Journal of Statistical Computation & Simulation
Vol. 00, No. 00, Month 200x, 1–12

RESEARCH ARTICLE

A Coordinate Majorization Descent Algorithm for ℓ_1 Penalized Learning

Yi Yang [a] and Hui Zou [a*]

[a] *School of Statistics, University of Minnesota, U.S.A.*

(*Revised version: 6 May, 2012*)

The glmnet package by [1] is an extremely fast implementation of the standard coordinate descent algorithm for solving ℓ_1 penalized learning problems. In this paper, we consider a family of coordinate majorization descent algorithms for solving the ℓ_1 penalized learning problems by replacing each coordinate descent step with a coordinate-wise majorization descent operation. Numerical experiments show that this simple modification can lead to substantial improvement in speed when the predictors have moderate or high correlations.

Keywords: Coordinate decent; Majorization minimization; Glmnet; Lasso.

1. Introduction

The lasso [2] is a very popular technique for high-dimensional modeling. A key contributor to the tremendous popularity of the lasso is the celebrated *Least Angle Regression* (LARS) algorithm proposed by [3]. LARS efficiently produces the piecewise linear solution paths of the lasso penalized least squares with the computational cost of a single least squares fit. Another efficient algorithm for solving the lasso is the *cyclical coordinate descent* algorithm. [4] developed the first working coordinate descent algorithm for solving the lasso. Some recent papers have made coordinate descent a popular computational algorithm for sparse regression. See [5], [6], [7], among others.

[8] proposed the elastic net penalty as an improved variant of the lasso to better handle correlated variables and to stabilize the lasso solution paths. The ℓ_1 component of the elastic net is responsible for achieving sparsity. Hence one can regard the elastic net as a member of the ℓ_1 penalized methods. [8] developed the LARS-EN algorithm for computing the entire solution paths of the elastic net penalized least squares. [1] developed the glmnet package to implement a coordinate descent algorithm for fitting the entire lasso or elastic net regularization paths for generalized linear models. Numerical experiments in [1] showed that glmnet is faster than the other publicly available packages for solving the ℓ_1 penalized models.

In this paper, we consider a family of coordinate majorization descent algorithms including the classical coordinate descent as a special case. The generalization is actually very straightforward. We simply replace each of the coordinate descent step with a coordinate majorization descent (CMD) operation and everything else in glmnet stays the same. Numerical experiments show that this simple CMD trick

*Corresponding author. Email: zouxx019@umn.edu.

ISSN: 0094-9655 print/ISSN 1563-5163 online

DOI:
http://www.informaworld.com

can lead to substantial improvement in speed when the predictors have moderate or high correlations.

2. Coordinate Majorization Descent

2.1. *Review of glmnet*

Because we present an improved glmnet algorithm, it is convenient and necessary to review some key elements of glmnet first. Consider the penalized least squares (PLS) problem. Given a training dataset with N observations (x_i, y_i) where x denotes a p dimensional predictor vector and y is a continuous response. Without loss of generality, let us assume that the predictors are standardized: $\sum_{i=1}^N x_{ij} = 0$, $\frac{1}{N}\sum_{i=1}^N x_{ij}^2 = 1$, for $j = 1, \ldots, p$. We use a linear function $\beta_0 + x^\intercal\beta$ to predict y. Define a penalized residual sum squares as follows

$$R(\beta_0, \beta) = \frac{1}{2N}\sum_{i=1}^N (y_i - \beta_0 - x_i^\intercal\beta)^2 + P_{\lambda,\alpha}(\beta), \tag{1}$$

where $P_{\lambda,\alpha}(\beta)$ is the elastic net penalty [8] and it is defined as

$$P_{\lambda,\alpha}(\beta) = \lambda\textstyle\sum_{j=1}^p p_\alpha(\beta_j) = \lambda\displaystyle\sum_{j=1}^p \left[\frac{1}{2}(1-\alpha)\beta_j^2 + \alpha|\beta_j|\right]. \tag{2}$$

Then the fitted model is obtained via $(\hat\beta, \hat\beta_0) = \underset{(\beta_0,\beta)\in\mathbb{R}^{p+1}}{\arg\min}\ R(\beta_0, \beta)$. The elastic net with $\alpha = 1$ reduces to the lasso. When the predictors exhibit strong correlation, using some $\alpha < 1$ yields better prediction accuracy.

For each fixed λ, cyclic coordinate descent can be easily implemented for solving the elastic net. To keep our discussion concise, we refer interested readers to [1] for more details. We just discuss the main ideas. Let $r_i = y_i - \tilde\beta_0 - x_i^\intercal\tilde\beta$ be the current residual. To update the estimate for β_j we need to solve a univariate elastic net problem

$$\hat\beta_j = \underset{\beta_j}{\arg\min} R(\beta_j|\tilde\beta_0, \tilde\beta), \tag{3}$$

where

$$R(\beta_j|\tilde\beta_0, \tilde\beta) = \frac{1}{2}\left(\beta_j - \tilde\beta_j\right)^2 - \frac{1}{N}\sum_{i=1}^N r_i x_{ij}\left(\beta_j - \tilde\beta_j\right) + \lambda p_\alpha(\beta_j). \tag{4}$$

It turns out that (3) has a simple closed form solution [8]

$$\hat\beta_j = \frac{S\left(\frac{1}{N}\sum_{i=1}^N x_{ij}r_i + \tilde\beta_j, \lambda\alpha\right)}{1 + \lambda(1-\alpha)}, \tag{5}$$

where $S(z, t) = (|z| - t)_+\mathrm{sgn}(z)$. We next set $\tilde\beta_j = \hat\beta_j$ as the new estimate. The operation is sequentially conducted on each coordinate β_j till convergence. See Algorithm **1**.

