# Regularization Paths for Generalized Linear Models via Coordinate Descent

Jerome Friedman Trevor Hastie*
Rob Tibshirani
Department of Statistics, Stanford University

May 19, 2008

**Abstract**

We develop fast algorithms for estimation of generalized linear models with convex penalties. The models include linear regression, two-class logistic regression, and multinomial regression problems while the penalties include $\ell_1$ (the lasso), $\ell_2$ (ridge regression) and mixtures of the two (the elastic net). The algorithms use cyclical coordinate descent, computed along a regularization path. The methods can handle large problems and can also deal efficiently with sparse features. In comparative timings we find that the new algorithms are considerably faster than competing methods.

## 1 Introduction

The lasso [Tibshirani, 1996] is a popular method for regression that uses an $\ell_1$ penalty to achieve a sparse solution. In the signal processing literature, the lasso is also known as *basis pursuit* [Chen et al., 1998]. This idea has been broadly applied, for example to generalized linear models [Tibshirani, 1996] and Cox's proportional hazard models for survival data [Tibshirani, 1997]. In recent years, there has been an enormous amount of research activity devoted to related regularization methods:

1. The grouped lasso [Yuan and Lin, 2007, Meier et al., 2008], where variables are included or excluded in groups;

*Corresponding author. email: hastie@stanford.edu. Sequoia Hall, Stanford University, CA94305.

2. The Dantzig selector [Candes and Tao, 2007, and discussion], a slightly modified version of the lasso;

3. The *elastic net* [Zhou and Hastie, 2005] for correlated variables, which uses a penalty that is part $\ell_1$, part $\ell_2$;

4. $\ell_1$ regularization paths for generalized linear models [Park and Hastie, 2006];

5. Regularization paths for the support-vector machine [Hastie et al., 2004].

6. The graphical lasso [Friedman et al., 2007b] for sparse covariance estimation and undirected graphs

Efron et al. [2004] developed an efficient algorithm for computing the entire regularization path for the lasso. Their algorithm exploits the fact that the coefficient profiles are piecewise linear, which leads to an algorithm with the same computational cost as the full least-squares fit on the data (see also Osborne et al. [2000]).

In some of the extensions above [2,3,5], piecewise-linearity can be exploited as in Efron et al. [2004] to yield efficient algorithms. Rosset and Zhu [2007] characterize the class of problems where piecewise-linearity exists—both the loss function and the penalty have to be quadratic or piecewise linear.

Here we instead focus on cyclical coordinate descent methods. These methods have been proposed for the lasso a number of times, but only recently was their power fully appreciated. Early references include Fu [1998] and Daubechies et al. [2004]. Van der Kooij [2007] independently used coordinate descent for solving elastic-net penalized regression models. Recent rediscoveries include Friedman et al. [2007a] and Wu and Lange [2007]. The first paper recognized the value of solving the problem along an entire path of values for the regularization parameters, using the current estimates as warm starts. This strategy turns out to be remarkably efficient for this problem. Several other researchers have re-discovered coordinate descent, many for solving the same problems we address in this paper—notably Krishnapuram and Hartemink [2005] and Genkin et al. [2007].

In this paper we extend the work of Friedman et al. [2007a] and develop fast algorithms for fitting generalized linear models with elastic-net penalties. In particular, our models include regression, two-class logistic regression, and multinomial regression problems. Our algorithms can work on

very large datasets, and can take advantage of sparsity in the feature set. We provide a publicly available R package `glmnet`.

In section 2 we present the algorithm for the elastic net, which includes the lasso and ridge regression as special cases. Section 3 and 4 discuss (two-class) logistic regression and multinomial logistic regression. Comparative timings are presented in Section 5.

## 2 Algorithms for the Lasso, Ridge Regression and the Elastic Net

We consider the usual setup for linear regression. We have a response variable $Y \in \mathbb{R}$ and a predictor vector $X \in \mathbb{R}^p$, and we approximate the regression function by a linear model $E(Y|X = x) = \beta_0 + x^T\beta$. We have $N$ observation pairs $(x_i, y_i)$. For simplicity we assume the $x_{ij}$ are standardized: $\sum_{i=1}^N x_{ij} = 0$, $\frac{1}{N}\sum_{i=1}^N x_{ij}^2 = 1$, for $j = 1, \ldots, p$. Our algorithms generalize naturally to the unstandardized case. The elastic net solves the following problem

$$\min_{(\beta_0,\beta)\in\mathbb{R}^{p+1}} \left[\frac{1}{2N}\sum_{i=1}^N (y_i - \beta_0 - x_i^T\beta)^2 + \lambda P_\alpha(\beta)\right], \tag{1}$$

where

$$\begin{aligned} P_\alpha(\beta) &= (1-\alpha)\frac{1}{2}||\beta||_{\ell_2}^2 + \alpha||\beta||_{\ell_1} &(2)\\ &= \sum_{j=1}^p \left[\tfrac{1}{2}(1-\alpha)\beta_j^2 + \alpha|\beta_j|\right] &(3)\end{aligned}$$

is the *elastic-net penalty* [Zhou and Hastie, 2005]. $P_\alpha$ is a compromise between the ridge-regression penalty ($\alpha = 0$) and the lasso penalty ($\alpha = 1$). This penalty is particularly useful in the $p \gg N$ situation, or any situation where there are many correlated predictor variables.

Ridge regression is known to shrink the coefficients of correlated predictors towards each other, allowing them to borrow strength from each other. In the extreme case of $k$ identical predictors, they each get identical coefficients with $1/k$th the size that any single one would get if fit alone. From a Bayesian point of view, the ridge penalty is ideal if there are many predictors, and all have non-zero coefficients (drawn from a Gaussian distribution).

Lasso, on the other hand, is somewhat indifferent to very correlated predictors, and will tend to pick one and ignore the rest. In the extreme

case above, the lasso problem breaks down. The Lasso penalty corresponds to a Laplace prior, which expects many coefficients to be zero or close to zero, and a small subset of non-zero coefficients.

