# REGRESSION MODELS AND OPTIMIZATION TECHNIQUES FOR ORDINAL, SURVIVAL, AND EXPONENTIAL FAMILY DISTRIBUTIONS

by

Michael J. Wurm

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

Doctor of Philosophy
(Statistics)

at the

UNIVERSITY OF WISCONSIN–MADISON

2017

Date of final oral examination: 07/20/2017

The dissertation is approved by the following members of the Final Oral Committee:
Bret M. Hanlon, Assistant Professor, Department of Statistics
Paul J. Rathouz, Professor and Chair, Department of Biostatistics and Medical Informatics
Alexander M. Tahk, Assistant Professor, Department of Political Science
Sijian Wang, Associate Professor, Department of Statistics and Department of Biostatistics and Medical Informatics
Richard J. Chappell, Professor, Department of Statistics and Department of Biostatistics and Medical Informatics

## ACKNOWLEDGMENTS

To my wife, Jeea: Graduate school has been a time full of rewarding challenges and personal growth, but it turns out that my best reason for attending graduate school was to meet you. We completed this journey together. Thank you for taking it with me and leading the way.

To my parents and grandparents, my brother, and my family: Thank you for supporting all of my pursuits, both personal and professional. You make everything possible.

To my advisor, Bret Hanlon: I have been continually inspired by your passion for statistics and finding new knowledge. It is welcomely contagious, and I hope to carry it with me throughout of my career. The generous time you spent with me in deep conversation helped foster my most creative research ideas. Your constant words of encouragement helped push me through each sticking point. Thank you for being a truly great teacher.

To my advisor, Paul Rathouz: You have shown a genuine interest in my development since day one. I mean that literally because on my first day in graduate school you agreed to mentor my first research project. I'm sure that helping me was often more working than if you had done the project yourself, but that just shows how much you care about the Biostatistics and Medical Informatics Department and the people in it. I will always be thankful for your unselfish willingness to help. Furthermore, I will always be amazed how even after weeks of studying a problem, I could rarely tell you something about statistics you didn't already know.

To my prelim and defense committee members, Professors Alex Tahk, Rick Chappell, Sijian Wang, and Rick Nordheim: You vastly improved my work with many thoughtful suggestions. In many ways, your understanding of my work was better than my own. I have the highest admiration of your expertise.

I am grateful to the National Institutes for Health and the Ruth L. Kirschstein National Research Service Award for helping to fund my education. This grant also gave me the opportunity to work with many wonderful mentors and collaborators. Thank you especially to Professors Paul Rathouz, Heather Johnson, Margie Rosenberg, Jun Zhu, Moo Chung, and Heather Neuman.

To my fellow graduate students and colleagues from the UW–Madison Statistics Department: You have been a terrific support group, and more importantly, the best of friends. Thank you for the lifelong memories. As we go our separate ways, I look forward to seeing what the future holds for each of you.

Contents iii

**1** Introduction 1

**2** Regularized Ordinal Regression and the ordinalNet R Package 3
*2.1* *Introduction* 3
*2.2* *Elementwise link multinomial-ordinal (ELMO) class* 5
*2.3* *Coordinate descent optimization algorithm* 13
*2.4* *Simulation* 23
*2.5* *Method comparison* 25
*2.6* *The ordinalNet R package* 29
*2.7* *Demonstration in R* 32
*2.8* *Discussion* 38

**3** Cox-assisted clustering extensions and optimization 40
*3.1* *Inroduction* 40
*3.2* *CAC assumptions and semiparametric likelihood* 42
*3.3* *EM algorithm optimization procedure* 44
*3.4* *Standard errors and inference* 49
*3.5* *Simulation experiments* 51
*3.6* *Comparison of random starting value methods* 58
*3.7* *Application to TCGA data* 62
*3.8* *Future work* 64
*3.9* *Discussion* 67

**4** Semiparametric Generalized Linear Models and the gldrm R Package 69
*4.1* *Introduction* 69
*4.2* *Optimization algorithm* 71
*4.3* *Inference* 77
*4.4* *Simulation experiments* 78
*4.5* *Computational time and scaling* 86
*4.6* *R package: gldrm* 88

*4.7 R example: iris data* 89

*4.8 Discussion* 93

**A** Appendix to Chapter 2: Regularized Ordinal Regression and the ordinalNet R Package 94

*A.1 Uniqueness of the semi-parallel model estimator* 94

*A.2 The inverse link function and its Jacobian for specific MO families* 97

*A.3 Quadratic approximation to the log-likelihood* 99

*A.4 Statistical inference*100

**B** Appendix to Chapter 4: Semiparametric Generalized Linear Models and the gldrm R Package101

*B.1 Notation*101

*B.2 $\tilde{f}_0$ score function*101

*B.3 $\tilde{f}_0$ information matrix*103

*B.4 $\beta$ score function*104

*B.5 Proof that transformed response with modified link function has equivalent log-likelihood*105

Bibliography106

# 1 INTRODUCTION

This dissertation focuses on regression models for three types of response variables: ordinal, survival, and exponential family. Each chapter is dedicated to one of the three types with a particular concentration on optimization techniques for reliable software. An optimization algorithm is presented for each model with attention to computational efficiency and numerical stability. An R software package has been written for each model which incorporates the optimization techniques discussed in the chapter. Numerical studies were conducted to demonstrate reliability of the software and algorithm. We explore the topics of estimation, statistical inference, and out-of-sample prediction where they are pertinent.

Chapter 2 addresses regularized regression for an ordinal response variable. Regularization techniques such as the lasso (Tibshirani, 1996) and elastic net (Zou and Hastie, 2005) can be used to improve regression model coefficient estimation and prediction accuracy, as well as to perform variable selection. Ordinal regression models are widely used in applications where the use of regularization could be beneficial; however, these models are not included in many popular software packages for regularized regression. We propose a coordinate descent algorithm to fit a broad class of ordinal regression models with an elastic net penalty. Furthermore, we demonstrate that each model in this class generalizes to a more flexible form, for instance to accommodate unordered categorical data. We introduce an elastic net penalty class that applies to both model forms. Additionally, this penalty can be used to shrink a non-ordinal model toward its ordinal counterpart. Finally, we introduce the R package **ordinalNet**, which implements the algorithm for this model class.

Chapter 3 addresses semiparametric finite mixture of regression models for survival data. Finite mixture of regression models are used to model heterogeneous populations, where covariate effects differ by a latent subpopulation classes. Cox-assisted clustering (CAC) is a semiparametric mixture of Cox regression models for survival analysis proposed by Eng and Hanlon (2014). We expand upon this model framework by proposing restrictions on the baseline hazard functions that improve model identifiability. Our simulation results suggest that, under these restrictions, the estimators are asymptotically normal; however, without these restrictions they may have considerable bias and mean squared error. We propose an improved method of generating starting values for the EM algorithm optimization procedure. Finally, we propose a method to calculate standard errors of parameter estimators.

Chapter 4 addresses semiparametric regression for exponential family data. This chapter introduces a new algorithm to estimate and perform inferences on a recently proposed and developed semiparametric generalized linear model (glm). Rather than selecting a particular exponential family model, such as the Poisson distribution, this semiparametric glm assumes that the response is drawn from the more general exponential tilt family. The regression coefficients and unspecified reference distribution are estimated by maximizing a semiparametric likelihood. The new algorithm incorporates several computational stability and efficiency improvements over the algorithm originally proposed. In particular, the new algorithm performs well for either small or large support for the nonparametric response distribution. The algorithm is implemented in a new R package called **gldrm**.

# 2 REGULARIZED ORDINAL REGRESSION AND THE ORDINALNET R PACKAGE

## 2.1 Introduction

Ordinal regression models arise in contexts where the response variable belongs to one of several ordered categories (such as 1="poor", 2="fair", 3="good", 4="excellent"). One of the most common regression models for this type of data is the cumulative logit model (McCullagh, 1980), which is also known as the proportional odds model or the ordinal logistic regression model. Other ordinal regression models include the stopping ratio model, the continuation ratio model, and the adjacent category model. The **VGAM** R package (Yee and Wild, 1996; Yee, 2010, 2015, 2017) fits all of the aforementioned models, but without regularization or variable selection. The SAS CATMOD procedure also fits some of these models (SAS Institute Inc, 2017). Popular CRAN packages for penalized regression, such as **penalized** (Goeman *et al.*, 2017) and **glmnet** (Friedman *et al.*, 2010), do not currently fit ordinal models.

Some algorithms and software already exist for penalized ordinal regression models. The R package **lrm** (Harrell, 2015, 2017) fits the cumulative logit model with quadratic (ridge regression) penalty. The R packages **glmnetcr** (Archer, 2014a) and **glmpathcr** (Archer, 2014b) fit stopping ratio models with the elastic net penalty. Archer (2014a,b) refers to these as continuation ratio models, but we define stopping ratio and continuation ratio models in the same way as Yee (2010).

Archer *et al.* (2014, 2016) also implemented the generalized monotone incremental forward stagewise (GMIFS) algorithm for regularized ordinal regression models in the R package **ordinalgmifs**. This procedure finds a solution path similar to the $L_1$ norm (lasso) penalty. In fact it is the same solution path if the lasso path is monotone for each coefficient, but in other cases the GMIFS and lasso solution paths differ (Hastie *et al.*, 2009). Some drawbacks of this algorithm are that it fits a single solution path and does not have the flexibility of the elastic net mixing parameter (usually denoted by $\alpha$). It can also be computationally expensive because the entire solution path must be fit in small increments, whereas the lasso and elastic net solution path can be obtained only at specified values of the regularization tuning parameter (usually denoted by $\lambda$). A sequence of, say, twenty values may be enough to tune a model by cross validation and will usually be faster than fitting a longer sequence.

To summarize, algorithms for ordinal regression either do not allow regularization, or they apply to specific models. Hence, options are limited for ordinal regression with a large number of predictors. In that context, our contribution to this growing body of software and literature is threefold. First, we propose a general coordinate descent algorithm to fit a rich class of multinomial regression models, both ordinal and non-ordinal, with elastic net penalty.

Second, we define a class of models that (a) can be fit with the elastic net penalty by the aforementioned algorithm, (b) contains some of the most common ordinal regression models, (c) is convenient for modularizing the fitting algorithm, and (d) has both a parallel (ordinal) and a nonparallel form for each model (discussed in the next paragraph). We call this the *elementwise link multinomial-ordinal* (ELMO) class of models. This class is a subset of vector generalized linear models (Yee, 2015). Each model in this class uses a multivariate link function to link multinomial probabilities to a set of linear predictors. The link function can be conveniently written as a composite of two functions. The first determines the model family (e.g. cumulative probability, stopping ratio, continuation ratio, or adjacent category). The second is a standard link function (e.g. logit, probit, or complementary log-log), which is applied elementwise to the result of the first function.

Another feature of the ELMO class is that each model has a form that is appropriate for ordinal response data, as well as a more flexible form that can be applied to either ordinal or unordered categorical responses. We will refer to the first as the *parallel* form and the second as the *nonparallel* form. For the parallel form, the linear predictors of a given observation only differ by their intercept values—the other coefficients are the same. This restriction is what Yee (2010) refers to as the parallelism assumption. The nonparallel form allows all of the coefficients to vary. An example from the ELMO class is the proportional odds model, which is a parallel model that has a nonparallel counterpart, the partial proportional odds model (Peterson *et al.*, 1990). For more details, see Section 2.2.5.

Finally, we propose an elastic net penalty class that applies to both the parallel and nonparallel forms. It can also be used to shrink the nonparallel model toward its parallel counterpart. This can be useful in a situation where one would like to fit an ordinal model but relax the parallelism assumption. This can be achieved by over-parameterizing the nonparallel model to include both the nonparallel and parallel coefficients. We call this alternate parametrization the *semi-parallel* model. Although the regression model itself is not identifiable under this parametrization, the penalized likelihood has a unique optimum (or almost unique, as discussed in Appendix A.1).

We provide an outline for the remainder of the chapter. Section 2.2 defines the ELMO class with specific examples. We also define the parallel, nonparallel, and semi-parallel parametrizations with the elastic net penalty. Section 2.3 provides the proposed algorithm for fitting multinomial regression models with the elastic net penalty. Section 2.4 presents a simulation study to compare prediction accuracy of the penalized parallel, nonparallel, and semi-parallel models. Section 2.5 demonstrates the use of penalized ELMO class models for out-of-sample prediction and variable selection alongside other methods. Section 2.6 provides details about the **ordinalNet** R package (Wurm *et al.*, 2017), which is available on the Comprehensive R Archive Network. Section 2.7 provides a demonstration of the **ordinalNet** package. Section 2.8 contains a summary of the findings and contribution.

## 2.2 Elementwise link multinomial-ordinal (ELMO) class

This section is organized as follows. Section 2.2.1 introduces commonly used notation. Section 2.2.2 is the heart of Section 2.2, defining the ELMO model class. The remaining subsections then discuss particular elements of the ELMO class and issues related to elastic net penalization of this class. Sections 2.2.3 and 2.2.4 provide details for the family function and elementwise link function, respectively. The parallel, nonparallel, and semi-parallel forms are discussed in Sections 2.2.5 and 2.2.6. Section 2.2.7 discusses the elastic net penalty and formulates the objective function under the three model forms. Finally, Section 2.2.8 describes the Jacobian of the inverse link function for the ELMO class.

### 2.2.1 Notation

We introduce commonly used notation. Paper-specific notation is developed throughout the work. For a vector $x$, we use $x^T$ to denote its transpose. We use $\mathbb{1}_K$ to denote the length-$K$ column vector of ones and $I_{K\times K} = I$ to denote the $K \times K$ identity matrix. In both cases, the dependence on $K$ will be suppressed when it is clear from the context. We use $\nabla$ for the gradient operator and $D$ for the Jacobian operator. Consider a vector-valued function $f$ with vector argument $x$. As is standard, the Jacobian of $f$ is defined as

$$Df(x) = \frac{\partial f(x)}{\partial x^T},$$

in other words

$$[Df(x)]_{mn} = \frac{\partial f(x)_m}{\partial x_n}.$$

### 2.2.2 An Introduction to the ELMO model class

We now define the ELMO model class. Models within this class are completely specified by their multivariate link function, which is a composite of two functions. The first function determines the model family (e.g. cumulative probability, stopping ratio, continuation ratio, or adjacent category). We refer to these as *multinomial-ordinal* (MO) families because each has a parallel form specifically for ordinal data, as well as a nonparallel form for any multinomial data, ordinal or unordered. The second function is an *elementwise link* (EL) function, which applies a standard link function on $(0, 1) \to \mathbb{R}$ (e.g. logit, probit, or complementary log-log) elementwise to the result of the first function. McCullagh (1980), Wilhelm *et al.* (1998), and Yee (2010) provide more background on categorical regression models.

Let $\boldsymbol{y}_i|\boldsymbol{x}_i = x_i \overset{\text{indep.}}{\sim} \text{Multinomial}(n_i, p_{i1}, p_{i2}, \ldots, p_{i(K+1)})$ for $i = 1, \ldots, N$. Observations (e.g. subjects or patients) are indexed by $i$, and $N$ is the total number of observations. Here, $x_i$ is an observed vector of covariates (without an intercept), and $\boldsymbol{y}_i = (y_{i1}, \ldots, y_{i(K+1)})^T$ is a random vector of counts summing to $n_i$. The conditional distribution represents $n_i$ independent trials which fall into $K + 1$ classes with probabilities $(p_{i1}, p_{i2}, \ldots, p_{i(K+1)})$ that are a function of $x_i$. The $K + 1$ probabilities sum to one, so they can be parametrized by the vector $p_i = (p_{i1}, p_{i2}, \ldots, p_{iK})^T$.

Let $P$ be the length of $x_i$ and let $B$ be a $P \times K$ matrix of regression coefficients. Let $b_0$ be a vector of $K$ intercept values. The covariates are mapped to a vector of $K$ linear predictors, $\eta_i$, by the relationship $\eta_i = b_0 + B^T x_i$. Class probabilities are linked to the linear predictors by $\eta_i = g(p_i)$, where $g : \mathcal{S}^K \to \mathbb{R}^K$ is a multivariate invertible link function and $\mathcal{S}^K = \{p : p \in (0, 1)^K, \|p\|_1 < 1\}$. Furthermore, $g$ is a composite of two functions, $g_{EL} : (0, 1)^K \to \mathbb{R}^K$ and $g_{MO} : \mathcal{S}^K \to (0, 1)^K$. More specifically, ELMO class models have a link function of the form

$$g(p) = (g_{EL} \circ g_{MO})(p) \ ,$$

where

$$g_{MO}(p) = \delta = (\delta_1, \ldots, \delta_K)^T$$

and

$$g_{EL}(\delta) = (g_{EL*}(\delta_1), \ldots, g_{EL*}(\delta_K))^T \ .$$

### 2.2.3 Family function

The function $g_{MO}$ determines the family of multinomial-ordinal models, such as cumulative probability, stopping ratio, continuation ratio, or adjacent category. In order to belong to a multinomial-ordinal family, the function $g_{MO}$ must be invertible and have the following Monotonicity Property. This Monotonicity Property ensures that all parallel models in the ELMO class are in fact ordinal models (discussed in Section 2.2.5). Examples of MO families are given in Table 2.1.

**Definition (Monotonicity Property)** *For any $p \in \mathcal{S}^K$, define $\gamma_j(p)$ for $j = 1, \dots, K$ as the sum of the first $j$ elements (i.e. cumulative probabilities). Define $\delta_i = (\delta_{i1}, \dots, \delta_{iK})^T = g_{MO}(p_i)$ for $i \in \{1, 2\}$. All MO families have either Property 1 or Property 2 below.*

1. *$\gamma_j(p_1) \leq \gamma_j(p_2)$ for all $j$ if and only if $\delta_{1j} \leq \delta_{2j}$ for all $j$.*
2. *$\gamma_j(p_1) \leq \gamma_j(p_2)$ for all $j$ if and only if $\delta_{1j} \geq \delta_{2j}$ for all $j$.*

| MO Family | $\delta_j$ |
|---|---|
| Cumulative Probability (forward) | $\text{P}(Y \leq j)$ |
| Cumulative Probability (backward) | $\text{P}(Y \geq j+1)$ |
| Stopping Ratio (forward) | $\text{P}(Y = j \mid Y \geq j)$ |
| Stopping Ratio (backward) | $\text{P}(Y = j+1 \mid Y \leq j+1)$ |
| Continuation Ratio (forward) | $\text{P}(Y > j \mid Y \geq j)$ |
| Continuation Ratio (backward) | $\text{P}(Y < j \mid Y \leq j)$ |
| Adjacent Category (forward) | $\text{P}(Y = j+1 \mid j \leq Y \leq j+1)$ |
| Adjacent Category (backward) | $\text{P}(Y = j \mid j \leq Y \leq j+1)$ |

Table 2.1: Examples of multinomial-ordinal (MO) families. For each example, $Y$ is a categorical random variable with class probability vector $(p_1, p_2, \dots, p_K) = g_{MO}^{-1}(\delta_1, \delta_2, \dots, \delta_K)$.

Recall that $p_i = (p_{i1}, p_{i2}, \dots, p_{iK})^T$, and let $r(p_i) = (p_{i(K+1)}, p_{iK}, \dots, p_{i2})$ denote the class probabilities in reverse order, leaving out class 1 instead of class $K+1$. If $g_{MO}$ is a MO function with Property 1, then the $(g_{MO} \circ r)$ is a MO function with Property 2 and vice versa. We can refer to one as the "forward" family and the other as the "backward" family. Although the terms "forward" and "backward" are commonly used in the literature, they do not have a consistent interpretation. We follow the naming conventions used in Yee (2010). By these definitions, the forward cumulative probability and stopping ratio families have Property 1, and the forward continuation ratio and adjacent category families have Property 2.

In addition, if $g_{MO}$ is an MO function, then $g^*_{MO}(p) = 1 - g_{MO}(p)$ is also an MO function. For the cumulative probability and adjacent category families, this is simply a transformation between the forward and backward families. On the other hand, applying this transformation to the forward (backward) stopping ratio family yields the forward (backward) continuation ratio family.

### 2.2.4 Elementwise link function

The elementwise link function $g_{EL*} : (0, 1) \to \mathbb{R}$ must be a monotone, invertible function. It can be almost any link function used for binary data regression, such as logit, probit, or complementary log-log. An important property that some elmentwise link functions satisfy is symmetry, that is

$$g_{EL*}(\delta) = -g_{EL*}(1 - \delta)\ .$$

For example, logit and probit are symmetric, but complementary log-log is not. Under symmetry, the following model pairs are equivalent, with reversed signs on the coefficients: 1) cumulative probability forward and backward models; 2) adjacent category forward and backward models; 3) forward stopping ratio and forward continuation ratio; and 4) backward stopping ratio and backward continuation ratio. However, if $g_{EL*}$ is not symmetric, such as the complementary log-log, then none of these equivalences hold.

### 2.2.5 Parallel and nonparallel forms

Each model in the ELMO class has a parallel form and a nonparallel form. The difference between them is that the parallel form restricts the columns of $B$ to be identical. This restriction is referred to as the parallelism assumption by Yee (2010). Let $b$ denote the common column vector and consider the distribution of $\boldsymbol{y}_i|\boldsymbol{x}_i = x_i$. The Monotonicity Property of $g_{MO}$, along with the monotonicity requirement on $g_{EL*}$, ensures that a change in $b^T x_i$ will shift all cumulative class probabilities in the same direction. This is the defining characteristic of an ordinal regression model.

In contrast, the nonparallel form places no restriction on $B$, and it does not force the cumulative class probabilities to “shift together” in any way. The nonparallel form is appropriate for unordered multinomial data, although it can also be used as a more flexible model for ordinal data.

A word of caution: the nonparallel cumulative probability model must a have linear predictor vector, $B^T x$, that is monotone to ensure that the cumulative probabilities are non-decreasing. For example, $B^T x$ must be monotone increasing for the forward model and monotone decreasing for the backward model. Thus, $B$ should be constrained such that $B^T x$ is monotone for any feasible $x$ in the population of interest. This constraint is difficult to implement in practice, especially because the range of feasible $x$ values may not be known. It is more practical to constrain $B^T x$ to be monotone for all $x$ in the training sample. However, this may lead to non-monotone probabilities for out-of-sample $x$, so it is important to be mindful of this. This is not a concern for the parallel cumulative probability model because the MLE (or penalized MLE) will always have monotone intercepts, and hence monotone linear predictors for all $x$. The other families in Table 2.1 do not have any restriction on the parameter space.

### 2.2.6 Semi-parallel form

In most applications with ordinal response data, domain knowledge does not make it clear whether to use the parallel or nonparallel form. With enough observations, one could simply estimate the nonparallel model by maximum likelihood and obtain a good fit. After all, the nonparallel model includes the parallel model as a special case, and the parallel model will give inconsistent coefficient estimates if it is incorrect.

When the number of predictors is large relative to the number of observations, a regularization method such as lasso or elastic net is required. In this case, it is not possible to estimate each coefficient with a high degree of accuracy—a more realistic modeling goal is to build a model for out-of-sample prediction and determine the most important predictors. In this context, one might forgive some incorrectness of the parallel model if it is accurate enough to accomplish the modeling goals. Even if a nonparallel model were the true data generating mechanism, the regularized parallel model could still outperform the regularized nonparallel model for prediction.

The question becomes: how "parallel" does the data need to be to make the parallel model a better choice? If the response categories have a natural ordering, then it seems prudent to leverage this fact by using an ordinal regression model. However, the fact alone that the response is ordinal does not mean that a parallel regression model will be a good fit. Therefore, it also seems prudent to use a model that is sufficiently flexible to allow deviation from the strict parallelism assumption.

With this motivation, we propose a model that (1) is ordinal in nature and (2) allows deviation from the parallelism assumption. We call this the *semi-parallel* model. Recall that the nonparallel model specifies $\eta_i = b_0 + B^T x_i$, where $b_0$ is a vector of $K$ intercepts and $B$ is an unrestricted $P \times K$ matrix of coefficients. The parallel model restricts the columns of $B$ to be identical and can be parametrized as $\eta_i = b_0 + (b^T x_i) \cdot \mathbb{1}$. The semi-parallel model specifies $\eta_i = b_0 + B^T x_i + (b^T x_i) \cdot \mathbb{1}$. It is the nonparallel model but overparametrized to include both the parallel and nonparallel coefficients. With the elastic net penalty, the penalized likelihood has a unique solution in most cases (see Appendix A.1 for details). We use the term *semi-parallel* because for some covariates, the penalized semi-parallel model fit might only contain the parallel coefficient, with the nonparallel coefficients all set to zero. For other covariates, the fit might contain both parallel and nonparallel coefficients.

### 2.2.7 Elastic net penalty

This section discusses the elastic net penalty for ELMO models. There are many useful resources on regularization, penalized regression, and variable selection, which provide further details in various settings (Bickel *et al.*, 2006; Hesterberg *et al.*, 2008; Hastie *et al.*, 2009; Friedman *et al.*, 2010; Schifano *et al.*, 2010; Vidaurre *et al.*, 2013).

If the sample size is large enough, it may be possible to accurately estimate a regression model by maximum likelihood. However, in many applications the sample size is not large enough to obtain reliable or even unique estimates. In situations like this, it may be advantageous to optimize a penalized version of the log likelihood function. One such penalty is the elastic net (Zou and Hastie, 2005), which is a generalization of both the lasso (Tibshirani, 1996) and ridge regression penalties.

Lasso and ridge regression are techniques that minimize a penalized likelihood objective function, defined as the negative log-likelihood plus a penalty term that is a function of the coefficient vector. For lasso, the penalty is proportional to the $L_1$ norm of the coefficient vector, and for ridge regression it is proportional to the squared $L_2$ norm. Both of these penalties result in a coefficient estimate that is closer to zero than the maximum likelihood estimator, i.e. the estimate is "shrunk" toward zero. This biases the estimates toward zero, but the trade-off is a reduction in variance which often reduces the overall mean squared error. The lasso also has the property that some of the coefficient estimates are shrunk to zero exactly. This provides a natural way to perform variable selection because only the the predictors most associated with the response variable will have nonzero coefficients.

The elastic net penalty is a weighted average between the lasso and ridge regression penalties, and it shares the lasso property of shrinking some coefficients to zero exactly. The weighting parameter, typically denoted by $\alpha$, must be selected or tuned on the data set. The degree of penalization is controlled by another tuning parameter, typically denoted by $\lambda$. Typical practice is to fit the penalized model for a sequence of $\lambda$ values and use a tuning procedure to select the best value (Hesterberg *et al.*, 2008; Hastie *et al.*, 2009; Arlot and Celisse, 2010; Sun *et al.*, 2013). One tuning procedure is to select the model with best fit as determined by $C_p$, AIC, BIC, or another fitness measure. An alternative approach is using cross-validation to select the value that gives the best out-of-sample prediction. We use cross-validation.

Let $b_j$ be the $j^{\text{th}}$ element of $b$ and $B_{jk}$ be the $(j,k)^{\text{th}}$ element of $B$. Let $N_* = \sum_{i=1}^{N} n_i$ be the total number of multinomial trials in the data set. Let $\ell(\cdot)$ denote the log-likelihood function for each ELMO model form. Below, we write the elastic net objective function for each model form.

**Parallel model**

The objective function is

$$\mathcal{M}(b_0, b; \alpha, \lambda) = -\frac{1}{N_*}\ell(b_0, b) + \lambda \sum_{j=1}^{P} \left(\alpha |b_j| + \tfrac{1}{2}(1-\alpha) b_j^2\right) \ .$$

**Nonparallel model**

The objective function is

$$\mathcal{M}(b_0, B; \alpha, \lambda) = -\frac{1}{N_*}\ell(b_0, B) + \lambda \sum_{j=1}^{P}\sum_{k=1}^{K} \left(\alpha |B_{jk}| + \tfrac{1}{2}(1-\alpha) B_{jk}^2\right) \ .$$

**Semi-parallel model**

The objective function is

$$\begin{aligned}\mathcal{M}(b_0, b, B; \alpha, \lambda, \rho) = -\frac{1}{N_*}\ell(b_0, b, B)+ \\ + \lambda\left(\rho\sum_{j=1}^{P}\left(\alpha|b_j| + \tfrac{1}{2}(1-\alpha)b_j^2\right) + \sum_{j=1}^{P}\sum_{k=1}^{K}\left(\alpha|B_{jk}| + \tfrac{1}{2}(1-\alpha)B_{jk}^2\right)\right) .\end{aligned}$$

Here, $\lambda \geq 0$ and $\alpha \in [0, 1]$ are the tuning parameters previously described. Also, $\rho \geq 0$ is a tuning parameter that determines the degree to which the parallel terms are penalized. Fixing $\rho$ at a very large value will set all parallel coefficients to zero, which is equivalent to the nonparallel model. Fixing $\lambda$ at a very large value and choosing $\rho$ such that $\lambda\rho = \lambda^*$ is equivalent to the parallel model with regularization parameter $\lambda^*$. Hence, the semi-parallel model includes both the parallel and nonparallel models as special cases. Fixing $\rho = 0$ will leave the parallel coefficients unpenalized, so the fit will shrink from the maximum likelihood nonparallel model fit toward the maximum likelihood parallel model fit as $\lambda$ increases from zero.

We follow the common convention of scaling the negative log-likelihood by the number of observations (Hesterberg *et al.*, 2008). This way, when fitting a model to a sample from a given population, a given $\lambda$ value will have roughly the same degree of penalization regardless of the sample size. This is convenient when tuning $\lambda$ by cross validation because the tuning data set may have a different sample size than the training data set. We define the sample size as $N_*$ rather than $N$ so the model fit is invariant to whether multinomial trials are grouped into a single observation or split into multiple observations with $n_i = 1$ and identical $x$.

### 2.2.8 Inverse link function Jacobian

The Jacobian of the inverse link function is required for the coordinate descent algorithm. This computation can be compartmentalized for link functions in the ELMO class because of their composite form. Define $h$, $h_{EL}$, $h_{EL*}$, and $h_{MO}$ to be the inverses of $g$, $g_{EL}$, $g_{EL*}$, and $g_{MO}$, respectively. The inverse link function can be written as

$$h(\eta) = h_{MO}(\delta) = h_{MO}(h_{EL}(\eta)) ,$$

where $h_{EL}(\eta) = (h_{EL*}(\eta_1),\ h_{EL*}(\eta_2),\ \ldots,\ h_{EL*}(\eta_K))$.

The Jacobian of the inverse link can be written as

$$Dh(\eta) = Dh_{EL}(\eta)\ Dh_{MO}(p)\ ,$$

where $Dh_{EL}(\eta) = \operatorname{diag}\{h'_{EL*}(\eta_1),\ h'_{EL*}(\eta_2), \ldots,\ h'_{EL*}(\eta_K)\}$.

The inverse and its derivative are well-known for common elementwise link functions, so we do not discuss these any further (see, e.g., the `make.link` function in the R package **stats**). To calculate $h(\eta)$, it only remains to calculate $h_{MO}(\delta)$ and $Dh_{MO}(\delta)$. These calculations are presented for specific MO families in Appendix A.2.

## 2.3 Coordinate descent optimization algorithm

We propose optimizing ELMO class models with the elastic net penalty using a coordinate descent algorithm. Our algorithm mirrors that of Friedman *et al.* (2007, 2010) for generalized linear models. The algorithm is iterative and has an outer and inner loop. The outer loop constructs a quadratic approximation to the log-likelihood—the same quadratic approximation used for iteratively reweighted least squares (IRLS). This approximation is the second order Taylor expansion at the current coefficient estimates. In the spirit of Fisher scoring, the Hessian matrix is replaced by its expectation, the negative Fisher information matrix. This approximation is used as a replacement for the true likelihood in the elastic net objective function, resulting in an expression that can be optimized by coordinate descent. The inner loop cycles through the coefficient estimates, updating each one with the value that marginally optimizes the approximate objective function.

Wilhelm *et al.* (1998) demonstrated the use of IRLS to obtain the maximum likelihood estimates for a broad class of multinomial regression models. This class includes ELMO models but is even more general. This algorithm can easily be applied to any multinomial regression model that links a vector of $K$ probabilities to a vector of $K$ linear combinations of covariates. The link function $g$ does not need to be a composite function or have the Monotonicity Property of the ELMO class. It simply needs to have an inverse $h$. To apply the coordinate descent algorithm to another model, all that is required is to derive the Jacobian of $h$. Although ELMO is a rich class of models, there are multinomial regression models outside this class (e.g. multinomial logistic regression). The coordinate descent algorithm is very general. One could use the basic ideas for fitting an elastic net penalized multinomial regression model that is outside the ELMO class.

The rest of this section is organized as follows. In order to formulate the IRLS quadratic approximation, it is more convenient to parametrize ELMO models with a single coefficient vector instead of $b_0$, $b$, and $B$. Section 2.3.1 discusses this alternative parameterization. Section 2.3.2 discusses the elastic net penalty under the alternative parametrization. In Section 2.3.3 we derive the general form of the score and information matrix for multinomial regression models. Section 2.3.4 discusses the outer loop of the optimization procedure, which updates the quadratic approximation to the log-likelihood. Section 2.3.5 discusses the coordinate descent inner loop. Section 2.3.6 discusses computational efficiency and numerical stability for the coordinate descent updates.