Algorithm 1 The coordinate descent algorithm for the elastic net PLS

(1) Initialize $(\tilde{\beta}_0, \tilde{\beta})$.
(2) Cyclic coordinate descent, for $j = 1, 2, \ldots, p$: compute $r_i = y_i - \tilde{\beta}_0 - x_i^\intercal \tilde{\beta}$ and

$$\hat{\beta}_j = \frac{S\left(\frac{1}{N}\sum_{i=1}^N x_{ij}r_i + \tilde{\beta}_j, \lambda\alpha\right)}{1 + \lambda(1-\alpha)}.$$

(3) Set $\tilde{\beta}_j = \hat{\beta}_j$.
(4) Repeat steps $2 - 3$ until convergence of $\hat{\beta}$.

Algorithm 2 The CMD algorithm for the elastic net PLS

(1) Initialize $(\tilde{\beta}_0, \tilde{\beta})$.
(2) Cyclic coordinate descent, for $j = 1, 2, \ldots, p$: compute $r_i = y_i - \tilde{\beta}_0 - x_i^\intercal \tilde{\beta}$ and

$$\hat{\beta}_j^{\mathrm{B}} = \frac{S\left(\frac{1}{N}\sum_{i=1}^N x_{ij}r_i + f \cdot \tilde{\beta}_j, \lambda\alpha\right)}{f \cdot 1 + \lambda(1-\alpha)}. \quad (f \geq 1)$$

(3) Set $\tilde{\beta}_j = \hat{\beta}_j^{\mathrm{B}}$.
(4) Repeat steps $2 - 3$ until convergence of $\hat{\beta}$.

2.2. *The majorization trick*

We now introduce a family of generalized coordinate descent algorithms. We consider modifying the update formula (5) as follows

$$\hat{\beta}_j^{\mathrm{B}} = \frac{S\left(\frac{1}{N}\sum_{i=1}^N x_{ij}r_i + f \cdot \tilde{\beta}_j, \lambda\alpha\right)}{f \cdot 1 + \lambda(1-\alpha)} \quad (f \geq 1) \tag{6}$$

and hence Algorithm 1 becomes Algorithm **2**.

Comparing (5) and (6), one can see that the generalization lies in an extra constant factor f. When $f = 1$, (6) reduces to (5). We will show that as long as f is greater or equal to 1, Algorithm 2 is guaranteed to converge. We have verified that with $f = 1$ glmnet2 and glmnet not only give exactly identical solutions but also use the same timing. Interestingly, when using some f greater than 1, Algorithm 2 can enjoy faster convergence than Algorithm 1. In numerical experiments presented in this paper we set $f = 2$ unless stated otherwise.

The convergence property of Algorithm 1 comes from the fact that each operation by (5) minimizes the objective function along the jth coordinate direction, which is the basic idea of coordinate descent. We show that each operation by (6) at least decreases the objective function along the jth coordinate direction if $f > 1$. To appreciate this fact, we invoke the Majorization-Minimization (MM) principle [9–11] which can be regarded as a more generalized form of the famous Expectation-Maximization algorithm [12].

To apply the MM principle, define

$$Q(\beta_j) = \frac{1}{2} f \cdot \left(\beta_j - \tilde{\beta}_j\right)^2 - \frac{1}{N} \sum_{i=1}^{N} r_i x_{ij} \left(\beta_j - \tilde{\beta}_j\right) + \lambda p_\alpha(\beta_j). \tag{7}$$

Note that $\hat{\beta}_j^{\mathrm{B}}$ actually minimizes $Q(\beta_j)$, i.e.,

$$\hat{\beta}_j^{\mathrm{B}} = \arg\min_{\beta_j} Q(\beta_j). \tag{8}$$

On the other hand, we have

$$Q(\beta_j) - R(\beta_j|\tilde{\beta}_0, \tilde{\beta}) = \frac{1}{2}(f-1)(\beta_j - \tilde{\beta}_j)^2. \tag{9}$$

Therefore, for any $f > 1$, $Q(\beta_j) > R(\beta_j|\tilde{\beta}_0, \tilde{\beta})$ unless $\beta_j = \tilde{\beta}_j$. Hence we have

$$\begin{aligned} R(\hat{\beta}_j^{\mathrm{B}}|\tilde{\beta}_0, \tilde{\beta}) &= Q(\hat{\beta}_j^{\mathrm{B}}) + R(\hat{\beta}_j^{\mathrm{B}}|\tilde{\beta}_0, \tilde{\beta}) - Q(\hat{\beta}_j^{\mathrm{B}}) && (10) \\ &\leq Q(\tilde{\beta}_j) && (11) \\ &= R(\tilde{\beta}_j|\tilde{\beta}_0, \tilde{\beta}). && (12) \end{aligned}$$

Obviously, $R(\hat{\beta}_j^{\mathrm{B}}|\tilde{\beta}_0, \tilde{\beta}) = R(\tilde{\beta}_j|\tilde{\beta}_0, \tilde{\beta})$ if and only if $\hat{\beta}_j^{\mathrm{B}} = \tilde{\beta}_j$. The above arguments show that Algorithm 2 retains the essential descent property of the original coordinate descent algorithm. So it is a genuine coordinate-wise descent algorithm. Because the MM principle is crucial to its descent property, Algorithm 2 is named coordinate majorization descent algorithm.