The elastic net with $\alpha = 1 - \epsilon$ for some small $\epsilon > 0$ performs much like the lasso, but removes any degeneracies and wild behavior caused by extreme correlations. More generally, the entire family $P_\alpha$ creates a useful compromise between ridge and lasso. As $\alpha$ increases from 0 to 1, for a given $\lambda$ the sparsity of the solution to (1) (i.e. the number of coefficients equal to zero) increases monotonically from 0 to the sparsity of the lasso solution.

Figure 1 shows an example. The dataset is from Golub et al. [1999], consisting of 72 observations on 3571 genes measured with DNA microarrays. The observations fall in two classes: we treat this as a regression problem for illustration. The coefficient profiles from the first 10 steps from each of the three regularization methods are shown. The lasso admits at most $N = 72$ genes into the model, while ridge regression gives all 3571 genes non-zero coefficients. The elastic net a provides a compromise between these two methods, and has the effect of averaging genes that are highly correlated and then entering the averaged gene into the model. Using the algorithm described below, computation of the entire path of solutions for each method, at 100 values of the regularization parameter evenly spaced on the log-scale, took under a second in total. Because of the large number of non-zero coefficients for ridge regression, they are individually much smaller than the coefficients for the other methods.

Consider a coordinate descent step for solving (1). That is, suppose we have estimates $\tilde{\beta}_0$ and $\tilde{\beta}_\ell$ for $\ell \neq j$, and we wish to partially optimize with respect to $\beta_j$. Denote by $R(\beta_0, \beta)$ the objective function in (1). We would like to compute the gradient at $\beta_j = \tilde{\beta}_j$, which only exists if $\tilde{\beta}_j \neq 0$. If $\tilde{\beta}_j > 0$, then

$$\frac{\partial R}{\partial \beta_j}|_{\beta=\tilde{\beta}} = -\frac{1}{N}\sum_{i=1}^{N} x_{ij}(y_i - \tilde{\beta}_o - x_i^T\tilde{\beta}) + \lambda(1-\alpha)\beta_j + \lambda\alpha. \tag{4}$$

A similar expression exists if $\tilde{\beta}_j < 0$. Simple calculus shows [Donoho and Johnstone, 1994, Friedman et al., 2007a] that the coordinate-wise update has the form

$$\tilde{\beta}_j \leftarrow \frac{S\Big(\frac{1}{N}\sum_{i=1}^{N} x_{ij}(y_i - \tilde{y}_i^{(j)}),\ \lambda\alpha\Big)_+}{1 + \lambda(1-\alpha)} \tag{5}$$

where

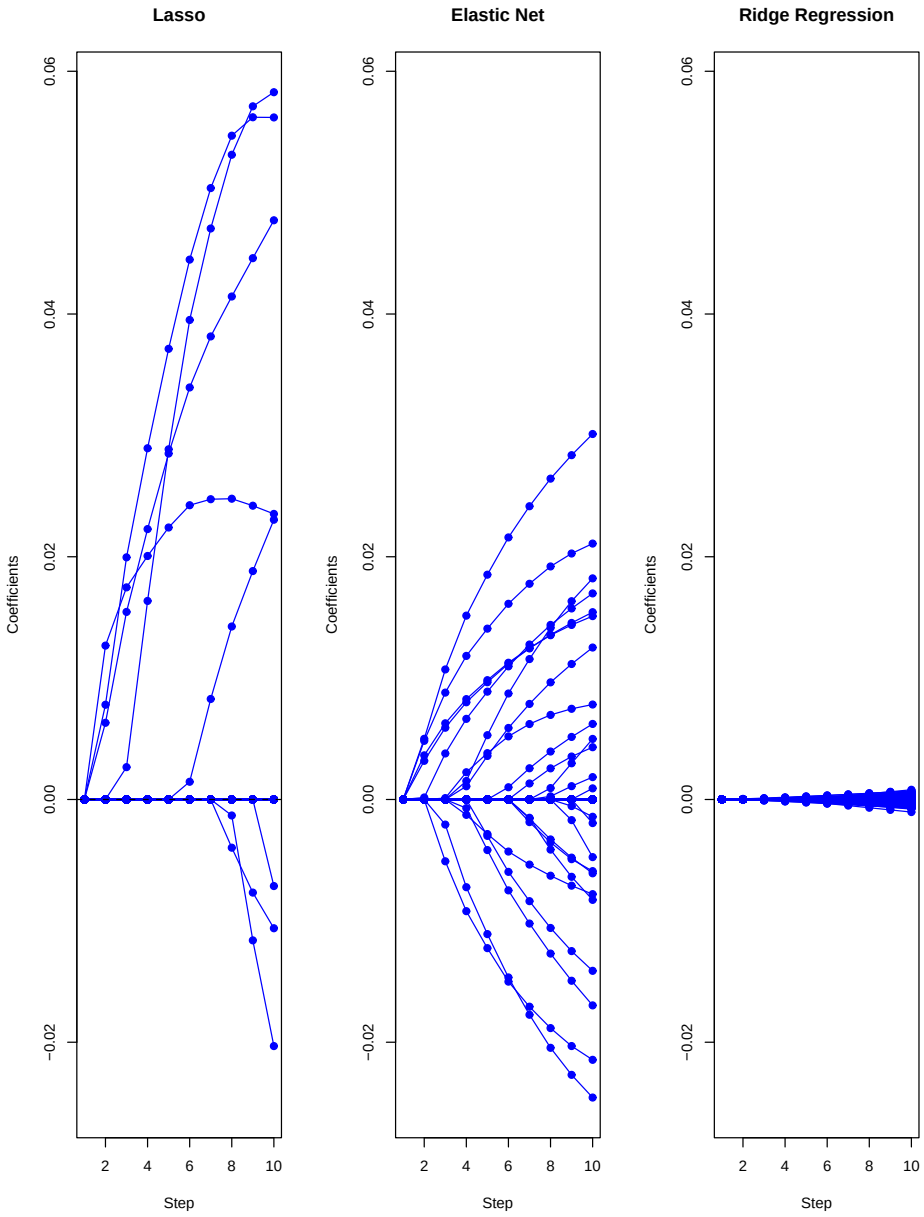


Figure 1: *Leukemia data: profiles of estimated coefficients for three methods, showing only first 10 steps in each case. For the elastic net,* $\alpha = 0.2$.

