# Chapter 4: Classification

Brook Chuang

## Chapter 4: classification

Predicting a qualitative response for an observation can be referred to as *classifying* that observation since it involves assigning the observation to a category or class.

The three most popular classification strategies include: 1. logistic regression 2. linear discriminant analysis 3. k-nearest neighbors

### 4.1 overview of classification

In the classification setting we have a set of training observations $(x_1, y_1), \ldots, (x_n, y_n)$ that are used to build a classifier This classifier must perform well on both training and testing data.

### 4.2 Why not linear regression?

Linear regression is not appropriate because intrinsically has a ordering of variables. For example a predictor of 1 as yes and 0 as no, 2 as maybe, would be very different compared to 0 as yes and 1 as no for linear regression. Each of these coding produce fundamentally different linear models that would lead to a different set of predictions on test observations.

If the response variable's values did take on a natural ordering it is hard to convert these values into a linear regression model.

For *binary* (two level) qualitative responses, the situation is better. Regression by least squares does make sense because we can show that $X\hat{\beta}$ is an estimate of $Pr(drug_{overdoses} \mid X)$ - but if we do use linear regression some of our estimates can be outside the interval [0,1] and therefore may be hard to interpret. - dummy variables can not be easily extended to accommodate qualitative responses with more than two levels

### 4.3 Logisitic regression

Logistic regression models the *probability* that $Y$ belongs to a particular category.

For the default data, the logistic regression models the probability of default such that,

$$Pr(default = Yes \mid balance)$$

This probability, which we will abbreviate to $p(balance)$ is between 0 and 1. Then for any given value of balance, a prediction can be made for default. One might use 0.1, 0.5 ect for determining if the default = yes.

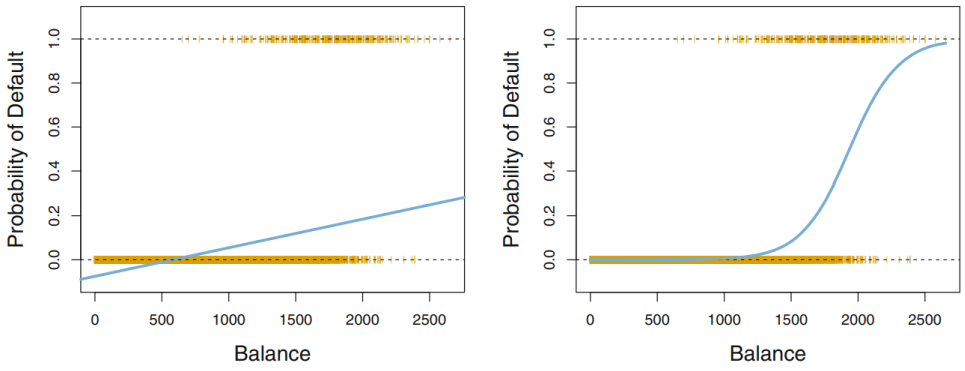



Figure 1: Classificaiton Using Logistic Regression vs linear rgeression

**Logistic model** If we model a linear regression model of $p(X) = Pr(Y = 1 \mid X)$ and $X$ as, $p(X) = \beta_0 + \beta_1 X$.

we have a problem. When balances are near 0, we predict a negative probability of default. Whenever a linear fit is used for a binary response that is coded as 0 or 1, in principle we can always predict $p(X) < 0$ for some values of $X$ and $p(X) > 1$ for others. This can be seen in the visualization below.

We have to model $p(X)$ using th function that gives outputs between 0 and 1 for all values of X. In logistic regression, this equation is the *logistic function*,

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}} \tag{4.2}$$

To fit the model we use the method called *maximum likelihood*. After some manipulation we can find that,

$$\frac{p(X)}{1 - p(X)} = e^{\beta_0+\beta_1 X} \tag{4.3}$$

*Odds*: $p(X)/[1 - p(X)]$ - the odds can take on value between 0 and $\infty$ and indicate very low or very high probabilities of the event occurring.

By taking the log on both sides, we can get,

$$log(\frac{p(X)}{1 - p(X)}) = \beta_0 + \beta_1 X \tag{4.4}$$

where the left hand side is called the *log-odds* or *logit*. In 4.2 we can see that the logistic regression model has a logit that is linear in $X$.

**In the logistic regression model, increasing $X$ by one unit changes the log odds by $\beta_1$.** Equivalent, we can say it multiplies the odds by $e^{\beta_1}$**.** - because the relationship between $p(X)$ and $X$ is not linear, $\beta_1$ does not correspond to the change in $p(X)$ associated with one until increase in $X$.

The amount $p(X)$ changes due to the one unit increase in$X$ will depend on the current value of $X$. If $\beta_1$ is positive, increasing $X$ will correspond to an increasing $p(X)$ and if negative increasing $X$ will be associated with decreasing $p(X)$.

**Estimating regressoin coefficients** Using maximum likelihood is preferred over non linear least squares fit when predicting coefficients.

The basic intuition is as follows: - we seek to predict $\beta_0, \beta 1$, sculpt that the predicted probability $p(\hat{x}_i)$ corresponds as close as possible to the individuals observed status.

The intuition can be formalized using th emaximum likelihood function:

$$l(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1 - p(x_{i'})) \tag{4.5}$$

The estimates of $\hat{\beta}_0$ and $\hat{\beta}_1$ are chosen to maximize this likelihood function.

The z-stat plays the same role as the t-stat in a linear regression output. The z-stat indicates evidence against the null hypothesis $H_0 : \beta_1 = 0$. The null hypothesis implies that $p(X) = \frac{e^{\beta_0}}{1+e^{\beta_0}}$. In other words the at the response does not depend on the observable predictor. The estimated intercept is typically not o fitness,t its main purpose s to adjust the average fitted probabilities to the proportion of ones in the data.

**Making predicitons**

$$p(\hat{X}) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 X}}{1 + e^{\hat{\beta}_0+\hat{\beta}_1 X}}$$

One can use qualitative predictors with the logistic regression model using the dummy variable approach. Using the default data set containing the qualitative variable student, we fit the dummy variable that takes on 1 for students and 0 for non-students. The results are shown below.

| | Coefficient | Std. error | Z-statistic | P-value |
|---|---|---|---|---|
| `Intercept` | −3.5041 | 0.0707 | −49.55 | <0.0001 |
| `student[Yes]` | 0.4049 | 0.1150 | 3.52 | 0.0004 |

$$\hat{Pr}(default = Yes \mid student = yes) = \frac{e^{-}3.5041 + 0.4049 \times 1}{1 + e^{-3.5041+0.4049\times 1}} = 0.0431 \hat{Pr}(default = Yes \mid student = no) = \frac{e^{-}3.5041 + 0.40^{4}}{1 + e^{-3.5041+0.4}}$$

This indicates that students ten to have higher default probabilities than non-students.

**Multiple logistic regression** Predicting a binary response using multiple predictors can be done by extending MLR such that,

$$log(\frac{p(X)}{1 - p(X)}) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p \tag{4.7}$$

where $X = (X_1, \ldots, X_p)$ are p predictors. Then we can rewrite 4.6 as,

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}{1 + e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}} \tag{4.7}$$

Then we use the maximum likelihood method to estimate $\beta_0, \ldots, \beta_p$.

Given multiple logistic regression, there is a surprising result where the p values associated with the dummy variable for student status are very small but the coefficient is negative.

How is it possible for student status to be associated with an *increase* and *decrease* in probability of default given multiple and single logistic regression.

The negative coefficient in the multiple logistic regression indicates that for *a fixed value* of balance and income a student is likely to default than a non-student.

The example shows that two predictors are correlated to each other aka that given a singular predictor there are others that are also relevant. In general this is called *confounding*.

**Log regression for >2 response classes** The two class log regression model have multiple class extensions but they do not tend to be useful. Instead *discriminant analysis* can be used for multiple class classification.

### 4.4 Linear Discrimimant analysis

While log regression involves directly modeling $Pr(Y = k \mid X = x)$ using the log function, we model the conditional distribution of the response $Y$ given the predictors $X$.

