Handout 15

Lasso

Lasso is similar in spirit to the ridge regression except that the penalty is different. For the purpose of discussion we will assume that all the variables have been standardized and call them $Y, X_1, \ldots, X_6$. In ridge regression we have minimized

$$||Y - X\beta||^2 + k||\beta||^2 = \sum (Y_i - \beta_1 X_{i1} - \cdots \beta_p X_{ip})^2 + k \sum \beta_j^2$$

with the penalty $k||\beta||^2$ leading to an estimate $\hat{\beta}(k) = (X^T X + kI)^{-1-1} X^T Y$. In Lasso method the penalty is changed to $k \sum |\beta_j|$. So we minimize

$$||Y - X\beta||^2 + k||\beta||_1 = \sum (Y_i - \beta_1 X_{i1} - \cdots \beta_p X_{ip})^2 + k \sum |\beta_j|,$$

where $||\beta||_1 = \sum |\beta_j|$. Unlike the ridge case, it is not possible to obtain a closed form expression for Lasso. In ridge regression, usually all the estimated beta coefficients are non-zero. However, Lasso may produce some zero estimated beta coefficients, i.e., it allows variable deletion. As a matter of fact, under some mathematical conditions, it has been shown that Lasso can select the "right model" (assuming that such a thing exists) with probability close to 1 for large sample size n.

As in ridge, estimate of k can be chosen via-cross-validation. For computational efficiency and stability of estimates multifold cross-validation (usually 10-fold) is often used in the packages.

R Package "glmnet" and some important conventions:

In the package it is assumed that the quantity to be minimized is

$$(1/n) \sum (Y_i - \beta_1 X_{i1} - \cdots - \beta_p X_{ip})^2 + 2\lambda \sum |\beta_j|$$

so that $k = 2n\lambda$.

(a) Choice of the penalty parameter: ff=cv.glmnet(x,y,intercept=FALSE)

[The default is intercept=TRUE. The command "ff=cv.glmnet(x,y)" lead to fitting a model with the intercept. Here we are fitting without intercept, since we have standardized all the variables and also for comparison purposes with ridge and partial least squares.]

(b) The command "plot(ff)" will plot the cross-validate quantity against λ. The command "ff$lamda.min" will print $\hat{\lambda}$, the value of λ at which the cross-validation criterion attains its minimum. For cars93 data, $\hat{\lambda} = 0.0185$.

(c) The command "yfit=predict(ff,newx=x)" will save the fitted Y values in the array yfit when the fitted values have been obtained by using Lasso at the value $\lambda = \hat{\lambda}(=0.0185)$.

(d) The command "coef(ff)" will yield the estimated beta coefficients.

Analysis of cars93 data.

We have transformed and standardized all the variables. The cross-validation criterion (10-fold default) lead to $\hat{\lambda} = 0.0185$. The observed Y values are calculated using the lasso method with $\hat{\lambda} = 0.0185$. The plot of CV, observed Y vs. fitted Y etc are given in the plot. The method has deleted variables X_2, X_3 and X_5. Thus the estimated beta coefficients are:

$\hat{\beta}_1 = -0.14851394$, $\hat{\beta}_2 = 0, \hat{\beta}_3 = 0, \hat{\beta}_4 = 0.49343716$,$\hat{\beta}_5 = 0$ and $\hat{\beta}_6 = 0.08177461$.

Recall that stepwise regression (given in Handout 13) selected only variables X_1 and X_4.

Multifold cross-validation used in choosing λ.

The discussion here mirrors those given in Handout 10. For notation convenience, let us write $\beta_1 X_{i1} + \cdots + \beta_p X_{ip}$ as $\beta^T x_i$, i.e., $x_i^T = (X_{i1}, \ldots, X_{ip})$. In v-fold cross validatiion, the data is split into v subsets of roughly equal size so that the j^{th} group has $n_j \approx n/v$ observations. For each j, call the j^{th} group of observations the test test set and the all the other $j-1$ groups put together as the learning set. Note that the test set has about $n_j \approx n/v$ observations and the learning set has about $n - n_j \approx n - n/v$ observations.

Fix λ. Obtain the Lasso estimate of β on the basis of the $n - n_j$ observation in the learning set using the penalty λ. Let $\hat{\beta}^{(j)}(\lambda)$ be vector of estimated beta parameters. Now use this estimate to predict the observations in the test set in order to get an estimate of the prediction error, i.e.

$$(1/n_j) \sum_{i \in I_j} (Y_i - \hat{\beta}^{(j)}(\lambda)^T x_i)^2$$

is an estimate of the prediction error, where I_j is the indices of the observation in the j^{th} partition.

Now the last calculation has been done for only one test set j. If we do the similar calculations for $j = 1, ..., v$, we get v estimates of the prediction errors. A weighted average of these will lead to an estimate of the prediction error, i.e.,

$$\begin{aligned} CV^{(v)}(\lambda) &= \sum_{j=1}^{v} (n_j/n)(1/n_j) \sum_{i \in I_j} (Y_i - \hat{\beta}^{(j)}(\lambda)^T x_i)^2 \\ &= (1/n) \sum_{j=1}^{v} \sum_{i \in I_j} (Y_i - \hat{\beta}^{(j)}(\lambda)^T x_i)^2. \end{aligned}$$

A generalizations of Lasso. (Elastic Net)

One may wish to combine the ridge and Lasso in order to keep the good properties of both. One way to do this is to use a combination of ridge and Lasso penalties and this can be done as follows. Minimize

$$(1/n) \sum (Y_i - \beta_1 X_{i1} - \cdots \beta_p X_{ip})^2 + k_1 ||\beta||^2 + k_2 ||\beta||_1$$

with respect to β for given $k_1, k_2 > 0$. Then one needs to choose k_1 and k_2 by cross-validation. In the package "glmnet" this is done by a slight rewriting of the penalty $k_1||\beta||^2 + k_2||\beta||_1$ as follows

$$k_1||\beta||^2 + k_2||\beta||_1 = \lambda[(1-\alpha)||\beta||^2 + 2\alpha||\beta||_1], 0 \leq \alpha \leq 1,$$

with $\lambda = k_1 + k_2/2$ and $\alpha = k_2/(2k_1 + k_2)$. Thus the quantity to be minimized with respect to β is

$$(1/n)\sum(Y_i - \beta_1 X_{i1} - \cdots \beta_p X_{ip})^2 + \lambda[(1-\alpha)||\beta||^2 + 2\alpha||\beta||_1],\ \lambda > 0, 0 \leq \alpha \leq 1.$$

When $\alpha = 0$, this leads to the ridge regression, where $\alpha = 1$ leads to Lasso. In general, $0 < \alpha < 1$ produces a compromise of the two.

Figure 1: Lasso regression for cars93 data