Unlike Algorithm 1, Algorithm 2 does not take the steepest descent step along each coordinate direction. This seems counterintuitive, as we usually want to decrease the objective function as much as we can at each iteration. In fact, when the predictors are uncorrelated, Algorithm 1 gives the exact solution after one cycle, while Algorithm 2 still needs to iterate. Thus we expect to see Algorithm 1 is faster than Algorithm 2 when the predictors are uncorrelated or nearly uncorrelated. However, in high dimensional data the predictors often have strong correlations or many moderate correlations. What we have found is that in such more complex situations Algorithm 2 can be substantially faster than Algorithm 1. In Section 3.4 we offer some explanation to this interesting phenomenon.

2.3. *Penalized weighted least squares and logistic regression*

Often we need to assign a weight ω_i (other than $1/N$) to each observation in least squares. For example, weighted least squares handles linear regression with heteroscadestic variance and iterative weighted least squares is a classical algorithm for fitting many statistical models. Both Algorithm 1 and Algorithm 2 can be easily modified to deal with the elastic net penalized weighted least squares (PWLS) in which the objective function is

$$\min_{(\beta_0, \beta) \in \mathbb{R}^{p+1}} \frac{1}{2N} \sum_{i=1}^{N} \omega_i (y_i - \beta_0 - x_i^{\intercal} \beta)^2 + P_{\lambda,\alpha}(\beta). \tag{13}$$

Algorithm 3 The coordinate descent algorithm for the elastic net PWLS

(1) Initialize $(\tilde{\beta}_0, \tilde{\beta})$.
(2) Cyclic coordinate descent, for $j = 1, 2, \dots, p$: compute $r_i = y_i - \tilde{\beta}_0 - x_i^\intercal \tilde{\beta}$ and

$$\hat{\beta}_j = \frac{S\left(\frac{1}{N}\sum_{i=1}^N \omega_i x_{ij} r_i + \left(\frac{1}{N}\sum_{i=1}^N \omega_i x_{ij}^2\right)\tilde{\beta}_j, \lambda\alpha\right)}{\frac{1}{N}\sum_{i=1}^N \omega_i x_{ij}^2 + \lambda(1-\alpha)}. \tag{14}$$

(3) Set $\tilde{\beta}_j = \hat{\beta}_j$.
(4) Repeat steps $2-3$ until convergence of $\hat{\beta}$.

Algorithm 4 The CMD algorithm for the elastic net PWLS

(1) Initialize $(\tilde{\beta}_0, \tilde{\beta})$.
(2) Cyclic coordinate descent, for $j = 1, 2, \dots, p$: compute $r_i = y_i - \tilde{\beta}_0 - x_i^\intercal \tilde{\beta}$ and

$$\hat{\beta}_j^{\mathrm{B}} = \frac{S\left(\frac{1}{N}\sum_{i=1}^N \omega_i x_{ij} r_i + f\cdot\left(\frac{1}{N}\sum_{i=1}^N \omega_i x_{ij}^2\right)\tilde{\beta}_j, \lambda\alpha\right)}{f\cdot\frac{1}{N}\sum_{i=1}^N \omega_i x_{ij}^2 + \lambda(1-\alpha)}. \quad (f > 1) \tag{15}$$

(3) Set $\tilde{\beta}_j = \hat{\beta}_j^{\mathrm{B}}$.
(4) Repeat steps $2-3$ until convergence of $\hat{\beta}$.

Algorithm **3** computes that the solution to (13) [1]. Following the arguments in Section 2.2 we derive a coordinate majorization descent algorithm for solving (13). See Algorithm **4**.

In a logistic regression model we have a binary response variable $Y = \{0,\ 1\}$ and assume

$$\Pr(Y = 1|x) = \frac{1}{1 + \exp\left(-\beta_0 + x^\intercal \beta\right)} = p_i.$$

We consider the elastic net penalized maximum likelihood estimate

$$(\hat{\beta}_0, \hat{\beta}) = \underset{(\beta_0,\beta)\in\mathrm{R}^{p+1}}{\arg\min}\left[-\frac{1}{N}\sum_{i=1}^N \{y_i \log p_i + (1-y_i)\log(1-p_i)\} + P_{\lambda,\alpha}(\beta)\right]. \tag{16}$$

The un-penalized logistic regression is often solved by using the Newton-Raphson algorithm in many standard statistical packages. glmnet uses a similar strategy and it is so far the fastest algorithm for computing the elastic net penalized logistic regression with high dimensional data. Basically, glmnet uses coordinate descent within the iterative re-weighted least squares loop to solve the penalized logistic regression problems. Let $(\tilde{\beta}_0, \tilde{\beta})$ be the current estimate in the iterative re-weighted least squares. As in the usual logistic regression, we define the following quantities

$$\eta_i = \tilde{\beta}_0 + x_i^\intercal \tilde{\beta}, \quad \tilde{p}_i = \frac{1}{1 + \exp(-\tilde{\eta}_i)},$$
$$z_i(\tilde{\eta}_i) = \tilde{\eta}_i + \frac{y_i - \tilde{p}_i}{\tilde{p}_i(1-\tilde{p}_i)}, \quad \omega_i(\tilde{\eta}_i) = \tilde{p}_i(1-\tilde{p}_i).$$