Sections 2.3.7, 2.3.8, and 2.3.9 discuss different aspects of the algorithm specifications. Specifically, Section 2.3.7 presents a method for choosing a sequence of regularization parameter values for the solution path. Section 2.3.8 presents a method for choosing starting coefficient values for the optimization algorithm. And Section 2.3.9 suggests a stopping rule for terminating the algorithm. Section 2.3.10 summarizes the algorithm in outline form. Finally, Section 2.3.11 discusses specific optimization issues that can arise with the cumulative probability model family.

### 2.3.1 ELMO parametrization with a single coefficient vector

Until now, we have written ELMO coefficients in a compact form using an intercept vector $b_0$, a coefficient vector $b$, and a coefficient matrix $B$. For coordinate descent, it is more convenient to write the model with a single coefficient vector $\beta$. To do this, we need to introduce a covariate matrix $X_i$, which is a function of $x_i$. The vector of $K$ linear combinations, $\eta_i$, can then be written as $\eta_i = X_i^T \beta$ for any of the three ELMO model forms.

Let $B_{\cdot k}$ denote the $k^{\text{th}}$ column of $B$. The form of $X_i$ and $\beta$ is given below for the parallel, nonparallel, and semi-parallel models.

**Parallel model**

$$X_i = \left( \begin{array}{c|c} & x_i^T \\ I_{K\times K} & \vdots \\ & x_i^T \end{array} \right)_{K\times(P+K)}, \qquad \beta = \begin{pmatrix} b_0 \\ b \end{pmatrix}_{(P+K)\times 1}.$$

**Nonparallel model**

$$X_i = \left( \begin{array}{c|cccc} & x_i^T & 0 & \cdots & 0 \\ & 0 & x_i^T & \cdots & 0 \\ I_{K\times K} & \vdots & \vdots & \ddots & \vdots \\ & 0 & 0 & \cdots & x_i^T \end{array} \right)_{K\times(PK+K)} , \qquad \beta = \begin{pmatrix} b_0 \\ B_{\cdot 1} \\ B_{\cdot 2} \\ \vdots \\ B_{\cdot K} \end{pmatrix}_{(PK+K)\times 1} .$$

**Semi-parallel model**

$$X_i = \left( \begin{array}{c|ccccc} & x_i^T & x_i^T & 0 & \cdots & 0 \\ & x_i^T & 0 & x_i^T & \cdots & 0 \\ I_{K\times K} & \vdots & \vdots & \vdots & \ddots & \vdots \\ & x_i^T & 0 & 0 & \cdots & x_i^T \end{array} \right)_{K\times(P(K+1)+K)} , \qquad \beta = \begin{pmatrix} b_0 \\ b \\ B_{\cdot 1} \\ B_{\cdot 2} \\ \vdots \\ B_{\cdot K} \end{pmatrix}_{(P[K+1]+K)\times 1} .$$

### 2.3.2 Elastic net penalty

Suppose $\beta$ is length $Q$ and let $\beta_j$ denote the $j^{\text{th}}$ element. We write the elastic net objective function as

$$\mathcal{M}(\beta; \alpha, \lambda, c_1, \ldots, c_Q) = -\frac{1}{N_*}\ell(\beta) + \lambda \sum_{j=1}^{Q} c_j \left( \alpha|\beta_j| + \tfrac{1}{2}(1-\alpha)\beta_j^2 \right) ,$$

where $\lambda > 0$ and $0 \leq \alpha \leq 1$. For ELMO models, $c_j = 0$ for all intercept terms. For the parallel and nonparallel models, $c_j = 1$ for all non-intercept coefficients. For the semi-parallel model, $c_j = \rho$ for the parallel non-intercepts ($b$), and $c_j = 1$ for the nonparallel non-intercepts ($B$). These are not firm rules regarding $c_j$, as there may be situations where one wishes to modify the $c_j$ to accommodate more elaborate penalization schemes. For example, one might wish leave to some covariates unpenalized or to penalize them with varying degrees. The only requirement is that the $c_j$ be nonnegative.

Typically, each covariate is standardized to have its sample standard deviation equal to one so that the scale of a covariate does not affect the degree to which its coefficient is penalized (Hesterberg *et al.*, 2008). However, this is not a requirement.

### 2.3.3 Score and information matrix

In this section, we derive the general form of the score and information matrix for multinomial regression models. The log-likelihood of an observation with probability vector $p_i$ can be written as

$$L_i(p_i) = \sum_{j=1}^{K} y_{ij} \log(p_{ij}) + y_{i(K+1)} \log \left(1 - \sum_{j=1}^{K} p_{ij}\right) .$$

Note that we drop the multinomial term $\log \binom{n_i}{yi1, y_{i2}, \ldots, y_{i(K+1)}}$ because it does not depend on the unknown coefficients, and hence does not affect the model fit. The log-likelihood as a function of $\beta$ is

$$\ell_i(\beta) = L_i(h(X_i^T \beta)) .$$

The score function can be obtained by a chain rule decomposition:

$$U_i(\beta) = X_i^T Dh(\eta_i)^T \nabla L_i(p_i) = X_i^T v_i ,$$

$$\begin{aligned} \text{where } Dh(\eta_i) &= \left(\frac{\partial h_1}{\partial \eta_i}, \ldots, \frac{\partial h_K}{\partial \eta_i}\right) , \\ \nabla L_i(p_i) &= \left(\frac{y_{i1}}{p_{i1}}, \ldots, \frac{y_{iK}}{p_{iK}}\right)^T - \left(\frac{y_{i(K+1)}}{p_{i(K+1)}}\right) \cdot \mathbb{1} \text{ , and} \\ v_i &= Dh(\eta_i)^T \nabla L_i(p_i) . \end{aligned}$$

The Fisher information matrix is given by

$$\mathcal{I}_i(\beta) = E_\beta\{U_i(\beta) U_i(\beta)^T\} = X_i^T W_i X_i ,$$

where $W_i = Dh(\eta_i)^T \Sigma_i^{-1} Dh(\eta_i)$ and

$$\begin{aligned} \Sigma_i^{-1} &= E_\beta \left\{\nabla L_i(p_i) \nabla L_i(p_i)^T\right\} \\ &= \left[E_\beta \left\{ \left(\frac{y_{im}}{p_{im}} - \frac{y_{i(K+1)}}{p_{i(K+1)}}\right) \left(\frac{y_{in}}{p_{in}} - \frac{y_{i(K+1)}}{p_{i(K+1)}}\right) \right\}\right]_{mn} \\ &= \left[n_i \left(\frac{I(m=n)}{p_{im}} + \frac{1}{p_{i(K+1)}}\right)\right]_{mn} \\ &= n_i \left([\text{diag}(p_i)]^{-1} + \frac{1}{p_{i(K+1)}} \cdot \mathbb{1}\mathbb{1}^T\right) . \end{aligned}$$

Because the $y_i$ are independent, the full data log-likelihood, score, and information are defined as

$$\begin{aligned}
\ell(\beta) &= \sum_{i=1}^{N} \ell_i(\beta) \,, \\
U(\beta) &= \sum_{i=1}^{N} U_i(\beta) = X^T v \text{ , and} \\
\mathcal{I}(\beta) &= \sum_{i=1}^{N} \mathcal{I}_i(\beta) = X^T W X \,,
\end{aligned}$$

$$\text{where } X = \begin{pmatrix} X_1 \\ \vdots \\ X_N \end{pmatrix} , \;\; v = \begin{pmatrix} v_1 \\ \vdots \\ v_N \end{pmatrix} , \;\; \text{and } \; W = \begin{pmatrix} W_1 & 0 & \cdots & 0 \\ 0 & W_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & W_N \end{pmatrix} .$$

### 2.3.4 Optimization outer loop (quadratic approximation)

The optimization outer loop updates the quadratic approximation to the log-likelihood using a Taylor expansion around the current coefficient estimates. Let $\hat{\beta}^{(r)}$ denote the coefficient estimates after the $r^{\text{th}}$ outer loop iteration, and let the $(r)$ superscript denote terms that depend on $\hat{\beta}^{(r)}$. Of course, starting values are required for the first iteration, and this topic will be discussed later.

We define the quadratic approximation of $\ell(\beta)$ as

$$\ell^{(r)}(\beta) = -\tfrac{1}{2}(z^{(r)} - X\beta)^T \, W^{(r)} \, (z^{(r)} - X\beta) \,,$$

where $z^{(r)} = X\hat{\beta}^{(r)} + \{W^{(r)}\}^{-1} v^{(r)}$. This is the second order Taylor series expansion of $\ell$ at $\hat{\beta}^{(r)}$, up to an additive constant that does not depend on $\beta$. Also, the Hessian is replaced by its expectation, $-\mathcal{I}(\hat{\beta}^{(r)})$. The derivation is provided in Appendix A.3. The inner loop computes $\hat{\beta}^{(r+1)}$ by optimizing the *penalized* quadratic approximation.

### 2.3.5 Optimization inner loop (coordinate descent)

For unpenalized maximum likelihood estimation, the quadratic approximation is maximized by the usual weighted least squares solution $\hat{\beta}^{(r+1)} = (X^T W^{(r)} X)^{-1} X^T W^{(r)} z^{(r)}$. With the elastic net penalty, we can still follow the IRLS procedure, but the optimization step no longer has a closed form solution. This is because partial derivatives of the elastic net penalty do not exist at zero for any of the penalized coefficients. The optimization step can instead be done with a coordinate descent procedure. This involves cycling through the coefficient estimates, updating each one with the value that marginally optimizes the approximate objective function. The cycle is iterated until convergence.

Let $\mathcal{M}^{(r)}(\beta)$ denote the elastic net objective function with $\ell^{(r)}(\beta)$ in place of $\ell(\beta)$. Let $\hat{\beta}_j^{(r,s)}$ denote the estimate of $\beta_j$ at the $s^{\text{th}}$ inner loop iteration of the $r^{\text{th}}$ outer loop iteration. Let $\mathcal{M}_j^{(r,s)}(t) = \mathcal{M}^{(r)}(\hat{\beta}_1^{(r,s+1)}, \ldots, \hat{\beta}_{j-1}^{(r,s+1)}, t, \hat{\beta}_{j+1}^{(r,s)}, \ldots, \hat{\beta}_Q^{(r,s)})$ denote $\mathcal{M}^{(r)}$ as a marginal function of the $j^{\text{th}}$ regression coefficient only, with all other coefficients fixed at their current estimates. The $s^{\text{th}}$ iteration coordinate descent update of the $j^{\text{th}}$ coefficient is $\arg\min \mathcal{M}_j^{(r,s)}(t)$. If $c_j = 0$ (i.e. $\beta_j$ is unpenalized), then this can be solved straightforwardly by setting $\frac{d}{dt}\mathcal{M}_j^{(r,s)}(t) = 0$. In general, for $t \neq 0$,

$$\frac{d}{dt}\mathcal{M}_j^{(r,s)}(t) = -\frac{1}{N_*} X_{\cdot j}^T W^{(r)} \left( z^{(r)} - X_{\cdot -j} \hat{\beta}_{-j}^{(r,s)} - t X_{\cdot j} \right) + \lambda \left( \alpha \cdot \text{sign}(t) + (1-\alpha) \cdot t \right) ,$$

where $X_{\cdot j}$ denotes the $j^{\text{th}}$ column of $X$, $X_{\cdot -j}$ denotes $X$ with the $j^{\text{th}}$ column deleted, and $\hat{\beta}_{-j}^{(r,s)} = (\hat{\beta}_1^{(r,s+1)}, \ldots, \hat{\beta}_{j-1}^{(r,s+1)}, \hat{\beta}_{j+1}^{(r,s)}, \ldots, \hat{\beta}_Q^{(r,s)})$. $\mathcal{M}_j^{(r,s)}(t)$ is convex because $\frac{d}{dt}\mathcal{M}_j^{(r,s)}(t)$ runs from $-\infty$ to $+\infty$ and is monotonically increasing. The only point where the derivative does not exist is at $t = 0$, where it jumps up by $2\lambda\alpha c_j$. If $\left| \frac{1}{N_*} X_{\cdot j}^T W^{(r)} \left( z^{(r)} - X_{\cdot -j} \hat{\beta}_{-j} \right) \right| > \lambda\alpha c_j$, then the derivative attains zero, and the value at which this occurs is $\arg\min \mathcal{M}_j^{(r,s)}(t)$. Otherwise the derivative changes sign at $t = 0$, and $\arg\min \mathcal{M}_j^{(r,s)}(t) = 0$. Hence, the coordinate descent update can be written as

$$\hat{\beta}_j^{(r,s+1)} = \frac{S\left( \frac{1}{N_*} X_{\cdot j}^T W^{(r)} \left( z^{(r)} - X_{\cdot -j} \hat{\beta}_{-j}^{(r,s)} \right) ,\ \lambda\alpha c_j \right)}{\frac{1}{N_*} X_{\cdot j}^T W^{(r)} X_{\cdot j} + \lambda(1-\alpha) c_j} ,$$

where $S(x, y) = \text{sign}(x)(|x| - y)_+$ is the soft-thresholding operator, as defined in Friedman *et al.* (2010).

On a side note, in some situations one may wish to include a model constraint that forces some or all of the coefficients to be nonnegative (e.g. to implement the nonnegative garrote penalty discussed in Breiman (1995)). With this constraint, the coordinate descent update becomes $\hat{\beta}_j^{(r,s+1)} = \underset{t \geq 0}{\arg\min} \mathcal{M}_j^{(r,s)}(t)$, where the only difference is the restriction that $t \geq 0$. When $\arg\min \mathcal{M}_j^{(r,s)}(t) < 0$, then $\underset{t \geq 0}{\arg\min} \mathcal{M}_j^{(r,s)}(t) = 0$ because $\mathcal{M}_j^{(r,s)}(t)$ is convex. The coordinate descent update can be written as

$$\hat{\beta}_j^{(r,s+1)} = \frac{\left(\frac{1}{N_*} X_{\cdot j}^T W^{(r)} \left(z^{(r)} - X_{\cdot -j} \hat{\beta}_{-j}^{(r,s)}\right) - \lambda \alpha c_j\right)_+}{\frac{1}{N_*} X_{\cdot j}^T W^{(r)} X_{\cdot j} + \lambda(1-\alpha) c_j} .$$

### 2.3.6 Computational efficiency and numerical stability

It would be computationally inefficient to compute the coordinate descent coefficient updates in the form presented in Section 2.3.5. First of all, $W^{(r)}$ is a sparse block diagonal matrix and should not be multiplied without taking this into consideration. Secondly, it is not necessary to explicitly compute $z^{(r)}$, which depends on $W_1^{-1}$, $W_2^{-1}$, …, $W_N^{-1}$. It is not necessarily computationally expensive to compute these $K \times K$ matrix inverses when $K$ is small, but it is still better to avoid doing so. For computational efficiency, it is better to write the coordinate descent updates in terms of the score and information matrix. This is done by making the following two substitutions in (2.3.5)

$$\begin{aligned} X_{\cdot j}^T W^{(r)} \left(z^{(r)} - X_{\cdot -j} \hat{\beta}_{-j}^{(r,s)}\right) &= \left[U(\hat{\beta}^{(r)})\right]_j + \left[\mathcal{I}(\hat{\beta}^{(r)})\hat{\beta}^{(r)}\right]_j - \left[\mathcal{I}(\hat{\beta}^{(r)})\hat{\beta}^{(r,s)}\right]_j , \\ X_{\cdot j}^T W^{(r)} X_{\cdot j} &= \left[\mathcal{I}(\hat{\beta}^{(r)})\right]_{jj} . \end{aligned}$$

The subscripts on terms with square brackets indicate the $j^{\text{th}}$ vector element or matrix diagonal. Only the third term on the right hand side of the first substitution needs to be updated with each inner loop iteration because it depends on $\hat{\beta}^{(r,s)}$. The other terms are a function of $\hat{\beta}^{(r)}$, so they only need to be updated during the outer loop.

Furthermore, the $X_i$ matrices are sparse for ELMO models. In some programming languages, it may be advantageous to use this block structure to compute $X_i\hat{\beta}$ and $X_i^T W_i X_i$. This has the additional advantage that it is not necessary to store $X_1$, $X_2$, …, $X_N$ in memory, but rather just $x_1$, $x_2$, …, $x_N$, which are smaller.

Numerical instability of the information matrix can arise when the fitted class probabilities approach zero for any observation. This is problematic even if the near-zero probability occurs for an observation $i$ and class $j$ such that $y_{ij} = 0$. A way to prevent numerical instability is to cap the fitted probabilities at some minimum value just for the information matrix calculation. (The `pMin` argument sets this threshold for the optimization function `ordinalNet` in the **ordinalNet** R package.) The score and likelihood are computed with uncapped fitted probabilities.

### 2.3.7 Regularization parameter sequence

Often we are interested in computing solutions for a sequence of $\lambda$ values, rather than a single value. For $\alpha > 0$, there always exists a threshold value $\lambda_{\max}$ where the first coefficient enters the solution path. All penalized coefficients are set to zero for any $\lambda > \lambda_{\max}$. An off-the-shelf method to generate a reasonable sequence of $\lambda$ values is to let $\lambda_{\min} = 0.01 \times \lambda_{\max}$ and consider a sequence of $\lambda$ values that is uniform between $\lambda_{\max}$ and $\lambda_{\min}$ on the log scale (Friedman *et al.*, 2010).

To calculate $\lambda_{\max}$, we first fit the intercept-only model by unpenalized maximum likelihood. Also include any unpenalized non-intercept coefficients if there are any. We then calculate the quadratic approximation at this solution. Each penalized coefficient has a threshold value of $\lambda$ where its coordinate descent update becomes nonzero. The minimum threshold value among all coefficients is $\lambda_{\max}$, as this is the value where the first coefficient enters the solution path. Specifically,

$$\lambda_{\max} = \min_j \frac{1}{N_* \alpha c_j} \left| X_{\cdot j}^T W \left( z - X_{\cdot -j} \hat{\beta}_{-j} \right) \right| ,$$

where $\hat{\beta}$ is the intercept-only maximum likelihood estimate, and $W$ and $z$ are calculated at $\hat{\beta}$.

### 2.3.8 Starting values

Park and Hastie (2007) proposed an efficient estimation procedure for a decreasing sequence of $\lambda$ values. The $\hat{\beta}$ solution for each $\lambda$ value is used as the starting value for the next $\lambda$ in the sequence. This technique is known as "warm starts." Furthermore, it is not necessary to update all coefficient estimates during the coordinate descent inner loop. Many coefficient estimates will begin and remain at zero throughout the inner loop, especially for larger $\lambda$ values. It is more efficient to cycle through only the coefficients that had nonzero estimates at the previous step. The set of nonzero coefficients is known as the "active set." After the coordinate descent inner loop converges, one pass can be made over each coefficient outside the active set. If the coordinate descent update is zero for all of them, then the optimal solution has been reached. If the final pass changes any coefficient estimate from zero to a nonzero value, then the coordinate descent loop should be continued including these new nonzero coefficients in the active set.

A reasonable set of starting values can usually be obtained by passing the observed response category frequencies into the link function—this provides intercept starting values, and all other coefficients can start at zero. This is also the solution corresponding to $\lambda_{\text{max}}$ if there are no unpenalized non-intercept coefficients. If the first $\lambda$ value is not $\lambda_{\text{max}}$ or some of the non-intercepts are unpenalized, then this is still usually a reasonable set of starting values for the first $\lambda$ value.

### 2.3.9 Stopping rule

The coordinate descent procedure has an inner and outer loop, both of which require convergence definitions and thresholds. A suggestion is to define convergence using the relative change in the elastic net objective. For the outer loop, the definition is $\left|\frac{\mathcal{M}(\hat{\beta}^{(r)})-\mathcal{M}(\hat{\beta}^{(r-1)})}{\mathcal{M}(\hat{\beta}^{(r-1)})}\right| < \epsilon_{\text{out}}$. For the inner loop, the quadratic approximation to the log-likelihood should be used instead of the true log-likelihood, so the definition is $\left|\frac{\mathcal{M}^{(r)}(\hat{\beta}^{(r,s)})-\mathcal{M}^{(r)}(\hat{\beta}^{(r,s-1)})}{\mathcal{M}^{(r)}(\hat{\beta}^{(r,s-1)})}\right| < \epsilon_{\text{in}}$. A small constant can also be added to the denominator to allow convergence when the log-likelihood is near zero. Based on some trial and error, it seems efficient to set the outer and inner convergence thresholds, $\epsilon_{\text{out}}$ and $\epsilon_{\text{in}}$, to the same value.

### 2.3.10 Algorithm summary

1. Fit the intercept-only model by maximum likelihood. (Also include any unpenalized non-intercept coefficients if there are any.)

2. Calculate $\lambda_{\text{max}}$ and choose a decreasing sequence $\lambda_{\text{max}} = \lambda_1, \lambda_2, \ldots, \lambda_M = \lambda_{\text{min}}$.

3. Set $\hat{\beta}[\lambda_1]$ equal the solution of the intercept-only model.

4. For $m = 2$ to $M$:

   a) Set $r \leftarrow 0$ and $\hat{\beta}^{(0)} \leftarrow \hat{\beta}[\lambda_{m-1}]$.

   b) While $\left|\frac{\mathcal{M}(\hat{\beta}^{(r)})-\mathcal{M}(\hat{\beta}^{(r-1)})}{\mathcal{M}(\hat{\beta}^{(r-1)})}\right| > \epsilon_{\text{out}}$:

```
        i. Calculate $U(\hat{\beta}^{(r)})$ and $\mathcal{I}(\hat{\beta}^{(r)})$.
       ii. Set $s \leftarrow 0$ and $\hat{\beta}^{(r,0)} \leftarrow \hat{\beta}^{(r)}$.
      iii. While $\left|\frac{\mathcal{M}^{(r)}(\hat{\beta}^{(r,s)})-\mathcal{M}^{(r)}(\hat{\beta}^{(r,s-1)})}{\mathcal{M}^{(r)}(\hat{\beta}^{(r,s-1)})}\right| > \epsilon_{\text{in}}$:
            A. Calculate $\hat{\beta}^{(r,s+1)}$ with a single cycle of coordinate descent updates over
               the coefficient active set.
            B. Set $s \leftarrow s+1$.
       iv. Do one loop of coordinate descent updates over coefficients outside the
           active set. If any coefficient estimate changes to a nonzero value, then repeat
           the previous loop with the new nonzero coefficients in the active set.
        v. Set $\hat{\beta}^{(r+1)} \leftarrow \hat{\beta}^{(r,s)}$.
       vi. Set $r \leftarrow r+1$.
    c) Set $\hat{\beta}[\lambda_m] \leftarrow \hat{\beta}^{(r)}$.
```

### 2.3.11 Issues with the cumulative probability family

The cumulative probability family has a constrained parameter space because the cumulative probabilities must be monotone for every $x$ in the population. It was discussed in Section 2.2.5 that if the constraint is only enforced for $x$ in the training sample, then difficulties may arise because an out-of-sample $x$ may have fitted probabilities that are not monotone.

The parameter space constraint can also create difficulties during optimization. Although the likelihood is undefined outside the constrained parameter space, we could define an improper likelihood on the unrestricted parameter space. This improper likelihood would allow observations to have fitted probabilities greater than zero or less than one, and it would be defined as zero anywhere that an observation has negative probability in a class that was observed. This is essentially the likelihood that coordinate descent algorithm is designed to optimize. As a result, the algorithm may seek a solution that lies outside the constrained parameter space.

When the solution path leaves the constrained parameter space, we simply terminate the optimization algorithm at the $\lambda$ value where this occurred. Further work is required to devise a constrained optimization procedure for the cumulative probability family.

## 2.4 Simulation

We have discussed three penalized ELMO model forms for ordinal data: the parallel model, the nonparallel model, and the semi-parallel model. The purpose of the following simulation experiments is to show scenarios where each of these three model types yields better out-of-sample prediction accuracy than the others.

We conducted three simulation experiments, each based on 100 replicates, i.e. simulated datasets. For each replicate, the data were simulated from a forward stopping ratio model with the parameters shown in Table 2.2. In all of the experiments, the covariates were simulated as independent, standard normal random variables.

Now consider the estimation procedures. For each of the three models, parallel, nonparallel, and semi-parallel, the elastic net tuning parameter was set to $\alpha = 1$ (i.e. lasso penalty). For the semi-parallel model, the tuning parameter $\rho$ was set to one. A $\lambda$ sequence of twenty values was generated uniformly on the log scale between $\lambda_{\max}$ of the full training data and $0.01 \times \lambda_{\max}$. For each simulated dataset, five-fold cross validation was used to select the optimal tuning parameter.

Simulation results are shown in Table 2.3. For a given replicate, out-of-sample prediction accuracy was evaluated as the average log-likelihood of 10,000 observations generated from the same distribution as the training set. The validation data set was chosen large enough that the within-replicate standard error of the mean was negligible relative to the across-replicate standard error. More precisely, the results were produced in the following way. For a given replicate, generate 10,000 observations from the same distribution as the training set. Fix replicate $i$, for these 10,000 observations, compute log-likelihoods $x_{i,j}$, $j = 1, ..., 10000$ and denote the sample average as $v_i$,

$$v_i = \frac{1}{10000} \sum_{j=1}^{10000} x_{i,j}.$$

Table 2.3 reports the sample average of the $v_i$'s (across the $100$ simulated datasets), $\bar{v} = \frac{1}{100} \sum_{i=1}^{100} v_i$ and the standard error of $\bar{v}$.

For Simulation 1, the data generating model is nonparallel, and the sample size is relatively large. This results in the nonparallel and semi-parallel models having better fits than the parallel model. For Simulation 2, the data generating mechanism is parallel, and only one in three covariates has a nonzero effect. This results in the parallel and

| | $N$ | $b_0$ | $B$ |
|---|---|---|---|
| Sim 1 | 500 | $\begin{pmatrix} -0.5 \\ 0 \end{pmatrix}$ | $\begin{pmatrix} 0 & 2 \end{pmatrix}$ |
| Sim 2 | 50 | $\begin{pmatrix} -0.5 \\ 0 \end{pmatrix}$ | $\begin{pmatrix} \left.\begin{matrix} 2 & 2 \\ \vdots & \vdots \\ 2 & 2 \end{matrix}\right\} \times 5 \\ \left.\begin{matrix} 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{matrix}\right\} \times 10 \end{pmatrix}$ |
| Sim 3 | 50 | $\begin{pmatrix} -0.5 \\ 0 \end{pmatrix}$ | $\begin{pmatrix} -2 & 2 \\ \left.\begin{matrix} 2 & 2 \\ \vdots & \vdots \\ 2 & 2 \end{matrix}\right\} \times 4 \\ \left.\begin{matrix} 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{matrix}\right\} \times 10 \end{pmatrix}$ |

Table 2.2: Training data parameters for the three simulation experiments. For each experiment, the data generating mechanism was a forward stopping ratio model.

semi-parallel models having better fits than the nonparallel model. Simulation 3 is similar to Simulation 2, but the first covariate has a highly nonparallel effect. This results in the semi-parallel model having a better fit than both the parallel and nonparallel models.

Note that in every simulation scenario, the semi-parallel model fit is nearly as good, if not better, than both the parallel and nonparallel model fits. This is not to say that the semi-parallel model should always be preferred, but it is evidence that it is a highly versatile model. In practice, cross validation could be used to determine which of the three methods performs the best, using out-of-sample log-likelihood, or another measure of fit. In addition, it could be worthwhile to use cross validation to compare different values of $\rho$ for the semi-parallel fit.

| | Parallel | Nonparallel | Semi-parallel |
|---|---|---|---|
| Sim 1 | -1.05 (0.00045) | -0.95 (0.00052) | -0.95 (0.00052) |
| Sim 2 | -0.59 (0.0089) | -0.71 (0.0073) | -0.62 (0.0077) |
| Sim 3 | -0.74 (0.008) | -0.71 (0.010) | -0.64 (0.011) |

Table 2.3: Results for the simulation experiment. For a given replicate, out-of-sample prediction accuracy was evaluated as the average log-likelihood of 10,000 observations generated from the same distribution as the training set. The values reported are the sample average (standard error) of these values across the 100 simulation replicates.

## 2.5 Method comparison

We now demonstrate ELMO class models alongside other methods for out-of-sample prediction and variable selection. We use a dataset from a cancer genomics example presented by Archer *et al.* (2014). The data come from the Gene Expression Omnibus GSE18081. The CRAN package **ordinalgmifs** contains a filtered version of this dataset called *hccmethyl*. It contains expression levels of 46 genes from 56 human subjects. The measurements come from liver tissue samples assayed using the Illumina GoldenGat Methylation BeadArray Cancer Panel I (Archer *et al.*, 2010). Twenty subjects have a normal liver, 16 have cirrhosis (disease), and 20 have hepatocellular carcinoma (severe disease). These categories have a natural ordering according to disease severity.

The analysis goal was to use gene expression values to predict liver disease—more specifically, to estimate the probability that each subject's liver would be classified as healthy, diseased, or severely diseased. The number of predictors is large relative to the sample size, making regularization and/or variable selection imperative. We use five-fold cross validation to compare the out-of-sample prediction performance of various methods. We use out-of-sample log-likelihood and misclassification rate as performance criteria. A total of seven methods were compared. We summarize them below.

1. Cumulative logit models with lasso penalty.
    - Three models: parallel, nonparallel, and semi-parallel.
    - $\lambda$ was tuned by five-fold cross validation within each cross validation fold, selecting the value with the best average out-of-sample log-likelihood.

- $\lambda$ was selected from the same sequence within each fold. $\lambda_{\max}$ was calculated from the full data, and a sequence of twenty values was generated uniformly on the logarithmic scale from $\lambda_{\max}$ to $\lambda_{\max}/100$.
- Fit by **ordinalNet**.

2. Multinomial logistic regression models with lasso penalty.
   - Two models: standard lasso and group lasso.
   - $\lambda$ was tuned by five-fold cross validation within each cross validation fold, selecting the value with the best average out-of-sample log-likelihood.
   - $\lambda$ was selected from the same sequence within each fold. $\lambda_{\max}$ was calculated from the full data, and a sequence of twenty values was generated uniformly on the logarithmic scale from $\lambda_{\max}$ to $\lambda_{\max}/100$.
   - Fit by **glmnet**.
3. Cumulative logit model with GMIFS solution path.
   - The solution path was fit with step size $0.01$.
   - AIC was used for tuning within each fold. Specifically, the model with the best AIC was selected from all of the models in the solution path.
   - Fit by **ordinalgmifs**.
4. Cumulative logit model with AIC, forward stepwise variable selection.
   - Fit by **MASS** function `polr` (Venables and Ripley, 2002).

Results for the experiment are summarized in Table 2.4. The GMIFS method performed the best according to both performance criteria. The parallel cumulative logit lasso, the semi-parallel cumulative logit lasso, and the multinomial logistic regression with (standard) lasso performed comparably to the GMIFS method. The other methods did not perform well.

Poor performance of the nonparallel, cumulative logit lasso model can be attributed to the parameter space restriction discussed in Sections 2.2.5 and 2.3.11. With this dataset, it is easy for the nonparallel model to predict non-monotone cumulative probabilities for out-of-sample observations. However, the largest $\lambda$ value in the sequence corresponds to the null model, which cannot have non-monotone cumulative probabilities. As a result, the cross validation tuning procedure tends to select the null model or a highly penalized model; neither model is good for prediction.

The AIC stepwise method performed poorly in terms of out-of-sample log-likelihood. This poor performance is likely due to overfitting. More specifically, this method occasionally estimates a very small probability in the observed class of an out-of-sample observation. The out-of-sample misclassification rate is more reasonable than the out-of-sample log-likelihood, but other methods perform better.

| | Log-likelihood | Misclassification rate |
|---|---|---|
| Cumulative logit lasso - parallel | -2.37 (0.60) | 0.091 (0.050) |
| Cumulative logit lasso - nonparallel | -10.92 (0.70) | 0.373 (0.036) |
| Cumulative logit lasso - semi-parallel | -2.47 (0.52) | 0.091 (0.050) |
| Multinomial logistic lasso | -2.84 (0.41) | 0.108 (0.019) |
| Multinomial logistic group lasso | -6.56 (0.49) | 0.158 (0.047) |
| Cumulative logit GMIFS | -1.76 (0.58) | 0.073 (0.034) |
| Cumulative logit AIC forward stepwise | -12.87 (4.03) | 0.162 (0.035) |

Table 2.4: Out-of-sample log-likelihood and misclassification rates for the method comparison based on the liver disease dataset (GSE18081). The value reported is the mean (standard error) across five cross validation folds.

