



Courage  
Inspiration  
Trust  
Youth  
Uniqueness

# SDSC3006 Lab 3-Classification

Langming LIU    [langmiliu2-c@my.cityu.edu.hk](mailto:langmiliu2-c@my.cityu.edu.hk)

School of Data Science  
City University of Hong Kong

# Contents

---

- Logistic regression
- LDA and QDA
- ROC curve
- KNN

# Logistic regression

# Preliminary

---

- Model structure

$$\log \left( \frac{p(X)}{1 - p(X)} \right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p,$$

$$p(X) = \frac{e^{\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p}}{1 + e^{\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p}}.$$

- Parameters estimation method: maximum likelihood

$$\ell(\beta_0, \beta_1) = \prod_{i: y_i=1} p(x_i) \prod_{i': y_{i'}=0} (1 - p(x_{i'})).$$

- Accuracy of estimates: z-statistics (p-value)

# Preliminary

---

- **Smarket** data set in the ISLR2 package.
- Percentage returns for the S&P 500 stock index in 2001~2005.
- Predict the Direction (Up/Down) of the stock market on a day based on the values in the previous days (Lag1,...,Lag5), etc.
- Code:

```
library(ISLR2)
names(Smarket)
dim(Smarket)
attach(Smarket)
```

# Step1-Training model

---

- Code of logistic model

```
logistic.fit=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume, data=Smarket, family=binomial)
summary(logistic.fit)
```

- What is the meaning of **Number of Fisher Scoring iterations**?

We use Fisher's Scoring algorithm(numerical method) here instead of calculating MLE.

# Step2-predictions

---

- Get the prediction of probability:  
logistic.probs=predict(logistic.fit,type="response")  
##print the first ten probabilities  
logistic.probs[1:10]
- Meaning of prob: prob of going up  
##check dummy variable  
contrasts (Direction)
- Convert the prob to class:  
logistic.pred=rep("Down",1250) #create all "down" array  
logistic.pred[logistic.probs>0.5]="Up" #set threshold 0.5

# Step3-Accuracy

- Confusion matrix  
`table(logistic.pred, Direction)`
- calculate prediction accuracy: 0.5216  
 `#(507+145)/1250` or  
 `mean(logistic.pred==Direction)`
- What this accuracy means?  
It appears that the logistic regression model is working a little better than random guessing. However, this result is misleading because we trained and tested the model on the same set. In other words,  **$1 - 52.2\% = 47.8\%$  is the training error rate.**  
**Training error rate often underestimate the test error rate!**



# Cross validation

---

- Split the data set into a training set and a test set:  
For example, use data in 2001~2004 for training, and data in 2005 for test.
- Fit a logistic regression model using the training set
- Find the test error rate using the test set

# Cross validation

##Step 1: Split data (2001~2004 for training, 2005 for test)

```
train=(Year<2005)
```

```
Smarket.2005=Smarket[!train,]
```

```
Direction.2005=Direction[!train]
```

##Step 2: Train model on training data

```
logistic.fit=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume,  
data=Smarket,family=binomial,subset=train)
```

##Step 3: Make Prediction on test data

```
logistic.probs=predict(logistic.fit,Smarket.2005,type="response")
```

```
logistic.pred=rep("Down", 252) #length(Direction.2005)
```

```
logistic.pred[logistic.probs>0.5] = "Up"
```

##Step 4: Assess prediction accuracy

```
table(logistic.pred,Direction.2005)
```

```
mean(logistic.pred==Direction.2005)
```

```
[1] 0.4801587
```

# Summary of Results

---

- Test error rate =  $1 - 48\% = 52\%$  (Training error rate = 47.8%). Prediction of the logistic regression is even worse than random guessing!
- It is not surprising, since the stock market is too random to predict.
- **Maybe** removing some predictors which have high p-values can improve the prediction performance.

