ML HW1

Ryan Gallagher

2023-09-29

### Generate a training data set with n = 500 independent samples from the following logistic regression:

where , , , , , and

set.seed(1337)  
  
n=500  
b0 = -1  
b1 = 1  
b2 = -1  
x1 = rnorm(500)  
x2 = rnorm(500)  
x3 = runif(500)  
  
y\_i = b0 + b1 \* x1 \* x2 + exp(b2 \* x3)  
prob\_y\_i = exp(y\_i) / (1+exp(y\_i))  
  
y = rbinom(n,1, prob\_y\_i)  
x = data.frame(x1 = x1, x2 = x2, x3 = x3)  
  
head(x)

## x1 x2 x3  
## 1 0.1924919 0.1708936 0.6696283938  
## 2 -1.4467018 -0.7152583 0.7409912099  
## 3 -0.3231805 -0.2272849 0.7035011353  
## 4 1.6222961 0.0754860 0.0003537976  
## 5 -0.6890241 1.4837221 0.3694174737  
## 6 2.0421222 -0.3116930 0.6910913778

head(y)

## [1] 1 0 0 1 0 0

df = data.frame(x1=x1, x2=x2, x3=x3, y=y)

### 

### (1) Run 10-fold CV for KNN classifier with k=3,4,…,20 to select the optimal k

K = 10 #For 10 Fold CV  
xg = sample(rep(1:K,times=rep(n/K,K)))  
mse = matrix(0,K,18)  
  
for(k in 1:K){  
 trainx = x[-which(xg==k),]  
 trainy = y[-(1:n)[xg==k]]  
 testx = x[which(xg==k),]  
 testy = y[(1:n)[xg==k]]  
   
 for(lil\_k in 3:20){  
 knnreg = knn.reg(train=trainx,test=testx,y=trainy,  
 k=lil\_k) #knn=lil\_k (3,4,...,20)  
 knnpred=knnreg$pred  
 mse[k,lil\_k-2] = 1/(length(testy))\*sum((testy-knnpred)^2)  
 }  
}  
apply(mse,2,mean)

## [1] 0.2953333 0.2653750 0.2541600 0.2423333 0.2379184 0.2329688 0.2318272  
## [8] 0.2278600 0.2270579 0.2252500 0.2229112 0.2227653 0.2195200 0.2179844  
## [15] 0.2184706 0.2163210 0.2173684 0.2168700

print(min(apply(mse,2,mean)))

## [1] 0.216321

Here, we find that **k=18 yields the lowest test MSE**. Thus, k=18 is our optimal k.  
**NOTE:** If I change seed, sometimes I’m getting k=17 or k=20. The difference is pretty negligable since their MSE is always pretty close.

### 

### Run LDA, QDA, and naive Bayes. Also run the logistic regression as follows:

Calculate training error rates for KNN, LDA, QDA, naive Bayes, and the logistic regression using the training data set. Which classifier is the best?

### LDA

df$train <- ifelse(runif(nrow(df)) < 0.8, TRUE, FALSE)  
  
lda.fit=lda(y~x1+x2+x3, data=df, subset=train)  
lda.fit

## Call:  
## lda(y ~ x1 + x2 + x3, data = df, subset = train)  
##   
## Prior probabilities of groups:  
## 0 1   
## 0.5456853 0.4543147   
##   
## Group means:  
## x1 x2 x3  
## 0 -0.06565143 -0.001612948 0.5273808  
## 1 0.06169653 0.112447743 0.4354995  
##   
## Coefficients of linear discriminants:  
## LD1  
## x1 0.3469896  
## x2 0.2440342  
## x3 -3.0091016

lda.pred = predict(lda.fit, df[df$train == 0,])  
lda.class=lda.pred$class  
table(lda.class,df[df$train == 0,]$y)

##   
## lda.class 0 1  
## 0 48 25  
## 1 19 14

mean(lda.class==df[df$train == 0,]$y)