Algorithm 5 Glmnet and Glmnet2 for penalized logistic regression

(1) Initialize $(\tilde{\beta}_0, \tilde{\beta})$, and set $\tilde{\eta}_i = \tilde{\beta}_0 + x_i^\intercal \tilde{\beta}$ for $i = 1, \ldots, N$.
(2) Compute $z_i = z_i(\tilde{\eta}_i)$ and $\omega_i = \omega_i(\tilde{\eta}_i)$.
(3) Glmnet calls Algorithm 3 and Glmnet 2 calls Algorithm 4 to solve

$$(\hat{\beta}_0, \hat{\beta}) = \underset{(\beta_0,\beta)\in\mathbb{R}^{p+1}}{\arg\min} \frac{1}{2N} \sum_{i=1}^{N} \omega_i (z_i - \beta_0 - x_i^\intercal \beta)^2 + P_{\lambda,\alpha}(\beta) .$$

(4) Set $\tilde{\beta} = \hat{\beta}, \tilde{\beta}_0 = \hat{\beta}_0$.
(5) Repeat steps $2 - 4$ until convergence of $\hat{\beta}$.

The Newton-Raphson algorithm finds the updated solution by solving

$$\min_{(\beta_0,\beta)\in\mathbb{R}^{p+1}} \left\{ \frac{1}{2N} \sum_{i=1}^{N} \omega_i(\tilde{\eta}_i)(z_i(\tilde{\eta}_i) - \beta_0 - x_i^\intercal \beta)^2 + P_{\lambda,\alpha}(\beta) \right\} . \tag{17}$$

Glmnet calls Algorithm 3 to solve (17). The complete glmnet algorithm for penalized logistic regression is given in Algorithm **5**. Our package glmnet2 is almost identical to glmnet except that we use Algorithm 4 to solve (17).

3. Numerical Experiments

Glmnet uses several tricks to boost its speed, including pathwise descent, warm start and active set convergence. For the sake of space we do not repeat the details of these tricks here. The readers are referred to [1] for the warn start trick, which deals with initial values for the iterative coordinate descent, and the active set trick. In the latest version of glmnet (version 1.7), glmnet further uses the strong rule trick [13]. In order for us to show that the timing difference between glmnet and glmnet2 is solely due to the extra factor f, we need to make sure that the two algorithms use the same implementation tricks. To do so, we took the core `Fortran` routines used in glmnet and added the extra f factor in those used for doing the soft-thresholding operation.

In glmnet version 1.7 the convergence criterion is $\max_j \left(\hat{\beta}_j^{\text{old}} - \hat{\beta}_j^{\text{new}}\right)^2 < \epsilon^2$. The same convergence criterion is used in glmnet2. In this section $\epsilon = 10^{-5}$. We compare the run times of glmnet and glmnet2. All timings were carried out on an Intel Core 2 Duo 2.4 GHz processor.

3.1. *Simulated data*

To fix idea, we use $f = 2$ in this subsection. We further explore the effect of f on timing in Section 3.3. Consider Friedman's model for timing comparison [1]. We simulated data with N observations and p predictors where each pair of predictors X_j and $X_{j'}$ have the same population correlation ρ, with ρ ranges from zero to 0.95. We tried $(N = 5000, p = 100)$ and $(N = 100, p = 5000)$. The response variable was generated by

$$Y = \sum_{j=1}^{p} X_j \beta_j + k \cdot N(0,1),$$

where $\beta_j = (-1)^j \exp(-(2j-1)/20)$ and k is set to make the signal-to-noise ratio equal 3. For logistic regression, we used the same simulation setup as above, except we define $p = 1/(1+\exp(-Y))$ and generate a two class response Y' is generated with $\Pr(\mathrm{Y}' = 0) = p \quad \Pr(\mathrm{Y}' = +1) = 1 - p$. For each data set we computed its elastic net solution paths with $\alpha = 1$ and $\alpha = 0.5$ for 100 λ values. In Table 1 We report the average run time of glmnet and glmnet2 over 10 independent runs.

3.2. *Real data*

We also compared glmnet and glmnet2 on some benchmark data sets. See Table 2. Colon [14] and Prostate [15] are the typical examples of the $p \gg N$ data. In the other three data sets, Wisconsin Breast Cancer Diagnostic (WBCD) data; Ionosphere data and Sonar data [16], the original dimension is less than the sample size. We expanded the predictor set by including the second order polynomials and pairwise interactions of the original predictors. Then the expanded dimension becomes much larger or at least similar to the sample size; see row 2 of Table 2. We fit the elastic net penalized logistic regression model on each data set and used 10-fold cross-validation to choose α; see row 3 of Table 2. Fix $\alpha = \alpha_{CV}$. We compared the running time of glmnet and glmnet2. The relative speed improvement is defined as $(t_{\text{glmnet}} - t_{\text{glmnet2}})/t_{\text{glmnet2}}$. We can see than glmnet2 is noticeably faster than glmnet, especially on the WBCD and Sonar.

