q)^T - \frac{\gamma}{n}\sum_{i:g_i=q}\sum_{j=1}^{G}\hat{p}_j(x_i-\hat{\mu}_j)(x_i-\hat{\mu}_j)^T\right). \tag{3.16}$$

First, I evaluate the asymptotic behavior of the first term inside the parenthesis of (3.16) for a single class q:

$$\frac{1}{n}\sum_{i:g_i=q}(x_i-\hat{\mu}_q)(x_i-\hat{\mu}_q)^T.$$

As $n\to\infty$, this term approaches a scaled version of the covariance matrix for class q. Let n_q be the number of class q training examples, then as $n\to\infty$,

$$\frac{1}{n}\sum_{i:g_i=q}(x_i-\hat{\mu}_q)(x_i-\hat{\mu}_q)^T \to \frac{n_q}{n}\Sigma_q = p_q\Sigma_q. \tag{3.17}$$

Now, I analyze the asymptotic performance of the second term inside the parenthesis in (3.16) for a single class q:

$$\frac{\gamma}{n}\sum_{i:g_i=q}\sum_{j=1}^{G}\hat{p}_j(x_i-\hat{\mu}_j)(x_i-\hat{\mu}_j)^T.$$

This term can be rewritten as

$$\frac{\gamma}{n}\sum_{j=1}^{G}\hat{p}_j\sum_{i:g_i=q}x_ix_i^T-x_i\hat{\mu}_j^T-\hat{\mu}_jx_i^T+\hat{\mu}_j\hat{\mu}_j^T. \tag{3.18}$$

Asymptotically, $\hat{p}_j \to p_j$ and $\hat{\mu}_j \to \mu_j$. Substituting in these values, and adding and subtracting $\mu_q\mu_q^T$ from (3.18) gives

$$\frac{\gamma}{n}\sum_{j=1}^{G}p_j\sum_{i:g_i=q}(x_ix_i^T-\mu_q\mu_q^T)-x_i\mu_j^T-\mu_jx_i^T+\mu_j\mu_j^T+\mu_q\mu_q^T. \tag{3.19}$$

As $n \to \infty$, the first term again becomes a scaled version of the class q covariance matrix. Additionally, $\sum_{i:g_i=q} x_i\mu_j^T \to n_q\mu_q\mu_j^T$ and $\sum_{i:g_i=q}\mu_jx_i^T \to n_q\mu_j\mu_q^T$. Therefore, (3.19) can be replaced with

$$\gamma\frac{n_q}{n}\sum_{j=1}^{G}p_j\left(\Sigma_q+(\mu_j-\mu_q)(\mu_j-\mu_q)^T\right)=\gamma p_q\left(\Sigma_q+\sum_{j=1,j\neq q}^{G}p_j(\mu_j-\mu_q)(\mu_j-\mu_q)^T\right). \tag{3.20}$$

Substituting (3.17) and (3.20) into (3.16) gives the asymptotic behavior of Z:

$$\begin{aligned}Z &\to \sum_{q=1}^{G}p_q\left((1-\gamma)\Sigma_q-\gamma\sum_{j=1,j\neq q}^{G}p_j(\mu_j-\mu_q)(\mu_j-\mu_q)^T\right)\\ &=(1-\gamma)\sum_{q=1}^{G}p_q\Sigma_q-\gamma\sum_{q=1}^{G}\sum_{j=1,j\neq q}^{G}p_j(\mu_j-\mu_q)(\mu_j-\mu_q)^T.\end{aligned} \tag{3.21}$$

The first term in (3.21) is the within-class covariance matrix, scaled by $1-\gamma$. The second term is the between-class covariance matrix, scaled by γ.

Substituting (3.21) back into (3.15) and (3.12) reveals the asymptotic LDG objective as the number of training samples $n \to \infty$:

$$\underset{B\in\mathbb{R}^{d\times\ell}}{\arg\min}\ \mathrm{Tr}\left(B^T\left((1-\gamma)\sum_{q=1}^{G}p_q\Sigma_q-\gamma\sum_{q=1}^{G}\sum_{j=1,j\neq q}^{G}p_j(\mu_j-\mu_q)(\mu_j-\mu_q)^T\right)B\right) \tag{3.22}$$

subject to $B^TB=I$.

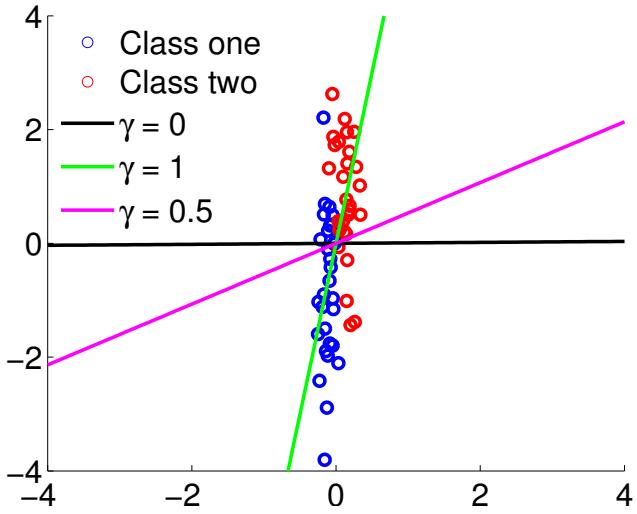

Figure 3.3: The class one and class two training data are represented by the blue and red circles. The lines show the one-dimensional subspaces that the data is mapped onto for different values of γ.

Equation (3.22) reveals that as $n \to \infty$, the LDG objective has two components. The first component is the within-class covariance matrix that LDG tries to make small after the mapping B. The second component is the between-class scatter matrix that LDG tries to make large after the mapping by forcing the class-conditional means to be separated. Unlike FDA, which tries to maximize the ratio of these two quantities in the projected space, as shown in (2.7), LDG balances the tradeoff between these two objectives through the parameter γ.

3.2.1 Illustrative Experiment

I present a simulation experiment to illustrate the relationship between γ and the LDG solution as revealed in the asymptotic analysis of the previous section. The simulation is a two-class problem, thirty training examples are provided for each class, and the data from each class is conditionally Gaussian. The mean of the data for class one is $[-0.1, -1]^T$ and the mean for class two $[0.1, 1]^T$. The covariances are equal, $\Sigma_1 = \Sigma_2$, and are diagonal, with the diagonal vector being $[0.01, 1]^T$. Therefore, the mapping that minimizes the class-

conditional covariance term in (3.21) for each class is $B = [1,\ 0]^T$. The mapping that minimizes the difference of means term in (3.21) is $B = [0.0995,\ 0.995]^T$.

Figure (3.3) shows the result of this experiment for three different values of γ. When $\gamma = 0$, B maps onto a one-dimensional space that mirrors the horizontal axis. When $\gamma = 1$, on the other hand, B maps onto a one-dimensional space that maximizes the difference between the two means. When $\gamma = 0.5$, we see that B achieves a balance between the $\gamma = 0$ and $\gamma = 1$ solutions. This is the behavior that is predicted as a result of the analysis of the previous section.

3.3 Extensions: Kernel LDG and Sparse LDG

This section describes two extensions of LDG: kernel LDG and sparse LDG. Kernel LDG finds a non-linear LDG mapping in Euclidean space via the kernel trick, and sparse LDG finds an LDG matrix such that the columns of B contain few nonzero elements.

Before describing kernel LDG, it is necessary to show how the V and A matrices in the LDG objective function (3.12) can be computed in block matrix form. After rewriting (3.12) using the block matrix form, kernel LDG is derived in a very similar manner to kernel PCA [57].

3.3.1 Block Matrix Forms for the LDG Matrices

The V and A matrices in the LDG objective function (3.12) can be written in block matrix form as a function of the data matrix X, a matrix whose i^{th} column is the i^{th} training feature vector, x_i.

Define $\mathcal{H}_{i,j}$ as a set that contains the class j local neighbors of training point x_i. Matrix V (3.9) can then be expanded as

$$V = \sum_{i=1}^{n} \left(x_i x_i^T - x_i \frac{1}{k} \sum_{q:q\in\mathcal{H}_{i,g_i}} x_q^T - \frac{1}{k} \sum_{q:q\in\mathcal{H}_{i,g_i}} x_q x_i^T + \frac{1}{k^2} \sum_{q:q\in\mathcal{H}_{i,g_i}} \sum_{c:c\in\mathcal{H}_{i,g_i}} x_q x_c^T \right). \quad (3.23)$$

Now, define N_{sc} as a *same-class* neighbor matrix, where $N_{\text{sc}}(i,q) = \frac{1}{k}$ if x_q is a class g_i neighbor of x_i and is zero otherwise. Similarly, define N_j as a class j neighbor matrix,

where $N_j(i,q) = \frac{1}{k}$ if x_q is a class j neighbor of x_i and is zero otherwise. Finally, let $E_{i,q}$ be an all-zero matrix except for element (i,q) that is set to 1. Using these definitions (3.23) can be rewritten as as

$$\begin{aligned} V &= X\left(\sum_{i=1}^{n}\left(E_{i,i} - \frac{1}{k}\sum_{q:q\in\mathcal{H}_{i,g_i}}(E_{i,q}+E_{q,i}) + \frac{1}{k^2}\sum_{q:q\in\mathcal{H}_{i,g_i}}\sum_{c:c\in\mathcal{H}_{i,g_i}}E_{q,c}\right)\right)X^T \\ &= X\left(I - N_{\text{sc}} - N_{\text{sc}}^T + N_{\text{sc}}^T N_{\text{sc}}\right)X^T. \end{aligned} \tag{3.24}$$

Similarly, the LDG A matrix (3.10) can be rewritten

$$\begin{aligned} A &= X\left(\sum_{j=1}^{G}p_j\sum_{i=1}^{n}\left(E_{i,i} - \frac{1}{k}\sum_{q:q\in\mathcal{H}_{i,j}}(E_{i,q}+E_{q,i}) + \frac{1}{k^2}\sum_{q:q\in\mathcal{H}_{i,j}}\sum_{c:c\in\mathcal{H}_{i,j}}E_{q,c}\right)\right)X^T \\ &= X\left(\sum_{j=1}^{G}p_j\left(I - N_{\text{j}} - N_{\text{j}}^T + N_{\text{j}}^T N_{\text{j}}\right)\right)X^T. \end{aligned} \tag{3.25}$$

3.3.2 Kernel LDG

Define $L_V = \left(I - N_{\text{sc}} - N_{\text{sc}}^T + N_{\text{sc}}^T N_{\text{sc}}\right)$ and $L_A = \sum_{j=1}^{G}p_j\left(I - N_{\text{j}} - N_{\text{j}}^T + N_{\text{j}}^T N_{\text{j}}\right)$. Substituting these definitions into (3.24) and (3.25), the LDG optimization problem (3.12) can be rewritten as

$$\begin{aligned} B_\gamma^* = \underset{B\in\mathbb{R}^{d\times l}}{\arg\min}\ &\frac{1}{2}\text{Tr}\left(B^T X(L_V - \gamma L_A)X^T B\right) \\ &\text{subject to } B^T B = I, \end{aligned}$$

which is solved by an eigendecomposition of the matrix $X(L_V - \gamma L_A)X^T$. Let λ be an eigenvalue corresponding to eigenvector ν. This pair must satisfy the linear relationship

$$X(L_V - \gamma L_A)X^T\nu = \lambda\nu. \tag{3.26}$$

From (3.26), it is possible to derive a kernelized version of LDG using a derivation similar to that for kernel PCA [57, 58]. First, notice that in order for (3.26) to be satisfied, ν must be a linear combination of the columns of X. Therefore, $\nu = X\alpha$ for some vector

α. Substituting this into (3.26), and pre-multiplying by X^T gives

$$X^T X(L_V - \gamma L_A)X^T X\alpha = \lambda X^T X\alpha. \tag{3.27}$$

The term $X^T X$ in (3.27) is the linear kernel function for the training data. Using the kernel trick (see Section 2.1.4), $X^T X$ can be replaced with any positive semi-definite kernel matrix K in order to find the LDG components in some high-dimensional reproducing kernel Hilbert space (RKHS). Performing this substitution gives

$$K(L_V - \gamma L_A)K\alpha = \lambda K\alpha. \tag{3.28}$$

Eigenvector and eigenvalue pairs that satisfy 3.28 can be found by solving

$$(L_V - \gamma L_A)K\alpha = \lambda\alpha. \tag{3.29}$$

It is evident that all solutions to (3.29) also satisfy (3.28). The opposite is not true, as it is possible that there are solutions to (3.28) that do not satisfy (3.29). However, these solutions have been shown to be irrelevant [58].

Let F be the kernel LDG solution that is constructed by setting the columns of F to be the ℓ smallest eigenvectors of $(L_V - \gamma L_A)K$. F has several properties that are different than the standard LDG solution. First, $(L_V - \gamma L_A)K$ is an $n \times n$ matrix, and thus $F \in \mathbb{R}^{\ell \times n}$ with $\ell \leq n$. Furthermore, in linear LDG, the columns of B were set to be the ν satisfying (3.26), whereas the columns of F are set to the α satisfying (3.29). Due to the relationship $\nu = X\alpha$, and the fact that B maps a test vector according to $B^T x$, F maps a test vector according to

$$\begin{aligned} &F^T X^T x \\ &= F^T[K(x_1, x)\ K(x_2, x)\ ...\ K(x_n, x)]^T, \end{aligned}$$

where $K(x_i, x)$ is the evaluation of the kernel function between training sample x_i and test sample x.

3.3.3 Sparse LDG

Sparse LDG adapts the LDG objective function so that each column of B has few nonzero elements. Since LDG and PCA are both solved via the maximal eigenvalue decomposition of a square matrix, we can adapt methods for sparse PCA, such as those found in [32, 59, 72], to find sparse LDG matrices as well. In this section, I adapt the direct sparse PCA algorithm of d'Aspremont, et al. [15] to LDG. The adaptation is essentially a direct copy of the sparse PCA algorithm found in [15], except that the LDG matrix $(V - \gamma A)$ is substituted in for the PCA covariance matrix S in (2.4).

Suppose that we wish to find a single column of B_γ in (3.12), denoted by $b_\gamma \in \mathbb{R}^{d\times 1}$. For brevity, in the rest of this section I will drop the γ and also substitute $Z = (V - \gamma A)$. Then the LDG optimization problem (3.12) can be rewritten as

$$\begin{aligned} b^* = \underset{b\in\mathbb{R}^{d\times 1}}{\arg\min}\ & b^T Z b \\ \text{subject to } & b^T b = 1. \end{aligned} \tag{3.30}$$

In order to ensure that b^* is s sparse, meaning that no more than s of the d terms of b^* are nonzero, a cardinality constraint can be added to (3.30):

$$\begin{aligned} b^* = \underset{b\in\mathbb{R}^{d\times 1}}{\arg\min}\ & b^T Z b \\ \text{subject to } & b^T b = 1, \\ & \text{Card}(b) \le s. \end{aligned} \tag{3.31}$$

However, the cardinality constraint makes the above problem NP-hard.

The authors of [15] show that by making several relaxations, (3.31) can be converted into a convex semidefinite program. Let $\tilde{B} = bb^T$, $|\ \tilde{B}\ |$ be a matrix whose entries take the absolute values of the entries of $\tilde{B}$, and $\mathbf{1}$ be a $d \times 1$ vector of all ones. Using these

definitions, a convex relaxation of (3.31) is

$$\begin{aligned} \tilde{B}^* = \underset{\tilde{B}\in\mathbb{R}^{d\times d}}{\arg\min}\ & \mathrm{Tr}\left(Z\tilde{B}\right) \\ \text{subject to } & \mathrm{Tr}(\tilde{B}) = 1, \\ & \mathbf{1}^T \mid \tilde{B} \mid \mathbf{1} \leq s, \\ & \tilde{B} \succeq 0. \end{aligned} \tag{3.32}$$

After solving (3.32), the solution b^* is found by decomposing the rank one matrix $\tilde{B}^*$. The relaxed optimization problem (3.32) does not guarantee that the solution b^* will have s or fewer nonzero components; however, in practice it does produce a sparse solution, with smaller s resulting in increased sparsity. See [15] for empirical results showing how s relates to the sparsity of b^*.

The sparse LDG optimization problem (3.32) is capable of finding only a single column of B^*. In order to find ℓ columns of B^*, a two-step iteration must be used where the first step in the iteration is to solve (3.32) and the second step is matrix *deflation*. Let b_i be the solution to (3.32) at the i^{th} iteration. The deflation step removes the influence of b_i from Z at the end of each iteration. The authors in [15] proposed to perform deflation as $Z_{i+1} = Z_i - (b_i^T Z_i b_i) b_i b_i^T$. I call this deflation procedure *sparse deflation*, as it produces sparser columns than the alternative described in the next paragraph, but does not guarantee that the columns are orthonormal.

The following *orthonormal deflation* procedure guarantees that the columns of B^* are orthonormal. Let $B_i = [b_1, b_2, ..., b_i] \in \mathbb{R}^{d\times i}$ be the current sparse LDG solution after i iterations. Furthermore, let $G_i \in \mathbb{R}^{d\times d-i}$ be a matrix whose columns are orthonormal to the columns of B_i. The deflated matrix is $Z_{i+1} = G_i^T Z G_i \in \mathbb{R}^{d-i\times d-i}$. Solving (3.32) using Z_{i+1} results in the solution $b \in \mathbb{R}^{d-i\times 1}$, and using this b we can solve for b_{i+1} as $b_{i+1} = G_i b$. Since the columns of G_i are orthonormal to B_i, b_{i+1} is orthonormal to all the columns of B_i.

Table 3.1: Mean training time in seconds and mean classification accuracy when the number of dimensions is chosen by cross-validation. Bold font highlights methods that were statistically the best or not statistically different from the best with 95% confidence. Symbol legend: "-" method did not converge in under three hours per training/test split, "nc" not computable.

	Orig	Train	LDG		PCA		FDA		LFDA		ITML		NCA	
	Dim	Ex	acc	time	acc	time	acc	time	acc	time	acc	time	acc	time
Diabetes	8	538	**71.3**	1	67.9	<1	**72.7**	<1	69.2	<1	69.7	40	**70.7**	135
Wine	13	125	**97.7**	<1	95.8	<1	**98.5**	<1	**98.5**	<1	**97.2**	54	**97.9**	6
Image Seg	19	1617	**96.5**	4	92.5	2	**96.2**	1	95.0	3	95.6	138	95.4	4641
German	20	700	**71.1**	<1	68.9	<1	**71.4**	<1	**71.9**	<1	69.5	15	**71.1**	1015
Ringnorm	20	3000	**86.9**	9	85.8	5	71.9	1	85.8	6	80.6	23	85.7	9119
Derm	33	256	**89.2**	<1	92.5	<1	91.5	<1	92.5	<1	**93.4**	139	**95.5**	381
Ion	34	246	**86.2**	<1	85.2	<1	83.2	<1	83.1	<1	84.3	10	**89.1**	222
Statlog	36	3000	**90.1**	10	**90.1**	5	86.6	3	88.3	6	88.0	232	-	-
USPS	256	3000	**93.5**	24	**92.0**	10	90.9	5	92.6	7	90.8	4886	-	-
Isolet	617	3000	87.9	94	73.1	15	**88.9**	18	**90.2**	21	-	-	-	-
MNIST	784	3000	**89.1**	55	**87.5**	13	76.2	10	32.7	34	-	-	-	-
Gisette	5000	3000	**95.5**	466	**96.7**	63	51.5	1983	49.8	2313	-	-	-	-
Mutants	5408	200	**90.0**	2908	64.8	2	52.0	1946	48.0	2003	-	-	-	-
Arcene	10K	70	**76.0**	578	61.3	15	53.7	39	nc	nc	-	-	-	-
Dexter	20K	210	**84.0**	4365	58.1	2	nc	nc	nc	nc	-	-	-	-

The drawback of the orthonormal deflation procedure compared to the sparse deflation procedure is that as i increases, the sparsity of $b_{i+1} = G_i b$ decreases. In Section 3.4.1, I will highlight the differences between these two deflation procedures through experiments.

3.4 Dimensionality Reduction Experiments

I perform experiments to compare LDG to several different dimensionality reduction methods: PCA, FDA, LFDA, NCA, and information theoretic metric learning (ITML) [16] with feature selection using the maximum-relevance, minimum redundancy criterion (MRMR) [46]. For NCA, LFDA, ITML, and MRMR feature selection, I use code provided by the authors. I evaluate the performance of the dimensionality reduction methods via k-NN classification accuracy with $k = 3$, as was done in [68].

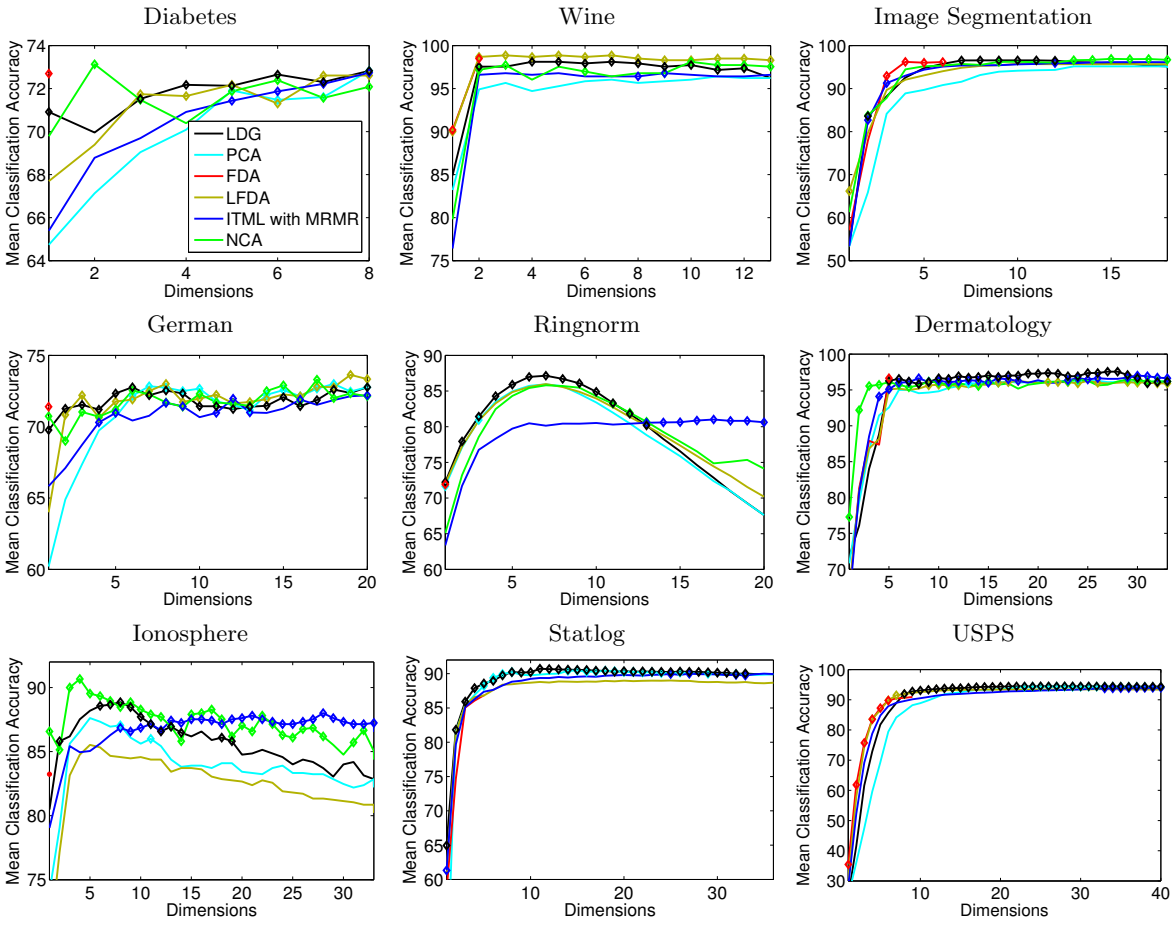

Figure 3.4: Mean classification accuracy over ten random splits of the data. Diamonds mark methods that are statistically the best or not statistically different from the best with 95% confidence for that dimensionality.

