# Feature Selection and Post-Selection Statistical Inference in Multinomial Models

Matthew Pencer

Department of Mathematics and Statistics

McGill University, Montreal

April 18, 2016

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements of the degree of Master of Science

# ABSTRACT

This thesis presents a detailed study of multinomial regression, with a special focus on its application to high-dimensional datasets. After reviewing the basic properties of the multinomial model, the concept of sparsity - when many of the regression coefficients are zero - is introduced. This is followed by an overview of several penalized estimators and the distinction between "grouped" and "ungrouped" penalties. Using the elastic net and adaptive lasso penalties, the performances of the grouped and ungrouped penalties are compared when applied to both an artificial dataset as well as a real dataset dealing with voice recognition.

The second part of this thesis deals with post-selection inference - that is, assessing the significance of covariates that have been selected by a penalized estimator. This thesis demonstrates the danger of performing classical significance tests on models which had undergone variable selection, and uses simulations to quantify its inaccuracy. Next we describe a "valid" method of post selection inference, the "desparsified" lasso estimator (first proposed by [24], and extended to general linear models by [2]), which can be used to create true confidence intervals using models selected using the lasso. Finally, we extend this method to the case of a multinomial model selected via a grouped lasso penalty.

# ABRÉGÉ

Cette thèse présente une étude détaillée de régression multinomiale, avec un accent particulier sur son application aux ensembles de données de grande dimension. Après avoir examiné les propriétés du modèle multinomial, le concept de “sparsité” - quand la plupart des coefficients de régression sont nuls - est introduit. Cette discussion est suivie par un aperçu de plusieurs estimateurs pénalisés et la distinction entre des penalités “groupées” et “dégroupées”. Avec les penalités “elastic net” et lasso adaptatif, les performances des penalités groupées et dégroupées sont comparées lorsqu'elles sont appliquées à un ensemble de données artificielle ainsi qu'un ensemble de données réelles traitant de la reconnaissance vocale.

La deuxième partie de cette thèse traite de l'inférence post-sélection - l'évaluation de l'importance des variables qui ont été sélectionnés par un estimateur pénalisé. Cette thèse démontre le danger d'effectuer des tests de signification classiques sur les modèles qui avaient subi une sélection de variables, et utilise des simulations pour quantifier son imprécision. Ensuite, nous décrivons un méthode de sélection “valide” de poste inférence, le “desparsified lasso” (d'abord proposée par [24], et étendu à des modèles linéaires généraux par [2]), qui peut être utilisé pour créer des intervalles de confiance en utilisant des modèles sélectionnés à l'aide du lasso. Enfin, nous étendons cette méthode pour le cas d'un modèle multinomial sélectionné via une pénalité lasso groupés.

# ACKNOWLEDGEMENTS

I would like to thank my supervisors, Professor Abbas Khalili and Professor Masoud Asgharian, for their assistance and supervision during the writing of this thesis and throughout my graduate studies.

# Contents

**ABSTRACT** ii

**ABRÉGÉ** iii

**ACKNOWLEDGEMENTS** iv

**LIST OF FIGURES** vii

**I Multinomial Regression** 1

**1 Introduction** 2

1.1 Previous Research . . . . . 4

1.2 Original Contributions of this Thesis . . . . . 5

**2 Sparsity, Likelihood, and Penalization** 6

2.1 Assumptions . . . . . 9

2.2 Sparsity . . . . . 10

2.3 Likelihood Function and Maximum Likelihood Estimator . . . . . 13

2.4 Penalized Estimators . . . . . 15

**3 Comparison of Grouped and Ungrouped Penalization Methods** 21

3.1 Simulations - Artificial Dataset . . . . . 22

3.1.1 Methodology . . . . . 22

3.1.2 Prediction . . . . . 25

3.1.3 Results . . . . . 27

3.2 Experiment - Real Dataset . . . . . 32

3.2.1 Description of the Dataset . . . . . 32

3.2.2 Prediction . . . . . 33

3.2.3 Results . . . . . 33

**4 Generalization to Other Penalties** **34**

## II Post-Selection Inference 38

**5 Naive Methods of Performing Inference** **39**

5.1 Forward Selection . . . . . 39

5.2 Lasso . . . . . 41

5.3 Naively Estimating P-Values . . . . . 42

**6 Finding Emperical P-Values** **44**

6.1 Logistic Regression . . . . . 46

**7 Finding a Valid "Desparsified" Lasso Estimator** **47**

7.1 Verification . . . . . 49

7.2 Desparsified Estimator for GLMs . . . . . 49

7.3 Group Lasso . . . . . 50

7.4 Extension to Multinomial Regression . . . . . 53

## III Conclusion and Future Work 56

**8 Conclusion** **57**

8.1 Future Work . . . . . 57

## IV Appendix and References 59

**9 Full Results** **60**

**10 R Code** **63**

**11 References** **63**

# List of Figures


1 Effect of $p$, $K$ on misclassification rate when $n = 50$ . . . . . 29

2 Effect of $p$, $K$ on misclassification rate when $n = 100$ . . . . . 30

3 Effect of $p$, $K$ on misclassification rate when $n = 200$ . . . . . 31

4 P-values "naively" assigned to various values of $|\hat{\beta}|$ . . . . . 43

5 P-values "naively" assigned to various t-scores . . . . . 43

6 Empirical p-values assigned to various values of $|\hat{\beta}|$ . . . . . 45

7 Empirical p-values assigned to various t-scores . . . . . 46

8 P-values "naively" assigned to various t-scores and values of $|\hat{\beta}|$ . . . . . 47

9 Empirical p-values assigned to various t-scores and values of $|\hat{\beta}|$ . . . . . 47

# Part I

# Multinomial Regression

# 1 Introduction

Logistic Regression, which is used to model a dataset where the response is binary, has been well-studied. Its "big brother", *Multinomial Logistic Regression* or *Multinomial Regression*, is used for prediction of a *categorical* response, where the response falls in one of $K \geq 3$ classes (as opposed to logistic regression, where $K = 2$). For example, multinomial regression can be used to predict a students course selection, a potential customer's favourite colour, or the preferred candidate of a voter (in an election with three or more major candidates). While similar, in that both use maximum likelihood estimation to perform classification, multinomial regression has some added complexities.

In spite of its usefulness, multinomial regression has been far less studied than logistic regression. Few have examined how multinomial regression behaves when the predictors are of large dimension (large $p$), especially with respect to variable selection.

This thesis begins with an overview of the notation and definitions of the multinomial model. Next, the concept of *sparsity* is introduced, where we present an overview of the different types of sparsity, along with their respective motivations.

Once the reasoning behind multinomial regression has been introduced, a mathematical overview of the likelihood function is presented, along with the score functions and Hessian matrix. Naturally, this is followed by a short description of the maximum likelihood estimator, the simplest and most used method of estimating the coefficients of a multinomial model.

As this thesis focuses on high-dimensional models, more attention is given to *penalized estimators*, rather than the maximum likelihood estimator. As the number of covariates grows, the "full model" becomes less and less effective, due to most predictors being "inactive" (having no effect on the response). Among many problems, prediction power may decrease due to excessive noise, interpretibility becomes impossible if a human is trying to understand the effects of thousands of covariates, and computation can become overwhelmingly complex. The need for *variable selection* has led to the invention of a wide array of penalized estimators [5].

Specifically, when dealing with a multinomial model, the regression coefficients are in the form of a matrix, rather than a vector. Therefore, the desire for a *row-sparse* matrix of coefficients motivates the use of *grouped* penalized estimators, which perform variable selection on a row-wise rather than element-wise basis. A detailed mathematical overview of multiple types of penalized estimators, including the grouped and ungrouped versions of the lasso [22], elastic net [10], SCAD [4], MCP [23], and Adaptive Lasso [26], is provided, along with their respective pros and cons.

While the group-lasso [17] and its application to multinomial regression [6] has been relatively well studied, the grouped version of the Adaptive Lasso is less well-known. Although it has been discussed and implemented for linear regression ([11] and [15]), its application to multinomial regression has yet to been studied. The implementation is described in this thesis. The code to implement this algorithm in `R` is also given in the appendix.

To provide evidence to back up our intuitive reasoning of why a row-sparse matrix is preferable, a simulation study is performed. After creating an artificial multinomial dataset, we estimate the coefficients using the maximum likelihood estimator, the lasso, elastic net, and Adaptive Lasso, as well as their respective grouped versions. By computing the misclassification rate, we can confirm whether or not using a grouped method (for the purpose of outputting a row-sparse estimated matrix of coefficients) is in fact superior to using an ungrouped penalized estimator. This simulation study is followed by an application of the described methods to a real-life multinomial dataset, and a comparison of the performances of the various estimators.

After exploring the empirical performance of grouped and ungrouped versions of the lasso, it is of interest to study its theoretical properties. Although the Lasso is very useful in performing regularization and variable selection, its "Achilles heel" lies in the difficulty in assessing the significance of the lasso-selected parameters, as classical significance tests do not take the variability inherent in variable selection into account.

First, we demonstrate the danger of performing classical significance tests on models selected through regularization. For simplicity, this will be demonstrated on linear models. In Section

8.1, we look at what happens when inference is performed on a model selected through forward selection. Although this thesis focuses on the lasso, the simplicity of forward selection provides an easy "case study" of the effects of variable selection on hypothesis testing. The experiment is repeated on a linear regression model selected using the lasso, with similar results.

Next, we begin to quantify the effect of variable selection on classical significance tests (specifically, the t-score and p-value). In Section 9, the p-values are calculated directly from the t-scores, despite the warnings of Section 8 that the results will not be accurate. We call this the "Naive Method" of estimating p-values.

In Section 10, we propose estimating the p-values emperically, rather than via the t-scores (which are shown to be inaccurate). This method provides us with accurate (or "valid") p-values; however, it is often impractical due to its computationally intensive nature.

In Section 11, we describe another method to produce valid p-values via a "desparsified" lasso estimator, using a method first proposed by Zhang & Zhang [24]. Furthermore, we describe the contributions of van de Geer et al [2], who showed how to extend this method to General Linear Models and to applying joint significance tests. Using the results of van de Geer et al, we extend this method to the Group Lasso and finally to Multinomial Regression, which is the focus of this thesis.

## 1.1 Previous Research

There has recently been an abundance of research on penalized estimators for high-dimensional models. Tibshirani first proposed the Lasso [22] in 1996, and demonstrated how the $L_1$ penalty performs variable selection. Since then, many other penalties have been proposed. Fan and Li [4] introduced the *SCAD* penalty. They showed how SCAD has the *oracle* property, meaning that the penalized estimator is optimal in the sense that it performs as if the active variables are known. This is elaborated upon by Huang and Xie [12]. Zhang [23] introduced the *MCP*, which is similar to SCAD.

Zou and Hastie [10] showed how the Lasso performs poorly when there is high correlation

between predictors or when $p >> n$. They extended the Lasso by adding an $L_2$ penalty term to create the *Elastic Net* penalty, which performs better than the Lasso in many cases while maintaining the variable selection property.

Another penalized estimator that posseses the oracle property is the *Adaptive Lasso*, proposed by Zou in 2006 [26]. Adaptive Lasso has the advantage of being far simpler than the SCAD and MCP penalties, which is why this thesis will extend it to multinomial regression.

Since the "grouped" versions of penalized estimators for multinomial regression work by "grouping" sets of coefficients (all of which correspond to the same covariate), they are all special cases of the *Group Lasso* [17]. Fortunately, much research has been done on the group lasso, notably by Simon, Friedman, and Hastie [6], who gave a coordinate descent algorithm to find the group lasso estimator of a multinomial model. Furthermore, they extended their algorithm to compute the grouped lasso estimator for multinomial regression. Their methods have been implemented in `R` with the `glmnet` library [7].

The SCAD and MCP penalties have also been generalized to a "grouped" version, although not in the case of multinomial regression. Breheny and Huang [1] describe how SCAD and MCP can be extended to a group selection problem, and gave algorithms to compute the group-penalized estimators. In his 2012 doctoral dissertation, Jiang [13] proposed a new algorithm called *Majorization Minimization by Coordinate Descent*, or *MMCD*, to compute the SCAD and MCP penalties. He extended this method to multinomial regression, albeit not for the case of group-selection.

Finally, the second part of this thesis relies strongly on the research of Zhang & Zhang [24] and van de Geer et al [2], who introduced methods to produce valid p-values for models selected using a lasso penalty.

## 1.2 Original Contributions of this Thesis

This thesis extends the Grouped Adaptive Lasso, Smoothly Clipped Absolute Deviation (SCAD), and Minimax Concave Penalty (MCP) penalties to multinomial models. This the-

sis also compares the performances of grouped and ungrouped elastic net estimators for multinomial models, providing empirical evidence that grouped penalties are usually superior. These experiments are performed on both artificial and real datasets.
In addition to reviewing the concept of post-selection inference and performing simulations to motivate its use, this thesis extends the desparsified lasso estimator to multinomial regression. This is accompanied by a simulation (performed in `R`) that verifies the validity of the desparsified estimator.

# 2 Sparsity, Likelihood, and Penalization

We will begin this section with a short overview of the notation that will be used throughout the thesis.

**Notation** Let $\boldsymbol{X}$, the predictor matrix, be an $n \times p$ matrix, where $\boldsymbol{x_i} = \{x_{i1}, x_{i2}, ..., x_{ip}\}$ is the $i$-th row of $\boldsymbol{X}$, ie the $i$-th observation. The response vector $\boldsymbol{Y} = \{Y_i, Y_2, ..., Y_n\}$ is an $n \times 1$ vector, where the $i$-th response $Y_i$ is categorical, taking a value $y_i \in \{1, 2, ..., K\}$ for $K \geq 3$.
If there are $p$ variables and $K$ classes, then we require that $\boldsymbol{\beta}$ be a $p \times K-1$ matrix of coefficients,

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_{1,1} & ... & \beta_{1,K-1} \\ \beta_{2,1} & ... & \beta_{2,K-1} \\ ... & ... & ... \\ \beta_{p,1} & ... & \beta_{p,K-1} \end{pmatrix}$$

We will refer to the $j$-th row of $\boldsymbol{\beta}$ by $\boldsymbol{\beta_{j\cdot}}$ and the the $k$-th column of $\boldsymbol{\beta}$ by $\boldsymbol{\beta_{\cdot k}}$.
There is also a $1 \times K-1$ vector of intercepts,

$$\boldsymbol{\beta_{0\cdot}} = \begin{pmatrix} \beta_{0,1} & ... & \beta_{0,K-1} \end{pmatrix}$$

**Probability Model** When performing logistic regression, we model the probabilities by making the *log-odds* a linear function of the predictor matrix $\boldsymbol{X}$. Each row of $\boldsymbol{X}$, $\boldsymbol{x_i}$, repre-

sents a single observation of $p$ covariates.

Specifically,

$$\log(\frac{P(Y_i = 1|\boldsymbol{x_i})}{P(Y_i = 0|\boldsymbol{x_i})}) = \beta_0 + \boldsymbol{x_i}\boldsymbol{\beta^T} \tag{2.1}$$

This form is chosen because the log-odds, unlike $P(Y_i = 1|\boldsymbol{x_i})$ or the untransformed odds, is unbounded, allowing a probability to be assigned to any value $\boldsymbol{x_i}\boldsymbol{\beta^T} \in (-\infty, \infty)$.

The probabilities for multinomial models can be written in a similar fashion. If there are $K$ classes, $y_i \in \{1, ..., K\}$, we can allow one of the classes to be the *pivot*, a role similar to that of $y_i = 0$ in the logistic model. By convention, the "last" class $K$ is chosen to be the pivot. Then for $k \in \{1, ..., K-1\}$, we can use the predictor matrix to model the partial odds $\dfrac{P(Y_i = k|\boldsymbol{x_i})}{P(Y_i = K|\boldsymbol{x_i})}$. This requires the modelling of $K-1$ probabilities (rather than $K$), and therefore $K-1$ different vectors of coefficients, $\boldsymbol{\beta_1}, ..., \boldsymbol{\beta_{K-1}}$, each of which is a $p \times 1$ vector. Specifically, our probability model will be

$$\log(\frac{P(Y_i = 1|\boldsymbol{x_i})}{P(Y_i = K|\boldsymbol{x_i})}) = \beta_{0,1} + \boldsymbol{x_i}\boldsymbol{\beta_1} \tag{2.2}$$

$$...$$

$$...$$

$$...$$

$$\log(\frac{P(Y_i = K-1|\boldsymbol{x_i})}{P(Y_i = K|\boldsymbol{x_i})}) = \beta_{0,K-1} + \boldsymbol{x_i}\boldsymbol{\beta_{K-1}}$$

By taking the exponents, we can solve for the individual probabilities

$$P(Y_i = k|\boldsymbol{x_i}) = P(Y_i = K|\boldsymbol{x_i})e^{\beta_{0,k} + \boldsymbol{x_i}\boldsymbol{\beta_k}} \tag{2.3}$$

For $k = 1, ..., K-1$.

Using the fact that all the probabilities must sum to one, we can solve explicitly:

$$P(Y_i = k|\boldsymbol{x_i}) = \frac{e^{\beta_{0,k} + \boldsymbol{x_i}\boldsymbol{\beta_k}}}{1 + \sum_{\ell=1}^{K-1} e^{\beta_{0,\ell} + \boldsymbol{x_i}\boldsymbol{\beta_\ell}}}, \quad k = 1, ..., K-1 \tag{2.4}$$

$$P(Y_i = K|\boldsymbol{x_i}) = \frac{1}{1 + \sum_{\ell=1}^{K-1} e^{\beta_{0,\ell} + \boldsymbol{x_i}.\boldsymbol{\beta_{.\ell}}}} \tag{2.5}$$

**Alternate Form** In some cases, the following probability equations are used:

$$P(Y_i = k|\boldsymbol{x_i}) = \frac{e^{\beta_{0,k} + \boldsymbol{x_i}.\boldsymbol{\beta_{.k}}}}{\sum_{\ell=1}^{K} e^{\beta_{0,\ell} + \boldsymbol{x_i}.\boldsymbol{\beta_{.\ell}}}}, \quad k = 1, ..., K \tag{2.6}$$

This is the form used (and computed) by the `glmnet R` package [7] (see [9] for more details). While simpler, this requires $K$ different $p \times 1$ vectors $\boldsymbol{\beta_{.k}}$. Furthermore, the $p \times K$ matrix $\boldsymbol{\beta}$ will not be of full rank, as a column $\boldsymbol{\beta_{.k}}$ can be written as a function of the other $K - 1$ columns.

Therefore, if we are given a $p \times K$ matrix $\boldsymbol{\beta'}$ (for example, this can be the output of the `glmnet` package), it may be of interest to calculate the corresponding $p \times K - 1$ matrix $\boldsymbol{\beta}$. We can find these by comparing the two different probability equations.

