Stat230hw6

Yifan Zheng

4/5/2020

```
library(MASS)
library(mlbench)
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 3.0-2
```

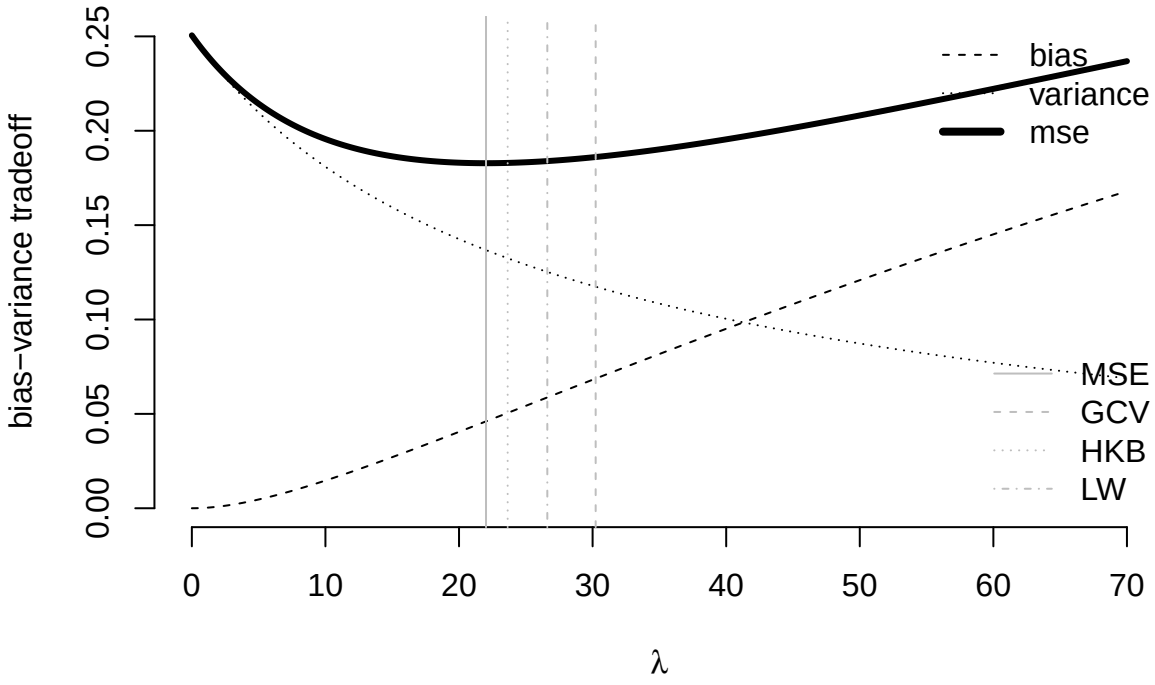
Lecture 13 Problem 6

Setting 1: independent covariates

(1) re-run simulation of the first plot of figure 1 (minimize mse with independent covariates)

