HW7

Dhondup Dolma

12/4/2022

```
dat=read.table("meatspec.txt")
dat = as.matrix(dat[,-1:-20])
dat = as.matrix(dat[,-61:-80])



get.std.mat=function(X)
{
  mx = apply(X[,-1], 2, mean)
  sx = apply(X[,-1], 2, sd)
  A = diag(ncol(X))
  A[1,-1]=-mx
  B=diag(c(1,1/sx))
  return(A%*%B)

}
ridge=function(X,y, lam, std=TRUE, use.svd=(ncol(X) >= nrow(X)))
{
p=ncol(X)
if(std)
{
T=get.std.mat(X)
X=X%*%T
}
if(!use.svd)
{
M=diag(c(0, rep(1,p-1)))
bhat=qr.solve(crossprod(X)+lam*M, crossprod(X,y))
if(std) bhat=T%*%bhat
} else
{
n=nrow(X)
xbart=crossprod(rep(1,n), X[,-1])/n
ybar=mean(y)
Xc= X[,-1] - rep(1,n)%*%xbart
yc=y-ybar
## compute the reduced SVD of Xc
q=min(c(n-1, p-1))
out=svd(Xc, nu=q, nv=q)
H=diag(out$d[1:q]/(out$d[1:q]^2 + lam))
bhatm1=out$v%*%H%*%t(out$u)%*%yc
bhat1=ybar - (xbart%*%bhatm1)[1]
```

```
bhat=c(bhat1, bhatm1)
if(std) bhat=T%*%bhat
}
return(bhat)
}
ridgecv=function(X, y, K=5, permute=FALSE, std=TRUE,
lam.vec=10^seq(from=-8, to=8, by=0.5))
{
n=length(y)
p=ncol(X)
if(permute) ind=sample(n) else ind=1:n
val.sq.err=numeric(length(lam.vec))
for(k in 1:K)
{
leave.out=ind[ (1+floor((k-1)*n/K)):floor(k*n/K) ]
X.tr=X[-leave.out,,drop=FALSE]; y.tr=y[-leave.out]
X.va=X[leave.out,,drop=FALSE]; y.va=y[leave.out]
if(std)
{
  T=get.std.mat(X.tr)
X.tr=X.tr%*%T
}
n.tr=length(y.tr)
## compute the centered training predictor matrix
xbart=crossprod(rep(1,n.tr), X.tr[,-1])/n.tr
ybar=mean(y.tr)
Xc= X.tr[,-1] - rep(1,n.tr)%*%xbart
## compute the centered training response vector
yc=y.tr-ybar
## compute the reduced SVD of Xc
q=min(c(n.tr-1, p-1))
out=svd(Xc, nu=q, nv=q)
for(j in 1:length(lam.vec))
{
## compute beta.hat^(lam.vec[j], -k)
H=diag(out$d[1:q]/(out$d[1:q]^2 + lam.vec[j]))
bhatm1=out$v%*%H%*%t(out$u)%*%yc
bhat1=ybar - (xbart%*%bhatm1)[1]
bhat.tr=c(bhat1, bhatm1)
if(std) bhat.tr=T%*%bhat.tr
## compute its validation error
val.sq.err[j] = val.sq.err[j] + sum((y.va - X.va%*%bhat.tr)^2 )
}
}
best.lam=lam.vec[which.min(val.sq.err)]
## compute the ridge estimator with the selected tuning parameter
## using all the data
beta.hat=ridge(X=X,y=y, lam=best.lam, std=std, use.svd=TRUE)
return(list(best.lam=best.lam, beta.hat=beta.hat, val.sq.err=val.sq.err))
}
X = cbind(1,as.matrix(dat[,-61]))
train=1:129
test=130:215
```

```
y = as.numeric(dat[,61])
lam.vec=10^seq(from=-8, to=6, by=0.5)
out=ridgecv(X=X[train,], y=y[train], K=5, std=TRUE, permute=FALSE, lam.vec=lam.vec)

log10(out$best.lam)
```

```
## [1] -5.5
```

```
rmpse=function(a,b) return(sqrt(mean((a-b)^2)))

beta.hat = qr.coef(qr(X[train,]), y=y[train])
result= matrix(NA, nrow=2, ncol=1)
result[1,1] = rmpse(y[test], X[test,]%*%beta.hat)
result[2,1] = rmpse(y[test], X[test,]%*%out$beta.hat)
colnames(result) = "Testing RMPSE"
rownames(result) = c("Least squares", "Ridge-std-5-fold CV")
result
```

```
##                       Testing RMPSE
## Least squares              3.591686
## Ridge-std-5-fold CV        2.384019
```