For $k \in \{1, ..., K - 1\}$,

$$P(Y_i = k|\boldsymbol{x_i}) = \frac{e^{\beta'_{0,k} + \boldsymbol{\beta'_{.k}}\boldsymbol{x_i}}}{\sum_{\ell=1}^{K} e^{\beta'_{0,\ell} + \boldsymbol{\beta'_{.\ell}}\boldsymbol{x_i}}} = \frac{e^{\beta_{0,k} + \boldsymbol{\beta_{.k}}\boldsymbol{x_i}}}{1 + \sum_{\ell=1}^{K-1} e^{\beta_{0,\ell} + \boldsymbol{\beta_{.\ell}}\boldsymbol{x_i}}} \tag{2.7}$$

$$P(Y_i = K|\boldsymbol{x_i}) = \frac{e^{\beta'_{0,K} + \boldsymbol{\beta'_{.K}}\boldsymbol{x_i}}}{\sum_{\ell=1}^{K} e^{\beta'_{0,\ell} + \boldsymbol{\beta'_{.\ell}}\boldsymbol{x_i}}} = \frac{1}{1 + \sum_{\ell=1}^{K-1} e^{\beta_{0,\ell} + \boldsymbol{\beta_{.\ell}}\boldsymbol{x_i}}} \tag{2.8}$$

This gives us

$$\frac{P(Y_i = k|\boldsymbol{x_i})}{P(Y_i = K|\boldsymbol{x_i})} = e^{\beta_{0,k} + \boldsymbol{\beta_{.k}}\boldsymbol{x_i}} = \frac{e^{\beta'_{0,k} + \boldsymbol{\beta'_{.k}}\boldsymbol{x_i}}}{e^{\beta'_{0,K} + \boldsymbol{\beta'_{.K}}\boldsymbol{x_i}}} \tag{2.9}$$

Or, for $k \in \{1, ..., K - 1\}$,

$$\beta_{0,k} + \boldsymbol{\beta_{.k}}\boldsymbol{x_i} = \beta'_{0,k} + \boldsymbol{\beta'_{.k}}\boldsymbol{x_i} - \beta'_{0,K} - \boldsymbol{\beta'_{.K}}\boldsymbol{x_i} \tag{2.10}$$

This holds $\forall \boldsymbol{X}$ and $\forall k$, so

$$\beta_{j,k} = \beta'_{j,k} - \beta'_{j,K} \quad \forall j \in \{0, 1, ..., p\}, k \in \{1, 2, ..., K - 1\} \tag{2.11}$$

**Choice of Reference Class** When estimating the coefficients using the maximum likelihood estimator, the choice of reference class is irrelevant with respect to prediction. However, the same does not hold when using penalized estimators [8]. In this case, the choice of reference class can affect prediction and the "alternate form" should be used [8]. This thesis will therefore use the alternate form when dealing with models chosen via a penalized estimator.

## 2.1 Assumptions

Due to its flexibility, multinomial regression is very attractive - Starkweather and Moske [19] noted that unlike more powerful alternatives such as discriminant function analysis, the assumption of normality, linearity, and homoscedasticity are *not* required.

**Independence of Irrelevant Alternatives** A major assumption is that of "Independence of Irrelevant Alternatives" (IIA), which requires that adding a $K+1$-th class does not affect the relative probabilities of the response falling in certain classes.
Specifically, for all $k, \ell \in \{1, ..., K\}$, the relative probabilities

$$\frac{P(Y_i = k|\boldsymbol{X_i} = \boldsymbol{x_i})}{P(Y_i = \ell|\boldsymbol{X_i} = \boldsymbol{x_i})} \tag{2.12}$$

do *not* change if an additional class is added. One should be aware of this requirement before performing multinomial regression.
A common example of this requirement being violated is if the response is a person's preferred mode of transportation, where the options (categories) are a car or a blue bicycle. Say the odds ratio of these choices is $1:1$. If the option of a red bicycle is introduced (and the person does not care about the colour), the odds ratio of car to blue bicycle will decrease to $1:0.5$ (and the odds ratio of car to red bicycle will also be $1:0.5$).
For more information about the IIA and Diagnostic Tests, see [18].

## 2.2 Sparsity

While the last section provided a theoretical overview of the multinomial model, we failed to mention its main drawback. As a matrix, $\boldsymbol{\beta}$ is far more difficult to interpret, both from a human and computational standpoint, than a simple vector of coefficients. Furthermore, when compared to the case of a binomial response, a model with the same number of covariates has $K-1$ times more coefficients when the response is multinomial with $K$ classes. Therefore, it is natural to try to find patterns in the matrix of coefficients. This is especially important when $p$ is large. As a matrix with $p \times K-1$ coefficients, it is relatively difficult to interpret individual elements of $\boldsymbol{\beta}$. Recall that

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_{1,1} & \dots & \beta_{1,K-1} \\ \beta_{2,1} & \dots & \beta_{2,K-1} \\ \dots & \dots & \dots \\ \beta_{p,1} & \dots & \beta_{p,K-1} \end{pmatrix}$$

Consider $\beta_{j,k} = 0$. Since $P(Y_i = k|\boldsymbol{X_i} = \boldsymbol{x_i}) = P(Y_i = K|\boldsymbol{X_i} = \boldsymbol{x_i}) \exp(\beta_{0,k} + \sum_{j=1}^{p} \beta_{j,k} x_{i,j})$, this implies that $\dfrac{P(Y_i = k|\boldsymbol{X_i} = \boldsymbol{x_i})}{P(Y_i = K|\boldsymbol{X_i} = \boldsymbol{x_i})}$ does not depend on $x_{i,j}$. In other words, the $j$-th covariate is of no help in distinguishing between the $k$-th and $K$-th classes.

Since this thesis deals with variable selection, it is natural to ask if having elements set to exactly 0 is a desirable property in the context of multinomial regression. Unlike a linear or logistic model where $\boldsymbol{\beta}$ is one-dimensional, the concept of sparsity is not as straightforward. When $\boldsymbol{\beta}$ is a vector, sparsity is defined by having many elements of $\boldsymbol{\beta}$ being zero, or relatively few coefficients being nonzero.

The simplest extension of sparsity to a matrix setting would be what we call an *element-sparse* matrix, where many entries in the matrix are zero. However, we will show that a *row-sparse* matrix, where entire rows of $\boldsymbol{\beta}$ are zero, is more desireable. We will also provide a short description of a *column-sparse* matrix. While the latter is not as useful as a row-sparse matrix in most applications to high-dimensional models, it does possess some interesting properties. These are discussed in more detail in the "Future Work" section of this thesis.

**Element-Sparse Matrix** Consider a matrix which is *element-sparse*. Instead of entire columns being zero, many individual entries throughout the matrix are zero.

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_{1,1} & ... & \beta_{1,k} & ... & 0 & ... & \beta_{1,K-1} \\ \beta_{2,1} & ... & 0 & ... & \beta_{2,\ell} & ... & \beta_{2,K-1} \\ ... & ... & 0 & ... & 0 & ... & ... \\ \beta_{p,1} & ... & \beta_{p,k} & ... & \beta_{p,\ell} & ... & \beta_{p,K-1} \end{pmatrix}$$

While there may be few non-zero entries in $\boldsymbol{\beta}$, the model corresponding to $\boldsymbol{\beta}$ is not at all simple since $\boldsymbol{\beta_{i\cdot}} \neq \mathbf{0}\ \forall i$ and therefore every covariate $\boldsymbol{x_i}$ must be kept in the model. All we may say is that for certain combinations of $j$ and $k$, the $j$-th covariate is of no help distinguishing the $k$-th class from the $K$-th class. Unfortunately, this information is not very useful in the context of variable selection.

In the case of high-dimensional data, an element-sparse matrix is also unrealistic, since it implies that all $p$ covariates have an effect on the response. Yet some covariates have zero effect on the probability of certain classes of the response (but not others) - for example, the second coveriate has an effect on the probability of the response being in the 5-th class but not the 6-th class, even though the classes are not independent.

**Row-Sparse Matrix** A much more desirable outcome is that $\boldsymbol{\beta}$ be *row-sparse*.

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_{1,1} & ... & \beta_{1,K-1} \\ \beta_{2,1} & ... & \beta_{2,K-1} \\ ... & ... & ... \\ 0 & ... & 0 \\ ... & ... & ... \\ 0 & ... & 0 \\ \beta_{p,1} & ... & \beta_{p,K-1} \end{pmatrix}$$

Specifically, we have that for certain $j \in \{1, ..., p\}, \boldsymbol{\beta_{j\cdot}} = \mathbf{0}$. This implies that the $j$-th covariate has no effect whatsoever on the response. In this case, we may perform variable selection by omitting the $j$-th covariate from the model. Unlike a column-sparse or element-sparse matrix, this is of much use when dealing with high-dimensional models.

In addition to being of much practical help, since it allows us to greatly simplify the model, a row-sparse matrix is likely to reflect the reality of the underlying data. When large amounts of predictor variables are collected, it is likely that many of these are unnecessary and have zero effect on the response. We may conclude that having $\boldsymbol{\beta}$ be row-sparse is both appropriate and desireable when fitting a high-dimensional model.

**Column-Sparse Matrix** Lastly, consider a column-sparse matrix where multiple columns are set to zero.

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_{1,1} & ... & 0 & ... & 0 & ... & \beta_{1,K-1} \\ \beta_{2,1} & ... & 0 & ... & 0 & ... & \beta_{2,K-1} \\ ... & ... & 0 & ... & 0 & ... & ... \\ \beta_{p,1} & ... & 0 & ... & 0 & ... & \beta_{p,K-1} \end{pmatrix}$$

Recall that $P(Y_i = k|\boldsymbol{X_i} = \boldsymbol{x_i}) = P(Y_i = K|\boldsymbol{X_i} = \boldsymbol{x_i}) \exp(\beta_{0,k} + \boldsymbol{\beta_{\cdot k}} \boldsymbol{x_i})$. Therefore $\boldsymbol{\beta_{\cdot k}} = \mathbf{0_p}$ implies that $P(Y_i = k) = P(Y_i = K) \exp(\beta_{0,k})$. In other words, the predictor matrix $X$ does not help distinguish between the $k$ and $K$-th classes. Therefore, it may be desirable to "merge" these classes (the ones that correspond to the zero columns) into the $K$-th class. This allows for a simpler model (when there are more classes, it is much harder to predict the most likely class - see the simulations) and a smaller (specifically, narrower) $\boldsymbol{\beta}$ matrix, so there are less coefficients to estimate. However, it still requires that we keep every variable $j \in \{1, ...p\}$ in the model. Unfortunately, in the common scenario where $p$ is very large but $K$ is of a reasonable size, a column sparse matrix does not help simplify the model.

However, we must not only think about simplifying the model. Preferably, we would like that our chosen model reflect reality as much as possible. In the aforementioned case of high-dimensional data, this scenario implies that each variable has a non-zero effect on the response - doubtful when $p$ is very high. On the other hand, with so much information thrown into the predictor matrix (large $p$) the model is still unable to make any distinction whatsoever between two different classes. These two features seem unlikely in the case of high-dimensional data, and we may conclude that a column-sparse $\boldsymbol{\beta}$ matrix is inappropriate

in that context.

## 2.3 Likelihood Function and Maximum Likelihood Estimator

A natural candidate for the loss function of the multinomial model is the log-likelihood. Specifically, the negative log-likelihood can be used as a loss function, as minimizing the negative log-likelihood corresponds to maximizing the likelihood. The derivatives (score function) and double-derivarives (Hessian matrix) of the likelihood also have to be calculated because they will be useful for optimization (of $\hat{\boldsymbol{\beta}}$) and Post-Selection Inference, which is described later in this thesis.

**Derivation of the Likelihood** Assume we are given data $(\boldsymbol{x_{i\cdot}}, \boldsymbol{y_i}), i = 1, ..., n$, where $\boldsymbol{y_i} = \{y_{i,1}, ..., y_{i,K-1}\}$ and $y_{ik} = 1_{(y_i=k)}$. The likelihood is

$$L_n(\boldsymbol{\beta}) = \prod_{i=1}^{n} P_{\boldsymbol{\beta}}(Y_i = y_i|\boldsymbol{x_i}) = \prod_{i=1}^{n} \left[ \frac{\prod_{k=1}^{K-1} \exp(\beta_{0,k} + \boldsymbol{x_{i\cdot}}\boldsymbol{\beta_{\cdot k}})^{y_{ik}}}{1 + \sum_{\ell=1}^{K-1} \exp(\beta_{0,\ell} + \boldsymbol{x_{i\cdot}}\boldsymbol{\beta_{\cdot \ell}})} \right] \tag{2.13}$$

The log-likelihood function is

$$\ell_n(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left[ \sum_{k=1}^{K-1} y_{ik}(\beta_{0,k} + \boldsymbol{x_{i\cdot}}\boldsymbol{\beta_{\cdot k}}) - \log(\sum_{\ell=1}^{K-1} \exp(\beta_{0,\ell} + \boldsymbol{x_{i\cdot}}\boldsymbol{\beta_{\cdot \ell}}) + 1) \right] \tag{2.14}$$

**Differentiating the Likelihood Equation** The derivative of the loss function with respect to one column $\boldsymbol{\beta_{\cdot \ell}}$ is

$$\dot{\ell}(\boldsymbol{\beta_{\cdot \ell}}) = \sum_{i=1}^{n} \left[ y_{i\ell}\boldsymbol{x_i} - \frac{\boldsymbol{x_i} \exp(\beta_{0,\ell} + \boldsymbol{x_{i\cdot}}\boldsymbol{\beta_{\cdot \ell}})}{\sum_{k=1}^{K-1} \exp(\beta_{0,k} + \boldsymbol{x_{i\cdot}}\boldsymbol{\beta_{\cdot k}}) + 1} \right] \tag{2.15}$$

Note that $\dot{\ell}(\boldsymbol{\beta_{\cdot \ell}})$ is a $p \times 1$ vector. The entire gradient vector is

$$\dot{\ell}(\boldsymbol{\beta}) = \{\dot{\ell}(\boldsymbol{\beta_{\cdot 1}}), ..., \dot{\ell}(\boldsymbol{\beta_{\cdot K-1}})\},\ a\ p \times (K-1)\ matrix \tag{2.16}$$

**"Vectorized" Notation** When dealing with score functions and Hessian matrices, it is often preferable to write $\boldsymbol{\beta}$ in a "vectorized" form $\boldsymbol{\beta^v}$, instead of as a matrix. Therefore $\beta_{jk}$, the $j$-th covariate's effect of the probability of being in the $k$-th class, will be the $((k-1)p+j)$-th entry in the vector $\boldsymbol{\beta^v}$, rather than the entry in the $j$-th row and $k$-th column of the matrix $\boldsymbol{\beta}$. This is so that the double derivative of the likelihood is a two-dimensional matrix instead of a three-dimensional matrix.

Furthermore, we will continue to write $\boldsymbol{\beta_{\cdot k}}$ to represent the coefficients corresponding to the $k$-th class. This allows us to rewrite the gradient vector of the log-likelihood as

$$\dot{\ell}(\boldsymbol{\beta^v}) = \{\dot{\ell}(\boldsymbol{\beta_{\cdot 1}}), ..., \dot{\ell}(\boldsymbol{\beta_{\cdot K-1}})\} \tag{2.17}$$

Now, we can write $\dot{\ell}(\boldsymbol{\beta^v})$ as a vector of length $p \cdot (K-1)$ instead of a matrix of dimension $p \times (K-1)$.

**Hessian Matrix** For the double derivatives, $\ddot{\ell}(\boldsymbol{\beta})$ is a $p*(K-1) \times p*(K-1)$ matrix, using the vectorized notation. On the diagonal, there are $K-1$ blocks

$$\ddot{\ell}(\boldsymbol{\beta})_{\ell,\ell} := \frac{\partial^2 \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta_\ell} \partial \boldsymbol{\beta_\ell}}, \ell = 1, ..., K-1 \tag{2.18}$$

, each of which is a $p \times p$ matrix. The off diagonal contains the blocks

$$\ddot{\ell}(\boldsymbol{\beta})_{\ell,m} := \frac{\partial^2 \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta_\ell} \partial \boldsymbol{\beta_m}},\ (\ell, m) \in \{1, ..., K-1\} \times \{1, ..., K-1\}, \ell \neq m \tag{2.19}$$

Specifically, the matrix $\ddot{\ell}(\boldsymbol{\beta})$ is given by

$$\ddot{\ell}(\boldsymbol{\beta}) = \begin{pmatrix} \ddot{\ell}(\boldsymbol{\beta})_{1,1} & \ddot{\ell}(\boldsymbol{\beta})_{1,2} & ... & \ddot{\ell}(\boldsymbol{\beta})_{1,K-1} \\ \ddot{\ell}(\boldsymbol{\beta})_{2,2} & \ddot{\ell}(\boldsymbol{\beta})_{2,2} & ... & \ddot{\ell}(\boldsymbol{\beta})_{2,K-1} \\ ... & ... & ... & ... \\ \ddot{\ell}(\boldsymbol{\beta})_{K-1,1} & \ddot{\ell}(\boldsymbol{\beta})_{K-1,2} & ... & \ddot{\ell}(\boldsymbol{\beta})_{K-1,K-1} \end{pmatrix} \tag{2.20}$$

These blocks on the diagonal are given by

$$\ddot{\ell}(\boldsymbol{\beta})_{\ell,\ell} = -\sum_{i=1}^{n} \left[ \boldsymbol{x_i^T x_i} \exp(\beta_{0,\ell} + \boldsymbol{x_i \beta_{\cdot \ell}}) \frac{\sum_{k \neq \ell} \exp(\beta_{0,k} + \boldsymbol{x_i \cdot \beta_{\cdot k}}) + 1}{(\sum_{k=1}^{K-1} \exp(\beta_{0,k} + \boldsymbol{x_i \beta_{\cdot k}}) + 1)^2} \right] \tag{2.21}$$

The off-diagonal blocks are

$$\ddot{\ell}(\boldsymbol{\beta})_{\ell,m} = \sum_{i=1}^{n} \left[ \frac{\boldsymbol{x_i^T x_i} \exp(\beta_{0,\ell} + \boldsymbol{x_i^T \beta_{\cdot \ell}}) \exp(\beta_{0,m} + \boldsymbol{x_i^T \beta_{\cdot m}})}{(\sum_{k=1}^{K-1} \exp(\beta_{0,k} + \boldsymbol{x_i^T \beta_{\cdot k}}) + 1)^2} \right] \tag{2.22}$$

### Maximum Likelihood Estimator

When the dimension is small and no variable selection needs to be performed, the matrix of coefficients $\boldsymbol{\beta}$ is estimated to maximize the log-likelihood $\ell_n(\hat{\boldsymbol{\beta}})$, which was given in equation (4.2).

