Pathwise Coordinate Descent

Presentation for PhD Group

Johan Larsson

Department of Statistics, Lund University

November 27, 2020

Overview

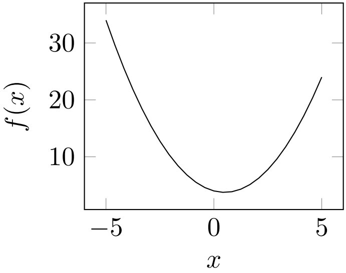
Coordinate Descent

Setting

The general problem we want to solve is

$$\underset{x \in \mathbb{R}^p}{\text{minimize}}\, f(x),$$

with $f : \mathbb{R}^p \mapsto \mathbb{R}$ convex and continuous.



standard methods: gradient descent, Newton's method, etc.

Coordinate Descent

The (very simple) idea of coordinate descent is to minimize $f(x)$ **one coordinate (variable) at the time**.

Algorithm 1 Coordinate Descent

---

while stopping criterion not reached **do**
    pick coordinate j from $\{1, 2, \ldots, p\}$
    $x_j \leftarrow \arg\min_{x_j \in \mathbb{R}} f(x)$
end while

Important: always use most recent coordinate update

Convex and Differentiable

For $f(x)$ **convex** and **differentiable**, CD always obtains the global minimum.

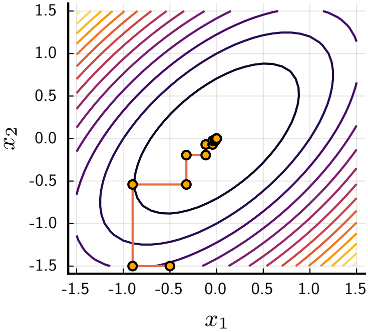


Figure 1: Coordinate descent for $f(x_1, x_2) = 5x_1^2 - 6x_1x_2 + 5x_2^2$.

Convex and Non-Differentiable

For **non-differentiable** f, however, the algorithm need not converge.

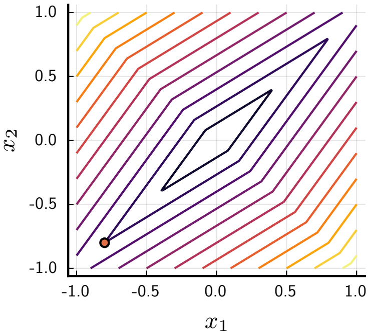


Figure 2: Coordinate descent for $f(x_1, x_2) = |x + y| + 3|x - y|$.

Convex, Non-Differentiable, but Separable

It turns out that if f is of the form

$$f(x) = g(x) + \sum_{j=1}^{p} h(x_j),$$

i.e. **separable**, then the algorithm converges even if each h_j is non-differentiable.

Convex, Non-Differentiable, but Separable

It turns out that if f is of the form

$$f(x) = g(x) + \sum_{j=1}^{p} h(x_j),$$

i.e. **separable**, then the algorithm converges even if each h_j is non-differentiable.

Why does this matter? Because many useful penalties have exactly this form!

- lasso (and elastic net)
- the nonnegative garotte
- LAD-lasso
- group lasso

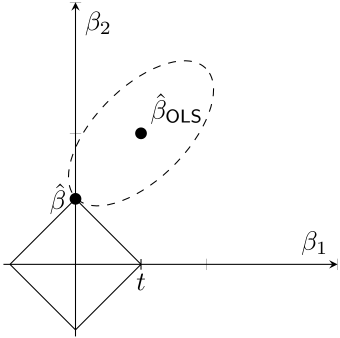
The Lasso

The Lasso

The lasso solves the following problem:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 \\ \text{subject to} \quad & \sum_{j=1}^{p} |\beta_j| \le t. \end{aligned}$$

t is kind of a **budget** on the magnitude of the coefficient vector.



