**Chapter 6: Linear Model Selection and Regularization**

**1(a)**

Best Subset方法的training RSS最小，因为这个方法是从所有2p个模型中选择使training RSS最小的k个预测变量，另外两种方法包含k个预测变量的模型的training RSS大小路径依赖于模型之前选择的变量，不一定能保证training RSS是最小的。

**1(b)**

Best Subset方法可能会有最小的test RSS，如上所述，Best Subset方法是在2p个模型中选择最优的模型，应该比另外两种方法得到的模型对数据拟合得更好，从而test RSS更小。

**1(c)**

i. True.

ii. True.

iii. False.

iv. False.

v. False.

**2(a)**

iii is true. lasso yields sparse models—that is, sparse models that involve only a subset of the variables. As a result, Lasso is less flexible than Linear Regression because of less predictors, hence the bias will increase while the variance will decrease more.

**2(b)**

iii is true. As λ increases, the flexibility of the ridge regression fit decreases, leading to decreased variance but increased bias.

**2(c)**

ii is true. non-linear methods are more flexible than least squares, leading to decreased bias but increased variance.

**3(a)**

iv is true. As we increase s from 0, all coefficients increase from 0 to their least square estimate values. Training error is the maximum when s equals to 0 and it steadily decreases to the Ordinary Least Square training RSS.

**3(b)**

ii is true. When s is 0, all coefficients are 0, the model is extremely simple and has high test RSS. As we increase s, model becomes more flexible and starts to fit well on test data and so test RSS decreases. Eventually, as coefficients approach to their OLS values, they start to overfit to the training data and increase test RSS.

**3(c)**

iii is true. when s is 0, the model is less flexible and the variance is small, as we increase s, the model is more flexible and the variance steadily increase.

**3(d)**

iv is true. when s is 0, the model is less flexible and the bias is big, as we increase s, the model is more flexible and the bias steadily decrease.

**3(e)**

v is true. Irreducible error cannot be predicted using X, no matter how well we estimate f, we cannot reduce the error.

**4(a)**

iii is true. As we increase λ from 0, the model becomes less flexible, hence, it can’t fit the training data very well. The training RSS will increase.

**4(b)**

ii is true. As λ increases, the flexibility of the ridge regression fit decreases, leading to decreased variance but increased bias. When λ is small, as λ increases, the shrinkage of the ridge coefficient estimates leads to a substantial reduction in the variance of the predictions, at the expense of a slight increase in bias. But when λ is large enough, the decrease in variance due to increasing λ slows, and the shrinkage on the coefficients causes them to be significantly underestimated, resulting in a large increase in the bias. The test RSS will decrease initially, and then eventually start increasing in a U shape.

**4(c)**

iv is true. As we increase λ from 0, the model become less flexible, the variance will decrease steadily.

**4(d)**

iii is true. As we increase λ from 0, the model become less flexible, the bias will increase steadily.

**4(e)**

v is true. Irreducible error cannot be predicted using X, no matter how well we estimate f, we cannot reduce the error.

**5(a)**



**5(b)**

5(a)中的式子分别对和求导并令其等于0得到





又，从而。

**5(c)**



**5(d)**

5(c)的式子等价于



对于，可转化为



解集为线与图形的所有交点。

**6(a)**

时，(6.12)化为



令

> y = 2

> lambda = 1

> beta1 = seq(-10, 10, 0.1)

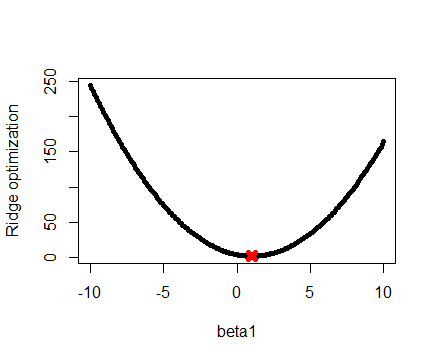
> f = (y - beta1)^2 + lambda \* beta1^2

> plot(beta1, f, pch = 20, xlab = "beta1", ylab = "Ridge optimization")

> est.beta = y/(1 + lambda)

> est.f = (y - est.beta)^2 + lambda \* est.beta^2

> points(est.beta, est.f, col = "red", pch = 4, lwd = 5, cex = est.beta)



红叉就是最小值，此时。

**6(b)**

时，(6.13)化为



令

> y = 2

> lambda = 1

> beta1\_ = seq(-10, 10, 0.1)

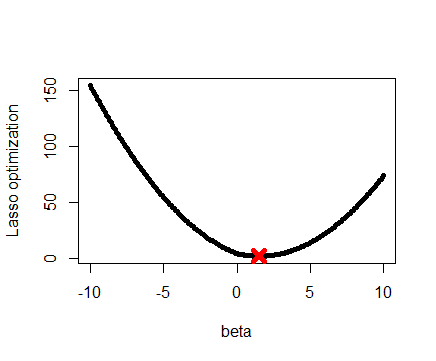
> f\_ = (y - beta1\_)^2 + lambda \* abs(beta1\_)

> plot(beta1\_, f\_, pch = 20, xlab = "beta", ylab = "Lasso optimization")

> est.beta\_ = y - lambda/2

> est.f\_ = (y - est.beta\_)^2 + lambda \* abs(est.beta\_)

> points(est.beta\_, est.f\_, col = "red", pch = 4, lwd = 5, cex = est.beta\_)



红叉就是最小值，此时。

**7(a)**

****

**7(b)**

由****

得****

**7(c)**

****

****

****

****

****

令****，得到原式等价于

****

****

**7(d)**

****

由****

得到

****

**7(e)**

****

****

****

****

****

****

****

**8(a)**

> set.seed(1)

> X = rnorm(100)

> eps = rnorm(100)

**8(b)**

>Y=1+2\*X+3\*X^2+4\*X^3+eps

**8(c)**

>data=data.frame(X,Y)