This is a transcendal equation and therefore cannot be solved analytically. Instead it must be computed numerically, often by using *iteratively reweighted least squares* or *IRLS* or by *coordinate descent* [7] (which is also used to find penalized estimators).

Although the MLE is optimal from a likelihood standpoint, it is not practical when the model is of high dimension (large $p$). Since it performs no variable selection ($\beta_{jk} \neq 0 \; \forall j, k$), too many covariates will be in the final model, resulting in a lack of interpretability and excess noise [5].

## 2.4 Penalized Estimators

**Notation** As mentioned in the preliminaries section, when dealing with penalized estimators we will let $\boldsymbol{\beta}$ have dimension $p \times K$ instead of $p \times K - 1$.

Recall that for observations $\{\boldsymbol{(x_1, y_1), ..., (x_n, y_n)}\}$, where $\boldsymbol{x_i} = (x_{i,1}, ..., x_{i,p})$ and $\boldsymbol{y_i} =$

$(y_{i,1}, ..., y_{i,K})$, $y_{ik} = 1_{y_i=k}$, the log-likelihood is given by

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n}[\sum_{k=1}^{K} y_{ik}(\beta_{0,k} + \boldsymbol{x_i^T}\boldsymbol{\beta_{.k}}) - \log(\sum_{k=1}^{K}\exp(\beta_{0,k} + \boldsymbol{x_i^T}\boldsymbol{\beta_{.k}}))] \tag{2.23}$$

While in many cases the maximum likelihood estimator (which maximizes the above equation) is desirable, it is sometimes inadequate. If the goal is to perform variable selection, it is often necessary to add a *penalty term* to the likelihood equation. If the tuning parameter of the penalty function (usually given by $\lambda$) is large enough, many estimated coefficients will be set to zero..

In the case of multinomial regression, where each covariate corresponds to a row, our goal will be to get a row-sparse matrix. Therefore, we would like to perform variable selection on a row-by-row basis. To do so, each row must be penalized as a group, so the penalty term will take a row $\boldsymbol{\beta_{j.}}$ as an argument.

The penalty term will take the form

$$p(\boldsymbol{\beta_{j.}}, \lambda), \boldsymbol{\beta_{j.}} \in \mathbb{R}^K, \lambda \in \mathbb{R}^+ \tag{2.24}$$

By adding the penalty term to the negative log-likelihood equation, we get the *penalized negative log-likelihood equation*

$$\tilde{\ell}(\boldsymbol{\beta}, \lambda) = -\ell(\boldsymbol{\beta}) + \sum_{j=1}^{p} p(|\boldsymbol{\beta_{j.}}|, \lambda) \tag{2.25}$$

Note that we apply the penalty to the absolute value of each element of $\boldsymbol{\beta}$, since the goal is to shrink the coefficients towards zero.

**Maximum Likelihood Estimator** As previously described, the maximum likelihood estimator has no penalty term (or $p(\beta_{j.}) = 0$). Therefore

$$\tilde{\ell}(\boldsymbol{\beta}, \lambda) = -\ell(\boldsymbol{\beta}) = -\sum_{i=1}^{n}[\sum_{k=1}^{K} y_{ik}(\beta_{0,k} + \boldsymbol{x_i^T}\boldsymbol{\beta_{.k}}) - \log(\sum_{k=1}^{K}\exp(\beta_{0,k} + \boldsymbol{x_i^T}\boldsymbol{\beta_{.k}}))] \tag{2.26}$$

**Ungrouped Lasso Penalty** The lasso estimator, which puts an $L_1$ penalty on $\boldsymbol{\beta}$ and results in some elements of $\boldsymbol{\beta}$ being set to zero [22], can be generalized to the multinomial model.

In this situation, since $\boldsymbol{\beta}$ is a matrix, the lasso estimator will penalize the sum of all entries in the $\boldsymbol{\beta}$ matrix

$$\sum_{j=1}^{p}\sum_{k=1}^{K}|\beta_{j,k}| \tag{2.27}$$

Specifically, the penalty term is

$$p(\boldsymbol{\beta_j}.,\lambda) = \lambda\sum_{k=1}^{K}|\boldsymbol{\beta_{jk}}| \tag{2.28}$$

The penalized log-likelihood then becomes

$$\begin{aligned}\tilde{\ell}(\boldsymbol{\beta},\lambda) &= -\ell(\boldsymbol{\beta}) + \sum_{j=1}^{p} p(|\boldsymbol{\beta_j}.|,\lambda) \\ &= -\sum_{i=1}^{n}[\sum_{k=1}^{K} y_{ik}(\beta_{0,k} + \boldsymbol{x_i^T\beta_{\cdot k}}) - \log(\sum_{k=1}^{K}\exp(\beta_{0,k} + \boldsymbol{x_i^T\beta_{\cdot k}}))] \\ &+ \lambda\sum_{j=1}^{p}\sum_{k=1}^{K}|\beta_{j,k}|\end{aligned} \tag{2.29}$$

where the the lasso estimator is given by

$$\hat{\boldsymbol{\beta}}(\lambda) = \text{argmin}_{\boldsymbol{\beta}}\tilde{\ell}(\boldsymbol{\beta},\lambda) \tag{2.30}$$

Since the penalty term penalizes each entry in $\boldsymbol{\beta}$ independently, and because the $L_1$ penalty shrinks many entries to zero, $\hat{\boldsymbol{\beta}}(\lambda)$ will be an element-sparse matrix. As was previously mentioned, this matrix is both difficult to interpret and fails to perform variable selection.

**Grouped Lasso Penalty** Instead, we would like the penalty term to perform true variable selection, which can be achived by setting entire rows of $\boldsymbol{\beta}$ to zero. To achieve this, the $L_1$ penalty must be applied on a row-by-row basis; specifically, there should be an $L_1$ penalty on the magnitude of each of the $p$ rows [17].

In this case, we should penalize

$$\sum_{j=1}^{p} ||\boldsymbol{\beta_j},||_2 = \sum_{j=1}^{p} \sqrt{\sum_{k=1}^{K} \beta_{j,k}^2} \tag{2.31}$$

Specifically, the penalty term is

$$p(\boldsymbol{\beta_j}\cdot, \lambda) = \lambda \sqrt{\sum_{k=1}^{K} \boldsymbol{\beta_{jk}}^2} \tag{2.32}$$

And the penalized log-likelihood is

$$\begin{aligned} \tilde{\ell}(\boldsymbol{\beta}, \lambda) &= -\ell(\boldsymbol{\beta}) + \sum_{j=1}^{p} p(|\boldsymbol{\beta_j}.|, \lambda) \\ &= -\sum_{i=1}^{n} [\sum_{k=1}^{K} y_{ik}(\beta_{0,k} + \boldsymbol{x_i^T}\boldsymbol{\beta_{\cdot k}}) - \log(\sum_{k=1}^{K} \exp(\beta_{0,k} + \boldsymbol{x_i^T}\boldsymbol{\beta_{\cdot k}}))] \\ &+ \lambda \sum_{j=1}^{p} \sqrt{\sum_{k=1}^{K} \beta_{j,k}^2} \end{aligned} \tag{2.33}$$

Note that this is a special form of the *Group-Lasso* penalty, where each row of $\boldsymbol{\beta}$ forms a group of coefficients (so either all or none of the coefficients in each group are selected). With this penalty and a suitable choice of $\lambda$, $\hat{\boldsymbol{\beta}}(\lambda) = \text{argmin}_{\boldsymbol{\beta}} \tilde{\ell}(\boldsymbol{\beta}, \lambda)$ will be a row-sparse matrix.

**Elastic Net** This can also be extended to an *elastic-net* penalty [10] by adding an $L_2$ penalty term.

$$\sum_{j=1}^{p} \sum_{k=1}^{K} \beta_{j,k}^2 \tag{2.34}$$

The elastic net penalty has many desireable properties. Importantly, unlike the lasso it is effective when variables are highly correlated. Furthermore, the $L_2$ penalty is much more "smooth" than the $L_1$ penalty, so it allows for easier computation [10].

The penalized log-likelihood equation for the ungrouped elastic-net penalty is given by

$$\begin{aligned}\tilde{\ell}(\boldsymbol{\beta}, \lambda, \alpha) = = &- \sum_{i=1}^{n}[\sum_{k=1}^{K} y_{ik}(\beta_{0,k} + \boldsymbol{x_i^T}\boldsymbol{\beta_{\cdot k}}) - \log(\sum_{k=1}^{K}\exp(\beta_{0,k} + \boldsymbol{x_i^T}\boldsymbol{\beta_{\cdot k}}))] \\ &+ \lambda[(1-\alpha)\sum_{j=1}^{p}\sum_{k=1}^{K}\beta_{j,k}^2 + \alpha\sum_{j=1}^{p}\sum_{k=1}^{K}|\beta_{j,k}|]\end{aligned} \tag{2.35}$$

The penalized log-likelihood equation for the grouped elastic-net penalty is given by

$$\begin{aligned}\tilde{\ell}(\boldsymbol{\beta}, \lambda, \alpha) = = &- \sum_{i=1}^{n}[\sum_{k=1}^{K} y_{ik}(\beta_{0,k} + \boldsymbol{x_i^T}\boldsymbol{\beta_{\cdot k}}) - \log(\sum_{k=1}^{K}\exp(\beta_{0,k} + \boldsymbol{x_i^T}\boldsymbol{\beta_{\cdot k}}))] \\ &+ \lambda[(1-\alpha)\sum_{j=1}^{p}\sum_{k=1}^{K}\beta_{j,k}^2 + \alpha\sum_{j=1}^{p}\sqrt{\sum_{k=1}^{K}\beta_{j,k}^2}]\end{aligned} \tag{2.36}$$

The lasso penalty is equivalent to the elastic-net penalty with $\alpha = 1$. Setting $\alpha = 0$ results in ridge regression.

The elastic net or lasso estimate $\hat{\boldsymbol{\beta}}(\lambda, \alpha)$ can be computed in `R` using the `glmnet` package [7], which uses coordinate descent to minimize the penalized negative log-likelihood. This package outputs a $p \times K$ matrix $\beta'$ instead of a full-rank $p \times K - 1$ matrix, as described in section **2**. We can recover the full rank matrix $\boldsymbol{\beta}$ by setting $\boldsymbol{\beta_{\cdot k}} = \boldsymbol{\beta'_{\cdot k}} - \boldsymbol{\beta'_{\cdot K}}$ for $k = 1, ..., K-1$.

**Adaptive Lasso** Zou [26] introduced the adaptive lasso, which is defined as follows. Let $\hat{\boldsymbol{\beta}}_{ML}$ be the maximum likelihood estimator of $\boldsymbol{\beta}$.

Next, let the *weights* $\boldsymbol{w}$ be defined by

$$w_{jk} = \frac{1}{|\hat{\beta}_{ML_{jk}}|^{\nu}} \tag{2.37}$$

where $\nu > 0$ (sometimes chosen by cross validation).

The penalty term is given by

$$p(\lambda, \nu, |\beta_{jk}|) = \lambda w_{jk} |\beta_{jk}| \tag{2.38}$$

Therefore, the penalized negative log-likelihood for the adaptive lasso is given by

$$\tilde{\ell}(\boldsymbol{\beta}, \lambda, \nu) = -\ell(\boldsymbol{\beta}) + \sum_{j=1}^{p} \sum_{k=1}^{K} p(\lambda, \nu, |\beta_{j,k}|) \tag{2.39}$$

It is important to understand the intuition behind the adaptive lasso. A major downside of the lasso is that it is biased; specifically, as $\lambda$ grows so does the bias. This is somewhat desireable since it sets (or attempts to set) the coefficients corresponding to inactive variables to zero. However, for active variables, the corresponding non-zero coefficients will be biased downwards. Adaptive Lasso attempts to mitigate this effect through the use of weights. For $j, k$ where $|\hat{\beta}_{ML_{j,k}}|$ is large, $w_{jk}$ will be small. This is desireable because having $|\hat{\beta}_{ML_{j,k}}|$ large implies that the true value coefficient is probably nonzero. Therefore the "effective penalty" applied to $\hat{\beta}_{jk}$, $w_{jk}\lambda$, will be small, leading to a small bias for $\beta_{jk}$.

Conversely, for $j, k$ where $|\hat{\beta}_{ML_{j,k}}|$ is small, $w_{jk}$ is large. This leads to a large downwards bias being applied, increasing the likelihood of $\hat{\beta}_{jk}$ being set to zero. Since it is likely that $\beta_{jk} = 0$ (ie, the corresponding variable is truly inactive), this effect is desireable.

Note that the size of the tuning paramater $\nu$ determines the magnitude of this affect; as $\nu$ increases, more importance is places on coefficients with small $|\hat{\beta}_{ML_{j,k}}|$.

As with all the previous "ungrouped" penalties, applying this "ungrouped" version of the adaptive lasso will output an element-sparse matrix $\hat{\boldsymbol{\beta}}$.

**Grouped Adaptive Lasso** As was previously mentioned, we would prefer that the matrix of coefficients $\hat{\boldsymbol{\beta}}$ be row-sparse. We can extend the adaptive lasso penalty by penalizing the coefficients on a group by group basis.

Let the *weights* $\boldsymbol{w}$ be defined by

$$w_j = \frac{1}{||\hat{\boldsymbol{\beta}}_{MLj.}||_2^{\nu}} \tag{2.40}$$

where $\nu > 0$.

The penalty term is given by

$$p(\lambda, \nu, |\boldsymbol{\beta_j}.|) = \lambda \boldsymbol{w_j} ||\boldsymbol{\beta_j}.||_2 \tag{2.41}$$

The penalized negative log-likelihood for the adaptive lasso is given by

$$\tilde{\ell}(\boldsymbol{\beta}, \lambda, \nu) = -\ell(\boldsymbol{\beta}) + \sum_{j=1}^{p} p(\lambda, \nu, |\boldsymbol{\beta}_j.|) = -\ell(\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} \boldsymbol{w_j} ||\boldsymbol{\beta_j}.||_2 \tag{2.42}$$

This can be implemented in `R` through the `glmnet` package, using the option `penalty.factor` to define the weights.

# 3 Comparison of Grouped and Ungrouped Penalization Methods

To assess whether or not a grouped estimator actually performs better than the ungrouped version, and to compare the performance of various estimators listed above, we use simulations to estimate the misclassification rate by appliyng the estimators to artificially-created datasets. This is followed by a comparison of the grouped and ungrouped penalties' respective performances on a real dataset.

## 3.1 Simulations - Artificial Dataset

### 3.1.1 Methodology

**Methods Used** The matrix of coefficients, $\boldsymbol{\beta}$, was estimated by minimizing the penalized negative log-likelihood function $\tilde{\ell}(\boldsymbol{\beta}, \lambda)$, so that $\hat{\boldsymbol{\beta}}(\lambda) = \text{argmin}_{\boldsymbol{\beta}} \tilde{\ell}(\boldsymbol{\beta}, \lambda)$. $\lambda$ was chosen by cross-validation, and $\alpha$ (the elastic net tuning parameter) is set to 0.9 (to help counter the effects of covariance between columns of the predictor matrix and smooth out the computation). When adaptive lasso is used, $\nu$ is set to 1 and an $L_2$ (elastic net) penalty term is added, using $\alpha = 0.9$.

The following penalized likelihood functions were used:

1. Maximum-Likelihood Estimator (Equation 3.15)
2. Ungrouped Elastic Net (3.24)
3. Grouped Elastic Net (3.25)
4. Ungrouped Adaptive Lasso (3.28) *(with an $L_2$ penalty term added)*
5. Grouped Adaptive Lasso (3.31) *(with an $L_2$ penalty term added)*

**Parameters Used** The following parameters were used to construct the predictor and response matrices.

1. $n$, the number of observations
2. $\gamma$, the rate of divergence of the number of covariates (in the `R` code, this parameter is called `alpha`. In this section, we will use $\gamma$ to avoid confusion with the elastic-net parameter)
3. $p$, the number of covariates, given by $\lfloor n^{\gamma} \rfloor$
4. $s_0$, the number of active (nonzero) covariates, set to 5
5. $K$, the number of classes of the categorical response

6. $\rho$, which determines the correlation between the covariates (columns of the predictor matrix), set to 0.5

The following range of parameters are used:

$n \in \{50, 100, 200\}$

*(The simulations run very slowly for large n; n=500 will be added in the future)*

$\gamma \in \{0.5, 0.7, 1.0, 1.5\}$

$K \in \{3, 6\}$ for $n < 200$

$K \in \{3, 6, 10, 20\}$ for $n \geq 200$

**Details of the Experiment** `R` is used to generate artificial datasets and compute the estimated matrix of coefficients. For each set of parameters 200 pairs of predictor matrices and response vectors were created, given by

$$\begin{aligned} &\{(\boldsymbol{X_1}, \boldsymbol{Y_{1,1}}), (\boldsymbol{X_1}, \boldsymbol{Y_{1,2}}), ..., (\boldsymbol{X_1}, \boldsymbol{Y_{1,20}}), \\ &(\boldsymbol{X_2}, \boldsymbol{Y_{2,1}}), (\boldsymbol{X_2}, \boldsymbol{Y_{2,2}}), ..., (\boldsymbol{X_2}, \boldsymbol{Y_{2,20}}), \\ &... \\ &(\boldsymbol{X_{10}}, \boldsymbol{Y_{2,1}}), (\boldsymbol{X_{10}}, \boldsymbol{Y_{2,2}}), ..., (\boldsymbol{X_{10}}, \boldsymbol{Y_{2,20}})\} \end{aligned} \tag{3.1}$$

Ten instances of $\boldsymbol{\beta}$ are also randomly generated, given by

$$\{\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, ..., \boldsymbol{\beta}_{10}\} \tag{3.2}$$

The predictor matrices $\boldsymbol{X_1}, \boldsymbol{X_2}, ..., \boldsymbol{X_{10}}$ were all generated independently. For each $i = 1, 2, ..., 10$, $\boldsymbol{Y_{i,1}}, \boldsymbol{Y_{i,2}}, ..., \boldsymbol{Y_{i,20}}$ were generated using $\boldsymbol{X_i}$ and $\boldsymbol{\beta_i}$ and independently of each other.

Furthermore, each pair $(\boldsymbol{X_i}, \boldsymbol{Y_{i,j}})$ has a corresponding "testing" pair $(\boldsymbol{X_i'}, \boldsymbol{Y_{i,j}}')$, which is independent of and identically distributed to $(\boldsymbol{X_i}, \boldsymbol{Y_{i,j}})$.

**Construction of the Predictor Matrices** Given the parameters $n, p, \rho$, the predictor matrix $\boldsymbol{X_i}$ follows a multivariate normal distribution given by

$$\boldsymbol{X_i} \sim \mathcal{N}_p(\mathbf{0_p}, \boldsymbol{\Sigma}) \tag{3.3}$$

for $i = 1, 2, ..., 10$.

