Statistics for High-Dimensional Data Methods, Theory and Applications

Chapter 2: Lasso for linear models

Yewen Chen

chenyw68@mail2.sysu.edu.cn

July 27, 2020

Outline

1. Convex optimization with constraints

2. Lasso

   2.1 Coordinate descent algorithm

   2.2 Pathwise coordinate descent

   2.3 Least Angle Regression

3 Speed Trials

Convex optimization with constraints

Consider a general optimization problem with constraints:

$$\min f(\boldsymbol{\beta}) \text{ subject to } J_k(\boldsymbol{\beta}) \leq 0, k = 1, 2, \cdots, m \tag{1}$$

Since $f(\boldsymbol{\beta})$ and $J_k(\boldsymbol{\beta})$ are convex, there is a unique global minimum. Suppose first that each $J_k(\boldsymbol{\beta})$ is differentiable. Then necessary conditions for $\boldsymbol{\beta}^*$ to be the global solution are as follows: there exists $\lambda_k^* \geq 0$ such that

1) primal feasibility: $J_k(\boldsymbol{\beta}) \leq 0$.
2) complementary slackness: $\lambda_k^* J_k(\boldsymbol{\beta}) = 0, k = 1, 2, \cdots, m$.
3) stationarity: $0 \in \partial f(\boldsymbol{\beta}^*) + \sum_{k=1}^{m} \lambda_k^* \partial J_k(\boldsymbol{\beta}^*)$.

These conditions are known as the Karush-Kuhn-Tucker(KKT) optimality conditions.

Lasso

In the lasso, let $f(\boldsymbol{\beta}) = \frac{\|\boldsymbol{y}-\boldsymbol{x}\boldsymbol{\beta}\|_2^2}{n}, J(\boldsymbol{\beta}) = \|\boldsymbol{\beta}\|_1$, optimization problem will be

$$\hat{\beta}(\lambda) = \arg\min_{\beta} \left(\frac{\|\boldsymbol{y} - \boldsymbol{x}\boldsymbol{\beta}\|_2^2}{n} + \lambda\|\boldsymbol{\beta}\|_1 \right). \tag{2}$$

The lasso has two important benefits for variable selection:

1. The lasso objective function in Eqs. (2) is convex.
2. Estimates are continuous with respect to both λ and the data.

One way is to transform non-differentiable into differentiable

L_1 penalty function in Eqs. (2) is not differentiable if any of the β_j are equal to 0. One way to finesse this is to write each β_j in terms of its positive and negative parts:

$$|\beta_j| = \theta_j^+ + \theta_j^-.$$

where $\theta_j^+ = \max\{\beta_j, 0\}$ and $\theta_j^- = \max\{-\beta_j, 0\}$ and we reparametrize the problem in terms of the θ_j^+ and θ_j^-, $j = 1, 2, \cdots, p$.

However this approach doubles the number of parameters and increases the number of constraints.

Coordinate descent and separability

In Eqs. (1), let $g(\beta_1, \cdots, \beta_p) = f(\beta_1, \cdots, \beta_p) + \sum_{k=1}^{m} J_k(\beta_k)$, A coordinate descent algorithm successively minimizes the function over each parameter, holding the others fixed:

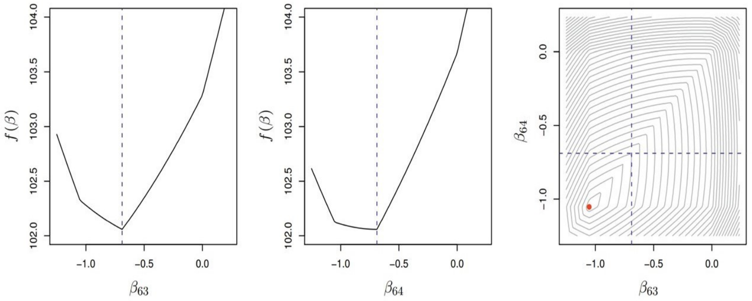
$$\hat{\beta}_j \leftarrow \arg\min \left\{ g(\hat{\beta}_1, \cdots, \hat{\beta}_{j-1}, \beta_j, \hat{\beta}_{j+1}, \cdots, \hat{\beta}_p) \right\} \tag{3}$$

Cycle around by Eqs. (3) till coefficients stabilize.

