10 More About Entropy

10.1 Introduction

In this chapter we return to the subject of the *entropy* of a training set, which was introduced in Chapter 5. The idea of entropy is not only used in data mining; it is a very fundamental one, which is widely used in Information Theory as the basis for calculating efficient ways of representing messages for transmission by telecommunication systems.

We will start by explaining what is meant by the entropy of a set of distinct values and then come back to look again at the entropy of a training set.

Suppose we are playing a game of the 'twenty questions' variety where we try to identify one of M possible values by asking a series of yes/no questions. The values in which we are really interested are mutually exclusive classifications of the kind discussed in Chapter 3 and elsewhere, but the same argument can be applied to any set of distinct values.

We will assume at present that all M values are equally likely and for reasons that will soon become apparent we will also assume that M is an exact power of 2, say 2^N, where $N \geq 1$.

As a concrete example we will take the task of identifying an unknown capital city from the eight possibilities: London, Paris, Berlin, Warsaw, Sofia, Rome, Athens and Moscow (here $M = 8 = 2^3$).

There are many possible ways of asking questions, for example random guessing:

M. Bramer, *Principles of Data Mining*, Undergraduate Topics in Computer Science, DOI 10.1007/978-1-4471-7307-6_10

Is it Warsaw? *No*
Is it Berlin? *No*
Is it Rome? *Yes*

This works well if the questioner makes a lucky guess early on, but (unsurprisingly) it is inefficient in the general case. To show this, imagine that we make our guesses in the fixed order: London, Paris, Berlin etc. until we guess the correct answer. We never need guess further than Athens, as a 'no' answer will tell us the city must be Moscow.

If the city is London, we need 1 question to find it.
If the city is Paris, we need 2 questions to find it.
If the city is Berlin, we need 3 questions to find it.
If the city is Warsaw, we need 4 questions to find it.
If the city is Sofia, we need 5 questions to find it.
If the city is Rome, we need 6 questions to find it.
If the city is Athens, we need 7 questions to find it.
If the city is Moscow, we need 7 questions to find it.

Each of these possibilities is equally likely, i.e. has probability 1/8, so on average we need $(1 + 2 + 3 + 4 + 5 + 6 + 7 + 7)/8$ questions, i.e. $35/8 = 4.375$ questions.

A little experiment will soon show that the best strategy is to keep dividing the possibilities into equal halves. Thus we might ask

Is it London, Paris, Athens or Moscow? *No*
Is it Berlin or Warsaw? *Yes*
Is it Berlin?

Whether the third question is answered yes or no, the answer will tell us the identity of the 'unknown' city.

The halving strategy always takes three questions to identify the unknown city. It is considered to be the 'best' strategy not because it invariably gives us the answer with the smallest number of questions (random guessing will occasionally do better) but because if we conduct a long series of 'trials' (each a game to guess a city, selected at random each time) the halving strategy will invariably find the answer and will do so with a smaller number of questions on average than any other strategy. With this understanding we can say that the smallest number of yes/no questions needed to determine an unknown value from 8 equally likely possibilities is three.

It is no coincidence that 8 is 2^3 and the smallest number of yes/no questions needed is 3. If we make the number of possible values M a higher or a lower power of two the same occurs. If we start with 8 possibilities and halve the number by the first question, that leaves 4 possibilities. We can determine the unknown value with 2 further questions. If we start with 4 possibilities and halve the number down to 2 by the first question we can determine the unknown value by just one further question ('is it the first one?'). So for $M = 4$ the smallest number of questions is 2 and for $M = 2$ the smallest number of questions is 1.

We can extend the argument to look at higher values of M, say 16. It takes one 'halving' question to reduce the number of possibilities to 8, which we know we can handle with 3 further questions. So the number of questions needed in the case of 16 values ($M = 16$) must be 4.

In general, we have the following result. **The smallest number of yes/no questions needed to determine an unknown value from $M = 2^N$ equally likely possibilities is N.**

Using the mathematical function $\log_2$,[1] we can rewrite the last result as: the smallest number of yes/no questions needed to determine an unknown value from M equally likely possibilities is $\log_2 M$ (provided M is a power of 2; see Figure 10.1).

M	$\log_2 M$
2	1
4	2
8	3
16	4
32	5
64	6
128	7
256	8
512	9
1024	10

Figure 10.1 Some values of $\log_2 M$ (where M is a power of 2)

We will define a quantity called the **entropy** of a set of M distinct values as follows.