The variance-covariance matrix $\boldsymbol{\Sigma}$ is a *Toeplitz* matrix, given by

$$\boldsymbol{\Sigma}_{\ell,m} = \rho^{|\ell - m|} \tag{3.4}$$

for $\ell = 1, ..., n$ and $m = 1, ..., p$.

$\boldsymbol{X_i}'$ is independently generated from the same distribution $\mathcal{N}_p(\mathbf{0_p}, \boldsymbol{\Sigma})$.

**Construction of $\boldsymbol{\beta}$** Given $K$ and $s_0$, $\boldsymbol{\beta_i}$ (a $p \times K$ matrix) is given as follows (for $i = 1, 2, ..., 10$):

$$\boldsymbol{\beta} = \begin{pmatrix} U_{1,1} & ... & U_{1,K} \\ U_{2,1} & ... & U_{2,K} \\ ... & ... & ... \\ U_{s_0,1} & ... & U_{s_0,K} \\ 0 & ... & 0 \\ ... & ... & ... \\ 0 & ... & 0 \end{pmatrix} \tag{3.5}$$

where the $U_{\ell,m}$'s are independent and identically distributed

$$U_{\ell,m} \sim \mathcal{U}(-2, 2) \tag{3.6}$$

for $\ell = 1, ..., s_0$, $m = 1, ..., K$, and where $\mathcal{U}$ is the uniform distribution.

The vector of intercepts $\boldsymbol{\beta}_{0\cdot}$ is set to $\mathbf{0}_K$.

**Construction of the Response Vectors** Given $X$ and $\boldsymbol{\beta}$, the response vector $\boldsymbol{Y}$ must be generated.

The first step is to (deterministically) determine the probabilities of each observation falling in a given class, which can be calculated using equation (2.6).

Finally, each element $Y_i$ of the response vector $\boldsymbol{Y}$ can be randomly generated from a multinomial distribution.

$$Y_i \sim \text{Multinomial}(P(Y_i = 1|\boldsymbol{x_i}, \boldsymbol{\beta}), P(Y_i = 2|\boldsymbol{x_i}, \boldsymbol{\beta}), ..., P(Y_i = K|\boldsymbol{x_i}, \boldsymbol{\beta})) \tag{3.7}$$

Using these equations, the response vector can be simulated using the `rmultinom` function in `R`.

The "test" response vectors $\boldsymbol{Y}'$ are calculated in the same manner, using $\boldsymbol{X}'$ and $\boldsymbol{\beta}$

### 3.1.2 Prediction

For each pairing $(\boldsymbol{X_i}, \boldsymbol{Y_{i,j}})$, a model is fit using each of the 5 estimators.

The following `R` functions were used to estimate the coefficients:

1. For Maximum-Likelihood Estimator, the `multinom` function from the `nnet` package [21] is used.

2. For the Ungrouped Elastic Net Penalty, the `glmnet` package was used.
   Specifically, the function `cv.glmnet` was used to find an optimal choice of the tuning parameter $\lambda$. The default setting of 10-fold cross validation was used.
   The argument `type.multinomial` was set to `"ungrouped"`
   $\alpha$ was set to 0.9 (using the `glmnet` option `alpha=0.9`), adding a small $L_2$ penalty.

3. For the Grouped Elastic Net, `cv.glmnet` was also used.
   In this case, `type.multinomial` was set to `"grouped"`

4. For the Ungrouped Adaptive Lasso, the `glmnet` package was used. The argument `type.multinomial` is set to `ungrouped`, and the weights $w$ are set using the `penalty.factor` option. The weights are set using the maximum likelihood estimator coefficients (calculated using the `nnet` package), with $\nu$ set to 1.

   For computational simplicity, each "row" $\boldsymbol{w_{j\cdot}}$ of weights is uniform, with $w_{jk} = ||\boldsymbol{w_{j\cdot}}||_2 \ \forall k$. This still results in each coefficient being penalized seperately, since the ungrouped elastic net penalty is used. The magnitude of the penalty being applied to $\beta_{jk}$ depends only on the relationship between the $j$-th covariate and the response, and is independent of the class $k$ that corresponds to $\beta_{jk}$.

   As with the elastic net penalty, an $L_2$ penalty term is added by setting `alpha=0.9`.

5. For the Grouped Adaptive Lasso, the `glmnet` package was used. The argument `type.multinomial` is set to `grouped`, and the weights $\boldsymbol{w}$ are set using the `penalty.factor` option. $\alpha$ is again set to 0.9. The weights are set using the maximum likelihood estimator coefficients (calculated using the `nnet` package, , with $\nu$ set to 1.

After the coefficients $\hat{\boldsymbol{\beta}}$ are estimated using one of the methods listed above, the probabilities for the test set are calculated using the following formulas (for all $i = 1, ..., n, k = 1, ..., K$):

$$P(Y_i' = k | \boldsymbol{X_i} = \boldsymbol{x_i}, \boldsymbol{\beta} = \hat{\boldsymbol{\beta}}) = \frac{e^{\hat{\beta}_{0,k} + \hat{\boldsymbol{\beta}}_{\boldsymbol{\cdot k}} \boldsymbol{x_i}}}{\sum_{\ell=1}^{K} e^{\hat{\beta}_{0,\ell} + \hat{\boldsymbol{\beta}}_{\boldsymbol{\cdot \ell}} \boldsymbol{x_i}}}, \quad k = 1, ..., K \tag{3.8}$$

**Calculation of Misclassification Rate** Using the estimated probabilities $P(Y'_{i,j,a} = k | \boldsymbol{X}_{\boldsymbol{i_a\cdot}} = \boldsymbol{x_a}, \boldsymbol{\beta} = \hat{\boldsymbol{\beta}})$ for $i = 1, ..., 10, j = 1, ..., 20, a = 1, ..., n, k = 1, ..., K$, we can calculate the misclassification rates of the different methods.

First, $Y'_{i,j,a}$ is "assigned to" (or predicted to be in) the class $m_{i,j,a}$ which maximises the

aformentioned probability,

$$m_{i,j,a} = \text{argmax}_k P(Y'_{i,j,a} = k | \boldsymbol{X_{i_a\cdot}} = \boldsymbol{x_a}, \boldsymbol{\beta} = \hat{\boldsymbol{\beta}}) \tag{3.9}$$

The misclassification rate $e_{i,j}$ can be calculated using the test response $\boldsymbol{Y_{i,j}}'$, by seeing how often the above prediction was correct.

$$e_{i,j} = \frac{1}{n} \sum_{a=1}^{n} 1(m_{i,j,a} \neq Y'_{i,j,a}) \tag{3.10}$$

The average misclassification rate for a given set of parameters $n, p, s_0, \rho, K$ can be determined by averaging the misclassification rate $e_{i,j}$ over all $i = 1, ..., 10, j = 1, ..., 20$.

$$\bar{e} = \frac{1}{200} \sum_{i=1}^{10} \sum_{j=1}^{20} e_{i,j} \tag{3.11}$$

This is the misclassification rate that is reported in the results.

### 3.1.3 Results

In preliminary tests, the MLE and the Adaptive Lasso (both the grouped and ungrouped versions) performed very poorly. Therefore, the detailed simulations will focus on comparing the respective performances of the grouped and ungrouped elastic net.

Although both the grouped and ungrouped lasso perform somewhat poorly (with misclassification rates at around 40%) in these simulations, they perform much better when applied to a real dataset, as shown in the next section. The purpose of this section is to compare their relative performances, and show how the advantage of the group penalty varies with the parameters of the model.

**Effect of $n$, $p$, and $K$ on Misclassification Rates** We can study how the difference in performance between the grouped and ungrouped penalties varies with $n, p$, and $K$. If the grouped penalty was no better than the ungrouped penalty, we would expect that the difference in misclassification rates between the two penalties, all else being equal, would be

zero (depicted by the horizontal red line in the boxplots). If the grouped penalty was better (has a smaller misclassification rate), we would expect that the difference would be positive. While it is difficult to formally test the statistical significance of these results, the following data and boxplots will provide strong visual evidence that the grouped penalty is superior, especially when $n$, $p$, and $K$ are large.

For $\boldsymbol{n = 50}$, the grouped elastic net had an average (across all parameters other than $n$) misclassification rate of 45.0% for the grouped elastic net and 46.0% for the ungrouped elastic net. On average, the misclassification rate for the grouped elastic net was **0.95**% **lower** than the ungrouped elastic net.

For $\boldsymbol{n = 100}$, the grouped elastic net had an average (across all parameters other than $n$) misclassification rate of 40.1% for the grouped elastic net and 42.1% for the ungrouped elastic net. On average, the misclassification rate for the grouped elastic net was **1.98**% **lower** than the ungrouped elastic net.

For $\boldsymbol{n = 200}$, the grouped elastic net had an average (across all parameters other than $n$) misclassification rate of 48.3% for the grouped elastic net and 50.3% for the ungrouped elastic net. On average, the misclassification rate for the grouped elastic net was **2.00**% **lower** than the ungrouped elastic net.

We can create graphs to visualize how the difference in misclassification rates between the grouped and ungrouped estimators depends on $p, K$, and $s_0$ (a positive difference means that the grouped elastic net performed better than the ungrouped elasic net). Note that in this case, `alpha` refers to $\gamma$, where $p = n^{\gamma}$.

The graphs for $n = 50, 100, 200$ are below.

Figure 1: Effect of $p$, $K$ on misclassification rate when $n = 50$

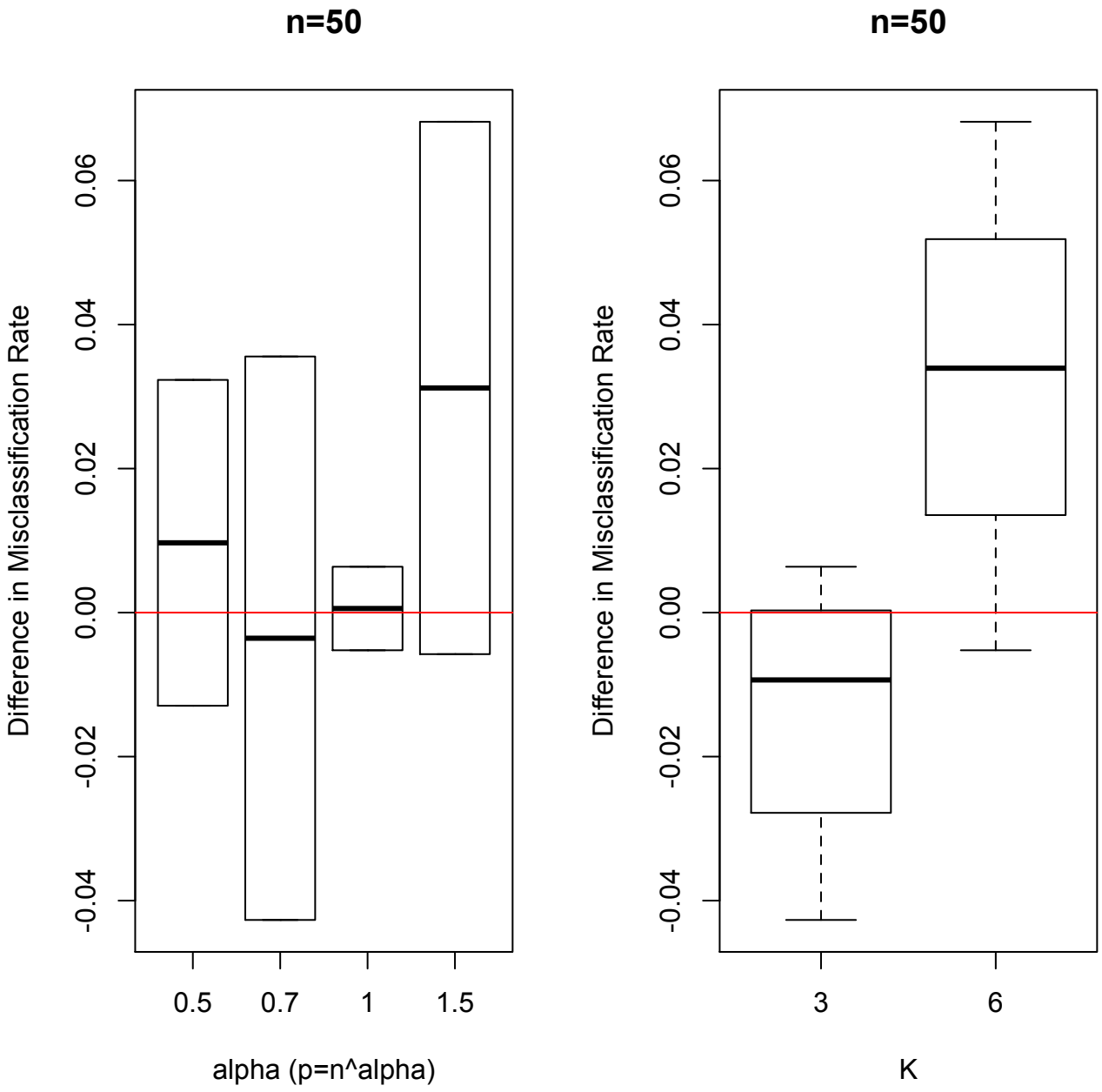
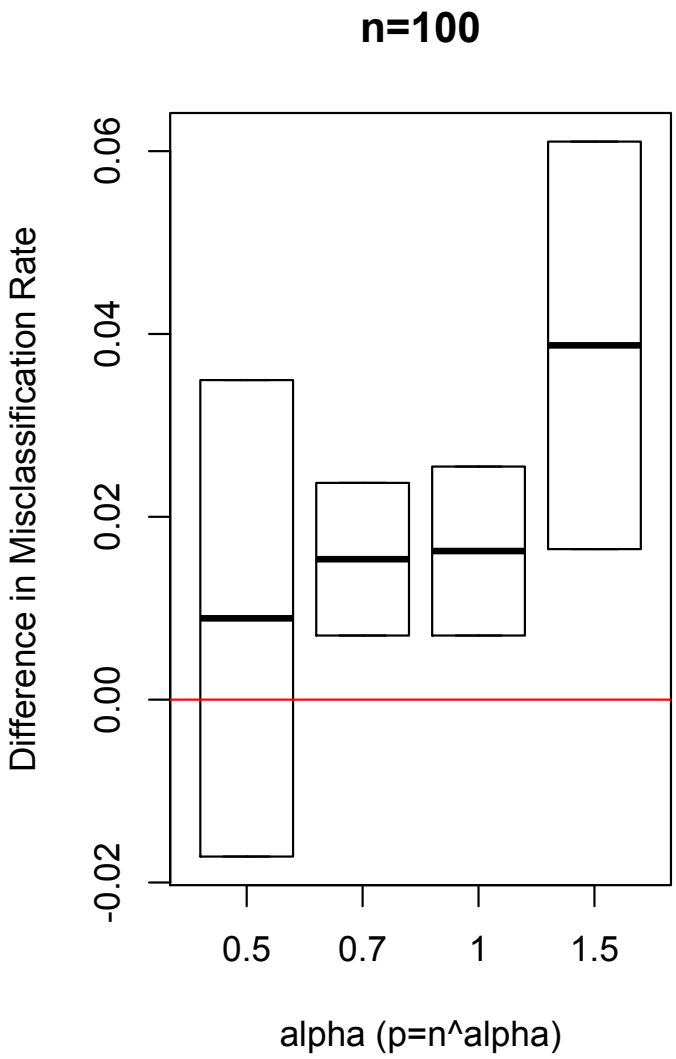


Figure 2: Effect of $p$, $K$ on misclassification rate when $n = 100$



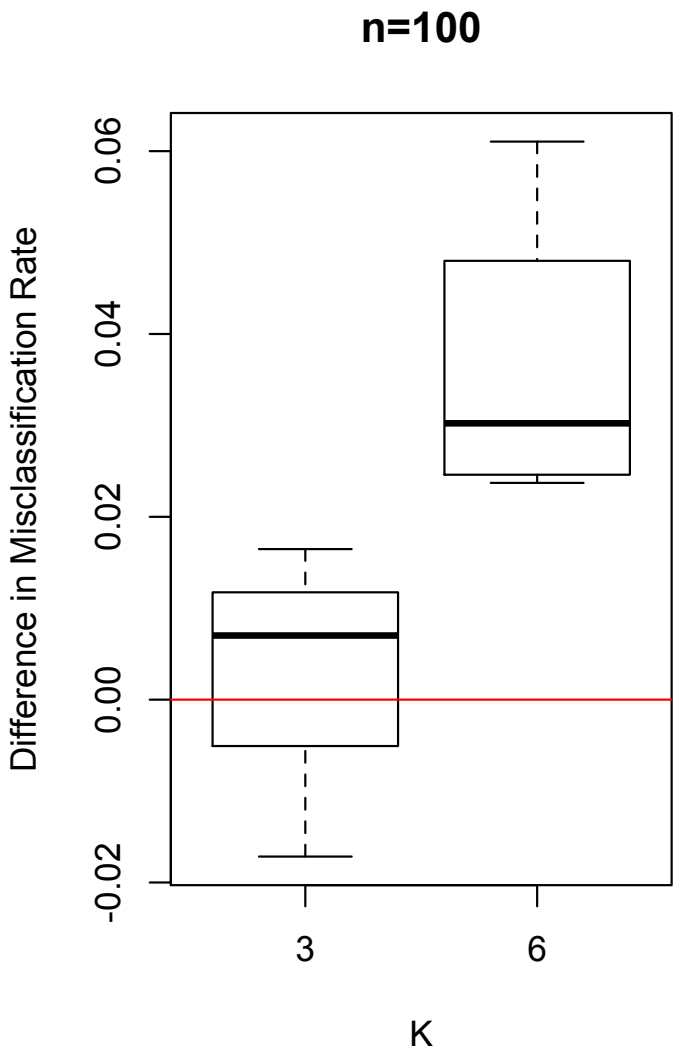
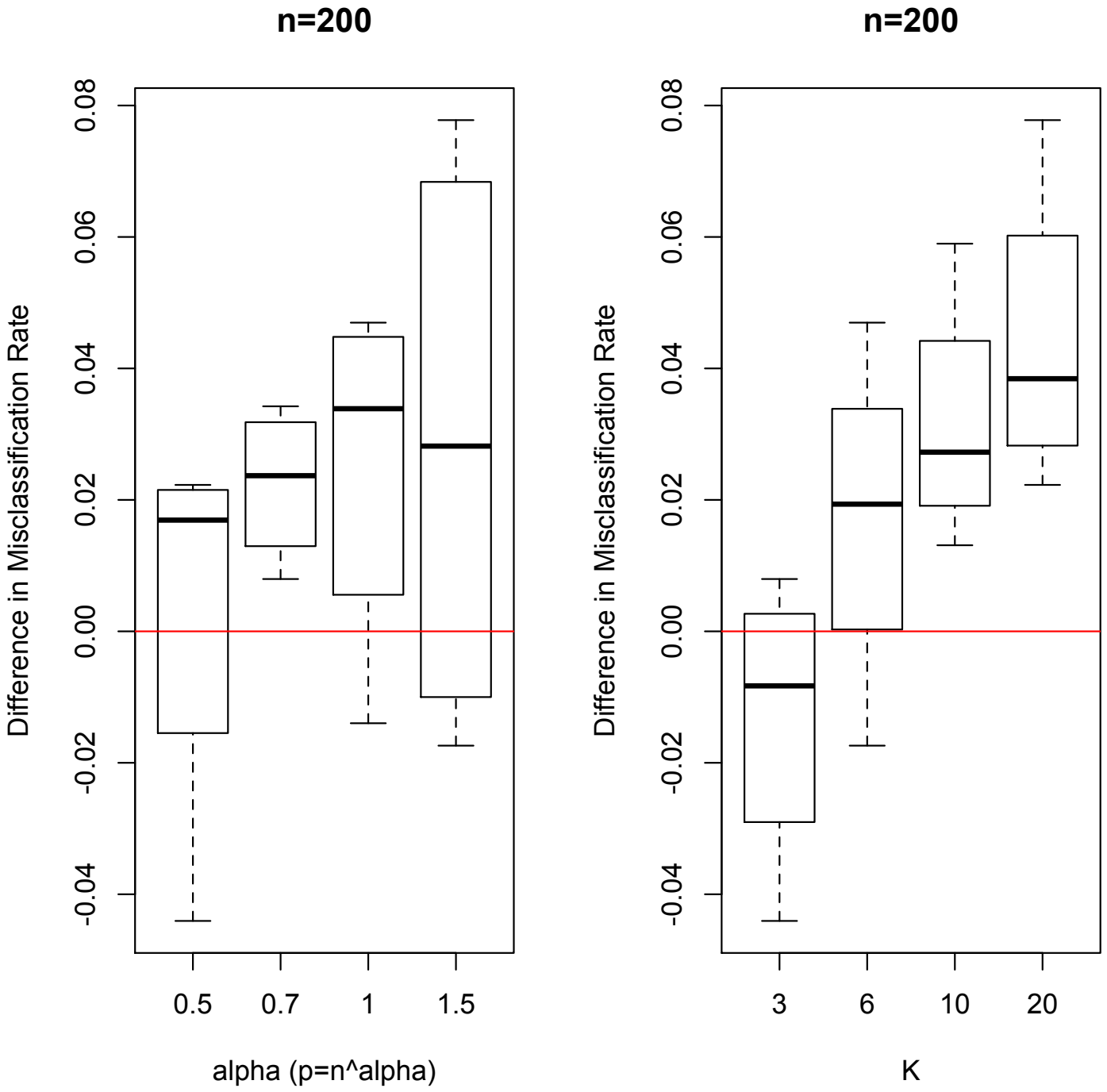