Tseng (1988, 2001) shows that coordinate descent converges to the minimizer of g. The key to this result is the **separability** of the penalty function $\sum_{k=1}^{m} J_k(\beta_k)$, a sum of functions of each individual parameter. This result implies that the coordinate-wise algorithms for the lasso converge to their optimal solutions.

An example of failure for coordinate descent

Counterexample: $\sum_j^p |\beta_j - \beta_{j-1}|$.



Failure of coordinate-wise descent in a fused lasso problem with 100 parameters. The optimal values for two of the parameters, β_{63} and β_{64}, are both -1.05, as shown by the dot in the right panel. The left and middle panels show slices of the objective function f as a function of β_{63} and β_{64}, with the other parameters set to the global minimizers.

Coordinate descent algorithm by means of the subgradient method

If in addition to centering, the variables are standardized to have unit variance, coordinate descent algorithm is as follows:

Algorithm 1: Coordinate descent for the lasso

```
Input: Standardize the predictors to have mean zero and unit $\ell_2$ norm. Start with the
       residual $\boldsymbol{r}_0 = \mathbf{y}, \boldsymbol{\beta}^0 = (\beta_1, \beta_2, \ldots, \beta_p) = \mathbf{0}$.
1 Cycle over $j = 1, 2, \cdots, p, 1, 2, \cdots$ till convergence:
    1) Compute the partial residuals $r_{ij} = y_i - \sum_{k \neq j}^{p} x_{i,k}\beta_k$.
    2) Compute the simple least squares coefficient of these residuals on $j$th predictor:
       $\beta_j^* = \frac{1}{n}\sum_{i=1}^{n} x_{ij} r_{ij}$
    3) Update $\beta_j$ by soft-thresholding:
       $$\beta_j = \mathrm{sign}(\beta_j^*)\left(|\beta_j^*| - \lambda\right).$$
```

Why is coordinate descent so fast?

There are a number of tricks and strategies that we use exploit the structure of the problem.

Naive Updates: $\beta_j^* = \frac{1}{n}\sum_{i=1}^n x_{ij} r_{ij} = \frac{1}{n}\sum_{i=1}^n x_{ij} r_i + \beta_j$, where r_i is current model residual for observation i; this requires making $O(n)$ calculations.

Covariance Updates: $n\beta_j^* = \boldsymbol{x}_j' \boldsymbol{r} = \boldsymbol{x}_j' \boldsymbol{y} - \sum_{k:|\beta_k|>0}^p \boldsymbol{x}_j' \boldsymbol{x}_k \beta_k$ Cross-covariance terms are computed once for active variables and s-tored (each step does not require making $O(n)$ calculations, so helps a lot when $n \gg p$). Computational cost per iteration: $O(np + sp)$, where s denotes the number of nonzero coefficients.

Why is coordinate descent so fast?

Sparse Updates: If design $\boldsymbol{X}$ is sparse (many zeros), inner products can be computed efficiently.

Active Set Convergence: After a cycle through p variables, we can restrict further iterations to the active set till convergence $+$ one more cycle through p to check if active set has changed. Helps when $n \gg p$.

Warm Starts

Typically, one wants a sequence of lasso solutions, say for a decreasing sequence of values $\{\lambda_l, l = 0, 1, \cdots, L\}$. It is easy to see that the largest value that we need consider is:

$$\lambda_0 = \frac{2}{n} \max_{0 \leq j \leq p} |\boldsymbol{x}_j' \boldsymbol{y}|$$

by condition 3) of KKT, since any value larger would yield an empty model.