This data set has significantly better ridge regression estimators than the data set with all 100 waveleghts. This also data set also differs compare to the in class example, because the least squares beta estimator is closer to ridge regression estimators beta(lamda,std) which means that the prediction using the wavelength from 21st to 80th is better.

```
#2a
library(glmnet)
```

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

```
## Loaded glmnet 4.1-6
```

```
set.seed(3301)
n=200; p=100; rho=1;beta=rep(0,p)
X = cbind(1, matrix(rnorm(n*(p-1)), nrow=n, ncol=p-1))
y = X%*%beta + rnorm(n)
lam.vec=10^seq(from=-6, to=6, by=0.5)
cvout = cv.glmnet(x=X[,-1], y=y, nfolds=5, standardize=TRUE, lambda=lam.vec)
log10(cvout$lambda.min)
```

```
## [1] 6
```

```
out = glmnet(x=X[,-1], y=y, lambda= log10(cvout$lambda.min), standardize = TRUE)
bhat = c(as.numeric(out$a0), as.numeric(out$beta))
print("nonzero betahat")
```

```
## [1] "nonzero betahat"
```

```
length(which(bhat != 0))
```

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

```
#2b
reps =100
set.seed(3301)
X = cbind(1,matrix(rnorm(n*(p-1)), nrow =n , ncol=p-1))
number.nonzero=numeric(reps)
```

```
for(i in 1:reps){
  y = X%*%beta + rnorm(n)
  cvout = cv.glmnet(x=X[,-1], y = y, nfolds = 5, standardize = TRUE, lambda= lam.vec)
  out = glmnet(x=X[,-1], y=y, lambda=cvout$lambda.min, standardize= TRUE)
  number.nonzero[i] = length(which(out$beta[-1]!=0))
}
alpha=.05
num.mean = mean(number.nonzero)
num.sd = sd(number.nonzero)
conf.interval = num.mean+c(-1,1)*qt(1-alpha/2, reps-1)*num.sd/sqrt(reps)
print("confidence interval")
```

```
## [1] "confidence interval"
```

```
conf.interval
```

```
## [1] 0.9566204 2.9233796
```

```
#3a
trainingdata = read.table("wbca-train.txt")
fit=glm(trainingdata$Class~trainingdata$Adhes+trainingdata$BNucl+trainingdata$Chrom+trainingdata$Epith+
coef(fit)
```

```
##        (Intercept) trainingdata$Adhes trainingdata$BNucl trainingdata$Chrom
##        11.31706210        -0.39453536        -0.36281736        -0.60079835
## trainingdata$Epith trainingdata$Mitos trainingdata$NNucl trainingdata$Thick
##        -0.10083408        -0.26079069        -0.26074801        -0.73732474
## trainingdata$UShap trainingdata$USize
##        -0.20433881         0.02325025
```

#3b is in pdf

```
#3c
eval.all.subsets=function(X, y)
{
if(is.null(colnames(X)))
columnnames=2:ncol(X)
else
columnnames=colnames(X)[-1]
X=as.matrix(X); y=as.numeric(y)
name.vec=c("AIC", "BIC", columnnames)
keep.status=as.matrix(expand.grid( replicate(ncol(X)-1, c(0,1), simplify=FALSE) ))
aicbic=matrix(NA, nrow=nrow(keep.status), ncol=2)
for(j in 1:nrow(keep.status))
{
keep=as.logical(c(1, keep.status[j,]))
fit=glm(y ~ X[,keep]-1, family=binomial)
aicbic[j,] = c(AIC(fit), BIC(fit))
}
result=cbind(aicbic, keep.status)
colnames(result)=name.vec
ind.aic=which.min(aicbic[,1])
ind.bic=which.min(aicbic[,2])
name.vec=name.vec[-c(1,2)]
best_result=result[c(ind.aic, ind.bic),]
rownames(best_result)=c("AIC picked", "BIC picked")
print(best_result)
```

```
return(result)
}
X = cbind(1,trainingdata$Adhes,trainingdata$BNucl,trainingdata$Chrom,trainingdata$Epith,trainingdata$Mi
y = trainingdata$Class
res = eval.all.subsets(X,y)
```

```
##                 AIC      BIC 2 3 4 5 6 7 8 9 10
## AIC picked 91.40176 117.3801 1 1 1 0 0 1 1 0  0
## BIC picked 91.40176 117.3801 1 1 1 0 0 1 1 0  0
```