- $\tilde{y}_i^{(j)} = \tilde{\beta}_0 + \sum_{\ell \neq j} x_{i\ell}\tilde{\beta}_\ell$ is the fitted value excluding the contribution from $x_{ij}$, and hence $y_i - \tilde{y}_i^{(j)}$ the *partial residual* for fitting $\beta_j$. Because of the standardization, $\frac{1}{N}\sum_{i=1}^n x_{ij}(y_i - \tilde{y}_i^{(j)})$ is the simple least-squares coefficient when fitting this partial residual to $x_{ij}$.

- $S(z, \gamma)$ is the soft-thresholding operator with value

$$\text{sign}(z)(|z| - \gamma)_+ = \begin{cases} z - \gamma & \text{if } z > 0 \text{ and } \gamma < |z| \\ z + \gamma & \text{if } z < 0 \text{ and } \gamma < |z| \\ 0 & \text{if } \gamma \geq |z|. \end{cases} \tag{6}$$

Thus we compute the simple least-squares coefficient on the partial residual, apply soft-thresholding to take care of the lasso contribution to the penalty, and then apply a proportional shrinkage for the ridge penalty. This algorithm was suggested by Van der Kooij [2007].

### 2.1 Naive Updates

Looking more closely at (5), we see that

$$\begin{aligned} y_i - \tilde{y}_i^{(j)} &= y_i - \hat{y}_i + x_{ij}\tilde{\beta}_j \\ &= r_i + x_{ij}\tilde{\beta}_j, \end{aligned} \tag{7}$$

where $\hat{y}_i$ is the current fit of the model for observation $i$, and hence $r_i$ the current residual. Thus

$$\frac{1}{N}\sum_{i=1}^N x_{ij}(y_i - \tilde{y}_i^{(j)}) = \frac{1}{N}\sum_{i=1}^N x_{ij}r_i + \tilde{\beta}_j, \tag{8}$$

because the $x_j$ are standardized. The first term on the right-hand side is the gradient of the loss with respect to $\beta_j$. It is clear from (8) why coordinate descent is computationally efficient. Many coefficients are zero, remain zero after the thresholding, and so nothing needs to be changed. Such a step costs $O(N)$ operations— the sum to compute the gradient. On the other hand, if a coefficient does change after the thresholding, $r_i$ is changed in $O(N)$ and the step costs $O(2N)$. Thus a complete cycle through all $p$ variables costs $O(pN)$ operations. We refer to this as the *naive algorithm*, since it is generally less efficient than the *covariance updating* algorithm to follow. Later we use these algorithms in the context of iteratively reweighted least squares (IRLS), where the observation weights change frequently; there the naive algorithm dominates.

### 2.2 Covariance Updates

Further efficiencies can be achieved in computing the updates in (8). We can write the first term on the right (up to a factor $1/N$) as

$$\sum_{i=1}^{N} x_{ij} r_i = \langle x_j, y \rangle - \sum_{k:|\tilde{\beta}_k|>0} \langle x_j, x_k \rangle \tilde{\beta}_k, \tag{9}$$

where $\langle x_j, y \rangle = \sum_{i=1}^{N} x_{ij} y_i$. Hence we need to compute inner products of each feature with $y$ initially, and then each time a new feature $x_k$ enters the model (for the first time), we need to compute and store its inner product with all the rest of the features ($O(Np)$ operations). We also store the $p$ gradient components (9). If one of the coefficients currently in the model changes, we can update each gradient in $O(p)$ operations. Hence with $m$ non-zero terms in the model, a complete cycle costs $O(pm)$ operations if no new variables become non-zero, and costs $O(Np)$ for each new variable entered. Importantly, $O(N)$ calculations do not have to be made at every step. This is the case for all penalized procedures with squared error loss.

### 2.3 Sparse Updates

We are sometimes faced with problems where the $N \times p$ feature matrix $X$ is extremely sparse. A leading example is from document classification, where the feature vector uses the so-called "bag-of-words" model. Each document is scored for the presence/absence of each of the words in the entire dictionary under consideration (sometimes counts are used, or some transformation of counts). Since most words are absent, the feature vector for each document is mostly zero, and so the entire matrix is mostly zero. We store such matrices efficiently in *sparse column format*, where we store only the non-zero entries and the coordinates where they occur.

Coordinate descent is ideally set up to exploit such sparsity, in an obvious way. The $O(N)$ inner-product operations in either the naive or covariance updates can exploit the sparsity, by summing over only the non-zero entries. Note that in this case scaling of the variables will not alter the sparsity, but centering will. So scaling is performed up front, but the centering is incorporated in the algorithm in an efficient manner.

### 2.4 Weighted Updates

Often a weight $w_i$ (other than $1/N$) is associated with each observation. This will arise naturally in later sections where observations receive weights

in the IRLS algorithm. In this case the update step (5) becomes only slightly more complicated:

$$\tilde{\beta}_j \leftarrow \frac{S\Big(\sum_{i=1}^N w_i x_{ij}(y_i - \tilde{y}_i^{(j)}),\ \lambda\alpha\Big)_+}{\sum_{i=1}^N w_i x_{ij}^2 + \lambda(1-\alpha)}. \tag{10}$$

If the $x_j$ are not standardized, there is a similar sum-of-squares term in the denominator (even without weights). The presence of weights does not change the computational costs of either algorithm much, as long as the weights remain fixed.

### 2.5 Pathwise Coordinate Descent

We compute the solutions for a decreasing sequence of values for $\lambda$, starting at the smallest value $\lambda_{max}$ for which the entire vector $\hat{\beta} = 0$. Apart from giving us a path of solutions, this scheme exploits *warm starts*, and leads to a more stable algorithm. We have examples where it is faster to compute the path down to $\lambda$ (for small $\lambda$) than the solution only at that value for $\lambda$.