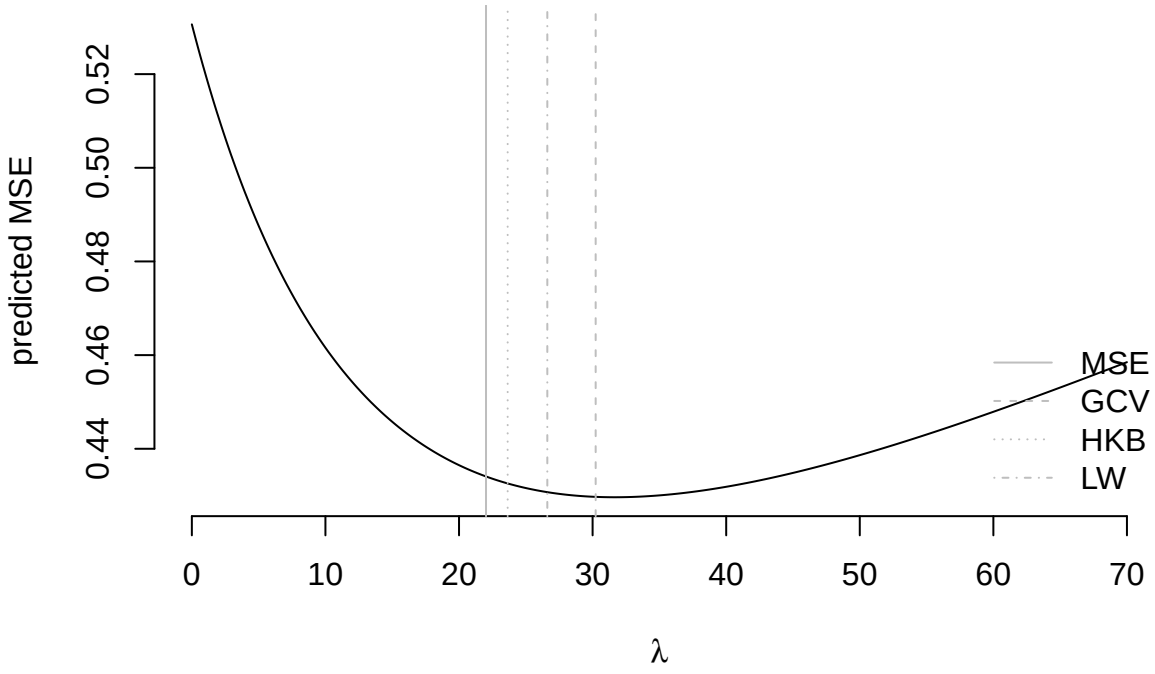
```
set.seed(111)

## setting 1
n = 200
p = 100

beta = rep(1/sqrt(p), p)
sig  = 1/2
## x is independent Normals
X    = matrix(rnorm(n*p), n, p)
## standardize the covariates
X    = scale(X)
X    = X*sqrt(n/(n-1))
## yi = xi1+...+xip+zi
Y    = as.vector(X%*%beta + rnorm(n, 0, sig))

## eigen-decomposition
eigenxx = eigen(t(X)%*%X)
## diagnal
xis     = eigenxx$values
## eigen-vectors
gammas  = t(eigenxx$vectors)%*%beta

# range of lambda
lambda.seq = seq(0, 70, 0.01)
bias2.seq  = lambda.seq
```

```
var.seq    = lambda.seq
mse.seq    = lambda.seq

for(i in 1:length(lambda.seq))
{
       ll = lambda.seq[i]
       bias2.seq[i]  = ll^2*sum(gammas^2/(xis + ll)^2)
       var.seq[i]    = sig^2*sum(xis/(xis + ll)^2)
       mse.seq[i]    = bias2.seq[i] + var.seq[i]
}

y.min = min(bias2.seq, var.seq, mse.seq)
y.max = max(bias2.seq, var.seq, mse.seq)

#pdf("biasvariancetradeoffridgeplot.pdf",
#    height = 10, width = 8.5)
#par(mfrow = c(2, 2))
plot(bias2.seq ~ lambda.seq, type = "l",
     ylim = c(y.min, y.max),
     xlab = expression(lambda), main = "",
     ylab = "bias-variance tradeoff",
     lty = 2, bty = "n")
lines(var.seq ~ lambda.seq, lty = 3)
lines(mse.seq ~ lambda.seq, lwd = 3, lty = 1)
abline(v = lambda.seq[which.min(mse.seq)],
       lty = 1, col = "grey")
legend("topright", c("bias", "variance", "mse"),
       lty = c(2, 3, 1), lwd = c(1, 1, 4), bty = "n")


## ridge regression
ridge.fit = lm.ridge(Y ~ X, lambda = lambda.seq)
abline(v = lambda.seq[which.min(ridge.fit$GCV)],
       lty = 2, col = "grey")
abline(v = ridge.fit$kHKB, lty = 3, col = "grey")
abline(v = ridge.fit$kLW, lty = 4, col = "grey")
legend("bottomright",
       c("MSE", "GCV", "HKB", "LW"),
       lty = 1:4, col = "grey", bty = "n")
```



(2) re-run simulation of the second plot of figure 1 (minimize prediction error with independent covariates)

```
## prediction
X.new    = matrix(rnorm(n*p), n, p)
X.new    = scale(X.new)
X.new    = X.new*matrix(sqrt(n/(n-1)), n, p)
Y.new    = as.vector(X.new%*%beta + rnorm(n, 0, sig))
predict.error = Y.new - X.new%*%ridge.fit$coef
predict.mse   = apply(predict.error^2, 2, mean)
plot(predict.mse ~ lambda.seq, type = "l",
     xlab = expression(lambda),
     ylab = "predicted MSE", bty = "n")
abline(v = lambda.seq[which.min(mse.seq)],
       lty = 1, col = "grey")
abline(v = lambda.seq[which.min(ridge.fit$GCV)],
       lty = 2, col = "grey")
abline(v = ridge.fit$kHKB, lty = 3, col = "grey")
abline(v = ridge.fit$kLW, lty = 4, col = "grey")
legend("bottomright",
       c("MSE", "GCV", "HKB", "LW"),
       lty = 1:4, col = "grey", bty = "n")

mtext("independent covariates", side = 1,
      line = -58, outer = TRUE, font.main = 1, cex=1.5)
```



(3) Report λ selected by Dempster et al. (1977)'s method

```
## Dempster et al. (1977)
## use ols sigma and ols beta to calculate FOC of minimizing MSE
beta_ols = solve(t(X) %*% X) %*% t(X) %*% Y
y_hat = as.vector(X %*% beta_ols)
eps_hat = Y - y_hat
sigma2_ols = var(eps_hat) *n/(n-p)
gamma_hat = t(eigenxx$vectors)%*%beta_ols

## by FOC
lambda.seq = seq(0, 40, 0.01)
left  = lambda.seq
right    = lambda.seq
abs_diff    = lambda.seq


for(i in 1:length(lambda.seq))
{
       ll = lambda.seq[i]
       left[i]  = ll*sum((gamma_hat^2)*xis/(xis + ll)^3)
       right[i]    = sigma2_ols*sum(xis/(xis + ll)^3)
       abs_diff[i]    = abs(left[i] - right[i])
}
```

```
lambda.dempster = lambda.seq[which.min(abs_diff)]
cat("Lambda selected by Dempster et al.(1977)'s method is: ", lambda.dempster)
```

```
## Lambda selected by Dempster et al.(1977)'s method is:  18.37
```