3.3. *Exploring the factor size*

We have fixed $f = 2$ in the previous numerical examples. Now we further explore the effect of f on timing. In Figure 1 we plotted the run time (in log scale) against f for different correlation levels varying from 0 to 0.95. The dotted vertical reference line in each panel indicates the run time of glmnet. We see that when $\rho = 0$ increasing f only slows down the convergence, which is expected. However, when the correlation becomes stronger ($\rho \geq 0.2$), the curve starts to have a valley and using some $f > 1$ can reduce the computing time. From Figure 1 it seems that 2 is a good default value for f. We also see that when the correlation is very high such as $\rho = 0.8$ or higher, $f = 4$ or $f = 6$ can even work slightly better than $f = 2$. On the other hand, using $f = 4$ or $f = 6$ can have much bigger loss in speed when correlation is low compared to using $f = 2$. It would be also interesting to decide f's value based on the empirical correlations. We tested this idea and did not find this strategy works noticeably better than just using $f = 2$.

3.4. *Some explanation of the acceleration effect*

We have shown by numerical experiments that using $f > 1$ in the CMD could lead to faster convergence than the ordinary coordinate descent using $f = 1$, especially when the predictors are highly correlated. Now we attempt to provide some explanation to this acceleration effect. To gain some insight, we consider a simpler case where we use the CMD to solve the ordinary least squares prob-

	Correlation					
	0	0.1	0.2	0.5	0.8	0.95
	Least Squares $\alpha = 1$					
	$N = 5000$, $p = 100$					
glmnet	0.0719	0.0746	0.0787	0.0988	0.1641	0.3144
glmnet2	0.0752	0.0757	0.0764	0.0833	0.1094	0.1921
	$N = 100$, $p = 5000$					
glmnet	0.2222	0.2339	0.2979	0.4606	0.7919	1.9016
glmnet2	0.2533	0.2519	0.2886	0.3758	0.5450	1.0735
	Logistics Regression $\alpha = 1$					
	$N = 5000$, $p = 100$					
glmnet	1.4282	1.5591	1.9760	4.1573	8.9447	35.5548
glmnet2	1.8754	1.9002	2.0551	2.6598	5.2532	14.5804
	$N = 100$, $p = 5000$					
glmnet	0.2756	0.2734	0.2938	0.3940	0.8790	1.6118
glmnet2	0.2744	0.2734	0.2876	0.3302	0.5328	0.9422
	Correlation					
	0	0.1	0.2	0.5	0.8	0.95
	Least Squares $\alpha = 0.5$					
	$N = 5000$, $p = 100$					
glmnet	0.0718	0.0750	0.0802	0.1003	0.1704	0.5112
glmnet2	0.0750	0.0755	0.0770	0.0859	0.1145	0.2526
	$N = 100$, $p = 5000$					
glmnet	0.2107	0.2189	0.2356	0.3669	0.7765	2.1528
glmnet2	0.2225	0.2285	0.2414	0.2861	0.4876	1.3335
	Logistics Regression $\alpha = 0.5$					
	$N = 5000$, $p = 100$					
glmnet	1.5438	1.6540	2.0519	4.3477	11.9141	37.5434
glmnet2	2.0086	2.0039	2.1666	2.9319	6.1041	16.8284
	$N = 100$, $p = 5000$					
glmnet	0.3193	0.3573	0.4661	0.6217	1.1428	2.0621
glmnet2	0.3176	0.3253	0.3456	0.4042	0.5980	1.3126

Table 1. Timings (in seconds) for **glmnet** and **glmnet2** in the elastic net penalized ($\alpha = 1, 0.5$) regression and logistic regression. Total time for 100 λ values, averaged over 10 independent runs.

lem with p predictors, defined as $\hat{\beta} = \arg\min_{\beta \in \mathbb{R}^p} \frac{1}{2N} \left\| \mathbf{y} - \mathbf{x}\beta^T \right\|^2$. This model is a special point on the ℓ_1 penalized least squares solution path. Without loss of generality assume all predictors are standardized such that $\overline{x_j} = \frac{1}{N}\sum_{i=1}^N x_{ij} = 0$, $s_{x_j}^2 = \frac{1}{N}\sum_{i=1}^N (x_{ij} - \overline{x_j})^2 = 1$, for $j = 1, \cdots, p$. To simplify the analysis, we further assume that the pairwise sample correlation is a constant, i.e., $\frac{1}{N}\sum_{i=1}^N x_{ij}x_{ik} = \rho$ for $j, k \in \{1, \cdots, p\}$.

	Colon	Prostate	WBCD	Ionosphere	Sonar
N	62	102	569	351	208
p	2000	6033	495 (30)	560 (32)	1890 (60)
$\alpha_{\rm CV}$	0.6	0.5	0.6	0.4	0.4
Test Error	8.3%	5%	1.77%	2.86%	24.39%
glmnet	0.1166	0.3283	9.4039	0.5158	2.0828
glmnet2	0.0910	0.2938	4.9593	0.3667	1.0945
Improv. %	+28%	+11.7%	+89.6%	+40.6%	+90.3%

Table 2. Timings (in seconds) of **glmnet** and **glmnet2** for some real data, averaged over 10 runs.