The best subset is {Adhes, BNucl, Chrom, NNucl, Thick}

```
#3d
testdata= read.table("wbca-test.txt")
h = function(w) return(exp(w)/(1+exp(w)))
Xtest = cbind(1,testdata$Adhes,testdata$BNucl,testdata$Chrom,testdata$NNucl,testdata$Thick)
ytest = testdata$Class
fit=glm(trainingdata$Class~trainingdata$Adhes+trainingdata$BNucl+trainingdata$Chrom+trainingdata$NNucl+
est.prob.benign= h(Xtest%*%coef(fit))
est.Class=1*(est.prob.benign>0.5)
mean(est.Class==ytest)
```

```
## [1] 0.9833333
```

```
#.9833 is the proportion of  the test subjects were classified properly
```

```
#3e
X = cbind(1,trainingdata$Adhes,trainingdata$BNucl,trainingdata$Chrom,trainingdata$Epith,trainingdata$Mi
y = trainingdata$Class
lam.vec=10^seq(from=-8, to =8, by =.5)
n = nrow(X)
cvout = cv.glmnet(x=X[,-1], y =y, family="binomial", nfolds=5, lambda= lam.vec, alpha=0, standardize=TRU
cvout$lambda.min
```

```
## [1] 0.003162278
```

```
gfit = glmnet(x=X[,-1], y =y, family="binomial", lambda=cvout$lambda.min, alpha=0,standardize= TRUE)
bhat.best =c(as.numeric(gfit$a0), as.numeric(gfit$beta))
X.test = cbind(1,testdata$Adhes,testdata$BNucl,testdata$Chrom,testdata$Epith,testdata$Mitos,testdata$NNu
y.test = testdata$Class
est.prob.benign.ridge = h(X.test%*%bhat.best)
print("the proportion of the test subjects that were correctly classified using ridge penalized method"
```

```
## [1] "the proportion of the test subjects that were correctly classified using ridge penalized method
```

```
est.Class.ridge=1*(est.prob.benign.ridge>0.5)
mean(est.Class.ridge==y.test)
```

```
## [1] 0.9833333
```

```
log10(cvout$lambda.min)
```

```
## [1] -2.5
```

```
#3f
lam.vec= 10^seq(from =-8, to =8, by =.5)
n = nrow(X)
cvout.lasso= cv.glmnet(x=X[,-1], y =y, family="binomial", nfolds=5, lambda= lam.vec, alpha=1, standardi
cvout.lasso$lambda.min
```

```
## [1] 0.003162278
```

```
gfit.lasso = glmnet(x=X[,-1], y =y, family="binomial",lambda= cvout$lambda.min, alpha=1, standardize=TRU
gfit.lasso$beta
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##             s0
## V1 -0.30397257
## V2 -0.33352205
## V3 -0.47649153
## V4 -0.07995549
## V5  .
## V6 -0.21053303
## V7 -0.60976896
## V8 -0.21497956
## V9  .
```

```
bhat.lasso =c(as.numeric(gfit.lasso$a0), as.numeric(gfit.lasso$beta))
est.prob.benign.lasso= h(X.test%*%bhat.lasso)
est.Class.lasso = 1*(est.prob.benign.lasso>.5)
print("the proportion of the test subjects that were correctly classified using ridge penalized method")
```

```
## [1] "the proportion of the test subjects that were correctly classified using ridge penalized method
```

