Journal of Statistical Software

September 2021, Volume 99, Issue 6. doi: 10.18637/jss.v099.i06

Regularized Ordinal Regression and the ordinalNet R Package

Michael J. Wurm
University of Wisconsin–Madison

Paul J. Rathouz
University of Texas at Austin

Bret M. Hanlon
University of Wisconsin–Madison

Abstract

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, that can be used to model either ordered or unordered categorical response data. We call this the *elementwise link multinomial-ordinal* class, and it includes widely used models such as multinomial logistic regression (which also has an ordinal form) and ordinal logistic regression (which also has an unordered multinomial form). We introduce an elastic net penalty class that applies to either model form, and 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.

Keywords: ordinal regression, multinomial regression, variable selection, lasso, elastic net.

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 ordinal logistic regression (McCullagh 1980), which is also known as the cumulative logit or proportional odds model. Other ordinal regression models include the stopping ratio model, the continuation ratio model, and the adjacent category model.

The prediction accuracy of regression models can often be improved by regularization such as lasso (Tibshirani 1996) and elastic net (Zou and Hastie 2005), particularly in cases where the number of predictors is large relative to the sample size and especially when the number of predictors exceeds the sample size. Regularization shrinks the coefficient estimators toward zero, resulting in estimators that are biased but with reduced variance. With an appropriate degree of regularization, this trade-off can improve prediction accuracy. Furthermore, the lasso and elastic net shrink some coefficients to zero exactly, so only those covariates with the most predictive power are actually used to calculate predictions. This is an effective method for variable selection in a high dimensional predictor space.

A variety of software packages exist for unpenalized ordinal regression. In R, **VGAM** (Yee and Wild 1996; Yee 2010, 2015, 2021) is a comprehensive software package for ordinal models, including all those mentioned in the first paragraph. The SAS CATMOD procedure also fits some of these models (SAS Institute Inc 2017). However, these packages do not currently have functionality for regularization or variable selection. Popular CRAN packages for penalized regression, such as **penalized** (Goeman, Meijer, and Chaturvedi 2018) and **glmnet** (Friedman, Hastie, and Tibshirani 2010; Friedman, Hastie, Tibshirani, Simon, Narasimhan, and Qian 2021), do not currently fit ordinal models.

Some algorithms and software do already exist for penalized ordinal regression. The `lrm` function in the R package **rms** (Harrell, Jr. 2015, 2021) fits the cumulative logit model with quadratic (ridge regression) penalty. The R packages **glmnetcr** (Archer 2020a) and **glmpathcr** (Archer 2020b) fit stopping ratio models with the elastic net penalty. (Note that we follow the naming convention of Yee (2010) in using the term "stopping ratio", although some authors, including Archer (2020a,b), use the term "continuation ratio" for these models.)

Archer, Hou, Zhou, Ferber, Layne, and Gentry (2014, 2019) 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, Tibshirani, and Friedman 2009). A drawback of this algorithm is that it does not have the flexibility of the elastic net mixing parameter (usually denoted by α). It can also be computationally expensive for some use cases because the 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 λ). 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, existing 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.

1. 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.

2. 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.

A 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.

The ELMO class includes ordinal logistic regression (a.k.a., the proportional odds model), which is a parallel model that has a nonparallel counterpart, the partial proportional odds model (Peterson and Harrell, Jr. 1990). Another example from this class is multinomial logistic regression (see, e.g., Agresti (2003)), which is a nonparallel model that also has a parallel counterpart. Note that the usual multinomial logistic regression parameterization does not fit the ELMO class; however, the ELMO class does contain an alternative but equivalent parameterization called the nonparallel adjacent category model with logit link. For more details, see Appendix A.

3. 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 overparameterizing the nonparallel model to include both the nonparallel and parallel coefficients. We call this alternate parameterization the *semi-parallel* model. Although the regression model itself is not identifiable under this parameterization, the penalized likelihood has a unique optimum (or almost unique, as discussed in Appendix B). Tutz and Gertheiss (2016) discuss related ideas about penalized regression models for ordinal response data that allow the parallelism assumption to be relaxed.

We provide an outline for the remainder of the work. Section 2 defines the ELMO class with specific examples. We also define the parallel, nonparallel, and semi-parallel parameterizations with the elastic net penalty. Section 3 provides the proposed algorithm for fitting multinomial regression models with the elastic net penalty. Section 4 presents a simulation study to compare prediction accuracy of the penalized parallel, nonparallel, and semi-parallel models. Section 5 demonstrates the use of penalized ELMO class models for out-of-sample prediction and variable selection alongside other methods. Section 6 provides details about the **ordinalNet** R package (Wurm, Rathouz, and Hanlon 2021), which is available on the Comprehensive R Archive Network at `https://CRAN.R-project.org/package=ordinalNet`. Section 7 provides a demonstration of the **ordinalNet** package. Section 8 contains a summary of the findings and contributions.

2. Elementwise link multinomial-ordinal (ELMO) class

This section is organized as follows. Section 2.1 introduces some common notation used in this paper. Section 2.2 is the heart of Section 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 2.3 and Section 2.4 provide details for the family function and elementwise link function, respectively. The parallel and nonparallel forms are introduced in Section 2.5, and the semi-parallel form is introduced in Section 2.6. Finally, Section 2.7 discusses the elastic net penalty and formulates the objective function under the three model forms.

2.1. Notation

We introduce some common notation used in this paper. Other paper-specific notation is developed throughout the work. For a vector x, we use $x^\top$ 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 ∇ 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

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

in other words

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

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.

The data consists of vector pairs (y_i, x_i) 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 $y_i = (y_{i1}, \ldots, y_{i(K+1)})^\top$ is a random vector of counts summing to n_i. We assume the conditional distribution $y_i \mid x_i \overset{\text{indep.}}{\sim} \text{Multinomial}(n_i, p_{i1}, p_{i2}, \ldots, p_{i(K+1)})$. In other words, we have n_i independent trials which fall into $K + 1$ classes with probabilities $(p_{i1}, p_{i2}, \ldots, p_{i(K+1)})$. These probabilities are a function of x_i, and the choice of ELMO model determines this function. The $K + 1$ probabilities sum to one, so they can be parameterized by the first K values, $p_i = (p_{i1}, p_{i2}, \ldots, p_{iK})^\top$.

Let P be the length of x_i and let B be a $P \times K$ matrix of regression coefficients. Let c be a vector of K intercept values. The covariates are mapped to a vector of K linear predictors, $\eta_i = (\eta_{i1}, \eta_{i2}, \ldots, \eta_{iK})$, by the relationship $\eta_i = c + B^\top x_i$.

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, $g_{\text{EL}} : (0,1)^K \to \mathbb{R}^K$ and $g_{\text{MO}} : \mathcal{S}^K \to (0,1)^K$. The function g_{EL} simply applies a function $\tilde{g}_{\text{EL}} : (0,1) \to \mathbb{R}$ elementwise to a vector of length K. More concisely, ELMO class models have a link function of the form

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

where

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

and

$$g_{\text{EL}}(\delta) = (\tilde{g}_{\text{EL}}(\delta_1), \ldots, \tilde{g}_{\text{EL}}(\delta_K))^\top \ .$$

To summarize, an ELMO model specifies a family function g_{MO} and an elementwise link function $\tilde{g}_{\text{EL}}$ such that

$$B^\top x_i = \begin{pmatrix} \eta_{i1} \\ \eta_{i2} \\ \vdots \\ \eta_{iK} \end{pmatrix} = \begin{pmatrix} \tilde{g}_{\text{EL}}(\delta_{i1}) \\ \tilde{g}_{\text{EL}}(\delta_{i2}) \\ \vdots \\ \tilde{g}_{\text{EL}}(\delta_{iK}) \end{pmatrix} = g_{\text{EL}} \left(\begin{pmatrix} \delta_{i1} \\ \delta_{i2} \\ \vdots \\ \delta_{iK} \end{pmatrix} \right) = (g_{\text{EL}} \circ g_{\text{MO}}) \left(\begin{pmatrix} p_{i1} \\ p_{i2} \\ \vdots \\ p_{iK} \end{pmatrix} \right) = g(p_i) \ .$$

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 2.5). Examples of MO families are given in Table 1.