The **entropy** of a set of M distinct values that are equally likely is the smallest number of yes/no questions needed to determine an unknown value drawn from the M possibilities. As before, the words 'in all cases' are implicit and by smallest we mean the smallest number of questions averaged over a series of trials, not just one single trial (game).

[1] The $\log_2$ function is defined in Appendix A for readers who are unfamiliar with it.

In the phrase 'the smallest number of yes/no questions needed' in the definition of entropy, it is implicit that each question needs to divide the remaining possibilities into two equally probable halves. If they do not, for example with random guessing, a larger number will be needed.

It is not sufficient that each question looked at in isolation is a 'halving question'. For example, consider the sequence

Is it Berlin, London, Paris or Warsaw? *Yes*
Is it Berlin, London, Paris or Sofia? *Yes*

Both questions are 'halving questions' in their own right, but the answers leave us after two questions still having to discriminate amongst three possibilities, which cannot be done with one more question.

It is not sufficient that each question asked is a halving question. It is necessary to find a *sequence* of questions that take full advantage of the answers already given to divide the remaining possibilities into two equally probable halves. We will call this a 'well-chosen' sequence of questions.

So far we have established that the entropy of a set of M distinct values is $\log_2 M$, provided that M is a power of 2 and all values are equally likely. We have also established the need for questions to form a 'well-chosen' sequence. This raises three questions:

- What if M is not a power of 2?
- What if the M possible values are not equally likely?
- Is there a systematic way of finding a sequence of well-chosen questions?

It will be easier to answer these questions if we first introduce the idea of coding information using bits.

10.2 Coding Information Using Bits

There is an obvious everyday sense in which the more questions that are answered the more *information* we have. We can formalise this by saying that the answer to a question that can only be answered yes or no (with equal probability) can be considered as containing one *unit of information*. The basic unit of information is called a *bit* (short for 'binary digit'). This usage of the word 'bit' has a close connection with its use for the basic unit of storage in computer memory. It is a fundamental two-valued unit that corresponds to a switch being open or closed, a light being on or off, an electric current flowing or not flowing, or the dot and dash of Morse code.

The unit of information can also be looked at as the amount of information that can be *coded* using only a zero or a one. If we have two possible values, say male and female, we might use the coding

0 =	male
1 =	female

We can encode four possible values (say: man, woman, dog, cat) using two bits, e.g.

00 =	man
01 =	woman
10 =	dog
11 =	cat

To code eight values, say the eight capital cities, we need to use three bits, for example

000 =	London
001 =	Paris
010 =	Berlin
011 =	Warsaw
100 =	Sofia
101 =	Rome
110 =	Athens
111 =	Moscow

Coding the 2^N equally likely possibilities with N binary digits shows that it is always possible to discriminate amongst the values with a sequence of N well-chosen questions, for example:

Is the first bit zero?
Is the second bit zero?
Is the third bit zero?
and so on.

This leads to the following alternative (and equivalent) definition of entropy:

The **entropy** of a set of M distinct values is the number of bits needed to encode the values in the most efficient way.

As for the previous definition, the words ‘in all cases’ are implicit and by ‘the most efficient way’ we mean the smallest number of bits averaged over a series of trials, not just one single trial. This second definition also explains why the entropy is often given not as a number but as so many ‘bits of information’.

10.3 Discriminating Amongst M Values (M Not a Power of 2)

So far we have established that the entropy of a set of M distinct values that are equally likely is $\log_2 M$ for cases where M is a power of 2. We now need to consider the case when it is not.

Is there any sense in which we can say that the entropy is $\log_2 M$ bits of information? We cannot have a non-integer number of questions or encode with a non-integer number of bits.

To answer this we need to think of identifying not just one value out of M possibilities but a sequence of k such values (each one chosen independently of the others). We will denote the smallest number of yes/no questions needed to determine a sequence of k unknown values drawn independently from M possibilities, i.e. the entropy, by V_{kM}. This is the same as the number of questions needed to discriminate amongst M^k distinct possibilities.

To take a concrete example, say M is 7 and k is 6 and the task is to identify a sequence of six days of the week, for example {Tuesday, Thursday, Tuesday, Monday, Sunday, Tuesday}. A possible question might be

Is the first day Monday, Tuesday or Wednesday
and the second day Thursday
and the third day Monday, Saturday, Tuesday or Thursday
and the fourth day Tuesday, Wednesday or Friday
and the fifth day Saturday or Monday
and the sixth day Monday, Sunday or Thursday?