> reg.fit8=regsubsets(Y~poly(X,10),data = data,nvmax = 10)

> summary(reg.fit8)

Subset selection object

Call: regsubsets.formula(Y ~ poly(X, 10), data = data, nvmax = 10)

10 Variables (and intercept)

Forced in Forced out

poly(X, 10)1 FALSE FALSE

poly(X, 10)2 FALSE FALSE

poly(X, 10)3 FALSE FALSE

poly(X, 10)4 FALSE FALSE

poly(X, 10)5 FALSE FALSE

poly(X, 10)6 FALSE FALSE

poly(X, 10)7 FALSE FALSE

poly(X, 10)8 FALSE FALSE

poly(X, 10)9 FALSE FALSE

poly(X, 10)10 FALSE FALSE

1 subsets of each size up to 10

Selection Algorithm: exhaustive

poly(X, 10)1 poly(X, 10)2 poly(X, 10)3 poly(X, 10)4 poly(X, 10)5 poly(X, 10)6 poly(X, 10)7 poly(X, 10)8 poly(X, 10)9 poly(X, 10)10

1 ( 1 ) "\*" " " " " " " " " " " " " " " " " " "

2 ( 1 ) "\*" " " "\*" " " " " " " " " " " " " " "

3 ( 1 ) "\*" "\*" "\*" " " " " " " " " " " " " " "

4 ( 1 ) "\*" "\*" "\*" " " "\*" " " " " " " " " " "

5 ( 1 ) "\*" "\*" "\*" "\*" "\*" " " " " " " " " " "

6 ( 1 ) "\*" "\*" "\*" "\*" "\*" " " " " " " " " "\*"

7 ( 1 ) "\*" "\*" "\*" "\*" "\*" " " "\*" " " " " "\*"

8 ( 1 ) "\*" "\*" "\*" "\*" "\*" " " "\*" " " "\*" "\*"

9 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" " " "\*" "\*"

10 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"

> reg.summary8=summary(reg.fit8)

> which.min(reg.summary8$cp)

[1] 4

> which.min(reg.summary8$bic)

[1] 3

> which.min(reg.summary8$adjr2)

[1] 1

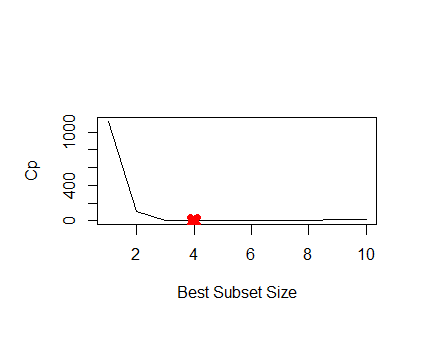
对于Cp，模型中有预测变量X1，X2，X3，X5时模型是最好的。

对于BIC，模型中有预测变量X1，X2，X3时模型是最好的。

对于调整R2，模型中有预测变量X3时模型是最好的。

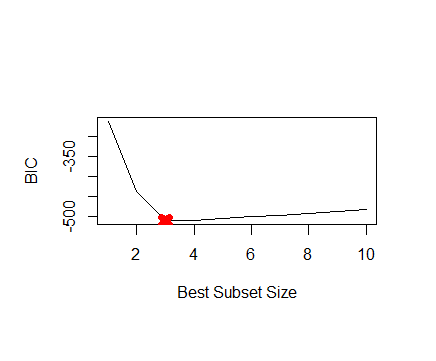
> plot(reg.summary8$cp, xlab = "Best Subset Size", ylab = "Cp", pch = 20, type = "l")

> points(4, reg.summary8$cp[4], pch = 4, col = "red", lwd = 7)



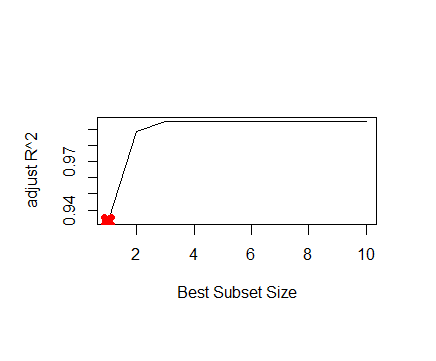
> plot(reg.summary8$bic, xlab = "Best Subset Size", ylab = "BIC", pch = 20, type = "l")

> points(3, reg.summary8$bic[3], pch = 4, col = "red", lwd = 7)



> plot(reg.summary8$adjr2, xlab = "Best Subset Size", ylab = "adjust R^2", pch = 20, type = "l")

> points(1, reg.summary8$adjr2[1], pch = 4, col = "red", lwd = 7)



> coef(reg.fit8,4)

(Intercept) poly(X, 10, raw = TRUE)1 poly(X, 10, raw = TRUE)2 poly(X, 10, raw = TRUE)3 poly(X, 10, raw = TRUE)5

1.07200775 2.38745596 2.84575641 3.55797426 0.08072292

> coef(reg.fit8,3)

(Intercept) poly(X, 10, raw = TRUE)1 poly(X, 10, raw = TRUE)2 poly(X, 10, raw = TRUE)3

1.061507 1.975280 2.876209 4.017639

>coef(reg.fit8,1)

(Intercept) poly(X, 10, raw = TRUE)3

3.437156 4.828270

**8(d)**

> which.min(reg.summary8\_for$cp)

[1] 4

> which.min(reg.summary8\_for$bic)

[1] 3

> which.min(reg.summary8\_for$adjr2)

[1] 1

> which.min(reg.summary8\_back$cp)

[1] 4

> which.min(reg.summary8\_back$bic)

[1] 3

> which.min(reg.summary8\_back$adjr2)

