# 5. Classification

## 5.1 Introduction to Classification

Classification problems aim to identify the characteristics that indicate the group to which each case belongs. This pattern can be used both to understand the existing data and to predict how new instances will behave. For example, you may want to predict whether individuals can be classified as likely to respond to a direct mail solicitation, vulnerable to switching over to a competing long distance phone service, or a good candidate for a surgical procedure.

**Data mining creates classification models by examining already classified data (cases) and inductively finding a predictive pattern**. These existing cases may come from an historical database, such as people who have already undergone a particular medical treatment or moved to a new long distance service. They may come from an experiment in which a sample of the entire database is tested in the real world and the results used to create a classifier. For example, a sample of a mailing list would be sent an offer, and the results of the mailing used to develop a classification model to be applied to the entire database. Sometimes an expert classifies a sample of the database, and this classification is then used to create the model which will be applied to the entire database. Given the **class** of each case is **known** in the **training set** when building **classification models**, classification models are therefore known as **supervised methods**.

Fraud detection and credit-risk applications are particularly well suited to this type of analysis. This approach frequently employs decision tree classification algorithms. The use of classification algorithms begins with a training set of pre-classified example transactions. For a fraud detection application, this would include complete records of both fraudulent and valid activities, determined on a record-by-record basis. The classifier training algorithm uses these pre-classified examples to determine the set of parameters required for proper discrimination. The algorithm then encodes these parameters into a model called a classifier.

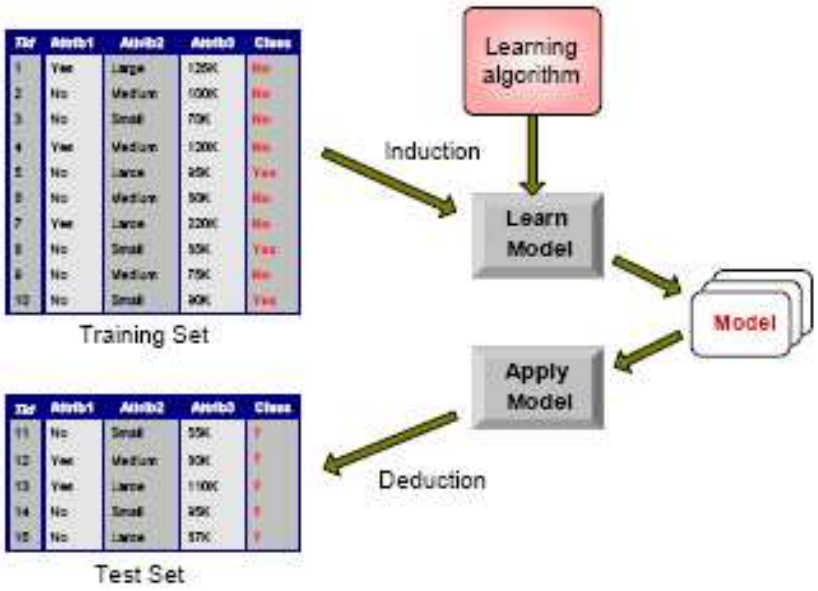
The approach affects the explanation capability of the system. Once an effective classifier is developed, it is used in a predictive mode to classify new records (known as a *test set*) into these same predefined classes. For example, a classifier capable of identifying risky loans could be used to aid in the decision of whether to grant a loan to an individual.

### 5.1.1 Classification Steps

- Given a collection of records (**training set**).
  - Each record contains a set of attributes (or variables), **one** attribute is the **class**.
- Find a **model** for **class attribute as a function of** the values of **other attributes**.
- Goal: previously unseen records should be assigned a class as accurately as possible.

– A **test set** is used to determine the **accuracy** of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to assess the accuracy of the model.

### 5.1.2 Classification Illustration

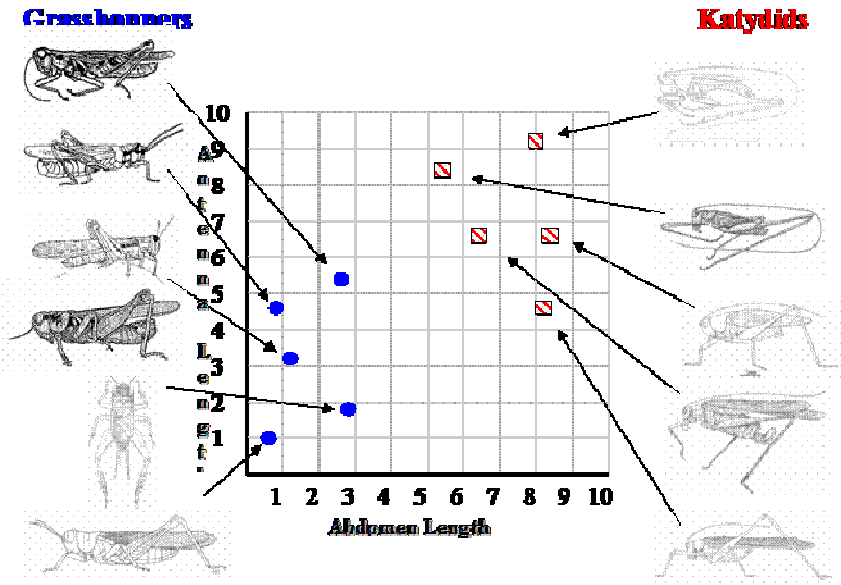



### 5.1.3 Examples of Classification Tasks

- Predicting tumour cells as benign or malignant
- Classifying credit card transactions as legitimate or fraudulent
- Classifying secondary structures of protein as alpha-helix, beta- sheet, or random coil.
- Categorising news stories as finance, weather, entertainment, sports, etc.

### 5.2 Examples of Classifiers and Classification Error

Consider the classification problem for a group of insects. The Antenna length and Abdomen length of 10 insects was measured and recorded. The data points are plotted on the graph below.



Classification deals with associating a class label to a point, according to the class to which the point is believed to belong. Classification can be based on a **distance measure** between the point under consideration and other points belonging to known classes in point space (see Section 3.2). Figure 5.2 shows an example of the classification task in two dimensional point space, where the task is to label a new insect as either belonging to the Katydid class or the Grasshopper class (denoted by a purple diamond).

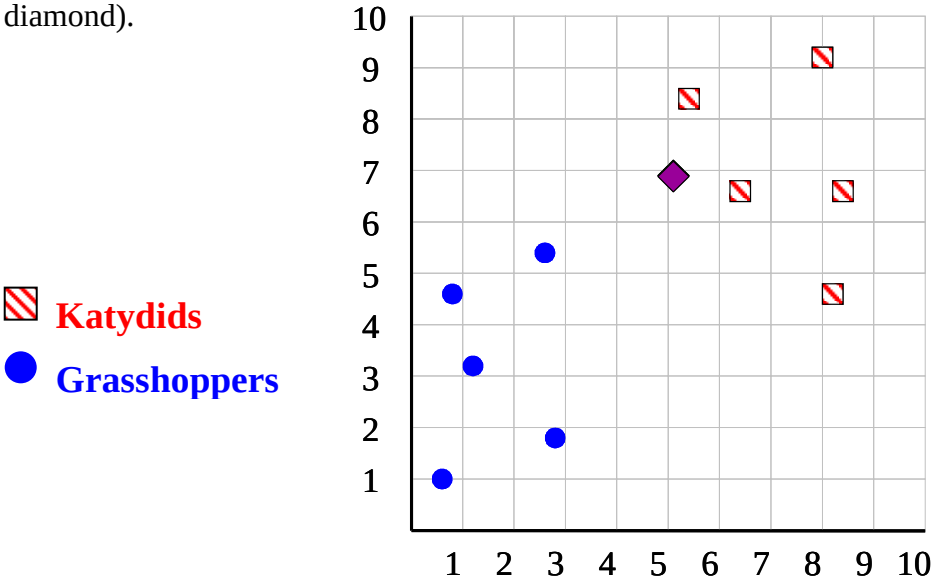



FIGURE 5.2: Two-dimensional point space for Katydids and Grasshoppers.

### 5.2.1 Simple Linear Classifier

The goal of classification is to group items that have similar feature values, into groups. A linear classifier achieves this by making a classification decision based on the value of the linear combination of the features. For a two-class classification problem (e.g. two classes are “yes”/“no”), one can visualise the operation of a linear classifier as splitting a high-dimensional input space with a **hyperplane**: all points on one side of the hyperplane are classified as "yes", while the others are classified as "no". That is, linear classification deals with classifying points according to their position relative to a hyperplane in the point space. All points on a given side of the hyperplane are considered as belonging to the same class. Figure 5.2.1 shows an example of a linear classification in two dimensional point space, where the hyperplane is a line. All points to the left of the hyperplane belong to class Grasshoppers, and all points to the right belong to class Katydids. Any new point is assigned a class label according which side of the line it falls. Linear classification is considered a simple and robust method of classification in the absence of a known model for class data.

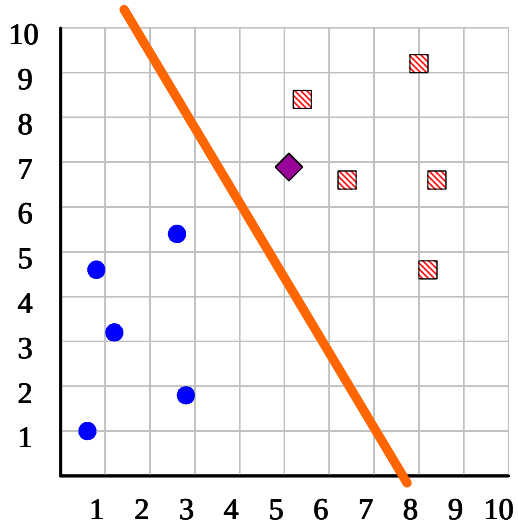



FIGURE 5.2.1: Illustrating a linear classifier.

If previously unseen instance is above the line
then
    class is Katydid
else
    class is Grasshopper

Problems that can be solved by a linear classifier are called **linearly separable**. Alternatively a **piecewise linear classifier** can be generalised to *N* classes, by fitting *N*-1 lines.

**Naïve Bayes** is an example of a **linear classifier** (see Section 5.4.2).

### 5.2.2 Piecewise Linear Classifier

It is not hard to think of configurations of points for which simple linear classification will perform poorly, however. There may exist no hyperplane capable of uniquely separating point classes, because of geometrical reasons or simply because there are more than 2 classes. Figure 5.2.2A shows an example of this problem.

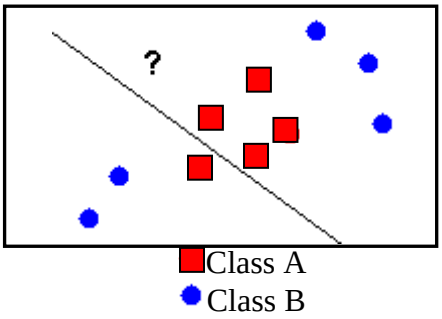



FIGURE 5.2.2A: Linear classification problem

Piecewise linear classification solves this problem by defining boundaries between classes using several hyperplanes. Using multiple hyperplanes, it is possible to define more than 2 regions in space, making it possible to classify a set of points more precisely. Figure 5.2.2B shows how piecewise linear classification works.

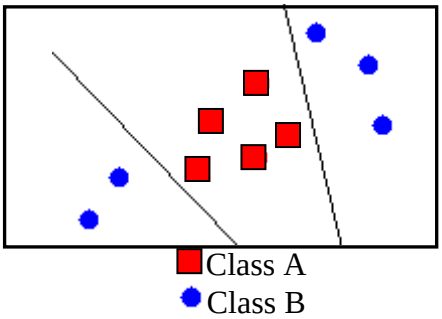



FIGURE 5.2.2B: Piecewise linear classification.

Consider an example which looks at the classification of three types of flowers Setosa, Versicolor and Virginica (see Figure 5.2.2C) – a famous dataset known as **Fisher's iris data set**. In this case we first learned the line to (perfectly) discriminate between Setosa and Virginica/Versicolor, then we learned to approximately discriminate between Virginica and Versicolor. An example of a piecewise linear classification line may be:

If petal width > 3.272 – (0.325 * petal length) then class = Virginica

Elseif petal width …..

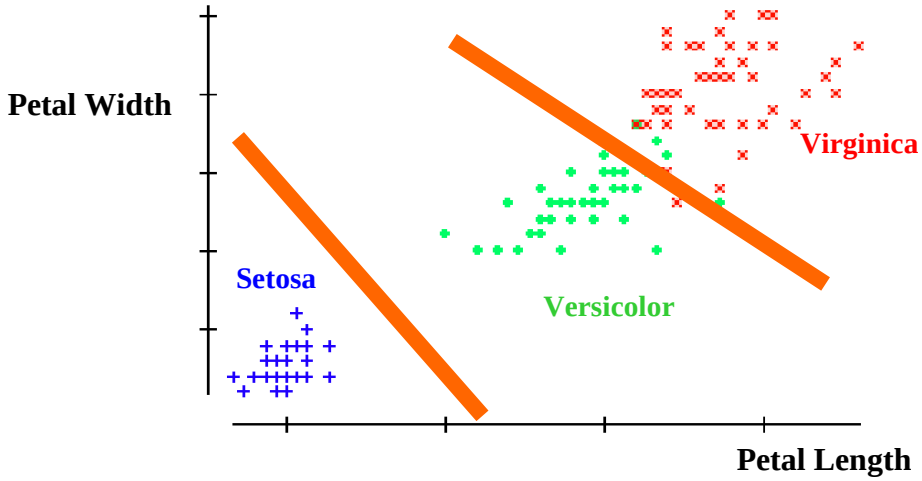



FIGURE 5.2.2C: Piecewise linear classification for Iris Flower data.

### 5.2.3 Non-linear Classifier

A non-linear classifier makes its classification decision based on a non-linear combination of the features. An example of a non-linear classifier is the ***k*-Nearest Neighbour (*k*-NN) algorithm** (see Section 5.5). The decision boundaries of *k*-NN are locally linear segments, but in general have a complex shape that is not equivalent to a line in 2D or a hyperplane in higher dimensions (see Figure 5.2.3 below).

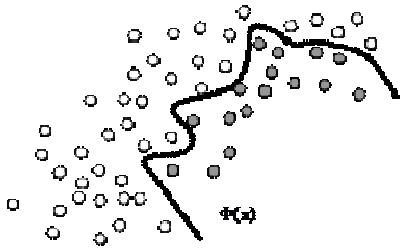


FIGURE 5.2.3: Non-linear classification.

### 5.2.4 Classification Error

Classification is not perfect; points may be misclassified due to factors such as data noise. Classification error can be determined by attempting to classify a set of test points, and counting the number of misclassifications. The **classification error rate** or **accuracy** of the classifier can be estimated by the number of misclassifications divided by the total number of points in the **test set**. In Figure 5.2.4, a point from class A is misclassified as belonging to class B. Since there are 11 points in the test set, and 1 has been misclassified, the classification error rate is 1/11 = 0.09, or 9%.

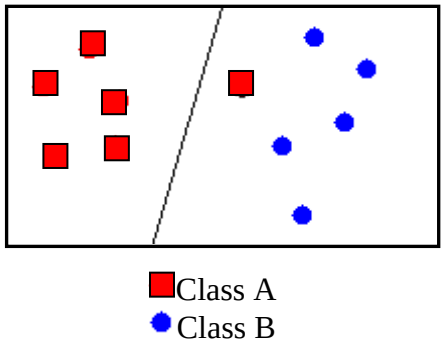



FIGURE 5.2.4: Classification error.

## 5.3 Decision Trees

Tree models, which begin by producing a classification of observations into groups, are usually divided into one of two categories: **regression trees**, when the response variable is continuous; and **classification trees**, when the response variable is quantitative discrete or qualitative.

The idea behind a decision tree is to learn the structure of it using a training sample of data, and then use the resulting tree structure to **create a set of classification rules**. Once such rules are created, new unknown samples can be classified accordingly.