Lasso in Lagrangian form

To solve the lasso, we typically transform it into an **unconstrained** optimization problem:

$$\text{minimize} \quad \frac{1}{2}\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|.$$

high values of λ: strong penalization, sparse solutions

Lasso in Lagrangian form

To solve the lasso, we typically transform it into an **unconstrained** optimization problem:

$$\text{minimize} \quad \frac{1}{2}\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|.$$

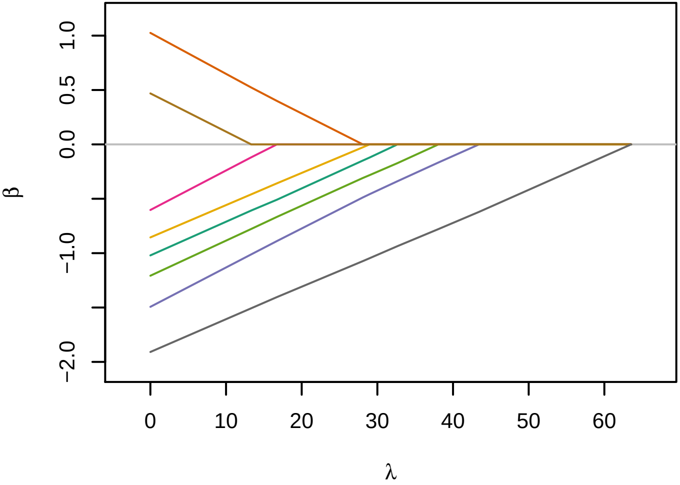
high values of λ: strong penalization, sparse solutions

Regularization Path

usually don't know λ in advance

typically select it using **grid search** (with cross-validation): start at large λ; finish at small λ (OLS)

The Lasso Path



Coordinate Descent for the Lasso: One Predictor

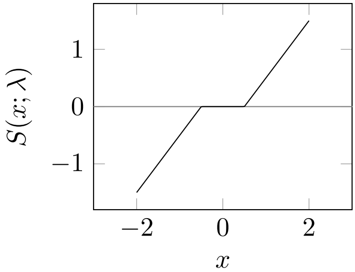
Assume x standardized, then the problem reduces to

$$\text{minimize}_\beta \quad \frac{1}{2}\left(\beta - \hat{\beta}_{\mathrm{OLS}}\right)^2 + \lambda|\beta|,$$

leading to

$$\hat{\beta}_{\mathrm{lasso}} = \mathrm{S}(\hat{\beta}_{\mathrm{OLS}}; \lambda) = \mathrm{sign}(\hat{\beta}_{\mathrm{OLS}})\left(|\hat{\beta}_{\mathrm{OLS}}| - \lambda\right)_+ .$$

$\mathrm{S}(\cdot; \lambda)$ is the **soft-thresholding** operator.



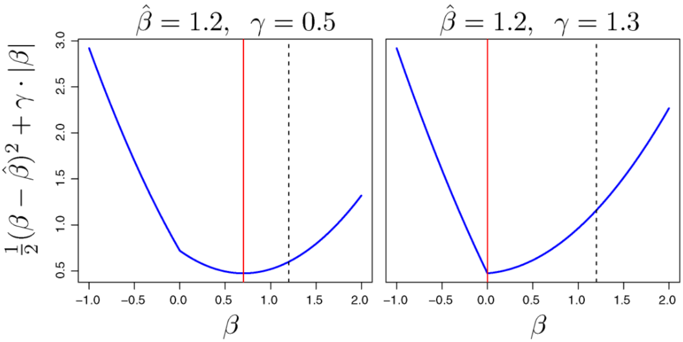


Figure 3: The solution to two lasso problems with one predictor each. The dashed line marks the (unpenalized) ordinary least-squares solution. The red line marks the lasso solution. ($\gamma := \lambda$ in our notation.)

Coordinate Descent for the Lasso: Several Predictors

Rewrite lasso objective as

$$f(\beta) = \frac{1}{2}\sum_{i=1}^{n}\left(y_i - \sum_{k\neq j} x_{ik}\beta_k - x_{ij}\beta_j\right)^2 + \lambda\sum_{k\neq j}|\beta_k| + \lambda|\beta_j|,$$

hold β_k for $k \neq j$ fixed, and minimize with respect to β_j:

$$\arg\min_{\beta_j\in\mathbb{R}} f(\beta) = \mathrm{S}\left(\sum_{i=1}^{n} x_{ij}\Big(y_i - \sum_{k\neq j} x_{ik}\beta_k\Big); \lambda\right)$$

Key: updates are cheap!