# LDA and QDA

# Keys

---

- Smarket data set in the ISLR2 package.
- Predict the Direction (Up/Down) of the stock market.
- Split the data set into training data and test data.
- Apply function `lda()` and `qda()` in the MASS library
- Next are the steps of LDA, You can change the words `lda` to `qda` to implement QDA

# Steps of LDA

---

- Step 1 - Obtain dataset and Split it

```
library(ISLR2)
library(MASS)
attach(Smarket)
train=(Year<2005)
Smarket.2005 = Smarket[!train,]
Direction.2005 = Direction[!train]
```

- Step 2 - Train model and predict

```
lda.fit=lda(Direction~Lag1+Lag2, data=Smarket,
subset=train)
lda.pred = predict(lda.fit,Smarket.2005)
names(lda.pred)    #see what prediction contains
##lda.pred$class
##lda.pred$posterior
```

# Step

---

- Step 3 - Calculate prediction accuracy

```
lda.class = lda.pred$class  
table(lda.class, Direction.2005)  
mean(lda.class == Direction.2005)
```

- Step 4 - Change threshold (Extra)

```
lda.class = rep("Down", length(Direction.2005))  
lda.class[lda.pred$posterior[,2] > 0.49] = "Up"  
table(lda.class, Direction.2005)  
mean(lda.class == Direction.2005)
```

ROC curve



# ROC curve

---

- Smarket data set in the ISLR2 package.
- Why we draw ROC curve: to compare the performance between methods.
- Compare two methods: logistic regression, LDA
- Method: Write a function `roc.curve()` which calculate and print the ROC curve for a given method.

# ROC curve of Logistic Regression

---

```
library(ISLR2)
attach(Smarket)
##fit logistic regression to all data (2001~2005)
LR.fit = glm(Direction~Lag1+Lag2+Lag3,family=binomial,data=Smarket)
##predict probability of "UP"
LR.pred = predict(LR.fit,type="response")
```

# ROC curve of Logistic Regression

```
## Calculate FPR and TPR under a given threshold
```

```
roc.curve=function(s,print=FALSE){  
  Ps=(LR.pred>s)*1  
  FP=sum((Ps==1)*(Direction=="Down"))/sum(Direction=="Down")  
  TP=sum((Ps==1)*(Direction=="Up"))/sum(Direction=="Up")  
  if(print==TRUE){  
    print(table(Observed=Direction,Predicted=Ps))  
  }  
  vect=c(FP,TP)  
  names(vect)=c("FPR","TPR")  
  return(vect) }
```

```
threshold=0.5
```

```
roc.curve(threshold,print=TRUE)
```

```
## Plot ROC curve
```

```
ROC.curve=Vectorize(roc.curve)  
M.ROC=ROC.curve(seq(0,1,by=0.01))  
plot(M.ROC[1,],M.ROC[2,],col="grey",lwd=2,type="l",xlab="False  
positive rate",ylab="True positive rate")
```

# ROC curve of LDA

---

```
library(ISLR2)
attach(Smarket)
## fit model to all data
library(MASS)
LDA.fit = lda(Direction~Lag1+Lag2+Lag3,data=Smarket)
## predict probabilities of training data
LDA.pred0 = predict(LDA.fit,type="response")
LDA.pred = LDA.pred0$posterior[,2]
```

# ROC curve of LDA

## ## Calculate FPR and TPR under a given threshold

```
roc.curve=function(s,print=FALSE){  
  Ps=(LDA.pred>s)*1  
  FP=sum((Ps==1)*(Direction=="Down"))/sum(Direction=="Down")  
  TP=sum((Ps==1)*(Direction=="Up"))/sum(Direction=="Up")  
  if(print==TRUE){  
    print(table(Observed=Direction,Predicted=Ps))  
  }  
  vect=c(FP,TP)  
  names(vect)=c("FPR","TPR")  
  return(vect) }  
threshold=0.5  
roc.curve(threshold,print=TRUE)
```

## ## Plot ROC Curve

```
ROC.curve=Vectorize(roc.curve)  
M.ROC=ROC.curve(seq(0,1,by=0.01))  
plot(M.ROC[1,],M.ROC[2,],col="blue",lwd=2,type="l",xlab="False  
positive rate",ylab="True positive rate")
```

KNN

# Notice in KNN

---

- Use the `knn()` function in the `class` library, which requires 4 inputs (training observation, test observation, training response, number of K).
- This function does not follow the two-step (first model fitting, then prediction) approach; it generates predictions using a single command.

# Code of KNN

```
library(ISLR2)
attach(Smarket)
library(class)
train=(Year<2005)
train.X=cbind(Lag1,Lag2)[train,] #training data of observation
test.X=cbind(Lag1,Lag2)[!train,] #test data of observation
train.Direction=Direction[train] #training data of response
knn.pred1=knn(train.X,test.X,train.Direction,k=1)
table(knn.pred1,Direction.2005)
mean(knn.pred1==Direction.2005)      #predict accuracy
##change k to get different results
knn.pred2=knn(train.X,test.X,train.Direction,k=3)
table(knn.pred2,Direction.2005)
mean(knn.pred2==Direction.2005)
```