When $\tilde{\beta} = 0$, we see from (5) that $\tilde{\beta}_j$ will stay zero if $\frac{1}{N}|\langle x_j, y\rangle| < \lambda\alpha$. Hence $N\alpha\lambda_{max} = \max_\ell |\langle x_\ell, y\rangle|$. Our strategy is to select a minimum value $\lambda_{min} = \epsilon\lambda_{max}$, and construct a sequence of $K$ values of $\lambda$ decreasing from $\lambda_{\max}$ to $\lambda_{min}$ on the log scale. Typical values are $\epsilon = 0.001$ and $K = 100$.

### 2.6 Other Details

Irrespective of whether the variables are standardized to have variance 1, we always center each predictor variable. Since the intercept is not regularized, this means that $\hat{\beta}_0 = \bar{y}$, the mean of the $y_i$, for all values of $\alpha$ and $\lambda$.

It is easy to allow different penalties $\lambda_j$ for each of the variables. We implement this via a penalty scaling parameter $\gamma_j \geq 0$. If $\gamma_j > 0$, then the penalty applied to $\beta_j$ is $\lambda_j = \lambda\gamma_j$. If $\gamma_j = 0$, that variable does not get penalized, and always enters the model.

Considerable speedup is obtained by organizing the iterations around the *active set* of features—those with nonzero coefficients. After a complete cycle through all the variables, we iterate on only the active set till convergence. If another complete cycle does not change the active set, we are done, otherwise the process is repeated.

## 3 Regularized Logistic Regression

When the response variable is binary, the linear logistic regression model is often used. Denote by $G$ the response variable, taking values in $\mathcal{G} = \{1, 2\}$ (the labeling of the elements is arbitrary). The logistic regression model represents the class-conditional probabilities through a linear function of the predictors

$$\begin{aligned} \Pr(G = 1|x) &= \frac{1}{1 + e^{-(\beta_0 + x^T\beta)}}, \\ \Pr(G = 2|x) &= \frac{1}{1 + e^{+(\beta_0 + x^T\beta)}} \\ &= 1 - \Pr(G = 1|x). \end{aligned} \tag{11}$$

Alternatively, this implies that

$$\log \frac{\Pr(G = 1|x)}{\Pr(G = 2|x)} = \beta_0 + x^T\beta. \tag{12}$$

Here we fit this model by regularized maximum (binomial) likelihood. Let $p(x_i) = \Pr(G = 1|x_i)$ be the probability (11) for observation $i$ at a particular values for the parameters $(\beta_0, \beta)$, then we maximize the penalized log likelihood

$$\max_{(\beta_0,\beta)\in\mathbb{R}^{p+1}} \left[ \frac{1}{N} \sum_{i=1}^{N} \big\{ I(g_i = 1) \log p(x_i) + I(g_i = 2) \log(1 - p(x_i)) \big\} - \lambda P_\alpha(\beta) \right]. \tag{13}$$

Denoting $y_i = I(g_i = 1)$, the log-likelihood part of (13) can be written in the more explicit form

$$\ell(\beta_0, \beta) = \frac{1}{N} \sum_{i=1}^{N} y_i \cdot (\beta_0 + x_i^T\beta) - \log(1 + e^{(\beta_0 + x_i^T\beta)}), \tag{14}$$

a concave function of the parameters. The Newton algorithm for maximizing the (unpenalized) log-likelihood (14) amounts to iteratively reweighted least squares. Hence if the current estimates of the parameters are $(\tilde{\beta}_0, \tilde{\beta})$, we form a quadratic approximation to the negative log-likelihood (Taylor expansion about current estimates), which is

$$\ell_Q(\beta_0, \beta) = -\frac{1}{2N} \sum_{i=1}^{N} w_i(z_i - \beta_0 - x_i^T\beta) + C(\tilde{\beta}_0, \tilde{\beta})^2 \tag{15}$$

where

$$z_i = \tilde{\beta}_0 + x_i^T \tilde{\beta} + \frac{y_i - \tilde{p}(x_i)}{\tilde{p}(x_i)(1-\tilde{p}(x_i))}, \quad \text{(working response)} \tag{16}$$

$$w_i = \tilde{p}(x_i)(1-\tilde{p}(x_i)), \quad \text{(weights)} \tag{17}$$

and $\tilde{p}(x_i)$ is evaluated at the current parameters. The last term is constant. The Newton update is obtained by minimizing $\ell_Q$.

Our approach is similar. For each value of $\lambda$, we create an outer loop which computes the quadratic approximation $\ell_Q$ about the current parameters $(\tilde{\beta}_0, \tilde{\beta})$. Then we use coordinate descent to solve the penalized weighted least-squares problem

$$\min_{(\beta_0,\beta)\in\mathbb{R}^{p+1}} \left\{-\ell_Q(\beta_0,\beta) + \lambda P_\alpha(\beta)\right\}. \tag{18}$$

This amounts to a sequence of nested loops:

OUTER LOOP: Decrement $\lambda$.

MIDDLE LOOP: Update the quadratic approximation $\ell_Q$ using the current parameters $(\tilde{\beta}_0, \tilde{\beta})$.

INNER LOOP: Run the coordinate descent algorithm on the penalized weighted-least-squares problem (18).

There are several important details in the implementation of this algorithm.

- When $p \gg N$, one cannot run $\lambda$ all the way to zero, because the saturated logistic regression fit is undefined (parameters wander off to $\pm\infty$ in order to achieve probabilities of 0 or 1). Hence the default $\lambda$ sequence runs down to $\lambda_{min} = \epsilon\lambda_{max} > 0$.

- Care is taken to avoid coefficients diverging in order to achieve fitted probabilities of 0 or 1. When a probability is within $\varepsilon = 10^{-5}$ of 1, we set it to 1, and set the weights to $\varepsilon$. 0 is treated similarly.

- Our code has an option to approximate the Hessian terms by an exact upper-bound. This is obtained by setting the $w_i$ in (17) all equal to 0.25 [Krishnapuram and Hartemink, 2005].

- We allow the response data to be supplied in the form of a two-column matrix of counts, sometimes referred to as *grouped* data. We discuss this in more detail in Section 4.2.

