3 Introduction to Classification: Naïve Bayes and Nearest Neighbour

3.1 What Is Classification?

Classification is a task that occurs very frequently in everyday life. Essentially it involves dividing up objects so that each is assigned to one of a number of mutually exhaustive and exclusive categories known as *classes*. The term 'mutually exhaustive and exclusive' simply means that each object must be assigned to precisely one class, i.e. never to more than one and never to no class at all.

Many practical decision-making tasks can be formulated as classification problems, i.e. assigning people or objects to one of a number of categories, for example

– customers who are likely to buy or not buy a particular product in a supermarket

– people who are at high, medium or low risk of acquiring a certain illness

– student projects worthy of a distinction, merit, pass or fail grade

– objects on a radar display which correspond to vehicles, people, buildings or trees

– people who closely resemble, slightly resemble or do not resemble someone seen committing a crime

M. Bramer, *Principles of Data Mining*, Undergraduate Topics in Computer Science, DOI 10.1007/978-1-4471-7307-6_3

- houses that are likely to rise in value, fall in value or have an unchanged value in 12 months' time
- people who are at high, medium or low risk of a car accident in the next 12 months
- people who are likely to vote for each of a number of political parties (or none)
- the likelihood of rain the next day for a weather forecast (very likely, likely, unlikely, very unlikely).

We have already seen an example of a (fictitious) classification task, the 'degree classification' example, in the Introduction.

In this chapter we introduce two classification algorithms: one that can be used when all the attributes are categorical, the other when all the attributes are continuous. In the following chapters we come on to algorithms for generating classification trees and rules (also illustrated in the Introduction).

3.2 Naïve Bayes Classifiers

In this section we look at a method of classification that does not use rules, a decision tree or any other explicit representation of the classifier. Rather, it uses the branch of Mathematics known as *probability theory* to find the most likely of the possible classifications.

The significance of the first word of the title of this section will be explained later. The second word refers to the Reverend Thomas Bayes (1702–1761), an English Presbyterian minister and Mathematician whose publications included "Divine Benevolence, or an Attempt to Prove That the Principal End of the Divine Providence and Government is the Happiness of His Creatures" as well as pioneering work on probability. He is credited as the first Mathematician to use probability in an inductive fashion.

A detailed discussion of probability theory would be substantially outside the scope of this book. However the mathematical notion of probability corresponds fairly closely to the meaning of the word in everyday life.

The *probability* of an *event*, e.g. that the 6.30 p.m. train from London to your local station arrives on time, is a number from 0 to 1 inclusive, with 0 indicating 'impossible' and 1 indicating 'certain'. A probability of 0.7 implies that if we conducted a long series of *trials*, e.g. if we recorded the arrival time of the 6.30 p.m. train day by day for N days, we would expect the train to be on time on $0.7 \times N$ days. The longer the series of trials the more reliable this estimate is likely to be.

Usually we are not interested in just one event but in a set of alternative possible events, which are *mutually exclusive* and *exhaustive*, meaning that one and only one must always occur.

In the train example, we might define four mutually exclusive and exhaustive events

$E1$ – train cancelled
$E2$ – train ten minutes or more late
$E3$ – train less than ten minutes late
$E4$ – train on time or early.

The probability of an event is usually indicated by a capital letter P, so we might have

$P(E1) = 0.05$
$P(E2) = 0.1$
$P(E3) = 0.15$
$P(E4) = 0.7$

(Read as ‘the probability of event $E1$ is 0.05’ etc.)

Each of these probabilities is between 0 and 1 inclusive, as it has to be to qualify as a probability. They also satisfy a second important condition: the sum of the four probabilities has to be 1, because precisely one of the events must always occur. In this case

$$P(E1) + P(E2) + P(E3) + P(E4) = 1$$

In general, the sum of the probabilities of a set of mutually exclusive and exhaustive events must always be 1.