Convergence

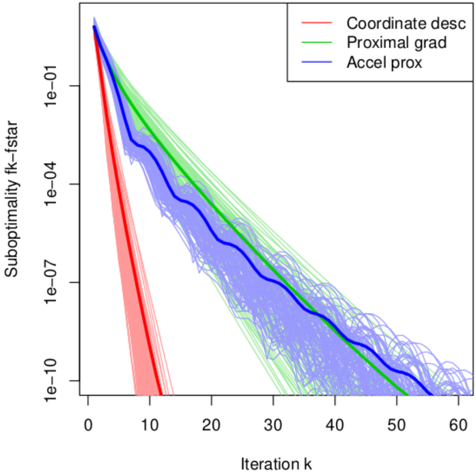


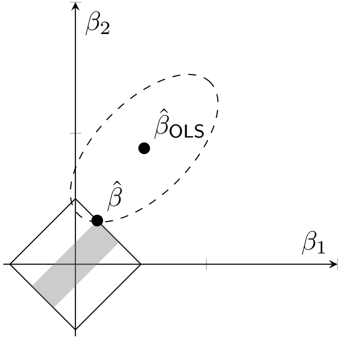
Figure 4: Coordinate descent and proximal gradient for lasso with $n = 200$, $p = 50$.

The Fused Lasso

The Fused Lasso

The Fused Lasso minimizes

$$\frac{1}{2}\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_j \sum_{j=2}^{p} |\beta_j - \beta_{j-1}|.$$



Standard CD Does not Work for the Fused Lasso

The fused lasso penalty, however, isn't **separable**, which means that convergence is no longer guaranteed!

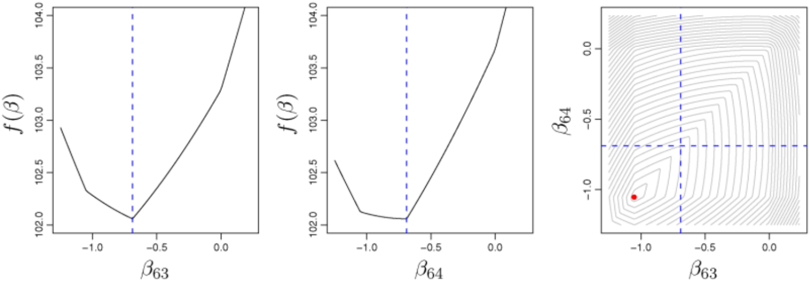


Figure 5: Coordinate descent failure for fused lasso problem.

FLSA

We begin with a special case of the fused lasso: the fused lasso signal approximator (FLSA):

$$\underset{\beta}{\text{minimize}} \left\{ \frac{1}{2} \sum_{i=1}^{n} (y_i - \beta_i)^2 + \lambda_1 \sum_{i=1}^{n} |\beta_i| + \lambda_2 \sum_{i=2}^{n} |\beta_i - \beta_{i=1}| \right\}$$

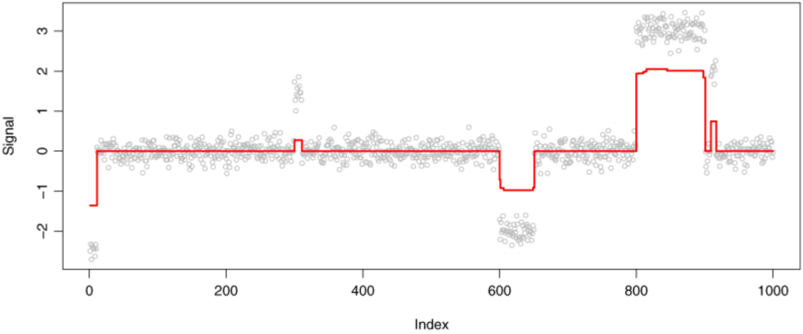


Figure 6: Fused lasso solution for signal approximation.

