Classification Problems
From 'An Introduction to Statistical Learning'

July 1, 2016

1 Logistic Model

Logistic regression involves directly modeling $P(Y = k|X = x)$ using the logistic function. In other words, the conditional probability of the response Y given X.

Why use a logistic function?

If we have a linear regression model that represents probabilities, such as:

$$P(X) = \beta_0 + \beta_1 X$$

we may have a function where $P(X) < 0$, or $P(x) > 1$. This is not possible, of course. But we may use the logistic equation to produce a function $P(X)$ that follows all of the axioms of probability:

$$0 \leq P(X) \leq 1.$$

Here is the logistic function:

$$P(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

Which may be manipulated into *odds*:

$$\frac{P(X)}{1 - P(X)} = (P(X))\,(1 - P(X))^{-1}$$
$$\frac{P(X)}{1 - P(X)} = e^{\beta_0+\beta_1 X}$$
$$\Longleftrightarrow$$
$$\ln\left(\frac{P(X)}{1 - P(X)}\right) = \beta_0 + \beta_1 X$$

where

$$\frac{P(X)}{1 - P(X)} \in (0, \infty).$$

The expression $\ln\left(\frac{P(X)}{1-P(X)}\right)$ is called the *log-odds* or *logit*.
This function in linear in X, but is not a line. We somehow need to estimate β_0 and β_1 based on training data. The most widely used technique is called *maximum likelihood*.

1.1 Estimating β_0 and β_1

We could find the coefficients β_0 and β_1 via nonlinear least squares. However a more general method called *maximum likelihood* is preferred. Essentially we seek coefficients such that the predicted probability $\hat{p}(x_i)$ corresponds closely to the training set for each x_i. We seek to maximize the *likelihood function*:

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

This can easily be done in R (see "mle" in stats4 package) or Python (see "mle" package).

1.2 Multiple Logistic Regression

We can generalize the *logit* expression (above) to predict a binary response using multiple predictors.

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

where

$$X = (X_1, X_2, ..., X_p)$$

are p predictors. Thus

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

Again we estimate $\beta_0, \beta_1, ..., \beta_p$ using a *maximum likelihood* function. Functions for multiple variables are messy and won't be repeated here.

2 Linear Discriminant Analysis

Consider a less direct alternative to logistic regression where we model the distribution of the predictors X separately in each of the response classes Y then use Bayes' theorem to change these models into estimates for $P(Y = k|X = x)$. When these distributions are assumed to be normal, it turns out that the model is very similar in form to logistic regression.

Why use a different method?

- When classes are well-separated, the parameter estimates for logistic regression are unstable.
- If n is small and the distribution of the predictors X is approximately normal in each of the classes, linear discriminant analysis is more stable than logistic regression.
- Linear discriminant analysis is useful when we have more than 2 response classes.

2.1 Bayes' Theorem for classification

Suppose we wish to classify an observation into one of K classes, where $K \geq 2$. Thus the qualitative response variable Y can take on K distinct values. Let

π_k = the overall (prior) probability that a randomly chosen observation comes from the k^{th} class

and let

$f_k(X) \equiv P(X = x|Y = k)$
= the density function of X for an observation that comes from the k^{th} class

So $f_k(x)$ is relatively large if there is a high probability that an observation in the k^{th} class has $X \approx x$, and $f_k(x)$ is relatively small if it is very unlikely that an observation in the k^{th} class has $X \approx x$.

Bayes' Theorem:

$$\begin{aligned} P(Y = k, X = x) &= P_k(X) \\ &= \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)} \end{aligned}$$

then instead of directly computing $p_k(X)$, we can simply plug in estimates for π_k and $f_k(x)$.

- Predicting π_k is easy. We may just calculate how many of the observations from the training data belong to k.
- Predicting f_k (called the posterior probability) is harder. We may have to assume some simple forms for these probability densities.

Doing so allows us to approximate the Bayes classifier.

2.2 Linear Discriminant Analysis for $p = 1$

If we have only one predictor ($p = 1$), we may obtain an estimate for $f_k(x)$ that we can use to apply Bayes' Theorem. We have to make some assumptions about its form.

Suppose we assume $f_k(x)$ is *normal* or *Gaussian*. In one dimension the normal density has the form

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left[-\frac{1}{2\sigma_k^2}(x-\mu_k)^2\right] \tag{1}$$

where μ_k and σ_k^2 are the mean and variance parameters for the k^{th} class. Further, we will assume that the variance is shared across all k classes. That is, $\sigma_1^2 = ...\sigma_k^2$. Let this be notated as σ^2 for convenience. Substituting equation (1) into the expression for Bayes' Theorem yields:

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2\sigma^2}(x-\mu_k)^2\right]}{\sum_{i \le K} \pi_i \exp\left[-\frac{1}{2\sigma^2}(x-\mu_i)^2\right]} \tag{2}$$

The Bayes classifier involves assigning an observation $X = x$ to the class for which (2) is largest. By doing a little algebra we can see that (2) is equivalent to assigning the observation to the class for which