Generally we are not in a position to know the true probability of an event occurring. To do so for the train example we would have to record the train's arrival time for all possible days on which it is scheduled to run, then count the number of times events $E1$, $E2$, $E3$ and $E4$ occur and divide by the total number of days, to give the probabilities of the four events. In practice this is often prohibitively difficult or impossible to do, especially (as in this example) if the trials may potentially go on forever. Instead we keep records for a *sample* of say 100 days, count the number of times $E1$, $E2$, $E3$ and $E4$ occur, divide by 100 (the number of days) to give the frequency of the four events and use these as estimates of the four probabilities.

For the purposes of the classification problems discussed in this book, the ‘events’ are that an instance has a particular classification. Note that classifications satisfy the ‘mutually exclusive and exhaustive’ requirement.

The outcome of each trial is recorded in one row of a table. Each row must have one and only one classification.

For classification tasks, the usual terminology is to call a table (dataset) such as Figure 3.1 a *training set*. Each row of the training set is called an *instance*. An instance comprises the values of a number of attributes and the corresponding classification.

The training set constitutes the results of a sample of trials that we can use to predict the classification of other (unclassified) instances.

Suppose that our training set consists of 20 instances, each recording the value of four attributes as well as the classification. We will use classifications: *cancelled*, *very late*, *late* and *on time* to correspond to the events $E1$, $E2$, $E3$ and $E4$ described previously.

day	season	wind	rain	class
weekday	spring	none	none	on time
weekday	winter	none	slight	on time
weekday	winter	none	slight	on time
weekday	winter	high	heavy	late
saturday	summer	normal	none	on time
weekday	autumn	normal	none	very late
holiday	summer	high	slight	on time
sunday	summer	normal	none	on time
weekday	winter	high	heavy	very late
weekday	summer	none	slight	on time
saturday	spring	high	heavy	cancelled
weekday	summer	high	slight	on time
saturday	winter	normal	none	late
weekday	summer	high	none	on time
weekday	winter	normal	heavy	very late
saturday	autumn	high	slight	on time
weekday	autumn	none	heavy	on time
holiday	spring	normal	slight	on time
weekday	spring	normal	none	on time
weekday	spring	normal	slight	on time

Figure 3.1 The *train* Dataset

How should we use probabilities to find the most likely classification for an unseen instance such as the one below?

weekday	winter	high	heavy	????

One straightforward (but flawed) way is just to look at the frequency of each of the classifications in the training set and choose the most common one. In this case the most common classification is *on time*, so we would choose that.

The flaw in this approach is, of course, that all unseen instances will be classified in the same way, in this case as *on time*. Such a method of classification is not necessarily bad: if the probability of *on time* is 0.7 and we guess that every unseen instance should be classified as *on time*, we could expect to be right about 70% of the time. However, the aim is to make correct predictions as often as possible, which requires a more sophisticated approach.

The instances in the training set record not only the classification but also the values of four attributes: *day*, *season*, *wind* and *rain*. Presumably they are recorded because we believe that in some way the values of the four attributes affect the outcome. (This may not necessarily be the case, but for the purpose of this chapter we will assume it is true.) To make effective use of the additional information represented by the attribute values we first need to introduce the notion of *conditional probability*.

The probability of the train being on time, calculated using the frequency of *on time* in the training set divided by the total number of instances is known as the *prior probability*. In this case $P(\text{class} = \text{on time}) = 14/20 = 0.7$. If we have no other information this is the best we can do. If we have other (relevant) information, the position is different.

What is the probability of the train being on time if we know that the season is winter? We can calculate this as the number of times class = on time and season = winter (in the same instance), divided by the number of times the season is winter, which comes to $2/6 = 0.33$. This is considerably less than the prior probability of 0.7 and seems intuitively reasonable. Trains are less likely to be on time in winter.

The probability of an event occurring if we know that an attribute has a particular value (or that several variables have particular values) is called the *conditional probability* of the event occurring and is written as, e.g.

$P(\text{class} = \text{on time} \mid \text{season} = \text{winter})$.

The vertical bar can be read as 'given that', so the whole term can be read as 'the probability that the class is *on time* given that the season is *winter*'.