Figure 3: Effect of $p$, $K$ on misclassification rate when $n = 200$



For $n = 50$, the grouped and ungrouped penalized estimators performed similarly for small $p$. This is to be expected, since when $n = 50$ and $p < 50$, variable selection is not very necessary. However, when $p = 50^{1.5} = 353$, the need for variable selection increases. This is reflected in the above graph - for $\gamma = 1.5/p = 353$, the grouped estimator has a misclassification rate that is almost 4 percentage points lower than that of the ungrouped estimator.

When $K = 3$, the grouped and ungrouped estimators again performed similarly. This can also be expected, since $\boldsymbol{\beta}$ is relatively small (only 3 columns), so setting an entire row to zero is not very important. However, when $K = 6$, $\boldsymbol{\beta}$ is larger an more complex, and it is undesireable to have a row containing both zeros and nonzeros. This is also reflected in the above graph, as the grouped estimator outperforms in this case.

For $n = 100$, we see similar results, for the same reasons. he grouped estimator performed slightly better, except in the case of $\gamma = 1.5/p = 1000$ and $K = 6$, where the grouped estimator performed much better.

When we have a high-dimension model with $n$ and $p$ large, feature selection becomes ever more important. For $n = 200$, the grouped estimator is consistently better. As $p$ and $K$ increase, so does the difference in misclassification rates, confirming that the need for grouped variable selection increases as $\boldsymbol{\beta}$ increases in size.

## 3.2 Experiment - Real Dataset

To test the performance of grouped and ungrouped penalized estimators on a real-life dataset, we use the "ISOLET" dataset from the University of California, Irvine's Machine Learning Repository [16], created by Ron Cole and Mark Fanty [3].

### 3.2.1 Description of the Dataset

The ISOLET dataset was generated by 150 English-speaking subjects (with different genders, accents, and dialects), with each pronouncing the name of each letter of the alphabet

twice, resulting in $2 \times 150 \times 26 = 7800$ total instances. Three instances are missing, so there is a total of $n = 7797$. The response is categorical, with its domain in the integers between 1 and 26, each representing a letter of the alphabet (so in the context of multinomial regression, $K = 26$).

There are 617 features, which include "spectral coefficients; contour features; sonorant features; pre-sonorant features; and post-sonorant features" [16]. All the features are continuous and scaled to be between $-1$ and 1. The features are described in more detail in [3].

### 3.2.2 Prediction

The authors of the dataset split the data into two sets: a testing set with 6238 observations, and a training set with 1559 observations. We performed two experiments. In the first, we fit ungrouped and grouped elastic-net models on the training set (the maximum likelihood estimator and the grouped and ungrouped adaptive lasso estimators performed poorly) and used these models to predict the responses of the test set to obtain misclassification rates.

Although this is a high dimensional dataset with $p = 617$ and $K = 26$, when we have $n = 6238$ prediction becomes much easier. Since $p << n$, variable selection is less important and the noise from 617 features is less of a concern than in cases where $p \approx n$ or $p > n$. To try to emulate a common real-world situation where the number of observations is small relative to the number of features, we refit grouped and ungrouped elastic-net models using a training set of only 500 randomly-chosen observations and computed a misclassification rate by testing on a testing set of another 500 randomly-chosen observations. This was repeated 100 times (using different training and testing sets).

### 3.2.3 Results

**Whole Dataset** When using the training set of 6238 observations, the ungrouped elastic net model had a misclassification rate (when predicting the responses of the test set) of 4.426%. The grouped elastic net actually performed worse, with a misclassification rate of

5.067%. We would hypothesize that because $p$ is small relative to $n$, the benefits of variable selection are outweighed by the fact that the model chosen using the ungrouped penalized estimator contains more information.

**Smaller Training Sets** The results were very different when a smaller training set with $n = 500$ is used. With $p > n$, variable selection becomes much more important, both for the simplicity of the model and for reducing noise from 617 features. The average misclassification rate was 12.054% when using the ungrouped elastic net, and 9.516% when using the grouped elastic net. This means that for the smaller training sets of $n = 500$, grouping reduced the msiclassification rate by 21%.

# 4 Generalization to Other Penalties

**SCAD Penalty** Fan and Li (2001) proposed the *SCAD* penalty [4], which is defined on $[0, \infty)$. In the usual case where the penalty is applied individually to each coefficient, the penalty is given by

$$p_a(\beta_{j\cdot}, \lambda) = \begin{cases} \lambda|\beta_{j\cdot}| & |\beta_{j\cdot}| \leq \lambda \\ \dfrac{a\lambda|\beta_{j\cdot}| - 0.5(\beta_{j\cdot}^2 + \lambda^2)}{a-1} & \lambda < |\beta_{j\cdot}| \leq a\lambda \\ \dfrac{\lambda^2(a^2-1)}{2(a-1)} & |\beta_{j\cdot}| > a\lambda \end{cases} \tag{4.1}$$

For $a > 1$. Fan and Li suggested using $a = 3.7$ in most situations [4].
The reasoning behind this penalty can be better understood when looking at its derivative, which is given by

$$p'_a(\beta_{j\cdot}, \lambda) = \begin{cases} \lambda & |\beta_{j\cdot}| \leq \lambda \\ \dfrac{a\lambda - |\beta_{j\cdot}|}{a-1} & \lambda < |\beta_{j\cdot}| \leq a\lambda \\ 0 & |\beta_{j\cdot}| > a\lambda \end{cases} \tag{4.2}$$

For $\beta_{j\cdot}$ relatively small ($\beta_{j\cdot} < \lambda$), the rate of penalization is given by $\lambda$ - the same penalty as the lasso. This relatively high rate of penalization strongly "pushes" the coefficient towards zero. This is desirable so that the SCAD estimator maintains the selection property of the lasso, since relatively small instances of $\hat{\beta}$ will instead be set to zero.

Unlike the lasso, the penalty tapers off as $\beta_{j\cdot}$ increases. The rate of penalization decreases until $\beta_{j\cdot} > a\lambda$, in which case the penalty term plateaus with a value of $\dfrac{\lambda^2(a^2-1)}{2(a-1)}$. As a result, SCAD does not perform shrinkage as strongly as the lasso. Therefore, we can view SCAD as a penalty which fully maintains the selection "power" of the lasso, while putting less emphasis on shrinkage. This may be useful in cases where we would like as little bias as possible on coefficients which are deemed to be nonzero.

When $\boldsymbol{\beta}$ is a matrix, the penalized log-likelihood is then given by

$$\tilde{\ell}(\boldsymbol{\beta}, \lambda) = -\ell(\boldsymbol{\beta}) + \sum_{j=1}^{p} \sum_{k=1}^{K} p_a(|\beta_{j,k}|, \lambda) \tag{4.3}$$

where $p_a$ is given by equation (6.1).

As with the ungrouped Lasso, this penalty will result in an element-sparse matrix of coefficients.

**Grouped SCAD Penalty** As with the Lasso, we would like to apply a grouped version of the SCAD penalty so as to get a row-sparse matrix of coefficients. Therefore we should apply the SCAD penalty (which performs shrinkage and selection) on a row-by-row basis, and use an $\mathcal{L}_2$ penalty (which performs shrinkage but not selection) within each row.

Therefore we need to introduce the *Grouped SCAD* penalty, given by

$$p_a(\boldsymbol{\beta_j}., \lambda) = \begin{cases} \lambda||\boldsymbol{\beta_j}.||_2 & ||\boldsymbol{\beta_j}.||_2 \leq \lambda \\ \dfrac{a\lambda||\boldsymbol{\beta_j}.||_2 - 0.5(||\boldsymbol{\beta_j}.||_2^2 + \lambda^2)}{a-1} & \lambda < ||\boldsymbol{\beta_j}.||_2 \leq a\lambda \\ \dfrac{\lambda^2(a^2-1)}{2(a-1)} & ||\boldsymbol{\beta_j}.||_2 > a\lambda \end{cases} \tag{4.4}$$

where $||\boldsymbol{\beta_j}.||_2 = \sqrt{\sum \beta_{jk}^2}$.
The penalized log-likelihood is then given by

$$\tilde{\ell}(\boldsymbol{\beta}, \lambda) = -\ell(\boldsymbol{\beta}) + \sum_{j=1}^{p} p_a(|\boldsymbol{\beta_j}.|, \lambda) \tag{4.5}$$

Where $p_a$ is given by equation (6.4).
As with the grouped lasso, $\hat{\boldsymbol{\beta}}(\lambda) = argmin_{\boldsymbol{\beta}} \tilde{\ell}(\boldsymbol{\beta}, \lambda)$ will be row-sparse.

**MCP Penalty** Zhang (2010) proposed the *MCP* penalty [23], also defined on $[0, \infty)$. In the usual case where the penalty is applied individually to each coefficient, the penalty is given by

$$p_a(\beta_{j\cdot}, \lambda) = \begin{cases} \lambda|\beta_{j\cdot}| - \dfrac{\beta_{j\cdot}^2}{2a} & |\beta_{j\cdot}| \leq a\lambda \\ \dfrac{1}{2}a\lambda^2 & |\beta_{j\cdot}| > a\lambda \end{cases} \tag{4.6}$$

The derivative is given by

$$p_a'(\beta_{j\cdot}, \lambda) = \begin{cases} \lambda - \dfrac{|\beta_{j\cdot}|}{a} & |\beta_{j\cdot}| \leq a\lambda \\ 0 & |\beta_{j\cdot}| > a\lambda \end{cases} \tag{4.7}$$

The MCP penalty is similar to that of the SCAD penalty, in that the penalty term increases at a rate of $\lambda$ for $\beta_{j\cdot} = 0$, but the penalty "tapers off" as $|\beta_{j\cdot}|$ increases, plateauing

at $|\beta_{j\cdot}| > a\lambda$.

When $\boldsymbol{\beta}$ is a matrix, the penalized log-likelihood is then given by

$$\tilde{\ell}(\boldsymbol{\beta}, \lambda) = -\ell(\boldsymbol{\beta}) + \sum_{j=1}^{p} \sum_{k=1}^{K} p_a(|\beta_{j,k}|, \lambda) \tag{4.8}$$

where $p_a$ is given by equation (6.6).

Like SCAD, this penalty will result in an element-sparse matrix of coefficients.

**Grouped MCP Penalty** As with SCAD, we can instead extend the MCP penalty to a matrix of coefficients by grouping each row of the matrix, using an $\mathcal{L}_2$ penalty within each row.

The *Grouped MCP* penalty is given by

$$p_a(\boldsymbol{\beta_{j\cdot}}, \lambda) = \begin{cases} \lambda ||\boldsymbol{\beta_{j\cdot}}||_2 - \dfrac{||\boldsymbol{\beta_{j\cdot}}||_2^2}{2a} & ||\boldsymbol{\beta_{j\cdot}}||_2 \leq a\lambda \\ \dfrac{1}{2} a\lambda^2 & ||\boldsymbol{\beta_{j\cdot}}||_2 > a\lambda \end{cases} \tag{4.9}$$

where $||\boldsymbol{\beta_{j\cdot}}||_2 = \sqrt{\sum \beta_{jk}^2}$.

As with the group SCAD penalty, the penalized log-likelihood is then given by

$$\tilde{\ell}(\boldsymbol{\beta}, \lambda) = -\ell(\boldsymbol{\beta}) + \sum_{j=1}^{p} p_a(|\boldsymbol{\beta_{j\cdot}}|, \lambda) \tag{4.10}$$

where $p_a$ is given by equation (6.9).

As with SCAD and the grouped lasso, $\hat{\boldsymbol{\beta}}(\lambda) = argmin_{\boldsymbol{\beta}} \tilde{\ell}(\boldsymbol{\beta}, \lambda)$ will be row-sparse.

# Part II

# Post-Selection Inference

# 5 Naive Methods of Performing Inference

In this section, all experiments will be performed on a linear model, of the form

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

where $\boldsymbol{Y}$ is an $n \times 1$ vector, $\boldsymbol{X}$ is an $n \times p$ matrix, $\boldsymbol{\beta}$ is a $p \times 1$ vector, and $\boldsymbol{\epsilon} \sim \mathcal{N}_n(\mathbf{0}_n, \boldsymbol{\Sigma})$.

## 5.1 Forward Selection

Forward selection, when variables are added one at a time to a model in order of their marginal effect on reducing the sum of squares of the residuals, is a very commonly used method to perform variable selection. Its popularity is due to a combination of it being very easy to implement and understand. However, it can be very dangerous when its effects are not properly understood by the user.

Say we wish to test the significance of a group of covariates $\{x_1, ..., x_m\}$, out of the full model $\{x_1, ..., x_p\}$. Let $RSS_{FM}$ be the residual sum of squares of the fitted model containing all the covariates (the *full model*), and let $RSS_{RM}$ be the residual sum of squares of the "restricted" or "reduced" fitted model, where the covariates of interest are not included. Let $p_{FM}$ and $p_{RM}$ be the respective number of parameters of the full and restricted model (so $p_{FM} - p_{RM} = m$), and let $n$ be the number of observations. We can then calculate the $F$ statistic:

$$F = \frac{\frac{RSS_{RM} - RSS_{FM}}{p_{FM} - p_{RM}}}{\frac{RSS_{FM}}{n - p_{FM}}} \tag{5.1}$$

Under the null hypothesis $H_0 : \beta_1 = ... = \beta_m = 0$, we have that $F \sim F_{m, n-p_{FM}}$, the $F$ distribution with $m = p_{FM} - p_{RM}$ and $n - p_{FM}$ degrees of freedom.

When the full model is used (no variable selection), using an $F$ or $\chi^2$ test to perform significance tests on one or multiple variables is very simple. One may think that we can use an $F$ test on the drop in RSS to test the significance of a variable added at a given step of forward selection. However, the test statistic under the null no longer has the expected $F$ distribution. This is due to the "greediness" of the algorithm, which "cheats" at each

step by choosing the model which maximises the test statistics. In other words, the full and reduced model are not fixed - rather, they have been selected based on the data.

**Experiment Design** In this and all future experiments, the following parameters are used. The design matrix $\boldsymbol{X}$ is multivariate normal with mean $\mathbf{0_{60}}$ and variance-covariance matrix $\boldsymbol{\Sigma}$, where $\boldsymbol{\Sigma}_{i,j} = 0.8^{|i-j|}$. To generate the vector of responses $\boldsymbol{Y}$, $\boldsymbol{\beta}$ is set to 0 (ie, $\boldsymbol{Y}$ does not actually depend on $\boldsymbol{X}$). The variance $\boldsymbol{\Sigma}$ is set to the identity matrix $\boldsymbol{I}_{60}$, so each response observation $Y_i$ is therefore normally distributed with mean 0 and variance 1.
For simplicity, the intercept term $\beta_0$ is set to 0.

**The Experiment** We use `R` to implement simulations of forward selection and study the results. 10 design matrices $\boldsymbol{X} \sim N_{60}(\mathbf{0_{60}}, \boldsymbol{\Sigma})$ are generated, and for each design matrix 100 response vectors $\boldsymbol{Y}$ are generated, for a total of 1000 pairings

$$\{(\boldsymbol{X_1}, \boldsymbol{Y_{1,1}}), ..., (\boldsymbol{X_1}, \boldsymbol{Y_{1,100}}), (\boldsymbol{X_2}, \boldsymbol{Y_{2,1}}), ..., (\boldsymbol{X_{10}}, \boldsymbol{Y_{10,100}})\}$$

For each pairing we found the "best" covariate (out of 60) using forward selection (using the `step` function in the `stats` package [20]). We then fit a linear regression model using only the "selected" parameter and calculated the $F$ statistic. Then the $F$ statistic is used to decide whether or not to reject the null hypothesis $H_0 : \beta = 0$ at the 5% level, and we record how often the null hypothesis was rejected. Since the null hypothesis is in fact correct, the proportion of times when the null is rejected provides an estimate of the true Type I error of this test. Using the reasoning described in the previous section, we suspect that the true Type I error will be greater than the desired 5%.
In fact, the null hypothesis was rejected 790 times out of the 1000 iterations of the experiment. Under these conditions, this corresponds to an estimated Type I error of 79%! We can attribute much of this horrendous outcome to spurious correlation - when forward selection has 60 parameters to choose from, one of them is bound to be highly correlated with the response. By design, this will be the variable chosen by the forward selection algorithm.

