Stats 598z: Homework 6

Due before midnight, Sunday Apr 14

Important:

R code, tables and figures should be part of a single .pdf or .html files from `R Markdown` and `knitr`. See the class reading lists for a short tutorial.
Include R commands for all output unless explicitly told not to.
If you collaborated with anyone else, mention their names and the nature of the collaboration

1 Problem 1: LASSO [40pts]

(a) Write a function `gen_data` to generate a training dataset (X, Y). Your function should take in 4 arguments, `n, p, sparsity` and `level`. `n` is the number of observations, and `p` is their dimensionality, and generate `X` as an $n \times p$ matrix of mean-0, variance-1 Gaussian elements. The weight vector w is a p-dimensional vector, all of whose elements are 0 except the first `sparsity` elements, which all take value `level`. Generate the output vector `Y` as

$$Y_i = X_i w + \epsilon_i$$

where X_i is the ith input, and ϵ_i is Gaussian noise. Do not use for loops. [10]

(b) Write a function `lasso_loss` that takes two inputs `w` and `lambda` and returns the values of the LASSO loss function for (X, Y). You can treat (X, Y) as additional inputs, or as global variables. [5]

(c) Generate a dataset with `n=50, p = 100, sparsity=5, level=5`. [5]

(d) Use the `optim` function to find the best-values of w for the dataset above on the LASSO loss function. Set `lambda=1`. Plot the true w and the returned w. [10]

(e) Use the `optim` function to find the best-values of w and `lambda` for the dataset above on the LASSO loss function. Plot the true w and the returned w. [10]

Now we are going to directly solve the LASSO problem.

2 Problem 2: Coordinate descent [60]

1. Write a function `soft_threshold` that takes in two scalar inputs, `w` and `th`. It should output the result of soft-thresholding, so that if the absolute value of `w` is less than `th`, it returns `0`, else it returns `w` shifted by `th` towards 0. (see the slides). Plot the curve traced by this for `th` equal to 1, as you vary `w`, this should resemble the red curve in the slides. [10]

2. First we'll solve the 1-d case. Write a function `lasso1d` that takes three inputs, length-n inputs `x, y` and `lambda`, and returns a scalar weight `w` by first calculating the OLS solution (correlation coefficient) and then soft-thresholding it. See the slides. [5]

3. Given a p-dimensional weight vector, write a function `get_residual` to calculate the residual for some dimension `dim`. This function should take two inputs `w` and `dim` (and `X,Y` unless they are global), and return the residual error from trying to predict `Y` using all dimensions of `X` except `dim`. The simplest way to do this is to set `w[dim] <- 0`, and then calculate `Y_pred = X · w`. The residual is the difference between the true `Y` and `Y_pred`. [10]

4. Now we will solve for the p-dimensions `w` vector by coordinate descent. Initialize `w` to some value. Cycle through each dimension, first calculating its residual, and then updating the corresponding component of `w`. Repeat this until the change in `w` aftern *a*n entire sweep is less than some threshold. [15]

5. Try this on your earlier dataset, again with `lambda = 1`. Comment on your solution obtained this way versus the solution obtained from `optim` [10]

6. Rerun your algorithm from the first n elements of X, where n varies from 0 to 50 in steps of 5. Plot the L_2 error between the resuling w and the true w. [10]