$P(\text{class} = \text{on time} \mid \text{season} = \text{winter})$ is also called a *posterior probability*. It is the probability that we can calculate for the classification *after* we have obtained the information that the season is winter. By contrast, the prior probability is that estimated *before* any other information is available.

To calculate the most likely classification for the 'unseen' instance given previously we could calculate the probability of

$$\begin{aligned} P(\text{class} = \text{on time} \quad | \quad & \text{day} = \text{weekday and season} = \text{winter} \\ & \text{and wind} = \text{high and rain} = \text{heavy}) \end{aligned}$$

and do similarly for the other three possible classifications. However there are only two instances in the training set with that combination of attribute values and basing any estimates of probability on these is unlikely to be helpful.

To obtain a reliable estimate of the four classifications a more indirect approach is needed. We could start by using conditional probabilities based on a single attribute.

For the *train* dataset

$P(\text{class} = \text{on time} \mid \text{season} = \text{winter}) = 2/6 = 0.33$
$P(\text{class} = \text{late} \mid \text{season} = \text{winter}) = 1/6 = 0.17$
$P(\text{class} = \text{very late} \mid \text{season} = \text{winter}) = 3/6 = 0.5$
$P(\text{class} = \text{cancelled} \mid \text{season} = \text{winter}) = 0/6 = 0$

The third of these has the largest value, so we could conclude that the most likely classification is very late, a different result from using the prior probability as before.

We could do a similar calculation with attributes day, rain and wind. This might result in other classifications having the largest value. Which is the best one to take?

The **Naïve Bayes** algorithm gives us a way of combining the prior probability and conditional probabilities in a single formula, which we can use to calculate the probability of each of the possible classifications in turn. Having done this we choose the classification with the largest value.

Incidentally the first word in the rather derogatory sounding name Naïve Bayes refers to the assumption that the method makes, that the effect of the value of one attribute on the probability of a given classification is independent of the values of the other attributes. In practice, that may not be the case. Despite this theoretical weakness, the Naïve Bayes method often gives good results in practical use.

The method uses conditional probabilities, but the other way round from before. (This may seem a strange approach but is justified by the method that follows, which is based on a well-known Mathematical result known as Bayes Rule.)

Instead of (say) the probability that the class is very late given that the season is winter, $P(\text{class} = \text{very late} \mid \text{season} = \text{winter})$, we use the conditional probability that the season is winter given that the class is very late, i.e. $P(\text{season} = \text{winter} \mid \text{class} = \text{very late})$. We can calculate this as the number of times that season = winter and class = very late occur in the same instance, divided by the number of instances for which the class is *very late.*

In a similar way we can calculate other conditional probabilities, for example $P(\text{rain} = \text{none} \mid \text{class} = \text{very late})$.

For the *train* data we can tabulate all the conditional and prior probabilities as shown in Figure 3.2.

	class = on time	class = late	class = very late	class = cancelled
day = weekday	**9/14 = 0.64**	**1/2 = 0.5**	**3/3 = 1**	**0/1 = 0**
day = saturday	2/14 = 0.14	1/2 = 0.5	0/3 = 0	1/1 = 1
day = sunday	1/14 = 0.07	0/2 = 0	0/3 = 0	0/1 = 0
day = holiday	2/14 = 0.14	0/2 = 0	0/3 = 0	0/1 = 0
season = spring	4/14 = 0.29	0/2 = 0	0/3 = 0	1/1 = 1
season = summer	6/14 = 0.43	0/2 = 0	0/3 = 0	0/1 = 0
season = autumn	2/14 = 0.14	0/2 = 0	1/3 = 0.33	0/1 = 0
season = winter	**2/14 = 0.14**	**2/2 = 1**	**2/3 = 0.67**	**0/1 = 0**
wind = none	5/14 = 0.36	0/2 = 0	0/3 = 0	0/1 = 0
wind = high	**4/14 = 0.29**	**1/2 = 0.5**	**1/3 = 0.33**	**1/1 = 1**
wind = normal	5/14 = 0.36	1/2 = 0.5	2/3 = 0.67	0/1 = 0
rain = none	5/14 = 0.36	1/2 =0.5	1/3 = 0.33	0/1 = 0
rain = slight	8/14 = 0.57	0/2 = 0	0/3 = 0	0/1 = 0
rain = heavy	**1/14 = 0.07**	**1/2 = 0.5**	**2/3 = 0.67**	**1/1 = 1**
Prior Probability	**14/20 = 0.70**	**2/20 = 0.10**	**3/20 = 0.15**	**1/20 = 0.05**

