Computation & optimization for Lasso - part 2

Luyang Han & Janosch Ott

ETH Zürich

22 October 2018

Computation & optimization for Lasso - part 2

Luyang Han & Janosch Ott

ETH Zürich

22 October 2018

Overview

Overview

1. Coordinate Descent
2. Least Angle Regression
3. Comparison of Optimization Methods
4. Recall: Duality
5. ADMM
6. Screening Rules
7. Minor-Max Algorithms
8. Alternating Minimizations

Coordinate Descent Algorithm

What is Coordinate Descent (CD) Algorithm?

$$\beta_k^{t+1} = \underset{\beta_k}{\operatorname{argmin}}\ f(\beta_1^t\ , \beta_2^t\ , \ldots \beta_k\ , \beta_{k+1}^t\ , \ldots \beta_p^t) \tag{1}$$

and $\beta_j^{t+1} = \ \beta_j^t$ for $j \neq k$

- An iterative algorithm that updates from β^t to β^{t+1} by choosing a single coordinate, and minimizing over this coordinate.

Coordinate Descent Algorithm

What is Coordinate Descent (CD) Algorithm?

$$\beta_k^{t+1} = \underset{\beta_k}{\operatorname{argmin}}\ f(\beta_1^t\ , \beta_2^t\ , \ldots \beta_k\ , \beta_{k+1}^t\ , \ldots \beta_p^t) \tag{1}$$

and $\beta_j^{t+1} = \ \beta_j^t$ for $j \neq k$

- An iterative algorithm that updates from β^t to β^{t+1} by choosing a single coordinate, and minimizing over this coordinate.

Separability Condition

Motivation

Does CD procedure converge to the global minimum of a convex function?

Separability Condition

Motivation

Does CD procedure converge to the global minimum of a convex function?

restrictive regarding its application to Lasso, regularizers leads to optimization problems that need not be differentiable.

Separability Condition

Motivation

Does CD procedure converge to the global minimum of a convex function?

- **Sufficient Condition:** the function is continuously differentiable and strictly convex in each coordinate.

$\Rightarrow$ restrictive

Separability Condition
Motivation
Does CD procedure converge to the global minimum of a convex function?
- Sufficient Condition: the function is continuously differentiable and strictly convex in each coordinate.
⇒ restrictive

restrictive regarding its application to Lasso, regularizers leads to optimization problems that need not be differentiable.

Separability Condition

Suppose the cost function f has the additive decomposition:

$$f(\beta_1, ..., \beta_p) = g(\beta_1, ..., \beta_p) + \sum_{j=1}^{p} h_j(\beta_j) \tag{2}$$

where $g : \mathbb{R}^p \to \mathbb{R}$ is differentiable and convex, and the univariate functions $h_j : \mathbb{R} \to \mathbb{R}$ is convex.

- Lasso: $g(\beta) = \frac{1}{2N}\|\mathbf{y} - \mathbf{X}\beta\|_2^2$ and $h_j(\beta_j) = \lambda|\beta_j|$ satisfies the condition

Separability Condition

Suppose the cost function f has the additive decomposition:

$$f(\beta_1, ..., \beta_p) = g(\beta_1, ..., \beta_p) + \sum_{j=1}^{p} h_j(\beta_j) \tag{2}$$

where $g : \mathbb{R}^p \to \mathbb{R}$ is differentiable and convex, and the univariate functions $h_j : \mathbb{R} \to \mathbb{R}$ is convex.

- Lasso: $g(\beta) = \frac{1}{2N}\|\mathbf{y} - \mathbf{X}\beta\|_2^2$ and $h_j(\beta_j) = \lambda|\beta_j|$ satisfies the condition

Separability Condition: Example

An Example of failure of Coordinate Descent

$$\underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \frac{1}{2N}\|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=2}^{p} |\beta_j - \beta_{j-1}|$$

- $h(\beta)$ is not separable
- Fused Lasso: coordinate descent procedure is not guaranteed to find the global minimum

Separability Condition: Example

An Example of failure of Coordinate Descent

$$\underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \frac{1}{2N}\|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=2}^{p} |\beta_j - \beta_{j-1}|$$

- $h(\beta)$ is not separable
- Fused Lasso: coordinate descent procedure is not guaranteed to find the global minimum

Separability Condition: Example

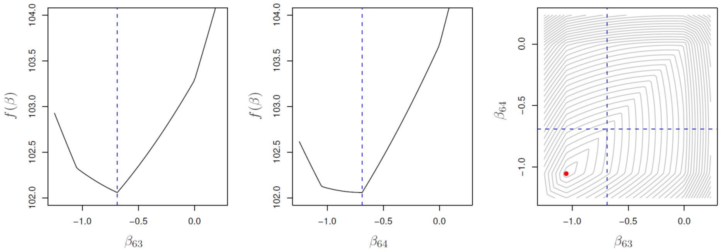
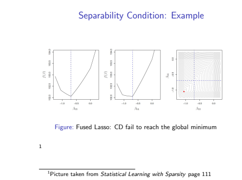


Figure: Fused Lasso: CD fail to reach the global minimum

1

[1] Picture taken from *Statistical Learning with Sparsity* page 111



Lasso & Coordinate Descent

Optimality Condition:

$-\frac{1}{2N}\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{k=1}^{p} x_{ik}\beta_k)x_{ij} + \lambda s_j = 0$

where $s_j \in sign(\beta_j)$ for $j = 1, 2, ...p$

- Define the **partial residual**: $r_i^{(j)} = y_i - \sum_{k\neq j} x_{ik}\hat{\beta}_k$
- Then the solution for $\hat{\beta}_j$ satisfies:

$$\hat{\beta}_j = \frac{S_\lambda(\frac{1}{N}\sum_{i=1}^{N} r_i^{(j)} x_{ij})}{\frac{1}{N}\sum_{i=1}^{N} x_{ij}^2}$$

where $S_\lambda(\theta) = sign(\theta)(|\theta| - \lambda)_+$

Lasso & Coordinate Descent

Optimality Condition:

$-\frac{1}{2N}\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{k=1}^{p} x_{ik}\beta_k)x_{ij} + \lambda s_j = 0$

where $s_j \in sign(\beta_j)$ for $j = 1, 2, ...p$

- Define the **partial residual**: $r_i^{(j)} = y_i - \sum_{k\neq j} x_{ik}\hat{\beta}_k$
- Then the solution for $\hat{\beta}_j$ satisfies:

$\hat{\beta}_j = \frac{S_\lambda(\frac{1}{N}\sum_{i=1}^{N} r_i^{(j)} x_{ij})}{\frac{1}{N}\sum_{i=1}^{N} x_{ij}^2}$

where $S_\lambda(\theta) = sign(\theta)(|\theta| - \lambda)_+$

Lasso & Coordinate Descent

- Illustration of Coordinate Descent in R

Strategies to make the operation efficient:

Naive Updating

$r_i^{(j)} = y_i - \sum_{k \neq j} x_{ik} \hat{\beta}_k = r_i + x_{ij} \hat{\beta}_j$

$\frac{1}{N} \sum_{i=1}^{N} x_{ij} r_i^{(j)} = \frac{1}{N} \sum_{i=1}^{N} x_{ij} r_i + \hat{\beta}_j$

Covariance Updating

$\sum_{i=1}^{N} x_{ij} r_i = \langle x_j, y \rangle - \sum_{k||\beta_{\hat{k}}|>0} \langle x_j, x_k \rangle \beta_{\hat{k}}$

Warm Starts: For a decreasing sequence of values $\{\lambda_0^L\}$, $\hat{\beta}(\lambda_l)$ is typically a very good warm start for the solution $\hat{\beta}(\lambda_{l+1})$. We set $\lambda_0 = \frac{1}{N} max|\langle x_j, y \rangle|$ and $\lambda_L \approx 0$

Lasso & Coordinate Descent

- Illustration of Coordinate Descent in R

Strategies to make the operation efficient:

Naive Updating

$r_i^{(j)} = y_i - \sum_{k \neq j} x_{ik} \hat{\beta}_k = r_i + x_{ij} \hat{\beta}_j$

$\frac{1}{N} \sum_{i=1}^{N} x_{ij} r_i^{(j)} = \frac{1}{N} \sum_{i=1}^{N} x_{ij} r_i + \hat{\beta}_j$

Covariance Updating

$\sum_{i=1}^{N} x_{ij} r_i = \langle x_j, y \rangle - \sum_{k||\beta_{\hat{k}}|>0} \langle x_j, x_k \rangle \beta_{\hat{k}}$

Warm Starts: For a decreasing sequence of values $\{\lambda_0^L\}$, $\hat{\beta}(\lambda_l)$ is typically a very good warm start for the solution $\hat{\beta}(\lambda_{l+1})$. We set $\lambda_0 = \frac{1}{N} max|\langle x_j, y \rangle|$ and $\lambda_L \approx 0$

Covariance updating: In this approach, we compute inner products of each feature with y initially,and then each time a new feature xk enters the model for the first time, wecompute and store its inner product with all the rest of the features, requiringO(Np) operations. We also store the p gradient components. If one of the coefficients currently in the model changes, we can update each gradient in O(p) operations. Hence with k nonzero terms in the model, a complete cycle costs O(pk) operations if no new variables become nonzero, and costs O(Np) for each new variable entered. Importantly, each step does not require making. O(N) calculations; **Warm Starts**: sequence of lambda values; double number of L, would not double the computational time; fewer iteration for each lambda.