[1] 1

首先是Forward

对于Cp，模型中有预测变量X1，X2，X3，X5时模型是最好的。

对于BIC，模型中有预测变量X1，X2，X3时模型是最好的。

对于调整R2，模型中有预测变量X3时模型是最好的。

没有变化

然后是Backward

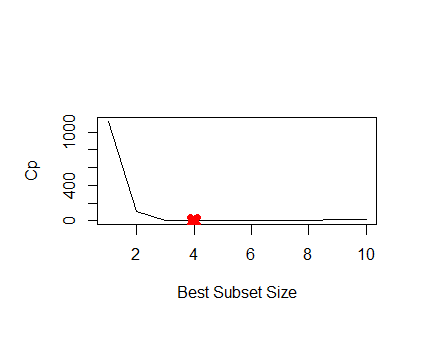
对于Cp，模型中有预测变量X1，X2，X3，X9时模型是最好的。

对于BIC，模型中有预测变量X1，X2，X3时模型是最好的。

对于调整R2，模型中有预测变量X3时模型是最好的。

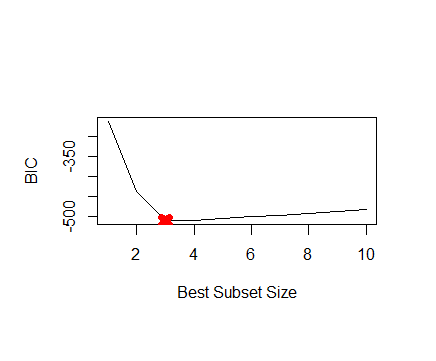
> plot(reg.summary8\_for$cp, xlab = "Best Subset Size", ylab = "Cp", pch = 20, type = "l")

> points(4, reg.summary8\_for$cp[4], pch = 4, col = "red", lwd = 7)



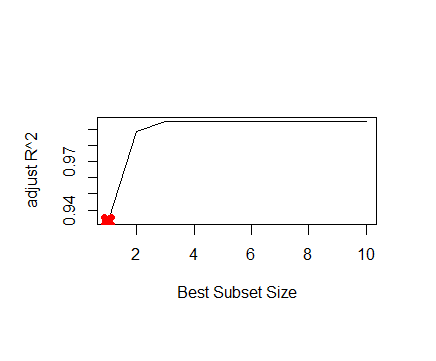
> plot(reg.summary8\_for$bic, xlab = "Best Subset Size", ylab = "BIC", pch = 20, type = "l")

> points(3, reg.summary8\_for$bic[3], pch = 4, col = "red", lwd = 7)



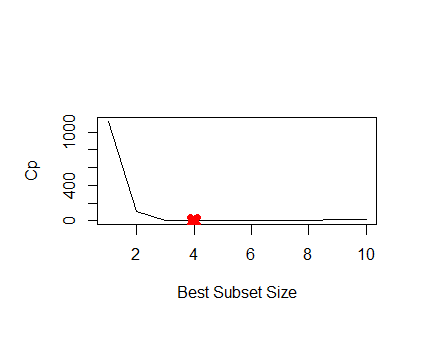
> plot(reg.summary8\_for$adjr2, xlab = "Best Subset Size", ylab = "adjust R^2", pch = 20, type = "l")

> points(1, reg.summary8\_for$adjr2[1], pch = 4, col = "red", lwd = 7)



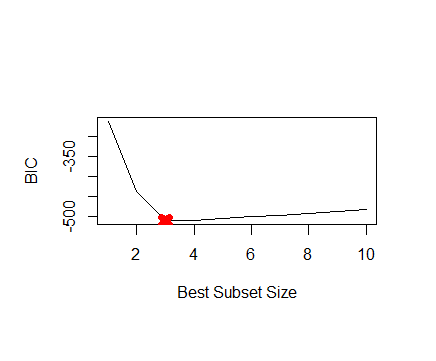
> plot(reg.summary8\_back$cp, xlab = "Best Subset Size", ylab = "Cp", pch = 20, type = "l")

> points(4, reg.summary8\_back$cp[4], pch = 4, col = "red", lwd = 7)



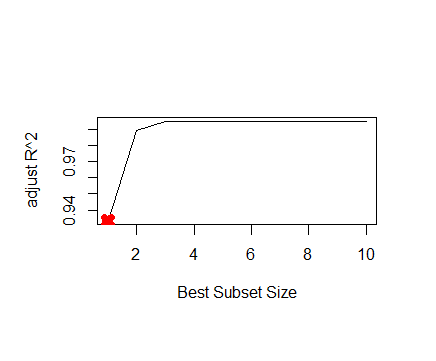
> plot(reg.summary8\_back$bic, xlab = "Best Subset Size", ylab = "BIC", pch = 20, type = "l")

> points(3, reg.summary8\_back$bic[3], pch = 4, col = "red", lwd = 7)



> plot(reg.summary8\_back$adjr2, xlab = "Best Subset Size", ylab = "adjust R^2", pch = 20, type = "l")

> points(1, reg.summary8\_back$adjr2[1], pch = 4, col = "red", lwd = 7)



> reg.summary8\_for

Subset selection object

Call: regsubsets.formula(Y ~ poly(X, 10, raw = TRUE), data = data,

nvmax = 10, method = "forward")

10 Variables (and intercept)

Forced in Forced out

poly(X, 10, raw = TRUE)1 FALSE FALSE

poly(X, 10, raw = TRUE)2 FALSE FALSE

poly(X, 10, raw = TRUE)3 FALSE FALSE

poly(X, 10, raw = TRUE)4 FALSE FALSE

poly(X, 10, raw = TRUE)5 FALSE FALSE

poly(X, 10, raw = TRUE)6 FALSE FALSE

poly(X, 10, raw = TRUE)7 FALSE FALSE

poly(X, 10, raw = TRUE)8 FALSE FALSE

poly(X, 10, raw = TRUE)9 FALSE FALSE

poly(X, 10, raw = TRUE)10 FALSE FALSE

1 subsets of each size up to 10

Selection Algorithm: forward

poly(X, 10, raw = TRUE)1 poly(X, 10, raw = TRUE)2 poly(X, 10, raw = TRUE)3 poly(X, 10, raw = TRUE)4 poly(X, 10, raw = TRUE)5 poly(X, 10, raw = TRUE)6