- Decision tree
  - A flow-chart-like tree structure
  - Internal node denotes a test on an attribute (variable)
  - Branch represents an outcome of the test
  - Leaf nodes represent class labels or class distribution

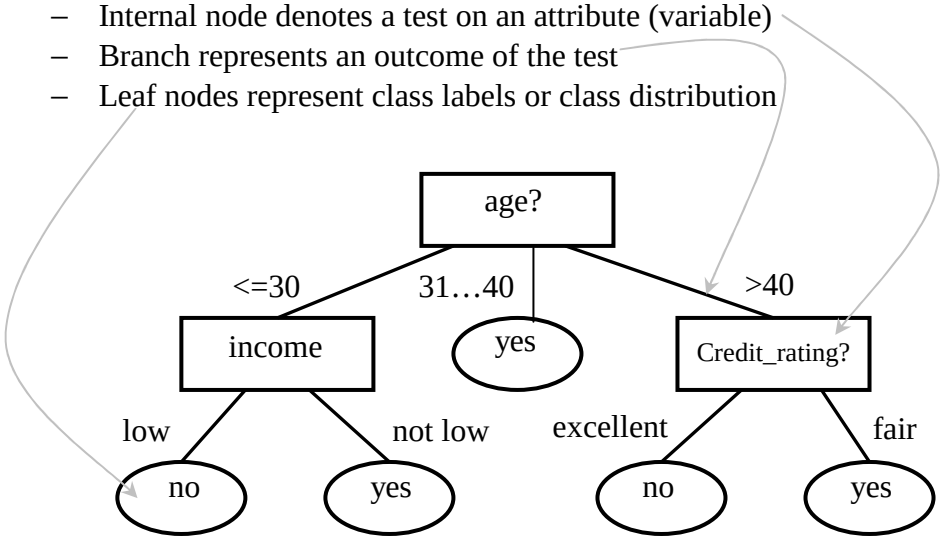
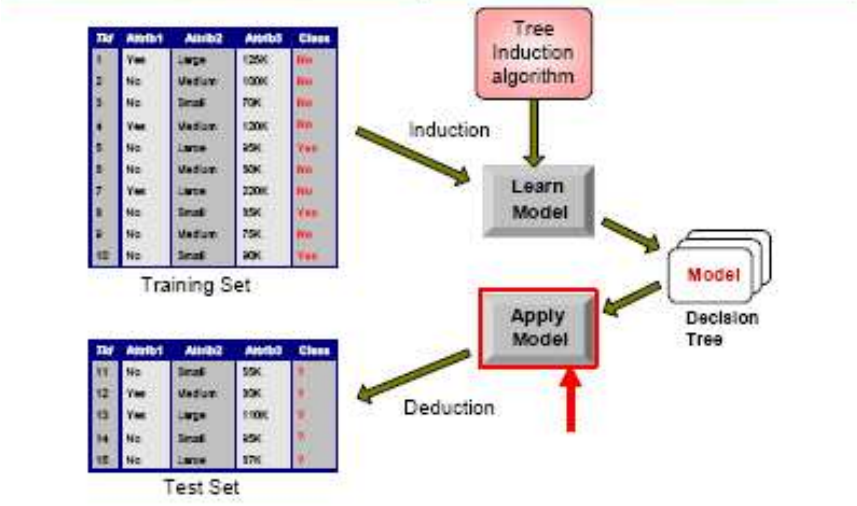



FIGURE 5.3A: A DECISION TREE FOR THE CONCEPT OF BUYING A COMPUTER. THE ATTRIBUTES (VARIABLES) INCLUDED IN THIS CLASSIFICATION ARE THE *AGE* OF THE POTENTIAL PURCHASER (<=30/31-40/>40), THEIR *INCOME* LEVEL (LOW/MEDIUM/HIGH), THEIR STUDENT STATUS (YES/NO) AND WHETHER THEY HAVE EXCELLENT OR FAIR *CREDIT RATING*.

- Decision tree generation consists of two phases
  - **Tree construction**
    - At start, all the training examples are at the root
    - Partition examples recursively based on selected attributes
  - **Tree pruning**
    - Either pre-prune the tree, by halting its construction early (e.g. decide not to add a further split at some node and so node becomes a leaf.)
    - Or post-prune the tree by identifying and removing branches that reflect noise or outliers.
- Use of decision tree: Classifying an unknown sample
  - Test the variable values of the sample against the decision tree

| Customer ID | Student | Age | Income | Credit_rating | Buys Computer |
|---|---|---|---|---|---|
| 1001 | Y | 25 | Medium | F | ? |
| 1002 | N | 56 | High | E | ? |

**Decision Tree Classification Task**




**Tree Induction/ Tree Construction**

- Basic algorithm (a greedy algorithm) - **<u>ID3 algorithm</u>** - split the records based on an attribute test that optimises certain criterion.
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, the tree starts as a single node (the root) representing the training samples.
  - If samples are all in the same class, then the node becomes a leaf and is labeled with that class.
  - Variables are categorical (if continuous-valued, they can be discretised in advance)
  - The algorithm uses a statistical measure (e.g. **information gain**) for selecting the variable that will best separate the samples into individual classes. This variable becomes the decision variable at the node.
  - A branch is created for each known value of the decision variable, and the samples are portioned accordingly.
  - The algorithm uses the same process recursively to form a decision tree for the samples at each partition. Once a variable has been used at a node, it is not considered for any of the node's descendents.

- **Conditions for stopping partitioning**
  - All samples for a given node belong to the same class.
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf.
  - There are no samples left.

***Requirements of the sample data used in ID3 algorithm***

- Variable value description – the same variables must describe each example and have a fixed number of values.
- Predefined classes – an example's variables must already be defined, that is, they are not learned by ID3.
- Sufficient examples – there must be enough test cases to distinguish valid patterns from chance occurrences.

***How does the ID3 decide which variable is the best to use for splitting?***

- Splitting Criteria: A statistical property called *Information Gain* is used for selecting the variable that will best separate the samples into individual classes.
  - Information gain measures how well a given variable separates training examples into targeted classes.
  - The one with the highest information gain (information being the most important for classification) is selected for branching.
- In order to define *information gain*, we use a statistic called *entropy* (see Section 2.2.3), which measures the amount of information (or dispersion) in a variable or sample.
  - Given a collection $S$ of $c$ possible outcomes, the entropy of $S$, $E(S)$, is given by

$$E(S) = \sum_{I=1}^{c} -p(I)\log_c p(I)$$

  - Thus if there are two possible outcomes (e.g. yes/no) the entropy is given by:

$$E(S) = \sum_{I=1}^{2} -p(I)\log_2 p(I)$$

  - The entropy is 0 if all members of $S$ belong to the same class (i.e. perfect homogeneity) and 1 if all observations are uniformly distributed across the $c$ possible outcomes (i.e. maximum heterogeneity).

Quick example:

If $S$ is a collection of 15 answers with 9 YES and 6 NO examples (i.e. two possible outcomes), then

$$E(S) = -(9/15)\log_2(9/15) - (6/15)\log_2(6/15) = 0.9710\,.$$

- The information gain measures the *expected* reduction in entropy caused by knowing the value of a variable (or feature or attribute) $A$. That is, the information gain of a example set $S$ by branching on a variable $A$ (where the variable $A$ has $\nu$ distinct values $\{a_1, a_2, \ldots, a_\nu\}$) is defined as

$$\text{Information Gain}(S, A) = E(S) - \sum_{i=1}^{\nu} \frac{|S_i|}{|S|} E(S_i)$$

where

- $S_i$ = subset of $S$ for which variable $A$ has value $a_i$,
- $|S_i|$ = number of elements in $S_i$,
- $|S|$ = number of elements in $S$,
- $E(S_i)$ is the entropy of the subset $S_i$,

**Example 1**

Let's build a classification tree to decide where a game of baseball should go ahead or not. We are provided with four variables regarding the weather and one variable describing whether previous games have been played or not:

*Outlook* = {sunny, overcast, rain}
*Temperature* = {hot, mild, cool}
*Humidity* = {high, normal}
*Wind* = {weak, strong}

*Play ball* = {yes, no}

We are provided with the following example set *S* and ask to build a decision tree using the criterion of information gain:

| Day | *Outlook* | *Temperature* | *Humidity* | *Wind* | *Play ball?* |
|---|---|---|---|---|---|
| 1 | Sunny | Hot | High | Weak | No |
| 2 | Sunny | Hot | High | Strong | No |
| 3 | Overcast | Hot | High | Weak | Yes |
| 4 | Rain | Mild | High | Weak | Yes |
| 5 | Rain | Cool | Normal | Weak | Yes |
| 6 | Rain | Cool | Normal | Strong | No |
| 7 | Overcast | Cool | Normal | Strong | Yes |
| 8 | Sunny | Mild | High | Strong | No |
| 9 | Sunny | Cool | Normal | Weak | Yes |
| 10 | Rain | Mild | Normal | Weak | Yes |
| 11 | Sunny | Mild | Normal | Weak | Yes |
| 12 | Overcast | Mild | High | Strong | Yes |
| 13 | Overcast | Hot | Normal | Weak | Yes |
| 14 | Rain | Mild | High | Strong | No |
| 15 | Sunny | Cool | High | Weak | No |

We need to find out which variable will be the root node in our decision tree. The gain is calculated for the four variables:

∴

TEMPERATURE

Gain(S,temperature) = E(S) – (4/15)E($S_{hot}$) – (6/15)E($S_{mild}$) – (5/15)E($S_{cool}$)
where:
E(S) = -(9/15)$\log_2$(9/15) – (6/15)$\log_2$(6/15) = 0.9710
E($S_{hot}$) = -(2/4)$\log_2$(2/4) – (2/4)$\log_2$(2/4) = 1
E($S_{mild}$) = -(4/6)$\log_2$(4/6) – (2/6)$\log_2$(2/6) = 0.9183
E($S_{cool}$) = -(3/5)$\log_2$(3/5) – (2/5)$\log_2$(2/5) = 0.9710

∴ Gain(S,temperature) = 0.013

HUMIDITY

Gain(S,humidity) = E(S) – (8/15)E($S_{high}$) – (7/15)E($S_{norm}$)
where:
E(S) = -(9/15)$\log_2$(9/15) – (6/15)$\log_2$(6/15) = 0.9710
E($S_{high}$) = -(3/8)$\log_2$(3/8) – (5/8)$\log_2$(5/8) = 0.9544
E($S_{norm}$) = -(6/7)$\log_2$(6/7) – (1/7)$\log_2$(1/7) = 0.5917

∴ Gain(S,humidity) = 0.186

WIND

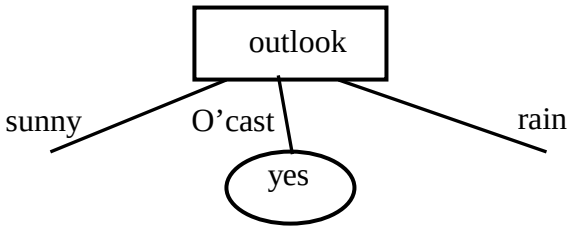
Gain(S,wind) = E(S) – (9/15)E($S_{weak}$) – (6/15)E($S_{strong}$)
where:
E(S) = -(9/15)$\log_2$(9/15) – (6/15)$\log_2$(6/15) = 0.9710
E($S_{weak}$) = -(7/9)$\log_2$(7/9) – (2/9)$\log_2$(2/9) = 0.7642
E($S_{strong}$) = -(2/6)$\log_2$(2/6) – (4/6)$\log_2$(4/6) = 0.9183

∴ Gain(S,wind) = 0.145

The *outlook* variable has the greatest gain; therefore we use it as the decision variable in the root node.

From observing the above table of data, we can see that on any day in which the outlook was overcast, the game was played; therefore we can create a leaf along the "overcast" branch and label it by the class "yes" (see stopping criterion 1).

Thus so far we have:




The next step is to determine which variables may be added to the "sunny" branch and the "rain"

branch. There are only three remaining variables: *temperature*, *humidity* and *wind*.

Consider the "sunny" branch first. Now we only have to consider those examples in *S* that belong to *outlook* = "sunny".

∴

HUMIDITY

$Gain(S_{sunny},humidity) = E(S_{sunny}) - (4/6)E(S_{sunny,high}) - (2/6)E(S_{sunny,norm})$
where:
$E(S_{sunny,high}) = -(0/4)\log_2(0/4) - (4/4)\log_2(4/4) = 0$
$E(S_{sunny,norm}) = -(2/2)\log_2(2/2) - (0/2)\log_2(0/2) = 0$

∴ $Gain(S_{sunny},humidity) = 0.9183$

WIND

$Gain(S_{sunny},wind) = E(S_{sunny}) - (4/6)E(S_{sunny,weak}) - (2/6)E(S_{sunny,\ strong})$
where:
$E(S_{sunny,weak}) = -(2/4)\log_2(2/4) - (2/4)\log_2(2/4) = 1$
$E(S_{sunny,strong}) = -(0/2)\log_2(0/2) - (2/2)\log_2(2/2) = 0$

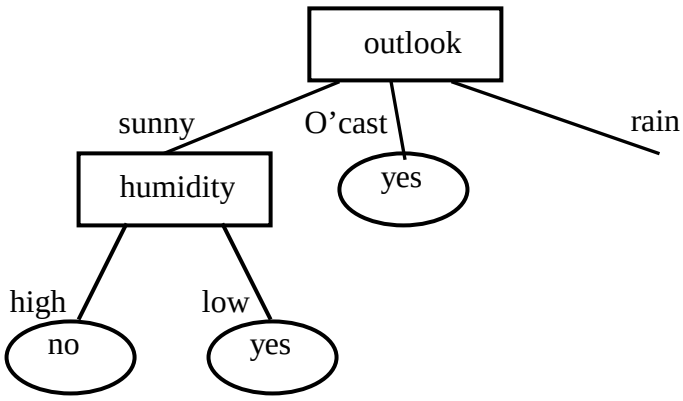
∴ $Gain(S_{sunny},wind) = 0.252$

The *humidity* variable has the greatest gain along the sunny branch; therefore we use it as the decision variable at the end of the "sunny" branch.

Note also on any day in which the outlook was sunny, with high humidity, the game was not played; therefore we can create a leaf at this branch and label it by the class "no". Also, in any day in which

the outlook was sunny, with low humidity, the game was played; therefore we can create a leaf at this branch and label it by the class "yes".

Thus so far we have:




The next step is to determine which variables may be added to the "rain" branch. At this point, there are only three remaining variables: *temperature*, *humidity* and *wind*. Now we only have to consider those examples in *S* that belong to *outlook* = "rain".

HUMIDITY

$Gain(S_{rain},humidity) = E(S_{rain}) - (2/5)E(S_{rain,high}) - (3/5)E(S_{rain,norm}) = 0.020$
given:
$E(S_{rain,high}) = -(1/2)\log_2(1/2) - (1/2)\log_2(1/2) = 1$
$E(S_{rain,norm}) = -(2/3)\log_2(2/3) - (1/3)\log_2(1/3) = 0.9183$

WIND

$Gain(S_{rain},wind) = E(S_{rain}) - (3/5)E(S_{rain,weak}) - (2/5)E(S_{rain,strong}) = 0.971$

given:
$E(S_{rain,weak}) = -(3/3)\log_2(3/3) – (0/3)\log_2(0/3) = 0$
$E(S_{rain,strong}) = -(0/2)\log_2(0/2) – (2/2)\log_2(2/2) = 0$

The *wind* variable has the greatest gain. And so our final decision tree looks like:

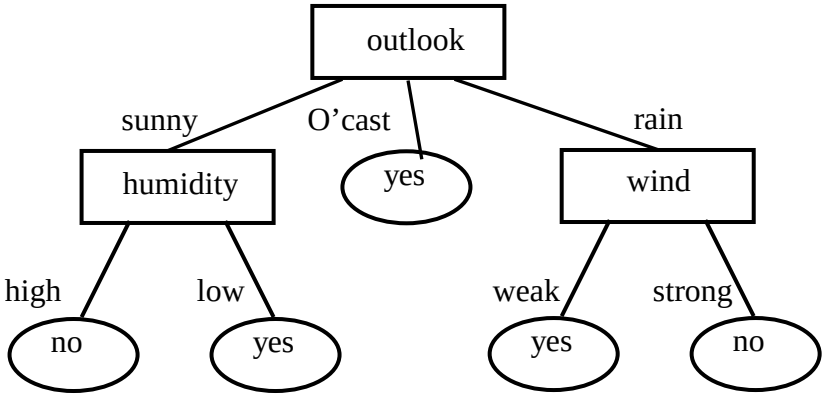



FIGURE 5.3B: A DECISION TREE FOR THE CONCEPT OF PLAYING A GAME OF BASEBALL.

Such a decision tree could then be used to classify a new example:

| | | | | | |
|---|---|---|---|---|---|
| 26 | Rain | Mild | High | Strong | ?? |

For example, here it would predict that play would not occur.

**When constructing a decision tree, traditional methods use log to the base 2. Log to base 10, $\log_{10}$, or natural logs, $\log_e$, can be also be used in gain information calculations, as comparisons in the calculated value are only being compared.** Note however that the $0 \leq \text{entropy} \leq 1$ constraint indicating no/perfect classification will not be valid if using $\log_{10}$ or $\log_e$.

### 5.3.1 Extracting Classification Rules from Decision Trees.

The knowledge represented in decision trees can be extracted and represented in the form of classification IF-THEN rules. One rule is created for each path from the root to a leaf. The rules extracted from Figure 5.3B are:

IF outlook = sunny AND humidity = high THEN play ball = no
IF outlook = sunny AND humidity = low THEN play ball = yes
IF outlook = overcast THEN play ball = yes
IF outlook = rain AND wind = weak THEN play ball = yes
IF outlook = rain AND wind = strong THEN play ball = no

Classification trees have been implemented in a number of real-world applications including medical diagnosis, credit risk assessment of loan applications, equipment failures etc.

### 5.3.2 Advantages/Disadvantages of Decision Trees

- Advantages:
  - Easy to understand.
  - Easy to simple and easily-understood classification rules.
  - Relatively fast (compared to other classification methods).
- Disadvantages:
  - May suffer from overfitting.
  - Classifies by rectangular partitioning (so does not handle correlated features very well).
  - Can be quite large – pruning is sometimes necessary.
  - May suffer from "noisy" data

## 5.3.3 Other decision tree algorithms

Thus far we have only considered the ID3 algorithm for constructing the decision trees, however many other algorithms exist. For example, the **C4.5 algorithm**, while it is based on many of the principles of the ID3 algorithm, includes a number of improvements:

- C4.5 can handle both continuous and discrete attributes - in order to handle continuous variables, C4.5 creates a threshold and then splits the list into those whose variable value is above the threshold and those that are less than or equal to it.
- C4.5 can handle training data with missing variable values - C4.5 allows variable values to be marked as missing. Missing variable values are simply not used in gain and entropy calculations.
- C4.5 can handle variables with differing costs – it can account for the fact that some variables may be more expensive to ascertain than others.
- C4.5 can prune trees after creation

As with the ID3 algorithm, C4.5 uses the fact that each attribute of the data can be used to make a decision that splits the data into smaller subsets. That is, C4.5 examines the normalised information gain (difference in entropy) that results from choosing an attribute for splitting the data. The attribute with the highest normalised information gain is the one used to make the decision. The algorithm then recurs on the smaller sub-lists.