Lasso & Coordinate Descent

Active-set Convergence: Define the active set A and iterate the algorithm using only variables in A.

Strong-set Convergence: Define the strong set S and iterate the algorithm using only variables in S.

Sparsity: Sparsity of the design matrix X makes the operation of inner product efficient.

Details in page 113 and page 114.

Lasso & Coordinate Descent

Active-set Convergence: Define the active set A and iterate the algorithm using only variables in A.

Strong-set Convergence: Define the strong set S and iterate the algorithm using only variables in S.

Sparsity: Sparsity of the design matrix X makes the operation of inner product efficient.

Details in page 113 and page 114.

Convergence criterion; covered in the following section of screening rule covered by JanoschSparsity:sparsity matrices can be stored efficiently in sparse-column format, where we store only the nonzero entries and the coordinates where they occur. Now when we compute inner products, we sum only over the nonzero entries.

Elastic Net & Coordinate Descent

$$minimize_{\beta_0,\beta_p} \frac{1}{2}\sum_{i=1}^{N}(y_i-\beta_0-x_i^T\beta)^2+\lambda\left[\frac{1}{2}(1-\alpha)||\beta||_2^2+\alpha||\beta||\right]_1$$

- Combination of L1 and L2 penaLty
- Satisfy the separability condition
- The solution satisfies:

$$\hat{\beta}_j = \frac{S_{\alpha\lambda}(\frac{1}{N}\sum_{i=1}^{N} r_i^{(j)} x_{ij})}{\frac{1}{N}\sum_{i=1}^{N} x_{ij}^2+(1-\alpha)\lambda}$$

Elastic Net & Coordinate Descent

$$minimize_{\beta_0,\beta_p} \frac{1}{2}\sum_{i=1}^{N}(y_i-\beta_0-x_i^T\beta)^2+\lambda\left[\frac{1}{2}(1-\alpha)||\beta||_2^2+\alpha||\beta||\right]_1$$

- Combination of L1 and L2 penaLty
- Satisfy the separability condition
- The solution satisfies:

$$\hat{\beta}_j = \frac{S_{\alpha\lambda}(\frac{1}{N}\sum_{i=1}^{N} r_i^{(j)} x_{ij})}{\frac{1}{N}\sum_{i=1}^{N} x_{ij}^2+(1-\alpha)\lambda}$$

Logistic Regression & Coordinate Descent

Background

- **Class Label G**: Take values 1 and -1

  Denote $p(x_i; \beta_0, \beta) = Pr(G = 1|x_i)$

  Define log odds: $log \frac{Pr(G=-1|x)}{Pr(G=1|x)} = \beta_0 + x^T \beta$

Logistic Regression & Coordinate Descent

Background

- Class Label G: Take values 1 and -1
Denote $p(x_i; \beta_0, \beta) = Pr(G = 1|x_i)$
Define log odds: $log \frac{Pr(G=-1|x)}{Pr(G=1|x)} = \beta_0 + x^T \beta$

Logistic Regression & Coordinate Descent

Background

- **Class Label G**: Take values 1 and -1

  Denote $p(x_i; \beta_0, \beta) = Pr(G = 1|x_i)$

  Define log odds: $log\frac{Pr(G=-1|x)}{Pr(G=1|x)} = \beta_0 + x^T\beta$

- **Maximize penalized log-likelihood:**

  $\frac{1}{N}\sum_{i=1}^{N}\{I(g_i = 1) \cdot logp(x_i; \beta_0, \beta) + I(g_i = -1) \cdot log(1 - p(x_i; \beta_0, \beta))\} - \lambda||\beta||_1$

  Denote $y_i = I(g_i = -1)$

Explicit form of log likelihood (without penalty):
$l(\beta_0, \beta) = \frac{1}{N}\sum_{i=1}^{N}\left[y_i \cdot (\beta_0 + x_i^T\beta) - log(1 + e^{\beta_0 + x_i^T\beta})\right]$

Logistic Regression & Coordinate Descent

Background

- Class Label G: Take values 1 and -1
  Denote $p(x_i; \beta_0, \beta) = Pr(G = 1|x_i)$
  Define log odds: $log\frac{Pr(G=-1|x)}{Pr(G=1|x)} = \beta_0 + x^T\beta$
- Maximize penalized log-likelihood:
  $\frac{1}{N}\sum_{i=1}^{N}\{I(g_i = 1) \cdot logp(x_i; \beta_0, \beta) + I(g_i = -1) \cdot log(1 - p(x_i; \beta_0, \beta))\} - \lambda||\beta||_1$
  Denote $y_i = I(g_i = -1)$

Explicit form of log likelihood (without penalty):
$l(\beta_0, \beta) = \frac{1}{N}\sum_{i=1}^{N}\left[y_i \cdot (\beta_0 + x_i^T\beta) - log(1 + e^{\beta_0 + x_i^T\beta})\right]$

Logistic Regression & Coordinate Descent

Background

- Form a **quadratic objective function** using Taylor expansion about current estimates $(\tilde{\beta}_0, \tilde{\beta})$: Idea of Newton method, Iterated Weighted Least Square problem

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

- Use Coordinate Descent to solve the problem

  $minimize_{(\beta_0,\beta)}\{l_Q\{\beta_0, \beta) + \lambda||\beta||_1\}$

Logistic Regression & Coordinate Descent

Background

- Form a quadratic objective function using Taylor expansion about current estimates $(\tilde{\beta}_0, \tilde{\beta})$: Idea of Newton method, Iterated Weighted Least Square problem
  $l_Q(\beta_0, \beta) = -\frac{1}{2N}\sum_{i=1}^{N}(w_i(z_i - \beta_0 - x_i^T\beta)^2 + C(\tilde{\beta}_0, \tilde{\beta})$
- Use Coordinate Descent to solve the problem
  $minimize_{(\beta_0,\beta)}\{l_Q\{\beta_0, \beta) + \lambda||\beta||_1\}$

By analogy with Section 5.3.3, this is known as a generalized Newton algorithm,and the solution to the minimization problem (5.56)) defines a proximal Newton map

Logistic Regression & Coordinate Descent

Algorithm

OUTER LOOP: Decrement λ

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

INNER LOOP: Run the coordinate descent algorithm on the penalized weighted least squares problem

Logistic Regression & Coordinate Descent

Algorithm

OUTER LOOP: Decrement λ

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

INNER LOOP: Run the coordinate descent algorithm on the penalized weighted least squares problem

Least Angle Regression

Introduction

- Relates to Forward Selection method
- Relates to Lasso method
- Able to deliver the entire solution path of the lasso problem with squared-error loss as a function of the regularization parameter λ

Least Angle Regression

Introduction

- Relates to Forward Selection method
- Relates to Lasso method
- Able to deliver the entire solution path of the lasso problem with squared-error loss as a function of the regularization parameter λ

Least Angle Regression: Algorithm

- Start with all coefficients β_j equal to zero
- Find the predictor X_j **most correlated** with y
- **Increase** the coefficient β_j in the direction of the **sign** of its **correlation** with y
- Take **residuals** $r = y - \hat{y}$ along the way; Stop when some other predictor X_k has **as much correlation** with r as X_j has
- **Increase** β_j, β_k in their **joint least squares direction**, until some other predictor has as much correlation with the residual r

Continue until: all predictors are in the model

Least Angle Regression: Algorithm
- Start with all coefficients β_j equal to zero
- Find the predictor X_j **most correlated** with y
- **Increase** the coefficient β_j in the direction of the **sign** of its **correlation** with y
- Take **residuals** $r = y - \hat{y}$ along the way; Stop when some other predictor X_k has **as much correlation** with r as X_j has
- **Increase** β_j, β_k in their **joint least squares direction**, until some other predictor has as much correlation with the residual r

Continue until: all predictors are in the model

Least Angle Regression: Algorithm

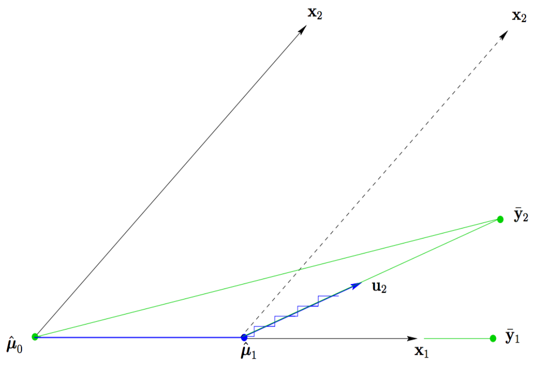
Algorithm 5.1 LEAST ANGLE REGRESSION.

