manipulate data using mathematical operations, we need to have a mathematical model for the data. Obviously, the better the model (i.e., the closer the model matches the aspects of reality that are of interest to us), the more likely it is that we will come up with a satisfactory technique. There are several approaches to building mathematical models.

2.3.1 Physical Models

If we know something about the physics of the data generation process, we can use that information to construct a model. For example, in speech-related applications, knowledge about the physics of speech production can be used to construct a mathematical model for the sampled speech process. Sampled speech can then be encoded using this model. We will discuss speech production models in more detail in Chapter 7.

Models for certain telemetry data can also be obtained through knowledge of the underlying process. For example, if residential electrical meter readings at hourly intervals were to be coded, knowledge about the living habits of the populace could be used to determine when electricity usage would be high and when the usage would be low. Then instead of the actual readings, the difference (residual) between the actual readings and those predicted by the model could be coded.

In general, however, the physics of data generation is simply too complicated to understand, let alone use to develop a model. Where the physics of the problem is too complicated, we can obtain a model based on empirical observation of the statistics of the data.

2.3.2 Probability Models

The simplest statistical model for the source is to assume that each letter that is generated by the source is independent of every other letter, and each occurs with the same probability. We could call this the *ignorance model*, as it would generally be useful only when we know nothing about the source. (Of course, that *really* might be true, in which case we have a rather unfortunate name for the model!) The next step up in complexity is to keep the independence assumption, but remove the equal probability assumption and assign a probability of occurrence to each letter in the alphabet. For a source that generates letters from an alphabet $\mathcal{A} = \{a_1, a_2, \ldots, a_M\}$, we can have a *probability model* $\mathcal{P} = \{P(a_1), P(a_2), \ldots, P(a_M)\}$.

Given a probability model (and the independence assumption), we can compute the entropy of the source using Equation (2.4). As we will see in the following chapters using the probability model, we can also construct some very efficient codes to represent the letters in $\mathcal{A}$. Of course, these codes are only efficient if our mathematical assumptions are in accord with reality.

If the assumption of independence does not fit with our observation of the data, we can generally find better compression schemes if we discard this assumption. When we discard the independence assumption, we have to come up with a way to describe the dependence of elements of the data sequence on each other.

2.3.3 Markov Models

One of the most popular ways of representing dependence in the data is through the use of Markov models (named after the Russian mathematician A.A. Markov). For models used in lossless compression, we use a specific type of Markov process called a *discrete time Markov chain.* Let $\{x_n\}$ be a sequence of observations. This sequence is said to follow a kth-order Markov model if

$$P(x_n|x_{n-1},\ldots,x_{n-k}) = P(x_n|x_{n-1},\ldots,x_{n-k},\ldots). \tag{2.5}$$

In other words, knowledge of the past k symbols is equivalent to the knowledge of the entire past history of the process. The values taken on by the set $\{x_{n-1},\ldots,x_{n-k}\}$ are called the *states* of the process. If the size of the source alphabet is l, then the number of states is l^k. The most commonly used Markov model is the first-order Markov model, for which

$$P(x_n|x_{n-1}) = P(x_n|x_{n-1},x_{n-2},x_{n-3},\ldots). \tag{2.6}$$

The relationships defined by Equations (2.5) and (2.6) indicate the existence of dependence between samples. Exactly how this dependence was introduced is not made explicit. We can develop different first-order Markov models depending on our assumption about how the dependence between samples was introduced.

If we assumed that the dependence was introduced in a linear manner, we could view the data sequence as the output of a linear filter driven by white noise. The output of such a filter can be given by the difference equation

$$x_n = \rho x_{n-1} + \epsilon_n \tag{2.7}$$

where ϵ_n is a white noise process. This model is often used when developing coding algorithms for speech and images.

The use of the Markov model does not require the assumption of linearity. For example, consider a binary image. The image has only two types of pixels, white pixels and black pixels. We know that the appearance of a white pixel as the next observation depends, to some extent, on whether the current pixel is white or black. Therefore, we can model the pixel process as a discrete time Markov chain. Define two states S_w and S_b (S_w would correspond to the case where the current pixel is a white pixel and S_b corresponds to the case where the current pixel is a black pixel). We define the transition probabilities $P(w|b)$ and $P(b|w)$, and the probability of being in each state $P(S_w)$ and $P(S_b)$. The Markov model can then be represented by the state diagram shown in Figure 2.1.

The entropy of a finite state process with states S_i is simply the average value of the entropy at each state.

$$H = \sum_{i=1}^{M} P(S_i)H(S_i) \tag{2.8}$$

For our particular example of a binary image

$$H(S_w) = -P(b|w)\log P(b|w) - P(w|w)\log P(w|w)$$

where $P(w|w) = 1 - P(b|w)$. $H(S_b)$ can be calculated in a similar manner.

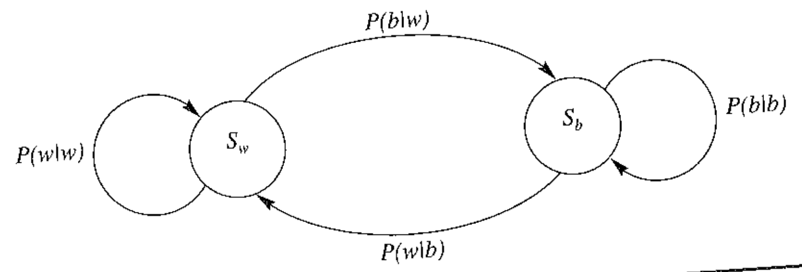

FIGURE 2.1 A two-state Markov model for binary images.

