hw3.R

Brian Kang

Tue May 07 17:00:04 2019

# HW3  
# 1  
# a)  
set.seed(1)  
x <- rnorm(100)  
y <- x-2\*x^2+rnorm(100)  
# n=100; p=2; model: y=x-2x^2+e (the error term)  
  
# b)  
plot(x,y)  
# We see a negative parabolic curve. As we expect from  
# normal distributions, the center is at 0 and there are  
# more data points clustered around the center than the  
# sparse few on each end of the tails.  
  
# c)  
library(boot)  
mat1 <- matrix(data = NA, nrow = 4, ncol=2) # record errors  
set.seed(1)  
dat1 <- data.frame(x,y)  
glm.1 <- glm(y~x)  
cv.1 <- cv.glm(dat1, glm.1)  
mat1[1,1] <- cv.1$delta[1]  
  
glm.2 <- glm(y~poly(x,2))  
cv.2 <- cv.glm(dat1, glm.2)  
mat1[2,1] <- cv.2$delta[1]  
  
glm.3 <- glm(y~poly(x,3))  
cv.3 <- cv.glm(dat1, glm.3)  
mat1[3,1] <- cv.3$delta[1]  
  
glm.4 <- glm(y~poly(x,4))  
cv.4 <- cv.glm(dat1, glm.4)  
mat1[4,1] <- cv.4$delta[1]  
  
# d)  
set.seed(3000)  
glm.1 <- glm(y~x)  
cv.1 <- cv.glm(dat1, glm.1)  
mat1[1,2] <- cv.1$delta[1]  
  
glm.2 <- glm(y~poly(x,2))  
cv.2 <- cv.glm(dat1, glm.2)  
mat1[2,2] <- cv.2$delta[1]  
  
glm.3 <- glm(y~poly(x,3))  
cv.3 <- cv.glm(dat1, glm.3)  
mat1[3,2] <- cv.3$delta[1]  
  
glm.4 <- glm(y~poly(x,4))  
cv.4 <- cv.glm(dat1, glm.4)  
mat1[4,2] <- cv.4$delta[1]  
mat1 # display errors

## [,1] [,2]  
## [1,] 7.2881616 7.2881616  
## [2,] 0.9374236 0.9374236  
## [3,] 0.9566218 0.9566218  
## [4,] 0.9539049 0.9539049

# The results are the same from each of the seeds. Results  
# are identical because LOOCV uses the same MSE calculation  
# process on all observations with a set n value.  
# I.e. every single observation is evaluated n folds.  
  
# e)  
# The model that goes up to the 2nd power has the smallest  
# LOOCV error. This can be what I expected because the  
# original data had a clear quadratic shape. But I expected  
# the 4th power model to do as well or better because, as   
# the direct square of a quadratic, although it may overfit,  
# the errors could have been smaller.  
  
# f)  
summary(glm.1)

##   
## Call:  
## glm(formula = y ~ x)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -9.5161 -0.6800 0.6812 1.5491 3.8183   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -1.6254 0.2619 -6.205 1.31e-08 \*\*\*  
## x 0.6925 0.2909 2.380 0.0192 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 6.760719)  
##   
## Null deviance: 700.85 on 99 degrees of freedom  
## Residual deviance: 662.55 on 98 degrees of freedom  
## AIC: 478.88  
##   
## Number of Fisher Scoring iterations: 2

# The coefficients do not mean much when we are fitting  
# a quadratic with just the intercept and linear slope.  
# The 1st power is significant at the 0.01 level.  
summary(glm.2)

##   
## Call:  
## glm(formula = y ~ poly(x, 2))  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.9650 -0.6254 -0.1288 0.5803 2.2700   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -1.5500 0.0958 -16.18 < 2e-16 \*\*\*  
## poly(x, 2)1 6.1888 0.9580 6.46 4.18e-09 \*\*\*  
## poly(x, 2)2 -23.9483 0.9580 -25.00 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 0.9178258)  
##   
## Null deviance: 700.852 on 99 degrees of freedom  
## Residual deviance: 89.029 on 97 degrees of freedom  
## AIC: 280.17  
##   
## Number of Fisher Scoring iterations: 2