Figure 3.2 Conditional and Prior Probabilities: *train* Dataset

For example, the conditional probability $P(\text{day} = \text{weekday} \mid \text{class} = \text{on time})$ is the number of instances in the *train* dataset for which day = weekday and class = on time, divided by the total number of instances for which class = on time. These numbers can be counted from Figure 3.1 as 9 and 14, respectively. So the conditional probability is 9/14 = 0.64.

The prior probability of class = very late is the number of instances in Figure 3.1 for which class = very late divided by the total number of instances, i.e. 3/20 = 0.15.

We can now use these values to calculate the probabilities of real interest to us. These are the posterior probabilities of each possible class occurring for a specified instance, i.e. for known values of all the attributes. We can calculate these posterior probabilities using the method given in Figure 3.3.

Naïve Bayes Classification

Given a set of k mutually exclusive and exhaustive classifications c_1, c_2, …, c_k, which have prior probabilities $P(c_1)$, $P(c_2)$, …, $P(c_k)$, respectively, and n attributes a_1, a_2, …, a_n which for a given instance have values v_1, v_2, …, v_n respectively, the posterior probability of class c_i occurring for the specified instance can be shown to be proportional to

$$P(c_i) \times P(a_1 = v_1 \text{ and } a_2 = v_2 \ldots \text{ and } a_n = v_n \mid c_i)$$

Making the assumption that the attributes are independent, the value of this expression can be calculated using the product

$$\boldsymbol{P(c_i) \times P(a_1 = v_1 \mid c_i) \times P(a_2} = v_2 \boldsymbol{\mid c_i) \times \ldots \times P(a_n = v_n \mid c_i)}$$

We calculate this product for each value of i from 1 to k and choose the classification that has the largest value.

Figure 3.3 The Naïve Bayes Classification Algorithm

The formula shown in bold in Figure 3.3 combines the prior probability of c_i with the values of the n possible conditional probabilities involving a test on the value of a single attribute.

It is often written as $P(c_i) \times \prod_{j=1}^{n} P(a_j = v_j \mid class = c_i)$.

Note that the Greek letter $\prod$ (pronounced pi) in the above formula is not connected with the mathematical constant 3.14159…. It indicates the product obtained by multiplying together the n values $P(a_1 = v_1 \mid c_i)$, $P(a_2 = v_2 \mid c_i)$ etc.

($\prod$ is the capital form of 'pi'. The lower case form is π. The equivalents in the Roman alphabet are P and p. P is the first letter of 'Product'.)

When using the Naïve Bayes method to classify a series of unseen instances the most efficient way to start is by calculating all the prior probabilities and also all the conditional probabilities involving one attribute, though not all of them may be required for classifying any particular instance.

Using the values in each of the columns of Figure 3.2 in turn, we obtain the following posterior probabilities for each possible classification for the unseen instance:

weekday	winter	high	heavy	????

<u>class = on time</u>

$0.70 \times 0.64 \times 0.14 \times 0.29 \times 0.07 = 0.0013$

<u>class = late</u>

$0.10 \times 0.50 \times 1.00 \times 0.50 \times 0.50 = 0.0125$

<u>class = very late</u>

$0.15 \times 1.00 \times 0.67 \times 0.33 \times 0.67 = 0.0222$

<u>class = cancelled</u>

$0.05 \times 0.00 \times 0.00 \times 1.00 \times 1.00 = 0.0000$

The largest value is for class = very late.