1. Standardize the predictors to have mean zero and unit ℓ_2 norm. Start with the residual $\boldsymbol{r}_0 = \mathbf{y} - \bar{\mathbf{y}}$, $\beta^0 = (\beta_1, \beta_2, \ldots, \beta_p) = \mathbf{0}$.
2. 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 λ_0, define the active set $\mathcal{A} = \{j\}$, and $\mathbf{X}_\mathcal{A}$, the matrix consisting of this single variable.
3. For $k = 1, 2, \ldots, K = \min(N-1, p)$ do:
   (a) Define the least-squares direction $\delta = \frac{1}{\lambda_{k-1}}(\mathbf{X}_\mathcal{A}^T \mathbf{X}_\mathcal{A})^{-1} \mathbf{X}_\mathcal{A}^T \boldsymbol{r}_{k-1}$, and define the p-vector Δ such that $\Delta_\mathcal{A} = \delta$, and the remaining elements are zero.
   (b) Move the coefficients β from β^{k-1} in the direction Δ toward their least-squares 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 $\boldsymbol{r}(\lambda) = \mathbf{y} - \mathbf{X}\beta(\lambda) = \boldsymbol{r}_{k-1} - (\lambda_{k-1} - \lambda)\mathbf{X}\Delta$.
   (c) Keeping track of $|\langle \mathbf{x}_\ell, \boldsymbol{r}(\lambda) \rangle|$ for $\ell \notin \mathcal{A}$, identify the largest value of λ 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" λ_k.
   (d) Set $\mathcal{A} = \mathcal{A} \cup \{j\}$, $\beta^k = \beta(\lambda_k) = \beta^{k-1} + (\lambda_{k-1} - \lambda_k)\Delta$, and $\boldsymbol{r}_k = \mathbf{y} - \mathbf{X}\beta^k$.
4. Return the sequence $\{\lambda_k, \beta^k\}_0^K$.

2

[2] Picture taken from *Statistical Learning with Sparsity* page 119

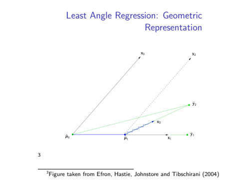
Least Angle Regression: Geometric Representation



[3]

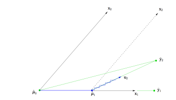
[3] Figure taken from Efron, Hastie, Johnstore and Tibschirani (2004)

Least Angle Regression: Geometric Representation



The LARS algorithm in the case of $m = 2$ covariates; y_2 is the projection of y into $L(x1, x2)$. Beginning at $\mu_0 = 0$, the residual vector $y_2 - \mu_0$ has greater correlation with x_1 than x_2;the next LARS estimate is $\mu_1 = \mu_0 + \gamma_1 x_1$, where γ_1 is chosen such that $y_2 - \mu_1$ bisects the angle between x_1 and x_2; then $\mu_2 = \mu_1 + \gamma_2 u_2$, where u_2 is the unit bisector; $\mu_2 = y_2$ in the case $m = 2$,but not for the case $m > 2$; see Figure 4. The staircase indicates a typical Stagewise path. Here LARS gives the Stagewise track as $\epsilon \to 0$, but a modification is necessary to guarantee agreement in higher dimensions.

Least Angle Regression: Geometric Representation



[3]Figure taken from Efron, Hastie, Johnstore and Tibschirani (2004)

The LARS algorithm in the case of $m = 2$ covariates; y_2 is the projection of y into $L(x1, x2)$. Beginning at $\mu_0 = 0$, the residual vector $y_2 - \mu_0$ has greater correlation with x_1 than x_2;the next LARS estimate is $\mu_1 = \mu_0 + \gamma_1 x_1$, where γ_1 is chosen such that $y_2 - \mu_1$ bisects the angle between x_1 and x_2; then $\mu_2 = \mu_1 + \gamma_2 u_2$, where u_2 is the unit bisector; $\mu_2 = y_2$ in the case $m = 2$,but not for the case $m > 2$; see Figure 4. The staircase indicates a typical Stagewise path. Here LARS gives the Stagewise track as $\epsilon \to 0$, but a modification is necessary to guarantee agreement in higher dimensions.

Connection between LAR and Lasso

LAR

$\mathbf{x}_j^T(\mathbf{y}-\mathbf{X}\beta(\lambda)) = \lambda \cdot s_j,\ \forall j \in \mathbb{A}$ where s_j is the sign of inner product λ. [3(c)]

Connection between LAR and Lasso

LAR

$\mathbf{x}_j^T(\mathbf{y}-\mathbf{X}\beta(\lambda)) = \lambda \cdot s_j,\ \forall j \in \mathbb{A}$ where s_j is the sign of inner product λ. [3(c)]

Connection between LAR and Lasso

LAR

$\mathbf{x}_j^T(\mathbf{y} - \mathbf{X}\beta(\lambda)) = \lambda \cdot s_j, \ \forall j \in \mathbb{A}$ where s_j is the sign of inner product λ. [3(c)]

LASSO

Let $\mathbb{B}$ be the active set of variables in the solution for a given value of λ.
$R(\beta) = \frac{1}{2}||\mathbf{y} - \mathbf{X}\beta||_2^2 + \lambda||\beta||_1$

For differentiable $R(\beta)$, the stationary conditions give:
$\mathbf{x}_j^T(\mathbf{y} - \mathbf{X}\beta) = \lambda \cdot sign(\beta_j), \forall j \in \mathbb{B}$

If sign (β_j) matches s_j, the coefficient would be identical.

Connection between LAR and Lasso

LAR
$\mathbf{x}_j^T(\mathbf{y} - \mathbf{X}\beta(\lambda)) = \lambda \cdot s_j, \ \forall j \in \mathbb{A}$ where s_j is the sign of inner product λ. [3(c)]

LASSO
Let $\mathbb{B}$ be the active set of variables in the solution for a given value of λ.
$R(\beta) = \frac{1}{2}||\mathbf{y} - \mathbf{X}\beta||_2^2 + \lambda||\beta||_1$

For differentiable $R(\beta)$, the stationary conditions give:
$\mathbf{x}_j^T(\mathbf{y} - \mathbf{X}\beta) = \lambda \cdot sign(\beta_j), \forall j \in \mathbb{B}$

If sign (β_j) matches s_j, the coefficient would be identical.

Connection between LAR and Lasso

- R Example
- LAR algorithm explains that the coefficient paths for the lasso are **piecewise linear**
- Coefficient paths differ if $\text{sign}(\beta_j)$ is different from s_j
- **Modification** of LAR for computing Lasso solution [3(c)+]:
  If a **nonzero** coefficient **crosses zero** before the next variable enters, **drop** it from $\mathbb{A}$ and recompute the current joint least squares direction.

Connection between LAR and Lasso

- R Example
- LAR algorithm explains that the coefficient paths for the lasso are **piecewise linear**
- Coefficient paths differ if $\text{sign}(\beta_j)$ is different from s_j
- **Modification** of LAR for computing Lasso solution [3(c)+]:
  If a **nonzero** coefficient **crosses zero** before the next variable enters, **drop** it from $\mathbb{A}$ and recompute the current joint least squares direction.