There are $7^6 = 117649$ possible sequences of six days. The value of $\log_2 117649$ is 16.84413. This is between 16 and 17 so to determine any possible value of a sequence of 6 days of the week would take 17 questions. The average number of questions for each of the six days of the week is $17/6 = 2.8333$. This is reasonably close to $\log_2 7$, which is approximately 2.8074.

A better approximation to the entropy is obtained by taking a larger value of k, say 21. Now $\log_2 M^k$ is $\log_2(7^{21}) = 58.95445$, so 59 questions are needed for the set of 21 values, making an average number of questions per value of $59/21 = 2.809524$.

Finally, for a set of 1000 values ($k = 1000$), $\log_2 M^k$ is $\log_2(7^{1000}) = 2807.3549$, so 2808 questions are needed for the set of 1000 values, making an average per value of 2.808, which is very close to $\log_2 7$.

It is not a coincidence that these values appear to be converging to $\log_2 7$, as is shown by the following argument for the general case of sequences of length k drawn from M distinct equally likely values.

There are M^k possible sequences of k values. Assuming now that M is *not* a power of 2, the number of questions needed, V_{kM} is the next integer above $\log_2 M^k$. We can put lower and upper bounds on the value of V_{kM} by the relation

$$\log_2 M^k \leq V_{kM} \leq \log_2 M^k + 1$$

Using the property of logarithms that $\log_2 M^k = k \log_2 M$ leads to the relation

$$k \log_2 M \leq V_{kM} \leq k \log_2 M + 1$$

so $\log_2 M \leq V_{kM}/k \leq \log_2 M + 1/k$.

V_{kM}/k is the average number of questions needed to determine each of the k values. By choosing a large enough value of k, i.e. a long enough sequence, the value of $1/k$ can be made as small as we wish. Thus the average number of questions needed to determine each value can be made arbitrarily close to $\log_2 M$. Thus the entropy of a set of M distinct values can be said to be $\log_2 M$, even when M is not a power of 2 (see Figure 10.2).

M	$\log_2 M$
2	1
3	1.5850
4	2
5	2.3219
6	2.5850
7	2.8074
8	3
9	3.1699
10	3.3219

Figure 10.2 $\log_2 M$ for M from 2 to 10

10.4 Encoding Values That Are Not Equally Likely

We finally come to the general case of encoding M distinct values that are not equally likely. (We assume that values that never occur are not included.)

When M possible values are equally likely the entropy has previously been shown to be $\log_2 M$. When M values are unequally distributed the entropy will always have a lower value than $\log_2 M$. In the extreme case where only one value ever occurs, there is no need to use even one bit to represent the value and the entropy is zero.

We will write the frequency with which the ith of the M values occurs as p_i where i varies from 1 to M. Then we have $0 \le p_i \le 1$ for all p_i and
$\sum_{i=1}^{i=M} p_i = 1$.

For convenience we will give an example where all the p_i values are the reciprocal of an exact power of 2, i.e. 1/2, 1/4 or 1/8, but the result obtained can be shown to apply for other values of p_i using an argument similar to that in Section 10.3.

Suppose we have four values A, B, C and D which occur with frequencies 1/2, 1/4, 1/8 and 1/8 respectively. Then $M = 4$, $p_1 = 1/2$, $p_2 = 1/4$, $p_3 = 1/8$, $p_4 = 1/8$.

When representing A, B, C and D we could use the standard 2-bit encoding described previously, i.e.

A	10
B	11
C	00
D	01

However, we can improve on this using a *variable length encoding*, i.e. one where the values are not always represented by the same number of bits. There are many possible ways of doing this. The best way turns out to be the one shown in Figure 10.3.

A	1
B	01
C	001
D	000

Figure 10.3 Most Efficient Representation for Four Values with Frequencies 1/2, 1/4, 1/8 and 1/8

If the value to be identified is A, we need examine only one bit to establish this. If it is B we need to examine two bits. If it is C or D we need to examine 3 bits. In the average case we need to examine $1/2 \times 1 + 1/4 \times 2 + 1/8 \times 3 + 1/8 \times 3 =$ 1.75 bits.