In addition to class prediction, penalized regression can be used for variable selection, i.e. to determine which of the 46 genes are most associated with liver disease. Table 2.5 shows which genes were selected by each method. All models were tuned on the full data the same way that they were trained in the cross validation study.

| | CLP | CLN | CLS | ML | MLG | GMIFS | AIC |
|---|---|---|---|---|---|---|---|
| CDKN2B_seq_50_S294_F | -14.74 | · | 2 | 1 | 3 | -1.30 | · |
| DDIT3_P1313_R | -8.88 | · | 1 | · | · | -1.29 | -148.94 |
| ERN1_P809_R | 1.20 | · | 1 | · | · | 0.36 | · |
| GML_E144_F | 7.84 | · | 1 | 1 | 3 | 1.92 | · |
| HDAC9_P137_R | · | · | · | · | · | 0.08 | · |
| HLA-DPA1_P205_R | · | · | · | · | · | 0.36 | · |
| HOXB2_P488_R | · | · | · | · | 3 | -0.08 | -171.45 |
| IL16_P226_F | 14.43 | · | 1 | 1 | 3 | 1.82 | · |
| IL16_P93_R | 2.25 | · | 1 | · | · | 0.34 | 77.05 |
| IL8_P83_F | 1.36 | · | 2 | 1 | 3 | 0.38 | · |
| MPO_E302_R | 11.88 | · | 1 | 1 | 3 | 0.72 | 188.97 |
| MPO_P883_R | · | · | 1 | 1 | · | 0.17 | · |
| PADI4_P1158_R | -4.40 | · | 1 | 1 | 3 | -0.96 | · |
| SOX17_P287_R | -9.41 | · | 1 | 2 | 3 | -1.94 | · |
| TJP2_P518_F | -24.62 | · | 1 | 1 | 3 | -2.05 | -71.70 |
| WRN_E57_F | 5.75 | · | 1 | · | · | 0.54 | 145.49 |
| CRIP1_P874_R | · | 1 | · | 1 | 3 | · | · |
| SLC22A3_P634_F | · | 1 | 1 | 2 | 3 | · | · |
| CCNA1_P216_F | · | · | · | · | · | · | · |
| SEPT9_P374_F | · | · | · | · | · | · | · |
| ITGA2_E120_F | · | · | · | · | · | · | · |
| ITGA6_P718_R | · | · | · | · | · | · | · |
| HGF_P1293_R | · | · | · | · | · | · | · |
| DLG3_E340_F | · | · | · | · | · | · | · |
| APP_E8_F | · | · | · | · | · | · | · |
| SFTPB_P689_R | 5.05 | · | 1 | 1 | 3 | 0.31 | · |
| PENK_P447_R | · | · | · | · | · | · | · |
| COMT_E401_F | 3.74 | · | 1 | · | 3 | 0.59 | · |
| NOTCH1_E452_R | · | · | · | · | · | · | · |
| EPHA8_P456_R | · | · | · | · | · | · | · |
| WT1_P853_F | · | · | · | · | · | · | · |
| KLK10_P268_R | · | · | · | 1 | 3 | · | · |
| PCDH1_P264_F | 0.08 | · | 1 | 1 | 3 | · | · |
| TDGF1_P428_R | · | · | · | · | · | · | · |
| EFNB3_P442_R | · | · | · | · | · | · | -137.05 |
| MMP19_P306_F | · | · | · | · | · | · | · |
| FGFR2_P460_R | · | · | · | 1 | 3 | · | · |
| RAF1_P330_F | · | · | · | · | · | · | · |
| BMPR2_E435_F | · | · | · | 1 | 3 | · | · |
| GRB10_P496_R | · | · | · | · | · | · | · |
| CTSH_P238_F | · | · | · | · | · | · | · |
| SLC6A8_seq_28_S227_F | · | · | · | · | · | · | · |
| PLXDC1_P236_F | · | · | · | · | · | · | · |
| TFE3_P421_F | · | · | · | · | · | · | · |
| TSG101_P139_R | · | · | · | · | · | · | · |

Table 2.5: Regression coefficient estimates for the method comparison based on the liver disease dataset (GSE18081). The column headers are abbreviations for the methods listed in the same order as Table 2.4. For the methods with multiple coefficients per predictor, the number of nonzero coefficient estimates is reported instead of the coefficient value. The nonparallel cumulative logit model can have up to two nonzero coefficients per predictor. The semi-parallel cumulative logit and multinomial logistic regression models can have up to three nonzero coefficients per predictor. By design, the group lasso penalty either selects all three coefficients or sets them all to zero. Note that the coefficient estimates from `MASS::polr` are multiplied by -1 to be consistent with the **ordinalNet** and **ordinalgmifs** parameterizations of the cumulative logit model.

## 2.6 The ordinalNet R package

The **ordinalNet** package (version 2.2) (Wurm *et al.*, 2017) contains the following functions:

- `ordinalNet` is the main function for fitting parallel, nonparallel, and semi-parallel regression models with the elastic net penalty. It returns an "ordinalNet" S3 object. This object has `print`, `summary`, `coef`, and `predict` methods.

- `ordinalNetTune` uses K-fold cross validation to obtain out-of-sample log-likelihood and misclassification rates for a sequence of lambda values. It returns an "ordinalNetTune" S3 object. This object has `print`, `summary`, and `print` methods.

- `ordinalNetCV` uses K-fold cross validation to obtain out-of-sample log-likelihood and misclassification rates. Lambda is tuned within each cross validation fold. It returns an "ordinalNetCV" S3 object. This object has `print` and `summary` methods.

Below is a description of the `ordinalNet` function and its arguments.

```
ordinalNet(x, y, alpha = 1, standardize = TRUE, penaltyFactors = NULL,
  positiveID = NULL, family = c("cumulative", "sratio", "cratio", "acat"),
  reverse = FALSE, link = c("logit", "probit", "cloglog", "cauchit"),
  customLink = NULL, parallelTerms = TRUE, nonparallelTerms = FALSE,
  parallelPenaltyFactor = 1, lambdaVals = NULL, nLambda = 20,
  lambdaMinRatio = 0.01, includeLambda0 = FALSE, alphaMin = 0.01,
  pMin = 1e-08, stopThresh = 1e-04, threshOut = 1e-08, threshIn = 1e-08,
  maxiterOut = 100, maxiterIn = 1000, printIter = FALSE,
  printBeta = FALSE, warn = TRUE, keepTrainingData = TRUE)
```

### Arguments

**x** Covariate matrix. It is recommended that categorical covariates are converted to a set of indicator variables with a variable for each category (i.e. no baseline category); otherwise the choice of baseline category will affect the model fit.

**y** Response variable. Can be a factor, ordered factor, or a matrix where each row is a multinomial vector of counts. A weighted fit can be obtained using the matrix option, since the row sums are essentially observation weights. Non-integer matrix entries are allowed.

**alpha** The elastic net mixing parameter, with `0 <= alpha <= 1`. `alpha=1` corresponds to the lasso penalty, and `alpha=0` corresponds to the ridge penalty.

**standardize** If `standardize=TRUE`, the predictor variables are scaled to have unit variance. Coefficient estimates are returned on the original scale.

**penaltyFactors** Nonnegative vector of penalty factors for each variable. This vector is multiplied by lambda to get the penalty for each variable. If `NULL`, the penalty factor is one for each coefficient.

**positiveID** Logical vector indicating whether each coefficient should be constrained to be non-negative. If `NULL`, the default value is `FALSE` for all coefficients.

**family** Specifies the type of model family. Options are "cumulative" for cumulative probability, "sratio" for stopping ratio, "cratio" for continuation ratio, and "acat" for adjacent category.

**reverse** Logical. If `TRUE`, then the "backward" form of the model is fit, i.e. the model is defined with response categories in reverse order. For example, the reverse cumulative model with $K+1$ response categories applies the link function to the cumulative probabilities $P(Y \geq 2), \ldots, P(Y \geq K+1)$, rather then $P(Y \leq 1), \ldots, P(Y \leq K)$.

**link** Specifies the link function. The options supported are logit, probit, complementary log-log, and cauchit. Only used if `customLink=NULL`.

**customLink** Optional list containing a vectorized link function `g`, a vectorized inverse link `h`, and the Jacobian function of the inverse link `getQ`. The Jacobian should be defined as $\partial h(\eta)/\partial \eta^T$ (as opposed to the transpose of this matrix).

**parallelTerms** Logical. If `TRUE`, then parallel coefficient terms will be included in the model. `parallelTerms` and `nonparallelTerms` cannot both be `FALSE`.

**nonparallelTerms** Logical. if `TRUE`, then nonparallel coefficient terms will be included in the model. `parallelTerms` and `nonparallelTerms` cannot both be `FALSE`.

**parallelPenaltyFactor** Numeric value greater than or equal to zero. Lambda is multiplied by this factor (as well as variable-specific `penaltyFactors`) to obtain the penalties for parallel terms. Only used if `parallelTerms=TRUE`.

**lambdaVals** An optional user-specified lambda sequence (vector). If `NULL`, a sequence will be generated based on `nLambda` and `lambdaMinRatio`. In this case, the maximum lambda is the smallest value that sets all penalized coefficients to zero, and the minimum lambda is the maximum value multiplied by the factor `lambdaMinRatio`.

**nLambda** Positive integer. The number of lambda values in the solution path. Only used if `lambdaVals=NULL`.

**lambdaMinRatio** A factor greater than zero and less than one. Only used if `lambdaVals=NULL`.

**includeLambda0** Logical. If `TRUE`, then zero is added to the end of the sequence of `lambdaVals`. This is not done by default because it can significantly increase computational time. An unpenalized saturated model may have infinite coefficient solutions, in which case the fitting algorithm will still terminate when the relative change in log-likelihood becomes small. Only used if `lambdaVals=NULL`.

**alphaMin** Value greater than zero, but much less than one. If `alpha < alphaMin`, then `alphaMin` is used to calculate the maximum lambda value. When `alpha=0`, the maximum lambda value would be infinite otherwise.

**pMin** Value greater than zero, but much less than one. During the optimization routine, the Fisher information is calculated using fitted probabilities. For this calculation, fitted probabilities are capped below by this value to prevent numerical instability.

**stopThresh** In the relative log-likelihood change between successive lambda values falls below this threshold, then the last model fit is used for all remaining lambda.

**threshOut** Convergence threshold for the coordinate descent outer loop. The optimization routine terminates when the relative change in the penalized log-likelihood between successive iterations falls below this threshold. It is recommended to set `theshOut` equal to `threshIn`.

**threshIn** Convergence threshold for the coordinate descent inner loop. Each iteration consists of a single loop through each coefficient. The inner loop terminates when the relative change in the penalized approximate log-likelihood between successive iterations falls below this threshold. It is recommended to set `theshOut` equal to `threshIn`.

**maxiterOut** Maximum number of outer loop iterations.

**maxiterIn** Maximum number of inner loop iterations.

**printIter** Logical. If `TRUE`, the optimization routine progress is printed to the terminal.

**printBeta** Logical. If `TRUE`, coefficient estimates are printed after each coordinate descent outer loop iteration.

**warn** Logical. If `TRUE`, the following warning message is displayed when fitting a cumulative probability model with `nonparallelTerms=TRUE` (i.e. nonparallel or semi-parallel model). "Warning message: For out-of-sample data, the cumulative probability model with nonparallelTerms=TRUE may predict cumulative probabilities that are not monotone increasing." The warning is displayed by default, but the user may wish to disable it.

**keepTrainingData** Logical. If `TRUE`, then `x` and `y` are saved with the returned "ordinalNet" object. This allows `predict.ordinalNet` to return fitted values for the training data without passing a `newx` argument.

## 2.7 Demonstration in R

This section contains five examples that demonstrate different aspects of the **ordinalNet** package (Wurm *et al.*, 2017) using the GSE18081 dataset from the Gene Expression Omnibus.

Examples 1-3 demonstrate how to fit the parallel, semi-parallel, and nonparallel model forms, respectively, using the `ordinalNet` function. The model form is determined by the boolean arguments `parallelTerms` and `nonparallelTerms`, as shown in Table 2.6.

| | `parallelTerms` | `nonparallelTerms` |
|---:|:---:|:---:|
| Parallel | `TRUE` | `FALSE` |
| Nonparallel | `FALSE` | `TRUE` |
| Semi-parallel | `TRUE` | `TRUE` |

Table 2.6: Argument settings for the parallel, nonparallel, and semi-parallel model forms.

Example 4 demonstrates how one might use the `ordinalNetTune` function to select an appropriate value for the tuning parameter $\lambda$. This function uses cross validation to evaluate the out-of-sample prediction performance of a sequence of $\lambda$ values.

Example 5 demonstrates the `ordinalNetCV` function, which tunes the model within each cross validation fold and evaluates the out-of-sample prediction performance. This provides an unbiased estimate of the out-of-sample performance of a tuned model. A variety of tuning procedures can be used, including AIC, BIC, and cross validation (within each fold).

## Example 1

We fit a parallel cumulative probability model with logit link (proportional odds model). We set `parallelTerms = TRUE` and `nonparallelTerms = FALSE` to obtain the parallel model fit. We use the default elastic net tuning parameter `alpha = 1` to select the lasso penalty. We use the default settings of `lambdaVals = NULL`, `nLambda = 20`, and `lambdaMinRatio = 0.01` to generate a sequence of twenty $\lambda$ values, with $\lambda_{\text{max}}$ equal to the smallest value that sets every coefficient to zero and $\lambda_{\text{min}} = \lambda_{\text{max}} \cdot 0.01$. The sequence runs from $\lambda_{\text{min}}$ to $\lambda_{\text{max}}$ uniformly on the logarithmic scale.

```
R> library(ordinalNet)
R> library(ordinalgmifs)
R> library("Biobase")
R> data(hccmethyl)
R> y <- pData(hccmethyl)$group
R> x <- t(exprs(hccmethyl))
R> fit1 <- ordinalNet(x, y, family = "cumulative", link = "logit",
R+   parallelTerms = TRUE, nonparallelTerms = FALSE)
```

The `summary` method displays the lambda sequence (`lambdaVals`), number of nonzero coefficients (`nNonzero`), the log-likelihood (`loglik`), percent deviance explained (`pctDev`), and AIC and BIC. The AIC and BIC are calculated using `nNonzero` as the approximate degrees of freedom

```
R> head(summary(fit1))
```

```
  lambdaVals nNonzero    loglik    devPct        aic       bic
```

```
1  0.4287829        2 -61.22898 0.0000000 126.45797 130.5087
2  0.3364916        6 -49.70793 0.1881634 111.41586 123.5680
3  0.2640652       10 -40.97485 0.3307932 101.94970 122.2032
4  0.2072278       11 -33.86289 0.4469467  89.72579 112.0047
5  0.1626241       12 -28.29049 0.5379560  80.58097 104.8852
6  0.1276209       15 -23.15157 0.6218855  76.30313 106.6834
```

The `coef` method returns the coefficient estimates of any model fit in the $\lambda$ sequence. The best AIC fit is selected by default. The `matrix = TRUE` option returns the coefficients in matrix form with a column corresponding to each linear predictor. Because this is a parallel model, the coefficient columns are identical except for the intercepts.

```
R> head(coef(fit1, matrix = TRUE))

                     logit(P[Y<=1]) logit(P[Y<=2])
(Intercept)              -27.997567     -19.157113
CDKN2B_seq_50_S294_F     -13.774058     -13.774058
DDIT3_P1313_R             -8.393522      -8.393522
ERN1_P809_R                1.215556       1.215556
GML_E144_F                 7.263032       7.263032
HDAC9_P137_R               0.000000       0.000000
```

**Example 2**

By setting `parallelTerms = TRUE` and `nonparallelTerms = TRUE`, we obtain the semi-parallel model fit. Because this is a cumulative probability model, we set `warn = FALSE` to suppress the warning that the semi-parallel form is susceptible to non-monotone cumulative probabilities for out-of-sample predictions. We use the default semi-parallel tuning parameter $\rho$, which is `parallelPenaltyFactor = 1`. The coefficient matrix of the best AIC fit has nearly identical columns for the first several predictors, but they differ for the first predictor.

```
R> fit2 <- ordinalNet(x, y, family = "cumulative", link = "logit",
R+   parallelTerms = TRUE, nonparallelTerms = TRUE, warn = FALSE)
R>
R> head(coef(fit2, matrix = TRUE))
```

```
                     logit(P[Y<=1]) logit(P[Y<=2])
(Intercept)              -23.518682     -22.199966
CDKN2B_seq_50_S294_F      -5.732730     -18.218945
DDIT3_P1313_R             -8.604492      -8.604492
ERN1_P809_R                1.010048       1.010048
GML_E144_F                 7.414796       7.414796
HDAC9_P137_R               0.000000       0.000000
```

### Example 3

By setting `parallelTerms = FALSE` and `nonparallelTerms = TRUE`, we obtain the nonparallel model fit. This example demonstrates the problem with the nonparallel cumulative probability model discussed in Sections 2.2.5 and 2.3.11. (The semi-parallel model is also susceptible to this issue, but it is less prone.) The solution path is terminated after the third $\lambda$ value where the optimum leaves the constrained parameter space. This can be seen from the `summary` method output.

```
R> fit3 <- ordinalNet(x, y, family = "cumulative", link = "logit",
R+   parallelTerms = FALSE, nonparallelTerms = TRUE, warn = FALSE)
R>
R> head(summary(fit3))
```

```
  lambdaVals nNonzero    loglik    devPct      aic      bic
1  0.4046054        2 -61.22898 0.0000000 126.4580 130.5087
2  0.3175182        4 -52.35095 0.1449972 112.7019 120.8033
3  0.2491755        6 -44.89073 0.2668386 101.7815 113.9336
4  0.1955430        6 -44.89073 0.2668386 101.7815 113.9336
5  0.1534543        6 -44.89073 0.2668386 101.7815 113.9336
6  0.1204248        6 -44.89073 0.2668386 101.7815 113.9336
```

### Example 4

The `ordinalNetTune` function uses $K$-fold cross validation to obtain out-of-sample performance for a sequence on $\lambda$ values. We demonstrate this function using the parallel cumulative logit model. We use the default setting of `nFolds = 5` and the default sequence of twenty $\lambda$ values obtained from the model fit to the full data. We also set `lambdaMinRatio = 1e-4` instead of the default value 0.01.

The `summary` method returns the average out-of-sample log-likelihood and misclassification rate for each $\lambda$ value. The average is taken across all cross validation folds. The user can use this information to tune the model, for example by selecting the $\lambda$ value with the best average out-of-sample log-likelihood, as demonstrated below. We obtain the coefficient estimates at this $\lambda$ value, which happens to be the thirteenth value in the sequence.

```
R> set.seed(123)
R> fit1_tune <- ordinalNetTune(x, y, family = "cumulative", link = "logit",
R+   lambdaMinRatio = 1e-04, printProgress = FALSE)
R>
R> summary(fit1_tune)
```

```
         lambda loglik_avg misclass_avg
1  4.287829e-01 -12.116291   0.55151515
2  2.640652e-01  -9.072777   0.28484848
3  1.626241e-01  -7.060731   0.21363636
4  1.001517e-01  -5.300055   0.10606061
5  6.167827e-02  -3.914672   0.08787879
6  3.798445e-02  -3.025536   0.08787879
7  2.339266e-02  -2.442411   0.06969697
8  1.440633e-02  -2.042826   0.06969697
9  8.872110e-03  -1.792743   0.05151515
10 5.463873e-03  -1.665196   0.06969697
11 3.364916e-03  -1.659950   0.06969697
12 2.072278e-03  -1.613671   0.06969697
13 1.276209e-03  -1.600095   0.06969697
14 7.859507e-04  -1.623936   0.08787879
15 4.840264e-04  -1.688270   0.06969697
16 2.980868e-04  -1.773509   0.06969697
17 1.835762e-04  -1.883374   0.06969697
18 1.130551e-04  -2.006193   0.06969697
19 6.962477e-05  -2.124836   0.06969697
20 4.287829e-05  -2.248505   0.06969697
```

```
R> head(coef(fit1_tune$fit, matrix = TRUE, whichLambda = 13))

                     logit(P[Y<=1]) logit(P[Y<=2])
(Intercept)              -74.491490     -60.134706
CDKN2B_seq_50_S294_F     -22.251784     -22.251784
DDIT3_P1313_R            -12.443911     -12.443911
ERN1_P809_R                3.567562       3.567562
GML_E144_F                12.108139      12.108139
HDAC9_P137_R               0.000000       0.000000
```

**Example 5**

The `ordinalNetCV` function uses $K$-fold cross validation to evaluate the out-of-sample performance of models that are tuned within each cross validation fold. We demonstrate this function again using the parallel cumulative logit model. We use the default value of `nFolds = 5`. We also use the default settings of `nFoldsCV = 5` and `tuneMethod = "cvLoglik"` to tune $\lambda$ by 5-fold cross validation within each fold, each time selecting the $\lambda$ value with the best average out-of-sample log-likelihood. As in Example 4, we set `lambdaMinRatio = 1e-4`.

The `summary` method returns the $\lambda$ value selected by the tuning procedure for each cross validation fold, along with the out-of-sample log-likelihood and misclassification rate.

```
R> set.seed(123)
R> fit1_cv <- ordinalNetCV(x, y, family = "cumulative", link = "logit",
R+   lambdaMinRatio = 1e-04, printProgress = FALSE)
R>
R> summary(fit1_cv)

           lambda     loglik   misclass
fold1 0.008872110 -3.7368700 0.16666667
fold2 0.005463873 -0.6340910 0.00000000
fold3 0.008872110 -0.9913706 0.00000000
fold4 0.023392656 -2.6007982 0.09090909
fold5 0.037984451 -3.7018244 0.09090909
```

## 2.8 Discussion

This chapter introduced the elmentwise link, multinomial-ordinal (ELMO) model class, a rich class of multinomial regression models that includes commonly used categorical regression models. Each of these models has both a parallel and nonparallel form. The parallel form is appropriate for ordinal data, while the nonparallel form is a more flexible model which can be used with an unordered categorical response. We also introduced the semi-parallel model form, which can be used with the elastic net penalty to shrink the nonparallel model toward the parallel model.

The motivation for this work began with a need to extend variable selection tools for ordinal logistic regression. For instance, consider the problem of developing a gene signature to predict response to a novel therapy, where the observed patient response belongs to one of the following categories: no response, partial response, or complete response. We developed these tools in the general ELMO framework. Specifically, we proposed a coordinate descent fitting algorithm for the ELMO class with the elastic net penalty. The algorithm is general and can also be applied to multinomial regression models outside the ELMO class.

We considered numerical experiments to highlight different features of the model class and to demonstrate the use of the related R code. We presented different simulation scenarios to demonstrate cases where the parallel, nonparallel, and semi-parallel each achieved better out-of-sample prediction performance than the other two models. With the Gene Expression Omnibus GSE18081 data set, we demonstrated the use of the penalized ELMO class for prediction and variable selection.

Finally, we introduced the R package **ordinalNet**, which implements a coordinate descent algorithm for parallel, nonparallel, and semi-parallel models of the ELMO class.

We consider two possible directions for future research: code speedup via and questions of statistical inference. **Rcpp** and **RcppArmadillo** are R packages which allow integration of C++ code into R (Eddelbuettel and François, 2011; Eddelbuettel, 2013; Eddelbuettel and Sanderson, 2014). Our code is written with separate functions for the inner and outer coordinate descent loops. Because of the number of calls to it in a typical run of the algorithm, the inner loop, in particular, could benefit from speed up via C++ .

The **ordinalNet** package does not provide standard errors for estimates. We quote a relevant section from the **penalized** vignette (Goeman *et al.*, 2017).

It is a very natural question to ask for standard errors of regression coefficients

> or other estimated quantities. In principle such standard errors can easily be calculated, e.g. using the bootstrap. Still, this package deliberately does not provide them. The reason for this is that standard errors are not very meaningful for strongly biased estimates such as arise from penalized estimation methods. Penalized estimation is a procedure that reduces the variance of estimators by introducing substantial bias. The bias of each estimator is therefore a major component of its mean squared error, whereas its variance may contribute only a small part.

The topic of post-selection inference has been studied in both the classic setting (Zhang, 1992; Leeb and Pötscher, 2005; Wang and Lagakos, 2009; Berk *et al.*, 2013), where the number of observations exceeds the number of predictors, and the high-dimensional setting (Javanmard and Montanari, 2014; Lockhart *et al.*, 2014; Tibshirani *et al.*, 2016). In the high-dimensional setting, we would like to highlight the groundbreaking work of Lockhart *et al.* (2014). They proved the asymptotic distribution of their test statistic specifically for the linear model, but their simulation results suggest that the same test statistic could be used for generalized linear models. This work may provide a path for post-selection inference for penalized multinomial and ordinal regression models.

# 3 COX-ASSISTED CLUSTERING EXTENSIONS AND OPTIMIZATION

## 3.1 Introduction

Cox-assisted clustering (CAC) (Eng and Hanlon, 2014) is a semiparametric finite mixture of regression model for survival analysis. It can be used to detect and model heterogeneity in a population. Each member of the population is assumed to belong to one of finitely many latent classes. Within each class, survival times are assumed to follow a proportional hazards model. The model coefficients are allowed to vary by class. The baseline hazard function of each class is estimated nonparametrically.

Eng and Hanlon (2014) provide a concrete example from cancer genomics, where a gene effect appears to vary between two subpopulations. The data come from The Cancer Genome Atlas (TCGA) ovarian cancer study population. Patients were classified as having either optimal or suboptimal cytoreduction, which is the amount of residual tumor remaining after surgery. Suboptimally cytoreduced patients have a a clinically significant amount of residual tumor that can cause disease recurrence and progression (Bhoola and Hoskins, 2006). Increased expression of the epiregulin gene has a protective effect among suboptimally cytoreduced patients, but a detrimental effect among optimally cytoreduced patients. In this example, the population subclasses (cytoreduction levels) are known, so the CAC model is not required to identify them. However, the example illustrates the type of covariate effect heterogeneity that CAC is intended to model. In cases where the sub-classes are unknown, CAC may be used to identify them.

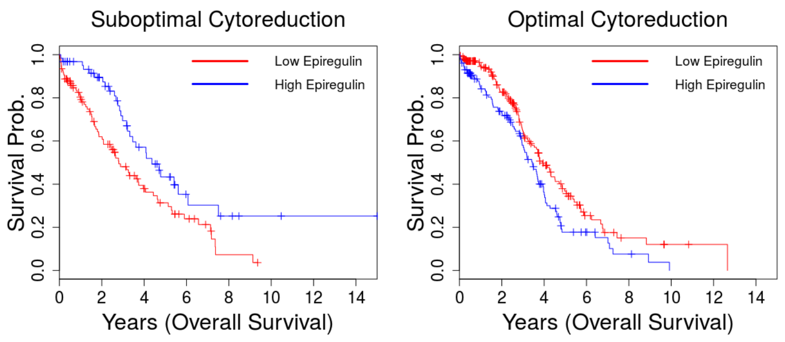

Figure 3.1: Kaplan-Meier plots for high and low epiregulin expression groups within each cytoreduction subpopulation. Data come from the TCGA ovarian cancer study population.

Our contribution to the development of CAC is threefold. First, we propose restrictions on the baseline hazard functions that can improve model identifiability. Second, we propose an improved method for generating random starting values for the EM algorithm. Third, we propose a method for inference.

The original CAC model (Eng and Hanlon, 2014) allowed a different baseline hazard function in each latent class, with no restriction. This is a rather flexible class of models, which can make it weakly identifiable. In fact, our simulation study in Section 3.4 suggests that the maximum likelihood coefficient estimators have considerable bias and mean squared error under this model. We propose restricting the baseline hazard functions to be either identical or proportional. This greatly reduces the number of parameters in the semiparametric likelihood and should improve efficiency if either assumption holds. Furthermore, our simulation study suggests that the estimators are consistent and asymptotically normal under either of these restrictions.

The CAC semiparametric likelihood function used for estimation is non-convex. We follow a typical procedure for mixture models by generating multiple sets of random starting values to find as many local optima as possible and choose the estimate to be the solution among the local optima with the greatest likelihood. It is important to have sufficient variation in the starting values to find more local optima and increase the chance of finding the maximum likelihood solution. To this end, we propose a new method of generating random starting values.

We also propose a method for estimating standard errors for parameter estimators. To do this, we use numerical differentiation techniques to approximate the Hessian of the profile semiparametric likelihood.

In Section 3.2, we present the CAC model and assumptions. In Section 3.3, we present variations of the EM algorithm optimization procedure for each of the proposed baseline hazard restrictions. We also introduce the new method for generating random starting values. In Section 3.4, we present a method for estimating standard errors. In Section 3.5, we present a simulation study to check the finite sample performance of the estimators and estimated standard errors. In Section 3.6, we present an example where the new random start method shows benefit over the previously proposed starting method for CAC. In Section 3.7, we apply CAC to a data set from cancer genomics where genes appear to affect survival differently in different subpopulations. In Section 3.8, we present ideas for future work on CAC. Section 3.9 is a discussion of our findings.

## 3.2 CAC assumptions and semiparametric likelihood

### 3.2.1 Assumptions

For observations $i = 1, \ldots, N$, we define the following random variables.

$T_i$ is the survival time.

$C_i$ is the right censoring time.

$Y_i = \min(T_i, C_i)$ is the follow-up time.

$\Delta_i = I(T_i \leq C_i)$ is the failure indicator.

$X_i$ is a vector of covariates.

$U_i \in \{1, \ldots, K\}$ is the latent class.

We use lowercase letters $(t_i, c_i, y_i, \delta_i, x_i, u_i)$ to denote the realized values of the random variables. We drop the subscript to denote the entire sample, e.g. $T = (T_1, \ldots, T_N)$ and $t = (t_1, \ldots, t_N)$. The data actually observed are $(y, \delta, x)$.

Eng and Hanlon (2014) cites the seminal work by Cox (1972), stating "assume Cox's proportional hazards model within each class". We present a more explicit set of CAC model assumptions, which generalizes Cox's proportional hazards model to a mixture model.

A1. $C \perp\!\!\!\perp (T, U) | X$

A2. $U_i$ are mutually independent conditional on $X$, and $\mathrm{P}(U_i = k|X) = \pi_k$, where $\pi_k$ is a constant.

A3. $T_i$ are mutually independent conditional on $(U, X)$, and the conditional hazard of $T_i|(X, U)$, as a function of $t$, is $\sum_{k=1}^{K} I(U_i = k)\lambda_k(t)\exp(x_i^T\beta_k)$, where each $\lambda_k(\cdot)$ is a baseline hazard function with a continuous distribution.

Note that the class probabilities $P(U_i = k|X_i = x_i)$ could be a function of $x_i$, but we do not address this scenario.

### 3.2.2 Semiparametric likelihood

The parameters of the model are $\pi = (\pi_1, \ldots, \pi_k)$, $\beta = (\beta_1, \ldots, \beta_k)$, and $\lambda = (\lambda_1(\cdot), \ldots, \lambda_K(\cdot))$. Let $\Lambda_k(t) = \int_0^t \lambda_k(s)ds$ be the cumulative hazard function of each class. The likelihood for the parameters of interest $(\pi, \beta, \lambda)$ can be written

$$\begin{aligned}
&L(\pi, \beta, \lambda; y, \delta | x) = \\
&\quad = \mathrm{P}(Y = y, \Delta = \delta | X = x) \\
&\quad = \mathrm{P}\Bigg( \Big\{ \bigcap_{i:\delta_i = 1} \{C_i \geq y_i, T_i = y_i\} \Big\} \cap \Big\{ \bigcap_{i:\delta_i = 0} \{C_i = y_i, T_i \geq y_i\} \Big\} \Bigg| X = x \Bigg) \\
&\quad = \mathrm{P}\Bigg( \Big\{ \bigcap_{i:\delta_i = 1} \{C_i \geq y_i\} \Big\} \cap \Big\{ \bigcap_{i:\delta_i = 0} \{C_i = y_i\} \Big\} \Bigg| X = x \Bigg) \times \\
&\qquad \times \mathrm{P}\Bigg( \Big\{ \bigcap_{i:\delta_i = 1} \{T_i = y_i\} \Big\} \cap \Big\{ \bigcap_{i:\delta_i = 0} \{T_i \geq y_i\} \Big\} \Bigg| X = x \Bigg) && \text{by (A1)} \\
&\quad \propto \mathrm{P}\Bigg( \Big\{ \bigcap_{i:\delta_i = 1} \{T_i = y_i\} \Big\} \cap \Big\{ \bigcap_{i:\delta_i = 0} \{T_i \geq y_i\} \Big\} \Bigg| X = x \Bigg) && \text{by (A1)} \\
&\quad = \sum_{u \in \{1, \ldots, K\}^N} \Bigg\{ \mathrm{P}(U = u | X = x) \times \\
&\qquad \times \mathrm{P}\Bigg( \Big\{ \bigcap_{i:\delta_i = 1} \{T_i = y_i\} \Big\} \cap \Big\{ \bigcap_{i:\delta_i = 0} \{T_i \geq y_i\} \Big\} \Bigg| X = x, U = u \Bigg) \Bigg\} \\
&\quad = \sum_{u \in \{1, \ldots, K\}^N} \prod_{i=1}^N \sum_{k=1}^K \Big\{ I(u_i = k) \pi_k [\lambda_k(y_i) \exp(x_i^T \beta_k)]^{\delta_i} \times \\
&\qquad \times \exp\{-\Lambda_k(y_i) \exp(x_i^T \beta_k)\} \Big\} && \text{by (A2)-(A3)} \\
&\quad = \prod_{i=1}^N \sum_{k=1}^K \pi_k [\lambda_k(y_i) \exp(x_i^T \beta_k)]^{\delta_i} \exp\{-\Lambda_k(y_i) \exp(x_i^T \beta_k)\} \, .
\end{aligned}$$

As with the partial likelihood of the standard Cox model, we have factored out the censoring portion of the likelihood because it does not depend on the parameters of interest. For estimation, we work with a semiparametric likelihood, replacing each $\lambda_k(y_i)$ with point mass $h_{ik}$. Let $h_k = (h_{1k}, \ldots, h_{Nk})$ be the finite-dimensional representation of $\lambda_k(\cdot)$, and let $h = (h_1, \ldots, h_K)$. Likewise, we replace $\Lambda_k(y_i)$ with $H_{ik} = \sum_{\{j: y_j \leq y_i\}} h_{ik}$. For simplicity of notation we assume no tied failure times. Letting $h = (h_1, \ldots, h_K)$, the observed data likelihood becomes

$$L(\beta, \pi, h; y, \delta|x) = \prod_{i=1}^{N} \sum_{k=1}^{K} \pi_k [h_{ik} \exp(x_i^T \beta_k)]^{\delta_i} \exp\{-H_{ik} \exp(x^T \beta_k)\} \ .$$

We leave the likelihood in its semiparametric form, rather than profiling over the baseline hazard because the profile likelihood does not have a closed form as it does for the standard Cox model. We estimate the parameters $(\pi, \beta, h)$ by optimizing the semiparametric likelihood, which is nonconvex like most mixture model likelihood functions. We use an EM algorithm approach to find multiple local optima by choosing different sets of starting values. We choose our estimate to be the local optimum with the maximum likelihood among all local optima found.