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

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

Let $r(p_i) = (p_{i(K+1)}, p_{iK}, \ldots, p_{i2})$ 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 $(g_{\text{MO}} \circ r)$ 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

MO family	δ_k
Cumulative Probability (forward)	$\mathsf{P}(Y \leq k)$
Cumulative Probability (backward)	$\mathsf{P}(Y \geq k+1)$
Stopping Ratio (forward)	$\mathsf{P}(Y = k \mid Y \geq k)$
Stopping Ratio (backward)	$\mathsf{P}(Y = k+1 \mid Y \leq k+1)$
Continuation Ratio (forward)	$\mathsf{P}(Y > k \mid Y \geq k)$
Continuation Ratio (backward)	$\mathsf{P}(Y < k \mid Y \leq k)$
Adjacent Category (forward)	$\mathsf{P}(Y = k+1 \mid k \leq Y \leq k+1)$
Adjacent Category (backward)	$\mathsf{P}(Y = k \mid k \leq Y \leq k+1)$

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

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 $g_{\text{MO}}^*(p) = 1 - g_{\text{MO}}(p)$ 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.

2.4. Elementwise link function

The elementwise link function $\tilde{g}_{\text{EL}} : (0,1) \to \mathbb{R}$ 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 elementwise link functions satisfy is symmetry, that is

$$\tilde{g}_{\text{EL}}(\delta) = -\tilde{g}_{\text{EL}}(1-\delta) \ .$$

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 $\tilde{g}_{\text{EL}}$ is a non-symmmetric function, such as complementary log-log, then none of these equivalences hold.

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. Yee (2010) refer to this restriction as the parallelism assumption. Under the parallelism assumption, we can write $\eta_i = b^\top x_i + c$, where b is a vector. Now consider the distribution of $y_i \mid x_i$. The Monotonicity Property of g_{MO}, along with the monotonicity requirement on $\tilde{g}_{\text{EL}}$, ensures that a change in $b^\top x_i$ 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.

We point out here that the parallel cumulative probability model with logit link (either forward or backward) is the ordinal logistic regression model (a.k.a. cumulative logit or proportional odds model). We also point out that the nonparallel adjacent category model with logit link (either forward or backward) is an alternative but equivalent parameterization of multinomial logistic regression (see Appendix A for details).

A word of caution: the nonparallel *cumulative probability* model must a have linear predictor vector, $B^\top x$, that is monotone to ensure that the cumulative probabilities are non-decreasing. Specifically, $B^\top x$ must be monotone increasing for the forward model and monotone decreasing for the backward model. Thus, B should be constrained such that $B^\top x$ 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 $B^\top x$ 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 1 do not have any restriction on the parameter space.

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. If the number of observations is large enough to accurately estimate the nonparallel model by maximum likelihood, one might prefer this option. After all, the nonparallel model includes the parallel model as a special case.

When the number of predictors is large relative to the number of observations, it is not possible to estimate each coefficient with a high degree of accuracy by maximum likelihood. A more realistic modeling goal is to build the best possible model for out-of-sample prediction and determine which predictors are most important predictors. A regularization method such as lasso or elastic net can help. In this context, one might forgive some incorrectness of the parallelism assumption if it is reasonable 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. Tutz and Gertheiss (2016) note that although the parallelism assumption "is very popular and reduces model complexity, it is quite restrictive and sometimes not reasonable".

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 $\eta_i = c + B^\top x_i$, where c 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 parameterized as $\eta_i = c + (b^\top x_i) \cdot \mathbb{1}$. The semi-parallel model specifies $\eta_i = c + B^\top x_i + (b^\top x_i) \cdot \mathbb{1}$. It is the nonparallel model but overparameterized 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 B 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.

We point out that overparameterization of penalized regression models is used in other settings. For example, Ollier and Viallon (2017) use overparameterization to avoid an arbitrary choice of reference stratum when modeling interactions between a categorical covariate and other predictors.

2.7. Elastic net penalty

This section discusses the elastic net penalty for the parallel, nonparallel, and semi-parallel ELMO model forms. There are many useful resources on regularization, penalized regression, and variable selection, which provide further details in various settings (Bickel and Li 2006; Hesterberg, Choi, Meier, and Fraley 2008; Hastie *et al.* 2009; Friedman *et al.* 2010; Schifano, Strawderman, and Wells 2010; Vidaurre, Bielza, and Larrañaga 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 α, must be selected or tuned on the data set. The degree of penalization is controlled by another tuning parameter, typically denoted by λ. Typical practice is to fit the penalized model for a sequence of λ values and use a tuning procedure to select the best value (Hesterberg *et al.* 2008; Hastie *et al.* 2009; Arlot and Celisse 2010; Sun, Wang, and Fang 2013). One tuning procedure is to select the model with best fit as determined by C_p, AIC, BIC, or another fitness measure. Another approach, which we generally prefer, is using cross-validation to select the value that gives the best out-of-sample prediction.

Let b_j be the j-th element of b and B_{jk} be the (j,k)-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. Below, we write the elastic net objective function for each of the three model forms.

Parallel model

The objective function is

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

Nonparallel model

The objective function is

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

Semi-parallel model

The objective function is

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

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 ρ at a very large value will set all parallel coefficients to zero, which is equivalent to the nonparallel model. Fixing λ at a very large value and choosing ρ such that $\lambda\rho = \lambda^*$ is equivalent to the parallel model with regularization parameter λ^*. 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 λ increases from zero.

We follow the common convention of scaling the negative log-likelihood by the number of observations (Friedman *et al.* 2010). 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.

3. Coordinate descent optimization algorithm

We propose optimizing ELMO models with the elastic net penalty using a coordinate descent algorithm. Our algorithm mirrors that of Friedman, Hastie, Höfling, and Tibshirani (2007); Friedman *et al.* (2010) for generalized linear models. The algorithm is iterative with 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, Carter, and Hubert (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 by an invertible multivariate link function. To apply the coordinate descent algorithm to any such model, all that is required is to derive the Jacobian of the inverse link function.

The rest of this section is organized as follows. In order to formulate the IRLS quadratic approximation, it is more convenient to parameterize ELMO models with a single coefficient vector instead of c, b, and B. Section 3.1 discusses this alternative parameterization. Section 3.2 discusses the elastic net penalty under the alternative parameterization. Section 3.3 discusses the outer loop of the optimization procedure, which updates the quadratic approximation to the log-likelihood. Section 3.4 discusses the coordinate descent inner loop. Section 3.5 discusses computational efficiency and numerical stability for the coordinate descent updates. Sections 3.6, Section 3.7, and Section 3.8 discuss different aspects of the algorithm specifications. Specifically, Section 3.6 presents a method for choosing a sequence of regularization parameter values for the solution path. Section 3.7 presents a method for choosing starting coefficient values for the optimization algorithm. And Section 3.8 suggests a stopping rule for terminating the algorithm. Section 3.9 summarizes the algorithm in outline form. Finally, Section 3.10 discusses specific optimization issues that can arise with the cumulative probability model family.

3.1. ELMO parameterization with a single coefficient vector

Until now, we have written ELMO coefficients in a compact form using an intercept vector c, 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 β. 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, η_i, can then be written as $\eta_i = X_i^\top \beta$ for all three model forms – parallel, nonparallel, and semi-parallel.

Let B_{*k} denote the k-th column of B. Below, the structure of X_i and β is given for each of the three model forms, with the dimensions written in subscript.

Parallel model

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

Nonparallel model

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

Semi-parallel model

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

3.2. Elastic net penalty

Let Q denote the length of β, and let β_j denote the j-th element. We write the elastic net objective function as

$$\mathcal{M}(\beta; \alpha, \lambda, \omega_1, \ldots, \omega_Q) = -\frac{1}{N_+}\ell(\beta) + \lambda \sum_{j=1}^{Q} \omega_j \left(\alpha|\beta_j| + \tfrac{1}{2}(1-\alpha)\beta_j^2 \right),$$

where $\lambda > 0$ and $0 \leq \alpha \leq 1$. For all three model forms, $\omega_1 = \omega_2 = \cdots = \omega_K = 0$, so the intercept terms in c are not penalized. For $j > K$, $\omega_j = 1$, with the only exception being that $\omega_{K+1} = \omega_{K+2} = \cdots = \omega_{K+P} = \rho$ for the semi-parallel model (i.e., the ω_j corresponding to the coefficients in b are equal to ρ for the semi-parallel model).

