**HW 2**

**2015129053 김형철**

1.

(a)

**I simulated the correlated data sets using the following codes:**

*rm(list=ls())*

*N=150*

*P=50*

*X=matrix(NA,nrow=N,ncol=P)*

*#correlated predictors*

*set.seed(100)*

*covmat=matrix(rnorm(P^2,sd=2),nrow=P)*

*covmat=covmat+t(covmat)*

*U=eigen(covmat)$vectors*

*D=diag(rexp(P,rate=10))*

*covmat=U%\*%D%\*%t(U)*

*library(mvtnorm)*

*set.seed(100)*

*for(i in 1:N){*

*X[i,]=rmvnorm(1,mean=rep(0,P),sigma=covmat)*

*}*

*X=data.frame(X)*

*betas.true=c(1,2,3,4,5,-1,-2,-3,-4,-5,rep(0,P-10))*

*#simulate the y*

*sigma=15.7*

*X=as.matrix(X)*

*set.seed(100)*

*y=X%\*%betas.true+rnorm(N,mean=0,sd=sigma)*

**Then I divided the data sets into training and test group using these codes:**

*##split the date to test and train*

*alldata=data.frame(cbind(y,X))*

*names(alldata)[1] <- "y"*

*train=alldata[1:100,]*

*test=alldata[101:150,]*

**Now, I fit the ordinary linear regression using training data sets**

*#fit the ordinary linear regression using training data sets. calculate the variance inflation factor.*

*fit=lm(y~.,data=train)*

*summary(fit)*

*betas.lm=coef(fit)*

A picture containing table

Description automatically generated

**Then I checked the variance inflation factor using this code:**

*library(car)*

*vif(fit)*

A picture containing text, scoreboard

Description automatically generated

**It seems that there are few variables that have multicollinearity issues. If the value of variance inflation factor is bigger than or equal to 10, it is assumed that there are multicollinearity issues. In that sense, there are few variables with values bigger than 10.**

(b)

**I used following cv.glmnet and glmnet functions codes to get the best lambda value for the ridge regression:**

*library(glmnet)*

*## fit ridge (trying 100 different lambda values)*

*rr=glmnet(x=as.matrix(train[,-1]),y=as.numeric(train[,1]),alpha=0,nlambda=100)*

*## use 10-fold crossvalidation to find the best lambda*

*cv.rr=cv.glmnet(x=as.matrix(train[,1]),y=as.numeric(train[,1]),alpha=0,nfolds=10,nlambda=100)*

*## get lambda and best rr fit*

*lambda.rr=cv.rr$lambda.min*

*lambda.rr*

Text

Description automatically generated

**I then compared the estimate ridge regression coefficients and standard regression coefficients. Like in the lecture, the coefficients of ridge regression are closer to 0.**

*## compare the ridge coefficients with standard regression coefficients*

*plot(betas.rr,betas.lm,xlim=c(-6,6),ylim=c(-26,26))*

*abline(0,1)*

Chart, scatter chart

Description automatically generated

*## beta estimates for best lambda*

*betas.rr=coef(cv.rr,s="lambda.min")*

*betas.rr*

A picture containing graphical user interface

Description automatically generated

(c)

**I repeated (b) for lasso regression:**

*## fit lasso (trying 100 different lambda values)*

*lasso=glmnet(x=as.matrix(train[,-1]),y=as.numeric(train[,1]),alpha=1,nlambda=100)*

*## use 10-fold crossvalidation to find the best lambda*

*cv.lasso=cv.glmnet(x=as.matrix(train[,-1]),y=as.numeric(train[,1]),alpha=1,nfolds=10)*

*## get lambda and best lasso fit*

*lambda.lasso=cv.lasso$lambda.min*

*lambda.lasso*

*Text

Description automatically generated*

*## beta estimates for best lambda*

*betas.lasso=coef(cv.lasso,s="lambda.min")*

*par(mfrow=c(1,2))*

*plot(cv.lasso)*

*abline(v=log(lambda.lasso))*

*plot(lasso,xvar="lambda", main="Lasso Regression Betas for Different Values of the Tuning Parameter")*