- The Newton algorithm is not guaranteed to converge without step-size optimization. Our code does not implement any checks for divergence. We have a closed form expression for the starting solutions, and each subsequent solution is warm-started from the previous close-by solution, which generally makes the quadratic approximations quite accurate. We have not encountered any divergence problems so far.

# 4 Regularized Multinomial Regression

When the categorical response variable $G$ has $K > 2$ levels, the linear logistic regression model can be generalized to a multi-logit model. The traditional approach is to extend (12) to $K-1$ logits

$$\log \frac{\Pr(G=\ell|x)}{\Pr(G=K|x)} = \beta_{0\ell} + x^T\beta_\ell,\ \ell = 1,\ldots,K-1. \tag{19}$$

Here $\beta_\ell$ is a $p$-vector of coefficients. As in Zhu and Hastie [2004], here we choose a more symmetric approach. We model

$$\Pr(G=\ell|x) = \frac{e^{\beta_{0\ell}+x^T\beta_\ell}}{\sum_{k=1}^K e^{\beta_{0k}+x^T\beta_k}} \tag{20}$$

This parametrization is not estimable without constraints, because for any values for the parameters $\{\beta_{0\ell},\beta_\ell\}_1^K$, $\{\beta_{0\ell}-c_0,\beta_\ell-c\}_1^K$ gives identical probabilities (20). Regularization deals with this ambiguity in a natural way; see Section 4.1 below.

We fit the model (20) by regularized maximum (multinomial) likelihood. Using a similar notation as before, let $p_\ell(x_i) = \Pr(G=\ell|x_i)$, and let $g_i \in \{1,2,\ldots,K\}$ be the $i$th response. We maximize the penalized log-likelihood

$$\max_{\{\beta_{0\ell},\beta_\ell\}_1^K \in \mathbb{R}^{K(p+1)}} \left[\frac{1}{N}\sum_{i=1}^N \log p_{g_i}(x_i) - \lambda\sum_{\ell=1}^K P_\alpha(\beta_\ell)\right]. \tag{21}$$

Denote by $\mathbf{Y}$ the $N \times K$ *indicator* response matrix, with elements $y_{i\ell} = I(g_i = \ell)$. Then we can write the log-likelihood part of (21) in the more explicit form

$$\ell(\{\beta_{0\ell},\beta_\ell\}_1^K) = \frac{1}{N}\sum_{i=1}^N\left[\sum_{\ell=1}^K y_{i\ell}(\beta_{0\ell}+x_i^T\beta_\ell) - \log\left(\sum_{\ell=1}^K e^{\beta_{0\ell}+x_i^T\beta_\ell}\right)\right]. \tag{22}$$

The Newton algorithm for multinomial regression can be tedious, because of the vector nature of the response observations. Instead of weights

$w_i$ as in (17), we get weight *matrices*, for example. However, in the spirit of coordinate descent, we can avoid these complexities. We perform *partial Newton steps* by forming a partial quadratic approximation to the log-likelihood (22), allowing only $(\beta_{0\ell}, \beta_\ell)$ to vary for a single class at a time. It is not hard to show that this is

$$\ell_{Q\ell}(\beta_{0\ell}, \beta_\ell) = -\frac{1}{2N}\sum_{i=1}^{N} w_{i\ell}(z_{i\ell} - \beta_{0\ell} - x_i^T\beta_\ell)^2 + C(\{\tilde{\beta}_{0k}, \tilde{\beta}_k\}_1^K), \qquad (23)$$

where as before

$$z_{i\ell} = \tilde{\beta}_{0\ell} + x_i^T\tilde{\beta}_\ell + \frac{y_{i\ell} - \tilde{p}_\ell(x_i)}{\tilde{p}_\ell(x_i)(1-\tilde{p}_\ell(x_i))}, \qquad (24)$$

$$w_{i\ell} = \tilde{p}_\ell(x_i)(1-\tilde{p}_\ell(x_i)), \qquad (25)$$

Our approach is similar to the two-class case, except now we have to cycle over the classes as well in the outer loop. For each value of $\lambda$, we create an outer loop which cycles over $\ell$ and computes the partial quadratic approximation $\ell_{Q\ell}$ about the current parameters $(\tilde{\beta}_0, \tilde{\beta})$. Then we use coordinate descent to solve the penalized weighted least-squares problem

$$\min_{(\beta_{0\ell}, \beta_\ell)\in\mathbb{R}^{p+1}} \left\{-\ell_{Q\ell}(\beta_{0\ell}, \beta_\ell) + \lambda P_\alpha(\beta_\ell)\right\}. \qquad (26)$$

This amounts to the sequence of nested loops:

OUTER LOOP: Decrement $\lambda$.

MIDDLE LOOP (OUTER): Cycle over $\ell \in \{1, 2, \ldots, K, 1, 2 \ldots\}$.

MIDDLE LOOP (INNER): Update the quadratic approximation $\ell_{Q\ell}$ using the current parameters $\{\tilde{\beta}_{0k}, \tilde{\beta}_k\}_1^K$.

INNER LOOP: Run the co-ordinate descent algorithm on the penalized weighted-least-squares problem (26).

### 4.1 Regularization and Parameter Ambiguity

As was pointed out earlier, if $\{\beta_{0\ell}, \beta_\ell\}_1^K$ characterizes a fitted model for (20), then $\{\beta_{0\ell} - c_0, \beta_\ell - c\}_1^K$ gives an identical fit ($c$ is a $p$-vector). Although this means that the log-likelihood part of (21) is insensitive to $(c_0, c)$, the penalty is not. In particular, we can always improve an estimate $\{\beta_{0\ell}, \beta_\ell\}_1^K$ (w.r.t. (21)) by solving

$$\min_{c\in\mathbb{R}^p} \sum_{\ell=1}^{K} P_\alpha(\beta_\ell - c). \qquad (27)$$

This can be done separately for each coordinate, hence

$$c_j = \arg\min_t \sum_{\ell=1}^{K} \left[\tfrac{1}{2}(1-\alpha)(\beta_{j\ell} - t)^2 + \alpha|\beta_{j\ell} - t|\right]. \tag{28}$$

**Theorem 1** *Consider problem (28) for values $\alpha \in [0, 1]$. Let $\bar{\beta}_j$ be the mean of the $\beta_{j\ell}$, and $\beta_j^M$ a median of the $\beta_{j\ell}$ (and for simplicity assume $\bar{\beta}_j \leq \beta_j^M$. Then we have*

$$c_j \in [\bar{\beta}_j, \beta_j^M], \tag{29}$$

*with the left endpoint achieved if $\alpha = 0$, and the right if $\alpha = 1$.*

The two endpoints are obvious. The proof of Theorem 1 is given in the appendix. A consequence of the theorem is that a very simple search algorithm can be used to solve (28). The objective is piecewise quadratic, with knots defined by the $\beta_{j\ell}$. We need only evaluate solutions in the intervals including the mean and median, and those in between.