# This shows that all coefficients up to the 2nd power are  
# statistically significant.  
summary(glm.3)

##   
## Call:  
## glm(formula = y ~ poly(x, 3))  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.9765 -0.6302 -0.1227 0.5545 2.2843   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -1.55002 0.09626 -16.102 < 2e-16 \*\*\*  
## poly(x, 3)1 6.18883 0.96263 6.429 4.97e-09 \*\*\*  
## poly(x, 3)2 -23.94830 0.96263 -24.878 < 2e-16 \*\*\*  
## poly(x, 3)3 0.26411 0.96263 0.274 0.784   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 0.9266599)  
##   
## Null deviance: 700.852 on 99 degrees of freedom  
## Residual deviance: 88.959 on 96 degrees of freedom  
## AIC: 282.09  
##   
## Number of Fisher Scoring iterations: 2

summary(glm.4)

##   
## Call:  
## glm(formula = y ~ poly(x, 4))  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.0550 -0.6212 -0.1567 0.5952 2.2267   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -1.55002 0.09591 -16.162 < 2e-16 \*\*\*  
## poly(x, 4)1 6.18883 0.95905 6.453 4.59e-09 \*\*\*  
## poly(x, 4)2 -23.94830 0.95905 -24.971 < 2e-16 \*\*\*  
## poly(x, 4)3 0.26411 0.95905 0.275 0.784   
## poly(x, 4)4 1.25710 0.95905 1.311 0.193   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 0.9197797)  
##   
## Null deviance: 700.852 on 99 degrees of freedom  
## Residual deviance: 87.379 on 95 degrees of freedom  
## AIC: 282.3  
##   
## Number of Fisher Scoring iterations: 2

# These two show that the coefficients up to the 2nd power  
# are statistically significant. The 3rd and 4th power are  
# insignificant, which agrees with our conclusions from the  
# cross-validation results.  
  
# 2  
# a)  
library(MASS)  
attach(Boston) # used in lecture; every name is like a vars  
mu.hat <- mean(medv)  
mu.hat

## [1] 22.53281

# b)  
# standard error of the sample mean =   
# sd(sample) / sqrt(observations count)  
sd(medv)/sqrt(nrow(Boston))

## [1] 0.4088611

# c)  
# bootstrap for mu  
# output should incluse SE of sample mean  
fun <- function(data, index) {  
 mu <- mean(data[index])  
 return (mu)  
}  
library(boot)  
boot(medv, fun, R = 1000)

##   
## ORDINARY NONPARAMETRIC BOOTSTRAP  
##   
##   
## Call:  
## boot(data = medv, statistic = fun, R = 1000)  
##   
##   
## Bootstrap Statistics :  
## original bias std. error  
## t1\* 22.53281 0.004424506 0.4012229

# SE = 0.4033299  
# The bootstrap estimated standard error of the samle mean  
# is very close to the calculated SE from the previous.  
  
# d)  
# approx 95% confidence interval using   
# [mu.hat-2SE(mu.hat) , mu.hat+2SE(mu.hat)].  
c(mu.hat-2\*0.4033299 , mu.hat+2\*0.4033299)

## [1] 21.72615 23.33947

t.test(medv)$conf.int

## [1] 21.72953 23.33608  
## attr(,"conf.level")  
## [1] 0.95

# The 95% confidence intervals from bootstrapping and   
# the t.test() method are very close.  
  
# e)  
med.hat <- median(medv)  
med.hat

## [1] 21.2

# f)   
fun <- function(data, index) {  
 med <- median(data[index])  
 return (med)  
}  
boot(medv, fun, R = 1000)

##   
## ORDINARY NONPARAMETRIC BOOTSTRAP  
##   
##   
## Call:  
## boot(data = medv, statistic = fun, R = 1000)  
##   
##   
## Bootstrap Statistics :  
## original bias std. error  
## t1\* 21.2 -0.0089 0.3795522

# The estimated standard error of the median using bootstrap  
# is reasonably small. The median is equal to the value we  
# calculated previously.  
  
# g)  
quant.hat <- quantile(medv, 0.1)  
quant.hat

## 10%   
## 12.75

# h)  
fun <- function(data, index) {  
 quant <- quantile(data[index], 0.1)  
 return (quant)  
}  
boot(medv, fun, 1000)

##   
## ORDINARY NONPARAMETRIC BOOTSTRAP  
##   
##   
## Call:  
## boot(data = medv, statistic = fun, R = 1000)  
##   
##   
## Bootstrap Statistics :  
## original bias std. error  
## t1\* 12.75 0.00365 0.5074413

# The estimated standard error of the 10th percentile using  
# bootstrap is again reasonably small. The 10th percentile  
# is equal to the value we calculated preciously.  
  