This is the most efficient representation. Flipping some or all of the bits consistently will give other equally efficient representations that are obviously equivalent to it, such as

A	0
B	11
C	100
D	101

Any other representation will require more bits to be examined on average. For example we might choose

A	01
B	1
C	001
D	000

With this representation, in the average case we need to examine $1/2 \times 2 + 1/4 \times 1 + 1/8 \times 3 + 1/8 \times 3 = 2$ bits (the same as the number for the fixed length representation).

Some other representations, such as

A	101
B	0011
C	10011
D	100001

are much worse than the 2-bit representation. This one requires $1/2 \times 3 + 1/4 \times 4 + 1/8 \times 5 + 1/8 \times 6 = 3.875$ bits to be examined on average.

The key to finding the most efficient coding is to use a string of N bits to represent a value that occurs with frequency $1/2^N$. Writing this another way, represent a value that occurs with frequency p_i by a string of $\log_2(1/p_i)$ bits (see Figure 10.4).

p_i	$\log_2(1/p_i)$
1/2	1
1/4	2
1/8	3
1/16	4

Figure 10.4 Values of $\log_2(1/p_i)$

This method of coding ensures that we can determine any value by asking a sequence of 'well-chosen' yes/no questions (i.e. questions for which the two possible answers are equally likely) about the value of each of the bits in turn.

Is the first bit 1?
If not, is the second bit 1?

If not, is the third bit 1?
etc.

So in Figure 10.3 value A, which occurs with frequency 1/2 is represented by 1 bit, value B which occurs with frequency 1/4 is represented by 2 bits and values C and D are represented by 3 bits each.

If there are M values with frequencies p_1, p_2, ..., p_M the average number of bits that need to be examined to establish a value, i.e. the entropy, is the frequency of occurrence of the ith value multiplied by the number of bits that need to be examined if that value is the one to be determined, summed over all values of i from 1 to M. Thus we can calculate the value of entropy E by

$$E = \sum_{i=1}^{M} p_i \log_2(1/p_i)$$

This formula is often given in the equivalent form

$$E = -\sum_{i=1}^{M} p_i \log_2(p_i)$$

There are two special cases to consider. When all the values of p_i are the same, i.e. $p_i = 1/M$ for all values of i from 1 to M, the above formula reduces to

$$\begin{aligned} E &= -\sum_{i=1}^{M} (1/M)\log_2(1/M) \\ &= -\log_2(1/M) \\ &= \log_2 M \end{aligned}$$

which is the formula given in Section 10.3.

When there is only one value with a non-zero frequency, $M = 1$ and $p_1 = 1$, so $E = -1 \times \log_2 1 = 0$.

10.5 Entropy of a Training Set

We can now link up the material in this chapter with the definition of the entropy of a training set given in Chapter 5. In that chapter the formula for entropy was simply stated without motivation. We can now see the entropy of a training set in terms of the number of yes/no questions needed to determine an unknown classification.

If we know that the entropy of a training set is E, it does not imply that we can find an unknown classification with E 'well-chosen' yes/no questions. To do so we would have to ask questions about the classification itself, e.g. 'Is the classification A or B, rather than C or D?' Obviously we cannot find a way of predicting the classification of an unseen instance by asking questions of this kind. Instead we ask a series of questions about the value of a set of attributes measured for each of the instances in a training set, which collectively determine the classification. Sometimes only one question is necessary, sometimes many more.

Asking any question about the value of an attribute effectively divides the training set into a number of subsets, one for each possible value of the attribute (any empty subsets are discarded). The TDIDT algorithm described in Chapter 4 generates a decision tree from the top down by repeatedly splitting on the values of attributes. If the training set represented by the root node has M possible classifications, each of the subsets corresponding to the end nodes of each branch of the developing tree has an entropy value that varies from $\log_2 M$ (if the frequencies of each of the classifications in the subset are identical) to zero (if the subset has attributes with only one classification).

When the splitting process has terminated, all the 'uncertainty' has been removed from the tree. Each branch corresponds to a combination of attribute values and for each branch there is a single classification, so the overall entropy is zero.

Although it is possible for a subset created by splitting to have an entropy greater than its 'parent', at every stage of the process splitting on an attribute reduces the average entropy of the tree or at worst leaves it unchanged. This is an important result, which is frequently assumed but seldom proved. We will consider it in the next section.

10.6 Information Gain Must Be Positive or Zero

The Information Gain attribute selection criterion was described in Chapter 5. Because of its name, it is sometimes assumed that Information Gain must always be positive, i.e. information is always gained by splitting on a node during the tree generation process.