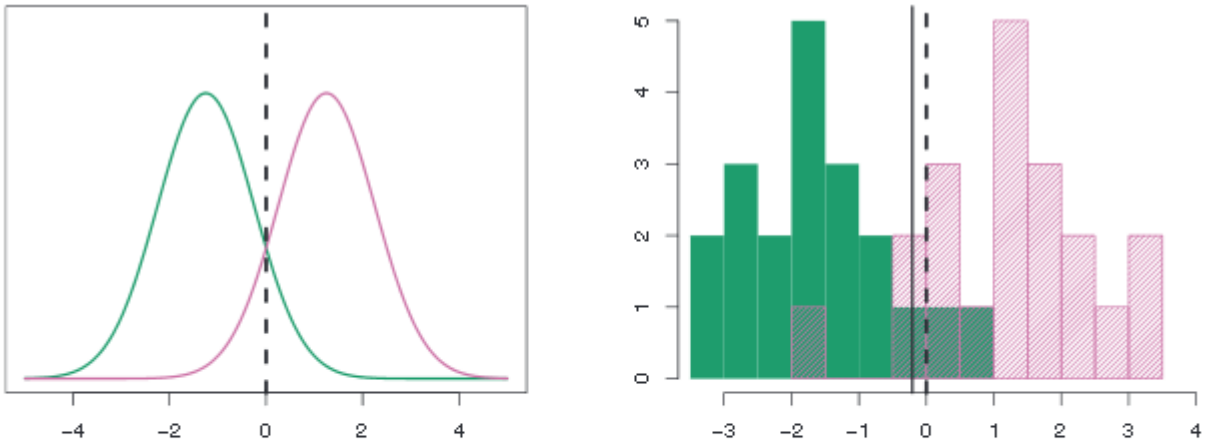
$$\delta_k(x) = x\,\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \ln(\pi_k) \tag{3}$$

is largest. For example, if $K = 2$ and $\pi_1 = \pi_2$, then the Bayes classifier assigns an observation to class 1 if:

$$2x(\mu_1 - \mu_2) > \mu_1^2 - \mu_2^2$$

and to class 2 otherwise.

In the diagram below, if an observation x is just as likely to come from either class, then x belongs to group 1 (left) if $x < 0$ and to group 2 (right) otherwise. In practice we will not often be able to calculate the Bayes classifier.



Left: *Two one-dimensional normal density functions are shown. The dashed vertical line represents the Bayes decision boundary.* Right: *20 observations were drawn from each of the two classes, and are shown as histograms. The Bayes decision boundary is again shown as a dashed vertical line. The solid vertical line represents the LDA decision boundary estimated from the training data.*

In practice, even if we are certain that X is drawn from a Gaussian distribution within each class, we still have to estimate many parameters: $\mu_1, \mu_2, ..., \mu_K$, $\pi_1, \pi_2, ..., \pi_K$ and σ^2. The **Linear Discriminant Analysis (LDA)** method approximates the Bayes classifier by plugging estimates π_i, μ_i and σ^2 into (3).

The following estimates are used:

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i \tag{4}$$

and

$$\hat{\sigma}^2 = \frac{1}{n-K} \sum_{j \leq K} \sum_{i:y_i=k} (x_i - \hat{\mu}_j)^2 \tag{5}$$

where n_k is the number of training observation in the k^{th} class. The value μ_k is just the mean of all of the known observations in the k^{th} class. $\hat{\sigma}^2$ is the weighted average of the sample variances for each of the K classes. Sometimes the class membership probabilities $(\pi_1, \pi_2, ..., \pi_K)$ are known and may be used directly.

LDA estimates π_j using the proportion of the training observations that belong to the j^{th} class.

$$\hat{\pi}_j = n_j / n \tag{6}$$

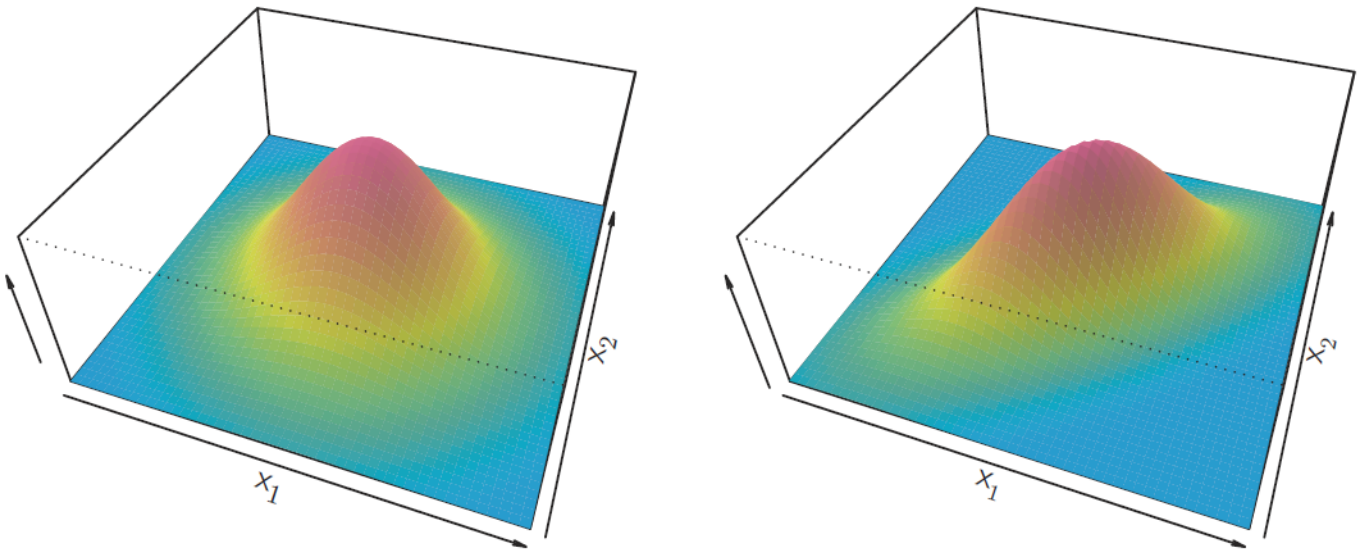
The LDA classifier then plugs (4), (5), and (6) into (3) and assigns an observation $X = x$ to the class for which