In this alternative approach, we model the distribution of the predictors $X$ desperately in each of the response classes given $Y$ and then use the Bayes theorem to flip these around to estimate $Pr(Y = k \mid X = x)$. When these distributions are assumed to be normal. it turns out that the model is very similar in form to logistic regression.

Why do we need another method when we have logistic regression?

- when classes are well-separated the parameter estimated for logistic regression model are surprisingly unstable.
- if n is small and distribution of the predictors $X$ is approx normal in each of the classes linear discriminant model is again more stable than the logistic regression model
- linear discriminant analysis is popular when we have more than two response classes

**using bayes theorem for classification** If we want to classify an observation into one of $K$ classes, where $K \geq 2$ let $\pi_k$ represent the overall or *prior* probability that a randomly chosen observation comes form the kth class. In other words the probability that a given observation is associated with the kth category of the response variable $Y$. Let $f_k = Pr(X = x \mid Y = k)$ denote the *density function* of $X$ for an observation that comes form the kth class. In other words, $f_k(x)$ is relatively large if there is a high probability that an observation in the kth class has $X \approx x$ and $f_k(x)$ is small if it is very unlikely that an observation in the kth class has $X \approx x$. Then the bayes theorem states,

$$Pr(Y = k \mid X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)} \tag{4.10}$$

We use $p_k(X) = Pr(Y = k \mid X)$. This suggests that instead of directly computing $p_k(X)$ we simply plug in estimates of $\pi_k$ and $f_k(X)$ into 4.10

$\pi_k$ calculation is easy given that if we have a random sample of Ys for the population we simply have to compute the fraction of the training observations that belong to the kth class. Estimating $f_k(X)$ tends to be more challenging unless we assume some simple forms for these densities. We refer to $p_k(X)$ as the *posterior* probability that an observation $X = x$ belongs to the kth class, that id the probability that the observation belongs the the kth class given the predictor value for that observation.

If we can find a way to estimate $f_k(X)$ then we can develop a classifier that approximates the bayes classifier.

**Linear discriminat analyss for p =1** Assume that $p = 1$. Then we like to obtain an estimate fo $f_k(X)$ to estimate $p_k(X)$ and then classification the observation to the class which $p_k(X)$ is greatest.

First we make some assumptions of the form:

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} exp(-\frac{1}{2\sigma_k^2}(x - \mu_k)^2) \tag{4.11}$$

We can then plug in 4.11 into 4.10 such that,

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} exp(1/(2\sigma^2)(x - \mu_k)^2)}{\sum_{l=1}^{K} \pi_l 1/(2\pi\sigma) exp(-1/(s\sigma^2)(x - \mu_l)^2)} \tag{4.12}$$

The Bayes classifier involves assigning an observation $X = x$ to the class for which is largest. Taking the log and rearranging the terms, it is not hard to show that this is equivalent to assigning the observation to the class for which 4.13 is largest:

$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + log(\pi_k) \tag{4.13}$$

For instance, if K = 2, and $\pi_1 = \pi_2$ then 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. The bayes decision boundary corresponds to the point where

$$x = \frac{\mu_1^2 - \mu_2^2}{2(\mu_1 - \mu_2)} = \frac{\mu_1 + \mu_2}{2} \tag{4.14}$$

In this example we have all the parameters and distributions know, in actual practice we can not calculate the bayes classifier.

The *linear discriminant analysis* (LDA) method approximates the bayes classifier by plugging the estimates for $\pi_k, \mu_k, \sigma^2$ int0 4.13. The following estimates are used,

$$\hat{\mu_k} = \frac{1}{n} \sum_{i:y_i=k} x_i \hat{\mu^2} = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu_k}^2) \tag{4.15}$$

The estimate of $\mu_k$ is the average of of all training observations from the kth class while $\hat{\sigma^2}$ is the weighted average of the sample variances for each of the K classes. Sometimes we know the class probabilities of $\pi_1, ... \pi_k$ and can be used directly other times we assume $\pi_k$ to be (in LDA ect) the proportion of training observations that belong to the kth class

$$\hat{\pi_k} = n_k \backslash n \tag{4.16}$$

The LDA classifier plugs estimates 4.15, 4.16 into 4.13 to get

$$\hat{\delta_k}(x) = x \cdot \frac{\hat{\mu_k}}{\hat{\sigma^2}} - \frac{\hat{\mu_k^2}}{2\hat{\sigma^2}} + log(\hat{\pi_k}) \tag{4.17}$$

for which 4.17 is largest. "linear" in LDA comes from the discriminant functions $\hat{\delta_k}(x)$ that are linear functions of $x$.

Example 4.4: To impelled LDA we begin by 1. estimating $\pi_k, \mu_k, \sigma^2$ using 4.15 and 4.16. 2. compute the decision boundary that results after assigning an observation to the class for which 4.17 is largest 3. because $n_1 = n)2 = 20$, we have $\hat{\pi_1} = \hat{\pi_2}$ and the boundary corresponds to the midpoint between sample means for the two classes $(\mu_1 + \mu_2)/2 = 0$.

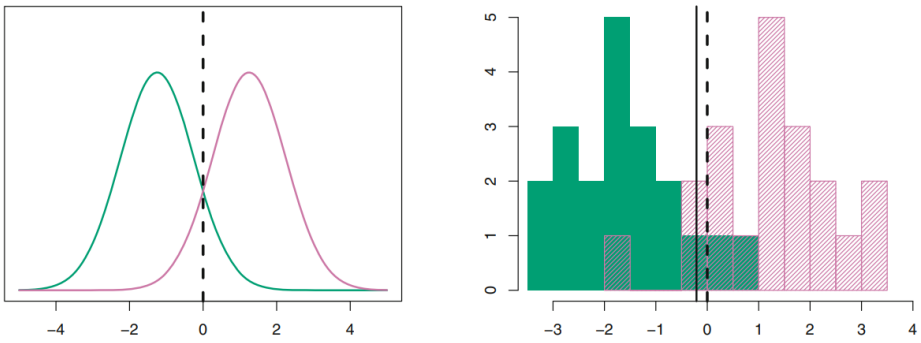


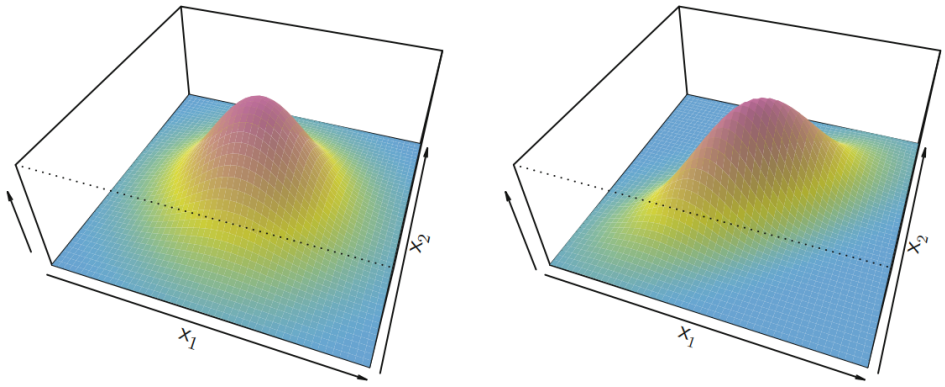

Figure 2: LDA

To reiterate, the LDA classifier results form assuming that the observations within each class come from a normal distribution with a class-specific mean vector and a common variance $\sigma^2$ and plugging estimates for these parameters into the bayes classifier.

**linear discrimination analyss for p > 1** We now extend the LDA classifier to the case of multiple predictors. WE will assume $X = (X_1, dots, X_p)$ is drawn from a *multi-variate Gaussian* distribution with a class specific mean vector and a common covariance matrix.

**Multivariate guassian** The multivariate Gaussian distribution assumes that each indv. predictor follows a one-dimensional normal distribution as in 4.11 with some correlation between each pair of predictors.

