Data Mining: 36-462/36-662
Homework 4

Due Thursday March 28 2013
(at the beginning of lecture)

Append your R code to the end of your homework. In your solutions, you should just present your R output (e.g. numbers, table, figures) or snippets of R code as you deem it appropriate. Make sure to present your results (i.e., your R ouput) in a clear and readable fashion. Careless or confusing presentations will be penalized.

Problem 1

Consider the usual linear regression setup, with response vector $y \in \mathbb{R}^n$ and predictor matrix $X \in \mathbb{R}^{n \times p}$. Let $x_1, \ldots x_p$ be the columns of X. Suppose that $\hat{\beta} \in \mathbb{R}^p$ is a minimzer of the least squares criterion

$$\|y - X\beta\|_2^2.$$

(a) Show that if $v \in \mathbb{R}^p$ is a vector such that $Xv = 0$, then $\hat{\beta} + c \cdot v$ is also a minimizer of the least squares criterion, for any $c \in \mathbb{R}$.

(b) If $x_1, \ldots x_p \in \mathbb{R}^n$ are linearly independent, then what vectors $v \in \mathbb{R}^p$ satisfy $Xv = 0$?

(c) Suppose that $p > n$. Show that there exists a vector $v \neq 0$ such that $Xv = 0$. Argue, based on part (a), that there are infinitely many linear regression estimates. Further argue that there is a variable $i \in \{1, \ldots p\}$ such that the regression coefficient of variable i can have different signs, depending on which estimate we choose. Comment on this.

Problem 2

Given a response vector $y \in \mathbb{R}^n$, predictor matrix $X \in \mathbb{R}^{n \times p}$, and tuning parameter $\lambda \geq 0$, recall the ridge regression estimate

$$\hat{\beta}^{\text{ridge}} = \underset{\beta \in \mathbb{R}^p}{\text{argmin}} \ \|y - X\beta\|_2^2 + \lambda \|\beta\|_2^2.$$

(a) Show that $\hat{\beta}^{\text{ridge}}$ is simply the vector of linear regression coefficients from regressing the response $\tilde{y} = \begin{bmatrix} y \\ 0 \end{bmatrix} \in \mathbb{R}^{n+p}$ onto the predictor matrix $\tilde{X} = \begin{bmatrix} X \\ \sqrt{\lambda} I \end{bmatrix} \in \mathbb{R}^{(n+p) \times p}$, where here $0 \in \mathbb{R}^p$, and $I \in \mathbb{R}^{p \times p}$ is the identity matrix.

(b) Show that the matrix $\tilde{X}$ always has full column-rank, i.e., its columns are always linearly independent, regardless of the columns of X. Hence argue that the ridge regression estimate is always unique, for any matrix of predictors X.

(c) Write out an explicit formula for $\hat{\beta}^{\text{ridge}}$ involving X, y, λ. Conclude that for any $a \in \mathbb{R}^p$, the estimate $a^T\hat{\beta}^{\text{ridge}}$ is a linear function of y.

(d) Now consider the estimation of $a^T\beta^*$, with β^* being the true coefficient vector. Based on what we've seen in lecture, ridge regression can have a lower MSE than linear regression. But on the last homework we proved that the linear regression estimate is the BLUE. Given that it is indeed linear (from part (c)), what does this imply about the ridge regression estimate, $a^T\hat{\beta}^{\text{ridge}}$?

(e) Let X have singular value decomposition $X = UDV^T$, where $U \in \mathbb{R}^{n\times r}, D \in \mathbb{R}^{r\times r}, V \in \mathbb{R}^{p\times r}$, U, V have orthonormal columns, and D is diagonal with elements $d_1 \geq \ldots d_r \geq 0$. Rewrite your formula for the ridge regression solution $\hat{\beta}^{\text{ridge}}$ from (c) by replacing X with UDV^T, and simplifying the expression as much as possible.