(4) Report λ selected by PRESS

```
## PRESS
lambda.seq = seq(10, 40, 0.01)
PRESS  = lambda.seq

for(i in 1:length(lambda.seq))
{
       ll = lambda.seq[i]
       H = X.new %*% solve(t(X.new) %*% X.new + ll * diag(1,p,p)) %*% t(X.new)
       h = as.vector(diag(H))
       eps_hat.new = Y.new - H %*% Y.new
       eps_hat_i.new = eps_hat.new/(1-h)
       PRESS[i] = sum(eps_hat_i.new^2)
}

lambda.press = lambda.seq[which.min(PRESS)]

cat("Lambda selected by PRESS method is: ", lambda.press)
```

```
## Lambda selected by PRESS method is:  25.27
```

(5) Report λ selected by K-fold CV

```
# split the data into k folds
lambda.seq = seq(10, 40, 0.01)
kcv = lambda.seq

k = 5
nk = n/k
for (j in 1:length(lambda.seq)){
  pred.error = 0
  for (i in 1:k){
    x.test = X.new[((i-1)*nk+1):(i*nk),]
    y.test = Y.new[((i-1)*nk+1):(i*nk)]
    x.train = X.new[-(((i-1)*nk+1):(i*nk)),]
    y.train = Y.new[-(((i-1)*nk+1):(i*nk))]

    beta.ridge = solve(t(x.train) %*% x.train + lambda.seq[j]*diag(p)) %*% t(x.train) %*% y.train
    pred.error = pred.error + sum((y.test - x.test %*% beta.ridge)^2)
  }
  kcv[j] = pred.error/k
}

lambda.kcv = lambda.seq[which.min(kcv)]

cat("Lambda selected by k-fold CV method is: ", lambda.kcv)
```

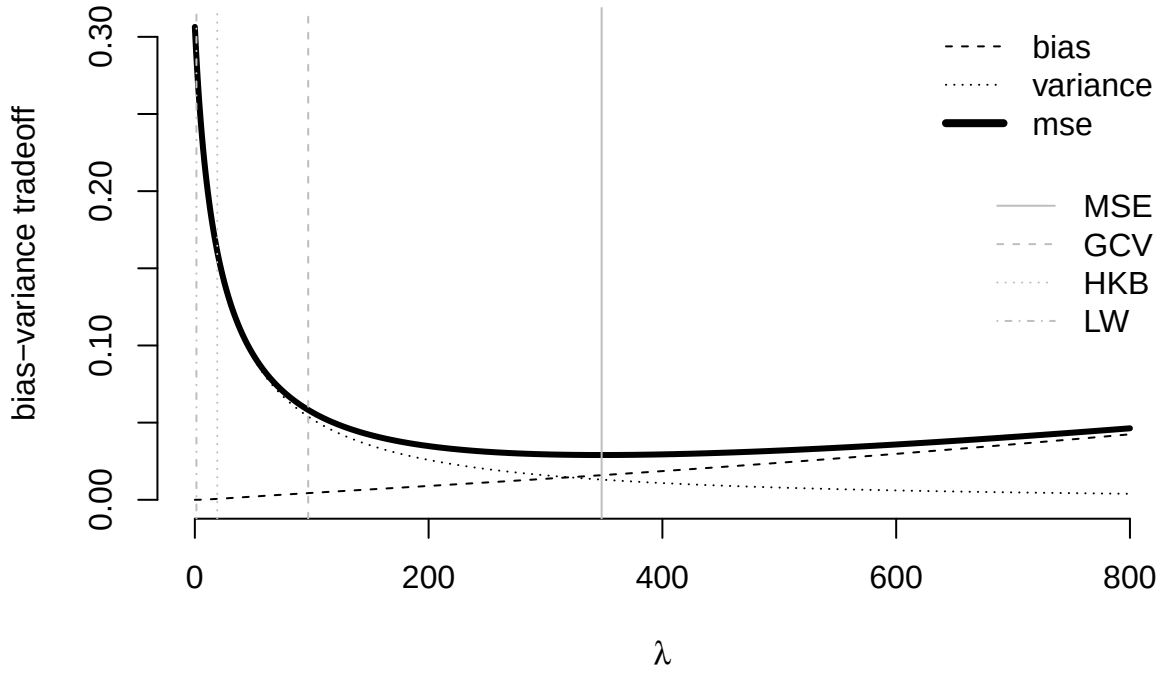
```
## Lambda selected by k-fold CV method is:  19.03
```