1 ( 1 ) " " " " "\*" " " " " " "

2 ( 1 ) " " "\*" "\*" " " " " " "

3 ( 1 ) "\*" "\*" "\*" " " " " " "

4 ( 1 ) "\*" "\*" "\*" " " "\*" " "

5 ( 1 ) "\*" "\*" "\*" " " "\*" "\*"

6 ( 1 ) "\*" "\*" "\*" " " "\*" "\*"

7 ( 1 ) "\*" "\*" "\*" " " "\*" "\*"

8 ( 1 ) "\*" "\*" "\*" " " "\*" "\*"

9 ( 1 ) "\*" "\*" "\*" " " "\*" "\*"

10 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*"

poly(X, 10, raw = TRUE)7 poly(X, 10, raw = TRUE)8 poly(X, 10, raw = TRUE)9 poly(X, 10, raw = TRUE)10

1 ( 1 ) " " " " " " " "

2 ( 1 ) " " " " " " " "

3 ( 1 ) " " " " " " " "

4 ( 1 ) " " " " " " " "

5 ( 1 ) " " " " " " " "

6 ( 1 ) " " " " "\*" " "

7 ( 1 ) "\*" " " "\*" " "

8 ( 1 ) "\*" "\*" "\*" " "

9 ( 1 ) "\*" "\*" "\*" "\*"

10 ( 1 ) "\*" "\*" "\*" "\*"

> reg.summary8\_back

Subset selection object

Call: regsubsets.formula(Y ~ poly(X, 10, raw = TRUE), data = data,

nvmax = 10, method = "backward")

10 Variables (and intercept)

Forced in Forced out

poly(X, 10, raw = TRUE)1 FALSE FALSE

poly(X, 10, raw = TRUE)2 FALSE FALSE

poly(X, 10, raw = TRUE)3 FALSE FALSE

poly(X, 10, raw = TRUE)4 FALSE FALSE

poly(X, 10, raw = TRUE)5 FALSE FALSE

poly(X, 10, raw = TRUE)6 FALSE FALSE

poly(X, 10, raw = TRUE)7 FALSE FALSE

poly(X, 10, raw = TRUE)8 FALSE FALSE

poly(X, 10, raw = TRUE)9 FALSE FALSE

poly(X, 10, raw = TRUE)10 FALSE FALSE

1 subsets of each size up to 10

Selection Algorithm: backward

poly(X, 10, raw = TRUE)1 poly(X, 10, raw = TRUE)2 poly(X, 10, raw = TRUE)3 poly(X, 10, raw = TRUE)4 poly(X, 10, raw = TRUE)5 poly(X, 10, raw = TRUE)6

1 ( 1 ) " " " " "\*" " " " " " "

2 ( 1 ) " " "\*" "\*" " " " " " "

3 ( 1 ) "\*" "\*" "\*" " " " " " "

4 ( 1 ) "\*" "\*" "\*" " " " " " "

5 ( 1 ) "\*" "\*" "\*" " " " " " "

6 ( 1 ) "\*" "\*" "\*" " " " " " "

7 ( 1 ) "\*" "\*" "\*" " " " " "\*"

8 ( 1 ) "\*" "\*" "\*" "\*" " " "\*"

9 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*"

10 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*"

poly(X, 10, raw = TRUE)7 poly(X, 10, raw = TRUE)8 poly(X, 10, raw = TRUE)9 poly(X, 10, raw = TRUE)10

1 ( 1 ) " " " " " " " "

2 ( 1 ) " " " " " " " "

3 ( 1 ) " " " " " " " "

4 ( 1 ) " " " " "\*" " "

5 ( 1 ) " " "\*" "\*" " "

6 ( 1 ) " " "\*" "\*" "\*"

7 ( 1 ) " " "\*" "\*" "\*"

8 ( 1 ) " " "\*" "\*" "\*"

9 ( 1 ) " " "\*" "\*" "\*"

10 ( 1 ) "\*" "\*" "\*" "\*"

> coef(reg.fit8\_for,4)

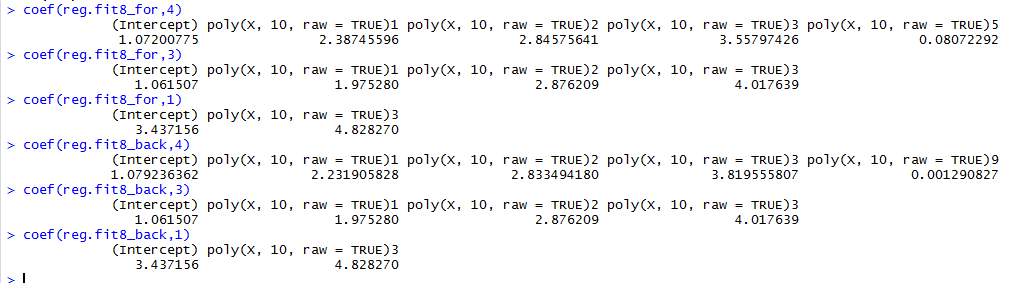
> coef(reg.fit8\_for,3)

> coef(reg.fit8\_for,1)

> coef(reg.fit8\_back,4)

> coef(reg.fit8\_back,3)

> coef(reg.fit8\_back,1)



**8(e)**

> mat = model.matrix(Y ~ poly(X, 10), data = data)[, -1]

> set.seed(1)

> View(mat)

> lasso.fit8=cv.glmnet(mat,Y,alpha = 1)

> lambda=lasso.fit8$lambda.min

> lasso.fit8\_=glmnet(mat, Y, alpha = 1)

> lasso.pred8=predict(lasso.fit8,s=lambda, type = "coefficients")

> lasso.pred8

11 x 1 sparse Matrix of class "dgCMatrix"

1

(Intercept) 4.4541625

poly(X, 10)1 107.9556155

poly(X, 10)2 44.6358314

poly(X, 10)3 59.7488786

poly(X, 10)4 0.8491130

poly(X, 10)5 1.0722064

poly(X, 10)6 .