**Cross Validation** It is possible to perform a valid significance test by using cross validation. If the data is split into a training and testing set and the model was not chosen using the test set, we can perform valid inference (using the $F$ statistic or other methods) using the data from the test set.

However, in practice this method is not always feasible. Often variable selection is performed when the number of parameters is very high but the number of observations if low. For example, if one has only 50 observations cross validation would require an average of only 25 observations in the training and testing set. Unfortunately, 25 observations is not enough to get anything useful out of the data. There is too little data to perform accurate model selection, the only reason forward selection is being used in the first place. However, since the model was *not* selected using the data (specifically, the testing data) the significance tests performed would be valid. Unfortunately, valid and useful are two seperate qualities. Since the length of the confidence intervals are roughly proportional to the inverse of the square-root of $n$, the number of observations, they would be far too wide to be of practical use.

## 5.2 Lasso

We can perform similar simulations to those of the previous section using the Lasso. Again, we simulated the data using $\boldsymbol{\beta} = \mathbf{0}$ and selecting one variable using the lasso (by using a large enough $\lambda$ such that only one variable is selected). Then the $F$ statistic is used to (naively) test the significance of the added variable at the 5% level. This is repeated for 1000 pairings of $(\boldsymbol{X}, \boldsymbol{Y})$ (as described in the previous section) and we recorded how often $H_0$ was rejected in order to get an estimate of the true Type I error. In the simulations, the null hypothesis was rejected 44% of the time. While still very bad, the effect is slightly less dramatic than the results from forward selection. This is due to the fact that when using the Lasso, variables are not added to the model for the explicit purpose of reducing the RSS (as they are with forward selection).

## 5.3 Naively Estimating P-Values

Rather than deal with individual hypothesis tests, it may be of interest to find the p-value of all the parameters selected in the model. Classical statistical theory gives us a way to compute these p-values: If there are $p$ parameters in the model, we have that under $H_0 : \beta_j = 0$, for $1 \leq j \leq p$,

$$T_j = \frac{\hat{\beta}_j}{SE(\hat{\beta}_j)} \sim t_{n-p} \tag{5.2}$$

where $t_{n-p}$ follows a Student-t distribution with $n - p$ degrees of freedom.

Of course, we should realize that when the model is selected using the data, the standard errors will not be accurately estimated. However, comparing the p-values estimated using the above formula to those estimated (in the next section) using Monte Carlo methods gives us insight into how one can easily be misled.

In this experiment, we used the previously described methods to generate 1000 datasets (using $\boldsymbol{\beta} = \mathbf{0}$). For each response vector, certain covariates are selected using the Lasso (with $\lambda = 0.1$), and refit an unpenalized regression model using only these covariates. Then, every estimated coefficient is recorded - that is, for each response vector we recorded every $\hat{\beta}_j, j \in \{j : \hat{\beta}_j(\lambda = 0.1) \neq 0\}$. The $T$ statistic for each of these coefficients is also recorded, and used to calculate a corresponding p-value (using the quantiles of the Student-t distribution from equation 3). This gives us a large amount of estimated coefficients and their (naively estimated) p-values, which we can plot to get an idea of "how large must $\hat{\beta}_j$ be to reject $H_0 : \beta_j = 0$?" This question cannot be answered definitively since it fails to take the standard errors into account. Instead, we can plot the $T$ statistics and their corresponding p-values (calculated using equation 3) to get an idea of what significance we can assign to a given $T$ statistic using classical statistical theory.

Figure 4: P-values "naively" assigned to various values of $|\hat{\beta}|$

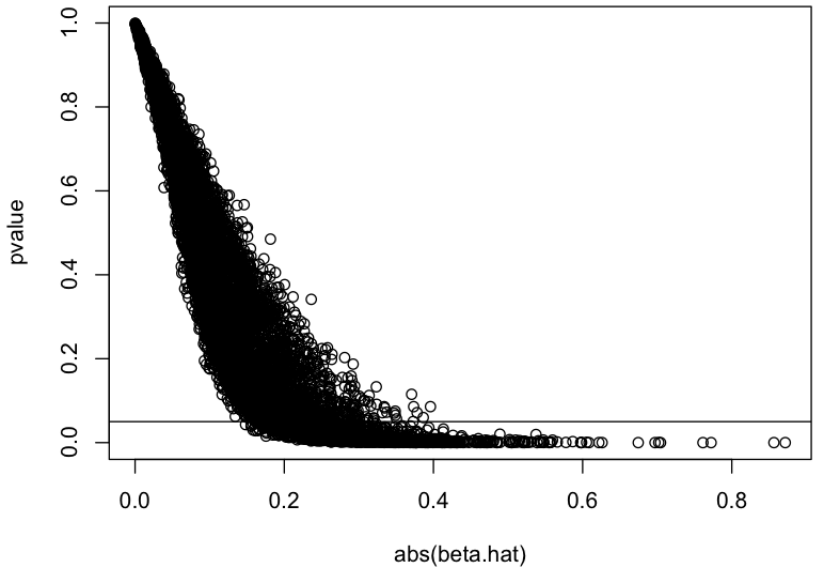
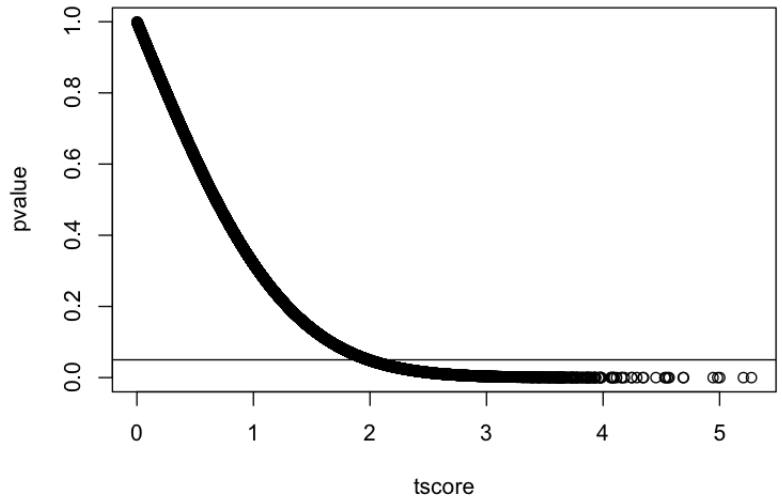



Figure 5: P-values "naively" assigned to various t-scores




As we can see, a t-score of $\geq 2$ corresponds to a p-value of $\leq 0.05$ (In fact, the above plot is simply a plot of the absolute student-t distribution). Similarly, we see that having $\hat{\beta} \geq 0.2 - 0.4$ would lead us to reject the null hypothesis (this is less exact, since the standard error estimates are needed for significance testing).

# 6 Finding Emperical P-Values

While it is easy to prove that the $T$ statistics will not follow the distribution given in equation 3, it is of interest to find their true distributions. Using equation 3, we have that

$$P(|T_j| \geq t_{n-p,0.975} | \beta_j = 0) = 0.05 \tag{6.1}$$

where $t_{n-p,0.975}$ is the 97.5-th percentile of the student-t distribution with $n - p$ degrees of freedom.

However, we know that when variable selection is performed the above formula no longer holds. To estimate the true value of $P(|T_j| \geq t_{n-p,0.975} | \beta_j = 0)$, we can look at all the $T$ statistics generated using the methods described in the previous section. Since the null hypothesis $H_{0,j} : \beta_j = 0$ is true for all $j$, we can study the empirical distribution of the $T$ statistics.

This allows us to find the true "a posteriori" (ie, taking the selection step into account) p-value for $T_j = a$. Specifically, under the null hypothesis $\beta_j = 0$, what is the probability that the $T_j \geq a$, given that the $j$-th covariate was selected? This can be easily estimated using the quantiles of the *observed* $T$ statistics.

**Calculating Emperical P Values** Let $M$ be the number of estimated (non-zero) coefficients (over all of the iterations) and $t_i$ be the $i$-th t-score calculated. Note that $M > 1000$ since more than one parameter was selected in each iteration. Also note that by definition, under the null hypothesis a t-score of $a$ would have a p-value of $p$ s.t. $E[\frac{1}{M}\sum_{i=1}^{M} I(t_i > a)] = p$. Therefore, since the null hypothesis does hold in this case, an "empirical p-value" $\hat{p}$ for a t-score of $a$ would be $\hat{p} = \frac{1}{M}\sum_{i=1}^{M} I(t_i > a)$. This provides us with an unbiased estimate of the true p-value $p$ of a t-score $a$.

Using this formula we can assign an empirical p-value for every t-score. We also found empirical p-values for the estimated coefficients $\hat{\beta}$. These are helpful to compare against the "naively calculated" p-values, but are not very useful on their own since it is impossible to

get an accurate estimate without taking standard errors into account.

**Results** Below is a plot of the $M$ estimated coefficients $\hat{\beta}$ (x-axis) with their corresponding empirical p-values (y-axis), followed by a plot of the $M$ estimated T-scores $\hat{t}$ (x-axis) with their corresponding empirical p-values (y-axis)

Figure 6: Empirical p-values assigned to various values of $|\hat{\beta}|$

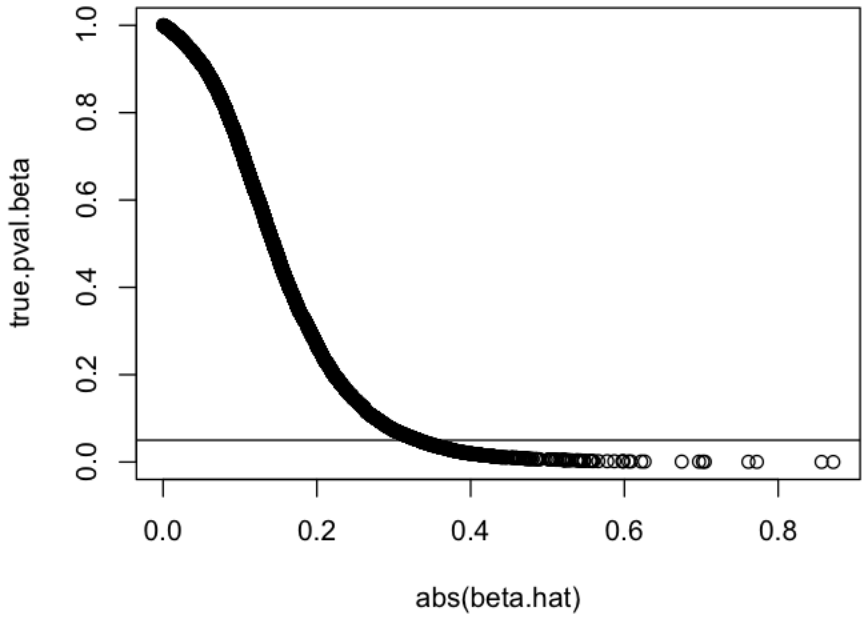
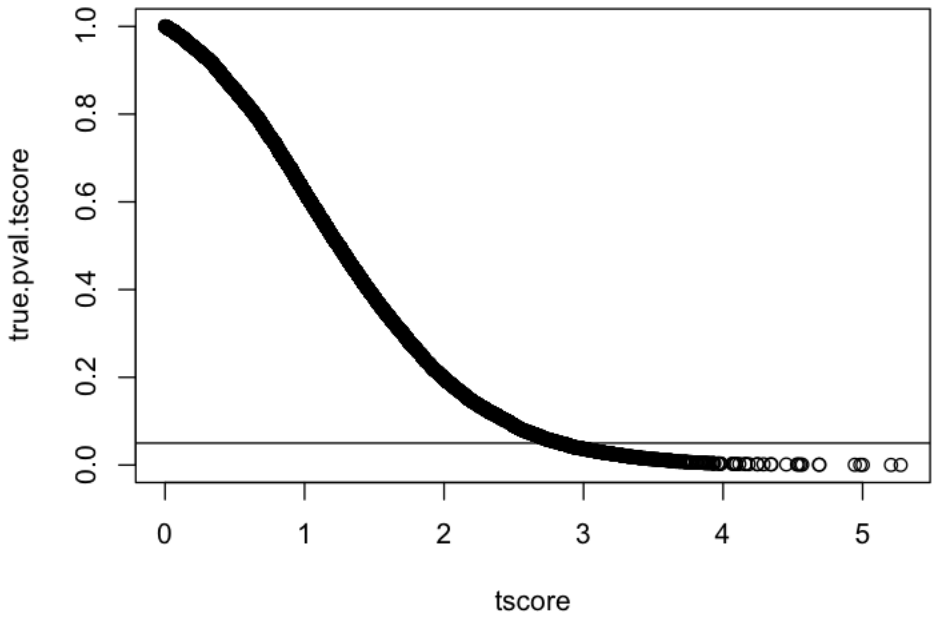


Figure 7: Empirical p-values assigned to various t-scores

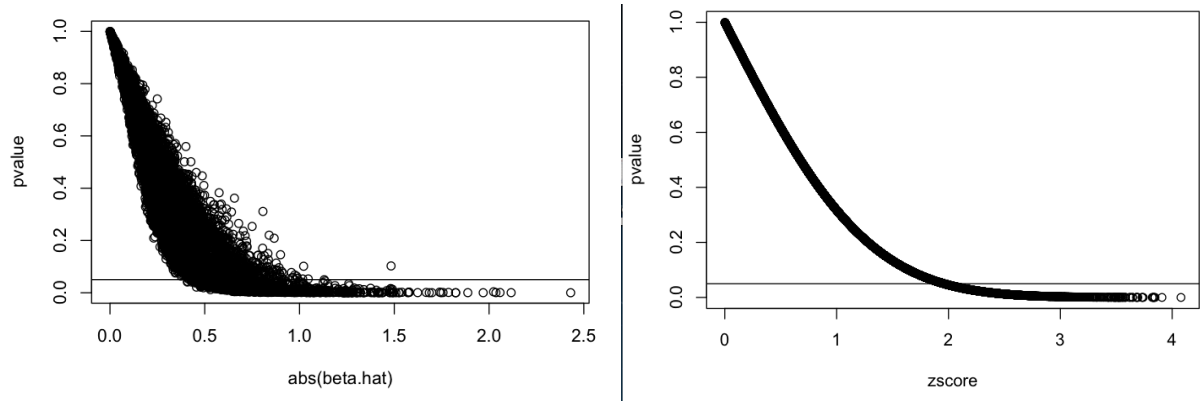



We see that in reality, we get a t-score of $\geq 2$ about 20% of the time, meaning the naive tests at the 5% level result in Type I errors of 20%. Instead, these Monte Carlo simulations tell us that we should only reject the null hypothesis if $t \geq 3$.
Specifically, this experiment shows us that the *empirical* p-value of a T-score of 2 is 0.2, and a T-score of 3 has an *empirical* p-value of 0.05.

## 6.1 Logistic Regression

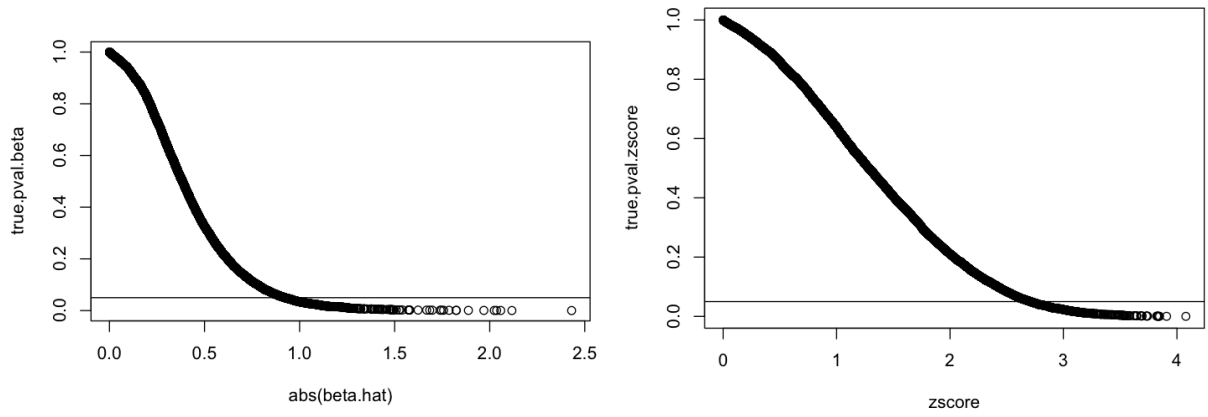
We can repeat the above simulations using logistic regression instead of linear regression. The plots of the estimated coefficients and t-scores (x-axis) against the "naive" p-values are:

Figure 8: P-values "naively" assigned to various t-scores and values of $|\hat{\beta}|$




The plots of the estimated coefficients and t-scores (x-axis) against the "empirical" p-values are:

Figure 9: Empirical p-values assigned to various t-scores and values of $|\hat{\beta}|$




The results are very similar to those from linear regression.

# 7 Finding a Valid "Desparsified" Lasso Estimator

While the above algorithm can be used to find p-values, it is a computationally expensive method. Instead, Zhang and Zhang [24] and Van de Geer et al [2] demonstrated that

it is possible to find the distribution of the Lasso-estimated coefficients $\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}}(\lambda)$ using the Karush-Kuhn-Tucker conditions. In this section we will explain the theory behind this method, and in the next section we will explain the approach we took to verify these methods through simulations.

We have that:

$$\hat{\boldsymbol{\beta}}(\lambda) = argmin \frac{1}{n} ||\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2 + 2\lambda ||\boldsymbol{\beta}||_1 \tag{7.1}$$

The KKT conditions can be seen by setting the gradient to zero:

$$0 = -\boldsymbol{X}^T(\boldsymbol{Y} - \boldsymbol{X}\hat{\boldsymbol{\beta}})/n + \lambda \hat{\boldsymbol{\kappa}} \tag{7.2}$$

For $\hat{\kappa}_j \in [-1, 1]\ \forall j, \hat{\kappa}_j = sign(\hat{\beta}_j)$ if $\hat{\beta}_j \neq 0$

Let $\boldsymbol{\beta}^0$ be the true value of $\boldsymbol{\beta}$, so that $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta^0} + \boldsymbol{\epsilon}$.

