2020

DataMining

HW5

192STG11 우나영

1. Description

지금까지 배운 모델링 기법들은 데이터 셋이 observation 수가 feature 의 수보다 클 때 적절하다. 그러나, 실제 데이터들은 observation 수 보다 feature 수가 더 많은 high dimensional 이 많다. 이번 장에서 다루는 high dimension 은 HW1 에서 다룬 curse of dimensionality 의 개념과 다르다. HW2 에서는 data set 이 observation 의 수가 feature 수보다 크다는 것을 가정하고, feature 의 수가 커짐에 따라 생기는 variance 와 overfitting 문제를 다뤘다면, 이번 장에서 high dimensional setting 은 feature 의 수가 observation 과 비슷하거나 근소하게 작을 때를 가정한다.

위와 같은 high dimensional data 일 때, least squares 로 regression 을 적합하면 overfitting 문제가 발생한다. 이러한 문제는 logistic regression, linear discriminant analysis, 그리고 다른 classical statistical model 에서도 발생한다. Overfitting 문제가 발생한 모델은 test set 에서 성능이 떨어진다. High dimensional data 에서 이러한 문제가 발생하는 이유는 least squares regression line 이 매우 flexible 하기 때문이다. 따라서, high dimensional data 에서 이러한 문제를 해결하기 위해서 least squares 대신 forward stepwise selection, ridge regression, the lasso 그리고 principal components regression 과 같은 방법을 사용하여 linear model 을 적합할 수 있다.

High dimensional data 에서 hyper parameter 인 λ의 값이 적절할 때, ridge 와 lasso 와 같은 regularization 방법은 매우 효과적이다. 그럼에도 불구하고, 설명력이 부족한 feature 들이 model 에 추가되면 curse of dimensionality 에 따라, test error 가 증가한다. Response 와 연관있는 feature 를 추가하더라도 모델의 variance 가 bias 의 감소량보다 크게 증가한다. 따라서, 만약 feature 가 response 에 대한 설명력이 높다면 추가하는 것이 model 의 성능을 개선시키지만, 설명력이 낮다면, variance 가 증가하여 성능이 떨어지게 된다. 또한, high dimensional data 로 적합한 모델은 overfitting 이 생겨 residual 이 0 인 경우가 빈번하게 발생하기 때문에, sum of squared errors, p-values, R^2 와 같이 기존의 모델 평가 measure 를 절대 사용해서는 안된다. 대신에, test set 의 MSE 또는 R^2 또는 cross-validation 을 이용한 measure 를 사용해야한다.

2. Implementation

Lab 6.5~6.7 에서 least squares 의 대안으로 제시된 기법인 Subset Selection method, Regularization(ridge and lasso regression)그리고 Dimension reduction(PCR and PLS regression) R 코드로 실습한다. 이를 바탕으로 예제 1, 2, 5, 9 그리고 11 번을 풀 것이다.

1. We perform best subset, forward stepwise, and backward stepwise selection on a single data set. For each approach, we obtain p+1 models, containing 0, 1, 2, …p predictors. Explain your answers:

(a) Which of the three models with k predictors has the smallest training RSS?

best subset 은 모든 가능한 모델 중 RSS 값을 작게 만드는 predictor 의 조합을 선택하므로 best subset selection 을 이용한 k predictor 모델의 training RSS 가 가장 작을 것이다.

(b) Which of the three models with k predictors has the smallest test RSS?

알 수 없다. Best subset selection 으로 선택한 모델은 training 에서 overfitting 이 발생할 가능성이 높아 test 에서 성능이 낮은 가능성이 높다.

(c) True or False:

i. The predictors in the k-variable model identified by forward stepwise are a subset of the predictors in the (k+1)-variable model identified by forward stepwise selection.

TRUE, forward 로 구한 k-variable model 에서 predictor 하나 추가한 것이 forward 로 구한 k+1 variable model 이다.

ii. The predictors in the k-variable model identified by backward stepwise are a subset of the predictors in the (k+1)-variable model identified by backward stepwise selection.