# 3  
# a)  
library(ISLR)

## Warning: package 'ISLR' was built under R version 3.5.3

data("College")  
head(College)

## Private Apps Accept Enroll Top10perc  
## Abilene Christian University Yes 1660 1232 721 23  
## Adelphi University Yes 2186 1924 512 16  
## Adrian College Yes 1428 1097 336 22  
## Agnes Scott College Yes 417 349 137 60  
## Alaska Pacific University Yes 193 146 55 16  
## Albertson College Yes 587 479 158 38  
## Top25perc F.Undergrad P.Undergrad Outstate  
## Abilene Christian University 52 2885 537 7440  
## Adelphi University 29 2683 1227 12280  
## Adrian College 50 1036 99 11250  
## Agnes Scott College 89 510 63 12960  
## Alaska Pacific University 44 249 869 7560  
## Albertson College 62 678 41 13500  
## Room.Board Books Personal PhD Terminal  
## Abilene Christian University 3300 450 2200 70 78  
## Adelphi University 6450 750 1500 29 30  
## Adrian College 3750 400 1165 53 66  
## Agnes Scott College 5450 450 875 92 97  
## Alaska Pacific University 4120 800 1500 76 72  
## Albertson College 3335 500 675 67 73  
## S.F.Ratio perc.alumni Expend Grad.Rate  
## Abilene Christian University 18.1 12 7041 60  
## Adelphi University 12.2 16 10527 56  
## Adrian College 12.9 30 8735 54  
## Agnes Scott College 7.7 37 19016 59  
## Alaska Pacific University 11.9 2 10922 15  
## Albertson College 9.4 11 9727 55

attach(College)  
# split data to training and testing  
collegeTrain <- College[1:(0.8\*nrow(College)),] # 80% for train  
collegeTest <- College[(0.8\*nrow(College)+1):nrow(College),] # 20% for test  
  
# b)  
lm.1 <- lm(Apps~., data = collegeTrain)  
summary(lm.1)

##   
## Call:  
## lm(formula = Apps ~ ., data = collegeTrain)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -5249.5 -379.4 2.6 283.3 7372.5   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -674.81356 449.81107 -1.500 0.13408   
## PrivateYes -492.49111 159.85957 -3.081 0.00216 \*\*   
## Accept 1.68045 0.04429 37.944 < 2e-16 \*\*\*  
## Enroll -1.18227 0.20477 -5.774 1.24e-08 \*\*\*  
## Top10perc 43.67102 6.13052 7.124 3.01e-12 \*\*\*  
## Top25perc -11.60268 4.94035 -2.349 0.01917 \*   
## F.Undergrad 0.07072 0.03604 1.962 0.05017 .   
## P.Undergrad 0.02374 0.04871 0.487 0.62620   
## Outstate -0.08905 0.02099 -4.242 2.56e-05 \*\*\*  
## Room.Board 0.14926 0.05276 2.829 0.00482 \*\*   
## Books 0.05611 0.25857 0.217 0.82828   
## Personal 0.03970 0.07157 0.555 0.57932   
## PhD -8.61642 5.02397 -1.715 0.08685 .   
## Terminal -2.45096 5.49433 -0.446 0.65569   
## S.F.Ratio 29.32492 14.57724 2.012 0.04470 \*   
## perc.alumni 1.14958 4.57727 0.251 0.80178   
## Expend 0.09614 0.01460 6.585 9.89e-11 \*\*\*  
## Grad.Rate 6.10467 3.21809 1.897 0.05831 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1032 on 603 degrees of freedom  
## Multiple R-squared: 0.9279, Adjusted R-squared: 0.9258   
## F-statistic: 456.3 on 17 and 603 DF, p-value: < 2.2e-16

mse.1 <- mean((predict(lm.1, collegeTest)-collegeTest$Apps)^2)  
mse.1 # test error

## [1] 1251886

# c)  
# ridge regression with CV choosing lambda  
x <- model.matrix(Apps~.,data=College)[,-1] # take out intercept  
xtrain <- model.matrix(Apps~.,data=collegeTrain)[,-1]  
xtest <- model.matrix(Apps~.,data=collegeTest)[,-1]  
ytrain <- Apps[1:(0.8\*nrow(College))]  
ytest <- Apps[(0.8\*nrow(College)+1):nrow(College)]  
  
library(glmnet)