$$\hat{\delta}_k(x) = x \frac{\hat{\mu_k}}{\hat{\sigma}^2} - \frac{\hat{\mu_k}^2}{2\hat{\sigma}^2} + \ln(\hat{\pi}_k) \tag{7}$$

is largest. Note: This is called 'Linear Discriminant Analysis' since the 'Linear Discriminant' equation above is linear in x. There are $p(p+1)/2$ total parameters to estimate.

2.3 Linear Discriminant Analysis for $p > 1$

Two examples of multivariate Gaussian distributions with $p = 2$. *Left: x_1 and x_2 are uncorrelated. Right: x_1 and x_2 have correlation of 0.7*



In the case where we have multiple predictors we need to use covariance. The covariance of two variables x and y measures how the two are linearly related. A positive covariance indicates a positive linear relationship between the variables.

- Population Covariance:

$$\sigma_{xy} = \frac{1}{N} \sum_{i \leq N} (x_i - \mu_x)(y_i - \mu_y)$$

- Sample Covariance:

$$s_{xy} = \frac{1}{n-1} \sum_{i \leq n} (x_i - \bar{x})(y_i - \bar{y})$$

- Covariance matrices:

  A covariance matrix for a random vector $\vec{X} = (X_1, X_2, ..., X_p)^T$ is

$$\text{Cov}(X) = \Sigma = \begin{pmatrix} \text{Var}(X_1) & \text{Cov}(X_1, X_2) & \cdots & \text{Cov}(X_1, X_p) \\ \text{Cov}(X_2, X_1) & \text{Var}(X_2) & \cdots & \text{Cov}(X_2, X_p) \\ \vdots & \vdots & \ddots & \vdots \\ \text{Cov}(X_p, X_1) & \text{Cov}(X_p, X_2) & \cdots & \text{Var}(X_p) \end{pmatrix}$$

  Note that $\text{Cov}(X)$ is symmetric positive semi-definite. I.e., $\vec{a}^T \sum \vec{a} \geq 0$ for any $\vec{a} \in \mathbb{R}^n$.

To indicate that a p-dimensional random variable X has a multivariate Gaussian distribution, we write

$$X \sim N(\mu, \Sigma)$$

where the mean μ is the expected value of X: $\mu = E(X)$, and $\Sigma = \text{Cov}(X)$ is the $p \times p$ covariance matrix of X.

The multivariate Gaussian density is defined as

$$f(x) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right]$$

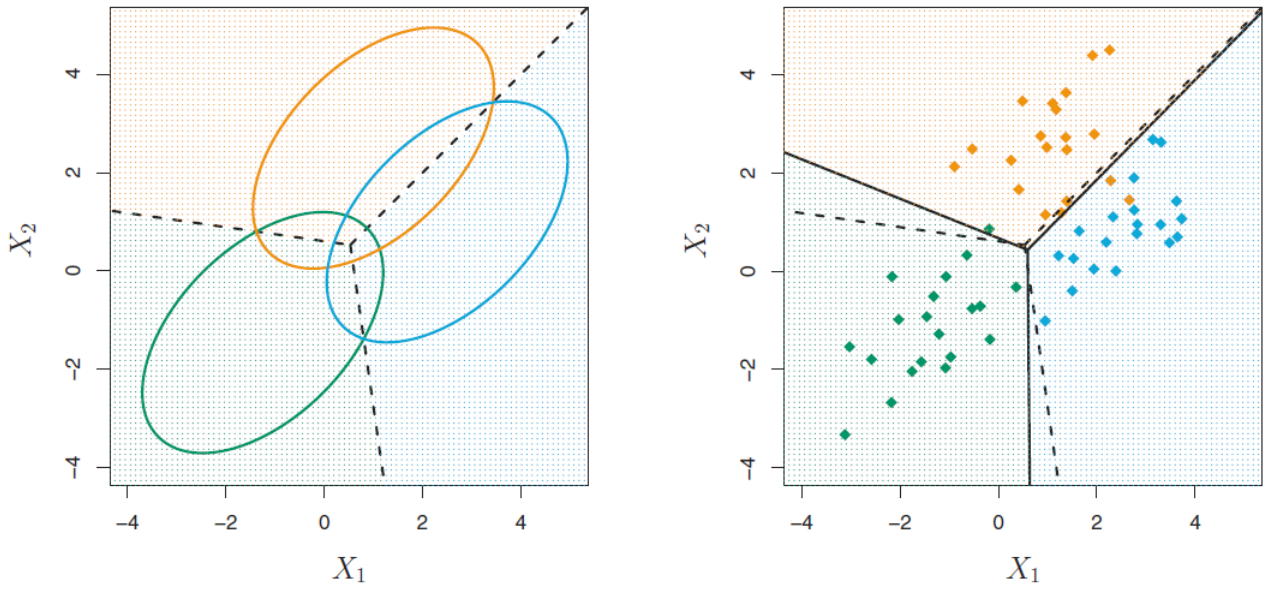
The LDA classifier assumes that the observations in the k^{th} class are drawn from a multivariate Gaussian distribution $N(\mu_k, \Sigma)$, where μ_k is a class-specific vector containing means and Σ is a covariance matrix common to all K classes.