As a preprocessing step, the training data is standard normalized so that each feature has a mean of zero and standard deviation of one. I choose the number of neighbors used to estimate the local Gaussians for LDG dimensionality reduction by five-fold cross-validation using a local QDA classifier [24] on the original data. For LDG, $\gamma \in \{0.2, 0.4, 0.6, 0.8, 1\}$, and I choose whichever γ minimizes the k-NN leave-one-out cross-validation error at dimensionality equal to the number of classes plus five. I have found that, in general, a few more dimensions than the number of classes present in the data is a good dimensionality at which to choose γ. In the case of ties, I select the largest value of γ. MRMR requires that I

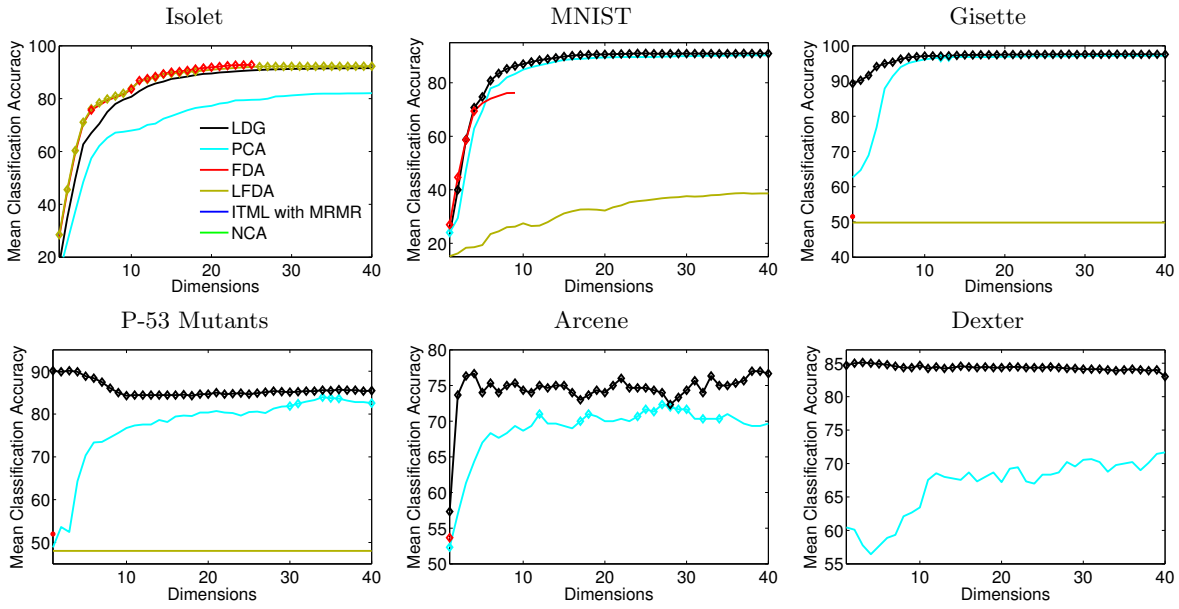

Figure 3.5: Mean classification accuracy over ten random splits of the data. Diamonds mark methods that are statistically the best or not statistically different from the best with 95% confidence for that dimensionality.

discretize the ITML features for feature selection, and I do so by thresholding at the mean, as recommended in the authors' code.

Experiments are performed on fifteen datasets, and for each the accuracy is averaged over ten random 70/30 splits of the training and test data (up to a maximum of 3000 training samples). The datasets can be found either at the UCI Machine Learning Repository or the Machine Learning Dataset Repository. The P53-Mutants dataset contained a large degree of class asymmetry. Therefore, I randomly sampled 143 of the *inactive* class samples and discarded the rest in order to make a 50/50 split between *inactive* and *active* class data (as opposed to the 1% vs 99% split in the original dataset).

Figures 3.4 and 3.5 and Table 3.1 show that for small datasets, LDG is comparable to other state-of-the-art methods. However, LDG provides a clear advantage on the datasets with the largest feature dimensionality.

NCA and ITML failed to converge in under three hours per training/test split on a standard 2.8 GHz PC for the datasets marked with "-" in Table 3.1, and results for these

datasets are not plotted in Figures 3.4 and 3.5. Figures 3.4 and 3.5 also shows that ITML has difficulty with the Ringnorm dataset which has some features that are only noise.

LDG also outperforms FDA and LFDA on some of the datasets. FDA can provide dimensionality only up to one fewer than the number of classes, which limits its performance on the Ionoshpere and Ringnorm datasets. Furthermore, FDA and LFDA exhibit numerical instability in some of the datasets with large feature dimensionality due to the fact that the within-class covariance matrix is underdetermined. Thus, the generalized eigenvalue decomposition that these algorithms solve fails to find discriminative dimensions. LFDA returns complex eigenvalues for the Arcene and Dexter datasets, and FDA does the same on the Dexter dataset; thus, the LFDA and FDA results are not computable for these datasets.

Table 3.1 shows the average classification accuracy when the dimensionality is chosen by leave-one-out cross-validation. I do this in a greedy fashion by increasing the dimensionality, up to forty dimensions, until the cross-validation accuracy decreases by adding another dimension. The run-time numbers measure the mean time it takes, in seconds, for the method to produce the dimensions shown in Figures 3.4 and 3.5 and to select the best dimensionality.

3.4.1 Sparse LDG Experiments

The experiments in this section serve to highlight the differences between LDG, sparse LDG using the sparse deflation method, and sparse LDG using the orthonormal deflation method. All experiments in this section are performed on the Wine dataset, and the data preprocessing matches that described in Section 3.4. All results are again averaged over ten random 70/30 splits of the training vs. test data.

The value of γ for sparse LDG is set to be the same value of γ that is selected for non-sparse LDG. The sparseness parameter is set to $s = 3$ for all experiments. Furthermore, for simplicity I use CVX [27] running Sedumi [63] to solve the sparse LDG semidefinite program (3.32); however, a faster algorithm is proposed in [15].

I first compare the sparseness of the B matrices produced by the three different methods. Figure 3.6 plots the mean number of nonzero elements in each column of B, where the mean

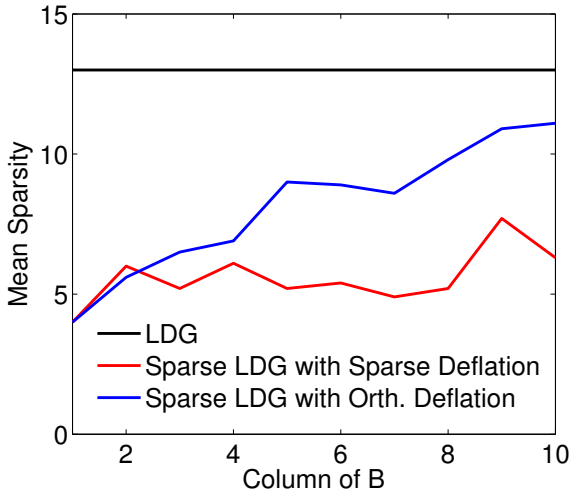

Figure 3.6: The figure plots the mean number of nonzero elements in each column of B for LDG and sparse LDG with two different deflation methods.

is taken over the ten random training vs. test splits. The figure shows that standard LDG finds a matrix B that has no sparsity. Sparse LDG, on the other hand, finds columns of B that are sparse. Comparing sparse deflation to orthonormal deflation, we can see that when there are few columns of B, the methods produce columns with similar sparsity. In fact, the solution for the first column of B is the same for the two methods as it involves no deflation. However, as we add more columns to B, the sparse deflation method produces sparser columns than the orthonormal method.

Sparse deflation does not produce a B matrix that is orthonormal. Figure 3.7 plots the mean of the absolute value of the inner product between the columns of B for sparse LDG with sparse deflation and sparse LDG with orthonormal deflation. If the columns of B are orthonormal, then the off-diagonal elements of the plots in Figure 3.7 will be zero and the diaganol elements will be one. The figure shows that orthonormal deflation produces columns that are orthonormal and sparse LDG does not.

Figure 3.8 compares the mean classification accuracy of the three different dimensionality reduction methods for three nearest neighbor classification. The figure shows that LDG is the best or statistically significantly tied for the best at all dimensions. Sparse LDG with

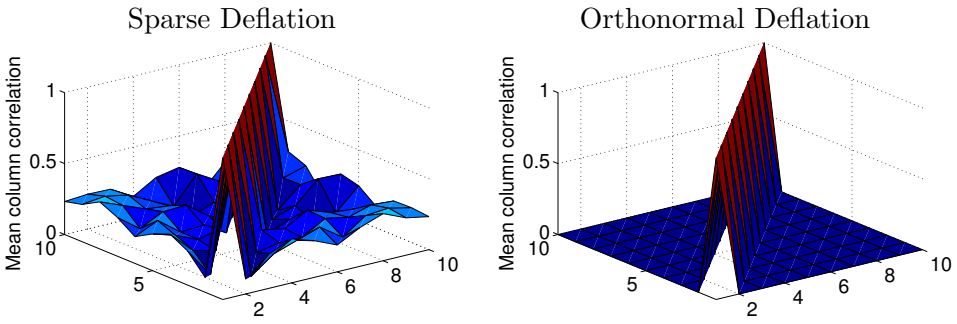

Figure 3.7: The figure shows the mean of the absolute value of the inner product between the columns of B produced by sparse LDG with sparse deflation and sparse LDG with orthonormal deflation.

orthonormal deflation is statistically tied with LDG at most dimensions. Sparse LDG with sparse deflation results in the lowest mean classification accuracy, but is statistically tied with the other two methods at several dimensions.

3.5 Small Sample Size Dimensionality Reduction

The results in Figures 3.4 and 3.5 show that FDA and LFDA perform extremely poorly on the datasets with the largest feature dimensionality. This is due to the *small sample size* problem in dimensionality reduction. FDA and LFDA both try to maximize the ratio of the between-class covariance to within-class covariance by solving the optimization problem

$$B^* = \underset{B \in \mathbb{R}^{d \times \ell}}{\arg\max} \left(\frac{B^T S_b B}{B^T S_w B} \right).$$

The solution to this problem is achieved by setting the columns of B to the ℓ largest eigenvectors of the generalized eigenvalue decomposition

$$S_b \nu = \lambda S_w \nu,$$

which is equivalent to the eigenvalue decomposition of

$$S_w^{-1} S_b \nu = \lambda S_w \nu.$$

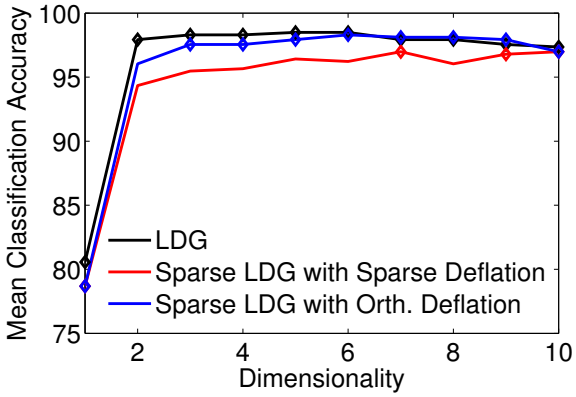

Figure 3.8: The figure plots the mean classification accuracy for ten random training vs. test splits of the Wine dataset. Diamonds indicate that the method is either the best or not statistically significantly worse than the best at that dimensionality.

Therefore, if S_w is underdertermined, the above problem is ill-posed and numerically unstable.

The rank of S_w for FDA is at most $n - G$, the number of training samples minus the number of classes, and thus the small sample problem arises whenever $n < d+G$. The small sample size problem can occur if even more samples are provided for $LFDA$, as is apparent in the results for the MNIST dataset in Figures 3.4 and 3.5.

Many possible solutions to this problem have been proposed. One of the earliest is *Fisher faces* [5]. The idea of Fisher faces is to first reduce the dimensionality of the training data to $n - G$ using PCA and then to perform FDA on the resulting data. Let $B_{PCA} \in \mathbb{R}^{d \times n-G}$ be the PCA matrix and $B_{FDA} \in \mathbb{R}^{n-G \times \ell}$, $\ell < n - G$ be the FDA matrix. The resulting Fisher faces dimensionality reduction matrix is $B_{PCA}B_{FDA} \in \mathbb{R}^{d \times n-G}$. I abbreviate this algorithm as *PCA FDA*, and it it can easily be extended to *PCA LFDA* by first using PCA to reduce the data to a dimensionality such that the LFDA S_w matrix is full rank, and then performing LFDA on the resulting data.

Another possible solution is to regularize the within-class covariance matrix. Ridge regularization [28] does so by adding a scaled identity matrix: $S_w(\epsilon) = S_w + \epsilon I$. The scaling parameter ϵ can be chosen by cross-validation in order to maximize classification accuracy.

I call this method *Regularized FDA* or *Regularized LFDA* for Fisher discriminant analysis and local Fisher discriminant analysis, respectively.

Another approach to solving the small sample size problem is called *direct LDA* (DLDA) [71]. DLDA first decomposes the between-class covariance matrix and then uses the resulting decomposition to decompose the within-class covariance matrix. The final dimensionality reduction matrix is the result of both of these decompositions. First, DLDA finds a dimensionality reduction matrix $V_{\text{DLDA}} \in \mathbb{R}^{d\times u}, u < d$ that is composed of the u non-zero eigenvectors of the between class covariance matrix S_b. The columns of V_{DLDA} are normalized so that $V_{\text{DLDA}}^T S_b V_{\text{DLDA}} = I$, and thus V_{DLDA} spheres the between-class covariance. Then, let $W_{\text{DLDA}} \in \mathbb{R}^{u\times G}$ be the top G eigenvectors of $V_{\text{DLDA}}^T S_w V_{\text{DLDA}}$. The final DLDA dimensionality reduction matrix is $B_{\text{DLDA}} = V_{\text{DLDA}} W_{\text{DLDA}} \in \mathbb{R}^{d\times G}$.

Generalized discriminant analysis (GDA) is another small sample size solution that reformulates standard Fisher discriminant analysis as a generalized singular value decomposition [29]. This generalized singular value decomposition does not become ill-posed for the small sample size problem. For details on how FDA is reformulated as GDA, see [29].

The most similar method to LDG within the small sample size literature is the maximum margin criteria (MMC) [34]. MMC finds the dimensionality reduction matrix by solving the following optimization problem:

$$B^*_{MMC} = \underset{B\in\mathbb{R}^{d\times\ell}}{\arg\max}\ \text{Tr}\left(B^T(S_b - S_w)B\right) \qquad (3.33)$$
$$\text{subject to } (B^T B) = I,$$

where S_b and S_w are the between-class and within-class matrices used in FDA (see (2.5) and (2.6)). The authors of [34] motivate the MMC objective function as finding the B that maximizes the margin between the same-class and different-class training data; thus their analysis is quite different than that for LDG. Furthermore, the MMC objective differs from that of LDG in that LDG adds the regularization term γ and LDG acts locally to each training point whereas MMC considers the global scatter matrices.

3.5.1 Computational Savings Using Kernelized LDG

In Section 3.3.2, kernel LDG was motivated as a method to find non-linear mappings of the input data using the kernel trick. However, kernel LDG can also provide computational savings when there are fewer training examples than there are features.

Equation (3.29) showed that the kernel LDG solution requires that we find the ℓ smallest eigenvectors of $(L_V - \gamma L_A)K$. If there are n training examples, then the size of this matrix is $n \times n$. Compare this to the standard linear form of the LDG objective function (3.12) that requires us to find the ℓ smallest eigenvectors of a $d \times d$ matrix. If $n < d$, the eigendecomposition is simplified using kernel LDG. Furthermore, we can find the linear LDG solution using kernel LDG with the linear kernel: $K(x_i, x_j) = x_i^T x_j$.

3.5.2 Small Sample Size Experiments

I conduct experiments to compare LDG to the methods proposed above, as well as ITML, on a large number of datasets that exhibit the small sample size problem. I again compare performance by $k = 3$ nearest-neighbor classification accuracy on the mapped data. The datasets that I use are all time-series datasets available at the *UCR time-series classification and clustering page*. I use this database for these experiments due to the large number of time-series datasets that exhibit the small sample size problem.

Each dataset at the UCR database consists of sampled time-series signals from a number of classes, and each dataset has a standard training vs. test split. It is common to use the time-series samples as the features for classification, and doing so results in twenty-five datasets that exhibit the small sample size problem. Table 3.2 gives details for the twenty-five datasets. Although the UCR database pre-separates the data into standard training vs. test splits, I instead combine these two sets and randomly sample a training vs. test split (with the same number of samples for training and test as exist in the standard split) ten times for each dataset in order to get statistically significant results.

The only methods requiring cross-validation parameters are LDG, regularized FDA, regularized LFDA, and ITML. For LDG I cross-validate the γ parameter over [0.01, 0.1, 0.3, 0.5, 0.7, 0.9]. For regularized LDA and regularized LFDA, I cross-validate the ϵ param-

Table 3.2: Details for the time-series datasets that exhibit the small sample size problem.

Dataset	Time-series Length	Classes	Training Samples	Test Samples
Beef	470	5	30	30
CBF	128	3	30	900
Cinc ECG Torso	1639	4	40	1380
Coffee	286	2	28	28
Diatom Size Reduction	345	4	16	306
ECG Five Days	136	2	23	861
Face Four	350	4	24	88
Fish	463	7	175	175
Gun Point	150	2	50	150
Haptics	1092	5	155	308
Inline Skate	1882	7	100	550
Lightning 2	637	2	60	61
Lightning 7	319	7	70	73
Mallat	1024	2	67	1029
Mote Strain	84	2	20	1252
Olive Oil	570	4	30	30
OSU Leaf	427	6	200	242
Sony AIBO Robot Surface	70	2	20	601
Sony AIBO Robot Surface II	65	2	27	953
Starlight Curves	1024	3	1000	8236
Symbols	398	6	25	995
Trace	275	4	100	100
Two Lead ECG	82	2	23	1139
Word Synonyms	270	25	267	638
Yoga	426	2	300	3000

eter over [0.001, 0.01, 0.1, 1, 10, 100, 100]. For ITML, I use the default cross-validation parameters provided in the authors' code.

Due to the large number of datasets, I do not include plots like those shown in Figures 3.4 and 3.5. Instead, I present all results in Tables 3.3 and 3.4. These tables show results for ITML only on datasets with fewer than three hundred features due to the excessive run-time of ITML on datasets with large feature dimensionality. Table 3.3 shows the classification accuracy of the various dimensionality reduction methods when reducing the data to a final dimensionality equal to one fewer than the number of classes: $G - 1$. I choose to show the results at this dimensionality because this is the inherent dimensionality chosen by the variants of FDA. Showing results at this dimensionality shows how well the methods can do when reducing to a small final dimensionality.

Table 3.4 shows the results for PCA, LDG, PCA LFDA, and regularized LFDA at a number of dimensions chosen in order to maximize classification accuracy by cross-validation. In the experiments of Section 3.4, I used a completely greedy method to choose the best dimensionality. In the small sample size problem of this section; however, I have found that this greedy method can incorrectly choose too few dimensions. Therefore, I instead choose the best dimensionality from a fixed set of candidate dimensions. The set of candidate dimensions is [G - 1, G, G+1, G+2, G+4, G+8, G+16, G+32], where G is the number of classes. The reason behind choosing this set of dimensions to cross-validate over is that if the data is unimodal in each dimension, then a final dimensionality of $G-1$ often performs very well. Otherwise, a few more dimensions than this is needed, as can be seen in Figures 3.4 and 3.5. LDG and regularized LFDA must cross-validate over dimensions and parameters. As opposed to doing a grid search, I instead first choose the γ parameter for LDG or the ϵ parameter for regularized LFDA based on the $k = 3$ nearest-neighbor cross-validation accuracy at dimensionality $G - 1$. I then use this parameter to cross-validate over the number of dimensions. For PCA LDA, Regularized LDA, DLDA, and GDA, the results are again shown at dimensionality $G - 1$.

For ITML, the results in Table 3.4 involve no dimensionality reduction. Instead, the performance is evaluated using the full Mahalanobis metric found by ITML (see Section 2.2.5). I choose to use the full metric for ITML because the results in the previous section

in Figure (3.4) showed that ITML rarely saw accuracy improvements with reduced dimensionality. Therefore, the performance using the full Mahalanobis metric is a good point of comparison for results chosen to maximize classification accuracy.

Table (3.3) shows that in the special case that the dimensionality is constrained to be $G-1$, regularized FDA is either the best method or statistically significantly tied for the best in seventeen of twenty-five datasets, followed by LDG and regularized LFDA each at fifteen of twenty-five. ITML is is either the best or statistically significantly tied for the best only six of the ten datasets with fewer than three-hundred features, while regularized LFDA is the best or statistically significantly tied for the best on nine of those ten. PCA LDA, DLDA, GDA, and PCA LFDA are each the best or statistically significantly tied for the best on fewer than ten of twenty-five datasets.

Table (3.4) shows that when the dimensionality is chosen by cross-validation, LDG is the best or statistically significantly tied for the best on nineteen of twenty-five datasets, and regularized LFDA comes in second with sixteen of twenty-five. Regularized FDA loses ground in this setting due to its shortcomings of not being able to separate multi-modal data, and not being able to find more then $G-1$ dimensions. ITML is the best or statistically significantly tied for the best on six of ten datasets, while LDG and regularized LFDA are the best or statistically signficantly tied for the best on eight of those ten. In addition, run-time comparisons in Table (3.5) show that ITML requires significantly more training time than the other methods.

3.6 Conclusions

I have motivated and derived LDG dimensionality reduction as a mapping that finds an approximate solution to that which minimizes the error of a local QDA classifier. Asymptotically, LDG attempts to simultaneously minimize the within-class scatter while maximizing the distance between the means of different classes. This objective is similar to FDA; however, LDG does not require an inversion of the within-class scatter matrix and can therefore be applied directly to small sample size problems.

Experiments showed that LDG is comparable with other state-of-the-art algorithms for linear dimensionality reduction. The fact that LDG requires only a maximal eigenvalue

Table 3.3: Results of the small sample size dimensionality reduction experiments when the final dimensionality of the data is one fewer than the number of classes. The results are averaged over ten random splits of the training and test data, and methods that are the best or not different from the best with statistical significance of 95% are shown in bold.

Dataset	LDG	PCA	PCA FDA	Reg. FDA	DLDA	GDA	PCA LFDA	Reg. LFDA	ITML
Beef	**87.7**	42.3	**86.3**	**87**	37.3	**87.3**	59.7	81.7	-
CBF	88.3	76.8	85.5	89.3	83	82.1	85.3	**92.4**	91.2
Cinc ECG Torso	**63.4**	**63.5**	46.4	48.4	55.8	45.2	38.4	**60.8**	-
Coffee	**99.6**	58.2	**100**	**100**	61.4	**100**	**100**	**100**	**99.6**
Diatom	**90.1**	86.7	87.8	**89.4**	88.7	**88.6**	80.8	87	-
ECG Five Days	**93.6**	51.3	**94.8**	**94.8**	77.1	**94.8**	**94.8**	92	**93.8**
Face Four	**81.7**	67.5	**82.2**	**82.3**	77.6	**83.1**	65.9	80.9	-
Fish	74.2	64.5	65.3	**83.1**	70.2	62.6	35.7	78.9	-
Gun Point	73.3	64	77.5	**84.4**	69.4	78.1	77.9	**80.9**	79.5
Haptics	**40.3**	32	32.2	**40.5**	36	33	26.8	38.7	-
Inline Skate	**25**	**26.5**	24.2	24.5	**24.9**	24.6	19.3	**26.5**	-
Lightning 2	**70.7**	65.2	60.7	**67**	**70**	61	60.5	66.9	-
Lightning 7	62.1	**65.3**	48.1	60.3	58.6	45.3	29	61	-
Mallat	92.6	81.2	87.6	91.5	62.1	60.1	61	**93.6**	-
Mote Strain	**85.3**	74.7	82.2	**84.6**	**84.6**	80.9	82.3	**84.3**	**83.3**
Olive Oil	**87.3**	66	**87**	**88.3**	78.7	**86**	60.7	**86.3**	-
OSU Leaf	47.5	**49.7**	32.5	44	47.4	33.2	23.7	**50.4**	-
Sony	**87.4**	67.1	86.2	**87.8**	**88.1**	85.9	86.5	**87.8**	**88.6**
Sony II	**80.7**	75.1	81.1	**81.4**	**79.2**	**81.1**	81.1	**81.6**	**82.2**
Starlight Curves	83.8	84.4	54.9	**85.2**	84.1	46.7	63.9	**84.9**	-
Symbols	**73.4**	**73.5**	71	**74.4**	59.9	58.3	55.7	70	-
Trace	75.2	71.3	73.9	74.9	76	63.4	68.2	**78.8**	**76.4**
Two Lead ECG	92	54.1	**95.8**	**95.8**	64.8	**96**	**95.9**	**95.8**	95.1
Word Synonyms	**58.7**	58	20.9	54.7	40.5	5.3	7.1	**59.3**	56.3
Yoga	56.8	58.9	58.9	**66**	56.4	58.9	59	58.2	-

Table 3.4: Results of the small sample size dimensionality reduction experiments when the final dimensionality of the data chosen by cross-validation for LDG, PCA, PCA LFDA and Regularized LFDA. For easy analysis, we also compare the results for the other methods at dimensionality of one fewer than the number of classes, where they commonly achieve the best performance. The results are averaged over ten random splits of the training and test data, and methods that are the best or not different from the best with statistical significance of 95% are shown in bold.

