HW 3 M574

Shrey

2022-09-30

1

The paper compares LDA and logistic regression over binary classification and multi-class classification when independent vectors (X) may or may not contain categorical data. Logistic regression is more general and can work for categorical data assumptions as well as multivariate normal data assumption in contrast to LDA which works for only multivariate normal data with equal covariance matrix.

MLE for logistic regression are slower to compute and their accuracies are 50-66% of that of LDA only when the multivariate normal assumptions are met. However for general data MLEs are more robust. LDA might include redundant features or produce non zero slop when there is no relation (unique solutions) in the data. MLE logistic regression produces a warning in this case and drops redundant features. MLE Logistic regression's sum of predicted values is equal to sum of observed values, which is not the case in LDA, moreover LDA is sometimes biased.

When trained on breast cancer data (mix of continuous and categorical data) and US census population data, Logistic regression performed better although taking higher time to train.

2

(a)

Bayes rule for classification compares the ratio posterior probabilities of Y_i

Since probabilities are non negative values the ratio can be between zero and positive infinity, but in regression models the modeled response

$$\hat{Y}$$

can take values from negative infinity to positive infinity, we take the log odds of the posterior probabilities and model them as linear models.

$log(P(Y = 1|X)/(1 - P(Y = 1|X)) = \beta_0 + \beta_1^T X$

(b)

From the above equation, taking anti-log on both sides

$P(Y = 1|X)/(1 - P(Y = 1|X)) = e^{\beta_0 + \beta_1^T X}$

$P(Y = 1|X) = e^{\beta_0 + \beta_1^T X}/(1 + e^{\beta_0 + \beta_1^T X})$

(c)

Baye's rule for classifiction under 0,1 Loss.

Yi will be 1 if $P(Y=1|X)/(1-P(Y=1|X)) >= 1$

by taking log on both sides we get

$\beta\_0 + \beta_1^T X \quad >= 0$

3

(a)

Baye's classifiation rule under equal covariance matrix

$P(Y=1|X) = \pi_1 N(\mu 1, \sum_1)$

$P(Y=0|X) = \pi_0 N(\mu 0, \sum\_0)$

$N(\mu_j, \sum_j)$ is symbol for Multivariate Gaussian distribution Where μ_j is class mean vector and $\sum_j$ is class covariance matrix

Given $\sum_0 = \sum_1 = \sum$

Take log odds of posterior probabilities

$-> log(P(Y=1|X)/(1-P(Y=1|X)) = log(\frac{\pi_1 N(\mu 1, \sum_1)}{\pi_0 N(\mu 0, \sum_0)})$

$-> log(P(Y=1|X)/(1-P(Y=1|X)) = log(\frac{\pi_1}{\pi_0}) + log(\frac{N(\mu_1, \sum_1)}{N(\mu_0, \sum_0)})$

Given $\sum_0 = \sum_1 = \sum$

Because of equal covariance the sqrt(2pi)^p and quadratic parts of X will get cancelled and we get

$-> log(P(Y=1|X)/(1-P(Y=1|X)) = log(\frac{\pi_1}{\pi_0}) - 0.5(X-\mu_1)^T \sum^{-1}(X-\mu_1) + 0.5(X-\mu_0)^T \sum^{-1}(X-\mu_0)$

$log(P(Y=1|X)/(1-P(Y=1|X)) = log(\frac{\pi_1}{\pi_0}) + 0.5(\mu 1 + \mu_0)^T \sum^{-1}(\mu 1 - \mu_0) + X^T \sum^{-1}(\mu 1 - \mu_0)$

The above expression is linear model with below parameters $\beta_0 = log(\frac{\pi_1}{\pi_0}) + 0.5(\mu 1 + \mu_0)^T \sum^{-1}(\mu 1 - \mu_0)$ $\beta_1 = \sum^{-1}(\mu 1 - \mu_0)$

(b)

When $sum{0} =sum{1} $ sqrt(2pi)^p and quadratic parts of X will not get cancelled and we get

$log(P(Y=1|X)/(1-P(Y=1|X)) = log(\frac{\pi_1}{\pi_0}) - 0.5log(\frac{|\sum_1|}{|\sum_0|}) - 0.5(X-\mu_1)^T \sum_1^{-1}(X-\mu_1) + 0.5(X-\mu_0)^T \sum_0^{-1}(X-\mu_0)$

(c)

The result in (a) is linear but in part(b) will contain quadratic terms of X as well. Therefore decision boundary in part(a) will be a line but in part(b) it will be a quadratic curve.