These are not firm rules regarding ω_j, as there may be situations where one wishes to modify the ω_j to accommodate more elaborate penalization schemes. For example, one might wish leave to some covariates unpenalized or to penalize them with varying degrees. Such modifications can easily be made and do not require any adjustment to the algorithm.

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; Friedman *et al.* 2010). However, this is not a requirement.

3.3. 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. This requires the score function and Fisher information.

Log-likelihood

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 $\log \binom{n_i}{y_{i1}, y_{i2}, \ldots, y_{i(K+1)}}$ because it does not depend on the unknown coefficients, and hence does not affect the model fit. Now let h denote the inverse link function, i.e., $h = g^{-1}$. The log-likelihood as a function of β is

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

Score function

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

$$U_i(\beta) = \nabla \ell_i(\beta) = X_i^\top Dh(\eta_i)^\top \nabla L_i(p_i) = X_i^\top W_i(z_i - X_i\beta) ,$$

where

$$z_i = W_i^{-1}(Dh(\eta_i)^\top \nabla L_i(p_i)) + X_i\beta$$

and

$$\nabla L_i(p_i) = \left(\frac{y_{i1}}{p_{i1}}, \ldots, \frac{y_{iK}}{p_{iK}}\right)^\top - \left(\frac{y_{i(K+1)}}{p_{i(K+1)}}\right) \cdot \mathbb{1} .$$

See Appendix C for details about the inverse link Jacobian, $Dh(\cdot)$, specifically for ELMO models.

Fisher information

The Fisher information matrix is

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

where

$$W_i = Dh(\eta_i)^\top \Sigma_i^{-1} Dh(\eta_i) ,$$

and

$$\Sigma_i^{-1} = n_i\left([\mathrm{diag}(p_i)]^{-1} + \frac{1}{p_{i(K+1)}} \cdot \mathbb{1}\mathbb{1}^\top\right)$$

since

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

Quadratic approximation

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

$$\ell(\beta) = \sum_{i=1}^{N} \ell_i(\beta) \ ,$$
$$U(\beta) = \sum_{i=1}^{N} U_i(\beta) = X^\top W(z - X\beta) \ , \text{ and}$$
$$\mathcal{I}(\beta) = \sum_{i=1}^{N} \mathcal{I}_i(\beta) = X^\top W X \ ,$$

$$\text{where } X = \begin{pmatrix} X_1 \\ \vdots \\ X_N \end{pmatrix} \ , \quad z = \begin{pmatrix} z_1 \\ \vdots \\ z_N \end{pmatrix} \ , \quad \text{and} \quad W = \begin{pmatrix} W_1 & 0 & \cdots & 0 \\ 0 & W_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & W_N \end{pmatrix} \ .$$

Let $\hat{\beta}^{(r)}$ denote the coefficient estimates after the r^{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 the weighted sum of squares expression

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

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

3.4. Optimization inner loop (coordinate descent)

For unpenalized maximum likelihood estimation, the quadratic approximation is maximized by the usual weighted least squares solution $\hat{\beta}^{(r+1)} = (X^\top W^{(r)} X)^{-1} X^\top W^{(r)} z^{(r)}$. 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 β_j at the s-th inner loop iteration of the r-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-th regression coefficient only, with all other coefficients fixed at their current estimates. The s-th iteration coordinate descent update of the j-th coefficient is $\arg\min \mathcal{M}_j^{(r,s)}(t)$. If $\omega_j = 0$ (i.e., β_j is unpenalized), then this can be solved straightforwardly by setting $\frac{d}{dt}\mathcal{M}_j^{(r,s)}(t) = 0$.

In general, for $t \neq 0$,

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

where X_{*j} denotes the j-th column of X, X_{*-j} denotes X with the j-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 $2\lambda\alpha\omega_j$. If $|\frac{1}{N_+}X_{*j}^T W^{(r)}(z^{(r)} - X_{*-j}\hat{\beta}_{-j})| > \lambda\alpha\omega_j$, 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

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

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 non-negative (e.g., to implement the non-negative garrote penalty discussed in Breiman (1995)). With this constraint, the coordinate descent update becomes $\hat{\beta}_j^{(r,s+1)} = \arg\min_{t\geq 0} \mathcal{M}_j^{(r,s)}(t)$, where the only difference is the restriction that $t \geq 0$. When $\arg\min \mathcal{M}_j^{(r,s)}(t) < 0$, then $\arg\min_{t\geq 0} \mathcal{M}_j^{(r,s)}(t) = 0$ because $\mathcal{M}_j^{(r,s)}(t)$ is convex. The coordinate descent update in this case is

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

3.5. Computational efficiency and numerical stability

The inner loop update can alternatively be written in terms of the score and information as

$$\hat{\beta}_j^{(r,s+1)} = \frac{S\left(\frac{1}{N_+}\left(\left[U(\hat{\beta}^{(r)})\right]_j + \left[\mathcal{I}(\hat{\beta}^{(r)})\hat{\beta}^{(r)}\right]_j - \left[\mathcal{I}(\hat{\beta}^{(r)})\hat{\beta}^{(r,s)}\right]_j\right),\ \lambda\alpha\omega_j\right)}{\frac{1}{N_+}\left[\mathcal{I}(\hat{\beta}^{(r)})\right]_{jj} + \lambda(1-\alpha)\omega_j} \ ,$$

where the subscripts on terms with square brackets indicate the j-th vector element or matrix diagonal. Written in this form, we see that only the numerator term $[\mathcal{I}(\hat{\beta}^{(r)})\hat{\beta}^{(r,s)}]_j$ needs to be updated with each inner loop iteration. Furthermore, only one element of $\hat{\beta}^{(r,s)}$ changes with each inner loop iteration, so the numerator term can be updated without performing a full matrix multiplication.

For computational efficiency, it is important to recognize the sparse block structure of W when performing matrix multiplication. Also, the X_i matrices of the nonparallel and semi-parallel models have a sparse block structure, so it may be advantageous to consider this as well.

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 score and likelihood can be computed with uncapped probabilities. (In the **ordinalNet** R package, the `pMin` argument sets the minimum probability threshold for the information matrix calculation.)

3.6. Regularization parameter sequence

Often we are interested in computing solutions for a sequence of λ 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 λ values is to let $\lambda_{\min} = 0.01 \times \lambda_{\max}$ and consider a sequence of λ 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 λ 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,

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

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

3.7. Starting values

Park and Hastie (2007) proposed an efficient estimation procedure for a decreasing sequence of λ values. The $\hat{\beta}$ solution for each λ value is used as the starting value for the next λ 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 λ 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 $\lambda_{\max}$ if there are no unpenalized non-intercept coefficients. If the first λ value is not $\lambda_{\max}$ or some of the non-intercepts are unpenalized, then this is still usually a reasonable set of starting values for the first λ value.

3.8. 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, ϵ_{out} and ϵ_{in}, to the same value.

3.9. Algorithm summary

Let $\hat{\beta}[\lambda]$ denote the final coefficient vector estimate at a particular λ value in the solution path. The full algorithm is as follows.

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

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

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 $\hat{\beta}[\lambda_m] \leftarrow \hat{\beta}^{(r)}$.

3.10. 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 2.5 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 that allows the cumulative probabilities to be non-monotone. In this case, the class probabilities could be greater than one or less than zero for some observations. The coefficient values that optimize the objective function may even result in negative class probabilities provided no observation has negative probability in a class that was observed. This is essentially the likelihood that our 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 λ value where this occurred. Further work is required to devise a constrained optimization procedure for the cumulative probability family.

4. Simulation

We have discussed three penalized ELMO model forms for ordinal data – parallel, nonparallel, and semi-parallel. 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. For each replicate, we simulated a training data set of 50 observations, along with a very large validation set of 10,000 observations for evaluating out-of-sample prediction accuracy. The validation data set was chosen to be large so that the out-of-sample prediction accuracy could be estimated precisely for each randomly generated training data set. Both the training and test data sets were simulated from a forward stopping ratio model with the parameters shown in Table 2. In each 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 ρ was set to one. A λ 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 data set, five-fold cross-validation was used to select the optimal tuning parameter. The λ value with the highest average out-of-sample log-likelihood across the five folds was selected.

