HW#3 High-dimensional data analysis, Spring 2015.

102225014 Ai-Chun

1. Reproduce Table 3.3(p.63) for the LSE, Best subset and ridge (except Error). Add the result of the compound univariate estimates to the table.

   The data come from the book's inference website:
   http://statweb.stanford.edu/~tibs/ElemStatLearn/datasets/prostate.data
   And the last column of data is logical judgments of training data or testing data. We captured the training data by function "subset" in R. We found there are $n = 67$ observations.

   (i)

   LS can calculate by $\widehat{\boldsymbol{\beta}} = (\mathrm{X}^{\mathrm{T}}\mathrm{X})^{-1}\mathrm{X}^{\mathrm{T}}y$, where X is standardized such that

   $\langle \mathbf{1}, \mathbf{x}_i \rangle = 0$ and $\langle \mathbf{x}_i, \mathbf{x}_i \rangle = n, i = 1, \ldots, p$ with $\mathbf{x}_i$ be the column vector of X.

   (ii)

   Best subset approach first find the subset size k by cross-validation or AIC. Here, we assume $k = 2$. Then, we can use $\widehat{\boldsymbol{\beta}} = (\mathrm{X}_s^{\mathrm{T}}\mathrm{X}_s)^{-1}\mathrm{X}_s^{\mathrm{T}}y$ where X_s is the sub-matrix only remain the significant columns of X (include lcavol and lweight). Also, we can use the formula in HW#2. $\hat{\beta}_0 = \bar{y} - \hat{\beta}_1\bar{x}_{\cdot 1} - \hat{\beta}_2\bar{x}_{\cdot 2}$,

   $\hat{\beta}_1 = \dfrac{\mathrm{SS}_{\mathbf{x}_1\mathbf{y}}\mathrm{SS}_{\mathbf{x}_2} - \mathrm{SS}_{\mathbf{x}_2\mathbf{y}}\mathrm{SS}_{\mathbf{x}_1\mathbf{x}_2}}{\mathrm{SS}_{\mathbf{x}_1}\mathrm{SS}_{\mathbf{x}_2} - \mathrm{SS}^2_{\mathbf{x}_1\mathbf{x}_2}}$, $\hat{\beta}_2 = \dfrac{\mathrm{SS}_{\mathbf{x}_1\mathbf{y}}\mathrm{SS}_{\mathbf{x}_1\mathbf{x}_2} - \mathrm{SS}_{\mathbf{x}_2\mathbf{y}}\mathrm{SS}_{\mathbf{x}_1}}{\mathrm{SS}^2_{\mathbf{x}_1\mathbf{x}_2} - \mathrm{SS}_{\mathbf{x}_1}\mathrm{SS}_{\mathbf{x}_2}}$ to derive the coefficient estimation.

   (iii)

   Ridge regression: $\widehat{\boldsymbol{\beta}}^{\mathrm{Ridge}} = (\mathrm{X}_p^{\mathrm{T}}\mathrm{X}_p + \lambda\mathrm{I})^{-1}\mathrm{X}_p^{\mathrm{T}}y$, where X_p is from eliminating intercept of X and λ is chosen from $df(\lambda) = \sum_{j=1}^{p} \dfrac{d_j^2}{d_j^2 + \lambda} = 5$ where d_i are the singular values of X_p. And $\lambda = 23.122$ here by calculate the uniroot in R.

   And the intercept estimation is $\hat{\beta}_0^{\mathrm{Ridge}} = \bar{y}$.

   (iv)

   Compound univariate estimates:

First, estimate intercept by $\hat{\beta}_0 = \bar{y}$. Then, estimate other terms by univariate estimates $\hat{\beta}_j = \frac{\langle \mathbf{x}_j, \mathbf{y} \rangle}{\langle \mathbf{x}_j, \mathbf{x}_j \rangle}$.

Table 3.3 Estimated coefficient

Term	LS	Best Subset	Ridge	Compound univariate
Intercept	2.452	2.452	2.452	2.452
lcavol	0.711	0.774	0.432	0.878
lweight	0.290	0.349	0.251	0.581
age	-0.141		-0.046	0.272
lbph	0.210		0.168	0.315
svi	0.307		0.234	0.667
lcp	-0.286		0.003	0.586
gleason	-0.020		0.041	0.410
pgg45	0.275		0.134	0.537

Comments:

I use the training data, and the standardization procedure is same, too. Therefore, I think the difference between this table and the book's is acceptable.

The compound univariate estimator is much easier to compute, but the magnitude of coefficient estimate (without intercept term) is quite large according to other method. The Best subset keep two variables, but ridge approach state that variable "svi" may also important.

2. Reproduce Figure 3.8 (p.65)

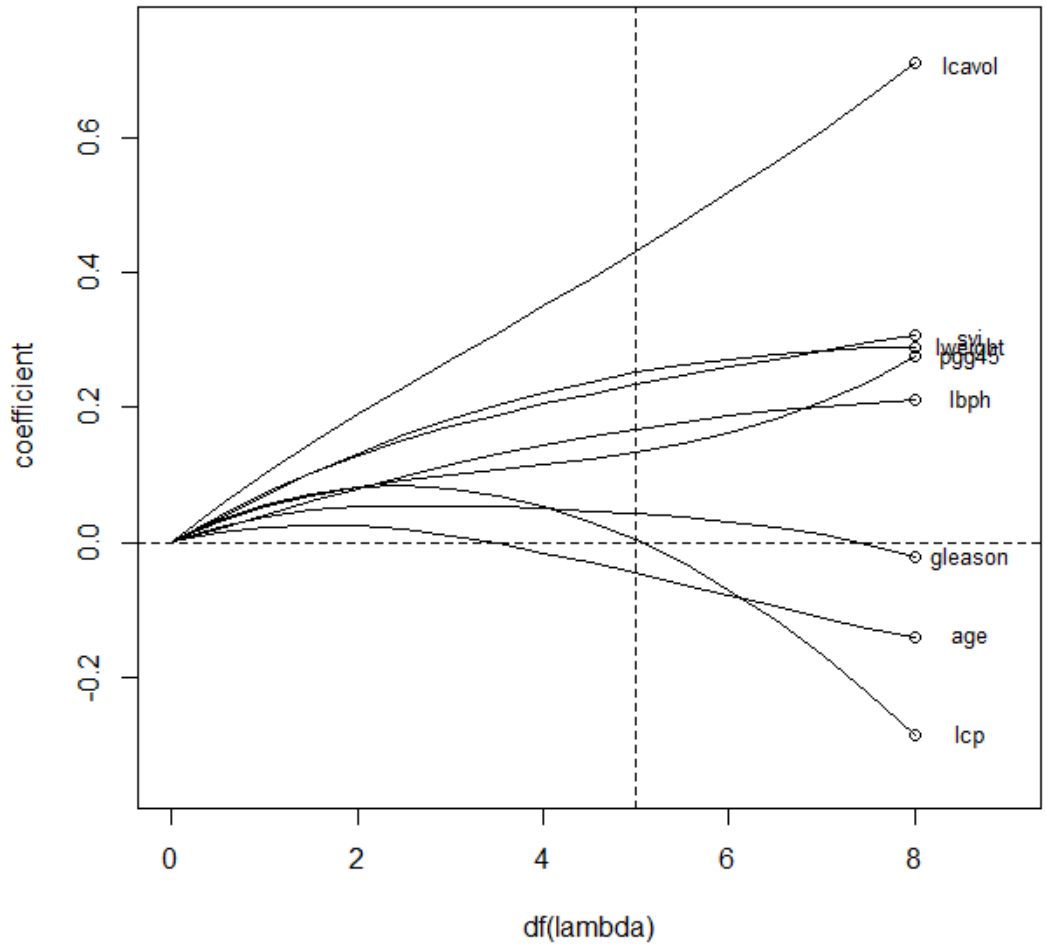

Figure 3.8 Profiles of ridge coefficients for the prostate cancer example, as parameter $df(\lambda)$ from 0 to 8.

3. Appendix – R code

```
#High-dimensional data analysis
#HW3

data =
read.table("http://statweb.stanford.edu/~tibs/ElemStatLearn/datasets/prostate.data",
h=T)
attach(data)
data_train = subset(data, train==TRUE) # capture the training data
n = dim(data_train)[1]
```

```
x = as.matrix(data_train[,1:8])
#standardization
x_std = sweep(x, 2, colMeans(x), FUN="-")
Xp = sweep(x_std, 2, sqrt((n-1)/n*apply(x, 2, var)), FUN="/")
X = cbind(1,Xp)
p = dim(Xp)[2]
y = as.vector(data_train[,9])

#LS
model = lm(y ~ X[,2]+X[,3]+X[,4]+X[,5]+X[,6]+X[,7]+X[,8]+X[,9])
beta_LS = solve(t(X)%*%X)%*%t(X)%*%y
summary(model)

#Best subset
X_s = X[,1:3]
beta_BS = solve(t(X_s)%*%X_s)%*%t(X_s)%*%y
#Use HW2
attach(data.frame(Xp))
SS1y=sum(t(lcavol)%*%y)-n*mean(lcavol)*mean(y)
SS12=sum(t(lcavol)%*%lweight)-n*mean(lcavol)*mean(lweight)
SS2y=sum(t(y)%*%lweight)-n*mean(y)*mean(lweight)
SS1= sum(lcavol^2)-n*mean(lcavol)^2
SS2= sum(lweight^2)-n*mean(lweight)^2
b1 = (SS1y*SS2-SS2y*SS12)/(SS1*SS2-SS12^2)
b2 = (SS1y*SS12-SS2y*SS1)/(SS12^2-SS1*SS2)
b0 = mean(y)-b1*mean(lcavol)-b2*mean(lweight)

#Ridge
d = svd(Xp)$d
df = 5
df_lam = function(lambda){
   s=0
   for(i in 1:p){
     s = s + d[i]^2/(d[i]^2+lambda)
   }
   return(s-df)
}
library(rootSolve)
```

```
lambda = uniroot.all(df_lam, interval=c(0,100))
beta_r = solve(t(Xp)%*%Xp+lambda*diag(x=1,p,p))%*%t(Xp)%*%y
b0 = mean(y)

#figure 3.8
#Ridge
d = svd(Xp)$d
xaxis = c(0.01, seq(0.5,8,by=0.5))    #x-axis in plot
coef = matrix(rep(0,p*length(xaxis)),p, length(xaxis))    #coefficients
lambda = numeric(length(xaxis))
  for(j in 1:length(xaxis)){
     WIL = function(lam){       #what is lambda function
       s=0
       for(i in 1:p){
         s = s + d[i]^2/(d[i]^2+lam)
       }
       return(s-xaxis[j])
     }

    lambda[j] = uniroot(WIL, interval=c(0,100^100))$root
  }

for( j    in 1:length(xaxis)){
       coef[,j] = solve(t(Xp)%*%Xp+lambda[j]*diag(x=1,p,p))%*%t(Xp)%*%y
}

plot( xaxis, coef[1,], type="l", ylim = c(-0.35, 0.75), xlim=c(0, 9),
ylab="coefficient", xlab="df(lambda)" )
lines(xaxis, coef[2,], type = "l")
lines(xaxis, coef[3,], type = "l")
lines(xaxis, coef[4,], type = "l")
lines(xaxis, coef[5,], type = "l")
lines(xaxis, coef[6,], type = "l")
lines(xaxis, coef[7,], type = "l")
lines(xaxis, coef[8,], type = "l")
coef[,11]
for(i in 1:p){    #draw points when lambda=0
  points(8, beta_LS[i+1], cex=1)
```

```
}
abline(a=0, b=0 , lty=2)
abline(v=5, lty=2)
text(8.6,beta_LS[2:(p+1)],names(data)[1:p],cex=0.8)
```