OPT2023: 15th Annual Workshop on Optimization for Machine Learning

Feature Selection in Generalized Linear models via the Lasso: To Scale or Not to Scale?

Anant Mathur ANANT.MATHUR@UNSW.EDU.AU
Sarat Moka S.MOKA@UNSW.EDU.AU
Zdravko Botev BOTEV@UNSW.EDU.AU
University of New South Wales, Kensington, Sydney, NSW 2052, Australia

Abstract

The Lasso regression is a popular regularization method for feature selection in statistics. Prior to computing the Lasso estimator in both linear and generalized linear models, it is common to conduct a preliminary rescaling of the feature matrix to ensure that all the features are standardized. Without this standardization, it is argued, the Lasso estimate will unfortunately depend on the units used to measure the features. We propose a new type of iterative rescaling of the features in the context of generalized linear models. Whilst existing Lasso algorithms perform a single scaling as a preprocessing step, the proposed rescaling is applied iteratively throughout the Lasso computation until convergence. We provide numerical examples, with both real and simulated data, illustrating that the proposed iterative rescaling can significantly improve the statistical performance of the Lasso estimator without incurring any significant additional computational cost.

1. Introduction and Background

We begin with providing the background notation and facts for the linear regression model and its extensions to generalized linear models. With this requisite notation and background material, we will then be able to explain the main contribution of our paper without unnecessary verbosity.

We denote the $n \times p$ regression matrix (or feature matrix) containing the p features $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_p$ as

$$\mathbf{X} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_p] = \begin{bmatrix} \boldsymbol{x}_1^\top \\ \vdots \\ \boldsymbol{x}_n^\top \end{bmatrix},$$

and the corresponding regression response vector as $\boldsymbol{Y} \in \mathbb{R}^n$ (with $\boldsymbol{y}$ being the realization of the random vector $\boldsymbol{Y}$). We make the standard assumptions (see, for example, [6]) that the regression function, or the mean of $\boldsymbol{Y}$ conditional on $\mathbf{X}$, satisfies $\boldsymbol{Y} = \mathbb{E}_{\mathbf{X}}[\boldsymbol{Y}] + \boldsymbol{\epsilon}$ for some noise vector $\boldsymbol{\epsilon}$ with conditional expectation $\mathbb{E}_{\mathbf{X}}[\boldsymbol{\epsilon}] = \mathbf{0}$ and conditional variance $\mathbb{V}\mathrm{ar}_{\mathbf{X}}(\boldsymbol{\epsilon}) = \sigma^2 \mathbf{I}_n$, where σ is an unknown parameter. Recall that in a linear model, we assume that

$$\mathbb{E}_{\mathbf{X}}[\boldsymbol{Y}] = \mathbf{1}\beta_0 + \mathbf{X}\boldsymbol{\beta}$$

is a linear function of some model coefficients $\boldsymbol{\beta} \in \mathbb{R}^p$ and $\beta_0 \in \mathbb{R}$, the last one being called the intercept and corresponding to the constant feature $\mathbf{1} \in \mathbb{R}^n$. Thus, the model for $\boldsymbol{Y}$ is

$$\boldsymbol{Y} = \mathbf{1}\beta_0 + \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}.$$

Define the projection (idempotent) matrix $\mathbf{C} := \mathbf{I}_n - \mathbf{1}\mathbf{1}^\top/n$ and let

$$\eta_i^2 := \frac{\|\mathbf{C}\boldsymbol{v}_i\|^2}{n} = \frac{\boldsymbol{v}_i^\top \mathbf{C}\boldsymbol{v}_i}{n}, \quad i = 1, \ldots, p$$

be the empirical variance of the components of the i-th feature vector. We define the Lasso estimate [7] of $\boldsymbol{\beta}$ as the solution to the penalized least squares:

$$(\hat{\beta}_{0,\lambda}, \hat{\boldsymbol{\beta}}_\lambda) = \underset{b_0, \boldsymbol{b}}{\operatorname{argmin}} \frac{\|\boldsymbol{y} - \mathbf{1}b_0 - \mathbf{X}\boldsymbol{b}\|^2}{2n} + \lambda \sum_{i=1}^{p} \eta_i \times |b_i|, \tag{1}$$

where $\lambda > 0$ is a suitably chosen regularization parameter and the intercept b_0 is not penalized. It will shortly become clear why the definition (1) of the lasso estimator appears prima facie to be different from the one used in textbooks [4] and statistical packages [2]. We remind the reader that the main advantage of the Lasso regularization is that many of the components of $\hat{\boldsymbol{\beta}}_\lambda$ are estimated as zeros, making it very easy to tell which features are important and which are not. This phenomenon is behind the well-known ability of the Lasso estimator to perform simultaneous shrinkage and model selection [4, 7].

Feature Standardization. As mentioned in the abstract, it is common practice to standardize the features $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_p$ so that the variance of each $\boldsymbol{v}_i$ is unity [4, 7]. In other words, the columns $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_p$ of $\mathbf{X}$ are all rescaled in such a way that $\|\mathbf{C}\boldsymbol{v}_i\|^2 = n$ for all i. This standardization ensures that the Lasso estimate $\hat{\boldsymbol{\beta}}_\lambda$ is not affected by the units in which the features are measured, and in general improves the performance of the estimator [3]. The standardization can be accomplished by working with the matrix $\mathbf{X}\boldsymbol{\Upsilon}$, rather than $\mathbf{X}$, where $\boldsymbol{\Upsilon}$ is the rescaling matrix $\boldsymbol{\Upsilon} := \operatorname{diag}(\eta_1^{-1}, \ldots, \eta_p^{-1})$.