In order to handle continuous variables, the best **split-point** for a continuous variable $A_k$ must be determined. That is, in order to branch by this variable $A_k$ we need to decide a value for the continuous variable (the split-point), where observations below this value will follow one branch while values above will follow the alternative branch. Typically the observed values of $A_k$ in the training set would be sorted and the information gain would be calculated by having the split-point at each and every of the mid-points of the sorted values in the observed training set. The position of the split-point with the best information gain is then chosen for the actual split-point. The information gain with this variable (using this best split point) can then be compared in the usual way to the information gained through branching by the other possible variables.

### 5.3.4 Noisy Data

- Frequently, training data contains "noise" - i.e. examples which are misclassified, or where one or more of the attribute values is wrong.
- In such cases, one is like to end up with a part of the decision tree which considers say 100 examples, of which 99 are in class $C_1$ and the other is apparently in class $C_2$ (because it is misclassified).
- If there are any unused attributes, we *might* be able to use them to elaborate the tree to take care of this one case, but the subtree we would be building would in fact be wrong, and would likely misclassify real data.
- Thus, particularly if we know there is noise in the training data, it may be wise to "prune" the decision tree to remove nodes which, statistically speaking, seem likely to arise from noise in the training data.

### 5.3.5 Using validation data to prune a decision tree

Decision trees like many classifiers are prone to **overfitting** (that is they can be overly complex, describing even the noise in the data). To overcome this, the data can be split into a training, test and **validation** set.

Training set:

A set of examples used for learning - that is to fit the parameters of the classifier.

Validation set:

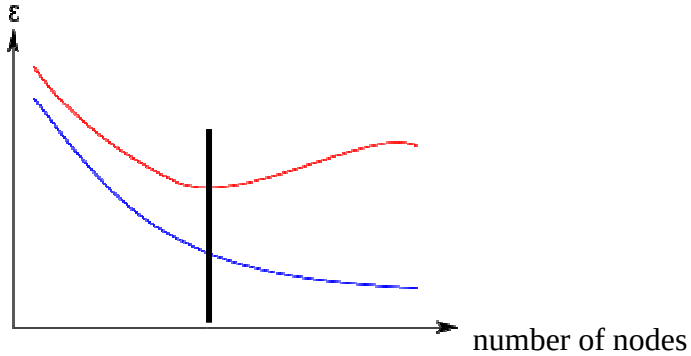
A set of examples used to tune the parameters of a classifier - for example to choose the number of nodes to keep in a decision tree.

Test set:

A set of examples used only to assess the performance of a fully-specified classifier.

For example, imagine you are considering having 15 or 20 nodes in your decision tree – you want to decide whether you should prune back or not. The validation set is used to compare the performance of the 15-node and 20-node decision trees and so decide on which one to take. Thus it is likely that the error in the training data is less for the decision tree with 20 nodes. However applying the 15-node and 20-node decision trees to the validation data, the error may actually be worse for the 20-node decision tree – due to overfitting. If this is the case the 15-node decision tree is preferred.

In general while the error in the training set decreases as the decision tree becomes more complex (blue line below), the same may not occur for the validation data. The best predictive and fitted model would be where the error in the validation set (red line below) has its global minimum.



## 5.4 Bayesian Classifiers

What are **Bayesian Classifiers**? They are statistical classifiers that can predict class membership probabilities, such as the probability that a given sample belongs to a particular class. Bayesian classification is based on Bayes' Theorem, described below. One particular kind of a Bayesian classifier is a ***naïve Bayesian classifier***, also called:

- Idiot Bayes
- Simple Bayes

This assumption is called **class conditional independence**. It is made to simplify the computations and, in this sense, is considered naïve. First, let's introduce Bayes' theorem.

### 5.4.1 Bayes' Theorem

Let $X$ be a data sample whose class label is unknown. Let $H$ be some hypothesis, such as that the data sample $X$ belongs to a specified class $C$. For classification purposes we want to determine $P(H|X)$ (**posterior probability of $H$ given $X$**), the probability that the hypothesis $H$ holds given the observed data sample $X$.

For example, let $X$ be a data sample describing some piece of fruit. Let's pretend we have apples, oranges and grapes. Let $X$ = red and round. Let $H$ be the hypothesis that $X$ is an apple. Then $P(H|X)$ reflects our confidence that $X$ is an apple given that we have seen that $X$ is red and round. We can determine $P(H|X)$ using Bayes' formula:

where $P(H)$ (prior probability) is simply the probability that any given data sample is an apple. This can be calculated from our data set. $P(X|H)$ is the conditional probability that X is red and round, given that we knew it was an apple (likelihood of the sample data). This can also be calculated from some data set. Finally, $P(X)$ is the probability that a data sample from our data set is red and round. Again, this can be estimated from our data set. Now let's see how we can use Bayes' theorem in **Naïve Bayes Classification**.

### 5.4.2 Naïve Bayes Classification

We are about to see some of the mathematical formalisms, and more examples, but keep in mind the basic idea.

1. Each data sample is represented by an $n$-dimensional feature vector, $\boldsymbol{X} = (x_1, x_2, \ldots, x_n)$,

depicting $n$ measurements made on a sample from $n$ variables, respectively, $A_1, A_2, \ldots, A_n$.

2. Suppose that there are $m$ classes, $C_1, C_2, \ldots, C_m$. Given an unknown data sample, $\boldsymbol{X}$, (i.e. having no class label), the classifier will predict that $\boldsymbol{X}$ belongs to the class having the highest posterior probability, conditioned on $\boldsymbol{X}$. That is, the naïve Bayes classifier assigns an unknown sample $\boldsymbol{X}$ to the class $C_i$ if and only if

$$P(C_i \mid \boldsymbol{X}) > P(C_j \mid \boldsymbol{X}) \text{ for all } 1 \le j \le m, j \ne i$$

Thus we maximise $P(C_i|\boldsymbol{X})$. By Bayes' Theorem

$$P(C_i \mid \boldsymbol{X}) = \frac{P(\boldsymbol{X} \mid C_i)P(C_i)}{P(\boldsymbol{X})}.$$

3. The objective therefore is to calculate the right hand side of the above equation and maximise it. As $P(\boldsymbol{X})$ is constant for all classes, $P(\boldsymbol{X} \mid C_i)P(C_i)$ need only be maximised.
4. The class prior probabilities may be estimated from the data by $P(C_i) = s_i/s$ where $s_i$ is the number of training samples of class $C_i$ and $s$ is the total number of training samples. If it is assumed that each of the classes are equally likely (that is, the priors are all equal: $P(C_1) = P(C_2) = \ldots. = P(C_m)$), then we only need to maximise $P(\boldsymbol{X}|C_i)$.
5. In order to reduce computational time, we assume that the naïve assumption of class conditional independence holds, and thus

$$P(\boldsymbol{X} \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$$

where $P(x_1 \mid C_i), P(x_2 \mid C_i), \ldots, P(x_n \mid C_i)$ are estimated from the training sample (the data).

Note:
(a) if variable $A_k$ is categorical, $P(x_k \mid C_i)$ can be estimated from a training sample by counting the number of observations in the class $C_i$ having the value $x_k$ for $A_k$, and dividing by the total number of observations in the class $C_i$.
(b) if variable $A_k$ is continuous-valued, then the variable is typically assumed to have a normal distribution so that:

$$P(x_k \mid C_i) = \frac{1}{\sqrt{2\pi}\sigma_{C_i}} \exp\left(-\frac{(x_k - \mu_{C_i})^2}{2\sigma_{C_i}^2}\right)$$

where $\mu_{C_i}$ and $\sigma_{C_i}$ are the mean and standard deviation, respectively, of the values of the variable $A_k$ for training samples of class $C_i$.

6. In order to classify an unknown sample $\boldsymbol{X}$, $P(\boldsymbol{X}|C_i)P(C_i)$ is evaluated for all $C_i$. Sample $\boldsymbol{X}$ is then assigned to the class $C_i$ for which $P(\boldsymbol{X}|C_i)P(C_i)$ is maximum.

***Find out the probability of a previously unseen instance belonging to each class, then simply pick the most probable class.***

The Naive Bayes classifiers are often represented as this type of graph…

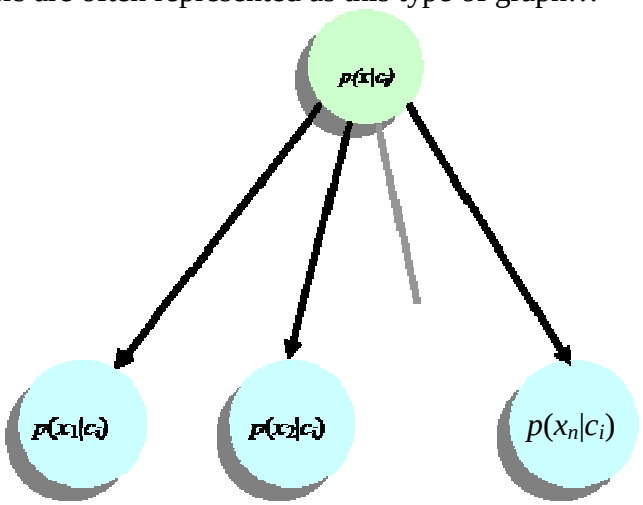



FIGURE 5.4.2: GENERALISED NAÏVE BAYESIAN CLASSIFICATION TREE.

### 5.4.3 Advantages/Disadvantages of Naïve Bayes

- Advantages:
  - Fast to train (single scan). Fast to classify.
  - Speed compares to decision trees.
  - Handles real and discrete data
- Disadvantages:
  - Assumes independence of features – often a "difficult" assumption in reality.

**Example 2: using Bayesian Classifiers**

The following table refers to data for an electronics company. Such a company may wish to use classification techniques to determine if a new customer is likely to buy a computer or not, based on the knowledge they know from previous sales. The data here refers to the transactions of 14 previous customers. We wish to decide if customer with ID15 purchases a new computer using the **Naïve Bayes Classification**.

| ID | Age | Income | student | Credit_rating | Class: Buys? |
|---|---|---|---|---|---|
| 1 | <=30 | High | No | Fair | No |
| 2 | <=30 | High | No | Excellent | No |
| 3 | 31 – 40 | High | No | Fair | Yes |
| 4 | >40 | Medium | No | Fair | Yes |
| 5 | >40 | Low | Yes | Fair | Yes |
| 6 | >40 | Low | Yes | Excellent | No |
| 7 | 31 – 40 | Low | Yes | Excellent | Yes |
| 8 | <=30 | Medium | No | Fair | No |
| 9 | <=30 | Low | Yes | Fair | Yes |

| | | | | | |
|---|---|---|---|---|---|
| 10 | >40 | Medium | Yes | Fair | Yes |
| 11 | <=30 | Medium | Yes | Excellent | Yes |
| 12 | 31 – 40 | Medium | No | Excellent | Yes |
| 13 | 31 – 40 | High | Yes | Fair | Yes |
| 14 | >40 | Medium | No | Excellent | No |
| 15 | <=30 | Medium | Yes | Fair | ?? |

We need to maximise $P(\boldsymbol{X}|C_i)P(C_i)$ for $i$=1,2.

$P(C_i)$, the prior probability of each class, can be computed straight from the above table of data (training samples):

To compute $P(\boldsymbol{X}|C_i)$ for ID15, we compute the following conditional probabilities: $P(x_1|C_1)$, $P(x_2|C_1)$, $P(x_1|C_2)$ etc, again directly from the above data set.

Variable $A_2$: income (categorical variable 3 classes: “low”, “medium”, “high”)
$P(x_2|C_1)$=$P$(income = “medium” | buys = “yes”) = 4/9 = 0.444
$P(x_2|C_2)$= $P$ (income = “medium” | buys = “no”) = 2/5 = 0.400
Variable $A_3$: student (categorical variable 2 classes: “yes”, “no”)
$P(x_3|C_1)$=$P$ (student = “yes” | buys = “yes”) = 6/9 = 0.667
$P(x_3|C_2)$=$P$ (student = “yes” | buys = “no”) = 1/5 = 0.200
Variable $A_4$: credit rating (categorical variable 2 classes: “fair”, “excellent”)
$P(x_4|C_1)$=$P$ (credit_rating = “fair” | buys = “yes”) = 6/9 = 0.667
$P(x_4|C_2)$=$P$ (credit_rating = “fair” | buys = “no”) = 2/5 = 0.400

Using these probabilities we then obtain

$$P(\boldsymbol{X}|\text{buys} = \text{“yes”}) = \prod_{k=1}^{4} P(x_k \mid C_1) = 0.222 \text{ x } 0.444 \text{ x } 0.667 \text{ x } 0.667 = 0.044$$

And therefore:

$P(\boldsymbol{X}|C_1)P(C_1)$ =$P(\boldsymbol{X}|$buys = “yes”)$P$(buys = “yes”) = 0.044 x 0.643 = 0.028
$P(\boldsymbol{X}|C_2)P(C_2)$ =$P(\boldsymbol{X}||$buys = “no”)P(buys = “no”) = 0.019 x 0.357 = 0.007

### 5.4.4 The Laplacian correction to the naïve Bayesian classifier

Using the naïve Bayesian classifier we estimate $P(\boldsymbol{X}\,|\,C_i)$ as a product of the probabilities $P(x_1\,|\,C_i), P(x_2\,|\,C_i), \ldots, P(x_n\,|\,C_i)$, based on the assumption of class conditional independence. These probabilities $P(x_k\,|\,C_i)$ are estimated from the training set, and are based on the number of observations seen in the training set. It is possible that if the true conditional probability $P(x_k\,|\,C_i)$ is some small value, the training set may not contain any observations for a particular $x_k$ and $C_i$, thus the probability $P(x_k\,|\,C_i)$ would be estimated as zero. Thus this zero probability would result in $P(\boldsymbol{X}\,|\,C_i)$ being inappropriately estimated as zero also.

To avoid this problem we can add a small number to the counts in both the numerator and denominator of the probability estimation. If the training set is large this Laplacian correction makes only a negligible difference to the probability estimation, but ensures no probability is zero.

For example, suppose we considered the observations in a training set in one particular class (e.g. *buys_computer=yes*). If there were 1000 observations within this class - 0 with *income=low*, 990 with *income=medium* and 10 with *income=high*. The probabilities of these events without the Laplacian correction are 0, 0.990, and 0.010 respectively. Using the Laplacian correction we instead obtain:

$$\frac{1}{1003} = 0.001, \frac{991}{1003} = 0.988, \text{ and } \frac{11}{1003} = 0.011$$

respectively.

### 5.4.5 Bayesian Belief Networks

The naïve Bayesian classifier makes the assumption of class conditional independence in order to simplify computation. In practice however, dependencies can exist between variables. **Bayesian belief networks** (also known as belief networks, Bayesian networks, probabilistic networks) specify **joint conditional probability distributions**. A belief network is defined by a **graphical model of causal relationships** (a directed acyclic graph) on which learning has been performed to produce a set of conditional probability tables. Trained Bayesian belief networks can be used for classification.

Each node of the directed acyclic graph represents a random variable, (which can be discrete or continuous-valued). If an arc (or arrow) is drawn from node *Y* to node *Z*, it indicates that there is a probabilistic dependence between the variables, where *Z* is a **descendant or child** of *Y* (and *Y* is a **parent or immediate predecessor** of *Z*). Each variable is conditionally independent of its non-descendents in the graph, given its parents.

For discrete random variables, the **joint probability function** of a Bayesian network (containing variables $X_1,\ldots,X_n$) is given by:

$$\mathrm{P}\left(X_1=x_1,\ldots,X_n=x_n\right)=\prod_{i=1}^{n}\mathrm{P}\left(X_i=x_i\middle|X_j=x_j \text{ for each } X_j \text{ which is a parent of } X_i\right).$$

In order to train or learn Bayesian belief networks, the **network topology** (or "layout" of the nodes and arrows) may be given in advance or inferred from the data. The network variables may be observable or hidden in all or some of the training observations.

If the network topology is **known** and the variables are observable, training of the network is straightforward – the **conditional probability tables** for each variable (with entries specifying $P(Z|Parents(Y))$) are computed based on the observations in the training set.

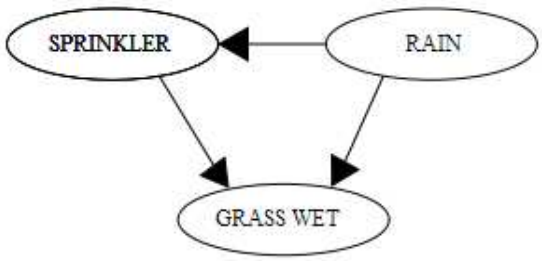
When the network topology is known and some of the variables are hidden, the **gradient descent** iterative strategy can be used.

---

**Example 3: using Bayesian Belief Networks**

Suppose that there are two events which could cause grass to be wet: either the sprinkler is on or it's raining. Also, suppose that the rain has a direct effect on the use of the sprinkler (namely that when it rains, the sprinkler is usually not turned on). Then the situation can be modelled with the Bayesian network illustrated below. All three variables have two possible values T (for true) and F (for false).

| | SPRINKLER | |
|---|---|---|
| RAIN | T | F |
| F | 0.4 | 0.6 |
| T | 0.01 | 0.99 |




| | RAIN | |
|---|---|---|
| T | F | |
| 0.2 | 0.8 | |

| | | GRASS WET | |
|---|---|---|---|
| SPRINKLER | RAIN | T | F |
| F | F | 0.0 | 1.0 |
| F | T | 0.8 | 0.2 |
| T | F | 0.9 | 0.1 |
| T | T | 0.99 | 0.01 |

Question: Determine the probability that it is raining, given the grass is wet. Thus, determine the classification or whether it is raining or not given the grass is wet.

Solution:
The joint probability distribution is:

$P(G,S,R) = P(G \mid S,R)P(S \mid R)P(R)$

where the names of variables have been abbreviated to $G$ = Grass wet, $S$ = Sprinkler, and $R$ = Rain.