Solving FLSA via Three Cycles

Descent cycle

run coordinate descent for each β_j

Fusion cycle

consider fusing neighboring parameters

Algorithm 2 CD for the fused lasso (smoothing cycle).

Require: $\delta > 0$

$\lambda_2 \leftarrow 0$

repeat

    $\lambda_2 \leftarrow \lambda_2 + \delta$

    repeat

        run *descent cycle*

        run *fusion cycle*

    until no changes in coefficient estimates occur

until until λ_2 reaches target value

Descent Cycle

Assume that $\beta_i \notin \{0, \beta_{i-1}, \beta_{i+1}\}$, hold all β_k, $k \neq i$ fixed. Then,

$$\frac{\partial f(\beta)}{\partial \beta_i} = -(y_i - \beta_i) + \lambda_1 \operatorname{sign}(\beta_i) - \lambda_2 \operatorname{sign}(\beta_{i+1} - \beta_i) + \lambda_2 \operatorname{sign}(\beta_i - \beta_{i-1}),$$

which is **piecewise linear**.

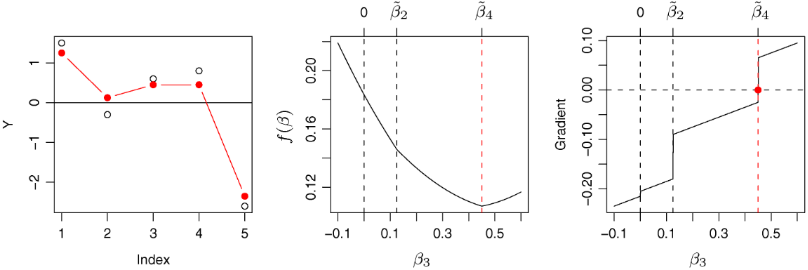
Descent Cycle

Assume that $\beta_i \notin \{0, \beta_{i-1}, \beta_{i+1}\}$, hold all β_k, $k \neq i$ fixed. Then,

$$\frac{\partial f(\beta)}{\partial \beta_i} = -(y_i - \beta_i) + \lambda_1 \operatorname{sign}(\beta_i) - \lambda_2 \operatorname{sign}(\beta_{i+1} - \beta_i) + \lambda_2 \operatorname{sign}(\beta_i - \beta_{i-1}),$$

which is **piecewise linear**. Then proceed to

1. check for a zero in each interval
2. if no zero is found, examine objective values at $\beta_i = 0, \beta_{i-1}, \beta_{i+1}$



Fusion Cycle

Descent cycle can get stuck! The fusion cycle considers fusing **two** coefficients and moving them together.

Enforce $|\beta_i - \beta_{i-1}| = 0$ by letting $\gamma \coloneqq \beta_i = \beta_{i=1}$ and minimize with respect to γ.

$$\begin{aligned}\frac{\partial f(\beta)}{\partial \gamma} &= (-y_{i-1} - \gamma) - (y_i - \gamma) \\ &\quad + 2\gamma_1 \operatorname{sign}(\gamma) - \gamma_2 \operatorname{sign}(\beta_{i+1} - \gamma) \\ &\quad + \gamma_2 \operatorname{sign}(\gamma - \beta_{i-2})\end{aligned}$$

If optimal γ decreases objective, accept the move (set $\gamma^* = \beta_i = \beta_{i-1}$).

