Simulation of Prediction Modeling

Lecture 16 Handout

Statistics 139

Topics

- Large Simulation
- Sequential Variable Selection vs. Ridge vs. LASSO

The material in this lab corresponds to the Lecture 16 Notes.

In this lab we will be exploring the effects of various types of situation on the

Problem 1: Sampling Data: add in correlation

a) Let Z_1, Z_2, Z_3 all be i.i.d. $N(0,1)$ r.v.s, and let $\rho \in [0,1]$. Determine the joint distribution of (X_1, X_2) where $X_1 = (\sqrt{\rho})\, Z_1 + (\sqrt{1-\rho})\, Z_2$ and $X_2 = (\sqrt{\rho})\, Z_1 + (\sqrt{1-\rho})\, Z_3$.

b) Modify the code below to allow for a fixed correlation between all predictors in the resulting predictor matrix, `X`, by incorporating the result above (Note: this is computationally faster than using `mvrnorm`). Briefly check that the results agree with the theory.

```
generate.data = function(n = 500, p = 100, beta0 = 2, beta =
                           rep(0,p),sigma=5,rho=0){
  Z = matrix(rnorm(n*p),nrow=n,ncol=p)
  # create z1 and use it with Z above to create X


  y = beta0+X%*%beta+rnorm(n,sd=sigma)
  data=cbind(y=y,X=X)
  colnames(data) = c("y",paste("x",1:100,sep=""))
  return(as.data.frame(data))
}
```

c) Investigate the affect of correlation in the predictors on the methods seen in class today. Below is the relevant code from class (to perform the analyses and to perform). Note: we only

consider $p = 50$ here to speed up the `step` function. Suggested correlations: $\rho = 0, 0.4, 0.8$.

```
fit.step = function(data,test){
  n=nrow(data)
  p=ncol(data)-1
  MSEs.step.train=MSEs.step.test=rep(NA,p+1)
  coefs = matrix(0,nrow=p+1,ncol=p+1)
  rownames(coefs) = c("(Intercept)",names(data)[2:(p+1)])
  lmstep = lm(y~.,data)
  for(i in 1:(p+1)){
    coefs[rownames(coefs) %in% names(coef(lmstep)),i] = coef(lmstep)
    MSEs.step.train[i] = sum((predict(lmstep)-data$y)^2)/n
    MSEs.step.test[i] = sum((predict(lmstep,newdata=test)-test$y)^2)/n
    lmstep = step(lmstep,steps=1,k=100,trace=0)
  }

  return(list(MSEs.train = MSEs.step.train,MSEs.test = MSEs.step.test,beta=coefs))
}


fit.glmnet = function(data,test,alpha=1){
  n=nrow(data)

  lmtrain = lm(y~.,data=data)
  X = model.matrix(lmtrain)[,-1]
  lmtest = lm(y~.,data=test)
  Xtest = model.matrix(lmtest)[,-1]

  fits = glmnet(X,data$y,alpha=alpha)
  MSEs.fits.train = apply((predict(fits,X)-data$y)^2,2,sum)/n
  MSEs.fits.test = apply((predict(fits,Xtest)-test$y)^2,2,sum)/n

  return(list(MSEs.train = MSEs.fits.train,
              MSEs.test = MSEs.fits.test,
              lambdas=fits$lambda,beta=fits$beta))
}

plot.MSE = function(steps,ridges,lassos){
  par(mfrow=c(1,3))
  x = 100:0

  ylimit = c(min(steps$MSEs.train,steps$MSEs.test),
             max(steps$MSEs.train,steps$MSEs.test))
  plot(steps$MSEs.train~x,type="l",
       ylim=ylimit,
       xlab="number of predictors",
       ylab="MSE",main="Backwards Step")
  lines(steps$MSEs.test~x,col="darkorange")
```

```
  ylimit = c(min(ridges$MSEs.train,ridges$MSEs.test),
             max(ridges$MSEs.train,ridges$MSEs.test))
  plot(ridges$MSEs.train~log(ridges$lambda),type="l",
       ylim=ylimit,main="Ridge")
  lines(ridges$MSEs.test~log(ridges$lambda),col="chartreuse")

  ylimit = c(min(lassos$MSEs.train,lassos$MSEs.test),
             max(lassos$MSEs.train,ridges$MSEs.test))
  plot(lassos$MSEs.train~log(lassos$lambda),type="l",
       ylim=ylimit,main="LASSO")
  lines(lassos$MSEs.test~log(lassos$lambda),col="hotpink")
}
```

```
#install.packages("glmnet")
library(glmnet)

data0 <- generate.data(rho=0)
test0 <- generate.data(rho=0)

steps0 = fit.step(data0,test0)
ridges0 = fit.glmnet(data0,test0,alpha=0)
lassos0 = fit.glmnet(data0,test0,alpha=1)

plot.MSE(steps0,ridges0,lassos0)
```

```
# play around with correlation
```

Problem 2: Playing around with sample size

a) In the lecture notes we used $n = 500$ and $p = 100$ (above it was $n = 500,\ p = 50$. What would happen if instead $n = 100$ and $p = 50$? What about if $n = 2000$ and $p = 50$?

b) Investigate the effects of the ratio of n to p based on the conditions above.

Problem 3: Playing around with $\vec{\beta}$

a) In what situation did backwards sequential variable selection 'win' out? In what situations did Ridge win out? In what situation did LASSO win out? For the method that did not win any situation, come up with a setting in which it might.

b) Modify the simulation code so that the true vector of $\vec{\beta}$ is sampled in a way to address whether the setting you propose in part (a) results in the winner you expect.

c) Interpret the results of the simulation above.

d) In a real-life application what are the advantages to each method, and which method would you use? How should you compare the methods? What can this comparison tell you about the data structure?