Boxplots of the simulation results are shown in Figure 1. Prediction accuracy was measured using the percentage of deviance explained and the misclassification rate on the validation data. Now, we compare the medians of these metrics for the three models in each simulation. As one might expect, the nonparallel model performs the best in Simulation 1 because the parallelism assumption is violated for every predictor with nonzero coefficients. The parallel model performs the best in Simulation 2 because the parallelism assumption holds for every

Parameter	Sim 1	Sim 2	Sim 3
c	$\begin{pmatrix} -0.5 \\ 0 \end{pmatrix}$	$\begin{pmatrix} -0.5 \\ 0 \end{pmatrix}$	$\begin{pmatrix} -0.5 \\ 0 \end{pmatrix}$
B	$\begin{pmatrix} 2 & -2 \\ \vdots & \vdots \\ 2 & -2 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{pmatrix} \begin{matrix} \\ \times 5 \\ \\ \\ \times 10 \\ \end{matrix}$	$\begin{pmatrix} 2 & 2 \\ \vdots & \vdots \\ 2 & 2 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{pmatrix} \begin{matrix} \\ \times 5 \\ \\ \\ \times 10 \\ \end{matrix}$	$\begin{pmatrix} -2 & 2 \\ 2 & 2 \\ \vdots & \vdots \\ 2 & 2 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{pmatrix} \begin{matrix} \\ \\ \times 4 \\ \\ \\ \times 10 \\ \end{matrix}$

Table 2: Data simulation parameters for the three simulation experiments. For each experiment, the data generating mechanism was a forward stopping ratio model. The sample size was $N = 50$ for training data and $N = 10{,}000$ for validation.

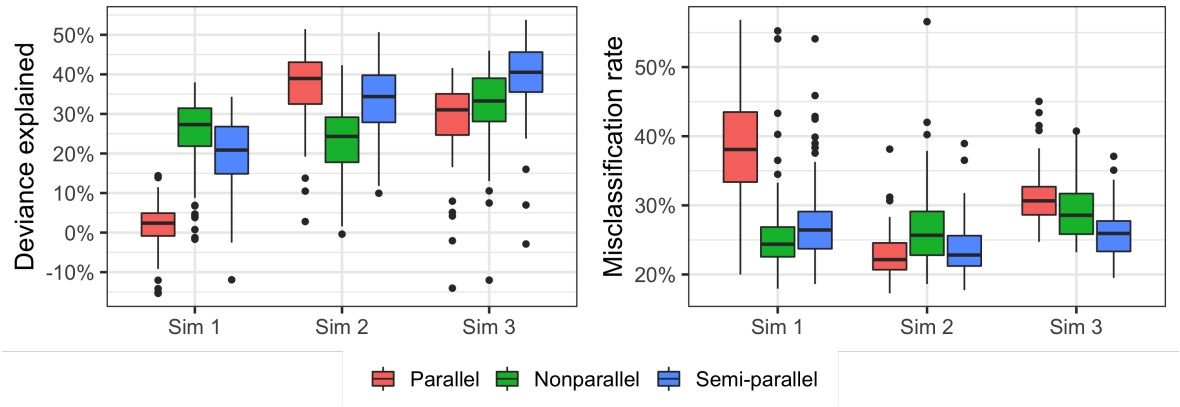

Figure 1: Out-of-sample percentage of deviance explained and misclassification rate across 100 replicates for the simulation experiment. For each replicate, out-of-sample prediction accuracy was evaluated on 10,000 observations generated from the same distribution as the training set. Deviance explained was calculated as $1 - \text{loglik}_1/\text{loglik}_0$ where loglik_1 is the log-likelihood of the out-of-sample data calculated from the fitted model and loglik_0 is the within-sample log-likelihood of a null model (no covariates) fit to the out-of-sample data. This quantity is usually positive but can be negative.

predictor. The semi-parallel model performs the best in Simulation 3 because the parallelism assumption holds for most of the predictors, but it is violated for the first predictor.

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 ρ for the semi-parallel fit.

5. Method comparison

We now demonstrate ELMO class models alongside other established methods for out-of-sample prediction and variable selection. We use a data set from a cancer genomics example presented by Archer *et al.* (2014). The data are a subset of the Gene Expression Omnibus `GSE18081`. The R package **ordinalgmifs** contains a filtered version of this data set called `hccframe`. It contains methylation levels of 46 CpG sites from 56 human subjects. The measurements come from liver tissue samples assayed using the Illumina GoldenGate Methylation BeadArray Cancer Panel I (Archer, Mas, Maluf, and Fisher 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 CpG site methylation levels 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.

To compare the out-of-sample prediction performance of various methods, we used cross-validation by fitting the models 100 times, each time holding out ten randomly selected subjects from the training sample. The out-of-sample prediction performance was evaluated using the Brier score (Brier 1950) and misclassification rate. We chose the Brier score as the performance metric because likelihood-based metrics (e.g., percentage of deviance explained) are highly sensitive to rare cases where the predicted probability of an observed class is very small – this problem is known as *hypersensitivity to the logarithmic scoring rule* (Selten 1998).

A total of six models were compared, based on three methods summarized below. An abbreviation for each model is written in parentheses for reference.

1. Adjacent category models with elastic net penalty.
    - Three models: parallel (ACAT-P), nonparallel (ACAT-N), and semi-parallel (ACAT-S).
    - Logit link with forward parameterization.
    - $\alpha = 0.5$
    - λ was tuned by ten-fold cross-validation within each training sample, and the value with the best average out-of-sample log-likelihood was selected.
    - The same λ sequence was used for each of the ten cross-validation folds. $\lambda_{\max}$ was calculated from the full data, and a sequence of 100 values was generated uniformly on the logarithmic scale from $\lambda_{\max}$ to $\lambda_{\max} \times 10^{-4}$.
    - Fit by **ordinalNet**.
2. Multinomial logistic regression models with elastic net penalty.
    - Two models: standard lasso (ML) and group lasso (ML-G).
    - $\alpha = 0.5$
    - λ was tuned by ten-fold cross-validation within each training sample, and the value with the best average out-of-sample log-likelihood was selected.

- The same λ sequence was used for each of the ten cross validation folds. $\lambda_{\max}$ was calculated from the full data, and a sequence of 100 values was generated uniformly on the logarithmic scale from $\lambda_{\max}$ to $\lambda_{\max} \times 10^{-4}$.
- Fit by **glmnet**.

3. Adjacent category model with GMIFS solution path (GMIFS).

- Logit link with forward parameterization.
- The GMIFS solution path was fit with step size 0.01.
- The number of steps was tuned by ten-fold cross-validation within each training sample, and the step number with the best out-of-sample-log-likelihood was selected.
- The maximum number of steps was calculated from the full training data. If the GMIFS algorithm terminated short of the maximum step limit for any of the ten cross-validation folds, then model where the algorithm terminated was used for the remaining steps.
- Fit by **ordinalgmifs**.

All of these models are forms of the forward adjacent category model with logit link (see Appendix A for details). ACAT-N, ACAT-S, ML, and ML-G are all the same model but with different forms of the elastic net penalty. ACAT-P and GMIFS also have the same model space except that it is restricted by the parallelism assumption.

For both **ordinalNet** and **glmnet**, we set the argument `standardize = TRUE` to standardize the training data and return the fitted model with coefficients on the scale of the original data each time the model was fit. For **ordinalgmifs**, setting the argument `scale = TRUE` returns a model on the scale of the standardized data. To be consistent with **ordinalNet** and **glmnet**, we set `scale = FALSE` and standardized the training data before passing it to **ordinalgmifs**, then scaled the coefficients and intercepts back to the scale of the original data to make out-of-sample predictions.

Results for the experiment are summarized in Table 3, Figure 2, and Figure 3. All models perform reasonably well, but ACAT-P had the best average Brier score and misclassification rate by a slight margin. ACAT-S had the second best average Brier score while GMIFS had the second best average misclassification rate. Undoubtedly the best model and regularization technique combination will vary by application, but this comparison illustrates that ELMO models with parallel, semi-parallel, and nonparallel penalties are a viable option.

In addition to class prediction, penalized regression can be used for variable selection, i.e., to determine which of the 46 CpG site methylation levels are most associated with liver disease. Table 4 shows which CpG sites 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. GMIFS is the most sparse model, although ACAT-S selects nearly the same set of CpG sites.