Connection between LAR and Lasso

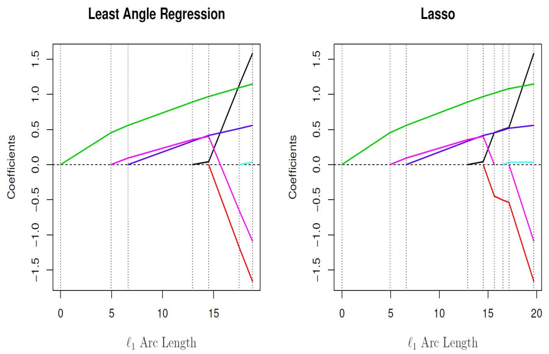


Figure: Cases where signs of λ and β disagree

4

[4] Picture taken from *Statistical Learning with Sparsity* page 120

Connection between LAR and Lasso

Figure: Cases where signs of λ and β disagree

[4] Picture taken from *Statistical Learning with Sparsity* page 120

Algorithm Performance

Simulation: Comparison of computation efficiency between CD and LAR

Set Up: [5]

- Generate Gaussian data with N observations and p predictors, with each pair of predictors X_j, X_k having the same population correlation ρ.
- Try different combination of N and p; Range ρ from 0 to 0.95.
  $Y = \sum_{j=1}^{p} X_j \beta j + kZ$ where
  $\beta_j = (-1)^j exp(\frac{-2(j-1)}{20}), Z \sim N(0,1)$ and k is a constant.

[5] Friedman,Hastie, Tibshirani (2010)

Algorithm Performance

Simulation: Comparison of computation efficiency between CD and LAR

Set Up: [5]

- Generate Gaussian data with N observations and p predictors, with each pair of predictors X_j, X_k having the same population correlation ρ.
- Try different combination of N and p; Range ρ from 0 to 0.95.
  $Y = \sum_{j=1}^{p} X_j \beta j + kZ$ where
  $\beta_j = (-1)^j exp(\frac{-2(j-1)}{20}), Z \sim N(0,1)$ and k is a constant.

[5] Friedman,Hastie, Tibshirani (2010)

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 λ values, averaged over 3 runs.

Algorithm Performance

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
	$N = 100, p = 50000$					

Figure: Comparison of computing time

Algorithm Performance

Figure: Comparison of computing time

Algorithm Performance

Simulation: Comparison of computation efficiency between Coordinate Descent, Proximal Gradient Descent and Nestrov Method

Set Up: [7]

- Generated an $N \times p$ predictor matrix X with standard Gaussian entries and pairwise correlation 0 or 0.5 between the features.
- $|\beta_j| = exp\big[-0.5(u(j-1))^2\big]$ and $u = \sqrt{\frac{\pi}{20}}$ and alternating signs -1,+1,-1...

[7] *Statistical Learning with Sparsity* Page 117

Algorithm Performance

Simulation: Comparison of computation efficiency between Coordinate Descent, Proximal Gradient Descent and Nestrov Method

Set Up: [7]

- Generated an $N \times p$ predictor matrix X with standard Gaussian entries and pairwise correlation 0 or 0.5 between the features.
- $|\beta_j| = exp\big[-0.5(u(j-1))^2\big]$ and $u = \sqrt{\frac{\pi}{20}}$ and alternating signs -1,+1,-1...

[7] *Statistical Learning with Sparsity* Page 117

Algorithm Performance

Table 5.1 *Lasso for linear regression: Average (standard error) of CPU times over ten realizations, for coordinate descent, generalized gradient, and Nesterov's momentum methods. In each case, time shown is the total time over a path of 20 λ values.*

	$N = 10000,\ p = 100$		$N = 200,\ p = 10000$	
Correlation	0	0.5	0	0.5
Coordinate descent	0.110 (0.001)	0.127 (0.002)	0.298 (0.003)	0.513 (0.014)
Proximal gradient	0.218 (0.008)	0.671 (0.007)	1.207 (0.026)	2.912 (0.167)
Nesterov	0.251 (0.007)	0.604 (0.011)	1.555 (0.049)	2.914 (0.119)

Figure: Comparison of computing efficiency between 3 methods

8

[8] *Statistical Learning with Sparsity* Page 117

Algorithm Performance

Figure: Comparison of computing efficiency between 3 methods

8

[8] *Statistical Learning with Sparsity* Page 117

Recall: Duality in optimization

Recall: Duality in optimization

In various section, I came across terms like "dual" and "dual problem"

Primal	
Optimize	$\min f(x)$
Constraints	$g_i(x) \leq 0, h_j(x) = 0, x \in X$
Function	$L(x, \lambda, \mu) := f(x) + \sum_i \lambda_i g_i(x) + \sum_j \mu_j h_j(x)$
Dual	
Function	$q(\lambda, \mu) = \inf_{x \in X} L(x, \lambda, \mu)$
Constraints	$\lambda \geq 0$
Optimize	$\max q(\lambda, \mu)$

Why though? - **Dual problem is always convex!**

Primal	
Optimize	$\min f(x)$
Constraints	$g_i(x) \leq 0, h_j(x) = 0, x \in X$
Function	$L(x, \lambda, \mu) := f(x) + \sum_i \lambda_i g_i(x) + \sum_j \mu_j h_j(x)$
Dual	
Function	$q(\lambda, \mu) = \inf_{x \in X} L(x, \lambda, \mu)$
Constraints	$\lambda \geq 0$
Optimize	$\max q(\lambda, \mu)$

Why though? - **Dual problem is always convex!**

$x \in X$ for e.g. solutions in a cone or integer solutions
Terms: Primal problem, Lagrange function with dual variables/Lagrange-multipliers, dual function (λ and μ now in vector notation), dual problem (max q)
Dual problem is always convex! - I don't know much about optimization yet, but they really like convexity.
"(Convexity confers two advantages. The first is that, in a constrained problem, a convex feasible region makes it easier to ensure that you do not generate infeasible solutions while searching for an optimum.)
The second advantage is that all local optima are global optima. That allows local search algorithms to guarantee optimal solutions. And local search is often faster." [Rubin, 2016])

Alternating Direction Method of Multipliers (ADMM)

Problem

$$\underset{\beta\in\mathbb{R}^m,\theta\in\mathbb{R}^n}{\text{minimize}}\, f(\beta)+g(\theta) \quad \text{subject to } \mathbf{A}\beta+\mathbf{B}\theta-c=0$$

Lagrangian

$$f(\beta)+g(\theta)+\langle\mu,\mathbf{A}\beta+\mathbf{B}\theta-c\rangle$$

Augmented Lagrangian

$$L_\rho(\beta,\theta,\mu):=f(\beta)+g(\theta)+\langle\mu,\mathbf{A}\beta+\mathbf{B}\theta-c\rangle+\frac{\rho}{2}||\mathbf{A}\beta+\mathbf{B}\theta-c||_2^2$$

Alternating Direction Method of Multipliers (ADMM)

Problem

$$\underset{\beta\in\mathbb{R}^m,\theta\in\mathbb{R}^n}{\text{minimize}}\, f(\beta)+g(\theta) \quad \text{subject to } \mathbf{A}\beta+\mathbf{B}\theta-c=0$$

Lagrangian

$$f(\beta)+g(\theta)+\langle\mu,\mathbf{A}\beta+\mathbf{B}\theta-c\rangle$$

Augmented Lagrangian

$$L_\rho(\beta,\theta,\mu):=f(\beta)+g(\theta)+\langle\mu,\mathbf{A}\beta+\mathbf{B}\theta-c\rangle+\frac{\rho}{2}||\mathbf{A}\beta+\mathbf{B}\theta-c||_2^2$$

Alternating Direction Method of Multipliers (ADMM)

Problem - decomposable !

$$\underset{\beta\in\mathbb{R}^m,\theta\in\mathbb{R}^n}{\text{minimize}}\, f(\beta)+g(\theta) \quad \text{subject to } \mathbf{A}\beta+\mathbf{B}\theta-c=0$$

Lagrangian

$$f(\beta)+g(\theta)+\langle\mu,\mathbf{A}\beta+\mathbf{B}\theta-c\rangle$$

Augmented Lagrangian

$$L_\rho(\beta,\theta,\mu):=f(\beta)+g(\theta)+\langle\mu,\mathbf{A}\beta+\mathbf{B}\theta-c\rangle+\frac{\rho}{2}||\mathbf{A}\beta+\mathbf{B}\theta-c||_2^2$$

Alternating Direction Method of Multipliers (ADMM)

Problem - decomposable !

$$\underset{\beta\in\mathbb{R}^m,\theta\in\mathbb{R}^n}{\text{minimize}}\, f(\beta)+g(\theta) \quad \text{subject to } \mathbf{A}\beta+\mathbf{B}\theta-c=0$$

Lagrangian

$$f(\beta)+g(\theta)+\langle\mu,\mathbf{A}\beta+\mathbf{B}\theta-c\rangle$$

Augmented Lagrangian

$$L_\rho(\beta,\theta,\mu):=f(\beta)+g(\theta)+\langle\mu,\mathbf{A}\beta+\mathbf{B}\theta-c\rangle+\frac{\rho}{2}||\mathbf{A}\beta+\mathbf{B}\theta-c||_2^2$$

decomposable problem and constraints!