Note that the four values calculated are not themselves probabilities, as they do not sum to 1. This is the significance of the phrasing 'the posterior probability ... can be shown to be proportional to' in Figure 3.3. Each value can be 'normalised' to a valid posterior probability simply by dividing it by the sum of all four values. In practice, we are interested only in finding the largest value so the normalisation step is not necessary.

The Naïve Bayes approach is a very popular one, which often works well. However it has a number of potential problems, the most obvious one being that it relies on all attributes being categorical. In practice, many datasets have a combination of categorical and continuous attributes, or even only continuous attributes. This problem can be overcome by converting the continuous attributes to categorical ones, using a method such as those described in Chapter 8 or otherwise.

A second problem is that estimating probabilities by relative frequencies can give a poor estimate if the number of instances with a given attribute/value combination is small. In the extreme case where it is zero, the posterior probability will inevitably be calculated as zero. This happened for class = cancelled in the above example. This problem can be overcome by using a more complicated formula for estimating probabilities, but this will not be discussed further here.

3.3 Nearest Neighbour Classification

Nearest Neighbour classification is mainly used when all attribute values are continuous, although it can be modified to deal with categorical attributes.

The idea is to estimate the classification of an unseen instance using the classification of the instance or instances that are *closest* to it, in some sense that we need to define.

Supposing we have a training set with just two instances such as the following

a	*b*	*c*	*d*	*e*	*f*	Class
yes	no	no	6.4	8.3	low	negative
yes	yes	yes	18.2	4.7	high	positive

There are six attribute values, followed by a classification (positive or negative).

We are then given a third instance

yes	no	no	6.6	8.0	low	????

What should its classification be?

Even without knowing what the six attributes represent, it seems intuitively obvious that the unseen instance is *nearer* to the first instance than to the second. In the absence of any other information, we could reasonably predict its classification using that of the first instance, i.e. as 'negative'.

In practice there are likely to be many more instances in the training set but the same principle applies. It is usual to base the classification on those of the k nearest neighbours (where k is a small integer such as 3 or 5), not just the nearest one. The method is then known as *k-Nearest Neighbour* or just *k-NN classification* (Figure 3.4).

Basic *k*-Nearest Neighbour Classification Algorithm

– Find the k training instances that are closest to the unseen instance.

– Take the most commonly occurring classification for these k instances.

Figure 3.4 The Basic k-Nearest Neighbour Classification Algorithm

We can illustrate *k-NN* classification diagrammatically when the *dimension* (i.e. the number of attributes) is small. The following example illustrates the case where the dimension is just 2. In real-world data mining applications it can of course be considerably larger.

Figure 3.5 shows a training set with 20 instances, each giving the values of two attributes and an associated classification.

How can we estimate the classification for an 'unseen' instance where the first and second attributes are 9.1 and 11.0, respectively?

For this small number of attributes we can represent the training set as 20 points on a two-dimensional graph with values of the first and second attributes measured along the horizontal and vertical axes, respectively. Each point is labelled with a + or − symbol to indicate that the classification is positive or negative, respectively. The result is shown in Figure 3.6.

Attribute 1	Attribute 2	Class
0.8	6.3	$-$
1.4	8.1	$-$
2.1	7.4	$-$
2.6	14.3	$+$
6.8	12.6	$-$
8.8	9.8	$+$
9.2	11.6	$-$
10.8	9.6	$+$
11.8	9.9	$+$
12.4	6.5	$+$
12.8	1.1	$-$
14.0	19.9	$-$
14.2	18.5	$-$
15.6	17.4	$-$
15.8	12.2	$-$
16.6	6.7	$+$
17.4	4.5	$+$
18.2	6.9	$+$
19.0	3.4	$-$
19.6	11.1	$+$

Figure 3.5 Training Set for k-Nearest Neighbour Example

A circle has been added to enclose the five nearest neighbours of the unseen instance, which is shown as a small circle close to the centre of the larger one.