The probability that it is raining, given the grass is wet is given by:

$$P(R=T \mid G=T) = \frac{P(G=T,R=T)}{P(G=T)} = \frac{\sum_{S \in \{T,F\}} P(G=T,S,R=T)}{\sum_{S,\,R \in \{T,F\}} P(G=T,S,R)}$$

That is:

$$P(R=T \mid G=T) = \frac{\sum_{S \in \{T,F\}} P(G=T \mid S,R=T)P(S \mid R=T)P(R=T)}{\sum_{S,\,R \in \{T,F\}} P(G=T \mid S,R)P(S \mid R)P(R)}$$

and so:

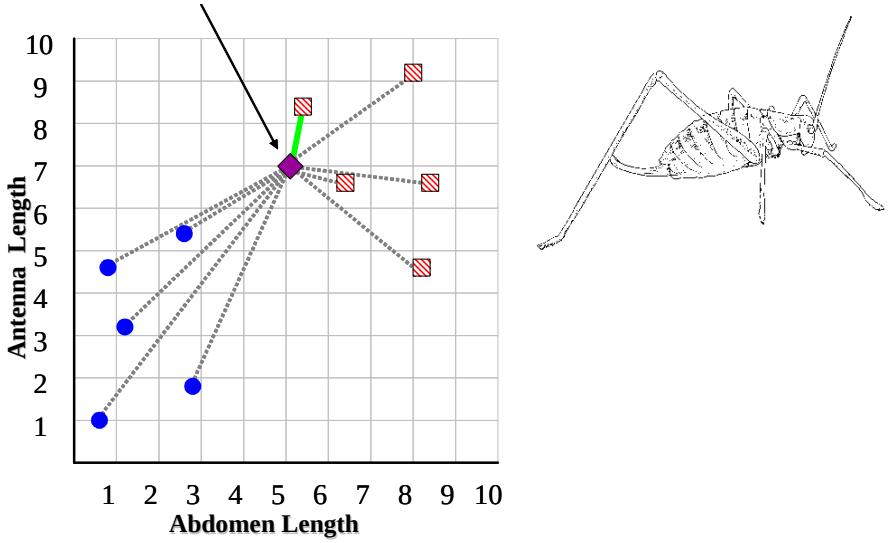
Note that $P(R=F \mid G=T) = 1 - 35.8\% = 64.2\%$, thus it is more likely that it is not raining, given the grass is wet.

### 5.4.5.1 Estimating conditional probabilities

Calculating conditional probabilities from Bayesian Belief Networks by enumerating all matching entries in the joint distribution (like has been illustrated in Example 4) can be expensive. Thus it may be easier to **sample** from the joint distribution to **estimate** a conditional probability e.g. $P(E_1 \mid E_2)$ would be based on the ratio of $N_s / N_c$ ($N_s$ - number of samples in which $E_1 \cap E_2$, $N_c$ - number of samples in which $E_2$).

## 5.5 Nearest Neighbour Classifier

The **nearest neighbour classifier** is a supervised learning algorithm where the result of a new query is classified based on the class of its nearest neighbour. The purpose of this algorithm is to classify a new object based on attributes and training samples. The classifiers do not use any model to fit and are only based on memory.




So how do we measure a new object's neighbours? If the variables are quantitative the **Euclidean distance** is the most commonly used **distance measure**. If the variables are qualitative, then **similarity measures** are used. (See Section 3.2).

The nearest neighbour algorithm is **sensitive to outliers** in the data, see Figure 5.5A.

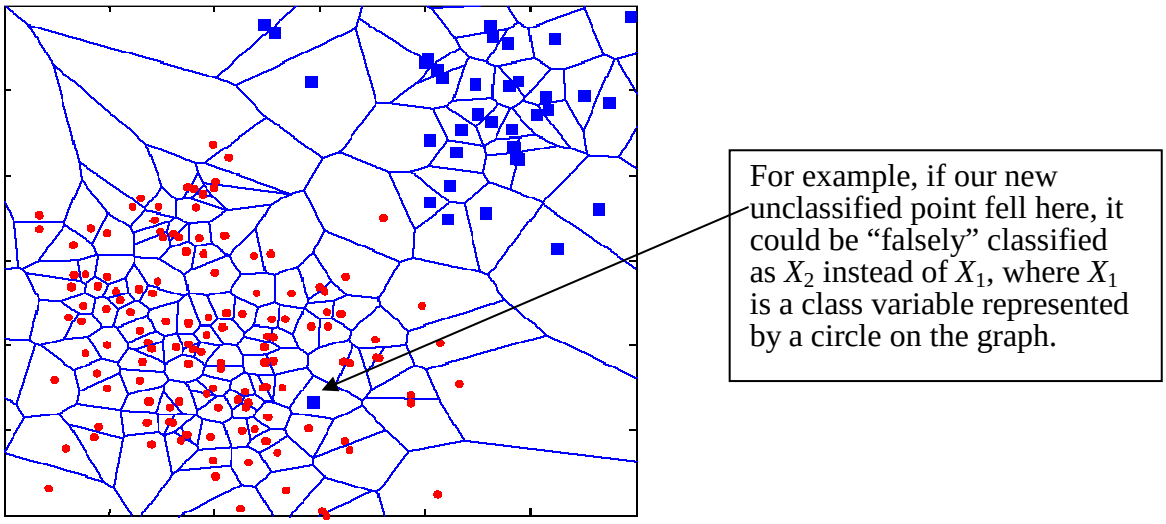



FIGURE 5.5A: MISCLASSIFICATION POSSIBLE FROM USING NEAREST NEIGHBOUR APPROACH

The nearest neighbour algorithm can be generalised to the *k*- Nearest Neighbour (*k*-NN) algorithm. It works on the minimum distance from the query to the training samples to determine the *k*-nearest neighbours. After we gather nearest neighbours, we take simple majority (*majority vote*) of these *k*-neighbours to be the prediction of the query instance (see Figure 5.5B). (*k* is typically an odd number). Generalising the nearest neighbour algorithm to *k*-nearest neighbours helps deal with the outliers.

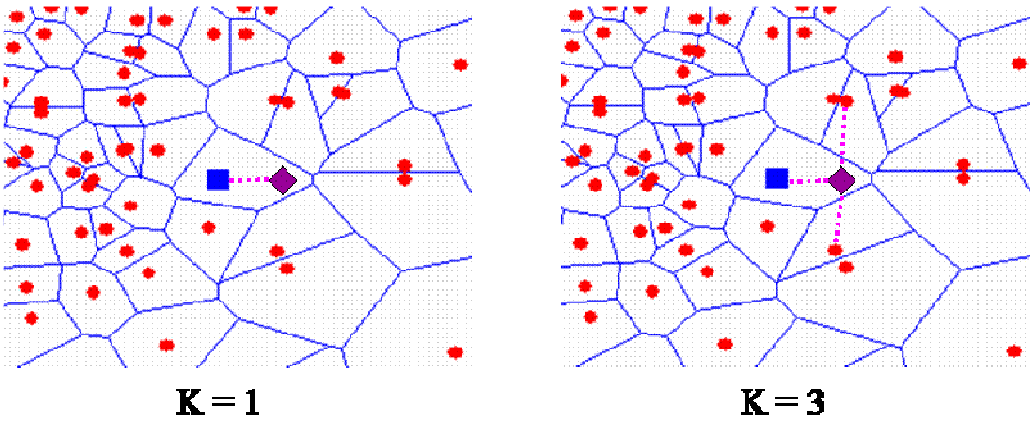


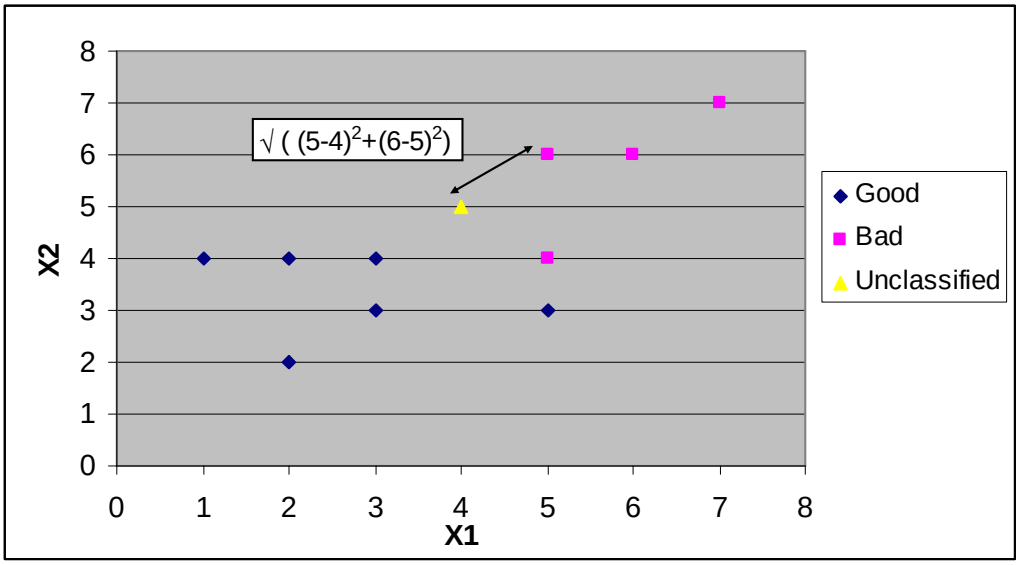

FIGURE 5.5B: SOLUTION IS TO GENERALISE THE NEIGHBOUREST NEIGHBOUR APPROACH TO A K-NEAREST NEIGHBOUR CLASSIFICATION.

**Example 4**

This is a very simple example to illustrate the use of the *k*-NN algorithm for classification. The data refers to a survey carried out to gauge people's opinions regarding whether a special paper tissue is good or bad. Measurements were taken for two variables of the paper: *acid durability* (seconds) and *strength* (kg/ square metre). In this illustrative example we have just 10 samples and our objective is to classify the 11$^{th}$ sample:

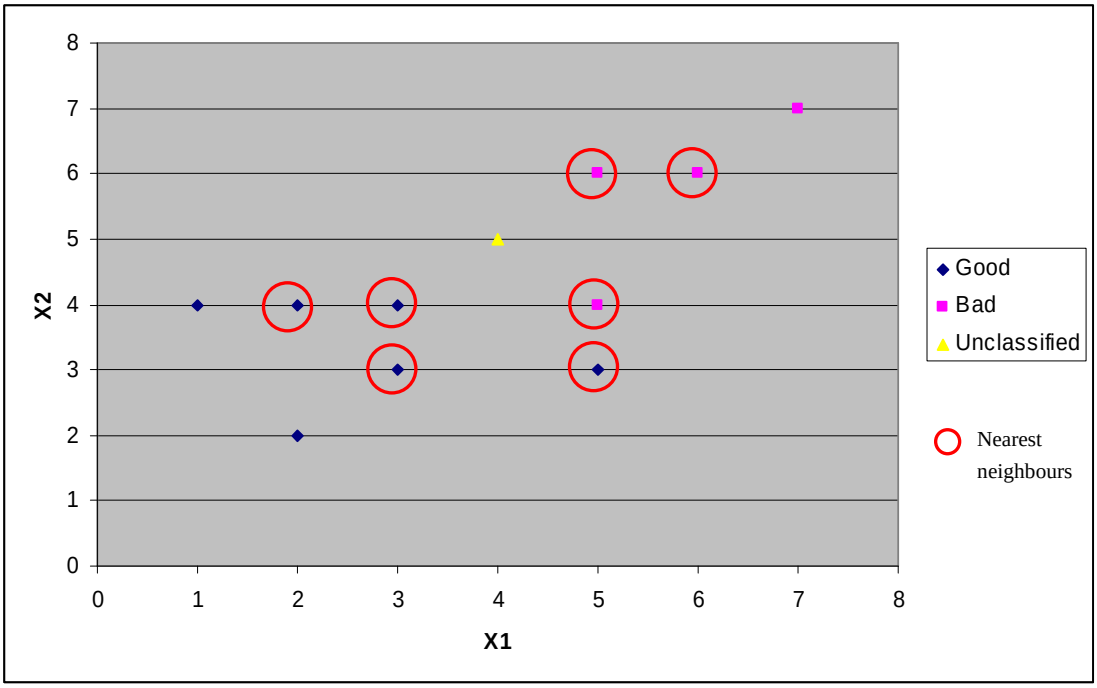
| X1 (acid durability) | X2 (strength) | Y (classification) |
|---|---|---|
| 3 | 4 | Good |
| 2 | 4 | Good |
| 2 | 2 | Good |
| 3 | 3 | Good |
| 1 | 4 | Good |
| 5 | 3 | Good |
| 7 | 7 | Bad |
| 5 | 4 | Bad |
| 6 | 6 | Bad |
| 5 | 6 | Bad |
| 4 | 5 | ?? |

Graphically these points look like:



We calculate the Euclidean distance for each possible neighbour.

| X1 (acid durability) | X2 (strength) | Y (classification) | Euclidean distance | Rank |
|---|---|---|---|---|
| 3 | 4 | Good | 1.414214 | 1 |
| 2 | 4 | Good | 2.236068 | 4 |
| 2 | 2 | Good | 3.605551 | 9 |
| 3 | 3 | Good | 2.236068 | 4 |
| 1 | 4 | Good | 3.162278 | 8 |
| 5 | 3 | Good | 2.236068 | 4 |
| 7 | 7 | Bad | 3.605551 | 9 |
| 5 | 4 | Bad | 1.414214 | 1 |
| 6 | 6 | Bad | 2.236068 | 4 |
| 5 | 6 | Bad | 1.414214 | 1 |
| 4 | 5 | ?? | | |




Letting $k = 7$, we consider the 7 nearest neighbours. Of the 7 nearest neighbours, 4 are classified as good and 3 are classified as bad.

## 5.5.1 Possible problems of Nearest Neighbour algorithms

**Advantages/Disadvantages of Nearest Neighbour Algorithms**

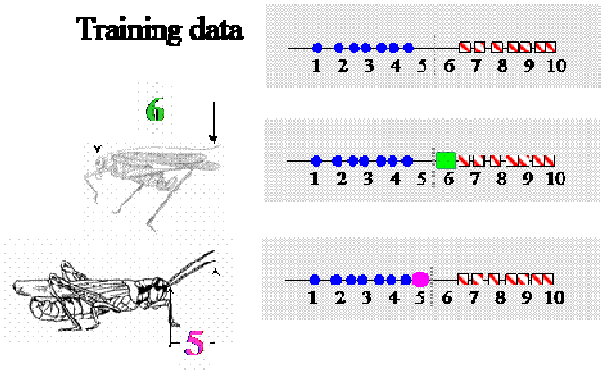
- Advantages:
  - Simple to implement
  - Handles correlated features
  - Defined for any distance measure
- Disadvantages:
  - Very sensitive to irrelevant features.

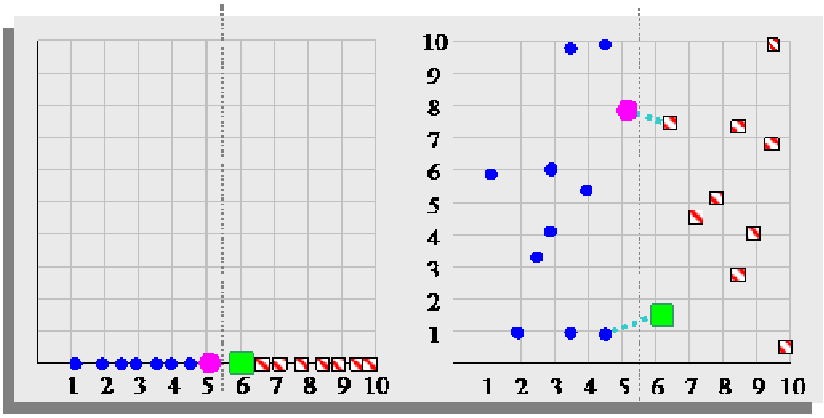
- Slow classification time for large datasets
- Sensitive to scale of measurement

- *Sensitive to irrelevant features*

Suppose the following is true, if an insect's antenna is longer than 5.5 it is a **Katydid**, otherwise it is a **Grasshopper**.

Using just the antenna length we get perfect classification!



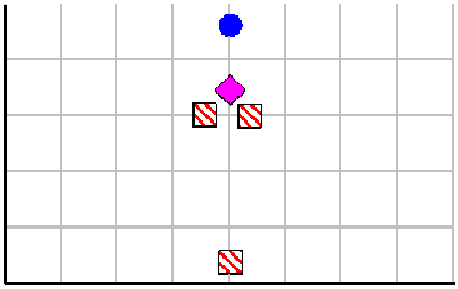

Suppose however, we add in an **irrelevant** feature, for example the insect's mass.

Using both the antenna length and the insect's mass with the 1-NN algorithm we get the wrong classification!

How do we mitigate the nearest neighbour algorithm's sensitivity to irrelevant features?

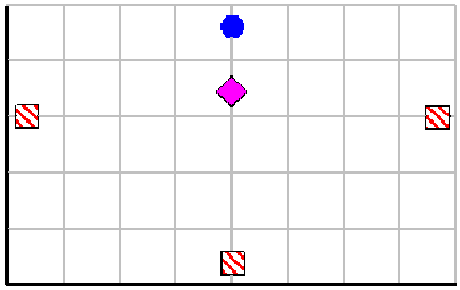
- Use more training instances
- Ask an expert what features are relevant to the task
- Use statistical tests to try to determine which features are useful

- *Sensitive to scale of measurement*



X axis measured in cm
Y axis measured in pounds

The nearest neighbour to the (unclassified) diamond is a stripped square



X axis measured in mm
Y axis measured in pounds

The nearest neighbour to the (unclassified) diamond is the circle

One possible solution is to standardise the variables before the classification process.

