Homework 3

Jimmy Hickey

Due @ 5pm on February 21, 2020

Part 1. We will construct and analyze the convergence of an MM algorithm for fitting the smoothed least absolute deviations (LAD) regression. We first set some notation: $\mathbf{x}_i \in \mathbb{R}^p, y_i \in \mathbb{R}$ for $i = 1, \ldots, n$ and $\epsilon > 0$. Throughout Part 1, assume that $n > p$ and that the design $\mathbf{X} \in \mathbb{R}^{n \times p}$ is full rank. Recall the objective function in smoothed LAD regression is

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} \sqrt{(y_i - \mathbf{x}_i^{\mathsf{T}} \boldsymbol{\beta})^2 + \epsilon}.$$

1

1. Prove that the function $f_\epsilon(u) = \sqrt{u + \epsilon}$ is concave on its domain $[0, \infty)$.

We can prove this by showing that $-f$ is convex. Take $\lambda \in [0, 1]$ and arbitrary x and y. Then,

$$\begin{aligned}
-f(\lambda x + (1-\lambda)y) &= -\sqrt{\lambda x + (1-\lambda)y + \varepsilon} \\
&= -\sqrt{\lambda x + (1-\lambda)y + \lambda\varepsilon + (1-\lambda)\varepsilon} \\
&\leq -\sqrt{\lambda x + \lambda\varepsilon} - \sqrt{(1-\lambda)y + (1-\lambda)\varepsilon} && \text{triangle inequality} \\
&= -\sqrt{\lambda}\sqrt{x + \varepsilon} - \sqrt{1-\lambda}\sqrt{y + \varepsilon} \\
&\leq -\lambda\sqrt{x + \varepsilon} - (1-\lambda)\sqrt{y + \varepsilon} && \lambda < 1
\end{aligned}$$

2

2. Fix $\tilde{u} \in [0, \infty)$. Prove that

   $$g_\epsilon(u \mid \tilde{u}) = \sqrt{\tilde{u} + \epsilon} + \frac{u - \tilde{u}}{2\sqrt{\tilde{u} + \epsilon}}$$

   majorizes $f_\epsilon(u)$.

Using the univariate majorization above enables us to construct a majorization of $\ell(\boldsymbol{\beta})$, namely

$$g(\boldsymbol{\beta} \mid \tilde{\boldsymbol{\beta}}) = \sum_{i=1}^{n} g_\epsilon(r_i(\boldsymbol{\beta})^2 \mid r_i(\tilde{\boldsymbol{\beta}})^2),$$

where $r_i(\boldsymbol{\beta}) = (y_i - \mathbf{x}_i^{\mathsf{T}} \boldsymbol{\beta})$ is the ith residual.

$$
\begin{aligned}
g_\varepsilon(\theta|\tilde{\theta}) &= \sqrt{\tilde{\theta}+\varepsilon} + \frac{\theta-\tilde{\theta}}{2\sqrt{\tilde{\theta}+\varepsilon}} \\
&= \frac{2\tilde{\theta}+2\varepsilon+\theta-\tilde{\theta}}{2\sqrt{\tilde{\theta}+\varepsilon}} \\
&= \frac{2\varepsilon+\tilde{\theta}+\theta}{2\sqrt{\tilde{\theta}+\varepsilon}} \\
&= \frac{(\theta+\varepsilon)+(\tilde{\theta}+\varepsilon)}{2\sqrt{\tilde{\theta}+\varepsilon}} \\
&= \frac{(\theta+\varepsilon)}{2\sqrt{\tilde{\theta}+\varepsilon}} + \frac{(\tilde{\theta}+\varepsilon)}{2\sqrt{\tilde{\theta}+\varepsilon}} \\
&\geq \frac{(\theta+\varepsilon)}{2\sqrt{\tilde{\theta}+\varepsilon}} \\
&\geq \sqrt{\theta+\varepsilon} \\
&\geq f(\theta)
\end{aligned}
$$

We want to minimize

$$\sqrt{\tilde{u}+\varepsilon} + \frac{u-\tilde{u}}{2\sqrt{\tilde{u}+\varepsilon}} - \sqrt{u+\varepsilon} \geq 0.$$