The five nearest neighbours are labelled with three $+$ signs and two $-$ signs, so a basic *5-NN* classifier would classify the unseen instance as 'positive' by a form of majority voting. There are other possibilities, for example the 'votes' of each of the k nearest neighbours can be weighted, so that the classifications of closer neighbours are given greater weight than the classifications of more distant ones. We will not pursue this here.

We can represent two points in two dimensions ('in two-dimensional space' is the usual term) as (a_1, a_2) and (b_1, b_2) and visualise them as points in a plane.

When there are three attributes we can represent the points by (a_1, a_2, a_3) and (b_1, b_2, b_3) and think of them as points in a room with three axes at right angles. As the number of dimensions (attributes) increases it rapidly becomes impossible to visualise them, at least for anyone who is not a physicist (and most of those who are).

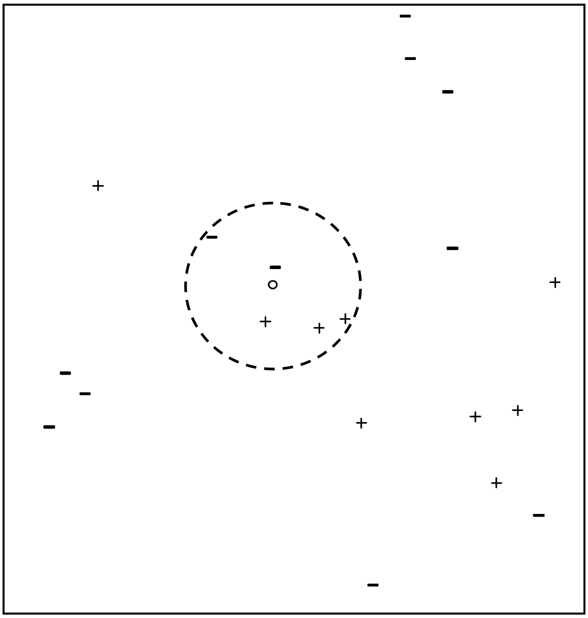

Figure 3.6 Two-dimensional Representation of Training Data in Figure 3.5

When there are n attributes, we can represent the instances by the points $(a_1, a_2, \ldots, a_n)$ and $(b_1, b_2, \ldots, b_n)$ in 'n-dimensional space'.

3.3.1 Distance Measures

There are many possible ways of measuring the distance between two instances with n attribute values, or equivalently between two points in n-dimensional space. We usually impose three requirements on any distance measure we use. We will use the notation $\mathbf{dist}(X, Y)$ to denote the distance between two points X and Y.

1. The distance of any point A from itself is zero, i.e. $\mathbf{dist}(A, A) = 0$.
2. The distance from A to B is the same as the distance from B to A, i.e. $\mathbf{dist}(A, B) = \mathbf{dist}(B, A)$ (the *symmetry condition*).

The third condition is called the *triangle inequality* (Figure 3.7). It corresponds to the intuitive idea that 'the shortest distance between any two points is a straight line'. The condition says that for any points A, B and Z: $\mathbf{dist}(A, B) \leq \mathbf{dist}(A, Z) + \mathbf{dist}(Z, B)$.

As usual, it is easiest to visualise this in two dimensions.

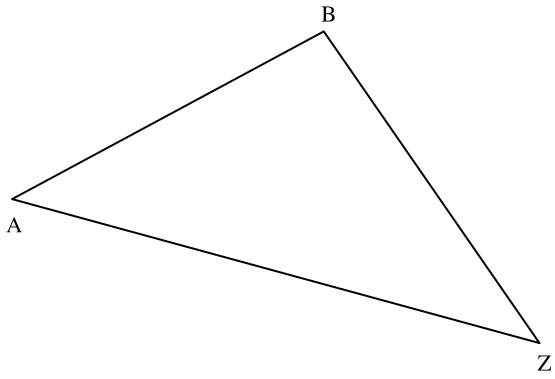

Figure 3.7 The Triangle Inequality

The equality only occurs if Z is the same point as A or B or is on the direct route between them.