We *recenter* the parameters in each index set $j$ after each INNER MIDDLE LOOP step, using the the solution $c_j$ for each $j$.

Not all the parameters in our model are regularized. The intercepts $\beta_{0\ell}$ are not, and with our penalty modifiers $\gamma_j$ (section 2.6) others need not be as well. For these parameters we use mean centering.

### 4.2 Grouped and Matrix Responses

As in the two class case, the data can be presented in the form of a $N \times K$ matrix $m_{i\ell}$ of non-negative numbers. For example, if the data are grouped: at each $x_i$ we have a number of multinomial samples, with $m_{i\ell}$ falling into category $\ell$. In this case we divide each row by the row-sum $m_i = \sum_\ell m_{i\ell}$, and produce our response matrix $y_{i\ell} = m_{i\ell}/m_i$. $m_i$ becomes an observation weight. Our penalized maximum likelihood algorithm changes in a trivial way. The working response (24) is defined exactly the same way (using $y_{i\ell}$ just defined). The weights in (25) get augmented with the observation weight $m_i$:

$$w_{i\ell} = m_i \tilde{p}_\ell(x_i)(1 - \tilde{p}_\ell(x_i)). \tag{30}$$

Equivalently, the data can be presented directly as a matrix of class proportions, along with a weight vector. From the point of view of the algorithm, any matrix of positive numbers and any non-negative weight vector will be treated in the same way.

# 5 Timings

In this section we compare the run times of the coordinate-wise algorithm to some competing algorithms. These use the lasso penalty in both the regression and logistic regression settings. All timings were carried out on an Intel Xeon 2.80GH processor.

## 5.1 Regression with the Lasso

We generated Gaussian data with $N$ observations and $p$ predictors, with each pair of predictors $X_j, X_{j'}$ having the same population correlation $\rho$. We tried a number of combinations of $N$ and $p$, with $\rho$ varying from zero to 0.95. The outcome values were generated by

$$Y = \sum_{j=1}^{p} X_j \beta_j + k \cdot Z \tag{31}$$

where $\beta_j = (-1)^j \exp(-2(j-1)/20)$, $Z \sim N(0,1)$ and $k$ is chosen so that the signal-to-noise ratio is 3.0. The coefficients are constructed to have alternating signs and to be exponentially decreasing.

Table 1 shows the average CPU timings for the coordinatewise algorithm, and the LARS procedure [Efron et al., 2004]. All algorithms are implemented as R language functions. The coordinate-wise algorithm does all of its numerical work in Fortran, while LARS (written by Efron and Hastie) does much of its work in R, calling Fortran routines for some matrix operations. However comparisons in [Friedman et al., 2007a] showed that LARS was actually faster than a version coded entirely in Fortran. Comparisons between different programs are always tricky: in particular the LARS procedure computes the entire path of solutions, while the coordinate-wise procedure solves the problem for a set of pre-defined points along the solution path. In the orthogonal case, LARS takes $\min(N,p)$ steps: hence to make things roughly comparable, we called the latter two algorithms to solve a total of $\min(N,p)$ problems along the path. Table 1 shows timings in seconds averaged over three runs. We see that `glmnet` is considerably faster than LARS; the covariance-updating version of the algorithm is a little faster than the naive version when $N > p$ and a little slower when $p > N$. We had expected that high correlation between the features would increase the run time of `glmnet`, but this does not seem to be the case.

**Linear Regression — Dense Features**

| | Correlation | | | | | |
|---|---|---|---|---|---|---|
| | 0 | 0.1 | 0.2 | 0.5 | 0.9 | 0.95 |
| | $N = 1000,\ p = 100$ | | | | | |
| glmnet-naive | 0.05 | 0.06 | 0.06 | 0.09 | 0.08 | 0.07 |
| glmnet-cov | 0.02 | 0.02 | 0.02 | 0.02 | 0.02 | 0.02 |
| lars | 0.11 | 0.11 | 0.11 | 0.11 | 0.11 | 0.11 |
| | $N = 5000,\ p = 100$ | | | | | |
| glmnet-naive | 0.24 | 0.25 | 0.26 | 0.34 | 0.32 | 0.31 |
| glmnet-cov | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 |
| lars | 0.29 | 0.29 | 0.29 | 0.30 | 0.29 | 0.29 |
| | $N = 100,\ p = 1000$ | | | | | |
| glmnet-naive | 0.04 | 0.05 | 0.04 | 0.05 | 0.04 | 0.03 |
| glmnet-cov | 0.07 | 0.08 | 0.07 | 0.08 | 0.04 | 0.03 |
| lars | 0.73 | 0.72 | 0.68 | 0.71 | 0.71 | 0.67 |
| | $N = 100,\ p = 5000$ | | | | | |
| glmnet-naive | 0.20 | 0.18 | 0.21 | 0.23 | 0.21 | 0.14 |
| glmnet-cov | 0.46 | 0.42 | 0.51 | 0.48 | 0.25 | 0.10 |
| lars | 3.73 | 3.53 | 3.59 | 3.47 | 3.90 | 3.52 |
| | $N = 100,\ p = 20000$ | | | | | |
| glmnet-naive | 1.00 | 0.99 | 1.06 | 1.29 | 1.17 | 0.97 |
| glmnet-cov | 1.86 | 2.26 | 2.34 | 2.59 | 1.24 | 0.79 |
| lars | 18.30 | 17.90 | 16.90 | 18.03 | 17.91 | 16.39 |
| | $N = 100,\ p = 50000$ | | | | | |
| glmnet-naive | 2.66 | 2.46 | 2.84 | 3.53 | 3.39 | 2.43 |
| glmnet-cov | 5.50 | 4.92 | 6.13 | 7.35 | 4.52 | 2.53 |
| lars | 58.68 | 64.00 | 64.79 | 58.20 | 66.39 | 79.79 |