So, we'll take the derivative and set it equal to 0

$$
\begin{aligned}
\frac{\partial}{\partial u} = 0 &= \frac{1}{2\sqrt{\tilde{u}+\varepsilon}} - \frac{1}{2\sqrt{u+\varepsilon}} \\
\tilde{u} &= u.
\end{aligned}
$$

And then we will check that it's second derivative is positive (to ensure that it is a minimum).

$$\frac{\partial^2}{\partial u^2} = \frac{1}{4(\varepsilon+u)^{3/2}}$$

Thus,

$$g(\tilde{\theta}|\tilde{\theta}) - f(\tilde{\theta}) = 0.$$

3

3. Derive the MM update, namely write an explicit formula for

$$\boldsymbol{\beta}^+ = \arg\min_{\boldsymbol{\beta}} g(\boldsymbol{\beta} \mid \tilde{\boldsymbol{\beta}}).$$

We are looking to minimize

$$\arg\min_{\beta} \sum_{i=1}^{n} \sqrt{r_i(\tilde{\beta})^2+\varepsilon} - \frac{r_i(\tilde{\beta})^2}{2\sqrt{r_i(\tilde{\beta})+\varepsilon}} - \frac{(y_i - X_i^T\beta)^2}{2\sqrt{r_i(\tilde{\beta})^2+\varepsilon}}.$$

We can rewrite this as

$$\frac{1}{2}\sum \tilde{w}_i(y_i - x_i^T\beta)^2.$$

We can take the derivative elementwise and then recombine it.

$$\frac{\partial}{\partial \beta_j} g(\beta|\tilde{\beta}) = \sum_i \tilde{w}_i x_{ij}(x_i^T\beta - y_i).$$

Recall that

$$z_j = \sum_i \gamma_i x_{ij} \Rightarrow X^T\Gamma.$$

Then the deriviative has the following form and we can set it equal to 0.

$$X^T\tilde{W}(X\beta - Y) = X^T\widetilde{W}X\beta - X^T\widetilde{W}Y = 0$$

Where $\widetilde{W}$ is a diagonal matrix. So we need to solve.

$$X^T\widetilde{W}X\beta = X^T\widetilde{W}Y$$

4

4. What is the computational complexity of computing the MM update?

Since we cannot assume any special structure, this will take $\mathcal{O}(p^3)$ because we have to invert. However the matrix multiplication $X^T\widetilde{W}X$ will take $\mathcal{O}(np^2)$, so the overall complexity will be which of these two is bigger (depending on n and p).

5

5. Prove that $\ell(\boldsymbol{\beta})$ has a unique global minimum for all $\epsilon > 0$.

To show that we have a unique global minimum, we must show coercivity and strong convexity.

We will start with coercivity. We can see that as $\beta \to \infty$

$$\sum \sqrt{(y_i - x_i^T\beta)^2 + \varepsilon} \to \infty$$

because the $x_i^T - \beta$ term is squared.

We can show strong convexity by showing

$$\ell(\beta) - \frac{m}{2}||\beta||_2^2.$$

We can start by taking elementwise derivatives.

$$
\begin{aligned}
\frac{\partial \ell}{\partial \beta_j} &= \sum \frac{1}{2}((y_i - x_i^T\beta)^2 + \varepsilon)^{-1/2} \cdot 2(y_i - x_i^T\beta) \cdot -x_{ij} \\
&= \sum (r_i(\beta)^2 + \varepsilon)^{-1/2}(r_i(\beta))x_{ij}
\end{aligned}
$$

We can then take the second derivative.

$$
\begin{aligned}
\frac{\partial^2 \ell}{\partial \beta \partial \beta^T} &= \sum \frac{\partial}{\partial \beta^T}(r_i(\beta)^2 + \varepsilon)^{-1/2}(r_i(\beta))x_{ij} \\
&= \sum \frac{\varepsilon x_i}{(r_i(\beta)^2 + \varepsilon)^{3/2}} \frac{\partial}{\partial \beta^T} r_i(\beta) \\
&= \sum \frac{\varepsilon x_i x_i^T}{(r_i(\beta)^2 + \varepsilon)^{3/2}}
\end{aligned}
$$

This is $X^T D X$ where D is a diagonal matrix. Since X is also full column rank, we have a full row rank matrix times a diagonal matrix times a full column rank matrix, which will net us a positive definite matrix.