Alternating Direction Method of Multipliers (ADMM)

Problem - decomposable !

$$\underset{\beta\in\mathbb{R}^m,\theta\in\mathbb{R}^n}{\text{minimize}}\, f(\beta)+g(\theta) \quad \text{subject to } \mathbf{A}\beta+\mathbf{B}\theta-c=0$$

Lagrangian - decomposable !

$$f(\beta)+g(\theta)+\langle\mu,\mathbf{A}\beta+\mathbf{B}\theta-c\rangle$$

Augmented Lagrangian

$$L_\rho(\beta,\theta,\mu):=f(\beta)+g(\theta)+\langle\mu,\mathbf{A}\beta+\mathbf{B}\theta-c\rangle+\frac{\rho}{2}||\mathbf{A}\beta+\mathbf{B}\theta-c||_2^2$$

Alternating Direction Method of Multipliers (ADMM)
Problem - decomposable !
$\underset{\beta\in\mathbb{R}^m,\theta\in\mathbb{R}^n}{\text{minimize}}\, f(\beta)+g(\theta) \quad \text{subject to } \mathbf{A}\beta+\mathbf{B}\theta-c=0$
Lagrangian - decomposable !
$f(\beta)+g(\theta)+\langle\mu,\mathbf{A}\beta+\mathbf{B}\theta-c\rangle$
Augmented Lagrangian
$L_\rho(\beta,\theta,\mu):=f(\beta)+g(\theta)+\langle\mu,\mathbf{A}\beta+\mathbf{B}\theta-c\rangle+\frac{\rho}{2}||\mathbf{A}\beta+\mathbf{B}\theta-c||_2^2$

Lagrangian problem can still be decomposed into β and μ terms
this has nice algorithm where we can execute some stuff in parallel, because we can decompose the Lagrangian

Alternating Direction Method of Multipliers (ADMM)

Problem - decomposable !

$$\underset{\beta\in\mathbb{R}^m,\theta\in\mathbb{R}^n}{\text{minimize}}\, f(\beta)+g(\theta) \quad \text{subject to } \mathbf{A}\beta+\mathbf{B}\theta-c=0$$

Lagrangian - decomposable !

$$f(\beta)+g(\theta)+\langle\mu,\mathbf{A}\beta+\mathbf{B}\theta-c\rangle$$

Augmented Lagrangian - NOT decomposable !

$$L_\rho(\beta,\theta,\mu):=f(\beta)+g(\theta)+\langle\mu,\mathbf{A}\beta+\mathbf{B}\theta-c\rangle+\frac{\rho}{2}||\mathbf{A}\beta+\mathbf{B}\theta-c||_2^2$$

Alternating Direction Method of Multipliers (ADMM)
Problem - decomposable !
$\underset{\beta\in\mathbb{R}^m,\theta\in\mathbb{R}^n}{\text{minimize}}\, f(\beta)+g(\theta)$ subject to $\mathbf{A}\beta+\mathbf{B}\theta-c=0$
Lagrangian - decomposable !
$f(\beta)+g(\theta)+\langle\mu,\mathbf{A}\beta+\mathbf{B}\theta-c\rangle$
Augmented Lagrangian - NOT decomposable !
$L_\rho(\beta,\theta,\mu):=f(\beta)+g(\theta)+\langle\mu,\mathbf{A}\beta+\mathbf{B}\theta-c\rangle+\frac{\rho}{2}||\mathbf{A}\beta+\mathbf{B}\theta-c||_2^2$

Augmented: scalar product with ρ gets added,
Method of Multipliers: is a way to make the algorithm more robust
advantage: better convergence
disadvantage: no longer parallel execution of subtasks due to l2-term, no longer decomposable in beta and theta terms, as l2 norm dquares every entry of the vector
alternating direction: semi-decomposable, i.e. keeping one variable fixed while updating the other
ρ is step length of iterative algorithm
All notes on this slide: see the slides by [Boyd]

Dual Variable Update

Alternating Direction Method of Multipliers

$$\beta^{t+1} = \arg\min_{\beta \in \mathbb{R}^m} L_\rho(\beta, \theta^t, \mu^t)$$

$$\theta^{t+1} = \arg\min_{\theta \in \mathbb{R}^m} L_\rho(\beta^{t+1}, \theta, \mu^t)$$

$$\mu^{t+1} = \mu^t + \rho(\mathbf{A}\beta^{t+1} + \mathbf{B}\theta^{t+1} - c)$$

Dual Variable Update
Alternating Direction Method of Multipliers

$$\beta^{t+1} = \arg\min_{\beta \in \mathbb{R}^m} L_\rho(\beta, \theta^t, \mu^t)$$
$$\theta^{t+1} = \arg\min_{\theta \in \mathbb{R}^m} L_\rho(\beta^{t+1}, \theta, \mu^t)$$
$$\mu^{t+1} = \mu^t + \rho(\mathbf{A}\beta^{t+1} + \mathbf{B}\theta^{t+1} - c)$$

Dual Variable Update

Alternating Direction Method of Multipliers

$$\beta^{t+1} = \underset{\beta \in \mathbb{R}^m}{\arg\min}\, L_\rho(\beta, \theta^t, \mu^t)$$

$$\theta^{t+1} = \underset{\theta \in \mathbb{R}^m}{\arg\min}\, L_\rho(\beta^{t+1}, \theta, \mu^t)$$

$$\mu^{t+1} = \mu^t + \rho(\mathbf{A}\beta^{t+1} + \mathbf{B}\theta^{t+1} - c)$$

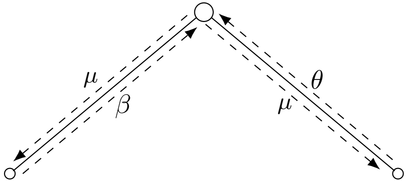
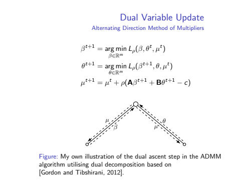


Figure: My own illustration of the dual ascent step in the ADMM algorithm utilising dual decomposition based on [Gordon and Tibshirani, 2012].



Method of Multipliers: is a way to make the algorithm more robust, (if in second line β^t statt β^{t+1})
alternating direction: semi-decomposable, i.e. keeping one variable fixed while updating the other
think of it as only the last line, sending μ to the updaters for β and θ
in this context ρ in last line can be thought of as "step length"

All notes on this slide: see the slides by [Boyd]

ADMM - Why?

- convex problems with nondifferentiable constraints
- blockwise computation
  - sample blocks
  - feature blocks

ADMM - Why?

- convex problems with nondifferentiable constraints
- blockwise computation
  - sample blocks
  - feature blocks

Details for blockwise computation in Exercise 5.12.

ADMM for the Lasso

Problem

Problem in Lagrangian form

$$\underset{\beta \in \mathbb{R}^p, \theta \in \mathbb{R}^p}{\text{minimize}} \left\{ \frac{1}{2} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda \|\theta\|_1 \right\} \quad \text{such that } \beta - \theta = 0$$

Augmented Lagrangian

$$L_\rho(\beta, \theta, \mu) := \left\{ \frac{1}{2} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda \|\theta\|_1 \right\} + \langle \mu, \beta - \theta \rangle + \frac{\rho}{2} ||\beta - \theta||_2^2$$

ADMM for the Lasso
Problem
Problem in Lagrangian form
$\underset{\beta \in \mathbb{R}^p, \theta \in \mathbb{R}^p}{\text{minimize}} \left\{ \frac{1}{2} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda \|\theta\|_1 \right\}$ such that $\beta - \theta = 0$
Augmented Lagrangian
$L_\rho(\beta, \theta, \mu) := \left\{ \frac{1}{2} \|\mathbf{y} - \mathbf{X}\beta\|_2^2 + \lambda \|\theta\|_1 \right\} + \langle \mu, \beta - \theta \rangle + \frac{\rho}{2} ||\beta - \theta||_2^2$

In the problem, I can decompose into beta and theta terms, i.e.show $f(\beta)$ and $g(\theta)$
the problem itself and the constraints,
A and B are unit matrices here

ADMM for the Lasso

Update

Update

$$\beta^{t+1} = (\mathbf{X}^T\mathbf{X} + \rho\mathbf{I})^{-1}(\mathbf{X}^T\mathbf{y} + \rho\theta^t - \mu^t)$$
$$\theta^{t+1} = \mathcal{S}_{\lambda/\rho}(\beta^{t+1} + \mu^t/\rho)$$
$$\mu^{t+1} = \mu^t + \rho(\beta^{t+1} - \theta^{t+1})$$

where $\mathcal{S}_{\lambda/\rho}(z) = \text{sign}(z)(|z| - \frac{\lambda}{\rho})_+$.