In 4.5 the height of the surface at any particular point represents the probability that both $X_1$ and $X_2$ fall into a small region around that point. The cross section of the shape will have a 1d normal distribution.




The left hand side of the image shoes an example wehre the variance of $Var(x_1) = Var(X_2)$ and correlation between the two are . This results in a *bell shape* and will distort with predictors are correlated or have unequal variances (seen on the right).

To indicate p-dim r.v. $X$ has a multivariate Gaussian distribution, we write $X \sim N(\mu, \sum)$. - $\mathbb{E}(X) = \mu$ - $Cov(X) = \sum$ is the $p * p$ covariance matrix of $X$.

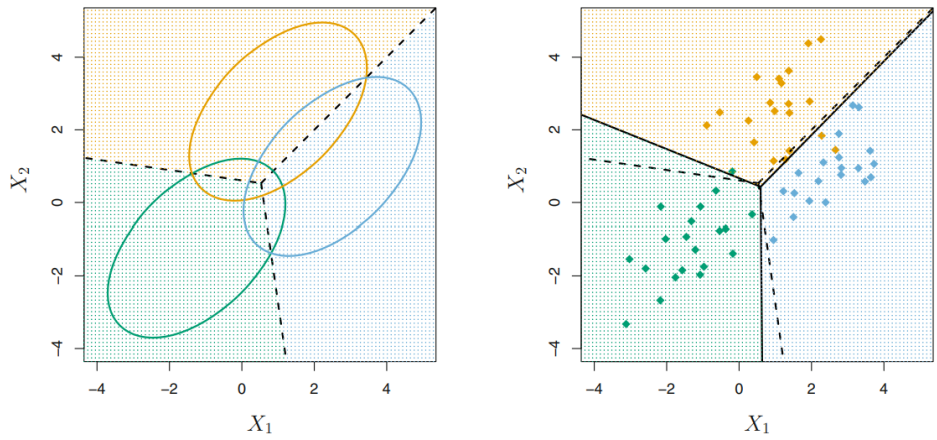
Formally the multivariate Gaussian density is,

$$f(x) = \frac{1}{(2\pi)^{p/2} \mid \sum \mid^{1/2}} exp(-1/2(x-\mu)^T \sum^{-1}(x-\mu)) \tag{4.18}$$

In the case of $p > 1$ the LDA classifier assumes that the observations in the kth class are drawn from a multivariate Gaussian distribution $N(\mu_k, \sum)$ where $\mu_k$ is a class specific mean vector and $|sum$ is a covariance matrix that is common to all$ K$ classes.

Plugging the density function for the $kth$ class $f_k(X = x)$ into 4.10 and performing some algebra we ge that the bayes classifier assigns an observation $X = x$ to the class for which,

$$\delta_k(x) = x^T \sum^{-1} \mu_k - 1/2\mu_k^T \sum^{-1} \mu_k + log\pi_k \tag{4.19}$$



In example 4.6 on the left three equally sized Gaussian classes are show with class-specific mean vectors and a common covariance matrix The three ellipses represent regions that contain 95% of the probability for each of the three classes. The dashed lines are the bayes decision boundaries and they represent the set of values x for which $ _k(x) = _l(x) such that,

$$x^T \sum^{-1} \mu_k - 1/2\mu_k^T \sum^{-1} \mu_k = x^T \sum^{-1} \mu_l - 1/2\mu_l^t \sum^{-1} \mu_l \tag{4.20}$$

In the visualization thee are three lines representing the boundary because there are *three pairs of classes* among the three classes.

We need to estimate the unknown parameters $\mu_1, \dots, \mu_k$ and $\pi_1, \dots, \pi_k$ and $\sum$. To assign a new observation $X = x$ LDA plugs these estimates into 4.19 and classifies to the class ft which $\hat{\delta_k}(x)$ is the largest. - in 4.19 $\delta_k(x)$ is a linear function of $x$. The LDA decision rule depends on x only through a linear combination of its elements.

On the right of th visualization above, the decision boundaries in black are constructed form LDA and are pretty close to the Bayes decision boundary.

when constructing on data, - training error rates will usually be lower than test error rates (which are real quantities of interest) We specifically adjust parameters of our model to do well of training vs test. - the high ratio of $p$ parameters to the sample of $n$ we will expect *over fitting* to play a role - the *trivial null classifier* will achieve an error rate that is only a big higher that the LDA training set error rate

In practice the binary classier makes two different types of errors (type I or type II) in this case construct a *confusion matrix*

| | | *True default status* | | |
|---|---|---|---|---|
| | | No | Yes | Total |
| *Predicted default status* | No | 9,644 | 252 | 9,896 |
| | Yes | 23 | 81 | 104 |
| | Total | 9,667 | 333 | 10,000 |

#### Regression validation

Class-specific performance is also important. *sensitivity* and *specificity* characterize the performance of a classifier.

*Sensitivity* is the percentage of correctly identified events. The specificity is the percentage of correctly identified complements of the event.

Why does LDA have such low sensitivity? LDA is trying to approximate the Bayes classifier which has the lowest total error rate out of all the classifiers (if the Gaussian model is correct). The Bayes classifier will yield the smallest possible total number of misclassified observations irrespective of which class the errors come from.

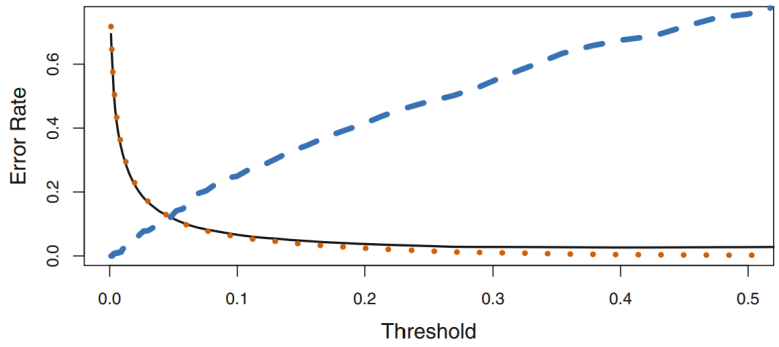
The bayes classifier works by assigning an observation to the class for with the posterior probability $p_k(X)$ is greatest. In the two-class case this amounts to assigning an observation to the *default* class if

$$Pr(defualt = yes \mid X = x) > 0.5 \tag{4.21}$$

The bayes and LDA uses a threshold of 50% for the posterior probability of default in order to assign a observation to the default class.

If we are concerned about incorrectly predicting the default status for individuals who default, then we can consider lowering this threshold. For instance, we might label any customer with a posterior probability of default above 20% to the default class. Instead of assigning an observation to the default class if 4.21 holds we can assign an observation if

$$Pr(default = yes \mid X = x) > 0.2 \tag{4.21}$$



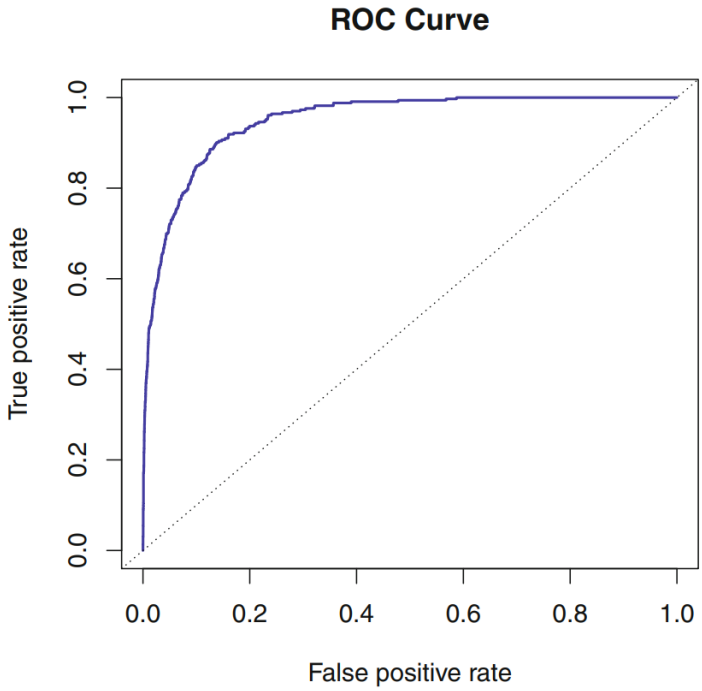

The visualization above illustrates the trade-off that results from modifying the threshold value for the posterior probability of default. Various error rates are shown as a function of the threshold value. WHen the threshold is 0.5 for the bayes classifier this is known t have the lowest overall error rate but the error rate amount individuals who default is quite high (blue line).