Since we know that the second derivative is positive definite, we know that $\nabla^2 \ell(\beta)$ has positive singular values. So we can take the smallest one to be our m. Thus, we have strong convexity and coercivity, so we have a unique global minimum.

6

6. Fix $\epsilon > 0$. Use the version of Meyer's monotone convergence theorem discussed in class to prove that the algorithm using the updates you derived in 3 converges to the unique global minimum of $\ell(\boldsymbol{\beta})$.

We need to show 4 conditions.

Part 2. MM algorithm for smooth LAD regression and Newton's method

Please complete the following steps.

Step 1: Write a function "smLAD" that implements the MM algorithm derived above for smooth LAD regression

```
#' MM algorithm for smooth LAD regression
#'
#' @param y response
#' @param X design matrix
#' @param beta Initial regression coefficient vector
#' @param epsilon smoothing parameter
#' @param max_iter maximum number of iterations
#' @param tol convergence tolerance
smLAD <- function(y,X,beta,epsilon=0.25,max_iter=1e2,tol=1e-3) {

}
```

Your function should return

- The final iterate value
- The objective function values
- The relative change in the function values
- The relative change in the iterate values

Step 2: Apply smoothed LAD regression and least squares regression on the telephone data below. Plot the two fitted lines and data points.

```
## Number of International Calls from Belgium,
## taken from the Belgian Statistical Survey,
## published by the Ministry of Economy,
##
## 73 subjects, 2 variables:
##  Year(x[i])
##  Number of Calls (y[i], in tens of millions)
##
## http://www.uni-koeln.de/themen/statistik/data/rousseeuw/
## Datasets used in Robust Regression and Outlier Detection (Rousseeuw and Leroy, 1986).
## Provided on-line at the University of Cologne.

x <- c(50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68,
  69, 70, 71, 72, 73)

y <- c(0.44, 0.47, 0.47, 0.59, 0.66, 0.73, 0.81, 0.88, 1.06, 1.20, 1.35, 1.49, 1.61,
  2.12, 11.90, 12.40, 14.20, 15.90, 18.20, 21.20, 4.30, 2.40, 2.70, 2.90)
```

Step 3: Plot the objective function values for smooth LAD evaluated at the MM iterate sequence, i.e. $\ell(\beta^{(k)})$ versus k.

For the rest of Part 2, we will investigate the effect of using the Sherman-Morrison-Woodbury identity in improving the scalability of the Newton's method algorithm for ridge LAD regression in the case when $p > n$. We seek to minimize the following objective function

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} \sqrt{(y_i - \mathbf{x}_i^\mathsf{T}\boldsymbol{\beta})^2 + \epsilon} + \frac{\lambda}{2}\|\boldsymbol{\beta}\|_2^2.$$

Let $W(\boldsymbol{\beta})$ be a $n \times n$ diagonal matrix that depends on $\boldsymbol{\beta}$.

Step 4: Write a function "newton_step_naive" that computes the solution $\Delta\boldsymbol{\beta}_{\text{nt}}$ to the linear system

$$(\lambda\mathbf{I} + \mathbf{X}^\mathsf{T}\mathbf{W}(\boldsymbol{\beta})\mathbf{X})\Delta\boldsymbol{\beta}_{\text{nt}} = \nabla\ell(\boldsymbol{\beta}).$$

Use the **chol**, **backsolve**, and **forwardsolve** functions in the base package.

```r
#' Compute Newton Step (Naive) for logistic ridge regression
#'
#' @param y response
#' @param X Design matrix
#' @param beta Current regression vector estimate
#' @param g Gradient vector
#' @param lambda Regularization parameter
#' @param epsilon smoothing parameter
newton_step_naive <- function(y, X, beta, g, lambda, epsilon=0.25) {

}
```

Your function should return the Newton step $\Delta\boldsymbol{\beta}_{\text{nt}}$.