Let $\hat{\boldsymbol{\Sigma}} = \boldsymbol{X}^T\boldsymbol{X}/n$. Then:

$$\hat{\boldsymbol{\Sigma}}(\boldsymbol{\beta} - \hat{} \boldsymbol{\beta^0}) + \lambda \hat{\boldsymbol{\kappa}} = \boldsymbol{X}^T\boldsymbol{\epsilon}/n \tag{7.3}$$

We will need to invert $\hat{\boldsymbol{\Sigma}}$. If instead we use a "reasonable approximation" $\hat{\boldsymbol{\Theta}}$, we get:

$$\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}^0 + \hat{\boldsymbol{\Theta}}\lambda\hat{\boldsymbol{\kappa}} = \hat{\boldsymbol{\Theta}}\boldsymbol{X^T}\boldsymbol{\epsilon}/n - \Delta/\sqrt{n} \tag{7.4}$$

where $\Delta := \sqrt{n}(\hat{\boldsymbol{\Theta}}\hat{\boldsymbol{\Sigma}} - \boldsymbol{I})(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta^0})$

Note that $\Delta$ is zero if $\hat{\boldsymbol{\Theta}}$ if the true inverse of $\hat{\boldsymbol{\Sigma}}$, and small if $\hat{\boldsymbol{\Theta}}$ is a good approximation. Van de Geer et al showed that $\Delta$ is asymptotically negligible ($o_P(1)$) under "certain sparsity assumptions" (see Theorem 2.2 in [2]).

Now, we can get a promising "de-sparsified" estimator for $\boldsymbol{\beta}^0$:

$$\hat{\boldsymbol{b}} = \hat{\boldsymbol{\beta}} + \hat{\boldsymbol{\Theta}}\lambda\hat{\boldsymbol{\kappa}} = \hat{\boldsymbol{\beta}} + \hat{\boldsymbol{\Theta}}\boldsymbol{X^T}(\boldsymbol{Y} - \boldsymbol{X}\hat{\boldsymbol{\beta}})/n \tag{7.5}$$

We then get

$$\sqrt{n}(\hat{\boldsymbol{b}} - \boldsymbol{\beta^0}) = \hat{\boldsymbol{\Theta}}\boldsymbol{X^T}\boldsymbol{\epsilon}/\sqrt{n} - \Delta \tag{7.6}$$

Note that $\boldsymbol{\epsilon} \sim \mathcal{N}_n(0, \sigma_\epsilon^2 \boldsymbol{I}) \Rightarrow \frac{\hat{\boldsymbol{\Theta}}_{\boldsymbol{p}\times\boldsymbol{p}}\boldsymbol{X^T}_{p\times n}}{\sqrt{n}}\boldsymbol{\epsilon} \sim \mathcal{N}_p(0, \sigma_\epsilon^2(\hat{\boldsymbol{\Theta}}\boldsymbol{X^T})(\hat{\boldsymbol{\Theta}}\boldsymbol{X^T})^T/n) = \mathcal{N}_p(0, \sigma_\epsilon^2\hat{\boldsymbol{\Theta}}\hat{\boldsymbol{\Sigma}}\hat{\boldsymbol{\Theta}}^T)$

Let $\hat{\boldsymbol{\Omega}} = \hat{\boldsymbol{\Theta}}\hat{\boldsymbol{\Sigma}}\hat{\boldsymbol{\Theta}}^{\boldsymbol{T}}$. This gives us our pivot distribution:

$$\sqrt{n}(\hat{\boldsymbol{b}} - \boldsymbol{\beta^0}) \sim \mathcal{N}_p(0, \sigma_\epsilon^2\hat{\boldsymbol{\Omega}}) + o_P(1) \tag{7.7}$$

This gives us the $1-\alpha$ confidence intervals, $P(\beta_j^0 \in \hat{b}_j \pm c(\alpha, n, \sigma_\epsilon)) = 1-\alpha$, where

$$c(\alpha, n, \sigma_\epsilon) := \Phi^{-1}(1-\alpha/2)\sigma_\epsilon\sqrt{\hat{\Omega}_{j,j}/n},\ \Phi \textit{ is the cdf of } \mathcal{N}(0,1) \tag{7.8}$$

We can use this method even if $\boldsymbol{\epsilon}$ is not normally distributed. In this case,

$$\sqrt{n}(\hat{\boldsymbol{b}} - \boldsymbol{\beta^0}) = \boldsymbol{W} + \boldsymbol{\Delta}$$

where $\boldsymbol{W} = \hat{\boldsymbol{\Theta}}\boldsymbol{X^T}\boldsymbol{\epsilon}/\sqrt{n}, |\Delta_j| = o_P(1)\ \forall j$. Note that unlike in the previous case, $\boldsymbol{W}$ need not be normal (though the CLT can be used to get a normal approximation).

## 7.1 Verification

To verify the results claimed in the thesis, we generated (using the methods previously described) 1000 datasets $(\boldsymbol{X_i}, \boldsymbol{Y_i})$, where $\boldsymbol{X_i}$ is a $100 \times 60$ matrix and $\boldsymbol{Y_i} \sim \boldsymbol{X\beta} + \boldsymbol{\epsilon}$ (where $\boldsymbol{\beta} = \boldsymbol{0_{60}}$) is a $60 \times 1$ vector.

The lasso estimator is used to estimate $\hat{\boldsymbol{\beta}}$. $\hat{\boldsymbol{\Sigma}}, \hat{\boldsymbol{\Theta}}, \hat{\boldsymbol{\Omega}}$ are calculated and used to find the desparsified estimator $\hat{\boldsymbol{b}}$ and its confidence interval. For these experiments, $\boldsymbol{\Theta}$ is solved explicitly by inverting $\boldsymbol{\Sigma}$. Note that in situations where $p$ is very large, inverting $\boldsymbol{\Sigma}$ directly cannot be done. Instead we can compute a "reasonable approximation" using a method such as *Node-Wise Regression*, which is described in detail in [2]. The null hypothesis $H_{0,j} : \beta_j^0 = 0$ is rejected if zero is not contained in the 5% confidence interval $\hat{b}_j \pm c(\alpha, n, \sigma_\epsilon)$.

Since $H_{0,j}$ is in fact true for all $j$, if this test is valid then $H_{0,j}$ should be rejected 5% of the time (a Type I error of 5%). In the simulations performed in `R`, the null hypotheses $H_{0,j} : \beta_j^0 = 0$ were rejected an average of 5.000467% of the time, as expected.

## 7.2 Desparsified Estimator for GLMs

This method can also be extended to General Linear Models. We will use the notation of [2]. Let $\rho_\beta := \rho_{\boldsymbol{\beta}}(y, \boldsymbol{x}) = \rho(y, \boldsymbol{x\beta})$ be the loss function (usually the negative log likelihood) of a single observation. Keeping with the notation of [2], we let $P_n$ be the "mean operator",

so $P_n\rho_\beta = \frac{1}{n}\sum_{i=1}^{n}\rho_\beta(y_i, x_i)$.

Let $\hat{\boldsymbol{\beta}} = argmin_{\boldsymbol{\beta}}(P_n\rho_{\boldsymbol{\beta}} + \lambda||\boldsymbol{\beta}||_1)$. Also let $\hat{\boldsymbol{\Sigma}} = P_n\ddot{\rho}_{\hat{\boldsymbol{\beta}}}$ (in general, $\hat{\boldsymbol{\Sigma}}$ may depend on $\boldsymbol{\beta}$, although it does not in the case of linear regression).

Let $\hat{\sigma}_j^2 = (\hat{\Theta}P_n\dot{\rho}_{\hat{\boldsymbol{\beta}}}\dot{\rho}_{\hat{\boldsymbol{\beta}}}^T\hat{\Theta}^T)_{j,j}$ and $\hat{\boldsymbol{b}} = \hat{\boldsymbol{\beta}} - \hat{\boldsymbol{\Theta}}P_n\dot{\rho}_{\hat{\boldsymbol{\beta}}}$.

Using a Taylor Expansion and the Central Limit Theorem, we can show that

$\sqrt{n}(\hat{b}_j - \beta_j^0)/\hat{\sigma}_j = V_j + o_P(1)$ (see [2] for the proof).

We verified the validity of the desparsified estimator for logistic regression, using the methods described in section **11.1**. In this case it is necessary to compute the loss function and its derivatives and double derivatives. They are:

$$\rho_{\boldsymbol{\beta}}(y, x) = -(y\boldsymbol{x^T\beta} - \log(1 + e^{\boldsymbol{x^T\beta}}))$$

$$\dot{\rho}_{\boldsymbol{\beta}} = -(y\boldsymbol{x^T} - \boldsymbol{x^T}\frac{e^{\boldsymbol{x^T\beta}}}{1 + e^{\boldsymbol{x^T\beta}}}) \text{ (a px1 vector)}$$

$$\ddot{\rho}_\beta = \boldsymbol{x^T x}\frac{e^{\boldsymbol{x^T\beta}}}{(1 + e^{\boldsymbol{x^T\beta}})^2} \text{ (a pxp matrix)}$$

$$P_n\dot{\rho}_{\hat{\boldsymbol{\beta}}} = -\frac{1}{n}\sum_{i=1}^{n}(y_i\boldsymbol{x_i^T} - \boldsymbol{x_i^T}\frac{e^{\boldsymbol{x_i^T\hat{\beta}}}}{1 + e^{\boldsymbol{x_i^T\hat{\beta}}}}) \text{ (px1 vector)}$$

$$P_n\dot{\rho}_{\hat{\boldsymbol{\beta}}}\dot{\rho}_{\hat{\boldsymbol{\beta}}^T} = \frac{1}{n}\sum_{i=1}^{n}(y_i\boldsymbol{x_i^T} - \boldsymbol{x_i^T}\frac{e^{\boldsymbol{x_i^T\hat{\beta}}}}{1 + e^{\boldsymbol{x_i^T\hat{\beta}}}})(y_i\boldsymbol{x_i^T} - \boldsymbol{x_i^T}\frac{e^{\boldsymbol{x_i^T\hat{\beta}}}}{1 + e^{\boldsymbol{x_i^T\hat{\beta}}}})^T \text{ (pxp matrix)}$$

$$\hat{\boldsymbol{\Sigma}} = P_n\ddot{\rho}_{\hat{\boldsymbol{\beta}}} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x_i^T x_i}\frac{e^{\boldsymbol{x_i^T\hat{\beta}}}}{(1 + e^{\boldsymbol{x_i^T\hat{\beta}}})^2} \text{ (pxp matrix)}$$

When this technique was verified using `R` (with the methods described in section **11.1**), the Type I error was 4.9%.

## 7.3 Group Lasso

Let $\boldsymbol{\beta} = (\boldsymbol{\beta_1}, ..., \boldsymbol{\beta_G})$ where $\boldsymbol{\beta_g} = (\beta_{g1}, ..., \beta_{gn_g})$.

Say we want to select all or none of each group $\boldsymbol{\beta_g}$ of variables. This situation comes up often - you may believe (before fitting a model) that certain covariates are highly related and that it would not make sense to say that some are important and others are not. This

is also an issue when dealing with categorical variables - in this case, a $K$-class categorical variable must be substituted by $K-1$ "dummy" variables. Since all these dummy variables represent the same original classification, they should all be grouped together. It would make no sense to select some dummy variables but not others, as this would imply that the variable being in certain classes affects the response while the variable being in other classes does not. Since there is a direct relationship between the dummy variables (if one of them is 1 the others must all be 0), it would be difficult to interpret a model where only a few classes are in the model.

The group lasso estimator is:

$$\hat{\boldsymbol{\beta}}_\lambda = argmin_\beta \frac{1}{2n}\sum_{i=1}^{n}(y_i - \boldsymbol{x_i^T}\boldsymbol{\beta}) + \lambda \sum_{g=1}^{G}\sqrt{n_g}||\boldsymbol{\beta}_g||_2 \tag{7.9}$$

Note that within each group, the coefficients are penalized using the $L_2$ norm, so no selection takes place. Between groups, we use the usual $L_1$ penalty, which will shrink some of the groups of coefficients to zero.

The KKT Conditions are:

$$0 = -\boldsymbol{X_g^T}(\boldsymbol{Y} - \boldsymbol{X\beta})/n + \lambda s(df_g)\hat{\boldsymbol{\kappa}}_g \tag{7.10}$$

where $\hat{\boldsymbol{\kappa}}_g \in [-1,1] \forall g, \hat{\boldsymbol{\kappa}}_g = \dfrac{\hat{\boldsymbol{\beta}}_g}{||\hat{\boldsymbol{\beta}}_g||_2}$ if $\hat{\boldsymbol{\beta}}_g \neq \mathbf{0}$.

If we let $\hat{\boldsymbol{\kappa}} = \{s(df_1)\hat{\kappa}_1, ..., s(df_n)\hat{\kappa}_n\}$, we can combine the groups to get the same formula as we had in the ungrouped case: $0 = -\boldsymbol{X^T}(\boldsymbol{Y} - \boldsymbol{X}\hat{\boldsymbol{\beta}})/n + \lambda\hat{\boldsymbol{\kappa}}$

This gives us the same pivotal quantity $\hat{b}$ where $\sqrt{n}(\hat{\boldsymbol{b}} - \boldsymbol{\beta^0}) \sim \mathcal{N}_p(0, \sigma_\epsilon^2\hat{\boldsymbol{\Omega}}) + o_P(1)$

However, unlike in the ungrouped case, when performing inference after group lasso we would usually want to perform inference on a group by group basis, rather than coefficient by coefficient. Therefore we should not study each $\hat{\beta}_j$ individually. Instead, we need to find a test statistic which provides insight into whether the group is important. A natural candidate would be $\max_{j\in g}\hat{b}_j$. If one of the coefficients in the group is very large, we should conclude that this coefficient must be important and therefore its group is important. On the other hand, if each coefficient in the group is moderately large but there is no obviously significant

coefficient, we may want to conclude that the magnitude of each of the coefficients can be explained by noise, and therefore there is not enough evidence to reject the null hypotheis that the group of variables is significant.

The method to compute joint confidence intervals is slightly more complicated than the method to compute individual confidence intervals, and involves using Monte Carlo simulations to estimate distributions.

We have

$$P(\max_{j\in g} \frac{\sqrt{n}|\hat{b}_j - \beta_j^0|}{\sigma_\epsilon\sqrt{\hat{\Omega}_{j,j}}} \leq z) = P(\max_{j\in g} \frac{|W_j|}{\sigma_\epsilon\sqrt{\hat{\Omega}_{j,j}}} \leq z) \tag{7.11}$$

where $\boldsymbol{W} \sim \mathcal{N}_p(0, \sigma_\epsilon^2\hat{\boldsymbol{\Omega}})$

Therefore, under $H_0 : \beta_j^0 = 0 \; \forall j \in g$, we have that

$$\max_{j\in g} \frac{n\hat{b}_j^2}{\sigma_\epsilon^2\hat{\Omega}_{j,j}} \sim \max_{j\in g} \chi_j^2(1) \tag{7.12}$$

However, the $\chi^2$ variables are dependendent, with

$$\chi_j^2(1) = \frac{|W_j|^2}{\sigma_\epsilon^2\hat{\Omega}_{j,j}} \tag{7.13}$$

This distribution does not have a closed form solution, but Monte Carlo simulations can be used to create a good approximation.

To create the confidence intervals we must find the 95-th percentile of $\max_{j\in g} \frac{|W_j|^2}{\sigma_\epsilon^2\hat{\Omega}_{j,j}}$

**Algorithm** The algorithm to test the null hypothesis $H_{0,g} : \beta_j = 0 \; \forall j \in g$ for the $g$-th group, $g = 1, ..., G$, is:

1) Generate $\boldsymbol{W_1}, ..., \boldsymbol{W_m}$ for $m$ large, where each $\boldsymbol{W_i} \sim \mathcal{N}_p(0, \sigma_\epsilon^2\hat{\boldsymbol{\Omega}})$ (a $1 \times p$ vector)

2) Let $\chi_{i,j}^2(1) = \frac{W_{i,j}^2}{\sigma_\epsilon^2\hat{\Omega}_{j,j}}$

3) Calculate $m_{i,g} = \max_{j\in g} \chi_{i,j}^2(1) \; \forall i, g$

4) Find $\hat{q}_{0.95}$, the 0.95 quantile of $m_{\cdot,g}$ for each $g$ (using the `quantile` function from the `stats` package in `R`).

5) Given our Lasso estimators $\hat{\beta}_j, j \in \{1, ..., p\}$, we reject $H_{0,g}: \beta_j = 0 \ \forall j \in g$ if

$$\max_{j \in g} \frac{n\hat{b}_j^2}{\sigma_\epsilon^2 \hat{\Omega}_{j,j}} > \hat{q}_{0.95,g} \tag{7.14}$$

**Verification** To verify the results claimed in the thesis, we generated (using `R`) 1000 datasets $(\boldsymbol{X_i}, \boldsymbol{Y_i})$, where $\boldsymbol{X_i}$ is a $100 \times 60$ matrix and $\boldsymbol{Y_i} \sim \boldsymbol{X\beta} + \boldsymbol{\epsilon}$ (where $\boldsymbol{\beta} = \boldsymbol{0_{60}}$) is a $60 \times 1$ vector.

We tested the null hypothesis $H_{0,g} : \beta_j^0 = 0 \ \forall j \in g$ for $g \in \{\{1, 2, ..., 10\}, \{11, 12, ..., 20\}, ..., \{51, 52, ..., 60\}\}$. Using `R`, the null hypothesis $H_{0,g}$ is rejected or accepted using the the algorithm described above. Since $H_{0,g}$ is in fact true for all $g$, if this test is valid then $H_{0,g}$ should be rejected 5% of the time (a Type I error of 5%). In the simulations performed in `R`, the null hypotheses $H_{0,g} : \beta_g^0 = 0$ were rejected an average of 5.11% of the time.

## 7.4 Extension to Multinomial Regression

While there is an abundance of literature analyzing logistic regression, multinomial regression gets less attention, especially in the field of post-selection inference. Having three or more categories greatly complicates regression - unlike in logistic regression, where the response is a binary variable $y_i$, a $K$-class multinomial model requires $K$ responses $y_{i,k}, k = 1, ..., K$, where $y_{ik} = 1$ if the $i$-th observation falls into the $k$-th class.

**Likelihood Functions** When performing selection in a multinomial model, we want to either select a whole row $\boldsymbol{\beta}_{j\cdot}$ or none of it.