Warm Starts: We fits a sequence of models from λ_0 down to $\lambda_L = \varepsilon \lambda_0$. λ_0 is smallest value of λ for which all coefficients are zero. Solutions don't change much from one λ to the next. Convergence is often faster for entire sequence than for single solution at small value of λ.

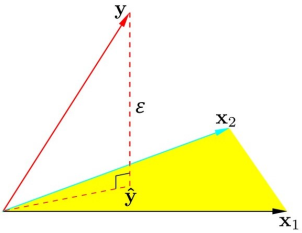
Pathwise coordinate descent for lasso

Start with a large value of λ, eg. λ_0. Run Algorithm 1 until convergence.

1. Outer loop (pathwise strategy, eg, for $\lambda_0 > \lambda_1 > \cdots > \lambda_L$): Decrease λ using previous solution as a warm start.
2. Inner loop (active set strategy):
   2.1 Perform one coordinate cycle by Algorithm 1, and record active set $\mathcal{A}$ of coefficients that are nonzero.
   2.2 Cycle over coefficients in $\mathcal{A}$ until convergence.
   2.3 Check KKT conditions over all coefficients; if not all satisfied, add offending coefficients to $\mathcal{A}$, go back one step.

OLS

We all known that $\boldsymbol{X}'\hat{\varepsilon} = \mathbf{0}$ under least squares, where $\hat{\varepsilon} = \left(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}\right)$, which means that correlation between $X_j, j = 1, \cdots, p$, and residual vector $\hat{\varepsilon}$ has disappeared through least squares fitting.



The geometric interpretation of least squares regression with two predictors.

Now, let $\boldsymbol{\mu} = \boldsymbol{X}\boldsymbol{\beta}$, some variables selection method will be obtained by starting with $\hat{\boldsymbol{\mu}} = \mathbf{0}$.

A comparison of some variable selection methods

If $\hat{\boldsymbol{\mu}}$ is the current estimate, let $c(\hat{\boldsymbol{\mu}})$ be the vector of current correlations,

$$\hat{c} = c(\hat{\boldsymbol{\mu}}) = \boldsymbol{X}'\left(\boldsymbol{y} - \hat{\boldsymbol{\mu}}\right),$$

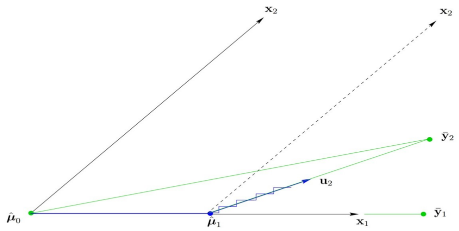
Then the next step of the algorithm is taken in the direction of the greatest current correlation:

$$j = \arg\max\{\hat{c}_j\} \text{ and } \hat{\boldsymbol{\mu}} = \hat{\boldsymbol{\mu}} + \mathsf{sign}(\hat{c}_j)\varepsilon x_j$$

with ε some constant.

Here different choices of ε correspond to different approaches. For example, the 'big' choice $\varepsilon = |\hat{c}_j|$ leads to the classic Forward selection technique, some very small choices results in Forward stagewise algorithm, while Least Angle Regression (LARS, see Efron, etc., 2003) uses an intermediate value.

Least Angle Regression



The LARS algorithm in the case of m $=2$ covariates.

LARS delivers the entire solution path as a function of the regularization parameter λ, the idea behind this algorithm is to

(1) Project the residuals onto the active variables

(2) Calculate how far we can proceed in that direction before another variable reaches the necessary level of correlation with the residuals, then adding it to the set of active variables and repeating (1) and (2), and so on.