Table 1: *Timings (secs) for* `glmnet` *and lars algorithms for linear regression with lasso penalty. The first line is* `glmnet` *using* `naive` *updating while the second uses* `covariance` *updating. Total time for 100 $\lambda$ values, averaged over 3 runs.*

### 5.2 Lasso-logistic regression

We used the same simulation setup as above, except that we took the continuous outcome $y$, defined $p = 1/(1 + \exp(-y))$ and used this to generate a two-class outcome $z$ with $\text{Prob}(z = 1) = p, \text{Prob}(z = 0) = 1 - p$. We compared the speed of `glmnet` to the interior point method `l1lognet` proposed by Koh et al. [2007], Bayesian Logistic Regression (BBR) due to Genkin et al. [2007] and the Lasso Penalized Logistic (LPL) program supplied by Ken Lange [Wu and Lange, 2007]. The latter two methods also use a coordinate descent approach.

The BBR software automatically performs ten-fold cross-validation when given a set of $\lambda$ values. Hence we report the total time for ten-fold cross-validation for all methods using the same 100 $\lambda$ values for all. Table 2 shows the results; in some cases, we omitted a method when it was seen to be very slow at smaller values for $N$ or $p$.

Again we see that `glmnet` is the clear winner: it slows down a little under high correlation. The computation seems to be roughly linear in $N$, but grows faster that linear in $p$.

Table 3 shows some results when the feature matrix is sparse: we randomly set 95% of the feature values to zero. Again, the `glmnet` procedure is significantly faster than `l1lognet`.

### 5.3 Real data

Table 4 shows some timing results for four different datasets. For the Leukemia and Internet ad datasets, the BBR program used fewer than 100 $\lambda$ values so we estimated the total time by scaling up the time for smaller number of values. Again `glmnet` is considerably faster than the competing methods.

## 6 Discussion

Cyclical coordinate descent methods are a natural approach for solving convex problems with $\ell_1$ or $\ell_2$ constraints, or mixtures of the two (elastic net). Each coordinate-descent step is fast, with an explicit formula for each coordinate-wise minimization. The method also exploits the sparsity of the model, spending much of its time evaluating only inner products for variables with non-zero coefficients. Its computational speed both for large $N$ and $p$ are quite remarkable.

**Logistic Regression — Dense Features**

| | Correlation | | | | | |
|---|---|---|---|---|---|---|
| | 0 | 0.1 | 0.2 | 0.5 | 0.9 | 0.95 |
| | $N = 1000,\ p = 100$ | | | | | |
| glmnet | 1.65 | 1.81 | 2.31 | 3.87 | 5.99 | 8.48 |
| l1lognet | 31.475 | 31.86 | 34.35 | 32.21 | 31.85 | 31.81 |
| BBR | 40.70 | 47.57 | 54.18 | 70.06 | 106.72 | 121.41 |
| LPL | 24.68 | 31.64 | 47.99 | 170.77 | 741.00 | 1448.25 |
| | $N = 5000,\ p = 100$ | | | | | |
| glmnet | 7.89 | 8.48 | 9.01 | 13.39 | 26.68 | 26.36 |
| l1lognet | 239.88 | 232.00 | 229.62 | 229.49 | 22.19 | 223.09 |
| | $N = 100,000,\ p = 100$ | | | | | |
| glmnet | 78.56 | 178.45 | 205.94 | 274.33 | 552.48 | 638.50 |
| | $N = 100,\ p = 1000$ | | | | | |
| glmnet | 1.06 | 1.07 | 1.09 | 1.45 | 1.72 | 1.37 |
| l1lognet | 25.99 | 26.40 | 25.67 | 26.49 | 24.34 | 20.16 |
| BBR | 70.19 | 71.19 | 78.40 | 103.77 | 149.05 | 113.87 |
| LPL | 11.02 | 10.87 | 10.76 | 16.34 | 41.84 | 70.50 |
| | $N = 100,\ p = 5000$ | | | | | |
| glmnet | 5.24 | 4.43 | 5.12 | 7.05 | 7.87 | 6.05 |
| l1lognet | 165.02 | 161.90 | 163.25 | 166.50 | 151.91 | 135.28 |
| | $N = 100,\ p = 100,000$ | | | | | |
| glmnet | 137.27 | 139.40 | 146.55 | 197.98 | 219.65 | 201.93 |

Table 2: *Timings (seconds) for logistic models with lasso penalty. Total time for tenfold cross-validation over a grid of 100 $\lambda$ values.*

| | Logistic Regression — Sparse Features | | | | | |
|---|---|---|---|---|---|---|
| | Correlation | | | | | |
| | 0 | 0.1 | 0.2 | 0.5 | 0.9 | 0.95 |
| | $N = 1000,\ p = 100$ | | | | | |
| glmnet | 0.77 | 0.74 | 0.72 | 0.73 | 0.84 | 0.88 |
| l1lognet | 5.19 | 5.21 | 5.14 | 5.40 | 6.14 | 6.26 |
| BBR | 2.01 | 1.95 | 1.98 | 2.06 | 2.73 | 2.88 |
| | $N = 100,\ p = 1000$ | | | | | |
| glmnet | 1.81 | 1.73 | 1.55 | 1.70 | 1.63 | 1.55 |
| l1lognet | 7.67 | 7.72 | 7.64 | 9.04 | 9.81 | 9.40 |
| BBR | 4.66 | 4.58 | 4.68 | 5.15 | 5.78 | 5.53 |
| | $N = 10,000,\ p = 100$ | | | | | |
| glmnet | 3.21 | 3.02 | 2.95 | 3.25 | 4.58 | 5.08 |
| l1lognet | 45.87 | 46.63 | 44.33 | 43.99 | 45.60 | 43.16 |
| BBR | 11.80 | 11.64 | 11.58 | 13.30 | 12.46 | 11.83 |
| | $N = 100,\ p = 10,000$ | | | | | |
| glmnet | 10.18 | 10.35 | 9.93 | 10.04 | 9.02 | 8.91 |
| l1lognet | 130.27 | 124.88 | 124.18 | 129.84 | 137.21 | 159.54 |
| BBR | 45.72 | 47.50 | 47.46 | 48.49 | 56.29 | 60.21 |