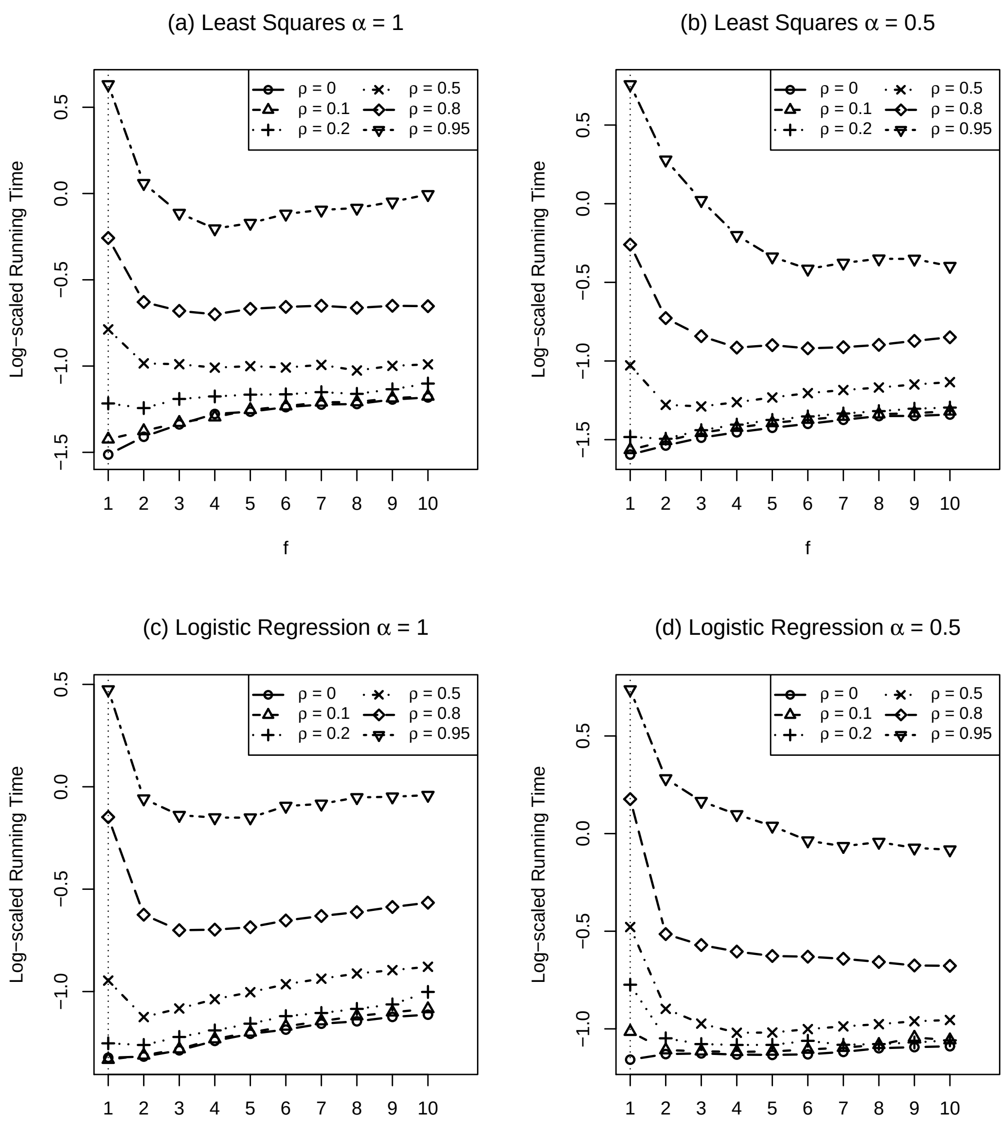

Figure 1. The running time of **glmnet2** for computing solution paths at 100 λs of the elastic net penalized regression and logistic regression with $\alpha = 1$ and $\alpha = 0.5$, averaged over 10 independent runs. The factor size f varies from 1 to 10. The data were generated from the simulation model in Section 3.1 with $N = 100$, $p = 5000$. Each curve corresponds to a different correlation level.

Given below is the CMD algorithm for the least squares regression problem:

(1) Initialize $\tilde{\beta} = \beta^{(0)}$.
(2) For $k = 1, 2, 3, \cdots$, iterate step 3 until convergence of $\tilde{\beta}$.
(3) For $j = 1, \cdots, p$, fix $\tilde{\beta} = (\beta_1^{(k)}, \cdots, \beta_{j-1}^{(k)}, \beta_j^{(k-1)}, \beta_{j+1}^{(k-1)}, \cdots, \beta_p^{(k-1)})$, we update the j-th coordinate of $\tilde{\beta}$,

$$\tilde{\beta}_j^{(k)} = \arg\min_{\beta_j} \frac{1}{2} f(\beta_j - \beta_j^{(k-1)})^2 - \frac{1}{N} x_j^\intercal \left(y - \mathbf{x}\tilde{\beta}^\intercal\right)(\beta_j - \beta_j^{(k-1)}) \tag{18}$$

$$= \tilde{\beta}(\underbrace{-\frac{\rho}{f}, \cdots, -\frac{\rho}{f}}_{j-1}, 1 - \frac{1}{f}, \underbrace{-\frac{\rho}{f}, \cdots, -\frac{\rho}{f}}_{p-j})^\intercal + \frac{x_j^\intercal y}{fN}. \tag{19}$$

We have used the equal correlation assumption to simplify (18) to get (19).