(f) Assume that

$$y = X\beta^* + \epsilon, \quad \text{with } \mathrm{E}[\epsilon] = 0, \mathrm{Cov}(\epsilon) = \sigma^2 I,$$

and let $a \in \mathbb{R}^p$. Prove that $a^T\hat{\beta}^{\text{ridge}}$ is indeed a biased estimate of $a^T\beta^*$, for any $\lambda > 0$.

Problem 3

In this problem, you will consider choosing the tuning parameters for both ridge regession and the lasso, using 10-fold cross-validation. First download the files "hw4prob3.R", "plotfuns.R", and "bstar.Rdata" from the course website. The first line of the file "hw4prob3.R" has you install the package `glmnet`. Once you have done this (i.e., once you have installed this package), you can comment this line out.

We begin with a true signal `bstar`. Although this is stored as a vector of length $p = 2500$, `bstar` really represents an image of dimension 50×50. You can plot it by calling `plot.image(bstar)`. This image is truly sparse, in the sense that 2084 of its pixels have a value of 0, while 416 pixels have a value of 1. You can think of this image as a toy version of an MRI image that we are interested in collecting.

Suppose that, because of the nature of the machine that collects the MRI image, it takes a long time to measure each pixel value individually, but it's faster to measure a linear combination of pixel values. We measure $n = 1300$ linear combinations, with the weights in the linear combination being random, in fact, independently distributed as $N(0, 1)$. These measurements are given by the entries of the vector `x %*% bstar` in our R code. Because the machine is not perfect, we don't get to observe this directly, but we see a noisy version of this. Hence, in terms of our R code, we observe `y = x %*% bstar + rnorm(n,sd=5)`. Now the question is: can we model `y` as a linear combination of the columns of `x` to recover some coefficient vector that is close to `bstar`? Roughly speaking, the answer is *yes*. Key points here: although the number of measurements $n = 1300$ is smaller than the dimension $p = 2500$, the true vector `bstar` is sparse, and the weights in a linear combination are i.i.d normal. This is the idea behind the field of *compressed sensing*.

The file "hw4prob3.R" is setup to perform ridge regression of `y` on `x`, and the lasso of `y` on `x`, with the tuning parameter for each method selected by cross-validation. You will fill in the missing pieces. It's helpful to read through the whole file to get a sense of what's to be accomplished. Try to understand all the parts, even if it doesn't seem related to what you have to fill in; this should be good practice for working with R in the future, etc.

(a) Fill in the missing parts. There are 4 missing parts marked by `# TODO`. When you're getting started, just to check that things are running without errors, it could be helpful to run the cross-validation loop for only `k=1`. It might also be helpful to read the documentation for the `glmnet` function, which you will use to perform ridge regression and the lasso.

(b) Plot the cross-validation error curves for each of ridge regression and the lasso. You can do this using the function `plot.cv`, as demonstrated by the code at the end. For both ridge regression and the lasso, what value of λ is chosen by the usual rule? What value is chosen by the one standard error rule? Which method, ridge regression or the lasso, has a smaller minimum cross-validation error?

(c) Now run ridge regression and the lasso on the entire data set `x,y`, for the same tuning parameter values as you did before. Save the objects returned by `glmnet` as `a.rid`, `a.las`, respectively. Plot the coefficient images corresponding to the values of λ chosen by the usual rule and the one standard error rule, for each of ridge regression and the lasso. For this, you'll want to use the indices that you computed in parts (a) and (b), `i1.rid`, `i1.las` (usual rule) and `i2.rid`, `i2.las` (one standard error rule), as well as the coefficients `a.rid$beta`, `a.las$beta` that you just computed. What is the difference between the usual rule and the one standard error rule for the lasso? Which image looks better? What is the difference between the ridge regression images and the lasso images? Which do you think matches the true image `bstar` more closely?

(d) Look at the squared error between the ridge regression and the lasso coefficients that you computed in (c), for both the estimate chosen by cross-validation and that from the one standard error rule, and the true coefficient vector `bstar`. What has the lowest squared error?