As the threshold is reduced, the trade off is seen. We determine the threshold value using *domain knowledge* such as s costs associated with the default.

**Roc curve** The *ROC curve* is a popular graphic for displaying the two types of errors for all possible thresholds. The name ROC is short for *receiver operating characteristics* and the overall performance of a classifier summarized over all possible thresholds is given by the area under the curve (ROC).

AN ideal ROC will hug the top left corner so the *larger the AUC the better.*

We expect a classifier that performs no better than chance to have a AUC of 0.5. ROC curves are useful for comparing different classifiers since they take into account all possible thresholds.




Varying classifier thresholds changes its true positive and dlase postive rate. THese are also called *sensitivity* and *1-specificity* of out classifier.

| Name | Definition | Synonyms |
|---|---|---|
| False Pos. rate | FP/N | Type I error, 1−Specificity |
| True Pos. rate | TP/P | 1−Type II error, power, sensitivity, recall |
| Pos. Pred. value | TP/P* | Precision, 1−false discovery proportion |
| Neg. Pred. value | TN/N* | |

Figure 3: Popular Terms

**Quadratic Discriminant analysis** *Quadratic discriminant analysis* (QDA) provides an alternative approach assuming observations from each class are drawn from a Gaussian distribution and plugging estimates for the parameters into Bye's theorem in order to perform predictions.

Unlike LDA, QDA assumes that each class has its own covariance matrix. That is, it assumes that an observation from the kth class is the form $X \sim N(\mu_k, \sum_k)$ where $\sum_k$ is the covariance matrix for the kth class. Under the assumption, the bayes classifier assigns an observation $X = x$ to the class for which

$$\delta_k(x) = -1/2(x-\mu)k)^T \sum_k^{-1}(x-\mu_k) - 1/2log \mid \sum_k \mid + log\pi_k = 1/2x^T \sum_k^{-1} + x^T \sum_k^{-1} \mu_k - 1/2\mu_k^T \sum_k^{-1} \mu_k - 1/2log \mid \sum_k \mid + log\pi_k \tag{4.23}$$

So the the QDA classifier involves plugging estimates for $\sum_k, \mu_k$ and $\pi_k$ into 4.23 and then assigning the observation $X = x$ to the class for which this quantity is the largest.

Unlike 4.18, the quantity x appears as the *quadratic function in 4.23. This is where QDA gets its name.

Why does it matter whether or not we assume that the $K$ classes share a common covariance matrix? Why would one prefer LDA to QDA or vice versa? - the bias-variance trade off. When there are $p$ predictors then estimating a covariance matrix requires estimating $p(p+1)/2$ parameters. QDA estimates a seperate covariance matrix for each class for a total of $Kp(p+1)/2$ parameters.

Instead assuming that the $K$ classes share a common covariance matrix, the LDA model becomes linear in x which means that there are $Kp$ linear coefficients to estimate. Consequently LDA is much less flexible classifer than QDA and lower variance. - this could possibly lead to better prediction performance but the trade off is that LDA's assumption that the $K$ classes share a common covariance matrix is badly off, LDA can suffer from high bias.

QDA is recommended if the training set is very large so that the variance of the classifier is not a major concern or if the assumption of a common covariance for the K classes is clearly untenable.

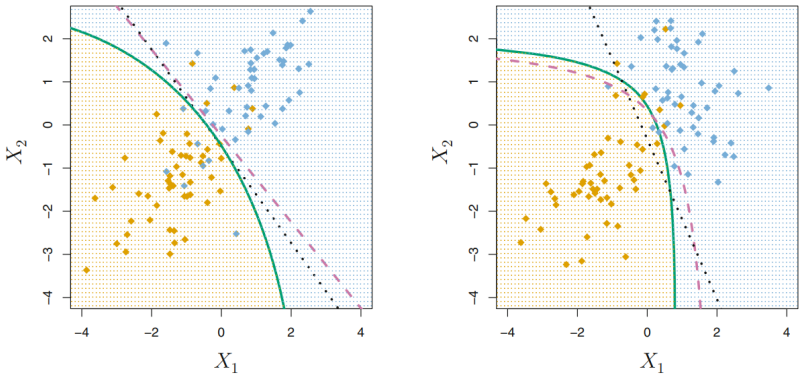



Figure 4: Popular Terms

The difference between LDA (black) and QDA (green) left where $\sum_1 = \sum_2$. Because the bayes decisions

boundary (purple) is linear, LDA is a better approximation.(QDA suffers from higher variance without a corresponding decrease in variance).

On the left where $\sum_1 \neq \sum_2$ we see that the bayes boundary is nonlienar and QDA is a better approximation.

### 4.5 comparison of classification methods

So far we have anamlyzed logistic regression, LDA, and QDA. Logistic regressiona nd LDA mehtods are closely connected.

**Log reg vs LDA** Consider the two-class setting with $p = 1$ predictor and let $p_1(x)$ and $p_2(x) = 1 - p_1(x)$ be the probabilities that the observation $X = x$ belongs to class 1 and class 2. In the LDA framework in 4.12 and 4.13 the log odds can be given as

$$log(\frac{p_1(x)}{1 - p_1(x)}) = log(\frac{p_1(x)}{p_2(x)}) = c_0 + c_1 x \tag{4.24}$$

where $c_0$ and $c_1$ are functions of $\mu_1, \mu_2, \sigma^2$. Then we know that the logistic regression is

$$log(\frac{p_1}{1 - p_1}) = \beta_0 + \beta_1 x \tag{4.25}$$

Both 4.24 and 4.25 are linear functions of x. Hence both logistic regressions and LDA produce linear decision boundaries and the only difference is that $\beta_0, \beta_1$ are estimated by MLE in logistic whereas $c_0, c_1$ are computed using the estimated mean and variance from the normal distribution This same connection between DLA and logistic regression holds for multdim data where $p > 1$.

While LDA and logistic regression only differ in their fitting procedures, their results are not always the same. If the assumptions of LDA are not met, ie same covariance matrix x and observations follow a Gaussian distribution, logistic regression can outperform LDA.

**KNN** In KNN, the K training observations that are closest to x are identified. Then $X$ is assigned to the class to which the plurality of these observations belong. KNN is a competency non-parametric approach: no assumptions are made about the shape of the decision boundary. - we expect this approach to dominate LDA and log regression but can not tell us which predictors are important.

**QDA** QDA serves as a compromise between the non-parametric KNN method and the linear LDA and logistic regression approaches. Since QDA assumes a quadratic decision boundary, it can accurately model a wider range of problems. While not as flexible as KNN, QDA can perform better in the presence of limited number of training observations because it does make some assumptions of the decisions boundry.