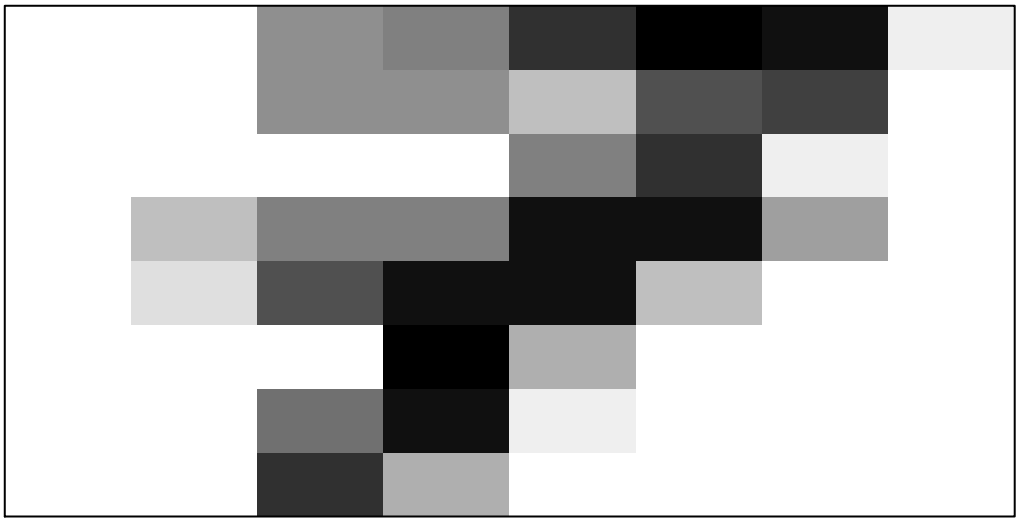
```
mean(est.Class.lasso==y.test)
```

```
## [1] 0.9833333
```

The predictors that the lasso method automatically removed are Mitos and USize The best subset selected by AIC removed Epith, Mitos, UShap, and USize, while the lasso only removed Mitos and USize. The proportion of test subjects that the lasso method correctly classified by the AIC-selected subset fit by maximum likelihood, the ridge-penalized and the lasso-penalized likelihood method are all .9833333, so they have the same performance in this case.

```
#4a
dat.train = read.table("digits-train.txt")
dat.test = read.table("digits-test.txt")
x1.train = cbind(1, as.matrix(dat.train[which(dat.train$digit==1), -65]))
x7.train = cbind(1, as.matrix(dat.train[which(dat.train$digit==7), -65]))
X.train = rbind(x1.train, x7.train)
y.train = c(rep(1, nrow(x1.train)), rep(0, nrow(x7.train)))
x1.test= cbind(1, as.matrix(dat.test[which(dat.test$digit==1), -65]))
x7.test= cbind(1, as.matrix(dat.test[which(dat.test$digit==7), -65]))
X.test = rbind(x1.test, x7.test)
y.test = c(rep(1,nrow(x1.test)), rep(0, nrow(x7.test)))
```

```
#4b
display.digit=function(x){
  imat = matrix(x[-1], nrow=8, ncol=8)
  image(z=imat[,8:1], col=grey((16:0)/16), axes=FALSE)
  box()
}
display.digit(x=x7.test[1,])
```



```
#4c
to.remove=which(apply(X.train, 2, var)==0)[-1]
to.remove
```

```
##  V1  V9 V33 V40 V41 V48 V49 V57
##   2  10  34  41  42  49  50  58
```

```
X.train = X.train[,-to.remove]
X.test= X.test[,-to.remove]
dim(X.train)
```

```
## [1] 776  57
```

```
dim(X.test)
```

```
## [1] 361  57
```

```
#4d
lam.vector = 10^seq(from=-6, to =6, by=.5)
cvout = cv.glmnet(x=X.train[,-1], y=y.train, nfolds=5, family="binomial", lambda=lam.vector, alpha=1, s
print("the selected tuning parameter")
```

```
## [1] "the selected tuning parameter"
```

```
cvout$lambda.min
```

```
## [1] 0.001
```

```
gfit=glmnet(x=X.train[,-1], y=y.train, family="binomial", lambda=cvout$lambda.min, alpha=1, standardize=
beta=c(as.numeric(gfit$a0), as.numeric(gfit$beta))
print("the number of nonzero entires in the estimate")
```

```
## [1] "the number of nonzero entires in the estimate"
```

```
length(which(beta!=0))
```

```
## [1] 14
```

```
#4e
est.prob = h(X.test%*%beta)
yhat.test = 1*(est.prob>=0.5)
print("the proportion of the test subjects that were correctly classified ")
```

```
## [1] "the proportion of the test subjects that were correctly classified "
```

```
mean(yhat.test == y.test)
```

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