Note that we can rewrite (19) in step 3 as follows

$$\tilde{\beta}^{\text{new}} = \tilde{\beta}\mathbf{W}_j + \upsilon_j, \tag{20}$$

where

$$\tilde{\beta}^{\text{new}} = \left(\beta_1^{(k)}, \cdots, \beta_{j-1}^{(k)}, \beta_j^{(k)}, \beta_{j+1}^{(k-1)}, \cdots, \beta_p^{(k-1)}\right), \qquad \upsilon_j = (0, \cdots, \frac{x_j^\intercal y}{fN}, \cdots, 0).$$

$$\mathbf{W}_j = \mathbf{I}_{p\times p} + \left[\mathbf{0}_{p\times(j-1)}\ \mathbf{u}_j\ \mathbf{0}_{p\times(N-j)}\right], \qquad \mathbf{u}_j = (u_{kj})_{p\times 1} = \begin{cases} -\frac{1}{f} & k = j \\ -\frac{\rho}{f} & k \neq j \end{cases}.$$

Using (20), we find that after a complete cycle from $j = 1$ to $j = p$ we can write

$$\tilde{\beta}^{(k)} = \tilde{\beta}^{(k-1)}\mathbf{A} + \mu, \tag{21}$$

where

$$\mathbf{A} = \prod_{j=1}^{p} \mathbf{W}_j, \qquad \mu = \sum_{s=1}^{p-1}\left(\upsilon_s \prod_{j=s+1}^{p} \mathbf{W}_j\right) + \upsilon_p. \tag{22}$$

Then by a simple transformation $\tilde{\gamma}^{(k)} = \tilde{\beta}^{(k)} + \omega$ where $\omega = (\mathbf{I} - \mathbf{A})^{-1}\mu$ we can express (21) in terms of $\tilde{\gamma}^{(k)}$ and $\tilde{\gamma}^{(k-1)}$ as follows

$$\gamma^{(k)} = \mathbf{A}\gamma^{(k-1)}, \tag{23}$$

which means that

$$\gamma^{(k)} = \mathbf{A}^k\gamma^{(0)}, \tag{24}$$

and

$$\left\|\gamma^{(k)}\right\| = \left\|\mathbf{A}^k\gamma^{(0)}\right\| \leq \sqrt{\eta_{\max}\left((\mathbf{A}^k)^\intercal \mathbf{A}^k\right)}\left\|\gamma^{(0)}\right\|, \tag{25}$$

where $\eta_{\max}\left((\mathbf{A}^k)^\intercal \mathbf{A}^k\right)$ is the maximum eigenvalue of $\left(\mathbf{A}^k\right)^\intercal \mathbf{A}^k$.

From (25) one can see that the convergence rate of the CMD algorithm for the least squares problem is determined by $\eta_{\max}\left(\left(\mathbf{A}^k\right)^\top \mathbf{A}^k\right)$, which is affected by both f and ρ. Although we do not find an explicit expression of $\eta_{\max}\left(\left(\mathbf{A}^k\right)^\top \mathbf{A}^k\right)$, we can compute it numerical values easily. We did the calculation for $p = 10$ and Figure 2 displays the calculated $\eta_{\max}\left(\left(\mathbf{A}^k\right)^\top \mathbf{A}^k\right)$ as a function of $\log(k)$ for different combinations of (f, ρ). It is not surprising to see that as k (the number of iterations) increases, $\eta_{\max}\left(\left(\mathbf{A}^k\right)^\top \mathbf{A}^k\right)$ goes to zero, for all factors considered there. As shown in Figure 2 panel (a), when the correlation is low, $f = 1$ has the fastest convergence. However, when $\rho = 0.5$ as shown in Figure 2 panel (b), $f = 2$ starts to outperform $f = 1$. When the correlation is even higher like in Figure 2 panels (c) and (d), $f = 2, 3, 4, 5$ clearly dominates $f = 1$.

The above theoretical results are derived for the least squares problem. The analysis is not directly generalized to the more general ℓ_1 penalized least squares or logistic regression. However, the analysis does show us that in using the coordinate decent scheme to solve a multivariate optimization problem, taking the steepest descent in each coordinate direction is not necessarily the best strategy for achieving convergence in the multi-dimension space.

4. Conclusions

By using the MM principle, we have developed a generalized coordinate descent algorithm called CMD for solving the ℓ_1 penalized regression and logistic regression. The empirical examples suggest that the CMD algorithm can be substantially faster than the original CD algorithm used for the R package glmnet, as long as the correlations among predictors are not weak. The gain in speed is solely due to the use of MM principle within the coordinate decent loop. The R package glmnet2 and the functions used for the simulation in this paper are available at the following publicly accessible webpage `http://code.google.com/p/glmnet2`.

Acknowledgements

The authors thank the editor, an associate editor and referee for their helpful comments and suggestions. This work is supported in part by NSF Grant DMS-08-46068.

References

[1] J. Friedman, T. Hastie, and R. Tibshirani, *Regularization paths for generalized linear models via coordinate descent*, Journal of statistical software 33 (2010), p. 1.
[2] R. Tibshirani, *Regression shrinkage and selection via the lasso*, Journal of the Royal Statistical Society, Series B. (1996), pp. 267–288.
[3] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, *Least angle regression*, Annals of statistics 32 (2004), pp. 407–451.
[4] W. Fu, *Penalized regressions: the bridge versus the lasso*, Journal of Computational and Graphical Statistics 7 (1998), pp. 397–416.
[5] 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 (2004), pp. 1413–1457.
[6] J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani, *Pathwise coordinate optimization*, The Annals of Applied Statistics 1 (2007), pp. 302–332.
[7] T. Wu and K. Lange, *Coordinate descent algorithms for lasso penalized regression*, The Annals of Applied Statistics 2 (2008), pp. 224–244.
[8] H. Zou and T. Hastie, *Regularization and variable selection via the elastic net*, Journal of the Royal Statistical Society: Series B 67 (2005), pp. 301–320.
[9] K. Lange, D. Hunter, and I. Yang, *Optimization transfer using surrogate objective functions*, Journal of Computational and Graphical Statistics 9 (2000), pp. 1–20.

