FSAN815/ELEG815: Foundations of Statistical Learning

Gonzalo R. Arce

Chapter 13: Classification

Fall 2014
The course provides an introduction to the mathematics of data analysis and a detailed overview of statistical models for inference and prediction.

**Course Structure:**
- Weekly lectures [notes: www.ece.udel.edu/~arce/Courses]
- Homework & computer assignments [30%]
- Midterm & Final examinations [70%]

**Textbooks:**
- Papoulis and Pillai, Probability, random variables, and stochastic processes.
- Hastie, Tibshirani and Friedman, The elements of statistical learning.
- Haykin, Adaptive Filter Theory.
Introduction

Classification

As in the regression setting, in the classification setting we have a set of training observations $(x_1, y_1), \ldots, (x_n, y_n)$ that we can use to build a classifier.

- Illustrate classification using the simulated Default data set.
- Predict whether an individual will default on credit card payment, on the basis of annual income and monthly credit card balance.
FIGURE 4.1. The Default data set. Left: The annual incomes and monthly credit card balances of a number of individuals. The individuals who defaulted on their credit card payments are shown in orange, and those who did not are shown in blue. Center: Boxplots of balance as a function of default status. Right: Boxplots of income as a function of default status.
Predict the medical condition of a patient. For example, there are three possible diagnoses: stroke, drug overdose, and epileptic seizure.

\[
Y = \begin{cases} 
1 & \text{if stroke;} \\
2 & \text{if drug overdose;} \\
3 & \text{if epileptic seizure.}
\end{cases}
\]

This coding implies an ordering on the outcomes.

In general there is no natural way to convert a qualitative response variable with more than two levels into a quantitative response that is ready for linear regression.
Suppose there are only two possibilities for the patient’s medical condition: stroke and drug overdose.

\[
Y = \begin{cases} 
0 & \text{if stroke;} \\
1 & \text{if drug overdose;} 
\end{cases}
\] (2)

- Fit a linear regression to this binary response, and predict drug overdose if \( \hat{Y} > 0.5 \) and stroke otherwise.
- \( X\hat{\beta} \) obtained using linear regression is an estimate of \( \Pr(\text{drug overdose}|X) \) in this special case.
**Binary (two level) Qualitative Response**

**Drawback** If using linear regression, some of estimates might be outside the $[0, 1]$ interval, making them hard to interpret as probabilities.
Consider again the Default data set, the response default falls into one of two categories, Yes or No. Logistic regression models the probability that $Y$ belongs to a particular category. The probability of default given balance can be written as

$$Pr(\text{default}=\text{Yes}|\text{balance}).$$

$p(\text{balance})$ for abbreviation.
The Logistic Model

How should we model the relationship between $p(X) = Pr(Y = 1|X)$ and $X$? We tried linear regression model

$$p(X) = \beta_0 + \beta_1 X$$ (3)

- Probability would fall outside interval $[0,1]$.
- Model $p(X)$ using a function that gives outputs between 0 and 1 for all values of $X$. 
The Logistic Model

In logistic regression, we use the *logistic function*,

\[ p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}} \]  

(4)

To fit the model, we use a method called maximum likelihood.

- For low balances we now predict the probability of default as close to, but never below, zero.
- For high balances we predict a default probability close to, but never above, one.
The Logistic Model

FIGURE 4.2. Classification using the Default data. Left: Estimated probability of default using linear regression. Some estimated probabilities are negative! The orange ticks indicate the 0/1 values coded for default (No or Yes). Right: Predicted probabilities of default using logistic regression. All probabilities lie between 0 and 1.
The Logistic Model

Rewrite equation (4), we have

\[
\frac{p(X)}{1 - p(X)} = e^{\beta_0 + \beta_1 X}
\]  

(5)

The quantity \(p(X)/[1 - p(X)]\) is called the *odds*, and can take on any value between 0 and \(\infty\).

- Odds close to 0 indicates very low probability of default. 1 in 5 people will default, \(p(X) = 0.2\), *odds* = 1/4
- Odds close to \(\infty\) indicates very high probability of default. 9 in 10 people will default, \(p(X) = 0.9\), *odds* = 9
The Logistic Model

Take the logarithm of both sides of equation 2, we have

$$log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X$$

(6)

The left-hand side is called the \textit{log-odds} or \textit{logit}. We see that the logistic regression model has a logit that is linear in $X$.