The solution to (1) (without preliminary standardization of the features) is equivalent to the solution

$$(\hat{\beta}_{0,\lambda}, \boldsymbol{\Upsilon}^{-1}\hat{\boldsymbol{\beta}}_\lambda) = \underset{b_0, \boldsymbol{b}}{\operatorname{argmin}} \frac{\|\boldsymbol{y} - \mathbf{1}b_0 - [\mathbf{X}\boldsymbol{\Upsilon}]\boldsymbol{b}\|^2}{2n} + \lambda \sum_{i=1}^{p} |b_i|, \tag{2}$$

so that $\hat{\boldsymbol{\beta}}_\lambda$ is now in agreement with the definition of the Lasso estimator given in textbooks and statistical packages [2].

Extensions to GLMs. Suppose that the joint density of the response vector $\boldsymbol{Y}$ given $\beta_0, \boldsymbol{\beta}, \mathbf{X}$ is $g(\boldsymbol{y} \mid \beta_0, \boldsymbol{\beta}, \mathbf{X})$, where the dependence on $\beta_0, \boldsymbol{\beta}, \mathbf{X}$ is through the linear map $(\beta_0, \boldsymbol{\beta}) \mapsto \mathbf{1}\beta_0 + \mathbf{X}\boldsymbol{\beta}$. Here, the cross-entropy training loss [6] (negative average log-likelihood) is $-\frac{1}{n} \ln g(\boldsymbol{y} \mid \beta_0, \boldsymbol{\beta}, \mathbf{X})$ and the extension of the Lasso estimator (2) to the setting of generalized linear models is then given by :

$$(\hat{\beta}_{0,\lambda}, \boldsymbol{\Upsilon}^{-1}\hat{\boldsymbol{\beta}}_\lambda) = \underset{b_0, \boldsymbol{b}}{\operatorname{argmin}} \frac{-\ln g(\boldsymbol{y} \mid b_0, \boldsymbol{b}, \mathbf{X}\boldsymbol{\Upsilon})}{n} + \lambda \sum_{i=1}^{p} |b_i|. \tag{3}$$

Observe that, just like in the linear Lasso estimator (2), we scale the features so that their variance is unity [4]. This scaling need only be applied once on $\mathbf{X}$, possibly as a preprocessing step prior to the main optimization, and then reversed at the end of the optimization (in order to obtain the regression coefficients in the original units of measurement). The Lasso solution (3) can be also be

rewritten in an equivalent form to (1), namely,

$$(\hat{\beta}_{0,\lambda}, \hat{\boldsymbol{\beta}}_\lambda) = \underset{b_0,\boldsymbol{b}}{\operatorname{argmin}} \frac{-\ln g(\boldsymbol{y} \,|\, b_0, \boldsymbol{b}, \mathbf{X})}{n} + \lambda \sum_{i=1}^{p} \eta_i \times |b_i|. \tag{4}$$

We are now ready to describe our proposed methodology of rescaling.

New Rescaling Method for GLMs. Let $r(b_0, \boldsymbol{b}) := \frac{-\ln g(\boldsymbol{y} \,|\, b_0, \boldsymbol{b}, \mathbf{X})}{n}$ be our shorthand notation for the cross-entropy loss. We define $\eta_i^2(\boldsymbol{\beta})$ to be the i-th diagonal element of the $p \times p$ Hessian matrix of second derivatives:

$$\frac{\partial^2}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}^\top} \min_{b_0} r(b_0, \boldsymbol{\beta}).$$

This is the Hessian matrix of the cross-entropy loss, evaluated at the true parameter $\boldsymbol{\beta}$, and after the nuisance parameter β_0 is eliminated from the optimization. Then, instead of the usual rescaled Lasso estimator (4), in this article we propose the following alternative *iteratively rescaled Lasso* (IRL):

$$\underset{\boldsymbol{b},b_0}{\operatorname{argmin}} \frac{-\ln g(\boldsymbol{y} \,|\, b_0, \boldsymbol{b}, \mathbf{X})}{n} + \lambda \sum_{i=1}^{p} \eta_i(\boldsymbol{\beta}) \times |b_i|. \tag{5}$$

We now make three observations.

First, since each $\eta_i(\boldsymbol{\beta})$ depends on the unknown $\boldsymbol{\beta}$, the approximate computation of (5) will be iterative — this will be explained carefully in the next section — and is the main reason for naming the method IRL.

Second, the linear Lasso estimator (1) is a special case of (5) when $\boldsymbol{Y}$ is a multivariate Gaussian with mean $\mathbb{E}_{\mathbf{X}}[\boldsymbol{Y}] = \mathbf{1}\beta_0 + \mathbf{X}\boldsymbol{\beta}$ and variance $\mathbb{V}\mathrm{ar}_{\mathbf{X}}(\boldsymbol{Y}) = \mathbf{I}$, because then $\eta_i^2(\boldsymbol{\beta}) = \|\mathbf{C}\boldsymbol{v}_i\|^2/n$.

Third, one may ask what is the motivation for the proposed IRL estimator. The answer is that the IRL estimator coincides with the traditional linear regression estimator (1), provided that the cross-entropy loss $r(b_0, \boldsymbol{b})$ is replaced by its quadratic approximation in the neighborhood of the true coefficients $\beta_0, \boldsymbol{\beta}$. In other words, our proposed IRL estimator uses exactly the same scaling as the linear Lasso estimator (1) when the generalized linear model is *linearized* at the true solution. Note that there is no such agreement in the scaling between the currently accepted linear estimator (1) and its GLM counterpart (4), that is, the current widely-used scaling is not consistent across linear and nonlinear models. Our proposal is thus motivated by the desire for consistency in the scaling applied to linear and nonlinear models.

The rest of the paper is organized as follows. In Section 2 we explain how to approximately compute the Lasso estimator in (5), given that we do not actually have apriori knowledge of the Hessian matrix $\mathbf{H}$ at the true parameter $\boldsymbol{\beta}$. Then, in Section 3 we provide a numerical example with real data illustrating the scope of improvement in the statistical performance of the Lasso estimator.