## 3.3 EM algorithm optimization procedure

Eng and Hanlon (2014) used the EM algorithm (Dempster *et al.*, 1977) to optimize the semiparametric log likelihood. We present the algorithm here with three different variations to accommodate proportional and identical baseline hazard functions, as well as the original version with no restriction.

### 3.3.1 Parameter updates

The EM algorithm procedure alternates between (1) calculating the expectation of the complete data log-likelihood, conditional on the observed data, using the parameter estimates from the previous iteration and (2) optimizing the expected complete data log-likelihood over the parameters $(\pi, \beta, h)$. Step (1) is known as the E-step, and step (2) is known as the M-step. The complete data likelihood is simply defined as

$$L(\beta, \pi, h; y, \delta, u|x) = \mathrm{P}(Y = y, \Delta = \delta, U = u|X = x) \ ,$$

which is what the likelihood function would be if the latent class of each observation were known. In order to express the complete data log-likelihood, we define a set of indicator variables $(U_{i1}, \ldots, U_{iK})$ to indicate the latent class of the $i^{\text{th}}$ observation. (Each indicator is zero, except for the one in the $U_i^{\text{th}}$ position.) The complete data likelihood can then be written

$$\log L(\beta, \pi, h; y, \delta, u|x) = \log L_1(\pi; u) + \log L_2(\beta, h; y, \delta|u, x) ,$$

where

$$\log L_1(\pi; u) = \sum_{i=1}^{N} \sum_{k=1}^{K} u_{ik} \log \pi_k$$

and

$$\log L_2(\beta, h; y, \delta|u, x) = \sum_{i=1}^{N} \sum_{k=1}^{K} \delta_i u_{ik} \log h_{ik} + \delta_i u_{ik} x_i^T \beta_k - u_{ik} H_{ik} \exp(x_i^T \beta_k) .$$

## E-step

The E-step calculates $\mathrm{E}[\log L(\pi, \beta, h; y, \delta, U|x)|Y = y, \Delta = \delta, X = x]$. The indicators $U$ are the only random variables in the conditional expectation. Because $L_1$ and $L_2$ are both linear in the class indicators, taking the conditional expectation simply amounts to replacing each $U_{ik}$ with

$$\begin{aligned} \hat{u}_{ik} &= \mathrm{E}(U_{ik}|Y = y, \Delta = \delta, X = x) \\ &= \frac{\pi_k (h_{ik} \exp(x_i^T \beta_k))^{\delta_i} \exp(-H_{ik} e^{x_i^T \beta_k})}{\sum\limits_{k'=1}^{K} \pi_{k'} (h_{k'i} \exp(x_i^T \beta_{k'}))^{\delta_i} \exp(-H_{ik'} e^{x_i^T \beta_{k'}})} . \end{aligned}$$

## M-step

The M-step optimizes $\log L(\beta, \pi, h; y, \delta, \hat{u}|x)$ over $(\pi, \beta, h)$. Because $L_1$ is a function of $\pi$ only and $L_2$ is a function of $(\beta, h)$ only, these components can be optimized independently. The function $L_1(\pi; \hat{u})$ attains its optimum at the value

$$\hat{\pi} = \left( \frac{1}{N} \sum_{i=1}^{N} \hat{u}_{i1}, \ldots, \frac{1}{N} \sum_{i=1}^{N} \hat{u}_{iK} \right) .$$

To optimize $L_2(\beta, h; y, \delta|u, x)$ over respect to $(\beta, h)$, we first profile over $h$, replacing it with $h^*(\beta) = \arg\max_h \log L_2(\beta, h; y, \delta|u, x)$. The profile likelihood then takes the form of a standard Cox model profile likelihood. The exact steps depend on the baseline hazard restriction. When the baseline hazards are non-proportional (i.e. unrestricted), the profile log-likelihood is the sum of $K$ separate weighted Cox log-likelihoods with different coefficients and baseline hazard functions. These likelihoods can be optimized independently. When the baseline hazard functions are restricted to be proportional or identical, the profile log-likelihood is a weighted Breslow log-likelihood (Breslow, 1974) with $K$ tied failures at each failure time. We present the technique for optimizing $L_2$ over $(\beta, h)$ under each baseline hazard assumption.

### Non-proportional baseline hazard functions

Setting $\frac{\partial L_2}{\partial h_{ik}} = 0$, we obtain the profile likelihood estimate for $h_{ik}$:

$$h^*_{ik}(\beta) = \frac{\delta_i \hat{u}_{ik}}{\sum_{j \in R_i} u_{jk} \exp(x_j^T \beta_k)} \; .$$

The profile log-likelihood is

$$\begin{aligned}
&\log L_2(\beta, h^*(\beta); y, \delta, u|x) = \\
&= \sum_{k=1}^{K} \sum_{i=1}^{N} \Bigg\{ \delta_i u_{ik} \Bigg[ \log(\delta_i u_{ik}) - \log\Bigg( \sum_{j \in R_i} u_{jk} \exp(x_j^T \beta_k) \Bigg) + x_i^T \beta_k \Bigg] \\
&\qquad - u_{ik} \exp(x_i^T \beta_k) \sum_{j: y_j \le y_i} \frac{\delta_j u_{jk}}{\sum_{s \in R_j} u_{sk} \exp(x_s^T \beta_k)} \Bigg\} \\
&= \sum_{k=1}^{K} \sum_{i=1}^{N} \delta_i u_{ik} \Bigg[ x_i^T \beta_k - \log\Bigg( \sum_{j \in R_i} u_{jk} \exp(x_j^T \beta_k) \Bigg) \Bigg] + C(u),
\end{aligned}$$

where $C(u)$ is a constant that does not depend on $\beta$. As stated previously, this is the sum of $K$ separate weighted Cox log-likelihoods, which can be optimized independently to obtain $\hat{\beta}$ and $\hat{h} = h^*(\hat{\beta})$.

### Proportional baseline hazard functions

To enforce the proportionality restriction, let $h_{ik} = h_i e^{\zeta_k}$ for all $i$ and $k = 1, \ldots, K$. For identifiability, we arbitrarily select the $K^{\text{th}}$ class to be the baseline, setting $\zeta_K = 0$. Setting $\frac{\partial L_2}{\partial h_i} = 0$, we obtain the profile likelihood estimate for $h_i$:

$$h^*_i(\beta) = \frac{\sum_{k=1}^{K} \delta_i u_{ik}}{\sum_{k'=1}^{K} \sum_{j \in R_i} u_{jk'} \exp(x_j^T \beta_{k'} + \zeta_{k'})}.$$

The profile likelihood is

$$
\begin{aligned}
&\log L_2(\beta, h^*(\beta); y, \delta, u|x) = \\
&= \sum_{k=1}^{K}\sum_{i=1}^{N}\Bigg\{\delta_i u_{ik}\Bigg[\log\left(\sum_{k'=1}^{K}\delta_i u_{ik'}\right) - \log\left(\sum_{k'=1}^{K}\sum_{j\in R_i} u_{jk'}\exp(x_j^T\beta_{k'}+\zeta_{k'})\right) + \\
&\quad + x_i^T\beta_k + \zeta_k\Bigg] - u_{ik}\exp(x_i^T\beta_k+\zeta_k)\sum_{j:y_j\le y_i}\frac{\sum_{k'=1}^{K}\delta_j u_{jk'}}{\sum_{k''=1}^{K}\sum_{s\in R_j} u_{sk''}\exp(x_s^T\beta_{k''}+\zeta_{k''})}\Bigg\} \\
&= \sum_{k=1}^{K}\sum_{i=1}^{N}\delta_i u_{ik}\left[x_i^T\beta_k + \zeta_k - \log\left(\sum_{k'=1}^{K}\sum_{j\in R_i} u_{jk'}\exp(x_j^T\beta_{k'}+\zeta_{k'})\right)\right] + C(u)\ ,
\end{aligned}
$$

where $C(u)$ is a constant that does not depend on $\beta$. As stated above, this is a weighted Breslow log-likelihood with $K$ tied failures at each failure time. The coefficient vector is $(\zeta_1, \ldots, \zeta_{K-1}, \beta_1^T, \ldots, \beta_K^T)^T$, and the covariate matrix is expanded into a block form. Let $\boldsymbol{X} = (x_1, \ldots x_N)^T$ be an $N \times P$ covariate matrix, and let $\mathbf{0}$ be an $N \times P$ matrix of zeros. Also let $\tilde{0}$ be a vector of $N$ zeros, and let $\tilde{1}$ be vector of $N$ ones. The block covariate matrix takes the form

$$
\begin{pmatrix}
\tilde{1} & \tilde{0} & \cdots & \tilde{0} & \boldsymbol{X} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\
\tilde{0} & \tilde{1} & \cdots & \tilde{0} & \mathbf{0} & \boldsymbol{X} & \cdots & \mathbf{0} & \mathbf{0} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\tilde{0} & \tilde{0} & \cdots & \tilde{1} & \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{X} & \mathbf{0} \\
\tilde{0} & \tilde{0} & \cdots & \tilde{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \boldsymbol{X}
\end{pmatrix} .
$$

The profile likelihood is optimized to obtain $\hat{\beta}$ and $\hat{h} = h^*(\hat{\beta})$.

### Identical baseline hazard functions

This is a special case of proportional hazards where $\zeta_1 = \ldots = \zeta_K = 0$. The profile log-likelihood is exactly the same, but it does not contain the intercept terms $\zeta_1, \ldots, \zeta_K$.

### 3.3.2 Starting values

The observed data semiparametric log-likelihood is non-convex and can have many local optima. Different starting values will result in convergence to different solutions. To estimate the parameters, we choose multiple starting points and select the local optimum with the greatest likelihood.

Starting values must be provided either for $\{\hat{u}_{ik}\}$ (in which case we start with the M step) or for $(\hat{\beta}, \hat{\pi}, \hat{h})$ (in which case we start with the E step). To randomly generate starting values for $\hat{u}$, we can randomly select a class for each observation to which we assign the majority weight (say 0.8). The remaining weight (0.2) is divided among the remaining classes. This method was suggested in Eng and Hanlon (2014).

Other methods have been proposed for generating EM algorithm starting values for mixture models. For example, Schepers (2015) and McLachlan and Peel (2004) generate starting values for a $K$-class finite mixture of Gaussian regression models by randomly choosing $K$ subsets of the sample and fitting a Gaussian model to each one. Henderson and Rathouz (2015) use a similar method for autoregressive count data. The subsets are chosen to be small enough that the starting values have considerable variation when this method is repeated, thus finding more local optima.

We propose using a similar method for CAC. For each random start, we randomly select $K$ equally sized subsets of the sample and fit a Cox model to each one. This gives starting $\hat{\beta}$ values for each class. We draw subsets without replacement, but this could also be done with replacement. Various methods can be used to initialize $\hat{\pi}$ and $\hat{h}$. A simple options is to assign equal probability to each class and use the full sample Nelson-Aaelen or Kaplan-Meier estimate of $h$.

Smaller subsets will produce more variation in the starting values, which can find more local optima. When selecting the subset size, consideration needs to be given to the number of covariates and the censoring rate. For example, if a subset has fewer uncensored observations than the number of covariates, then coefficient estimates cannot be obtained for that subset. In this case, we can simply draw a different $K$ subsets; however, if this occurs too frequently, then it may be better to increase the subset size.

### 3.3.3 Stopping rule

We assess convergence based on the relative change in the semiparametric log-likelihood between successive iterations. The algorithm is terminated when the relative change falls below a specified threshold. For simulations and analyses in this chapter, we used a convergence threshold of $10^{-8}$.

### 3.3.4 Algorithm summary

Let the $(m)$ superscript denote the $m^{\text{th}}$ iteration of the algorithm. We present the version of the algorithm where we initialize $(\hat{\beta}^{(0)}, \hat{h}^{(0)}, \hat{\pi}^{(0)})$ and begin with the E-step. Alternatively, we could initialize $\hat{u}^{(0)}$ and begin with the M-step.

1. Initialize $(\hat{\pi}^{(0)}, \hat{\beta}^{(0)}, \hat{h}^{(0)})$.
2. Set $m \leftarrow 0$.
3. While $\left|\log \frac{L(\hat{\pi}^{(m)},\hat{\beta}^{(m)},\hat{h}^{(m)};y,\delta|x))-\log L(\hat{\pi}^{(m-1)},\hat{\beta}^{(m-1)},\hat{h}^{(m-1)};y,\delta|x)}{\log L(\hat{\pi}^{(m)},\hat{\beta}^{(m)},\hat{h}^{(m)};y,\delta|x)}\right|$ exceeds the convergence threshold,

   *E-step*

   a) Set $\hat{u}_{ik}^{(m+1)} \leftarrow \mathrm{E}(U_{ik}|Y_i = y_i, \Delta_i = \delta_i, X_i = x_i)$
   $$= \frac{\hat{\pi}_k^{(m)}[\hat{h}_{ik}^{(m)} \exp(x_i^T \hat{\beta}_k^{(m)})]^{\delta_i} \exp\{-\hat{H}_{ik}^{(m)} \exp(x_i^T \hat{\beta}_k^{(m)})\}}{\sum_{k'=1}^K \hat{\pi}_{k'}^{(m)}[\hat{h}_{k'i}^{(m)} \exp(x_i^T \hat{\beta}_{k'}^{(m)})]^{\delta_i} \exp\{-\hat{H}_{k'i}^{(m)} \exp(x_i^T \hat{\beta}_{k'}^{(m)})\}}$$

   *M-step*

   b) Set $\hat{\pi}^{(m+1)} \leftarrow \arg\max_{\pi} \log L_1(\pi; \hat{u}^{(m+1)}) = \left(\frac{1}{N}\sum_{i=1}^N \hat{u}_{i1}^{(m+1)}, \ldots, \frac{1}{N}\sum_{i=1}^N \hat{u}_{iK}^{(m+1)}\right)$.

   c) Set $\hat{\beta}^{(m+1)} \leftarrow \arg\max_{\beta} \log L_2(\beta, h^*(\beta); y, \delta|\hat{u}^{(m+1)}, x)$, which is the optimum of a weighted Cox model profile likelihood.

   d) Set $\hat{h}^{(m+1)} \leftarrow h^*(\hat{\beta}^{(m+1)})$.

   e) Set $\hat{H}_{ik}^{(m+1)} \leftarrow \sum_{j:y_j \le y_i} \hat{h}_{jk}^{(m+1)}$.

   f) Set $m \leftarrow m + 1$.

## 3.4 Standard errors and inference

We would like to estimate standard errors for the parameters of interest. We consider the parameters of interest to be $\theta = (\pi, \beta)$ for the identical and non-proportional models, and $\theta = (\pi, \beta, \zeta)$ for the proportional model. There may be applications where standard errors for the cumulative baseline hazard functions could be of use, but we do not address this here. We propose using the inverse negative Hessian of the profile semiparametric likelihood as an asymptotic variance matrix for $\hat{\theta}$. The Hessian is difficult to obtain analytically, so we consider two numerical approximation methods for latent variable models proposed by Jamshidian and Jennrich (2000).

We define the following notation:

- $L(\theta; y, \delta|x) = \sup_h L(\theta, h; y, \delta|x)$ is the profile semiparametric likelihood.
- $L(\theta; y, \delta, u|x)$ is the complete data profile semiparametric likelihood.
- $Q(\theta', \theta) = \mathrm{E}_\theta[\log L(\theta'; y, \delta, U|x)|Y = y, \Delta = \delta, X = x]$, where $E_\theta$ indicates that the conditional expectation is taken using parameter values $\theta$ (and $h$ equal to the values that optimize the likelihood at $\theta$).
- $\dot{Q}(\theta', \theta)$ and $\ddot{Q}(\theta', \theta)$ are the gradient and Hessian of $Q(\theta', \theta)$ viewed as a function of $\theta'$.
- $M(\theta) = \arg\max_{\theta'} Q(\theta', \theta)$ is the EM algorithm M-step update, resulting when $\theta$ is the current parameter estimate. The function $M(\cdot)$ is known as the EM operator.
- $\dot{M}(\theta)$ is the Jacobian of $M(\theta)$.
- $S(\theta)$ is the score and $H(\theta)$ is the Hessian of $L(\theta; y, \delta|x)$.

## Approach 1: NDS

The first approach uses numerical differentiation of the score function, so Jamshidian and Jennrich (2000) refers to this method as NDS. The score function can be calculated using the Fisher (1925) result

$$S(\theta) = \dot{Q}(\theta, \theta) \ .$$

A proof of this result can be found in Chapter 3 of McLachlan and Krishnan (2007). The quantity $\dot{Q}(\theta, \theta)$ is simply the score function of a weighted Cox model semiparametric likelihood. Its form depends on the baseline hazard assumption (see Section 3.3).

## Approach 2: NDM

The second approach was orginally introduced by Meng and Rubin (1991) as the SEM (supplement EM) algorithm. It uses numerical differential of the EM operator $M(\cdot)$, so

Jamshidian and Jennrich (2000) refers to this method as NDM. The method is based on the Dempster *et al.* (1977) result

$$H(\hat{\theta}) = \ddot{Q}(\hat{\theta}, \hat{\theta})[I - \dot{M}(\hat{\theta})] \, .$$

A proof of this result can be found in Jamshidian and Jennrich (2000). The quantity $\ddot{Q}(\hat{\theta}, \hat{\theta})$ is the Hessian of a weighted Cox model semiparametric log-likelihood. Its form depends on the baseline hazard assumption (see Section3.3). The quantity $\dot{M}(\hat{\theta})$ must be approximated using numerical differentiation.

NDS requires numerical differentiation of the score function, and NDM requires numerical differentiation of the EM operator. We follow the recommendation of Jamshidian and Jennrich (2000) and use a first order Richardson extrapolation. For a function $f(x)$ with scalar argument $x$, a first order Richardson extrapolation of the central difference is

$$\frac{f(x-2\epsilon) - 8f(x-\epsilon) + 8f(x+\epsilon) - f(x+2\epsilon)}{12\epsilon} \, ,$$

where $\epsilon$ is a small positive value. This approximation is applied to each coordinate of the score function or EM operator to obtain the approximate Jacobian. Again following the recommendation of Jamshidian and Jennrich (2000), we use $\epsilon = 10^{-4} \max(|x|, 1)$ for each coordinate. In order to evaluate these functions at points surrounding $\hat{\theta}$, one must first optimize over $h$ at each point. This is an important step because we are working with a profile likelihood.

One minor issue is that these methods produce an approximate Hessian that is not exactly symmetric due approximation error. We prefer the approximation $\frac{1}{2}(H + H^T)$, where $H$ is the approximate Hessian obtained by NDS or NDM.

## 3.5 Simulation experiments

To check consistency of CAC estimators and accuracy of the asymptotic standard errors, we conducted the following simulation. Data sets were generated from a two-class CAC model with parameters $X_i \overset{iid}{\sim} N(0,1)$, $(\beta_1, \beta_2) = (-1, 1)$, $(\pi_1, \pi_2) = (0.35, 0.65)$. The baseline hazard was exponential with mean four in both classes. For each observation, the censoring time was drawn from an independent exponential distribution with mean $20 \times \exp\{-x_i\}$, which resulted in approximately 20% censoring. The simulation was replicated 2,000 times with values of $N$ equal to 100, 500, and 2,500.

For each simulated data set, we fit CAC with identical, proportional, and non-proportional baseline hazards. The assumptions of all three models are met because the data generating mechanism has identical hazards, which is the strongest assumption of the three. Ideally, we would like to find the maximum likelihood estimate for each model, but in practice we can only search for local optima and do not have a way to determine if any of them are the maximum likelihood estimate. Instead, we compare two different estimators under each baseline hazard assumption. One estimator is the local optimum resulting when starting weight one is placed in the true class. We refer to this as the "true start" method. The other estimator is the best of five random starts, where $\hat{\beta}$ is initialized by fitting Cox models fit to two random subsets of 10 observations each, $\hat{\pi}$ is initialized to (0.5, 0.5), and $\hat{h}$ is initialized to the Nelson-Aalen estimate for the full data set. We refer to this as the "random start" estimator.

Note that in order to assess the empirical distribution of an estimator, it is necessary to assign class labels to the parameter estimates. We assign class labels in the order that minimizes the mean squared error of $\hat{\beta}$, which in this case simply amounts to assigning $\hat{\beta}_1$ to be the smaller coefficient estimate and $\hat{\beta}_2$ to be the greater coefficient estimate.

Table 3.1 shows the root mean squared error of each estimator. We see that the average error diminishes as the sample size increases under all three baseline hazard assumptions and under both starting value methods. However, the non-proportional model with random starts has considerably larger error than the others. Even at $N = 2,500$, the average error is still quite large. For all the other models, it appears that the error is going to zero.

Table 3.2 shows the empirical mean of the estimators. Only the non-proportional model with random starts has considerable bias.

Table 3.3 shows the 95% confidence interval coverage rates based on standard errors estimated by the NDM method. For the most part, these intervals are quite accurate for large sample sizes. The non-proportional model with random starts is again the only exception with coverage rates far below 95%.

Further evidence of asymptotic normality for the proportional model with random starts is provided in Figures 3.2 and 3.3. The kernel density estimators in Figure 3.2 show that as the sample size increases, the distribution of each estimator becomes more normal-shaped with decreasing variance. The Q-Q plots in Figure 3.3 show that the distributions are very close to normal with sample size $N = 2,500$.

| | | True weight starts | | | Random starts | | |
|---|---|---|---|---|---|---|---|
| | | I | P | N | I | P | N |
| $\hat{\pi}_1$ | N = 100 | 0.129 | 0.130 | 0.053 | 0.139 | 0.147 | 0.167 |
| | N = 500 | 0.055 | 0.056 | 0.022 | 0.056 | 0.057 | 0.145 |
| | N = 2500 | 0.024 | 0.024 | 0.010 | 0.024 | 0.024 | 0.130 |
| $\hat{\beta}_1$ | N = 100 | 0.57 | 0.59 | 0.32 | 0.61 | 0.63 | 1.17 |
| | N = 500 | 0.20 | 0.21 | 0.12 | 0.20 | 0.21 | 0.64 |
| | N = 2500 | 0.08 | 0.08 | 0.05 | 0.08 | 0.08 | 0.39 |
| $\hat{\beta}_2$ | N = 100 | 0.33 | 0.38 | 0.22 | 0.37 | 1.33 | 2.67 |
| | N = 500 | 0.12 | 0.12 | 0.09 | 0.12 | 0.12 | 1.36 |
| | N = 2500 | 0.05 | 0.05 | 0.04 | 0.05 | 0.05 | 1.07 |
| $\hat{\zeta}_1$ | N = 100 | | 0.73 | | | 2.92 | |
| | N = 500 | | 0.21 | | | 0.21 | |
| | N = 2500 | | 0.09 | | | 0.09 | |

Table 3.1: Root mean squared error of each estimator over on 2,000 simulation replicates. Root mean squared error is defined as $\sqrt{\frac{1}{2000}\sum_{i=1}^{2000}(\hat{\theta}_i-\theta)^2}$, where $\theta$ represents the true value of any of the four parameters and $\hat{\theta}_i$ is its estimator for the $i^{\text{th}}$ simulation replicate. The experiment was repeated for $N \in \{100,\ 500,\ 2500\}$, and CAC was fit under each of the three baseline hazard assumptions: I = Identical; P = Proportional; N = Non-proportional.

| | | True weight starts | | | Random starts | | |
|---|---|---|---|---|---|---|---|
| | | I | P | N | I | P | N |
| $\hat{\pi}_1$ | N = 100 | 0.356 | 0.368 | 0.356 | 0.380 | 0.393 | 0.497 |
| | N = 500 | 0.351 | 0.354 | 0.352 | 0.352 | 0.355 | 0.483 |
| | N = 2500 | 0.349 | 0.350 | 0.350 | 0.350 | 0.350 | 0.474 |
| $\hat{\beta}_1$ | N = 100 | -1.10 | -1.08 | -1.04 | -1.01 | -0.99 | -1.22 |
| | N = 500 | -1.02 | -1.02 | -1.00 | -1.02 | -1.01 | -1.06 |
| | N = 2500 | -1.01 | -1.00 | -1.00 | -1.00 | -1.00 | -0.99 |
| $\hat{\beta}_2$ | N = 100 | 1.06 | 1.11 | 1.04 | 1.08 | 1.19 | 2.87 |
| | N = 500 | 1.01 | 1.01 | 1.01 | 1.01 | 1.02 | 2.14 |
| | N = 2500 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.98 |
| $\hat{\zeta}_1$ | N = 100 | | 0.09 | | | 0.12 | |
| | N = 500 | | 0.01 | | | 0.01 | |
| | N = 2500 | | 0.00 | | | 0.00 | |

Table 3.2: Empirical mean of each estimator over 2,000 simulation replicates. The experiment was repeated for $N \in \{100,\ 500,\ 2500\}$, and CAC was fit under each of the three baseline hazard assumptions: I = Identical; P = Proportional; N = Non-proportional. True parameter values are $(\pi_1, \beta_1, \beta_2, \zeta_1) = (0.35, -1, 1, 0)$.

| | | True weight starts | | | Random starts | | |
|---|---|---|---|---|---|---|---|
| | | I | P | N | I | P | N |
| $\hat{\pi}_1$ | N = 100 | 0.881 | 0.869 | 0.949 | 0.879 | 0.852 | 0.332 |
| | N = 500 | 0.939 | 0.937 | 0.957 | 0.938 | 0.935 | 0.052 |
| | N = 2500 | 0.949 | 0.948 | 0.946 | 0.945 | 0.946 | 0.001 |
| $\hat{\beta}_1$ | N = 100 | 0.867 | 0.857 | 0.963 | 0.813 | 0.794 | 0.378 |
| | N = 500 | 0.910 | 0.913 | 0.957 | 0.907 | 0.908 | 0.274 |
| | N = 2500 | 0.939 | 0.944 | 0.952 | 0.937 | 0.942 | 0.188 |
| $\hat{\beta}_2$ | N = 100 | 0.919 | 0.906 | 0.951 | 0.919 | 0.898 | 0.201 |
| | N = 500 | 0.944 | 0.941 | 0.951 | 0.943 | 0.941 | 0.057 |
| | N = 2500 | 0.943 | 0.940 | 0.944 | 0.941 | 0.939 | 0.001 |
| $\hat{\zeta}_1$ | N = 100 | | 0.912 | | | 0.904 | |
| | N = 500 | | 0.945 | | | 0.945 | |
| | N = 2500 | | 0.940 | | | 0.940 | |

Table 3.3: NDM 95% confidence interval coverage rates over 2,000 simulation replicates. The experiment was repeated for $N \in \{100,\ 500,\ 2500\}$, and CAC was fit under each of the three baseline hazard assumptions: I = Identical; P = Proportional; N = Non-proportional. NDS confidence interval coverage was nearly identical.

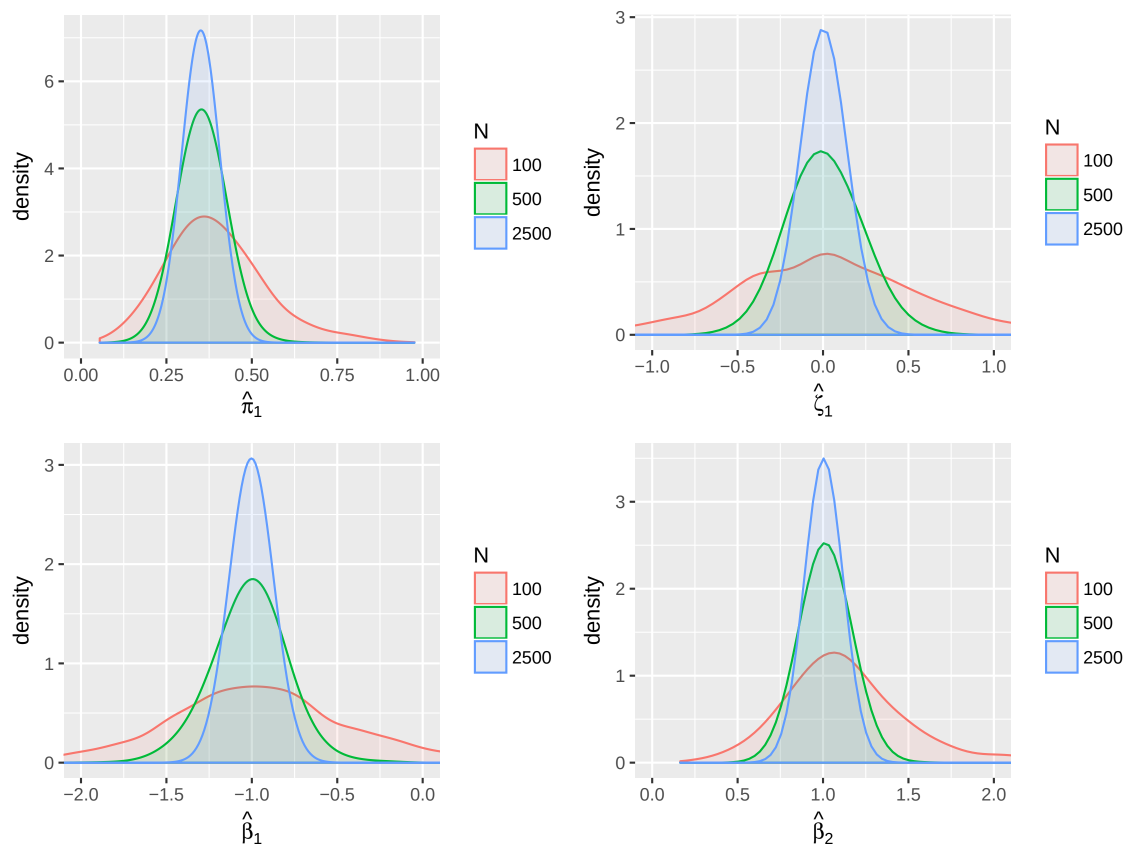

Figure 3.2: Smoothed histograms of the empirical distribution for each parameter estimator under the proportional baseline hazard assumption with random starts.