However this is not correct. Although it is generally true that information gain is positive it is also possible for it to be zero. The following demonstration that information gain can be zero is based on the principle that for C possible classifications, the entropy of a training set takes the value $\log_2 C$ (its largest possible value) when the classes are balanced, i.e. there are the same number of instances belonging to each of the classes.

The training set shown in Figure 10.5 has two equally balanced classes.

The probability of each class is 0.5, so we have

$$E_{start} = -(1/2)\log_2(1/2) - (1/2)\log_2(1/2) = -\log_2(1/2) = \log_2(2) = 1$$

X	Y	Class
1	1	A
1	2	B
2	1	A
2	2	B
3	2	A
3	1	B
4	2	A
4	1	B

Figure 10.5 Training Set for 'Information Gain Can be Zero' Example

This is the value of $\log_2 C$ for $C = 2$ classes.

The training set has been constructed to have the property that whichever attribute is chosen for splitting, each of the branches will also be balanced.

For splitting on attribute X the frequency table is shown in Figure 10.6(a).

	Attribute value			
Class	1	2	3	4
A	1	1	1	1
B	1	1	1	1
Total	2	2	2	2

Figure 10.6(a) Frequency Table for Attribute X

Each column of the frequency table is balanced and it can easily be verified that $E_{new} = 1$.

For splitting on attribute Y the frequency table is shown in Figure 10.6(b).

	Attribute value	
Class	1	2
A	2	2
B	2	2
Total	4	4

Figure 10.6(b) Frequency Table for Attribute Y

Again both columns are balanced and $E_{new} = 1$. Whichever value is taken, E_{new} is 1 and so the Information Gain $= E_{start} - E_{new} = 0$.

The absence of information gain does not imply that there is no value in splitting on either of the attributes. Whichever one is chosen, splitting on the other attribute for all the resulting branches will produce a final decision tree with each branch terminated by a leaf node and thus having an entropy of zero.

Although we have shown that Information Gain can sometimes be zero, it can never be negative. Intuitively it would seem wrong for it to be possible to lose information by splitting on an attribute. Surely that can only give more information (or occasionally the same amount)?

The result that Information Gain can never be negative is stated by many authors and implied by others. The name Information *Gain* gives a strong suggestion that information loss would not be possible, but that is far from being a formal proof.

The present author's inability to locate a proof of this crucial result led him to issue a challenge to several British academics to find a proof in the technical literature or generate one themselves. An excellent response to this came from two members of the University of Ulster in Northern Ireland who produced a detailed proof of their own [1]. The proof is too difficult to reproduce here but is well worth obtaining and studying in detail.

10.7 Using Information Gain for Feature Reduction for Classification Tasks

We conclude this chapter by looking at a further use for entropy, in the form of Information Gain, this time as a means of reducing the number of features (i.e. attributes) that a classification algorithm (of any kind) needs to consider.

The method of feature reduction described here is specific to classification tasks. It uses information gain, which was introduced in Chapter 5 as a criterion for selecting attributes at each stage of the TDIDT tree generation algorithm. However for purposes of feature reduction, information gain is applied at the top level only as an initial pre-processing stage. Only the attributes meeting a specified criterion are retained for use by the classification algorithm. There is no assumption that the classification algorithm used is TDIDT. It can potentially be any algorithm.

Broadly the method amounts to asking for each attribute in turn 'how much information is gained about the classification of an instance by knowing the value of this attribute?' Only the attributes with the largest values of information gain are retained for use with the preferred classification algorithm. There are three stages.

```
1. Calculate the value of information gain for each attribute in the original
dataset.
2. Discard all attributes that do not meet a specified criterion.
3. Pass the revised dataset to the preferred classification algorithm.
```

The method of calculating information gain for categorical attributes using frequency tables was described in Chapter 6. A modification that enables the method to be used for continuous attributes by examining alternative ways of splitting the attribute values into two parts was described in Chapter 8. The latter also returns a 'split value', i.e. the value of the attribute that gives the largest information gain. This value is not needed when information gain is used for feature reduction. It is sufficient to know the largest information gain achievable for the attribute with any split value.

There are many possible criteria that can be used for determining which attributes to retain, for example:

– Only retain the best 20 attributes

– Only retain the best 25% of the attributes

– Only retain attributes with an information gain that is at least 25% of the highest information gain of any attribute

– Only retain attributes that reduce the initial entropy of the dataset by at least 10%.