Setting 2: dependent covariates

(1) re-run simulation of the 3rd plot of figure 1 (minimize mse with dependent covariates)

```
## setting 2
n = 200
p = 100
beta = rep(1/sqrt(p), p)
sig  = 1/2
## correlated Normals
X    = matrix(rnorm(n*p), n, p) + rnorm(n, 0, 0.5)
## standardize the covariates
X    = scale(X)
X    = X*matrix(sqrt(n/(n-1)), n, p)
Y    = as.vector(X%*%beta + rnorm(n, 0, sig))
eigenxx = eigen(t(X)%*%X)
xis     = eigenxx$values
gammas  = t(eigenxx$vectors)%*%beta

lambda.seq = seq(0, 800, 1)
bias2.seq  = lambda.seq
var.seq    = lambda.seq
mse.seq    = lambda.seq
for(i in 1:length(lambda.seq))
{
  ll = lambda.seq[i]
  bias2.seq[i]  = ll^2*sum(gammas^2/(xis + ll)^2)
  var.seq[i]    = sig^2*sum(xis/(xis + ll)^2)
  mse.seq[i]    = bias2.seq[i] + var.seq[i]
}

y.min = min(bias2.seq, var.seq, mse.seq)
y.max = max(bias2.seq, var.seq, mse.seq)
plot(bias2.seq ~ lambda.seq, type = "l",
     ylim = c(y.min, y.max),
     xlab = expression(lambda), main = "",
     ylab = "bias-variance tradeoff",
     lty = 2, bty = "n")
lines(var.seq ~ lambda.seq, lty = 3)
lines(mse.seq ~ lambda.seq, lwd = 3, lty = 1)
abline(v = lambda.seq[which.min(mse.seq)],
       lty = 1, col = "grey")
legend("topright", c("bias", "variance", "mse"),
       lty = c(2, 3, 1), lwd = c(1, 1, 4), bty = "n")



## ridge regression
ridge.fit = lm.ridge(Y ~ X, lambda = lambda.seq)
abline(v = lambda.seq[which.min(ridge.fit$GCV)],
       lty = 2, col = "grey")
abline(v = ridge.fit$kHKB, lty = 3, col = "grey")
abline(v = ridge.fit$kLW, lty = 4, col = "grey")
legend("right",
```

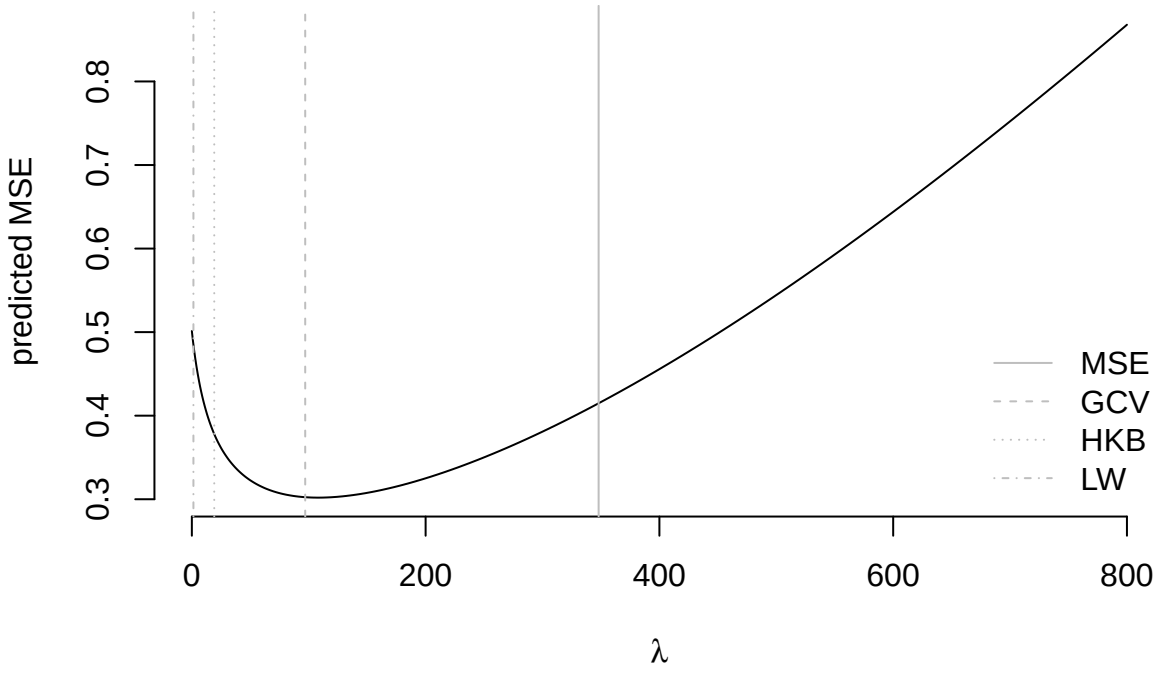
```
       c("MSE", "GCV", "HKB", "LW"),
       lty = 1:4, col = "grey", bty = "n")
```



(2) re-run simulation of the 4th plot of figure 1 (minimize prediction error with dependent covariates)

```
## prediction
X.new    = matrix(rnorm(n*p), n, p) + rnorm(n, 0, 0.5)
X.new    = scale(X.new)
X.new    = X.new*matrix(sqrt(n/(n-1)), n, p)
Y.new    = as.vector(X.new%*%beta + rnorm(n, 0, sig))
predict.error = Y.new - X.new%*%ridge.fit$coef
predict.mse   = apply(predict.error^2, 2, mean)
plot(predict.mse ~ lambda.seq, type = "l",
     xlab = expression(lambda),
     ylab = "predicted MSE", bty = "n")
abline(v = lambda.seq[which.min(mse.seq)],
       lty = 1, col = "grey")
abline(v = lambda.seq[which.min(ridge.fit$GCV)],
       lty = 2, col = "grey")
abline(v = ridge.fit$kHKB, lty = 3, col = "grey")
abline(v = ridge.fit$kLW, lty = 4, col = "grey")
legend("bottomright",
       c("MSE", "GCV", "HKB", "LW"),
       lty = 1:4, col = "grey", bty = "n")
```

```
mtext("correlated covariates", side = 1,
      line = -28, outer = TRUE, font.main = 1, cex=1.5)
```