2. Computation via Iterative Reweighted Least Squares

Since the true $\boldsymbol{\beta}$ is not known apriori, in this section we explain how to approximately compute (5) using an iterative reweighted least squares (IRLS) method. We begin by reviewing the well-known IRLS for computing the estimator (4) and then explain how it is easily modified to approximately compute our proposed estimator (5). To this end, we introduce the notation $\breve{\boldsymbol{b}} := [b_0, \boldsymbol{b}^\top]^\top$ and $\breve{\mathbf{X}} := [\mathbf{1}, \mathbf{X}]$, so that $\breve{\mathbf{X}}\breve{\boldsymbol{b}} = \mathbf{1}b_0 + \mathbf{X}\boldsymbol{b}$. Then, computing (4) is equivalent to minimizing $r(\breve{\boldsymbol{b}}) +$

$\lambda \sum_{i=1}^{p} \eta_i \times |b_i|$ with respect to $\breve{\boldsymbol{b}}$. This problem is nonlinear, but it can be solved by successive and repeated linearizations of $r(\breve{\boldsymbol{b}})$. Suppose that at iteration t, we have a current best guess $\breve{\boldsymbol{b}}_t$ for the minimizer of (4). Given this $\breve{\boldsymbol{b}}_t$, we consider the quadratic multivariate Taylor approximation to the cross-entropy loss at the point $\breve{\boldsymbol{b}}_t$:

$$r(\breve{\boldsymbol{b}}_t) + (\breve{\boldsymbol{b}} - \breve{\boldsymbol{b}}_t)^\top \nabla r(\breve{\boldsymbol{b}}_t) + \frac{1}{2}(\breve{\boldsymbol{b}} - \breve{\boldsymbol{b}}_t)^\top \mathbf{H}(\breve{\boldsymbol{b}}_t)(\breve{\boldsymbol{b}} - \breve{\boldsymbol{b}}_t).$$

Then, we update $\breve{\boldsymbol{b}}_t$ to $\breve{\boldsymbol{b}}_{t+1}$ by computing the linear Lasso estimator:

$$\breve{\boldsymbol{b}}_{t+1} := \underset{\breve{\boldsymbol{b}}}{\operatorname{argmin}} \ (\breve{\boldsymbol{b}} - \breve{\boldsymbol{b}}_t)^\top \nabla r(\breve{\boldsymbol{b}}_t) + \frac{1}{2}(\breve{\boldsymbol{b}} - \breve{\boldsymbol{b}}_t)^\top \mathbf{H}(\breve{\boldsymbol{b}}_t)(\breve{\boldsymbol{b}} - \breve{\boldsymbol{b}}_t) + \lambda \sum_{i=1}^{p} \eta_i \times |b_i|. \tag{6}$$

This computation is then iterated until convergence [2].

To keep the mathematical detail simple, we henceforth use the Logit model to illustrate the computations that are typically required for all GLMs. In the Logit model the binary response $Y_1, \ldots, Y_n$ are assumed to be independent Bernoulli random variables with conditional mean

$$\mathbb{E}_{\mathbf{X}}[Y_i] = \frac{1}{1 + \exp(-\beta_0 - \boldsymbol{x}_i^\top \boldsymbol{\beta})}, \quad i = 1, \ldots, n,$$

yielding the cross-entropy loss:

$$r(\breve{\boldsymbol{b}}) = \frac{1}{n} \sum_{i=1}^{n} (1 - y_i) \breve{\boldsymbol{x}}_i^\top \breve{\boldsymbol{b}} + \ln(1 + \exp(-\breve{\boldsymbol{x}}_i^\top \breve{\boldsymbol{b}})). \tag{7}$$

Then, the gradient $\nabla r(\breve{\boldsymbol{b}}_t)$ and Hessian $\mathbf{H}(\breve{\boldsymbol{b}}_t)$ are given by the formulas:

$$\begin{aligned}
\mu_{t,i} &:= (1 + \exp(-\breve{\boldsymbol{x}}_i^\top \breve{\boldsymbol{b}}_t))^{-1}, \\
\boldsymbol{\mu}_t &:= [\mu_{t,1}, \ldots, \mu_{t,n}]^\top, \\
\nabla r(\breve{\boldsymbol{b}}_t) &= \frac{1}{n} \breve{\mathbf{X}}^\top (\boldsymbol{\mu}_t - \boldsymbol{y}), \\
\boldsymbol{w}_t &:= \sqrt{\boldsymbol{\mu}_t \odot (\mathbf{1} - \boldsymbol{\mu}_t)}, \qquad (w_{t,i} = \sqrt{\mu_{t,i}(1 - \mu_{t,i})}, \ \forall i), \\
\mathbf{D}_t &:= \operatorname{diag}(\boldsymbol{w}_t), \\
\mathbf{H}(\breve{\boldsymbol{b}}_t) &= \frac{1}{n} \breve{\mathbf{X}}^\top \mathbf{D}_t^2 \breve{\mathbf{X}} = \frac{1}{n} \sum_{i=1}^{n} w_{t,i}^2 \breve{\boldsymbol{x}}_i \breve{\boldsymbol{x}}_i^\top.
\end{aligned}$$

If we define the quantities:

$$\begin{aligned}
\mathbf{X}_t &:= \mathbf{D}_t \mathbf{X}, \\
\boldsymbol{y}_t &:= \boldsymbol{w}_t b_{t,0} + \mathbf{X}_t \boldsymbol{b}_t + \mathbf{D}_t^{-1}(\boldsymbol{y} - \boldsymbol{\mu}_t),
\end{aligned}$$

then straightforward algebra shows that the estimator in (6) can also be written in terms of the following (weighed) regularized least-squares:

$$(b_{t+1,0}, \boldsymbol{b}_{t+1}) := \underset{b_0, \boldsymbol{b}}{\operatorname{argmin}} \|\boldsymbol{y}_t - \boldsymbol{w}_t b_0 - \mathbf{X}_t \boldsymbol{b}\|^2 + \lambda \sum_{i=1}^{p} \eta_i \times |b_i|.$$