Note that $|\Sigma|$ denotes $\det(\Sigma)$.

LDA plugs the density function for the k^{th} class $f_k(X = x)$ into Bayes' Theorem and then assigns an observation $X = x$ to the class for which

$$\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + \ln(\pi_k) \tag{8}$$

is largest. This is the matrix / vector version of (7).



An example with three classes. The observations from each class are drawn from a multivariate Gaussian distribution with $p = 2$, with a class-specific mean vector and a common covariance matrix. Left: *Ellipses that contain 95 % of the probability for each of the three classes are shown. The dashed lines are the Bayes decision boundaries.* Right: 20 *observations were generated from each class, and the corresponding LDA decision boundaries are indicated using solid black lines. The Bayes decision boundaries are once again shown as dashed lines.*

3 Quadratic Discriminant Analysis

In the case of LDA, each class are drawn from a multivariate Gaussian distribution with a class-specific mean and covariance matrix that is common to all K classes.

Quadratic Discriminant Analysis (QDA) also assumes that each class are drawn from a Gaussian distribution and uses Bayes' theorem to make a prediction. However, QDA assumes that each class has its own covariance matrix.

For QDA:

- Assume that an observation from the k^{th} class is of the form

$$X \sim N(\mu_k, \Sigma_k)$$

  where Σ_k is a covariance matrix for the k^{th} class.

- The Bayes classifier assigns an observation $X = x$ to the class for which

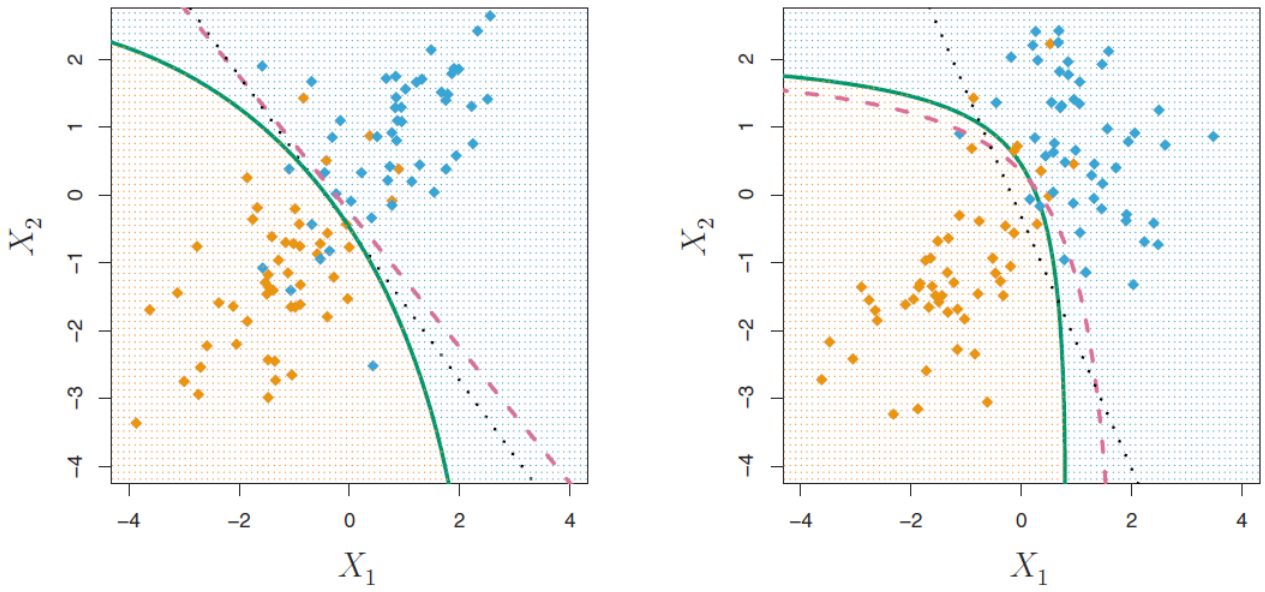
$$\begin{aligned}\delta_k(x) &= -\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) - \frac{1}{2}\ln(|\Sigma_k|) + \ln(\pi_k) \\ &= -\frac{1}{2}x^T\Sigma_k^{-1}x + x^T\Sigma_k^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma_k^{-1}\mu_k - \frac{1}{2}\ln(|\Sigma_k|) + \ln(\pi_k)\end{aligned}$$

  is largest.

- To calculate QDA, we need to plug estimates for Σ_k, μ_k, and π_k into the above. There are $Kp(p+1)/2$ observations to be estimated.

- An observation $X = x$ is assigned to the class for which this quantity is largest.

- Note that the QDA expression is *quadratic* in x, which gives QDA it's name.