Wrong classification due to presence of many irrelevant attributes is often termed as the **curse of dimensionality**. There are a number of different possible approaches which may help classification accuracy including:

Approach 1: Associate weights with the attributes

- Assign random weights
- Calculate the classification error
- Adjust the weights according to the error
- Repeat till acceptable level of accuracy is reached

Approach 2: Backward Elimination

- Starts with the full set of features and greedily removes the one that most improves performance, or degrades performance slightly

### 5.6 Binary Logistic Regression

Binary logistic regression deals with one binary dependent ('target') variable $Y$=0 or 1, and $p$ independent variables $X_j$ which may be continuous, binary or qualitative (for which the indicators lead to the case of a binary variable). In the following notation the $p$ variables $X_j$ are grouped in a vector $\mathbf{X}=(X_1, X_2, \ldots, X_p)$.

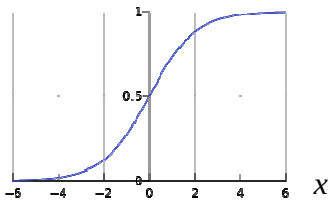
The aim of logistic regression is therefore the same as in all regressions, namely to model the conditional expectation $E(Y|\mathbf{X})$. That is, we wish to know the mean of $Y$ for any value of $\mathbf{X}$. For a value $Y$ equal to 0 or 1 (Bernoulli distribution), this mean is the probability that $Y$=1. Thus we have

$$E(Y|\mathbf{X}) = \Pr(Y=1|\mathbf{X})$$

In linear regression, with the model:

$$E(Y|\mathbf{X}) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots \beta_p X_p,$$

the term $\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots \beta_p X_p$ is unbounded and thus is inappropriate for modelling $\Pr(Y=1|\mathbf{X})$ - which should be bounded in the interval [0,1]. Often the shape of this probability takes a *S*-shaped curve.

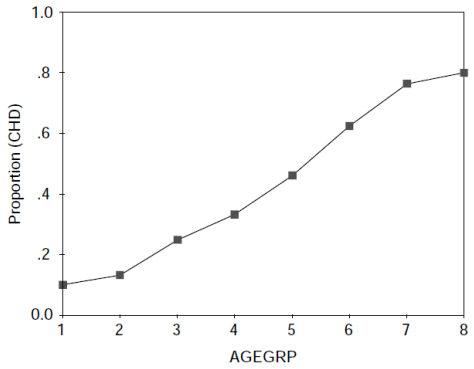



S-shaped curve

For example, consider the following data from 100 patients, whose ages ranged from 20 to 69 years - the occurrence (CHD=1) or nonoccurrence (CHD=0) of coronary heart disease was noted in them.

| Age class | No. of patients | CHD = 0 | CHD = 1 | Proportion of CHD = 1 |
|---|---|---|---|---|
| 20–29 | 10 | 9 | 1 | 0.10 |
| 30–34 | 15 | 13 | 2 | 0.13 |
| 35–39 | 12 | 9 | 3 | 0.25 |
| 40–44 | 15 | 10 | 5 | 0.33 |
| 45–49 | 13 | 7 | 6 | 0.46 |
| 50–54 | 8 | 3 | 5 | 0.63 |
| 55–59 | 17 | 4 | 13 | 0.76 |
| 60–69 | 10 | 2 | 8 | 0.80 |
| TOTAL | 100 | 57 | 43 | 0.43 |

giving a S-shaped curve:




In logistic regression analysis, we write $\pi(\mathbf{X}) = \Pr(Y=1|\mathbf{X})$ in the form:

$$\pi(\mathbf{X}) = \frac{e^{\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots\beta_p X_p}}{1+e^{\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots\beta_p X_p}} = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots\beta_p X_p)}}$$

or equivalently:

The function $f(p) = \log\left(\frac{p}{1-p}\right)$ is called the *logit*.

For each data point, $\mathbf{X} = \mathbf{x}_i$ and $Y = y_i$. The probability of that class was either $\pi$ if $y_i = 1$, or $1-\pi$ if $y_i = 0$.The likelihood of the logistic regression model is therefore:

$$L = \prod_{i=1}^{n}\left[\pi(\mathbf{x}_i)\right]^{y_i}\left[1-\pi(\mathbf{x}_i)\right]^{1-y_i}$$

The log-likelihood is:

$$\begin{aligned}\ell = \log L &= \sum_{i=1}^{n}\left[y_i \log\left(\pi(\mathbf{x}_i)\right)+(1-y_i)\log\left(1-\pi(\mathbf{x}_i)\right)\right]\\ &= \sum_{i=1}^{n}\log\left(1-\pi(\mathbf{x}_i)\right)+\sum_{i=1}^{n} y_i \log\left(\frac{\pi(\mathbf{x}_i)}{(1-\pi(\mathbf{x}_i))}\right)\\ &= \sum_{i=1}^{n}-\log\left(1+e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\ldots\beta_p x_{ip}}\right)+\sum_{i=1}^{n} y_i\left[\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\ldots\beta_p x_{ip}\right]\end{aligned}$$

In maximum likelihood estimation, partial derivatives of this with respect to each of the unknown parameters are set equal to zero.

$$\frac{\partial \ell}{\partial \beta_j} = 0 \text{ for } j = 0,1,\ldots, p$$

<u>Odds Ratios:</u>

The odds ratio of an independent variable measures the variation of the ratio of the probabilities of the occurrence of the event $Y$=1 against $Y$=0, when a variable $X_j$ changes from $X_j$ to $X_j$ +1 (all other variables remain the same). In this case, logit($\pi(\mathbf{x})$) increases by the coefficient $\beta_j$, and the odds $\frac{\pi(\mathbf{x})}{1-\pi(\mathbf{x})}$ are multiplied by $\exp(\beta_j)$. This is written has

If $X_j$ is a **binary** variable, then the odds ratio formula becomes

$$OR = \frac{\pi\left(\mathbf{X}\middle|X_j=1\right)\Big/\left(1-\pi\left(\mathbf{X}\middle|X_j=1\right)\right)}{\pi\left(\mathbf{X}\middle|X_j=0\right)\Big/\left(1-\pi\left(\mathbf{X}\middle|X_j=0\right)\right)} = \frac{\Pr\left(Y=1\middle|\mathbf{X},X_j=1\right)\Big/\Pr\left(Y=0\middle|\mathbf{X},X_j=1\right)}{\Pr\left(Y=1\middle|\mathbf{X},X_j=0\right)\Big/\Pr\left(Y=0\middle|\mathbf{X},X_j=0\right)} = e^{\beta_j}$$

Confidence intervals for odds ratios can be determined based on the standard error of each $\beta_j$ parameter estimate. For example a 95% confidence interval for $\beta_j$ is $\hat{\beta}_j \pm 1.96 \text{ s.e.}\left(\hat{\beta}_j\right)$. Thus the limits of the 95% confidence interval for OR are

**Example 5**

Consider a study of the analgesic effects of treatments on elderly patients with neuralgia. Two test treatments and a placebo are compared. The response variable is whether the patient reported pain or not. Researchers recorded the age and gender of 60 patients and the duration of complaint before the treatment began.

The data set Neuralgia contains five variables: Treatment, Sex, Age, Duration, and Pain. The last variable, Pain, is the response variable. A specification of Pain=Yes indicates there was pain, and

Pain=No indicates no pain. The variable Treatment is a categorical variable with three levels: A and B represent the two test treatments, and P represents the placebo treatment. The gender of the patients is given by the categorical variable Sex. The variable Age is the age of the patients, in years, when treatment began. The duration of complaint, in months, before the treatment began is given by the variable Duration.
Using the following SAS code:

```
   ods graphics on;
   proc logistic data=Neuralgia plots(only)=(oddsratio(range=clip) effect)
outest=betas covout;
      class Treatment Sex /param=ref;
      model Pain= Treatment Sex age;
      oddsratio Treatment;
      oddsratio Sex;
      oddsratio age;
      contrast 'Pairwise' Treatment 1  0,
                          Treatment 0  1,
                          Treatment 1 -1 / estimate=exp;
      contrast 'Female vs Male' Sex 1 / estimate=exp;
        output out=pred p=phat lower=lcl upper=ucl
             predprob=(individual crossvalidate);
   run;

   ods graphics off;
```

gives the following output:

*Probability modeled is Pain='No'.*

**Analysis of Maximum Likelihood Estimates**

| Parameter | | DF | Estimate | Standard Error | Wald Chi-Square | Pr > ChiSq |
|---|---|---|---|---|---|---|
| Intercept | | 1 | 15.8669 | 6.4056 | 6.1357 | 0.0132 |
| Treatment | A | 1 | 3.1790 | 1.0135 | 9.8375 | 0.0017 |
| Treatment | B | 1 | 3.7264 | 1.1339 | 10.8006 | 0.0010 |
| Sex | F | 1 | 1.8235 | 0.7920 | 5.3013 | 0.0213 |
| Age | | 1 | -0.2650 | 0.0959 | 7.6314 | 0.0057 |

Here the model is:

where

$$X_1 = \begin{cases} 1 & \text{if treatment=A} \\ 0 & \text{if treatment} \neq \text{A} \end{cases}, X_2 = \begin{cases} 1 & \text{if treatment=B} \\ 0 & \text{if treatment} \neq \text{B} \end{cases}, X_3 = \begin{cases} 1 & \text{if sex=F} \\ 0 & \text{if sex} \neq \text{F} \end{cases}, X_4 = \text{age of individual}$$

Estimates for $\beta_o,\ldots\beta_4$ are in the column "Estimate". Notice that using Wald's test, each of these $\beta$ parameters are significantly different from zero.

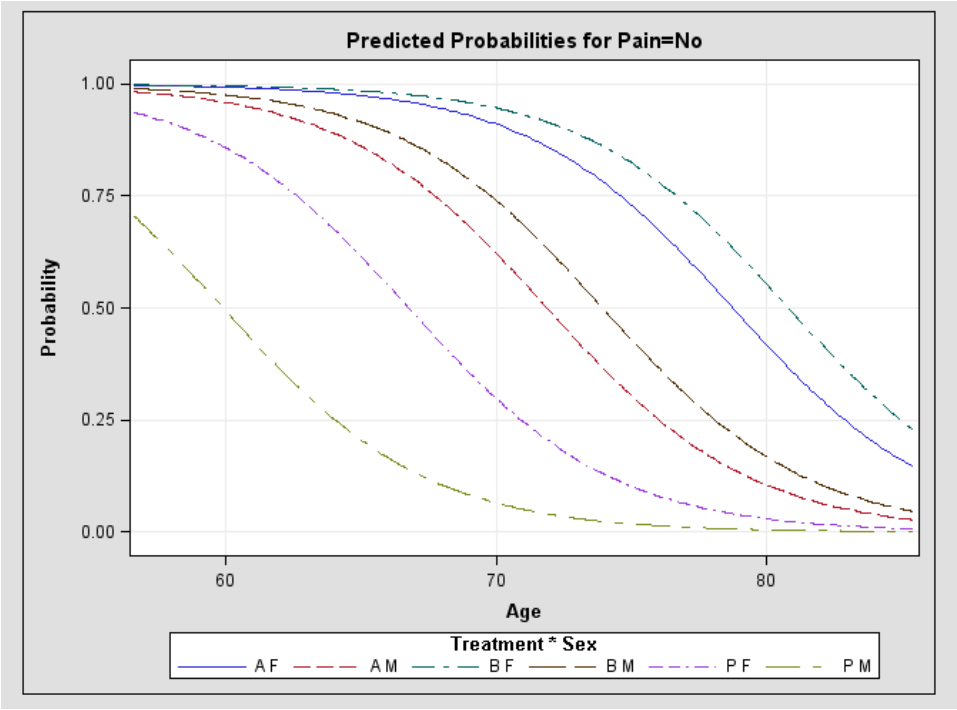
Based on this model the prediction for a 60 year old female on treatment A would be:

While the prediction for a 65 year old male on treatment P would be:

Similarly, the prediction for a 70 year old male on treatment B would be:

$$\Pr\left(\text{Pain="No"}\middle|\mathbf{X}\right)=\frac{1}{1+\exp\left(-\left(\beta_0+\beta_2+70\beta_4\right)\right)}=\frac{1}{1+\exp\left(-\left(15.8669+3.7264+70\times\left(-0.2650\right)\right)\right)}=0.740$$

The output from SAS gives the predictions based on the various sex/treatment combinations.



The odds ratios for comparing to the baseline treatment (treatment P) and comparing female versus males are straightforward to determine. For example

And the 95% confidence limits are

The table below shows the other OR results:

| Wald Confidence Interval for Odds Ratios | | | |
|---|---|---|---|
| **Label** | **Estimate** | **95% Confidence Limits** | |
| Treatment A vs B | 0.578 | 0.093 | 3.589 |
| Treatment A vs P | 24.022 | 3.295 | 175.121 |
| Treatment B vs P | 41.528 | 4.500 | 383.262 |
| Sex F vs M | 6.194 | 1.312 | 29.248 |
| Age | 0.767 | 0.636 | 0.926 |

Notice that the comparison of treatment A and treatment B requires:

$$OR(\text{comparing treatment A vs B}) = \frac{\text{odds treatment A}}{\text{odds treatment B}} = \frac{\dfrac{\text{odds treatment A}}{\text{odds treatment P}}}{\dfrac{\text{odds treatment B}}{\text{odds treatment P}}} = \frac{e^{\beta_1}}{e^{\beta_2}} = e^{\beta_1-\beta_2} = e^{(3.179-3.7264)} = 0.578$$

The confidence interval of $e^{\beta_1-\beta_2}$ depends on the variance and covariance of the parameter estimates (output in the data set `betas` - see below).

VIEWTABLE: Parameter Estimates and Covariance Matrix

| | Link function | Type of Statistics | Convergence Status | Row Names for Parameter Estimates and Covariance Matrix | Intercept: Pain=No | Treatment A | Treatment B | Sex F | Age | Model Log Likelihood |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | LOGIT | PARMS | 0 Converged | Pain | 15.86689093 | 3.1789636677 | 3.7263780036 | 1.8235260499 | -0.264959606 | -24.3837144 |
| 2 | LOGIT | COV | 0 Converged | Intercept | 41.031551587 | 2.0247966882 | 3.7418953347 | 0.4157805669 | -0.609455441 | -24.3837144 |
| 3 | LOGIT | COV | 0 Converged | TreatmentA | 2.0247966882 | 1.0272762992 | 0.7228283407 | 0.3176108501 | -0.038680646 | -24.3837144 |
| 4 | LOGIT | COV | 0 Converged | TreatmentB | 3.7418953347 | 0.7228283407 | 1.2856558125 | 0.3225902404 | -0.064273714 | -24.3837144 |
| 5 | LOGIT | COV | 0 Converged | SexF | 0.4157805669 | 0.3176108501 | 0.3225902404 | 0.6272483036 | -0.012769365 | -24.3837144 |
| 6 | LOGIT | COV | 0 Converged | Age | -0.609455441 | -0.038680646 | -0.064273714 | -0.012769365 | 0.0091993312 | -24.3837144 |

Since

the confidence limits of this odds ratio are:

The results from the `contrast` statements agree with the information provided in the odds ratio table:

| Contrast Rows Estimation and Testing Results | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Contrast | Type | Row | Estimate | Standard Error | Alpha | Confidence Limits | Wald Chi-Square | Pr > ChiSq |
| Pairwise | EXP | 1 | 24.0218 | 24.3473 | 0.05 | 3.2951 175.1 | 9.8375 | 0.0017 |
| Pairwise | EXP | 2 | 41.5284 | 47.0877 | 0.05 | 4.4998 383.3 | 10.8006 | 0.0010 |
| Pairwise | EXP | 3 | 0.5784 | 0.5387 | 0.05 | 0.0932 3.5889 | 0.3455 | 0.5567 |
| Female vs Male | EXP | 1 | 6.1937 | 4.9053 | 0.05 | 1.3116 29.2476 | 5.3013 | 0.0213 |

For example, the confidence limits in row 1 of the 'pairwise' contrast correspond to the confidence limits of the odds ratio comparing treatment A versus treatment P. Similarly, the confidence limits in row 3 of the 'pairwise' contrast correspond to the confidence limits of the odds ratio comparing treatment A versus treatment B.

Notice however the corresponding Wald's test statistic for $\beta_1 - \beta_2$ is now also given:

Predicting the Classification of Observations

In order to give a prediction of whether pain occurs or not in an individual, the cut-off probability of 50% could be used.

If this is the case the 60 year old female on treatment A could be classified as Pain= "No" since $\Pr(\text{Pain="No"}|\mathbf{X}) = 0.993 > 0.5$. Similarly:

One can then examine how well the predictions compared to the actual classifications of individuals in the training data:

| Frequency | Table of Pain by pred_pain | | |
|---|---|---|---|
| | pred_pain | | |
| Pain | No | Yes | Total |
| No | 30 | 5 | 35 |
| Yes | 4 | 21 | 25 |
| Total | 34 | 26 | 60 |

5 persons who indicated that they suffered no pain are actually incorrectly predicted as suffering pain by the model

4 persons who indicated that they suffered pain are actually incorrectly predicted as suffering no pain by the model

The **sensitivity of the model** using the probability cut-off of 50% is:

$$\text{sensitivity} = \Pr\left(\text{model predicts the positive outcome} \middle| \text{positive outcome occurs}\right) = \frac{TP}{TP + FN}$$

The **specificity of the model** using the probability cut-off of 50% is:

$$\text{specificity} = \Pr\left(\text{model predicts the negative outcome} \middle| \text{negative outcome occurs}\right) = \frac{TN}{TN + FP}$$

Receiver Operating Characteristic (ROC) Curves

A ROC curve is a graphical plot which illustrates the performance of a binary classifier system as its discrimination threshold is varied. It is created by plotting sensitivity vs. (1-specificity) at various threshold settings. For example with the logistic regression classifier, the probability cut-off could be varied from 50% to as high as 100% and as low as 0%. With a cut-off near 100% the probabilities predicted for individuals need to be very high if a person is to be classified as Pain="No" (e.g. if the cut-off was 95%, we see that the 70 year old male on treatment B (Pr=0.74) will be classified as Pain="Yes"), thus it is likely many with actually no pain will be given the classification of having pain i.e. the sensitivity may be low. Similarly, since those with pain are likely to be a given this classification the specification may be high (or (1-specificity) may be low). In contrast, when the cut-off is near 0%, more individuals will be classified as Pain="No" (e.g. with a cut-off of 5%, even the 65 year old male on treatment P (Pr=0.205) would be classified as having no pain). Thus it is likely that individuals with pain will be given the classification of having no pain i.e. the specificity may be low or (1-specificity) may be high. Similarly, since those with no pain are likely to be a given this classification the sensitivity may be high

The sensitivity and specificity can be calculated for any logistic regression model at different cut-off thresholds in SAS by adding the following statement within the **`proc logistic`** statements:

```
score out=scores outroc=rocs;
```

The dataset `rocs` gives the sensitivity and specificity at different cut-off thresholds.