**Scenario testing**

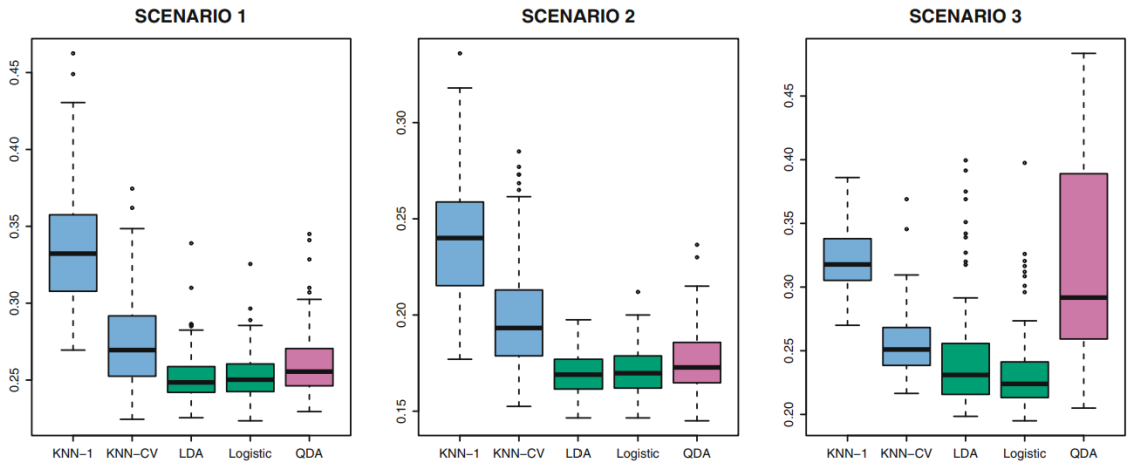



**FIGURE 4.10.** *Boxplots of the test error rates for each of the linear scenarios described in the main text.*

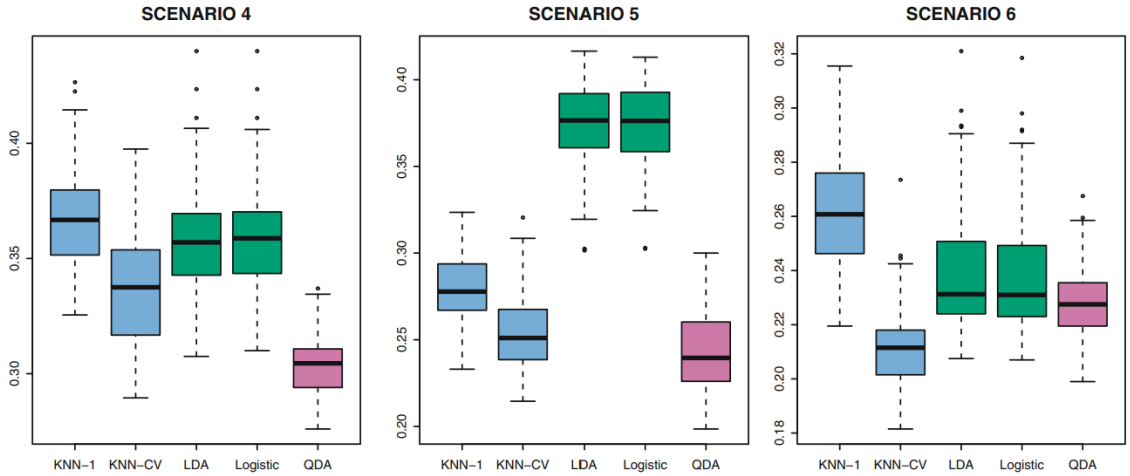



**FIGURE 4.11.** *Boxplots of the test error rates for each of the non-linear scenarios described in the main text.*

1. 20 training observations in each of two classes. Uncorrelated, random normal variables with a different mean in each. LDA performed well, KNN performed badly (paid price in variance). QDA performed worse than LDA since its fit was more flexible.

2. same as scenario 1 but now the two predictors have a correlation of -0.5.

3. $X_1$ and $X_2$ from the t-distribution with 50 observed per class. The t-distribution has a similar shape to the normal distribution but it has a tendency to yield more extreme point. The boundary is still linear.

4. Data generated form a normal distribution with a correlation of 0.5 between the predictors in the first class and correlation of -0.5 between the predictors in the second class. QDA assumption and resulted in quadratic bounds.

5. Observations generated form normal distribution with uncorrelated predictors. Responses samples from logistic function - quadratic boundary.

6. Responses samples form non-linear function.

### 4.6 Lab: Logistic regression, LDA, QDA, KNN

```
library("ISLR")
```

```
## Warning: package 'ISLR' was built under R version 4.2.3
```

```
names(Smarket)
```

```
## [1] "Year"      "Lag1"      "Lag2"      "Lag3"      "Lag4"      "Lag5"
## [7] "Volume"    "Today"     "Direction"
```

```
summary(Smarket)
```

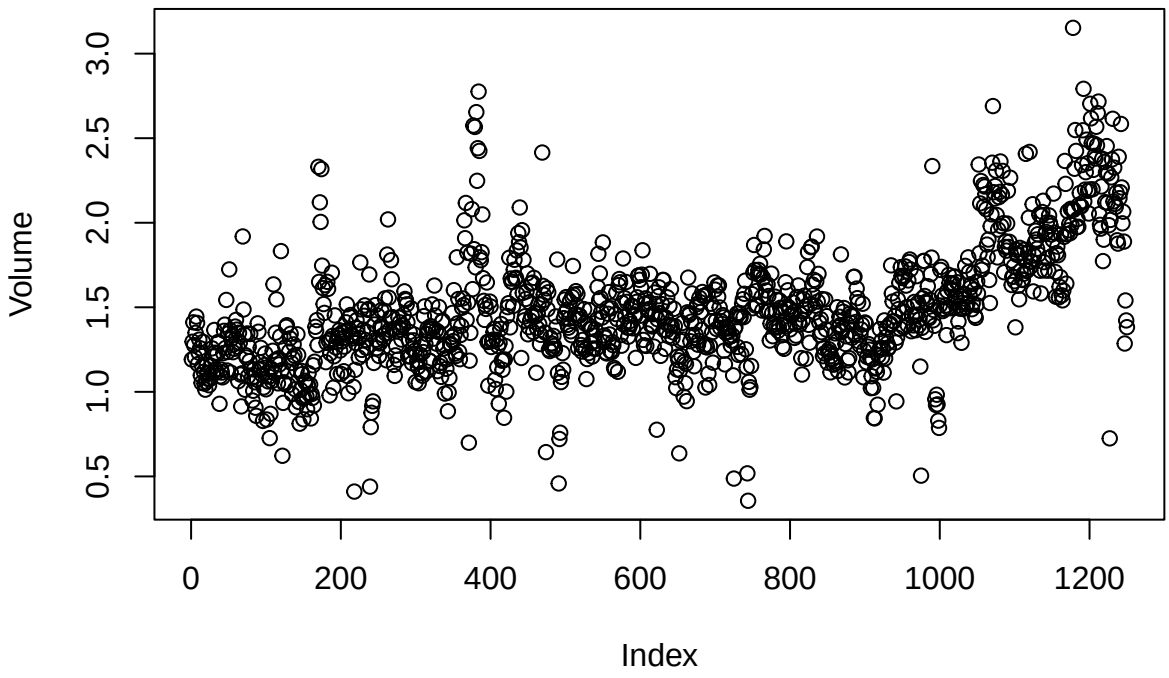
```
##       Year           Lag1                Lag2                Lag3
##  Min.   :2001   Min.   :-4.922000   Min.   :-4.922000   Min.   :-4.922000
##  1st Qu.:2002   1st Qu.:-0.639500   1st Qu.:-0.639500   1st Qu.:-0.640000
##  Median :2003   Median : 0.039000   Median : 0.039000   Median : 0.038500
##  Mean   :2003   Mean   : 0.003834   Mean   : 0.003919   Mean   : 0.001716
##  3rd Qu.:2004   3rd Qu.: 0.596750   3rd Qu.: 0.596750   3rd Qu.: 0.596750
##  Max.   :2005   Max.   : 5.733000   Max.   : 5.733000   Max.   : 5.733000
##       Lag4                Lag5              Volume           Today
##  Min.   :-4.922000   Min.   :-4.92200   Min.   :0.3561   Min.   :-4.922000
##  1st Qu.:-0.640000   1st Qu.:-0.64000   1st Qu.:1.2574   1st Qu.:-0.639500
##  Median : 0.038500   Median : 0.03850   Median :1.4229   Median : 0.038500
##  Mean   : 0.001636   Mean   : 0.00561   Mean   :1.4783   Mean   : 0.003138
##  3rd Qu.: 0.596750   3rd Qu.: 0.59700   3rd Qu.:1.6417   3rd Qu.: 0.596750
##  Max.   : 5.733000   Max.   : 5.73300   Max.   :3.1525   Max.   : 5.733000
##  Direction
##  Down:602
##  Up  :648
##
##
##
##
```

```
# pairwise correlations
cor(Smarket[,-9])
```

```
##              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
```

```
attach(Smarket)
plot(Volume)
```