We can eliminate the intercept term b_0 from the optimization by applying the centering (projection) matrix $\mathbf{C}_t := \mathbf{I}_n - \boldsymbol{w}_t\boldsymbol{w}_t^\top/\|\boldsymbol{w}_t\|^2$ to both $\mathbf{X}_t$ and $\boldsymbol{y}_t$. Once the $\boldsymbol{b}_{t+1}$ is computed, then we recover $b_{0,t+1} = \boldsymbol{w}_t^\top(\boldsymbol{y}_t - \mathbf{X}_t\boldsymbol{b}_{t+1})/\|\boldsymbol{w}_t\|^2$. This gives the following formulas for updating $(b_{0,t}, \boldsymbol{b}_t)$ to $(b_{0,t+1}, \boldsymbol{b}_{t+1})$:

$$\boldsymbol{b}_{t+1} := \underset{\boldsymbol{b}}{\operatorname{argmin}} \|\mathbf{C}_t\boldsymbol{y}_t - \mathbf{C}_t\mathbf{X}_t\boldsymbol{b}\|^2 + \lambda\sum_{i=1}^{p}\eta_i \times |b_i|$$

$$b_{0,t+1} := \frac{\boldsymbol{w}_t^\top(\boldsymbol{y}_t - \mathbf{X}_t\boldsymbol{b}_{t+1})}{\|\boldsymbol{w}_t\|^2}.$$

We iterate for $t = 1, 2, \ldots$ until a convergence criterion is met. This iterative reweighed penalized least squares is summarized in the following pseudo-algorithm [2], which assumes that we compute the Lasso estimate $(\hat{\beta}_{0,\lambda}, \hat{\boldsymbol{\beta}}_\lambda)$ on a grid of m values (with $m = 200$ being typical values):

$$\lambda_1 < \lambda_2 < \cdots < \lambda_m,$$

where λ_m is large enough so that $\hat{\boldsymbol{\beta}}_{\lambda_m} = \mathbf{0}$; see [2].

Algorithm 2.1: IRLS for Lasso-Logistic model [2].

input: $\boldsymbol{y}, \mathbf{X}$, error tolerance $\epsilon > 0$ and grid $\lambda_1 < \lambda_2 < \cdots < \lambda_m$.
output: Regularized solution $(b_{0,j}, \boldsymbol{b}_j)$ for each λ_j $j = 1, \ldots, m$

1 $\boldsymbol{b} \leftarrow \mathbf{0}$, $b_0 \leftarrow -\ln(1/\bar{y} - 1)$ // initializing values
2 **for** $j = m, m-1, m-2, \ldots, 2, 1$ **do** // outer loop over $\{\lambda_j\}$
3 $t \leftarrow 0$
4 **repeat** // middle loop is over the quadratic approximation
5 $\boldsymbol{b}_{\text{old}} \leftarrow \boldsymbol{b}$
6 $\boldsymbol{\mu} \leftarrow (\mathbf{1} + \exp(-\mathbf{1}b_0 - \mathbf{X}\boldsymbol{b}))^{-1}$ // mean response
7 $\boldsymbol{w} \leftarrow \sqrt{\boldsymbol{\mu} \odot (\mathbf{1} - \boldsymbol{\mu})}$ // weights
8 $\mathbf{X}_t \leftarrow \operatorname{diag}(\boldsymbol{w})\mathbf{X}$
9 $\boldsymbol{y}_t \leftarrow b_0\boldsymbol{w} + \mathbf{X}_t\boldsymbol{b} + (\boldsymbol{y} - \boldsymbol{\mu}) \div \boldsymbol{w}$ // adjusted & weighted response
10 $c_y \leftarrow \boldsymbol{y}_t^\top\boldsymbol{w}/\|\boldsymbol{w}\|^2$, $\boldsymbol{c}_x \leftarrow \mathbf{X}_t^\top\boldsymbol{w}/\|\boldsymbol{w}\|^2$ // adjustments for intercept
11 $\eta_i \leftarrow \sqrt{\|\mathbf{C}\boldsymbol{v}_i\|^2/n}$ for $i = 1, \ldots, p$ // square root of feature variance
12 $\boldsymbol{b} \leftarrow \text{lassoCD}(\boldsymbol{b}, \boldsymbol{y}_t - \boldsymbol{w}c_y, \mathbf{X}_t - \boldsymbol{w}\boldsymbol{c}_x^\top, \epsilon, \lambda_j, \boldsymbol{\eta})$ // inner loop
13 $b_0 \leftarrow c_y - \boldsymbol{c}_x^\top\boldsymbol{b}$ // intercept update given $\boldsymbol{b}$
14 $t \leftarrow t + 1$
15 **until** $\|\boldsymbol{b}_{\text{old}} - \boldsymbol{b}\| < \epsilon$
16 $(b_{0j}, \boldsymbol{b}_j) \leftarrow (b_0, \boldsymbol{b})$ // store values for each grid point
17 **return** $b_{0j}, \boldsymbol{b}_j, j = 1, \ldots, m$

Line 12 in Algorithm 2.1 calls the subroutine $\text{lassoCD}(\boldsymbol{b}, \boldsymbol{y}, \mathbf{X}, \epsilon, \lambda, \boldsymbol{\eta})$ to compute the Lasso estimate:

$$\underset{\boldsymbol{b}}{\operatorname{argmin}} \frac{\|\boldsymbol{y} - \mathbf{X}\boldsymbol{b}\|^2}{2n} + \lambda\sum_{i=1}^{p}\eta_i \times |b_i| \tag{8}$$

to within an error tolerance of ϵ via the method of coordinate descent. For completeness, we include the pseudocode of this subroutine in the Appendix and refer the reader to [2] for more details of this well-known algorithm.