TRUE, backward 로 구한 k-variable model 에서 predictor 하나 추가한 것이 backward 로 구한 k+1 variable model 이다.

iii. The predictors in the k-variable model identified by backward stepwise are a subset of the predictors in the (k+1)-variable model identified by forward stepwise selection.

FALSE, backward 와 forward 로 구한 모델의 상관관계 없다.

iv. The predictors in the k-variable model identified by forward stepwise are a subset of the predictors in the (k+1)-variable model identified by backward stepwise selection.

FALSE, backward 와 forward 로 구한 모델의 상관관계 없다.

v. The predictors in the k-variable model identified by best subset are a subset of the predictors in the (k+1)-variable model identified by best subset selection.

FALSE, best subset 은 주어진 predictor 개수의 모델 중 가장 모델 성능이 좋은 predictor 의 조합을 구하는 것이다. 따라서, best subset 으로 구한 k-variable 모델이 꼭 best subset 으로 구한 (k+1)-variable 모델의 subset 이 아닐 수 있다.

2. For parts (a) through (c), indicate which of i. through iv. is correct. Justify your answer.

(a) The lasso, relative to least squares, is:

iii. Less flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance..

=> Lasso 는 least squares 보다 flexible 하지 않으며 variance 를 줄임으로써 prediction accuracy 를 개선한다.

(b) The ridge, relative to least squares, is:

iii. Less flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance.

=> Ridge 도 least squares 보다 flexible 하지 않으며 variance 를 줄임으로써 prediction accuracy 를 개선한다.

(c) Non linear methods relative to least squares, is:

ii. More flexible and hence will give improved prediction accuracy when its increase in variance is less than its decrease in bias.

=> Non-linear method 는 least squares 보다 flexible 하며 bias 를 줄임으로써 prediction accuracy 를 개선한다.

5. It is well-known that ridge regression tends to give similar coefficient values to correlated variables, whereas the lasso may give quite different coefficient values to correlated variables. We will now explore this property in a very simple setting.

Suppose that $\mathrm{n} = 2, \mathrm{p} = 2, x_{11} = x_{12}, x_{21} = x_{22}$. Furthermore, suppose that $y_1 + y_2 = 0$ and $x_{11} + x_{21} = 0$ and $x_{12} + x_{22} = 0$, so that the estimate for the intercept in a least squares, ridge regression, or lasso model is zero: $\widehat{\beta_0} = 0$.

$$\text{model} : y_i = \beta_1 x_{1i} + \beta_2 x_{2i} + \epsilon_i \; where \; i = 1, 2$$

$$\begin{aligned}\text{RSS} &= \big(y_1 - (\beta_1 x_{11} + \beta_2 x_{12})\big)^2 + \big(y_2 - (\beta_1 x_{21} + \beta_2 x_{22})\big)^2 \\ &= [y_1 - (\beta_1 x_{11} + \beta_2 x_{11})]^2 + [y_2 - (\beta_1 x_{21} + \beta_2 x_{21})]^2 \\ &= [y_1 - (\beta_1 + \beta_2) x_{11}]^2 + [y_1 - (\beta_1 + \beta_2) x_{21}]^2 \\ &= [y_1 - (\beta_1 + \beta_2) x_{11}]^2 + [-y_1 - (\beta_1 + \beta_2)(-x_{11})]^2 \\ &= 2[y_1 - (\beta_1 + \beta_2) x_{11}]^2\end{aligned}$$

(a) Write out the ridge regression optimization problem in this setting.

$$\begin{aligned}\widehat{\beta^R} &= \underset{\beta}{\text{argmin}}\big[\text{RSS} + \lambda\big({\beta_1}^2 + {\beta_2}^2\big)\big] \\ &= \underset{\beta}{\text{argmin}}\big[2[y_1 - (\beta_1 + \beta_2) x_{11}]^2 + \lambda\big({\beta_1}^2 + {\beta_2}^2\big)\big]\end{aligned}$$

(b) Argue that in this setting, the ridge coefficient estimates satisfy $\widehat{\beta_1} = \widehat{\beta_2}$.