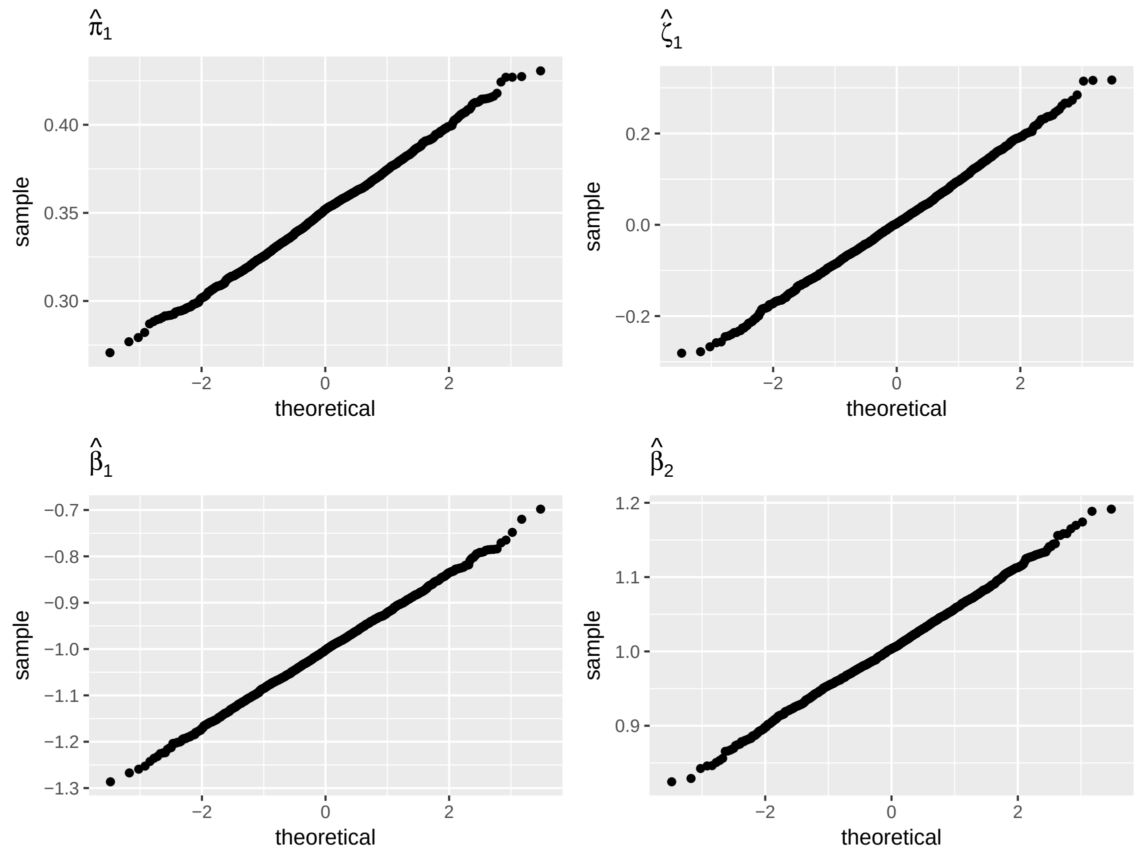

Figure 3.3: Q-Q plot of the empirical distribution for each parameter under the proportional baseline hazard assumption with random starts and sample size $N = 2,500$.

To better understand these results, we examine the true and random start estimators more closely. Table 3.4 shows the number of simulation replicates where either the true or random start estimator has a greater log-likelihood than the other. If one estimator has greater log-likelihood, then the other estimator clearly is not the MLE for that simulation replicate. It turns out that for the identical and proportional models, the two estimators nearly always coincide. The estimators coincide more often as the sample size increases, and for $N = 2,500$ they matched for all 2,000 replicates. For the non-proportional model, the random start estimate essentially always has a greater log-likelihood than the true start estimate. This is despite the fact that the random start estimate has much larger mean squared error.

| | Identical | | Proportional | | Non-proportional | |
|---|---|---|---|---|---|---|
| | True | Random | True | Random | True | Random |
| N = 100 | 86 | 49 | 83 | 96 | 0 | 1999 |
| N = 500 | 2 | 3 | 6 | 5 | 0 | 2000 |
| N = 2500 | 0 | 0 | 0 | 0 | 0 | 2000 |

Table 3.4: This table shows the number of simulation replicates out of 2,000 where true weight starts and random starts converged to solutions with different log-likelihood values. Let $\ell_t$ and $\ell_r$ denote the log-likelihood of the solution obtained by true weight starts and random starts, respectively. The "True" column shows the number of replicates where $(\ell_t - \ell_r)/|\ell_t| > 10^{-6}$. The "Random" column shows the number of replicates where $(\ell_r - \ell_t)/|\ell_t| > 10^{-6}$. For the remaining replicates, $|\ell_r - \ell_t|/|\ell_t| < 10^{-6}$.

Recall that for the random start estimators, we chose the EM algorithm solution with the greatest log-likelihood among five random starts. In Table 3.5, we explore how often the best solution of the five is matched by multiple random starts. When multiple random starts converge to the best solution, this provides evidence that this may be the MLE. Conversely, if the five random starts converge to five different solutions, then it seems likely that the MLE may not be among them. We see that for the identical and proportional models, the best solution is nearly always matched by multiple starts, especially for large sample sizes. The non-proportional model exhibits the opposite pattern: as the sample size increases it becomes more likely that the best solution is not matched by multiple random starts.

| | Identical | | | | | Proportional | | | | | Non-proportional | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 5 |
| N = 100 | 60 | 57 | 62 | 121 | 1699 | 62 | 48 | 37 | 97 | 1755 | 820 | 280 | 252 | 322 | 325 |
| N = 500 | 2 | 1 | 1 | 21 | 1975 | 3 | 1 | 2 | 23 | 1971 | 1393 | 328 | 159 | 88 | 32 |
| N = 2500 | 0 | 0 | 0 | 22 | 1978 | 0 | 0 | 0 | 13 | 1987 | 1731 | 204 | 54 | 10 | 1 |

Table 3.5: Tabulation of the number of random starts that converged to the same solution. For each simulation replicate, the solution with the best log-likelihood among five random starts was selected. In many cases, the best log-likelihood was reached by multiple random starts (up to five). This table stratifies the 2,000 replicates according to the number of random starts that converged to the best log-likelihood. This was done for each baseline hazard assumption and sample size. Let $\ell_r$ denote the maximum log-likelihood among the five random starts for a given replicate. For $i = 1, \ldots, 5$, the log-likelihood $\ell_i$ of the $i^{\text{th}}$ random start was considered to match $\ell_r$ if $|\ell_i - \ell_r|/|\ell_r| < 10^{-6}$.

In summary, these simulation results suggest that estimators under the identical and proportional baseline hazard assumption are asymptotically normal, and that the NDM and NDS methods provide accurate estimates of the standard errors. The fact that the true and random starting value estimators usually coincide suggests that the estimate obtained by both methods could often be the MLE.

Under the non-proportional baseline hazard assumption, it appears that the true start estimator is asymptotically normal; however, this estimator clearly does not coincide with the MLE as evidenced by the fact that the random start estimator nearly always has greater log-likelihood. Despite having greater log-likelihood, the random start estimator has much larger mean squared error and bias. Because the estimator with greater log-likelihood is not better, it would appear that the MLE may not be a good estimator for this model. It could also be the case that the random start method does not reliably converge to the MLE, but the non-proportional model is problematic regardless of the explanation because the true start estimator cannot be used in applications.

## 3.6 Comparison of random starting value methods

We compare the "random weight" and "random subset" starting value methods by plotting the local optima found by each method. Of particular interest is the number of unique local optima each method found and whether finding more local optima increases the chance of finding the maximum likelihood solution, which lies near the true coefficient values.

A single data set was generated for Figures 3.4, 3.5, and 3.6. The data was generated from a two-class CAC model with parameters $N = 100$, $X_i \overset{iid}{\sim} N(0,1)$, $(\hat{\beta}_1, \hat{\beta}_2) = (-2, 2)$, $(\hat{\pi}_1, \hat{\pi}_2) = (0.2, 0.8)$. The censoring time for each observation was drawn from an independent exponential distribution with mean one, which resulted in 54 censored observations. The baseline hazard in all classes was exponential with mean one. This data set is meant to illustrate an example where the random start methods produce different results. Obviously, the results could vary if this experiment were repeated. The simulation parameters were selected such that there is strong separation between classes, and the maximum likelihood solution is near the true parameters. However, the class are imbalanced, making it difficult to detect the class separation by random 80%/20% starting weights.

We fit CAC using all three baseline hazard assumptions: identical, proportional, and non-proportional. The assumptions of all three models are met because the data generating mechanism has identical hazards, which is the strongest assumption of the three. For each of these assumptions, we fit CAC using four starting value methods.

1. The “random weight (80-20)” method randomly assigns 80% weight to one class and 20% weight to the other for each observation.

2. The “random subset” method initializes $\hat{\beta}$ by fitting Cox models fit to two random subsets of 50 observations each. For each start, $\hat{\pi}$ is initialized to (0.5, 0.5), and $\hat{h}$ is initialized to the Nelson-Aalen estimate for the full data set.

3. Same as 2, but with subsets of size 25.

4. Same as 2, but with subsets of size 10.

For each starting method, we used 50 random starts resulting in 50 local optima. Each local optimum is plotted with a block dot. Not all local optima are unique, and in fact there is considerable overlap. The solution with the highest likelihood is marked with a red triangle. On each plot, there is also a blue square marking the model fit resulting from the “true weight” start, which places a starting weight of one in the true class. For all three baseline hazard assumptions, the “true weight” solution lies near the true coefficient values.

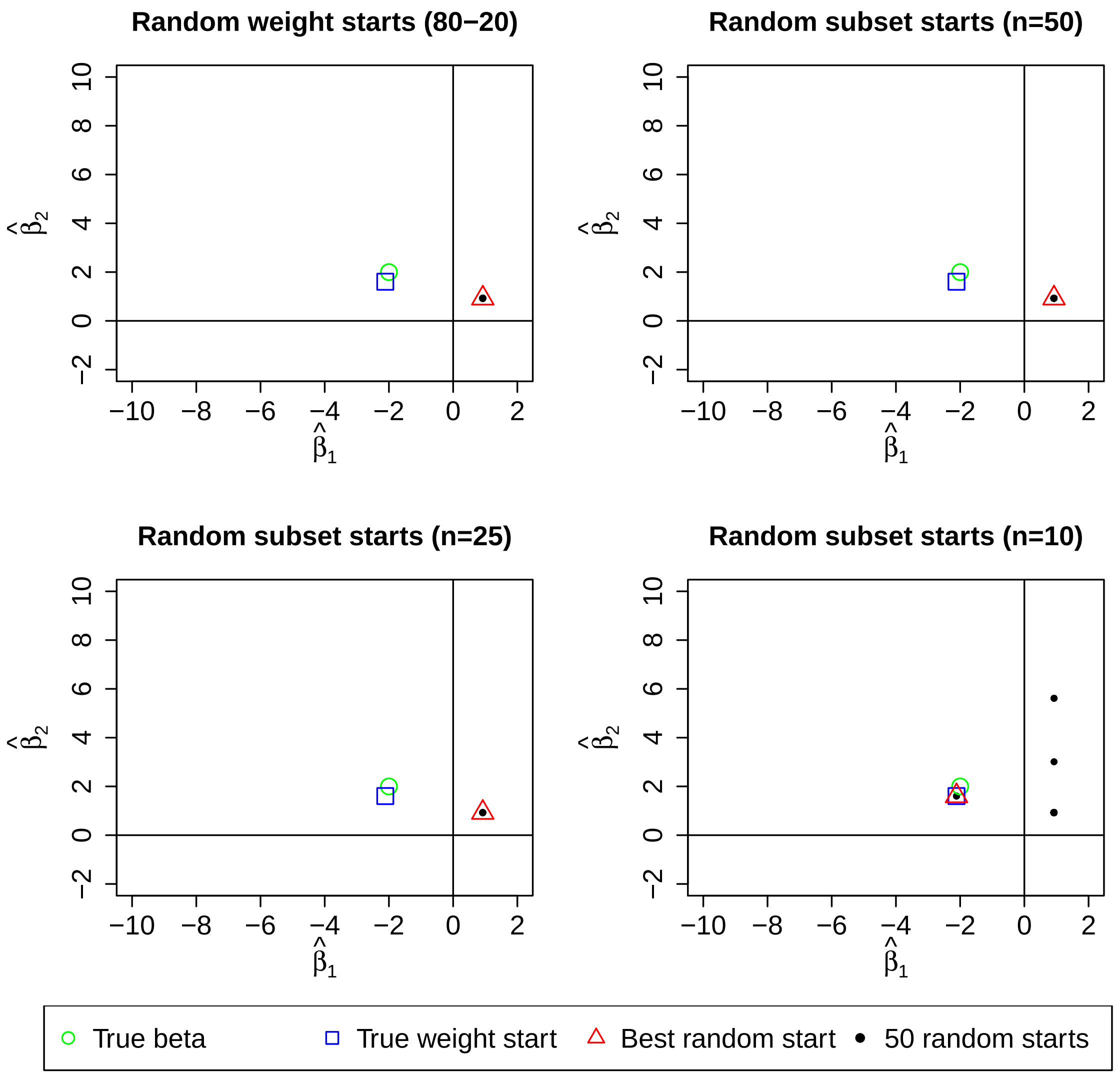

Figure 3.4: Identical baseline hazards.

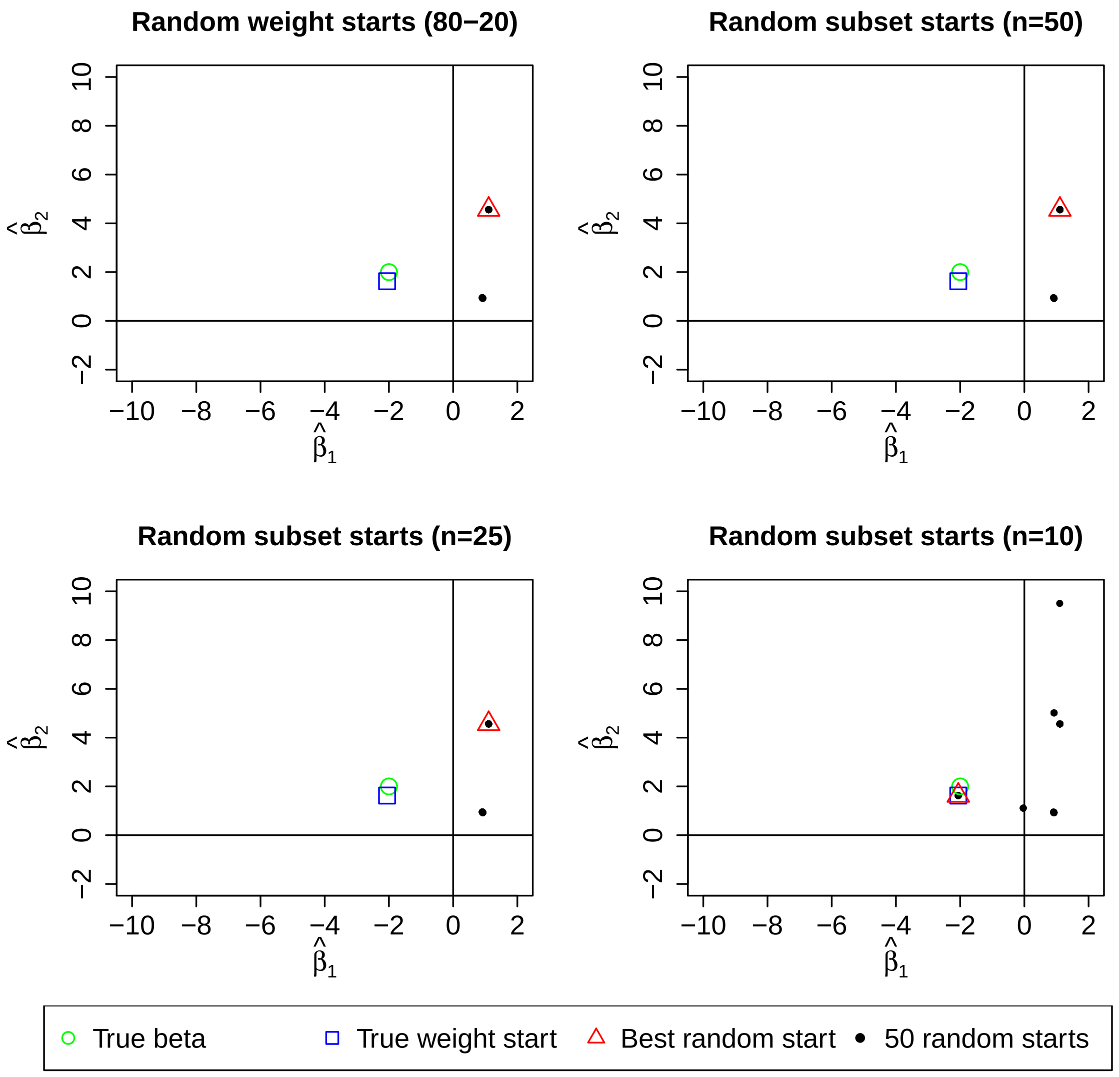

Figure 3.5: Proportional baseline hazards.

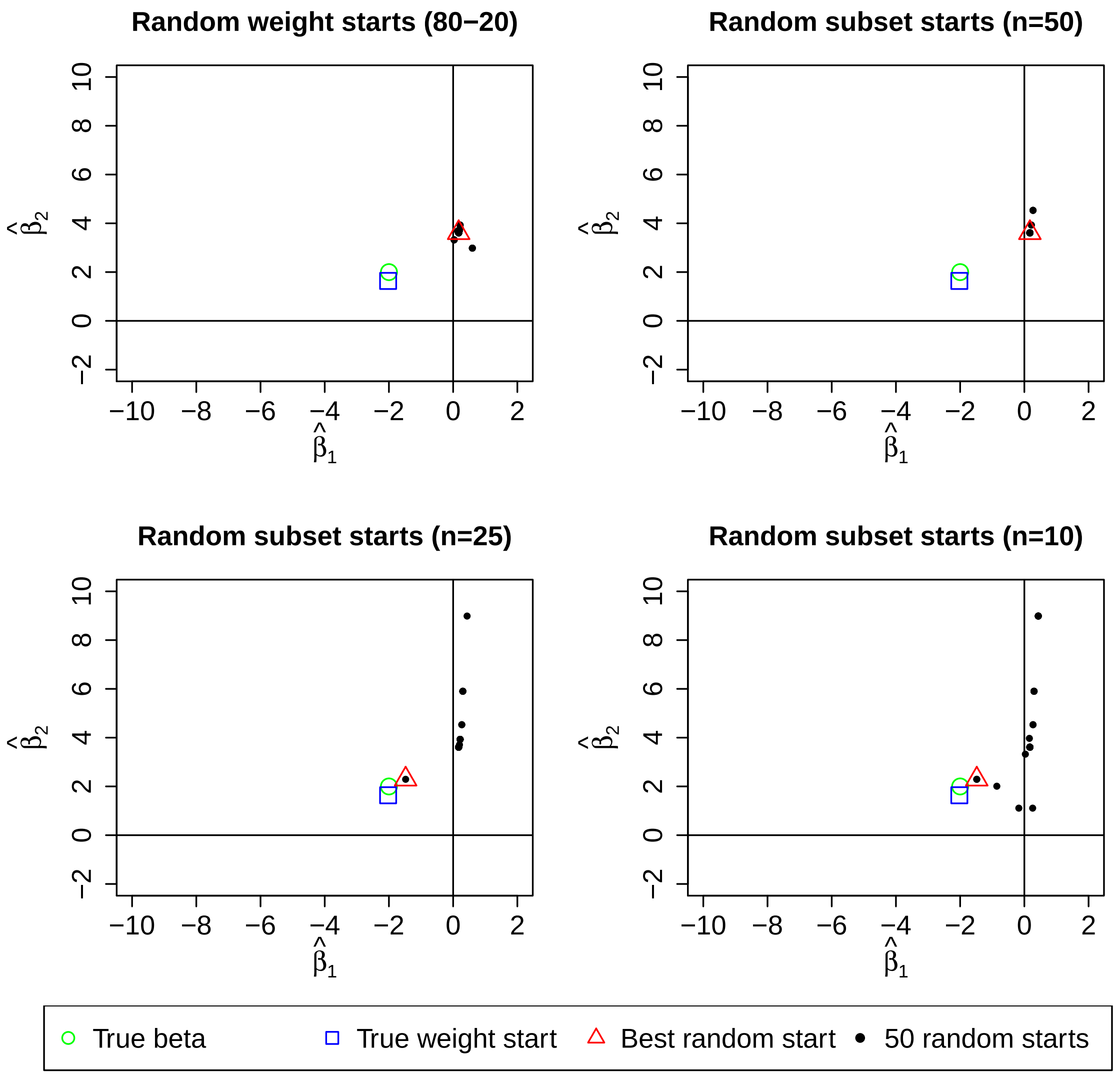

Figure 3.6: Non-proportional baseline hazards.

## 3.7 Application to TCGA data

Returning to the TCGA ovarian cancer example from Eng and Hanlon (2014), we model overall survival as a function of epiregulin expression level. There are 276 events 227 censored observtations, for a total of 503 observations (Table 3.6). We center and scale epiregulin expression levels to have mean zero and standard deviation one. By fitting separate Cox models to the optimal and suboptimal cytoreduction subpopulations, Eng and Hanlon (2014) showed that increased epiregulin expression appears to have a detrimental effect for optimally cytoreduced patients ($\hat{\beta} = 0.156$, $P = 0.014$), but a protective effect for suboptimally cytoreduced patients ($\hat{\beta} = -0.452$, $P = 0.012$).

| | Suboptimal | Optimal |
|---|---|---|
| Censored | 72 | 155 |
| Event | 97 | 179 |

Table 3.6: Censoring distribution of patients in each cytoreduction class.

Now suppose we did not have cytoreduction class information available. We can check to see if CAC is able to detect the differential effect of epiregulin across the latent cytoreduction classes. We fit CAC using three baseline hazard assumptions: identical, proportional, and non-proportional. For each of these assumptions, we use two starting value methods:

1. Starting weight of one in the true cytoreduction class.
2. $\hat{\beta}$ initialized by Cox models fit to two random subsets of ten subjects each. $\hat{\pi}$ initialized to (0.5, 0.5). $\hat{h}$ initialized to the Nelson-Aalen estimate for the full data set. The model fit with the greatest semiparametric log-likelihood is selected from 50 random starts.

Under each hazard restriction, we are looking for two things. First, we use the model fit resulting from true starting weights to determine whether the semiparametric likelihood has a local maximum near the coefficient estimates from separate Cox model fits in each class. If a close solution is not found with these starting values, then it is unlikely that one exists. If a close solution exists, then we check if this solution can be found using random starting values and if it has the highest likelihood among local optima.

It should be emphasized that the ground truth is unknown, and this is purely an illustrative example. Even if CAC is a good model, we do not know the true number of classes because there could be other latent variables besides cytoreduction. We chose this particular example because there is fairly strong empirical evidence that the epiregulin effect differs by cytoreduction class. We do not claim that the results would generalize to a population outside the TCGA sample.

The results are summarized in Table 3.7. Under the identical hazard restriction, no solution exists near the separated class coefficients. With both true and random starts, the coefficient estimates are positive in both classes. Under the proportional hazard restriction, a solution does exist near the separated class coefficients, and this is also the best solution found by the random start method. With non-proportional hazards (i.e. no restriction), there is a local optimum near the separated class coefficients, but this is not the best local optimum. This suggests that the proportional hazard CAC model may be the best model for this data set, because it is the only model that both has a local solution near the separated class coefficients and selects that solution using random starts.

| | Suboptimal | Optimal | Log-likelihood |
|---:|---:|---:|---:|
| Separate Cox model fits | -0.45 | 0.16 | NA |
| Identical: true start | 0.03 | 0.09 | -1676.14 |
| Identical: random start | 0.01 | 0.18 | -1676.12 |
| Proportional: true start | -0.24 | 0.25 | -1675.30 |
| Proportional: random start | -0.24 | 0.25 | -1675.30 |
| Non-proportional: true start | -0.34 | 0.17 | -1674.92 |
| Non-proportional: random start | -6.58 | 0.23 | -1602.99 |

Table 3.7: Epiregulin coefficient estimates for each cytoreduction class. Epiregulin expression values are standardized to have mean zero and standard deviation one. "True start" refers to the model fit resulting when each patient is assigned starting weight of one the true cytoreduction class, and "random start" refers to the model fit resulting when the best fit of 50 random starts is selected.

## 3.8 Future work

As CAC is formulated here and in Eng and Hanlon (2014), it is assumed that class probabilities do not depend on the covariates. This may not reflect reality in some applications. Furthermore, CAC cannot be used to predict the class of an out-of-sample observation unless the survival time is already known. This may not be much help in certain applications, such as cancer genomics, where it would be useful to predict a patient's class without this information. If class probabilities vary, conditional on the covariates, then it would be possible to make better class predictions without information about survival time. This would require a model for the conditional class probabilities, such as a multinomial regression model. To the best of our knowledge there is not work done on this topic. Khalili and Chen (2007) write that their model can be generalized to allow class probabilities to depend on covariates, but they don't pursue it in that paper.

It may be beneficial to explore methods for regularization and variable selection within the CAC model framework. We expect that the lasso and sparse group lasso penalties could improve model performance in settings with more covariates relative to the sample size. Städler *et al.* (2010) and Khalili and Chen (2007) have shown that the lasso penalty, with some modification, produces good results in the case of Gaussian finite mixture of regression models. Perhaps these results could be extended to CAC.

It might be interesting to explore methods for inference that do not involve numerical differentiation. Louis (1982) proposed a method to formulate the Hessian of the observed data likelihood using the complete data likelihood. This method requires the conditional expectation of the complete data Hessian and score outer product. We did not find this method to be convenient for CAC because the complete data Hessian and score outer product are complicated functions of the latent class indicators. In contrast, the NDS and NDM methods, which we suggest for CAC, use the score and Hessian of the expected complete data log-likelihood (the expectation is taken before differentiation). We found this to be much more convenient for the CAC profile likelihood.

The CAC non-profile complete data semiparametric likelihood, on the other hand, is linear in the latent class indicators, which would be more convenient for application of the Louis (1982) method. In addition to avoiding numerical differentiation, this method may have the potential to provide standard errors for the cumulative baseline hazard functions. It has been shown in the context of Cox frailty models that the Hessian of the non-profile semiparametric likelihood can be used for baseline hazard inference (Parner, 1998; Andersen *et al.*, 1997; Murphy, 1995). Perhaps this could hold true for CAC as well. A disadvantage of working with the non-profile semiparametric likelihood is the high dimension of the information matrix, which must be inverted to obtain the asymptotic variance matrix. The number dimension of this matrix scales with the number of unique failure times, which is a serious computational limitation.

It may also be worthwhile exploring how sensitive CAC estimation is to the assumptions in Section 3.2. In particular, the assumption that censoring time is independent of class, conditional on $X$ may not be critical for estimation. The following simulation results suggest that CAC estimators may still be consistent when this assumption is violated; however, the standard error estimates proposed in Section 3.4 are no longer correct. This simulation was conducted exactly as in Section 3.5, except that for each observation, the censoring time was drawn from an independent exponential distribution with mean $20 \times \exp\{-x_i^T \sum_{k=1}^{K} I(U_i = k)\beta_k\}$. This still resulted resulted in approximately 20% censoring, but the censoring time is no longer independent of the latent class. Table 3.8 shows that all estimators appear to be consistent, but Table 3.9 shows that 95% confidence interval coverage is well below 95%, even at $N = 2,500$.

| | | True weight starts | | | Random starts | |
|---|---|---|---|---|---|---|
| | | I | P | N | I | P |
| $\hat{\pi}_1$ | N = 100 | 0.120 | 0.120 | 0.051 | 0.125 | 0.126 |
| | N = 500 | 0.045 | 0.046 | 0.023 | 0.046 | 0.046 |
| | N = 2500 | 0.023 | 0.023 | 0.011 | 0.023 | 0.024 |
| $\hat{\beta}_1$ | N = 100 | 0.46 | 0.49 | 0.33 | 0.46 | 0.52 |
| | N = 500 | 0.15 | 0.16 | 0.14 | 0.15 | 0.16 |
| | N = 2500 | 0.07 | 0.07 | 0.09 | 0.07 | 0.07 |
| $\hat{\beta}_2$ | N = 100 | 0.33 | 0.35 | 0.22 | 0.34 | 0.37 |
| | N = 500 | 0.12 | 0.12 | 0.10 | 0.12 | 0.12 |
| | N = 2500 | 0.07 | 0.07 | 0.06 | 0.07 | 0.07 |
| $\hat{\zeta}_1$ | N = 100 | | 0.64 | | | 0.72 |
| | N = 500 | | 0.17 | | | 0.17 |
| | N = 2500 | | 0.08 | | | 0.08 |

Table 3.8: Root mean squared error of each estimator over on 2,000 simulation replicates. Root mean squared error is defined as $\sqrt{\frac{1}{2000}\sum_{i=1}^{2000}(\hat{\theta}_i - \theta)^2}$, where $\theta$ represents the true value of any of the four parameters and $\hat{\theta}_i$ is its estimator for the $i^{\text{th}}$ simulation replicate. The experiment was repeated for $N \in \{100,\ 500,\ 2500\}$, and CAC was fit under each of the three baseline hazard assumptions: I = Identical; P = Proportional; N = Non-proportional. This is a repeat of the simulation study in Section 3.5, except the censoring distribution depends on the latent class.

| | | True weight starts | | | Random starts | |
|---|---|---|---|---|---|---|
| | | I | P | N | I | P |
| | N = 100 | 0.892 | 0.881 | 0.945 | 0.888 | 0.875 |
| $\hat{\pi}_1$ | N = 500 | 0.944 | 0.941 | 0.945 | 0.944 | 0.939 |
| | N = 2500 | 0.922 | 0.917 | 0.927 | 0.916 | 0.910 |
| | N = 100 | 0.878 | 0.849 | 0.945 | 0.857 | 0.824 |
| $\hat{\beta}_1$ | N = 500 | 0.912 | 0.908 | 0.896 | 0.912 | 0.906 |
| | N = 2500 | 0.877 | 0.891 | 0.678 | 0.877 | 0.892 |
| | N = 100 | 0.897 | 0.879 | 0.940 | 0.896 | 0.877 |
| $\hat{\beta}_2$ | N = 500 | 0.907 | 0.901 | 0.901 | 0.907 | 0.899 |
| | N = 2500 | 0.802 | 0.792 | 0.729 | 0.796 | 0.784 |
| | N = 100 | | 0.905 | | | 0.898 |
| $\hat{\zeta}_1$ | N = 500 | | 0.940 | | | 0.939 |
| | N = 2500 | | 0.918 | | | 0.916 |

Table 3.9: NDM 95% confidence interval coverage rates over 2,000 simulation replicates. The experiment was repeated for $N \in \{100,\ 500,\ 2500\}$, and CAC was fit under each of the three baseline hazard assumptions: I = Identical; P = Proportional; N = Non-proportional. NDS confidence interval coverage was nearly identical. This is a repeat of the simulation study in Section 3.5, except the censoring distribution depends on the latent class.

## 3.9 Discussion

We proposed several innovations to CAC, including restrictions on the baseline hazard that can improve model identifiability, a method for generating random EM algorithm starting values that can find more local optima of the likelihood, and a method for estimating standard errors.

Perhaps most significant was the finding that without baseline hazard restrictions, the CAC estimators can have considerable bias and mean squared error. A simulation study demonstrated that this occurs even with a large sample size. The simulation suggested that without baseline hazard restrictions, the likelihood has a local optimum that is asymptotically normal, but we currently can only identify this solution when the class of each observation is known, which is not possible outside of simulation studies. Imposing baseline hazard restrictions allows us to find a good estimator via the EM algorithm without knowing the class of each observation. The simulation demonstrated that this estimator has good finite sample behavior and standard errors which we can accurately estimate.

We proposed two methods to estimate standard errors based on work by Jamshidian and Jennrich (2000). Both methods numerically approximate the Hessian of the profile semiparametric log-likelihood.

We also proposed a method to obtain starting values for the EM algorithm optimization algorithm. This method generates random sets of starting values that have much more variability than the method originally proposed. This allows the EM algorithm to find more local optima of the log-likelihood and potentially a better estimator. We showed an example where the proposed method greatly improves the chance of finding a good estimator.

Future work might involve regularization and variable selection, as well development of a model that allows class probabilities to depend on the covariates. These steps have the potential to open a wide range of applications for CAC and help researchers to derive meaningful insights from this model.

# 4 SEMIPARAMETRIC GENERALIZED LINEAR MODELS AND THE GLDRM R PACKAGE

## 4.1 Introduction

Rathouz and Gao (2009) introduced the generalized linear density ratio model (gldrm), which is a novel semiparametric formulation of the classical glm. Although Rathouz and Gao did not use the term gldrm, we refer to it as such because it is a natural extension of the density ratio model (see e.g. Lovric (2011)). Like a standard glm, the gldrm relates the conditional mean of the response to a linear function of the predictors through a known link function $g$. To be specific, let the random variable $Y$ be a scalar response, and let $X$ be a $p \times 1$ covariate vector for a particular observation. The model for the conditional mean is

$$\mathrm{E}(Y|X = x) = g^{-1}(x^T\beta) \ , \tag{4.1}$$

Because Equation 4.1 holds for both gldrm and glm, the regression coefficients have the same interpretation for both models. The gldrm relaxes the standard glm assumption that the distribution of $Y|X$ comes from a particular exponential family model. Instead, assume that $Y|X$ comes from an exponential tilt model of the form

$$f(y|X = x) = f_0(y)\exp\{\theta y - b(\theta)\} \ , \tag{4.2}$$

where

$$b(\theta) = \log \int f_0(y)\exp(\theta y)\, d\lambda(y) \ , \tag{4.3}$$

and $\theta$ is defined implicitly as the solution to

$$g^{-1}(x^T\beta) = \int y \exp\{\theta y - b(\theta)\}\, d\lambda(y) \ . \tag{4.4}$$

Here $f_0$ is an unspecified probability density with respect to a measure $\lambda$. We call $f_0$ the *reference distribution*. Measure $\lambda$ is Lebesgue if $Y|X$ is continuous, a counting measure if $Y|X$ is discrete, or a mixture of the two if $Y|X$ has a mixture distribution. Note that $\mathrm{E}(Y|X) = b'(\theta)$ and $\mathrm{Var}(Y|X) = b''(\theta)$, which are standard glm properties.