(3) Report λ selected by Dempster et al. (1977)'s method

```
## Dempster et al. (1977)
## use ols sigma and ols beta to calculate FOC of minimizing MSE

beta_ols = solve(t(X) %*% X) %*% t(X) %*% Y
y_hat = as.vector(X %*% beta_ols)
eps_hat = Y - y_hat
sigma2_ols = var(eps_hat) *n/(n-p)
gamma_hat = t(eigenxx$vectors)%*%beta_ols

## by FOC
lambda.seq = seq(0, 500, 0.01)
left  = lambda.seq
right    = lambda.seq
abs_diff    = lambda.seq


for(i in 1:length(lambda.seq))
{
       ll = lambda.seq[i]
       left[i]  = ll*sum((gamma_hat^2)*xis/(xis + ll)^3)
```

```
       right[i]    = sigma2_ols*sum(xis/(xis + ll)^3)
       abs_diff[i]    = abs(left[i] - right[i])
}

lambda.dempster = lambda.seq[which.min(abs_diff)]
cat("Lambda selected by Dempster et al.(1977)'s method is: ", lambda.dempster)
```

```
## Lambda selected by Dempster et al.(1977)'s method is:  47.5
```

(4) Report λ selected by PRESS

```
## PRESS

lambda.seq = seq(60, 100, 0.01)
PRESS  = lambda.seq

for(i in 1:length(lambda.seq))
{
       ll = lambda.seq[i]
       H = X.new %*% solve(t(X.new) %*% X.new + ll * diag(1,p,p)) %*% t(X.new)
       h = as.vector(diag(H))
       eps_hat.new = Y.new - H %*% Y.new
       eps_hat_i.new = eps_hat.new/(1-h)
       PRESS[i] = sum(eps_hat_i.new^2)
}

lambda.press = lambda.seq[which.min(PRESS)]

cat("Lambda selected by PRESS method is: ", lambda.press)
```

```
## Lambda selected by PRESS method is:  89.32
```

(5) Report λ selected by K-fold CV

```
# split the data into k folds
lambda.seq = seq(60, 100, 0.01)
kcv = lambda.seq

k = 10
nk = n/k
for (j in 1:length(lambda.seq)){
  pred.error = 0
  for (i in 1:k){
    x.test = X.new[((i-1)*nk+1):(i*nk),]
    y.test = Y.new[((i-1)*nk+1):(i*nk)]
    x.train = X.new[-(((i-1)*nk+1):(i*nk)),]
    y.train = Y.new[-(((i-1)*nk+1):(i*nk))]

    beta.ridge = solve(t(x.train) %*% x.train + lambda.seq[j]*diag(p)) %*% t(x.train) %*% y.train
    pred.error = pred.error + sum((y.test - x.test %*% beta.ridge)^2)
  }
```

```
  kcv[j] = pred.error/k
}

lambda.kcv = lambda.seq[which.min(kcv)]

cat("Lambda selected by k-fold CV method is: ", lambda.kcv)
```

```
## Lambda selected by k-fold CV method is:  82.2
```

Lecture 14 Problem 4, More noise in the Boston housing data

```
data(BostonHousing)
n = nrow(BostonHousing)
set.seed(123)
```