Table 3: *Timings (seconds) for logistic model with lasso penalty and sparse features. Total time for ten-fold cross-validation over a grid of 100 $\lambda$ values.*

| Name | Type | $N$ | $p$ | glmnet | l1logreg | BBR/BMR |
|---|---|---|---|---|---|---|
| | | | Dense | | | |
| Cancer | 14 class | 144 | 16,063 | 2.5 mins | | 2.1 hrs |
| Leukemia | 2 class | 72 | 3571 | 2.50 | 55.0 | 450 |
| | | | Sparse | | | |
| Internet ad | 2 class | 2359 | 1430 | 5.0 | 20.9 | 34.7 |
| Newsgroup | 2 class | 11,314 | 4,802,169 | 2 mins | 3.5 hrs | |

Table 4: *Timings (seconds, unless staed otherwise) for some real datasets. For the Cancer, Leukemia and Internet Ad datasets, times are for ten-fold cross-valdiation over 100 $\lambda$ values; for Newsgroup we performed a single run with* 100 *values of $\lambda$, with $\lambda_{min} = 0.05\lambda_{max}$.*

A public domain R language package `glmnet` is available from the CRAN website, as well as from the second author's website.

## Acknowledgments

We would like to thank Holger Hoefling for helpful discussions. Friedman was partially supported by grant DMS-97-64431 from the National Science Foundation. Hastie was partially supported by grant DMS-0505676 from the National Science Foundation, and grant 2R01 CA 72028-07 from the National Institutes of Health. Tibshirani was partially supported by National Science Foundation Grant DMS-9971405 and National Institutes of Health Contract N01-HV-28183.

## Appendix

*Proof of theorem 1.* We have

$$c_j = \arg\min_t \sum_{\ell=1}^{K} \left[\tfrac{1}{2}(1-\alpha)(\beta_{j\ell} - t)^2 + \alpha|\beta_{j\ell} - t|\right]. \tag{32}$$

Suppose $\alpha \in (0,1)$. Differentiating w.r.t. $t$ (using a sub-gradient respresentation), we have

$$\sum_{\ell=1}^{K}[-(1-\alpha)(\beta_{j\ell}-t)-\alpha s_{j\ell}]=0 \tag{33}$$

where $s_{j\ell}=\text{sign}(\beta_{j\ell}-t)$ if $\beta_{j\ell}\neq t$ and $s_{j\ell}\in[-1,1]$ otherwise. This gives

$$t=\bar{\beta}_j+\frac{1}{K}\frac{\alpha}{1-\alpha}\sum_{\ell=1}^{K}s_{j\ell} \tag{34}$$

It follows that $t$ cannot be larger than $\beta_j^M$ since then the second term above would be negative and this would imply that $t$ is less than $\bar{\beta}_j$. Similarly $t$ cannot be less than $\bar{\beta}_j$, since then the second term above would have to be negative, implying that $t$ is larger than $\beta_j^M$.

## References


E. Candes and T. Tao. The dantzig selector: Statistical estimation when p is much larger than n. *Annals of Statistics*, 35(6):2313–2351, 2007.

S. S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, pages 33–61, 1998.

I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57:1413–1457, 2004.

D. Donoho and I. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81:425–455, 1994.

B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, (2):407–499, 2004.

J. Friedman, T. Hastie, H. Hoefling, and R. Tibshirani. Pathwise coordinate optimization. *Annals of Applied Statistics*, 2(1):302–332, 2007a.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 2007b.

W. Fu. Penalized regressions: the bridge vs the lasso. *JCGS*, 7(3):397–416, 1998.

A. Genkin, D. Lewis, and D. Madigan. Large-scale bayesian logistic regression for text categorization. *Technometrics*, 49(3):291–304, 2007.

T. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 286:531–536, 1999.

T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, (5):1391–1415, 2004.

K. Koh, S.-J. Kim, and S. Boyd. An interior-point method for large-scale l1-regularized logistic regression. *Journal of Machine Learning Research*, 8:1519–1555, 2007.

B. Krishnapuram and A. J. Hartemink. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(6):957–968, 2005. ISSN 0162-8828. doi: http://dx.doi.org/10.1109/TPAMI.2005.127. Fellow-Lawrence Carin and Senior Member-Mario A. T. Figueiredo.

L. Meier, S. van de Geer, and P. Bhlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society B*, 70:53–71, 2008.

M. Osborne, B. Presnell, and B. Turlach. A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis*, 20: 389–404, 2000.

M. Y. Park and T. Hastie. An l1 regularization-path algorithm for generalized linear models. unpublished, 2006.

S. Rosset and J. Zhu. Adaptable, efficient and robust methods for regression and classification via piecewise linear regularized coefficient paths. *Annals of Statistics*, 35(3), 2007.

R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Royal. Statist. Soc. B.*, 58:267–288, 1996.

R. Tibshirani. The lasso method for variable selection in the cox model. *Statistics in Medicine*, 16:385–395, 1997.

A. Van der Kooij. Prediction accuracy and stability of regrsssion with optimal sclaing transformations. Technical report, Dept. of Data Theory, Leiden University, 2007.

T. Wu and K. Lange. Coordinate descent procedures for lasso penalized regression. *Annals of Applied Statistics*, 2007.

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *J. Royal. Statist. Soc. B*, 68(1):49–67, 2007.

H. Zhou and T. Hastie. Regularization and variable selection via the elastic net. *J. Royal. Stat. Soc. B.*, 67(2):301–320, 2005.

J. Zhu and T. Hastie. Classification of gene microarrays by penalized logistic regression. *Biostatistics*, 5(2):427–443, 2004.