The regression coefficients $\beta$ and reference distribution $f_0$ are estimated by maximizing a semiparametric likelihood function, which contains a nonparametric representation of

$f_0$ that has point mass only at values of $Y$ observed in the data. The quantity $\theta$ is not a parameter in this model, but rather a function of the free parameters $\beta$ and $f_0$, as well as of $X$. This work was fully developed by Huang and Rathouz (2012), who first focused on the case where the covariate vector takes one of finitely many values, drawing on theoretical arguments advanced in the empirical likelihood literature (e.g. Owen *et al.* (2001)). Drawing on semiparametric profile likelihood methods Huang (2014) went on to fully generalize the asymptotic arguments, proving consistency and asymptotic normality of the regression coefficient estimators, and deriving the asymptotic variance matrix using the profile likelihood. Huang also proved pointwise asymptotic normality of the reference cumulative distribution function estimator.

Despite these important theoretical advances, computation for this model has remained a practical challenge. The original algorithm in Rathouz and Gao (2009) is somewhat rudimentary and applies mostly to cases with finite support. It does not scale well, and stability and speed are challenges. This paper proposes a new algorithm to address these challenges and render the model more widely applicable.

In particular, the issue of optimizing the semiparametric likelihood over the $f_0$ point masses is improved with application of the Broyden-Fletcher-Goldfarb-Shanno (BFGS) technique (Broyden, 1970; Fletcher, 1970; Goldfarb, 1970; Shanno, 1970). Because the number of parameters in the semiparametric likelihood associated with $f_0$ is equal to the number of unique points in the response vector, the rank of the Hessian matrix can become unwieldy, especially for continuous response data. The BFGS method uses an easily-computed rank-two approximate Hessian matrix that has an inverse which is much less computationally intensive to calculate.

The optimization techniques in this paper have been incorporated into the R package **gldrm** (Wurm and Rathouz, 2017), the first CRAN package for gldrm. This package estimates the gldrm model and provides coefficient standard errors. In addition, since the publication of Rathouz and Gao (2009), Huang (2014) developed likelihood ratio tests and likelihood ratio-based confidence intervals for regression coefficients and demonstrated their strong properties. The **gldrm** package provides utilities for likelihood ratio tests of nested models, as well as confidence intervals.

We present our optimization algorithm in Section 4.2. In Section 4.3, we present simulation experiments to check the finite sample performance of the gldrm estimators and the accuracy of estimated standard errors using this algorithm. In Section 4.3, we discuss methods for inference. Section 4.5, we present a simulation that benchmarks the computational time and how it scales with the number of observations and number of covariates. In Section 4.6, we present a summary of the functions in the **gldrm** package. In Section 4.7, we demonstrate the functionality of the **gldrm** package with an example in R. In Section 4.8, we make concluding remarks.

## 4.2 Optimization algorithm

Suppose we have an observed sample $(x_1, y_1), \ldots, (x_n, y_n)$ generated by the model specified in Equations 4.1-4.4. The $x_i$ can be fixed or random, and the $y_i$ are conditionally independent given the covariates. Let $\mathcal{S} = (s_1, \ldots, s_K)$ be the observed support, i.e. a vector containing the unique values in the set $\{y_1, \ldots, y_n\}$. If the response follows a continuous distribution, then $K$ will equal $n$. Otherwise $K$ may be less than $n$ due to ties. Let the vector $\tilde{f}_0 = (f_1, \ldots, f_K)$ represent probability mass at the points $(s_1, \ldots, s_K)$. This is a nonparametric representation of $f_0$ because the true reference density may be continuous or have probability mass at values not contained in $\mathcal{S}$.

The semiparametric log-likelihood is

$$\ell(\beta, \tilde{f}_0) = \sum_{i=1}^{n} \left\{ \theta_i y_i - \log \sum_{k=1}^{K} f_k \exp(\theta_i s_k) + \sum_{k=1}^{K} I(y_i = s_k) \log f_k \right\} , \tag{4.5}$$

where each $\theta_i$ is defined implicitly as the solution to

$$g^{-1}(x_i^T \beta) = \frac{\sum\limits_{k=1}^{K} s_k f_k \exp\{\theta_i s_k\}}{\sum\limits_{k=1}^{K} f_k \exp\{\theta_i s_k\}} . \tag{4.6}$$

There exists a $\theta_i$ that satisfies Equation 4.6 as long as $g^{-1}(x_i^T \beta) \in (m, M)$ for all $i$, where $m \equiv \min(\mathcal{S})$ and $M \equiv \max(\mathcal{S})$.

We obtain parameter estimates from this likelihood as $\underset{(\beta, \tilde{f}_0)}{\arg\max}\, \ell(\beta, \tilde{f}_0)$, subject to the following constraints:

C1. $g^{-1}(x_i^T \beta) \in (m, M)$ for all $i$

C2. $f_k > 0$ for all $k = 1, \ldots, K$

C3. $\sum_{k=1}^{K} f_k = 1$

C4. $\sum_{k=1}^{K} s_k f_k = \mu_0$ for some chosen $\mu_0 \in (m, M)$

Constraint (C4) is necessary for identifiability because for any nonparametric density $\tilde{f}_0 = (f_1, \ldots, f_k)^T$, the "exponentially tilted" density $\tilde{f}_0^{(\alpha)} = (f_1 e^{\alpha s_1}, \ldots, f_K e^{\alpha s_K})^T / \sum_{k=1}^{K} f_k e^{\alpha s_k}$ has the same semiparametric log-likelihood for any $\alpha \in \mathbb{R}$, i.e. $\ell(\beta, \tilde{f}_0) = \ell(\beta, \tilde{f}_0^{(\alpha)})$ for all $\beta$. We can set $\mu_0$ to be any value within the range of observed response values, but choosing a value too extreme can lead to numerical instability in estimating $\tilde{f}_0$. It usually works well to choose $\mu_0 = \frac{1}{n} \sum_{i=1}^{n} y_i$.

To perform the optimization, we take an iterative approach, alternating between optimizing over $\tilde{f}_0$ and over $\beta$ (Rathouz and Gao, 2009). Each optimization step marginally optimizes the log-likelihood over one set of parameters, holding the other fixed. Neither optimization step has a closed form solution, so iterative procedures must be used. To update $\tilde{f}_0$, we propose using the BFGS technique. To update $\beta$, we use Fisher scoring, which is equivalent to iteratively re-weighted least squares (IRLS). We propose iterating BFGS until convergence for each $\tilde{f}_0$ update, while only using a single IRLS iteration to update $\beta$ in between $\tilde{f}_0$ updates; this is tantamount to a profile likelihood approach to estimation of $\beta$.

Although the log-likelihood in Equation 4.5 is a function of $(\beta, \tilde{f}_0)$, we have expressed it in terms of $(\theta_1, \ldots, \theta_n)$; each $\theta_i$ is implicitly a function of $(\beta, \tilde{f}_0)$. The score function is also best expressed in terms of the $\theta_i$'s (see Equations 4.12 and 4.15). Consequently, our optimization algorithm requires the $\theta_i$'s to be computed each time $\beta$ or $\tilde{f}_0$ is updated. To do this, we use an iterative Newton-Raphson procedure after each iteration of the $\beta$ update or the $\tilde{f}_0$ update procedures.

In what follows, we detail updates of the $\theta_i$'s, of $\tilde{f}_0$, and of $\beta$. We use the notation $b(\theta)$ to denote the function defined in Equation 4.4 with respect to the discrete probability distribution specified by $\tilde{f}_0$, i.e. $b(\theta) = \log \sum_{k=1}^{K} f_k \exp(\theta s_k)$. We also define $\mu_i \equiv g^{-1}(x_i^T \beta)$.

### 4.2.1 $\theta$ update procedure

$\theta_i$ is defined implicitly as the solution to Equation 4.6, which can be written as

$$\mu_i = b'(\theta_i) \ . \tag{4.7}$$

To calculate $\theta_i$, we use the Newton-Raphson procedure provided in Appendix C of Rathouz and Gao (2009). To satisfy equation 4.7, we need to find $\theta$ such that $\mu_i = b'(\theta)$, or equivalently $g_l(\mu_i) = g_l\{b'(\theta)\}$, where $g_l(s) = \text{logit}\left(\frac{s-m}{M-m}\right) = \log\left(\frac{s-m}{M-s}\right)$. (The logit transformation stabilizes the solution.)

We use Newton-Raphson to find the root of $t(\theta) = g_l\{b'(\theta)\} - g_l(\mu_i)$. Let $\theta^{(r)}$ denote the approximate solution at the $r^{\text{th}}$ Newton-Raphson iteration. The Newton-Raphson update is given by

$$\theta^{(r+1)} = \theta^{(r)} - \left\{t'(\theta^{(r)})\right\}^{-1} t(\theta^{(r)}) \ , \tag{4.8}$$

where

$$t'(\theta) = \frac{M-m}{\{b'(\theta)-m\}\{M-b'(\theta)\}} b''(\theta) \ , \tag{4.9}$$

and

$$b''(\theta) = \sum_{k=1}^{K} \{s_k - b'(\theta)\}^2 f_k \exp\{\theta s_k - b(\theta)\} \ . \tag{4.10}$$

We typically initialize $\theta^{(0)}$ to be the value obtained from the previous $\theta$ update procedure. The first time $\theta$ is updated, we initialize $\theta^{(0)}$ to zero for every observation. We define convergence when $|t(\theta^{(r)})| < \epsilon$, where $\epsilon$ is a small threshold such as $10^{-10}$.

As $\mu_i \to M^-$, $\theta_i \to +\infty$. Likewise, as $\mu_i \to m^+$, $\theta_i \to -\infty$. To prevent numerical instability when $\mu_i$ is at or near these boundaries, we cap $|\theta_i|$ at a maximum value (500 by default). The appropriateness of this threshold would depend on the scale of response variable. Rather than adjust the threshold, we center and scale the response variable to the interval [-1, 1] (see Section 4.2.4).

### 4.2.2 $\tilde{f}_0$ optimization procedure

Holding $\beta$ fixed at its current estimate, we need to marginally optimize the log-likelihood $\ell(\beta, \tilde{f}_0)$ over $\tilde{f}_0$, subject to constraints (C2)-(C4). The linear constraints (C3) and (C4) could be enforced using constrained optimization techniques such as Lagrange multipliers or reducing the dimension of the parameter space by two. Huang and Rathouz (2012) used the former technique, while Rathouz and Gao (2009) used the latter. We propose a different method that is more computationally efficient. It is based on the BFGS technique. At each iteration, we apply a BFGS update to $\tilde{f}_0$ to improve the unconstrained log-likelihood and then apply constraints (C3) and (C4), which does not affect the log-likelihood of the estimate, as constraints (C3) and (C4) are only required for identifiability (i.e. uniqueness of the optimal $\tilde{f}_0$).

We define the transformation $\tilde{g}_0 = (g_1, \ldots, g_k) = (\log f_1, \ldots, \log f_K)$ and consider the log-likelihood as a function of $\tilde{g}_0$ only, with $\beta$ held fixed. Working on the log scale enforces constraint (C2) and also improves numerical stability. Specifically, numerical stability is improved by working with the score and Hessian as a function of $\tilde{g}_0$ rather than $\tilde{f}_0$.

BFGS is a quasi-Newton procedure, which makes iterative updates using approximate Hessian matrix along with the exact score function. Let $\tilde{g}_0^{(t)}$ be the estimate at the $t^{\text{th}}$ iteration. The updates take the form

$$\tilde{g}_0^{(t+1)} \leftarrow \tilde{g}_0^{(t)} - H_t^{-1} S(\tilde{g}_0^{(t)}; \beta) \ . \tag{4.11}$$

Here, $S(\tilde{g}_0; \beta)$ is the score as a function of $\tilde{g}_0$ only, holding $\beta$ fixed. It has $k^{\text{th}}$ element

$$\{S(\tilde{g}_0; \beta)\}_k = \sum_{i=1}^{n} \left\{ I(y_i = s_k) - \frac{\exp(g_k + \theta_i s_k)}{\exp\{b(\theta_i)\}} - \frac{\exp(g_k + \theta_i s_k)}{\exp\{b(\theta_i)\}} \frac{s_k - \mu_i}{b''(\theta_i)} (y_i - \mu_i) \right\} \tag{4.12}$$

for $k = 1, \ldots, K$ (Proof in Appendix A). Note that this score function ignores constraints (C3) and (C4). The matrix $H_t$ is an approximation to the Hessian of the log-likelihood as a function of $\tilde{g}_0$ only, holding $\beta$ fixed. This estimate is updated with each iteration. Letting $u_t = \tilde{g}_0^{(t)} - \tilde{g}_0^{(t-1)}$ and $v_t = S(\tilde{g}_0^{(t)}; \beta) - S(\tilde{g}_0^{(t-1)}; \beta)$, we can write the BFGS estimate of the Hessian recursively as

$$H_t = H_{t-1} + \frac{u_t u_t^T}{u_t^T v_t} - \frac{H_{t-1} v_t v_t^T H_{t-1}}{v_t^T H_{t-1} v_t} \ . \tag{4.13}$$

$H_{t+1}$ is a rank-2 update to $H_t$ that satisfies the secant condition: $H_t u_t = v_t$. Furthermore, $H_t$ is guaranteed to be symmetric and positive definite, even though the true Hessian is not full rank. (The true Hessian is not full rank because without imposing constraints (C3) and (C4), the log-likelihood does not have a unique optimum.) The BFGS update in Equation 4.11 requires the inverse of $H_t$, which can be calculated efficiently and directly, without calculating $H_t$. By the Sherman-Morrison-Woodbury formula,

$$H_t^{-1} = H_{t-1}^{-1} + \frac{(u_t^T v_t + u_t^T H_{t-1}^{-1} u_t)(u_t u_t^T)}{(u_t^T v_t)^2} - \frac{H_{t-1}^{-1} v_t u_t^T + u_t v_t^T H_{t-1}^{-1}}{u_t^T v_t} . \tag{4.14}$$

For an initial estimate, we propose $H_0^{-1} = \alpha I_K$, where $I_K$ is the $K \times K$ identity matrix. We perform a line search along the gradient to choose an appropriate value for $\alpha$ such that $\ell(\beta, \tilde{f}_0^{(1)}) > \ell(\beta, \tilde{f}_0^{(0)})$.

As previously mentioned, constraints (C3) and (C4) are only required for identifiability. After each BFGS iteration, we impose these constraints to the $\tilde{f}_0$ estimate, which does not affect the log-likelihood of the estimate. Specifically, we apply (C3) by scaling our estimate of $\tilde{f}_0$ to sum to one. We then "exponentially tilt" the estimate to enforce constraint (C4). In other words, we compute $\theta$ such that $\sum_{j=1}^{K} s_j f_j e^{\theta s_j} / \sum_{j=1}^{K} f_j e^{\theta s_j} = \mu_0$, and set our final estimate for the iteration to be $f_k \leftarrow f_k e^{\theta s_k} / \sum_{j=1}^{K} f_j e^{\theta s_j}$ for all $k$.

We initialize $\tilde{g}_0^{(0)}$ to be the log of the $\tilde{f}_0$ estimate obtained from the previous $\tilde{f}_0$ update procedure. We suggest using the empirical response distribution as an initial estimate of $\tilde{f}_0$. We define convergence using the relative change in the log-likelihood. Our default convergence threshold is $10^{-10}$. If the log-likelihood decreases after any iteration, we backtrack by half steps, setting $\tilde{g}_0^{(t+1)} \leftarrow \frac{1}{2}\left(\tilde{g}_0^{(t+1)} + \tilde{g}_0^{(t)}\right)$ until the log-likelihood improves. In our experience, we have found that the log-likelihood improves after most iterations without taking half steps, but log-likelihood decreases can occur sporadically (both at early and late iterations).

### 4.2.3 $\beta$ optimization procedure

Holding $\tilde{f}_0$ fixed at its current estimate, we could marginally optimize the log-likelihood over $\beta$ using iteratively re-weighted least squares (IRLS). Rather than iterating until convergence, however, we propose using a single IRLS iteration to update $\beta$ in between $\tilde{f}_0$ updates. The IRLS algorithm is simply the Newton-Raphson algorithm, but using the Fisher information in place of the negative Hessian matrix. This technique is commonly referred to as Fisher Scoring. As we now show, the Fisher Scoring update minimizes a weighted least squares expression, which is why we can refer to the algorithm as IRLS. The score function is given by

$$S(\beta;\tilde{f}_0) = \sum_{i=1}^{n} x_i \left(\frac{1}{g'(\mu_i)}\right)\left(\frac{1}{b''(\theta_i)}\right)(y_i - \mu_i) = \boldsymbol{X}^T\boldsymbol{W}\boldsymbol{r}\ , \tag{4.15}$$

(Proof in Appendix C.) and the Fisher information is

$$\begin{aligned}\mathcal{I}(\beta;\tilde{f}_0) &= \mathrm{E}\left(S(\beta;\tilde{f}_0)S(\beta;\tilde{f}_0)^T \middle| X_1 = x_1, \ldots, X_n = x_n\right) \\ &= \sum_{i=1}^{n} x_i \left(\frac{1}{g'(\mu_i)}\right)^2\left(\frac{1}{b''(\theta_i)}\right) x_i^T \\ &= \boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X}\ ,\end{aligned} \tag{4.16}$$

where $\boldsymbol{X}$ is an $n \times p$ matrix with rows $x_i^T$, $\boldsymbol{W}$ is an $n \times n$ diagonal matrix with entries $\left(\frac{1}{g'(\mu_i)}\right)^2 \frac{1}{b''(\theta_i)}$, and $\boldsymbol{r}$ is an $n \times 1$ vector with entries $g'(\mu_i)(Y_i - \mu_i)$.

Let $\beta^{(0)}$ be the estimate obtained from the previous $\beta$ update procedure. The IRLS step between $\beta^{(0)}$ and the updated estimate, $\beta^{(1)}$, is

$$\begin{aligned}\beta^{(1)} - \beta^{(0)} &= \left\{\mathcal{I}(\beta^{(0)};\tilde{f}_0)\right\}^{-1} S(\beta^{(0)};\tilde{f}_0) \\ &= \left(\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{W}\boldsymbol{r}\ .\end{aligned} \tag{4.17}$$

This is the solution to a weighted least squares expression, which can be computed efficiently using QR decomposition.

### 4.2.4 Response variable transformation

Numerical stability issues can occur if the $\exp\{\theta_i s_k\}$ terms become too large or small during optimization. To prevent these terms from exceeding R's default floating-point number range, we transform the response vector to the interval [-1, 1]. Specifically, the response values are transformed to $y_i^* = (y_i - \frac{m+M}{2}) \cdot \frac{2}{M-m}$. It turns out the semiparametric log-likelihood function with the transformed response and modified link function $g_*(\mu) = g(\frac{m+M}{2} + \frac{M-m}{2}\mu)$ is equivalent to the original log-likelihood. The parameters $\beta$ and $\tilde{f}_0$ that optimize the modified log-likelihood also optimize the original log-likelihood (Proof in Appendix D.).

As mentioned in the "$\theta$ update procedure" section, we choose to cap $|\theta|$ for each observation at 500 by default. This restricts the rescaled $\theta_i s_k$ terms to the interval [-500, 500]. Note that the optimization function in the **gldrm** R package returns the estimated $\theta_i$ values for each observation, and these values are on the original scale of the response variable.

## 4.3 Inference

The **gldrm** R package (Wurm and Rathouz, 2017) can perform inference on the regression coefficients using likelihood ratio tests for nested models. It can also calculate likelihood ratio, Wald, or score confidence intervals. All three confidence intervals should yield similar results for large samples, but the Wald form may be preferred for its computational simplicity. Likelihood ratio and score confidence intervals are more computationally expensive to calculate. Huang (2014) recommends likelihood ratio tests for small samples.

The Wald asymptotic variance estimate for the $\beta$ estimator can be obtained from the inverse of the information matrix given in Equation 4.16. Recall this information matrix is calculated with the $\tilde{f}_0$ parameters held fixed. Its inverse is a valid asymptotic variance matrix because the full information matrix is block diagonal; i.e., the $\beta$ and $\tilde{f}_0$ estimators are asymptotically independent. The **gldrm** optimization function returns standard error estimates for each coefficient, which are displayed by the print method along with p-values. Wald confidence intervals are straightforward to calculate from the standard errors and can be obtained from the `gldrmCI` function. For these single-coefficient hypothesis tests and confidence intervals, we approximate the null distribution by a $t$-distribution with $n - p$ degrees of freedom, where $n$ is the number of observations and $p$ is the rank of the covariate matrix.

Likelihood ratio tests for nested models are based on the usual test statistic: 2 times the log-likelihood difference between the full and reduced model. Following Huang (2014), we approximate the null distribution by an $F$-distribution with $q$ and $n - p$ degrees of freedom, where $q$ is the difference in the number of parameters between the full and reduced models. This test can be performed by the `gldrmLRT` function.

Likelihood ratio and score confidence intervals for a single coefficient can be obtained from the `gldrmCI` function. These confidence intervals are more computationally intensive than Wald confidence intervals because an iterative method is required to search for the interval boundaries. We use the following bisection method for this. Suppose we want to obtain the lower boundary of a confidence level $1 - \alpha$ interval for a single coefficient $\beta^*$, and that the gldrm estimate of this coefficient is $\hat{\beta}^*$. For the lower boundary, we need to find $\beta^*_{\text{lo}}$

such that $\beta^*_{\text{lo}} < \hat{\beta}^*$ and the one-sided likelihood ratio test has a p-value equal to $\alpha/2$ under the null hypothesis $H_0 : \beta^* = \beta^*_{\text{lo}}$. As explained in the next paragraph, we can compute the p-value for any guess of $\beta^*_{\text{lo}}$. We begin by finding a guess that is too low (p-value less than $\alpha/2$) and a guess that is too high (p-value greater than $\alpha/2$). We then obtain the p-value of the midpoint. If the p-value is less than $\alpha/2$ then the midpoint becomes the new low guess, and if it is greater than $\alpha/2$ then the midpoint becomes the new high guess. We iterate this process, each time halving the distance between the low guess and high guess, until we have a guess of $\beta^*_{\text{lo}}$ that has a p-value arbitrarily close to $\alpha/2$. The same procedure can be used to solve for $\beta^*_{\text{hi}}$.

Let $\beta_0$ be any constant. To calculate the likelihood ratio or score test p-value of $H_0 : \beta^* = \beta_0$, we need to optimize the log-likelihood over $\tilde{f}_0$ and all other $\beta$ values, holding $\beta^*$ fixed at $\beta_0$. This can be done by multiplying $\beta_0$ with the column of the $\boldsymbol{X}$ matrix corresponding to $\beta^*$ and treating this vector as an offset to the linear predictor. In other words, this vector contains a fixed component of the linear predictor for each observation. The $\beta^*$ column is dropped from $\boldsymbol{X}$, and the model is optimized with the offset term.

## 4.4 Simulation experiments

Under four different simulation scenarios, we check the finite sample performance of the regression coefficient estimators, $\hat{\beta}$, as well as the reference distribution cdf estimator $\hat{F}_0()$. For $\hat{\beta}$, we also check the accuracy of the estimated standard errors.

For each simulation, the data generating mechanism is a gamma glm. The covariates are the same under each simulation setting, and their values are fixed, not random. The covariate vector is $x = (x_0, x_1, x_2, x_3)$. Variable $x_0$ is an intercept column of ones. Variable $x_1$ is an indicator with exactly 20% ones in each sample (observations 5, 10, 15, 20, . . . have $x_1 = 1$, and the rest have $x_1 = 0$). Variable $x_2$ is a continuous variable that follows a uniform sequence between zero and one in each sample (the first observation has $x_2 = 0$, and the last observation has $x_2 = 1$). Lastly, $x_3 = x_1 \cdot x_2$.

The coefficient values were $(\beta_1, \beta_2, \beta_3) = (1, 1, -2)$ for all simulation scenarios. The intercepts were $\beta_0 = 0$ for simulations 1 & 2, and $\beta_0 = 1$ for simulations 3 & 4. These values were selected so the mean response values were centered close to one and, for the identity link scenarios, guaranteed to be positive.

The log link was used for simulations 1 & 2, and the identity link was used for simulations 3 & 4. For all four simulations, the response was drawn from a gamma distribution with Var$(y|x) = \phi \text{E}(y|x)^2$, where $\phi$ varies by simulation to achieve different $R^2$ values. $R^2$ is defined as the squared correlation between the response and linear combination of predictors.

The log link was used for simulations 1 & 2, and the identity link was used for simulations 3 & 4. For all four simulations, the response was drawn from a gamma distribution with Var$(y|x) = \phi E(y|x)^2$, where $\phi$ varies by simulation to achieve different $R^2$ values. $R^2$ is defined as the squared correlation between the response and linear combination of predictors.

1. Simulation 1: log link and high $R^2$ ($\phi = 0.5$, $R^2 \approx 0.13$)
2. Simulation 2: log link and low $R^2$ ($\phi = 4$, $R^2 \approx 0.019$)
3. Simulation 3: identity link and high $R^2$ ($\phi = 0.25$, $R^2 \approx 0.13$)
4. Simulation 4: identity link and low $R^2$ ($\phi = 2$, $R^2 \approx 0.018$)

A gldrm was fit to each data set with correct link function and with $\tilde{f}_0$ constrained to have mean $\mu_0 = 1$. If we had followed our usual practice of choosing $\mu_0$ to be the sample mean of the observed response values, then the reference distribution would have a different mean for each simulation replicate. By choosing $\mu_0 = 1$ for all replicates, the true $f_0$ always follows a gamma distribution with mean one and variance $\phi$, where $\phi$ varies by simulation scenario. The value $\mu_0 = 1$ was chosen because, by design, it fell within the range of observed response values for all 2,000 replicates of each simulation scenario. The cumulative reference distribution estimate, denoted as $\hat{F}_0(\cdot)$, was computed at percentiles 0.1, 0.25, 0.5, 0.75, 0.9.

For each simulation scenario, we display four tables. The first table (Tables 4.1, 4.5, 4.9, and 4.13) contains the sample mean of each $\beta$ estimator. For comparison, we also calculated the gamma glm coefficient estimates for each simulated data set and display the sample mean alongside in parentheses.

The second table (Tables 4.2, 4.6, 4.10, and 4.14) contains the root mean squared error (rmse) of each $\beta$ estimator. The rmse is calculated as $\sqrt{\frac{1}{2000}\sum_{i=1}^{2000}(\hat{\beta}_i - \beta)^2}$, where $\hat{\beta}_i$ is the estimator of the $i^{\text{th}}$ simulation replicate, and $\beta$ is the true parameter value. For comparison, we also show the relative efficiency compared to the gamma glm estimator. Relative efficiency is calculated as $\text{mse}_{\text{glm}}/\text{mse}_{\text{gldrm}}$, where mse is the mean squared error (not rmse).

The third table (Tables 4.3, 4.7, 4.11, and 4.15) contains Wald confidence interval coverage rates for each $\beta$ estimator. Confidence intervals were calculated based on a $t$-distribution with $n-4$ degrees of freedom.

The fourth table (Tables 4.4, 4.8, 4.12, and 4.16) contains the mean of $\hat{F}_0()$, calculated at the true $10^{\text{th}}$, $25^{\text{th}}$, $50^{\text{th}}$, $75^{\text{th}}$, and, $90^{\text{th}}$ percentiles.

To summarize the results, the gldrm estimators perform as expected. In all four simulation scenarios, the bias of $\hat{\beta}$ and $\hat{F}_0()$ goes to zero as the sample size increases. For the high $R^2$ scenarios, there is very little bias even at $n = 25$. Also, the relative efficiency of the $\beta$ estimators compared to the gamma glm estimators goes to one as the sample size increases. This is expected because, as Rathouz and Gao (2009) demonstrated, the $\beta$ and $f_0$ estimators are asymptotically orthogonal, so gldrm and glm have equal asymptotic efficiency when the glm model is correctly specified. For the high $R^2$ scenarios, the relative efficiency is close to one even at $n = 25$. For the low $R^2$ scenarios, the gldrm estimator is actually more efficient than the glm estimator for small sample sizes.

The standard error estimates for gldrm $\beta$ estimators are consistently low for small sample sizes, as demonstrated by low Wald confidence interval coverage rates. The coverage rates improve with increasing sample size, demonstrating good asymptotic properties of the standard error estimators. Likelihood ratio confidence intervals can be calculated with the **gldrm** R package and may have better small sample performance, as demonstrated by Huang (2014).

### 4.4.1 Simulation 1

Simulation 1 uses log link with $\beta = (0, 1, 1, -2)$ and $\phi = 0.5$. This results in an $R^2$ of approximately 0.13. The simulation was replicated 2,000 times.

| | $\hat{\beta}$ mean (glm mean) | | |
|---|---|---|---|
| | n = 25 | n = 100 | n = 400 |
| $\beta_0 = 0$ | -0.02 (-0.02) | 0.00 (0.00) | 0.00 (0.00) |
| $\beta_1 = 1$ | 0.93 (0.92) | 0.98 (0.99) | 0.99 (0.99) |
| $\beta_2 = 1$ | 0.99 (1.00) | 0.99 (1.00) | 1.00 (1.00) |
| $\beta_3 = -2$ | -2.00 (-2.00) | -2.00 (-2.01) | -1.99 (-1.99) |

Table 4.1: Sample mean of $\hat{\beta}$. For comparison, the gamma glm sample mean is shown in parentheses.

| | $\hat{\beta}$ rmse (relative efficiency) | | |
|---|---|---|---|
| | n = 25 | n = 100 | n = 400 |
| $\beta_0$ | 0.31 (1.01) | 0.16 (1.00) | 0.08 (1.00) |
| $\beta_1$ | 0.81 (1.05) | 0.38 (0.99) | 0.18 (1.00) |
| $\beta_2$ | 0.54 (1.00) | 0.27 (1.00) | 0.14 (1.01) |
| $\beta_3$ | 1.27 (1.04) | 0.63 (1.00) | 0.30 (1.00) |

Table 4.2: Root mean squared error (rmse) of $\hat{\beta}$. In parentheses is the relative efficiency compared to the gamma glm estimator.

| | 80% C.I. | | | 90% C.I. | | | 95% C.I. | | |
|---|---|---|---|---|---|---|---|---|---|
| | n = 25 | n = 100 | n = 400 | n = 25 | n = 100 | n = 400 | n = 25 | n = 100 | n = 400 |
| $\beta_0$ | 0.755 | 0.799 | 0.805 | 0.870 | 0.896 | 0.899 | 0.920 | 0.945 | 0.944 |
| $\beta_1$ | 0.677 | 0.764 | 0.800 | 0.782 | 0.860 | 0.900 | 0.837 | 0.919 | 0.949 |
| $\beta_2$ | 0.742 | 0.784 | 0.810 | 0.853 | 0.883 | 0.896 | 0.910 | 0.940 | 0.950 |
| $\beta_3$ | 0.681 | 0.767 | 0.806 | 0.800 | 0.869 | 0.907 | 0.867 | 0.920 | 0.948 |

Table 4.3: Wald confidence interval coverage rate for $\beta$.

| | | $\hat{F}_0(y)$ mean | | |
|---|---|---|---|---|
| $F_0(y)$ | $y$ | n = 25 | n = 100 | n = 400 |
| 0.10 | 0.27 | 0.082 | 0.097 | 0.099 |
| 0.25 | 0.48 | 0.222 | 0.244 | 0.249 |
| 0.50 | 0.84 | 0.487 | 0.498 | 0.499 |
| 0.75 | 1.35 | 0.759 | 0.751 | 0.750 |
| 0.90 | 1.94 | 0.911 | 0.902 | 0.900 |

Table 4.4: Sample mean of $\hat{F}_0$ at selected true percentiles.