There are many possible distance measures, but the most popular is almost certainly the *Euclidean Distance* (Figure 3.8). This measure is named after the Greek Mathematician Euclid of Alexandria, who lived around 300 BC and is celebrated as the founder of geometry. It is the measure of distance assumed in Figure 3.6.

We will start by illustrating the formula for Euclidean distance in two dimensions. If we denote an instance in the training set by (a_1, a_2) and the unseen instance by (b_1, b_2) the length of the straight line joining the points is

$$\sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$$

by Pythagoras' Theorem.

If there are two points (a_1, a_2, a_3) and (b_1, b_2, b_3) in a three-dimensional space the corresponding formula is

$$\sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + (a_3 - b_3)^2}$$

The formula for Euclidean distance between points $(a_1, a_2, \ldots, a_n)$ and $(b_1, b_2, \ldots, b_n)$ in n-dimensional space is a generalisation of these two results. The Euclidean distance is given by the formula

$$\sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_n - b_n)^2}$$

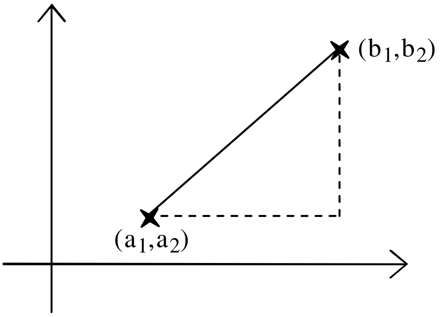

Figure 3.8 Example of Euclidean Distance

Another measure sometimes used is called *Manhattan Distance* or *City Block Distance*. The analogy is with travelling around a city such as Manhattan, where you cannot (usually) go straight from one place to another but only by moving along streets aligned horizontally and vertically.

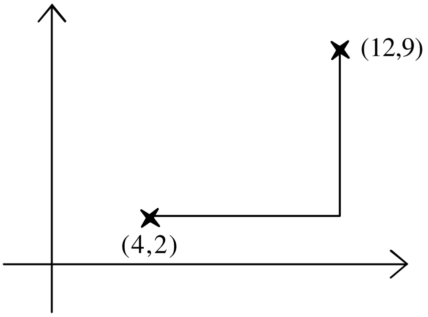

Figure 3.9 Example of City Block Distance

The City Block distance between the points $(4, 2)$ and $(12, 9)$ in Figure 3.9 is $(12 - 4) + (9 - 2) = 8 + 7 = 15$.

A third possibility is the *maximum dimension distance*. This is the largest absolute difference between any pair of corresponding attribute values. (The absolute difference is the difference converted to a positive number if it is negative.) For example the maximum dimension distance between instances

6.2	−7.1	−5.0	18.3	−3.1	8.9

and

8.3	12.4	−4.1	19.7	−6.2	12.4

is $12.4 - (-7.1) = 19.5$.

For many applications, Euclidean distance seems the most natural way of measuring the distance between two instances.

3.3.2 Normalisation

A major problem when using the Euclidean distance formula (and many other distance measures) is that the large values frequently swamp the small ones.

Suppose that two instances are as follows for some classification problem associated with cars (the classifications themselves are omitted).

Mileage (miles)	Number of doors	Age (years)	Number of owners
18,457	2	12	8
26,292	4	3	1

When the distance of these instances from an unseen one is calculated, the *mileage* attribute will almost certainly contribute a value of several thousands squared, i.e. several millions, to the sum of squares total. The number of doors will probably contribute a value less than 10. It is clear that in practice the only attribute that will matter when deciding which neighbours are the nearest using the Euclidean distance formula is the mileage. This is unreasonable as the unit of measurement, here the mile, is entirely arbitrary. We could have chosen an alternative measure of distance travelled such as millimetres or perhaps light years. Similarly we might have measured age in some other unit such as milliseconds or millennia. The units chosen should not affect the decision on which are the nearest neighbours.

To overcome this problem we generally *normalise* the values of continuous attributes. The idea is to make the values of each attribute run from 0 to 1. Suppose that for some attribute A the smallest value found in the training data is -8.1 and the largest is 94.3. First we adjust each value of A by adding 8.1 to it, so the values now run from 0 to $94.3+8.1 = 102.4$. The spread of values from highest to lowest is now 102.4 units, so we divide all values by that number to make the spread of values from 0 to 1.

