5 PQHS 471 Notes Week 5

5.1 Week 5 Day 1

ISLR Chapter 6 covers various ways to "contain" linear regression, which can be too flexible when there are many features (n/p is low or even $n < p$). Interpretation may also be an issue when there are many features in the model. The techniques introduced in this Chapter will be useful in other machine learning methods.

5.1.1 ISLR 6.1 Subset selection

For each k ($k = 1, \dots, p$), we pick the "best" model using the training set. Then we compare the p "best" models using one of the following approaches:

1. Direct estimates of test error, using a validation set or cross-validation;
2. Indirect estimates of test error, using traditional measures such as C_p, AIC, BIC, adjusted R^2.

Here, "best" means the resulting model has the best performance among all candidate models in that step according to a measure (e.g., RSS, deviance, R^2). Different measures could lead to different "best" models.

- 6.1.1: Best subset selection: Pick the "best" model among *all* models with k predictors.
- 6.1.2: Stepwise.
  - Forward: At each $k = 1, \dots, p$, select the predictor that, when added to the currect model, gives the "best" model. Works even when $n < p$.
  - Backward: At each $k = p, \dots, 1$, select the predictor that, when removed from the currect model, gives the "best" model.
  - These are greedy algorithms. But they are "guided" so that they effectively search over more models than those evaluated.

Note: At every step when comparing models to pick the "best" one, the models need to be evaluated on the same set of observations. The measures such as RSS, deviance, and R^2 are not comparable across different sets of observations. This means observations with any missing value have to be removed.

- 6.1.3: Some traditional measures for model comparison.
  - $C_p = \frac{RSS}{\hat{\sigma}_F^2} + 2d - n$. *Lower is better.* (Mallows, 1973)
    - where $\hat{\sigma}_F^2$ is from the full model and d is the number of parameters in the model being evaluated.
    - This definition is equivalent to $\frac{1}{n}(RSS + 2d\hat{\sigma}^2)$.
    - When n is very small, $\hat{\sigma}_F^2$ can be underestimated, leading to a smaller penalty on large d.
  - $AIC = -2\log(\hat{L}) + 2d$. *Lower is better.* (Akaike information criterion)
    - AIC is a penalized log-likelihood.
    - AIC is equivalent to C_p for linear regression with Gaussian errors with a *known* variance.
  - $BIC = -2\log(\hat{L}) + \log(n)d$. *Lower is better.* (Bayesian information criterion)
    - When $n > e^2 = 7.39$, which is almost always true in practice, $\log(n) > 2$ and $BIC > AIC$.
    - BIC puts more penalty on high d and tends to favor models with a smaller d than AIC.
  - $R^2_{adj} = 1 - \frac{RSS/(n-(d+1))}{TSS/(n-1)} = 1 - \frac{n-1}{n-(d+1)}(1 - R^2)$. *Higher is better.*
    - It is an adjustment of $R^2 = 1 - \frac{RSS}{TSS}$.

[Intuition about AIC: Consider two nested models where the reduced model is correct. Then approximately, $2(\log \hat{L}_F - \log \hat{L}_R) \sim \chi^2_{d_F - d_R}$. $AIC_F < AIC_R$ is equivalent to $2(\log \hat{L}_F - \log \hat{L}_R) > 2(d_F - d_R)$. If a model with d_R is the correct model, the (asymptotic) probability of wrongly selecting a richer model with d_F is 0.16 when $d_F = d_R + 1$, 0.09 when $d_F = d_R + 4$, 0.05 when $d_F = d_R + 7$.

For BIC: At $n = 100$, $\log(100) = 4.6$, if a model with d_R is the correct model, the (asymptotic) probability of wrongly selecting a richer model is 0.03 when $d_F = d_R + 1$, 0.01 when $d_F = d_R + 2$. At $n = 500$, $\log(500) = 6.2$, the probability is 0.01 when $d_F = d_R + 1$, 0.002 when $d_F = d_R + 2$.]

```r
library(ISLR)
?Hitters
names(Hitters); dim(Hitters)
```

```
row.names(Hitters)

## missing values
apply(is.na(Hitters), 2, sum)  ## only Salary variable has missing value
## Out of curiosity, I compare those with Salary information and those without
group = apply(is.na(Hitters), 1, sum)>0
par(mfrow=c(4,5))
for(i in names(Hitters)) {
    boxplot(as.numeric(Hitters[[i]]) ~ group, main=i)
}

#### Subset selection: regsubsets() in R/leaps package
library(leaps)
library(ISLR)
Hitters = na.omit(Hitters)
regfit.full = regsubsets(Salary ~ ., data=Hitters, nvmax=19)
reg.summary = summary(regfit.full)
which.min(reg.summary$bic)
plot(reg.summary$bic, xlab="Number of Variables", ylab="BIC", type='l')
points(6, reg.summary$bic[6], col="red", cex=2, pch=20)

regfit.fwd = regsubsets(Salary~.,data=Hitters,nvmax=19,method="forward")
regfit.bwd = regsubsets(Salary~.,data=Hitters,nvmax=19,method="backward")
```