VIEWTABLE: Receiver Operating Characteristics

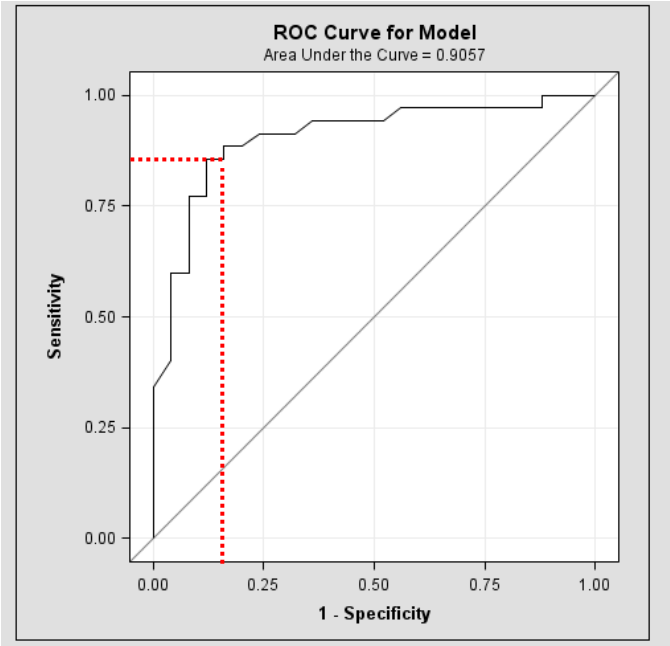
| | Probability Level | No. of Correctly Predicted Events | No. of Correctly Predicted Nonevents | No. of Nonevents Predicted as Events | No. of Events Predicted as Nonevents | Sensitivity | 1 - Specificity |
|---|---|---|---|---|---|---|---|
| 7 | 0.9582969006 | 10 | 25 | 0 | 25 | 0.2857142857 | 0 |
| 8 | 0.9575920932 | 11 | 25 | 0 | 24 | 0.3142857143 | 0 |
| 9 | 0.9320338311 | 12 | 25 | 0 | 23 | 0.3428571429 | 0 |

Notice that the values given for probability thresholds between [0.558166,0.496564) agree with those calculated at a cut-off of 50%.

VIEWTABLE: Receiver Operating Characteristics

| | Probability Level | No. of Correctly Predicted Events | No. of Correctly Predicted Nonevents | No. of Nonevents Predicted as Events | No. of Events Predicted as Nonevents | Sensitivity | 1 - Specificity |
|---|---|---|---|---|---|---|---|
| 7 | 0.9582969006 | 10 | 25 | 0 | 25 | 0.2857142857 | 0 |
| 8 | 0.9575920932 | 11 | 25 | 0 | 24 | 0.3142857143 | 0 |
| 9 | 0.9320338311 | 12 | 25 | 0 | 23 | 0.3428571429 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 25 | 0.6221495943 | 29 | 22 | 3 | 6 | 0.8285714286 | 0.12 |
| 26 | 0.6149203649 | 30 | 22 | 3 | 5 | 0.8571428571 | 0.12 |
| 27 | 0.5581661073 | 30 | 21 | 4 | 5 | 0.8571428571 | 0.16 |
| 28 | 0.4965645834 | 31 | 21 | 4 | 4 | 0.8857142857 | 0.16 |
| 29 | 0.49232925 | 31 | 20 | 5 | 4 | 0.8857142857 | 0.2 |
| 30 | 0.4845357844 | 32 | 19 | 6 | 3 | 0.9142857143 | 0.24 |

The ROC curve can be plotted using `plots=roc` within the **`proc logistic`** statement line.




The **area under the ROC curve** (often known as **AUC**) is equal to the probability that a classifier will assign a higher score to a randomly chosen positive example than to a randomly chosen negative example. Values close to 1 are desirable. This measure can be used in model comparison.

### 5.7 Multinomial Regression

Logistic regression may be extended beyond the analysis of dichotomous variables to the analysis of nominal dependent variables with more than two categories. Several methods exists and the following outlines one such method.

One value of the dependent variable is designated the reference category, $Y = h_o$, and the probability of membership in other categories is compared with the probability of membership in the reference category. Suppose a dependent variable has $M$ possible categories.

The probability that $Y$ is equal to any value $h$ (other than $h_0$) is given by:

$$\Pr\left(Y = h \middle| \mathbf{X}\right) = \frac{e^{\beta_{ho} + \beta_{h1}X_1 + \beta_{h2}X_2 + \ldots + \beta_{hp}X_p}}{1 + \sum_{j=1}^{M-1} e^{\beta_{jo} + \beta_{j1}X_1 + \beta_{j2}X_2 + \ldots + \beta_{jp}X_p}}$$

$$h = 1, \ldots, M-1$$

The probability that $Y$ is equal to the reference category $h_0$ is given by:

It can be verified that the sum of these probabilities adds to 1:

$$\Pr\left(Y = h_0 \middle| \mathbf{X}\right) + \sum_{h=1}^{M-1} \Pr\left(Y = h \middle| \mathbf{X}\right) = 1$$

Notice that there are $\beta$ parameters for <u>each</u> possible category $h\left(h = 1, \ldots, M-1\right)$, therefore the number of parameters to be evaluated has increased by a factor of $M-1$ compared to those in logistic regression.

The reference category is usually that with the <u>largest sample</u> (to improve standard errors), unless a reference category is otherwise more applicable (e.g. comparing drugs to a placebo).

**<u>Example 6</u>**

In the US students entering high school make a choice between the type of program they will follow: a general program, a vocational program and an academic program. This choice might be influenced by their writing score and their social economic status (ses).

The data set `hsb` contains variables on 200 students. The outcome variable is `prog`, program type (1=general, 2=academic, 3=vocational). The predictor variables are social economic status, `ses`, (a three-level categorical variable (from 1=low to 3=high)) and writing score, `write`, a continuous variable.

Using the following SAS code:

```
proc logistic data = hsb outest = mlogit_param;
class prog (ref = "2") ses (ref = "1") / param = ref;
model prog = ses write / link = glogit;
SES3_general_vs_SES3_vocational: test SES3_1 - SES3_3;
run;
```

produces:

| Analysis of Maximum Likelihood Estimates | | | | | | | |
|---|---|---|---|---|---|---|---|
| Parameter | | PROG | DF | Estimate | Standard Error | Wald Chi-Square | Pr > ChiSq |
| Intercept | | 1 | 1 | 2.8522 | 1.1664 | 5.9790 | 0.0145 |
| Intercept | | 3 | 1 | 5.2182 | 1.1635 | 20.1128 | <.0001 |
| SES | 2 | 1 | 1 | -0.5333 | 0.4437 | 1.4444 | 0.2294 |
| SES | 2 | 3 | 1 | 0.2914 | 0.4764 | 0.3742 | 0.5407 |
| SES | 3 | 1 | 1 | -1.1628 | 0.5142 | 5.1137 | 0.0237 |
| SES | 3 | 3 | 1 | -0.9827 | 0.5956 | 2.7224 | 0.0989 |
| WRITE | | 1 | 1 | -0.0579 | 0.0214 | 7.3200 | 0.0068 |
| WRITE | | 3 | 1 | -0.1136 | 0.0222 | 26.1392 | <.0001 |

The model in this case is:

$$\Pr\left(Y=\text{"general"}\middle|\mathbf{X}\right)=\frac{e^{\beta_{1o}+\beta_{11}X_1+\beta_{12}X_2+\beta_{13}X_3}}{1+e^{\beta_{1o}+\beta_{11}X_1+\beta_{12}X_2+\beta_{13}X_3}+e^{\beta_{3o}+\beta_{31}X_1+\beta_{32}X_2+\beta_{33}X_3}}$$

$$\Pr\left(Y=\text{"vocational"}\middle|\mathbf{X}\right)=\frac{e^{\beta_{3o}+\beta_{31}X_1+\beta_{32}X_2+\beta_{33}X_3}}{1+e^{\beta_{1o}+\beta_{11}X_1+\beta_{12}X_2+\beta_{13}X_3}+e^{\beta_{3o}+\beta_{31}X_1+\beta_{32}X_2+\beta_{33}X_3}}$$

$$\Pr\left(Y=\text{"academic"}\middle|\mathbf{X}\right)=\frac{1}{1+e^{\beta_{1o}+\beta_{11}X_1+\beta_{12}X_2+\beta_{13}X_3}+e^{\beta_{3o}+\beta_{31}X_1+\beta_{32}X_2+\beta_{33}X_3}}$$

where:

$$X_1=\begin{cases}1 & \text{if ses=2}\\ 0 & \text{if ses}\neq 2\end{cases},\ X_2=\begin{cases}1 & \text{if ses=3}\\ 0 & \text{if ses}\neq 3\end{cases},\ X_3=\text{writing score}$$

Using the parameter estimates indicated:

$$\Pr\left(Y=\text{"general"}\middle|\mathbf{X}\right)=\frac{e^{2.8522-0.5333X_1-1.1628X_2-0.0579X_3}}{1+e^{2.8522-0.5333X_1-1.1628X_2-0.0579X_3}+e^{5.2182+0.2914X_1-0.9827X_2-0.1136X_3}}$$

$$\Pr\left(Y=\text{"vocational"}\middle|\mathbf{X}\right)=\frac{e^{5.2182+0.2914X_1-0.9827X_2-0.1136X_3}}{1+e^{2.8522-0.5333X_1-1.1628X_2-0.0579X_3}+e^{5.2182+0.2914X_1-0.9827X_2-0.1136X_3}}$$

$$\Pr\left(Y=\text{"academic"}\middle|\mathbf{X}\right)=\frac{1}{1+e^{2.8522-0.5333X_1-1.1628X_2-0.0579X_3}+e^{5.2182+0.2914X_1-0.9827X_2-0.1136X_3}}$$

Thus:

Notice that:

- A one-unit increase in the variable **write** is associated with a **0.0579 decrease** in the relative log odds of being in general program vs. academic program.
- A one-unit increase in the variable **write** is associated with a **0.1136 decrease** in the relative log odds of being in vocation program vs. academic program.
- The relative log odds of being in general program vs. in academic program will **decrease by 1.1628** if moving from the lowest level of **ses** (**ses**==1) to the highest level of **ses** (**ses**==3).

The odds ratios in the table below compare e.g.

$$OR\left(\text{SES=2 vs SES=1}\middle|\text{"general" vs "academic"}\right)=\frac{\text{odds}\left(SES=2\middle|\text{"general" vs "academic"}\right)}{\text{odds}\left(SES=1\middle|\text{"general" vs "academic"}\right)}=e^{-0.5333}=0.587$$

with confidence limits:

NB: Since this confidence interval contains 1, SES=2 and SES=1 are not significantly different in influencing the “general” vs. “academic” comparison.

Also:

$$OR\left(\text{write=}x\text{+1 vs write=}x\middle|\text{"general" vs "academic"}\right)=\frac{\text{odds}\left(\text{write=}x+1\middle|\text{"general" vs "academic"}\right)}{\text{odds}\left(\text{write=}x\middle|\text{"general" vs "academic"}\right)}=e^{-0.0579}=0.944$$

This indicates that for 2 students with the same SES level and different writing scores, the one with the higher writing score is **more likely** to be in the academic program than the general program (or equivalently **less likely** to be in the general program than the academic program).

**Odds Ratio Estimates**

| Effect | PROG | Point Estimate | 95% Wald Confidence Limits | |
|---|---|---|---|---|
| SES 2 vs 1 | 1 | 0.587 | 0.246 | 1.400 |
| SES 2 vs 1 | 3 | 1.338 | 0.526 | 3.404 |
| SES 3 vs 1 | 1 | 0.313 | 0.114 | 0.856 |
| SES 3 vs 1 | 3 | 0.374 | 0.116 | 1.203 |
| WRITE | 1 | 0.944 | 0.905 | 0.984 |
| WRITE | 3 | 0.893 | 0.855 | 0.932 |

Consider a student with ses=2 and a writing score=33:

That is:

Thus for this student their most likely classification would be **"vocational".** Also, for a student with ses=1 and writing score=47:

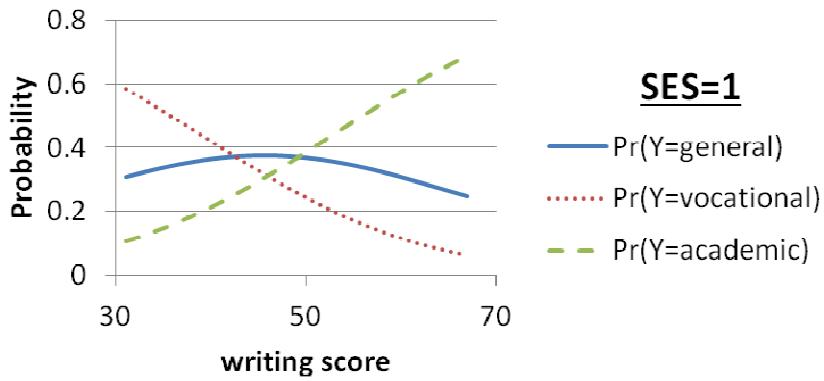
$$\Pr\left(Y=\text{"general"}\middle|\mathbf{X}\right)=\frac{1.138343}{1+1.138343+0.886034}=0.376$$

$$\Pr\left(Y=\text{"vocational"}\middle|\mathbf{X}\right)=\frac{0.886034}{1+1.138343+0.886034}=0.293$$

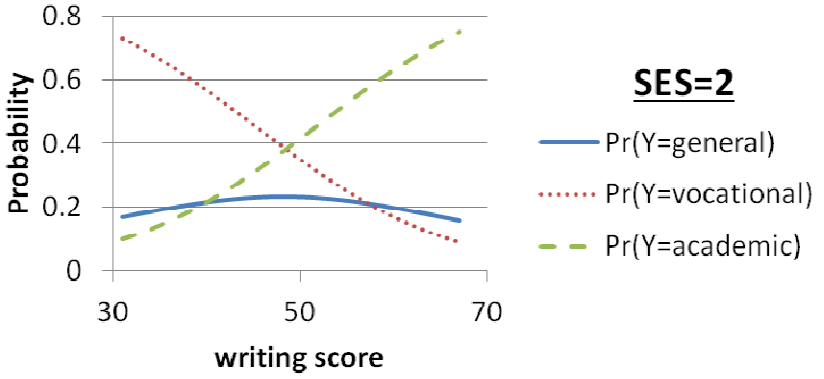
$$\Pr\left(Y=\text{"academic"}\middle|\mathbf{X}\right)=\frac{1}{1+1.138343+0.886034}=0.331$$

Thus for this student their most likely classification would be **"general".**

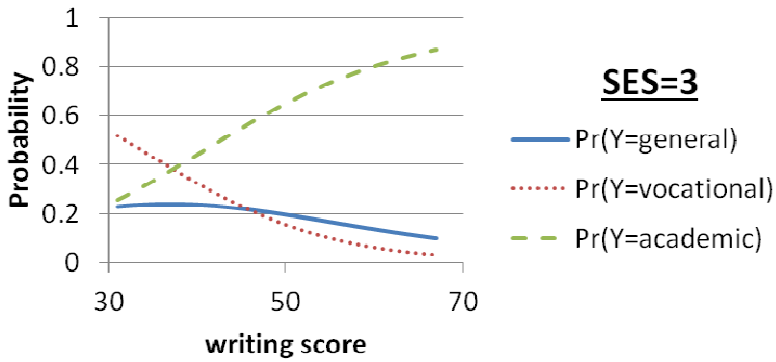
The results for this model across all writing scores are:




That is for students with ses=1 and a writing score below 43, their classification would be "vocational". However, for students with ses=1 and a writing score between 43 and 49, their classification would be "general". While for students with ses=1 and a writing score over 49, their classification would be "academic".




For students with ses=2 and a writing score below 49, their classification would be "vocational". For higher writing scores, their classification would be "academic".




For students with ses=3 and a writing score below 38, their classification would be "vocational". For higher writing scores, their classification would be "academic".