There is no one choice that is best in all situations, but analysing the information gain values of all the attributes can help make an informed choice.

10.7.1 Example 1: The *genetics* Dataset

As an example we will consider the *genetics* dataset, which is available from the UCI Repository. Some basic information about this is given in Figure 10.7.

Although 60 attributes is hardly a large number, it may still be more than is needed for reliable classification and is large enough to make overfitting a realistic possibility.

There are three classifications, distributed 767, 768 and 1655 amongst the three classes for the 3190 instances. The relative proportions are 0.240, 0.241 and 0.519, so the initial entropy is: $-0.240 \times \log_2(0.240) - 0.241 \times \log_2(0.241) - 0.519 \times \log_2(0.519) = 1.480$.

The values of information gain for some of the attributes A0 to A59 are shown in Figure 10.8.

The *genetics* Dataset: Basic Information

The *genetics* dataset contains 3190 instances. Each instance comprises the values of a sequence of 60 DNA elements and is classified into one of three possible categories: EI, IE and N. Each of the 60 attributes (named A0 to A59) is categorical and has 8 possible values: A, T, G, C, N, D, S and R.

For further information see [2].

Figure 10.7 *genetics* Dataset: Basic Information

Attribute	Information Gain
A0	0.0062
A1	0.0066
A2	0.0024
A3	0.0092
A4	0.0161
A5	0.0177
A6	0.0077
A7	0.0071
A8	0.0283
A9	0.0279
......	......
A27	0.2108
A28	0.3426
A29	0.3896
A30	0.3296
A31	0.3322
......	......
A57	0.0080
A58	0.0041
A59	0.0123

Figure 10.8 *genetics* Dataset: Information Gain for Some of the Attributes

The largest information gain is for A29. A gain of 0.3896 implies that the initial entropy would be reduced by more than a quarter if the value of A29 were known. The second largest information gain is for attribute A28.

Comparing values written as decimals to four decimal places is awkward (for people). It is probably easier to make sense of this table if it is adjusted by dividing all the information gain values by 0.3896 (the largest value), making a proportion from 0 to 1, and then multiplying them all by 100. The resulting values are given in Figure 10.9. An adjusted information gain of 1.60 for attribute A0 means that the information gain for A0 is 1.60% of the size of the largest value, which was the one obtained for A29.

Attribute	Info. Gain (adjusted)
A0	1.60
A1	1.70
A2	0.61
A3	2.36
A4	4.14
A5	4.55
A6	1.99
A7	1.81
A8	7.27
A9	7.17
......	......
A27	54.09
A28	87.92
A29	100.00
A30	84.60
A31	85.26
......	......
A57	2.07
A58	1.05
A59	3.16

Figure 10.9 *genetics* Dataset: Information Gain as Percentage of Largest Value

From this table it is clear that not only is the information gain for A29 the largest, it is considerably larger than most of the other values. Only a small number of other information gain values are even 50% as large.

Another way of looking at the information gain values is to consider *frequencies*. We can divide the range of possible adjusted values (0 to 100% in this case) into a number of ranges, generally known as *bins*. These might be labelled 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100. (It is not essential for the bins to be equally spaced.)

Each of the information gain values is then assigned to one of the bins. The first bin corresponds to values from 0 to 10 inclusive, the second bin corresponds to values greater than 10 but less than or equal to 20, and so on.

The frequency for each of the 10 bins is shown in Figure 10.10. The final two columns show the *cumulative frequency* (i.e. the number of values that are less than or equal to the bin label) and the cumulative frequency expressed as a percentage of the total number of values (i.e. 60).

Bin	Frequency	Cumulative frequency	Cumulative frequency (%)
10	41	41	68.33
20	9	50	83.33
30	2	52	86.67
40	2	54	90.00
50	0	54	90.00
60	2	56	93.33
70	0	56	93.33
80	0	56	93.33
90	3	59	98.33
100	1	60	100.00
Total	60		

Figure 10.10 *genetics* Dataset: Information Gain Frequencies

As many as 41 of the 60 attributes have an information gain that is no more than 10% as large as that of A29. Only six attributes have an information gain that is more than 50% of that of A29.

It is tempting to discard all but the best six attributes. Although this is not necessarily the best policy, it is interesting to look at the change in predictive accuracy that results if we do.