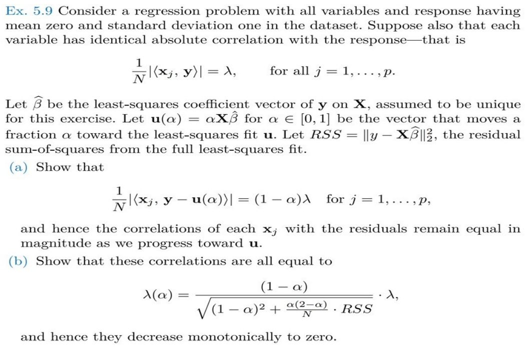
Least Angle Regression for lasso

Algorithm 2: Least Angle Regression

```
Input: Standardize the predictors to have mean zero and unit $\ell_2$ norm. Start with the
       residual $\boldsymbol{r}_0 = \mathbf{y}, \beta^0 = (\beta_1, \beta_2, \ldots, \beta_p) = \mathbf{0}$.
  1) Find the predictor $\mathbf{x}_j$ most correlated with $\boldsymbol{r}_0$; i.e., with largest value for
     $|\langle \mathbf{x}_j, \boldsymbol{r}_0 \rangle|$. Call this value $\lambda_0$, define the active set $\mathcal{A} = \{j\}$, and $\mathbf{X}_{\mathcal{A}}$, the
     matrix consisting of this single variable.
  2) for $k = 1, \ldots, \min(n-1, p)$ do
       (a) Define the least-squares direction $\delta = \frac{1}{\lambda_{k-1}} \left(\mathbf{X}_{\mathcal{A}}^T \mathbf{X}_{\mathcal{A}}\right)^{-1} \mathbf{X}_{\mathcal{A}}^T \boldsymbol{r}_{k-1}$, and
       define the $p$ -vector $\Delta$ such that $\Delta_{\mathcal{A}} = \delta$, and the remaining elements are zero.
       (b) Move the coefficients $\beta$ from $\beta^{k-1}$ in the direction $\Delta$ toward their
       leastsquares solution on $\mathbf{X}_{\mathcal{A}} : \beta(\lambda) = \beta^{k-1} + (\lambda_{k-1} - \lambda)\Delta$, for $0 < \lambda \leq \lambda_{k-1}$,
       keeping track of the evolving residuals
            $$\boldsymbol{r}(\lambda) = \mathbf{y} - \mathbf{X}\beta(\lambda) = \boldsymbol{r}_{k-1} - (\lambda_{k-1} - \lambda)\mathbf{X}_{\mathcal{A}}\Delta_{\mathcal{A}} \quad (4)$$
       (c) Keeping track of $|\langle \mathbf{x}_\ell, \boldsymbol{r}(\lambda) \rangle|$ for $\ell \notin \mathcal{A}$, identify the largest value of $\lambda$ at
       which a variable "catches up" with the active set; if the variable has index $j$, that
       means $|\langle \mathbf{x}_j, \boldsymbol{r}(\lambda) \rangle| = \lambda$. This defines the next "knot" $\lambda_k$.
       (d) Set $\mathcal{A} = \mathcal{A} \cup j, \beta^k = \beta(\lambda_k) = \beta^{k-1} + (\lambda_{k-1} - \lambda_k)\Delta$, and $\boldsymbol{r}_k = \mathbf{y} - \mathbf{X}\beta^k$.
  3) Return the sequence $\left\{\lambda_k, \beta^k\right\}_0^K, K = \min(N-1, p)$.
```

Some explanation of Least Angle Regression

* In step 2)(b), it is easy to check that $|\boldsymbol{X}'_{\mathcal{A}}, \boldsymbol{r}(\lambda)| = \lambda\mathbf{1}$, that is, the correlations remain tied along this path, and decrease to zero with λ. In fact, $\boldsymbol{\beta}(0) = \boldsymbol{\beta}^{k-1} + \boldsymbol{\lambda}_{k-1}\Delta$ is the least-squares coefficient vector corresponding to the subset $\mathcal{A}$.