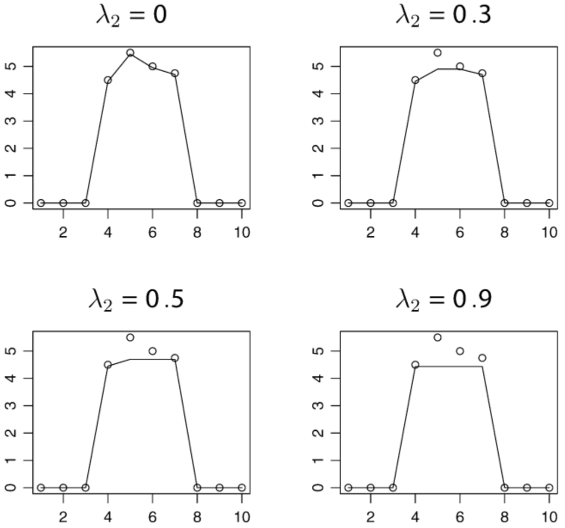


Figure 7: Smoothing cycle for the fused lasso.

Let's repeat:

Algorithm 3 CD for the fused lasso (smoothing cycle).

Require: $\delta > 0$

$\lambda_2 \leftarrow 0$

repeat

$\lambda_2 \leftarrow \lambda_2 + \delta$

repeat

run *descent cycle*

run *fusion cycle*

until no changes in coefficient estimates occur

until until λ_2 reaches target value

Assumptions

For the strategy to work, we need two assumptions.

Assumption 1

If the increments δ are sufficiently small, fusions will occur between no more than two neighboring points at a time.

This assumptions holds as long as the data have some randomness (no two adjacent y values have the same values).

Assumptions

For the strategy to work, we need two assumptions.

Assumption 1

If the increments δ are sufficiently small, fusions will occur between no more than two neighboring points at a time.

This assumptions holds as long as the data have some randomness (no two adjacent y values have the same values).

Assumption 2

Two parameters that are fused in the solution for (λ_1, λ_2) will be fused for all $(\lambda_1, \lambda_2' > \lambda_2)$.

This assumptions holds for the FLSA problem in general, **but not for the general fused lasso**.

Two-Dimensional Fused Lasso

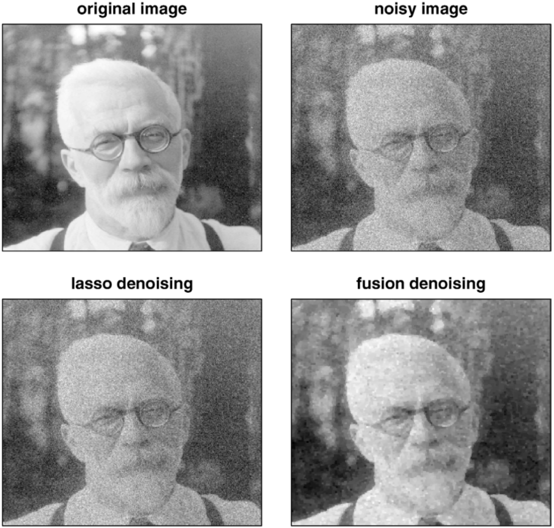


Figure 8: Lasso and 2D fused lasso denoising.

Coordinate Descent for SLOPE?

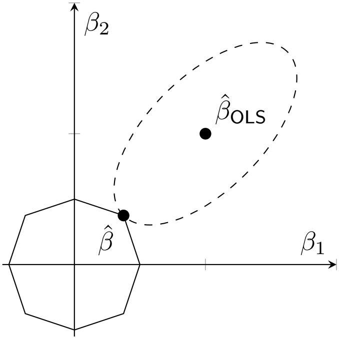
Sorted L-One Penalized Estimation (SLOPE)

SLOPE minimizes

$$g(\beta) + \sum_{i=1}^{p} \lambda_i |\beta|_{(i)}$$

where $\sum_{i=1}^{p} \lambda_i |\beta|_{(i)}$ is the **sorted ℓ_1 norm**, with

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p \geq 0, \qquad |\beta|_{(1)} \geq |\beta|_{(2)} \geq \cdots \geq |\beta|_{(p)}.$$



Wrap-Up

Wrap-Up

- CD can be extremely efficient but is not as generally applicable as standard first-order methods
- CD can be modified to handle non-separable penalty functions (fused lasso), with caveats
- CD works works extremely well with screening rules
- can we use ides from fused lasso to find a CD method for SLOPE?