STA 545 Statistical Data Mining I, Fall 2020

Homework 8

Stella Liao

November 4, 2020

Problem 1

(a)

```
library(ISLR)
data(College)
myCollege <- College
set.seed(2)
train.num=sample(1:nrow(myCollege),.7*nrow(myCollege),replace=FALSE)
College.train = myCollege[train.num,]
College.test = myCollege[-train.num,]
```

(b)Ridge Model

```
library(glmnet)
```

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

```
## Loaded glmnet 4.0-2
```

```
X.train <- data.matrix(College.train[,-2])
X.test <- data.matrix(College.test[,-2])

grid <-  10 ^ seq(4, -2, length = 100)
mod.ridge <-  cv.glmnet(X.train, College.train[ ,"Apps"],family="gaussian",
                        alpha = 0, lambda = grid, thresh = 1e-12, nfolds = 5)
lambda.opt <-  mod.ridge$lambda.min
lambda.opt
```

```
## [1] 0.01
```

```
pred.ridge = predict(mod.ridge, s = lambda.opt, newx = X.test)
test.error.kfold<-mean((pred.ridge - data.matrix(College.test[, "Apps"]))^2)
test.error.kfold
```

```
## [1] 1106918
```

(c)Ridge Model with GCV

```
library(MASS)
GCV.opt.lambda <-function(X,Y,lambda.threshold=1,interval=10){
  X <-cbind(t0=1,X)

  kvals=c(0,(1:interval)*(lambda.threshold/interval))
  elements_2_remove = c(0,1)
```

```
  kvals = kvals[!(kvals %in% elements_2_remove)]

  n=nrow(X)
  gcvs <-NULL
  for (k in kvals) {

    x.k = X%*%ginv(t(X)%*%X+k*diag(ncol(X)))%*%t(X)
    y.k = x.k%*%Y

    X.svd <- svd(X)
    d <- X.svd$d
    div <-d^2 +rep(k,rep(length(d),1))
    GCV <-sum((Y-y.k)^2)/(n-sum(matrix(d^2/div,length(d))))^2

    gcvs <-c(gcvs,GCV)
  }
  minLambda=kvals[which.min(gcvs)]
  return(minLambda)
}

opt.lambda.gcv = GCV.opt.lambda(scale(data.matrix(College.train[,-2])),
                                scale(data.matrix(College.train[,2])),
                                lambda.threshold=1,interval  = 1000)
opt.lambda.gcv
```

```
## [1] 0.395
```

```
Y.train <- data.matrix(College.train[,2])
Y.test <- data.matrix(College.test[,2])
ridge.fit <- glmnet(X.train, Y.train, intercept = FALSE, family="gaussian",
                    alpha=0, lambda = opt.lambda.gcv )
pred.ridge2 <- predict(ridge.fit, newx=X.test,type = "response")
test.error.ridge.gcv <- mean((pred.ridge2 - Y.test)^2)
test.error.ridge.gcv
```

```
## [1] 1103358
```