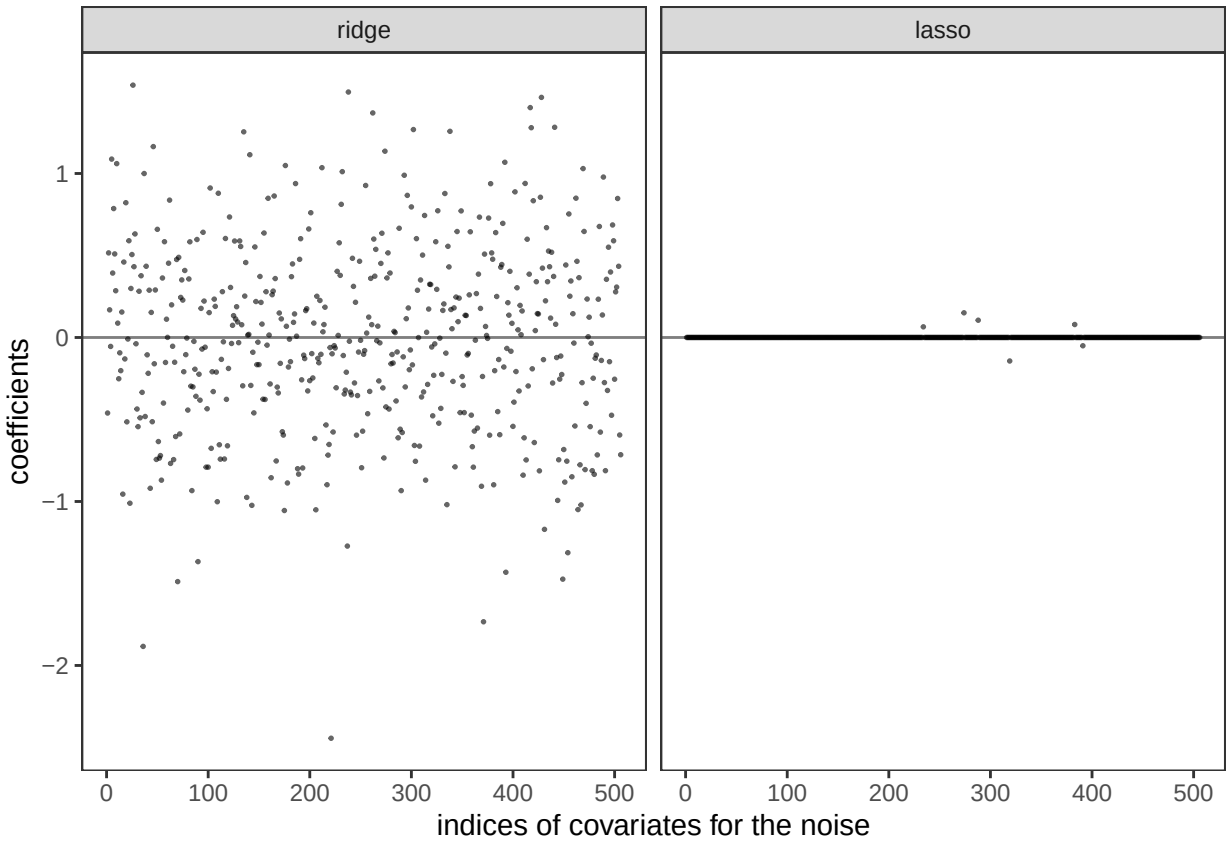
(1) add Add p=n columns of covariates of random noise, and compare OLS, ridge, and lasso.

```
dat.1 = BostonHousing
p.1 = n
# add p.1 columns random noise
xnoise = matrix(rnorm(n*p.1), n, p.1)
dat.1 = cbind(dat.1,xnoise)

# training and testing data
trainid.1 = sample(1:n, floor(n*0.9))
xcovar = subset(dat.1, select = -c(medv))
yvec = dat.1$medv
dat.1 = cbind(yvec,xcovar)


# linear regression
lm.1 = lm(yvec~., data = dat.1[trainid.1,])
predict.error.1 = dat.1$yvec[- trainid.1] - predict(lm.1, dat.1[- trainid.1, ])
mse.ols.1 = sum(predict.error.1^2)/length(predict.error.1)

# ridge regression
lambdas = seq(0,5,0.01)
rlm.1 = lm.ridge(yvec~.,data = dat.1[trainid.1,], lambda = lambdas)
coef.r.1 = coef(rlm.1)[ which.min (rlm.1$GCV), ]
predict.error.r.1 = dat.1$yvec[- trainid.1] - data.matrix(cbind (1, xcovar[-trainid.1,]))%*% coef.r.1
mse.ridge.1 = sum (predict.error.r.1^2)/length(predict.error.r.1)

# lasso
cv.lasso.1 = cv.glmnet(x = data.matrix(xcovar[trainid.1,]),y = yvec[trainid.1])
coef.l.1 = coef(cv.lasso.1, s="lambda.min")
predict.error.l.1 = dat.1$yvec[-trainid.1]-data.matrix(cbind (1, xcovar[-trainid.1,]))%*% coef.l.1
mse.lasso.1 = sum(predict.error.l.1^2)/length(predict.error.l.1)

# show mse of ols, ridge and lasso
```

```
cat("mse of ols is ", mse.ols.1, "\nmse of ridge is ", mse.ridge.1, "
    \nmse of lasso is ", mse.lasso.1)
```

```
## mse of ols is  10155.19
## mse of ridge is  259.1127
##
## mse of lasso is  19.56092
```

```
library(ggplot2)
dat.1.plot = rbind(data.frame(estimator = "ridge",
                              index = 1:p.1,
                              coef = coef.r.1[-(1:14)]),
                   data.frame(estimator = "lasso",
                              index = 1:p.1,
                              coef = coef.l.1[-(1:14)]))

ggplot(dat.1.plot) +
  geom_point(aes(x=index, y=coef),
             cex = 0.3, alpha = 0.6)+
  geom_hline(aes(yintercept = 0), alpha = 0.5) +
  facet_grid(~estimator) +
  theme_bw() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank()) +
  xlab("indices of covariates for the noise") +
  ylab("coefficients")
```




```
ggsave("ridge_lasso_coef_bostonhousingnoise.pdf",
       width = 8.5, height = 4)
```