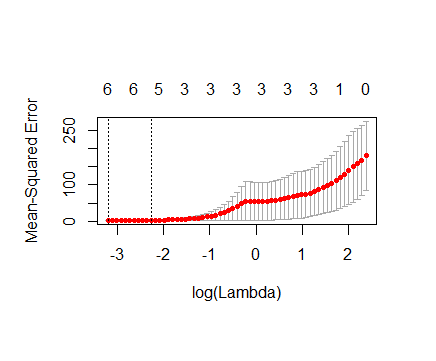
poly(X, 10)7 .

poly(X, 10)8 .

poly(X, 10)9 .

poly(X, 10)10 -0.5432475

> plot(lasso.fit8)



> Y=1+X^7+eps

> data = data.frame(X,Y)

> reg.fit8\_=regsubsets(Y ~ poly(X, 10), data = data, nvmax = 10)

> reg.fit8\_summary = summary(reg.fit8\_)

> which.min(reg.fit8\_summary$cp)

[1] 7

> which.min(reg.fit8\_summary$bic)

[1] 7

> which.min(reg.fit8\_summary$adjr2)

[1] 1

> reg.fit8\_summary

Subset selection object

Call: regsubsets.formula(Y ~ poly(X, 10), data = data, nvmax = 10)

10 Variables (and intercept)

Forced in Forced out

poly(X, 10)1 FALSE FALSE

poly(X, 10)2 FALSE FALSE

poly(X, 10)3 FALSE FALSE

poly(X, 10)4 FALSE FALSE

poly(X, 10)5 FALSE FALSE

poly(X, 10)6 FALSE FALSE

poly(X, 10)7 FALSE FALSE

poly(X, 10)8 FALSE FALSE

poly(X, 10)9 FALSE FALSE

poly(X, 10)10 FALSE FALSE

1 subsets of each size up to 10

Selection Algorithm: exhaustive

poly(X, 10)1 poly(X, 10)2 poly(X, 10)3 poly(X, 10)4 poly(X, 10)5 poly(X, 10)6 poly(X, 10)7 poly(X, 10)8 poly(X, 10)9 poly(X, 10)10

1 ( 1 ) " " " " "\*" " " " " " " " " " " " " " "

2 ( 1 ) "\*" " " "\*" " " " " " " " " " " " " " "

3 ( 1 ) "\*" " " "\*" " " "\*" " " " " " " " " " "

4 ( 1 ) "\*" "\*" "\*" " " "\*" " " " " " " " " " "

5 ( 1 ) "\*" "\*" "\*" "\*" "\*" " " " " " " " " " "

6 ( 1 ) "\*" "\*" "\*" "\*" "\*" " " "\*" " " " " " "

7 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" " " " " " "

8 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" " " " " "\*"

9 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" " " "\*" "\*"

10 ( 1 ) "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*" "\*"

对于Cp，模型中有预测变量X1，X2，X3，X4，X5，X6，X7时模型是最好的。

对于BIC，模型中有预测变量X1，X2，X3，X4，X5，X6，X7时模型是最好的。

对于调整R2，模型中有预测变量X3时模型是最好的。

> coef(reg.fit8\_,7)

(Intercept) poly(X, 10)1 poly(X, 10)2 poly(X, 10)3 poly(X, 10)4 poly(X, 10)5 poly(X, 10)6 poly(X, 10)7

5.182549 335.507160 137.987657 443.359221 82.477971 178.526321 15.705593 28.671059

> coef(reg.fit8\_,1)

(Intercept) poly(X, 10)3

5.182549 443.359221

没有拟合得特别好的。

> mat = model.matrix(Y ~ poly(X, 10), data = data)[, -1]

> lasso.fit8\_ = cv.glmnet(mat, Y, alpha = 1)

> lambda = lasso.fit8\_$lambda.min

> lambda

[1] 0.6738666

> bestmodel = glmnet(mat, Y, alpha = 1)

> predict(bestmodel, s = lambda, type = "coefficients")

11 x 1 sparse Matrix of class "dgCMatrix"

1

(Intercept) 5.182549

poly(X, 10)1 328.768495

poly(X, 10)2 131.248991

poly(X, 10)3 436.620555

poly(X, 10)4 75.739305

poly(X, 10)5 171.787655

poly(X, 10)6 8.966928

poly(X, 10)7 21.932393

poly(X, 10)8 .

poly(X, 10)9 .

poly(X, 10)10 .

模型中有预测变量X1，X2，X3，X4，X5，X6，X7时交叉验证误差最小。但是模型与原模型差距比较大。

**9(a)**

> library(ISLR)

> set.seed(1)

> sum(is.na(College))

[1] 0

> train = sample(1:dim(College)[1], dim(College)[1] / 2)

> test = -train

> College.train = College[train, ]

> College.test = College[test, ]

**9(b)**

> lm.fit9=lm(Apps~.,data = College.train)

> lm.pred9=predict(lm.fit9,College.test)

> mean((lm.pred9-College.test[,"Apps"])^2)

[1] 1108531

**9(c)**

> train.mat = model.matrix(Apps~., data=College.train)

> test.mat = model.matrix(Apps~., data=College.test)

> grid = 10 ^ seq(5, -5, length=100)

> cv.out=cv.glmnet(train.mat,College.train[, "Apps"], alpha=0, lambda=grid, thresh=1e-12)

> bestlam=cv.out$lambda.min

> rm(cv.out)