ADMM for the Lasso
Update

Update

$\beta^{t+1} = (\mathbf{X}^T\mathbf{X} + \rho\mathbf{I})^{-1}(\mathbf{X}^T\mathbf{y} + \rho\theta^t - \mu^t)$
$\theta^{t+1} = \mathcal{S}_{\lambda/\rho}(\beta^{t+1} + \mu^t/\rho)$
$\mu^{t+1} = \mu^t + \rho(\beta^{t+1} - \theta^{t+1})$

where $\mathcal{S}_{\lambda/\rho}(z) = \text{sign}(z)(|z| - \frac{\lambda}{\rho})_+$.

$\mathcal{S}$ is a soft-thresholding parameter
Computational cost: Initially $\mathcal{O}(p^3)$, which is a lot, for the SVD(singular value decomposition of $\mathbf{X}$), after that comparable to coordinate descent or composite gradient from earlier

Screening Rules

- Pre-processing to eliminate features
- very big data set, esp. huge number of predictors
- maybe too big to load into memory
- Screening rules eliminate predictors with minor calculation
- and very high / safe certainty (i.e. eliminated predictors would not show up in lasso model based on full data)

They achieve a reduction in the number of variables, typically by an order of mgnitude

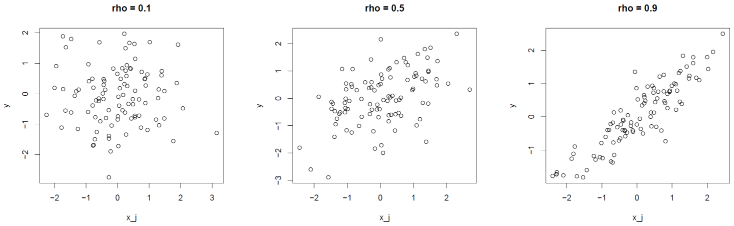
Screening Rules

- Pre-processing to eliminate features
- very big data set, esp. huge number of predictors
- maybe too big to load into memory
- Screening rules eliminate predictors with minor calculation
- and very high / safe certainty (i.e. eliminated predictors would not show up in lasso model based on full data)

They achieve a reduction in the number of variables, typically by an order of mgnitude

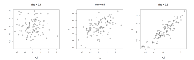
Imagine a big data set, a very big data set, with such a huge design matrix, that you cannot load it into memory (RAM).

What is a good predictor?



correlation is an inner product
high absolute correlation (=large absolute inner product)
$->$ high predictive power (compare plots)
$->$ x_j with largest inner product has highest predictive power, thus for that j we are most willing to accept some penalty from λ

What is a good predictor?



correlation is an inner product
high absolute correlation (=large absolute inner product)
$->$ high predictive power (compare plots)
$->$ x_j with largest inner product has highest predictive power, thus for that j we are most willing to accept some penalty from λ

Lasso - a different perspective I

Let $\mathcal{A}$ be the active set of predictors. Let λ take values on a decreasing sequence.

iterate

1. order predictors x_j not in $\mathcal{A}$ by their "effectiveness" using $\left|x_j^T y\right|$ or better $\left|x_j^T(y - \hat{y}_\lambda)\right|$, call the best predictor $x_{j_{\max}}$
2. move λ such that the positive effect from the best predictor $x_{j_{\max}}$ compensates the penalty by λ
3. calculate solution for chosen λ

Lasso - a different perspective I

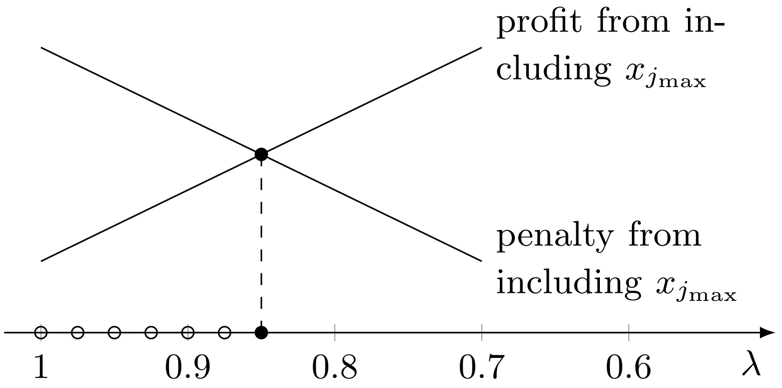
Let $\mathcal{A}$ be the active set of predictors. Let λ take values on a decreasing sequence.

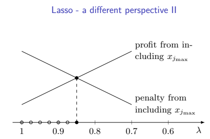
iterate

1. order predictors x_j not in $\mathcal{A}$ by their "effectiveness" using $\left|x_j^T y\right|$ or better $\left|x_j^T(y - \hat{y}_\lambda)\right|$, call the best predictor $x_{j_{\max}}$
2. move λ such that the positive effect from the best predictor $x_{j_{\max}}$ compensates the penalty by λ
3. calculate solution for chosen λ

this is just a formalisation of the previous slide
or better b/c we want to focus on the residuals once we have a preliminary solution

Lasso - a different perspective II





visualisation of step 2 of the previous slide

these lines are not linear, neither is usually the spacing on the lambda axis

we are walking along the lambda axis until we find a good point / the intersection between the penalty and the profit

Back to screening rules

Let λ take values on a decreasing sequence. Let $\lambda_{\max}$ be the λ where the first predictor has a non-zero coefficient.

$$\lambda_{\max} = \max_j \left| x_j^T y \right|$$

Let $\mathcal{A}$ be the active set of predictors.

$$\forall j \in \mathcal{A} \quad \lambda = \left| x_j^T (y - \hat{y}) \right|$$

$$\forall j \notin \mathcal{A} \quad \lambda \geq \left| x_j^T (y - \hat{y}) \right|$$

Back to screening rules

Let λ take values on a decreasing sequence. Let $\lambda_{\max}$ be the λ where the first predictor has a non-zero coefficient.

$$\lambda_{\max} = \max_j \left| x_j^T y \right|$$

Let $\mathcal{A}$ be the active set of predictors.

$$\forall j \in \mathcal{A} \quad \lambda = \left| x_j^T (y - \hat{y}) \right|$$

$$\forall j \notin \mathcal{A} \quad \lambda \geq \left| x_j^T (y - \hat{y}) \right|$$

for those wondering why in first equation not y-yhat? anybody?
yhat would come from the empty/intercept model, i.e. yhat=mean(y)
we assume standardised data (i.e. mean 0 and unit variance)
thus yhat = 0

Back to screening rules

Let λ take values on a decreasing sequence. Let $\lambda_{\max}$ be the λ where the first predictor has a non-zero coefficient.

$$\lambda_{\max} = \max_j \left| x_j^T y \right|$$

Let $\mathcal{A}$ be the active set of predictors.

$$\forall j \in \mathcal{A} \quad \lambda = \left| x_j^T (y - \hat{y}) \right|$$

$$\forall j \notin \mathcal{A} \quad \lambda \geq \left| x_j^T (y - \hat{y}) \right|$$

Back to screening rules

Let λ take values on a decreasing sequence. Let $\lambda_{\max}$ be the λ where the first predictor has a non-zero coefficient.

$$\lambda_{\max} = \max_j \left| x_j^T y \right|$$

Let $\mathcal{A}$ be the active set of predictors.

$$\forall j \in \mathcal{A} \quad \lambda = \left| x_j^T (y - \hat{y}) \right|$$

$$\forall j \notin \mathcal{A} \quad \lambda \geq \left| x_j^T (y - \hat{y}) \right|$$

this is the essential equation for screening rules, if, for a given lambda, a predictor does not fulfil this equation, we kick it out

SHOW R STUFF SCREENINGRULES 2

Global vs. Sequential

Global (one-time screening):
Suppose we want to calculate a lasso solution at $\lambda < \lambda_{\text{max}}$.