$\widehat{\beta^R}$ 을 β_1 과 β_2 각각에 대해 편미분 해준 후 0 으로 만드는 값으로 각각의 ridge coeffcient 로 구해주면 두개의 연립 정식이 나온다.

$$\frac{\partial \widehat{\beta^R}}{\partial \beta_1} = (4x_{11}^2 + 2\lambda)\beta_1 + 4x_{11}^2\beta_2^2 - 4y_1 x_{11} = 0$$

$$\frac{\partial \widehat{\beta^R}}{\partial \beta_2} = (4x_{11}^2 + 2\lambda)\beta_2 + 4x_{11}^2\beta_1^2 - 4y_1 x_{11} = 0$$

연립 방정식 계산 결과, $\widehat{\beta_1} = \widehat{\beta_2}$임을 알 수 있다.

(c) Write out the lasso optimization problem in this setting.

$$\begin{aligned}\widehat{\beta^L} &= \underset{\beta}{\text{argmin}}[\text{RSS} + \lambda(|\beta_1| + |\beta_2|)] \\ &= \underset{\beta}{\text{argmin}}[2x_{11}^2(\beta_1 + \beta_2)^2 - 4y_1 x_{11}(\beta_1 + \beta_2) + \lambda(|\beta_1| + |\beta_2|)]\end{aligned}$$

(d) Argue that in this setting, the lasso coefficients $\widehat{\beta_1}$ and $\widehat{\beta_2}$ are not unique – in other words, there are many possible solutions to the optimization problem in (c). Describe these solutions

$\widehat{\beta^L}$ 의 β_1 과 β_2 의 절대값 term 으로 인해 β_1 과 β_2 을 최소로 하는 값이 여러 개 존재한다.

9. In this exercise, we will predict the number of applications received using the other variables in the college data set.

(a) Split the data set into a training set and a test set.

train 70% test 30%로 나누었다. set.seed(1)

(b) Fit a linear model using least squares on the training set, and report the test error obtained.

MSE : 1261630

(c) Fit a ridge regression model on the training set, with λ chosen by cross-validation. Report the test error obtained.

MSE: 1261598

(d) Fit a lasso

MSE: 1233380

# of non-zero coefficient estimates : 15

(e) PCR model

MSE : 1578584

M : 10

(f) PLS model

MSE : 1279353

M: 10

(g) MSE 비교 표

	least squares	ridge	lasso	PCR	PLS
MSE	1261630	1261598	1233380	1837203	1279353

Lasso 의 MSE 가 가장 낮고 PCR 의 MSE 가 가장 높다. PCR 을 제외한 나머지 4 개의 모형의 MSE 는 1200000 대이다. 그러나 PCR 만 1800000 대이다.

11. We will now try to predict per capita crime rate in the Boston data set.

(a) Try out some of the regression methods explored in this chapter, such as best subset selection, the lasso, ridge regression, and PCR. Present and discuss results for the approaches that you consider.

train 과 test 는 각각 0.7 그리고 0.3 으로 나눴다. Best subset 의 k-fold cv error 를 구하기 위해 사용한 k=10 이다.

Method	CV error	Note
Best Subset	42.46014	# of predictors = 12
Ridge	57.37236	$\lambda = 0.2477076$
Lasso	59.55661	λ =0.04113335 # of predictors = 10
PCR	63.96279	M = 4
PLS	58.95384	M = 9

(b) Propose a model (or set of models) that seem to perform well on this data set, and justify your answer. Make sure that you are evaluating model performance using validation set error, cross-validation, or some other reasonable alternative, as opposed to using training error.

(a)의 결과를 바탕으로 CV error 가 가장 작은 Best subset 으로 구한 모델을 제안한다. 모델의 계수는 아래와 같다.

zn	indus	chas	nox	rm	dis
0.034651703	-0.077650190	-0.652952900	-7.221425363	0.110889976	-0.684897437
rad	tax	ptratio	black	lstat	medv
0.518258782	-0.002001553	-0.284337663	-0.011818440	0.209995205	-0.115036009

(c) Does your chosen model involve all of the features in the data set? Why or Why not? 아니요. Best subset 으로 구한 모델은 12 개의 predictor 를 갖는다. 이는 모델에서 제외된 age 라는 변수가 response 인 crim 에 대한 설명력이 부족하기 때문이다.