Dataset	LDG	PCA	PCA FDA	Reg. FDA	DLDA	GDA	PCA LFDA	Reg. LFDA	ITML
Beef	**87.7**	43	**86.3**	**87**	37.3	**87.3**	59.7	81.7	-
CBF	88.8	84	85.5	89.3	83	82.1	85.3	**92.4**	90.3
Cinc ECG Torso	**68.9**	**70.3**	46.4	48.4	55.8	45.2	38.4	66.9	-
Coffee	**99.6**	68.9	**100**	**100**	61.4	**100**	**100**	**100**	**99.6**
Diatom	**90.1**	88.7	**87.8**	**89.4**	88.7	**88.6**	80.8	87	-
ECG Five Days	**93.8**	76.8	**94.8**	**94.8**	77.1	**94.8**	**94.8**	**92.4**	**91.9**
Face Four	**82.8**	74.1	**82.2**	**82.3**	77.6	**83.1**	65.9	80.9	-
Fish	75.1	76.6	65.3	**83.1**	70.2	62.6	35.7	**82.9**	-
Gun Point	**88**	79.6	77.5	84.4	69.4	78.1	77.9	84.9	83.8
Haptics	**40.1**	37.2	32.2	**40.5**	36	33	26.7	**41.4**	-
Inline Skate	**25.1**	**26.9**	24.2	24.5	24.9	24.6	19.3	**26.9**	-
Lightning 2	**72.1**	**74.3**	60.7	67	70	61	60.5	**69.5**	-
Lightning 7	**62.3**	**64.2**	48.1	60.3	58.6	45.3	29	**61.4**	-
Mallat	92.6	80.1	87.6	91.5	62.1	60.1	61	**93.6**	-
Mote Strain	**85.2**	83	82.2	**84.6**	**84.6**	80.9	82.3	**84.3**	83.2
Olive Oil	**87.3**	75.3	**87**	**88.3**	78.7	**86**	60.7	**86.3**	-
OSU Leaf	**51**	**54.8**	32.5	44	47.4	33.2	23.7	**53.4**	-
Sony	**90**	86.4	86.2	**87.8**	88.1	85.9	86.5	87.8	**87.4**
Sony II	**81.2**	**78.4**	**81.1**	**81.4**	**79.2**	**81.1**	**81.1**	**81.6**	**81.2**
Starlight Curves	88.7	**90.7**	54.9	85.2	84.1	46.7	63.9	88.3	-
Symbols	**73.2**	**73.9**	71	**74.4**	59.9	58.3	55.7	70	-
Trace	**79.6**	**78.7**	73.9	74.9	76	63.4	68.2	**79.6**	**80.2**
Two Lead ECG	92	70.4	**95.8**	**95.8**	64.8	**96**	**95.9**	**95.8**	**95.1**
Word Synonyms	**58.6**	57.9	20.9	54.7	40.5	5.3	7.1	**59.2**	56.8
Yoga	78.9	**81.8**	58.9	66	56.4	58.9	59	79.7	-

Table 3.5: Average run-time, in seconds, for the small sample size dimensionality reduction methods.

Dataset	LDG	PCA	PCA FDA	Reg. FDA	DLDA	GDA	PCA LFDA	Reg. LFDA	ITML
Beef	3.5	0.1	< 0.1	1.3	0.4	0.4	0.5	1.3	-
CBF	0.2	< 0.1	< 0.1	0.5	< 0.1	< 0.1	0.1	0.5	68.2
Cinc ECG Torso	117.3	1.6	0.3	13.3	8.5	8.8	13.7	13.3	-
Coffee	1.1	0.1	< 0.1	0.9	0.1	0.1	0.2	0.9	346.5
Diatom	1.8	< 0.1	< 0.1	1.1	0.2	0.1	0.3	1.1	-
ECG Five Days	0.3	< 0.1	< 0.1	0.8	< 0.1	< 0.1	0.1	0.8	45.2
Face Four	1.8	0.1	< 0.1	1.2	0.2	0.1	0.3	1.2	-
Fish	3.9	0.3	0.1	1.6	0.4	1.8	1.4	1.6	-
Gun Point	0.2	< 0.1	< 0.1	0.5	< 0.1	< 0.1	0.2	0.5	38.6
Haptics	31.0	0.6	0.2	6.2	3.0	11.9	4.6	6.2	-
Inline Skate	159.7	4.4	0.9	18.9	12.4	26.9	24.5	18.9	-
Lightning 2	7.7	0.3	0.1	2.1	0.7	1.4	1.2	2.1	-
Lightning 7	1.4	0.1	< 0.1	1.1	0.1	0.2	0.5	1.1	-
Mallat	27.8	0.7	0.1	5.5	2.7	3.6	3.8	5.5	-
Mote Strain	0.2	< 0.1	< 0.1	0.6	< 0.1	< 0.1	0.1	0.6	8.6
Olive Oil	6.1	0.1	< 0.1	1.9	0.7	0.6	0.7	1.9	-
OSU Leaf	2.0	0.2	0.1	0.9	0.2	1.2	1.0	0.9	-
Sony	0.2	< 0.1	< 0.1	0.6	< 0.1	< 0.1	0.1	0.6	7.3
Sony II	0.2	< 0.1	< 0.1	0.6	< 0.1	< 0.1	< 0.1	0.6	5.6
Starlight Curves	44.8	4.6	11.2	6.0	2.6	38.8	13.5	6.0	-
Symbols	2.4	0.1	< 0.1	1.3	0.3	0.2	0.4	1.3	-
Trace	0.7	0.1	< 0.1	0.7	0.1	0.1	0.5	0.7	1637.0
Two Lead ECG	0.2	< 0.1	< 0.1	0.7	< 0.1	< 0.1	0.1	0.7	24.0
Word Synonyms	1.8	0.2	0.1	1.5	0.1	0.3	1.2	1.5	3358.2
Yoga	3.3	0.4	0.2	1.2	0.2	1.6	1.4	1.2	-

decomposition make it computationally efficient. LDG is also able to find reduced spaces where a nearest-neighbor classifier can achieve high accuracy in comparison to other dimensionality reduction algorithms. Experiments have also shown that LDG is able to achieve good performance on the small sample size dimensionality reduction problem.

Chapter 4

TRANSFER LOCAL DISCRIMINATIVE GAUSSIAN DIMENSIONALITY REDUCTION

This chapter focuses on the problem of *transfer learning*. In transfer learning, the objective is to classify test data drawn from some *target domain* distribution of feature vectors and class labels where either no or only a few labeled examples are available for training. However, a large number of training examples are provided from a *source domain* that differs from the target domain, but is thought to be useful for learning. The goal of transfer learning is to use the large set of source domain training data along with the much smaller set of target domain data to classify the test data drawn from the target domain.

As a visual example of this problem, Figure 4.1 shows handwritten digit images from two different image databases. It is clear from the figure that the digits have been processed in different ways. In the experiments, I will show that transfer learning techniques can boost the performance of a classifier that trains using images from one of these domains to classify images drawn from the other.

In this work, I assume that the learner is given a few labeled examples from the target domain and a much larger number of training examples from the source domain. The source domain dataset, $\mathcal{S} = \{(x_i, g_i)\}_{i=n_t+1}^{n_s+n_t}$, is drawn iid from some unknown joint distribution of feature vectors and class labels $p_{\mathcal{S}}(x, g)$. The smaller target domain dataset, $\mathcal{T} = \{(x_i, g_i)\}_{i=1}^{n_t}$, is drawn iid from unknown joint distribution $p_{\mathcal{T}}(x, g) \neq p_{\mathcal{S}}(x, g)$.

An implicit assumption of the supervised dimensionality reduction techniques described in the previous chapter is that the training and test data are drawn iid from the same distribution. Since this is not the case in the transfer learning scenario, these techniques often fail.

Transfer LDG takes into account the difference in source and target data, and adapts the LDG criteria in order to provide an improved mapping for transfer learning. The

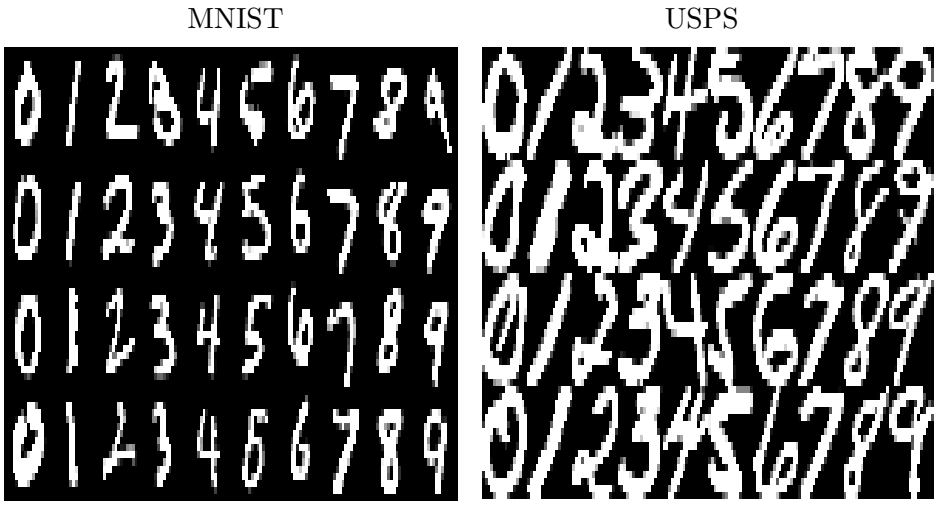

Figure 4.1: The MNIST and USPS handwritten digits datasets vary in how the digits are preprocessed. If we want to classify USPS handwritten digits using training data drawn from the collection of MNIST handwritten digits, then transfer learning techniques are useful in order to reconcile the differences between the two domains.

basic idea of transfer LDG is to find a dimensionality reduction matrix, B, such that the training data in the source domain is made to be similar to that in the target domain: $p_{\mathcal{T}}(B^T x, g) \approx p_{\mathcal{S}}(B^T x, g)$. Since the mapping will be used for classification of data drawn from the target domain, the mapped space should also separate the target domain training data according to class. These two criteria, similarity between source and target domain and separation in the target domain, will guide the approach.

The chapter is organized as follows. Section 4.1 reviews related work in transfer learning. In Section 4.2, I present the transfer LDG objective function. Sections 4.3 and 4.4 provide an asymptotic analysis and an illustrative experiment with simulated data showing the behavior of transfer LDG. Section 4.5 shows the performance of transfer LDG on real data problems, and Section 4.6 combines transfer LDG with another transfer technique called invariant features in order to achieve even better performance for handwritten digit classification. Some of the material in this chapter was conducted jointly with Maya Gupta and has been previously published [45].

4.1 Related Work for Transfer Learning

This section reviews several other approaches to dimensionality reduction for transfer learning. Since dimensionality reduction is a feature preprocessing step prior to classification, I also review another feature preprocessing method for transfer learning: invariant features. For a detailed overview of the different assumptions made for transfer learning, and the ways of adapting classifiers to work for transfer learning see [42, 6].

Bregman divergence based transfer learning [60] is a dimensionality reduction technique for transfer learning that does not use label information in the target domain data. The idea behind this approach is to perform standard supervised learning using the source domain data, but to regularize the objective function by minimizing the Bregman divergence between the source and target domain feature distributions. Let $f(B)$ be the objective function of the source domain dimensionality reduction technique (for instance FDA or PCA), and let $D_B(p_{\mathcal{S}}(x), p_{\mathcal{T}}(x))$ be the Bregman divergence between the source and target domain feature distributions after dimensionality reduction using matrix B. The objective function for Bregman divergence based dimensionality reduction is

$$B^* = \arg\min_{B \in \mathbb{R}^{d \times l}} f(B) + \lambda D_B(p_{\mathcal{S}}(x), p_{\mathcal{T}}(x)).$$

The parameter λ trades off between the source domain objective function, which tries to make the mapping discriminative in the source domain, and the Bregman divergence based criteria, which attempts to make the distributions similar after the mapping. Since the source and target domain distributions are unknown in practice, the authors estimate them using kernel density estimation with the source and target domain training data. The authors minimize the objective with gradient descent.

Another transfer dimensionality reduction method that does not use labels in the target domain is transfer component analysis (TCA) [41]. TCA follows a similar approach to the Bregman divergence based technique in that it uses a divergence measure between the distribution of the source and target domain feature vectors to regularize the objective function. The authors use the *minimum mean discrepancy* criteria to measure the divergence

between the two domains. They also incorporate two other components into their objective function: label dependence and locality preservation. The label dependence objective attempts to separate the data according to class in the source domain. The locality preserving objective attempts to preserve the local geometry of the data by forcing data points that are near each other prior to the mapping to still be near after the mapping. The authors propose a kernelized version of this approach that finds nonlinear mappings of the feature vectors. Furthermore, they show that the mapping can be found by an eigendecomposition.

ITML has also been adapted to the transfer learning scenario [54]. Let $d_M(x_i, x_j)^2 = (x_i - x_j)^T M(x_i - x_j)$ be the squared Mahalanobis distance between x_i and x_j. The original ITML objective is

$$\begin{aligned} M^* = \underset{M \succeq 0}{\arg\min}\ & \mathrm{Tr}(M) - \log\det(M) \\ \text{subject to } & d_M(x_i, x_j)^2 \leq u, \text{ if } g_i = g_j \\ & d_M(x_i, x_j)^2 \geq v, \text{ if } g_i \neq\ g_j. \end{aligned} \tag{4.1}$$

For transfer metric learning, the authors propose to use objective function (4.1), but generate constraints only between examples from different domains, ie. $x_i \in \mathcal{T}\ \forall i$ and $x_j \in \mathcal{S}\ \forall j$. In this way, they find an M that makes distances between examples *across* the two domains small for same-class data and large for different-class data. The Mahalanobis metric M can again be converted into a dimensionality reduction metric by using MRMR feature selection on the resulting features.

The first two dimensionality reduction methods described above try to find a mapping that reduces the divergence between the feature vectors in the two domains. Another preprocessing step for transfer learning problems that also attempts to alleviate the dissimilarity between the source and target distributions is the method of *invariant features*. The basic idea behind invariant features is to extract features from the data that do not vary between the two domains, and they are typically derived by assuming some physical relationship between the source and target domains. For this reason, invariant features are task specific. For instance, Simard et al. derived tangent distance features that are invariant to common

transformations which occur in handwritten character recognition [61]. Bruna and Mallat recently showed that convolutional wavelet networks can produce features that are invariant to deformations and translations [9] [10]. Okopal et al. developed features for acoustic signal classification that are invariant to dispersion effects [40]. Ahonen et al. use local phase quantization features as invariant features to Gaussian blurring in a face recognition system [1]. Invariant features can also be used in conjunction with other methods for overcoming the domain transfer problem, as invariant features are typically not completely robust to all domain transfer effects [39]. In the experiments section I will show that combing invariant features, which use a physical model to pre-process data for transfer learning, with transfer LDG, which is a data-driven approach, can provide increased performance for transfer learning problems.

4.2 Transfer LDG Objective

As mentioned earlier, the two goals for transfer LDG dimensionality reduction are: 1) find a dimensionality reduction matrix B that separates the target domain training data according to class, and 2) find a mapping that makes the source and target domain distributions similar after the mapping, ie. $p_{\mathcal{T}}(B^T x, y) \approx p_{\mathcal{S}}(B^T x, y)$. In Figure 4.2, two examples are shown of one-dimensional spaces that have been mapped from some higher-dimensional space (which is not shown). The left plot is a mapping in which the target domain data are separated according to class, but the source and target domain distributions are not similar. In the right plot, the target domain data is separated, and additionally, the source domain data distribution is similar to that of the target domain data. Therefore, both the source and target domain data can be used to train a classifier for the test data using the right-side mapping.

I make three changes to the discriminative Gaussian objective function (3.2) derived in the previous chapter in order to adapt it to the transfer learning scenario. First, the objective function is weighted over the source and target domain data separately:

$$f(B) = (1-\alpha)\sum_{i=1}^{n_t} \log\left(\frac{\sum_{j=1}^{G} p(B^T x_i|j)p_j}{p(B^T x_i|g_i)p_{g_i}}\right) + \alpha \sum_{i=n_t+1}^{n_t+n_s} \log\left(\frac{\sum_{j=1}^{G} p(B^T x_i|j)p_j}{p(B^T x_i|g_i)p_{g_i}}\right). \quad (4.2)$$

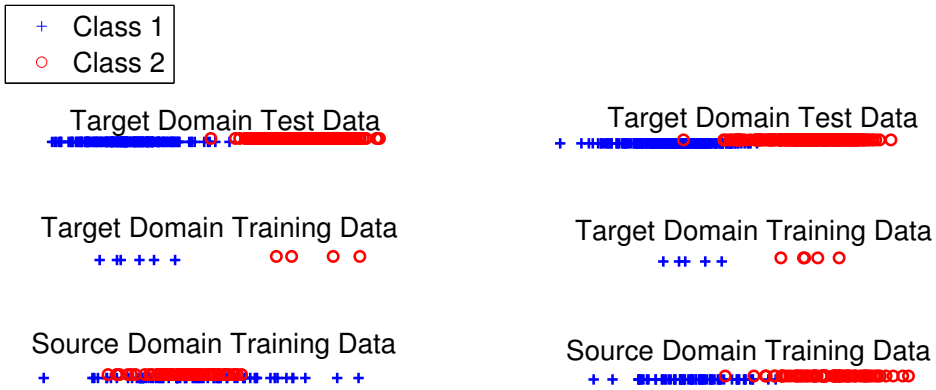

Figure 4.2: Two examples of one-dimensional mappings for transfer dimensionality reduction. In each example, the target domain is separated, but in the right example the target domain data matches the source domain data, making transfer learning more effective.

Second, the parameters of the Gaussian distribution for *target domain* point x_i are estimated using the k nearest *source domain* training examples. Third, since there are few target domain examples, the class priors are learned using the empirical distribution of the source domain data only.

The first term in (4.2) is the primary transfer term. The denominator in the first term of (4.2) finds a B that maximizes the likelihood of the target domain data for a Gaussian distribution trained using the local same-class source domain data, thus finding a B that brings the same-class source and target domain data close together. Conversely, the numerator in the first term of (4.2) seeks a B that minimizes the likelihood of the different-class source Gaussians, thus ensuring that the mapping is still discriminative.

The second term in (4.2) is the standard LDG objective function using only the source domain data. Therefore, if $\alpha = 0.5$, objective function (4.2) is similar to the standard LDG objective function acting on the pooled data with the difference being that the conditional distributions for the target domain data and source domain data are trained using only the source domain data (as opposed to the pooled data). The second term is included in the objective function because if the source and target domain data are similar, then it is beneficial to train the objective function using as much data as possible.

An alternative to the above approach would be to learn the class-conditional distributions from the pooled source and target data as opposed to the source domain data only. However, I choose not to do this because if the source and target domain distributions are different, then mixing data from the two different domains in order to learn the parameters may cause the learned parameters to be innacurate for each domain. On the other hand, if the source and target domains are similar, then the parameters learned from the pooled data would be similar to those learned using only one dataset or the other. Since there is more data in the source domain, then it makes sense to learn the parameters from the domain with the most data.

I make the same assumptions and approximations as in the previous chapter in order to find an approximate solution to (4.2) that requires only a maximal eigenvalue decomposition. In order to do this, I impose the constraint that $B^TB = I$, I assume that the covariance matrices of the local Gaussians are identity matrices, I then use Jensen's inequality to lower bound (4.2), and I introduce the γ parameter. Let $p(B^Tx_i|j) = \mathcal{N}(B^Tx_i; B^T\mu_{i,j}, I)$ be the local Gaussian for training point x_i, and let $\Delta_{i,j} = x_i - \mu_{i,j}$. Applying the approximations listed above yields the transfer LDG optimization problem

$$
\begin{aligned}
B^*_\gamma = \underset{B\in\mathbb{R}^{d\times l}}{\arg\min}\ (1-\alpha)\sum_{i=1}^{n_t}\left(\frac{1-p_{g_i}}{2}\Delta_{i,g_i}^TBB^T\Delta_{i,g_i} - \gamma\sum_{j=1,j\neq g_i}^{G}\left(\frac{p_j}{2}\Delta_{i,j}^TBB^T\Delta_{i,j}\right)\right) \\
+\alpha\sum_{i=n_t+1}^{n_t+n_s}\left(\frac{1-p_{g_i}}{2}\Delta_{i,g_i}^TBB^T\Delta_{i,g_i} - \gamma\sum_{j=1,j\neq g_i}^{G}\left(\frac{p_j}{2}\Delta_{i,j}^TBB^T\Delta_{i,j}\right)\right) \\
\text{subject to } B^TB = I.
\end{aligned}
\tag{4.3}
$$

As shown in the previous chapter, the B^*_γ that minimizes (4.4) is found by an eigendecomposition.

4.3 Asymptotic Analysis

In this section, I present an asymptotic analysis of the transfer LDG solution for B^*_γ in (4.3) as the number of target domain training examples $n_t \to \infty$ while also assuming infinite source domain training examples. The notation for this section is included in Table

Table 4.1: Notation for the asymptotic analysis of transfer LDG.

Symbol	Definition
$\mu_j^{(\mathcal{S})}$	Class j source domain mean
$\mu_j^{(\mathcal{T})}$	Class j target domain mean
$\Sigma_j^{(\mathcal{T})}$	Class j target domain covariance matrix
$p_j^{(\mathcal{S})}$	Class j source domain prior
$p_j^{(\mathcal{T})}$	Class j source domain prior
n_j	Number of class j target domain training points

4.1. Although in practice we expect to have few target domain examples, the asymptotic analysis will still provide insight on the behavior of the transfer LDG function, particularly with respect to the γ parameter. Since the second term of the LDG objective function has already been analyzed in the previous chapter, I perform analysis only for the first term by setting $\alpha = 0$.

As in the previous chapter, I again simplify the analysis by assuming that I don't perform *local* transfer LDG. Instead, I perform *global* transfer LDG where I model $p(x_i|j) = \mathcal{N}(x_i; \hat{\mu}_j^{(\mathcal{S})}, I)$, i.e. there is a single Gaussian per class. Since the number of source examples is assumed to be infinte, I will also assume that $\hat{\mu}_j^{(\mathcal{S})} = \mu_j^{(\mathcal{S})}$ and that $\hat{p}_j = p_j^{(\mathcal{S})}$.