**Logistic regression** Predict direction given predictors.

glm(): generalized linear models

```
# fit log reg
glm.fit <- glm(Direction~Lag1 + Lag2 + Lag3 + Lag4 + Lag5
+ Volume, data = Smarket, family = binomial)
summary(glm.fit)
```

```
##
```

```
## 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
```

```
# access coef
coef(glm.fit)
```

```
##  (Intercept)         Lag1         Lag2         Lag3         Lag4         Lag5
## -0.126000257 -0.073073746 -0.042301344  0.011085108  0.009358938  0.010313068
##       Volume
##  0.135440659
```

```
# access p val of coef
summary(glm.fit)$coef
```

```
##                 Estimate Std. Error    z value  Pr(>|z|)
## (Intercept) -0.126000257 0.24073574 -0.5233966 0.6006983
## Lag1        -0.073073746 0.05016739 -1.4565986 0.1452272
## Lag2        -0.042301344 0.05008605 -0.8445733 0.3983491
## Lag3         0.011085108 0.04993854  0.2219750 0.8243333
## Lag4         0.009358938 0.04997413  0.1872757 0.8514445
## Lag5         0.010313068 0.04951146  0.2082966 0.8349974
## Volume       0.135440659 0.15835970  0.8552723 0.3924004
```

predict(): predict probability that market will go up given values

```
glm.probs <- predict(glm.fit, type = "response")
glm.probs[1:10]
```

```
##         1         2         3         4         5         6         7         8
```

```
## 0.5070841 0.4814679 0.4811388 0.5152224 0.5107812 0.5069565 0.4926509 0.5092292
##         9        10
## 0.5176135 0.4888378
```

```
contrasts(Direction)
```

```
##      Up
## Down  0
## Up    1
```

```
# convert predicted probabilities into class labels
glm.pred <- rep("Down", 1250)
glm.pred[glm.probs > .5] = "Up"

# confusion matrix
table(glm.pred, Direction)
```

```
##         Direction
## glm.pred Down  Up
##     Down  145 141
##     Up    457 507
```

```
(507+145)/1250
```

```
## [1] 0.5216
```

```
# frac days where prediction was correct
mean(glm.pred == Direction)
```

```
## [1] 0.5216
```

```
# training error rate
100- 52.2
```

```
## [1] 47.8
```

```
# training on held out data

# create boolean vector
train <- (Year < 2005)
Smarket.2005 <- Smarket[!train, ]
dim(Smarket.2005)
```

```
## [1] 252   9
```

```
# separate vector
Direction.2005 <- Direction[!train]

# fit glm using training data
glm.fit <- glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume, data = Smarket, family = binomial, subset = t
```

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

# compute predictions for 2005
glm.pred = rep("Down", 252)
glm.pred[glm.probs > 0.5] = "Up"
table(glm.pred, Direction.2005)
```

```
##         Direction.2005
## glm.pred Down Up
##     Down   77 97
##     Up     34 44
```

```r
# fraction days correct
mean(glm.pred==Direction.2005)
```

```
## [1] 0.4801587
```

```r
# test set error rate
mean(glm.pred!=Direction.2005)
```

```
## [1] 0.5198413
```

```r
# refit using Lag1 Lag2
glm.fit <- glm(Direction~Lag1 + Lag2, data = Smarket, family = binomial, subset = train)

# probabilities
glm.probs <- predict(glm.fit, Smarket.2005, type = "response")
glm.pred <- rep("Down", 252)
glm.pred[glm.probs > 0.5] = "Up"

# confusion matrix
table(glm.pred, Direction.2005)
```

```
##         Direction.2005
## glm.pred Down  Up
##     Down   35  35
##     Up     76 106
```

```r
mean(glm.pred == Direction.2005)
```

```
## [1] 0.5595238
```

```r
106/(106+76)
```

```
## [1] 0.5824176
```

```r
# predict returns given values
predict(glm.fit, newdata = data.frame(Lag1 = c(1.2, 1.5), Lag2 = c(1.1, -0.8)), type = "response")
```

```
##         1         2
## 0.4791462 0.4960939
```

**LDA analysis** fit the LDA model using lda() function (identical to lm())

```
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 4.2.3
```

```
lda.fit = lda(Direction~Lag1+Lag2, data = Smarket, subset=train)
lda.fit
```

```
## Call:
## lda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)
##
## Prior probabilities of groups:
##     Down       Up
## 0.491984 0.508016
##
## Group means:
##             Lag1        Lag2
## Down  0.04279022  0.03389409
## Up   -0.03954635 -0.03132544
##
## Coefficients of linear discriminants:
##             LD1
## Lag1 -0.6420190
## Lag2 -0.5135293
```

Interpretation of results: $\hat{\pi}_1 = 0.492$ and $\hat{\pi}_2 = 0.508$. 49.2% of the training observations correspond to days during which the market went down and these are the average of each predictor within each class and are used by LDA as estimates of $\mu_k$. The group means suggests that there is a tendency for the previous 2 days returns to be negative on days when the market increases and a tendency for the previous days returns to be positive on days when the market declines.

The coefficients of linear discriminant provide the linear combination of lag1 and lag2 that are used to form LDA decision rule. IN other words, these are the multipliers of the elements fo $X = x$ in 4.19. If the LDA decision rule is large, then the classifier will predict a market increase and if small a decrease.

predict() outputs class: LDA predictions about movement of the market posterior: matrix whose $k$th column contains the posterior probability that corresponding observation belongs to the $k$th class x: linear discriminates

```
lda.pred <- predict(lda.fit, Smarket.2005)
names(lda.pred)
```

```
## [1] "class"     "posterior" "x"
```

```
lda.class <- lda.pred$class
table(lda.class, Direction.2005)
```

```
##          Direction.2005
## lda.class Down  Up
##      Down   35  35
##      Up     76 106
```

```
mean(lda.class==Direction.2005)
```

```
## [1] 0.5595238
```

```
# apply 50% threshold
sum(lda.pred$posterior[,1] >= 0.5)
```

```
## [1] 70
```

```
sum(lda.pred$posterior[,1] < 0.5)
```

```
## [1] 182
```

Posterior output by the model corresponds to the probability that the market will decrease:

```
lda.pred$posterior[1:20, 1]
```

```
##       999      1000      1001      1002      1003      1004      1005      1006
## 0.4901792 0.4792185 0.4668185 0.4740011 0.4927877 0.4938562 0.4951016 0.4872861
##      1007      1008      1009      1010      1011      1012      1013      1014
## 0.4907013 0.4844026 0.4906963 0.5119988 0.4895152 0.4706761 0.4744593 0.4799583
##      1015      1016      1017      1018
## 0.4935775 0.5030894 0.4978806 0.4886331
```

```
lda.class[1:20]
```

```
##  [1] Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Down Up   Up   Up
## [16] Up   Up   Down Up   Up
## Levels: Down Up
```

```
# different threshold
sum(lda.pred$posterior[,1] > 0.9)
```

```
## [1] 0
```

### Quadratic discriminant analysis

Fit a QDA model to Smarket data using qda() function

```
qda.fit <- qda(Direction~Lag1+Lag2, data = Smarket, subset = train)
qda.fit
```

```
## Call:
## qda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)
##
## Prior probabilities of groups:
##     Down       Up
## 0.491984 0.508016
```

```
##
## Group means:
##             Lag1        Lag2
## Down  0.04279022  0.03389409
## Up   -0.03954635 -0.03132544
```

Output contains group means but does not contain coeff of linear discriminant because QDA classifier involves quadratic rather than linear function of predictors. predict() functions works the same for QDA.