4

(a)

Volume of p-dimensional ball is proportional to radiusˆp. It contains total N points uniformly distributed in p dimensions

Ri = distance from the origin to the i'th data point

Probability of finding a point within distance

$ CDF(R) = rˆp $

PDF of R = pr^{p-1}

PDF of First order statistic minimum

$N[1-r^p]^{N-1}pr^{p-1}$

For Median distance $ int_{0}ˆ{R} N[1-rˆ{p}]$^{\text{\{N-1\}pr}}${p-1} ,dr = 1/2 $

$(1-R^p)^N = 1/2$

$R = [1-(1/2)^{1/N}]^{1/p}$

(b)

When N= 500 and P =10

R = $[1-(1/2)^{1/500}]^{1/10}$ R = 0.5177921

When N =500 and P =100

R = $[1-(1/2)^{1/500}]^{1/10}$ R = 0.9363011

Median Distance to the closest point is 0.5177 when n is 500 and p is 10. When P is increased to R becomes 0.9363, therefore we need to cover 93% percent of the total to for the median of the closest point, thus the points become farther from the origin and more pushed towards the boundary.

5

(a)

```r
library(MASS)

set.seed(2000)
green<-mvrnorm(100,c(2,1),diag(2))
green<-data.frame(green)
red<-mvrnorm(100,c(1,2),diag(2))
red<-data.frame(red)

set.seed(2014)
green_test<-mvrnorm(500,c(2,1),diag(2))
green_test<-data.frame(green_test)
red_test<- mvrnorm(500,c(1,2),diag(2))
red_test<-data.frame(red_test)

## bind training and testing data
```

```r
train_data = rbind(green,red)
train_data["Y"] = c(rep(1,100),rep(0,100))

test_data = rbind(green_test,red_test)
test_data["Y"] =  c(rep(1,500),rep(0,500))

## train model
lda_model = lda(Y~.,prior = c(1/2,1/2), data = train_data)
y_predict = data.frame(predict(lda_model, newdata = train_data[c(1,2)])$class)
train_error = mean(y_predict!=train_data["Y"])
y_predict_test = data.frame(predict(lda_model, newdata = test_data[c(1,2)])$class)
test_error = mean(y_predict_test!=test_data["Y"])
print(c(train_error,test_error))
```

```
## [1] 0.220 0.235
```

(b)

```r
logistic_regression = glm(Y~.,family = binomial(link="logit"), data= train_data)
y_fitted = data.frame(as.numeric(logistic_regression$fitted.values>=0.5))
train_error = mean(y_fitted!=train_data["Y"])
y_predicted = data.frame(as.numeric(predict(logistic_regression,newdata = test_data[c(1,2)])>=0))
test_error = mean(y_predicted!=test_data["Y"])
print(c(train_error,test_error))
```

```
## [1] 0.215 0.236
```

```r
linear_model = lm(Y~X1+X2,data = train_data)
y_fitted = as.numeric(linear_model$fitted.values >= 0.5)
tarin_error = mean(y_fitted!= train_data["Y"])
y_predicted = as.numeric(predict(linear_model,newdata = test_data[c("X1","X2")])>=0.5)
test_error = mean(y_predicted!=test_data["Y"])
print(c(train_error,test_error))
```

```
## [1] 0.215 0.235
```

(c)

```r
errors = data.frame(c(0.215,0.220,0.215,0.22),c(0.239,0.235,0.236,0.235))
colnames(errors) <- c("Train","Test")
row.names(errors) <- c("Bayes","LDA","Logistic","Linear")
print(errors)
```

```
##          Train  Test
## Bayes    0.215 0.239
## LDA      0.220 0.235
## Logistic 0.215 0.236
## Linear   0.220 0.235
```

By comparing test errors we can see that LDA performed the best as it should because the multivariate Gaussian assumption was correct.

LDA out-performed logistic regression by low margin.

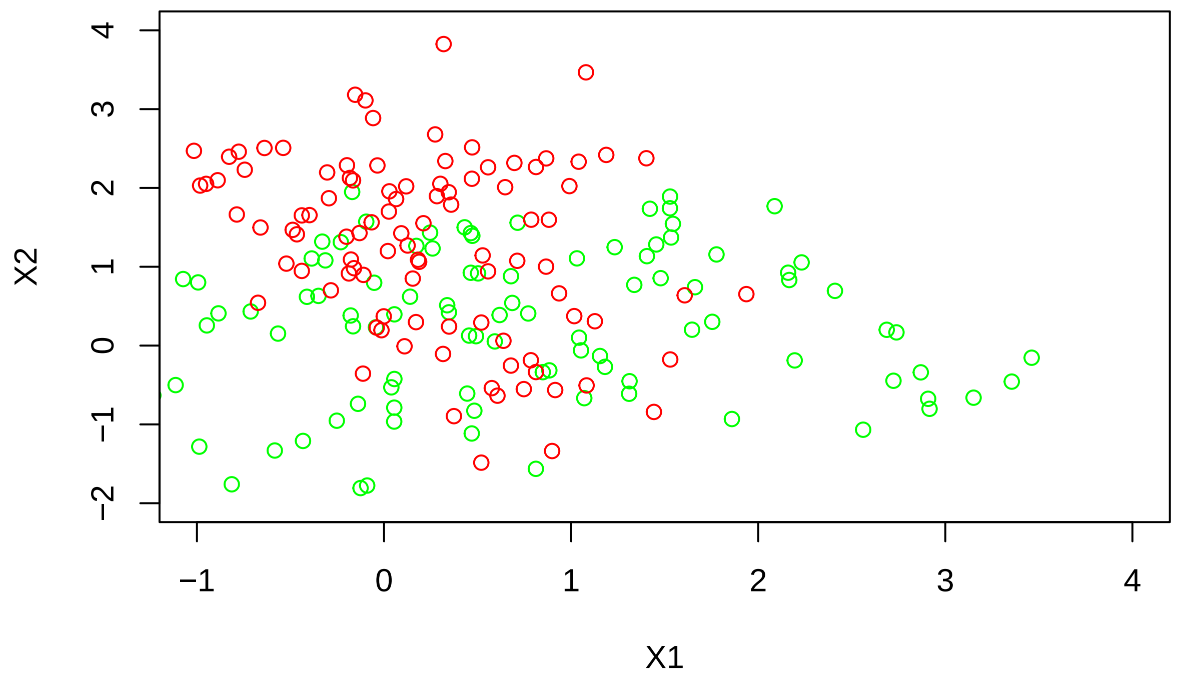
Linear regression and Logistic regression have equal test and train errors