Ex. 5.9 Consider a regression problem with all variables and response having mean zero and standard deviation one in the dataset. Suppose also that each variable has identical absolute correlation with the response—that is

$$\frac{1}{N}|\langle \mathbf{x}_j, \mathbf{y}\rangle| = \lambda, \qquad \text{for all } j = 1, \ldots, p.$$

Let $\widehat{\beta}$ be the least-squares coefficient vector of $\mathbf{y}$ on $\mathbf{X}$, assumed to be unique for this exercise. Let $\mathbf{u}(\alpha) = \alpha\mathbf{X}\hat{\beta}$ for $\alpha \in [0,1]$ be the vector that moves a fraction α toward the least-squares fit $\mathbf{u}$. Let $RSS = \|y - \mathbf{X}\widehat{\beta}\|_2^2$, the residual sum-of-squares from the full least-squares fit.

(a) Show that

$$\frac{1}{N}|\langle \mathbf{x}_j, \mathbf{y} - \mathbf{u}(\alpha)\rangle| = (1-\alpha)\lambda \quad \text{for } j = 1, \ldots, p,$$

and hence the correlations of each $\mathbf{x}_j$ with the residuals remain equal in magnitude as we progress toward $\mathbf{u}$.

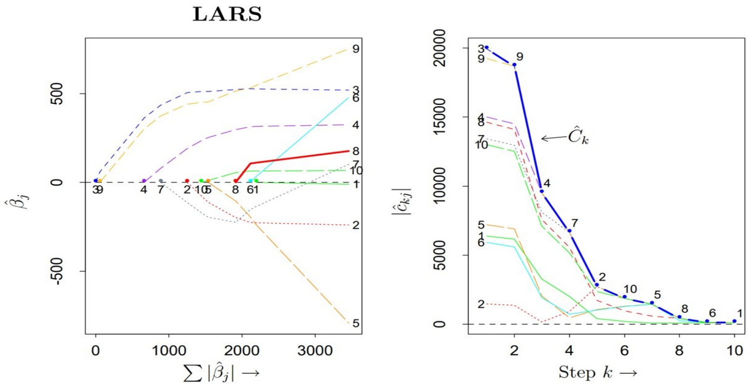
(b) Show that these correlations are all equal to

$$\lambda(\alpha) = \frac{(1-\alpha)}{\sqrt{(1-\alpha)^2 + \frac{\alpha(2-\alpha)}{N} \cdot RSS}} \cdot \lambda,$$

and hence they decrease monotonically to zero.

A example interpretation for the LARS algorithm keeps the correlations tied and monotonically decreasing.

Some explanation of Least Angle Regression



LARS analysis of the diabetes study. Left: estimates of regression coefficients $\beta_j, j = 1, \cdots, 10$. Right: Absolute current correlations as function of LARS step;variables enter active set in order $3, 9, 4, 7, \cdots, 1$; heavy curve shows maximum current correlation C_k declining with k

Some explanation of Least Angle Regression

* Consider setp 2) (c) Define $c_l = \boldsymbol{x}_l' \boldsymbol{r}_{k-1}$ and $A_l = \boldsymbol{x}_l' \boldsymbol{X}_{\mathcal{A}} \delta, l \notin \mathcal{A}$. Define

$$a_l = \mathsf{min}_+ \left\{ \frac{\lambda_{k-1} - c_l}{1 - A_l}, \frac{\lambda_{k-1} + c_l}{1 + A_l} \right\}$$

where $\min_+$ only considers positive entries. Show that the variable to enter at step k has index $j = \arg\min_{l \notin \mathcal{A}} \{a_l\}$, with value $\lambda_k = \lambda_{k-1} - a_j$.

Speed Trials

Some R packages of related algorithms:

lars: LARS algorithm.

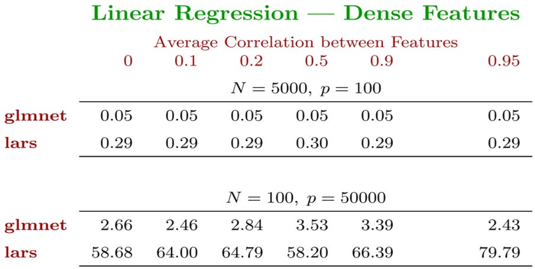
glmnet: Fortran based R package using coordinate descent.

l1logreg: Lasso-logistic regression package by Koh, Kim and Boyd, using state-of-art interior point methods for convex optimization.