3. Discussion

이번 과제를 통해 least squares 의 prediction accuracy 와 model interpretability 를 개선하는 모델 적합 방법으로 1) Subset selection 2) Regularization 3) Dimension reduction 을 배웠다. 기존의 least squares 는 high dimensional data setting 에서도 flexible 하기 때문에

overfitting 할 문제점이 있다. 그러나, 이번 과제에서 배운 방법들을 통해 least squares 보다 덜 flexible 한 동시에 variance 를 줄여 성능을 크게 개선할 수 있다. 이러한 방법들을 구현하기 위해 R 에서 구현하는 패키지와 코드를 배웠다. Best subset 의 경우 leaps 패키지의 regsubset 을, lasso 와 ridge 의 경우 glmnet 패키지의 glmnet 함수를 이용한다. 마지막으로 PCR 그리고 PLS 는 pls 패키지의 pcr 과 plsr 함수를 사용한다. 이러한 이론과 실습 내용을 바탕으로 예제를 풀어봄으로써 다양한 method 에 대해 깊이 이해할 수 있었다.

4. Appendix

```
rm(list=ls())

setwd('C:/Users/dnskd/Desktop/20Spring/datamining/week6')


# data library #

library(ISLR)


# -------- Lab1 : Subset Selection Methods ---------- #

# 1. Best Subset Selection

fix(Hitters)

names(Hitters)

dim(Hitters)

sum(is.na(Hitters$Salary))

Hitters = na.omit(Hitters)

dim(Hitters)

sum(is.na(Hitters))


library(leaps)

regfit.full = regsubsets(Salary ~ . , Hitters)

summary(regfit.full)

regfit.full = regsubsets(Salary ~ ., data = Hitters, nvmax = 19)

reg.summary = summary(regfit.full)

names(reg.summary)

reg.summary$rsq


par(mfrow = c(2,2))

plot(reg.summary$rss, xlab = "Number of Variables", ylab = "RSS", type = 'l')

plot(reg.summary$adjr2, xlab = "Number of Variables", ylab = "Adjusted RSq", type = "l")

which.max(reg.summary$adjr2)

points(11, reg.summary$adjr2[11], col = 'red', cex = 2, pch=20)

plot(reg.summary$cp, xlab = "Number of Variables", ylab = "Cp", type = 'l')

which.min(reg.summary$cp)

points(10, reg.summary$cp[10], col = 'red', cex = 2, pch = 20)

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)


# built-plot function

plot(regfit.full, scale = "r2")

plot(regfit.full, scale = "adjr2")

plot(regfit.full, scale = "Cp")

plot(regfit.full, scale = "bic")


coef(regfit.full, 6)


# 2. Forward and Backward Stepwise Selection

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

summary(regfit.fwd)

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

summary(regfit.bwd)


# 3. Choosing Among models Using the Validation Set Approach and Cross-Validation

set.seed(1)

train = sample(c(TRUE, FALSE), nrow(Hitters), rep = TRUE)

test = (!train)

regfit.best = regsubsets(Salary ~., data = Hitters[train, ], nvmax = 19)

test.mat = model.matrix(Salary ~ ., data = Hitters[test,])

val.errors = rep(NA, 19)

for(i in 1:19){

  coefi = coef(regfit.best, id = i)

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

  val.errors[i] = mean((Hitters$Salary[test] - pred)^2)

}

val.errors

which.min(val.errors)

coef(regfit.best, 7)

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

}

regfit.best = regsubsets(Salary ~., data = Hitters, nvmax = 19)

coef(regfit.best, 10)


k = 10

set.seed(1)
```

```
folds = sample(1:k, nrow(Hitters), replace = TRUE)

cv.errors = matrix(NA, k, 19, dimnames = list(NULL, paste(1:19)))

for(j in 1:k){

  best.fit = regsubsets(Salary ~., data = Hitters[folds!=j,], nvmax = 19)

  for(i in 1:19){

    pred = predict.regsubsets(best.fit, Hitters[folds ==j,], id = i)

    cv.errors[j,i] = mean( (Hitters$Salary[folds == j] - pred) ^ 2)

  }

}