6

(a)

```r
library(MASS)
#generate ten centers, which are treated as fixed parameters
Sig <- matrix(c(1,0,0,1),nrow=2)
seed_center <- 16
set.seed(seed_center)
center_green <- mvrnorm(n=10,c(1,0),Sig)
center_red <- mvrnorm(n=10,c(0,1),Sig)
##define a function "gendata2" first
gendata2 <-function(n,mu1,mu2,Sig1,Sig2,myseed)
{
set.seed(myseed)
mean1 <- mu1[sample(1:10,n,replace=T),]
mean2 <- mu2[sample(1:10,n,replace=T),]
green <- matrix(0,ncol=2,nrow=n)
red <- matrix(0,ncol=2,nrow=n)
for(i in 1:n){
green[i,] <- mvrnorm(1,mean1[i,],Sig1)
red[i,] <- mvrnorm(1,mean2[i,],Sig2)
}
x <- rbind(green,red)
return(x)
}
#generate the training set
seed_train <- 2000
ntrain <- 100
train2 <- gendata2(ntrain,center_green,center_red,Sig/5,Sig/5,seed_train)
ytrain <- c(rep(1,ntrain),rep(0,ntrain))
```

(b)



(c)

```
#generate the training set
seed_test <- 2014
ntest <- 500
test2 <- gendata2(ntest,center_green,center_red,Sig/5,Sig/5,seed_test)
ytest <- c(rep(1,ntest),rep(0,ntest))
```

7

(a)

```
train_df = data.frame(train2)
train_df["Y"] = ytrain
test_df = data.frame(test2)
test_df["Y"] = ytest
linear_model = lm(Y~X1+X2, data = train_df)
train_error = mean(as.numeric(linear_model$fitted.values>=0.5)!=ytrain)
y_predicted = as.numeric(predict(linear_model,newdata = test_df)>=0.5)
```

```
test_error = mean(y_predicted!=ytest)
print(c(train_error,test_error))
```

```
## [1] 0.310 0.295
```

(b)

```
lda_model = lda(train2,ytrain, cv= FALSE)
train_error= mean(predict(lda_model,newdata = train2)$class!=ytrain)
test_error= mean(predict(lda_model,newdata = test2)$class!=ytest)
print(c(train_error,test_error))
```

```
## [1] 0.310 0.295
```

(c)

```
train_data_logistic = data.frame(train2)
colnames(train_data_logistic)<-c("X1","X2")
train_data_logistic["Y"] = ytrain
test_data_logistic = data.frame(test2)
colnames(test_data_logistic)<-c("X1","X2")
test_data_logistic["Y"] = ytest
logistic_model = glm(Y~.,family=binomial(link="logit"),data = train_data_logistic)
train_error = mean(as.numeric(logistic_model$fitted.values>=0.5)!=ytrain)
test_error = mean(as.numeric(predict(logistic_model,newdata = test_data_logistic[c(1,2)])>=0)!=ytest)
print(c(train_error,test_error))
```

```
## [1] 0.305 0.292
```

(d)

```
errors = data.frame(c(0.310,0.310,0.305),c(0.295,0.295,0.377))
colnames(errors) <- c("Train","Test")
row.names(errors) <- c("Linear","LDA","Logistic")
print(errors)
```

```
##          Train  Test
## Linear   0.310 0.295
## LDA      0.310 0.295
## Logistic 0.305 0.377
```

By comparing test errors we can see that LDA performed the best as it should because the multivariate Gaussian assumption was correct.

LDA out-performed logistic regression by almost 27 percent.

Linear regression and LDA have same errors for test and train data.

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.