I begin by rearranging the terms in (4.3), and dividing by n_t, which does not effect the minimization. Therefore, the function to be minimized in (4.3) can be written as

$$\frac{1}{n_t}\sum_{q=1}^{G}\sum_{i:g_i=q}\left((x_i-\mu_q^{(\mathcal{S})})^T BB^T(x_i-\mu_q^{(\mathcal{S})}) - \gamma\sum_{j=1}^{G}\left(p_j^{(\mathcal{S})}(x_i-\mu_j^{(\mathcal{S})})^T BB^T(x_i-\mu_j^{(\mathcal{S})})\right)\right). \tag{4.4}$$

Equation (4.4) can be rewritten as

$$\mathrm{Tr}(B^T ZB), \tag{4.5}$$

where

$$Z = \sum_{q=1}^{G} \left(\frac{1}{n_t} \sum_{i:g_i=q} (x_i - \mu_q^{(\mathcal{S})})(x_i - \mu_q^{(\mathcal{S})})^T - \frac{\gamma}{n_t} \sum_{i:g_i=q} \sum_{j=1}^{G} p_j^{(\mathcal{S})} (x_i - \mu_j^{(\mathcal{S})})(x_i - \mu_j^{(\mathcal{S})})^T \right). \tag{4.6}$$

I begin by analyzing the first term inside the parenthesis in (4.6),

$$\frac{1}{n_t} \sum_{i:g_i=q} (x_i - \mu_q^{(\mathcal{S})})(x_i - \mu_q^{(\mathcal{S})})^T, \tag{4.7}$$

as $n_t \to \infty$. This term can be rewritten as

$$\begin{aligned} &\frac{1}{n_t} \sum_{i:g_i=q} x_i x_i^T - x_i \mu_q^{(\mathcal{S})^T} - \mu_q^{(\mathcal{S})} x_i^T + \mu_q^{(\mathcal{S})} \mu_q^{(\mathcal{S})^T} \\ &= \frac{1}{n_t} \sum_{i:g_i=q} x_i x_i^T - \mu_q^{(\mathcal{T})} \mu_q^{(\mathcal{T})^T} + \mu_q^{(\mathcal{T})} \mu_q^{(\mathcal{T})^T} - x_i \mu_q^{(\mathcal{S})^T} - \mu_q^{(\mathcal{S})} x_i^T + \mu_q^{(\mathcal{S})} \mu_q^{(\mathcal{S})^T}, \end{aligned} \tag{4.8}$$

where $\mu_q^{(\mathcal{T})} \mu_q^{(\mathcal{T})^T}$ has been added and subtracted in order to go from the first line to the second line. Asymptotically, $\sum_{i:g_i=q} x_i x_i^T - \mu_q^{(\mathcal{T})} \mu_q^{(\mathcal{T})^T}$ is equal to a scaled version of the class q target domain covariance matrix: $n_q \Sigma_q^{(\mathcal{T})}$. Furthermore, $\sum_{i:g_i=l} x_i \mu_q^{(\mathcal{S})^T} \to n_q \mu_q^{(\mathcal{T})} \mu_q^{(\mathcal{S})^T}$. Substiting these asymptotic solutions into (4.8), and also noting that $\frac{n_q}{n_t} \to p_q^{(\mathcal{T})}$, results in (4.7) being asymptotically equal to

$$p_q^{(\mathcal{T})} \left(\Sigma_q^{(\mathcal{T})} + (\mu_q^{(\mathcal{T})} - \mu_q^{(\mathcal{S})})(\mu_q^{(\mathcal{T})} - \mu_q^{(\mathcal{S})})^T \right). \tag{4.9}$$

I now analyze the second term inside parenthesis in (4.6) for a single class q:

$$\frac{\gamma}{n_t} \sum_{i:g_i=q} \sum_{j=1}^{G} p_j^{(\mathcal{S})} (x_i - \mu_j^{(\mathcal{S})})(x_i - \mu_j^{(\mathcal{S})})^T. \tag{4.10}$$

Equation (4.10) can be rewritten as

$$\frac{1}{n_t} \gamma \sum_{j=1}^{G} p_j^{(\mathcal{S})} \sum_{i:g_i=q} x_i x_i^T - \mu_j^{(\mathcal{S})} x_i^T - x_i \mu_j^{(\mathcal{S})^T} + \mu_j^{(\mathcal{S})} \mu_j^{(\mathcal{T})^T}. \tag{4.11}$$

Adding and subtracting $\mu_q^{(\mathcal{T})}\mu_q^{(\mathcal{T})^T}$ from (4.11) gives

$$\frac{1}{n_t}\gamma\sum_{j=1}^{G}p_j^{(\mathcal{S})}\sum_{i:g_i=q}x_ix_i^T-\mu_q^{(\mathcal{T})}\mu_q^{(\mathcal{T})^T}+\mu_q^{(\mathcal{T})}\mu_q^{(\mathcal{T})^T}-\mu_j^{(\mathcal{S})}x_i^T-x_i\mu_j^{(\mathcal{T})^T}+\mu_j^{(\mathcal{S})}\mu_j^{(\mathcal{S})^T}. \quad (4.12)$$

As $n_t\to\infty$, the term $\sum_{i:g_i=q}x_ix_i^T-\mu_q^{(\mathcal{T})}\mu_q^{(\mathcal{T})^T}$ in (4.12) again becomes a scaled version of the class q target domain covariance matrix. We can also replace the sums over x_i with the class q target domain means. Therefore, asymptotically (4.12) becomes

$$\begin{aligned}&\frac{n_q}{n_t}\gamma\sum_{j=1}^{G}p_j^{(\mathcal{S})}\left(\Sigma_q^{(\mathcal{T})}+(\mu_q^{(\mathcal{T})}-\mu_j^{(\mathcal{S})})(\mu_q^{(\mathcal{T})}-\mu_j^{(\mathcal{S})})^T\right)\\&=\gamma p_q^{(\mathcal{T})}\left(\Sigma_q^{(\mathcal{T})}+\sum_{j=1}^{G}p_j^{(\mathcal{S})}(\mu_q^{(\mathcal{T})}-\mu_j^{(\mathcal{S})})(\mu_q^{(\mathcal{T})}-\mu_j^{(\mathcal{S})})^T\right). \quad (4.13)\end{aligned}$$

Substituting (4.9) and (4.13) into (4.6) gives

$$\begin{aligned}Z=\sum_{q=1}^{G}p_q^{(\mathcal{T})}\Bigg((1-\gamma)\Sigma_q^{(\mathcal{T})}+(1-\gamma p_q^{(\mathcal{S})})(\mu_q^{(\mathcal{T})}-\mu_q^{(\mathcal{S})})(\mu_q^{(\mathcal{T})}-\mu_q^{(\mathcal{S})})^T \quad (4.14)\\-\gamma\sum_{j=1,j\neq q}^{G}p_j^{(\mathcal{S})}(\mu_q^{(\mathcal{T})}-\mu_j^{(\mathcal{S})})\mu_q^{(\mathcal{T})}-\mu_j^{(\mathcal{S})})^T\Bigg),\end{aligned}$$

Equation (4.14) reveals that Z is composed of three parts for each class q. The first part is the class q target domain covariance matrix, the second part is the distance vector between the class q mean in the source and target domains, and the third part is a summation over the distance vectors between the class q mean in the target domain and the different-class means in the source domain. These three components seek a B that does three different things: 1) bring the same-class target domain data close together by making the covariance in the mapped space small, 2) bring the same-class source and target domain data together by making the distance vector between their means small, and 3) separate target domain data from different-class source domain data by making the distance vector between their means large. The weightings between these different components are determined by the

priors in the two domains and the γ parameter. Large γ emphasizes the third component over the first two, and vice-versa for small γ.

The components of (4.14) can be mapped back to the original two objectives for transfer LDG that were described at the beginning of Section 4.2. The second transfer LDG objective, making the domains similar after the transformation, is achieved via the second component of (4.14) that brings the same-class source and target domain data together. The first objective for transfer LDG, that the target domain data is separated according to class, is achieved indirectly by combining the second and third components of (4.14). If the source and target same-class data are similar (component 2) and the different-class source and target domain data are separated (component 3), then the target domain data will also be separated according to class.

4.4 Illustrative Example

In this section I perform transfer LDG on a simulated dataset in order to verify and visualize the asymptotic analysis of the previous section. I generate two-class data in two different domains that are not matched. The class one and class two source domain feature vectors are Gaussian distributed with mean vectors [-1, -2]^T and [1, 2]^T, respectively. The class one and class two target domain feature vectors are Gaussian distributed with mean vectors [-1.2, 2]^T and [0.5, 2]^T, respectively. The diagonal covariance matrix is the same for the class one and class two data from both domains and has diagonal [0.2, 0.3]^T. Sixty training examples are provided in each domain (thirty from each class).

Figure (4.3) shows a scatter plot of the data for the simulation. The lines in the figure show the one-dimensional spaces onto which the data will be mapped using transfer LDG with $\gamma = 0$, $\gamma = 1$, and $\gamma = 0.5$. The figure also shows the one-dimensional scatter plots of the data after mapping. When $\gamma = 1$, transfer LDG prioritizes making the target domain training data far from different-class source domain training data. When $\gamma = 0$, transfer LDG prioritizes the first and second components of (4.14) which are to make the scatter of the target domain data as small as possible while at the same time bringing the same-class source and target domain data close together. Transfer LDG balances these objectives when $\gamma = 0.5$.

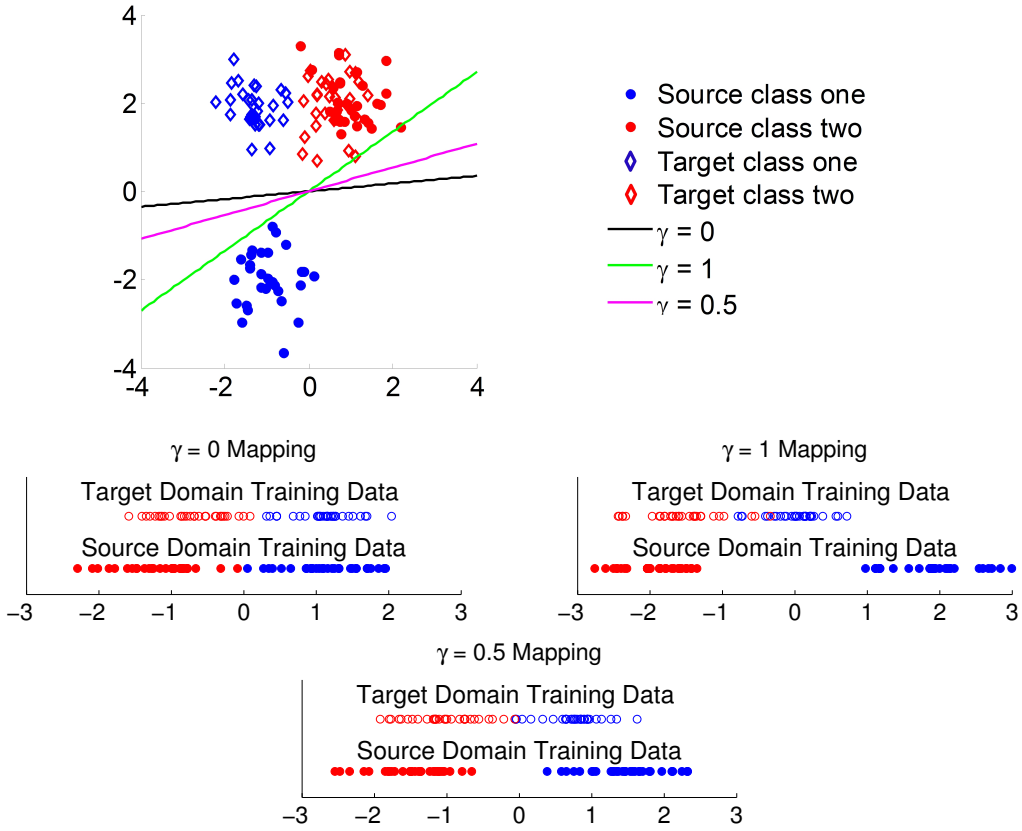

Figure 4.3: The top figure shows a scatter plot of the training data and the one dimensional subspace that the training data are mapped to for three choices of γ. The bottom three figures show the one-dimensional scatter of the data after three different mappings.

This simulation demonstrates the importance of including the γ parameter in the transfer LDG objective function. In practice, the γ parameter will again need to be cross-validated in order to find the best value.

4.5 Transfer Learning Experiments

In this section I present real data experiments using two different transfer learning datasets. The first dataset is an image classification dataset containing product images drawn from three different domains. The second dataset is the handwritten digits classification problem

described in the introduction with data drawn from two different domains. I describe the product images dataset first, and then describe the handwritten digits dataset.

4.5.1 Dataset Descriptions

The product images dataset contains images residing in three different domains: Amazon product images, images taken with a high-resolution DSLR camera, and images taken with a low-resolution webcam [54]. An example image of a backpack from each of these domains is shown in Figure 4.4. The figure shows that the Amazon domain significantly differs from the other two domains. The Amazon images are formatted and processed differently. Also, images of the same items appear in the DSLR and Webcam databases (for instance, the backpack pictured in Figure 4.4 is the same for these two domains) making these two domains somewhat similar. This dataset was first used by Saenko, et al., and we use the same preprocessing techniques as described in the paper [54] to extract features from the images. We describe the feature extraction process below.

Product Image Features: All images are resized to be three hundred by three hundred pixels and are converted to grayscale. The speeded-up robust features (SURF) [4] algorithm is then used to determine interest points in each image, and a sixty-four dimensional SURF descriptor is extracted for each interest point. K-means clustering is then performed using all the descriptors from a randomly chosen subset consisting of ten percent of Amazon training data. The k-means clustering provides eight hundred cluster centers, and these eight hundred cluster centers make up a code book. Each training image's set of SURF descriptors is converted to a histogram over this code book, and the resulting histogram is the final eight-hundred dimensional feature vector.

The handwritten digits dataset consists of handwritten digits coming from two different widely used datasets in the machine learning literature: MNIST and USPS both shown in Figure (4.1). I don't use all of the data from either of these datasets. Instead, I randomly sampled three hundred MNIST digits per class to be the MNIST domain dataset, and three hundred USPS digits per class to be the USPS domain dataset. This was done to speed up the run-time in the experiments, particularly for ITML. The only preprocessing performed

Figure 4.4: Examples of images taken from the Amazon, DSLR, and Webcam domains.

was resizing of the MNIST images to sixteen by sixteen pixels to match the size of the USPS digits. The feature vector for each image is the vectorized pixel intesities.

4.5.2 Dimensionality Reduction Methods

The first dimensionality reduction method is transfer LDG. For transfer LDG, I cross-validate to choose α from $[0,\ 0.1,\ 0.3,\ 0.5]$ by k-NN cross-validation at dimensionality equal to the number of classes plus five. I cross-validate α independently of γ prior to choosing γ by simply setting $\gamma = 0.9$. After choosing a value for α, I then cross-validate to choose γ from $[0.2,\ 0.4,\ 0.6,\ 0.8,\ 0.9]$.

I compare to three different dimensionality reduction approaches. First, I compare to to pooled PCA and pooled FDA dimensionality reduction. These approaches ignore the difference between the domains by pooling the training data from each domain and then performing standard PCA or FDA. I also compare to ITML for transfer learning as described in [54] with MRMR feature selection. Finally, I show the result of ITML metric learning with no dimensionality reduction.

4.5.3 Experimental Results

The source domain training data consists of all the available data in that particular domain, and I standard normalize so that it has mean zero and standard deviation one. The target

domain training data consists of exactly two examples per class and is standard normalized independently of the source domain training data. I remove any features that exhibit zero variance in the source or target domain. Figure 4.5 plots the accuracy averaged over ten random splits of the target domain test and training data.

Figure 4.5 shows that LDG is statistically the best or tied for the best in almost all the plots. The results also show that using the Webcam data to classify in the DSLR domain, and vice versa, provides the highest accuracy of all the product images domain transfer problems. This is due to the fact that the domains are actually quite similar, as can be seen from Figure (4.4). In the experiments that transfer between these two datasets, transfer LDG gives nonzero weight to the α parameter for every random split of the training and test data in order to leverage the source data. In all other transfer experiments, transfer LDG chooses $\alpha = 0$ due to the degree of dissimilarity between the domains.

4.6 Combining Transfer LDG and Invariant Features

As mentioned in the introduction, invariant feature transformations can aid in transfer learning. Bruna and Mallat showed that scattering transformation networks can produce features that are locally invariant to translation and that linearize deformations [9]. The scattering network is a bank of localized wavelet filters. At each layer of the network, the image is filtered and the average of the modulus is taken as a feature. This averaging operations removes the high frequency information. Therefore, at the next layer, the modulus coefficients of the previous layer are again filtered with another bank of localized wavelet filters and the resulting modulus is averaged for another feature.

Bruna and Mallat proposed using these invariant features for classification of the MNIST and USPS handwritten digits. However, they did not consider the problem that we consider here of using one of these databases to classify handwritten digits from the other database. Therefore, in this section, I perform the same handwritten digit classification experiment as performed in the previous section. However, instead of using the raw pixel values as the features for classification, I instead use the invariant features given by the scattering transformation network.

I compute the invariant features using code provided on the authors' website. I use the same parameters as the authors used in the paper [9]. This results in 2032 features per image. Since the feature size is large, I do not compare to ITML in this section, as the run-time for ITML becomes excessive if the feature dimensionality is large, as demonstrated in the previous chapter.

Figure 4.6 plots the results when the classifier is given all of the training data in one domain as the source data and exactly two images in the target domain. The results show that we can greatly increase performance by using the invariant features. Comparing the results of Figure (4.5) to Figure (4.6), the performance of transfer LDG and PCA dimensionality reduction is greatly boosted. FDA exhibits increased performance as well on the USPS to MNIST dataset but not on the MNIST to USPS dataset. Transfer LDG again achieves the best performance of the dimensionality reduction methods considered.

4.7 Conclusions

In this chapter, I have shown that LDG can be adapted to the transfer learning setting in order to find a reduced space where source data drawn from one domain can be used to classify target domain data drawn from a different domain. The two goals for transfer LDG were to find a reduced space where the target domain data were separated according to class, and where the source and target domain distributions were made to be similar. The asymptotic analysis showed that transfer LDG achieved these goals.

Experimental results showed that transfer LDG can perform well on real data. Furthermore, by combining transfer LDG with invariant features, excellent performance could be achieved when transferring between datasets of handwritten digits.

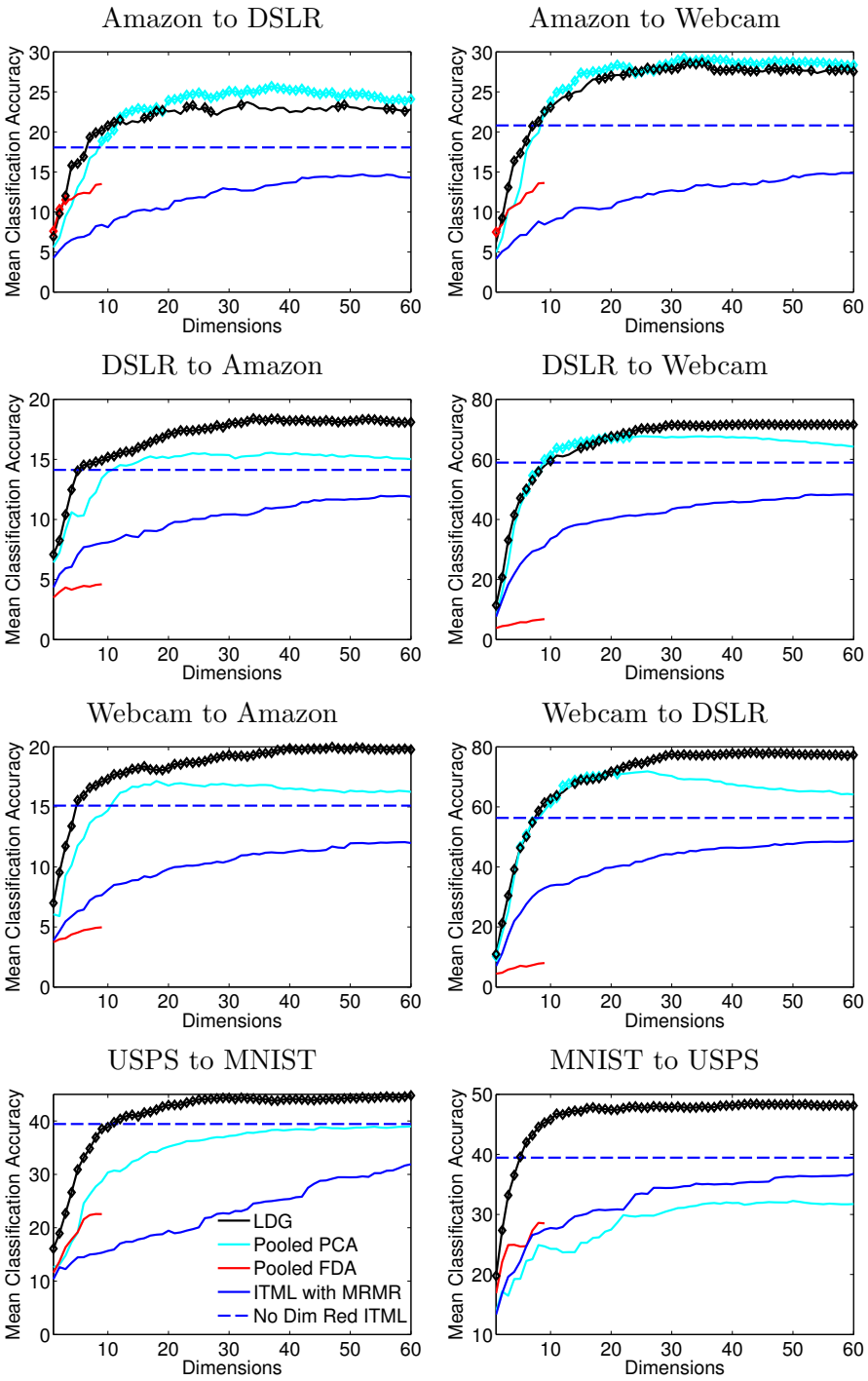

Figure 4.5: Transfer results when exactly two target domain training examples are provided for each class. Diamonds indicate that the method was statistically the best or not statistically different from the best with 95% confidence for that dimensionality.

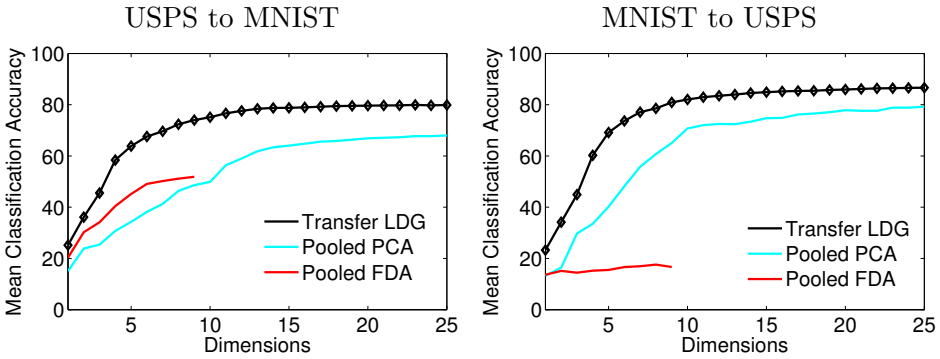

Figure 4.6: Transfer dimensionality reduction experiments for the handwritten digits datasets using invariant features when exactly two target domain images are chosen per class. Diamonds indicate that the method was statistically the best or not statistically different from the best with 95% confidence for that dimensionality.

Chapter 5

CLASSIFYING INCOMPLETE DATA

The focus of this chapter is incomplete test data classification. Incomplete test data classification is useful when one wishes to make a class decision without collecting all of the test data or before all of the test data is available. For example, in medical applications, it is beneficial to make a decision after subjecting a patient to as few tests as possible. When working with time-series data, it is beneficial to make time-sensitive decisions as early as possible. At the same time, though, one should wait until enough data is available to make a good decision. As a concrete example, suppose that a voice recognition system has been trained on three second audio signals of people saying "Hello World." Can it sometimes recognize a person as soon as it hears "Hello," or must it always wait for the complete signal to be sufficiently certain?

The focus of this chapter is the design of a classifier that classifies incomplete test data only if it can guarantee that the decision made using the incomplete data has a high probability of matching the decision that would be made given the complete test data. The approach also makes it possible to answer the related question, "If the current incomplete data is classified, what is the probability that the class decision will be the same as classifying from the complete data?"

Although this chapter does not specifically focus on dimensionality reduction, I will show that LDG dimensionality reduction as a preprocessing step for incomplete test data classification can provide significant computational savings at test time as well as increased accuracy in some cases. This chapter, therefore, shows the usefulness of LDG dimensionality reduction beyond simply being preprocessing step in basic classification scenarios, but also in more complex systems.

The material presented in this chapter was joint work with Hyrum Anderson, Maya Gupta, Kristi Tsukida, and Dun Yu Hsaio and has been published previously or is currently under review [2, 44].

The chapter is organized as follows. First, I propose optimal and practical decision rules for classifying incomplete data. In Sections 5.2, 5.3, and 5.4, I provide the details on how to efficiently and accurately implement the proposed practical decision rule for classifiers that use linear or quadratic discriminants, such as linear support vector machines and linear or quadratic discriminant analysis (LDA and QDA). Section 5.5 reviews related work on classifying with missing features and related work on early classification of time-series data. Experiments in Section 5.6 show that the proposed incomplete decision rule consistently provides enhanced reliability over the state-of-the-art in deciding when to classify incomplete data.

5.1 Incomplete Decision Rules

Let $\hat{g}(x)$ be a classifier function that assigns a class label to test sample x. However, suppose that at test time x is unavailable, and instead only some incomplete information is given as a vector z.

I wish to classify z if it provides enough information about x to make a good decision, otherwise, the decision is delayed until more information is available. Typically a good decision in the classification setting is measured by the accuracy on the test data. The incomplete decision rules in this chapter will instead consider the question: "Can z be classified knowing that some minimum probability threshold of making the same decision that would be made on x is met?" I define the objective of this question as *reliability*: the probability that the class label assigned to z matches that assigned to x. Therefore, the incomplete decision rules make a good decision by matching the class label of $\hat{g}(x)$, which is assumed to be trained for the goal of test accuracy.