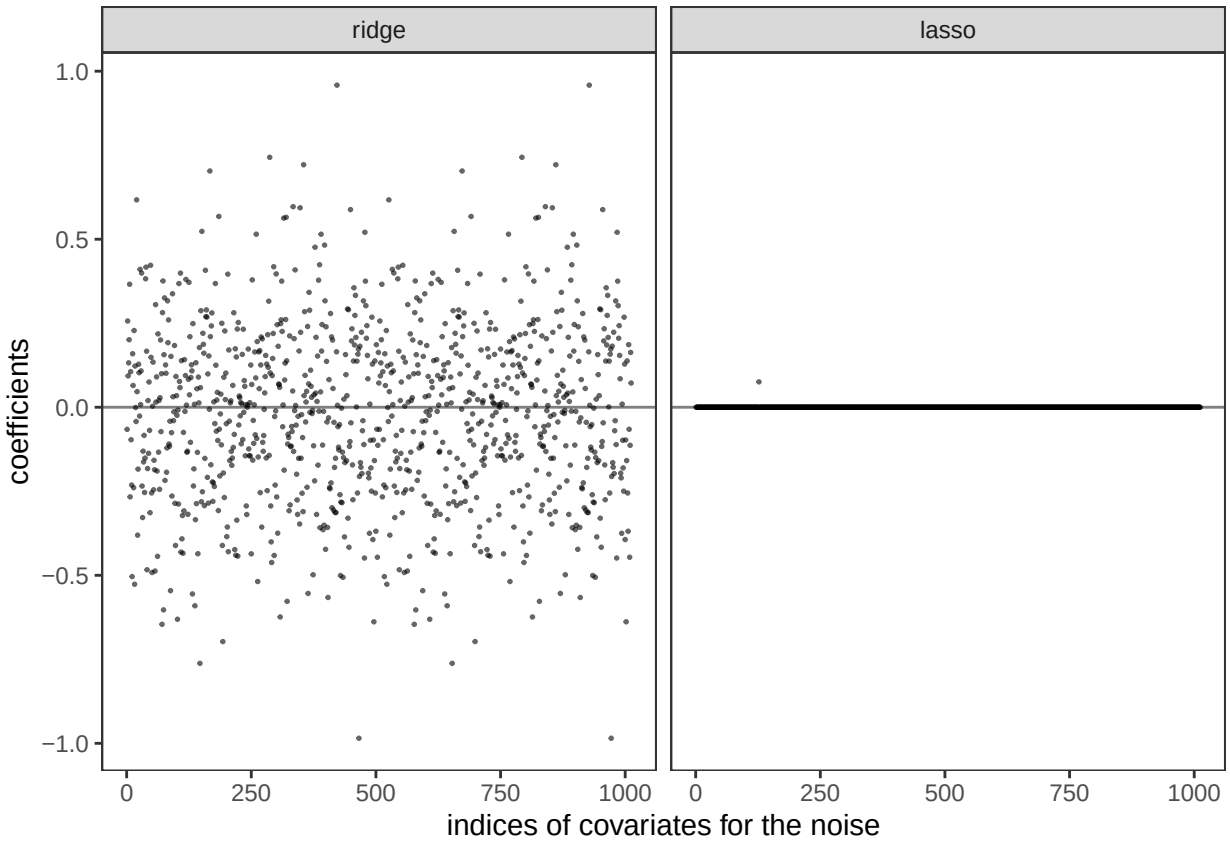
(2) Add p=2n columns of covariates of random noise, and compare OLS, ridge, and lasso.

```
dat.2 = BostonHousing
p.2 = n*2
# add p.2 columns random noise
xnoise = matrix(rnorm(n*p.1), n, p.2)
dat.2 = cbind(dat.2,xnoise)

# training and testing data
trainid.2 = sample(1:n, floor(n*0.9))
xcovar = subset(dat.2, select = -c(medv))
yvec = dat.2$medv
dat.2 = cbind(yvec,xcovar)


# linear regression
lm.2 = lm(yvec~., data = dat.1[trainid.2,])
predict.error.2 = dat.2$yvec[- trainid.2] - predict(lm.2, dat.2[- trainid.2, ])
mse.ols.2 = sum(predict.error.2^2)/length(predict.error.2)

# ridge regression
```

```
lambdas = seq(0,5,0.01)
rlm.2 = lm.ridge(yvec~.,data = dat.2[trainid.2,], lambda = lambdas)
coef.r.2 = coef(rlm.2)[ which.min (rlm.2$GCV), ]
predict.error.r.2 = dat.1$yvec[- trainid.2] - data.matrix(cbind (1, xcovar[-trainid.2,]))%*% coef.r.2
mse.ridge.2 = sum (predict.error.r.2^2)/length(predict.error.r.2)

# lasso
cv.lasso.2 = cv.glmnet(x = data.matrix(xcovar[trainid.2,]),y = yvec[trainid.2])
coef.l.2 = coef(cv.lasso.2, s="lambda.min")
predict.error.l.2 = dat.2$yvec[-trainid.2]-data.matrix(cbind (1, xcovar[-trainid.2,]))%*% coef.l.2
mse.lasso.2 = sum(predict.error.l.2^2)/length(predict.error.l.2)

# show mse of ols, ridge and lasso
cat("mse of ols is ", mse.ols.2, "\nmse of ridge is ", mse.ridge.2, "
    \nmse of lasso is ", mse.lasso.2)
```

```
## mse of ols is  4783.904
## mse of ridge is  151.8499
##
## mse of lasso is  23.34326
```

```
dat.2.plot = rbind(data.frame(estimator = "ridge",
                              index = 1:p.2,
                              coef = coef.r.2[-(1:14)]),
                   data.frame(estimator = "lasso",
                              index = 1:p.2,
                              coef = coef.l.2[-(1:14)]))

ggplot(dat.2.plot) +
  geom_point(aes(x=index, y=coef),
             cex = 0.3, alpha = 0.6)+
  geom_hline(aes(yintercept = 0), alpha = 0.5) +
  facet_grid(~estimator) +
  theme_bw() +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank()) +
  xlab("indices of covariates for the noise") +
  ylab("coefficients")
```




```
ggsave("ridge_lasso_coef_bostonhousingnoise.pdf",
       width = 8.5, height = 4)
```

The result shows that ridge and lasso performs better, when there are many columns are pure noise N(0,1). Especially, lasso shrink many coefficents to zero.