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

mean.cv.errors

par(mfrow = c(1,1))

plot(mean.cv.errors, type = "b")

reg.best = regsubsets(Salary ~ ., data = Hitters, nvmax = 19)

coef(reg.best, 10)


# -------- Lab2 : Ridge Regression and the Lasso ----------- #

if(!require(glmnet)) install.packages('glmnet'); library(glmnet)

x = model.matrix(Salary~., Hitters)[,-1]

y = Hitters$Salary


# 1. Ridge regression

grid = 10^seq(10, -2, length = 100)

ridge.mod = glmnet(x, y, alpha = 0, lambda = grid)

dim(coef(ridge.mod))

ridge.mod$lambda[50]

coef(ridge.mod)[, 50]

sqrt(sum(coef(ridge.mod)[-1, 50]^2))

ridge.mod$lambda[60]

coef(ridge.mod)[,60]

predict(ridge.mod, s = 50, type = "coefficients")[1:20,]


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)

ridge.pred = predict(ridge.mod, s = 4, newx = x[test,])

mean((ridge.pred - y.test)^2)

mean((mean(y[train])-y.test)^2)

ridge.pred = predict(ridge.mod, s = 1e10, newx = x[test,])

mean((ridge.pred - y.test)^2)

ridge.pred = predict(ridge.mod, s=0, newx = x[test, ], exact = TRUE)

mean((ridge.pred - y.test)^2)

lm(y~x, subset = train)

predict(ridge.mod, s = 0, exact = TRUE, type = "coefficients")[1:20,]


set.seed(1)

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

plot(cv.out)

bestlam = cv.out$lambda.min

bestlam

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

mean((ridge.pred - y.test)^2)

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

predict(out, type = "coefficients", s = bestlam)[1:20,]


# the lasso

lasso.mod = glmnet(x[train, ], y[train], alpha = 1, lambda = grid)

plot(lasso.mod)


set.seed(1)

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

plot(cv.out)

bestlam = cv.out$lambda.min

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

mean((lasso.pred - y.test)^2)

out = glmnet(x, y, alpha = 1, lambda = grid)

lasso.coef = predict(out, type ="coefficients", s = bestlam)[1:20,]

lasso.coef


# -------- Lab3 : PCR and PLS Regression -------- #

# 1. Principal Components Regression

if(!require(pls)) install.packages('pls');library(pls)

set.seed(2)

pcr.fit = pcr(Salary ~., data = Hitters, scale = TRUE, validation = "CV")
```

```
summary(pcr.fit)

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



set.seed(1)

pcr.fit = pcr(Salary ~., data = Hitters,
subset = train, scale = TRUE, validation =
"CV")

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

pcr.pred = predict(pcr.fit, x[test, ], ncomp
= 7)

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

pcr.fit = pcr(y ~ x, scale = TRUE, ncomp =
7)

summary(pcr.fit)


# 2. Partial Least Squares

set.seed(1)

pls.fit = plsr(Salary ~., data = Hitters,
subset = train, scale = TRUE, validation =
"CV")

summary(pls.fit)

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

pls.pred = predict(pls.fit, x[test, ], ncomp
= 2)

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

pls.fit = plsr(Salary ~., data = Hitters,
scale = TRUE, ncomp = 2)

summary(pls.fit)


# -------- EX9. ----------- #

# a.

data(College)

names(College)

sum(is.na(College$Grad.Rate))

dim(College)

set.seed(1)

train <- sample(1:nrow(College),
nrow(College)*0.7)

test <- seq(1:nrow(College))[-train]


