Homework 4: Coordinate-descent algorithm for LASSO

Attention: Because math rendering of .Rmd is not ideal, please see the enclosed pdf for correct rendering of all equations (an exact copy of this file)

Introduction

We consider the training data consisting of n samples (x_i, y_i), $x_i \in \mathbb{R}^p$ (vector of covariates for sample i), $y_i \in \mathbb{R}$ (response for sample i) supplied as matrix $X \in \mathbb{R}^{n \times p}$ and vector $Y\mathbb{R}^n$, respectively. We would like to fit linear model

$$Y = \beta_0 + X\beta + \varepsilon,$$

where the sample size n is small compared to the number of covariates p. We will use LASSO algorithm to fit this model (find β_0 and β), and use 5-fold cross-validation to select the tuning parameter.

(1) We will center Y, and center and scale X to form $\widetilde{Y}$ and $\widetilde{X}$, and fit

$$\widetilde{Y} = \widetilde{X}\widetilde{\beta} + \varepsilon.$$

Compared to original model, there is no intercept β_0 (because of centering), and $\widetilde{X}$ is such that each column satisfies $n^{-1}\widetilde{X}_j^\top \widetilde{X}_j = 1$ (because of scaling). See class notes for more.

(2) We will solve the following LASSO problem for various λ values

$$\widetilde{\beta} = \arg\min_{\beta} \left\{ (2n)^{-1} \|\widetilde{Y} - \widetilde{X}\beta\|_2^2 + \lambda \|\beta\|_1 \right\}.$$

To solve LASSO, we will use coordinate-descent algorithm with **warm starts**.

(3) We will use the K-fold cross-validation to select λ, and then find β_0, β for original X and Y based on back-scaling and back-centering (see class notes).

Starter code

The starter code for all functions with detailed description is provided in **LassoFunctions.R**. The described functions should have input/output **exactly as specified**, but you are welcome to create any additional functions you need. You are not allowed to use any outside libraries for these functions. I encourage you to work gradually through the functions and perform frequent testing as many functions rely on the previous ones.

Things to keep in mind when implementing:

- You want to make sure that parameters supplied to one function are correctly used in subsequent functions (i.e. the convergence level ε)
- You should check your code on simple examples before proceeding to the data (i.e. what happens when large lambda is supplied? What happens on toy example used in class?). I will use automatic tests to check that your code is correct on more than just the data example with different combinations of parameters.
- I will test your functions speed on large p, small n dataset (below), so you need to use coordinate-descent implementation that is optimized for this scenario to pass the speed requirements (see class notes). I recommend writing a correct code first using whichever version of algorithm you find the easiest, and then optimizing for speed (this will help avoid mistakes).
- I expect it will take you some time to figure out how to split the data for cross-validation. Keep in mind that the split should be random, in roughly equal parts, and should work correctly with any sample size n and any integer number of folds K as long as $n \geq K$.

- `glmnet` function within glmnet R package is probably the most popular LASSO solver, however the outputs are not directly comparable as glmnet does additional scaling of Y to have unit variance with 1/n formula, that is $\tilde{Y}$ for glmnet will satisfy $n^{-1}\tilde{Y}^\top\tilde{Y} = 1$ (see Details in ? `glmnet`). However, if you impose the same standardization, you may be able to use glmnet for testing. In terms of speed, **glmnet** is very highly optimized with underlying Fortran so it would be very fast compared to your code. On my computer, glmnet with 60 tuning parameters on riboflavin dataset takes around 42 milliseconds.

Application to Riboflavin data

Your implementation will be used for analysis of riboflavin data available from the R package `hdi`. The **RiboflavinDataAnalysis.R** gives starter code for loading the data and instructions. This is a high-dimensional dataset with the number of samples $n = 71$ much less than the number of predictors $p = 4088$. The goal is to predict the riboflavin production rate based on gene expression.

You will be asked to do the following:

- use **fitLASSO** function to see how the sparsity changes with λ value, and test the speed
- use **cvLASSO** function to select the tuning parameter, see how $CV(\lambda)$ changes with lambda

Grading for this assignment

Your assignment will be graded based on

- correctness *(50% of the grade)*

Take advantage of objective function values over iterations as a way to indirectly check the correctness of your function. Also recall that you know the right solution in special cases, so you can check your function in those cases (i.e. when λ is very large, or when $\lambda = 0$ and you have a nice problem)

- speed of implementations (use warm starts and coordinate descent optimized for large p settings) *(30% of the grade)*

You will get full points if your code is **at most twice slower** comparable to mine (fitLASSO with 60 tuning parameters on riboflavin data takes around 5.8 seconds on my laptop). You will loose 5 points for every fold over. You will get +5 bonus points if your **completely correct** code on 1st submission is faster than mine (median your time/median mine time < 0.9).

- code style/documentation *(10% of the grade)*

You need to comment different parts of the code so it's clear what they do, have good indentation, readable code with names that make sense. See guidelines on R style, and posted grading rubric.

- version control/commit practices *(10% of the grade)*

I expect you to start early on this assignment, and work gradually. You want to commit often, have logically organized commits with short description that makes sense. See guidelines on good commit practices, and posted grading rubric.