Sequential (iterative screening):
Suppose we have the lasso solution $\hat{\beta}(\lambda')$ at λ' and want to screen variables for solutions at $\lambda < \lambda'$.

Global vs. Sequential

Global (one-time screening):
Suppose we want to calculate a lasso solution at $\lambda < \lambda_{\text{max}}$.

Sequential (iterative screening):
Suppose we have the lasso solution $\hat{\beta}(\lambda')$ at λ' and want to screen variables for solutions at $\lambda < \lambda'$.

There are two main classes of Screening rules

Dual Polytope Projection (DPP)

Suppose we want to calculate a lasso solution at $\lambda < \lambda_{\text{max}}$. The DPP rule discards the j^{th} variable if

$$\left|\mathbf{x}_j^T \mathbf{y}\right| < \lambda_{\text{max}} - \|\mathbf{x}_j\|_2 \, \|\mathbf{y}\|_2 \, \frac{\lambda_{\text{max}} - \lambda}{\lambda}$$

Sequential DPP rule

Suppose we have the lasso solution $\hat{\beta}(\lambda')$ at λ' and want to screen variables for solutions at $\lambda < \lambda'$. We discard the j^{th} variable if

$$\left|\mathbf{x}_j^T (\mathbf{y} - \mathbf{X}\hat{\beta}(\lambda'))\right| < \lambda' - \|\mathbf{x}_j\|_2 \, \|\mathbf{y}\|_2 \, \frac{\lambda_{\text{max}} - \lambda}{\lambda}$$

Dual Polytope Projection (DPP)

Suppose we want to calculate a lasso solution at $\lambda < \lambda_{\text{max}}$. The DPP rule discards the j^{th} variable if

$$\left|\mathbf{x}_j^T \mathbf{y}\right| < \lambda_{\text{max}} - \|\mathbf{x}_j\|_2 \, \|\mathbf{y}\|_2 \, \frac{\lambda_{\text{max}} - \lambda}{\lambda}$$

Sequential DPP rule

Suppose we have the lasso solution $\hat{\beta}(\lambda')$ at λ' and want to screen variables for solutions at $\lambda < \lambda'$. We discard the j^{th} variable if

$$\left|\mathbf{x}_j^T (\mathbf{y} - \mathbf{X}\hat{\beta}(\lambda'))\right| < \lambda' - \|\mathbf{x}_j\|_2 \, \|\mathbf{y}\|_2 \, \frac{\lambda_{\text{max}} - \lambda}{\lambda}$$

Global Strong Rule

Suppose we want to calculate a lasso solution at $\lambda < \lambda_{\text{max}}$. The global strong rule discards the j^{th} variable if

$$\left|\mathbf{x}_j^T \mathbf{y}\right| < \lambda - (\lambda_{\text{max}} - \lambda) = 2\lambda - \lambda_{\text{max}}$$

Sequential Strong Rule

Suppose we have the lasso solution $\hat{\beta}(\lambda')$ at λ' and want to screen variables for solutions at $\lambda < \lambda'$. We discard the j^{th} variable if

$$\left|\mathbf{x}_j^T (\mathbf{y} - \mathbf{X}\hat{\beta}(\lambda'))\right| < 2\lambda - \lambda'$$

Global Strong Rule

Suppose we want to calculate a lasso solution at $\lambda < \lambda_{\text{max}}$. The global strong rule discards the j^{th} variable if

$$\left|\mathbf{x}_j^T \mathbf{y}\right| < \lambda - (\lambda_{\text{max}} - \lambda) = 2\lambda - \lambda_{\text{max}}$$

Sequential Strong Rule

Suppose we have the lasso solution $\hat{\beta}(\lambda')$ at λ' and want to screen variables for solutions at $\lambda < \lambda'$. We discard the j^{th} variable if

$$\left|\mathbf{x}_j^T (\mathbf{y} - \mathbf{X}\hat{\beta}(\lambda'))\right| < 2\lambda - \lambda'$$

Screening Rules - Example Setup

- simulated dataset
- $N = 200, p = 5000$ uncorrelated Gaussian predictors,
- $1/4$ true non-zero coefficients
- 100 decreassing lambda values equally spaced on the log-scale
- Compare Global DPP, Global Strong, Sequential DDP, Sequential Strong
- no violations for either of the strong rules

Screening Rules - Example Setup

- simulated dataset
- $N = 200, p = 5000$ uncorrelated Gaussian predictors,
- 1/4 true non-zero coefficients
- 100 decreassing lambda values equally spaced on the log-scale
- Compare Global DPP, Global Strong, Sequential DDP, Sequential Strong
- no violations for either of the strong rules

Figure: From [Hastie et al., 2015]

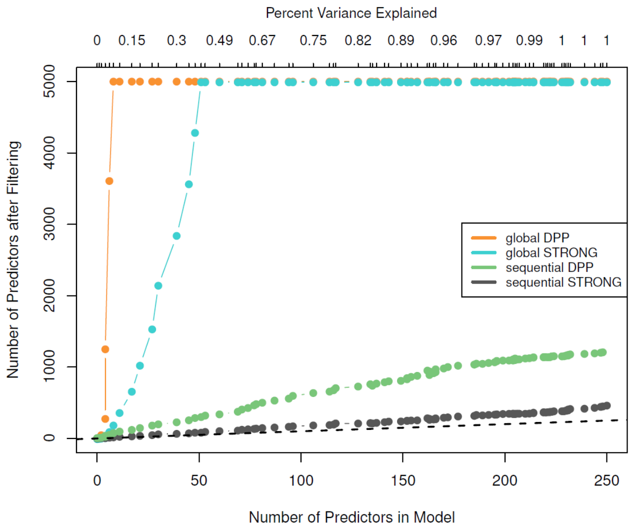
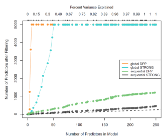


Figure: From [Hastie et al., 2015]



Lasso regression: Results of different rules applied to a simulated dataset. There are $N = 200$ observations and $p = 5000$ uncorrelated Gaussian predictors; one-quarter of the true coefficients are nonzero. Shown are the number of predictors left after screening at each stage, plotted against the number of predictors in the model for a given value of λ. The value of λ is decreasing as we move from left to right. In the plots, we are fitting along a path of 100 decreasing λ values equally spaced on the log-scale, **A broken line with unit slope is added for reference.** The proportion of variance explained by the model is shown along the top of the plot. There were no violations for either of the strong rules.

Summary I

Coordinate Descent

- An efficient algorithm implemented in glmnet but requires separability condition
- Application: Ridge, Lasso, Elastic Net, Logistic Regression, etc.

Least Angle Regression

- Similar to the idea of Forward Selection
- Computationally efficient but does not scale well to large problems

Connection between LASSO and LAR

- LAR could be modified to obtain Lasso solution
- Explains the fact that Lasso coefficient solution path is piece-wise linear

Summary I
Coordinate Descent
- An efficient algorithm implemented in glmnet but requires separability condition
- Application: Ridge, Lasso, Elastic Net, Logistic Regression, etc.
Least Angle Regression
- Similar to the idea of Forward Selection
- Computationally efficient but does not scale well to large problems
Connection between LASSO and LAR
- LAR could be modified to obtain Lasso solution
- Explains the fact that Lasso coefficient solution path is piece-wise linear

Summary II

ADMM

- Use duality to your advantage
- Limitations in speed for Lasso, but useful in more complex settings

Screening Rules

- Promising for very large p's
- Difficult to find best rule, field in development

Summary II

ADMM

- Use duality to your advantage
- Limitations in speed for Lasso, but useful in more complex settings

Screening Rules

- Promising for very large p's
- Difficult to find best rule, field in development

Minorization-Maximization Algorithms (MMA)

- Problem: minimize $f(\beta)$ over $\beta \in \mathbb{R}^p$ for f possibly non-convex
- Introduce additional variable θ
- Use θ to majorize (bound from above) the objective function to be minimized

Majorization-Minimization Algorithms work analoguosly.