# b.least squares

lm.fit <- lm(Apps ~., data =
College[train,])

pred <- predict(lm.fit, College[test,])

mean((College[test,'Apps']-pred)^2)


# c. ridge

set.seed(1)

X <- model.matrix(Apps~., College)[,-1]

y <- College$Apps

grid <- 10 ^ seq(4, -2, length = 100)

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

cv.out = cv.glmnet(X[train,], y[train],
alpha = 0, lambda = grid, thresh = 1e-12)

bestlam = cv.out$lambda.min

bestlam

ridge.pred = predict(ridge.mod, s = bestlam,
newx = X[test,])

mean((ridge.pred - y[test])^2)


# d. lasso

set.seed(1)

lasso.mod = glmnet(X[train, ], y[train],
alpha = 1, lambda = grid)

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

bestlam = cv.out$lambda.min

lasso.pred = predict(lasso.mod, s = bestlam,
newx = X[test, ])

mean((lasso.pred - y[test])^2)

predict(lasso.mod, s = bestlam, type =
"coefficients")


# e. PCR

set.seed(1)

pcr.fit = pcr(Apps ~., data = College,
subset = train, scale = TRUE, validation =
"CV")

summary(pcr.fit)

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

mean((pcr.pred - y[test])^2)


# f. PLS

set.seed(1)

pls.fit = plsr(Apps ~., data = College,
subset = train, scale = TRUE, validation =
"CV")

summary(pls.fit)

pls.pred = predict(pls.fit, X[test, ], ncomp
= 10)

mean((pls.pred - y[test])^2)


# -------- EX11. ----------- #

# a & b

library(MASS)

data("Boston")

predict.regsubset <- 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
}
dim(Boston)
names(Boston)


# train and test
set.seed(1)
train <- sample(1:nrow(Boston),
nrow(Boston)*0.7)
test <- seq(1:nrow(Boston))[-train]
X <- model.matrix(crim~., data = Boston)[,-
1]
y <- Boston$crim


# best subset
set.seed(1)
k = 10
folds = sample(1:k, nrow(Boston), replace =
TRUE)
cv.errors = matrix(NA, k, 13, dimnames =
list(NULL, paste(1:13)))
for(j in 1:k){
  best.fit = regsubsets(crim ~., data =
Boston[folds!=j,], nvmax = 13)
  for(i in 1:13){
    pred = predict.regsubset(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)
mean.cv.errors
plot(1:13, mean.cv.errors, type = "b")
reg.best <- regsubsets(crim~., data =
Boston[train,], nvmax = 13)
coef(reg.best, which.min(mean.cv.errors))



# ridge
set.seed(1)
grid <- 10 ^ seq(4, -2, length = 100)
cv.out = cv.glmnet(X[train,], y[train],
alpha = 0, lambda = grid,type.measure =
'mse', thresh = 1e-12)
bestlam = cv.out$lambda.min
bestlam
ridge.pred = predict(ridge.mod, s = bestlam,
newx = X[test,])
mean((ridge.pred - y[test])^2)


# lasso
set.seed(1)
lasso.mod = glmnet(X[train, ], y[train],
alpha = 1, lambda = grid)
cv.out = cv.glmnet(X[train, ], y[train],
alpha = 1)
bestlam = cv.out$lambda.min
lasso.pred = predict(lasso.mod, s = bestlam,
newx = X[test, ])
mean((lasso.pred - y[test])^2)
predict(lasso.mod, s = bestlam, type =
"coefficients")


# PCR
set.seed(1)
pcr.fit = pcr(crim ~., data = Boston, subset
= train, scale = TRUE, validation = "CV")
summary(pcr.fit)
pcr.pred = predict(pcr.fit, X[test, ], ncomp
= 4)
mean((pcr.pred - y[test])^2)


# f. PLS
set.seed(1)
pls.fit = plsr(crim ~., data = Boston,
subset = train, scale = TRUE, validation =
"CV")
summary(pls.fit)
pls.pred = predict(pls.fit, X[test, ], ncomp
= 9)
mean((pls.pred - y[test])^2)
```