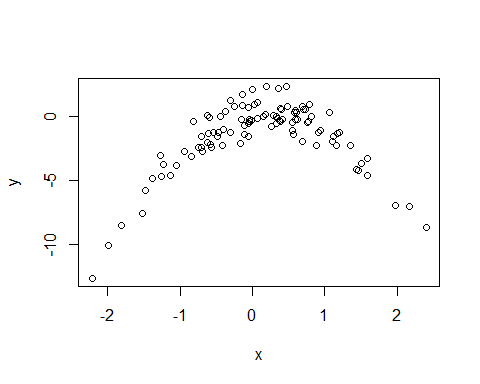
## Warning: package 'glmnet' was built under R version 3.5.2

## Loading required package: Matrix

## Loading required package: foreach

## Warning: package 'foreach' was built under R version 3.5.2

## Loaded glmnet 2.0-16



#set.seed(1)  
cv.ridge <- cv.glmnet(xtrain, ytrain, alpha = 0) # 0 for ridge  
cv.lambda <- cv.ridge$lambda.min # get smallest lambda (tuning param)  
# plot(cv.ridge)  
ridge <- glmnet(xtrain, ytrain, alpha = 0, lambda = cv.lambda)  
summary(ridge)

## Length Class Mode   
## a0 1 -none- numeric  
## beta 17 dgCMatrix S4   
## df 1 -none- numeric  
## dim 2 -none- numeric  
## lambda 1 -none- numeric  
## dev.ratio 1 -none- numeric  
## nulldev 1 -none- numeric  
## npasses 1 -none- numeric  
## jerr 1 -none- numeric  
## offset 1 -none- logical  
## call 5 -none- call   
## nobs 1 -none- numeric

pred <- predict(ridge, s = cv.lambda, newx = xtest)  
mse.2 <- mean((pred-ytest)^2)  
mse.2 # test error

## [1] 1064342

# d)  
set.seed(1)  
cv.lasso <- cv.glmnet(xtrain, ytrain, alpha = 1) # 1 for lasso  
cv.lambda <- cv.lasso$lambda.min # get smallest lambda (tuning param)  
# plot(cv.lasso)  
lasso <- glmnet(xtrain, ytrain, alpha = 1, lambda = cv.lambda)  
summary(lasso)

## Length Class Mode   
## a0 1 -none- numeric  
## beta 17 dgCMatrix S4   
## df 1 -none- numeric  
## dim 2 -none- numeric  
## lambda 1 -none- numeric  
## dev.ratio 1 -none- numeric  
## nulldev 1 -none- numeric  
## npasses 1 -none- numeric  
## jerr 1 -none- numeric  
## offset 1 -none- logical  
## call 5 -none- call   
## nobs 1 -none- numeric

pred <- predict(lasso, s = cv.lambda, newx = xtest)  
mse.3 <- mean((pred-ytest)^2)  
mse.3 # test error

## [1] 1238823

lassocoeffs <- predict(lasso, s = cv.lambda, type = "coefficients")  
summary(lassocoeffs)

## 18 x 1 sparse Matrix of class "dgCMatrix", with 18 entries   
## i j x  
## 1 1 1 -683.63462320  
## 2 2 1 -491.77079503  
## 3 3 1 1.66424223  
## 4 4 1 -1.05991856  
## 5 5 1 41.93178063  
## 6 6 1 -10.15428185  
## 7 7 1 0.05550041  
## 8 8 1 0.02178159  
## 9 9 1 -0.08544720  
## 10 10 1 0.14633720  
## 11 11 1 0.04959759  
## 12 12 1 0.03669838  
## 13 13 1 -8.31070697  
## 14 14 1 -2.34310583  
## 15 15 1 27.68487183  
## 16 16 1 0.28088548  
## 17 17 1 0.09474188  
## 18 18 1 5.79678580

lassocoeffs[lassocoeffs!=0] # nonzero lasso coeffs

## <sparse>[ <logic> ] : .M.sub.i.logical() maybe inefficient

## [1] -683.63462320 -491.77079503 1.66424223 -1.05991856 41.93178063  
## [6] -10.15428185 0.05550041 0.02178159 -0.08544720 0.14633720  
## [11] 0.04959759 0.03669838 -8.31070697 -2.34310583 27.68487183  
## [16] 0.28088548 0.09474188 5.79678580

# g)  
# In terms of test error there is not much of a huge   
# difference. Although we do see that the error from the   
# ridge regression is smaller than the least squares' and  
# lasso's. That is, it is better at prediction that other  
# models; however, we know that, like the LASSO, the  
# coefficients shrink to zero due to regularization and   
# it is "impossible" to interpret our results. The problem  
# with the lasso is that it works for low dimension models.  
# The College data we dealt with will not be considered  
# low dimensional data, but it is not high dimensional.