Lecture 17, Problem 7, Empirical comparison of logistic regression and linear discriminant analysis

Compare the performance of logistic regression and linear discriminant analysis, in terms of prediction accuracy.

case 1, model for linear discriminant analysis is correct

```
# simulate data
set.seed(123)
n.1 = 123
n.0 = 132
y.1 = rep(1, n.1)
y.0 = rep(0,n.0)
x.1 = matrix(rnorm(n.1*3,1,1),n.1,3)+rnorm(n.1,1,1)
x.0 = matrix(rnorm(n.0*3,3,1),n.0,3)+rnorm(n.0,3,1)
```

```r
x = rbind(x.1,x.0)
y = c(y.1,y.0)
n = n.1+n.0

df = cbind(y,x)
```

```r
# split traing and test data
train.id = sample(1:n, floor(n*0.8))
train.df = df[train.id,]
test.df = df[-train.id,]
```

```r
# logistic regression
logr = glm(y~., data = as.data.frame(train.df), family = binomial(link = "logit"))
pred.logr = ifelse(predict(logr, as.data.frame(test.df), type = "response") > 0.5,1,0)
acc.logr = sum((pred.logr-test.df[,1])==0)/nrow(test.df)
cat("The prediction accuracy of logistic regression is ", acc.logr)
```

```
## The prediction accuracy of logistic regression is  0.9607843
```

```r
# linear discriminant analysis
ldar = lda(y~.,data = as.data.frame(train.df))
pred.ldar = predict(ldar, as.data.frame(test.df))
acc.ldar = sum(as.numeric(as.character(pred.ldar$class))-test.df[,1]==0)/nrow(test.df)
cat("The prediction accuracy of linear discriminant analysis is ", acc.ldar)
```

```
## The prediction accuracy of linear discriminant analysis is  0.9607843
```

case 2, model for linear discriminant analysis is incorrect but the model for logistic regression is correct

```r
# simulate data
set.seed(123)
n = 200
x.1 = rnorm(n,2,3)
x.2 = runif(n,0,1)
prob = 1/(1+exp(-1-x.1-x.2))
y = rbinom(n,1,prob)
df = cbind(y,x.1,x.2)
```

```r
# split traing and test data
train.id = sample(1:n, floor(n*0.8))
train.df = df[train.id,]
test.df = df[-train.id,]
```

```r
# logistic regression
logr = glm(y~., data = as.data.frame(train.df), family = binomial(link = "logit"))
pred.logr = ifelse(predict(logr, as.data.frame(test.df), type = "response") > 0.5,1,0)
acc.logr = sum((pred.logr-test.df[,1])==0)/nrow(test.df)
cat("The prediction accuracy of logistic regression is ", acc.logr)
```

```
## The prediction accuracy of logistic regression is  0.925
```

```
# linear discriminant analysis
ldar = lda(y~.,data = as.data.frame(train.df))
pred.ldar = predict(ldar, as.data.frame(test.df))
acc.ldar = sum(as.numeric(as.character(pred.ldar$class))-test.df[,1]==0)/nrow(test.df)
cat("The prediction accuracy of linear discriminant analysis is ", acc.ldar)
```

```
## The prediction accuracy of linear discriminant analysis is  0.875
```

case 3, model for logistic regression is incorrect

```
# simulate data
set.seed(123)
n = 200
x.1 = rnorm(n,2,3)
x.2 = runif(n,0,1)
prob = pcauchy(1+x.1+x.2)
y = rbinom(n,1,prob)
df = cbind(y,x.1,x.2)
```

```
# split traing and test data
train.id = sample(1:n, floor(n*0.8))
train.df = df[train.id,]
test.df = df[-train.id,]
```

```
# logistic regression
logr = glm(y~., data = as.data.frame(train.df), family = binomial(link = "logit"))
pred.logr = ifelse(predict(logr, as.data.frame(test.df), type = "response") > 0.5,1,0)
acc.logr = sum((pred.logr-test.df[,1])==0)/nrow(test.df)
cat("The prediction accuracy of logistic regression is ", acc.logr)
```

```
## The prediction accuracy of logistic regression is  0.825
```

```
# linear discriminant analysis
ldar = lda(y~.,data = as.data.frame(train.df))
pred.ldar = predict(ldar, as.data.frame(test.df))
acc.ldar = sum(as.numeric(as.character(pred.ldar$class))-test.df[,1]==0)/nrow(test.df)
cat("The prediction accuracy of linear discriminant analysis is ", acc.ldar)
```

```
## The prediction accuracy of linear discriminant analysis is  0.775
```