5.1.2 ISLR 6.2 Shrinkage/regularization

In regularization, **standardize the features** unless there is a reason not to.

- 6.2.1 Ridge regression (also called penalized least squares, l_2 regularization):

$$\text{minimize}_{\beta_0,\beta} \sum_{i=1}^{n}(y-\beta_0-x_i^T\beta)^2+\lambda\sum_{j=1}^{p}\beta_j^2, \tag{6.5}$$

where $\lambda \geq 0$ and $\beta = (\beta_1, \ldots, \beta_p)$. When $\lambda = 0$, this is least squares. The penalty λ is a tuning parameter. In vector format,

$$\text{minimize}_{\beta_0,\beta}(\|y-\beta_0-X\beta\|^2+\lambda\|\beta\|_2^2).$$

where $\|\beta\|_2 = \sqrt{\sum_{j=1}^{p}\beta_j^2}$ is the l_2 norm of the vector β, and X is the $n \times p$ design matrix (with intercept).

Its solution is $\hat\beta_0 = \bar y$ and $\hat\beta(\lambda) = (X'X+\lambda I)^{-1}X'y$. Note that $\hat\beta(\lambda) = (S+\lambda I)^{-1}S\hat\beta^{ls}$, where $S = X'X$, and $\hat\beta^{ls}$ is the least squares solution.

Selection of λ through cross-validation.

Note: Ridge regression was originally introduced to stablize inverse calculation in least squares. (Hoerl and Kennard, 1970)

Figure 6.4 for the `Credit` data:

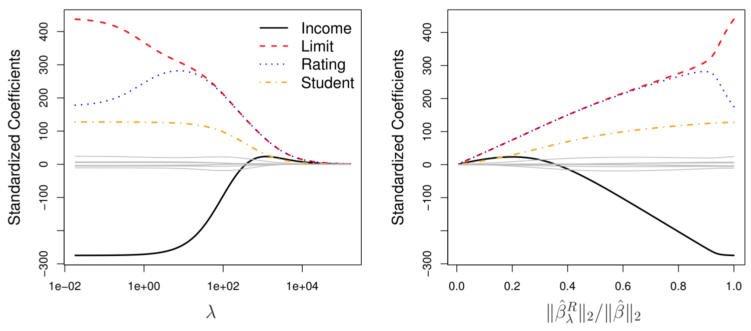
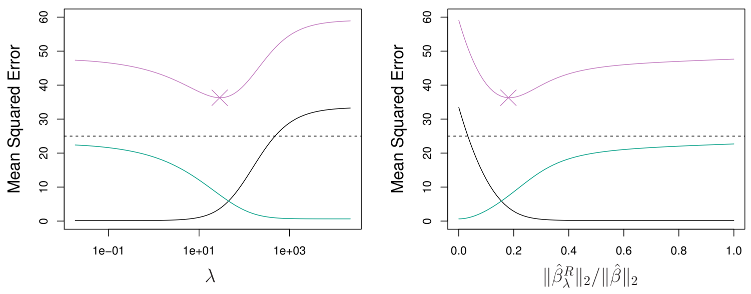


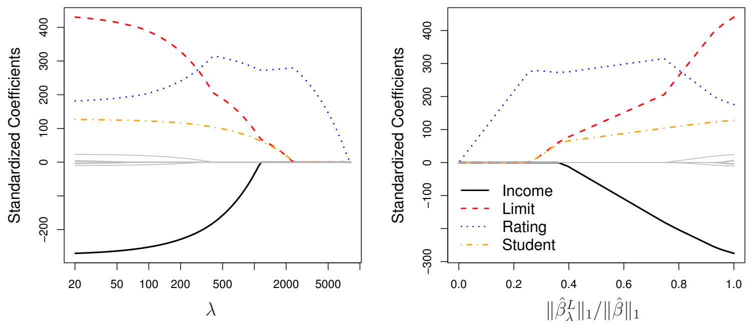
Figure 6.5: simulation ($n = 50$, $p = 45$)