> ridge.fit9=cv.glmnet(train.mat,College.train[, "Apps"], alpha=0, lambda=grid, thresh=1e-12)

> bestlam=ridge.fit9$lambda.min

> ridge.pred9 = predict(ridge.fit9, s= bestlam, newx=test.mat)

> mean((ridge.pred9-College.test[,"Apps"])^2)

[1] 1108531

**9(d)**

> lasso.fit9=cv.glmnet(train.mat,College.train[, "Apps"], alpha=1, lambda=grid, thresh=1e-12)

> bestlam=lasso.fit9$lambda.min

> lasso.pred9 = predict(lasso.fit9, s= bestlam, newx=test.mat)

> mean((lasso.pred9-College.test[,"Apps"])^2)

[1] 1035992

> lasso.fit9\_ = glmnet(train.mat,College.train[, "Apps"], alpha=1)

> predict(lasso.fit9\_, s=bestlam, type="coefficients")

19 x 1 sparse Matrix of class "dgCMatrix"

1

(Intercept) -3.426068e+02

(Intercept) .

PrivateYes -5.384014e+02

Accept 1.567282e+00

Enroll -4.689449e-01

Top10perc 5.114049e+01

Top25perc -1.043789e+01

F.Undergrad -1.148338e-02

P.Undergrad .

Outstate -5.920048e-02

Room.Board 1.979860e-01

Books 2.354755e-02

Personal 6.834524e-03

PhD -4.870375e+00

Terminal -3.040452e+00

S.F.Ratio .

perc.alumni -2.148537e+00

Expend 3.306974e-02

Grad.Rate 3.603527e+00

**9(e)**

> pcr.fit9 = pcr(Apps~., data=College.train, scale=T, validation="CV")

> validationplot(pcr.fit9, val.type="MSEP")

>

> summary(pcr.fit9)

Data: X dimension: 388 17

Y dimension: 388 1

Fit method: svdpc

Number of components considered: 17

VALIDATION: RMSEP

Cross-validated using 10 random segments.

(Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps 14 comps 15 comps 16 comps

CV 4335 4218 2413 2427 2215 1917 1908 1911 1916 1873 1860 1860 1863 1871 1863 1815 1327

adjCV 4335 4221 2407 2428 2009 1901 1899 1902 1907 1861 1848 1849 1852 1861 1855 1778 1311

17 comps

CV 1335

adjCV 1319

TRAINING: % variance explained

1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps 14 comps 15 comps 16 comps 17 comps

X 31.216 57.68 64.73 70.55 76.33 81.30 85.01 88.40 91.16 93.36 95.38 96.94 97.96 98.76 99.40 99.87 100.00

Apps 6.976 71.47 71.58 83.32 83.44 83.45 83.46 83.47 84.53 84.86 84.98 84.98 84.99 85.24 90.87 93.93 93.97

M选为5就可以了，再增加，MSE只有少量的减小。

> pcr.pred9 = predict(pcr.fit9, College.test, ncomp=5)

> mean((data.frame(pcr.pred9)-College.test[, "Apps"])^2)

[1] 1907827

**9(f)**

> set.seed(1)

> pls.fit9 = plsr(Apps~., data=College.train, scale=T, validation="CV")

> validationplot(pls.fit9, val.type="MSEP")

>

> summary(pls.fit9)

Data: X dimension: 388 17

Y dimension: 388 1

Fit method: kernelpls

Number of components considered: 17

VALIDATION: RMSEP

Cross-validated using 10 random segments.

(Intercept) 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps 14 comps 15 comps 16 comps

CV 4335 2176 1893 1725 1613 1406 1312 1297 1285 1280 1278 1279 1282 1281 1281 1281 1281

adjCV 4335 2171 1884 1715 1578 1375 1295 1281 1271 1267 1265 1266 1269 1268 1267 1267 1268

17 comps

CV 1281

adjCV 1268

TRAINING: % variance explained

1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps 8 comps 9 comps 10 comps 11 comps 12 comps 13 comps 14 comps 15 comps 16 comps 17 comps

X 26.91 43.08 63.26 65.16 68.50 73.75 76.10 79.03 81.76 85.41 89.03 91.38 93.31 95.43 97.41 98.78 100.00

Apps 76.64 83.93 87.14 91.90 93.49 93.85 93.91 93.94 93.96 93.96 93.96 93.97 93.97 93.97 93.97 93.97 93.97

M选为7就可以了，再增加，MSE只有少量的减小。

> pls.pred9 = predict(pls.fit9, College.test, ncomp=7)

> mean((data.frame(pls.pred9)-College.test[, "Apps"])^2)

[1] 1134225

**9(g)**

> test.avg9 = mean(College.test[, "Apps"])

> lm.test9.r2 = 1 - mean((College.test[, "Apps"] - lm.pred9)^2) /mean((College.test[, "Apps"] - test.avg9)^2)

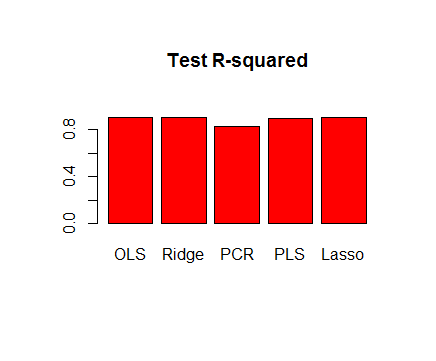
> ridge.test9.r2 = 1 - mean((College.test[, "Apps"] - ridge.pred9)^2) /mean((College.test[, "Apps"] - test.avg9)^2)

> lasso.test9.r2 = 1 - mean((College.test[, "Apps"] - lasso.pred9)^2) /mean((College.test[, "Apps"] - test.avg9)^2)

> pcr.test9.r2 = 1 - mean((College.test[, "Apps"] - data.frame(pcr.pred9))^2) /mean((College.test[, "Apps"] - test.avg9)^2)