*abline(v=log(lambda.lasso))*

Chart

Description automatically generated

**I then compared lasso regression coefficients and standard regression coefficients. Just like in the lecture, many coefficients of lasso regression are zero. This is the property of lasso regression.**

*plot(betas.lasso,betas.lm,xlim=c(-6,6),ylim=c(-55,55))*

*abline(0,1)*

Chart

Description automatically generated

(d) **Now I repeated (b) for alpha=0.5:**

*elastic=glmnet(x=as.matrix(train[,-1]),y=as.numeric(train[,1]),alpha=0.5,nlambda=100)*

*## use 10-fold cross validation to find the best lambda*

*cv.elastic=cv.glmnet(x=as.matrix(train[,1]),y=as.numeric(train[,1]),alpha=0.5,nfolds=10,nlambda=100)*

*## compare the elastic coefficients with standard coefficients*

*plot(betas.elastic,betas.lm,xlim=c(-6,6),ylim=c(-55,55))*

*abline(0,1)*

*Chart

Description automatically generated*

*## get lambda and best fit*

*lambda.elastic=cv.elastic$lambda.min*

*lambda.elastic*

*Text

Description automatically generated*

*## beta estimates for best lambda*

*betas.elastic=coef(cv.elastic,s="lambda.min")*

*betas.elastic*

*Graphical user interface

Description automatically generated with medium confidence*

**The objective function of elastic net is this:**

A picture containing text

Description automatically generated

**Elastic net regularization has several advantages over lasso. In fact, it was made to overcome the limitations of the LASSO. In the case of high-dimensional data that has few examples, lasso only selects those few examples (at most) before it saturates. Furthermore, lasso usually select only one variable from a group and ignore other variables when the group has highly correlated variables. To overcomes this situation, elastic net regularization added a quadratic part in the penalty.**

(e)

**If we compare the mean square prediction errors, we can see that the size of the error is**

**standard regression > ridge > elastic > lasso:**

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

2.

(a) **I scaled the variable sand divided the data into training and test sets using these codes:**

*#scale the variables*

*mydata=read.csv("BrainTumor.csv",sep=",", header=TRUE)*

*mydata12=mydata[,-1]*

*Class=mydata[,1]*

*scaled=scale(mydata12, center=TRUE, scale=TRUE)*

*mynewdata=cbind(Class, scaled)*

*mynewdata=as.data.frame(mynewdata)*

*rm(Class)*

*# split the data into 70, 30.*

*set.seed(100)*

*sample=sample.int(n = nrow(mynewdata), size = floor(nrow(mynewdata)\*0.70), replace = FALSE)*

*train=mynewdata[sample, ]*

*test=mynewdata[-sample, ]*

**I then fit the standard logistic regression:**

*# fit standard regression*

*attach(train)*

*fit=glm(Class~Mean+Variance+Standard.Deviation+Entropy+Skewness+Kurtosis+Contrast+Energy+ASM+Homogeneity+Dissimilarity+Correlation,family="binomial",data=train)*

*detach(train)*

*summary(fit)*

Text

Description automatically generated

(b) **I fitted a logistic regression with lasso penalty using training data set with these codes:**

*library(glmnet)*

*X=as.matrix(train[,-1])*

*y=train[,1]*

*lasso=glmnet(x=X,y=y,family="binomial",alpha=1,nlambda=100)*

*## use 10-fold crossvalidation to find the best lambda*

*cv.lasso=cv.glmnet(x=X,y=y,family="binomial",alpha=1,nfolds=10)*

*## get lambda and best lasso fit*

*lambda.lasso=cv.lasso$lambda.1se*

*lambda.lasso*

*Text

Description automatically generated*

*## cross validation plots*

*par(mfrow=c(1,2))*

*plot(cv.lasso)*

*abline(v=log(lambda.lasso))*

*plot(lasso,xvar="lambda")*

*abline(v=log(lambda.lasso))*

*Chart

Description automatically generated*

*## beta estimates for best lambda*

*betas.lasso=coef(cv.lasso)*

*betas.lasso*

*Text

Description automatically generated*

(c**) I fitted a logistic regression with ridge penalty using training data set with these codes:**