- 6.2.2 The Lasso

$$\text{minimize}_\beta \sum_{i=1}^{n}(y - \beta_0 - x_i^T\beta)^2 + \lambda\sum_{j=1}^{p}|\beta_j|,$$

where $\lambda \geq 0$ and $\beta = (\beta_1, \ldots, \beta_p)$.



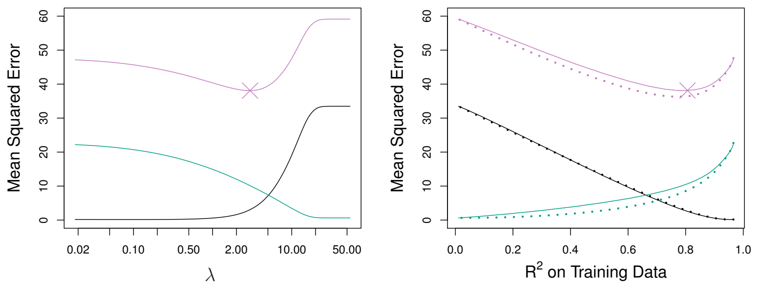
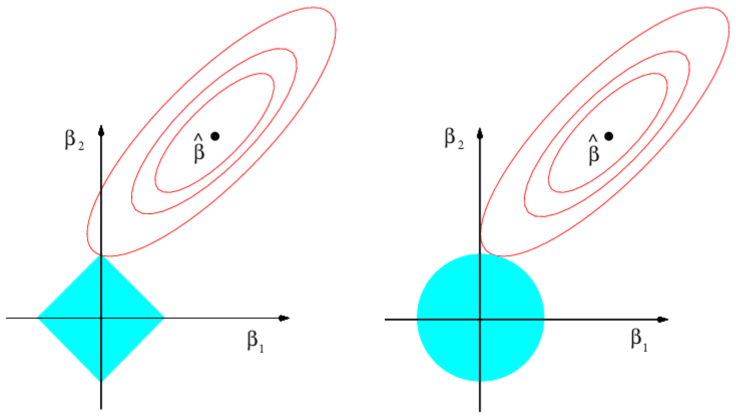


Figure 6.7 shows some intuition about ridge and lasso:



```r
#### Ridge/lasso/elastic net: glmnet() and cv.glmnet() in R/glmnet
## The alpha argument is the weight in elastic net (0<=alpha<=1).  The
## special cases are ridge regression (alpha=0) and lasso (alpha=1).
library(glmnet)
library(ISLR)
Hitters = na.omit(Hitters)
x = model.matrix(Salary ~ ., Hitters)[,-1]
y = Hitters$Salary

## ridge regression
ridge.mod = glmnet(x, y, alpha=0)
ridge.mod
names(ridge.mod)
cv.out = cv.glmnet(x, y, alpha=0)
plot(cv.out)
bestlam = cv.out$lambda.min
ridge.pred = predict(ridge.mod, s=bestlam, newx=x)

## lasso
lasso.mod = glmnet(x, y, alpha=1)
plot(lasso.mod)
cv.out = cv.glmnet(x, y, alpha=1)
names(cv.out)
cv.out$lambda
plot(cv.out)
bestlam=cv.out$lambda.min
lasso.pred=predict(lasso.mod, s=bestlam, newx=x)
lasso.coef=predict(lasso.mod, type="coefficients", s=bestlam)[1:20,]
lasso.coef
lasso.coef[lasso.coef!=0]
```