The `test` statement allows the significance of other linear combinations to be investigated. For example to test whether the null hypothesis $H_0 : \beta_{12} - \beta_{32} = 0$, a Wald's test was performed. This requires the variance and covariance terms, obtained using `covout`.

| | Type of Statistics | Convergence Status | Row Names for Parameter Estimates and Covariance Matrix | SES 3: PROG=1 | SES 3: PROG=3 | writing score: PROG=1 | writing score: PROG=3 | Model Log Likelihood |
|---|---|---|---|---|---|---|---|---|
| 1 | PARMS | 0 Converged | PROG | -1.162831959 | -0.982669691 | -0.057928408 | -0.113602625 | -179.9817262 |
| 2 | COV | 0 Converged | Intercept_1 | -0.059812872 | -0.067721851 | -0.023832352 | -0.010999203 | -179.9817262 |
| 3 | COV | 0 Converged | Intercept_3 | -0.038073931 | -0.128639745 | -0.011247614 | -0.024253496 | -179.9817262 |
| 4 | COV | 0 Converged | SES2_1 | 0.1225343344 | 0.064128933 | -0.00052163 | -0.000371912 | -179.9817262 |
| 5 | COV | 0 Converged | SES2_3 | 0.0636261289 | 0.1601998491 | 0.0000744261 | -0.001062758 | -179.9817262 |
| 6 | COV | 0 Converged | SES3_1 | 0.2644216572 | 0.0993138521 | -0.001182268 | -0.000461216 | -179.9817262 |
| 7 | COV | 0 Converged | SES3_3 | 0.0993138521 | 0.354699892 | 0.0000843924 | -0.00062836 | -179.9817262 |
| 8 | COV | 0 Converged | WRITE_1 | -0.001182268 | 0.0000843924 | 0.0004584286 | 0.0002039003 | -179.9817262 |
| 9 | COV | 0 Converged | WRITE_3 | -0.000461216 | -0.00062836 | 0.0002039003 | 0.0004937238 | -179.9817262 |

The test statistic is:

in agreement with the results output from SAS:

| Linear Hypotheses Testing Results | | | |
|---|---|---|---|
| Label | Wald Chi-Square | DF | Pr > ChiSq |
| SES3_general_vs_SES3_vocational | 0.0772 | 1 | 0.7811 |

**Independence from Irrelevant Alternatives (IIA)**

Notice that the ratio of the choice probabilities for alternatives $j$ and $h$, is

(with $\beta$ s equal to zero for the reference category).

Note that the ratio of the choice probabilities for alternatives $j$ and $h$ does not depend on any alternatives other than $j$ and $h$. The IIA property can be restrictive from the point of view of choice behaviour. Models that display the IIA property predict that a change in the attributes of one alternative changes the probabilities of the other alternatives proportionately such that the ratios of probabilities remain constant.

Consider the Red Bus/Blue Bus example. Commuters face a decision between car and red bus. Suppose that a commuter chooses between these two options with equal probability, 0.5, so that the odds ratio equals 1. Now suppose a third mode, a blue bus, is added. Assuming bus commuters do not care about the color of the bus, they are expected to choose between bus and car still with equal probability, so the probability of car is still 0.5, while the probabilities of each of the two bus types is 0.25. But IIA implies that this is not the case: for the odds ratio between car and red bus to be preserved, the new probabilities must be car 0.33; red bus 0.33; blue bus 0.33. In intuitive terms, the problem with the IIA axiom is that it leads to a failure to take account of the fact that red bus and blue bus are very similar, and are "perfect substitutes".

Random Utility Models

An alternative interpretation of data on individual choices is provided by the **random utility model**. For example suppose that $U^a$, $U^b$ and $U^c$ denote an individual's utility of three choices e.g. like the choice above between "general", "academic" and "vocational". The individual's observed preference allows one to know which utility is the largest, but does not reveal the unobservable utilities.

The above model can be expressed using a utility function. Notice that the denominator was the same when calculating $\Pr(Y = \text{"general"}|\mathbf{X})$, $\Pr(Y = \text{"vocational"}|\mathbf{X})$ and $\Pr(Y = \text{"academic"}|\mathbf{X})$ for the student with ses=2 and a writing score=33. Similarly the denominator was the same for $\Pr(Y = \text{"general"}|\mathbf{X})$, $\Pr(Y = \text{"vocational"}|\mathbf{X})$ and $\Pr(Y = \text{"academic"}|\mathbf{X})$ for the student with ses=1 and writing score=47 (but different to that of the other student).

In general, in order to identify the likely classification of an individual only the numerators need to be compared – the one that is largest is the most likely classification. That is in the example above, determine

$$\max\left(e^{2.8522-0.5333X_1-1.1628X_2-0.0579X_3}, e^{5.2182+0.2914X_1-0.9827X_2-0.1136X_3}, 1\right)$$

or equivalently:

$$\max\left(2.8522-0.5333X_1-1.1628X_2-0.0579X_3, 5.2182+0.2914X_1-0.9827X_2-0.1136X_3, 0\right).$$

That is multinomial regression can be expressed in the **linear** random utility model framework, where the utility a choice $h$ for individual $i$ is given by:

$$U^h = \beta_{ho} + \beta_{h1} X_{1,i} + \beta_{h2} X_{2,i} + \ldots + \beta_{hp} X_{p,i} + \varepsilon_{ij} \qquad (h = 1, \ldots, M-1)$$

$$U^{h_0} = 0 + \varepsilon_{ih_0}$$

where $\varepsilon_{ih}$ denote random error terms. Notice that a subscript $i$ has be added to the input variables to highlight the fact that these are individual specific characteristics.

The class observed for individual *i* is determined by which of these expected utilities is a maximum.

**NB**: Please note that multinomial regression can also be described as polytomous logistic regression for unordered categorical variables.

### 5.8 Conditional Logit Models

Suppose that data consist of choice-specific attributes instead of individual specific characteristics, or indeed attributes which vary for each individual across the *M* alternatives. That is, a set of inputs $X_{1,ih}, X_{2,ih}, \ldots, X_{p,ih}$ influence the choice of individual $i$ toward alternative $h$.

The term multinomial logit is often used to refer to the conditional logit model of McFadden (1974), in which:

That is, the utilities being compared for individual $i$ are:

$$U^h = \beta_1 X_{1,ih} + \beta_2 X_{2,ih} + \ldots + \beta_p X_{p,ih} + \varepsilon_{ih} \qquad (h = 1, \ldots, M)$$

In conditional logit the error terms are assumed type I extreme-value (or Gumbel) distribution with the probability density function and the cumulative distribution function of the random error given by $f(\varepsilon_{ih}) = \exp(-\varepsilon_{ih})\exp(-\exp(-\varepsilon_{ih}))$ and $F(\varepsilon_{ih}) = \exp(-\exp(-\varepsilon_{ih}))$ respectively.

**Example 7**

Consider individuals making a choice between three modes of transport for their journey to work. A researcher has collected the typical travel time of each mode of transportation specific to each of 50 individuals, together with their preferred mode of transport (i.e. travel time varies across individuals and transportation alternatives). The SAS procedure **`proc mdc`** can be used to assess whether the travel times influence the choice of the mode of transport.

The 50 observations were loaded into a dataset called `origdata` (see tables below). This data had to be transformed into order to use **`proc mdc`** using:

```
data newdata(keep=pid decision mode ttime);
   set origdata;
   array tvec{3} ttime1 - ttime3;
   retain pid 0;
   pid + 1;
   do i = 1 to 3;
      mode = i;
```

```
        ttime = tvec{i};
        decision = ( choice = i );
        output;
    end;
run;
```

| | ttime1 | ttime2 | ttime3 | choice |
|---|---|---|---|---|
| 1 | 16.481 | 16.196 | 23.89 | 2 |
| 2 | 15.123 | 11.373 | 14.182 | 2 |
| 3 | 19.469 | 8.822 | 20.819 | 2 |
| 4 | 18.847 | 15.649 | 21.28 | 2 |
| 5 | 12.578 | 10.671 | 18.335 | 2 |
| 6 | 11.513 | 20.582 | 27.838 | 1 |
| 7 | 10.651 | 15.537 | 17.418 | 1 |
| 8 | 8.359 | 15.675 | 21.05 | 1 |
| 9 | 11.679 | 12.668 | 23.104 | 1 |
| 10 | 23.237 | 10.356 | 21.346 | 2 |
| 11 | 13.236 | 16.019 | 10.087 | 3 |
| 12 | 20.052 | 16.861 | 14.168 | 3 |
| 13 | 18.917 | 14.764 | 21.564 | 2 |
| 14 | 18.2 | 6.868 | 19.095 | 2 |
| 15 | 10.777 | 16.554 | 15.938 | 1 |
| 16 | 20.003 | 6.377 | 9.314 | 2 |
| 17 | 19.768 | 8.523 | 18.96 | 2 |
| 18 | 8.151 | 13.845 | 17.643 | 2 |
| 19 | 22.173 | 18.045 | 15.535 | 1 |
| 20 | 13.134 | 11.067 | 19.108 | 2 |
| 21 | 14.051 | 14.247 | 15.764 | 1 |
| 22 | 14.685 | 10.811 | 12.361 | 3 |
| 23 | 11.666 | 10.758 | 16.445 | 1 |
| 24 | 17.211 | 15.201 | 17.059 | 3 |
| 25 | 13.93 | 16.227 | 22.024 | 1 |

| | ttime1 | ttime2 | ttime3 | choice |
|---|---|---|---|---|
| 26 | 15.237 | 14.345 | 19.984 | 2 |
| 27 | 10.84 | 11.071 | 10.188 | 1 |
| 28 | 16.841 | 11.224 | 13.417 | 2 |
| 29 | 13.913 | 16.991 | 26.618 | 3 |
| 30 | 13.089 | 9.822 | 19.162 | 2 |
| 31 | 16.626 | 10.725 | 15.285 | 3 |
| 32 | 13.477 | 15.509 | 24.421 | 2 |
| 33 | 20.851 | 14.557 | 19.8 | 2 |
| 34 | 11.365 | 12.673 | 22.212 | 2 |
| 35 | 13.296 | 10.076 | 17.81 | 2 |
| 36 | 15.417 | 14.103 | 21.05 | 1 |
| 37 | 15.938 | 11.18 | 19.851 | 2 |
| 38 | 19.034 | 14.125 | 19.764 | 2 |
| 39 | 10.466 | 12.841 | 18.54 | 1 |
| 40 | 15.799 | 16.979 | 13.074 | 3 |
| 41 | 12.713 | 15.105 | 13.629 | 2 |
| 42 | 16.908 | 10.958 | 19.713 | 2 |
| 43 | 17.098 | 6.853 | 14.502 | 2 |
| 44 | 18.608 | 14.286 | 18.301 | 2 |
| 45 | 11.059 | 10.812 | 20.121 | 1 |
| 46 | 15.641 | 10.754 | 24.669 | 2 |
| 47 | 7.822 | 18.949 | 16.904 | 1 |
| 48 | 12.824 | 5.697 | 19.183 | 2 |
| 49 | 11.852 | 12.147 | 15.672 | 2 |
| 50 | 15.557 | 8.307 | 22.286 | 2 |

| Obs | pid | mode | ttime | decision |
|---|---|---|---|---|
| 1 | 1 | 1 | 16.481 | 0 |
| 2 | 1 | 2 | 16.196 | 1 |
| 3 | 1 | 3 | 23.890 | 0 |
| 4 | 2 | 1 | 15.123 | 0 |
| 5 | 2 | 2 | 11.373 | 1 |
| 6 | 2 | 3 | 14.182 | 0 |
| 7 | 3 | 1 | 19.469 | 0 |
| 8 | 3 | 2 | 8.822 | 1 |
| 9 | 3 | 3 | 20.819 | 0 |

The dataset `newdata` contains 3 rows of data per individual (relating to each of the modes of transport). The 'decision' variable equals 1 for the preferred mode of transport (zero otherwise).

Using **`proc mdc`**:

```
proc mdc data=newdata;
    model decision = ttime / type=clogit nchoice=3
        optmethod=qn covest=hess;
    id pid;
      output out=probdata pred=p;
run;
```

NB: The `nchoice=3` indicates that 3 alternatives are available, while `type=clogit` specifies a conditional logit model, the `optmethod=qn` indicates that the quasi-Newton method is to be used in maximising the likelihood, while `covest=hess` specifies the covariance from the Hessian matrix.

The following results are obtained:

| Parameter Estimates | | | | | |
|---|---|---|---|---|---|
| Parameter | DF | Estimate | Standard Error | t Value | Approx Pr > \|t\| |
| ttime | 1 | -0.3572 | 0.0776 | -4.60 | <.0001 |

The effect of travel time is seen to be significant. Furthermore a utility will lower when the travel time is longer (due to the negative sign of the $\beta$ parameter). Thus in this model the classification will be based on the travel time that is the quickest.

Note that for individual $i$:

Consider the first individual (with travel times 16.481, 16.196, and 23.89):

Thus the model predicts that for this individual their preferred mode of transportation is mode 2 (since 50.8%>45.9% and 50.8%>3.3%). Notice this corresponds to the shortest of the three travel times.

Similarly, consider the 23rd individual (with travel times 11.666, 10.758 and 16.445):

$$\Pr(Y=1|\mathbf{X}) = \frac{e^{-0.3572\times 11.666}}{e^{-0.3572\times 11.666}+e^{-0.3572\times 10.758}+e^{-0.3572\times 16.445}} = 0.390$$

$$\Pr(Y=2|\mathbf{X}) = \frac{e^{-0.3572\times 10.758}}{e^{-0.3572\times 11.666}+e^{-0.3572\times 10.758}+e^{-0.3572\times 16.445}} = 0.539$$

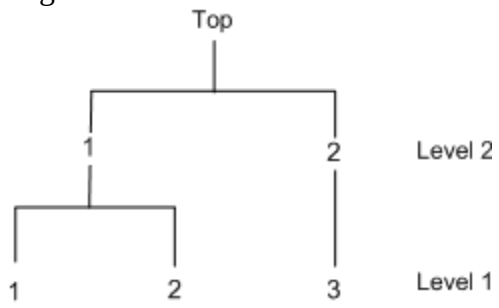
$$\Pr(Y=3|\mathbf{X}) = \frac{e^{-0.3572\times 16.445}}{e^{-0.3572\times 11.666}+e^{-0.3572\times 10.758}+e^{-0.3572\times 16.445}} = 0.071$$

Thus the model predicts that for this individual their preferred mode of transportation is mode 2 (since 53.9%>39.0% and 53.9%>7.1%). Notice this again corresponds to the shortest of the three travel times (and differs from the individual's actual preferred mode which is actually mode 1).

## 5.9 Nested Logit Models

Conditional logit models, like multinomial regression models still satisfy the IIA property where the ratio of the choice probabilities for alternatives $j$ and $h$ does not depend on any alternatives other than $j$ and $h$. Nested logit models can be used if the set of alternatives faced by an individual can be partitioned into subsets (or nests) such that the IIA property holds within subsets but not across subsets.

For example consider the following nested structure:




A nesting parameter $\lambda$, measures the degree of independence in unobserved utility among the alternatives in nests. Strong correlation and thus less independence is indicated by lower values of $\lambda$. When $\lambda = 1$ no correlations exist and the nested logit will collapse to the conditional logit model.

The probability that a decision maker $i$ prefer alternative $h$ in nest $g$ is given by:

$$\Pr\left(Y=h\middle|\mathbf{X}\right)=\frac{e^{\left(\beta_1 X_{1,ih}+\beta_2 X_{2,ih}+\ldots+\beta_p X_{p,ih}\right)/\lambda}\left(\sum_{\text{all alternatives } j \text{ in nest } g} e^{\left(\beta_1 X_{1,ij}+\beta_2 X_{2,ij}+\ldots+\beta_p X_{p,ij}\right)/\lambda}\right)^{\lambda-1}}{\sum_{\text{all nests}}\left(\sum_{\text{all alternatives } j \text{ in nest } g} e^{\left(\beta_1 X_{1,ij}+\beta_2 X_{2,ij}+\ldots+\beta_p X_{p,ij}\right)/\lambda}\right)^{\lambda}}$$

Notice that for alternatives within the same nest (1 and 2 above)

$$\frac{\Pr\left(Y=2\middle|\mathbf{X}\right)}{\Pr\left(Y=1\middle|\mathbf{X}\right)}=\frac{e^{\left(\beta_1 X_{1,i2}+\beta_2 X_{2,i2}+\ldots+\beta_p X_{p,i2}\right)/\lambda}\left(\sum_{\text{all alternatives } j \text{ in nest } 1} e^{\left(\beta_1 X_{1,ij}+\beta_2 X_{2,ij}+\ldots+\beta_p X_{p,ij}\right)/\lambda}\right)^{\lambda-1}}{e^{\left(\beta_1 X_{1,i1}+\beta_2 X_{2,i1}+\ldots+\beta_p X_{p,i1}\right)/\lambda}\left(\sum_{\text{all alternatives } j \text{ in nest } 1} e^{\left(\beta_1 X_{1,ij}+\beta_2 X_{2,ij}+\ldots+\beta_p X_{p,ij}\right)/\lambda}\right)^{\lambda-1}}$$

i.e. the IIA property holds for alternatives within the same nest.

Yet the ratio of probabilities for alternatives within different nests do not satisfy the IIA property. For example consider alternatives 3 and 1 (in different nests):

$$\frac{\Pr\left(Y=3\middle|\mathbf{X}\right)}{\Pr\left(Y=1\middle|\mathbf{X}\right)}=\frac{e^{\left(\beta_1 X_{1,i3}+\beta_2 X_{2,i3}+\ldots+\beta_p X_{p,i3}\right)/\lambda}\left(\sum_{\text{all alternatives } j \text{ in nest 2}} e^{\left(\beta_1 X_{1,ij}+\beta_2 X_{2,ij}+\ldots+\beta_p X_{p,ij}\right)/\lambda}\right)^{\lambda-1}}{e^{\left(\beta_1 X_{1,i1}+\beta_2 X_{2,i1}+\ldots+\beta_p X_{p,i1}\right)/\lambda}\left(\sum_{\text{all alternatives } j \text{ in nest 1}} e^{\left(\beta_1 X_{1,ij}+\beta_2 X_{2,ij}+\ldots+\beta_p X_{p,ij}\right)/\lambda}\right)^{\lambda-1}}$$

**Example 8**

Consider the transportation modes 1, 2 and 3 from Example 7. Perhaps 1 and 2 denote public transportation options, thus may be considered to be within the one nest – like is outlined in the nest structure above.