Minorization-Maximization Algorithms (MMA)

- Problem: minimize $f(\beta)$ over $\beta \in \mathbb{R}^p$ for f possibly non-convex
- Introduce additional variable θ
- Use θ to majorize (bound from above) the objective function to be minimized

Majorization-Minimization Algorithms work analoguosly.

MMA visually

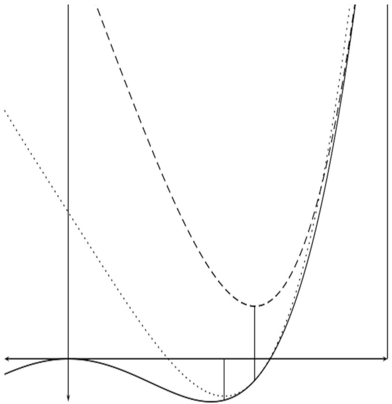
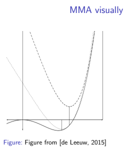


Figure: Figure from [de Leeuw, 2015]



MMA analytically I

Def. $\Psi : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ majorizes f at $\beta \in \mathbb{R}^p$ if

$$\forall \theta \in \mathbb{R}^p \quad \Psi(\beta, \theta) \geq f(\beta)$$

with equality for $\theta = \beta$.

Minor-Maxxalgorithm

- initialize β^0
- update with $\beta^{t+1} = \underset{\beta \in \mathbb{R}^p}{\arg\min}\, \Psi(\beta, \beta^t)$

2018-10-22

Computation & optimization
└Minor-Max Algorithms
└MMA analytically I

MMA analytically I

Def. $\Psi : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ majorizes f at $\beta \in \mathbb{R}^p$ if

$$\forall \theta \in \mathbb{R}^p \quad \Psi(\beta, \theta) \geq f(\beta)$$

with equality for $\theta = \beta$.

Minor-Maxxalgorithm

- initialize β^0
- update with $\beta^{t+1} = \underset{\beta \in \mathbb{R}^p}{\arg\min}\, \Psi(\beta, \beta^t)$

MMA analytically II

This scheme generates a sequence of β's for which the cost $f(\beta^t)$ is nonincreasing, because

$$f(\beta^t) \overset{(i)}{=} \Psi(\beta^t, \beta^t) \overset{(ii)}{\geq} \Psi(\beta^{t+1}, \beta^t) \overset{(iii)}{\geq} f(\beta^{t+1})$$

where

(i) & (iii) Definiton of majorize

(ii) β^{t+1} is a minimizer of $\beta \mapsto \Psi(\beta, \beta^t)$

MMA analytically II

This scheme generates a sequence of β's for which the cost $f(\beta^t)$ is nonincreasing, because

$$f(\beta^t) \overset{(i)}{=} \Psi(\beta^t, \beta^t) \overset{(ii)}{\geq} \Psi(\beta^{t+1}, \beta^t) \overset{(iii)}{\geq} f(\beta^{t+1})$$

where

(i) & (iii) Definiton of majorize

(ii) β^{t+1} is a minimizer of $\beta \mapsto \Psi(\beta, \beta^t)$

for inequalities: show previous slide

Biconvexity

Let's consider an example ...

$f(\alpha, \beta) = (1 - \alpha\beta)^2$

Biconvexity

Let's consider an example ...
$f(\alpha, \beta) = (1 - \alpha\beta)^2$

Mathematica: `3D plot (1-xy)^2, x in [-2,2], y in [-2,2]`
The formula is a link.

Biconvexity

Let's consider an example ...

$f(\alpha, \beta) = (1 - \alpha\beta)^2$

Def. A function $f(\alpha, \beta) : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$ is biconvex, if for each $\alpha \in \mathbb{R}^m$ the function $\alpha \mapsto f(\alpha, \beta)$ is convex and for each $\beta \in \mathbb{R}^n$ the function $\beta \mapsto f(\alpha, \beta)$ is convex. Analoguosly, a set $\mathcal{C} \subseteq \mathcal{A} \times \mathcal{B}$, for $\mathcal{A}, \mathcal{B}$ convex sets, is called biconvex, if it is convex

Biconvexity

Let's consider an example ...

$f(\alpha, \beta) = (1 - \alpha\beta)^2$

Def. A function $f(\alpha, \beta) : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$ is biconvex, if for each $\alpha \in \mathbb{R}^m$ the function $\alpha \mapsto f(\alpha, \beta)$ is convex and for each $\beta \in \mathbb{R}^n$ the function $\beta \mapsto f(\alpha, \beta)$ is convex. Analoguosly, a set $\mathcal{C} \subseteq \mathcal{A} \times \mathcal{B}$, for $\mathcal{A}, \mathcal{B}$ convex sets, is called biconvex, if it is convex

Mathematica: `3D plot (1-xy)^2, x in [-2,2], y in [-2,2]`
The formula is a link.

Alternate Convex Search

Block coordinate descent applied to α and β blocks

1. Initialize (α^0, β^0) at some point in the biconvex set to minimize over
2. For $t = 0, 1, 2, \ldots$
   (i) Fix $\beta = \beta^t$ and update $\alpha^{t+1} \in \underset{\alpha \in \mathcal{C}_{\beta^t}}{\arg\min} f(\alpha, \beta^t)$
   (ii) Fix $\alpha = \alpha^{t+1}$ and update $\beta^{t+1} \in \underset{\alpha \in \mathcal{C}_{\alpha^{t+1}}}{\arg\min} f(\alpha^{t+1}, \beta)$

For a function bounded from below, the algorithm converges to a partial optimum (i.e. as biconvexity, only optimal in one coordinate if the other coordinate is fixed).

Alternate Convex Search

Block coordinate descent applied to α and β blocks
1. Initialize (α^0, β^0) at some point in the biconvex set to minimize over
2. For $t = 0, 1, 2, \ldots$
(i) Fix $\beta = \beta^t$ and update $\alpha^{t+1} \in \arg\min_{\alpha \in \mathcal{C}_{\beta^t}} f(\alpha, \beta^t)$
(ii) Fix $\alpha = \alpha^{t+1}$ and update $\beta^{t+1} \in \arg\min_{\alpha \in \mathcal{C}_{\alpha^{t+1}}} f(\alpha^{t+1}, \beta)$

For a function bounded from below, the algorithm converges to a partial optimum (i.e. as biconvexity, only optimal in one coordinate if the other coordinate is fixed).

References I

Trevor Hastie, Robert Tibshirani, and Martin Wainwright (2015)
Statistical learning with sparsity: the Lasso and generalizations
CRC Press; Boca Raton, FL

Jan De Leeuw (2015)
Block Relaxation Methods in Statistics
doi.org/10.13140/RG.2.1.3101.9607 (last accessed: 02.10.18)

S. Boyd
Alternating Direction Method of Multipliers
https://web.stanford.edu/~boyd/papers/pdf/admm_slides.pdf (last accessed: 14.10.18)

Geoff Gordon and Ryan Tibshirani (2012)
Uses of Duality
https://www.cs.cmu.edu/~ggordon/10725-F12/slides/18-dual-uses.pdf (last accessed: 14.10.18)

References I

Trevor Hastie, Robert Tibshirani, and Martin Wainwright (2015)
Statistical learning with sparsity: the Lasso and generalizations
CRC Press; Boca Raton, FL

Jan De Leeuw (2015)
Block Relaxation Methods in Statistics
doi.org/10.13140/RG.2.1.3101.9607 (last accessed: 02.10.18)

S. Boyd
Alternating Direction Method of Multipliers
https://web.stanford.edu/~boyd/papers/pdf/admm_slides.pdf (last accessed: 14.10.18)

Geoff Gordon and Ryan Tibshirani (2012)
Uses of Duality
https://www.cs.cmu.edu/~ggordon/10725-F12/slides/18-dual-uses.pdf (last accessed: 14.10.18)

References II

Paul Rubin (2016)
What are the advantages of convex optimization compared to more general optimization problems?
https://www.quora.com/
What-are-the-advantages-of-convex-optimization-compared-to-m
(last accessed: 14.10.18)

References II

Paul Rubin (2016)
What are the advantages of convex optimization compared to more general optimization problems?
https://www.quora.com/
What-are-the-advantages-of-convex-optimization-compared-to-m
(last accessed: 14.10.18)

Comments . . .

Questions . . .

Suggestions . . .

Comments . . .
Questions . . .
Suggestions . . .

That's it.
Thanks for listening.

Fill out your feedback sheets!

That's it.
Thanks for listening.

Fill out your feedback sheets!