We now describe our proposed method for approximately computing the Lasso estimate (5). The basic idea is to modify the standard linearization (6) by replacing the penalty term $\sum_{i=1}^{p} \eta_i \times |b_i|$ with $\sum_{i=1}^{p} \eta_{t,i} \times |b_i|$, where $\boldsymbol{\eta}_t$ is determined from the columns of the Hessian of the cross-entropy loss evaluated at the current estimate $\breve{\boldsymbol{b}}_t$. This gives an iterative reweighed least squares in which the scaling in the lasso penalty term is modified from one iteration to the next:

$$\breve{\boldsymbol{b}}_{t+1} := \underset{\breve{\boldsymbol{b}}}{\operatorname{argmin}} \; (\breve{\boldsymbol{b}} - \breve{\boldsymbol{b}}_t)^\top \nabla r(\breve{\boldsymbol{b}}_t) + \frac{(\breve{\boldsymbol{b}} - \breve{\boldsymbol{b}}_t)^\top \mathbf{H}(\breve{\boldsymbol{b}}_t)(\breve{\boldsymbol{b}} - \breve{\boldsymbol{b}}_t)}{2} + \lambda \sum_{i=1}^{p} \eta_{t,i} \times |b_i|.$$

We implement this iterative rescaling with only one minor modification of Algorithm 2.1. In particular, we replace line 11 in Algorithm 2.1 with the variances of the features in $\mathbf{X}_t$ weighted by $\sqrt{\mu_{t,i}(1-\mu_{t,i})}$, $\forall i$:

$$\boldsymbol{\eta}_t \leftarrow \sqrt{\operatorname{diag}((\mathbf{X}_t - \boldsymbol{w}\boldsymbol{c}_x^\top)^\top (\mathbf{X}_t - \boldsymbol{w}\boldsymbol{c}_x^\top))/n}.$$

In other words, rather than computing the square root of the diagonal elements of the matrix $\mathbf{X}^\top \mathbf{C}\mathbf{X}/n$ (as currently done in Algorithm 2.1), we instead compute the square root of the diagonal elements of the matrix $\mathbf{X}_t^\top \mathbf{C}_t \mathbf{X}_t/n$. This is the only significant difference between the implementations of the current widely-used constant scaling and our proposed iteratively rescaled Lasso (IRL) algorithm.

3. Numerical Results

We study the performance of Glmnet and IRL on the UCI ML Breast Cancer Wisconsin (Diagnostic) dataset [8]. Further numerical simulations can be found in the appendix. The ordering of the 569 observations in this dataset is randomized and we use 399 and 170 observations for training and testing respectively. On this dataset, we fit Glmnet and IRL over a range of 100 λ values and plot the test loss (10) in Figure 1. The results indicate that IRL achieves the lowest test loss and this occurs when $\|\hat{\boldsymbol{\beta}}_\lambda\|_1 = 223.58$, whereas for Glmnet the minimum test loss occurs when $\|\hat{\boldsymbol{\beta}}_\lambda\|_1 = 346.44$. Moreover, at their corresponding minimizers, IRL selects 13 non-zero features whereas Glmnet selects 14. In other words, the IRL can achieve a smaller test loss using a sparser model.

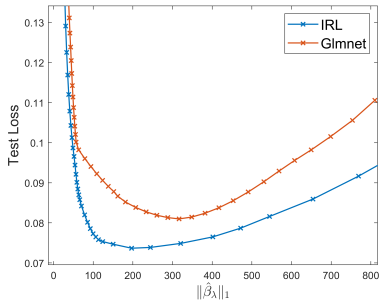


Figure 1: Test loss on the Breast Cancer Wisconsin (Diagnostic) dataset ($n = 569$, $p = 30$).

References

[1] Dimitris Bertsimas, Angela King, and Rahul Mazumder. Best subset selection via a modern optimization lens. 2016.

[2] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

[3] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[4] Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical learning with sparsity: the lasso and generalizations*. CRC press, 2015.

[5] Trevor Hastie, Robert Tibshirani, and Ryan Tibshirani. Best subset, forward stepwise or lasso? analysis and recommendations based on extensive comparisons. 2020.

[6] Dirk P Kroese, Zdravko Botev, Thomas Taimre, and Radislav Vaisman. *Data science and machine learning: mathematical and statistical methods*. CRC Press, 2019.

[7] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[8] William Wolberg, Olvi Mangasarian, Nick Street, and W. Street. Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository, 1995. DOI: https://doi.org/10.24432/C5DW2B.

Appendix A. Coordinate Descent for Linear Lasso Computations

The following code provides a simple coordinate descent algorithm [2] for computing (8). Note the absence here of the intercept term β_0, which is assumed to have been eliminated from the optimization. In the pseudocode below we use the Lasso shrinkage operator, defined as:

$$S_\lambda^{\text{Lasso}}(x) := x[1 - \lambda/|x|]_+,$$

where $x_+ = \max\{0, x\}$.