Let $y_{ik} = 1(y_i = k)$. The average negative log-likelihood and its derivatives are

$$P_n \rho_{\boldsymbol{\beta}} = -\frac{1}{n} \sum_{i=1}^{n} [\sum_{k=1}^{K} y_{ik} (\boldsymbol{x_i \beta_{\cdot k}}) - \log(\sum_{k=1}^{K} \exp(\boldsymbol{x_i \beta_{\cdot k}}))]$$

$$P_n \dot{\rho}_{\boldsymbol{\beta}_{\cdot \ell}} = -\frac{1}{n} \sum_{i=1}^{n} [y_{i\ell} \boldsymbol{x_i} - \frac{\boldsymbol{x_i} \exp(\boldsymbol{x_i \beta_{\cdot \ell}})}{\sum_{k=1}^{K} \exp(\boldsymbol{x_i \beta_{\cdot k}})}]$$

$P_n \dot{\rho}_\beta = (P_n \dot{\rho}_{\beta_{1\cdot}}, ..., P_n \dot{\rho}_{\beta_{K\cdot}})$, *a p*K x 1 vector*

$P_n \ddot{\rho}_\beta$ is a $p * K \times p * K$ matrix. The blocks on the diagonal are

$$P_n\ddot{\rho}_{\beta_\ell} = \frac{1}{n}\sum_{i=1}^{n} x_i^T x_i \exp(x_i.\beta_{.\ell})\frac{\sum_{k\neq\ell}\exp(x_i\beta_{.k})+1}{(\sum_{k=1}^{K}\exp(x_i\beta_{.k}))^2}$$

The off-diagonal blocks are $P_n\ddot{\rho}_{\boldsymbol{\beta}_{\ell m}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x_i^T x_i}\exp(\boldsymbol{x_i\beta_{.\ell}})\exp(\boldsymbol{x_i\beta_{.m}})/(\sum_{k=1}^{K}\exp(\boldsymbol{x_i\beta_{.k}}))^2$

**Performing Inference** My research focuses on how to perform valid Post-Selection Inference in Multinomial Regression, where there has been no previous work.

Let $\hat{\Omega}_{pK\times pK} = \hat{\Theta}P_n\dot{\rho}_{\hat{\beta}}\dot{\rho}_{\hat{\beta}}^T\hat{\Theta}^T$ and $V \sim \mathcal{N}_{p\cdot K}(0,\hat{\Omega})$

We want to "vectorize" the matrix $\boldsymbol{\beta}$, so we will write $\beta_{jk}$, the $((j-1)\cdot K+k)$-th element of the vector $\boldsymbol{\beta}$, to represent the $k$-th column of the $j$-th row of the matrix beta, and the same for $V_{jk}$.

Similarly, $\hat{\Omega}_{jk,jk}$ will be the $((j-1)\cdot K+k)$-th diagonal element of $\hat{\Omega}$.

Then for a GLM, we have that $\frac{\sqrt{n}(\hat{b}_j-\beta_j^0)}{\sqrt{\hat{\Omega}_{j,j}}} \sim \frac{|V_j|}{\sqrt{\hat{\Omega}_{j,j}}}$

Similarly to the linear version of the group lasso, we can show that under

$H_{0,j}: \beta_{jk}^0 = 0\ \forall j, \max_j \frac{n\hat{b}_{jk}^2}{\hat{\Omega}_{jk,jk}} \sim \max_k \chi_{jk}^2(1)$

As with the group lasso, we can approximate the distribution of $\max_k \chi_{jk}^2(1)$ using Monte-Carlo simulations.

**Algorithm** Since applying the lasso for multinomial regression is a special case of the group lasso, the algorithm to test for significance is a special case of the algorithm for grouped lasso (described in the previous section). The key step for multinomial regression is properly grouping the covariates, so that $\boldsymbol{\beta_j}.$ is contained in one grouping. This is done in "step 0" of the following algorithm. Steps 1-5 are similar to those of the algorithm for grouped lasso.

The algorithm to test the null hypothesis $H_{0,j}: \beta_{jk}^0 = 0\ \forall k$ for the $j$-th covariate, $j = 1,...,p$, is:

0) Create groups $j = 1,...,p$, where the $j$-th group is given by the

$\{(j-1)\cdot K+1, (j-1)\cdot K+2, ..., (j-1)\cdot K+(K-1), j\cdot K\}$-th elements of $\boldsymbol{\beta}$.

Notice that the $j$-th group represents the $j$-th covariate, or the $j$-th row of the matrix $\boldsymbol{\beta}$.

1) Generate $\boldsymbol{W_1, ..., W_m}$ for $m$ large, where each $\boldsymbol{W_i} \sim \mathcal{N}_{p \cdot K}(0, \sigma_\epsilon^2 \hat{\boldsymbol{\Omega}})$ (a $1 \times p \cdot K$ vector)

2) Let $\chi^2_{i,\ell}(1) = \dfrac{W^2_{i,\ell}}{\sigma_\epsilon^2 \hat{\Omega}_{\ell,\ell}}$

3) Calculate $m_{i,j} = \max_{\ell \in j} \chi^2_{i,\ell}(1)\ \forall i, j$

4) Find $\hat{q}_{0.95}$, the 0.95 quantile of $m_{\cdot,j}$ for each $j$ (using `quantile` function from the `stats` package in `R`).

5) Given our Lasso estimators $\hat{\beta}_\ell, \ell \in \{1, ..., p \cdot K\}$, we reject $H_{0,j} : \beta_\ell = 0\ \forall \ell \in j$ if

$$\max_{\ell \in j} \frac{n \hat{b}^2_\ell}{\sigma_\epsilon^2 \hat{\Omega}_{\ell,\ell}} > \hat{q}_{0.95,j} \tag{7.15}$$

**Verification** To verify the results claimed in the thesis, we generated (using `R`) 1000 multinimial datasets $(\boldsymbol{X_i, Y_i})$, using $\boldsymbol{\beta}_{jk} = 0\ \forall j, k$, where $\boldsymbol{X_i}$ is a $100 \times 60$ matrix and $\boldsymbol{Y}$ is a $60 \times 1$ vector.

For each dataset $(\boldsymbol{X_i, Y_i})$, we computed the likelihood functions described above. We tested the null hypothesis $H_{0,j} : \beta^0_{jk} = 0\ \forall k \in \{1, ..., K\}$ for $j \in \{1, ..., p\}$. Using `R`, the null hypothesis $H_{0,j}$ is rejected or accepted using the the algorithm described above. Since $H_{0,j}$ is in fact true for all $j$, if this test is valid then $H_{0,j}$ should be rejected 5% of the time (a Type I error of 5%). In the simulations performed in `R`, the null hypotheses $H_{0,j} : \beta^0_{j\cdot} = 0$ were rejected an average of 5.05% of the time.

# Part III

# Conclusion and Future Work

# 8 Conclusion

As a flexible model which can be used for prediction in many fields, multinomial regression should be included in the toolboxes of all analysts interested in classification. Since multinomial regression is often used in high-dimensional situations, including online advertising, biology, handwriting recognition, and many more, one should be aware of how this will affect the model. Specifically, if one desires to use variable selection to get a more manageable set of covariates, the properties of multinomial regression with respect to variable selection should be examined. The simulation-based experiments in this thesis demonstrated the need to include row-wise grouping in a penalty term to improve both interpretability of the model and performance of the predictions.

As the prominence of Big Data increases and variable selection is becoming increasingly common, one must be aware of the dangers involved in using classical techniques of inference. Unfortunately, many analysts are unaware of (or choose to ignore) the fact that classical techniques for assessing significance cannot be generalized to models chosen via variable selection. Instead, the techniques introduced by [24] and [2] can be used to perform proper post-selection inference. In addition, the methods described in this report can be used to create valid joint confidence intervals for models chosen by the group lasso or multinomial models.

## 8.1 Future Work

In addition to performing variable selection by taking advantage of row-sparse $\boldsymbol{\beta}$ matrices, we can look at the implications of having a column sparse matrix. Having a column of zeros in the $k$-th column implies that the predictor variables cannot help us decide whether to assign the response to the $k$-th or $K$-th class. Therefore we may want to merge classes since we can't decide whether to assign an observation to $k$ or $K$, we may as well create a new larger class which indicates that the response is in either of classes $k$ or $K$. This can be useful when we have too many classes, which makes for a complicated model. For example, this can arise if we want to predict the age interval of a person based on certain covariates.

Originally we may decide to use shorter, more precise intervals, for example 3-year intervals, but if it is too hard for the model to distinguish between age groups it may be best to merge some of them into one. For example, 30-32 and 33-35 may be merged into a new class 30-35. So our model is a little less precise, but potentially much more accurate. This does get complicated pretty quickly. For one thing, if we permute the order of the classes (so that the $K$-th class is no longer the reference class) our matrix will change, and a column of zeros will become two identical columns. So if we were to create an estimator which attempts to perform class-shrinkage, it should look to create not only columns of zeros but also identical columns, since these too can be merged. For this, one can try to create a special form of perhaps the fused lasso [14] and see if it is effective.

This thesis shows the importance of grouping when performing multinomial regression, and uses the elastic net and the adaptive lasso (or adaptive elastic net) to compare the performances of grouped and ungrouped versions. It would be interesting to see if a similar difference would be observed if the grouped and ungrouped SCAD and MCP penalties were implemented and compared.

# Part IV

# Appendix and References

# 9 Full Results

"n" "alpha" "p" "K" "Penalty" "Misc Rate"

50 0.5 7 3 "UEN" 0.283681592039801

50 0.5 7 3 "GEN" 0.296616915422886

50 0.5 7 6 "UEN" 0.489206349206349

50 0.5 7 6 "GEN" 0.456891191709845

50 0.5 7 10 "UEN" 0.624324324324324

50 0.5 7 10 "GEN" 0.6125

50 0.5 7 20 "UEN" 0

50 0.5 7 20 "GEN" 0

50 0.7 15 3 "UEN" 0.337313432835821

50 0.7 15 3 "GEN" 0.38

50 0.7 15 6 "UEN" 0.557704918032787

50 0.7 15 6 "GEN" 0.522139037433155

50 0.7 15 10 "UEN" 0.648695652173913

50 0.7 15 10 "GEN" 0.635185185185185

50 0.7 15 20 "UEN" 0

50 0.7 15 20 "GEN" 0

50 1 50 3 "UEN" 0.364776119402985

50 1 50 3 "GEN" 0.358407960199005

50 1 50 6 "UEN" 0.580648648648649

50 1 50 6 "GEN" 0.585882352941177

50 1 50 10 "UEN" 0.7048

50 1 50 10 "GEN" 0.65625

50 1 50 20 "UEN" 0

50 1 50 20 "GEN" 0

50 1.5 353 3 "UEN" 0.422388059701493

50 1.5 353 3 "GEN" 0.428159203980099

50 1.5 353 6 "UEN" 0.643548387096774

50 1.5 353 6 "GEN" 0.575384615384615

50 1.5 353 10 "UEN" 0.733214285714286

50 1.5 353 10 "GEN" 0.714603174603175

50 1.5 353 20 "UEN" 0

50 1.5 353 20 "GEN" 0

100 0.5 10 3 "UEN" 0.290845771144279

100 0.5 10 3 "GEN" 0.308009950248756

100 0.5 10 6 "UEN" 0.438507462686567

100 0.5 10 6 "GEN" 0.40355

100 0.5 10 10 "UEN" 0.576489361702128

100 0.5 10 10 "GEN" 0.519426751592357

100 0.5 10 20 "UEN" 0.643333333333333

100 0.5 10 20 "GEN" 0.60625

100 0.7 25 3 "UEN" 0.294626865671642

100 0.7 25 3 "GEN" 0.287611940298507

100 0.7 25 6 "UEN" 0.480746268656716

100 0.7 25 6 "GEN" 0.457035175879397

100 0.7 25 10 "UEN" 0.575398773006135

100 0.7 25 10 "GEN" 0.56491124260355

100 0.7 25 20 "UEN" 0.59

100 0.7 25 20 "GEN" 0.59875

100 1 100 3 "UEN" 0.35089552238806

100 1 100 3 "GEN" 0.343880597014925

100 1 100 6 "UEN" 0.540597014925373

100 1 100 6 "GEN" 0.5151

100 1 100 10 "UEN" 0.661666666666667

100 1 100 10 "GEN" 0.59814371257485

100 1 100 20 "UEN" 0.5

100 1 100 20 "GEN" 0.592

100 1.5 1000 3 "UEN" 0.376467661691542

100 1.5 1000 3 "GEN" 0.36

100 1.5 1000 6 "UEN" 0.597213930348259

100 1.5 1000 6 "GEN" 0.536169154228856

100 1.5 1000 10 "UEN" 0.674655172413793

100 1.5 1000 10 "GEN" 0.628524590163934

100 1.5 1000 20 "UEN" 0.654

100 1.5 1000 20 "GEN" 0.5825

200 0.5 14 3 "UEN" 0.232164179104478

200 0.5 14 3 "GEN" 0.276218905472637

200 0.5 14 6 "UEN" 0.411691542288557

200 0.5 14 6 "GEN" 0.390945273631841

200 0.5 14 10 "UEN" 0.511532663316583

200 0.5 14 10 "GEN" 0.498434343434343

200 0.5 14 20 "UEN" 0.675575221238938

200 0.5 14 20 "GEN" 0.653295454545455

200 0.7 40 3 "UEN" 0.274676616915423

200 0.7 40 3 "GEN" 0.266716417910448

200 0.7 40 6 "UEN" 0.431467661691542

200 0.7 40 6 "GEN" 0.413507462686567

200 0.7 40 10 "UEN" 0.564378109452736

200 0.7 40 10 "GEN" 0.534974489795918

200 0.7 40 20 "UEN" 0.68702479338843

200 0.7 40 20 "GEN" 0.65278947368421

200 1 200 3 "UEN" 0.267985074626866

200 1 200 3 "GEN" 0.281965174129353

```
200 1 200 6 "UEN" 0.459502487562189
200 1 200 6 "GEN" 0.412537313432836
200 1 200 10 "UEN" 0.612325
200 1 200 10 "GEN" 0.587213930348259
200 1 200 20 "UEN" 0.718760330578512
200 1 200 20 "GEN" 0.676126126126126
200 1.5 2828 3 "UEN" 0.319925373134328
200 1.5 2828 3 "GEN" 0.322537313432836
200 1.5 2828 6 "UEN" 0.467810945273632
200 1.5 2828 6 "GEN" 0.485199004975124
200 1.5 2828 10 "UEN" 0.64410447761194
200 1.5 2828 10 "GEN" 0.585124378109453
200 1.5 2828 20 "UEN" 0.767103174603175
200 1.5 2828 20 "GEN" 0.689327731092437
```

# 10 R Code

The R code used can be found at `https://www.dropbox.com/sh/khy8zuh322pnhqc/AAAKUIdZs3OqCkF6u-dl=0`

# 11 References


[1] BREHENY, P. and HUANG, J. (2015). Group Descent Algorithms for Nonconvex Penalized Linear and Logistic Regression Models with Grouped Predictors. *Statistics and Computing* **25** 173-187.

[2] BUHLMANN, P., DEZEURE, R., RITOV, Y. and VAN DE GEER, S. (2014). On Asymptotically Optimal Confidence Regions Tests for High-Dimensional Models. *The Annals of Statistics* **42** 1166-1202.

[3] COLE, R. and FANTY, M. (1991). Spoken Letter Recognition. *Advances in Neural Information Processiong Systems*, **3** 220-226. Lippmann, Moody and Touretzky, eds., Morgan Kaufmann, San Mateo.

[4] FAN, F. and LI, R. (2001). Variable Selection via Nonconcave Penalized Likelihood and its Oracle Properties. *Journal of American Statistical Association* **96** 1348-1360.

[5] FAN, J., and LV, J. (2010). A Selective Overview of Variable Selection in High-Dimensional Feature Space. *Statistica Sinica* **20** 101-148.

[6] FRIEDMAN, J., HASTIE, T. and SIMON, N. (2013). A Blockwise Descent Algorithm for Group-Penalized Multiresponse and Multinomial Regression. *arXiv:1311.6529*.

[7] FRIEDMAN, J., HASTIE, T. and TIBSHIRANI, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software* **33** 1-22, `http://www.jstatsoft.org/v33/i01/`.

[8] HASTIE, T., TIBSHIRANI, R. and WAINWRIGHT, M. (2015). *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman and Hall, 36-37.

[9] HASTIE, T. and QIAN, J. (2014). Glmnet Vignette. *Stanford Statistics Technical Report*, `http://www.stanford.edu/~hastie/glmnet/glmnet_alpha.html`.

[10] HASTIE,T. and ZOU, H. (2005). Regularization and Variable Selection via the Elastic Net. *Journal of the Royal Statistical Society Series B* **67** 301-320.

[11] HUANG, J. and WEI, F. (2010). Consistent Group Selection in High-Dimensional Linear Regression. *Bernoulli* **16** 1369-1384.

[12] HUANG, J. and XIE, H. (2007). Asymptotic Oracle Properties of SCAD-Penalized Least Squares Estimators. *Asymptotics: Particles, Processes and Inverse Problems* 149-166.

[13] JIANG, D. (2012). Concave Selection in Generalized Linear Models. *Doctoral Dissertation*, University of Iowa.

[14] KNIGHT, K., ROSSET, S., SAUNDERS, M., TIBSHIRANI, R. and ZHU, J. (2005). Sparsity and Smoothness via the Fused Lasso. *Journal of the Royal Statistical Society Series B* **67** 91-108.

[15] LENG, C. and WANG, H. (2008). A Note on Adaptive Group Lasso. *Computational Statistics & Data Analysis* **12** 5277-5286.

[16] LICHMANN, M. (2013). *UCI Machine Learning Repository* Irvine, CA: University of California, School of Information and Computer Science. `http://archive.ics.uci.edu/ml`.

[17] LIN, Y. and YUAN, M. (2006). Model Selection and Estimation in Regression with Grouped Variables. *Journal of the Royal Statistical Society Series B* **68** 49-67.

[18] McFADDEN, D., TALVITIE, A. and ASSOCIATES (1977). Demand Model Estimation and Validation. *Urban Travel Demand Forecasting Project, Final Report* **5** 222-288. Institute of Transportation Studies, University of California Berkeley.

[19] MOSKE, A. and STARKWEATHER, J. (2011). Multinomial Logistic Regression, `http://www.unt.edu/rss/class/Jon/Benchmarks/MLR_JDS_Aug2011.pdf`.

[20] R Core Team (2013). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. `http://www.R-project.org/`.

[21] RIPLEY, B. and VENABLES, W. (2002). *Modern Applied Statistics with S*, Springer, Fourth Edition, `http://www.stats.ox.ac.uk/pub/MASS4`.

[22] TIBSHIRANI, R. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society Series B* **58** 267-288.

[23] ZHANG, C. (2010). Nearly Unbiased Variable Selection Under Minimax Concave Penalty. *The Annals of Statistics* **38** 894-942.

[24] ZHANG, C. and ZHANG, S. (2014). Confidence Intervals for Low Dimensional Parameters in High Dimensional Models. *Journal of the Royal Statistical Society Series B* **76** 217-242.

[25] ZHANG, H. and ZOU, H. (2009). On the Adaptive Elastic-Net with a Diverging Number of Parameters. *The Annals of Statistics* **37** 1733-1751.

[26] ZOU, H. (2006). The Adaptive Lasso and its Oracle Properties. *Journal of the American Statistical Association* **101** 1418-1429.