- Increase $X$ by one unit changes the log odds by $\beta_1$, or equivalently it multiplies the odds by $e^{\beta_1}$. 

**Logistic Regression**

**Estimating the Regression Coefficients**

*Maximum likelihood* is used to estimate $\beta_0$ and $\beta_1$.

Seek estimates for $\beta_0$ and $\beta_1$ such that the predicted probability $\hat{p}(x_i)$ of default for each individual, using equation (4), corresponds as closely as possible to the individual’s observed default status.

This is formalized using a mathematical equation called a *likelihood function*:

$$l(\beta_0, \beta) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1 - p(x_{i'})).$$  \hspace{1cm} (7)

The estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ are chosen to maximize this likelihood function.
TABLE 4.1. For the Default data, estimated coefficients of the logistic regression model that predicts the probability of default using balance. A one-unit increase in balance is associated with an increase in the log odds of default by 0.0055 units.
Logistic Regression

Multiple Logistic Regression

Predict a binary response using multiple predictors.

\[ \log \left( \frac{p(X)}{1 - p(X)} \right) = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p \]  

where \( X = (X_1, \ldots, X_p) \) are \( p \) predictors.

\[ p(X) = \frac{e^{\beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p}}{1 + e^{\beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p}} \]  

Use the maximum likelihood method to estimate \( \beta_0, \ldots, \beta_p \).
Logistic Regression

The logistic regression model arises from the desire to model the posterior probabilities of the $K$ classes via linear functions in $x$, while at the same time ensuring that they sum to one and remain in $[0, 1]$.

\[
\log \frac{Pr(G = 1 | X = x)}{Pr(G = K | X = x)} = \beta_{10} + \beta_1^T x
\]

\[
\log \frac{Pr(G = 2 | X = x)}{Pr(G = K | X = x)} = \beta_{20} + \beta_2^T x
\]

\[
\log \frac{Pr(G = K - 1 | X = x)}{Pr(G = K | X = x)} = \beta_{(K-1)0} + \beta_1^T x
\]
Logistic Regression

Here $\mathbf{\beta}_k = [\beta_{k1}, \beta_{k2}, \ldots, \beta_{kp}]^T$, $\mathbf{x} = [x_1, x_2, \ldots, x_p]^T$.
The model is specified in terms of $K - 1$ log-odds or logit transformations (reflecting the constraint that the probabilities sum to one).
A simple calculation shows that

$$
Pr(G = k | X = \mathbf{x}) = \frac{\exp(\beta_{k0} + \mathbf{\beta}_k^T \mathbf{x})}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \mathbf{\beta}_l^T \mathbf{x})}, \quad k = 1, 2, \ldots, K - 1, \quad (11)
$$

$$
Pr(G = K | X = \mathbf{x}) = \frac{1}{1 + \sum_{l=1}^{K-1} \exp(\beta_{l0} + \mathbf{\beta}_l^T \mathbf{x})}
$$

To emphasize the dependence on the entire parameter set $\theta = \{\beta_{10}, \mathbf{\beta}_1^T, \ldots, \beta_{(K-1)0}, \mathbf{\beta}_{K-1}^T\}$, we denote the probabilities $Pr(G = k | X = \mathbf{x}) = p_k(\mathbf{x}; \theta)$.
Fitting Logistic Regression Models

- Logistic regression models are usually fit by maximum likelihood, using the conditional likelihood of $G$ given $X$.

The log-likelihood for $N$ observations is

$$l(\theta) = \sum_{i=1}^{N} \log p_{g_i}(x_i; \theta)$$  \hspace{1cm} (12)

where $p_{k}(x_i; \theta) = Pr(G = k | X = x_i; \theta)$
Fitting Logistic Regression Models

Discuss in detail the two-class case.

- Code the two-class \( g_i \) via a 0/1 response \( y_i \), where \( y_i = 1 \) when \( g_i = 1 \), and \( y_i = 0 \) when \( g_i = 2 \).