Suppose that $\beta = -0.402483$ and $\lambda = 0.8208647802$. Then for individual 1 (with times 16.481, 16.196 and 23.89 for modes 1-3 respectively):

$$\Pr\left(Y=1\middle|\mathbf{X}\right)$$

$$=\frac{e^{\left(16.481\times(-0.402483)\right)/0.82086}\left(e^{\left(16.481\times(-0.402483)\right)/0.82086}+e^{\left(16.196\times(-0.402483)\right)/0.82086}\right)^{0.82086-1}}{\left(\left(e^{\left(16.481\times(-0.402483)\right)/0.82086}+e^{\left(16.196\times(-0.402483)\right)/0.82086}\right)^{0.82086}+\left(e^{\left(23.89\times(-0.402483)\right)/0.82086}\right)^{0.82086}\right)}$$

$$\Pr\left(Y=1\middle|\mathbf{X}\right)=0.45$$

$$\Pr\left(Y=2\middle|\mathbf{X}\right)$$

$$=\frac{e^{\left(16.196\times(-0.402483)\right)/0.82086}\left(e^{\left(16.481\times(-0.402483)\right)/0.82086}+e^{\left(16.196\times(-0.402483)\right)/0.82086}\right)^{0.82086-1}}{\left(\left(e^{\left(16.481\times(-0.402483)\right)/0.82086}+e^{\left(16.196\times(-0.402483)\right)/0.82086}\right)^{0.82086}+\left(e^{\left(23.89\times(-0.402483)\right)/0.82086}\right)^{0.82086}\right)}$$

$$\Pr\left(Y=2\middle|\mathbf{X}\right)=0.52$$

$$\Pr\left(Y=3\middle|\mathbf{X}\right)$$

$$=\frac{e^{\left(23.89\times(-0.402483)\right)/0.82086}\left(e^{\left(23.89\times(-0.402483)\right)/0.82086}\right)^{0.82086-1}}{\left(\left(e^{\left(16.481\times(-0.402483)\right)/0.82086}+e^{\left(16.196\times(-0.402483)\right)/0.82086}\right)^{0.82086}+\left(e^{\left(23.89\times(-0.402483)\right)/0.82086}\right)^{0.82086}\right)}$$

$$\Pr\left(Y=3\middle|\mathbf{X}\right)=0.03$$

Note that SAS currently applies an unscaled version of the nested logit model (which is not compatible with the random utility maximisation hypothesis):

$$\Pr\left(Y=h\middle|\mathbf{X}\right)=\frac{e^{\left(\beta_1 X_{1,ih}+\beta_2 X_{2,ih}+\ldots+\beta_p X_{p,ih}\right)}\left(\sum_{\text{all alternatives } j \text{ in nest } g} e^{\left(\beta_1 X_{1,ij}+\beta_2 X_{2,ij}+\ldots+\beta_p X_{p,ij}\right)}\right)^{\lambda-1}}{\sum_{\text{all nests}}\left(\sum_{\text{all alternatives } j \text{ in nest } g} e^{\left(\beta_1 X_{1,ij}+\beta_2 X_{2,ij}+\ldots+\beta_p X_{p,ij}\right)}\right)^{\lambda}}$$

mlogit in R can be used for both the scaled and unscaled version of the model.

Note also that $\lambda$ can also be set to vary across the different nests i.e. replace $\lambda$ by $\lambda_g$.

### 5.10 Ordinal Regression

There are a number of options available for handling a dependent variable that is ordinal e.g.

- Ignoring the ordering of the categories of the dependent variable and treating it has nominal.
  - Information on the ordering of the categories of the dependent variable is ignored and thus lost.
- Treating the variable as though it were measured on an interval or ratio scale.
  - This approach has issues however, such as it cannot be fully efficient (in the minimum variance sense), with distributional assumptions made about the dependent variable likely to be incorrect. Also, unless certain conditions prevail (which includes a large number of ordered categories), the assumption of homoscedasticity is likely to be violated.
- Treating the variable as though it were measured on a true ordinal scale – ordinal logistic regression models.

Ordinal logistic models all explicitly incorporate the ordering of the categories of the dependent variables without regard to anything about the numerical codes other than their order. Different models exist including the cumulative logit model, the continuation ratio logit model, the adjacent categories logit model and the stereotype model.

The **cumulative logit model** is the most widely implemented ordinal logistic regression model. The model is motivated by the assumption that the ordinal dependent variable represents a crude measurement of an underlying continuous variable $\eta$, where $\eta=\beta_0+\beta_1 X_1+\ldots+\beta_p X_p$. We cannot observe $\eta$ directly, but we believe that $\eta$ can be transformed to $Y$ (an ordinal variable) using: i) $Y$ is in the first category $(Y=1)$ whenever $\eta<\theta_1$; ii) $Y=2$ whenever $\theta_1\le\eta<\theta_2$; iii) $Y=3$ whenever $\theta_2\le\eta<\theta_3$; (and so forth)...until for the final category, $I$, where $Y=I$ whenever $\eta>\theta_{I-1}$.

In the cumulative logit model, all categories at or below a given cut-off are compared with those categories above that cut-off according to:

In the **equal slopes model**, also called the **proportional odds model**, the subscript *i* is dropped from the $\beta$ coefficients. That is, the assumption of equal slopes but unequal intercepts corresponds to a model involving a set of parallel lines, one for each split $(Y > i \text{ vs. } Y \le i)$ in the cumulative logit model. Equivalently, the odds of being in one category as opposed to the next are equal regardless of which two categories are being compared. In the **generalised logit model**, the $\beta$ coefficients vary across the different values of *i*.

**Example 9**

A study wishes to look at factors that influence the decision of whether to apply to graduate school in the US. College juniors were asked if they are "unlikely", "somewhat likely", or "very likely" to apply to graduate school. Hence, the outcome variable "apply" has **three** (ordered) categories – coded 0, 1, 2 respectively. Data on parental educational status (pared - a 0/1 variable indicating whether at least one parent has a graduate degree), whether the undergraduate institution is public or private (public - which is a 0/1 variable where 1 indicates that the undergraduate institution is a public university and 0 indicates that it is a private university), and current Grade Point Average (GPA) of each student is also collected.

The following SAS code was run to perform a (proportional odds) cumulative logit model:

```
proc logistic data = grad_sch_info desc;
class pared(ref='0') public(ref='0') / param=reference;
model apply = pared public gpa;
output out=pred p=phat lower=lcl upper=ucl;
score out=scoress;
run;
```

**Response Profile**

| Ordered Value | APPLY | Total Frequency |
|---|---|---|
| 1 | 2 | 40 |
| 2 | 1 | 140 |
| 3 | 0 | 220 |

*Probabilities modeled are cumulated over the lower Ordered Values.*

**Analysis of Maximum Likelihood Estimates**

| Parameter | | DF | Estimate | Standard Error | Wald Chi-Square | Pr > ChiSq |
|---|---|---|---|---|---|---|
| Intercept | 2 | 1 | -4.2983 | 0.8092 | 28.2189 | <.0001 |
| Intercept | 1 | 1 | -2.2029 | 0.7844 | 7.8869 | 0.0050 |
| PARED | 1 | 1 | 1.0478 | 0.2684 | 15.2350 | <.0001 |
| PUBLIC | 1 | 1 | -0.0585 | 0.2886 | 0.0411 | 0.8393 |
| GPA | | 1 | 0.6156 | 0.2626 | 5.4963 | 0.0191 |

Notice that both pared and gpa are statistically significant; public is not. So for pared, we would say that for a one unit increase in pared (i.e., going from 0 to 1), we expect a 1.05 increase in the log odds of being in a higher level of apply, given all of the other variables in the model are held constant. For gpa, we would say that for a one unit increase in gpa, we would expect a 0.62 increase in the log odds of being in a higher level of apply, given that all of the other variables in the model are held constant.

To demonstrate this, consider calculating $\Pr(Y=0)/\Pr(Y=1)/\Pr(Y=2)$ for three individuals:

- Individual A: pared=0, public=0, gpa=3.26
- Individual B: pared=1, public=0, gpa=3.26
- Individual C: pared=1, public=0, gpa=4.9

Individual A:
Using:

$$\ln\left(\frac{\Pr(Y\le i)}{\Pr(Y>i)}\right)=\theta_i-\left(\beta_1X_1+\ldots+\beta_pX_p\right)\qquad i=1,\ldots,I-1$$

For the first category [note that due to the ordering within SAS (due to the `desc` option), the first category is *Y*=2 ("very likely")]:

NB: SAS (like many software) use the **opposite sign** for the $\beta$ coefficients.

Notice that

$$\text{logit}\left(\Pr(Y=\text{first category})\right)=-2.291444$$

Hence

$$\Pr(Y=\text{first category})=\frac{1}{1+e^{-(-2.291444)}}=0.09183$$

That is: $\Pr(Y=\text{"very likely"})=0.09183$

Now consider *i*=2 in the above formula:

$$\ln\left(\frac{\Pr(Y=\text{first or second category})}{\Pr(Y\ne\text{first or second category})}\right)=\theta_{2nd}-\left(\beta_1X_1+\ldots+\beta_pX_p\right)$$

$$=-2.2029-(-1.0478\times0+0.0585\times0-0.6156\times3.26)$$

$$=-0.196044$$

Hence

$$\Pr(Y=\text{first or second category})=\frac{1}{1+e^{-(-0.196044)}}=0.45115$$

Therefore:

$$\Pr(Y=\text{second category})=0.45115\text{-}0.09183=0.35932$$

That is: $\Pr(Y = \text{"somewhat likely"}) = 0.35932$

Finally:

Individual B:

Hence

$$\Pr(Y = \text{first category}) = \frac{1}{1+e^{-(-1.243644)}} = 0.22380 = \Pr(Y = \text{"very likely"})$$

While:

$$\Pr(Y = \text{first or second category}) = \frac{1}{1+e^{-(0.851756)}} = 0.70094$$

Hence:

$$\Pr(Y = \text{"somewhat likely"}) = 0.70094 - 0.22380 = 0.47714$$

$$\Pr(Y = \text{"unlikely"}) = 1 - 0.70094 = 0.29906$$

Hence the most likely classification for individual B would be **"somewhat likely"** (given 0.47714=max{0.22380, 0.47714,0.29906}).

Individual C:

$$\Pr(Y = \text{first category}) = \frac{1}{1+e^{-(-0.23406)}} = 0.44175 = \Pr(Y = \text{"very likely"})$$

$$\Pr(Y = \text{first or second category}) = \frac{1}{1+e^{-(1.86134)}} = 0.86545$$

$$\Pr(Y = \text{"somewhat likely"}) = 0.86545 - 0.44175 = 0.42370$$

$$\Pr(Y = \text{"unlikely"}) = 1 - 0.86545 = 0.13455$$

Hence the most likely classification for individual C would be **"very likely"** (given 0.44175=max{0.44175, 0.42370,0.13455}).

NB: A **generalised logit model** can be performed in SAS using the `/link=glogit` option within the `model` statement, which would produce three additional $\beta$ parameters.

## 5.11 Model Selection Methods within Regression Based Models

A number of criteria exist to aid the **selection of variables** to be included in a model based on the log likelihood values, $\ln(L)$.

Akaike information criterion (AIC)
Using AIC to compare models, the model with the smallest AIC is chosen. AIC is given by:

$$AIC = 2k - 2\ln(L)$$

where $k$ is the number of parameters to be evaluated in the model. Notice that the measure penalises models that have excessive number of parameters (i.e. they may be overfitting). Note that this measure can also be used to compare between different statistical models (e.g. logistic versus probit).

Bayesian information criterion (BIC)
The Bayesian information criterion (BIC) or Schwarz criterion has a larger penalty than the AIC. Again the model chosen would be the one with the smallest BIC. For large number of observations ($n$), the BIC is given by:

$$BIC = k\ln(n) - 2\ln(L)$$

Likelihood Ratio Tests
In the likelihood ratio test two nested models are compared. Model 1 is nested in Model 2 if the covariates in Model 1 are also contained in Model 2 e.g. Model 1 contains age, while Model 2 contain age and sex. The comparison takes the form:

$$LRT = -2\ln\left(\frac{L(\text{model 1})}{L(\text{model 2})}\right) \sim \chi^2_{\text{no of additional parameters}}$$

Under the null hypothesis that the additional $\beta$ parameter(s) in model 2 (and not in model 1) all equal zero this ratio should be an observation from a $\chi^2$ distribution. If the statistic lies in the tail of the distribution, there is evidence to reject the null hypothesis (i.e. some of the additional covariates in model 2 are required). Conversely, if we do not reject our null hypothesis, this suggests that all the additional covariates in model 2 are not required.

For example consider the following results from a regression-based model:

| Model | covariates | −2 log likelihood |
|---|---|---|
| 1 | Null | 309.716 |
| 2 | LogBUN HGB scalc age | 294.348 |
| 3 | LogBUN HGB scalc | 296.078 |

Comparing models 2 and 3 (so $H_0 : \beta_{age} = 0$):

This is an observation from a $\chi^2_1$ distribution, with a critical 5% value of 3.84. So we fail to reject the null hypothesis. This means there is no evidence that age is required in the model.

## 5.12 Comparing Classifiers

- Re-substitution error rate

  In order to calculate the **re-substitution error rate**, a classifier is first built based on some training data. The classifier is then applied to the same training data so that comparisons can be made between classifications predicted by the classifier and actual classifications recorded in the training data. The re-substitution error rate corresponds to the proportion of observations that are incorrectly predicted in the training data by the classifier. A re-substitution error rate can only indicate how good (or bad) a classifier is on the training data – it gives a performance measure for the algorithm (the lower, the better the algorithm). It does not correspond to how well the classifier would predict the classification of previously unseen observations.

- Predictive accuracy

  In order to assess the **predictive accuracy** of a classifier, observations recorded (with known classification) are split between **training** and **test** sets. The classifier is first built based on the training set and then applied to the test set. The predictive accuracy depends on the number of correctly classified observations in the test set:

There are a number of methods by which observations can be split between training and test sets:

1) Holdout

In the **holdout** method, a pre-defined percentage of the observations are randomly selected to be in the test set (usually around 20-30%). The predictive accuracy calculated can however be influenced by what observations are selected to be in the training and test set, so **repeated holdout** may be more reliable in giving a measure of the predictive accuracy of a classifier.

2) Repeated Holdout

In **repeated holdout**, the holdout method is repeated a large number of times. For each iteration, a certain proportion of observations are randomly selected for testing, the rest of the data is used for training. The predictive accuracy is calculated for each iteration and then averaged to yield an **overall predictive accuracy**. Repeated holdout still has the issue however that the different test sets considered potentially overlap and so this may influence the overall predictive accuracy calculated.

3) *K*-fold cross Validation

The ***K*-fold cross validation** method can be used to assess the accuracy of the classifier. The dataset is divided into *K* equal sized sections. The algorithm is tested *K* times, each time leaving out one of the *K* sections from building the classifier, but using it to *test* the classifier instead. As with the repeated holdout the predictive accuracy is calculated for each training/test set combination and then averaged to give an overall predictive accuracy. In *K*-fold cross validation however there is no overlap in the test sets considered.

- We can do *K*-fold cross validation for each possible classifier, and choose the model with the highest accuracy. Where there is a tie, we choose the simpler model.
- Actually, we should probably penalise the more complex models, even if they are more accurate, since more complex models are more likely to overfit.

4) Repeated *K*-fold cross Validation

In **repeated *K*-fold cross validation**, *K*-fold cross validation is repeated a number of times. That is, in each of the iterations, the observations in each of the *K* sections can potentially

change (due to their random partition into *K* sections at each iteration). Results are again averaged in determining the predictive accuracy.

5) Leave-one-out

For a dataset containing a total of *N* observations, leave-one-out corresponds to *N*-fold cross validation - *N* observations are divided into *N* equal-sized sections (i.e. 1 observation per section). In the leave-one-out method the classifier is built repeatedly leaving one observation out in each of the *N* iterations – in each iteration, the omitted observation is then used to *test* the classifier. The leave-one-out method involves **no random sub-sampling** and so is a **commonly-used** method. However **computationally it can be very expensive** for large datasets.

- Speed and scalability

  The time and space requirements for the two distinct phases of classification are:
  - Time to **construct** the classifier
    - In the case of the simpler linear classifier, the time taken to fit the line, this is linear in the number of instances.
  - Time to **use** the model
    - In the case of the simpler linear classifier, the time taken to test which side of the line the unlabeled instance is. This can be done in constant time.

  Note: Some classification algorithms are very efficient in one aspect, and very poor in the other.
- Robustness

  Handling noise, missing values and irrelevant features, streaming data
  - Consider what happens when we have:
    - Noise: e.g. a person's age could have been mistyped as 650 instead of 65, how does this affect our classifier? (This is important only for building the classifier, if the instance to be classified is noisy we can do nothing).
  - Missing values (For example suppose we want to classify an insect, but we only know the abdomen length (X-axis), and not the antennae length (Y-axis), can the classifier we were going to use still classify the instance?)
  - Irrelevant features (e.g. considered above included insect's weight)
  - Streaming data - For many real world problems, we don't have a single fixed dataset. Instead, the data continuously arrives, potentially forever… (stock market, weather data, sensor data etc)
- Interpretability
  - understanding and insight provided by the model
  - Some classifiers offer a *bonus* feature. The structure of the learned classifier tells us something about the domain. For example: if we try to classify peoples health risks based on just their height and weight, we could gain the following insight (Based on the observation that a single linear classifier does not work well, but two linear classifiers do). There are two ways to be unhealthy, being obese and being too skinny.