Algorithm A.1: Coordinate Descent for $\frac{1}{2n}\|\boldsymbol{y} - \mathbf{X}\boldsymbol{b}\|^2 + \lambda \sum_i \eta_i \times |b_i|$

input: initial $\boldsymbol{b}$, scaling parameter $\boldsymbol{\eta}$, and $\mathbf{X}, \epsilon, \lambda$
output: (global) minimizer $\boldsymbol{b}$

$\boldsymbol{r} \leftarrow \boldsymbol{y} - \mathbf{X}\boldsymbol{b}$ // initial residual
repeat // iterate over CD cycles
  $\boldsymbol{b}_{\text{old}} \leftarrow \boldsymbol{b}$
  for $k = 1, \ldots, p$ **do** // coordinate-descent cycle
    $b_{\text{new}} \leftarrow S^{\text{Lasso}}_{\lambda/\eta_k}(b_k + \boldsymbol{v}_k^\top \boldsymbol{r}/\|\boldsymbol{v}_k\|^2)$ // this step costs $\mathcal{O}(n)$
    if $b_{\text{new}} \neq b_k$ **then** // update only if necessary
      $\boldsymbol{r} \leftarrow \boldsymbol{r} + (b_k - b_{\text{new}})\boldsymbol{v}_k$ // this update costs $\mathcal{O}(n)$
      $b_k \leftarrow b_{\text{new}}$
until $\|\boldsymbol{b} - \boldsymbol{b}_{\text{old}}\| < \epsilon$
return $\boldsymbol{b}$

Appendix B. Further Numerical Simulations

In our simulations, we generate synthetic data from the model:

$$\boldsymbol{h} = \tau\left(\mathbf{1}\beta_0 + \mathbf{X}\boldsymbol{\beta}\right), \tag{9}$$

where the parameter $\tau > 0$ is a scaling parameter that determines the strength of the signal. When implementing Logistic regression, we sample a random vector $\boldsymbol{Y}$ by drawing components y_i from Bernoulli(μ_i), where $\mu_i = 1/(1 + \exp(-h_i))$ for all i. Each row of the predictor matrix $\mathbf{X}$ is generated from a multivariate normal distribution with zero mean and covariance $\boldsymbol{\Sigma}$ with diagonal elements $\boldsymbol{\Sigma}_{j,j} = 1$ and off-diagonal elements $\boldsymbol{\Sigma}_{i,j} = \rho^{\gamma|i-j|}$, for parameters $\rho \in (-1, 1)$ and $\gamma > 0$; see, e.g., [1, 5]. To induce sparsity in $\mathbf{X}$ we introduce the parameter $\xi > 0$ and set $\mathbf{X}_{i,j} = 0$ for all $\mathbf{X}_{i,j} < \xi$. We let the intercept $\beta_0 = 0$ and set the first 10 components of β as

$$\boldsymbol{\beta}_{[1:10]} = \begin{bmatrix} 25, & 4, & -4, & 50, & 4, & -4, & 75, & 4, & -4, & 100 \end{bmatrix}^\top,$$

and the remaining components to 0.

We run IRL and compare its statistical performance against the state-of-the-art implementation for generalized linear Lasso regression, Glmnet [2]. To tune the parameter λ we generate an independent validation set from the generating process (9) with identical parameter values for τ, ξ, ρ and γ. We then minimize the expected cross-entropy error on the validation set over a grid with 100 values.

Since the vector $\boldsymbol{\mu} = \mathbb{E}_{\mathbf{X}}[\boldsymbol{Y}]$ is known, when implementing Logistic regression, we can use the conditional expected cross-entropy error,

$$\mathbb{E}_{\mathbf{X}}\left[r(\breve{\boldsymbol{b}})\right] = \frac{1}{n}\sum_{i=1}^{n}(1-\mu_i)\breve{\boldsymbol{x}}_i^\top\breve{\boldsymbol{b}} + \ln(1+\exp(-\breve{\boldsymbol{x}}_i^\top\breve{\boldsymbol{b}})). \tag{10}$$

The expected loss conditioned on $\mathbf{X}$ allows one to estimate the true expected generalization risk.

In each simulation, after generating a training and validation set ($n = 1000$, $p = 100$) we run IRL and Glmnet to evaluate the λ that minimizes either (10) or (11) on the validation set. We denote this minimizer as λ^* and the corresponding model coefficient estimate as $\hat{\boldsymbol{\beta}}_{\lambda^*}$. We measure the following:

1. **Bias**: This is defined as $\|\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}_{\lambda^*}\|_2$.
2. **True Positives (TP)**: The number of correctly identified non-zero features in $\hat{\boldsymbol{\beta}}_{\lambda^*}$.
3. **False Positives (FP)**: The number of falsely identified non-zero features in $\hat{\boldsymbol{\beta}}_{\lambda^*}$.
4. **Test Loss**: The error (10) (Logistic) or (11) (Poisson) evaluated on an independently generated test set from model (9) of size n.

For each value of the pair (γ, ρ) we replicate the simulation 100 times and report the mean value of each performance measure in Table 1. Standard errors of the mean are provided in parentheses.

Table 1: **Logistic** - $\mathbf{X}$ is sparse ($\xi = 0.1$, $\tau = 1$).

		Average Variance = 0.047			
Method	(ρ, γ)	Bias	TP	FP	Test Loss
IRL	(0.1,0.1)	73.72	8.58 (0.13)	19.37 (0.43)	0.12
Glmnet		125.31	7.28 (0.11)	48.87 (0.74)	0.17
IRL	(0.1,1.0)	89.73	9.97 (0.02)	24.71 (0.53)	0.04
Glmnet		126.59	9.86 (0.04)	47.45 (0.71)	0.08
IRL	(0.9,0.1)	61.09	6.65 (0.13)	8.12 (0.29)	0.29
Glmnet		111.46	4.84 (0.08)	36.36 (0.68)	0.32
IRL	(0.9,1.0)	67.47	7.49 (0.13)	14.04 (0.38)	0.18
Glmnet		121.52	6.33 (0.12)	50.73 (0.73)	0.22