To estimate reliability, the classification features derived from the complete data are modeled as a random variable X that is jointly distributed with Z — a random variable modeling the incomplete data. Given a desired reliability $\tau \in [0, 1]$ and incomplete infor-

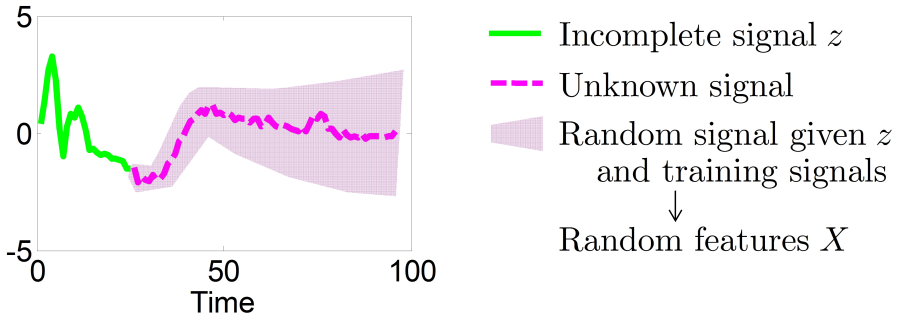

Figure 5.1: The figure illustrates incomplete time-series classification. The incomplete time signal is shown in green. The complete signal is treated as random, and its distribution is estimated assuming that it is iid with the training signals. From the pdf of the complete data we can estimate the pdf of the feature data $p(x|z)$, and, using this distribution, we can check whether or not one can make a reliable decision.

mation z, an ideal incomplete decision rule is to classify as class g if

$$P(X \text{ s.t. } \hat{g}(X) = g | Z = z) = \int_{x \text{ s.t. } \hat{g}(x)=g} p(x|z)\, dx \geq \tau, \tag{5.1}$$

and otherwise to wait for more information. Figure 5.2 illustrates this rule.

The ideal rule (5.1) could be checked in several ways. A straightforward approach would be to compute the integral directly and see if it is greater than or equal to τ. An alternative approach is to consider all sets A in the domain of X such that $P(X \in A | Z = z) \geq \tau$, and see if $\hat{g}(x)$ maps all x in one such set to a single class g. In general, I expect these approaches to be computationally intractable.

A more conservative, but computable, incomplete decision rule is to classify as class g if

$$\hat{g}(x) = g \text{ for all } x \in A \text{ for some set } A \text{ such that } P(X \in A | Z = z) \geq \tau. \tag{5.2}$$

Rule (5.2) differs from (5.1) in that only one set A that contains at least τ measure of X must be checked. This rule is more conservative than (5.1) because it does not check all sets A, and thus (5.1) could be satisfied without (5.2) being satisfied (but not vice-versa), as illustrated in Figure 5.2.

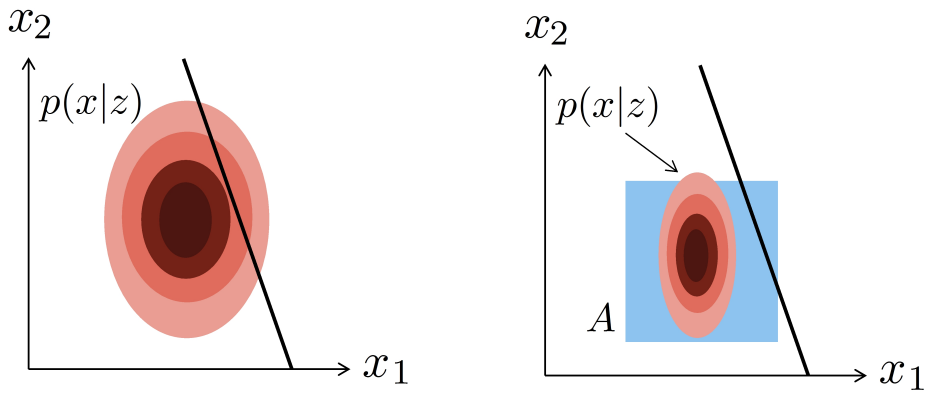

Figure 5.2: **Left:** A two-dimensional feature space and a linear class decision boundary. The mass of X lies mostly to the left of the decision boundary. For values of τ that are smaller than the mass of X that falls to the left of the decision boundary, the ideal incomplete decision rule would choose to classify rather than wait. **Right:** The entire mass of X falls on one side of the decision boundary, and thus the ideal incomplete decision rule would choose to classify rather than wait for any value of τ. On the other hand, the computable incomplete decision rule constructs some set A that captures a fraction τ of the mass of X, then requires that entire set A to lie on one side of the decision boundary. For the choice of A shown here, the set A crosses the decision boundary, and thus the computable decision rule would choose to wait.

Implementing the proposed rule given in (5.2) requires three steps. First, the conditional density $p(x|z)$ must be estimated. Second, the constraint set A must be constructed, and third, it must be checked if the rule is satisfied. The first of these steps that I discuss is the construction of set A in Section 5.2, where I show that only the first and second conditional moments of X need to estimated. Then in Section 5.3, I show how rule (5.2) can be efficiently checked for classifiers that have linear or quadratic class discriminant functions. The discussion of how to estimate the necessary moments of X is delayed until Section 5.4.

5.2 Defining a Set A that Contains Measure τ of X

To implement the incomplete decision rule (5.2), one must be able to construct a set A that contains at least τ measure of X given $Z = z$. In this section, I propose three ways to construct such a set A. Figure 5.3 illustrates these three constructions.

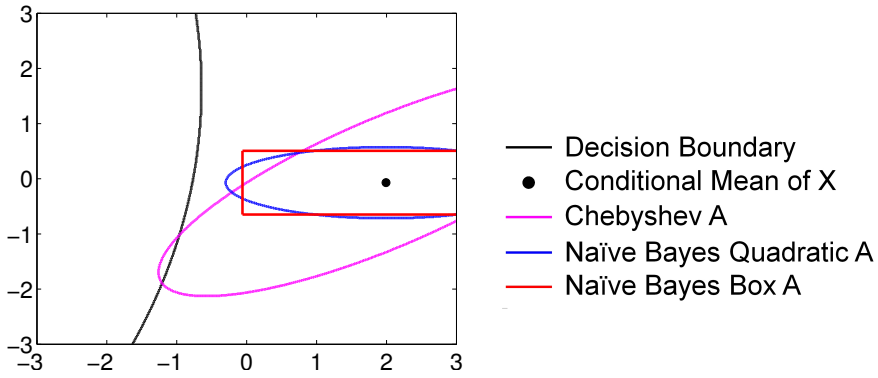

Figure 5.3: The sets A containing mass τ of the conditional p.d.f. of X using the three different construction methods proposed Section 5.2.

5.2.1 Chebyshev Construction for Set A

Suppose that only the first and second conditional moments of X are estimated, and no assumptions are made about the distribution other than that it has finite first and second moments. Then a set A can be constructed using the multidimensional Chebyshev inequality, which states that for $X \in \mathbb{R}^d$ with known mean m and covariance R, and any $\alpha > 0$:

$$P\left((X-m)^T R^{-1}(X-m) \leq \alpha^2\right) \geq 1 - \frac{d}{\alpha^2}.$$

Thus to satisfy $P(X \in A | Z = z) \geq \tau$, define

$$A = \left\{ x \text{ s.t. } (x-m)^T R^{-1}(x-m) \leq \frac{d}{1-\tau} \right\}. \tag{5.3}$$

The set A defined by (5.3) is non-empty for $\tau \in (-\infty, 1]$, although $\tau \leq 0$ does not give a useful bound for the incomplete classifier reliability.

5.2.2 Naive Bayes Constructions for Set A

The Chebyshev construction given in the previous section can be overly conservative, as it makes no assumptions about the conditional distribution of X other than a finite mean and covariance. If more is assumed about the distribution, a smaller constraint set A can be

defined that results in a less conservative decision rule, and therefore earlier classification for the same reliability requirement τ. For example, if it is assumed that the conditional distribution is Gaussian[1], then a quadratic set A that covers τ measure of X is

$$A = \left\{x \text{ s.t. } (x-m)^T R^{-1}(x-m) \leq \text{erf}(\tau)\right\}, \tag{5.4}$$

where one must compute the inverse cdf to determine the value $\text{erf}(\tau)$ to achieve a set A with measure τ. For a multi-dimensional Gaussian, computing the inverse cdf for (5.4) is non-trivial. Equation (5.4) can be simplified by making the conservative näive Bayes assumption that the components of X are independent, and thus R is diagonal. Then the quadratic function in (5.4) becomes $\sum_{\ell=1}^{d} \left(\frac{x(\ell)-m(\ell)}{\sqrt{R(\ell,\ell)}}\right)^2$. Under the independent Gaussian assumption, $\sum_{\ell=1}^{d} \left(\frac{X(\ell)-m(\ell)}{\sqrt{R(\ell,\ell)}}\right)^2$ is a chi-squared random variable with d degrees of freedom; thus, the $\text{erf}(\tau)$ function in (5.4) is easily computed using the inverse cdf of a chi-squared random variable [62]. I call the resulting set the *näive Bayes quadratic A*.

A related option is to force the set A to be a box centered about the mean of X. Again, make the näive Bayes assumption that elements of X are independent, then $p(x|z) = \prod_{\ell=1}^{d} p(x(\ell)|z)$. Then I can define a set

$$A = \{x \text{ s.t. } x(\ell) \in [m(\ell) - s_\tau(\ell), m(\ell) + s_\tau(\ell)] \; \forall \; \ell = 1, ..., d\}, \tag{5.5}$$

where s_τ is a vector defining the width of the box in each dimension such that the total measure of the box is τ. In this thesis, I implement this constraint by assuming that each marginal distribution $X(\ell)$ is Gaussian and by assigning each dimension equal measure $\tau^{1/d}$. I call the resulting set the *näive Bayes box A*.

The näive Bayes quadratic A (5.4) and näive Bayes box A (5.5) result from making the same assumptions about the conditional distribution of X. The sets differ, though, in that (5.4) finds the ellipsoidal footprint of the Gaussian that has measure τ, while (5.5) treats

[1]The Gaussian assumption is often justified by a central limit theorem argument, a maximum entropy argument, or a simplicity argument.

the dimensions completely independently, giving each of the marginals measure $\tau^{1/d}$ (see Figure 5.3).

5.3 Efficient Solutions for Linear or Quadratic Discriminants

In this section, I show that the incomplete data classification rule (5.2) with the constraint sets A proposed in Section 5.2 can be computed efficiently for classifiers of the form

$$\hat{g}(x) = \arg\max_g f_g(x), \tag{5.6}$$

where $f_g(x)$ is a linear or quadratic discriminant function for the g^{th} class, and according to (5.6), the classifier assigns x to the class with the maximum discriminant. For example, the linear support vector machine (SVM) has a linear discriminant, while the quadratic discriminant analysis (QDA) classifier has a quadratic discriminant [28].

Nearest-neighbor classifiers using a Euclidean (or Mahalanobis) distance have a discriminant that over the set $x \in A$ requires taking the minimum of a set of quadratic discriminants:

$$f_g(x) = \min_{x_i : y_i = g} (x - x_i)^T (x - x_i). \tag{5.7}$$

An optimal method for checking the incomplete decision rule (5.2) for this discriminant is an open question. A conservative reliability decision can be made by treating each sample as its own class in (5.6). That is, let $f_i(x) = (x - x_i)^T (x - x_i)$, solve (5.6) for the resulting quadratic discriminant, and then classify as the class y_i. A computationally simpler approach (but one that is not strictly conservative), is to only consider each class's nearest-neighbor to the posterior mean, which produces one quadratic discriminant per class. In the experiments, I do something similar to the latter approach using a local QDA classifier.

I begin with the two-class problem, and then show how this rule can be extended to multi-class problems.

5.3.1 Two-class problems

I first consider a two-class problem, where the set of class labels is $\mathcal{G} = \{1, 2\}$. Let $f_1(x)$ and $f_2(x)$ be the discriminants for classes one and two, and define

$$f(x) = f_2(x) - f_1(x).$$

An equivalent classifier to (5.6) can be defined using only $f(x)$ by noting that $f(x) = 0$ defines the decision boundary between classes 1 and 2. Therefore, classification rule (5.6) is equivalent to

$$\hat{g}(x) = \begin{cases} 1 & \text{if } f(x) \leq 0 \\ 2 & \text{if } f(x) > 0 \end{cases} . \tag{5.8}$$

The proposed incomplete data decision rule (5.2) is implemented:

$$\hat{g}(z) = \begin{cases} 1 & \text{if } \max\limits_{x \in A} f(x) \leq 0 \\ 2 & \text{if } \min\limits_{x \in A} f(x) > 0 \\ \text{no decision} & \text{otherwise.} \end{cases} \tag{5.9}$$

The decision rule in (5.9) classifies the incomplete data if the constraint set A lies completely on one side of the decision boundary or the other. If the set A lies across the decision boundary, then the rule withholds making a decision. Note that the decision rule (5.9) is dependent on the incomplete data through the dependence of A on z. The three different conditions in (5.9) are shown for a quadratic discriminant (and hence quadratic decision boundary) and a quadratic construction of the set A in Figure 5.4.

5.3.2 Linear Discriminants

In order to efficiently check (5.9), one must be able to efficiently compute the maximum and minimum of $f(x)$ over the set $x \in A$. For linear discriminant functions, $f(x)$ is also linear. Then for a quadratic set A, such as the Chebyshev or näive Bayes quadratic sets A given in Section 5.2, finding the maximum and minimum are quadratically constrained

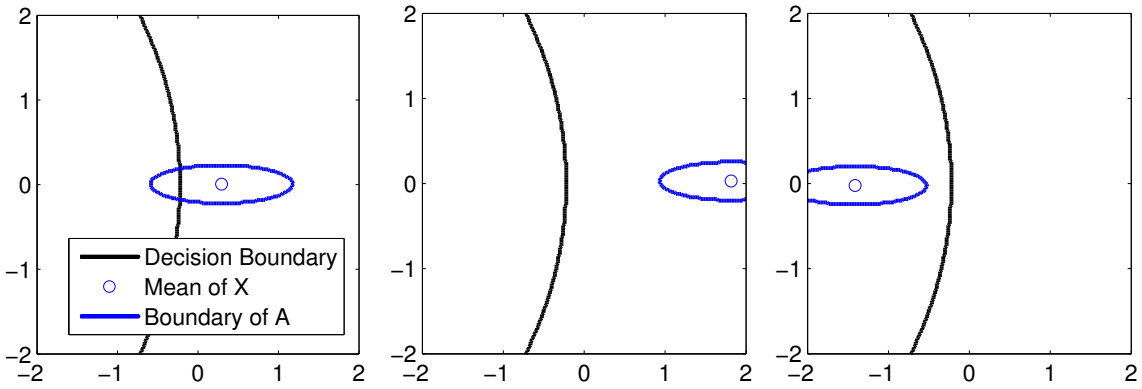

Figure 5.4: Three different scenarios for incomplete data classification. In the leftmost plot, the classifier withholds making a decision. In the center and rightmost plots, A lies completely on a single side of the decision boundary, so the classifier assigns a label to the incomplete data.

linear programs:

$$\max_{x \in A} f(x) = \max_{x} \beta^T x + b \tag{5.10}$$
$$\text{subject to } (x-m)^T R^{-1} (x-m) \leq \delta$$

$$\min_{x \in A} f(x) = \min_{x} \beta^T x + b \tag{5.11}$$
$$\text{subject to } (x-m)^T R^{-1} (x-m) \leq \delta.$$

Proposition 2: *The solutions to (5.10) and (5.11) are, respectively*

$$\max_{x \in A} f(x) = \beta^T m + \sqrt{\delta} \parallel R^{1/2} \beta \parallel_2 + b$$
$$\min_{x \in A} f(x) = \beta^T m - \sqrt{\delta} \parallel R^{1/2} \beta \parallel_2 + b.$$

The proof is in Appendix A.2.

For a linear set A such as the näive Bayes box constraint set given in (5.5), the maximum and minimum are:

$$\max_{x \in A} f(x) = \max_{x} \beta^T x + b \tag{5.12}$$
$$\text{subject to } m(\ell) - s_\tau(\ell) \leq x \leq m(\ell) + s_\tau(\ell) \; \forall \; \ell = 1, ..., d$$

$$\min_{x \in A} f(x) = \min_{x} \beta^T x + b \tag{5.13}$$
$$\text{subject to } m(\ell) - s_\tau(\ell) \leq x \leq m(\ell) + s_\tau(\ell) \; \forall \; \ell = 1, ..., d.$$

The solution of (5.12) is $\beta^T m + |\beta^T| s_\tau + b$, and the solution of (5.13) is $\beta^T m - |\beta^T| s_\tau + b$.

5.3.3 Quadratic Discriminants

If the class discriminant functions are quadratic, then $f(x) = f_2(x) - f_1(x)$ will also be quadratic and, thus, can be written

$$f(x) = (x - v)^T V (x - v) + b. \tag{5.14}$$

Since (5.14) is the difference of two quadratics, V will generally be indefinite even if $f_2(x)$ and $f_1(x)$ are both positive semi-definite.

First consider finding the maximum and minimum of (5.14), as required by the incomplete decision rule (5.9), over a quadratic constraint set A. Since V is indefinite, this is a non-convex optimization problem. However, strong duality holds for finding the minimum or maximum of any quadratic function subject to a quadratic constraint, as shown by [8, Appendix B]. The dual problem is a semi-definite program (SDP), and can therefore be solved using convex optimization such as an interior point methods. However, in our experiments, we found the SDP solution to be prohibitively slow. Therefore, we instead propose to use the two-step gradient descent approach described in Appendix B. Martinez showed that there is at most one local non-global solution to this non-convex problem [37]. Also, since we need only know if the minimum or maximum of $f(x)$ is less than or greater

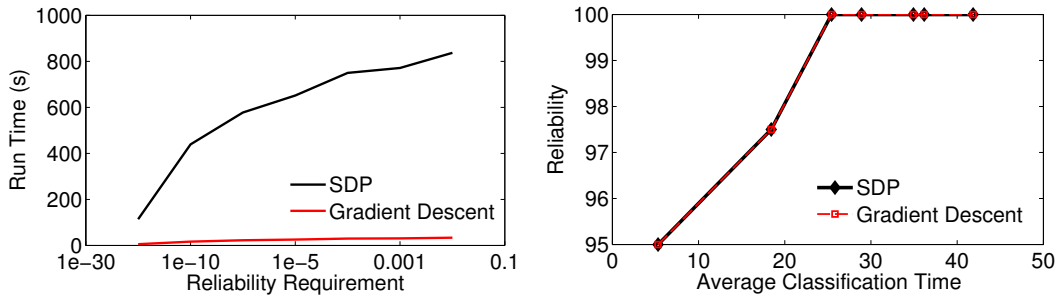

Figure 5.5: The left figure shows the time required by the SDP vs gradient descent solutions for different reliabilities. The right figure verifies that the solution for the methods is identical.

than zero, we can often stop the gradient descent before convergence. Figure (5.5) shows a run-time comparison between the SDP solution solved using `CVX` [27] running `SeDuMi` [63] and the gradient-descent solution.

Now consider finding the maximum and minimum of (5.14) over the box set A. An efficient solution is obtained by first performing a change of variables that diagonalizes V. Define $y = V^{1/2}x$ and $w = V^{1/2}v$, then $f(x) = f(y) = \| y - w \|^2 + b$. After this change of variables, the maximum and minimum computations required by the incomplete decision rule (5.9) can be greatly simplified by making the näive Bayes assumption on the random variable $Y = V^{1/2}X$ as opposed to on X. Defining the mean of Y as $m_y = V^{1/2}m$,

$$\max_{x \in A} f(x) = \max_{y \in A} \| y - w \|^2 + b \tag{5.15}$$

$$\min_{x \in A} f(x) = \min_{y \in A} \| y - w \|^2 + b, \tag{5.16}$$

with

$$A = \{ y \text{ s.t. } y(\ell) \in [m_y(\ell) - s_\tau(\ell),\ m_y(\ell) + s_\tau(\ell)],\ \forall\ \ell = 1, ..., d \},$$

where the $s_\tau(\ell)$ are determined by the inverse cdf of $Y(\ell)$.

After this change of variables, the y that maximizes (5.15) is found by assigning each $y(\ell)$ to the edge of the box that maximizes the distance from $w(\ell)$. Similarly, the y that

minimizes (5.16) assigns $y(\ell) = w(\ell)$ if $w(\ell) \in [m_y(\ell) - s_\tau(\ell),\ m_y(\ell) + s_\tau(\ell)]$. Otherwise, $y(\ell)$ is assigned to the edge of the box that minimizes the distance to $w(\ell)$.

5.3.4 Multi-class Classifiers

I now extend the results of the previous section to multi-class classifiers. For multi-class classifiers, the classification rule (5.6) can be expressed as

$$\hat{g}(x) = c \ \text{ if } \ f_c(x) - f_h(x) \geq 0 \text{ for all } \ h \neq c. \tag{5.17}$$

Using (5.17), the proposed incomplete data classification rule (5.2) can be written

$$\hat{g}(z) = \begin{cases} c & \text{if } \min\limits_{x \in A} f_c(x) - f_h(x) \geq 0 \text{ for all } h \neq c \\ \text{no decision} & \text{otherwise} \end{cases}. \tag{5.18}$$

That is, classify z as class c if the set A lies completely within the decision region for some class c, and do not decide at the requested reliability if the set A straddles a decision boundary.

If there are G total classes, then there are $2\binom{G}{2}$ possible checks, $\min_{x \in A} f_c(x) - f_h(x) \geq 0$, implied by (5.18). However, I show in the next section that regardless of the construction of set A, one must compute at most $2(G-1)$ of these checks. Furthermore, if the set A contains the posterior mean m (as it does in all of the proposed constructions), then a decision can be made with at most $G - 1$ checks:

Guess: Let $c = \arg\max\limits_{g} f_g(m)$.

Check: Sequentially check if $\min_{x \in A} f_c(x) - f_h(x) \geq 0$ for $h = 1, 2, \ldots, G,\ h \neq c$. If the check fails for any h, stop, and output the result *no decision*. If the check holds for all h, then classify early as class c.

5.3.5 General Multi-class Decision Process

I provide a provably efficient multi-class decision process for arbitrary constructions of the constraint set A. Let class c *dominate* class h and class h be *dominated* by c if $f_c(x) - f_h(x) \geq 0$ for all $x \in A$. If neither class dominates the other one, then the two classes are called *tied*. To classify the incomplete data z early as class c, class c must dominate all other classes.

Proposed Decision Process:

Initialize: Begin with all G classes labeled `candidate`.

Compare: Choose any two classes c and h that are labeled `candidate` and check if $\min_{x \in A} f_c(x) - f_h(x) \geq 0$. If yes, then label h as `dominated`. If no, then perform a second check to see if $\min_{x \in A} f_h(x) - f_c(x) \geq 0$, and if so then label c as `dominated`, and otherwise label both classes as `tied`. Continue this process until fewer than two classes are labeled `candidate`. If no classes remain that are labeled `candidate`, then output *no decision*. If one class is labeled `candidate`, then proceed to the *Final Comparison*.

Final Comparison: Check if the last class labeled `candidate` dominates every class labeled `tied`. If yes, classify the incomplete data as the class labeled `candidate`, if no, output *no decision*.

Proposition 3: *The above decision process correctly determines the dominating class or that there is no dominating class.*
The proof is in Appendix A.3.

Proposition 4: *Given G classes, the above decision process requires at most $2(G-1)$ and at least $G-1$ checks of $\min_{x \in A} f_h(x) - f_c(x) \geq 0$.*
The proof is in Appendix A.4.

5.4 Estimation of the Complete Test Data Distribution

In order to construct the sets A in Section 5.2, one must estimate the mean m and covariance R of the complete test data X. This can be done by leveraging the incomplete information about the test signal that is currently available along with the prior knowledge of the structure of the test signal that is available in the corpus of training data. Two estimation methods are proposed: 1) joint Gaussian estimation, and 2) Gaussian mixture model (GMM) estimation. These approaches are similar to those used in missing feature imputation, for example in speech recognition as described by Raj and Stern [48]. However, the approach here differs from that of missing feature imputation in that the latter constructs only a point estimate of the unknown data, whereas here I construct estimates of the mean and covariance of the unknown data.

5.4.1 Joint Gaussian Estimation

For joint Gaussian estimation, it is assumed that the complete data X is distributed jointly Gaussian with the incomplete data Z. Therefore, the model is

$$\begin{bmatrix} X \\ Z \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \bar{x} \\ \bar{z} \end{bmatrix}, \begin{bmatrix} \Sigma_{x,x} & \Sigma_{x,z} \\ \Sigma_{z,x} & \Sigma_{z,z} \end{bmatrix} \right). \tag{5.19}$$

The model parameters in (5.19) are estimated from the training data. The mean and covariance parameters of X conditioned on the realization of the partial information $Z = z$ are

$$\begin{aligned} m &= \hat{\bar{x}} + \hat{\Sigma}_{x,z}\hat{\Sigma}_{z,z}^{-1}(z - \hat{\bar{z}}) \\ R &= \hat{\Sigma}_{x,x} - \hat{\Sigma}_{x,z}\hat{\Sigma}_{z,z}^{-1}\hat{\Sigma}_{z,x}, \end{aligned}$$

where $\hat{\bar{x}}$, $\hat{\bar{z}}$, and $\hat{\Sigma}$ are the estimated parameter values.