(d)PCR with k-fold cv

```
library(pls)
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
#randomly separate data into k parts
cv.group<-function(k,datasize,seed){
    cvlist<-list()
    set.seed(seed)
    n<-rep(1:k,ceiling(datasize/k))[1:datasize]
    temp<-sample(n,datasize)
    x<-1:k
    dataseq<-1:datasize
  cvlist<-lapply(x,function(x) dataseq[temp==x])
```

```
  return(cvlist)
}

#implement PCR model and get the test error one time
#i refers to i-th dataset, the testing data
#j refers to the number of components, the tuning parameter
cv.test.pcr<-function(i,j,data,cv.list){
  data.train <- data[-cv.list[[i]],]
    data.test<- data[cv.list[[i]],]
    pcr.fit <- pcr(Apps~., data = data.train, scale = TRUE, ncomp = j)
    pcr.pred <- predict(pcr.fit, data.test, ncomp = j)
    train.err <- mean((pcr.pred-data.test[, "Apps"])^2)
    return(list(train.err,j))
}

#get the optimal tunning parameter M for PCR model
kfold.opt.M <- function(data,y.attr.name,k){

  i<-1:k
  total_comp = ncol(data)-1
  j <- seq(1,total_comp,by=1)
  i.s<-rep(i,times=length(j))
  j.s<-rep(j,each=k)
  ijs <- cbind(i.s,j.s)

  tmp.mses <-NULL
  mses <- rep(0,total_comp)
  comps <- rep(0,total_comp)
  index = 1
  old_comp = 1

  whole.cv.list = cv.group(k,nrow(College.train),2)

  for(p in 1:nrow(ijs)){
    if(ijs[p,2] == old_comp){
          mse_comp = cv.test.pcr(ijs[p,1],ijs[p,2],data,whole.cv.list)
          mse = mse_comp[[1]]
          comp = mse_comp[[2]]
          tmp.mses <- c(tmp.mses,mse)
    }
    mses[index] = mean(tmp.mses)
    comps[index] = comp
    index = ijs[p,2]
    old_comp = ijs[p,2]
}
opt.M = comps[which.min(mses)]
return(opt.M)
}

opt.M.pcr <- kfold.opt.M(College.train,"Apps",k=5)
opt.M.pcr
```

```
## [1] 17
```

```
pcr.fit <- pcr(Apps~., data = College.train, scale = TRUE, ncomp = opt.M.pcr)
pcr.pred = predict(pcr.fit, College.test, ncomp = opt.M.pcr)
test.error.pcr <- mean((pcr.pred - College.test[, "Apps"])^2)
test.error.pcr
```

```
## [1] 1106934
```

Problem2

(a)data preparing

```
library(datasets)
data(iris)
my.iris <- iris
whole.cv.list2 = cv.group(k=10,nrow(iris),2)
```

(b)1-NN

```
#1-NN
library(class)

cv.test.1nn<-function(i,data,cv.list){
  data.train <- data[-cv.list[[i]],]
    data.test<- data[cv.list[[i]],]

    pred.1NN = knn(train = data.train[,1:4], test = data.test[,1:4], cl = data.train$Species, k=1)
    test.error.1NN <- mean(pred.1NN!=data.test$Species)
    return(test.error.1NN)
}

all.test.error.1NN <- NULL
for(i in range(1:10)){
  i.test.error.1NN <- cv.test.1nn(i,iris,whole.cv.list2)
  all.test.error.1NN <- c(all.test.error.1NN,i.test.error.1NN)
}
test.error.1NN <- mean(all.test.error.1NN)
test.error.1NN
```

```
## [1] 0.03333333
```

(c)LDA

```
cv.test.LDA <- function(i,data,cv.list){
  data.train <- data[-cv.list[[i]],]
    data.test<- data[cv.list[[i]],]

    fit.LDA = lda( Species ~ ., data.train)
    pred.LDA = predict(fit.LDA, newdata=data.test[,1:4])$class
    test.error.LDA <- mean(pred.LDA != data.test$Species)
    return(test.error.LDA)
}

all.test.error.LDA <- NULL
for(i in range(1:10)){
  i.test.error.LDA <- cv.test.LDA(i,iris,whole.cv.list2)
  all.test.error.LDA <- c(all.test.error.LDA,i.test.error.LDA)
```

```
}
test.error.LDA <- mean(all.test.error.LDA)
test.error.LDA
```

```
## [1] 0.03333333
```

(d)QDA

```
cv.test.QDA<-function(i,data,cv.list){
  data.train <- data[-cv.list[[i]],]
    data.test<- data[cv.list[[i]],]

    fit.QDA = qda( Species ~ ., data.train)
    pred.QDA = predict(fit.QDA, newdata=data.test[,1:4])$class
    test.error.QDA <- mean(pred.QDA != data.test$Species)
    return(test.error.QDA)
}

all.test.error.QDA <- NULL
for(i in range(1:10)){
  i.test.error.QDA <- cv.test.QDA(i,iris,whole.cv.list2)
  all.test.error.QDA <- c(all.test.error.QDA,i.test.error.QDA)
}
test.error.QDA <- mean(all.test.error.QDA)
test.error.QDA
```

```
## [1] 0.06666667
```