```
qda.class <- predict(qda.fit, Smarket.2005)$class

# prior probabilities of groups
table(qda.class, Direction.2005)
```

```
##          Direction.2005
## qda.class Down  Up
##      Down   30  20
##      Up     81 121
```

```
# group means
mean(qda.class == Direction.2005)
```

```
## [1] 0.5992063
```

### K-nearest neighbors

Perform KNN using the knn() function; rather than a two step approach knn() forms predictions using a single command and requires four inputs,

1. matrix containing predictors associated with the training data train.X
2. matrix containing data for predictions test.X
3. vector for class labels for the training observations train.Direction
4. value for K to be used by classifier

cbind() using to column bind lag1 and lag2

```
library(class)
```

```
## Warning: package 'class' was built under R version 4.2.3
```

```
# set parameters
train.X <- cbind(Lag1, Lag2)[train, ]
test.X <- cbind(Lag1, Lag2)[!train, ]
train.Direction <- Direction[train]

# set seed for reproducibility
set.seed(1)
knn.pred <- knn(train.X, test.X, train.Direction, k = 1)
table(knn.pred, Direction.2005)
```

```
##         Direction.2005
## knn.pred Down Up
##     Down   43 58
##     Up     68 83
```

```
# correct predictions
(42 + 83) / 252
```

```
## [1] 0.4960317
```

Only about 50% of the observations are correctly predicted. K=1 might be over flexible fit to the data but we repeat the analaysis using k=3

```
# k = 3
knn.pred = knn(train.X, test.X, train.Direction, k=3)
table(knn.pred, Direction.2005)
```

```
##         Direction.2005
## knn.pred Down Up
##     Down   48 54
##     Up     63 87
```

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

```
## [1] 0.5357143
```

Results have improved but it's seen that QDA produces the best results.

**Application Example**

```
dim(Caravan)
```

```
## [1] 5822   86
```

```
attach(Caravan)
summary(Purchase)
```

```
##   No  Yes
## 5474  348
```

```
348/5822
```

```
## [1] 0.05977327
```

Becuase KNN classifier predicts the class of a given test observation by identifying the observations that are nearest to it, variables on a large scale will have a much larger *effect* on the distance between observations and vice versa.

Make sure to *standardize* the data such that all variables are given a mean of 0 and a std. dev. of 1. Then all variables will be on a comparable scale (we can use the function scale()) for this.

```r
# standardize data
standarized.X = scale(Caravan[, -86])
var(Caravan[, 1])
```

```
## [1] 165.0378
```

```r
var(Caravan[, 2])
```

```
## [1] 0.1647078
```

```r
var(standarized.X[, 1])
```

```
## [1] 1
```

```r
var(standarized.X[, 2])
```

```
## [1] 1
```

split the observations into test and traninng and fit the knn model

```r
# test v training split
test <- 1:1000
train.X = standarized.X[-test, ]
test.X = standarized.X[test,]

train.Y = Purchase[-test]
test.Y = Purchase[test]

# fit knn
set.seed(1)
knn.pred <- knn(train.X, test.X, train.Y, k=1)

# knn error rate
mean(test.Y != knn.pred)
```

```
## [1] 0.118
```

```r
mean(test.Y != "No")
```

```
## [1] 0.059
```

The error rate of 12% might look good but since only 6% of customers purchased insurance, we could get the 4rror rate down to 6% by always predicting “No” regardless of value of predictors.

It turns out K=1 does far better than random guessing among the customers that are predicted to buy insurance.

```
table(knn.pred, test.Y)
```

```
##         test.Y
## knn.pred  No Yes
##      No  873  50
##      Yes  68   9
```

```
# among predicted customers to say yes, 12% actually say yes
9 / (68+9)
```

```
## [1] 0.1168831
```

Using larger K = 5 sees the rate of 26.7%,

```
# fit knn model k = 3
knn.pred <- knn(train.X, test.X, train.Y, k = 3)
table(knn.pred, test.Y)
```

```
##         test.Y
## knn.pred  No Yes
##      No  920  54
##      Yes  21   5
```

```
# success rate given pred yes
5 / 26
```

```
## [1] 0.1923077
```

```
# fit knn model k = 5
knn.pred <- knn(train.X, test.X, train.Y, k = 5)
table(knn.pred, test.Y)
```

```
##         test.Y
## knn.pred  No Yes
##      No  930  55
##      Yes  11   4
```

```
# success rate given pred yes
4/15
```

```
## [1] 0.2666667
```

If fitting logistic regression model to the data we use the 0.25 cut off,

```
# fit log model
glm.fit <- glm(Purchase ~., data = Caravan, family = binomial, subset = -test)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
glm.probs <- predict(glm.fit, Caravan[test, ], type = "response")

# convert predicted probabilities into class labels
glm.pred <- rep("No", 1000)
glm.pred[glm.probs > 0.5] = "Yes"

table(glm.pred, test.Y)
```

```
##         test.Y
## glm.pred  No Yes
##      No  934  59
##      Yes   7   0
```

```
# 0.25 threshold
glm.pred[glm.probs > 0.25] = "Yes"
table(glm.pred, test.Y)
```

```
##         test.Y
## glm.pred  No Yes
##      No  919  48
##      Yes  22  11
```

```
# success rate given "Yes"
11 / 33
```

```
## [1] 0.3333333
```

## Exercises

10.

```
# summary
names(Weekly)
```

```
## [1] "Year"      "Lag1"      "Lag2"      "Lag3"      "Lag4"      "Lag5"
## [7] "Volume"    "Today"     "Direction"
```

```
summary(Weekly)
```

```
##       Year           Lag1               Lag2               Lag3
##  Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950
##  1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580
##  Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410
##  Mean   :2000   Mean   :  0.1506   Mean   :  0.1511   Mean   :  0.1472
##  3rd Qu.:2005   3rd Qu.:  1.4050   3rd Qu.:  1.4090   3rd Qu.:  1.4090
##  Max.   :2010   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260
##       Lag4               Lag5              Volume            Today
##  Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747   Min.   :-18.1950
##  1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202   1st Qu.: -1.1540
```

```
##  Median :  0.2380   Median :  0.2340   Median :1.00268   Median :  0.2410
##  Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462   Mean   :  0.1499
##  3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373   3rd Qu.:  1.4050
##  Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821   Max.   : 12.0260
##  Direction
##  Down:484
##  Up  :605
##
##
##
##
```

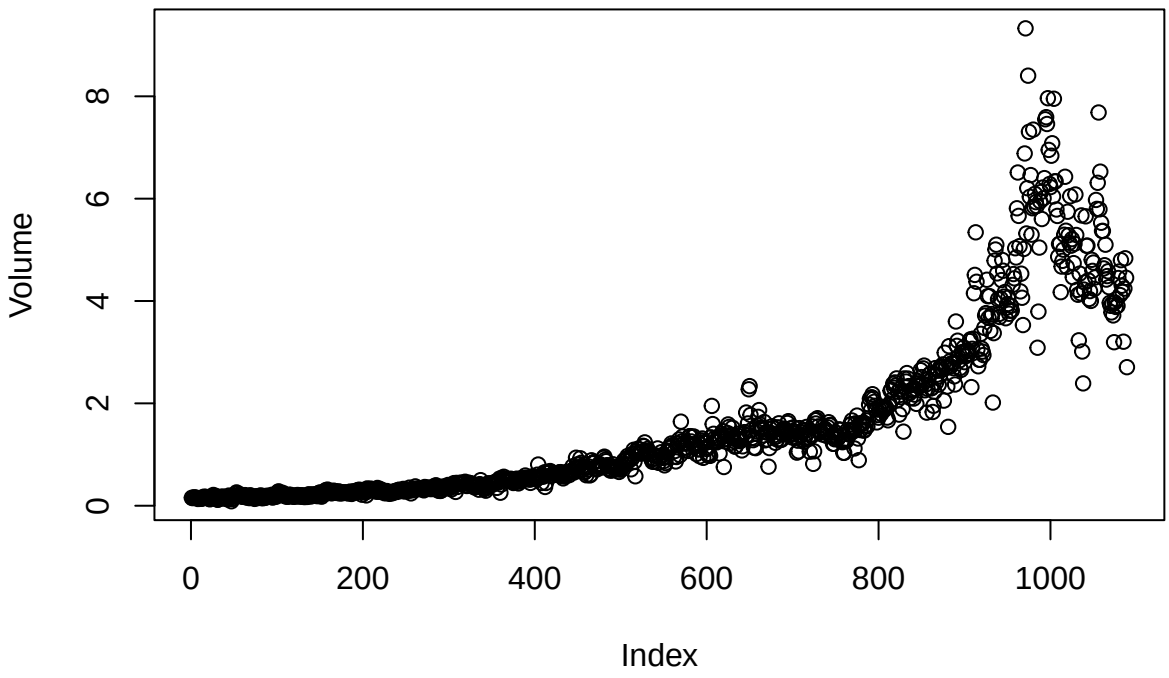
```
# pairwise correlations
cor(Weekly[, -9])
```