**## [1] 0.5849057**

### QDA

qda.fit=qda(y~x1+x2+x3, data=df, subset=train)  
qda.fit

## Call:  
## qda(y ~ x1 + x2 + x3, data = df, subset = train)  
##   
## Prior probabilities of groups:  
## 0 1   
## 0.5456853 0.4543147   
##   
## Group means:  
## x1 x2 x3  
## 0 -0.06565143 -0.001612948 0.5273808  
## 1 0.06169653 0.112447743 0.4354995

qda.class = predict(qda.fit, df[df$train == 0,])$class  
  
table(qda.class,df[df$train == 0,]$y)

##   
## qda.class 0 1  
## 0 53 19  
## 1 14 20

mean(qda.class==df[df$train == 0,]$y)

**## [1] 0.6886792**

### Naive Bayes

nb.fit = naiveBayes(y~x1+x2+x3, data=df, subset=train)  
nb.class = predict(nb.fit, df[df$train == 0,])  
table(nb.class, df[df$train == 0,]$y)

##   
## nb.class 0 1  
## 0 45 27  
## 1 22 12

mean(nb.class==df[df$train == 0,]$y)

**## [1] 0.5377358**

### Logistic Regression

train = df[df$train==1,]  
test = df[df$train==0,]  
  
logi.fit = glm(y~x1+x2+x3, data=train, family=binomial())  
test$y\_resp = predict(logi.fit, test,type='response')  
test$y\_pred <- ifelse(test$y\_resp > 0.5, 1, 0)  
  
table(test$y, test$y\_pred)

##   
## 0 1  
## 0 48 19  
## 1 25 14

mean(test$y == test$y\_pred)

**## [1] 0.5849057**

We find that **QDA** is the best classifier.

### Generate a test data set with n = 200 and calculate test error rates for KNN, LDA, QDA, naive Bayes, and the logistic regression. Which classifier is the best?

n\_2=200  
b0 = -1  
b1 = 1  
b2 = -1  
x1 = rnorm(200)  
x2 = rnorm(200)  
x3 = runif(200)  
  
y\_i = b0 + b1 \* x1 \* x2 + exp(b2 \* x3)  
prob\_y\_i = exp(y\_i) / (1+exp(y\_i))  
  
y = rbinom(n\_2,1, prob\_y\_i)  
x = data.frame(x1 = x1, x2 = x2, x3 = x3)  
  
df\_2 = data.frame(x1=x1, x2=x2, x3=x3, y=y)  
  
#KNN  
knnreg = knn.reg(train=trainx,test=subset(df\_2,select=c(x1,x2,x3)),y=trainy,  
 k=18) #proven optimal k  
knn\_pred = knnreg$pred  
knn\_y = ifelse(df\_2$knn\_pred > 0.5, 1, 0)  
mean(df\_2$y == knn\_y) #=0.61

## [1] 0.61

#LDA  
lda.pred = predict(lda.fit, subset(df\_2,select=c(x1,x2,x3)))$class  
mean(df\_2$y == lda.pred) #=0.52

## [1] 0.52

#QDA  
df\_2$qda.pred=predict(qda.fit,subset(df\_2,select=c(x1,x2,x3)))$class  
mean(df\_2$y == df\_2$qda.pred) #=0.66

## [1] 0.66

#Naive Bayes  
df\_2$nb.class = predict(nb.fit, subset(df\_2,select=c(x1,x2,x3)))  
mean(df\_2$y == nb.class) #=0.52

## [1] 0.52

#Logistic Regression  
df\_2$logi = predict(logi.fit, subset(df\_2,select=c(x1,x2,x3)),type='response')  
df\_2$logi\_pred <- ifelse(df\_2$logi > 0.5, 1, 0)  
mean(df\_2$y == df\_2$logi\_pred) #=0.56

## [1] 0.525

Once again, **QDA** is the best classifier - though the margin is not by as much with this data. Also – after running this data many times, I can see the variability in these models. The accuracy can stretch from ~0.61 to ~0.66 for something like QDA.