Left: *The Bayes (purple dashed), LDA (black dotted), and QDA (green solid) decision boundaries for a two-class problem with* $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2$. *The shading indicates the QDA decision rule. Since the Bayes decision boundary is linear, it is more accurately approximated by LDA than by QDA.* Right: *Details are as given in the left-hand panel, except that* $\boldsymbol{\Sigma}_1 \neq \boldsymbol{\Sigma}_2$. *Since the Bayes decision boundary is non-linear, it is more accurately approximated by QDA than by LDA.*

4 LDA and QDA: Comparison

Is there a particular reason why we should prefer LDA over QDA, or vise-versa? When there are p predictors:

- LDA requires estimating $p(p+1)/2$ parameters
- QDA requires estimating $Kp(p+1)/2$ parameters

Aside from having to estimate a separate covariance matrix for each class, there is also a bias-variance trade-off.

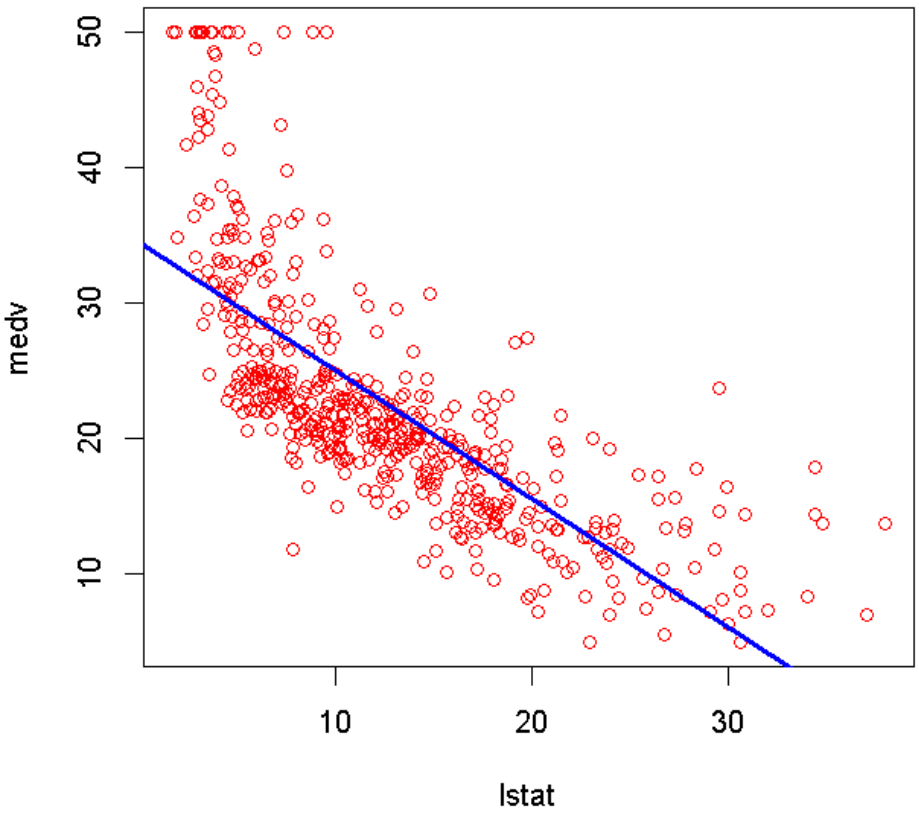
- LDA is much less flexible than QDA
- LDA has substantially less variance
- LDA can suffer a high bias if the K classes are assumed to share a covariance matrix but this assumption is far off.
- LDA is often better if there are few training observations
- QDA is recommended if the training set is very large
- QDA is recommending if the K classes clearly cannot have the same covariance matrix

5 Exercises

∮ 3.6.2 This is just a simple Linear Regression.

```
LinearRegressionLab <- function(){

# Usage:
# source("./LinearRegressionLab.R")
# LinearRegressionLab()

require(MASS)
require(ISLR)
# Causes error for some reason
# Should work now -- had to update R

# Use Boston Dataset
fix(Boston)
# fix invokes edit on x and then assigns the new (edited) version of x in the user's workspace.
names(Boston)
attach(Boston)
# Allows 'Boston' to be indexed -- but this rewrites over other variables.  Extreme care should be used - or u
lm.fit=lm(medv~lstat, data=Boston)
lm.fit
summary(lm.fit)
names(lm.fit)
coef(lm.fit)
# Obtain coefficients for results of lm
confint(lm.fit)
# Obtain cofidence interval

plot(lstat,medv, col="red")
abline(lm.fit, lwd=3, col="blue")
}
```

Here is the graphical output from the linear regression model:



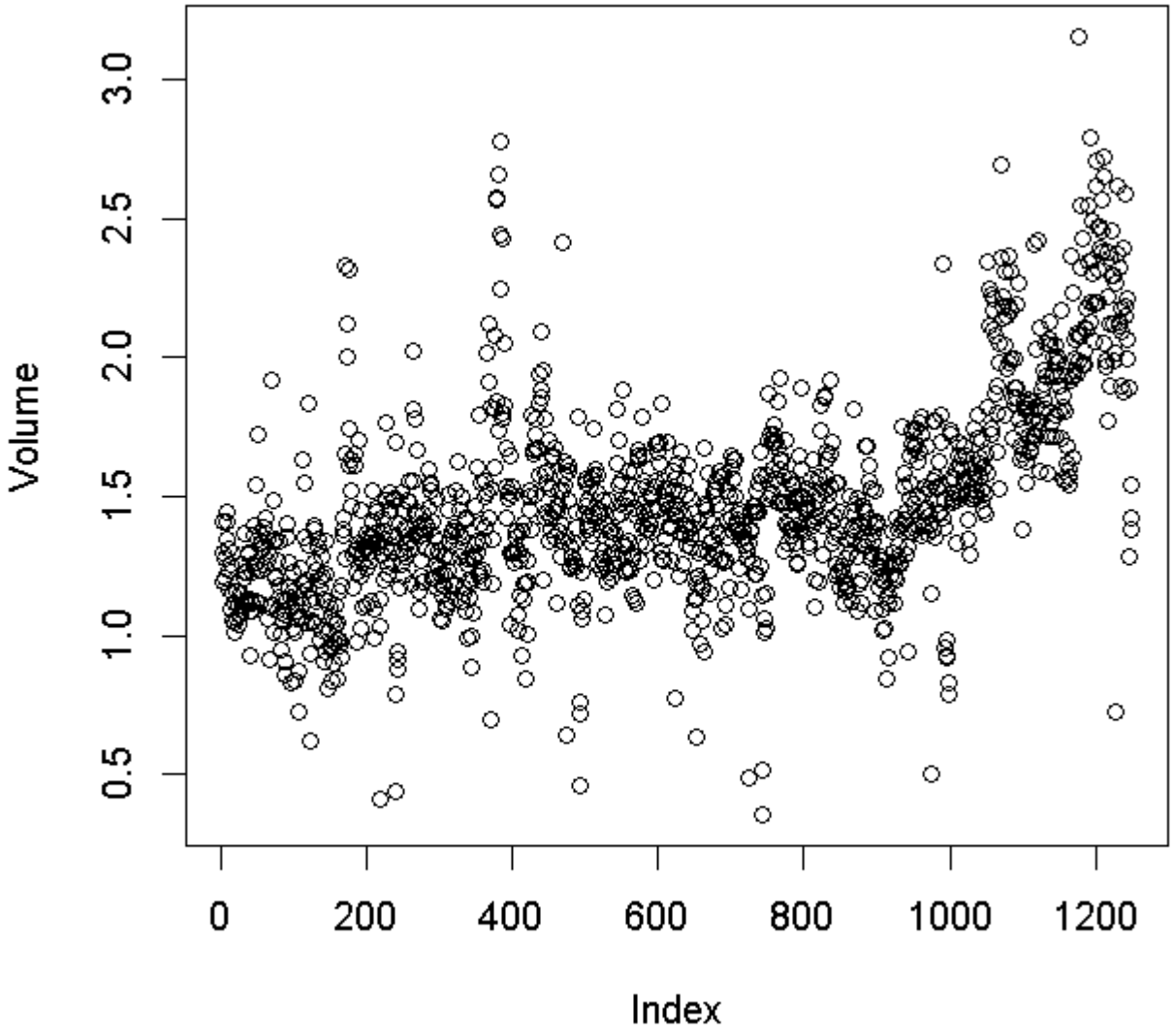
§ 4.6.1

This prints out some summary data for the *Smarket* data (part of the ISLR Library). The data consists of percentage returns for the S&P 500 over 1250 days from the beginning of 2001 to 2005. *Lag1, Lag2, ..., Lag5*: percentage returns for each of the previous five trading days. *Volume*: the total traded on previous day, in billions. *Today*: The percentage return on the date in question. *Direction*: whether the market was UP or DOWN.

R code:

```
Sec.4.6.1 <- function(){
require(ISLR)
print("Names for Smarket:")
print(names(Smarket))
print("Dim(Smarket):")
print(dim(Smarket))
summary(Smarket)
print("Pairs(Smarket):")
print(pairs(Smarket))
print("Correlation matrix for all pairwise correlations between predictors:")
print(cor(Smarket[,-9]))
attach(Smarket)
plot(Volume)
}
```

This plot shows the increase in volume in time.



Here is the function output:

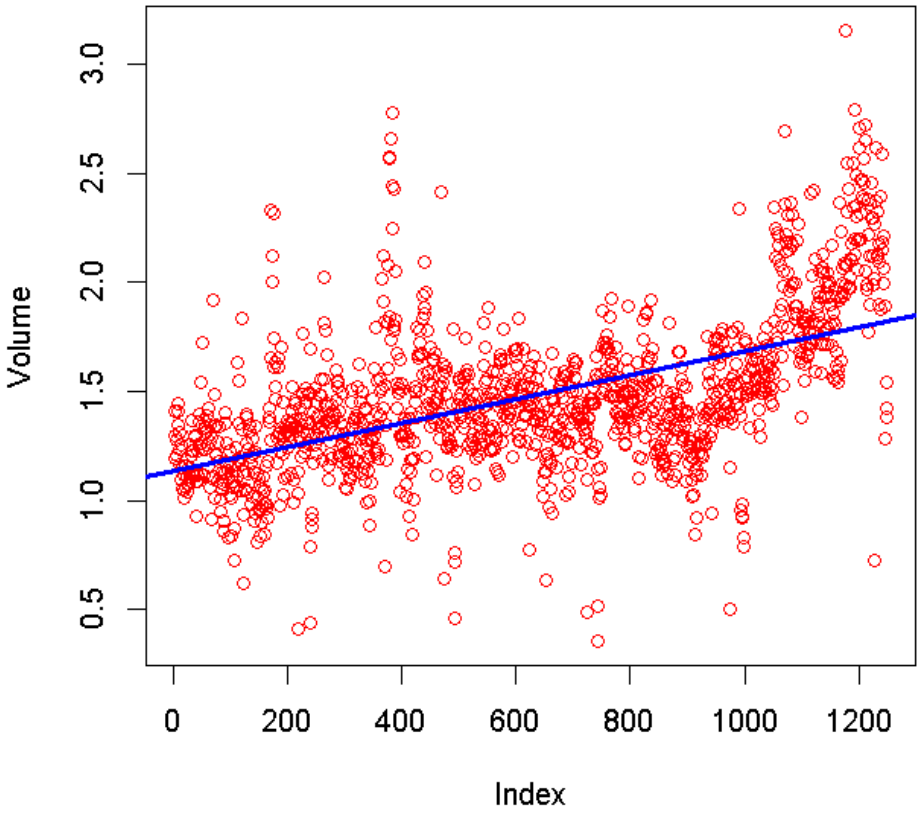
```
> Sec.4.6.1()
[1] "Names for Smarket:"
[1] "Year"      "Lag1"      "Lag2"      "Lag3"      "Lag4"      "Lag5"
[7] "Volume"    "Today"     "Direction"
[1] "Dim(Smarket):"
[1] 1250    9
[1] "Pairs(Smarket):"
NULL
[1] "Correlation matrix for all pairwise correlations between predictors:"
             Year         Lag1         Lag2         Lag3         Lag4
Year   1.00000000  0.029699649  0.030596422  0.033194581  0.035688718
Lag1   0.02969965  1.000000000 -0.026294328 -0.010803402 -0.002985911
Lag2   0.03059642 -0.026294328  1.000000000 -0.025896670 -0.010853533
Lag3   0.03319458 -0.010803402 -0.025896670  1.000000000 -0.024051036
Lag4   0.03568872 -0.002985911 -0.010853533 -0.024051036  1.000000000
Lag5   0.02978799 -0.005674606 -0.003557949 -0.018808338 -0.027083641
Volume 0.53900647  0.040909908 -0.043383215 -0.041823686 -0.048414246
Today  0.03009523 -0.026155045 -0.010250033 -0.002447647 -0.006899527
               Lag5      Volume        Today
Year    0.029787995  0.53900647  0.030095229
Lag1   -0.005674606  0.04090991 -0.026155045
Lag2   -0.003557949 -0.04338321 -0.010250033
Lag3   -0.018808338 -0.04182369 -0.002447647
Lag4   -0.027083641 -0.04841425 -0.006899527
Lag5    1.000000000 -0.02200231 -0.034860083
Volume -0.022002315  1.00000000  0.014591823
Today  -0.034860083  0.01459182  1.000000000
```

We can run a simple linear regression to get a line that best fits the data. An artificial 'x' variable was created for the given output.

R code:

```
LinearRegression4_6_1 <- function(){

# Usage:
# source("./LinearRegression4_6_1.R")
# LinearRegression4_6_1()

require(MASS)
require(ISLR)
fix(Smarket)
plotVec <- 1:length(Volume)
lm.fit=lm(Volume~plotVec)
lm.fit
plot(Volume, col="red")
abline(lm.fit, lwd=3, col="blue")

}
```

which produces the following graphical output:


Volume
0.5
1.0
1.5
2.0
2.5
3.0
0
200
400
600
800
1000
1200
Index

4.6.2 Logistic Regression

This will fit a logistic regression model to predict *Direction* using *Lag1, Lag2, ..., Lag5* and *Volume*. The glm() function is used to fit generalized linear models. The argument "family=binomial" is used to tell R to run logistic regression rather than another general linear model.

R code:

```
LogisticRegression <- function(){
# Note: family=binomial tells R to use a logistic model and not some other general linear model
require(ISLR)
glm.fit = glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume, data = Smarket, family = binomial)
print("Here are the coefficients from the glm.fit command.")
print(coef(glm.fit))
print("Here is the summary of the glm.fit command.  Note that none of the p-values are that strong.")
print(summary(glm.fit))
}
```

The code produces the following output.

```
[1] "Here are the coefficients from the glm.fit command."
 (Intercept)         Lag1         Lag2         Lag3         Lag4         Lag5       Volume
-0.126000257 -0.073073746 -0.042301344  0.011085108  0.009358938  0.010313068  0.135440659
[1] "Here is the summary of the glm.fit command.  Note that none of the p-values are that strong."

Call:
glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
    Volume, family = binomial, data = Smarket)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.446  -1.203   1.065   1.145   1.326

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.126000   0.240736  -0.523    0.601
Lag1        -0.073074   0.050167  -1.457    0.145
Lag2        -0.042301   0.050086  -0.845    0.398
Lag3         0.011085   0.049939   0.222    0.824
Lag4         0.009359   0.049974   0.187    0.851
Lag5         0.010313   0.049511   0.208    0.835
Volume       0.135441   0.158360   0.855    0.392

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1731.2  on 1249  degrees of freedom
Residual deviance: 1727.6  on 1243  degrees of freedom
AIC: 1741.6

Number of Fisher Scoring iterations: 3
```

Unfortunately this isn't a very strong example. It appears that the interaction is weak regardless of which 'Lag' variables are used. Instead I looked up "good logistic regression example R" and found another example online.