5.4.2 GMM Based Estimation

For the GMM model, it is assumed that the joint distribution of the complete data, X, and the incomplete data, Z, is a Gaussian mixture model, where the elements of the mixture are the class-conditional distributions. Under these assumptions the model is

$$\begin{bmatrix} X \\ Z \end{bmatrix} \sim \sum_{g \in \mathcal{G}} w(g) p \left(\begin{bmatrix} x \\ z \end{bmatrix} | g \right), \tag{5.20}$$

where $w(g)$ is the weight of the class g Gaussian and

$$p \left(\begin{bmatrix} x \\ z \end{bmatrix} | g \right) = \mathcal{N} \left(\begin{bmatrix} \bar{x}_g \\ \bar{z}_g \end{bmatrix}, \begin{bmatrix} \Sigma_{x,x}(g) & \Sigma_{x,z}(g) \\ \Sigma_{z,x}(g) & \Sigma_{z,z}(g) \end{bmatrix} \right).$$

The parameters of the model (the means, covariances, and weights) are again estimated from the training data.

Define

$$\begin{aligned} m_g &= \hat{\bar{x}}_g + \hat{\Sigma}_{x,z}(g) \hat{\Sigma}_{z,z}^{-1}(g)(z - \hat{\bar{z}}_g) \\ R_g &= \hat{\Sigma}_{x,x}(g) - \hat{\Sigma}_{x,z}(g) \hat{\Sigma}_{z,z}^{-1}(g) \hat{\Sigma}_{z,x}(g) \\ p(g|z) &= \frac{w_g p(z|G=g)}{\sum_{h \in \mathcal{G}} w_h p(z|G=h)}. \end{aligned}$$

Given a realization $Z = z$, one can compute the mean m of X as:

$$m = \mathrm{E}[X|z] = \sum_{g \in \mathcal{G}} \mathrm{E}[X, G|z] = \sum_{g \in \mathcal{G}} m_g p(g|z).$$

Furthermore, as shown in Appendix C, the covariance of X is

$$R = \sum_{g \in \mathcal{G}} p(g|z) \left(R_g + m_g m_g^T \right) - \sum_{q \in \mathcal{G}} \sum_{h \in \mathcal{G}} m_q m_h^T p(q|z) p(h|z).$$

5.5 Related Work

I first describe related work in early classification and missing features, then I contrast the proposed approach with optimal stopping, feature selection, online and incremental learning, and sequential hypothesis ratio testing.

5.5.1 Other Early Classification Work

Xing, et al. [70] considered the problem of making an early prediction on time-series data that matches that of a full length one nearest-neighbor classifier. Suppose that the labeled training dataset is $\{(x_i, g_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^d$. Their approach, called early classification on time-series (ECTS), is motivated by the idea of the *minimum prediction length* (MPL) of a training time-series x_i. Define $x_i(1:t) \in \mathbb{R}^t$ to be the first t samples of x_i. Furthermore, define, $\text{RNN}(x_i(1:t))$ to be the reverse nearest-neighbors of $x_i(1:t)$ which is the set of training samples that choose x_i to be their nearest-neighbor at time t. The MPL of x_i is the smallest time index k such that for all $k \leq \ell \leq d$ the following holds $\text{RNN}(x_i(1:\ell)) = \text{RNN}(x_i(1:d)) \neq \emptyset$. By this definition, the MPL is the smallest time index at which the reverse nearest-neighbors of x_i do not change as the rest of the time-series is revealed. At test time, a training point x_i can be used to assign a label to a test sample $x(1:t)$ once $t \geq \text{MPL}(x_i)$, the minimum prediction length of x_i.

The authors found that the above procedure was too conservative; therefore, they proposed a slightly modified way to find the MPL for ECTS. They first clustered the training data using a hierarchical clustering method and then selected the MPL for each training time-series depending on its cluster membership. They also introduced a parameter to control the 'earliness' of their approach called *minimum support* – a ratio that varies between zero and one, with zero resulting in the earliest classifier. However, the minimum support parameter is different from the τ parameter in my approach in that it does not provide an explicit guarantee on the reliability of the early decision.

Xing et al. cite [51] as the only existing study mentioning early classification on time-series data. Rodriguez and Alonso [51] propose to classify a time-series using a *literal* based classifier, where a literal is a descriptor describing what happens during a specified interval

of the time-series. For example, the literal *increases* would be set to one if the time-series increases during the specified interval, and would be set to zero otherwise. The authors mention that for early classification of time-series some of the literals will not yet have a value because the interval that they are measured in has not occurred yet. The authors propose to 'omit' these literals from the classifier in order to classifier early. This approach to early classification is the näive approach, as it does not consider whether or not a reliable decision can be made with the incomplete information.

5.5.2 Related Work on Missing and Noisy Features

Another related body of work is imputing (estimating) missing features. If missing features occur in the training data, then standard methods of classifier training cannot be used. One method of dealing with this problem, called single imputation, is to fill in the missing features with their estimated values. The missing features can be estimated using a multivariate regressor that is trained using the subset of training data with no missing features. [55] and [53] review missing feature methods for training data.

On the other hand, if features are missing in the test data, then they do not necessarily need to be imputed. A classifier can be trained for this test data by removing the same features that are missing in test data from the training data prior to classifier training. However, this approach requires the classifier to be retrained for each different subset of missing test features. This retraining can be avoided by estimating the missing features and then classifying the test data with the missing features filled in by their estimates. My method differs from methods that classify the imputed test features in that I use the estimated mean and variance in order to measure the reliability of a test decision made with the incomplete data. My work is complementary to imputation methods, and different estimation methods than the ones I used in Section 5.4 could be applied to the proposed approach.

If features are noisy rather than missing, then estimating the clean feature values can improve test accuracy. This problem arises, for example, in automatic speech recognition (ASR) systems when the test signal is masked by noise [12, 47, 48]. Raj and Stern [48]

compare MAP estimates for noisy features in ASR systems using Gaussian and GMM based estimators with models similar to those that I describe in Section 5.4.

5.5.3 Optimal Stopping Rules

Quoting [20], "The *theory of optimal stopping* is concerned with the problem of choosing a time to take a given action based on sequentially observed random variables in order to maximize an expected pay-off or to minimize an expected cost." While the high-level goal is the same, the optimal stopping perspective requires specification of misclassification costs and delay costs, which are often difficult to specify. Given such costs, an optimal stopping rule approach would attempt to estimate the probability of each class given the current incomplete information, and determine the expected costs of making a decision or waiting.

5.5.4 Feature Selection

A related problem in classification is to determine the best subset of features to use in classification. For example, the classic *forward selection method* sequentially adds in features based on their marginal value. Different stopping rules have been proposed to decide when to stop sequentially adding the features [13]. Many of these stopping rules are not applicable to the problem I focus on because they assume that all increasing sets of features can be compared, rather than that one only has the incomplete set of features and must make a decision [13]. In addition, these stopping rules are based only on analyzing the training data statistics, and from my perspective are strictly suboptimal in that they do not consider the current incomplete information.

5.5.5 Online and Incremental Learning

In this work I assume that a fixed set of training data is given, and that incremental features of a test sample become available. These assumptions differ from the usual set-up of online learning (also known as incremental learning), which assumes that incrementally more training data becomes available to train the classifier over time (e.g. [43],[18],[14]).

Also assuming the online learning set-up, Fu, et al. [23] propose a stopping rule for deciding when enough training samples have been received to classify with confidence.

5.5.6 Sequential Hypothesis Testing

The sequential probability ratio test (SPRT) [66] is a greedy alternate to the proposed work, designed for use with probabilistic models of two hypotheses. In the context of binary classification, and a generative model $p(y|x_k)$, it accumulates the log-likelihood ratio:

$$S_k = S_{k-1} + \log p(y_1|x_k) - \log(y_2|x_k), \tag{5.21}$$

and if S_k exceeds a preset threshold t_1, the signal would be called for class one, and if S_k goes below a pr-set negative threshold t_2, the signal would be called for class two. The thresholds are set to achieve desired error levels on class one and class two, respectively.

Armitage [3] expanded SPRT for the multi-hypothesis case and applied it to linear discriminant analysis classification (in which each class is assumed to be drawn from a distribution with the same covariance matrix) for a different problem than the one treated here: given a sequence of iid samples from one class, he prescribed how to use SPRT to give a rule for how and when to determine the class.

A key difference between the proposed approach and the SPRT approach is that (5.21) is greedy: new features do not change the contribution to the log-likelihood already made by previous features, which stems from the standard SPRT assumption successive observations are independent. But the classifiers considered in this work are not trained to consider the features independently. Furthermore, I assume correlations between the features in order to estimate a probability distribution over the unknown part of the feature vector, which I use to define a constraint set.

5.6 Experiments

Section 5.6.1 details the datasets, experimental set-up, and classifiers used. I first compare the proposed methods to construct sets A of measure τ, reported in Section 5.6.2, and the proposed estimation methods for the moments of $P_{X|z}$, reported in Section 5.6.3. Then in

Table 5.1: Time-series Datasets

Dataset	Time-series Length	Number of Classes	Training Samples	Test Samples
Chlorine Concentration	166	3	467	3840
Italy Power Demand	24	2	67	1029
Face (All)	131	14	560	1690
Medical Images	99	10	381	760
Non-Invasive Fetal ECG 1	750	42	1800	1965
Non-Invasive Fetal ECG 2	750	42	1800	1965
Starlight Curves	1024	3	1000	8236
Swedish Leaf	128	15	500	625
Synthetic Control	60	6	300	300
Two Patterns	128	4	1000	4000
U Wave Gesture Library X	315	8	896	3582
U Wave Gesture Library Y	315	8	896	3582
U Wave Gesture Library Z	315	8	896	3582
Wafer	152	2	1000	6174
Yoga	426	2	300	3000

Section 5.6.4, I show that applying a dimensionality reduction method can greatly reduce the computation needed at test time. Lastly, I compare the proposed reliable classifier to other approaches to early classification.

5.6.1 Experimental Set-up and Details

I demonstrate performance using all of the time-series datasets available on the *UCR Time-Series Classification and Clustering Page* [33] that have at least five hundred test samples and at least fifteen training examples per class when this paper was written. I use the given training and test splits, so all results can be reproduced. I also use the Synthetic Control dataset from this repository, a dataset of Gaussian data that has only three hundred test samples, to further the differences between the constraint sets and estimation methods that I have described for the proposed incomplete decision rule. Table 5.1 gives details for the used datasets.

The time-series classification experiments are performed as follows. The test dataset consists of n sampled time-series vectors and corresponding labels $\{x_i, g_i\}_{i=1}^n$, with $x_i \in \mathbb{R}^d$ and $g \in \mathcal{G}$. At time t, the incomplete data for the i^{th} test time-series is $z_i \in \mathbb{R}^t$, the first t

samples of x_i. At each time t I check the proposed incomplete decision rule and classify z_i if the reliability condition is met for τ. I plot results for a set of choices of τ.

Let $t_i(\tau)$ be the minimum time at which the i^{th} test signal can be classified with reliability constraint τ, and let $\hat{g}(z_i(\tau))$ be the class label assigned to z_i at this time. I measure the test reliability as $\frac{1}{n}\sum_{i=1}^{n} \mathrm{I}(\hat{g}(z_i(\tau)) = \hat{g}(x_i))$, where $\hat{g}(x_i)$ is the label assigned to the complete data and $\mathrm{I}(\cdot)$ is one if the argument is true and zero otherwise. I also measure the average classification time as the mean of the $t_i(\tau)$. Ideally, I would like to classify with the smallest average classification time while still meeting reliability requirement τ.

I perform incomplete classification experiments with two different discriminant classifiers. The first classifier is local QDA [24]. Local QDA learns the mean and covariance for the class g discriminant function for test point x, $f_g(x)$, by estimating them using the k nearest class g training points to test point x. I choose $k \in \{1, 2, 4, 8, 16, 32, 64, 128\}$ by cross-validation on the training data. In my implementation of local QDA, I use a diagonal covariance matrix, and I regularize the covariance estimate by adding $10^{-4}\mathrm{I}$, where I is the identity matrix. Since I do not have the complete data x, I instead estimate the mean and covariance for $f_g(x)$ by finding the nearest class g neighbors to the mean of X. The second classifier that I use is a linear SVM which I implement using LibSVM [11] with default settings.

5.6.2 Comparison of Construction of Sets of Measure τ

I first compare the three set construction methods proposed Section 5.2, the Chebyshev set (5.3), the Gaussian näive Bayes quadratic set (5.4), and the Gaussian näive Bayes box set (5.5).

I vary the reliability parameter between four values $\tau = [0.001, 0.1, 0.25, 0.9]$, and I perform prediction using the jointly Gaussian model (5.19). Figure 5.6 plots the results for the Synthetic Control, Medical Images, and Two Patterns datasets. In all cases, the empirical reliability rate exceeds the reliability requirement τ. Additionally, these plots verify that the Chebyshev set is the most conservative, as it waits the longest to classify the test data, and the näive Bayes quadratic set is the least conservative.

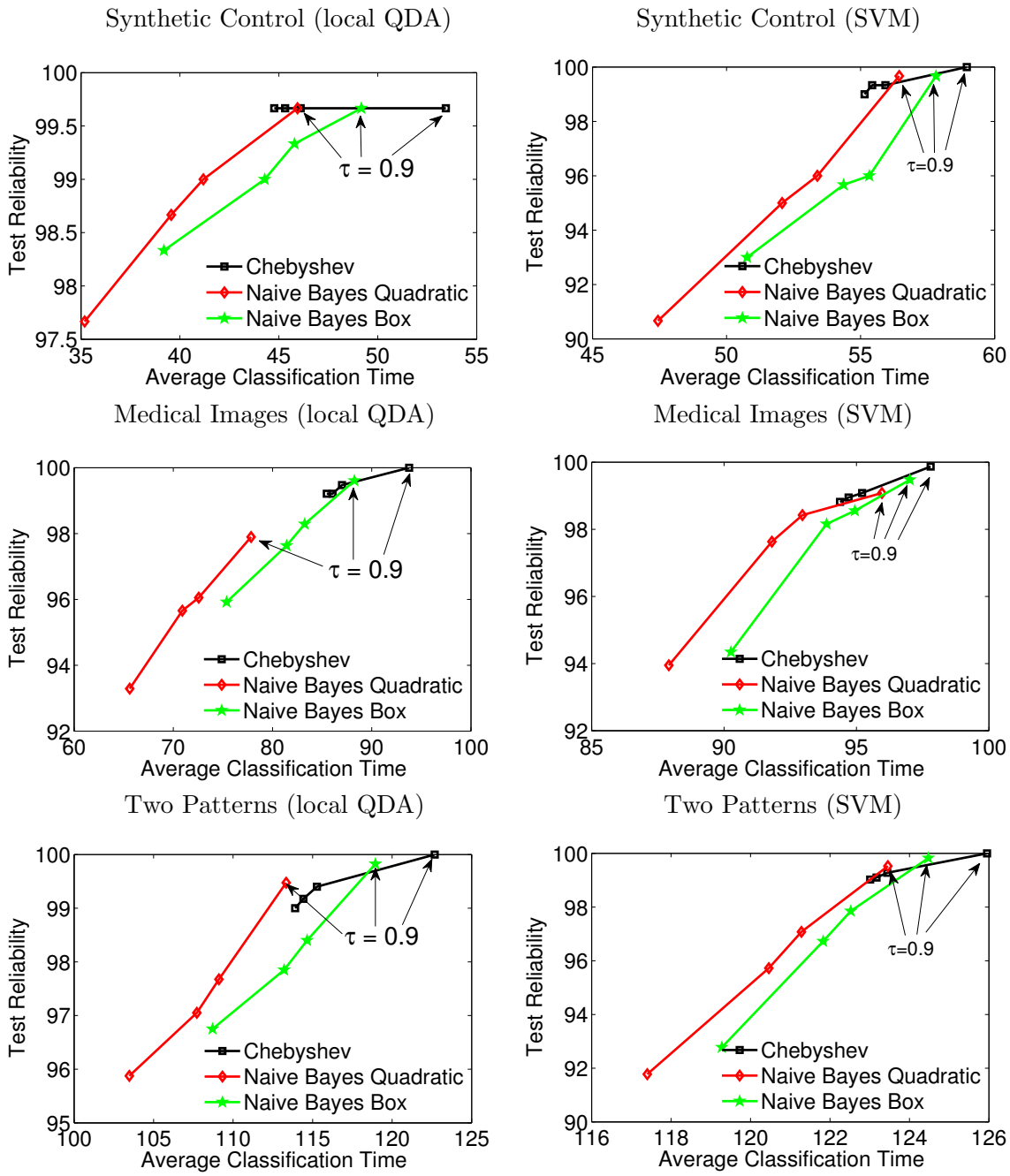

Figure 5.6: Average classification time vs test reliability for local QDA (left column) and linear SVM (right column) using jointly Gaussian prediction with τ varied between [0.001, 0.1, 0.25, 0.9].

Table 5.2 compares the average testing time per test sample for the three different constraint sets when $\tau = 0.9$. This table shows that the näive Bayes box set is the least computationally complex, followed by the näive Bayes quadratic set, and finally the Chebyshev set.

Table 5.2: Average test time per sample, in seconds, for the three different constraint sets.

	Local QDA			Linear SVM		
	Chebyshev	Quadratic	Box	Chebyshev	Quadratic	Box
Synthetic Control	1.8	0.7	0.4	0.4	0.3	0.3
Medical Images	27.1	2.7	1.4	1.9	1.0	0.8
Two Patterns	12.45	2.0	1.0	1.8	0.5	0.3

5.6.3 Comparison of Estimation Methods

In this section I compare the performance of reliable incomplete classification using jointly Gaussian estimation (5.19) to that using GMM estimation (5.20). I use the same classifiers and values for τ as given in the previous section.

Figure 5.7 plots the average classification time vs. test reliability for the jointly Gaussian and GMM estimation methods using the näive Bayes quadratic constraint set. The figure shows that on the Synthetic Control and Medical Images datasets, the GMM method dominates the jointly Gaussian over all τ values for both classifiers. On the Two Patterns dataset with local QDA classification, the GMM method is not uniformly better than the jointly Gaussian method.

Table 5.3 compares the total testing time of the two approaches, and as expected, the GMM method requires more test time than the simpler jointly Gaussian method.

5.6.4 Dimensionality Reduction Features

An advantage of the proposed reliable incomplete classification approach is that it can use any features derived from the data for which the mean and covariance can be estimated. Therefore, as an alternative to using the time-series samples as the features, I propose to

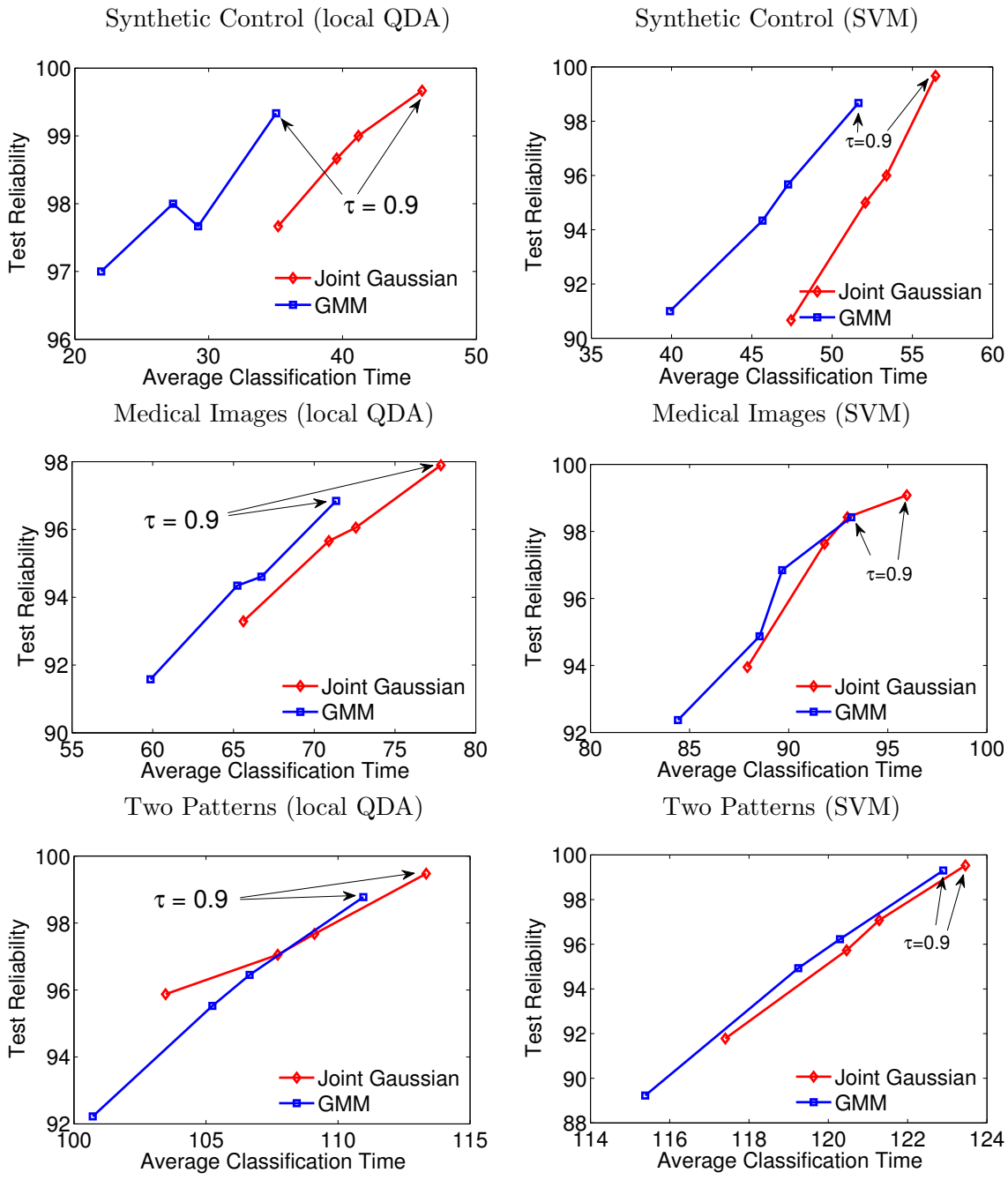

Figure 5.7: Average classification time vs test reliability for local QDA (left column) and linear SVM (right column) using the näive Bayes quadratic constraint set with τ varied between [0.001, 0.1, 0.25, 0.9].

Table 5.3: Average test time per sample, in seconds, for the two different estimation methods.

	Local QDA		Linear SVM	
	Joint Gaussian	GMM	Joint Gaussian	GMM
Synthetic Control	0.7	0.9	0.3	0.7
Medical Images	2.7	5.5	1.0	2.7
Two Patterns	2.0	3.4	0.5	0.8

select a smaller feature set by first preprocessing the time-series using supervised linear dimensionality reduction with matrix $B^T \in \mathbb{R}^{d \times \ell}$, $\ell < d$.

When using linear dimensionality reduction, the complete data becomes a vector $y = B^T x \in \mathbb{R}^\ell$ as opposed to $x \in \mathbb{R}^d$. Let Y be a random variable representing the distribution of the complete data. Due to the linear relationship $Y = B^T X$, it is straighforward to compute the mean and covariance parameters of Y using the models described in Section 5.4.

Linear dimensionality reduction can provide two advantages over classifying the time-series features. First, it can diminish the impact of noisy or non-discriminative features in the time-series data, thus providing increased accuracy. Second, reducing the number of features reduces the computational complexity. For a time-series with d samples, there are $d - t$ unknown samples at time t. Thus, if we simply use the time-series samples as the features for classification, the optimization problem that the reliable incomplete classifier must solve has $d - t$ free variables. For a long time series, this can cause the computational complexity to become extreme when t is small. However, performing linear dimensionality reduction reduces the number of unknowns to ℓ which can greatly reduce the number of variables in the optimization for reliable classification.

I use local discriminative Gaussian (LDG) dimensionality reduction [45] to learn B. I choose LDG dimensionality reduction because 1) it can separate multi-modal data, 2) the solution is fast, requiring only a maximal eigenvalue decomposition, and 3) it has been shown to work well even when few training samples are provided and the input dimensionality is large. Furthermore, I can choose the best input dimensionality by performing

Table 5.4: Time-series length and the number of features after LDG dimensionality reduction as well as a comparison of the testing time, in milliseconds, required to perform reliable local QDA classification with the näive Bayes quadratic constraint set and jointly Gaussian estimation. The test time shown measures the average time, per test sample, to perform reliable classification at time $t = 1$. Therefore, this is a worst case test time in terms of real-time performance as the number of unknowns in the optimization problem for reliable classification is maximized at time $t = 1$.

Dataset	Time-series length	Number of LDG features	Test time at t=1 (ms)	LDG test time at t=1 (ms)
Chlorine Concentration	166	42	76	4
Italy Power Demand	24	2	2	1
Face (All)	131	30	40	2
Medical Images	99	11	18	2
Non-Invasive Fetal ECG 1	750	30	6,107	4
Non-Invasive Fetal ECG 2	750	23	5,789	3
Starlight Curves	1024	26	15,697	2
Swedish Leaf	128	20	35	2
Synthetic Control	60	7	9	1
Two Patterns	128	22	31	2
U Wave Gesture Library X	315	12	418	2
U Wave Gesture Library Y	315	6	382	1
U Wave Gesture Library Z	315	10	393	2
Wafer	152	17	55	2
Yoga	426	26	945	2

cross-validation on the training dataset to find a reduced space that is both small and accurate. Table 5.4 shows the dimensionality of the training data after LDG dimensionality reduction. The table also compares the testing time required to perform reliable local QDA classification with the näive Bayes quadratic constraint set with jointly Gaussian estimation at time $t = 1$ with and without LDG dimensionality reduction. On the datasets with more than three hundred time-series samples, using LDG dimensionality reduction results in an orders of magnitude decrease in the testing time.

5.6.5 Comparison to Other Methods

In this section, I compare the performance of the reliable incomplete data classifier to ECTS [70][2] and several baselines. For all experiments in this section, I use the näive Bayes quadratic constraint set because it proved to be uniformly better than the box constraint set across all experiments in Section 5.6.2, while not being as overly conservative as the Chebyshev set. I also use the jointly Gaussian estimation method as it is faster to compute than the GMM method, particularly for the long time-series with many classes (the three U Wave Gesture Library datasets and two Non-Invasive Fetal ECG datasets). I also only show results for local QDA, as the reliable local QDA classifier classified earlier than reliable SVM in all experiments of Sections 5.6.2 and 5.6.3.

ECTS trades off between the objectives of classifying early and ensuring that early labels meet final labels by using a parameter, MS, that varies between zero and one, with zero resulting in the earliest classification time. However, I emphasize that this parameter is not the same as my reliability parameter τ, in that it provides no guarantee on reliability of the early predictions, but is instead a knob that the user can tune to trade off between earliness and reliability. [70] set this parameter to 0 in the majority of their experiments. I compare to ECTS by varying this parameter between MS = [0, 0.05, 0.1, 0.2, 0.4, 0.8].

I also compare to the performance of two baseline methods that I call *Fixed-time local QDA* and *Fixed-time 1-NN*. These methods use no predictive power, but instead classify all test signals at some user specified time: t samples.

---

[2]The authors provide code for ECTS at http://zhengzhengxing.blogspot.com/p/research.html.

The reliability results are shown in Figures 5.8 and 5.9. Reliable incomplete local QDA classification and reliable incomplete local QDA classification with LDG features perform well across all experiments. Reliable local QDA using the time series samples as features (the red line in the plots) dominates fixed-time local QDA in test reliability for all datasets over all values of τ. The only times that reliable local QDA with LDG features fails to dominate fixed-time local QDA are when $\tau = 0.001$ on the Italy Power Demand, U Wave Gesture Library Y, and Wafer datasets.

We also note that the leftmost pink circle in these plots is the earliest possible average classification time that ECTS can achieve, as MS = 0 is the smallest possible value for the minimum support parameter. On the other hand, reliable early classification can achieve earlier times than those shown in the figures by setting $\tau < 0.001$. Therefore, if someone wanted to set τ by cross-validation on the training dataset, the reliable incomplete classifier offers more flexibility than ECTS.

Figures 5.10 and 5.11 plot the test accuracy of the various approaches. The accuracy plots show that for some datasets, local QDA achieves higher accuracy than 1-NN; therefore, ECTS suffers in comparison to reliable local QDA due to the fact that it attempts to match a less accurate classifier.

The accuracy plots also show that although ECTS is typically more reliable than fixed-time 1-NN, it is less accurate for at least one value of MS on twelve of the fourteen datasets. On the other hand, reliable local QDA using the time-series samples as features is less accurate than fixed-time local QDA on only the Medical Images and Chlorine Concentration datasets. However, on the Chlorine Concentration dataset, reliable local QDA with LDG features exceeds the accuracy of fixed-time local QDA. Furthermore, we found that reliable local QDA classification using GMM estimation (not shown) exceeds the accuracy of fixed-time local QDA on the Medical Images dataset. The proposed reliable classification approach can be used with a wide variety of features, classifiers, and estimation methods in order to maximize accuracy for a particular application.

Finally, the accuracy in the Starlight Curves dataset of Figure 5.11 demonstrates an important consideration for ECTS and reliable local QDA. Both of these methods try to match the labels that are assigned by the respective fixed-time classifiers at time 1024.

However, the fixed-time local QDA plot shows that the underlying local QDA accuracy remains essentially flat from time two hundred to time 1024, and fixed-time 1-NN shows that accuracy decreases. Therefore, both incomplete classifiers would be better served by treating the complete data as only the first two hundred samples rather than all 1024.

5.7 Conclusions

I have proposed a practical incomplete decision rule that is a conservative approximation of the optimal rule. Experiments on a set of time-series data showed consistently more reliable and earlier prediction on average than other approaches. I have also showed that for classifiers that employ linear or quadratic discriminants, checking the proposed decision rule either has an analytic solution or can be solved using convex optimization.

This work has focused on answering the question "With probability τ, will the classification decision from this incomplete data be the same as from the complete data?" The presented tools can also be used to answer the related question: "If I classify based on the current incomplete data, what is the probability that the assigned label will match that which would be chosen using the complete data?" The answer can be computed by finding the largest τ that makes the first question a "Yes," which may require guessing a τ, solving the first question, refining τ up or down depending on the answer, and iterating.

Another related question that can be answered is, "Can I reliably classify as class g with this incomplete data?" That is, there may be only one class (or a subset of classes) which one would like to identify with incomplete data. For example, in determining if a cyst is cancerous or benign, doctors will often have a patient come back every few months to see how it changes over time. There is generally no rush to call it benign, but one would like to classify it as cancerous as soon as that is a reliable class label. This question can be answered by applying the incomplete decision rule given in (5.2) only to the class of interest.

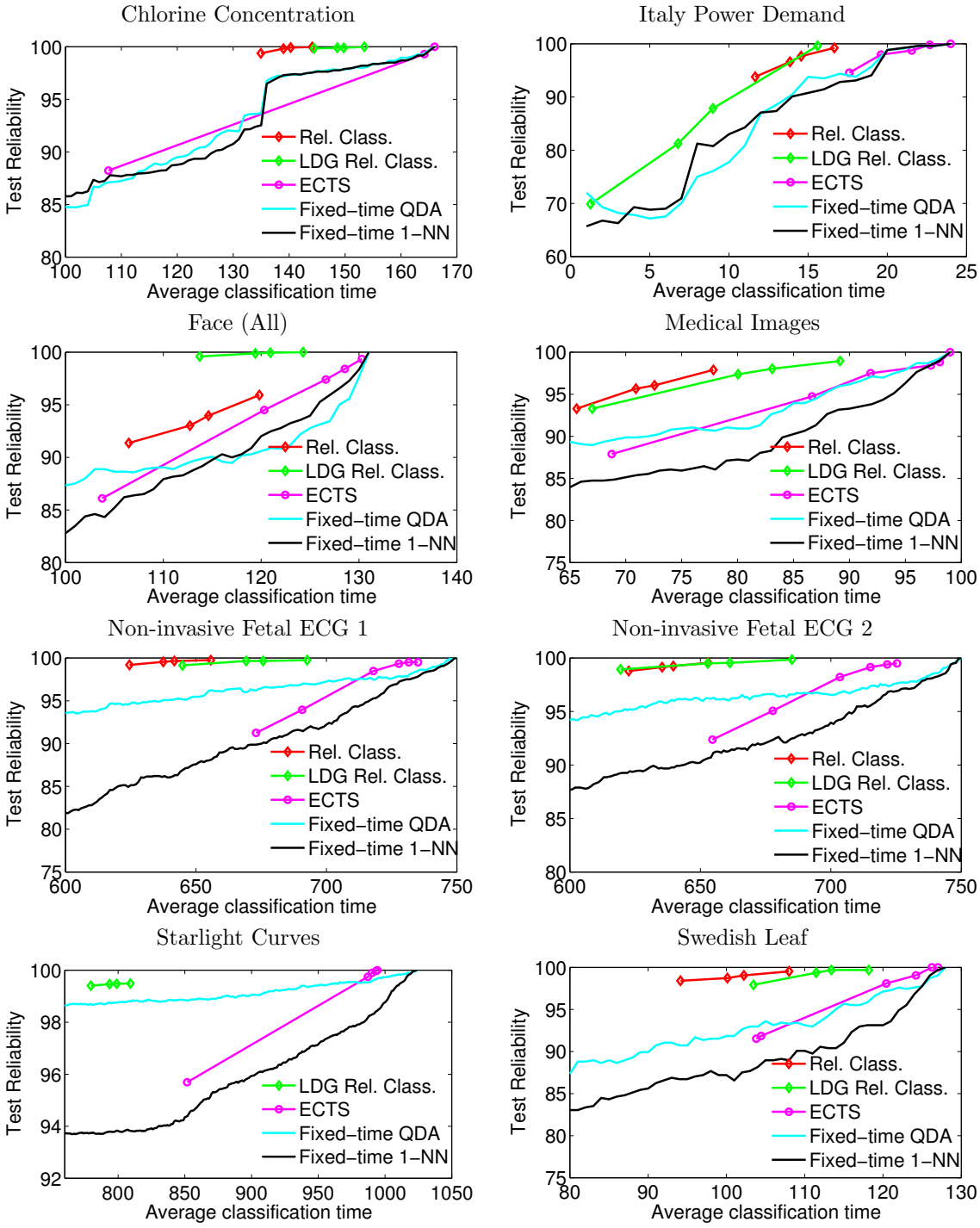

Figure 5.8: Average classification time vs test reliability for reliable incomplete local QDA classification (Rel. Class.), reliable incomplete local QDA classification with LDG features (LDG Rel. Class.), ECTS, Fixed-time local QDA, and Fixed-time 1-NN.

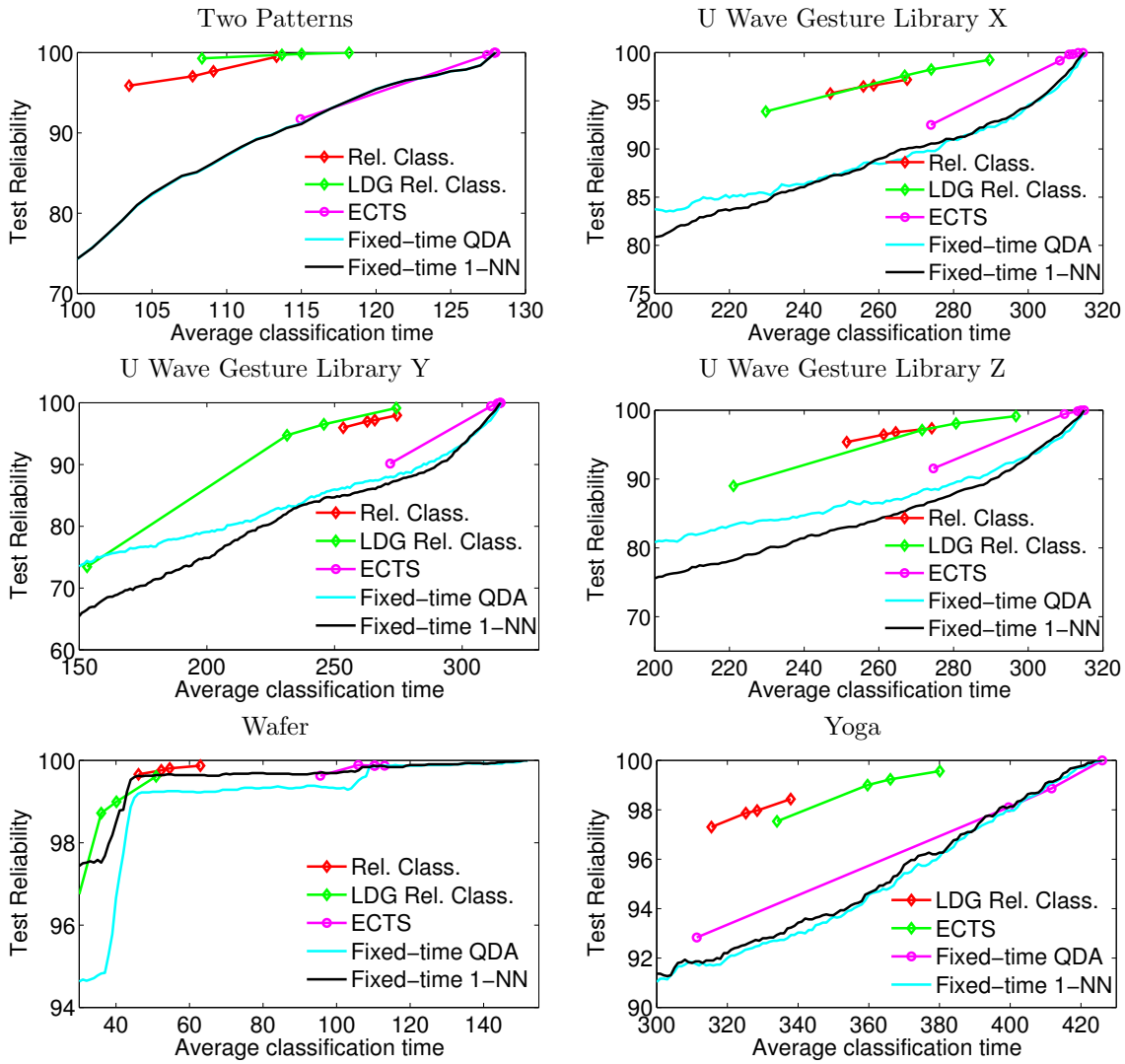

Figure 5.9: Average classification time vs test reliability for reliable incomplete local QDA classification (Rel. Class.), reliable incomplete local QDA classification with LDG features (LDG Rel. Class.), ECTS, Fixed-time local QDA, and Fixed-time 1-NN.

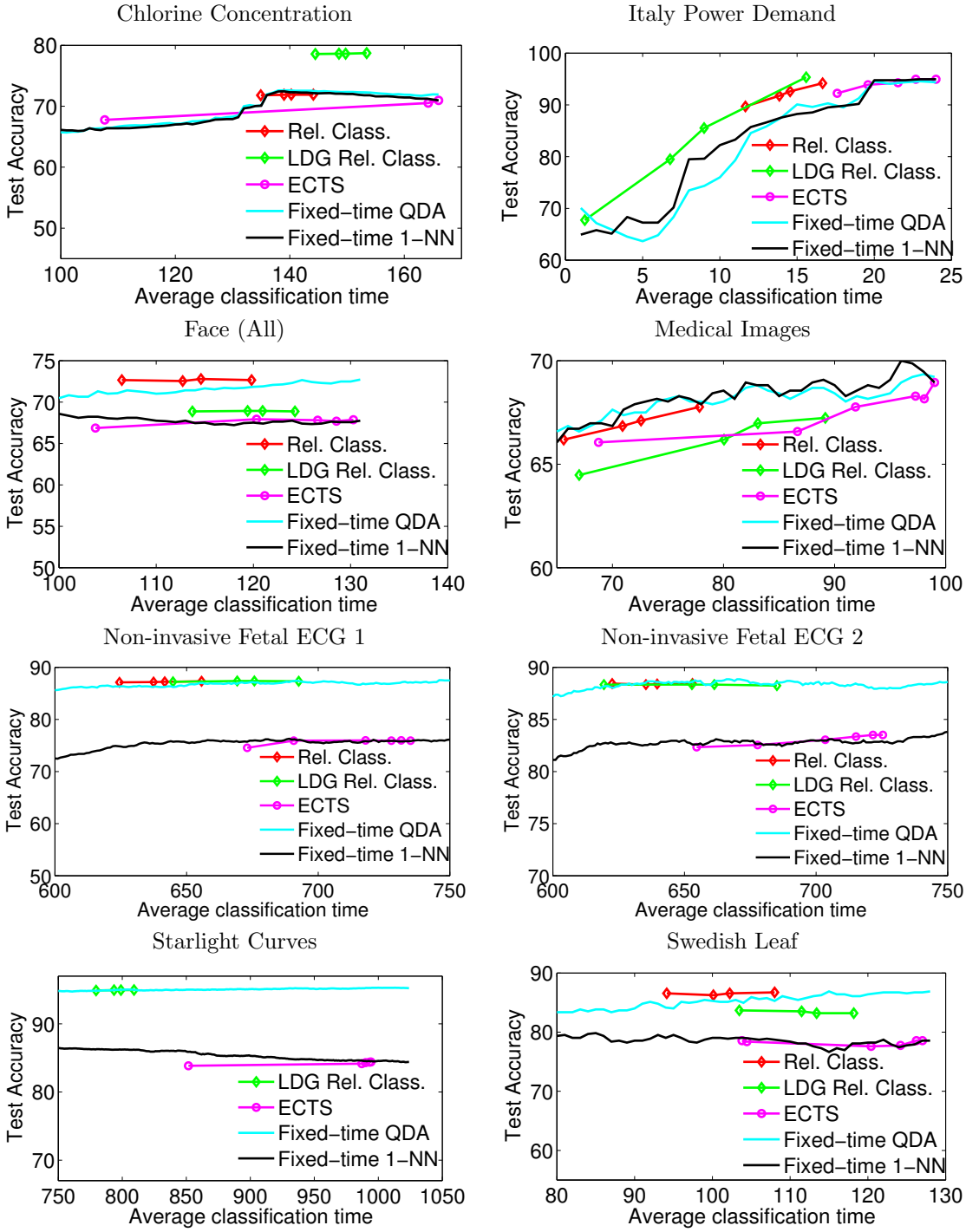

Figure 5.10: Average classification time vs test accuracy for reliable incomplete local QDA classification (Rel. Class.), reliable incomplete local QDA classification with LDG features (LDG Rel. Class.), ECTS, Fixed-time local QDA, and Fixed-time 1-NN.

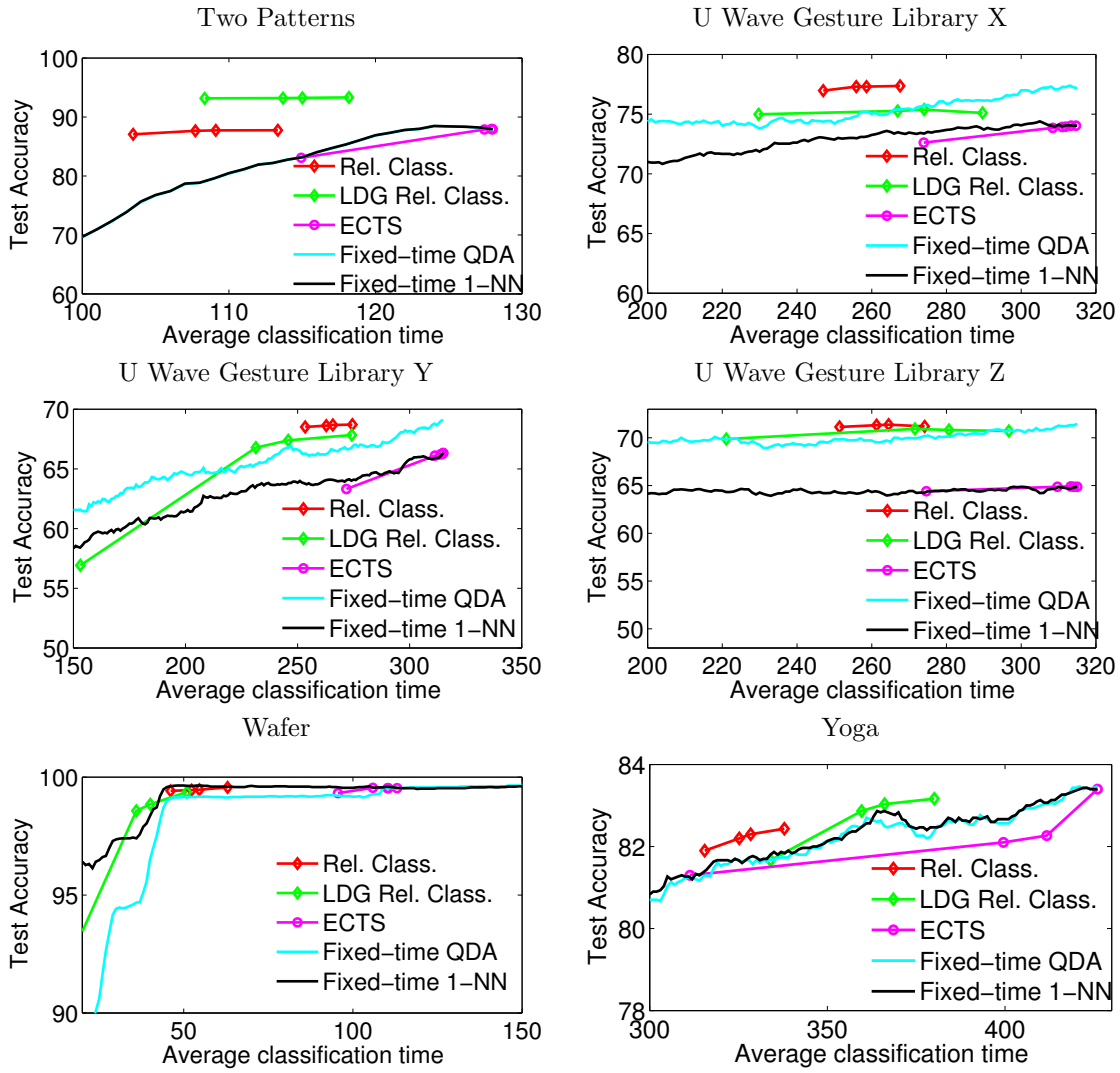

Figure 5.11: Average classification time vs test accuracy for reliable incomplete local QDA classification (Rel. Class.), reliable incomplete local QDA classification with LDG features (LDG Rel. Class.), ECTS, Fixed-time local QDA, and Fixed-time 1-NN.

Chapter 6

EXTENSIONS AND CONCLUSIONS

This thesis focused on the theory and use of LDG dimensionality reduction. This chapter reviews the contributions of this thesis and then discusses some limitations of LDG that can motivate future research.

6.1 Contributions

Chapter 3 motivated LDG dimensionality reduction through the goal of finding a dimensionality reduction matrix that minimized the leave-one-out cross-validation error of a local quadratic discriminant analysis classifier. By making several approximations to the leave-one-out error function, I was able to arrive at a final LDG optimization problem that could be solved via a maximal eigenvalue decomposition. This final optimization problem provided a trade-off between maximizing the likelihood that the data was drawn from the correct-class distribution, while minimizing the likelihood that it was drawn from one of the incorrect-class distributions. By adding a cross-validation term that controlled the trade-off between these two goals, I was able to achieve a more versatile objective. Furthermore, kernel LDG was shown to be able to find non-linear LDG mappings by using the kernel trick.

Experiments in Chapter 3 showed that LDG could find good dimensionality reduction matrices for classification. One of LDG's main advantages over other state-of-the art algorithms was the fact that it can be solved via a maximal eigenvalue decomposition. Other methods that require an iterative solution, such as NCA and ITML, proved to be more computationally expensive than LDG when the number of training features or training examples became large. LDG also exhibited good performance on small sample size problems.

Chapter 4 focussed on LDG dimensionality reduction for transfer learning. In this chapter, in addition to finding a reduced dimensional space such that the target domain

data was separated according to class, we also wished to find a space such that the source and target domain distributions were similar. In order to meet this goal, transfer LDG tried to find a mapping such that the target domain training data was brought closer to source domain data with the same class label, but separated from source domain data with a different class label. Experiments in this section showed that transfer LDG was effective on several different transfer learning datasets.

Finally, Chapter 5 motivated a framework for the classification of incomplete test data. The goal of this framework was to classify incomplete test data only if a probabilistic guarantee could be met that the label assigned to the incomplete data would match that which would be assigned to the unknown complete test data, and to otherwise wait for more data. This framework provided a trade-off between being able to classify with as little test data as possible, while also ensuring that the label assigned to the incomplete data was within some reasonable level of accuracy compared to the label assigned to the complete data. The resulting incomplete classifier showed good performance in the task of early time-series classification. Furthermore, it was shown that LDG dimensionality reduction could greatly reduce the computational complexity of incomplete classification as well as sometimes provide increased accuracy.

6.2 Future Work

One of the main limitations of LDG is that it cannot handle really big data. In fact, this is a drawback of all eigenvector based methods including PCA, FDA, and LFDA as well as LDG. Suppose that there are n training examples each of dimensionality d. In order to perform dimensionality reduction in Euclidean space, each of these methods must compute the eigenvectors of a $d \times d$ matrix. If $n < d$, then these methods can save some complexity by operating on the $n \times n$ kernel matrix. However, if both n and d are very large, then it may be computationally infeasible to do either.