*X2=as.matrix(train[,-1])*

*y2=train[,1]*

*rr=glmnet(x=X2,y=y2,family="binomial",alpha=0,nlambda=100)*

*## use 10-fold crossvalidation to find the best lambda*

*cv.rr=cv.glmnet(x=X2,y=y2,family="binomial",alpha=0,nfolds=10)*

*## get lambda and best rr fit*

*lambda.rr=cv.rr$lambda.1se*

*lambda.rr*

Text

Description automatically generated with medium confidence

*## cross validation plots*

*par(mfrow=c(1,2))*

*plot(cv.rr)*

*abline(v=log(lambda.rr))*

*plot(rr,xvar="lambda")*

*abline(v=log(lambda.rr))*

Chart

Description automatically generated

*## beta estimates for best lambda*

*betas.rr=coef(cv.rr)*

*betas.rr*

Text

Description automatically generated

(d) **I predicted a response for test data sets using these codes:**

*#ordinary glm*

*testnew=test[,-1]*

*eta=predict(fit,newdata=testnew,type="response")*

*plot(c(1:1129),eta)*

*tumor\_class=eta*

*for (i in 1:1129) {*

*if (eta[i]>=0.5) {*

*tumor\_class[i]=1*

*} else {*

*tumor\_class[i]=0*

*}*

*}*

*plot(c(1:1129), tumor\_class)*

*real=test[,1]*

*prediction=tumor\_class*

*accuracy=real-prediction*

*accuracy=unname(accuracy)*

*accuracy*

*check=0*

*for (i in 1:1129) {*

*if (accuracy[i] != 0) {*

*check=check+1*

*}*

*}*

*calculator=(1129-check)/1129*

*calculator*

A picture containing text

Description automatically generated

*#lasso case*

*lasso\_prob=predict(cv.lasso, s="lambda.1se", newx=as.matrix(test[,-1]), type="response")*

*lasso\_prob*

*plot(c(1:1129),lasso\_prob)*

*tumor\_class2=lasso\_prob*

*for (i in 1:1129) {*

*if (lasso\_prob[i]>=0.5) {*

*tumor\_class2[i]=1*

*} else {*

*tumor\_class2[i]=0*

*}*

*}*

*plot(c(1:1129), tumor\_class2)*

*real=test[,1]*

*prediction=tumor\_class2*

*accuracy=real-prediction*

*accuracy=unname(accuracy)*

*accuracy*

*check=0*

*for (i in 1:1129) {*

*if (accuracy[i] != 0) {*

*check=check+1*

*}*

*}*

*calculator=(1129-check)/1129*

*calculator*

*A picture containing text

Description automatically generated*

*#ridge case*

*ridge\_prob=predict(cv.rr, s="lambda.1se", newx=as.matrix(test[,-1]), type="response")*

*ridge\_prob*

*plot(c(1:1129),ridge\_prob)*

*tumor\_class3=ridge\_prob*

*for (i in 1:1129) {*

*if (ridge\_prob[i]>=0.5) {*

*tumor\_class3[i]=1*

*} else {*

*tumor\_class3[i]=0*

*}*

*}*

*plot(c(1:1129), tumor\_class3)*

*real=test[,1]*

*prediction=tumor\_class3*

*accuracy=real-prediction*

*accuracy=unname(accuracy)*

*accuracy*

*check=0*

*for (i in 1:1129) {*

*if (accuracy[i] != 0) {*

*check=check+1*

*}*

*}*

*calculator=(1129-check)/1129*

*calculator*

*A picture containing graphical user interface

Description automatically generated*

**Considering the result, it seems standard logistic regression has the best prediction accuracy (98.4%). The lasso also has quite a nice accuracy (98.0%) but it is lower than that of standard logistic regression. Finally, the ridge regression seems to have the lowest accuracy (96.8%). It seems that the penalty method used in lasso and ridge does not always ensure better accuracy.**