- Let \( p_1(\mathbf{x}; \theta) = p(\mathbf{x}; \theta) \), \( p_2(\mathbf{x}; \theta) = 1 - p(\mathbf{x}; \theta) \)

The log-likelihood can be written as

\[
 l(\theta) = \sum_{i=1}^{N} \left\{ y_i \log p(\mathbf{x}_i; \beta) + (1 - y_i) \log (1 - p(\mathbf{x}_i; \beta)) \right\}
\]

\[
 = \sum_{i=1}^{N} \left\{ y_i (\log p(\mathbf{x}_i; \beta) - \log (1 - p(\mathbf{x}_i; \beta))) + \log (1 - p(\mathbf{x}_i; \beta)) \right\} 
\]

\[
 = \sum_{i=1}^{N} \left\{ y_i \log \frac{Pr(G = 1|X = \mathbf{x}_i)}{Pr(G = 2|X = \mathbf{x}_i)} + \log (Pr(G = 2|X = \mathbf{x}_i)) \right\} 
\]

\[
 = \sum_{i=1}^{N} \left\{ y_i \beta^T \mathbf{x}_i - \log (1 + e^{\beta^T \mathbf{x}_i}) \right\} 
\]

(13)
Here $\beta = \{\beta_{10}, \beta_{1}\}$, and we assume that the vector of inputs $x_i$ includes the constant term 1 to accommodate the intercept.

To maximize the log-likelihood, we set its derivatives to zero. These score equations are

$$\frac{\partial l(\beta)}{\partial \beta} = \sum_{i=1}^{N} x_i (y_i - p(x_i; \beta)) = 0 \quad (14)$$

which are $p+1$ equations nonlinear in $\beta$.

Since the first component of $x_i$ is 1, the first score equation specifies that $\sum_{i=1}^{N} y_i = \sum_{i=1}^{N} p(x_i; \beta)$, the expected number of class ones matches the observed number.
Newton–Raphson algorithm

To solve the score equations, we use the Newton–Raphson algorithm, which requires the second-derivative or Hessian matrix

$$\frac{\partial^2 l(\beta)}{\partial \beta \partial \beta^T} = -\sum_{i=1}^{N} x_i x_i^T p(x_i; \beta)(1 - p(x_i; \beta))$$  \hspace{1cm} (15)

Starting with $\beta^{old}$, a single Newton update is

$$\beta^{new} = \beta^{old} - \left( \frac{\partial^2 l(\beta)}{\partial \beta \partial \beta^T} \right)^{-1} \frac{\partial l(\beta)}{\partial \beta}$$  \hspace{1cm} (16)

where the derivatives are evaluated at $\beta^{old}$. 
Newton Raphson algorithm

- Write the score and Hessian in matrix notation.
- \( y \): \( N \) vector of \( y_i \) values.
- \( x_i \): \( p + 1 \) vector of input.
- \( X \): \( N \times (p + 1) \) matrix of \( x_i \) values.
- \( p \): \( N \) vector of fitted probabilities with \( i \)th element \( p(x_i; \beta^{old}) \).
- \( W \): \( N \times N \) diagonal matrix of weights with \( i \)th diagonal element \( p(x_i; \beta^{old})(1 - p(x_i; \beta^{old})) \).
Then we have

\[
\frac{\partial l(\beta)}{\partial \beta} = X^T(y - p) \tag{17}
\]

\[
\frac{\partial^2 l(\beta)}{\partial \beta \partial \beta^T} = -X^T W X
\]

The Newton step is thus

\[
\beta_{\text{new}} = \beta_{\text{old}} + (X^T W X)^{-1} X^T (y - p)
\]

\[
= (X^T W X)^{-1} X^T W (X \beta_{\text{old}} + W^{-1} (y - p)) \tag{18}
\]

\[
= (X^T W X)^{-1} X^T W z
\]
In the second and third line we have re-expressed the Newton step as a weighted least squares step, with the response

\[ z = X\beta^{old} + W^{-1}(y - p) \]  

(19)

sometimes known as the *adjusted response*.

These equations get solved repeatedly, since at each iteration \( p \) changes, and hence so does \( W \) and \( z \).