An example of a dataset for which both the number of examples and number of features is very large is a recently compiled Google books dataset [38]. The dataset was compiled from a collection of 5,195,769 books, representing $\sim 4\%$ of the books ever published. Text data is typically featurized using *n-gram* features, where a 1-gram is a sequence of characters

without a space (for example "text" or "3.145"), and an n-gram is a sequence of n connected 1-grams (for example "a sunny day" is a 3-gram). The authors found that in the year 2000, there were 1,489,337 1-grams that occurred more than one billion times each.

Performing dimensionality reduction on a dataset of the magnitude of the Google books dataset will require a different approach. One approach may be to partition the feature data in multiple subsets, perform dimensionality reduction on each of these subsets of features, and then recombine the resulting features for a final feature dataset. Such an approach of partitioning the feature data into multiple subsets has been used before; however in this previous work the goal wasn't dimensionality reduction for large datasets, but was instead building an ensemble of classifiers [52].

The incomplete test data problem provides several avenues of future work as well. One future work item is determining how the method used to estimate the complete data relates to test reliability. This question is important because its answer will aid in determining what estimation method should be used. For instance, a standard approach is to select the estimator that produces the minimum mean square error (MMSE) estimate, but is the MMSE estimator the best choice for incomplete test data classification?

BIBLIOGRAPHY

[1] T. Ahonen, E. Rahtu, V. Ojansivu, and J. Heikkila. Recognition of blurred faces using local phase quantization. In *19th Intl. Conf. on Pattern Recognition*, pages 1–4, 2008.

[2] H. S. Anderson, N. Parrish, K. Tsukida, and M. R. Gupta. Reliable early classification of time series. In *Proc. IEEE Intl. Conf. Acoustics, Speech, and Signal Processing*, 2012.

[3] P. Armitage. Sequential analysis with more than two alternative hypotheses, and its relation to discriminant function analysis. *J. Royal Statistical Society. Series B (Methodological)*, 12(1):137–144, Jan. 1950.

[4] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *Computer Vision ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pages 404–417. Springer Berlin / Heidelberg, 2006.

[5] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(7):711–720, Jul. 1997.

[6] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Vaughan. A theory of learning from different domains. *Machine Learning*, 79:151–175, 2010.

[7] G. Bouchard and B. Triggs. The tradeoff between generative and discriminative classifiers. In *16th IASC International Symposium on Computational Statistics (COMPSTAT '04)*, pages 721–728, 2004.

[8] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, 2008.

[9] J. Bruna and S. Mallat. Classification with scattering operators. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 1561–1566, June 2011.

[10] J. Bruna and S. Mallat. Invariant scattering convolution networks. *CoRR*, abs/1203.1513, 2012.

[11] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Trans. Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[12] M. Cooke, P. Green, L. Josifovski, and A. Vizinho. Robust automatic speech recognition with missing and unreliable acoustic data. *Speech Communication*, 34(3):267–285, 2001.

[13] M. C. Costanza and A. A. Afifi. Comparison of stopping rules in forward stepwise discriminant analysis. *J. American Statistical Association*, 74(368):777–785, Jan. 1979.

[14] K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. *J. Machine Learning Research*, 3:951–991, 2003.

[15] A. d'Aspremont, L. El Ghaoui, M. I. Jordan, and G. R. G. Lanckriet. A direct formulation for sparse PCA using semidefinite programming. *SIAM Rev.*, 49(3):434–448, July 2007.

[16] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *Proc. Intl. Conf. on Machine Learning*, pages 209–216, 2007.

[17] D. Donoho. High-dimensional data analysis: The curses and blessings of dimensionality. *Aide-Memoire of a Lecture at AMS Conference on Math Challenges of the 21st Century*, 2000.

[18] M. Dredze, K. Crammer, and F. Pererira. Confidence-weighted linear classification. *Intl. Conf. Machine Learning (ICML)*, 2008.

[19] R. Duda, E. Hart, and D. Stork. *Pattern Classification*. Wiley-Interscience, 2nd edition, 2001.

[20] T. Ferguson. *Optimal Stopping Rules and Applications*. E-book available on the author's website., 2001.

[21] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Human Genetics*, 7(2):179–188, 1936.

[22] J. Friedman. Regularized discriminant analysis. *J. American Statistical Association*, 84(405):165–175, Mar. 1989.

[23] W. J. Fu, E. R. Dougherty, B. Mallick, and R. J. Carroll. How many samples are needed to build a classifier: A general sequential approach. *Bioinformatics*, 21(1):63–70, Jan. 2005.

[24] E. K. Garcia, S. Feldman, M. R. Gupta, and S. Srivastava. Completely lazy learning. *IEEE Trans. Knowledge and Data Engineering*, 22(9):1274–1285, Sept. 2010.

[25] A. Globerson and S. T. Roweis. Metric learning by collapsing classes. In *Proc. Advances in Neural Information Processing Systems 18*, pages 451–458. 2006.

[26] A. Globerson, S. T. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In *Proc. Advances in Neural Information Processing Systems 17*, pages 513–520. 2005.

[27] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 1.21. `http://cvxr.com/cvx/`, Apr. 2011.

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

[29] P. Howland, J. Wanb, and H. Park. Solving the small sample size problem in face recognition using generalized discriminant analysis. *Pattern Recognition*, 39(2):277–287, 2006.

[30] C. Hsu and C. Lin. A comparison of methods for multiclass support vector machines. *IEEE Trans. Neural Networks*, 13(2):415–425, 2002.

[31] P. Jain, B. Kulis, J. V. Davis, and I. S. Dhillon. Metric and kernel learning using a linear transformation. *J. Machine Learning Research*, 13:519–547, Mar. 2012.

[32] M. Journée, Y. Nesterov, P. Richtárik, and R. Sepulchre. Generalized power method for sparse principal component analysis. *J. Machine Learning Research*, 11:517–553, Mar. 2010.

[33] E. Keogh, Q. Zhu, B. Hu, Y. Hao., X. Xi, L. Wei, and C. A. Ratanamahatana. The UCR time series classification/clustering homepage. www.cs.ucr.edu/~eamonn/time_series_data/, 2011.

[34] H. Li, T. Jiang, and K. Zhang. Efficient and robust feature extraction by maximum margin criterion. *IEEE Trans. Neural Networks*, 17(1):157–165, Jan. 2006.

[35] S. Lohr. The age of big data. The New York Times, Feb. 2012.

[36] C. Ma and E. Chang. Comparison of discriminative training methods for speaker verification. In *Proc. IEEE Intl. Conf. Acoustics, Speech, and Signal Processing*, volume 1, Apr. 2003.

[37] J. M. Martinez. Local minimizers of quadratic functions on Euclidean balls and spheres. *SIAM J. on Optimization*, 4(1):159–176, 1994.

[38] J.-B. Michel, Y. K. Shen, A. P. Aiden, A. Veres, M. K. Gray, The Google Books Team, J. P. Pickett, D. Hoiberg, D. Clancy, P. Norvig, J. Orwant, S. Pinker, M. A. Nowak, and E. L. Aiden. Quantitative analysis of culture using millions of digitized books. *Science*, 331(6014):176–182, 2011.

[39] M. Nishiyama, A. Hadid, H. Takeshima, J. Shotton, T. Kozakaya, and O. Yamaguchi. Facial deblur inference using subspace analysis for recognition of blurred faces. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 33(4):838–845, 2011.

[40] G. Okopal, P. J. Loughlin, and L. Cohen. Dispersion-invariant features for classification. *J. Acoustical Society of America*, 123(2):832–841, 2008.

[41] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang. Domain adaptation via transfer component analysis. *IEEE Trans. Neural Networks*, 22(2):199–210, Feb. 2011.

[42] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Trans. Knowledge Data Engineering*, 22(10):1345–1359, Oct. 2010.

[43] S. Pang, S. Ozawa, and N. Kasabov. Incremental linear discriminant analysis for classification of data streams. *IEEE Trans. Systems, Man, and Cybernetics*, 35(5):905–914, Oct. 2005.

[44] N. Parrish, H. S. Anderson, M. R. Gupta, and D. Y. Hsaio. Classifying with confidence from incomplete test data. *Submitted for journal publication*, 2012.

[45] N. Parrish and M. R. Gupta. Dimensionality reduction by local discriminative Gaussians. In *Proc. Intl. Conf. on Machine Learning*, 2012.

[46] H. Peng, F Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(8):1226–1238, Aug. 2005.

[47] B. Raj, M. L. Seltzer, and R. M. Stern. Reconstruction of missing features for robust speech recognition. *Speech Communication*, 43(4):275–296, 2004.

[48] B. Raj and R. M. Stern. Missing-feature approaches in speech recognition. *IEEE Signal Processing Magazine*, 22(5):101–116, 2005.

[49] J. R. Rice. Experiments on Gram-Schmidt orthogonalization. *Mathematics of Computation*, 20(94):325–328, 1966.

[50] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *J. Machine Learning Research*, 5:101–141, Dec. 2004.

[51] J. J. Rodriguez and C. J. Alonso. Boosting interval-based literals: Variable length and early classification. In *ECAI'02 Workshop on Knowledge Discovery from (Spatio-) Temporal Data*, 2002.

[52] J. J. Rodriguez, L. I. Kuncheva, and C. J. Alonso. Rotation forest: A new classifier ensemble method. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(10):1619 –1630, Oct. 2006.

[53] A. Rogier, T. Donders, Geert J. M. G. van der Heijden, T. Stijnen, and K. G. M. Moons. Review: A gentle introduction to imputation of missing values. *J. of Clinical Epidemiology*, 59(10):1087–1091, 2006.

[54] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *Proc. Computer Vision ECCV 2010*, volume 6314 of *Lecture Notes in Computer Science*, pages 213–226. 2010.

[55] J. L. Schafer and J. W. Graham. Missing data: Our view of the state of the art. *Psychological Methods*, 7(2):147–177, 2002.

[56] B. Scholkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, 2001.

[57] B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In *Artificial Neural Networks ICANN'97*, volume 1327, pages 583–588. Springer Berlin / Heidelberg, 1997.

[58] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[59] H. Shen and J. Z. Huang. Sparse principal component analysis via regularized low rank matrix approximation. *J. Multivariate Analysis*, 99(6):1015–1034, 2008.

[60] S. Si, D. Tao, and B. Geng. Bregman divergence-based regularization for transfer subspace learning. *IEEE Trans. Knowledge and Data Engineering*, 22(7):929–942, July 2010.

[61] P. Simard, Y. LeCun, J. Denker, and B. Victorri. Transformation invariance in pattern recognition — Tangent distance and tangent propagation. In *Neural Networks: Tricks of the Trade*, volume 1524, pages 549–550. Springer Berlin / Heidelberg, 1998.

[62] M. K. Simon. *Probability Distributions Involving Gaussian Random Variables: A Handbook for Engineers, Scientists and Mathematicians*. Springer, New York, 2006.

[63] J. Sturm. Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11:625–653, 1999.

[64] M. Sugiyama. Dimensionality reduction of multimodal labeled data by local Fisher discriminant analysis. *J. Machine Learning Research*, 8:1027–1061, May 2007.

[65] P. D. Tao and L. T. H. An. Convex analysis approach to D.C. programming: Theory, algorithms, and applications. *ACTA Mathematica Vietnamica*, 22(1):289–355, 1997.

[66] A. Wald. *Sequential Analysis*. John Wiley, 1947.

[67] D. S. Watkins. Understanding the QR algorithm. *SIAM Review*, 24(4):427–440, 1982.

[68] K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *J. Machince Learning Research*, 10:207–244, June 2009.

[69] S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2:37–52, 1987.

[70] Z. Xing, J. Pei, and P. S. Yu. Early prediction on time series: A nearest neighbor approach. In *IJCAI*, pages 1297–1302, 1998.

[71] H. Yu and J. Yang. A direct LDA algorithm for high-dimensional data - with application to face recognition. *Pattern Recognition*, 34:2067–2070, 2001.

[72] H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *J. Computational and Graphical Statistics*, 15(2):265–286, 2006.

Appendix A

DERIVATIONS

A.1 Proof of Proposition 1

Since $V - A$ is a sum of outer products, it is symmetric. Thus, it can be rewritten as UMU^T, where $U \in \mathbb{R}^{d\times d}$ is a matrix whose columns are the orthonormal eigenvectors of $V - A$ and $M \in \mathbb{R}^{d\times d}$ is a diagonal matrix of the corresponding eigenvalues. Define m_i as the i^{th} diagonal element of M with $m_1 \leq m_2 \leq ... \leq m_d$. Also define E_i as a matrix that is equal to 1 at position (i, i) and is zero everywhere else. Then

$$\begin{aligned} \text{Tr}\left(B^T(V-A)B\right) &= \text{Tr}\left(B^TUMU^TB\right) \\ &= \text{Tr}\left(MU^TBB^TU\right) \\ &= \sum_{i=1}^{d} m_i \text{Tr}\left(E_iU^TBB^TU\right). \end{aligned} \tag{A.1}$$

Now define $w_i = \text{Tr}\left(E_iU^TBB^TU\right)$, and notice that w_i is the weight applied to m_i. First note that

$$\begin{aligned} \sum_{i=1}^{d} w_i &= \sum_{i=1}^{d} \text{Tr}\left(E_iU^TBB^TU\right) \\ &= \sum_{i=1}^{d} \text{Tr}\left(B^TUE_iU^TB\right) \\ &= \text{Tr}\left(B^TU\left(\sum_{i=1}^{d} E_i\right)U^TB\right) \\ &= \text{Tr}\left(B^TUU^TB\right) \\ &= \text{Tr}\left(B^TB\right) \\ &\overset{(a)}{=} \ell, \end{aligned} \tag{A.2}$$

where the equality indicated by (a) is due to the constraint that B^TB is equal to the $\ell \times \ell$ identity matrix.

Next, note that $w_i = \mathrm{Tr}\left(E_i U^T B B^T U\right)$ is equal to the number at position (i,i) of the matrix $U^T B B^T U$, which is equal to the squared norm of the i^{th} column of $B^T U$. Let b_i be the i^{th} column of B and u_i be the i^{th} column of U, then $w_i = |< b_i, u_i >|^2$ and thus

$$0 \leq w_i \overset{(c)}{\leq} \| b_i \|^2 \| u_i \|^2 \overset{(d)}{=} 1. \tag{A.3}$$

The inequality indicated by (c) is due to the Cauchy-Schwarz inequality. The equality indicated by (d) is due to the fact that $\| b_i \|^2 = 1$ by the constraint $B^TB = I$ and $\| u_i \|^2 = 1$ by definition.

Using (A.1), (A.2), and (A.3), the goal of the optimization problem (3.11) is to set B so that

$$\sum_{i=1}^{d} m_i w_i \tag{A.4}$$

is minimized with respect to w_i, subject to the constraints given by (A.2) and (A.3). Lemma 1, below, shows that (A.4) is minimized by by setting $w_1, ..., w_\ell = 1$. Since $w_i = | < b_i, u_i > |^2$, this is achieved by setting the ℓ columns of B equal to the ℓ smallest eigenvectors of $V - A$.

Lemma 1: *Recall that $m_1 \leq m_2, ..., \leq m_d$. The $\{w_i\}$ that minimize (A.4) subject to constraints (A.2) and (A.3) is w_1, w_2,...,$w_\ell = 1$ and $w_{\ell+1}$, $w_{\ell+2}$,...$w_d = 0$.*

Proof. The proof is by contradiction. When the $\{w_i\}$ are set in the manner indicated in the Lemma, the sum in (A.4) is

$$\sum_{i=1}^{d} w_i m_i = \sum_{i=1}^{\ell} m_i. \tag{A.5}$$

Now assume that we can reduce the sum in (A.5) by reducing some $w_j, 1 \leq j \leq \ell$ by $\epsilon, 0 < \epsilon \leq 1$.

In order to reduce w_j by setting it equal to $1-\epsilon$, some other w_i must be increased in order to satisfy $\sum_{i=1}^{d} w_i = \ell$. Since w_1 through w_ℓ already have maximum weight, ϵ must be distributed between $w_{\ell+1}$ through w_d. Without loss of generality, assume some $w_k, \ell+1 \leq k \leq d$ is set to ϵ, then the sum in (A.4) is

$$\sum_{i=1}^{d} m_i w_i = \sum_{i=1, i\neq j}^{\ell} m_i + (1-\epsilon)m_j + \epsilon m_k. \tag{A.6}$$

The increase in going from (A.5) to (A.6) is $\epsilon(m_k - m_j) \geq 0$, which is a contradiction. □

A.2 Proof of Proposition 2

For the minimum problem (5.11), the Lagrange dual function is $g(\lambda) = \beta^T m - \frac{1}{4\lambda} B^T R B + b - \lambda\delta$, a concave function of λ, and $g(\lambda)$ is maximized for $\lambda^* = \sqrt{\frac{1}{4\delta} B^T R B}$. Since $\lambda^* \geq 0$, it is dual feasible. Since the objective function is convex, strong duality holds, and thus the maximum of the dual problem equals the minimum of the primal problem. A similar analysis can be performed for the maximum problem.

A.3 Proof of Proposition 3

First, note that each pairwise check reduces the number of classes labeled `candidate` by either two classes if the classes tie, or by one class (the loser) if one class dominates. Second, once a class has tied with another class or has been dominated, it cannot be the correct dominating class. Thus the proposed decision process eventually reduces the number of classes labeled `candidate` to either zero or one. If there are zero classes left labeled `candidate`, then all classes have either tied or been dominated, and the above process correctly chooses not to classify. If there is one class remaining that is labeled `candidate` it must be compared to all the classes that tied on their first comparison. It is not necessary to also compare to the classes labeled `dominated` by the transitivity of the domination rule.

A.4 Proof of Proposition 4

We first note that in the *Compare* step, the pairwise comparison between classes c and h requires a single minimum computation if c dominates h, and two computations if the classes tie or if h dominates c. Furthermore, we define T to be the number of pairwise comparisons that result in ties during the *Compare* step, and D as the number of pairwise comparisons that do not result in a tie in the *Compare* step (such evaluations necessarily result in one class that was labeled `candidate` being re-labeled `dominated`). Thus, the compare step requires at most $2T + 2D$ minimum calculations.

On the other hand, each pairwise comparison in the *Final Comparison* check requires only a single minimum computation.

There are two cases to consider.

Case 1: Consider the case that the *Compare* step in the decision process results in one class left labeled `candidate`. Immediately prior to the *Final Comparison* step, there are $G-1$ classes that have been re-labeled `tied` or `dominated`, and since each tie results in two classes being re-labeled `tied`, it must be that

$$G - 1 = D + 2T. \tag{A.7}$$

In the *Final Comparison* step, the Gth class must be compared to at most the $2T$ classes labeled `tied`, each of which requires one minimum calculation. Thus the maximum number of calculations needed is

$$\begin{aligned} &2T + 2D + 2T \\ &= 2(G-1) \text{ by (A.7)}. \end{aligned}$$

Conversely, the best case is that there are no ties, and that each pairwise check requires only a single minimum calculation. This case requires $G - 1$ minimum calculations.

Case 2: The second case is that at the end of the *Compare* step there are zero classes labeled `candidate`. Therefore,

$$G = D + 2T, \tag{A.8}$$

and the total number of comparisons required is

$$\begin{aligned} & 2T + 2D \\ & = G + D \text{ by (A.8).} \end{aligned}$$

There can be at most $G - 2$ comparisons that result in one class dominating the other (otherwise, one class would remain labeled `candidate` after the *Compare* step), so the maximum number of minimum calculations is again $2(G - 1)$.

Since it requires at least one minimum calculation to change a class label from `candidate` to `dominated` or `tied`, the minimum number of calculations is G.

Appendix B

GRADIENT DESCENT SOLUTION FOR THE QUADRATIC MIN AND MAX PROBLEMS

The min problem with quadratic $f(x)$ subject to a quadratic constraint set is written as

$$\min_{x \in A} f(x) = \min_{x} (x - v)^T V (x - v) + b \tag{B.1}$$
$$\text{subject to } (x - m) R^{-1} (x - m) \leq \delta,$$

where V is indefinite and R is positive semi-definite. We propose to solve this problem using the two-step gradient descent approach described in [65].

We first reformulate (B.1) as the *trust region subproblem* (TRSP). Define

$$\begin{aligned} z &= R^{\frac{-1}{2}} (x - m) \\ A &= 2R^{\frac{1}{2}} V R^{\frac{1}{2}} \\ y &= 2R^{\frac{1}{2}} V (m - v) \\ b_{tsrp} &= b + v^T V v + m^T V m - 2m^T V v. \end{aligned}$$

Then rewrite (B.1) as

$$\min_{x \in A} f(x) = \min_{z} \frac{1}{2} z^T A z + y^T z + b_{tsrp} \tag{B.2}$$
$$\text{subject to } \parallel z \parallel \leq \sqrt{\delta}.$$

Let ρ equal the largest eigenvalue of A. The following two-step iteration converges to a z^* that is a local minimum of the TRSP (B.2):

$$\begin{aligned} \text{Step 1}: \quad & z_{k+1} = z_k - \frac{1}{\rho}(Az_k + y) \\ \text{Step 2}: \quad & z_{k+1} = \min \left[z_{k+1}, \ \frac{\| z_{k+1} \|}{\sqrt{\delta}} z_{k+1} \right], \end{aligned}$$

where Step 1 computes a gradient step, and Step 2 projects the z_{k+1} found in Step 1 onto the constraint set in (B.2).

The TRSP has been shown to have at most one local minimum that is not also the global minimum [37], and thus the above algorithm has proven to be robust in finding the minimum of (B.2).

Furthermore, for the incomplete data decision rule (5.9), it is not necessary to find the true minimum over A of $f(x)$, but it is instead sufficient to know only whether or not it is less than or equal to zero. Therefore, the iteration can be stopped early if $z_k^T A z_k + y^T z_k + b_{tsrp} \leq 0$.

Appendix C

VARIANCE OF A GAUSSIAN MIXTURE

Let $m_g = \mathrm{E}[X|g,z]$, $R_g = \mathrm{COV}[X|g,z]$, and $p(g|z)$ be defined as in Section 5.4.2.

$$
\begin{aligned}
R &= \int_x (x-m)(x-m)^T\, p(x|z)\, dx \\
&= \int_x \sum_{g\in\mathcal{G}} (x-m)(x-m)^T\, p(x,g|z)\, dx \\
&= \sum_{g\in\mathcal{G}} p(g|z) \int_x (x-m)(x-m)^T\, p(x|g,z)\, dx \\
&= \sum_{g\in\mathcal{G}} p(g|z) \int_x \left(xx^T - 2x \sum_{h\in\mathcal{G}} m_h^T p(h|z) + \sum_{q\in\mathcal{G}}\sum_{h\in\mathcal{G}} m_q m_h^T p(q|z)p(h|z) \right)\, p(x|g,z)\, dx \\
&= \sum_{g\in\mathcal{G}} p(g|z) \left(\int_x xx^T - 2xm_g^T + m_g m_g^T\, p(x|g,z)\, dx + \int_x 2xm_g^T - 2x \sum_{h\in\mathcal{G}} m_h^T p(h|z)\, p(x|g,z)\, dx \right. \\
&\quad \left. -m_g m_g^T + \sum_{q\in\mathcal{G}}\sum_{h\in\mathcal{G}} m_q m_h^T p(q|z)p(h|z) \right) \\
&= \sum_{g\in\mathcal{G}} p(g|z) \left(R_g + 2m_g m_g^T - 2m_g \sum_{h\in\mathcal{G}} m_h^T p(h|z) - m_g m_g^T + \sum_{q\in\mathcal{G}}\sum_{h\in\mathcal{G}} m_q m_h^T p(q|z)p(h|z) \right) \\
&= \sum_{g\in\mathcal{G}} p(g|z) \left(R_g + m_g m_g^T - 2m_g \sum_{h\in\mathcal{G}} m_h^T p(h|z) \right) + \sum_{q\in\mathcal{G}}\sum_{h\in\mathcal{G}} m_q m_h^T p(q|z)p(h|z) \\
&= \sum_{g\in\mathcal{G}} p(g|z) \left(R_g + m_g m_g^T \right) - \sum_{q\in\mathcal{G}}\sum_{h\in\mathcal{G}} m_q m_h^T p(q|z)p(h|z).
\end{aligned}
$$

VITA

Nathan Parrish received his PhD in Electrical Engineering from the University of Washington, Seattle in 2012. During his graduate career, he was awarded a National Defense Science and Engineering Graduate Fellowship and an EE Innovator Fellowship. From 2001 to 2006 he served in the Army, where he was promoted to the rank of Captain. Nathan received his BS degree in Electrical Engineering and graduated with honors from the United States Military Academy in 2001. Nathan grew up in Rusk, Texas where he was a Rusk High School Valedictorian.