Step 5: Write a function "newton_step_smw" that computes the Newton step using the Sherman-Morrison-Woodbury identity to reduce the computational complexity of computing the Newton step from $\mathcal{O}(p^3)$ to $\mathcal{O}(n^2 p)$. This is a reduction when $n < p$.

```r
#' Compute Newton Step (Sherman-Morrison-Woodbury) for logistic ridge regression
#'
#' @param y response
#' @param X Design matrix
#' @param beta Current regression vector estimate
#' @param g Gradient vector
#' @param lambda Regularization parameter
#' @param epsilon smoothing parameter
newton_step_smw <- function(y, X, beta, g, lambda, epsilon=0.25) {

}
```

Your function should return the Newton step $\Delta\boldsymbol{\beta}_{\text{nt}}$.

Step 6 Write a function "backtrack_descent"

```
#' Backtracking for steepest descent
#'
#' @param fx handle to function that returns objective function values
#' @param x current parameter estimate
#' @param t current step-size
#' @param df the value of the gradient of objective function evaluated at the current x
#' @param d descent direction vector
#' @param alpha the backtracking parameter
#' @param beta the decrementing multiplier
backtrack_descent <- function(fx, x, t, df, d, alpha=0.5, beta=0.9) {

}
```

Your function should return the selected step-size.

Step 7: Write functions 'fx_lad' and 'gradf_lad' to compute the objective function and its derivative for ridge LAD regression.

```
#' Objective Function for ridge LAD regression
#'
#' @param y response
#' @param X design matrix
#' @param beta regression coefficient vector
#' @param epsilon smoothing parameter
#' @param lambda regularization parameter
#' @export
fx_lad <- function(y, X, beta, epsilon=0.25,lambda=0) {

}

#' Gradient for ridge LAD regression
#'
#' @param y response
#' @param X design matrix
#' @param beta regression coefficient vector
#' @param epsilon smoothing parameter
#' @param lambda regularization parameter
#' @export
gradf_lad <- function(y, X, beta, epsilon=0.25, lambda=0) {

}
```

Step 8: Write the function "lad_newton" to estimate a ridge LAD regression model using damped Newton's method. Terminate the algorithm when half the square of the Newton decrement falls below the tolerance parameter

```
#' Damped Newton's Method for Fitting Ridge LAD Regression
#'
#' @param y response
#' @param X Design matrix
#' @param beta Initial regression coefficient vector
#' @param epsilon smoothing parameter
#' @param lambda regularization parameter
#' @param naive Boolean variable; TRUE if using Cholesky on the Hessian
#' @param max_iter maximum number of iterations
```

```r
#' @param tol convergence tolerance
lad_newton <- function(y, X, beta, epsilon=0.25,lambda=0, naive=TRUE, max_iter=1e2, tol=1e-3) {

}
```

Step 9: Perform LAD ridge regression (with $\lambda = 10$) on the following 3 data examples (y, X) using Newton's method and the naive Newton step calculation. Record the times for each using **system.time**.

```r
set.seed(12345)
## Data set 1
n <- 200
p <- 300

X1 <- matrix(rnorm(n*p),n,p)
beta0 <- matrix(rnorm(p),p,1)
y1 <- X1%*%beta0 + rnorm(n)

## Data set 2
p <- 600
X2 <- matrix(rnorm(n*p),n,p)
beta0 <- matrix(rnorm(p),p,1)
y2 <- X2%*%beta0 + rnorm(n)

## Data set 3
p <- 1200
X3 <- matrix(rnorm(n*p),n,p)
beta0 <- matrix(rnorm(p),p,1)
y3 <- X3%*%beta0 + rnorm(n)
```

Step 10: Perform LAD ridge regression (with $\lambda = 10$) on the above 3 data examples (y, X) using Newton's method and the Newton step calculated using the Sherman-Morrison-Woodbury identity. Record the times for each using **system.time**.

Step 11: Plot all six run times against p. Comment on the how the two run-times scale with p and compare it to what you know about the computational complexity for the two ways to compute the Newton update.