### 4.4.2 Simulation 2

Simulation 2 uses log link with $\beta = (0, 1, 1, -2)$ and $\phi = 4$. This results in an $R^2$ of approximately 0.019. The simulation was replicated 2,000 times.

| | $\hat{\beta}$ mean (glm mean) | | |
|---|---|---|---|
| | n = 25 | n = 100 | n = 400 |
| $\beta_0 = 0$ | -0.16 (-0.19) | -0.05 (-0.06) | -0.02 (-0.02) |
| $\beta_1 = 1$ | 0.21 (0.29) | 0.80 (0.84) | 0.96 (0.97) |
| $\beta_2 = 1$ | 0.90 (0.96) | 0.99 (1.01) | 1.00 (1.00) |
| $\beta_3 = -2$ | -1.74 (-1.98) | -1.93 (-2.01) | -1.99 (-2.00) |

Table 4.5: Sample mean of $\hat{\beta}$. For comparison, the gamma glm sample mean is shown in parentheses.

| | $\hat{\beta}$ rmse (relative efficiency) | | |
|---|---|---|---|
| | n = 25 | n = 100 | n = 400 |
| $\beta_0$ | 0.94 (0.88) | 0.45 (0.97) | 0.23 (1.00) |
| $\beta_1$ | 2.83 (0.70) | 1.12 (0.91) | 0.51 (0.98) |
| $\beta_2$ | 1.62 (0.84) | 0.78 (0.96) | 0.39 (0.99) |
| $\beta_3$ | 4.18 (0.63) | 1.88 (0.88) | 0.89 (0.98) |

Table 4.6: Root mean squared error (rmse) of $\hat{\beta}$. In parentheses is the relative efficiency compared to the gamma glm estimator.

| | 80% C.I. | | | 90% C.I. | | | 95% C.I. | | |
|---|---|---|---|---|---|---|---|---|---|
| | n = 25 | n = 100 | n = 400 | n = 25 | n = 100 | n = 400 | n = 25 | n = 100 | n = 400 |
| $\beta_0$ | 0.704 | 0.770 | 0.782 | 0.806 | 0.877 | 0.888 | 0.869 | 0.933 | 0.945 |
| $\beta_1$ | 0.626 | 0.724 | 0.781 | 0.732 | 0.832 | 0.883 | 0.794 | 0.897 | 0.935 |
| $\beta_2$ | 0.680 | 0.755 | 0.774 | 0.795 | 0.862 | 0.881 | 0.869 | 0.925 | 0.939 |
| $\beta_3$ | 0.633 | 0.722 | 0.762 | 0.750 | 0.829 | 0.879 | 0.819 | 0.895 | 0.935 |

Table 4.7: Wald confidence interval coverage rate for $\beta$.

| | | $\hat{F}_0(y)$ mean | | |
|---|---|---|---|---|
| $F_0(y)$ | $y$ | n = 25 | n = 100 | n = 400 |
| 0.10 | 0.00 | 0.093 | 0.100 | 0.101 |
| 0.25 | 0.01 | 0.230 | 0.247 | 0.250 |
| 0.50 | 0.17 | 0.465 | 0.496 | 0.499 |
| 0.75 | 1.04 | 0.727 | 0.747 | 0.749 |
| 0.90 | 3.00 | 0.903 | 0.899 | 0.900 |

Table 4.8: Sample mean of $\hat{F}_0$ at selected true percentiles.

### 4.4.3 Simulation 3

Simulation 3 uses identity link with $\beta = (1, 1, 1, -2)$ and $\phi = 0.25$. This results in an $R^2$ of approximately 0.13. The simulation was replicated 2,000 times.

| | $\hat{\beta}$ mean (glm mean) | | |
|---|---|---|---|
| | n = 25 | n = 100 | n = 400 |
| $\beta_0 = 1$ | 1.00 (1.00) | 0.99 (0.99) | 1.00 (1.00) |
| $\beta_1 = 1$ | 1.01 (1.01) | 1.00 (1.00) | 0.99 (0.99) |
| $\beta_2 = 1$ | 1.01 (1.01) | 1.01 (1.01) | 1.00 (1.00) |
| $\beta_3 = -2$ | -2.02 (-2.01) | -2.01 (-2.01) | -1.99 (-1.99) |

Table 4.9: Sample mean of $\hat{\beta}$. For comparison, the gamma glm sample mean is shown in parentheses.

| | $\hat{\beta}$ rmse (relative efficiency) | | |
|---|---|---|---|
| | n = 25 | n = 100 | n = 400 |
| $\beta_0$ | 0.25 (1.01) | 0.13 (1.01) | 0.07 (1.00) |
| $\beta_1$ | 0.88 (1.04) | 0.40 (1.01) | 0.20 (1.00) |
| $\beta_2$ | 0.55 (1.02) | 0.28 (1.01) | 0.14 (1.00) |
| $\beta_3$ | 1.26 (1.02) | 0.63 (1.02) | 0.32 (1.00) |

Table 4.10: Root mean squared error (rmse) of $\hat{\beta}$. In parentheses is the relative efficiency compared to the gamma glm estimator.

| | 80% C.I. | | | 90% C.I. | | | 95% C.I. | | |
|---|---|---|---|---|---|---|---|---|---|
| | n = 25 | n = 100 | n = 400 | n = 25 | n = 100 | n = 400 | n = 25 | n = 100 | n = 400 |
| $\beta_0$ | 0.735 | 0.793 | 0.802 | 0.830 | 0.888 | 0.892 | 0.888 | 0.939 | 0.947 |
| $\beta_1$ | 0.670 | 0.789 | 0.799 | 0.785 | 0.875 | 0.892 | 0.847 | 0.931 | 0.949 |
| $\beta_2$ | 0.737 | 0.782 | 0.796 | 0.849 | 0.885 | 0.893 | 0.916 | 0.941 | 0.943 |
| $\beta_3$ | 0.667 | 0.773 | 0.792 | 0.800 | 0.876 | 0.896 | 0.872 | 0.937 | 0.942 |

Table 4.11: Wald confidence interval coverage rate for $\beta$.

| | | $\hat{F}_0(y)$ mean | | |
|---|---|---|---|---|
| $F_0(y)$ | $y$ | n = 25 | n = 100 | n = 400 |
| 0.10 | 0.44 | 0.077 | 0.095 | 0.098 |
| 0.25 | 0.63 | 0.219 | 0.244 | 0.249 |
| 0.50 | 0.92 | 0.492 | 0.498 | 0.499 |
| 0.75 | 1.28 | 0.769 | 0.753 | 0.751 |
| 0.90 | 1.67 | 0.913 | 0.903 | 0.901 |

Table 4.12: Sample mean of $\hat{F}_0$ at selected true percentiles.

### 4.4.4 Simulation 4

Simulation 4 uses identity link with $\beta = (1, 1, 1, -2)$ and $\phi = 2$. This results in an $R^2$ of approximately 0.018. The simulation was replicated 2,000 times.

| | $\hat{\beta}$ mean (glm mean) | | |
|---|---|---|---|
| | n = 25 | n = 100 | n = 400 |
| $\beta_0 = 1$ | 1.02 (1.02) | 1.01 (1.01) | 1.00 (1.00) |
| $\beta_1 = 1$ | 0.91 (0.89) | 0.97 (1.00) | 0.99 (0.99) |
| $\beta_2 = 1$ | 0.99 (1.02) | 0.97 (0.98) | 0.99 (1.00) |
| $\beta_3 = -2$ | -1.86 (-1.85) | -1.92 (-1.95) | -1.98 (-1.99) |

Table 4.13: Sample mean of $\hat{\beta}$. For comparison, the gamma glm sample mean is shown in parentheses.

| | $\hat{\beta}$ rmse (relative efficiency) | | |
|---|---|---|---|
| | n = 25 | n = 100 | n = 400 |
| $\beta_0$ | 0.73 (0.85) | 0.39 (0.98) | 0.19 (1.00) |
| $\beta_1$ | 2.35 (0.95) | 1.16 (0.92) | 0.58 (0.98) |
| $\beta_2$ | 1.58 (0.85) | 0.81 (0.97) | 0.40 (1.00) |
| $\beta_3$ | 3.27 (0.85) | 1.83 (0.90) | 0.92 (0.98) |

Table 4.14: Root mean squared error (rmse) of $\hat{\beta}$. In parentheses is the relative efficiency compared to the gamma glm estimator.

| | 80% C.I. | | | 90% C.I. | | | 95% C.I. | | |
|---|---|---|---|---|---|---|---|---|---|
| | n = 25 | n = 100 | n = 400 | n = 25 | n = 100 | n = 400 | n = 25 | n = 100 | n = 400 |
| $\beta_0$ | 0.661 | 0.761 | 0.794 | 0.754 | 0.845 | 0.893 | 0.802 | 0.893 | 0.938 |
| $\beta_1$ | 0.595 | 0.718 | 0.767 | 0.692 | 0.822 | 0.868 | 0.741 | 0.870 | 0.921 |
| $\beta_2$ | 0.665 | 0.769 | 0.779 | 0.777 | 0.864 | 0.895 | 0.847 | 0.910 | 0.943 |
| $\beta_3$ | 0.666 | 0.711 | 0.777 | 0.789 | 0.824 | 0.877 | 0.871 | 0.892 | 0.932 |

Table 4.15: Wald confidence interval coverage rate for $\beta$.

| | | $\hat{F}_0(y)$ mean | | |
|---|---|---|---|---|
| $F_0(y)$ | $y$ | n = 25 | n = 100 | n = 400 |
| 0.10 | 0.02 | 0.085 | 0.097 | 0.100 |
| 0.25 | 0.10 | 0.223 | 0.246 | 0.249 |
| 0.50 | 0.45 | 0.474 | 0.493 | 0.499 |
| 0.75 | 1.32 | 0.741 | 0.749 | 0.750 |
| 0.90 | 2.71 | 0.907 | 0.902 | 0.900 |

Table 4.16: Sample mean of $\hat{F}_0$ at selected true percentiles.

### 4.4.5 Likelihood ratio and score confidence intervals

To support our claim that likelihood ratio confidence intervals may have better small sample performance, we compared the Wald, likelihood ratio, and score 95% confidence interval coverage rates under the settings of Simulation 1 with $n = 25$. The coverage rates are shown in Table 4.17. The Wald coverage rates are similar to those in Table 4.3, but not identical because this experiment used a new set of 2,000 replicates.

While the Wald confidence intervals tend to be too narrow, the score confidence intervals tend to be a bit too wide. The likelihood ratio intervals have coverage rates much closer to 0.95 than the corresponding Wald or score intervals. This is consistent with the findings of Huang (2014).

| | $\beta_0$ | $\beta_1$ | $\beta_2$ | $\beta_3$ |
|---|---|---|---|---|
| Wald | 0.918 | 0.823 | 0.909 | 0.859 |
| Likelihood ratio | 0.969 | 0.939 | 0.958 | 0.944 |
| Score | 0.976 | 0.966 | 0.967 | 0.967 |

Table 4.17: Coverage rates for 95% confidence intervals under the settings of Simulation 1.

## 4.5 Computational time and scaling

The following simulation explores the computation time of **gldrm** and how it scales with the number of observations $n$ and number of covariates $p$. For each value of $n$ and $p$, we fit **gldrm** to 100 randomly generated data sets. A gldrm was fit once to each data set. Variation in computation time is attributable to both variation in the time it takes to fit gldrm to a particular data set and variation among randomly generated data sets.

Covariates were drawn independently from a standard normal distribution, and coefficients were drawn independently from a uniform distribution on the interval (-1, 1). Simulation 1 used an identity link function with response drawn from a normal distribution with variance function $V(\mu) = 1$. Simulation 2 used a log link function with response drawn from an exponential distribution, i.e. a gamma distribution with variance function $V(\mu) = \mu^2$.

Iteration limits were set at 100 per $\theta$ update, 1,000 per $f_0$ update, and 100 per outer loop update (each consisting of a $\beta$ update, followed by an $f_0$ update), where each $\beta$ update consisted of a single iteration. Convergence thresholds were set at $10^{-10}$ for the $\theta$ update, $f_0$ update, and outer loop update. This experiment was run using a 2.2 GHz AMD Opteron 6174 processor. Figures 4.1 and 4.2 show the average CPU seconds for Simulations 1 and 2, respectively.

We also repeated this experiment with the number of support points fixed at 25. To do this, we generated $n$ response values from the model. The first 25 values were designated as the support, and the remaining response values were matched to the nearest support value and set to that value. This discrete data generating model is not actually a gldrm model, but we are only using it to evaluate computation time. Figures 4.3 and 4.4 show the average CPU seconds for Simulations 1 and 2, respectively, with support size fixed at 25.

In summary, the computation time scales roughly linearly with $n$ when the support is fixed at 25. When the support grows with $n$, the computation time grows faster than linearly. Computation time increases with $p$, but there is not a consistent pattern, especially with fixed support.

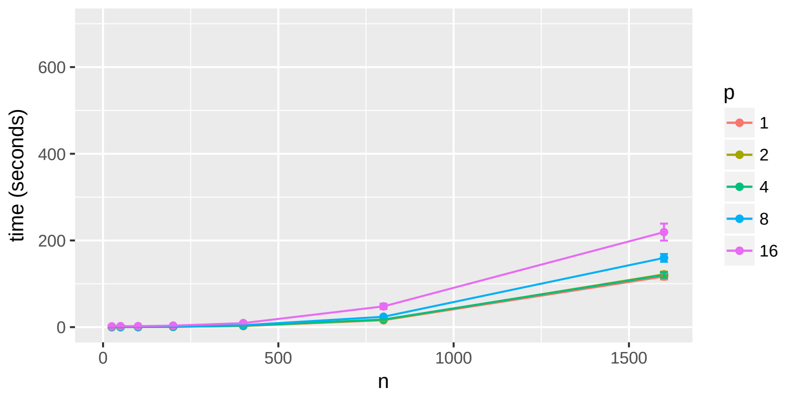

Figure 4.1: Mean computation time for Simulation 1. Error bars represent ±2 standard errors over 100 replicates.

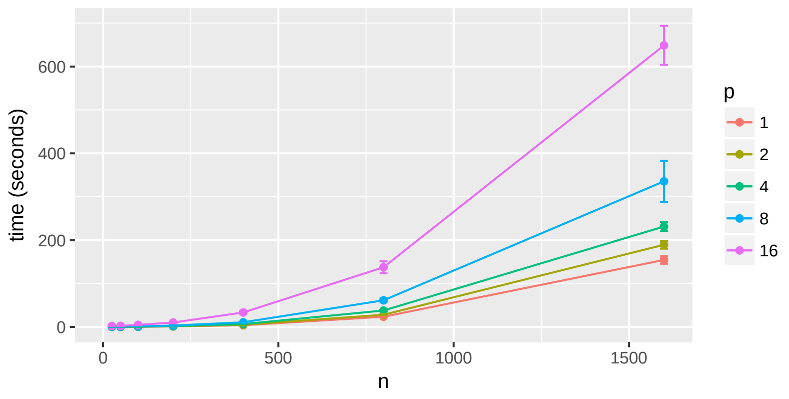

Figure 4.2: Mean computation time for Simulation 2. Error bars represent ±2 standard errors over 100 replicates.

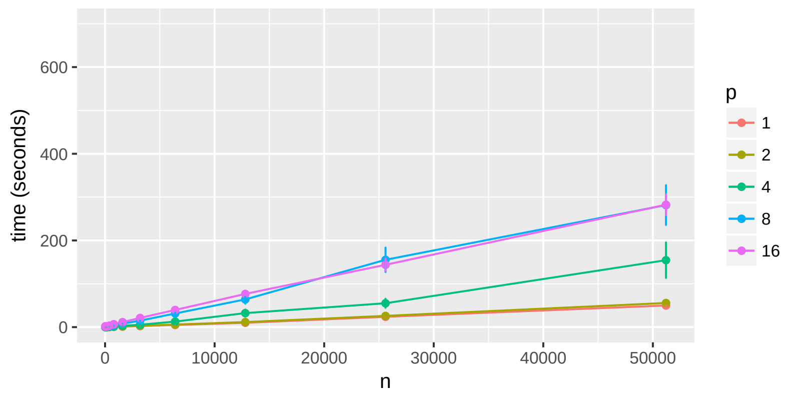

Figure 4.3: Mean computation time for Simulation 1 with support size fixed at 25. Error bars represent $\pm 2$ standard errors over 100 replicates.

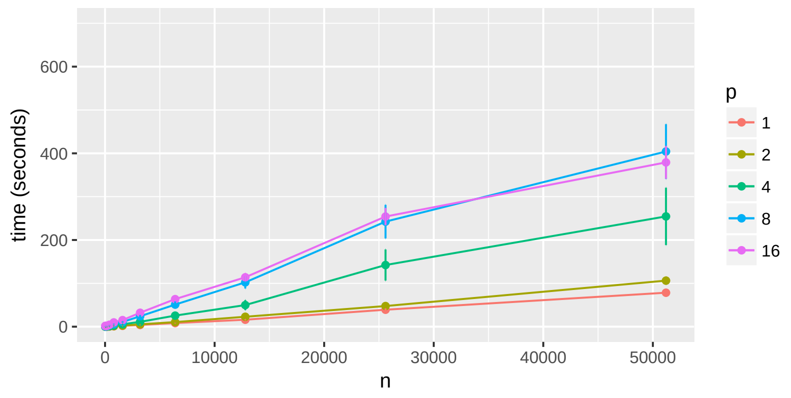

Figure 4.4: Mean computation time for Simulation 2 with support size fixed at 25. Error bars represent $\pm 2$ standard errors over 100 replicates.

## 4.6 R package: gldrm

The **gldrm** package (Wurm and Rathouz, 2017) was written entirely in the R programming language. Its primary functions are the following.

- `gldrm` is the main function for fitting gldrm models.
- `gldrmLRT` performs a likelihood ratio test between nested gldrm models. The test statistic is calculated as $2 \times (llik - llik_0)/r$, where $r$ is the difference is the number

of parameters between the full and null models. Under the null hypothesis, the test statistic follows an asymptotic F distribution with degrees of freedom $r$ and $n - p$, where $n$ is the number of observations and $p$ is the number of parameters in the full model.

- `gldrmCI` calculates Wald or likelihood ratio confidence intervals for a single coefficient.
- `predict.gldrmFit` is a predict method, which is modeled after the `print.glm` method in the **stats** package. Predictions can be obtained on the scale of the response or linear predictor, or the tilted nonparametric density for each observation can be returned.

## 4.7 R example: iris data

We demonstrate the **gldrm** package using the Iris data set from the **datasets** package. This is a data set of $n = 150$ observations. We choose sepal length to be the response variable. This variable has 35 unique values, so the support is $K = 35$. We first demonstrate how to fit a gldrm model with the optimization function, which is simply called `gldrm`. We then demonstrate how to perform inference on the regression coefficients. Finally we show how to obtain predictions for a set of observations, including the predicted mean and nonparametric estimate of the distribution function.

### 4.7.1 Fit gldrm model

The `gldrm` optimization function takes a formula and data argument, similar to R's `glm` function. Instead of passing both an error distribution and link function through a family argument, the `gldrm` only requires a link function. The link function is passed through three separate arguments:

1. `linkfun` A vectorized link function.
2. `linkinv` The inverse link function, also vectorized.
3. `mu.eta` The derivative of the inverse link function, also vectorized.

These arguments mirror the components of the `link-glm` class, which is constructed by the `make.link` function. The use of three separate link function arguments allows users to provide any custom link function, but it is important that the user provides a link function is both invertible and differentiable. It is possible to pass the three link function arguments directly from a `link-glm` object as demonstrated below.

```
R> ### Load gldrm package and Iris data from datasets package
R> library(gldrm)
R> data(iris, package="datasets")

R> ### Fit gldrm with all variables
R> lf <- make.link("log")
R> linkfun <- lf$linkfun  # this is equivalent to function(mu) log(mu)
R> linkinv <- lf$linkinv  # this is equivalent to function(eta) exp(eta)
R> mu.eta <- lf$mu.eta  # this is equivalent to function(eta) exp(eta)

R> fit <- gldrm(Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width + Species,
R+             data=iris, linkfun, linkinv, mu.eta)
R> fit

Summary of gldrm fit

Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept)         1.1832     0.0369   32.10  < 2e-16 ***
Sepal.Width         0.0788     0.0128    6.17  6.4e-09 ***
Petal.Length        0.1128     0.0102   11.04  < 2e-16 ***
Petal.Width        -0.0350     0.0248   -1.41    0.162
Speciesversicolor  -0.0561     0.0395   -1.42    0.157
Speciesvirginica   -0.0994     0.0557   -1.79    0.076 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Likelihood ratio test against null model:
F-statistic:  57.4
Numerator degrees of freedom:  5
Denominator degrees of freedom:  144
```

```
P-value:  < 2e-16
```

### 4.7.2 Inference

The gldrmLRT function performs a semiparametric likelihood ratio test between two nested models. We demonstrate this on the Iris data by fitting a sub-model that excludes "Species", which is a categorical variable with three levels and two degrees of freedom. We also obtain Wald and likelihood ratio confidence intervals for the petal width coefficient.

```
R> ### Fit gldrm without the categorical variable "Species"
R> fit0 <- gldrm(Sepal.Length ~ Sepal.Width + Petal.Length + Petal.Width,
R+               data=iris, linkfun, linkinv, mu.eta)

R> ### Likelihood ratio test for the nested models
R> gldrmLRT(fit, fit0)

Likelihood ratio test:

F-statistic:  2.03
Numerator degrees of freedom:  2
Denomicator degrees of freedom:  144
P-value:  0.135

R> ### Wald 95% confidence interval for Petal.Width
R> gldrmCI(fit, term="Petal.Width", test="Wald", type="2-sided", level=.95)

95% Wald confidence interval for Petal.Width:
(-0.084, 0.014)

R> ### Likelihood ratio 95% confidence interval for Petal.Width
R> gldrmCI(fit, term="Petal.Width", test="LRT", type="2-sided", level=.95)

95% likelihood ratio confidence interval for Petal.Width:
(-0.094, 0.025)
```

### 4.7.3 Prediction

We obtain predictions for three selected observations in the data set: one from each Species. These observations were contained in training data, but we could obtain predictions for out-of-sample observations in the same way. Using the `predict` method, we obtain the fitted mean response value for each observation, which is the estimate of $\mathrm{E}(Y|X = x)$.

We also use the `predict` method to obtain the nonparametric estimate of the conditional density $f(y|X = x)$. This is obtained by setting the argument `type = "fTilt"`, which returns an $n \times K$ matrix with $(i, k)^{\text{th}}$ entry $\tilde{f}_k \exp\{\theta_i s_k - b(\theta_i)\}$. Each row contains the nonparametric conditional density estimate for a given observation and sums to one. We use this matrix to calculate the conditional probabilities for the form $\mathrm{P}(y_1 < Y \leq y_2|X = x)$ for each observation. Note that all observed support values (sepal length values) fall between four and eight, so all probability mass falls within this interval.

```
R> ### Select three observations; one from each Species
R> newdata <- iris[c(1, 51, 101), ]

R> ### Fitted mean Sepal.Length
R> fitted_mean <- predict(fit, newdata=newdata, type="response")
R> fitted_mean <- round(fitted_mean, 2)

R> ### Estimated Sepal.Length distribution of each observation
R> ### Note: all Sepal.Length values are between 4 and 8
R> fTilt <- predict(fit, newdata=newdata, type="fTilt")
R> spt <- fit$spt
R> F4 <- rowSums(fTilt[ , spt <= 4])
R> F5 <- rowSums(fTilt[ , spt <= 5])
R> F6 <- rowSums(fTilt[ , spt <= 6])
R> F7 <- rowSums(fTilt[ , spt <= 7])
R> F8 <- rowSums(fTilt[ , spt <= 8])

R> Ftilt <- cbind(F5-F4, F6-F5, F7-F6, F8-F7)
R> Ftilt <- round(F, 3)
R> colnames(Ftilt) <- c("P(4=Y<=5)", "P(5<Y<=6)", "P(6<Y<=7)", "P(7<Y<=8)")
R> cbind(newdata, fitted_mean, Ftilt)
```

```
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species fitted_mean
1            5.1         3.5          1.4         0.2     setosa        5.00
51           7.0         3.2          4.7         1.4 versicolor        6.43
101          6.3         3.3          6.0         2.5  virginica        6.91
    P(4<Y<=5) P(5<Y<=6) P(6<Y<=7) P(7<Y<=8)
1       0.625     0.375     0.000     0.000
51      0.000     0.136     0.832     0.032
101     0.000     0.006     0.649     0.344
```

## 4.8 Discussion

We introduced a new optimization algorithm for gldrm, which computationally scales better with the number of unique observed response values. This is especially useful for estimation of continuous response data where the number of parameters in the $\tilde{f}_0$ parameter equals the sample size. In particular, the BFGS technique dramatically speeds up the $\tilde{f}_0$ optimization step, and makes gldrm much more computationally feasible for either discrete or continuous response data. The **gldrm** package utilizes the new optimization algorithm. Simulation results show that the algorithm and software obtain accurate estimates and standard errors. Computational time was shown to be feasible for support sizes well over one thousand.

Future research directions could include the incorporation of random effects into the gldrm framework. Optimization techniques for generalized linear mixed models, such as adaptive Gaussian quadrature, may be useful for model estimation (Pinheiro and Bates, 1995; Pinheiro and Chao, 2006).

# A Appendix to Chapter 2: Regularized Ordinal Regression and the ordinalNet R Package

## A.1 Uniqueness of the semi-parallel model estimator

This section addresses the uniqueness problem for the semi-parallel model. The semi-parallel model without the elastic net penalty is not identifiable, as there are infinitely many parametrizations for any particular model. However, different parametrizations have different elastic net penalty terms; therefore, the penalized likelihood favors some parametrizations over others. We will demonstrate that amongst almost all parametrizations for a given model, the elastic net penalty has a unique optimum; hence, the penalized likelihood has a unique optimum. There is one exception: the lasso penalty with integer-valued $\rho$. In this case, there may be a small range of optima on a closed interval.

We proceed in the following manner. First, we formulate the basic problem. Next we consider uniqueness for the ridge penalty ($\alpha = 0$), which is the simplest case. We then consider the lasso penalty ($\alpha = 1$), where this exception can occur. Finally, we consider the elastic net penalty with $\alpha \in (0, 1)$. These derivations are related to derivations for the lasso in the linear model setting (Osborne *et al.*, 2000; Rinaldo, 2008; Tibshirani, 2013).

To formulate the problem, take any row of the semi-parallel model coefficient matrix $(B_{j1}, B_{j2}, \dots, B_{jK})$ and the corresponding component of the coefficient vector, $b_j$. Denote their values as $(\delta_1, \delta_2, \dots, \delta_K)$ and $\zeta$, respectively. For any set of values $(\beta_1, \beta_2, \dots, \beta_K)$, there are an infinite number of parametrizations such that

$$(\delta_1 + \zeta, \delta_2 + \zeta, \dots, \delta_K + \zeta) = (\beta_1, \beta_2, \dots, \beta_K).$$

To see this, for any $\zeta$ set $\delta_k = \beta_k - \zeta$ for all $k$. All of these parametrizations have the same likelihood because they specify the same model, but they have different elastic net penalty terms proportional to

$$\alpha \left( \rho|\zeta| + \sum_{k=1}^{K} |\delta_k| \right) + \tfrac{1}{2}(1-\alpha) \left( \rho\zeta^2 + \sum_{k=1}^{K} \delta_k^2 \right).$$

Our goal is to find the value of $\zeta$ that minimizes the elastic net penalty.

We solve this as a constrained optimization problem, minimizing the penalty over

$(\zeta, \delta_1, \delta_2, \ldots, \delta_K)$ subject to the constraints $\delta_1 + \zeta = \beta_1$, $\delta_2 + \zeta = \beta_2$, ..., $\delta_K + \zeta = \beta_K$. This is equivalent to minimizing the Lagrangian

$$L(\zeta, \delta_1, \delta_2, \ldots, \delta_K, \lambda_1, \lambda_2, \ldots, \lambda_K) = \alpha \left( \rho|\zeta| + \sum_{k=1}^{K} |\delta_k| \right) + \tfrac{1}{2}(1-\alpha) \left( \rho\zeta^2 + \sum_{k=1}^{K} \delta_k^2 \right)$$
$$+ \lambda_1(\delta_1 + \zeta - \beta_1) + \lambda_2(\delta_2 + \zeta - \beta_1) + \cdots + \lambda_K(\delta_K + \zeta - \beta_K) \,.$$

**Ridge regression**

In this case, the Lagrangian is differentiable everywhere. Consider

$$0 \overset{\text{set}}{=} \frac{\partial L}{\partial \delta_k} = \delta_k + \lambda_k = \beta_k - \zeta + \lambda_k.$$

Solving this yields

$$\lambda_k = \zeta - \beta_k.$$

Now consider,

$$0 \overset{\text{set}}{=} \frac{\partial L}{\partial \zeta} = \rho\zeta + \lambda_1 + \lambda_2 + \cdots + \lambda_K = \rho\zeta + K\zeta - (\beta_1 + \beta_2 + \cdots + \beta_K).$$

Solving this yields

$$\zeta = \frac{1}{K+\rho}(\beta_1 + \beta_2 + \cdots + \beta_K).$$

The solution is unique for any $\rho \geq 0$.

**Lasso**

Consider

$$0 \overset{\text{set}}{=} \frac{\partial L}{\partial \delta_k} = \text{sign}(\delta_k) + \lambda_k = I\{\beta_k > \zeta\} - I\{\beta_k < \zeta\} + \lambda_k.$$

Solving this yields

$$\lambda_k = I\{\beta_k < \zeta\} - I\{\beta_k > \zeta\}.$$

Next, consider

$$\begin{aligned}\frac{\partial L}{\partial \zeta} &= \rho \cdot \text{sign}(\zeta) + \lambda_1 + \lambda_2 + \cdots + \lambda_K \\ &= \rho \cdot \text{sign}(\zeta) - \left(\sum I\{\beta_k > \zeta\} - \sum I\{\beta_k < \zeta\}\right).\end{aligned}$$

We want the solution where $\frac{\partial L}{\partial \zeta}$ equals or crosses zero. That is, we are searching for the value of $\zeta$ where $f(\zeta)$ equals or crosses $\rho$, with

$$f(\zeta) = \text{sign}(\zeta) \cdot \left(\sum I\{\beta_k > \zeta\} - \sum I\{\beta_k < \zeta\}\right).$$

Note that if $\rho \geq K$, then the solution will be $\zeta = 0$. Hence, if $\rho \geq K$, then all parallel coefficients will be penalized to zero, and the fit will be equivalent to the nonparallel model with the elastic net penalty.

Now, if $\rho$ is not an integer, then the solution is unique. If $\rho$ is an integer, then the solution could be unique, or there may be a range of solutions on a closed interval between two consecutive $\beta$'s, ranked by value.

**Elastic net**

Consider

$$0 \overset{\text{set}}{=} \frac{\partial L}{\partial \delta_k} = \alpha \cdot \text{sign}(\delta_k) + (1-\alpha)\delta_k + \lambda_k = \alpha \cdot (I\{\beta_k > \zeta\} - I\{\beta_k < \zeta\}) + (1-\alpha)(\beta_k - \zeta) + \lambda_k.$$

Solving this yields

$$\lambda_k = \alpha \cdot (I\{\beta_k < \zeta\} - I\{\beta_k > \zeta\}) + (1-\alpha)(\zeta - \beta_k).$$

Now, consider

$$\begin{aligned}\frac{\partial L}{\partial \zeta} &= \rho\alpha \cdot \text{sign}(\zeta) + \rho(1-\alpha)\zeta + \lambda_1 + \lambda_2 + \cdots + \lambda_K \\ &= \rho\alpha \cdot \text{sign}(\zeta) + \rho(1-\alpha)\zeta - \alpha\left(\sum I\{\beta_k > \zeta\} - \sum I\{\beta_k < \zeta\}\right) - \\ &\quad - (1-\alpha)(\beta_1 + \beta_2 + \cdots + \beta_K - K\zeta) \\ &= \rho\alpha \cdot \text{sign}(\zeta) + (1-\alpha)(\rho + K)\zeta - \alpha\left(\sum I\{\beta_k > \zeta\} - \sum I\{\beta_k < \zeta\}\right) - \\ &\quad - (1-\alpha)(\beta_1 + \beta_2 + \cdots + \beta_K - K\zeta).\end{aligned}$$

We want the solution where $\frac{\partial L}{\partial \zeta}$ equals or crosses zero. This solution is less transparent than that of ridge or lasso. However, the partial derivative is piecewise linear in $\zeta$ and never constant over a range of values. Hence, the solution is unique.

## A.2 The inverse link function and its Jacobian for specific MO families

Each MO family is defined by the $g_{MO}(p)$ component of its multivariate link function. For optimization, it is necessary to compute the inverse $h_{MO}(\delta)$ and its Jacobian $Dh_{MO}(\delta)$. In this section, we provide a method to compute these three functions for the cumulative probability, stopping ratio, continuation ratio, and adjacent category families. In some cases, it is convenient to compute the elements of these functions recursively. Although the Jacobian is, strictly speaking, a function of $\delta$, we write it in terms of both $p$ and $\delta$ when convenient. This can be done because there is a one-to-one correspondence between $\delta$ and $p$.