	Brier score	Misclassification rate
ACAT-P	0.123 (0.0091)	0.082 (0.0085)
ACAT-N	0.128 (0.0087)	0.093 (0.0078)
ACAT-S	0.123 (0.0088)	0.093 (0.0088)
ML	0.155 (0.0086)	0.116 (0.0081)
ML-G	0.153 (0.0084)	0.114 (0.0080)
GMIFS	0.132 (0.0097)	0.088 (0.0079)

Table 3: Out-of-sample Brier score and misclassification rate for the method comparison on the liver disease dataset (`GSE18081`). Each model was fit 100 times, and each time ten randomly selected subjects were held out from the training data. The Brier score and misclassification rate were calculated for the ten held out subjects. The value reported is the mean (standard error) across 100 replicates.

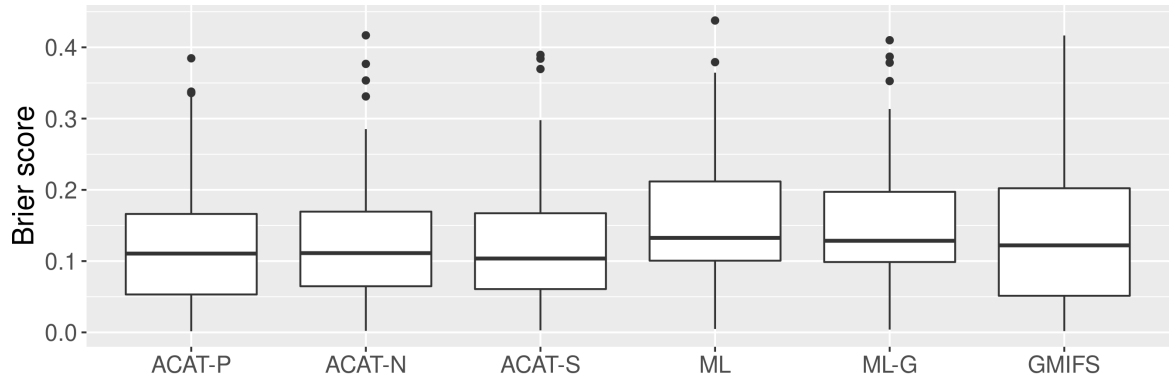

Figure 2: Out-of-sample Brier score for the method comparison on the liver disease dataset (`GSE18081`). The box plots summarize the results across 100 replicates. Note that a lower score indicates better prediction accuracy.

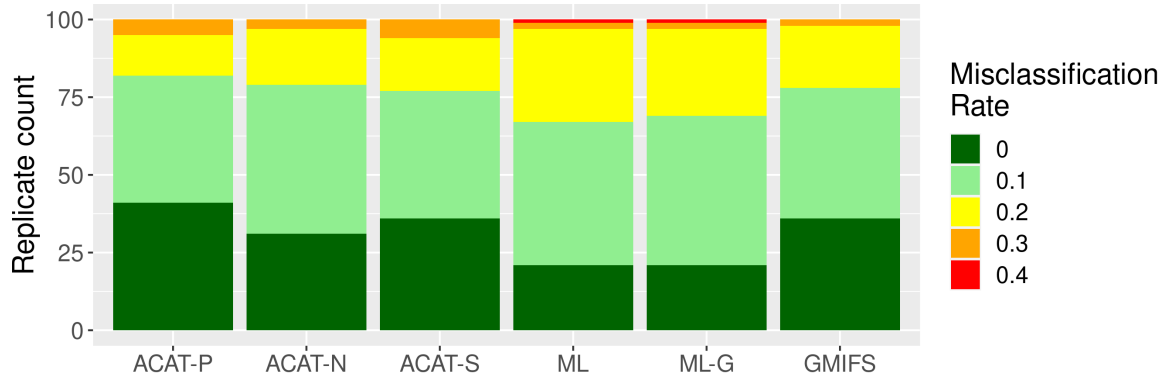

Figure 3: Out-of-sample misclassification rate for the method comparison on the liver disease dataset (`GSE18081`). Each replicate has a misclassification rate of 0, 0.1, 0.2, 0.3, or 0.4 corresponding to 0, 1, 2, 3, or 4 misclassifications among ten out-of-sample observations.

	ACAT-P	ACAT-N	ACAT-S	ML	ML-G	GMIFS
CDKN2B_seq_50_S294_F	14.0	1	2	2	3	14.4
DDIT3_P1313_R	7.6	2	1	2	3	8.1
ERN1_P809_R	− 2.4	1	1	2	3	−1.9
GML_E144_F	−6.6	2	2	2	3	−7.8
HDAC9_P137_R	−0.2	·	1	1	3	−0.3
HLA.DPA1_P205_R	−0.6	1	1	1	3	−1.3
HOXB2_P488_R	1.5	1	2	2	3	0.4
IL16_P226_F	−10.0	1	2	2	3	−13.7
IL16_P93_R	−3.2	1	1	1	3	−2.1
IL8_P83_F	−1.3	1	2	2	3	−1.1
MPO_E302_R	−10.0	1	2	1	3	−10.7
MPO_P883_R	−1.2	1	1	1	3	−1.0
PADI4_P1158_R	4.6	2	2	2	3	4.5
SOX17_P287_R	6.4	2	2	2	3	9.1
TJP2_P518_F	15.3	1	2	1	3	22.5
WRN_E57_F	−6.8	1	1	1	3	−4.8
CRIP1_P874_R	·	2	1	2	3	·
SLC22A3_P634_F	2.0	1	1	2	3	·
CCNA1_P216_F	0.6	1	·	2	3	·
SEPT9_P374_F	·	1	·	·	·	·
ITGA2_E120_F	·	·	·	1	3	·
ITGA6_P718_R	·	1	·	·	·	·
HGF_P1293_R	−2.0	·	·	·	·	·
DLG3_E340_F	−0.2	1	·	1	3	·
APP_E8_F	−8.8	1	·	·	3	·
SFTPB_P689_R	−7.0	1	2	1	3	−4.2
PENK_P447_R	·	1	·	1	3	·
COMT_E401_F	−1.7	1	2	2	3	−2.5
NOTCH1_E452_R	·	·	·	·	·	·
EPHA8_P456_R	−6.3	1	1	1	3	·
WT1_P853_F	·	·	·	·	·	·
KLK10_P268_R	·	1	·	2	3	·
PCDH1_P264_F	−0.3	2	1	1	3	·
TDGF1_P428_R	·	1	·	·	3	·
EFNB3_P442_R	·	1	·	·	·	·
MMP19_P306_F	·	1	·	·	·	·
FGFR2_P460_R	·	·	·	1	3	·
RAF1_P330_F	·	1	·	·	·	·
BMPR2_E435_F	·	1	·	1	3	·
GRB10_P496_R	·	1	·	·	·	·
CTSH_P238_F	3.2	1	·	·	3	·
SLC6A8_seq_28_S227_F	·	·	·	·	·	·
PLXDC1_P236_F	·	·	·	·	·	·
TFE3_P421_F	·	·	·	·	·	·
TSG101_P139_R	·	·	·	·	·	·

Table 4: Regression coefficient estimates for the method comparison on the liver disease dataset (`GSE18081`). For ACAT-P and GMIFS, the coefficients are reported on the scale of the unstandardized data, and positive coefficients shift probability toward higher response categories. For the methods with multiple coefficients per predictor, the number of nonzero coefficient estimates is reported instead of coefficient values. ACAT-N, ACAT-S, and ML can have up to two nonzero coefficients per predictor. (The latter two models have three coefficients per predictor, but at least one is always set to zero.) ML-G has three coefficients per predictor, and by design, either selects all three coefficients or sets them all to zero.