Using TDIDT with the entropy attribute selection criterion for classification, the predictive accuracy obtained using 10-fold cross-validation is 89.5% when all 60 attributes are used. This increases to 91.8% when only the best six attributes are used. Although this improvement is quite small, it certainly is an improvement and is obtained using only 6 out of the original 60 attributes.

10.7.2 Example 2: The *bcst96* Dataset

The next example makes use of a much larger dataset. The dataset *bcst96* has been used for experiments on automatic classification of web pages. Some basic information about it is given in Figure 10.11.

The *bcst96* Dataset: Basic Information

The *bcst96* dataset comprises 1186 instances (training set) and a further 509 instances (test set). Each instance corresponds to a web page, which is classified into one of two possible categories, B or C, using the values of 13,430 attributes, all continuous.
There are 1,749 attributes that each have only a single value for the instances in the training set and so can be deleted, leaving 11,681 continuous attributes.

Figure 10.11 *bcst96* Dataset: Basic Information

In this case the original number of attributes is more than 11 times as large as the number of instances in the training set. It seems highly likely that a large number of the attributes could safely be deleted, but which ones?

The initial value of entropy is 0.996, indicating that the two classes are fairly equally balanced.

As can be seen in Figure 10.11, having deleted the attributes that have a single value for all instances in the training set, there are 11,681 continuous attributes remaining.

Next we calculate the information gain for each of these 11,681 attributes. The largest value is 0.381.

The frequency table is shown in Figure 10.12.

The most surprising result is that as many as 11,135 of the attributes (95.33%) have an information gain in the 5 bin, i.e. no more than 5% of the largest information gain available. Almost 99% of the values are in the 5 and 10 bins.

Using TDIDT with the entropy attribute selection criterion for classification, the algorithm generates 38 rules from the original training set and uses these to predict the classification of the 509 instances in the test set. It does this with 94.9% accuracy (483 correct and 26 incorrect predictions). If we discard all but the best 50 attributes, the same algorithm generates a set of 62 rules, which again give 94.9% predictive accuracy on the test set (483 correct and 26 incorrect predictions).

Bin	Frequency	Cumulative frequency	Cumulative frequency (%)
5	11,135	11,135	95.33
10	403	11,538	98.78
15	76	11,614	99.43
20	34	11,648	99.72
25	10	11,658	99.80
30	7	11,665	99.86
35	4	11,669	99.90
40	1	11,670	99.91
45	2	11,672	99.92
50	1	11,673	99.93
55	1	11,674	99.94
60	2	11,676	99.96
65	2	11,678	99.97
70	0	11,678	99.97
75	1	11,679	99.98
80	0	11,679	99.98
85	1	11,680	99.99
90	0	11,680	99.99
95	0	11,680	99.99
100	1	11,681	100.00
Total	11,681		

Figure 10.12 *bcst96* Dataset: Information Gain Frequencies

In this case just 50 out of 11,681 attributes (less than 0.5%) suffice to give the same predictive accuracy as the whole set of attributes. However, the difference in the amount of processing required to produce the two decision trees is considerable. With all the attributes the TDIDT algorithm will need to examine approximately $1,186 \times 11,681 = 13,853,666$ attribute values at each node of the evolving decision tree. If only the best 50 attributes are used the number drops to just $1,186 \times 50 = 59,300$.

Although feature reduction cannot always be guaranteed to produce results as good as those in these two examples, it should always be considered, especially when the number of attributes is large.

10.8 Chapter Summary

This chapter returns to the subject of the entropy of a training set. It explains the concept of entropy in detail using the idea of coding information using bits. The important result that when using the TDIDT algorithm information gain must be positive or zero is discussed, followed by the use of information gain as a method of feature reduction for classification tasks.

10.9 Self-assessment Exercises for Chapter 10

1. What is the entropy of a training set of 100 instances with four classes that occur with relative frequencies 20/100, 30/100, 25/100 and 25/100? What is the entropy of a training set of 10,000 instances with those frequencies for its four classes?
2. Given the task of identifying an unknown person in a large group using only yes/no questions, which question is it likely to be best to ask first?

References

[1] McSherry, D., & Stretch, C. (2003). *Information gain* (University of Ulster Technical Note).

[2] Noordewier, M. O., Towell, G. G., & Shavlik, J. W. (1991). Training knowledge-based neural networks to recognize genes in DNA sequences. In *Advances in neural information processing systems* (Vol. 3). San Mateo: Morgan Kaufmann.