Each family has a forward and backward form. We present only one of these forms for each family. To fit the backward form, one can simply define the response categories in reverse order and fit the forward model, and vice versa.

### Forward cumulative probability family

This family is defined by $\delta_j = P(Y \leq j)$ ( see Table 2.1). From this definition, for all $j$,

$$[g_{MO}(p)]_j = \sum_{i=1}^{j} p_i.$$

Now, $h_{MO}$ has a closed form with

$$[h_{MO}(\delta)]_j = \delta_j - \delta_{j-1}, \quad \text{for all } j.$$

$Dh_{MO}$ also has a closed form with

$$[Dh_{MO}(\delta)]_{mn} = \begin{cases} \delta_m(1-\delta_m) & m = n \\ -\delta_m(1-\delta_m) & n = m-1 \\ 0 & \text{otherwise.} \end{cases}$$

### Forward stopping ratio family

This family is defined by $\delta_j = P(Y = j|Y \geq j)$ (see Table 2.1). From this definition, $[g_{MO}(p)]_1 = p_1$ and, for $j = 2, \ldots, K$,

$$[g_{MO}(p)]_j = \frac{p_j}{1 - \sum_{i=1}^{j-1} p_i} \; .$$

$h_{MO}(\delta)$ can be computed recursively, beginning with $[h_{MO}(\delta)]_1 = \delta_1$. For $j = 2, \ldots, K$,

$$[h_{MO}(\delta)]_j = \delta_j \left( 1 - \sum_{i=1}^{j-1} [h_{MO}(\delta)]_i \right) \; .$$

$Dh_{MO}(\delta)$ can also be computed recursively. For the first row we have

$$[Dh_{MO}(\delta)]_{1\cdot} = (1, 0, \ldots, 0).$$

For $m = 2, \ldots K$,

$$[Dh_{MO}(\delta)]_{m\cdot} = -\delta_m \sum_{i=1}^{m-1} [Dh_{MO}(\delta)]_{i\cdot} + \left( 1 - \sum_{i=1}^{m-1} [h_{MO}(\delta)]_i \right) \cdot (0, \ldots, 0, \underset{m^{\text{th}}}{1}, 0, \ldots, 0) \; .$$

### Forward continuation ratio family

This family is defined by $\delta_j = P(Y > j|Y \geq j)$ (Table 2.1). Let $g_{MO:FSR}$, $h_{MO:FSR}$, and $Dh_{MO:FSR}$ denote link, inverse link and inverse link Jacobian, respectively, for the forward stopping ratio family. Using these function definitions, it is straightforward to compute the corresponding functions for the forward continuation ratio family. We have

$$\begin{aligned} g_{MO}(p) &= \mathbb{1} - g_{MO:FSR}(p), \\ h_{MO}(\delta) &= h_{MO:FSR}(\mathbb{1} - \delta), \\ Dh_{MO}(\delta) &= -Dh_{MO:FSR}(\mathbb{1} - \delta). \end{aligned}$$

### Forward adjacent category family

This family is defined by $\delta_j = P(Y = j+1 | j \leq Y \leq j+1)$ (see Table 2.1). From this definition, for all $j$, we have

$$[g_{MO}(p)]_j = \frac{p_{j+1}}{p_j + p_{j+1}}.$$

Now, let $\Delta_j = \delta_j/(1-\delta_j)$. $h_{MO}(\delta)$ can be computed recursively, beginning with

$$[h_{MO}(\delta)]_1 = \frac{1}{1 + \sum_{i=1}^{K} \prod_{j=1}^{i} \Delta_j}.$$

For $j = 2, \ldots, K$,

$$[h_{MO}(\delta)]_j = [h_{MO}(\delta)]_{j-1} \Delta_{j-1}.$$

To compute $Dh_{MO}(\delta)$, we write $p = (p_1, p_1\Delta_1, p_2\Delta_2, \ldots, p_{K-1}\Delta_{K-1})$. $Dh_{MO}(\delta)$ can also be computed recursively. Beginning with the first row, we have

$$[Dh_{MO}(\delta)]_{1\cdot} = -\frac{p_1(1-p_1)}{\Delta_1}, -\frac{p_1(1-p_1-p_2)}{\Delta_2}, \ldots, -\frac{p_1(1-p_1-\cdots-p_K)}{\Delta_K}.$$

Then, for $m = 2, \ldots, K$,

$$[Dh_{MO}(\delta)]_{m\cdot} = \Delta_{m-1}[Dh_{MO}(\delta)]_{(m-1)\cdot} + p_{m-1}(0, \ldots, 0, \underset{m^{\text{th}}}{1}, 0, \ldots, 0).$$

## A.3 Quadratic approximation to the log-likelihood

The quadratic approximation $\ell^{(r)}(\beta)$ is the second order Taylor expansion of $\ell$ at $\hat{\beta}^{(r)}$, up to an additive constant that does not depend on $\beta$. Also, in $\ell^{(r)}(\beta)$, the Hessian is replaced by its expectation, $-\mathcal{I}(\hat{\beta}^{(r)})$. We give the derivation below, where $\overset{C}{=}$ denotes equality up to an additive constant. We have

$$\begin{aligned}
&\ell(\hat{\beta}^{(r)}) + (\beta - \hat{\beta}^{(r)})^T U(\hat{\beta}^{(r)}) - \tfrac{1}{2}(\beta - \hat{\beta}^{(r)})^T \, \mathcal{I}(\hat{\beta}^{(r)}) \, (\beta - \hat{\beta}^{(r)}) \\
&\overset{\mathsf{C}}{=} \beta^T U(\hat{\beta}^{(r)}) + \beta^T \mathcal{I}(\hat{\beta}^{(r)})\hat{\beta}^{(r)} - \tfrac{1}{2}\beta^T \mathcal{I}(\hat{\beta}^{(r)})\beta \\
&\overset{\mathsf{C}}{=} \beta^T X^T W^{(r)}([W^{(r)}]^{-1} v^{(r)} + X\hat{\beta}^{(r)}) - \tfrac{1}{2}\beta^T X^T W^{(r)} X\beta \\
&\overset{\mathsf{C}}{=} \beta^T X^T W^{(r)} z^{(r)} - \tfrac{1}{2}\beta^T X^T W^{(r)} X\beta - \tfrac{1}{2}[z^{(r)}]^T W^{(r)} z^{(r)} \\
&= -\tfrac{1}{2}(z^{(r)} - X\beta)^T \, W^{(r)} \, (z^{(r)} - X\beta) \\
&= \ell^{(r)}(\beta).
\end{aligned}$$

## A.4 Statistical inference

Goeman *et al.* (2017) assert that while standard errors of penalized regression coefficients could be calculated, e.g. by the bootstrap, these standard errors should be interpreted with care. In particular, they should not be used to construct confidence intervals. This is because penalized regression coefficient estimators have substantial bias. For this reason, they deliberately do not provide standard error estimates in the **penalized** package. In the classical setting, where the number of observations exceeds the number of predictors, then one can use the usual likelihood-based methods to construct confidence intervals and hypothesis tests. However, even in this setting, such methods can lead to incorrect inference (Zhang, 1992; Leeb and Pötscher, 2005; Wang and Lagakos, 2009; Berk *et al.*, 2013).

Tibshirani and Taylor (2011), Lockhart *et al.* (2014), and Tibshirani *et al.* (2016) laid the groundwork of an asymptotic inference method for penalized regression. They proved the asymptotic distribution of their test statistic specifically for least squares regression, but their simulation results suggest that the same test statistic could be used for generalized linear models. This may be a path toward high dimensional inference for penalized multinomial and ordinal regression models.

# B APPENDIX TO CHAPTER 4: SEMIPARAMETRIC GENERALIZED LINEAR MODELS AND THE GLDRM R PACKAGE

## B.1 Notation

In this appendix, we use $\partial$ to denote the partial derivative and $d$ to denote the total derivative. We require this notation for chain rule derivatives, specifically to derive the score function with respect to $\beta$ and $\tilde{f}_0$. Each $\theta_i$ is implicitly a function of $(\beta, \tilde{f}_0)$ and hence should not be held fixed when differentiating the log-likelihood with respect to either $\beta$ or $\tilde{f}_0$.

We also let

$$\ell_i = \theta_i y_i - \log \sum_{k=1}^{K} f_k \exp(\theta_i s_k) + \sum_{k=1}^{K} I(y_i = s_k) \log f_k \tag{B.1}$$

denote the contribution of the $i^{\text{th}}$ observation to the log-likelihood.

## B.2 $\tilde{f}_0$ score function

We derive Equation 4.12, which gives the score function with respect to $\tilde{f}_0$, holding $\beta$ fixed.

Using the definition of $\ell_i$ in Equation B.1 and applying the chain rule, the $k^{\text{th}}$ element of the score function is

$$\{S(\tilde{f}_0; \beta)\}_k = \frac{d\ell(\beta, \tilde{f}_0)}{df_k} = \sum_{i=1}^{n} \frac{d\ell_i}{df_k} = \sum_{i=1}^{n} \left( \frac{\partial \ell_i}{\partial f_k} + \frac{d\theta_i}{df_k} \frac{\partial \ell_i}{\partial \theta_i} \right) . \tag{B.2}$$

The first term on the RHS of Equation B.2 is

$$\frac{\partial \ell_i}{\partial f_k} = \frac{I(y_i = s_k)}{f_k} - \frac{\exp\{\theta_i s_k\}}{\sum\limits_{m=1}^{K} f_m \exp\{\theta_i s_m\}} . \tag{B.3}$$

The second term on the RHS of Equation B.2 is

$$\frac{\partial \ell_i}{\partial \theta_i} = y_i - \frac{\sum\limits_{k=1}^{K} s_k f_k \exp\{\theta s_k\}}{\sum\limits_{k=1}^{K} f_k \exp\{\theta s_k\}} = y_i - \mu_i . \tag{B.4}$$

Recall that $\mu_i = g^{-1}(x_i^T\beta)$, which does not vary with $\tilde{f}_0$. Therefore,

$$0 = \frac{d}{df_k}\mu_i = \frac{\partial\mu_i}{\partial f_k} + \frac{d\theta_i}{df_k}\frac{\partial\mu_i}{\partial\theta_i} \ , \tag{B.5}$$

and the third term on the RHS of Equation B.2 is

$$\frac{d\theta_i}{df_k} = -\frac{\partial\mu_i}{\partial f_k}\Big/\frac{\partial\mu_i}{\partial\theta_i} = -\frac{\exp\{\theta_i s_k\}(s_k - \mu_i)}{\exp\{b(\theta_i)\}}\Big/ b''(\theta_i) \ . \tag{B.6}$$

Plugging the results of Equations B.3, B.4, and B.6 into Equation B.2, we obtain

$$\{S(\tilde{f}_0;\beta)\}_k = \sum_{i=1}^{n}\left\{\frac{I(y_i = s_k)}{f_k} - \frac{\exp(\theta_i s_k)}{\exp\{b(\theta_i)\}} - \frac{\exp(\theta_i s_k)}{\exp\{b(\theta_i)\}}\frac{s_k - \mu_i}{b''(\theta_i)}(y_i - \mu_i)\right\} \ . \tag{B.7}$$

We obtain the result of Equation 4.12 by applying the Jacobian transformation $\{S(\tilde{g}_0;\beta)\}_k = f_k\{S(\tilde{f}_0;\beta)\}_k$.

## B.3 $\tilde{f}_0$ information matrix

The $\tilde{f}_0$ information matrix is not used during the BFGS optimization routine. However, we derive it here to correct a missing term in the Rathouz and Gao (2009) derivation. Each matrix element is

$$\begin{aligned} \left\{\mathcal{I}(\tilde{f}_0;\beta)\right\}_{(k,m)} &= \mathrm{E}\left(\left\{S(\tilde{f}_0;\beta)\right\}_k \left\{S(\tilde{f}_0;\beta)\right\}_m \middle| X_k = x_k, X_m = x_m\right) \\ &= \sum_{i=1}^{n} \mathrm{E}\left\{A_k A_m - 2A_k B_m + B_k B_m \middle| X_k = x_k, X_m = x_m\right\} , \end{aligned} \tag{B.8}$$

where

$$A_k = \frac{I(y_i = s_k)}{f_k} - \frac{\exp(\theta_i s_k)}{\exp\{b(\theta_i\}}$$

and

$$B_k = \frac{\exp(\theta_i s_k)}{\exp\{b(\theta_i)\}} - \frac{\exp(\theta_i s_k)}{\exp\{b(\theta_i)\}} \frac{s_k - \mu_i}{b''(\theta_i)} (y_i - \mu_i) \; .$$

From here on, all expectations are conditional on $X_k = x_k, X_m = x_m$, but we suppress the conditional expression. The first term on the RHS of Equation B.8 is

$$\mathrm{E}(A_k A_m) = \frac{\exp(\theta_i s_k) I(k = m)}{f_k \exp\{b(\theta_i)\}} - \frac{\exp(\theta_i s_k + \theta_i s_m)}{\exp\{2b(\theta_i)\}} \; . \tag{B.9}$$

The second term on the RHS of Equation B.8 is

$$E(B_k B_m) = \frac{\exp(\theta_i s_k + \theta_i s_m)}{\exp\{2b(\theta_i)\}} \frac{(s_k - \mu_i)(s_m - \mu_i)}{b''(\theta_i)} \; , \tag{B.10}$$

where we use the fact that $\mathrm{E}\{(y_i - \mu_i)^2\} = \mathrm{Var}(y_i) = b''(\theta_i)$. The third term on the RHS of Equation B.8 is

$$\mathrm{E}(A_k B_m) = \frac{\exp(\theta_i s_k + \theta_i s_m)}{\exp\{2b(\theta_i)\}} \frac{(s_k - \mu_i)(s_m - \mu_i)}{b''(\theta_i)} \; , \tag{B.11}$$

where we use the fact that $\mathrm{E}\{I(y_i = s_k)(y_i - \mu_i)\} = (y_k - \mu_i) f_k \exp(\theta_i s_k)/\exp\{b(\theta_i)\}$. Plugging the results of Equations B.9-B.11 into Equation B.8, we obtain

$$\begin{aligned}\{\mathcal{I}(\tilde{f}_0;\beta)\}_{(k,m)} = \sum_{i=1}^{n} \Bigg\{ & \frac{\exp(\theta_i s_k) I(k=m)}{f_k \exp\{b(\theta_i)\}} - \frac{\exp(\theta_i s_k + \theta_i s_m)}{\exp\{2b(\theta_i)\}} - \\ & - \frac{\exp(\theta_i s_k + \theta_i s_m)}{\exp\{2b(\theta_i)\}} \frac{(s_k - \mu_i)(s_m - \mu_i)}{b''(\theta_i)} \Bigg\} .\end{aligned} \tag{B.12}$$

## B.4 $\beta$ score function

We derive Equation 4.15, which gives the score function with respect to $\beta$, holding $\tilde{f}_0$ fixed. By the chain rule, each element is

$$S(\beta;\tilde{f}_0) = \frac{d\ell(\beta,\tilde{f}_0)}{d\beta} = \sum_{i=1}^{n} \frac{d\ell_i}{d\beta} = \sum_{i=1}^{n} \frac{d\mu_i}{d\beta}\frac{d\theta_i}{d\mu_i}\frac{\partial \ell_i}{\partial \theta_i} \tag{B.13}$$

We already derived the first term on the RHS of Equation B.13 in Equation B.4. Because $\mu_i = b'(\theta_i)$,

$$\frac{d\theta_i}{d\mu_i} = 1/\left(\frac{d\mu_i}{d\theta_i}\right) = \frac{1}{b''(\theta_i)} . \tag{B.14}$$

Because $g(\mu_i) = x_i^T \beta$,

$$\frac{d\mu_i}{d\beta} = x_i \left(\frac{1}{g'(\mu_i)}\right) . \tag{B.15}$$

Plugging the results of Equations B.4, B.14, and B.15 into Equation B.13, we obtain the result of Equation 4.15.

## B.5 Proof that transformed response with modified link function has equivalent log-likelihood

Let $\nu \equiv \frac{m+M}{2}$ and $\rho \equiv \frac{M-m}{2}$. Let $y_i^* = (y_i - \nu)/\rho$ and $s_k^* = (s_k - \nu)/\rho$. Equation B.1 can be rewritten

$$\begin{aligned} \ell_i &= \theta_i(\nu + \rho y_i^*) - \log \sum_{k=1}^{K} f_k \exp\left\{\theta_i(\nu + \rho s_k^*)\right\} + \sum_{k=1}^{K} I(y_i^* = s_k^*) \log f_k \\ &= \theta_i^* y_i^* - \log \sum_{k=1}^{K} f_k \exp\{\theta_i^* s_k^*\} + \sum_{k=1}^{K} I(y_i^* = s_k^*) \log f_k \ , \end{aligned} \tag{B.16}$$

where $\theta_i^* = \theta_i \rho$. Equation 4.6 can be rewritten as

$$\begin{aligned} g^{-1}(x_i^T\beta) &= \frac{\sum_{k=1}^{K} (\nu + \rho s_k^*) f_k \exp\left\{\theta_i(\nu + \rho s_k^*)\right\}}{\sum_{k=1}^{K} f_k \exp\{\theta_i(\nu + \rho s_k^*)\}} \\ &= \nu + \rho \left( \frac{\sum_{k=1}^{K} s_k^* f_k \exp\left\{\theta_i^* s_k^*\right\}}{\sum_{k=1}^{K} f_k \exp\{\theta_i^* s_k^*\}} \right) \ . \end{aligned} \tag{B.17}$$

Hence, if we define a modified link function $g_*(\mu) = g(\nu + \rho\mu)$ with inverse $g_*^{-1}(\eta) = (g^{-1}(\eta) - \nu)/\rho$, we can write

$$g_*^{-1}(x_i^T\beta) = \left( \frac{\sum_{k=1}^{K} s_k^* f_k \exp\{\theta_i^* s_k^*\}}{\sum_{k=1}^{K} f_k \exp\{\theta_i^* s_k^*\}} \right) \ . \tag{B.18}$$

In other words $\theta_i^*$ has the same implicit definition as $\theta_i$ when the modified link function and transformed response variable are used in place of the orginal.

This shows that, as a function of $(\beta, \tilde{f}_0)$, the log-likelihood with the transformed response and modified link function is equivalent to the original log-likelihood. Note that $\theta_i^*$ must be multiplied by $1/\rho$ if one would like calculate $\theta_i$ on the original scale.

Andersen PK, Klein JP, Knudsen KM, y Palacios RT (1997). "Estimation of variance in Cox's regression model with shared gamma frailties." *Biometrics*, pp. 1475–1484. doi:10.2307/2533513.

Archer KJ (2014a). *glmnetcr: Fit a penalized constrained continuation ratio model for predicting an ordinal response*. R package version 1.0.2, URL `https://CRAN.R-project.org/package=glmnetcr`.

Archer KJ (2014b). *glmpathcr: Fit a penalized continuation ratio model for predicting an ordinal response*. R package version 1.0.3, URL `https://CRAN.R-project.org/package=glmpathcr`.

Archer KJ, Hou J, Zhou Q, Ferber K, Layne JG, Gentry A (2016). *ordinalgmifs: Ordinal regression for high-dimensional data*. R package version 1.0.3, URL `https://CRAN.R-project.org/package=ordinalgmifs`.

Archer KJ, Hou J, Zhou Q, Ferber K, Layne JG, Gentry AE (2014). "ordinalgmifs: An R package for ordinal regression in high-dimensional data settings." *Cancer Informatics*, **13**, 187. doi:10.4137/CIN.S20806.

Archer KJ, Mas VR, Maluf DG, Fisher RA (2010). "High-throughput assessment of CpG site methylation for distinguishing between HCV-cirrhosis and HCV-associated hepatocellular carcinoma." *Molecular Genetics and Genomics*, **283**(4), 341–349. doi:10.1007/s00438-010-0522-y.

Arlot S, Celisse A (2010). "A survey of cross-validation procedures for model selection." *Statistics Surveys*, **4**, 40–79. doi:10.1214/09-SS054.

Berk R, Brown L, Buja A, Zhang K, Zhao L (2013). "Valid post-selection inference." *The Annals of Statistics*, **41**(2), 802–837. doi:10.1214/12-AOS1077.

Bhoola S, Hoskins WJ (2006). "Diagnosis and management of epithelial ovarian cancer." *Obstetrics & Gynecology*, **107**(6), 1399–1410. doi:10.1097/01.AOG.0000220516.34053.48.

Bickel PJ, Li B, B TA, *et al.* (2006). "Regularization in statistics." *Test*, **15**(2), 271–344. doi:10.1007/BF02607055.

Breiman L (1995). "Better subset regression using the nonnegative garrote." *Technometrics*, **37**(4), 373–384. doi:10.1080/00401706.1995.10484371.

Breslow N (1974). "Covariance analysis of censored survival data." *Biometrics*, **30**(1), 89–99. doi:10.2307/2529620.

Broyden CG (1970). "The convergence of a class of double-rank minimization algorithms." *IMA Journal of Applied Mathematics*, **6**(3), 222–231. doi:10.1093/imamat/6.3.222.

Cox DR (1972). "Regression models and life-tables." *Journal of the Royal Statistical Society, Series B*, **34**(2), 187–220. ISSN 00359246. URL `http://www.jstor.org/stable/2985181`.

Dempster AP, Laird NM, Rubin DB (1977). "Maximum likelihood from incomplete data via the EM algorithm." *Journal of the Royal Statistical Society, Series B*, pp. 1–38. URL `http://www.jstor.org/stable/2984875`.

Eddelbuettel D (2013). *Seamless R and C++ integration with Rcpp*. Springer. doi:10.1007/978-1-4614-6868-4.

Eddelbuettel D, François R (2011). "Rcpp: Seamless R and C++ integration." *Journal of Statistical Software*, **40**(8), 1–18. doi:10.18637/jss.v040.i08.

Eddelbuettel D, Sanderson C (2014). "RcppArmadillo: Accelerating R with high-performance C++ linear algebra." *Computational Statistics & Data Analysis*, **71**, 1054–1063. doi:10.1016/j.csda.2013.02.005.

Eng KH, Hanlon BM (2014). "Discrete mixture modeling to address genetic heterogeneity in time-to-event regression." *Bioinformatics*, **30**(12), 1690–1697. doi:10.1093/bioinformatics/btu065.

Fisher RA (1925). "Theory of statistical estimation." In "Mathematical Proceedings of the Cambridge Philosophical Society," volume 22, pp. 700–725. Cambridge University Press. doi:10.1017/S0305004100009580.

Fletcher R (1970). "A new approach to variable metric algorithms." *The Computer Journal*, **13**(3), 317–322. doi:10.1093/comjnl/13.3.317.

Friedman J, Hastie T, Höfling H, Tibshirani R (2007). "Pathwise coordinate optimization." *The Annals of Applied Statistics*, **1**(2), 302–332. doi:10.1214/07-AOAS131.

Friedman J, Hastie T, Tibshirani R (2010). "Regularization paths for generalized linear models via coordinate descent." *Journal of Statistical Software*, **33**(1), 1–22. doi:10.18637/jss.v033.i01.

Goeman JJ, Meijer RJ, Chaturvedi N (2017). *Penalized: L1 (lasso and fused lasso) and L2 (ridge) penalized estimation in GLMs and in the Cox model.* R package version 0.9-50, URL `https://CRAN.R-project.org/package=penalized`.

Goldfarb D (1970). "A family of variable-metric methods derived by variational means." *Mathematics of Computation*, **24**(109), 23–26. doi:10.1090/S0025-5718-1970-0258249-6.

Harrell Jr FE (2015). *Regression modeling strategies: With applications to linear models, logistic and ordinal regression, and survival analysis*. Springer, 2nd edition. doi:10.1007/978-3-319-19425-7.

Harrell Jr FE (2017). *rms: Regression Modeling Strategies*. R package version 5.1-0, URL `https://CRAN.R-project.org/package=rms`.

Hastie T, Tibshirani R, Friedman J (2009). *The Elements of Statistical Learning*. Springer, 2nd edition. doi:10.1007/b94608.

Henderson NC, Rathouz PJ (2015). "AR(1) Latent Class Models for Longitudinal Count Data." *ArXiv e-prints*. `1501.05961`.

Hesterberg T, Choi NH, Meier L, Fraley C (2008). "Least angle and $\ell_1$ penalized regression: A review." *Statistics Surveys*, **2**, 61–93. doi:10.1214/08-SS035.

Huang A (2014). "Joint estimation of the mean and error distribution in generalized linear models." *Journal of the American Statistical Association*, **109**(505), 186–196. doi:10.1080/01621459.2013.824892.

Huang A, Rathouz PJ (2012). "Proportional likelihood ratio models for mean regression." *Biometrika*, **99**(1), 223–229. doi:10.1093/biomet/asr075.

Jamshidian M, Jennrich RI (2000). "Standard errors for EM estimation." *Journal of the Royal Statistical Society, Series B*, **62**(2), 257–270. doi:10.1111/1467-9868.00230.

Javanmard A, Montanari A (2014). "Confidence intervals and hypothesis testing for high-dimensional regression." *Journal of Machine Learning Research*, **15**(1), 2869–2909. URL `http://jmlr.org/papers/v15/javanmard14a.html`.

Khalili A, Chen J (2007). "Variable selection in finite mixture of regression models." *Journal of the American Statistical Association*, **102**(479), 1025–1038. doi:10.1198/016214507000000590.

Leeb H, Pötscher BM (2005). "Model selection and inference: Facts and fiction." *Econometric Theory*, **21**(01), 21–59. doi:10.1017/S0266466605050036.

Lockhart R, Taylor J, Tibshirani RJ, Tibshirani R (2014). "A significance test for the lasso." *Annals of Statistics*, **42**(2), 413. doi:10.1214/13-AOS1175.

Louis TA (1982). "Finding the observed information matrix when using the EM algorithm." *Journal of the Royal Statistical Society, Series B*, pp. 226–233. URL `http://www.jstor.org/stable/2345828`.

Lovric M (2011). *International Encyclopedia of Statistical Science.* Springer.

McCullagh P (1980). "Regression models for ordinal data." *Journal of the Royal Statistical Society, Series B*, pp. 109–142. URL `http://www.jstor.org/stable/2984952`.

McLachlan G, Krishnan T (2007). *The EM algorithm and extensions*, volume 382. John Wiley & Sons, 2nd edition. doi:10.1002/9780470191613.

McLachlan G, Peel D (2004). *Finite mixture models*. John Wiley & Sons. doi:10.1002/0471721182.

Meng XL, Rubin DB (1991). "Using EM to obtain asymptotic variance-covariance matrices: The SEM algorithm." *Journal of the American Statistical Association*, **86**(416), 899–909. doi:10.1080/01621459.1991.10475130.

Murphy SA (1995). "Asymptotic theory for the frailty model." *The Annals of Statistics*, pp. 182–198. doi:10.1214/aos/1176324462.

Osborne MR, Presnell B, Turlach BA (2000). "On the lasso and its dual." *Journal of Computational and Graphical Statistics*, **9**(2), 319–337. doi:10.1080/10618600.2000.10474883.

Owen AB, Crc H, Raton B, New L, Washington Y, B AA, Owen A (2001). *Empirical Likelihood*. CRC Press. ISBN 1584880716.

Park MY, Hastie T (2007). "L1-regularization path algorithm for generalized linear models." *Journal of the Royal Statistical Society, Series B*, **69**(4), 659–677. doi:10.1111/j.1467-9868.2007.00607.x.

Parner E (1998). "Asymptotic theory for the correlated gamma-frailty model." *The Annals of Statistics*, **26**(1), 183–214. doi:10.1214/aos/1030563982.

Peterson B, Harrell FE, Petersont B (1990). "Partial proportional odds models for ordinal response variables." *Journal of the Royal Statistical Society, Series C*, **39**(2), 205–217. doi:10.2307/2347760.

Pinheiro JC, Bates DM (1995). "Approximations to the log-likelihood function in the nonlinear mixed-effects model." *Journal of Computational and Graphical Statistics*, **4**(1), 12–35. doi:10.1080/10618600.1995.10474663.

Pinheiro JC, Chao EC (2006). "Efficient Laplacian and adaptive Gaussian quadrature algorithms for multilevel generalized linear mixed models." *Journal of Computational and Graphical Statistics*, **15**(1), 58–81. doi:10.1198/106186006X96962.

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL `http://www.r-project.org/`.

Rathouz PJ, Gao L (2009). "Generalized linear models with unspecified reference distribution." *Biostatistics*, **10**(2), 205–218. doi:10.1093/biostatistics/kxn030.

Rinaldo A (2008). "A Note on the Uniqueness of the Lasso Solution." *Technical report*, Department of Statistics Carnegie Mellon University. URL `http://www.stat.cmu.edu/research/publications_and_reports/technical_reports`.

SAS Institute Inc (2017). *SAS/STAT User's Guide Procedures*. SAS Institute Inc., Cary, NC. URL `http://support.sas.com/documentation/onlinedoc/stat/indexproc.html`.

Schepers J (2015). "Improved random-starting method for the EM algorithm for finite mixtures of regressions." *Behavior Research Methods*, **47**(1). doi:10.3758/s13428-014-0468-9.

Schifano ED, Strawderman RL, Wells MT, Others (2010). "Majorization-minimization algorithms for nonsmoothly penalized objective functions." *Electronic Journal of Statistics*, **4**, 1258–1299. doi:10.1214/10-EJS582.

Shanno DF (1970). "Conditioning of quasi-Newton methods for function minimization." *Mathematics of Computation*, **24**(111), 647–656. doi:10.1090/S0025-5718-1970-0274029-X.

Städler N, Bühlmann P, Van De Geer S (2010). "L1-penalization for mixture regression models." *Test*, **19**(2), 209–256. doi:10.1007/s11749-010-0197-z.

Sun W, Wang J, Fang Y (2013). "Consistent selection of tuning parameters via variable selection stability." *Journal of Machine Learning Research*, **14**(1), 3419–3440. URL `http://jmlr.org/papers/v14/sun13b.html`.

Tibshirani R (1996). "Regression shrinkage and selection via the lasso." *Journal of the Royal Statistical Society, Series B*, pp. 267–288. URL `http://www.jstor.org/stable/2346178`.

Tibshirani RJ (2013). "The lasso problem and uniqueness." *Electronic Journal of Statistics*, **7**, 1456–1490. doi:10.1214/13-EJS815.

Tibshirani RJ, Taylor J (2011). "The solution path of the generalized lasso." *Annals of Statististics*, **39**(3), 1335–1371. doi:10.1214/11-AOS878.

Tibshirani RJ, Taylor J, Lockhart R, Tibshirani R (2016). "Exact post-selection inference for sequential regression procedures." *Journal of the American Statistical Association*, **111**(514), 600–620. doi:10.1080/01621459.2015.1108848.

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S*. Springer, New York, fourth edition. doi:10.1007/978-0-387-21706-2.

Vidaurre D, Bielza C, Larrañaga P (2013). "A survey of L1 regression." *International Statistical Review*, **81**(3), 361–387. doi:10.1111/insr.12023.

Wang R, Lagakos SW (2009). "Inference after variable selection using restricted permutation methods." *Canadian Journal of Statistics*, **37**(4), 625–644. doi:10.1002/cjs.10039.

Wilhelm MS, Carter EM, Hubert JJ (1998). "Multivariate iteratively re-weighted least squares, with applications to dose-response data." *Environmetrics*, **9**(3), 303–315. doi:10.1002/(SICI)1099-095X(199805/06)9:3<303::AID-ENV305>3.0.CO;2-1.

Wurm M, Rathouz P (2017). *gldrm: Generalized Linear Density Ratio Models*. R package version 1.1, URL `https://CRAN.R-project.org/package=gldrm`.

Wurm M, Rathouz P, Hanlon B (2017). *ordinalNet: Penalized Ordinal Regression*. R package version 2.2, URL `https://CRAN.R-project.org/package=ordinalNet`.

Yee TW (2010). "The VGAM Package for Categorical Data Analysis." *Journal of Statistical Software*, **32**(10), 1–34. doi:10.18637/jss.v032.i10.

Yee TW (2015). *Vector Generalized Linear and Additive Models: With an Implementation in R*. Springer. doi:10.1007/978-1-4939-2818-7.

Yee TW (2017). *VGAM: Vector generalized linear and additive models*. R package version 1.0-4, URL `https://CRAN.R-project.org/package=VGAM`.

Yee TW, Wild CJ (1996). "Vector Generalized Additive Models." *Journal of Royal Statistical Society, Series B*, **58**(3), 481–493. URL `http://www.jstor.org/stable/2345888`.

Zhang P (1992). "Inference after variable selection in linear regression models." *Biometrika*, **79**(4), 741–746. doi:10.1093/biomet/79.4.741.

Zou H, Hastie T (2005). "Regularization and variable selection via the elastic net." *Journal of the Royal Statistical Society, Series B*, **67**(2), 301–320. doi:10.1111/j.1467-9868.2005.00503.x.