This algorithm is referred to as *iteratively reweighted least squares* or IRLS, since each iteration solves the weighted least squares problem:

\[ \beta^{new} \leftarrow \arg \min_{\beta} (z - X\beta)^T W (z - X\beta) \]  

(20)
Linear Discriminant Analysis

Model the distribution of the predictors $X$ separately in each of the response classes (i.e. given $Y$), and then use Bayes’ theorem to flip these around into estimates for $Pr( Y = k | X = x )$.

- When these distributions are normal, the model is similar in form to logistic regression.
Bayes’ Theorem for Classification

- Suppose there are $K$ classes, ($K \geq 2$).
- $\pi_k$: overall or prior probability that a randomly chosen observation comes from the $k$th class.
- $f_k(X) \equiv Pr(X = x | Y = k)$: density function of $X$ for an observation that comes from the $k$th class.

Then Bayes’ theorem states that

$$Pr(Y = k | X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)} \quad (21)$$

$p_k(X) = Pr(Y = k | X)$ for abbreviation.
Linear Discriminant Analysis for $p = 1$

Assume that $p = 1$: there is only one predictor.

1. Obtain an estimate for $f_k(x)$.
2. Plug into equation in order to estimate $p_k(x)$.
3. Classify an observation to the class for which $p_k(x)$ is greatest.

Suppose we assume that $f_k$ is normal or Gaussian.

$$f_k(x) = \frac{1}{\sqrt{2\pi} \sigma_k} \exp\left(-\frac{1}{2\sigma_k^2} (x - \mu_k)^2\right)$$  \hspace{1cm} (22)

where $\mu_k$ and $\sigma_k^2$ are the mean and variance parameters for the $k$th class.
Linear Discriminant Analysis for $p = 1$

Assume $\sigma_1^2 = ... = \sigma_K^2 = \sigma$,

$$ p_k(x) = \frac{\pi_k}{\sum_{l=1}^{K} \pi_l} \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{1}{2\sigma^2} (x - \mu_k)^2\right) $$

$$ \sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{1}{2\sigma^2} (x - \mu_l)^2\right) $$

(23)

The Bayes classifier involves assigning an observation $X = x$ to the class for which equation (12) is largest.

Take log of equation (12),

$$ \log(p_k(x)) = \log(\pi_k) + \log\left(\frac{1}{\sqrt{2\pi\sigma}}\right) - \frac{1}{2\sigma^2} (x - \mu_k)^2 - \log(A) $$

$$ = \log(\pi_k) + \log\left(\frac{1}{\sqrt{2\pi\sigma}}\right) - \frac{x^2}{2\sigma^2} + x \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} - \log(A) $$

(24)

where $A = \sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{1}{2\sigma^2} (x - \mu_l)^2\right)$ is a constant. $\log\left(\frac{1}{\sqrt{2\pi\sigma}}\right)$ and $\frac{x^2}{2\sigma^2}$ is also fixed.
This is equivalent to assigning the observation to the class for which

\[ \delta_k(x) = x \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k) \]  

(25)

is largest.

For instance, \( K = 2 \), \( \pi_1 = \pi_2 \)

Assigns an observation to class 1 if

\[ \delta_1(x) > \delta_2(x) \Rightarrow 2x(\mu_1 - \mu_2) > \mu_1^2 - \mu_2^2. \]

The Bayes decision boundary,

\[ x = \frac{\mu_1^2 - \mu_2^2}{2(\mu_1 - \mu_2)} = \frac{\mu_1 + \mu_2}{2} \]  

(26)
Linear Discriminant Analysis for \( p = 1 \)

**Linear discriminant analysis (LDA):** Approximate the Bayes classifier by plugging estimates for \( \pi_k, \mu_k, \sigma^2 \) into equation (14).

\[
\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i \tag{27}
\]

\[
\hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)^2
\]

In the absence of any additional information, LDA estimates \( \pi_k \) using the proportion of the training observations that belong to the \( k \)th class.

\[
\hat{\pi}_k = \frac{n_k}{n} \tag{28}
\]

The LDA assigns an observation \( X = x \) to the class for which

\[
\hat{\delta}_k(x) = x \frac{\hat{\mu}_k}{\hat{\sigma}^2} - \frac{\hat{\mu}_k^2}{2\hat{\sigma}^2} + \log(\hat{\pi}_k) \tag{29}
\]

is largest.
Linear Discriminant Analysis for $p = 1$

FIGURE 4.4. Left: The two normal density functions that are displayed, $f_1(x)$ and $f_2(x)$. $\mu_1 = -1.25$, $\mu_2 = 1.25$, $\sigma_1^2 = \sigma_2^2 = 1$. 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 solid vertical line represents the LDA decision boundary estimated from the training data.
Extend the LDA classifier to the case of multiple predictors.