```
##              Year         Lag1        Lag2        Lag3         Lag4
## Year    1.00000000 -0.032289274 -0.03339001 -0.03000649 -0.031127923
## Lag1   -0.03228927  1.000000000 -0.07485305  0.05863568 -0.071273876
## Lag2   -0.03339001 -0.074853051  1.00000000 -0.07572091  0.058381535
## Lag3   -0.03000649  0.058635682 -0.07572091  1.00000000 -0.075395865
## Lag4   -0.03112792 -0.071273876  0.05838153 -0.07539587  1.000000000
## Lag5   -0.03051910 -0.008183096 -0.07249948  0.06065717 -0.075675027
## Volume  0.84194162 -0.064951313 -0.08551314 -0.06928771 -0.061074617
## Today  -0.03245989 -0.075031842  0.05916672 -0.07124364 -0.007825873
##                Lag5      Volume        Today
## Year   -0.030519101  0.84194162 -0.032459894
## Lag1   -0.008183096 -0.06495131 -0.075031842
## Lag2   -0.072499482 -0.08551314  0.059166717
## Lag3    0.060657175 -0.06928771 -0.071243639
## Lag4   -0.075675027 -0.06107462 -0.007825873
## Lag5    1.000000000 -0.05851741  0.011012698
## Volume -0.058517414  1.00000000 -0.033077783
## Today   0.011012698 -0.03307778  1.000000000
```

```
attach(Weekly)
```

```
## The following objects are masked from Smarket:
##
##     Direction, Lag1, Lag2, Lag3, Lag4, Lag5, Today, Volume, Year
```

```
plot(Volume)
```




```
# fit log model
glm.fit <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data = Weekly, family = binomial)
summary(glm.fit)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = Weekly)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6949  -1.2565   0.9913   1.0849   1.4579
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.26686    0.08593   3.106   0.0019 **
## Lag1        -0.04127    0.02641  -1.563   0.1181
## Lag2         0.05844    0.02686   2.175   0.0296 *
## Lag3        -0.01606    0.02666  -0.602   0.5469
## Lag4        -0.02779    0.02646  -1.050   0.2937
## Lag5        -0.01447    0.02638  -0.549   0.5833
## Volume      -0.02274    0.03690  -0.616   0.5377
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1496.2  on 1088  degrees of freedom
## Residual deviance: 1486.4  on 1082  degrees of freedom
## AIC: 1500.4
##
## Number of Fisher Scoring iterations: 4
```

```
# coef summary
coef(glm.fit)
```

```
## (Intercept)        Lag1        Lag2        Lag3        Lag4        Lag5
##  0.26686414 -0.04126894  0.05844168 -0.01606114 -0.02779021 -0.01447206
##      Volume
## -0.02274153
```

```
summary(glm.fit)$coef
```

```
##                Estimate Std. Error    z value    Pr(>|z|)
## (Intercept)  0.26686414 0.08592961  3.1056134 0.001898848
## Lag1        -0.04126894 0.02641026 -1.5626099 0.118144368
## Lag2         0.05844168 0.02686499  2.1753839 0.029601361
## Lag3        -0.01606114 0.02666299 -0.6023760 0.546923890
## Lag4        -0.02779021 0.02646332 -1.0501409 0.293653342
## Lag5        -0.01447206 0.02638478 -0.5485006 0.583348244
## Volume      -0.02274153 0.03689812 -0.6163330 0.537674762
```

```
# classification
glm.probs <- predict(glm.fit, type = "response")

glm.pred <- rep("Down", 1089)
glm.pred[glm.probs > 0.5] <- "Up"

table(glm.pred, Direction)
```

```
##         Direction
## glm.pred Down  Up
##     Down   54  48
##     Up    430 557
```

```
# correct predictions
(54 + 557) / (1089)
```

```
## [1] 0.5610652
```

Statistically significant predictors are lag2.

```
train <- (Weekly$Year >= 1990) & (Weekly$Year <= 2008)
test <- !train
glm.fit <- glm(Direction ~ Lag2, data = Weekly, family = binomial, subset = train)
```

```r
# predict()
glm.probs <- predict(glm.fit, newdata = Weekly[test, ], type = "response")

glm.pred <- rep("Down", sum(test))
glm.pred[glm.probs > 0.5] <- "Up"

table(glm.pred, Weekly$Direction[test])
```

```
##
## glm.pred Down Up
##     Down    9  5
##     Up     34 56
```

```r
# accuracy rate
(9 + 56) / length(glm.pred)
```

```
## [1] 0.625
```

LDA

```r
lda.fit <- lda(Direction~ Lag2, data = Weekly, subset = train)
lda.fit
```

```
## Call:
## lda(Direction ~ Lag2, data = Weekly, subset = train)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##             Lag2
## Down -0.03568254
## Up    0.26036581
##
## Coefficients of linear discriminants:
##            LD1
## Lag2 0.4414162
```

```r
lda.pred <- predict(lda.fit, newdata = Weekly[test, ])

names(lda.pred)
```

```
## [1] "class"     "posterior" "x"
```

```r
lda.class <- lda.pred$class

# confusion matrix
table(lda.class, Weekly$Direction[test])
```

```
##
## lda.class Down Up
##      Down    9  5
##      Up     34 56
```

```
# accuracy
mean(lda.class ==Weekly$Direction[test])
```

```
## [1] 0.625
```

```
# apply 50% threshold
sum(lda.pred$posterior[,1] >= 0.5)
```

```
## [1] 14
```

```
sum(lda.pred$posterior[,1] < 0.5)
```

```
## [1] 90
```

QDA

```
qda.fit <- qda(Direction ~ Lag2, data = Weekly, subset = train)
qda.pred <- predict(qda.fit, newdata = Weekly[test, ])
qda.class <- qda.pred$class

# prior probabilities of groups
table(qda.class, Weekly$Direction[test])
```

```
##
## qda.class Down Up
##      Down    0  0
##      Up     43 61
```

```
# group means
mean(qda.class == Weekly$Direction[test])
```

```
## [1] 0.5865385
```

KNN = 1

```
predictors <- Weekly[, "Lag2", drop = FALSE]
response <- Weekly$Direction

# Standardize
standardized.X <- scale(predictors)

# Split the data into training and test sets
set.seed(1)
test_indices <- 1:1000
```

```r
train.X <- standardized.X[-test_indices, , drop = FALSE]
test.X <- standardized.X[test_indices, , drop = FALSE]

train.Y <- response[-test_indices]
test.Y <- response[test_indices]

# KNN model k = 1
knn.pred <- knn(train = train.X, test = test.X, cl = train.Y, k = 1)

# Confusion matrix
table(knn.pred, test.Y)
```

```
##         test.Y
## knn.pred Down  Up
##     Down  172 217
##     Up    277 334
```

```r
# Error rate
mean(test.Y != knn.pred)
```

```
## [1] 0.494
```

```r
# Success rate
sum(knn.pred == "Up" & test.Y == "Up") / sum(knn.pred == "Up")
```

```
## [1] 0.5466448
```