> pls.test9.r2 = 1 - mean((College.test[, "Apps"] - data.frame(pls.pred9))^2) /mean((College.test[, "Apps"] - test.avg9)^2)

> barplot(c(lm.test9.r2, ridge.test9.r2, pcr.test9.r2, pls.test9.r2, lasso.test9.r2), col="red", names.arg=c("OLS","Ridge","PCR","PLS","Lasso"), main="Test R-squared")



除了PCR的test R2在0.8左右，其余几种方法的test R2都在0.9左右，OLS和Ridge略高一些，除了PCR，其余几种方法预测都挺准确的。

**10(a)**

> set.seed(1)

> p = 20

> n = 1000

> x = matrix(rnorm(n \* p), n, p)

> B = rnorm(p)

> B[5] = 0

> B[10] = 0

> B[15] = 0

> B[20] = 0

> eps = rnorm(p)

> y = x %\*% B + eps

**10(b)**

> train = sample(seq(1000), 100, replace = FALSE)

> y.train = y[train, ]

> y.test = y[-train, ]

> x.train = x[train, ]

> x.test = x[-train, ]

**10(c)**

> train.mat=model.matrix(y.train~.,data=data.frame(x.train,y.train))

> for (i in 1:p) {

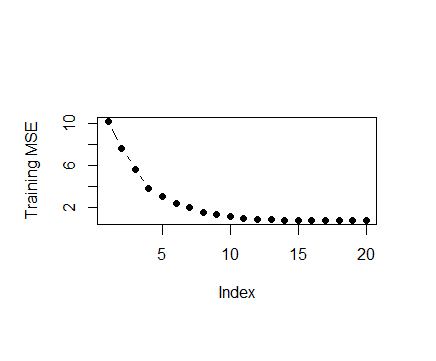
+ coefi=coef(reg.fit10,id=i)

+ pred=train.mat[,names(coefi)]%\*%coefi

+ val.errors[i]=mean((y.train-pred)^2)

+ }

> plot(val.errors, ylab = "Training MSE", pch = 19, type = "b")



**10(d)**

> test.mat=model.matrix(y.test~.,data=data.frame(x.test,y.test))

> for (i in 1:p) {

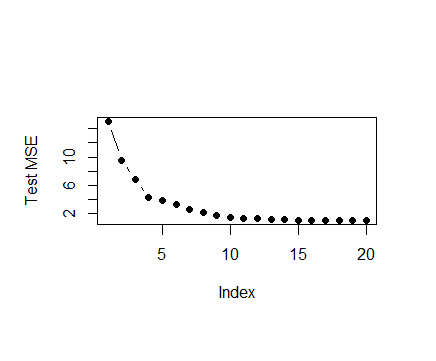
+ coefi=coef(reg.fit10,id=i)

+ pred=test.mat[,names(coefi)]%\*%coefi

+ val.errors[i]=mean((y.test-pred)^2)

+ }

> plot(val.errors, ylab = "Test MSE", pch = 19, type = "b")



**10(e)**

> which.min(val.errors)

[1] 16

即16个预测变量的模型有最小的test MSE。

**10(f)**

> coef(reg.fit10,id=16)



B5，B10，B15和B20预测为0，与原模型相符合。

**10(g)**

> x\_cols = colnames(x, do.NULL = FALSE, prefix = "X")

> View(x\_cols)

> val.errors = rep(NA, p)

> a = rep(NA, p)

> b = rep(NA, p)

> for (i in 1:p) {

+ coefi = coef(reg.fit10, id = i)

+ a[i] = length(coefi) - 1

+ b[i] = sqrt(sum((B[x\_cols %in% names(coefi)] - coefi[names(coefi) %in% x\_cols])^2) +

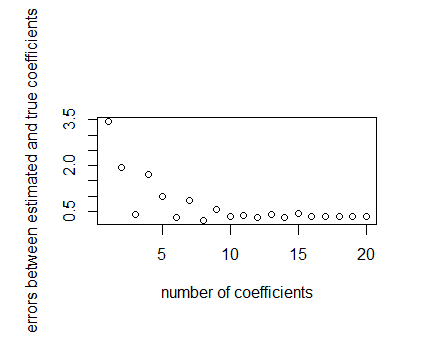
+ sum(B[!(x\_cols %in% names(coefi))])^2)

+ }

> plot(x = a, y = b, xlab = "number of coefficients", ylab = "errors between estimated and true coefficients")

> which.min(b)

[1] 8



**11(a)(b)**

> set.seed(1)

> library(MASS)

> names(Boston)

[1] "crim" "zn" "indus" "chas" "nox" "rm" "age" "dis" "rad" "tax" "ptratio" "black" "lstat" "medv"

> View(Boston)

> sum(is.na(Boston$crim))

[1] 0

> dim(Boston)

[1] 506 14

**Best subset selection**

使用K-折检验。

> predict.regsubsets =function (object ,newdata ,id){

+ form=as.formula (object$call [[2]])

+ mat=model.matrix (form ,newdata )

+ coefi =coef(object ,id=id)

+ xvars =names (coefi )

+ mat[,xvars ]%\*% coefi

+ }

> k=10

> folds=sample(1:k,nrow(Boston),replace = TRUE)

> cv.errors=matrix(NA,k,13,dimnames = list(NULL,paste(1:13)))

> View(cv.errors)

> for(j in 1:k){

+ best.fit =regsubsets (crim~.,data=Boston[folds!=j,],nvmax =13)

+ for(i in 1:13) {

+ pred=predict(best.fit ,Boston[folds ==j,],id=i)

+ cv.errors [j,i]=mean( (Boston$crim[folds ==j]-pred)^2)

+ }

+ }

> mean.cv.errors=apply(cv.errors, 2, mean)

> which.min(mean.cv.errors)