Assume that $X = (X_1, X_2, \ldots, X_p)$ is drawn from a multivariate Gaussian distribution, with a class-specific mean vector and a common covariance matrix.

For $p$-dimensional random variable $X$ has a multivariate Gaussian distribution, we write $X \sim N(\mu, \Sigma)$

- $E(X) = \mu$ is the mean of $X$ (a vector with $p$ components).
- $Cov(X) = \Sigma$ is the $p \times p$ covariance matrix of $X$.

The multivariate Gaussian density is

$$f(x) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)\right) \quad (30)$$
In the case of \( p > 1 \) predictors, LDA classifier assumes observations in the \( k \)th class are drawn from a multivariate Gaussian distribution \( \mathcal{N}(\mu_k, \Sigma) \).

- \( \mu_k \) is a class-specific mean vector.
- \( \Sigma \) is a covariance matrix that is common to all \( K \) classes.

\[
\log(p_k(x)) = \log(\pi_k) + \log\left(\frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}}\right) - \frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu) - \log(A)
\]

\[
-\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) = -\frac{1}{2} x^T \Sigma^{-1} x + x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k
\]

Bayes classifier assigns an observation \( X = x \) to the class for which

\[
\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log(\pi_k)
\]

(31)

is largest.
Linear Discriminant Analysis

Linear Discriminant Analysis for $p > 1$

FIGURE 4.6. An example with three classes. The observations from each class are drawn from a multivariate Gaussian distribution with $p = 2$. Left: The dashed lines are the Bayes decision boundaries. Right: 20 observations were generated from each class. Solid black lines indicate corresponding LDA decision boundaries.
Quadratic Discriminant Analysis (QDA): Assume each class has its own covariance matrix. An observation from the \( k \)th class is of the form \( X \sim N(\mu_k, \Sigma_k) \), where \( \Sigma_k \) is a covariance matrix for the \( k \)th class. Bayes classifier assigns an observation \( X = x \) to the class for which

\[
\delta_k(x) = -\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k) + \log(\pi_k)
\]

is largest.

\( (32) \)
FIGURE 4.9. Left: The Bayes (purple dashed), LDA (black dotted), and QDA (green solid) decision boundaries for a two-class problem with $\Sigma_1 = \Sigma_2$. Since the Bayes decision boundary is linear, it is more accurately approximated by LDA than by QDA. Right: $\Sigma_1 \neq \Sigma_2$. Since the Bayes decision boundary is non-linear, it is more accurately approximated by QDA than by LDA.
Use Smarket data for example. This data set consists of percentage returns for the S&P 500 stock index over 1,250 days, from the beginning of 2001 until the end of 2005.

<table>
<thead>
<tr>
<th>Lag1 to Lag5</th>
<th>The percentage returns for each of the five previous trading days</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume</td>
<td>The number of shares traded on the previous day, in billions</td>
</tr>
<tr>
<td>Today</td>
<td>The percentage return on the date in question</td>
</tr>
<tr>
<td>Direction</td>
<td>Whether the market was Up or Down on this date</td>
</tr>
</tbody>
</table>
Logistic Regression

```r
> library (ISLR)
> attach ( Smarket )
glm.fit = glm ( Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data = Smarket, family = binomial )

family = binomial controls R to run a logistic regression.

> coef(glm.fit )

(Intercept)    Lag1    Lag2    Lag3    Lag4
  -0.12600  -0.07307  -0.04230   0.01109  0.00936
   Lag5   Volume
   0.01031   0.13544
```
Logistic Regression

- predict() function is used to predict the probability that the market will go up, given values of the predictors.
- type="response" option controls to output probabilities of the form $P(Y = 1|X)$.

```r
> glm.probs = predict(glm.fit,type =" response ")
> glm.probs [1:5]

   1   2   3   4   5
0.507 0.481 0.481 0.515 0.511
```
Logistic Regression