∮ Another Logistic Regression Example

R code:

```
# A better logistic regression example found at:
# http://www.ats.ucla.edu/stat/r/dae/logit.htm

LogisticRegression2 <- function(){

require(aod)
library(aod)
require(ggplot2)
library(ggplot2)
require(Rcpp)
library(Rcpp)

# ficticious data from UCLA.  The multiple 'print' commands are needed so that
# R doesn't only print the last command

mydata <- read.csv("http://www.ats.ucla.edu/stat/data/binary.csv")
print("head(mydata)")
print(head(mydata))
print("Summary of 'mydata'")
print(summary(mydata))
print("Standard Deviations for 'mydata'")
print(sapply(mydata, sd))

## two-way contingency table of categorical outcome and predictors
## we want to make sure there are not 0 cells.  This produces a table.
xtabs(~ admit + rank, data = mydata)

# creates logistic model and prints a summary
mydata$rank <- factor(mydata$rank)
mylogit <- glm(admit ~ gre + gpa + rank, data = mydata, family = "binomial")
print(summary(mylogit))
# results:  gre, gpa are statistically significant.  The first three terms for rank are too.
# +1 gre -> +0.002 increase (log odds) of admission
# +1 gpa -> +0.804 increase (log odds) of admission

## CIs using profiled log-likelihood
print(confint(mylogit))

## CIs using standard errors
print(confint.default(mylogit))

## Wald-test
print("Wald-Test will give the overall effect of 'rank'")
print(wald.test(b = coef(mylogit), Sigma = vcov(mylogit), Terms = 4:6))

## Our results are pretty easy to read if we remove the logarithm
print(exp(coef(mylogit)))

## Odds ratios and 95% confidence intervals
exp(cbind(OR = coef(mylogit), confint(mylogit)))
}
```

The code produces the following output:

```
[1] "head(mydata)"
  admit gre  gpa rank
1     0 380 3.61    3
2     1 660 3.67    3
3     1 800 4.00    1
4     1 640 3.19    4
5     0 520 2.93    4
6     1 760 3.00    2
[1] "Summary of 'mydata'"
     admit             gre             gpa             rank
 Min.   :0.0000   Min.   :220.0   Min.   :2.260   Min.   :1.000
 1st Qu.:0.0000   1st Qu.:520.0   1st Qu.:3.130   1st Qu.:2.000
 Median :0.0000   Median :580.0   Median :3.395   Median :2.000
 Mean   :0.3175   Mean   :587.7   Mean   :3.390   Mean   :2.485
 3rd Qu.:1.0000   3rd Qu.:660.0   3rd Qu.:3.670   3rd Qu.:3.000
 Max.   :1.0000   Max.   :800.0   Max.   :4.000   Max.   :4.000
[1] "Standard Deviations for 'mydata'"
      admit         gre         gpa        rank
  0.4660867 115.5165364   0.3805668   0.9444602

Call:
glm(formula = admit ~ gre + gpa + rank, family = "binomial",
    data = mydata)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6268  -0.8662  -0.6388   1.1490   2.0790

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.989979   1.139951  -3.500 0.000465 ***
gre          0.002264   0.001094   2.070 0.038465 *
gpa          0.804038   0.331819   2.423 0.015388 *
rank2       -0.675443   0.316490  -2.134 0.032829 *
rank3       -1.340204   0.345306  -3.881 0.000104 ***
rank4       -1.551464   0.417832  -3.713 0.000205 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 499.98  on 399  degrees of freedom
Residual deviance: 458.52  on 394  degrees of freedom
AIC: 470.52

Number of Fisher Scoring iterations: 4

Waiting for profiling to be done...
                    2.5 %       97.5 %
(Intercept) -6.2716202334 -1.792547080
gre          0.0001375921  0.004435874
gpa          0.1602959439  1.464142727
rank2       -1.3008888002 -0.056745722
rank3       -2.0276713127 -0.670372346
rank4       -2.4000265384 -0.753542605
                    2.5 %       97.5 %
(Intercept) -6.2242418514 -1.755716295
gre          0.0001202298  0.004408622
```

```
gpa          0.1536836760  1.454391423
rank2       -1.2957512650 -0.055134591
rank3       -2.0169920597 -0.663415773
rank4       -2.3703986294 -0.732528724
[1] "Wald-Test will give the overall effect of 'rank'"
Wald test:
----------

Chi-squared test:
X2 = 20.9, df = 3, P(> X2) = 0.00011
(Intercept)         gre         gpa       rank2       rank3       rank4
  0.0185001   1.0022670   2.2345448   0.5089310   0.2617923   0.2119375
Waiting for profiling to be done...
                    OR       2.5 %    97.5 %
(Intercept) 0.0185001 0.001889165 0.1665354
gre         1.0022670 1.000137602 1.0044457
gpa         2.2345448 1.173858216 4.3238349
rank2       0.5089310 0.272289674 0.9448343
rank3       0.2617923 0.131641717 0.5115181
rank4       0.2119375 0.090715546 0.4706961
```

- The most important thing here is this:

```
## (Intercept)         gre         gpa       rank2       rank3       rank4
##      0.0185      1.0023      2.2345      0.5089      0.2618      0.2119
```

- A 1 unit increase in 'gpa' means there will increase the odds of being admitted to graduate school (versus not being admitted) by a factor of 2.23, etc.
- There is much more we can learn about reading the results of a logistic regression. The site has the following graph produced by logistic regression on this make-up data: