Fusion Penalties in Statistical Learning

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Bradley Scott Price

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Charles J. Geyer and
Adam J. Rothman, Advisers

June 2014

Acknowledgements

I would have never been able to complete any of the work contained in this dissertation without the support of my wife, family, friends, and the guidance by my advisers.

I want to thank my teachers and coaches from Boone County Schools for their patience and work, they never gave up on a struggling student. I would also like to thank the faculty at West Virginia University who helped me realize my goal of going to graduate school. A big thank you goes out to my family whose support and encouragement helped get me through graduate school and everything before it. I can not thank my parents and brother enough for their help throughout this process.

My graduate school experience has been amazing, which is due to people I have met and have to come to know while at the University of Minnesota School of Statistics. I specifically want to thank my advisers Dr. Charles Geyer and Dr. Adam Rothman. Working with them the past three years helped me grow as a researcher, teacher, and writer. I have thoroughly enjoyed working with them and appreciate the time and effort they've put into my development. I also want to thank them for allowing me to work on projects that I find interesting, and giving me the freedom in figuring out what I am interested in. I owe them both more gratitude than I would ever be able to express here, I'll truly miss being able to drop in their office to ask questions and pick their brains. All work presented in this document is joint work with both of them. I would like to thank Dr. Galin Jones and Dr. Julian Wolfson for agreeing serve on my preliminary and final dissertation committee.

Finally I would like to thank my wife, Danielle, who has put up with me throughout this process. I couldn't have done any of this without her support.

Abstract

In this dissertation we propose penalized likelihood estimators for use in statistical classification problems. Our main focus is on the development of *fusion penalties*. Chapter 1 presents an introduction to penalized likelihood estimation using fusion penalties. Chapter 2 introduces the ridge fusion method for jointly estimating precision matrices for use in quadratic discriminant analysis and semi-supervised model based clustering. A ridge and ridge fusion penalty are used to introduce shrinkage and promote similarity between the estimates. Blockwise coordinate descent is used for the optimization. Tuning parameter selection is also addressed for both the supervised and semi-supervised settings using cross validation with validation likelihood.

Chapter 3 presents a second method for jointly estimating multiple precision matrices for use in quadratic discriminant analysis, where a common correlation matrix exists between classes. The correlation decomposition of the precision matrix is penalized to create sparse estimates of the common inverse correlation matrix, and a ridge fusion penalty is used to promote similarity of the estimates of the inverse standard deviations of each variable. A two step algorithm is proposed which simultaneously selects tuning parameters for the two penalties. The merits of this method are show through simulations.

In Chapter 4 the we turn from fusion penalties in quadratic discriminant analysis to fusion penalties in multinomial logistic regression. We propose group fused multinomial regression, a novel method for reducing the number of response categories in multinomial logistic regression. An ADMM algorithm is used for optimization and convergence results are established. A line search algorithm and an AIC criterion are developed to select the group structure. A simulation study is presented to show

the ability of group fused multinomial regression to select the correct group structure. Finally Chapter 5 summarizes our work and discusses some future directions. We also provide some insight into the connection between the ridge fusion method proposed in Chapter 2 and fusion penalties in multinomial logistic regression.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

The term *classification* covers a wide variety of statistical problems with applications in many different fields. The goal of classification is to predict a categorical output variable based on a set of input variables. One example would be predicting if a banknote is fraudulent based on measurements of that banknote. Another example is predicting if a user will be interested in reading a document based on previous documents they have looked at and possibly documents that similar users have been interested in.

Classification can be separated into two settings; likelihood based methods and non-likelihood based methods. Commonly used methods such as logistic regression, multinomial regression (Nelder and Wedderburn, 1972), and discriminant analysis (Fisher, 1936) fall into the category of likelihood based methods. Other methods such as support vector machines (Boser and et al., 1992) and tree based methods (Kass, 1980; Breiman, 1996, 2001) fall into non-likelihood based methods. In the following chapters we propose shrinkage estimators for likelihood based classification methods. Specifically we develop penalized likelihood estimators for use in quadratic discriminant analysis and multinomial logistic regression.

1.2 Fusion Penalties

Shrinkage estimators introduce bias to achieve lower mean squared errors. Let $\hat{\theta}$ be an estimator of parameter $\theta \in \mathbb{R}^p$, the mean square error (MSE) is

$$\text{MSE}(\hat{\theta}) = \text{Var}(\hat{\theta}) + \text{Bias}^2(\hat{\theta}, \theta).$$

From this decomposition, we can see that there is a bias variance tradeoff and it is possible that biased estimators have a lower MSE. Throughout the years many variations of shrinkage estimators have been proposed, with most recent focus being on shrinkage estimators derived from penalized likelihoods to obtain sparse estimates for covariance matrices (or inverse covariance matrices) and regression coefficients.. Let $g(\theta)$ be an arbitrary negative log-likelihood. The penalized negative log-likelihood function is

$$g(\theta) + P(\theta),$$

where P is a non-negative real valued function on $\mathbb{R}^p$. The choice of P is used to exploit different structural assumptions. We will focus on penalties of the form

$$P(\theta) = \lambda |\theta|_q^q \tag{1.1}$$

where $|\theta|_q$ is the vector q-norm of θ, and $\lambda \geq 0$ is a tuning parameter. The case where $q = 1$ is called the *lasso penalty*, and $q = 2$, is called the *ridge penalty* (Tibshirani, 1994; Horel and Kennard, 1970). The ridge penalty promotes parameter estimates that are small, but non-zero, while the lasso penalty will promote sparse estimates of θ. Though the ridge and lasso penalties are the most common, other variants of these methods such as the group lasso, where $P(\theta) = \lambda |\theta|_2$, and the elastic net, which

combines the ridge and lasso penalties, have been proposed (Zou and Hastie, 2005; Yuan and Lin, 2006). For further discussion on penalized likelihood and shrinkage estimators in statistical learning we refer the reader to Hastie et al. (2009).

Our research focuses on penalized log-likelihood functions that use *fusion penalties*. This fusion promotes entry-wise similarity in the estimates. Let m and k denote disjoint sets of the elements of θ. The type fusion penalty that will be the focus of later chapters takes the form of

$$P(\theta) = \lambda \sum_{(m,k)\in\mathcal{S}} |\theta_m - \theta_k|., \tag{1.2}$$

where $\mathcal{S}$ is the set of pairs defines sets of elements of θ that are fused together. For instance if the sets of elements of θ have a specific order, we could define

$$\mathcal{S} = \{(m,k); k = (m+1),\ m = 1, 2, \ldots, p-1\}, \tag{1.3}$$

to promote similarity between consecutive elements of θ. The most common choices for the norm in (1.2) are the 1-norm, 2-norm , and the squared 2-norm.

Land and Friedman (1996) provide one of the first applications of fusion penalties in the context of signal regression, where they fuse successive regression coefficients to exploit the spatial and temporal nature of the problem. The authors introduce a q-norm penalty using $\mathcal{S}$ defined by (1.3), in the cases of $q = 0, 1$, and 2. One of the major contributions of the work is that they show that the fused signal regression problem with $q = 1$ can be rewritten as ordinary least squares (OLS) problem using the lasso penalty. Let θ represent the regression coefficients in the regression model, then rewrite $\alpha_j = \theta_j - \theta_{j+1}$ and reformulate the negative log-likelihood as a function of α. The problem becomes a lasso penalized regression of α and is equivalent to fusion of the θ's. This technique is used in many types of fusion penalties when spatial or

temporal dependence can be assumed.

The work of Land and Friedman (1996) was built upon by Tibshirani et al. (2005) with the introduction of the *fused lasso*. The penalty function that the fused lasso uses is

$$P(\theta) = \lambda_1|\theta|_1 + \lambda_2 \sum_{k=1}^{p-1} |\theta_k - \theta_{k+1}|_1, \tag{1.4}$$

where we recognize the term associated with λ_1 as the lasso penalty, and the term associated with λ_2 is the l_1 fusion penalty. The fused lasso promotes both similarity between successive elements and sparsity of the elements. Also, (1.4) shows an example of a penalty that has two tuning parameters that need selected, which makes tuning parameter selection more difficult. This work has been extended by Friedman et al. (2007a), which proposed the fused lasso signal approximator, Liu et al. (2010) which presents an efficient gradient descent algorithm for this problem, and others have provided pathwise algorithms to solve these problems (Hoefling, 2010).

Rinaldo (2009) introduced the adaptive fused lasso which introduces a total variation norm and the maximal one norm to the problem. The authors also propose a method that recovers sparse and block solutions for regression. Another extension is known as the group fused lasso which was introduced by Alaiz et al. (2013). This setting divides θ into groups and then uses l_2 fusion between consecutive groups, while shrinking each group towards zero.

1.3 Example: Fusion of Means

1.3.1 Ridge Fusion Penalty

While understanding the literature of fusion penalties is important it becomes more intuitive to see these methods in action. In this section we present a simple example

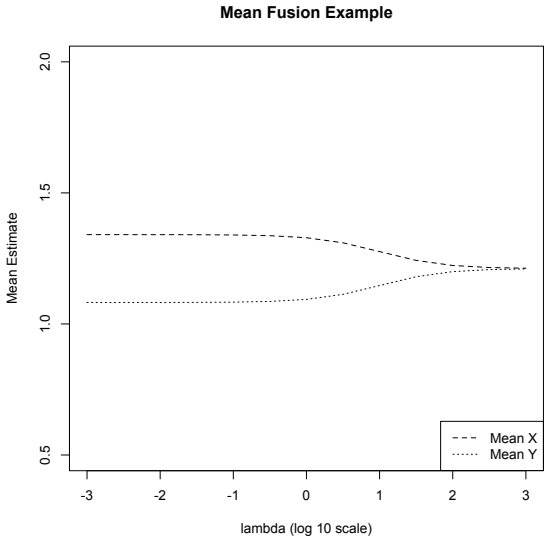

Figure 1.1: Mean estimates for $n = 20$ a two class example using the ridge fusion penalty where $\mu_1 = 1$ and $\mu_2 = 1.5$

to provide intuition into what fusion penalties actually do. Let $x_1, x_2, \ldots, x_n$ be n independent realizations of the random variable $X \sim N(\mu_{1*}, 1)$ and let $y_1, \ldots, y_n$ be a realization of n independent copies of the random variable $Y \sim N(\mu_{2*}, 1)$. We use a ridge fusion penalty to exploit similarity between μ_1 and μ_2. The penalized negative log-likelihood is

$$g(\mu_1, \mu_2) = \frac{1}{2}\left\{\sum_{i=1}^{n}\left((x_i - \mu_1)^2 + (y_i - \mu_2)^2\right) + \lambda|\mu_1 - \mu_2|_2^2\right\} \tag{1.5}$$

The penalized likelihood estimators for μ_{1*} and μ_{2*} are

$$\hat{\mu}_{1,\lambda} = \frac{n\bar{x} + \tilde{\lambda}\bar{y}}{n + \tilde{\lambda}},$$
$$\hat{\mu}_{2,\lambda} = \frac{n\bar{y} + \lambda\hat{\mu}_{1,\lambda}}{n + \lambda},$$

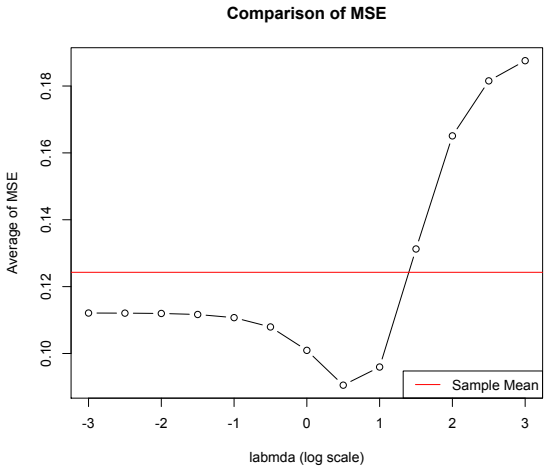

Figure 1.2: Comparison of the average sum of the MSE between the fusion estimates at different values of the tuning parameter and the sample means. The red line indicates the average sum of the MSE for the sample means.

where,

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n},$$
$$\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n},$$
$$\tilde{\lambda} = \frac{n\lambda}{n+\lambda},$$

Figure 1.1 shows the estimated means for a setting where $n = 20$, $\mu_1 = 1$ and $\mu_2 = 1.5$, for possible λ values contained in the set $\{10^m; m = -3, -2.5, \ldots, 2.5, 3\}$. We see that as λ increases the estimates become similar. To investigate the merits of using the fusion estimator over the estimates of the sample mean we compared the mean square error using 10 replications of this process. Figure 1.2 illustrates these results by comparing the average of the sum of the MSE for estimates of μ_{1*} and μ_{2*} for the 10 replications at different values of the tuning parameter, and the average

sum of the MSE using the sample means (denoted on the plot by the red line). The plot shows that for the smaller values of the tuning parameter the average of the sum of the MSE is lower than the average of the sum of the MSE of the sample means.

In this section the ridge fusion penalty, which is convex and differentiable, has been used to keep the example simple. Using other penalties would exploit slightly different characteristics of the model, but the overall message remains the same; fusion penalties exploit similarity.

1.4 An Overview of Our Work

In this chapter we have given a literature review of fusion penalties, and provided some intuition of how they work when applied to a simple two mean estimation problem. Most of the papers we have cited applied to fusion penalties in penalized least squares regression. There has been very little work in applying penalized likelihood methods using fusion penalties in classification. In Chapter 2, Chapter 3, and Chapter 4 we will introduce new methodology that uses fusion penalties in likelihood based classification methods with the goal of creating better classifiers.

In Chapter 2 we present the *ridge fusion method* to jointly estimate inverse covariance matrices for use in quadratic discriminant analysis (Price et al., 2014). We also extend this method and the sparse method of Danaher et al. (2013), called the Fused Graphical Lasso (FGL), to the semi-supervised model based clustering problem. This work also proposes a new cross-validation method using validation likelihood to select the tuning parameters. Another contribution of this work is a novel method to select tuning parameters in the semi-supervised model based clustering framework that uses the unlabeled data. In Section 2.6 we present a detailed simulation study where we investigate when the ridge fusion method will be preferred to its l_1 counterpart FGL, in terms of classification performance and computational

time. We also present a comparison to regularized discriminant analysis (RDA), which is a classical technique used for shrinkage estimation in quadratic discriminant analysis (Friedman, 1989). An R package called `RidgeFusion` is available on CRAN that implements our methods and is available for download on any computer that has an instillation of `R` (Price, 2014).

Chapter 3 introduces a second method to estimate inverse covariance matrices for use in classification with quadratic discriminant analysis and model based clustering. This method exploits an assumption that the inverse covariance matrices for each category are similar, and that each category has the same sparse inverse correlation matrix. In this chapter a unique two-step algorithm that first finds the inverse correlation matrix and the appropriate tuning parameter, and then selects the appropriate tuning parameter for the fusion of the inverse standard deviations. Cross validation using validation likelihood approach is used for tuning parameter selection.

In Chapter 4 we propose a novel penalized likelihood method for exploratory data analysis which has the goal of combining response categories in multinomial logistic regression, by using the group fused penalty. An alternating direction method of multipliers algorithm is proposed for *group fused multinomial logistic regression*, to find the response category combinations. We prove the algorithm converges. The response categories are selected using the solution path given by our group fused multinomial logistic regression algorithm. To select the category combinations an AIC method is developed. A simulation study showing the ability of group fused multinomial regression to select the correct group structure is also presented.

Finally Chapter 5 gives a summary of our work and discusses possible extensions. Specifically we will discuss different fusion penalties that could be used in the context of the multinomial logistic regression model discussed in Chapter 4, and a possible relationship of this model to the QDA model.

Chapter 2

Ridge Fusion in Statistical Learning

2.1 Introduction

Classification by quadratic discriminant analysis (QDA) requires the estimation of multiple inverse covariance matrices. In this model, the data $(x_1, y_1), \ldots, (x_n, y_n)$ are assumed to be a realization of n independent copies of the random pair (X, Y), where Y is supported on $\mathcal{C} = \{1, \ldots, C\}$ and $(X|Y = c) \sim N_p(\mu_{0c}, \Theta_{0c}^{-1})$ for each $c \in \mathcal{C}$. Let $n_c = \sum_{i=1}^{n} 1(y_i = c)$ be the sample size for the cth class, let $\bar{x}_c = n_c^{-1} \sum_{i=1}^{n} x_i 1(y_i = c)$ be the observed sample mean for the cth class, and let

$$S_c = \frac{1}{n_c} \sum_{i=1}^{n} (x_i - \bar{x}_c)(x_i - \bar{x}_c)^T 1(y_i = c), \qquad c \in \mathcal{C},$$

be the observed sample covariance matrix for the cth class. Simply inverting S_c to estimate Θ_{0c} is problematic when n_c is small and impossible when $p \geq n_c$.

The QDA classification rule requires estimates for each of the parameters of $(X|Y = c)$ for all $c \in \mathcal{C}$, and the category probabilities, defined as $P(Y = c) = \pi_c$. Let $\phi(x; \mu, \Theta)$ be the density function of a p-variate normal distribution with mean μ and inverse covariance matrix Θ. Given an x the QDA classification rule predicts

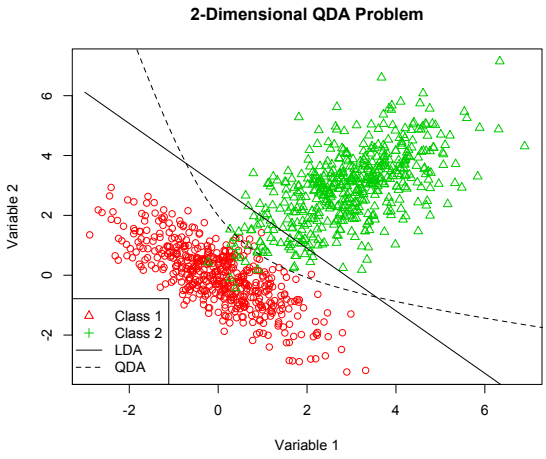

Figure 2.1: A two category example of two multivariate normal distributions with different means and different covariance matrices. The LDA and QDA lines imposed are the decision boundaries.

the category for Y that maximizes

$$P(Y = c|X = x) = \frac{\pi_c \phi(x; \mu_c, \Theta_c)}{\sum_{m \in \mathcal{C}} \pi_m \phi(x; \mu_m, \Theta_m)}. \tag{2.1}$$

The major assumption of QDA is that $\Theta_{0c} \neq \Theta_{0m}$ for atleast one $c, m \in \mathcal{C}$, $c \neq m$ and that Θ_{0c} must be positive definite $\forall c \in \mathcal{C}$. If $\Theta_{01} = \ldots = \Theta_{0C} = \Theta_0$ the problem becomes linear discriminant analysis (LDA). Figure 2.1 shows an example of a two category problem, where there categories are bi-variate normal distributions with different means and different covariance matrices. The parameters are obtained using the maximum likelihood estimates for each of the parameters of the LDA and QDA model using 500 observations from each class. The decision boundaries for the QDA and LDA classifiers are shown on the plot, these are the points where the $P(Y = 1 \mid X = x) = P(Y = 2 \mid X = x)$ for each classification rule. We see that the QDA classifier fits the data better as it is able to detect the different covariance matrices in the model.

In the QDA model, minus 2 times the profile log-likelihood function, profiling over mean and class probability parameters, is

$$g(\Theta_1, \ldots, \Theta_C) = \sum_{c \in \mathcal{C}} n_c \{\text{tr}(S_c \Theta_c) - \log \det(\Theta_c)\}, \tag{2.2}$$

where tr and det are the trace and determinant operators. A number of methods have been proposed to estimate $\Theta_1, \ldots, \Theta_C$ independently and maintain the positive definite constraint. Pourahmadi (2011) reviews several regularized covariance and inverse covariance estimators that could be used to estimate the Θ_{0c}'s, but this would not exploit similarities between them. Similarity between the Θ_{0c}^{-1}'s and low condition numbers for each Θ_{0c} are exploited in regularized discriminant analysis (RDA) (Friedman, 1989), which estimates Θ_{0c} by inverting a linear combination of S_c, the identity matrix, and the observed pooled sample covariance matrix. In the RDA problem two tuning parameters need selected the first is related to the joint estimation and the second controls the shrinkage towards a matrix where the condition number is 1.

In a more general setting, Guo et al. (2011) and Danaher et al. (2013) proposed estimates of $\Theta_{01}, \ldots, \Theta_{0C}$ by minimizing (2.2) plus penalties that promote entry equivalence across the estimates of $\Theta_{01}, \ldots, \Theta_{0C}$ and zero entries within estimates of the Θ_{0c}'s. The penalty of Guo et al. (2011) promoted zero entry equivalence across the inverse covariance estimates and the penalty of Danaher et al. (2013) promoted zero and non-zero entry equivalence of across the inverse covariance estimates. This sparse regularization is aimed at estimating multiple Gaussian graphical models, but is a natural regularization for quadratic discriminant analysis. We propose estimates of $\Theta_{01}, \ldots, \Theta_{0C}$ by minimizing g plus ridge penalties that promote entry-wise similarity between the estimates of the inverse covariance matrices and entry shrinkage for each inverse covariance estimate. This inverse shrinkage is primarily

aimed at improving QDA and is a natural alternative to regularized discriminant analysis, which promotes similarity between estimates of the covariance matrices, which need to be inverted for use in QDA. Our simulations and data examples illustrate cases where our estimators perform competitively in QDA. We show that in settings where the true inverse covariance matrices have large condition numbers and are not similar on the covariance scale our method has better classification performance than RDA. The result of our simulations show that when the estimated inverse covariance matrices are not sparse our algorithm is faster than the algorithm for the sparse solution proposed by Danaher et al. (2013). We also apply our method and the sparse method of Danaher et al. (2013) to semi-supervised model based clustering.

Let $|A|_q$ denote the q-norm of the vector formed from all the entries of the matrix A. Let $\mathbb{S}^p$ denote the set of symmetric $p \times p$ matrices, and let $\mathbb{S}^p_+$ the set of symmetric $p \times p$ positive definite matrices.

Computing our estimates relies on evaluating the function $Q(\cdot;\lambda) : \mathbb{S}^p \to \mathbb{S}^p$ defined by

$$Q(S,\lambda) = \underset{\Theta \in \mathbb{S}^p_+}{\arg\min} \big\{ \mathrm{tr}(\Theta S) - \log\det(\Theta) + \lambda |\Theta|_2^2/2 \big\}. \tag{2.3}$$

Witten and Tibshirani (2009) used the optimization in (2.3) in the context of covariance-regularized regression, where S is an observed sample covariance matrix and λ is a non-negative tuning parameter. For $\lambda > 0$, they derived the closed-form solution

$$Q(S,\lambda) = \frac{1}{2\lambda} V \{ -D + (D^2 + 4\lambda I)^{1/2} \} V^T,$$

where $S = VDV^T$ with V orthogonal and D diagonal. Iterative algorithms that evaluate $Q(\cdot;\lambda)$ include the Fused Graphical Lasso (FGL) algorithm of Danaher et al. (2013)

and an iterative algorithm developed by Rothman and Forzani (2013) that solves a modified version of (2.3) in which the term $\lambda|\Theta|_2^2/2$ is replaced by $\lambda\sum_{i,j} m_{ij}\theta_{ij}^2/2$, where the m_{ij}'s are user- specified non-negative penalty weights.

2.2 Derivation of Ridge Penalized Solution

The derivation of the Q function is critical to understanding the methods we propose here and is a useful derivation technique for methods that estimate the inverse covariance matrix. The Θ that minimizes the objective function in (2.3) satisfies

$$S - \Theta^{-1} + \lambda\Theta = 0. \tag{2.4}$$

Rewriting (2.4) equation we obtain the equality

$$S = \Theta^{-1} - \lambda\Theta. \tag{2.5}$$

Let spectral decomposition of $\Theta = VHV^T$, where V is an orthogonal matrix and H is diagonal. Then (2.5) can be rewritten as

$$S = V(H^{-1} - \lambda H)V^T. \tag{2.6}$$

The problem then becomes an eigenvalue estimation problem, since the eigenvectors of Θ are just the eigenvectors of S. Let the spectral decomposition of $S = VDV^T$, where D is a diagonal matrix. We can then solve for each element of H, H_{ii}, individually. Rewriting (2.6) using only H_{ii} and multiplying both sides by H_{ii} and grouping terms

we see quadratic equation

$$\lambda H_{ii}^2 + D_{ii}H_{ii} - 1 = 0.$$

The solution to this quadratic equation is

$$H_{ii} = \frac{1}{2\lambda}(-D_{ii} + \sqrt{D_{ii}^2 + 4\lambda}).$$

The resulting estimator is

$$\Theta = Q(S, \lambda) = \frac{1}{2\lambda}V\{-D + (D^2 + 4\lambda I)^{1/2}\}V^T.$$

We will discuss more about the Q function and the type of shrinkage it promotes Section 2.4.

2.3 Joint Estimation with Ridge Fusion

2.3.1 Method

We propose the penalized likelihood inverse covariance estimates

$$(\widehat{\Theta}_1, \ldots, \widehat{\Theta}_C) = \underset{\Theta_c \in \mathbb{S}_+^p, c \in \mathcal{C}}{\arg\min} \left\{ g(\Theta_1, \ldots, \Theta_C) + \frac{\lambda_1}{2} \sum_{c \in \mathcal{C}} |\Theta_c|_2^2 + \frac{\lambda_2}{4} \sum_{(c,m) \in \mathcal{C} \times \mathcal{C}} |\Theta_c - \Theta_m|_2^2 \right\}, \tag{2.7}$$

where λ_1 and λ_2 are non-negative tuning parameters. The term multiplied by λ_1 is called the *ridge penalty*, and the term multiplied by λ_2 is called the *ridge fusion penalty*. The former shrinks the elements of each $\widehat{\Theta}_c$ toward zero, and the latter

promotes entry-wise similarity between $\widehat{\Theta}_1, \ldots, \widehat{\Theta}_C$. Although these estimates are not invariant to scaling of the variables, invariance is easily achieved by standardizing the variables and then rescaling appropriately. The objective function in (2.7) is strictly convex, and, if $\lambda_1 > 0$, then a unique global minimizer always exists. We present an algorithm to solve (2.7) in Section 2.3.2 and continue by discussing some special cases.

If $\lambda_2 = 0$, then (2.7) decouples into C separate ridge penalized likelihood problems, the solutions for which are $\widehat{\Theta}_c = Q(S_c, n_c^{-1}\lambda_1)$ for $c \in \mathcal{C}$.

As λ_2 goes to infinity, $(\widehat{\Theta}_1, \ldots, \widehat{\Theta}_C)$ converges to $(\widehat{\Theta}_1^{\bullet}, \ldots, \widehat{\Theta}_C^{\bullet})$ defined to be the solution to (2.7) subject to the constraint $\Theta_1 = \cdots = \Theta_C$, which is

$$\widehat{\Theta}_1^{\bullet} = \cdots = \widehat{\Theta}_C^{\bullet} = \underset{\Theta \in \mathbb{S}_+^p}{\arg\min} \left\{ g(\Theta, \ldots, \Theta) + \frac{\lambda_1}{2} C |\Theta|_2^2 \right\} = Q\left(\frac{1}{n} \sum_{c \in \mathcal{C}} n_c S_c; \frac{\lambda_1 C}{n} \right). \quad (2.8)$$

This "edge case" is important both for computational efficiency — solving (2.7) is computationally unstable when either λ_1 or λ_2 is very large due to the limited precision of computer arithmetic — and because it is itself a parsimonious model appropriate for some data.

Note that our method can be considered the ridge equivalent to the sparse method of Danaher et al. (2013), called the Fused Graphical Lasso (FGL) and jointly estimates the inverse covariance matrix based on the penalized likelihood function

$$g(\Theta_1, \ldots, \Theta_C) + \lambda_2 \sum_{c \in \mathcal{C}} |\Theta_c|_1 + \lambda_2 \sum_{(c,m) \in \mathcal{C} \times \mathcal{C}} |\Theta_c - \Theta_m|_1. \quad (2.9)$$

The authors solution to (2.9) uses an alternating direction method of multipliers algorithm (ADMM) to solve the problem with divide and conquer algorithm to achieve a fast algorithm when solutions are sparse. If it is believed that the true inverse covariance matrices are sparse, and that effects the classification than FGL will be

preferred to ridge fusion.

2.3.2 Algorithm

We solve (2.7) using block-wise coordinate descent. The objective function in (2.7) is

$$f(\Theta_1, \ldots, \Theta_C) = g(\Theta_1, \ldots, \Theta_C) + \frac{\lambda_1}{2} \sum_{c \in \mathcal{C}} |\Theta_c|_2^2 + \frac{\lambda_2}{4} \sum_{(c,m) \in \mathcal{C} \times \mathcal{C}} |\Theta_c - \Theta_m|_2^2 \quad (2.10)$$

with g defined by (2.2). The blockwise coordinate descent step minimizes this with respect to one Θ_c, leaving the rest fixed. This step has a closed-form expression. Differentiating (2.10) with respect to Θ_c and setting the result equal to zero gives

$$n_c(S_c - \Theta_c^{-1}) + \lambda_1 \Theta_c + \lambda_2 \sum_{m \in \mathcal{C} \setminus \{c\}} (\Theta_c - \Theta_m) = 0$$

and dividing through by n_c gives

$$\widetilde{S}_c - \Theta_c^{-1} + \tilde{\lambda}_c \Theta_c = 0, \quad (2.11)$$

where

$$\widetilde{S}_c = S_c - \frac{\lambda_2}{n_c} \sum_{m \in \mathcal{C} \setminus \{c\}} \Theta_m \quad (2.12a)$$

$$\tilde{\lambda}_c = \frac{\lambda_1 + \lambda_2 (C-1)}{n_c} \quad (2.12b)$$

and, since the left-hand side of (2.11) is the same as the gradient of the objective function of (2.3) with S replaced by $\tilde{S}_c$ and λ replaced by $\tilde{\lambda}_c$, the solution to (2.11), considered as a function of Θ_c only, is $Q(\widetilde{S}_c; \tilde{\lambda}_c)$.

Algorithm 1

Initialize a convergence tolerance ε and $\Theta_1, \ldots, \Theta_C$.

Compute $\tilde{\lambda}_c$ using (2.12b).

repeat

for $c \in \mathcal{C}$

Compute $\widetilde{S}_c$ using (2.12a).

Set $\Theta_c^{\text{old}} = \Theta_c$

Set $\Theta_c := Q(\widetilde{S}_c; \tilde{\lambda}_c)$.

end for

until

$\sum_{c \in \mathcal{C}} |\Theta_c^{\text{old}} - \Theta_c|_1 < \varepsilon \sum_{c \in \mathcal{C}} |(S_c \circ I)^{-1}|_1$

end repeat □

The computational complexity of the blockwise descent algorithm is $O(Cp^3)$. The initial iterate for Algorithm 1 could be selected depending on the size of λ_2: when λ_2 is large, one could initialize at the edge-case estimates defined in (2.8); and when λ_2 is small, one could initialize at the solution to (2.7) when $\lambda_2 = 0$.

2.3.3 Tuning Parameter Selection

Traditionally the tuning parameters for QDA are selected by cross validation using the classification error rate of the validation sets. This method is problematic due to the discrete nature of classification error on a fixed sample size, and ignores the use of multivariate normal assumptions in fitting the QDA model. Instead we propose using a validation likelihood to select tuning parameters for (2.7). This is a generalization of its use in the single precision matrix estimation problem (Huang et al., 2006a; Rothman and Forzani, 2013). Randomly split the data into K subsets, dividing each of the C classes as evenly as possible. Let the subscript (v) index objects defined for

the vth subset of the data, and $(-v)$ index those defined for the data with the vth subset removed. The validation likelihood score is

$$V(\lambda_1, \lambda_2) = \sum_{v=1}^{K} \sum_{c \in \mathcal{C}} n_{c(v)} \{ \mathrm{tr}(S_{c(v)} \widehat{\Theta}_{c(-v)}) - \log \det(\widehat{\Theta}_{c(-v)}) \}, \tag{2.13}$$

noting that $\widehat{\Theta}_{c(-v)}$ depends on λ_1 and λ_2 even though the notation does not indicate this. Our selected tuning parameters are $\hat{\lambda}_1$ and $\hat{\lambda}_2$ defined as the values of the tuning parameters that minimize (2.13) over the set of their allowed values.

We suggest this approach not only for the ridge fusion method but for any method that estimates parameters for the QDA model. Since the validation likelihood method requires estimates of multiple inverse covariance matrices K times for each (λ_1, λ_2) method that produce results quickly will be preferred. It is known that RDA is a computationally efficient method, and in later sections we will show a comparison in speed between FGL and ridge fusion to show which of the methods would be preferable.

2.4 Differences between ridge fusion and RDA

To gain some understanding of the difference between our ridge fusion method and RDA, we consider the special case where there is no fusion. For RDA, this means that the coefficient multiplying the pooled sample covariance matrix is 0, so its cth covariance matrix estimate is $(1-\beta)S_c + \beta \bar{d} I$, where $\beta \in [0, 1]$ is a tuning parameter and $\bar{d}$ is the arithmetic mean of the eigenvalues of S_c. Our ridge fusion method without fusion is defined by ((2.7)), with $\lambda_2 = 0$. Decompose $S_c = VDV^T$ with V orthogonal and D diagonal. The cth covariance estimate for ridge fusion without fusion is

$$V \left\{ 0.5D + 0.5 \left(D^2 + 4\lambda_1 n_c^{-1} I \right)^{1/2} \right\} V^T,$$

and the cth covariance estimate for RDA without fusion is $V\left\{(1-\beta)D+\beta\bar{d}I\right\}V^T$. Both estimates have the same eigenvectors as S_c, but their eigenvalues are different. RDA shrinks or inflates the eigenvalues of S_c linearly toward their average $\bar{d}$. Ridge fusion inflates the eigenvalues of S_c nonlinearly, where the smaller eigenvalues of S_c are inflated more than the larger eigenvalues. Figure 2.2 shows an example of the difference in the shrinkage (or inflation) of RDA and ridge fusion on the covariance scale, when the average eigenvalue of a sample covariance matrix is 2.5. The vertical axis of these plots show how much larger the eigenvalues of the estimated covariance matrix when compared to the eigenvalue of the sample covariance matrix. Figure 2.2 illustrates that no matter the size of the tuning parameter, ridge fusion always inflates the eigenvalues of the sample covariance matrix. The smaller eigenvalues are affected most by the tuning parameter of ridge fusion, while larger eigenvalues are increased slightly. RDA on the other hand shrinks eigenvalues of the sample covariance matrix that above the mean eigenvalue and inflates the eigenvalues of the sample covariance matrix that are below the mean eigenvalue towards the mean eigenvalue. The amount of shrinkage is controlled by β. If $\beta=0$ then RDA returns the eigenvalues of the sample covariance matrix if $\beta=1$ then all eigenvalues are the mean eigenvalue.

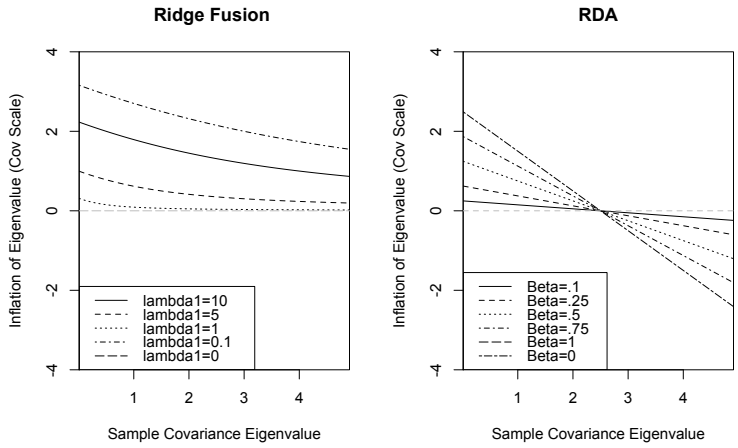

Figure 2.2: A comparison of the eigenvalue inflation of ridge fusion and RDA with no joint estimation. The average eigenvalue of the sample covariance matrix in this example is 2.5.

2.5 Extension to the Semi-Supervised Model

2.5.1 Introduction

Up to this point our discussion has focused on developing estimators where the category of each observations is known, which is called the supervised setting. When the categories of the observations are unknown the problem is called unsupervised. When only some of the labels are known the problem is called semi-supervised and can be considered a hybrid of the supervised and unsupervised settings. A natural extension of any supervised methods is to extend them to the semi-supervised and unsupervised methods. The semi-supervised problem is motivated by situations in which a realization of X is easily obtained, but a realization of Y is costly or difficult to obtain, an introduction of semi-supervised methodology can be found in Chapelle et al. (2006). One of the first examples shown the semi-supervised setting was predicting the gender of halibut based on measurements (Hosmer, 1973).

Traditionally this would be thought of as a supervised problem where the gender of each fish is observed along with the measurements. With halibut the difficulty is the gender can only be observed once the fish is dissected. So it easy to see why semi-supervised methodology will be needed in this case. A good review of the semi-supervised methods can be found in Zhu (2008).

The analogous methods to QDA are unsupervised and semi-supervised model based clustering. An overview of unsupervised model based clustering can be found in Fraley and Raftery (2002) and citations within, and an overview of many clustering methods can be found in Kaufman and Rousseuw (1990).

Just as in classification using QDA, semi-supervised model based clustering with Gaussian mixture models requires estimates for multiple inverse covariance matrices. In the semi-supervised model, let $\mathcal{L}$ and $\mathcal{U}$ be disjoint sets of cardinality n_L and n_U, respectively. The data are random pairs (X_i, Y_i), where for $i \in \mathcal{L}$ both X_i and Y_i are observed but for $i \in \mathcal{U}$ only X_i is observed (Y_i is latent). We denote $\mathcal{D}$ as the observed data. Otherwise, the setup is as in Section 2.3.

Let the conditional probability density function of X_i given $Y_i = c$, which, as in section 2, we assume is Gaussian, be denoted by $\phi_c(\,) = \phi(\,;\mu_c, \Theta_c)$, where μ_c is the mean vector and Θ_c is the inverse covariance matrix. Let π_c denote the probability of $Y_i = c$. And let $\Psi = \{\Theta_1, \ldots, \Theta_c, \mu_1, \ldots, \mu_c, \pi_1, \ldots, \pi_c\}$ denote all the parameters.

Define the log-likelihood for $\mathcal{D}$ with parameters Ψ as

$$l(\Psi) = \sum_{i \in \mathcal{L}} \log\{\pi_{y_i} \phi(x_i; \mu_{y_i}, \Theta_{y_i})\} + \sum_{i \in \mathcal{U}} \log \left\{ \sum_{c \in \mathcal{C}} \pi_c \phi(x_i; \mu_c, \Theta_c) \right\}, \tag{2.14}$$

and the complete data log-likelihood as

$$h(\Psi) = \sum_{i \in \mathcal{L} \cup \mathcal{U}} \log\{\pi_{y_i} \phi_{y_i}(x_i)\}. \tag{2.15}$$

Methods proposed in Ruan et al. (2011), Xie et al. (2008), and Zhou et al. (2009) seek to estimate the parameters of (2.14) using a penalized EM algorithm with assumptions of a specific structure or sparsity on both means and inverse covariances. We propose to estimate these parameters by maximizing (2.14) penalized by ridge or l_1 penalties to create the same kind of shrinkage discussed in Section 2.1 and Section 2.3. We also will address tuning parameter selection by introducing a validation likelihood that utilizes the unlabeled data.

2.5.2 Joint Estimation

Specifically, the penalized log likelihood defined is

$$l(\Psi) - \frac{\lambda_1}{j}\sum_{c\in\mathcal{C}}|\Theta_c|_j^j - \frac{\lambda_2}{j^2}\sum_{(c,m)\in\mathcal{C}\times\mathcal{C}}|\Theta_c-\Theta_m|_j^j, \tag{2.16}$$

over the parameter space for $j = 1, 2$. The case of (2.16) when $j = 1$ introduces the Fused Graphical Lasso (FGL) (Danaher et al., 2013) and when $j = 2$ introduces methodology presented in Section 2.3 in semi supervised model based clustering.

We use the penalized analog of the EM Algorithm to find maximum penalized likelihood estimates of (2.16) (Dempster et al., 1977; Wu, 1983; Green, 1990) . Let $\widehat{\Psi}$ denote the current iterate of the parameter estimates. Then the E-Step of the algorithm calculates

$$\begin{aligned} Q_{\widehat{\Psi}}(\Psi) =& E_{\widehat{\Psi}}\left(h(\Psi) - \frac{\lambda_1}{j}\sum_{c\in\mathcal{C}}|\Theta_c|_j^j - \frac{\lambda_2}{j^2}\sum_{(c,m)\in\mathcal{C}\times\mathcal{C}}|\Theta_c-\Theta_m|_j^j \,\middle|\, \mathcal{D}\right), \\ =& \sum_{i\in\mathcal{L}}\log\{\pi_{y_i}\phi_{y_i}(x_i)\} + \sum_{i\in\mathcal{U}}\sum_{c\in\mathcal{C}}\alpha_{ic}\log\{\pi_c\phi_c(x_i)\} \\ & - \frac{\lambda_1}{j}\sum_{c\in\mathcal{C}}|\Theta_c|_j^j - \frac{\lambda_2}{j^2}\sum_{(c,m)\in\mathcal{C}\times\mathcal{C}}|\Theta_c-\Theta_m|_j^j, \end{aligned} \tag{2.17}$$

where

$$\alpha_{ic} = \frac{\phi(x_i; \hat{\mu}_c, \hat{\Theta}_c)\hat{\pi}_c}{\sum_{m \in \mathcal{C}} \phi(x_i; \hat{\mu}_m, \hat{\Theta}_m)\hat{\pi}_m}, \qquad i \in \mathcal{U} \text{ and } c \in \mathcal{C}. \tag{2.18}$$

The M-Step of the algorithm calculates $\widehat{\Psi}$ that maximizes (2.17) with respect to Ψ. Define

$$\begin{aligned}
\tilde{n}_c &= \sum_{i \in \mathcal{L}} 1(y_i = c) + \sum_{i \in \mathcal{U}} \alpha_{ic} \\
\tilde{\pi}_c &= \frac{\tilde{n}_c}{n_L + n_U}, && (2.19) \\
\tilde{\mu}_c &= \frac{\sum_{i \in \mathcal{L}} x_i 1(y_i = c) + \sum_{i \in \mathcal{U}} \alpha_{ic} x_i}{\tilde{n}_c} && (2.20) \\
\tilde{S}_c^{(L)} &= \frac{\sum_{i \in \mathcal{L}} 1(y_i = c)(x_i - \tilde{\mu}_c)(x_i - \tilde{\mu}_c)^T}{n_c}, \\
\tilde{S}_c^{(U)} &= \frac{\sum_{i \in \mathcal{U}} \alpha_{ic}(x_i - \tilde{\mu}_c)(x_i - \tilde{\mu}_c)^T}{\sum_{i \in \mathcal{U}} \alpha_{ic}}, \\
\tilde{S}_c &= \frac{n_c \tilde{S}_c^{(L)} + (\sum_{i \in \mathcal{U}} \alpha_{ic}) \tilde{S}_c^{(U)}}{\tilde{n}_c}.
\end{aligned}$$

Then the profile of the negative penalized complete data log-likelihood for the Θ's replacing μ_c with $\tilde{\mu}_c$ and π_c with $\tilde{\pi}_c$ is

$$\sum_{c \in \mathcal{C}} \tilde{n}_c \left\{ \text{tr}(\tilde{S}_c \Theta_c) - \log \det(\Theta_c) \right\} + \frac{\lambda_1}{j} \sum_{c \in \mathcal{C}} |\Theta_c|_j^j + \frac{\lambda}{j^2} \sum_{(c,m) \in \mathcal{C} \times \mathcal{C}} |\Theta_c - \Theta_m|_j^j, \tag{2.21}$$

and maximizing this subject to $\Theta_c \in \mathbb{S}_+^p$ gives estimates of the Θ's for the next iteration, estimates of the other parameters for the next iteration being given by (2.19) and (2.20). In the $j = 1$ case of (2.21) solutions are found by the FGL algorithm (Danaher et al., 2013), and in the $j = 2$ case solutions are found by our coordinate descent algorithm (Algorithm 1). In our current implementation, both algorithms are run until convergence.

All of the steps above are repeated until the penalized EM (PEM) algorithm converges. Our convergence criterion here is similar to the one in Section 2.3.2, in particular, we are using the difference in the α's from iteration to iteration. Green (1990) gives convergence rates for the penalized EM algorithm that vary with the proportion of unlabeled data (the more unlabeled data the worse the convergence). Thus PEM should work well when the proportion of the unlabeled data is not too large. The initial estimates for our EM algorithm are obtained from the labeled data (Basu et al., 2002).

An alternative is to not iterate to convergence in the optimization of (2.21). Dempster et al. (1977) call a variant of the EM algorithm in which the M-step is not iterated to convergence but does make progress (goes uphill on the function it is optimizing) a generalized EM (GEM) algorithm and Wu (1983) proves this also converges to the MLE (under certain conditions). The analog here, not iterating the M-step to convergence, is penalized generalized EM (PGEM) and should also converge to the penalized maximum likelihood estimate, although we have not investigated this.

2.5.3 Validation Likelihood for Tuning Parameter Selection

In the semi-supervised setting it is not uncommon to have data in which the labeled sample size for each class is so small that it would not be practical to use the validation likelihood presented in Section 2.3.3 to select the tuning parameters. To address this we propose a validation likelihood that uses both labeled and unlabeled data. Define the negative log-likelihood of the observed data, $\mathcal{D}$, with parameters Ψ to be

$$L_{\mathcal{D}}(\Psi) = -\sum_{i\in\mathcal{L}} \log\{\pi_{y_i}\phi(x_i;\mu_{y_i},\Theta_{y_i})\} - \sum_{i\in\mathcal{U}} \log\left\{\sum_{c\in\mathcal{C}} \pi_c\phi(x_i;\mu_c,\Theta_c)\right\}. \quad (2.22)$$

Similar to the supervised case, randomly split the labeled and unlabeled data into K subsets. We define $\mathcal{L}_{(v)}$ and $\mathcal{U}_{(v)}$ to be the indices of the vth subset of the labeled

and unlabeled data, then let $\mathcal{D}_{(v)}$ be the vth subset of the data. Let $\widehat{\Psi}_{(-v)}$ denote the parameter estimates resulting from the semi-supervised model based clustering on the data with the vth subset of the data removed.

Our validation likelihood will evaluate $L_{\mathcal{D}_{(v)}}(\widehat{\Psi}_{(-v)})$, which is the negative log-likelihood for the vth subset of the data with parameters $\widehat{\Psi}_{(-v)}$, for each $v \in 1, \ldots, K$. The validation score is then defined as

$$V(\lambda_1, \lambda_2) = \sum_{v=1}^{K} L_{\mathcal{D}_{(v)}} \left(\widehat{\Psi}_{(-v)} \right) \tag{2.23}$$

where $\widehat{\Psi}_{(-v)}$ are the parameter estimates based on λ_1 and λ_2 though the notation does not say this specifically. We select the tuning parameters $(\hat{\lambda}_1, \hat{\lambda}_2)$ that minimize (2.23) over the set of allowed values to be the final tuning parameter.

2.6 Simulations

2.6.1 Regularization in quadratic discriminant analysis

We present simulation studies that compare the classification performance of QDA in which RDA, FGL, and the ridge fusion methods are used to estimate the inverse covariance matrices.

The data generating model and performance measurements

In the following simulations described in sections Section 2.6.1 – Section 2.6.1, we generated data from a two-class model where the class 1 distribution was $N_p(\mu_1, \Sigma_1)$ and the class 2 distribution was $N_p(\mu_2, \Sigma_2)$. We considered $p = 50$ and $p = 100$. The training data had 25 independent draws from the class 1 distribution and 25 independent draws from the class 2 distribution. These training observations were

used to compute parameter estimates. These estimates were used in QDA to classify observations in an independent testing dataset consisting of 500 independent draws from the class 1 distribution and 500 independent draws from the class 2 distribution. We measured performance with the classification error rate (CER) on these testing cases. This process was replicated 100 times.

The tuning parameters λ_1 and λ_2 for FGL and the ridge fusion estimates of Σ_1^{-1} and Σ_2^{-1} were selected from a subset of $\{10^x : x = -10, -9.5, \ldots, 9, 9.5, 10\}$ using the method described in section Section 2.3.3 unless otherwise stated. Specific subsets were determined from pilot tests for each simulation. An R package, `RidgeFusion`, implementing the ridge fusion and tuning parameter selection methods is available on CRAN (Price, 2014).

RDA tuning parameter selection simulation

In this simulation, we compared two cross-validation procedures to select tuning parameters for the RDA estimators of Σ_1^{-1} and Σ_2^{-1}. The first procedure minimizes the validation CER and the second maximizes the validation likelihood, as described in section Section 2.3.3. Weihs et al. (2005), in the documentation of the R package `klaR`, mentioned that cross validation minimizing validation CER is unstable when sample sizes are small. We used the klaR package to perform RDA with tuning parameter selection that minimizes validation CER, and we used our own code to perform RDA with tuning parameter selection that maximizes validation likelihood.

We set all elements of μ_1 to $5p^{-1}\log(p)$ and made μ_2 the vector of zeros. We generated Σ_1 and Σ_2 to have the same eigenvectors, which were the right singular vectors of the 100 by p matrix with rows independently drawn from $N_p(0, I)$. The jth eigenvalue of Σ_1 is

$$100\frac{p-j+1}{p}\mathrm{I}\{1 \leq j \leq 6\} + 10\frac{p-j+1}{p}\mathrm{I}\{7 \leq j \leq 11\} + \frac{p-j+1}{p}\mathrm{I}\{12 \leq j \leq p\}.$$

Table 2.1: Average CER for RDA reported with standard errors based on 100 independent replications for the simulation described in section Section 2.6.1 (the RDA tuning parameter selection simulation).

	$p = 20$	$p = 50$	$p = 100$
Validation Likelihood	0.02 (0.01)	0.05 (0.02)	0.13 (0.04)
Cross Validation with CER	0.07 (0.03)	0.09 (0.03)	0.13 (0.04)

The jth eigenvalue of Σ_2 is

$$500\frac{p-j+1}{p}\mathrm{I}\{1 \leq j \leq 6\} + 50\frac{p-j+1}{p}\mathrm{I}\{7 \leq j \leq 11\} + \frac{p-j+1}{p}\mathrm{I}\{12 \leq j \leq p\}.$$

We investigated cases where $p = 20, 50, 100$. The results of this simulation, found in Table Table 2.1, indicate that cross validation maximizing validation likelihood outperforms cross validation minimizing CER. This lead us to tune RDA with the validation likelihood method in the remaining simulation studies.

Dense, ill conditioned, and unequal inverse covariance matrices simulation: part 1

This simulation uses the parameter values described in section Section 2.6.1 to compare the QDA classification performance of FGL, RDA, and the ridge fusion methods. Since Σ_1^{-1} and Σ_2^{-1} are dense, it is unclear which method should perform the best. Based on section Section 2.4, we expect that RDA will perform poorly because Σ_1 and Σ_2 are ill conditioned. Table Table 2.2 has the average CER and corresponding standard errors. The ridge fusion method outperforms RDA. We also see that ridge fusion and FGL perform similarly.

Table 2.2: Average CER for QDA with standard errors based on 100 independent replications for the simulation described in section Section 2.6.1 (the dense, ill conditioned, and unequal inverse covariance matrices simulation: part 1).

	$p = 50$	$p = 100$
RDA	0.05 (0.02)	0.13 (0.04)
Ridge	0.03 (0.02)	0.08 (0.03)
FGL	0.03 (0.02)	0.09 (0.02)

Table 2.3: Average CER for QDA reported with standard errors based on 100 replications for the simulation described in section Section 2.6.1 (the dense, ill conditioned, and unequal inverse covariance matrices simulation: part 2).

	$p = 50$	$p = 100$
Ridge	0.00 (0.00)	0.00 (0.00)
RDA	0.16 (0.04)	0.31 (0.05)
FGL	0.00 (0.00)	0.00 (0.00)

Dense, ill conditioned, and unequal inverse covariance matrices simulation: part 2

In this simulation, Σ_1 has (i, j)th entry $0.5 \cdot 1(|i-j| = 1) + 1(i = j)$, and Σ_2 is defined in section Section 2.6.1. We set each element in μ_1 to p^{-1} and each element of μ_2 to be zero. We expect RDA to perform poorly because of the large condition numbers and lack of similarity between Σ_1 and Σ_2. The average classification error rate is reported in Table Table 2.3, where we see that ridge fusion and FGL outperform RDA for both values of p.

Sparse, well conditioned, and equal inverse covariance matrices simulation

In this simulation we set $\Sigma_1 = \Sigma_2 = I$ and all elements of μ_1 to $10p^{-1}\log(p)$ and all elements of μ_2 to zero. The average CER, based on 100 replications, is reported

Table 2.4: Average CER for QDA reported with standard errors based on 100 independent replications for the simulation described in section Section 2.6.1 (the sparse, well conditioned, and equal inverse covariance matrices simulation).

	$p = 50$	$p = 100$
RDA	0.01 (0.01)	0.03 (0.02)
Ridge	0.01 (0.01)	0.04 (0.02)
FGL	0.01 (0.01)	0.03 (0.02)

in Table Table 2.4: all three methods perform similarly when $p = 50$ and the ridge fusion method is outperformed by RDA and FGL when $p = 100$.

Sparse and similar inverse covariance matrices simulation

In this simulation, Σ_1 is block diagonal with two equal size blocks: the (i, j)th entry in the first block was $0.95^{|i-j|}$ and the (k, m)th entry in the second block was $0.8^{|k-m|}$. We also made Σ_2 block diagonal with two equal size blocks: the (i, j)th entry in the first block was $0.95^{|i-j|}$ and the (k, m)th entry in the second block was $\rho^{|k-m|}$, where $\rho = 0.25$, 0.50, and 0.95. This setting should favor FGL, which exploits the sparsity in Σ_1^{-1} and Σ_2^{-1}. We set each element in μ_1 to $20p^{-1}\log(p)$ and each element in μ_2 to zero. The classification performance is reported in Table Table 2.5. We see that FGL outperforms the other two methods for $\rho = 0.25, 0.50$ and all values of p. When $\rho = 0.95$, even though the covariance matrices are ill conditioned, RDA outperforms the ridge fusion method and FGL for both values of p.

Inverse covariance matrices with small entries simulation

In this simulation, Σ_1 has (i, j)th entry $0.4 \cdot 1(|i - j| = 1) + 1(i = j)$ and Σ_2 has (i, j)th entry $\rho \cdot 1(|i - j| = 1) + 1(i = j)$, where $\rho = 0.25, 0.30, 0.35$ and 0.50. We set each element in μ_1 to $10\log(p)p^{-1}$ and each element in μ_2 to zero. The

Table 2.5: Average CER for QDA reported with standard errors based on 100 independent replications for the simulation described in section Section 2.6.1 (the sparse and similar inverse covariance matrices simulation).

	ρ	$p = 50$	$p = 100$
RDA		0.10 (0.03)	0.21 (0.04)
Ridge	0.95	0.13 (0.04)	0.24 (0.04)
FGL		0.11 (0.03)	0.21 (0.04)
RDA		0.08 (0.03)	0.20 (0.04)
Ridge	0.50	0.06 (0.02)	0.13 (0.04)
FGL		0.04 (0.02)	0.09 (0.03)
RDA		0.06 (0.02)	0.15 (0.04)
Ridge	0.25	0.05 (0.02)	0.12 (0.03)
FGL		0.03 (0.02)	0.06 (0.02)

classification results are reported in Table Table 2.6, and show that RDA has the best classification performance for each value of p and ρ. Note that FGL has the same average classification error rate as RDA in the case where $p = 50$ when $\rho = 0.30$ and 0.35.

2.6.2 Computing time simulations: ridge fusion versus FGL

Although FGL performed as well or better than our ridge fusion method at classification in the simulations of sections Section 2.6.1 – Section 2.6.1, we found that computing FGL is much slower than our ridge fusion method when a dense estimate is desired. We present three timing simulations that illustrate this pattern. In each simulation we measured the computing time (in seconds) of ridge fusion and FGL, calculated by the R function `system.time`, where the tuning parameters (λ_1, λ_2) are selected from $\Lambda \times \Lambda$, where $\Lambda = \{10^x : x = -8, -7, \ldots, 7, 8\}$ and $p = 100$. We report the average of the difference in computing time between ridge fusion and FGL based on 100 independent replications, for each point in $\Lambda \times \Lambda$. FGL and ridge fusion were

Table 2.6: Average CER for QDA reported with standard errors based on 100 replications for the simulation described in section Section 2.6.1 (the inverse covariance matrices with small entries simulation).

	ρ	$p = 50$	$p = 100$
Ridge	0.25	0.04 (0.02)	0.09 (0.03)
RDA		0.02 (0.01)	0.06 (0.02)
FGL		0.03 (0.02)	0.07 (0.03)
Ridge	0.30	0.04 (0.02)	0.09 (0.03)
RDA		0.03 (0.02)	0.06 (0.02)
FGL		0.03 (0.02)	0.08 (0.03)
Ridge	0.35	0.04 (0.02)	0.10 (0.03)
RDA		0.03 (0.02)	0.07 (0.03)
FGL		0.03 (0.02)	0.08 (0.03)
Ridge	0.50	0.06 (0.02)	0.11 (0.03)
RDA		0.03 (0.02)	0.09 (0.03)
FGL		0.04 (0.02)	0.10 (0.03)

computed using the `JGL` (Danaher, 2013) and `RidgeFusion` (Price, 2014) R packages with default settings.

In each simulation setting, the ridge fusion algorithm is faster than FGL when λ_1 is small, and FGL is faster than ridge fusion when λ_1 is large. This result is not surprising as a large λ_1 when using FGL will produce sparse estimates of the inverse covariance matrices, which the algorithm exploits in estimation by using a divide and conquer algorithm. In summary, FGL will be faster when the true inverse covariance matrices are quite sparse and otherwise ridge fusion will be faster.

Dense, ill conditioned, and different inverse covariance matrices timing simulation

This simulation investigates the difference of the average speed over 100 replications of FGL and ridge fusion using the data generating model described in section Sec-

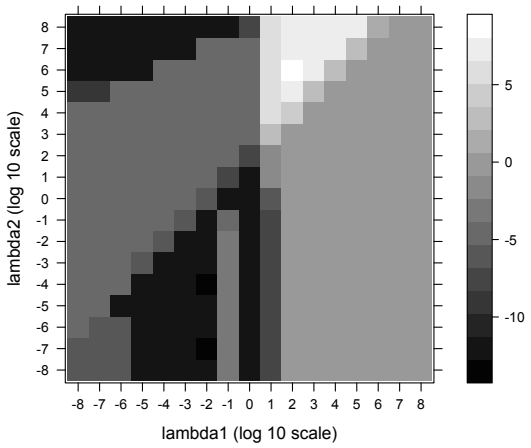

Figure 2.3: Difference of average computing time for Timing Simulation 1 for 100 replications at each grid point. Negative values represent where ridge fusion is faster than FGL.

tion 2.6.1 and parameter values used in section Section 2.6.1. The results are shown in in Figure Figure 2.3. The ridge fusion method is faster or comparable to FGL when λ_1 is small and otherwise FGL is faster. Over the entire grid we find that, on average, ridge fusion is 4 seconds faster than FGL. At one grid point, ridge fusion was 534 times faster than FGL and at another grid point FGL was 73 times faster than ridge fusion.

Sparse and similar inverse covariance matrices timing simulation

This simulation uses the data generating model described in section Section 2.6.1 and parameter values used in section Section 2.6.1 with $\rho = 0.95$ and $p = 100$. Here FGL performs much better than ridge fusion in classification. The results shown in Figure Figure 2.4 are the difference of the average computing time of FGL and ridge fusion. These are similar to those of section Section 2.6.2. As expected, ridge fusion is faster or comparable to FGL when λ_1 is small and otherwise FGL is faster. Averaging

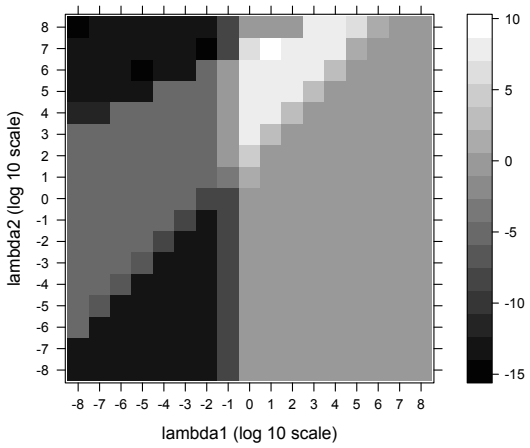

Figure 2.4: Difference of average computing time for Timing Simulation 2 for 100 replications at each grid point. Negative values represent where ridge fusion is faster than FGL.

across the grid, ridge fusion was approximately 5 seconds faster. At the extremes, there was one grid point at which ridge fusion was 564 times faster than FGL, and another point at which FGL was 63 times faster than ridge fusion.

Inverse covariance matrices with small entries timing simulation

This simulation uses the data generating model described in section Section 2.6.1 and parameter values used in section Section 2.6.1 when $\rho = 0.50$ and $p = 100$. The results in Figure Figure 2.5 show a similar result to the other timing simulations in sections Section 2.6.2 and Section 2.6.2: ridge fusion is faster or comparable to FGL when λ_1 is small and otherwise FGL is faster. We find that on average over the entire grid that ridge fusion is on average 5 seconds faster. At the extremes, there was one point on the grid where ridge fusion was 595 times faster than FGL and another point on the grid where FGL was 322 times faster than ridge fusion.

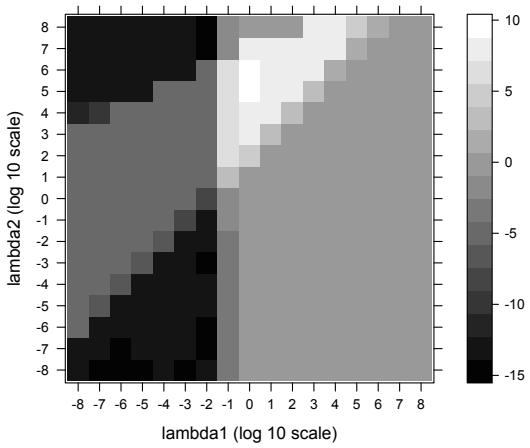

Figure 2.5: Difference of average computing time for Timing Simulation 3 for 100 replications at each grid point. Negative values represent where ridge fusion is faster than FGL.

2.6.3 Regularization in semi-supervised model based clustering

We evaluate the semi-supervised model based clustering methods proposed in section Section 2.5 by comparing the tuning parameter selection methods proposed in sections Section 2.5.3 and Section 2.3.3. This simulation uses the same data generating model as that used in the simulation study in section Section 2.6.1 with $\rho = 0.25$. Each replication will have 25 labeled and 250 unlabeled observations from each class. We compare the ridge fusion and FGL methods for semi-supervised model based clustering on their ability to classify the unlabeled data for 50 independent replications when the tuning parameters are selected using the labeled data only via the methodology proposed in section Section 2.5.3. For each replication the QDA classification rule is formed by using the corresponding parameter estimates from the regularized semi-supervised model based clustering. Results of this simulation are contained in Table Table 2.7 and show that using the method presented in section

Table 2.7: Average CER reported with standard errors for the semi-supervised model based clustering simulation based on 50 independent replications.

	$p = 50$	$p = 100$
Ridge	0.01 (0.01)	0.07 (0.04)
Ridge Labeled	0.02 (0.02)	0.22 (0.06)
FGL	0.01 (0.01)	0.01 (0.01)
FGL Labeled	0.04 (0.03)	0.31 (0.07)

Section 2.5.3 to select the tuning parameter outperforms the method that ignores the unlabeled data.

2.7 Data Example

We compare ridge fusion, FGL and RDA on the Libras movement data from the UCI Machine Learning Repository, which describes the hand movements in Brazilian sign language (Bache and Lichman, 2013). The original data has 15 classes corresponding to the type of hand movements, with 90 variables that represent 45 different time points in a video that shows the hand movement. The variables represent where the hand is in the frame at a given time point. For this example we selected 3 classes that correspond to the movements of curved swing, horizontal swing, and vertical swing. We have taken 18 observations from each class for training while keeping 45 observations from each class for validation. Tuning parameters for each method were selected using 3-fold validation likelihood due to the small sample size of the training set. The results of this analysis are reported in Table Table 2.8, and show that the ridge fusion method outperforms FGL and RDA with regard to the number of observations classified incorrectly for the validation data.

We also apply the methodology from section Section 2.5 on the Libras movement data where the 45 validation points from each class are treated as the unlabeled data.

Table 2.8: Fraction of the validation data that is classified incorrectly for the Libra data example.

	Fraction of Data Misclassified
Ridge	0/135
FGL	9/135
RDA	4/135

Table 2.9: Fraction of the unlabeled data that is classified incorrectly using semi-supervised model based clustering methods for the Libra data example.

	Fraction of Unlabeled Data Misclassified
Ridge	0/135
FGL	5/135

Again we use a 3-fold validation likelihood based on the method proposed in section Section 2.5.3. Table Table 2.9 contains the results. As we saw in the supervised case, the ridge fusion method has a smaller number of observations classified incorrectly when compared to FGL on the unlabeled data.

Chapter 3

A Common Correlation Extension

3.1 Common Correlation Model

The ridge fusion method proposed in Chapter 2 exploits the similarity of the elements of the inverse covariance matrices by using the ridge fusion penalty. In this chapter instead of exploiting element-wise similarity of the estimates of inverse covariance matrices, we choose to exploit similarity through the correlation decomposition of the inverse covariance matrix. We will assume the same data generating model as described in Section 2.1, where $\text{Var}(X|Y=c) = \Theta_{0c}^{-1} = (D_{0c}\Omega_0 D_{0c})^{-1}$, such that Ω_0^{-1} is the correlation matrix that is the same for all c. D_{0c}^{-1} is a diagonal matrix, where $\text{diag}(D_{0c}^{-1}) = (\sigma_{1c}, \ldots, \sigma_{pc})$, and σ_{jc} is the standard deviation of the jth element of the random vector from $(X|Y=c)$. Two times the negative log-likelihood of this model is

$$h(\Omega, D_1, \ldots, D_C) = \sum_{c\in\mathcal{C}} n_c \left\{\text{tr}(S_c D_c \Omega D_c) - 2\log\det(D_c) - \log\det(\Omega)\right\}. \quad (3.1)$$

Just as in Chapter 2 our goal is to estimate the inverse covariance matrices for use in QDA, which requires the estimates to be positive definite. Let $\mathbb{C} = \{A \in \mathbb{S}_+^p; \text{diag}(A^{-1}) = \vec{1}\}$, be the set of all positive definite inverse correlation matrices,

and $\mathcal{D} = \{A \in \mathbb{S}_+^p; A_{ij} = 0, \forall\, i \neq j\}$, where $\mathbb{S}_+^p$ is defined the same as in Chapter 2. We propose the inverse covariance matrix estimates $\widetilde{\Theta}_c = \widehat{D}_c \widehat{\Omega} \widehat{D}_c$, where

$$(\widehat{\Omega}, \widehat{D}_1, \ldots, \widehat{D}_C) = \underset{\Omega \in \mathbb{C},\, D_k \in \mathcal{D}}{\arg\min}\; h(\Omega, D_1, \ldots, D_C) + \lambda_1 |\Omega^-|_1 + \frac{\lambda_2}{2} \sum_{(c,m) \in \mathcal{C} \times \mathcal{C}} |D_c - D_m|_2^2, \tag{3.2}$$

where Ω^- is the matrix Ω with the diagonal elements set equal to zero. The penalty function $|\Omega^-|_1$ is a common penalty used to estimate the inverse covariance matrix and corresponds to penalizing only the off diagonal elements of Ω. Many proposals, described in Pourahmadi (2011) such as Huang et al. (2006b), Friedman et al. (2007b), and Rothman et al. (2008) have proposed estimating a single sparse inverse covariance matrix matrix using the penalized likelihood

$$\mathrm{tr}(S_c \Theta_c) - \log\det(\Theta_c) + \lambda \mid \Theta_c^- \mid_1 . \tag{3.3}$$

The most common algorithm to solve the penalized likelihood in (3.3) is called the graphical lasso, also called glasso (Friedman et al., 2007b). The glasso algorithm uses a coordinate descent algorithm to solve for the elements of the inverse covariance matrix. The major impact of this work is that the authors show that each coordinate descent step is a lasso regression of p variables. Define the graphical lasso solution to be

$$H(S_c, \lambda) = \underset{\Theta_c \in S_+^p}{\arg\min} \left\{ \mathrm{tr}(S_c \Theta_c) - \log\det(\Theta_c) + \lambda \mid \Theta_c^- \mid_1 \right\}. \tag{3.4}$$

Other variants of this solution have been proposed by Witten et al. (2011), which propose a sufficient condition for finding a block diagonal structure to create a faster algorithm. A review of the literature on the glasso algorithm can be found

in Mazumder and Hastie (2012).

Ideally we would like to estimate $D_1, \ldots, D_C$ and Ω jointly, but another option is to first estimate Ω, and use that estimate when estimating $D_1, \ldots, D_C$. We propose the latter where a two step algorithm is used with simultaneous tuning parameter selection. The first step of the algorithm is to estimate a sparse positive definite inverse correlation matrix and choose the corresponding tuning parameter. The second step of the algorithm estimates the inverse standard deviations and chooses the corresponding tuning parameter to the fusion penalty. Since a lasso penalty is used to estimate the inverse correlation matrix, then each inverse covariance matrix will have the same sparsity structure, that is they will have the same elements that are set equal to zero. A ridge penalty could also be used on the inverse correlation similar to that proposed in Rothman and Forzani (2013), just as a l_1 fusion penalty could be used to estimate the inverse standard deviations we will consider both of these settings in future work.

3.2 Common Correlation Algorithm

3.2.1 Penalized Likelihood Solution: Correlation

The first step of the algorithm requires solving the objective function in (3.4) for the optimal Ω for some fixed $\lambda_1 \geq 0$. The diagonal of Ω is not penalized due to the interpretation of Ω^{-1} as a correlation matrix. Let d_{cj} be the jth diagonal element of D_c where we initialize it to be $d_{cj} = \sqrt{1/s_{c,jj}}$, where $s_{c,jj}$ is the jth diagonal element of S_c. For a fixed $\lambda_1, \lambda_2 \geq 0$ let

$$L(\Omega, \lambda_1) = \sum_{c=1}^{C} n_c\{\text{tr}(S_c D_c \Omega D_c) - \log\det(\Omega)\} + \lambda_1 |\Omega^{-}|_1 \tag{3.5}$$

be the part of the function defined by (3.1) dependent on Ω. Define ω_{ij} to be the i,jth element of Ω, and Γ to be a $p \times p$ matrix where γ_{ij} is the i,jth element of Γ. Let $R = \frac{\sum_{c=1}^{C} n_c D_c S_c D_c}{n}$ and let $\tilde{\Omega}$ minimize $L(\Omega, \lambda_1)$ for a $\lambda_1 \geq 0$ then

$$R - \tilde{\Omega}^{-1} + \frac{\lambda_1}{n}\Gamma = 0, \tag{3.6}$$

where $\gamma_{ij} = \text{sign}(\omega_{ij})$ if $\omega_{ij} \neq 0$, and $\gamma_{ij} \in [-1, 1]$ if $\omega_{ij} = 0$. The resulting estimator is $\tilde{\Omega} = H(R, n^{-1}\lambda_1)$. Since the diagonal is not penalized the resulting estimates will be the inverse of the common correlation matrix of the categories.

3.2.2 Penalized Likelihood Solution: Fused Inverse Standard Deviations

The second step of the algorithm, fusing the inverse standard deviations, is not as straight forward as estimating the common correlation matrix. We propose using a coordinate descent algorithm to find the estimates for $D_1, \ldots, D_C$. To simplify the derivation of the estimator we reformulate the penalized negative log-likelihood using the identities,

$$\text{tr}(S_c D_c \Omega D_c) = \text{tr}(D_c S_c D_c \Omega) = \sum_{m=1}^{p}\sum_{l=1}^{p} d_{mm} s_{ml} d_{ll} \omega_{lm}, \tag{3.7}$$

and

$$\log\det(D_c) = \sum_{j=1}^{p} \log(d_{cj}). \tag{3.8}$$

We also assume Ω is fixed and positive definite for this discussion.

Let $D = \{D_1, \ldots, D_C\}$ and

$$G(D, \lambda_2) = \sum_{c \in \mathcal{C}} n_c \left\{ \operatorname{tr}(S_c D_c \Omega D_c) - 2 \log \det(D_c) \right\} + \frac{\lambda_2}{2} \sum_{(c,m) \in \mathcal{C} \times \mathcal{C}} |D_c - D_m|_2^2, \quad (3.9)$$

be the terms of (3.1) that are dependent on D.

The coordinate descent algorithm we propose solves for each d_{cj} individually and treats all others as fixed. The gradient of (3.9) with respect to d_{cj} is

$$\partial_{d_{cj}} G(D, \lambda_2) = 2n_c (d_{cj} s_{c,jj} \omega_{jj} + \sum_{l=1, j \neq l}^{p} d_{cl} s_{c,jl} w_{jl} - d_{cj}^{-1}) + 2\lambda_2 \sum_{m \in \mathcal{C}} (d_{cj} - d_{mj}). \quad (3.10)$$

Solving the zero gradient of (3.10) is equivalent to finding the $\hat{d}_{cj}$ that satisfies

$$a \hat{d}_{cj}^2 + b \hat{d}_{cj} - 1 = 0, \quad (3.11)$$

where

$$a = s_{c,jj} \omega_{jj} + \frac{\lambda_2 (C-1)}{n_c}, \quad (3.12)$$

$$b = \sum_{l=1, l \neq j}^{p} d_{cl} s_{c,jl} \omega_{jl} - \frac{\lambda_2}{n_c} \sum_{m \in \mathcal{C} \setminus \{c\}} d_{mj}. \quad (3.13)$$

The estimate for d_{cj} is

$$\hat{d}_{cj} = \frac{-b + \sqrt{b^2 + 4a}}{2a}. \quad (3.14)$$

The algorithm iterates until through each $c \in \mathcal{C}$ and $j = 1, 2, \ldots, p$ until convergence. The proposed algorithm is explicitly described in Algorithm 2.

Algorithm 2

Initialize a convergence tolerance ε and $d_{cj} = s_{c,jj}^{-.5}$ for all $c = 1, \ldots, C$ and $j = 1, 2, \ldots, p$.

repeat

for $c \in \mathcal{C}$

for $j \in 1, \ldots, p$

Calculate a using (3.12)

Calculate b using (3.13)

Set $d_{cj}^{\text{old}} = d_{cj}$

Set $d_{cj} := \frac{-b+\sqrt{b^2+4a}}{2a}$.

end for

end for

until

$\sum_{c \in \mathcal{C}} |D_c^{\text{old}} - D_c|_1 < \varepsilon$

end repeat □

3.3 Tuning Parameter Selection

Tuning parameter selection for the common correlation model will be done simultaneously with estimating the inverse covariance matrices. We propose using a cross validation method with a validation likelihood. While the objective function in (3.1) has two tuning parameters, Algorithm 3 separates the problem using a different penalty at each step. This allows us to select the tuning parameters using two line searches rather than a gird search as proposed in Section 2.3.3.

Algorithm 3

Calculate S_c

Initialize $\hat{d}_{c,j} = s_{c,jj}^{-.5}$, and $\lambda_1, \lambda_2 > 0$

Calculate R

$\Omega = H(R, n^{-1}\lambda_1)$

Implement Algorithm 2 to find $D_1, \ldots, D_C$

for $c \in \mathcal{C}$

$\Theta_c = D_c \Omega D_c$

end for □

We propose cross validation using validation likelihood, similar method as in Section 2.3.3, where we first split each of the classes into K subsets. Using a similar notation define $R_{(v)}$ to be the sample correlation matrix of the vth subset of the data, and $\Omega_{(-v)}$ be an estimate of Ω removing the vth subset of the data. Define the validation likelihood score for a fixed $\lambda_1 > 0$ as

$$VL(\lambda_1) = \sum_{v=1}^{K} \operatorname{tr}(R_{c(v)}\Omega_{(-v)}) - \log\det(\Omega_{(-v)}), \tag{3.15}$$

where $\Omega_{(-v)}$ is estimated using λ though the notation does not suggest it. We choose the $\hat{\lambda}_1$ as the value out of all possible values that minimizes $VL(\lambda_1)$.

We again use a validation likelihood to select λ_2. Define the validation score for selecting λ_2 as

$$M_{\hat{\lambda}_1}(\lambda_2) = \sum_{v=1}^{K} \sum_{c \in \mathcal{C}} n_{(v)c} \left(\operatorname{tr}(S_{(v)c} D_{(-v)c} \Omega_{(-v)} D_{(-v)c}) - 2\log\det(D_{(-v)c}) - \log\det(\Omega_{(-v)}) \right) \tag{3.16}$$

where $\Omega_{(-v)}$ is found using the tuning parameter $\hat{\lambda}_1$ though the notation does not suggest it. We choose the value $\hat{\lambda}_2$ as the value of all possible value that minimizes (3.16). Algorithm 4 explicitly shows each step of the algorithm.

Algorithm 4

Initlaize Λ_1 and Λ_2 to be possible values of λ_1, λ_2.

$\hat{\lambda}_1 = \arg\min_{\lambda \in \Lambda_1} VL(\lambda)$

$\hat{\lambda}_2 = \arg\min_{\lambda \in \Lambda_2} M(\lambda)$

Calculate $\widetilde{\Theta}_1, \ldots, \widetilde{\Theta}_C$ using Algorithm 3 with $\hat{\lambda}_1, \hat{\lambda}_2$ □

3.4 Simulations

3.4.1 Regularization in Quadratic Discriminant Analysis

In this section we present a simulation study that is similar to that presented in Section 2.6. The difference being in Chapter 3 we present methods that estimate a sparse inverse common correlation matrix so our focus will be on comparing to methods that could preform well when the inverse covariance matrices are sparse. We compare the common correlation method to the fused graphical lasso (FGL) and regularized discriminant analysis (RDA). We use RDA as our baseline method for comparison since it is the traditional method used for discriminant analysis in the $p > n_c$ setting.

Each simulation will investigate a specific setting of a two category problem. The data is generated from a model where the distribution of category 1 was generated from $N_p(\mu_1, \Sigma_1)$ and class 2 was generated from $N_p(\mu_2, \Sigma_2)$. The training data was created using 50 independent observations from class 1 and 50 independent observations from class 2. These training observations are used to create the estimates of the parameters of the normal distributions of both classes. The parameter estimates are used in the QDA model to classify 500 independent observations from class 1 and 500 independent observations from class 2. We measure the performance using the

classification error rate (CER) on the testing cases. This process was replicated 100 times.

The tuning parameters for FGL and the common correlation method were selected from a subset of $\{10^x : x = -10, -9.5\ , \ldots, 9.5, 10\}$ using the method discussed in Section 3.3 and Section 2.3.3 respectively. We also selected the tuning parameters for RDA using the method proposed in Section 2.3.3. To select the tuning parameters for the common correlation method, which we denote as Common Cor, wed used the method described in Section 3.3.

This simulation study is designed to investigate the performance of the common correlation in classification compared to a traditional method in RDA and a modern counterpart such as FGL. We do not use the ridge fusion method proposed in Chapter 2 due to the fact it cannot estimate sparse inverse covariance matrices well, as we have shown in the simulation study in Section 2.6. Since common random numbers are used for each simulation, instead of reporting standard errors we report the proportion of the replications that have a lower classification error rate as compared to the common correlation method.

The results of our simulation study show that when the assumption of a sparse inverse common correlation matrix is met the common correlation method has better classification performance under average CER and the proportion of replications that have a lower classification error rate. This is not an unreasonable result as the common correlation method is built for a specific class of problem.

3.4.2 Simulation 1

Simulation 1 investigates the ability of the methods to classify when the common correlation assumption between the classes is met. We set each element of μ_1 equal to $10p^{-1}\log(p)$ and each element of μ_2 equal to zero. We consider the setting where $\Sigma_1 = AR_{0.7}(1)$, and $\Sigma_2 = D\Sigma_1 D$, where D is a diagonal matrix with the first 20

Table 3.1: Average classification error rate for 100 independent replicates of the setting described in Simulation 1. In parenthesis we report the proportion of the replications that FGL and RDA have a lower classification rate than the common correlation method.

	$p = 20$	$p = 50$	$p = 100$
Common Cor	0.02	0.03	0.05
FGL	0.03 (0.10)	0.04 (0.02)	0.07 (0.09)
RDA	0.02 (0.21)	0.07 (0.00)	0.16 (0.00)

element equal to 1 and the remaining elements are set equal to 2.

We investigated the cases where $p = 20, 50$ and 100. The results of this simulation are contained in Table 3.1, and show that the common correlation method performs as well or better than RDA and FGL with regard to classification performance. RDA performs well when $p = 20$, the only case where $n_c > p$. Table 3.1 also reports (in parenthesis) the proportion of replications where FGL and RDA have a lower classification rate than the common correlation method. We see that based on this simulation study the common correlation method has a lower (or equal) classification error rate than FGL in atleast 90% of the replications. When compared to RDA, common correlation has a lower (or equal) classification error rate 80% of the time when $p = 20$, and in all replications for the cases when $p = 50$ and $p = 100$.

3.4.3 Simulation 2

Simulation 2 investigates a second case where the common correlation assumption is met but in this setting the inverse correlation is not sparse. We set each element of μ_1 equal to p^{-1} and each element of μ_2 is 0. We consider a model where Σ_1 is tri-diagonal with the diagonal elements equal to 1, and off diagonal elements are equal to 0.25. We set $\Sigma_2 = D\Sigma_1 D$ where D is the same matrix D as defined in Section 3.4.2.

We again investigated the settings where $p = 20, 50$, and 100. The classification

Table 3.2: Average classification error rate for 100 independent replicates of the setting described in Simulation 2. In parenthesis we report the proportion of the replications that FGL and RDA have a lower classification rate than the common correlation method.

	$p = 20$	$p = 50$	$p = 100$
Common Cor	0.10	0.14	0.17
FGL	0.11 (0.38)	0.13 (0.81)	0.18 (0.75)
RDA	0.12 (0.03)	0.22 (0.00)	0.30 (0.00)

results are contained in Table 3.2. We see that the common correlation method and FGL perform similar for each value of p when average classification error rate is used as the metric. When we use the proportion of replications that have a lower classification error rate for evaluation, this result changes. If we look in the parentheses in table Table 3.2 we see that FGL has a lower classification rate in 38% of the replications when $p = 20$, 81% of the replications when $p = 50$ and 75% when $p = 100$. When comparing RDA to the common correlation method we see that when $p = 20$ RDA has a lower classification error rate in 3% of the replications, and does not have a lower classification error rate in any of the replications when $p = 50$ or $p = 100$.

The case when $p = 50$ and $p = 100$ when comparing FGL and the common correlation method requires further comment. We see that the average classification error rates show similar classification performance but when the proportion of replications where FGL is better than common correlation is reported FGL has a substantial advantage. This is due to the properties of the l_1 penalties used by FGL. Slight increases in λ_1 and λ_2 with FGL can result in large differences in the estimates of the inverse covariance matrices, which will change the classification performance dramatically. When further investigated we found that the median classification error rate for $p = 50$ and $p = 100$ is lower than that of the common correlation method, but points out a disadvantage of an l_1 fusion penalty.

Table 3.3: Average classification error rate for 100 independent replicates of the setting described in Simulation 3. In parenthesis we report the proportion of the replications that FGL and RDA have a lower classification rate than the common correlation method.

	$p = 20$	$p = 50$	$p = 100$
Common Cor	0.12	0.05	0.03
FGL	0.13 (0.20)	0.10 (0.00)	0.08 (0.00)
RDA	0.14 (0.11)	0.18 (0.00)	0.19 (0.00)

3.4.4 Simulation 3

Simulation 3 investigates a third setting where the common correlation matrix is met, but unlike previous simulations neither Σ_1 or Σ_2 are on the correlation scale. In this model we let $\Omega = AR_{0.7}(1)$, be the common correlation matrix and $\Sigma_1 = D_1 \Omega D_1$ and $\Sigma_2 = D_2 \Omega$. We define D_1 and D_2 to be diagonal matrices where the first 20 diagonal elements of D_1 are 2 and the remaining are set equal to 1, and all diagonal elements of D_2 are set to $\sqrt{2}$. We set every element of μ_1 equal to $10p^{-1}\log(p)$ and every element of μ_2 is equal to zero.

We again investigate the setting with $p = 20, 50$ and 100. The classification results are contained in Table 3.3 and it shows that the common correlation method out performs FGL and RDA for all values of p. What we also see is that as p increases RDA loses the signal of the classification rule. The proportion of replications that each method has a lower (or equal) classification error rate than the common correlation method also shows that the common correlation method is better for each value of p.

3.4.5 Simulation 4

Simulation 4 investigates the classification performance of the three methods when the common correlation assumption between classes is not true. It is expected that

Table 3.4: Average classification error rate for 100 independent replicates of the setting described in Simulation 4. In parenthesis we report the proportion of the replications that FGL and RDA have a lower classification rate than the common correlation method.

	$p = 20$	$p = 50$	$p = 100$
Common Cor	0.11	0.23	0.32
FGL	0.07 (0.99)	0.07 (1.00)	0.11 (1.00)
RDA	0.07 (0.99)	0.13 (1.00)	0.17 (1.00)

the common correlation method will not perform well in this setting. In this model assume that $\Sigma_1 = AR_{0.7}(1)$ and $\Sigma_2 = AR_{0.9}(1)$. We set each element of μ_1 to be $10p^{-1}\log(p)$ and each element of μ_2 to be 0. We again investigate the cases where $p = 20, 50$ and 100.

The classification results are contained in Table 3.4 and show that FGL has a better classification performance for all values of p. Table 3.4 also shows that RDA and FGL have a lower classification error rate for almost every replication of this simulation.

3.5 Data Example

Just as in Section 2.7 we will show the merits of the common correlation method on the Libras movement data from the UCI Machine learning repository (Bache and Lichman, 2013). As a reminder the data has 3 classes that represent the type of hand movement and 90 variables that describe he hand movements at 45 time points from a video of the hand movement. Again we have taken 18 data points from each class to train our model while keeping 45 data points from each class for validation. The tuning parameter selection method used for the common correlation method is described in Section 3.3 and have been selected from a subset of the grid $\{10^x$:

$x = -8, -7, \ldots, 7, 8\}$ using 3-fold cross validation. The results show that 3 out of the possible 135 data points in the validation data were classified incorrectly. When compared to the results found in Table 2.8 we find that the common correlation method has a better classification performance than FGL and RDA, while having worse performance than the ridge fusion method.

3.6 Extensions

In this chapter we have discussed a unique approach to estimating multiple inverse covariance matrices for use in quadratic discriminant analysis, by using the assumption of a sparse inverse common correlation matrix between all of the classes. Our two step algorithm allows for tuning parameter selection to be done through two line searches rather than a grid search. This methodology promotes a common sparsity pattern for all inverse covariance matrices while promoting similarity and allows different inverse covariance matrices by fusing the inverse standard deviations of the categories. An extension of this work we are very interested in pursuing is the joint estimation in this same setting, which maintains Ω as an inverse correlation matrix for each iteration. We also have interest in extending this type of penalization and decomposition along with the penalties proposed in Chapter 2 to the covariance estimation problem.

A second extension of this work would be to use a fused lasso penalty rather than the ridge fusion penalty. This would allow for the resulting estimates of the inverse covariance matrix to be the same, which would be result in a penalized LDA rule. We can consider this as an extension of our current work because we could use a local quadratic approximation of the l_1 penalty with respect to our block descent algorithm. This would use the ridge penalized solution with thresholding to find completely fused solutions. We would also want to investigate the properties of when a ridge penalty is used on the common correlation matrix. This would require methods such as one

presented in Rothman and Forzani (2013) where the diagonal elements would not be penalized. This could be useful for settings such as the one presented in Section 3.4.3 where the assumption of a common correlation matrix is met but the inverse is not sparse.

A final extension of this work would be to take the common correlation method and apply it to the semi-supervised (and unsupervised) model based clustering problem. To do this we could not use the tuning parameter selection method we have proposed in Section 3.3 but would need to use the method proposed in Section 2.5.3.

Chapter 4

Group Fused Multinomial Logistic Regression

4.1 Introduction

In Chapter 2 and Chapter 3 we have proposed methods to estimate the inverse covariance matrix for use in quadratic discriminant analysis. In this chapter the discussion turns to the use of fusion penalties in multinomial logistic regression. The main difference between QDA and multinomial logistic regression is that QDA exploits a joint distribution of the categorical response and the predictor, while multinomial logistic regression directly models the distribution of the categorical response in sub-populations determined by the value of the predictor.

Let $x_{i1}, \ldots, x_{ip}$ be p explanatory variables for the ith case and define $x_i^T = (1, x_{i1}, \ldots, x_{ip})$. Let $y_i = (y_{i1}, \ldots, y_{iC})$ be the vector of C category counts for the ith case based on n_i independent and identical multinomial trails resulting in C categories. This model assumes that $y_1, \ldots, y_N$ are a realization of $Y_1, \ldots, Y_N$ where

$$Y_i \sim \text{Multi}\left(n_i, \pi_1(x_i), \ldots, \pi_C(x_i)\right), \tag{4.1}$$

and

$$\pi_c(x_i) = \frac{\exp(x_i^T \beta_c)}{\sum_{m \in \mathcal{C}} \exp(x_i^T \beta_m)}. \tag{4.2}$$

To make the model identifiable we set one of the β vectors equal to the zero vector. From (4.2) we have that

$$\log\left(\frac{\pi_c(x_i)}{\pi_k(x_i)}\right) = x_i^T \beta_c^{(k)}, \tag{4.3}$$

where the superscript (k) defines the baseline category, the category all other categories are being compared to, and $\beta_k^{(k)} = \vec{0}$. This notation allows for any category to be chosen as the baseline category and is useful when comparing different categories. For response categories c and m we have that

$$\begin{aligned}
\log\left(\frac{\pi_c(x)}{\pi_k(x)}\right) - \log\left(\frac{\pi_m(x)}{\pi_k(x)}\right) =& x^T \beta_c^{(k)} - x^T \beta_m^{(k)} \\
=& x^T (\beta_c^{(k)} - \beta_m^{(k)}) \\
=& \log\left(\frac{\pi_c(x)}{\pi_m(x)}\right) \\
=& x^T \beta_c^{(m)}.
\end{aligned}$$

This has taken a model defined by (4.3) with baseline class k and redefined the model with baseline category m, and results in the equality

$$\beta_c^{(m)} = \beta_c^{(k)} - \beta_m^{(k)}. \tag{4.4}$$

In this chapter we propose the group fused multinomial regression model which is a method used reduce the number of response categories in multinomial regression and investigate properties of fusion penalties in multinomial regression. Group fused

multinomial regression is used for exploratory data analysis and is not intended to estimate regression coefficients. An alternating direction method of multipliers algorithm is developed for group fused multinomial regression and tuning parameter selection is also addressed. A simulation study is presented that shows the merits of group fused multinomial regression to select the correct number of categories and category combinations.

Let $\beta^{(k)}$ be the vectorized form of the matrix $(\beta_1^{(k)}, \ldots, \beta_C^{(k)})$, where $\beta_k^{(k)} = \vec{0}$. The multinomial logistic regression log-likelihood using baseline category parameterization is

$$l(\beta^{(k)}) = \sum_{i=1}^{N} \left(\sum_{c \in \mathcal{C}} y_{ic} x_i^T \beta_c^{(k)} - n_i \log(\sum_{r \in \mathcal{C}} \exp\{x_i^T \beta_r^{(k)}\} \right) \tag{4.5}$$

and the penalized likelihood is

$$l(\beta^{(k)}) - \lambda P(\beta^{(k)}). \tag{4.6}$$

An active area of research is on the development of penalized likelihood estimators for use in multinomial and binomial logistic regression. Friedman et al. (2008) proposed a coordinate descent algorithm, known as GLMnet, that solves many penalized likelihood problems, such as lasso, ridge, and group penalties, in both binomial and multinomial logistic regression. GLMnet uses a penalized quadratic approximation of the likelihood, where the solution can be found using a weighted least squares algorithm. Simon et al. (2014) proposed a blockwise descent method for solving group penalized multinomial regression. This method uses an l_2 penalty on an alternative representation of (4.6) to promote shrinkage of the regression coefficient vectors towards the zero vector. Meier et al. (2008) proposed a group penalty for the binomial logistic regression model. Though these methods are more recent additions

to the literature one of the first papers to propose penalization in logistic regression was Anderson and Blair (1982). The application of the penalized logistic regression and penalized multinomial regression have been widely applied in gene micro-array studies and various other fields (Zhu and Hastie, 2004; Park and Hastie, 2007).

4.2 Group Fused Penalty: Combining Categories

We propose the penalized likelihood estimates for $\beta^{(k)}$.

$$\widehat{\beta}^{(k)} = \underset{\beta^{(k)} \in \mathcal{B}^{(k)}}{\arg\max} \left(l(\beta^{(k)}) - \lambda \sum_{(c,m) \in \mathcal{C} \times \mathcal{C}} \sqrt{\sum_{j=0}^{p} \left(\beta_{c,j}^{(k)} - \beta_{m,j}^{(k)} \right)^2} \right), \tag{4.7}$$

where $\mathcal{B}^{(k)} = \{\beta^{(k)} \in \mathbb{R}^{(p+1)C}; \beta_k^{(k)} = \vec{0}\}$. The penalty function in (4.7) is called the *group fused penalty*.

4.2.1 Interest in Fusion Penalties

Though we focus on the group fused penalty, to discuss the merits of using fusion penalties in multinomial logistic regression using the baseline parameterization we will use the general form of the fusion penalty

$$P(\beta^{(k)}) = \sum_{(c,m) \in \mathcal{C} \times \mathcal{C}} |\beta_c^{(k)} - \beta_m^{(k)}|. \tag{4.8}$$

An advantage of using fusion penalized multinomial logistic regression model with baseline parameterization is that the penalized likelihood is invariant with regard to the choice of the baseline category.

Proposition 1

The penalized log-likelihood defined by (4.6) when $P(\beta^{(k)})$ is of the form of (4.8) is invariant with regard to the choice of baseline category.

Proof 4.1

To prove Proposition 1 it is sufficient to show that the equality

$$\beta_c^{(k)} - \beta_m^{(k)} = \beta_c^{(q)} - \beta_m^{(q)},$$

holds for all $c, m, k, q \in \mathcal{C}$. Using (4.4) we have that

$$\begin{aligned}\beta_c^{(q)} - \beta_m^{(q)} &= \beta_c^{(q)} - \beta_m^{(q)} + \beta_k^{(q)} - \beta_k^{(q)}, \\ &= (\beta_c^{(q)} - \beta_k^{(q)}) - (\beta_m^{(q)} - \beta_k^{(q)}), \\ &= \beta_c^{(k)} - \beta_m^{(k)}.\end{aligned}$$

Since we have that $\beta_c^{(k)} - \beta_m^{(k)} = \beta_c^{(q)} - \beta_m^{(q)}$ this implies that

$$\sum_{(c,m)\in\mathcal{C}\times\mathcal{C}} |\beta_c^{(k)} - \beta_m^{(k)}|. = \sum_{(c,m)\in\mathcal{C}\times\mathcal{C}} |\beta_c^{(q)} - \beta_m^{(q)}|..$$

Fusion penalties and the likelihood are invariant to the choice of baseline class (4.6) when $P(\beta^{(k)})$ is a fusion penalty is invariant to the choice of k. □

The invariant property of fusion penalties is important because it is a property not found in non-fusion penalties (i.e. group lasso, lasso, and ridge) in multinomial regression using the baseline category parameterization. Though an interesting property of non-fusion penalties using baseline parameterization is they may be written

as fusion penalties, but not of the form of (4.8), for instance

$$P(\beta^{(k)}) = \sum_{c \in \mathcal{C}} |\beta_c^{(k)}|. = \sum_{c \in \mathcal{C}} |\beta_c^{(k)} - \beta_k^{(k)}|. = \sum_{c \in \mathcal{C}} |\beta_c^{(m)} - \beta_k^{(m)}|..$$

Under our notation any non-fusion penalty can be written as fusion towards a single category. These penalties are not invariant under the choice of baseline class without fusion, that is

$$\sum_{c \in \mathcal{C}} |\beta_c^{(k)}| \neq \sum_{c \in \mathcal{C}} |\beta_c^{(q)}|..$$

The penalty in (4.8) fuses all categories together, but there may be times where penalizing all $(c, m) \in \mathcal{C} \times \mathcal{C}$ is not appropriate. For instance there may be certain categories that should not be combined because the model would no longer make sense. Instead of using the penalty defined by (4.8), an alternative is using

$$\sum_{(c,m) \in \mathcal{S}} |\beta_c^{(k)} - \beta_m^{(k)}|., \tag{4.9}$$

where $\mathcal{S}$ defines a set of (c, m) that are permitted to be fused together.

There are also numerous functions that could be chosen for F. An l_1 fusion penalty could be used to promote similarity between all the classes and could be used to identify variables that matter when comparing different classes. Though this penalty is not discussed here it is something we plan to investigate in our future work. We propose using a group fusion penalty to promote vector-wise similarity between the β's. We will discuss the motivation and implications in the Section 4.2.

4.2.2 Group Fused Multinomial Regression

By Proposition 1 the penalized likelihood in (4.7) is invariant to the choice of k. The group fused penalty is is used to promote vector-wise similarity between the between the $\beta_c^{(k)}$'s.

Using the group fused penalty there exists a λ such that $\widehat{\beta}_c^{(k)} - \widehat{\beta}_m^{(k)} = \vec{0}$, and implies that

$$\log\left(\frac{\pi_c(x)}{\pi_m(x)}\right) = x^T(\widehat{\beta}_c^{(k)} - \widehat{\beta}_m^{(k)}) = 0. \tag{4.10}$$

The result of (4.10) is a model that indicates c and m occur with the same probability for all x. This means the model cannot tell the difference between these two categories. In terms of fusion, category c and category m are combined or fused together since $\widehat{\beta}_c^{(k)} = \widehat{\beta}_m^{(k)}$. We propose using the group fused penalty to combine categories in multinomial regression, where the tuning parameter will dictate the number of categories that are combined. Reducing the number of response categories will reduce the number of regression coefficients that need estimated and make the model easier to interpret. Large values of λ will lead to many or all categories having the same estimated regression coefficient vectors, while small values of λ will lead to no categories being combined. Though it may not be clear from our notation the penalty function used in (4.7) also promotes shrinkage of the estimates of $\beta_c^{(k)}$ towards the zero vector, which is fusion towards the baseline category. To show this we use the fact that $\beta_k^{(k)} = 0$ then

$$\lambda \sum_{(c,m)\in\mathcal{C}\times\mathcal{C}} |\beta_c^{(k)} - \beta_m^{(k)}|_2 = \lambda \left(2\sum_{c\in\mathcal{C}} |\beta_c^{(k)}|_2 + \sum_{(c,m)\in\mathcal{C}\setminus\{k\}\times\mathcal{C}\setminus\{k\}} |\beta_c^{(k)} - \beta_m^{(k)}|_2 \right). \tag{4.11}$$

We see that for large values of λ (4.11) not only fuses the regression coefficient vectors towards one another but also towards the zero vector.

Any algorithm used to find the estimates in (4.7) must meet the condition that for each $\widehat{\beta}_c^{(k)}$

$$X^T(\vec{Y}_c - n\hat{\pi}_c) - 2\lambda \sum_{m \in \mathcal{C}} \Gamma_{cm} = 0, \tag{4.12}$$

where $\vec{Y}_c = (y_{1c}, \ldots, y_{nc})^T$, where X is the $n \times (p+1)$ matrix of predictors, $n = (n_1, \ldots, n_N)$, and $\hat{\pi}_c = (\hat{\pi}_{1c}(x_1), \ldots, \hat{\pi}_{Nc}(x_i))^T$, and

$$\Gamma_{cm} = \begin{cases} \frac{\widehat{\beta}_c^{(k)} - \widehat{\beta}_m^{(k)}}{|\widehat{\beta}_c^{(k)} - \widehat{\beta}_m^{(k)}|_2} \text{if } \widehat{\beta}_c^{(k)} \neq \widehat{\beta}_m^{(k)} \\ \gamma_{cm} \text{ such that } |\gamma_{cm}|_2 \leq 1 \text{ if } \widehat{\beta}_c^{(k)} = \widehat{\beta}_m^{(k)}. \end{cases} \tag{4.13}$$

Though the $\beta^{(k)}$'s are being penalized the objective of group fused multinomial regression is find the categories that should be combined, we refer to this as estimating the group structure of the model. We define the group structure, $\mathcal{G} = (g_1, \ldots, g_G)$ as a partition of $\mathcal{C}$, where $c, m \in g_j$, if $\pi_c(x) = \pi_m(x)$ for all values of x. Let $\tilde{y}_i = (\tilde{y}_{i1}, \ldots, \tilde{y}_{iG})$ be the vector of G category counts of the ith case based on n_i independent and identical multinomial trials defined by the group structure $\mathcal{G}$. Let $\tilde{y}_1, \ldots, \tilde{y}_N$ be realizations of $\tilde{Y}_1, \ldots, \tilde{Y}_N$ where

$$\tilde{Y}_i \sim \text{Multi}(n_i, \theta_1(x_i), \ldots, \theta_G(x_i)), \tag{4.14}$$

and

$$\log\left(\frac{\theta_{g_j}(x)}{\theta_{g_l}(x)}\right) = x_i^T(\eta_{g_j} - \eta_{g_l}). \tag{4.15}$$

The connection between the the model with C categories and model with the group structure defined by $\mathcal{G}$ is that $\tilde{y}_{ig_j} = \sum_{c \in g_j} y_{ic}$ and $\theta_{g_j} = \sum_{c \in g_j} \pi_c I(c \in g_j)$.

4.2.3 ADMM Algorithm

We propose finding the estimates defined in (4.7) by using the alternating direction method of multipliers algorithm, most commonly known as the ADMM algorithm. The ADMM algorithm is a technique combines dual ascent methods with method of multipliers techniques to create an algorithm which solves (4.7). A great discussion of the general ADMM algorithm, examples, and convergence properties can be found in Boyd et al. (2010).

The ADMM algorithm solves problems of the form

$$\underset{\beta^{(k)}\in\mathcal{B}^{(k)},Z\in\mathbb{R}^{q(p+1)}}{\arg\min} \quad -l(\beta^{(k)}) + P(Z) \tag{4.16}$$

subject to the constraint $A\beta^{(k)} - BZ = c$, where $Z \in \mathbb{R}^{q(p+1)}$, $A \in \mathbb{R}^{m\times C(p+1)}$ and $B \in \mathbb{R}^{m\times q(p+1)}$. To solve (4.16), we define the augmented Lagrangian as

$$L_\rho(\beta^{(k)}, Z, v) = -l(\beta^{(k)}) + P(Z) + v^T(A\beta^{(k)} - BZ - c) + \frac{\rho}{2}|A\beta^{(k)} - BZ - c|_2^2, \tag{4.17}$$

where v is the dual variable and $\rho > 0$ is the penalty parameter. The solution to (4.16) requires an iterative procedure between the following three steps

$$\underset{\beta^{(k)}\in\mathcal{B}^{(k)}}{\arg\min}\ L_\rho(\beta^{(k)}, Z, v) \tag{4.18}$$

$$\underset{Z\in\mathbb{R}^{qC}}{\arg\min}\ L_\rho(\beta^{(k)}, Z, v) \tag{4.19}$$

$$v_{(q+1)} = v_{(q)} + \rho(A\beta^{(k)} + BZ - c), \tag{4.20}$$

where $v_{(q)}$ is the qth iteration of the algorithm. The literature has many ADMM convergence properties, but we will focus on a theorem that applies to the methods of the form (4.16). This result is presented in Mota et al. (2011) and requires three

assumptions:

A1) $-l(\beta^{(k)})$ and $P(Z)$ are proper, closed, convex functions.

A2) (4.16) is solvable and with an optimal objective function value of p^*

A3) Matrices A and B have full column rank

Theorem 4.1 (Mota et al. (2011))
If assumptions A1, A2, and A3 are met then the iterates produced by the ADMM algorithm, as the number of iterates goes to infinity, converge to the optimal value of the objective function, converges to $(\widehat{\beta}^{(k)}, \widehat{Z})$ which solves (4.16), and the dual variable converges to the dual optimal point. □

We refer the reader to Mota et al. (2011) for the proof.

To find the solution to the optimization in (4.7) using the ADMM algorithm we reformulate (4.7) as

$$\underset{\beta^{(k)} \in \mathcal{B}^{(k)},\ Z \in \mathbb{R}^{(p+1)C(C-1)}}{\arg\min} \left(-l(\beta^{(k)}) + \lambda \sum_{(c,m) \in \mathcal{C} \times \mathcal{C}} |Z_{cm}|_2 \right) \tag{4.21}$$

subject to the constraint that $Z_{cm} = \beta_c^{(k)} - \beta_m^{(k)} = A_{cm}\beta^{(k)}$. Notice that (4.21) can be split into two parts, the log-likelihood which has parameter $\beta^{(k)}$ and the penalty which has parameter Z which is the vectorized form of the matrix

$$\left(Z_{12}, \ldots, Z_{1C}, Z_{21}, Z_{23}, \ldots, Z_{C(C-1)}\right).$$

The scaled augmented Lagrangian form of (4.21) is

$$g(\beta^{(k)}, Z, U) = -l(\beta^{(k)}) + \sum_{(c,m) \in \mathcal{C} \times \mathcal{C}} \left(\lambda |Z_{cm}|_2 + \frac{\rho}{2} |\beta_c^{(k)} - \beta_m^{(k)} - Z_{cm} + U_{cm}|_2^2 \right), \tag{4.22}$$

where $\rho > 0$ is a penalty parameter and the scaled dual variable, U, is the vectorized form of the matrix

$$\left(U_{12}, \ldots, U_{1C}, U_{21}, \ldots, U_{C(C-1)}\right).$$

The ADMM solution of (4.21) is a iterative three step procedure that is shown in Algorithm 5.

Algorithm 5

Initialize $\widehat{Z}_{cm} = 0$ and $\widehat{U}_{cm} = 0$ for all $c, m \in \mathcal{C} \times \mathcal{C}$

repeat

$\widetilde{\beta}^{(k)} = \arg\min_{\beta^{(k)} \in \mathcal{B}^{(k)}} g(\beta^{(k)}, \widehat{Z}, \widehat{U})$

$\widehat{Z} = \arg\min_{Z \in \mathbb{R}^{C(C-1)(p+1)}} g(\widetilde{\beta}^{(k)}, Z, \widehat{U})$

$\widetilde{U} = \widehat{U} + A\widetilde{\beta}^{(k)} - \widehat{Z}$

$\widehat{U} = \widetilde{U}$

until

Convergence

end repeat □

Theorem 4.2 (Convergence of Algorithm 5)

The ADMM Algorithm that solves (4.21) , Algorithm 5, converges to the optimal objective function value, converges to the optimal values of $(\beta^{(k)}, Z)$, and the dual variable converges to the optimal dual variable. □

Proof 4.2

It is sufficient to show that (4.21) meets A1, A2, and A3 then we can apply Theorem 4.1 for convergence.

To simplify notation in this proof let

$$f_2(\beta^{(k)}) = \sum_{(c,m)\in\mathcal{C}\times\mathcal{C}} |\beta_c^{(k)} - \beta_m^{(k)}|_2.$$

To apply the result of Theorem 4.1 we must show that (4.7) is equivalent to the optimization problem

$$\underset{\beta^{(k)}\in\mathcal{B}^{(k)}, Z\in\mathbb{R}^{pC(C-1)}}{\arg\max} \; l(\beta^{(k)}) - \lambda f_3(Z), \tag{4.23}$$

subject to the constraint $Z_{cm} = \beta_c^{(k)} - \beta_m^{(k)}$, where

$$f_3(Z) = \sum_{(c,m)\in\mathcal{C}\times\mathcal{C}} |Z_{cm}|_2.$$

To do this we use that fact that that

$$|\beta_c^{(k)} - \beta_m^{(k)}|_2 = |A_{cm}\beta^{(k)}|_2$$

Then $f_2(\beta^{(k)}) = f_3(Z)$ under the constraint that $Z_{cm} = A_{cm}\beta^{(k)}$.

This allows us to say (4.23) and (4.7) are equivalent. It will be easier to use (4.7) moving forward in our proof.

Proof A1 The likelihood function $l(\beta^{(k)})$ is concave and continuous, so it is proper, closed and concave. The function $-\lambda f_2(\beta_k)$ is also concave and continuous and hence proper, closed and concave.

Proof A2 Since (4.21) is equivalent to (4.7), by Theorem 1.9 in Rockafellar and Wets (2004) (4.7) is solvable if the objective function is lower semi-continuous, proper and level bounded. Lower semi-continuity and proper are satisfied by A1 but bounded level sets requires more. Exercise 1.41 from Rockafellar and Wets

(2004) allows us to show that if $l(\beta^{(k)})$ and $-\lambda f_2(\beta^{(k)})$ are bounded above and $l(\beta^{(k)})$ or $-\lambda f_2(\beta^{(k)})$ have bounded level sets then $l(\beta^{(k)})-\lambda f_2(\beta^{(k)})$ has bounded level sets.

By basic properties of the multinomial log-likelihood $l(\beta^{(k)})$ is bounded above by 0. We also have that $-\lambda f_2(\beta^{(k)})$ is bounded above by 0, for $\lambda \geq 0$ since $f_2(\beta^{(k)})$ is the sum of l_2 norms.

Let $\hat{\beta} \in \{\beta \in \mathcal{B}^{(C)}; f_2(\beta) < \alpha\}$, to show f_2 has bounded level sets we must show that $|\hat{\beta}|_2$ is bounded above. Let $\hat{\beta}_C = \vec{0}$, where C denotes the baseline category. Then

$$\begin{aligned}
\alpha &> \sum_{(c,m)\in\mathcal{C}\times\mathcal{C}} |\hat{\beta}_c - \hat{\beta}_m|_2 \\
&= 2\sum_{c\in\mathcal{C}} |\hat{\beta}_c - \hat{\beta}_C|_2 + \sum_{(c,m)\in\mathcal{C}\setminus\{C\}\times\mathcal{C}\setminus\{C\}} |\hat{\beta}_c - \hat{\beta}_m|_2 \\
&\geq \sum_{c\in\mathcal{C}} |\hat{\beta}_c|_2 \\
&\geq \frac{1}{\sqrt{p}} |\hat{\beta}_c|1 \\
&= \frac{1}{\sqrt{p}} |\hat{\beta}|_1 \\
&\geq \frac{1}{\sqrt{p}\sqrt{C}} |\hat{\beta}|_2.
\end{aligned}$$

This results in the inequality

$$|\hat{\beta}|_2 < \sqrt{pC}\alpha, \tag{4.24}$$

and proves that f_2 has bounded level sets.

This allows us to say that (4.7) has bounded level sets and apply Theorem 1.9 from Rockafellar and Wets (2004) to say the problem is solvable.

Proof of A3 We can reformulate the constraint $Z_{cm} = \beta_c^{(k)} - \beta_m^{(k)} = A_{cm}\beta^{(k)}$ in (4.21) as $A\beta^{(k)} - BZ = \vec{0}$ where A_{cm} are the row blocks of A and B is the identity matrix. This results in A and B being full column rank by construction. □

Algorithm 5 iterates using solutions to three non-trivial optimization problems of the function g. The first step of the algorithm is to find the estimates

$$\widetilde{\beta}^{(k)} = \underset{\beta^{(k)} \in \mathcal{B}^{(k)}}{\arg\max} -g(\beta^{(k)}, \widehat{Z}, \widehat{U}). \tag{4.25}$$

To find the estimates in (4.25) we propose using a block coordinate descent algorithm which uses the Newton-Raphson algorithm, where each β_c is considered a block. The gradient of $g(\beta^{(k)}, \widehat{Z}, \widehat{U})$ with respect to $\beta_c^{(k)}$ is

$$G(\beta_c^{(k)}) = X^T(\vec{Y}_c - n\pi_c) - 2\rho \sum_{m \in \mathcal{C}} (\beta_c^{(k)} - \widetilde{\beta}_m^{(k)} - \widehat{Z}_{cm} + \widehat{U}_{cm}), \tag{4.26}$$

and $\pi_c = (\pi_c(x_1), \ldots, \pi_c(x_n))$. The hessian with respect to $\beta_c^{(k)}$ is

$$H(\beta_c^{(k)}) = -X^T W X - 2\rho(C-1)I_p, \tag{4.27}$$

where W is a diagonal matrix such that $W_{ii} = \pi_c(x_i)(1 - \pi_c(x_i))$.

Let $\hat{\beta}_{(k)}$ bet the current iterate of $\beta^{(k)}$ then, the update of $\beta_c^{(k)}$ is

$$\widetilde{\beta}_c^{(k)} = \hat{\beta}_c^{(k)} - H(\hat{\beta}_c^{(k)})^{-1} G(\hat{\beta}_c^{(k)}). \tag{4.28}$$

The algorithm iterates between each $c \in \mathcal{C} \setminus \{k\}$, where k is the baseline category, until convergence. Algorithm 6 explicitly shows this block coordinate descent algorithm.

Algorithm 6

Initialize $\widehat{Z}$, $\widehat{U}$, a convergence tolerance ε, and $\widetilde{\beta}_c^{(k)} = \vec{0}$ for $c \in \mathcal{C}$

repeat

for $c \in \mathcal{C}$

$\hat{\beta}_c^{(k)} = \widetilde{\beta}_c^{(k)}$

Calculate $\widetilde{\beta}_c^{(k)}$ using (4.28)

end for

until

Convergence

end repeat □

The second step of Algorithm 5 requires finding the solution to

$$\underset{Z \in \mathbb{R}^{C(C-1)(p+1)}}{\arg\min} \sum_{(c,m) \in \mathcal{C} \times \mathcal{C}} \left(\lambda |Z_{cm}|_2 + \frac{\rho}{2} |\beta_c^{(k)} - \beta_m^{(k)} - Z_{cm} + U_{cm}|_2^2 \right). \tag{4.29}$$

Since (4.29) can be written as the sum over (c, m), we can solve for each Z_{cm} individually. The Z_{cm} that minimizes the objective function in (4.29) is the Z_{cm} that satisfies

$$Z_{cm} - \beta_c^{(k)} + \beta_m^{(k)} - U_{cm} - \frac{\lambda}{\rho} \Upsilon_{cm} = 0, \tag{4.30}$$

where

$$\Upsilon_{cm} = \begin{cases} \frac{Z_{cm}}{|Z_{cm}|_2} & \text{if} |Z_{cm}|_2 \neq 0 \\ \upsilon_{cm} & \text{such that } |\upsilon_{cm}|_2 \leq 1 \text{ if } |Z_{cm}|_2 = 0. \end{cases}$$

Following a similar strategy to Yuan and Lin (2006) the solution to (4.30) is

$$\widehat{Z}_{cm} = A_{cm}(1 - \frac{\lambda}{\rho |A_{cm}|_2})_+, \tag{4.31}$$

where $A_{cm} = \beta_c^{(k)} - \beta_m^{(k)} + U_{cm}$, and $(a)_+ = \max(a, 0)$. This closed form solution

allows us to quickly calculate Z_{cm} for each $(c, m) \in \mathcal{C} \times \mathcal{C}$.

The third step of Algorithm 5 updates U_{cm} based on the current estimates of $\beta^{(k)}$ and Z. Let $\widetilde{U}_{cm}$ be the current iterate then the update of U_{cm} is

$$\widehat{U}_{cm} = \widetilde{U}_{cm} + \widetilde{\beta}_c^{(k)} - \widetilde{\beta}_m^{(k)} - \widehat{Z}_{cm}. \tag{4.32}$$

Define the residual of the current iteration as $\widetilde{\beta}_c^{(k)} - \widetilde{\beta}_m^{(k)} - \widehat{Z}_{cm}$, then we can interpret $\widehat{U}_{cm}$ as the sum of the residuals for the current iterate and all previous iterates.

4.2.4 Computational Issues

The ADMM algorithm proposed in Algorithm 5 has computational issues that need addressed. The first is that (4.25) is an l_2 penalized multinomial regression, meaning that fusion of the coefficient vectors will never happen. To solve this issue we report the final estimate,

$$\widehat{\beta}_c^{(k)} = \frac{\sum_{m \in \mathcal{C}} \widetilde{\beta}_m^{(k)} I(Z_{cm} = 0)}{\sum_{m \in \mathcal{C}} I(Z_{cm} = 0)}.$$

We also replace all $\widehat{\beta}_c^{(k)}$, where $|\widehat{\beta}_c^{(k)}|_2 < \tau$, with the zero vector. In our implementation of this algorithm we choose $\tau = 10^{-9}$. We consider this analogous to the thresholding proposed in Rothman et al. (2008) which produces estimates based on a local quadratic approximation.

The second issue that needs addressed is the convergence criterion for the algorithm. We propose using the same convergence criterion as proposed in Boyd et al. (2010), which uses the convergence of the dual and primal residuals. We also have implemented a varying penalty parameter ρ and described in the same paper, to help convergence speed of our algorithm.

In the case of (4.7) we can rewrite the penalty function as

$$\sum_{(c,m)\in\mathcal{C}\times\mathcal{C}} |\beta_c^{(k)} - \beta_m^{(k)}|_2 = \sum_{(c,m)\in\mathcal{S}} |\beta_c^{(k)} - \beta_m^{(k)}|_2,$$

where we define $\mathcal{S} = \mathcal{C}\times\mathcal{C}$, but as previously mentioned it maybe useful to have a more general form of $\mathcal{S}$. For a more general form of $\mathcal{S}$, which we assume to be symmetric (i.e. if $(c,m) \in \mathcal{S}$ then $(m,c) \in \mathcal{S}$), we can reformulate the objective function of the optimization problem as

$$l(\beta^{(k)}) - \sum_{(c,m)\in\mathcal{S}} \left(\lambda|Z_{cm}|_2 + \frac{\rho}{2}|\beta_c^{(k)} - \beta_m^{(k)} - Z_{cm} + U_{cm}|_2^2\right). \tag{4.33}$$

While this notation is mathematically convenient, it is not convenient for the optimization. The reformulation of (4.33) for the optimization is

$$l(\beta^{(k)}) - \sum_{(c,m)\in\mathcal{C}\times\mathcal{C}} \left(\lambda|Z_{cm}|_2 + \frac{\rho}{2}|\beta_c^{(k)} - \beta_m^{(k)} - Z_{cm} + U_{cm}|_2^2\right) E_{c,m}, \tag{4.34}$$

where

$$E_{c,m} = \begin{cases} 1 \text{ if } (c,m) \in \mathcal{S} \\ 0 \text{ if } (c,m) \notin \mathcal{S}\,. \end{cases}$$

This changes the optimization in the update of $\beta^{(k)}$, but in terms of Z and U only effects the size of the matrices since $E_{c,m}$ just serves as an indicator of which constraints should be in the problem. Since E is a symmetric matrix since $E_{c,m} = E_{m,c}$. The update for $\beta^{(k)}$ in the ADMM algorithm that solves (4.34) with current

iterate $\hat{\beta}_c^{(k)}$ is

$$G(\beta_c^{(k)}) = X^T(Y_c - n\pi_c) - 2\rho \sum_{m \in \mathcal{C}} (\beta_c^{(k)} - \widehat{\beta}_m^{(k)} - \widehat{Z}_{cm} + \widehat{U}_{cm}) E_{c,m} \tag{4.35}$$

$$H(\beta_c^{(k)}) = -X^T W X - 2\rho (\sum_{m \in \mathcal{C}} E_{c,m}) I_p \tag{4.36}$$

$$\widetilde{\beta}_c^{(k)} = \hat{\beta}_c^{(k)} - H(\hat{\beta}_c^{(k)})^{-1} G(\hat{\beta}_c^{(k)}). \tag{4.37}$$

The solution to the optimization problem presented in (4.34) is a more general form of (4.7) where E is just a matrix where each off diagonal element is set equal to 1.

A new definition of $\mathcal{S}$ other than that used to develop Algorithm 5 requires a slightly different proof of a convergence and but the result applies the same.

Theorem 4.3

If $\mathcal{S}$ can be represented by a connected undirected graph and assumptions A1, A2, and A3 are met then the ADMM algorithm associated with (4.34) converges to the optimal objective function value, converges to the optimal values of $(\beta^{(k)}, Z)$, and the dual variable converges to the optimal dual variable. □

Proof 4.3

The proof for Theorem 4.3 is the exact same as that for Theorem 4.2 with the exception of proving bounded level sets. We now must show that

$$f_4(\beta^{(k)}) = \sum_{(c,m) \in \mathcal{S}} |\beta_c^{(k)} - \beta_m^{(k)}|_2 = \sum_{(c,m) \in \mathcal{C} \times \mathcal{C}} |\beta_c^{(k)} - \beta_m^{(k)}|_2 E_{cm} \tag{4.38}$$

has bounded level sets and then apply example 1.41 from Rockafellar and Wets (2004) to show that (4.34) has bounded level sets.

To show that $f_4(\beta^{(k)})$ has bounded level sets let $\hat{\beta} \in \{\beta : f_4(\beta) < \alpha)$, and let

$\hat{\beta}_C = \vec{0}$. Since the graph representation of E is connected there is a path on that graph between each pair of vertices (in this case represented by the categories). The edges on this graph correspond to the values of the E matrix. Let P_{cm} represent the shortest path between vertices c and m, and d_{cm} represent the length of this path. By definition $P_{cm} \subset \mathcal{S}$. Then

$$\begin{aligned}
|\hat{\beta}|_2 &\leq \sqrt{C}|\hat{\beta}|_1 \\
&= \sqrt{C}\sum_{c\in\mathcal{C}}|\hat{\beta}_c - \hat{\beta}_C|_1 \\
&\leq \sqrt{Cp}\sum_{c\in\mathcal{C}}|\hat{\beta}_c - \hat{\beta}_C|_2 \\
&\leq \sqrt{Cp}\sum_{c\in\mathcal{C}}\sum_{(l,j)\in P_{cC}}|\hat{\beta}_j - \hat{\beta}_l|_2 \\
&\leq \sqrt{Cp}\sum_{c\in\mathcal{C}} d_{cC}\alpha.
\end{aligned}$$

Resulting in the equality

$$|\hat{\beta}|_2 \leq \alpha\sqrt{Cp}\left(\sum_{c\in Cs} d_{cC}\right), \tag{4.39}$$

and proving that $f_4(\beta^{(k)})$ has bounded level sets. □

The final computational issue we address is the fact that Algorithm 6 is an iterative algorithm inside of Algorithm 5. We choose to use a block descent algorithm rather than other algorithms such as the Newton-Raphson method that have traditionally been used for simplicity and the fact that the ADMM does not require exact convergence (Boyd et al., 2010). Other options would be to use the GLMnet algorithm, or to jointly optimize with respect to the $\beta^{(k)}$ matrix.

4.3 Tuning Parameter Selection

Group fused multinomial regression is proposed as an exploratory data analysis technique to estimate the number of response categories, and the group structure in multinomial logistic regression. To select tuning parameters for this method, means to select the group structure for the model. We propose selecting the tuning parameter using a variant of a solution path algorithm combined with AIC. Since we are only concerned with the category combinations produced by the model, there are only C discrete models that are candidates. To decide between these models we will use a two step procedure to find the possible group structures, and then use AIC for evaluation. For a given λ we use Algorithm 5 to estimate the group structure, and return the coefficients $\tilde{\beta}^{(k)}$. Let $\mathcal{G} = \{g_1, \ldots, g_G\}$ denote the group structure, where the cardinality of $\mathcal{G}$ is the number of unique regression coefficient vectors in $\tilde{\beta}^{(k)}$, where $G \leq C$. The elements of $\mathcal{G}$ are the category indicies that are included in that group. We then fit the unpenalized multinomial regression model using the $\tilde{y}_{ig}$'s as the category counts. The resulting regression coefficients are denoted as $\eta_{\mathcal{G}}$. We assume that the solution for the unpenalized multinomial regression exists. A line search is used to find the λ's that correspond to the C possible group structures.

To compare the models we develop an AIC criterion based on the fusion of the model. The observed data is generated from the model defined by (4.1), while the model that is imposed by the group structure is

$$\tilde{Y}_i \sim \text{Multi}(n_i, \theta_{g_1}(x_i), \ldots, \theta_{g_G}(x_i)). \tag{4.40}$$

To understand the difference between the two data generating models assume that they represent the same underlying model, meaning that there are categories that have the same probability for every value of the predictor variables. Let $n_{g_j} = \sum_{i=1}^{N} \tilde{y}_{ig_j}$ then the relationship between the log-likelihood of (4.1) the estimated group structure

is

$$l(\pi) = \sum_{i=1}^{N} \sum_{c \in \mathcal{C}} y_{ic} \log(\pi_{ic}) \tag{4.41}$$

$$= \sum_{i=1}^{N} \sum_{j=1}^{G} \sum_{c \in \mathcal{C}} y_{ic} \log \left(\frac{\theta_{ig_j}}{\text{card}(g_j)} \right) I(c \in g_j), \tag{4.42}$$

$$= \sum_{i=1}^{N} \sum_{j=1}^{G} \sum_{c \in \mathcal{C}} y_{ic} \left(\log(\theta_{ig_j}) - \log(\text{card}(g_j)) \right) I(c \in g_j), \tag{4.43}$$

$$= \sum_{i=1}^{N} \sum_{j=1}^{G} \left(\sum_{c \in g_j} y_{ic} \right) \left(\log(\theta_{g_j}) - \log(\text{card}(g_j)) \right) \tag{4.44}$$

$$= \left(\sum_{i=1}^{N} \sum_{j=1}^{G} \tilde{y}_{ig_j} \log(\theta_{g_j}) \right) - \sum_{j=1}^{G} n_{g_j} \log(\text{card}(g_j)). \tag{4.45}$$

Define $l_{\mathcal{G}}(\theta)$ to be the log-likelihood associated with the model defined by (4.40), then (4.45) can be rewritten as

$$l(\pi) = l_{\mathcal{G}}(\theta) - \sum_{j=1}^{G} n_{g_j} \log(\text{card}(g_j)). \tag{4.46}$$

From this derivation the AIC proposed for selecting models with category combinations is

$$AIC(\eta_{\widehat{\mathcal{G}}}) = -2 \left(l_{\widehat{\mathcal{G}}}(\tilde{\theta}) - \sum_{j=1}^{G} n_{g_j} \log(\text{card}(g_j)) \right) + 2(p+1)(G-1), \tag{4.47}$$

where $\widehat{\mathcal{G}}$ is the estimated group structure from group fused multinomial regression and $\tilde{\theta} = \left(\tilde{\theta}_{g_1}, \ldots, \tilde{\theta}_{g_G} \right)$ are category probabilities calculated from the multinomial regression coefficients $\eta_{\widehat{\mathcal{G}}}$. We choose the $\widehat{\mathcal{G}}$ that corresponds to the minimum AIC.

Figure 4.1 illustrates an example of the solution path of group fused multinomial regression as λ increases in a four category example. In this example we have three

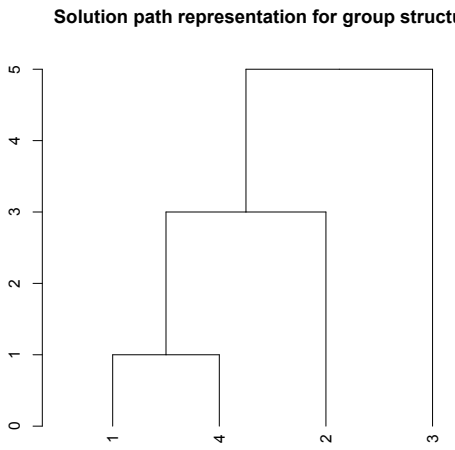

Figure 4.1: A dendrogram representation of the solution path produce by group fused multinomial regression.

groups, a group consisting of category 1 and 4, a group consisting of category 2, and a group consisting of category 3. The solution path shows that the first fusion to occur is that category 1 and 4 combine. From there this group is fused with category two, and then for larger λ category three. We see that the true model is in the solution path. Based on this solution path we fit the model with the different group structures and then use AIC to select the optimal group structure.

4.4 Practical Considerations

In this chapter we have proposed a novel method to combine categories for use in multinomial regression. In it's original form $\mathcal{S} = \mathcal{C} \times \mathcal{C}$ group fused multinomial regression should only be used for the purpose of exploratory data analysis. This leads to practical considerations, the first being that the category combinations interpretable. If they are not it may be hard to justify the multinomial or binomial

logistic regression the group structure indicates. This may lead to changes in how the user selects tuning parameters, or even the possible fusion that the user will allow in specification of the E matrix.

With this in mind, once the category combinations are selected the model should be refit to reflect the correct parameter estimates using the category combinations that were selected. The reason for refitting the model using unpenalized multinomial regression is that the estimates produced by group fused multinomial regression will be poor estimates of the coefficients. Though refitting results in a two step procedure it will give better parameter estimates of the final regression coefficients.

4.5 Simulations

4.5.1 Simulation Setup

In this section we show the merits of group fused multinomial regression to select the correct group structures for multinomial logistic regression. To select the tuning parameters needed we use the method proposed in Section 4.3. This simulation study investigates settings where the data is generated such that x_i is a realization of $X \sim N_9(0, I)$ and y_i is a realization of $Y_i \sim \text{Multi}\left(1, \pi_1(\tilde{x}_i), \pi_2(\tilde{x}_i), \pi_3(\tilde{x}_i), \pi_4(\tilde{x}_i)\right)$, where $\tilde{x}_i = (1, x_i)$ and $\pi_c(x)$ is found using (4.2). The fourth category is used as the baseline category.

We evaluate our method based on the ability to detect the correct group structure in each of the settings. In each of the settings it is possible to detect the correct number of groups but the incorrect structure, we will refer to these as correct and incorrect. It was also possible to select a group structure that had categories that had been partially combined, and were one combination away from selecting the correct structure, we refer to this situation as one-step away. In each setting we report the

Table 4.1: The fraction of the 100 replications specific group structures are selected for each N, δ combination. The label One-Step indicates that the correct group structure is still a possibility if the correct fusion was done at the next combination on the solution path.

	$N = 50$		$N = 75$	
	$\delta = 1$	$\delta = 3$	$\delta = 1$	$\delta = 3$
1 Group	3/100	0/100	0/100	0/100
2 Groups (Correct)	73/100	67/100	87/100	88/100
2 Groups (Incorrect)	0/100	0/100	0/100	0/100
3 Groups (One-Step)	22/100	12/100	10/100	11/100
3 Groups (Incorrect)	0/100	4/100	1/100	1/100
4 Groups	2/100	17/100	2/100	0/100

fraction of replications that detect each group structure of interest.

4.5.2 Detecting two groups

This simulation is designed to investigate the ability of group fused multinomial regression to detect the correct group structure when there are two groups. For this model $\beta_1^{(4)} = -\vec{\delta}$ and $\beta_2^{(4)} = \beta_3^{(4)} = \beta_4^{(4)} = \vec{0}$. The settings where $N = 50$ and 75 are investigated. We set $\delta = 1$ and 3 to indicate the amount of signal in the groups.

Table 4.1 reports the proportion of replications that select each number of groups for each N, δ combination. We see that for every of N and δ the method correctly picks the true number of groups more than any other group structure.

4.5.3 Detecting three groups

This simulation is designed to investigate the ability of group fused multinomial regression to detect the correct group structure of a problem with three groups in a four category problem. For this simulation we define $\beta_1^{(4)} = \beta_4^{(4)} = \vec{0}$, $\beta_2^{(4)} = -\vec{\delta}$, and $\beta_3^{(4)} = \vec{\delta}$. The cases where $N = 50$ and $\delta = 2$ and 3 were investigated. For each

Table 4.2: The fraction of 100 replications that selected group structures were selected for each δ, N combination for the simulation involving finding three groups in a four category problem.

	1 Group	2 Groups	3 Groups (Correct)	3 Groups (Incorrect)	4 Groups
$\delta = 2$	0/100	0/100	93/100	0/100	7/100
$\delta = 3$	0/100	0/100	98/100	0/100	2/100

Table 4.3: The fraction of 100 replications that selected each number of groups, for each N and δ combination for the simulation involving finding two groups of size two in a four category problem. The label One-Step indicates that the correct group structure is still a possibility if the correct fusion was done at the next combination on the solution path.

	$\delta = 5$	$\delta = 10$
1 Group	0/100	0/100
2 Groups (Correct)	58/100	54/100
2 Groups (Incorrect)	0/100	0/100
3 Groups (One-Step)	27/100	41/100
3 Groups (Incorrect)	12/100	2/100
4 Groups	3/100	3/100

of the 100 replications the number of optimal number of groups selected by AIC was recorded as was the selected group structure.

Table 4.2 gives report the fraction of group structures selected in the simulation for each δ. The results show that for both values of δ the correct group structure is selected most of the time. The results also show that no less than 3 groups were selected for any replication.

4.5.4 Detecting two groups of size two

This simulation investigates the ability of group fused multinomial regression to detect a two group structure where each group contains two categories. For this simulation

we set $\beta_1^{(4)} = \beta_4^{(4)} = \vec{0}$ and $\beta_2^{(4)} = \beta_3^{(4)} = \delta$. We looked at the settings where $\delta = 5$ and 10. For each replication $N = 100$. For each of the 100 replications the number of groups was recorded. Table 4.3 reports the fraction of replications that select specific group structures for each δ. The results show that in over half the replications of each δ the correct groups structure is selected. Since in this case one-step away indicates that one of the groups has been correctly combined Table 4.3 shows that atleast one group structure is correctly identified in 87 replications for $\delta = 5$ and 95 replications for $\delta = 10$. This indicates that the number of response categories is being reduced and a correct reduction is being made.

Chapter 5

Conclusions and Future Work

Throughout this work we have developed fusion penalized likelihood estimators to estimate the parameters needed for classification by quadratic discriminant analysis and multinomial logistic regression. Chapter 1 gives an introduction to regularization using fusion penalized likelihood estimators, and an example of fusion penalties for mean estimation. Chapter 2 and Chapter 3 introduce new methodology using ridge fusion penalties to estimate the inverse covariance matrices for use in classification by QDA. We do not claim to be the first to be interested in fusion penalties to estimate multiple inverse covariance matrices, but we are one of the first to be interested in the classification performance of these methods. While it may be of some interest to investigate the estimation performance of these methods we consider it beyond the scope of our work. A major contribution of the work presented in Chapter 2 is the cross validation procedures using validation likelihood for both the supervised and semi-supervised model. We not only show that this cross validation procedure better selects tuning parameters for the ridge fusion and common correlation method, but helps other methods as well. In the semi-supervised model the cross validation method is unique because it allows the unlabeled data to be used when selecting tuning parameters.

Chapter 4 moves away for the classification by QDA and investigates the use

of fusion penalties in multinomial logistic regression. We present a framework in which fusion penalties are invariant to the choice of the baseline category and discuss how non-fusion penalties (i.e. lasso, ridge, group lasso) can be rewritten as fusion penalties. We propose the group fused multinomial logistic regression model to help reduce the number of categories in the multinomial logistic regression model. An ADMM algorithm is proposed and we a detailed description to find the estimated group structure of the model. Tuning parameter selection is done using AIC that makes models with different response category size comparable.

While we discuss extensions of the work in Chapter 2 and Chapter 3 inside of the respective chapters, we believe there is work left to be done using the framework of Chapter 4. As previously mentioned using an l_1 fusion penalty rather than the group fused penalty could be useful for models where different variables matter when comparing different categories. A second direction would be to understanding how to find directions of recession for each of the group structures, and how it applies when the connected graph assumption of Theorem 4.2 is not met (Geyer, 2009).

An interesting direction of this work would be to investigate the relationship between fusion penalties in QDA and fusion penalties in multinomial logistic regression. The interest in this stems from Cook and Weisberg (1999), where the authors note that in binomial logistic regression if the conditional distribution of the predictors is $(X|Y = c) \sim N(\mu_c, \Theta_c)$ for each response category, the quadratic terms will be in regression the model. This begins to describe the relationship $(X|Y = c)$ and $(Y|X = x)$, two good discussions on this relationship can be found in Cook and Weisberg (1999) and Hastie et al. (2009). Here we will present an alternative way to think about this problem.

In Chapter 2 we presented the QDA classifier as assigning an observation x to the category, $c \in \mathcal{C}$ that maximizes $P(Y = c|X = x)$. A second way to think about the

QDA classifies is that x is assigned to the category, $c \in \mathcal{C}$ that satisfies

$$\log\left(\frac{P(Y=c|X=x)}{P(Y=m|X=x)}\right) > 0 \text{ for all } m \in \mathcal{C}\setminus\{c\}.$$

Writing out the comparison of category c and m explicitly we have that

$$\log\left(\frac{P(Y=c|X=x)}{P(Y=m|X=x)}\right) = K + x^T\alpha + .5x^T\Delta x, \tag{5.1}$$

where K is a constant depending on only the parameters of category c and m, $\alpha = -.5(\Theta_c\mu_c - \Theta_m\mu_m)$, and $\Delta = \Omega_2 - \Omega_1$. Using the fact that Ω_1 and Ω_2 are symmetric we have that

$$.5x^T\Delta x = .5\sum_{j=1}^{p}\sum_{k=1}^{p} x_j x_k \Delta_{jk} = \sum_{j=1}^{p}\sum_{k=j}^{p} x_j x_k \Delta_{jk} \tag{5.2}$$

Looking at (5.1) we can recognize this as the form of the full second order regression model, where K is the intercept, α represents the first order regression coefficients, and Δ_{jk} represents the second order regression coefficient associated with $x_j x_k$. The coefficients associated with higher order terms in this regression are Δ which are element wise difference of the inverse covariance matrices. We hope to investigate there relationship between fusion penalized likelihood estimates for inverse covariance matrices in QDA and fusion penalties in multinomial logistic regression with second order terms.

References

Alaiz, C., Barbero, A., and Dorronsoro, J. (2013). Group fused lasso. In *Leture Notes in Computer Science*, volume 8131, pages 66–73. Artifical Neural Networks and Machine Learning.

Anderson, J. and Blair, V. (1982). Penalized maximum likelihood estimation in logistic regression and discriminantion. *Biometrika*, 69(1):123–136.

Bache, K. and Lichman, M. (2013). UCI machine learning repository. http://archive.ics.uci.edu/ml.

Basu, S., Banerjee, A., and Mooney, R. (2002). Semi-supervised clustering by seeding. In *Proceedings of the 19th International Conference on Machine Learning*, pages 19–26.

Boser, B. E. and et al. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152. ACM Press.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2010). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(1):123–140.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Chapelle, O., Scholkoph, B., and Zien, A. (2006). Introduction to semi-supervised learning. In Chapelle, O., Scholkoph, B., and Zien, A., editors, *Semi-Supervised Learning*, chapter 1. The MIT Press.

Cook, R. and Weisberg, S. (1999). *Applied Regression Including Computing and Graphics*. John Wiley & Sons.

Danaher, P. (2013). *JGL: Performs the Joint Graphical Lasso for sparse inverse covariance estimation on multiple classes*. R package version 2.3.

Danaher, P., Wang, P., and Witten, D. (2013). The joint graphical lasso for inverse covariance estimation across multiple classes. *The Journal of Royal Statistical Society, Series B*, 76(2):373–397.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the em algorithm (with discussion). *Journal of The Royal Statistical Society*, 39(1):1–38.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(7):179–188.

Fraley, C. and Raftery, A. (2002). Model based clustering, discriminant analysis, and density estimation. *Journal of American Statistical Association*, 97(458):611–632.

Friedman, J. (1989). Regularized discriminant analysis. *Journal of the American Statistical Association*, 84:249–266.

Friedman, J., Hastie, T., Hfling, H., and Tibshirani, R. (2007a). Pathwise coordinate optimization. Technical report, Stanford University.

Friedman, J., Hastie, T., and Tibshirani, R. (2007b). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441.

Friedman, J., Hastie, T., and Tibshirani, R. (2008). Regularized paths for genearlized linear models via coordinate descent. *Journal of Statistial Softwawre*, 33(1).

Geyer, C. (2009). Likelihood inference in exponential families and directions of recession. *Electronic Journal of Statistics*, 3:259–289.

Green, P. (1990). On use of the em for penalized likelihood estimation. *Journal of the Royal Statistical Society. Series B*, 52(3):443–452.

Guo, J., Levina, E., Michailidis, G., and Zhu, J. (2011). Joint estimation of multiple graphical models. *Biometrika*, 98(1):1–15.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition.

Hoefling, H. (2010). A path algorithm for the fused lasso signal approximator. *Journal of Computational and Graphical Statistics*, 19(4):984–1006.

Horel, A. and Kennard, R. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12:55–67.

Hosmer, D. (1973). A comparison of iterative maximum likelihood estimates of the parameters of a mixture of two normal distributions under three different types of sample. *Biometrika*, 29(4):761–770.

Huang, J., Liu, N., Pourahmadi, M., and Liu, L. (2006a). Covariance matrix selection and estimation via penalised normal likelihood. *Biometrika*, 93(1):85–98.

Huang, J. Z., Liu, N., Pourahmadi, M., and Liu, L. (2006b). Covariance matrix selection and estimation via penalised normal likelihood. *Biometrika*, 93(1):85–98.

Kass, G. (1980). An Exploratory Technique for Investigating Large Quantities of Categorical Data. *Journal of the Royal Statistical Society: Series C*, 29(2):119–127.

Kaufman, L. and Rousseuw, P. J. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley.

Land, S. and Friedman, J. (1996). Variable fusion: A new adaptive signal regression. Technical report, Department of Statistics, Stanford University.

Liu, J., Yuan, L., and Ye, J. (2010). An efficient algorithm for a class of fused lasso problems. In *In ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Mazumder, R. and Hastie, T. (2012). The graphical lasso: New insights and alternatives. *Electronic Journal of Statistics*, 6:2125–2149.

Meier, L., Geer, S. V. D., and Buhlmann, P. (2008). The group lasso for logistic regression. *Journal of the Royal Statstical Society: Series B*, 70(1):53–71.

Mota, J., Xavier, J., Aguiar, P., and Puschel, M. (2011). A proof of convergence for the alternating method of multipliers applied to polyhedral-constrained functions.

Nelder, J. and Wedderburn, R. (1972). Generalized linear models. *Journal of the Royal Statistical Society: Series A*, 135:370–384.

Park, M. Y. and Hastie, T. (2007). Penalized logistic regression for detecting gene interactions. *Biostatistics*, 9(1):30–50.

Pourahmadi, M. (2011). Covariance estimation: The glm and regularization perspective. *Statistical Science*, 26(3):369–387.

Price, B. S. (2014). *RidgeFusion: R Package for Ridge Fusion in Statistical Learning*. R package version 1.0-3.

Price, B. S., Geyer, C. J., and Rothman, A. J. (2014). Ridge fusion in statistical learning. *Journal of Computational and Graphical Statistics*, To Appear.

Rinaldo, A. (2009). Properties and refinements of the fused lasso. *Annals of Statistics*, 37(5B):2597–3097.

Rockafellar, R. and Wets, R.-B. (2004). *Variational Analysis*, volume 317 of *A Series of Comprehensive Studies in Mathematics*. Springer, 2 edition.

Rothman, A., Bickel, P., Levina, E., and Zhu, J. (2008). Sparse permutation invariant covariance estimation. *Electronic Journal of Statistics*, 2:494–515.

Rothman, A. J. and Forzani, L. (2013). Properties of optimizations used in peanlized gaussian likelihood inverse covariance matrix estimation. Manuscript.

Ruan, L., Yuan, M., and Zou, H. (2011). Regularized parameter estimation in high-dimensional gaussian mixture models. *Neural Computation*, 23(6):1605–1622.

Simon, N., Friedman, J., and Hastie, T. (2014). A blockwise descent algortihm for group-penalized multiresponse and multinomial regression. http://statweb.stanford.edu/~jhf/ftp/noah.pdf.

Tibshirani, R. (1994). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288.

Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., and Knight, K. (2005). Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society Series B*, pages 91–108.

Weihs, C., Ligges, U., Luebke, K., and Raabe, N. (2005). klar analyzing german business cycles. In Baier, D., Decker, R., and Schmidt-Thieme, L., editors, *Data Analysis and Decision Support*, pages 335–343, Berlin. Springer-Verlag.

Witten, D., Friedman, J., and Simon, N. (2011). New insights and faster computations for the graphical lasso. *Journal of Computational and Graphical Statistics*, 20(4).

Witten, D. and Tibshirani, R. (2009). Covariance regularized regression and classification for high-dimensional problems. *Journal of Royal Statistical Society*, 71(3):615–636.

Wu, C. J. (1983). On the convergence properties of the em algorithm. *Annals of Statistics*, 11(1):95–103.

Xie, B., Pan, W., and Shen, X. (2008). Penalized model based clustering with cluster specific diagonal covariance matrices and grouped variables. *Electronic Journal of Statistics*, 2:168–212.

Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 68:49–67.

Zhou, H., Pan, W., and Shen, X. (2009). Penalized model-based clustering with unconstrained covariance matrices. *Electronic Journal of Statistics*, 3:1473–1496.

Zhu, J. and Hastie, T. (2004). Classification of gene microarrays by penalized logisitic regression. *Biostatistics*, 5(3):427–443.

Zhu, X. (2008). Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320.