- The `contrasts()` function indicates that R has created a dummy variable with a 1 for Up.
- Create a vector of class predictions based on whether the predicted probability of a market increase is greater than or less than 0.5.

```r
> contrasts ( Direction )

         Up
Down 0
Up  1

> glm.pred = rep (" Down ", 1250)
> glm.pred[glm.probs > .5] = " Up 
```
Logistic Regression

table() function is used to produce a confusion matrix.

```r
> table(glm.pred, Direction)

Direction

  glm.pred  Down  Up
Down  145  141
Up    457  507

> mean(glm.pred == Direction)

[1] 0.5216
```

Logistic regression correctly predicted the movement of the market 52.2% of the time.
To implement this strategy, create a vector corresponding to the observations from 2001 through 2004. Use this vector to create a held out data set of observations from 2005.

```r
> train = ( Year < 2005 )
> Smarket.2005 = Smarket [ !train , ]
> dim ( Smarket .2005 )
[1] 252 9
> Direction .2005 = Direction [ !train ]
> glm.fit = glm ( Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume , data = Smarket , family = binomial , subset = train )
> glm.probs = predict ( glm.fit , Smarket .2005 , type = " response " )
```
Logistic Regression

```r
> glm.pred = rep(" Down ", 252)
> glm.pred [glm.probs > .5] = " Up"
> table(glm.pred, Direction.2005)

<table>
<thead>
<tr>
<th>glm.pred</th>
<th>Direction.2005</th>
</tr>
</thead>
<tbody>
<tr>
<td>Down</td>
<td>77</td>
</tr>
<tr>
<td>Up</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>44</td>
</tr>
</tbody>
</table>

> mean(glm.pred == Direction.2005)
[1] 0.48

The test error rate is 52%!
```
Logistic Regression

```r
> glm.fit = glm(Direction ~ Lag1 + Lag2, data = Smarket, 
                   family = binomial, subset = train)
> glm.probs = predict(glm.fit, Smarket.2005, type = "response")
> glm.pred = rep("Down", 252)
> glm.pred[glm.probs > .5] = "Up"
> table(glm.pred, Direction.2005)

<table>
<thead>
<tr>
<th>glm.pred</th>
<th>Down</th>
<th>Up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Down</td>
<td>35</td>
<td>35</td>
</tr>
<tr>
<td>Up</td>
<td>76</td>
<td>106</td>
</tr>
</tbody>
</table>

> mean(glm.pred == Direction.2005)
[1] 0.56
```
Linear Discriminant Analysis

- Use `lda()` function, which is part of the MASS library, to achieve LDA.
- `predict()` function obtains prediction results.

```r
> library (MASS)
> lda.fit = lda(Direction ~ Lag1 + Lag2 , data = Smarket, subset = train)
> lda.pred = predict(lda.fit, Smarket.2005)
> lda.class = lda.pred$class
> table(lda.class, Direction.2005)

<table>
<thead>
<tr>
<th>Direction.2005</th>
<th>lda.pred</th>
</tr>
</thead>
<tbody>
<tr>
<td>Down</td>
<td>Down</td>
</tr>
<tr>
<td></td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>35</td>
</tr>
<tr>
<td>Up</td>
<td>Down</td>
</tr>
<tr>
<td></td>
<td>76</td>
</tr>
<tr>
<td></td>
<td>106</td>
</tr>
</tbody>
</table>

> mean(lda.class == Direction.2005)
[1] 0.56
```
Quadratic Discriminant Analysis

- Use qda() function, which is part of the MASS library, to achieve QDA.

```r
> qda.fit = qda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)
> qda.pred = predict(qda.fit, Smarket.2005)
> qda.class = qda.pred$class
> table(qda.class, Direction.2005)

Direction.2005
          qda.pred
   Down   Up
Down     30  20
Up       81 121
```

```r
> mean(qda.class == Direction.2005)
[1] 0.599
```

This level of accuracy is quite impressive for stock market data.
Exercise 4.10

Weekly data set is part of the ISLR package. It contains 1089 weekly percentage returns for 21 years, from the beginning of 1990 to the end of 2010.

<table>
<thead>
<tr>
<th>Lag1 to Lag5</th>
<th>The percentage returns for each of the five previous trading weeks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume</td>
<td>The number of shares traded on the previous week, in billions</td>
</tr>
<tr>
<td>Today</td>
<td>The percentage return on the week in question</td>
</tr>
<tr>
<td>Direction</td>
<td>Whether the market was Up or Down on this week</td>
</tr>
</tbody>
</table>
(b) Use the full data set to perform a logistic regression.

```r
library(ISLR)
attach(Weekly)
glm.fit=glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data=Weekly, family=binomial)
summary(glm.fit)
```

According to p-value, lag1, lag2 and lag4 appear to be statistically significant.
(c) Compute the confusion matrix and overall fraction of correct predictions.

```r
glm.probs = predict(glm.fit, type = "response")
glm.pred = rep("Down", 1089)
glm.pred[glm.probs > .5] = "Up"
table(glm.pred, Direction)
mean(glm.pred == Direction)
```

```
> table(glm.pred, Direction)
   Direction
 glm.pred Down Up
   Down  54  48
   Up    430 557
> mean(glm.pred == Direction)
[1] 0.5610652
```
(d) Fit the logistic regression model using a training data then predict.

\[
\begin{align*}
\text{train} &= (\text{Year}<2009) \\
A &= (\text{Year}==2009) \\
B &= (\text{Year}==2010) \\
\text{Weekly.2009} &= \text{Weekly}[A,] \\
\text{Weekly.2010} &= \text{Weekly}[B,] \\
\text{Direction.2009} &= \text{Direction}[A] \\
\text{Direction.2010} &= \text{Direction}[B] \\
\text{glm.fit} &= \text{glm(}\text{Direction} \sim \text{Lag2, data=Weekly,family=binomial,subset=train)} \\
\text{glm.probs} &= \text{predict(glm.fit,Weekly.2009,type="response")} \\
\text{glm.pred} &= \text{rep("Down",52)} \\
\text{glm.pred}[\text{glm.probs}>.5] &= "Up" \\
\text{table(glm.pred,Direction.2009)} \\
\text{mean(glm.pred==Direction.2009)}
\end{align*}
\]
Exercise

**Figure:** prediction for 2009

```
> table(glm.pred,Direction.2009)
    Direction.2009
  glm.pred    Down  Up
    Down    4   4
    Up      19  25
> mean(glm.pred==Direction.2009)
     [1] 0.5576923
```

**Figure:** prediction for 2010

```
> table(glm.pred,Direction.2010)
     Direction.2010
  glm.pred    Down  Up
    Down    5   1
    Up      15  31
> mean(glm.pred==Direction.2010)
     [1] 0.6923077
```
Exercise

(d) Fit the LDA model using a training data then predict.

library(MASS)
lda.fit=lda(Direction∼Lag2,data=Weekly,subset=train)
lda.pred=predict(lda.fit,Weekly.2009)
lda.class=lda.pred$class
table(lda.class,Direction.2009)
mean(lda.class==Direction.2009)

> table(lda.class,Direction.2009)

<table>
<thead>
<tr>
<th>lda.class</th>
<th>Direction.2009</th>
</tr>
</thead>
<tbody>
<tr>
<td>Down</td>
<td>4</td>
</tr>
<tr>
<td>Up</td>
<td>19</td>
</tr>
</tbody>
</table>

> mean(lda.class==Direction.2009)
[1] 0.5576923

> table(lda.class,Direction.2010)

<table>
<thead>
<tr>
<th>lda.class</th>
<th>Direction.2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>Down</td>
<td>5</td>
</tr>
<tr>
<td>Up</td>
<td>15</td>
</tr>
</tbody>
</table>

> mean(lda.class==Direction.2010)
[1] 0.6923077

Figure: prediction for 2009

Figure: prediction for 2010
(e) Fit the QDA model using a training data then predict.

```r
qda.fit = qda(Direction ~ Lag2, data = Weekly, subset = train)
qda.pred = predict(qda.fit, Weekly.2009)
qda.class = qda.pred$class
table(qda.class, Direction.2009)
mean(qda.class == Direction.2009)
```

Figure: prediction for 2009

Figure: prediction for 2010