BBR/BMR: Bayesian binomial/multinomial regression package by Genkin, Lewis and Madigan. Also uses coordinate descent to compute posterior mode with Laplace prior-the lasso fit.

The following simulation results from Trevor Hastie(2009).

Speed Trials



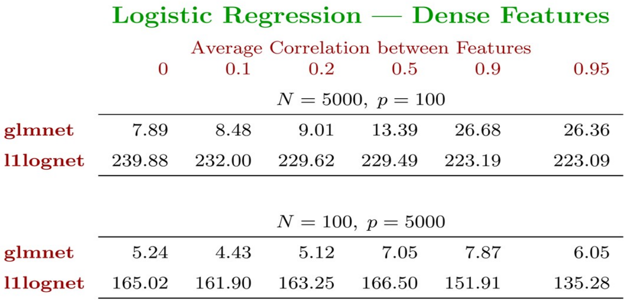
Linear Regression — Dense Features

Average Correlation between Features

	0	0.1	0.2	0.5	0.9	0.95
			$N = 5000,\ p = 100$			
glmnet	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
			$N = 100,\ p = 50000$			
glmnet	2.66	2.46	2.84	3.53	3.39	2.43
lars	58.68	64.00	64.79	58.20	66.39	79.79

Timings (secs) for glmnet and lars algorithms for linear regression with lasso penalty. Total time for 100 λ values, averaged over 3 runs.

Speed Trials

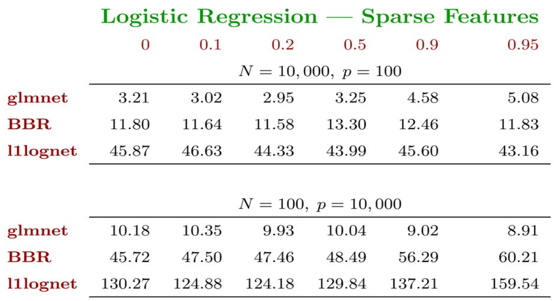


Logistic Regression — Dense Features

	Average Correlation between Features					
	0	0.1	0.2	0.5	0.9	0.95
			$N = 5000,\ p = 100$			
glmnet	7.89	8.48	9.01	13.39	26.68	26.36
l1lognet	239.88	232.00	229.62	229.49	223.19	223.09
			$N = 100,\ p = 5000$			
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

Timings (seconds) for logistic models with lasso penalty. Total time for tenfold cross-validation over a grid of 100 λ values.

Speed Trials



Logistic Regression — Sparse Features

	0	0.1	0.2	0.5	0.9	0.95
			$N = 10,000,\ p = 100$			
glmnet	3.21	3.02	2.95	3.25	4.58	5.08
BBR	11.80	11.64	11.58	13.30	12.46	11.83
l1lognet	45.87	46.63	44.33	43.99	45.60	43.16
			$N = 100,\ p = 10,000$			
glmnet	10.18	10.35	9.93	10.04	9.02	8.91
BBR	45.72	47.50	47.46	48.49	56.29	60.21
l1lognet	130.27	124.88	124.18	129.84	137.21	159.54

Timings (seconds) for logistic model with lasso penalty and sparse features (95% zeros in X). Total time for ten-fold cross-validation over a grid of 100 λ values.

Remarks

For coordinate descent algorithm, one full iteration can be completed at a computational cost of $O(2np)$ operations, thus, coordinate descent is linear in both n and p, scaling up to high dimensions even better than LARS, although it is worth noting that coordinate descent requires an unknown number of iterations, whereas LARS terminates in a known number of steps.

Geoff Gordon and Ryan Tibshirani point out that coordinate descent is competitve with fastest algorithms for L_1-norm penalized minimization problems.