When conducting experiments with generalized linear models it is useful to estimate the noise inherent in the model. In Logistic regression, the variance of y_i is given by

$$\mathbb{V}\mathrm{ar}[y_i] = \mu_i(1-\mu_i).$$

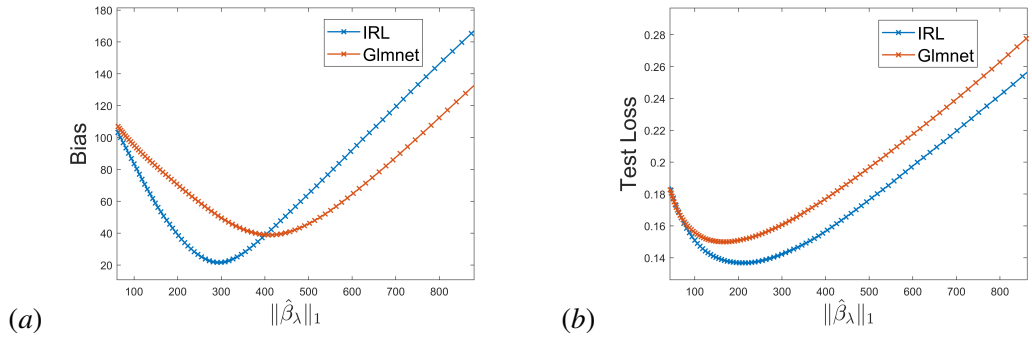
To quantify the signal strength we report the value

$$\sum_{i=1}^{n} \mu_i(1-\mu_i)/n$$

averaged over all the simulated responses in each table. The average variance has a maximum value of 0.25 which indicates high noise and a weak signal whereas an average variance closer to the minimum value of 0 indicates low noise and a strong signal.

In Table 1 we observe simulations with a strong signal. It is in this scenario that the IRL method significantly outperforms Glmnet. With IRL, we are able to obtain a sparser estimate $\hat{\boldsymbol{\beta}}_{\lambda^*}$ that contains significantly fewer false positives and achieves improved test loss. IRL outperforms Glmnet the most when the correlation is highest among the predictors ($\rho = 0.9$, $\gamma = 0.1$).

In Figure 2 we compare the bias and test loss of $\hat{\boldsymbol{\beta}}_\lambda$ computed with IRL and Glmnet over a grid of λ values. Since the parameter λ does not coincide between IRL and Glmnet, we plot the bias and test loss as a function of the dependent variable $\|\hat{\boldsymbol{\beta}}_\lambda\|_1$. Figure 2 shows that in the simulation setting where $\mathbf{X}$ is sparse, IRL attains lower test loss over a range of λ values and the lowest bias.



(*a*) (*b*)

Figure 2: Bias and test loss of $\hat{\boldsymbol{\beta}}_\lambda$ in a Logistic regression simulation. The estimate $\hat{\boldsymbol{\beta}}_\lambda$ is computed over a grid of 100 λ values where the simulation parameters are $\tau = 0.01$, $\xi = 0.1$, $\rho = 0.1$ and $\gamma = 1$.

We now present simulation results for Poisson regression. We generate data from the model (9) and sample $\boldsymbol{Y}$ by drawing y_i from Poisson(μ_i), where $\mu_i = \exp(h_i)$. We generate $\mathbf{X}$ as described in B and use the following conditional expected cross-entropy error,

$$\mathbb{E}_{\mathbf{X}}\left[r(\breve{\boldsymbol{b}})\right] = \frac{1}{n}\sum_{i=1}^{n} \exp(\breve{\boldsymbol{x}}_i^\top \breve{\boldsymbol{b}}) - \mu_i \breve{\boldsymbol{x}}_i^\top \breve{\boldsymbol{b}}. \tag{11}$$

In Poisson regression the variance of y_i is equal to its mean, that is, $\mathbb{V}\mathrm{ar}[y_i] = \mu_i$. To quantify the noise we report the value $\sum_{i=1}^n \mu_i/n$ averaged over all the simulated responses in each table. In Table 2 we observe simulations with low noise. Similar to Logistic regression, it is in this scenario that the IRL method significantly outperforms Glmnet. The IRL method is able to pick up significantly fewer false positives, thus exhibiting improved variable selection. The improvement is most significant when the correlation parameters are ($\rho = 0.9$, $\gamma = 1.0$). In Table 3, the simulated

Table 2: **Poisson** - $\mathbf{X}$ is sparse ($\xi = 0.1$, $\tau = 0.1$).

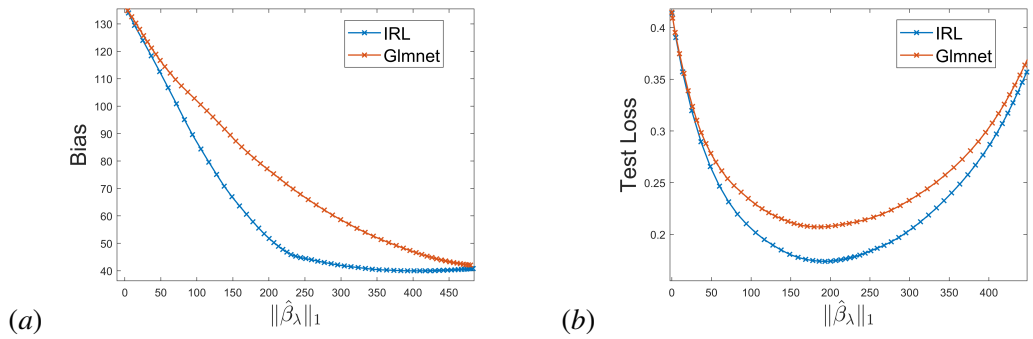
		Average Variance = 0.289			
Method	(ρ, γ)	Bias	TP	FP	Test Loss
IRL	(0.1,1.0)	35.98	9.14 (0.09)	20.76 (0.51)	0.16
Glmnet		73.62	9.62 (0.06)	47.56 (0.63)	0.19
IRL	(0.1,0.1)	36.13	5.69 (0.11)	11.03 (0.33)	0.28
Glmnet		71.19	6.30 (0.12)	43.78 (0.58)	0.31
IRL	(0.9,1.0)	35.65	5.80 (0.11)	8.66 (0.27)	0.36
Glmnet		69.55	6.09 (0.12)	41.54 (0.54)	0.39
IRL	(0.9,0.1)	41.86	6.09 (0.13)	4.80 (0.23)	0.49
Glmnet		61.78	5.36 (0.10)	23.26 (0.42)	0.51

Table 3: **Poisson** - $\mathbf{X}$ is not sparse ($\xi = \infty$, $\tau = 0.01$).

		Average Variance = 8.808			
Method	(ρ, γ)	Bias	TP	FP	Test Loss
IRL	(0.1,1.0)	11.6	8.22 (0.13)	19.02 (0.66)	-2.24
Glmnet		11.62	8.21 (0.13)	19.37 (0.66)	-2.24
IRL	(0.1,0.1)	11.64	6.44 (0.09)	8.14 (0.47)	-14.77
Glmnet		11.73	6.43 (0.10)	8.95 (0.49)	-14.77
IRL	(0.9,1.0)	11.98	6.58 (0.10)	6.72 (0.39)	-36.17
Glmnet		12.18	6.54 (0.10)	8.30 (0.45)	-36.17
IRL	(0.9,0.1)	19.91	6.32 (0.13)	2.81 (0.18)	-102.07
Glmnet		20.13	6.45 (0.13)	4.49 (0.30)	-102.07

$\mathbf{X}$ is non-sparse, and τ is decreased to 0.01, thereby increasing the average noise to 8.08. In this scenario, IRL outperforms Glmnet only in the high correlation settings. Figure 3 shows that in the sparse setting, IRL outperforms Glmnet over a range of λ values.

We now provide further numerical tables for Logistic regression which contain results from simulations with more values of τ. In Table 4, the simulated $\mathbf{X}$ is non-sparse, and τ is decreased thereby creating a weaker signal in the generated responses. In this high-noise setting, IRL achieves only marginally fewer false positives. In Table 5, $\mathbf{X}$ is simulated to be sparse and $\tau = 0.01$ which



(*a*) (*b*)

Figure 3: Bias and test loss of $\hat{\boldsymbol{\beta}}_\lambda$ in a Poisson regression simulation. The estimate $\hat{\boldsymbol{\beta}}_\lambda$ is computed over a grid of 100 λ values where the simulation parameters are $\tau = 0.1$, $\xi = 0.1$, $\rho = 0.1$ and $\gamma = 1$.

Table 4: **Logistic** - $\mathbf{X}$ is not sparse ($\xi = \infty$, $\tau = 0.01$).

		Average Variance = 0.156			
Method	(ρ, γ)	Bias	TP	FP	Test Loss
IRL	(0.1,0.1)	40.89	6.09 (0.12)	8.30 (0.26)	0.48
Glmnet		42.52	6.16 (0.12)	9.65 (0.26)	0.48
IRL	(0.1,1.0)	36.46	5.40 (0.10)	14.96 (0.42)	0.56
Glmnet		37.31	5.45 (0.11)	15.8 (0.43)	0.56
IRL	(0.9,0.1)	123.43	4.39 (0.11)	3.09 (0.18)	0.41
Glmnet		122.15	4.42 (0.11)	3.85 (0.20)	0.41
IRL	(0.9,1.0)	57.78	6.16 (0.13)	5.62 (0.22)	0.45
Glmnet		59.30	6.29 (0.12)	7.13 (0.24)	0.45

results in a weaker signal than that of in Table 1. In Table 6, $\mathbf{X}$ is simulated to be non-sparse and $\tau = 0.1$ which results in a stronger signal than that of in Table 4. In comparison to the weak signal setting in Table 4, there is a greater difference in the mean number of false positives in favor of the IRL method in Table 6.

Table 5: **Logistic** - $\mathbf{X}$ is sparse ($\xi = 0.1$, $\tau = 0.01$).

		Average Variance = 0.191			
Method	(ρ, γ)	Bias	TP	FP	Test Loss
IRL	(0.9,0.1)	146.29	3.55 (0.13)	3.61 (0.18)	0.55
Glmnet		143.38	3.66 (0.12)	4.56 (0.20)	0.55
IRL	(0.1,1.0)	55.2	4.49 (0.11)	12.27 (0.39)	0.57
Glmnet		58.07	4.64 (0.12)	13.85 (0.39)	0.57
IRL	(0.1,0.1)	63.74	5.79 (0.13)	7.37 (0.29)	0.56
Glmnet		66.26	5.87 (0.12)	9.29 (0.30)	0.56
IRL	(0.9,1.0)	80.68	5.62 (0.12)	5.94 (0.25)	0.56
Glmnet		81.78	5.72 (0.12)	7.90 (0.28)	0.56

Table 6: **Logistic** - $\mathbf{X}$ is not sparse ($\xi = \infty$, $\tau = 0.1$).

		Average Variance = 0.021			
Method	(ρ, γ)	Bias	TP	FP	Test Loss
IRL	(0.1,0.1)	49.29	6.97 (0.11)	23.59 (0.46)	0.08
Glmnet		53.75	6.85 (0.10)	30.79 (0.48)	0.09
IRL	(0.1,1.0)	48.54	9.34 (0.07)	43.09 (0.54)	0.12
Glmnet		50.32	9.40 (0.07)	47.15 (0.54)	0.12
IRL	(0.9,0.1)	58.08	6.34 (0.12)	4.65 (0.24)	0.06
Glmnet		66.95	6.42 (0.13)	10.96 (0.28)	0.06
IRL	(0.9,1.0)	45.34	6.77 (0.11)	15.47 (0.37)	0.07
Glmnet		53.09	6.53 (0.11)	23.93 (0.42)	0.07