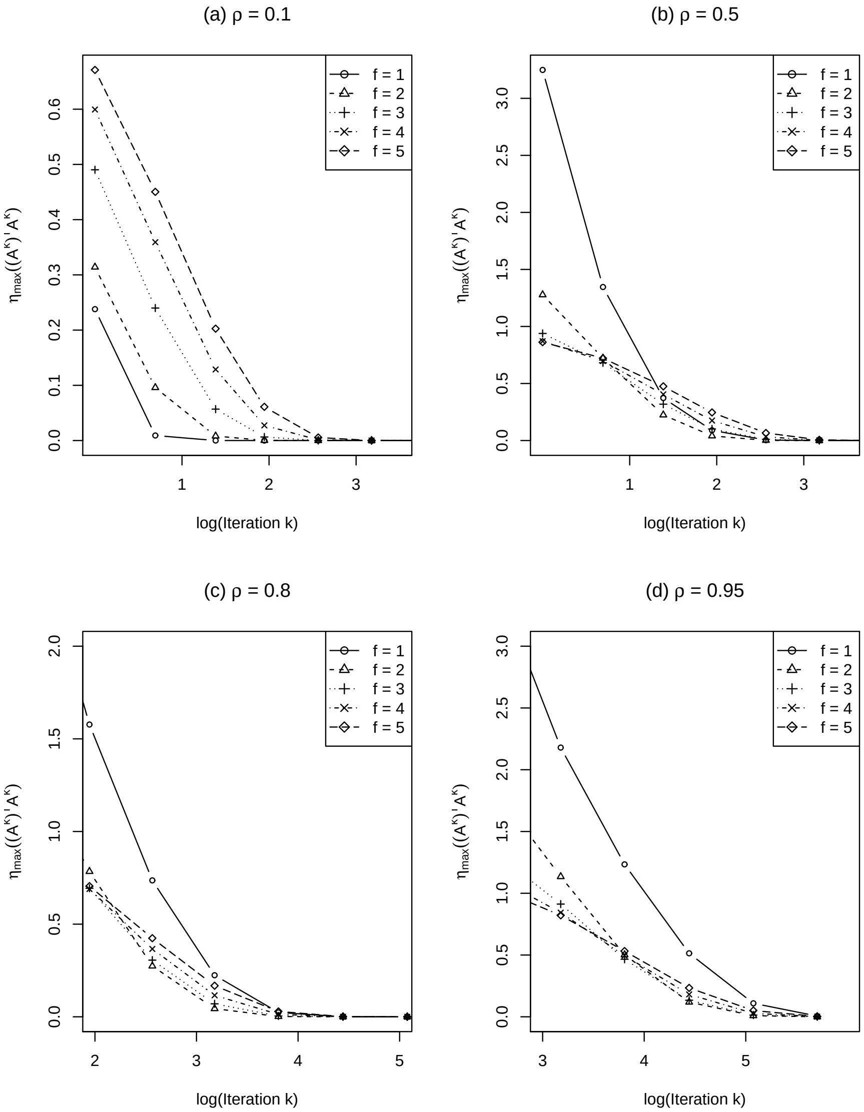

Figure 2. The CMD algorithm for the ordinary least squares problem with $p = 10$ predictors. $\eta_{\max}\left(\left(\mathbf{A}^k\right)^\intercal \mathbf{A}^k\right)$ (as defined in (25)) against the number of iterations (in logarithm scale) for 5 different factors ($f = 1, 2, 3, 4, 5$). Each panel corresponds to a different correlation level.

[10] D. Hunter and K. Lange, *A tutorial on MM algorithms*, The American Statistician 58 (2004), pp. 30–37.
[11] T. Wu and K. Lange, *The MM Alternative to EM*, Statistical Science 25 (2010), pp. 492–505.
[12] A. Dempster, N. Laird, and D. Rubin, *Maximum likelihood from incomplete data via the EM algorithm*, Journal of the Royal Statistical Society, Series B. 39 (1977), pp. 1–38.
[13] R. Tibshirani, J. Bien, J. Friedman, T. Hastie, N. Simon, J. Taylor, and R. Tibshirani, *Strong rules for discarding predictors in lasso-type problems*, Arxiv preprint arXiv:1011.2234 (2010).
[14] U. Alon, N. Barkai, D. Notterman, K. Gish, S. Ybarra, D. Mack, and A. Levine, *Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays*, Proceedings of the National Academy of Sciences 96 (1999), p. 6745.
[15] D. Singh et al., *Gene expression correlates of clinical prostate cancer behavior*, Cancer cell 1 (2002), pp. 203–209.
[16] A. Frank and A. Asuncion, UCI Machine Learning Repository,University of California, Irvine, School of Information and Computer Sciences, http://archive.ics.uci.edu/ml; .