6. The ordinalNet R package

The **ordinalNet** package (version 2.10) (Wurm *et al.* 2021) 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 the out-of-sample log-likelihood, misclassification rate, Brier score, or percentage of deviance explained for a sequence of λ values. It returns an '`ordinalNetTune`' S3 object. This object has `print`, `summary`, and `print` methods.
- `ordinalNetCV` uses K-fold cross-validation to obtain the out-of-sample log-likelihood, misclassification rate, Brier score, or percentage of deviance explained. 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-8, stopThresh = 1e-8, threshOut = 1e-8, threshIn = 1e-8,
  maxiterOut = 100, maxiterIn = 100, 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` Optional non-negative vector of penalty factors with length equal to the number of columns in `x`. If this argument is used, then the penalty for each variable is scaled by its corresponding factor. 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 $\partial h(\eta)/\partial \eta^\top$ (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` Non-negative numeric value equal to one by default. The penalty on all parallel terms is scaled by this factor (as well as variable-specific `penaltyFactors`). 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` `max(alpha, alphaMin)` is used to calculate the starting lambda value when `lambdaVals = NULL`. In this case, the default lambda sequence begins with the smallest lambda value such that all penalized coefficients are set to zero (i.e., the value where the first penalized coefficient enters the solution path). The purpose of this argument is to help select a starting value for the lambda sequence when `alpha = 0`, because otherwise it would be infinite. Note that `alphaMin` is only used to determine the default λ sequence and that the model is always fit using `alpha` to calculate the penalty.

`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 λ 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.

7. Demonstration in R

This section contains five examples that demonstrate different aspects of the **ordinalNet** package (Wurm *et al.* 2021) using the same subset of the Gene Expression Omnibus `GSE18081` data set used in Section 5.

	parallelTerms	nonparallelTerms
Parallel	TRUE	FALSE
Nonparallel	FALSE	TRUE
Semi-parallel	TRUE	TRUE

Table 5: Argument settings for the parallel, nonparallel, and semi-parallel model forms.

Examples 1–3 (Sections 7.1–7.3) 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 5.

Example 4 (Section 7.4) demonstrates how one might use the `ordinalNetTune` function to select an appropriate value for the tuning parameter λ. This function uses cross-validation to evaluate the out-of-sample prediction performance of a sequence of λ values.

Example 5 (Section 7.5) demonstrates the `ordinalNetCV` function, which tunes the model within each cross-validation fold and evaluates the out-of-sample prediction performance. This provides an 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).

7.1. 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 λ values, with $\lambda_{\max}$ equal to the smallest value that sets every coefficient to zero and $\lambda_{\min} = \lambda_{\max} \cdot 0.01$. The sequence runs from $\lambda_{\min}$ to $\lambda_{\max}$ uniformly on the logarithmic scale.

```
R> library("ordinalNet")
R> library("ordinalgmifs")
R> data("hccframe")
R> y <- hccframe$group
R> x <- as.matrix(subset(hccframe, select = -group))
R> fit1 <- ordinalNet(x, y, family = "cumulative", link = "logit",
+    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 can return the coefficient estimates of any model fit in the λ 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
```

The coefficients have relatively large magnitude. One reason is that the coefficients are on the scale of the unstandardized covariates, all of which have fairly small standard deviations. Another reason is that the best AIC fit explains over 96% of total deviance, so naturally some of the effects are quite large.

7.2. 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 ρ, which is `parallelPenaltyFactor = 1`. The coefficient matrix of the best AIC fit has identical columns for most, but not all, of the predictors.

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

```
  lambdaVals nNonzero    loglik    devPct       aic      bic
1  0.4287829        2 -61.22898 0.0000000 126.45797 130.5087
2  0.3364916        7 -49.66682 0.1888349 113.33363 127.5111
3  0.2640652        9 -40.70441 0.3352102  99.40881 117.6370
4  0.2072278       11 -33.66859 0.4501201  89.33718 111.6160
5  0.1626241       14 -27.93291 0.5437959  83.86583 112.2208
6  0.1276209       16 -22.97691 0.6247381  77.95381 110.3594
```

```
R> head(coef(fit2, matrix = TRUE))
```

```
                     logit(P[Y<=1]) logit(P[Y<=2])
(Intercept)              -23.518682     -22.199967
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
```

7.3. 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 2.5 and 3.10. The solution path is terminated after the third λ value where the optimum leaves the constrained parameter space. This can be seen from the `summary` method output.

The semi-parallel model is also susceptible to this issue, but it is less prone. The stopping ratio, continuation ratio, and adjacent category models avoid this issue altogether because their parameter space is not restricted. These may be better options than the nonparallel cumulative probability model in cases like this.

```
R> fit3 <- ordinalNet(x, y, family = "cumulative", link = "logit",
+    parallelTerms = FALSE, nonparallelTerms = TRUE, warn = FALSE)
R> head(summary(fit3), 8)
```

```
  lambdaVals nNonzero    loglik    devPct      aic      bic
1 0.40460545        2 -61.22898 0.0000000 126.4580 130.5087
2 0.31751816        4 -52.35095 0.1449972 112.7019 120.8033
3 0.24917554        7 -45.58072 0.2555696 105.1614 119.3389
4 0.19554299       20 -44.73266 0.2694202 129.4653 169.9724
5 0.15345431       27 -44.73265 0.2694204 143.4653 198.1498
6 0.12042480       27 -44.73265 0.2694204 143.4653 198.1498
7 0.09450456       27 -44.73265 0.2694204 143.4653 198.1498
8 0.07416340       27 -44.73265 0.2694204 143.4653 198.1498
```

```
R> head(coef(fit3, matrix = TRUE))
```

```
                     logit(P[Y<=1]) logit(P[Y<=2])
(Intercept)               1.192961      -1.008505
CDKN2B_seq_50_S294_F      0.000000       0.000000
DDIT3_P1313_R             0.000000       0.000000
ERN1_P809_R               0.000000       0.000000
GML_E144_F                0.000000       0.000000
HDAC9_P137_R              0.000000       0.000000
```

7.4. Example 4

The `ordinalNetTune` function uses K-fold cross-validation to obtain out-of-sample performance for a sequence on λ 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 λ 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, misclassification rate, Brier score, and percentage of deviance explained for each λ 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 λ value with the best average out-of-sample log-likelihood, as demonstrated below. We obtain the coefficient estimates at this λ value, which happens to be the thirteenth value in the sequence.

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

         lambda loglik_avg misclass_avg  brier_avg  devPct_avg
1  4.287829e-01 -12.116291   0.55151515 0.65478933 0.001124981
2  2.640652e-01  -9.072777   0.28484848 0.47093864 0.250327320
3  1.626241e-01  -7.060731   0.21363636 0.36088470 0.416917751
4  1.001517e-01  -5.300077   0.10606061 0.26720041 0.563612864
5  6.167827e-02  -3.914669   0.08787879 0.18989857 0.678678299
6  3.798445e-02  -3.025536   0.08787879 0.14636709 0.752340784
7  2.339266e-02  -2.442410   0.06969697 0.12236662 0.800617784
8  1.440633e-02  -2.042826   0.06969697 0.10569608 0.833781030
9  8.872110e-03  -1.792745   0.05151515 0.09531195 0.854663408
10 5.463873e-03  -1.665212   0.06969697 0.09020582 0.865216734
11 3.364916e-03  -1.659950   0.06969697 0.09250623 0.865655597
12 2.072278e-03  -1.613669   0.06969697 0.09051878 0.869552720
13 1.276209e-03  -1.600092   0.06969697 0.09060128 0.870756540
14 7.859507e-04  -1.623919   0.08787879 0.09322183 0.868875491
15 4.840264e-04  -1.688259   0.06969697 0.09790222 0.863648319
16 2.980868e-04  -1.773510   0.06969697 0.10269534 0.856739885
17 1.835762e-04  -1.883527   0.06969697 0.10755198 0.847825825
18 1.130551e-04  -2.006329   0.06969697 0.11185762 0.837903212
19 6.962477e-05  -2.124826   0.06969697 0.11568726 0.828309912
20 4.287829e-05  -2.248561   0.06969697 0.11947760 0.818317707

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

                    logit(P[Y<=1]) logit(P[Y<=2])
(Intercept)              -74.49220      -60.13542
```