12

12

> mean.cv.errors[12]

12

41.03457

> reg.best=regsubsets(crim~.,data = Boston,nvmax = 13)

> coef(reg.best,12)

(Intercept) zn indus chas nox rm dis rad tax ptratio black lstat

16.985713928 0.044673247 -0.063848469 -0.744367726 -10.202169211 0.439588002 -0.993556631 0.587660185 -0.003767546 -0.269948860 -0.007518904 0.128120290

medv

-0.198877768

**Ridge regression**

> x=model.matrix (crim~.,Boston)[,-1]

> y=Boston$crim

> set.seed (1)

> train=sample (1: nrow(x), nrow(x)/2)

> test=(- train )

> y.test=y[test]

> ridge.mod =glmnet (x[train,],y[train],alpha =0, lambda =grid,thresh =1e-12)

> cv.out=cv.glmnet(x[train,],y[train],alpha=0)

> bestlam=cv.out$lambda.min

> bestlam

[1] 0.5982585

> ridge.predict=predict(ridge.mod,newx=x[test,],s=bestlam)

> mean((ridge.predict-y.test)^2)

[1] 38.36277

> out=glmnet(x,y,alpha = 0)

> predict(out,type = "coefficients",s=bestlam)

14 x 1 sparse Matrix of class "dgCMatrix"

1

(Intercept) 8.552102217

zn 0.032255633

indus -0.081022623

chas -0.740357484

nox -5.051300376

rm 0.326971132

age 0.002092933

dis -0.681039588

rad 0.412977817

tax 0.003737536

ptratio -0.126371712

black -0.008539983

lstat 0.142708910

medv -0.135821569

**Lasso**

> x=model.matrix (crim~.,Boston)[,-1]

> y=Boston$crim

> set.seed (1)

> train=sample (1: nrow(x), nrow(x)/2)

> test=(- train )

> y.test=y[test]

> lasso.mod =glmnet (x[train,],y[train],alpha=1,lambda =grid,thresh =1e-12)

> cv.out=cv.glmnet(x[train,],y[train],alpha=1)

> bestlam=cv.out$lambda.min

> bestlam

[1] 0.082852

> lasso.predict=predict(lasso.mod,newx=x[test,],s=bestlam)

> mean((lasso.predict-y.test)^2)

[1] 38.37381

> out=glmnet(x,y,alpha = 1)

> predict(out,type = "coefficients",s=bestlam)

14 x 1 sparse Matrix of class "dgCMatrix"

1

(Intercept) 10.49327942

zn 0.03306034

indus -0.05836746

chas -0.53788430

nox -4.86675762

rm 0.09886801

age .

dis -0.66600513

rad 0.50162998

tax .

ptratio -0.13551807

black -0.00756184

lstat 0.12086746

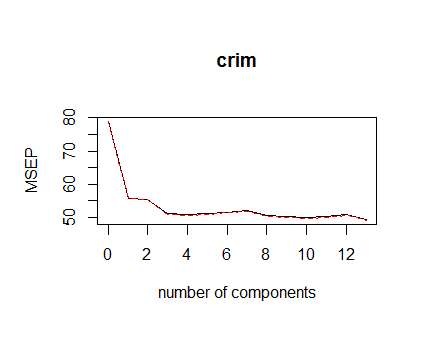
medv -0.13686346

**PCR**

> set.seed(1)

> pcr.fit=pcr(crim~.,data=Boston,subset=train,scale=TRUE,validation="CV")

> validationplot(pcr.fit,val.type="MSEP")



取M=10，

> pcr.pred=predict(pcr.fit ,x[test,],ncomp=10)

> mean((pcr.pred-y.test)^2)

[1] 39.66534

> pcr.pred=predict(pcr.fit,x[test,],ncomp=10)

> mean((pcr.pred -y.test)^2)

[1] 39.66534

> pcr.fit=pcr(y~x,scale=TRUE,ncomp=10)

> summary (pcr.fit)

Data: X dimension: 506 13

Y dimension: 506 1

Fit method: svdpc

Number of components considered: 10

TRAINING: % variance explained

1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps 8 comps 9 comps 10 comps

X 47.70 60.36 69.67 76.45 82.99 88.00 91.14 93.45 95.40 97.04

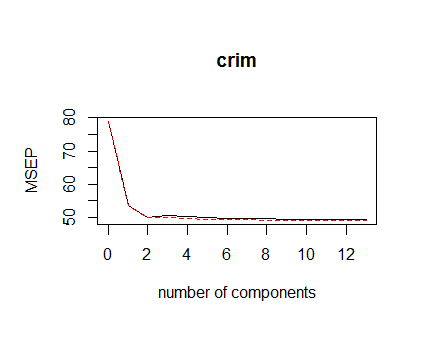
y 30.69 30.87 39.27 39.61 39.61 39.86 40.14 42.47 42.55 42.78

**PLS**

> set.seed(1)

> pls.fit=plsr(crim~.,data=Boston,subset=train,scale=TRUE,validation="CV")

> validationplot(pls.fit,val.type="MSEP")



取M=8，

> pls.pred=predict(pls.fit ,x[test,],ncomp=8)

> mean((pls.pred-y.test)^2)

[1] 39.39234

> pls.fit=plsr(y~x,scale=TRUE,ncomp=8)

> summary (pcr.fit)

Data: X dimension: 506 13

Y dimension: 506 1

Fit method: svdpc

Number of components considered: 10

TRAINING: % variance explained

1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps 8 comps 9 comps 10 comps

X 47.70 60.36 69.67 76.45 82.99 88.00 91.14 93.45 95.40 97.04

y 30.69 30.87 39.27 39.61 39.61 39.86 40.14 42.47 42.55 42.78

Ridge Regression和Lasso的test MSE较小，模型更好。

**11(c)**

Ridge Regression是的，Lasso没有，age和tax两个变量被排除了。