In general if the lowest value of attribute A is *min* and the highest value is *max*, we convert each value of A, say a, to $(a - min)/(max - min)$.

Using this approach all continuous attributes are converted to small numbers from 0 to 1, so the effect of the choice of unit of measurement on the outcome is greatly reduced.

Note that it is possible that an unseen instance may have a value of A that is less than *min* or greater than *max*. If we want to keep the adjusted numbers in the range from 0 to 1 we can just convert any values of A that are less than *min* or greater than *max* to 0 or 1, respectively.

Another issue that occurs with measuring the distance between two points is the *weighting* of the contributions of the different attributes. We may believe that the mileage of a car is more important than the number of doors it has (although no doubt not a thousand times more important, as with the unnormalised values). To achieve this we can adjust the formula for Euclidean distance to

$$\sqrt{w_1(a_1-b_1)^2+w_2(a_2-b_2)^2+\ldots+w_n(a_n-b_n)^2}$$

where $w_1, w_2, \ldots, w_n$ are the weights. It is customary to scale the weight values so that the sum of all the weights is one.

3.3.3 Dealing with Categorical Attributes

One of the weaknesses of the nearest neighbour approach to classification is that there is no entirely satisfactory way of dealing with categorical attributes. One possibility is to say that the difference between any two identical values of the attribute is zero and that the difference between any two different values is 1. Effectively this amounts to saying (for a colour attribute) red − red = 0, red − blue = 1, blue − green = 1, etc.

Sometimes there is an ordering (or a partial ordering) of the values of an attribute, for example we might have values *good*, *average* and *bad*. We could treat the difference between *good* and *average* or between *average* and *bad* as 0.5 and the difference between *good* and *bad* as 1. This still does not seem completely right, but may be the best we can do in practice.

3.4 Eager and Lazy Learning

The Naïve Bayes and Nearest Neighbour algorithms described in Sections 3.2 and 3.3 illustrate two alternative approaches to automatic classification, known by the slightly cryptic names of *eager learning* and *lazy learning*, respectively.

In eager learning systems the training data is ‘eagerly’ generalised into some representation or model such as a table of probabilities, a decision tree or a neural net without waiting for a new (unseen) instance to be presented for classification.

In lazy learning systems the training data is 'lazily' left unchanged until an unseen instance is presented for classification. When it is, only those calculations that are necessary to classify that single instance are performed.

The lazy learning approach has some enthusiastic advocates, but if there are a large number of unseen instances, it can be computationally very expensive to carry out compared with eager learning methods such as Naïve Bayes and the other methods of classification that are described in later chapters.

A more fundamental weakness of the lazy learning approach is that it does not give any idea of the underlying causality of the task domain. This is also true of the probability-based Naïve Bayes eager learning algorithm, but to a lesser extent. X is the classification for no reason deeper than that if you do the calculations X turns out to be the answer. We now turn to methods that give an explicit way of classifying any unseen instance that can be used (and critiqued) independently from the training data used to generate it. We call such methods *model-based*.

3.5 Chapter Summary

This chapter introduces classification, one of the most common data mining tasks. Two classification algorithms are described in detail: the Naïve Bayes algorithm, which uses probability theory to find the most likely of the possible classifications, and Nearest Neighbour classification, which estimates the classification of an unseen instance using the classification of the instances 'closest' to it. These two methods generally assume that all the attributes are categorical and continuous, respectively.

3.6 Self-assessment Exercises for Chapter 3

1. Using the Naïve Bayes classification algorithm with the *train* dataset, calculate the most likely classification for the following unseen instances.

weekday	summer	high	heavy	????

sunday	summer	normal	slight	????

2. Using the training set shown in Figure 3.5 and the Euclidean distance measure, calculate the 5-nearest neighbours of the instance with first and second attributes 9.1 and 11.0, respectively.