5.1.3 Assignment

1. Reading for next lecture: ISLR 6.3–6.4; HOML Chapter 4
2. ISLR Chapter 6 R Labs

5.2 Week 5 Day 2

5.2.1 ISLR 6.3

- 6.3.1: PC regression
- 6.3.2: Partial least squares

```
#### PC regression: pcr() in R/pls package
require(pls)
library(ISLR)
Hitters = na.omit(Hitters)
pcr.fit = pcr(Salary ~ ., data=Hitters, scale=T, validation="CV")
summary(pcr.fit)
validationplot(pcr.fit, val.type="MSEP")

## Partial least squares regression (not much useful): plsr() in R/pls package
pls.fit = plsr(Salary ~ ., data=Hitters, scale=T, validation="CV")
summary(pls.fit)
validationplot(pls.fit, val.type="MSEP")
```

```
####
#### PC is not regression line!
####
library(MASS)  ## for mvrnorm()

## Simulate (x,y) with some correlation and unequal variance
aa = mvrnorm(10000, mu=c(0,0), Sigma=matrix(c(4,1.3,1.3,1),2))
dim(aa)
colnames(aa)=c('x','y')

## Plot the data, add the regression line.
plot(aa)
lm1 = lm(aa[,2] ~ aa[,1])
lm1$coef
abline(lm1, col=2)  ## regression line

## Obtain the loadings of the first PC.  Draw the line from origin to
## it.  The first PC for standardized features recovers the "true"
## direction.
load1a = prcomp(aa, scale=T)$rotation[,1]
## back to the original scales for plotting
abline(0, (load1a[2]*sd(aa[,2]))/(load1a[1]*sd(aa[,1])), col=4, lwd=2)
## while the first PC for non-standardized features does not.
load1b = prcomp(aa)$rotation[,1]
abline(0, load1b[2]/load1b[1], col=3, lwd=2)

## Regression to the mean: Regress x on y to obtain the fitted line
## x=a+by, which is equivalent y=(x-a)/b.  It is DIFFERENT than the
## regression line of y on x!  This is because both models tend to
## regress to the mean.
coef2 = lm(aa[,1] ~ aa[,2])$coef
abline(-coef2[1]/coef2[2], 1/coef2[2], col=2, lty=2)


####
#### Principal components
```

```
####
#### Use prcomp(), not princomp()
## simulate three variables
N=100
x1 = 1 + rnorm(N, 0, 1)
x2 = x1*.2 + 5 + rnorm(N, 0, .2)
x3 = x1*.1 + x2*.2 + 5 + rnorm(N, 0, .2)

## feature matrices: original, centered, standardized
M=cbind(x1,x2,x3)
Mcen=cbind(x1-mean(x1), x2-mean(x2), x3-mean(x3))
Mstd=cbind((x1-mean(x1))/sd(x1), (x2-mean(x2))/sd(x2), (x3-mean(x3))/sd(x3))
t(Mstd) %*% Mstd  ## the diagonal is N-1

## visualize the data
library(rgl)
plot3d(x1,x2,x3, ylim=c(0,max(x2)), zlim=c(0,max(x3)))
plot3d(0,0,0, add=T, col=2, size=10)

plot3d(Mcen, col=3)
plot3d(0,0,0, add=T, col=2, size=10)

plot3d(Mstd, col=4)
plot3d(0,0,0, add=T, col=2, size=10)

## two ways of obtaining PCs, their loadings and SDs
a2 = prcomp(M, scale=T)  ## same as prcomp(Mstd)
s2 = svd(Mstd)

a2$sdev; s2$d/sqrt(N-1)  ## SDs for the PCs (same value)
sum((a2$sdev)^2)  ## sum of variance is p for standardized PCs
a2$rotation; s2$v  ## loadings (same value)
dim(a2$x); dim(s2$u %*% diag(s2$d))  ## PCs
head(a2$x); head(s2$u %*% diag(s2$d))  ## check the first few rows
sum(a2$x != s2$u %*% diag(s2$d))
range(a2$x - s2$u %*% diag(s2$d))
sum(round(a2$x - s2$u %*% diag(s2$d), 10))  ## the PCs are effectively same

## cumulative fraction explained by the first k PCs
(a2$sdev)^2  ## variances of the PCs
sum((a2$sdev)^2)
cumsum((a2$sdev)^2) / sum((a2$sdev)^2)  ## fractions

## PC regression
Y=rnorm(N)
summary(lm(Y ~ prcomp(M, scale=T)$x[,1:2]))
## see below for more details of pcr()
require(pls)
summary(pcr(Y ~ x1+x2+x3, scale=T, validataio="CV"))
```

5.2.2 ISLR 6.4

High dimensional data

5.2.3 HOML 4

5.2.4 Assignment

1. **Homework**: ISLR Chapter 6 Exercises ?
2. Reading for next lecture: ISLR 7.
3. ISLR Chapter 7 R Labs