Lab 2

August 11, 2016

Part 1. We will investigate algorithms for iteratively solving the lasso problem or 1-norm regularized least squares problem. We first set some notation. Let $x_i \in \mathbb{R}^p, y_i \in \mathbb{R}$ for $i = 1, \dots, n$. Let $X \in \mathbb{R}^{n \times p}$ denote the matrix with x_i as its ith row. Recall that the penalized negative log-likelihood $\ell(\beta)$ can be written as follows:

$$\ell(\beta) = \frac{1}{2}\|y - Xb|_2^2 + \lambda\|\beta\|_1$$

1. Write the gradient and Hessian of $f(\beta) = \frac{1}{2}\|y - Xb|_2^2$.

Answer:

$$\begin{aligned}\nabla f(\beta) &= X^T(Xb - y)\\ \nabla^2 f(\beta) &= X^T X.\end{aligned}$$

2. What is the computational complexity for a calculating the gradient and Hessian of $f(\beta)$?

Answer:

The gradient costs $\mathcal{O}(np)$ to compute, and the Hessian costs $\mathcal{O}(np^2)$ to compute.

3. Under what condition is $\ell(\beta)$ strictly convex?

Answer: $\ell(\beta)$ is strictly convex if and only if $\nabla^2 f(\beta)$ is positive definite, which occurs when X has full rank.

4. Prove that $f(\beta)$ is L-Lipschitz differentiable with $L = \|X\|_{\text{op}}^2$.

Answer:

$$\|\nabla f(\beta) - \nabla f(\tilde{\beta})\|_2 = \|X^T X(\beta - \tilde{\beta})\|_2 \leq \|X^T X\|_{\text{op}}\|\beta - \tilde{\beta}\|_2$$

5. Prove that for all $\lambda \geq \|X^T y\|_\infty$, the lasso solution is the all zeros vector.

Answer:

β is a solution to the lasso problem with regularization parameter λ if and only if

$$X^T(y - X\beta) \in \lambda\partial\|\beta\|_1.$$

So, if $\beta = 0$, then

$$X^T y \in \lambda\partial\|0\|_1.$$

Thus, $x_j^T y \in \lambda\partial|0|$ for all j where x_j denotes the jth column of X. But recall that $\partial|0| = [-1, 1]$. Therefore, $|x_j^T y| \leq \lambda$ for all j. Or in other words, $\|X^T y\|_\infty \leq \lambda$. The practical consequence is that you only need to consider lasso solutions for $\lambda \leq \|X^T y\|_\infty$. You already know the answer without computing anything for λ larger.

Part 2. Coordinate Descent, Proximal Gradient Descent, and ADMM

Step 1: Write a function "softthreshold" that performs softthresholding elementwise on a vector.

```r
#' Soft-threshold

#' Soft-threshold elements of a vector
#'
#' \code{softthreshold} softthresholds the elements of a vector
#'
#' @param x vector to shrink
#' @param lambda regularization parameter
#' @export
softthreshold <- function(x, lambda) {

}
```

Step 2: Write a function "lasso_cd" The function should terminate either when a maximum number of iterations has been taken or if the relative change in the objective function has fallen below a tolerance

$$\frac{|f(x_k) - f(x_{k-1})|}{|f(x_{k-1})| + 1} < \text{tolerance}$$

```r
#' Lasso (Coordinate Descent)
#'
#' \code{lasso_cd} solves the lasso problem using coordinate descent.
#'
#' @param y Response variable
#' @param X design matrix
#' @param beta0 initial guess of regression parameter
#' @param lambda regularization parameter
#' @param max_iter maximum number of iterations
#' @param tol convergence tolerance
#' @export
lasso_cd <- function(y, X, beta0, lambda, max_iter=1e2, tol=1e-3) {

}
```

- The final iterate value
- The objective function values

Step 3: Write a function "lasso_pgd."

```r
#' Lasso (Proximal Gradient Descent)
#'
#' \code{lasso_pgd} solves the lasso problem using proximal gradient descent with a fixed step size
#'
#' @param y Response variable
#' @param X design matrix
#' @param beta0 initial guess of regression parameter
#' @param lambda regularization parameter
#' @param t step-size
#' @param max_iter maximum number of iterations
#' @param tol convergence tolerance
#' @export
lasso_pgd <- function(y, X, beta0, lambda, t, max_iter=1e2, tol=1e-3) {

}
```

Your function should return

- The final iterate value
- The objective function values

Step 4: Write a function "lasso_admm."

```r
#' Lasso (ADMM)
#'
#' \code{lasso_admm} solves the lasso problem using ADMM
#'
#' @param y Response variable
#' @param X design matrix
#' @param beta0 initial guess of regression parameter
#' @param lambda regularization parameter
#' @param rho parameter
#' @param max_iter maximum number of iterations
#' @param tol convergence tolerance
#' @export
lasso_admm <- function(y, X, beta0, lambda, rho=1, max_iter=1e2, tol=1e-3) {

}
```

Your function should return

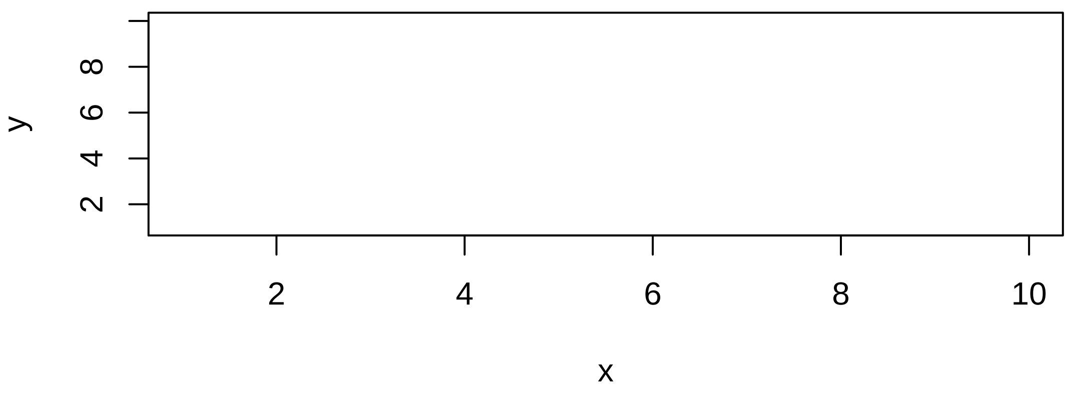
- The final iterate value
- The (primal) objective function values

Step 5: Perform lasso regression with different values of λ on the following data example (y, X). Use your answers to Part 1 to choose an appropriate fixed step-size for the proximal gradient algorithm. Plot the objective function values $\ell(\beta_k)$ versus the iteration k for all three methods. How does the lasso do at recovering the true sparse model?

```r
set.seed(12345)
n <- 100
p <- 2000

X <- matrix(rnorm(n*p),n,p)
beta0 <- matrix(0,p,1)
beta0[1:5] <- runif(5)

y <- X%*%beta0 + rnorm(n)
```



Part 3. If you have additional time,

- Try different values of ρ for ADMM. What is the effective on the convergence speed?
- Try adding in backtracking into the proximal gradient code.
- Implement FISTA, compare the run times with regular proximal gradient descent.