```
CDKN2B_seq_50_S294_F      -22.25175      -22.25175
DDIT3_P1313_R             -12.44390      -12.44390
ERN1_P809_R                 3.56758        3.56758
GML_E144_F                 12.10814       12.10814
HDAC9_P137_R                0.00000        0.00000
```

7.5. Example 5

Suppose we use `ordinalNetTune` to choose the λ value with, say, the best average out-of-sample misclassification rate. Then the average misclassification rate obtained by cross-validation for that λ value is a biased estimate for the tuned model's true out-of-sample misclassification rate (likely better than the true rate). In order to get an unbiased estimate, we need to evaluate the tuned model on data that has not been used to either train or tune the model. Rather than reserving data for that purpose, another option is to use the `ordinalNetCV` function, which uses cross-validation to evaluate the out-of-sample prediction performance of a model tuned by cross validation.

The `ordinalNetCV` function does this by performing a nested cross validation procedure. Essentially, `ordinalNetCV` calls `ordinalNetTune` on the training set of each training/test split and selects the λ value with the best average out-of-sample prediction according to the metric specified by the `tuneMethod` argument (misclassification rate, Brier score, etc.). The out-of-sample prediction is then assessed on the test set using the selected λ value.

We demonstrate this function again using the parallel cumulative logit model. We use the default value of `nFolds = 5` to obtain five estimates of the out-of-sample prediction. We also use the default settings of `nFoldsCV = 5` and `tuneMethod = "cvLoglik"` to tune λ by 5-fold cross-validation on the training set of each fold, each time selecting the λ value with the best average out-of-sample log-likelihood. As in Example 4 (Section 7.4), we set `lambdaMinRatio = 1e-4`.

The `summary` method returns the λ value selected by the tuning procedure for each cross-validation fold, along with the out-of-sample log-likelihood, misclassification rate, Brier score, and percentage of deviance explained.

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

           lambda     loglik   misclass      brier    devPct
fold1 0.008872110 -3.7368700 0.16666667 0.21204288 0.7110074
fold2 0.005463873 -0.6340910 0.00000000 0.02362936 0.9471179
fold3 0.008872110 -0.9913706 0.00000000 0.04707506 0.9155417
fold4 0.023392656 -2.6007982 0.09090909 0.13826233 0.7830979
fold5 0.037984451 -3.7018244 0.09090909 0.17728753 0.6912742
```

Taking the average across folds gives a single estimate for each metric.

```
R> colMeans(summary(fit1_cv))

    lambda      loglik    misclass       brier      devPct
0.01691704 -2.33299083  0.06969697  0.11965943  0.80960782
```

8. Discussion

This paper introduced the elementwise link multinomial-ordinal (ELMO) model class, a rich class of multinomial regression models that includes some of the most 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 either an ordered or unordered categorical response. We also introduced the semi-parallel model form, which can be used with 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 elastic net penalty. The algorithm is general and can also be applied to multinomial regression models outside the ELMO class.

We conducted 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 a subset of 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 the 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 C++ (Stroustrup 2013) 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.* 2018).

> 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, Brown, Buja, Zhang, and Zhao 2013), where the number of observations exceeds the number of predictors, and the high-dimensional setting (Javanmard and Montanari 2014; Lockhart, Taylor, Tibshirani, and Tibshirani 2014; Tibshirani, Taylor, Lockhart, and Tibshirani 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.

Acknowledgments

The authors thank Alex Tahk for the suggestion that led them to explore shrinking the nonparallel model to the parallel model. On this project, Wurm was supported by NIH grant T32HL083806, Rathouz was supported by NIH grant R01HL094786-05A1, and Hanlon was supported by NIH grant R01HG007377.

References

Agresti A (2003). *Categorical Data Analysis*, volume 482. John Wiley & Sons. doi:10.1002/0471249688.

Archer KJ (2020a). ***glmnetcr**: Fit a Penalized Constrained Continuation Ratio Model for Predicting an Ordinal Response*. R package version 1.0.6, URL https://CRAN.R-project.org/package=glmnetcr.

Archer KJ (2020b). ***glmpathcr**: Fit a Penalized Continuation Ratio Model for Predicting an Ordinal Response*. R package version 1.0.8, URL https://CRAN.R-project.org/package=glmpathcr.

Archer KJ, Hou J, Zhou Q, Ferber K, Layne JG, Gentry A (2019). ***ordinalgmifs**: Ordinal Regression for High-Dimensional Data*. R package version 1.0.6, 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.

Bickel PJ, Li B (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.

Brier GW (1950). "Verification of Forecasts Expressed in Terms of Probability." *Monthey Weather Review*, **78**(1), 1–3. doi:10.1175/1520-0493(1950)078<0001:vofeit>2.0.co;2.

Eddelbuettel D (2013). *Seamless R and C++ Integration with* ***Rcpp***. Springer-Verlag. 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.

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.

Friedman J, Hastie T, Tibshirani R, Simon N, Narasimhan B, Qian J (2021). ***glmnet****: Lasso and Elastic-Net Regularized Generalized Linear Models*. R package version 4.1-2, URL https://CRAN.R-project.org/package=glmnet.

Goeman JJ, Meijer RJ, Chaturvedi N (2018). ***penalized****: L1 (Lasso and Fused Lasso) and L2 (Ridge) Penalized Estimation in GLMs and in the Cox Model*. R package version 0.9-51, URL https://CRAN.R-project.org/package=penalized.

Harrell, Jr FE (2015). *Regression Modeling Strategies: With Applications to Linear Models, Logistic and Ordinal Regression, and Survival Analysis*. 2nd edition. Springer-Verlag. doi:10.1007/978-3-319-19425-7.

Harrell, Jr FE (2021). ***rms****: Regression Modeling Strategies*. R package version 6.2-0, URL https://CRAN.R-project.org/package=rms.

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

Hesterberg T, Choi NH, Meier L, Fraley C (2008). "Least Angle and ℓ_1 Penalized Regression: A Review." *Statistics Surveys*, **2**, 61–93. doi:10.1214/08-ss035.

Javanmard A, Montanari A (2014). "Confidence Intervals and Hypothesis Testing for High-Dimensional Regression." *Journal of Machine Learning Research*, **15**(1), 2869–2909. doi:10.1109/allerton.2013.6736695.

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." *The Annals of Statistics*, **42**(2), 413. doi:10.1214/13-aos1175.

McCullagh P (1980). "Regression Models for Ordinal Data." *Journal of the Royal Statistical Society B*, **42**(2), 109–142. doi:10.1111/j.2517-6161.1980.tb01109.x.

Ollier E, Viallon V (2017). "Regression Modelling on Stratified Data with the Lasso." *Biometrika*, **104**(1), 83–96. doi:10.1093/biomet/asw065.

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.

Park MY, Hastie T (2007). "L1-Regularization Path Algorithm for Generalized Linear Models." *Journal of the Royal Statistical Society B*, **69**(4), 659–677. doi:10.1111/j.1467-9868.2007.00607.x.

Peterson B, Harrell, Jr FE (1990). "Partial Proportional Odds Models for Ordinal Response Variables." *Journal of the Royal Statistical Society C*, **39**(2), 205–217. doi:10.2307/2347760.

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. URL http://support.sas.com/documentation/onlinedoc/stat/indexproc.html.

Schifano ED, Strawderman RL, Wells MT (2010). "Majorization-Minimization Algorithms for Nonsmoothly Penalized Objective Functions." *Electronic Journal of Statistics*, **4**, 1258–1299. doi:10.1214/10-ejs582.

Selten R (1998). "Axiomatic Characterization of the Quadratic Scoring Rule." *Experimental Economics*, **1**(1), 43–62. ISSN 1573-6938. doi:10.1007/bf01426214.

Stroustrup B (2013). *The C++ Programming Language*. 4th edition. Addison-Wesley.

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. doi:10.1109/icmlc.2006.258712.

Tibshirani R (1996). "Regression Shrinkage and Selection via the Lasso." *Journal of the Royal Statistical Society B*, pp. 267–288. doi:10.1111/j.2517-6161.1996.tb02080.x.

Tibshirani RJ (2013). "The Lasso Problem and Uniqueness." *Electronic Journal of Statistics*, **7**, 1456–1490. doi:10.1214/13-ejs815.

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.

Tutz G, Gertheiss J (2016). "Regularized Regression for Categorical Data." *Statistical Modelling*, **16**(3), 161–200. doi:10.1177/1471082x16642560.

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 MJ, Rathouz PJ, Hanlon BM (2021). ***ordinalNet**: Penalized Ordinal Regression*. R package version 2.10, 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-Verlag. doi:10.1007/978-1-4939-2818-7.

Yee TW (2021). ***VGAM**: Vector Generalized Linear and Additive Models*. R package version 1.1-5, URL https://CRAN.R-project.org/package=VGAM.

Yee TW, Wild CJ (1996). "Vector Generalized Additive Models." *Journal of Royal Statistical Society B*, **58**(3), 481–493. doi:10.1111/j.2517-6161.1996.tb02095.x.

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 B*, **67**(2), 301–320. doi:10.1111/j.1467-9868.2005.00503.x.

A. Equivalence of multinomial logit and nonparallel ACAT

We show that the nonparallel adjacent category model with logit link (either forward or backward) is an alternative but equivalent parameterization of multinomial logistic regression. To see the equivalence, let B_{*k} denote the k^{th} column of B, and let c_k denote the k^{th} intercept term under the backward adjacent category model with logit link. Note that for $k = 1, \ldots, K$,

$$B_{*k}^\top x + c_k = \text{logit}\left(P(Y = k \mid k \leq Y \leq k+1)\right) = \log\left(\frac{p_k}{p_{k+1}}\right) .$$

Therefore,

$$\begin{aligned}
\log\left(\frac{p_k}{p_{K+1}}\right) &= \log\left(\frac{p_k}{p_{k+1}} \cdot \frac{p_{k+1}}{p_{k+2}} \cdots \frac{p_K}{p_{K+1}}\right) \\
&= \log\left(\frac{p_k}{p_{k+1}}\right) + \log\left(\frac{p_{k+1}}{p_{k+2}}\right) + \cdots + \log\left(\frac{p_K}{p_{K+1}}\right) \\
&= (B_{*k}^\top x + c_k) + (B_{*(k+1)}^\top x + c_{k+1}) + \cdots + (B_{*K}^\top x + c_K) \\
&= \tilde{B}_k^\top x + \tilde{c}_k \ ,
\end{aligned}$$

where $\tilde{B}_k = B_{*k} + B_{*(k+1)} + \cdots + B_{*K}$ and $\tilde{c}_k = c_k + c_{k+1} + \cdots + c_K$. This is the usual parameterization for multinomial logistic regression, where class $K+1$ serves as a reference class and $\tilde{B}_1, \tilde{B}_2, \ldots, \tilde{B}_K$ are the coefficient vectors.

We point out that although these are equivalent parameterizations of the same model, the elastic net penalty function differs depending on which parameterization is used.

B. 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 parameterizations for any particular model. However, different parameterizations have different elastic net penalty terms; therefore, the penalized likelihood favors some parameterizations over others. We will demonstrate that among almost all parameterizations 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 ρ. 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, Presnell, and Turlach 2000; Rinaldo 2008; Tibshirani 2013).

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

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

To see this, for any ζ set $\delta_k = \beta_k - \zeta$ for all k. All of these parameterizations 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 ζ 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 $\delta_1 + \zeta = \beta_1$, $\delta_2 + \zeta = \beta_2$, $\ldots$, $\delta_K + \zeta = \beta_K$. 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

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

Solving this yields

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

Now consider,

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

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

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

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 ζ where $f(\zeta)$ equals or crosses ρ, 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 ρ is not an integer, then the solution is unique. If ρ is an integer, then the solution could be unique, or there may be a range of solutions on a closed interval between two consecutive β's, ranked by value.

Elastic net

Consider

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

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 ζ and never constant over a range of values. Hence, the solution is unique.

C. The inverse link function and its Jacobian for MO families

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}, $\tilde{h}_{\text{EL}}$, and h_{MO} to be the inverses of g, g_{EL}, $\tilde{g}_{\text{EL}}$, and g_{MO}, respectively. The inverse link function can be written as

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

where $h_{\text{EL}}(\eta) = \Big(\tilde{h}_{\text{EL}}(\eta_1),\ \tilde{h}_{\text{EL}}(\eta_2),\ \ldots,\ \tilde{h}_{\text{EL}}(\eta_K)\Big)$.
The Jacobian of the inverse link can be written as

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

where $Dh_{\text{EL}}(\eta) = \text{diag}\left\{\tilde{h}'_{\text{EL}}(\eta_1),\ \tilde{h}'_{\text{EL}}(\eta_2), \ldots,\ \tilde{h}'_{\text{EL}}(\eta_K)\right\}$.
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 $h_{\text{MO}}(\delta)$ and $Dh_{\text{MO}}(\delta)$.

Each MO family is defined by the $g_{\text{MO}}(p)$ component of its multivariate link function. For optimization, it is necessary to compute the inverse $h_{\text{MO}}(\delta)$ and its Jacobian $Dh_{\text{MO}}(\delta)$. 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 δ, we write it in terms of both p and δ when convenient. This can be done because there is a one-to-one correspondence between δ 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.

C.1. Forward cumulative probability family

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

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

Now, h_{MO} has a closed form with

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

Dh_{MO} also has a closed form with

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

C.2. Forward stopping ratio family

This family is defined by $\delta_j = P(Y = j \mid Y \geq j)$ (see Table 1). From this definition, $[g_{\text{MO}}(p)]_1 = p_1$ and, for $j = 2, \ldots, K$,

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

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

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

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

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

For $m = 2, \ldots K$,

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

C.3. Forward continuation ratio family

This family is defined by $\delta_j = P(Y > j \mid Y \geq j)$ (Table 1). Let $g_{\text{MO:FSR}}$, $h_{\text{MO:FSR}}$, and $Dh_{\text{MO:FSR}}$ 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

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

C.4. Forward adjacent category family

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

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

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

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

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

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

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

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

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

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

D. Quadratic approximation to the log-likelihood

We derive the quadratic approximation $\ell^{(r)}(\beta)$ defined in Section 3.3. We begin with the second order Taylor expansion of the log-likelihood at $\hat{\beta}^{(r)}$ with the Hessian replaced by its expectation, $-\mathcal{I}(\hat{\beta}^{(r)})$. We show that this equals the weighted sum of squares expression defined as $\ell^{(r)}(\beta)$, up to an additive constant that does not depend on β. Letting $\overset{\text{C}}{=}$ denote equality up to an additive constant, we have

$$\begin{aligned}
&\ell(\hat{\beta}^{(r)}) + (\beta - \hat{\beta}^{(r)})^\top U(\hat{\beta}^{(r)}) - \tfrac{1}{2}(\beta - \hat{\beta}^{(r)})^\top \mathcal{I}(\hat{\beta}^{(r)}) (\beta - \hat{\beta}^{(r)}) \\
&\overset{\text{C}}{=} \beta^\top U(\hat{\beta}^{(r)}) + \beta^\top \mathcal{I}(\hat{\beta}^{(r)})\hat{\beta}^{(r)} - \tfrac{1}{2}\beta^\top \mathcal{I}(\hat{\beta}^{(r)})\beta \\
&= \beta^\top X^\top W^{(r)}(z^{(r)} - X\hat{\beta}^{(r)}) + \beta^\top X^\top W^{(r)} X\hat{\beta}^{(r)} - \tfrac{1}{2}\beta^\top X^\top W^{(r)} X\beta \\
&= \beta^\top X^\top W^{(r)} z^{(r)} - \tfrac{1}{2}\beta^\top X^\top W^{(r)} X\beta \\
&\overset{\text{C}}{=} \beta^\top X^\top W^{(r)} z^{(r)} - \tfrac{1}{2}\beta^\top X^\top W^{(r)} X\beta - \tfrac{1}{2}\{z^{(r)}\}^\top W^{(r)} z^{(r)} \quad \text{(completing the square)} \\
&= -\tfrac{1}{2}(z^{(r)} - X\beta)^\top W^{(r)} (z^{(r)} - X\beta) \\
&= \ell^{(r)}(\beta).
\end{aligned}$$

Affiliation:

Michael J. Wurm
Department of Statistics
University of Wisconsin–Madison
1300 University Avenue, Madison, WI 53706, United States of America
E-mail: wurm@uwalumni.com

Paul J. Rathouz
Department of Population Health
Dell Medical School at the University of Texas at Austin
1601 Trinity Street, Building B, Austin, TX 78712, United States of America
E-mail: paul.rathouz@austin.utexas.edu

Bret M. Hanlon
Department of Biostatistics and Medical Informatics
University of Wisconsin–Madison
610 Walnut Street, Madison, WI 53726, United States of America
E-mail: bret.hanlon@wisc.edu

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
September 2021, Volume 99, Issue 6 *Submitted:* 2017-06-15
doi:10.18637/jss.v099.i06 *Accepted:* 2019-12-09