Example 2.3.1: Markov model

To see the effect of modeling on the estimate of entropy, let us calculate the entropy for a binary image, first using a simple probability model and then using the finite state model described above. Let us assume the following values for the various probabilities:

$$P(S_w) = 0.8 \quad P(S_b) = 0.2 \quad P(w|b) = 0.3 \quad P(b|w) = 0.01.$$

Then the entropy using a probability model and the *iid* assumption is

$$H = -0.8\log 0.8 - 0.2\log 0.2 = 0.722 \text{ bits}.$$

Now using the Markov model

$$H(S_b) = -0.3\log 0.3 - 0.7\log 0.7 = 0.881 \text{ bits}$$

and

$$H(S_w) = -0.01\log 0.01 - 0.99\log 0.99 = 0.081 \text{ bits}$$

Using (2.8), this results in an entropy for the Markov model of 0.241 bits, which is about a third of the entropy obtained using the *iid* assumption. ◆

Markov Models in Text Compression

As expected, Markov models are particularly useful in text compression, where the probability of the next letter is heavily influenced by the preceding letters. In fact, the use of Markov models for written English appears in the original work of Shannon [191]. In current text compression literature, the kth-order Markov models are more widely known as *finite context models*, with the word *context* being used for what we have earlier defined as state.

Consider the word *preceding*. Suppose we have already processed *precedin* and are going to encode the next letter. If we take no account of the context and treat each letter as a surprise, the probability of the letter g occurring is relatively low. If we use a first-order Markov model or single-letter context (that is, we look at the probability model given n), we can see that the probability of g would increase substantially. As we increase the context size (go from n to *in* to *din* and so on), the probability of the alphabet becomes more and more skewed, which results in lower entropy.

Shannon used a second-order model for English text consisting of the 26 letters and one space to obtain an entropy of 3.1 bits/letter [192]. Using a model where the output symbols were words rather than letters brought down the entropy to 2.4 bits/letter. Shannon then used predictions generated by people (rather than statistical models) to estimate the upper and lower bounds on the entropy of the 27-letter English language. For the case where the subjects knew the 100 previous letters, he estimated these bounds to be 1.3 and 0.6 bits/letter, respectively.

The longer the context, the better its predictive value. However, if we were to store the probability model with respect to all contexts of a given length, the number of contexts would grow exponentially with length of context. Furthermore, given that the source imposes some structure on its output, many of these contexts may correspond to strings that would never occur in practice. Consider a context model of order four (the context is determined by the last four symbols). If we take an alphabet size of 95, the possible number of contexts is 95^4—more than 81 million!

This problem is further exacerbated by the fact that different realizations of the source output may vary considerably in terms of repeating patterns. Therefore, context modeling in text compression schemes tends to be an adaptive strategy in which the probabilities for different symbols in the different contexts are updated as they are encountered. However, this means that we will often encounter symbols that have not been encountered before for any of the given contexts (this is known as the *zero frequency problem*). The larger the context, the more often this will happen. This problem could be resolved by sending a code to indicate that the following symbol was being encountered for the first time, followed by a prearranged code for that symbol. This would significantly increase the length of the code for the symbol on its first occurrence (in the given context). However, if this situation did not occur too often, the overhead associated with such occurrences would be small compared to the total number of bits used to encode the output of the source. Unfortunately, in context-based encoding, the zero frequency problem is encountered often enough for overhead to be a problem, especially for longer contexts.

This problem can be resolved by a process called *exclusion*. The exclusion approach works by first attempting to find if the symbol to be encoded has a nonzero probability with respect to the maximum context length. If this is so, the symbol is encoded and transmitted. If not, an escape symbol is transmitted and the context size is reduced by one and the process is repeated. This procedure is repeated until a context is found with respect to which the symbol has a nonzero probability. To guarantee that this process converges, a null context is always included with respect to which all symbols have equal probability. Initially, only the shorter contexts are likely to be used. However, as more and more of the source output is processed, the longer contexts, which offer better prediction, will be used more often. The probability of the escape symbol can be computed in a number of different ways, leading to different implementations [20].

The use of Markov models in text compression is a rich and active area of research. Our coverage here is rather sketchy. For a fuller discussion of this area, see [20].

2.3.4 Composite Source Model

In many applications, it is not easy to use a single model to describe the source. In such cases, we can define a *composite source*, which can be viewed as a combination or composition of

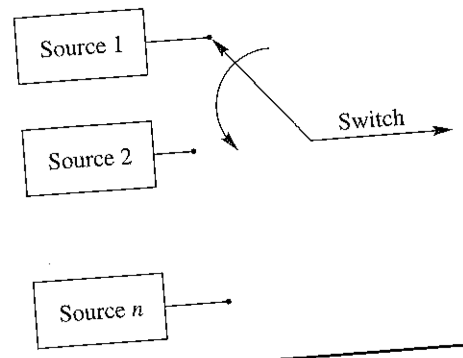

FIGURE 2.2 **A composite source.**

several sources, with only one source being *active* at any given time. A composite source can be represented as a number of individual sources $\mathcal{S}_i$, each with its own model $\mathcal{M}_i$, and a switch that selects a source $\mathcal{S}_i$ with probability π_i (as shown in Figure 2.2). This is an exceptionally rich model and can be used to describe some very complicated processes. We will describe this model in more detail when we need it.

2.4 Summary

In this chapter we learned some of the basic definitions of information theory. This was a rather brief visit, and we will revisit the subject in Chapter 7. However, the coverage in this chapter will be sufficient for the next four chapters. We also looked, rather briefly, at different approaches to modeling. When the need arises to understand a model in more depth later in the book, we will devote more attention to it at that time.

Further Reading

1. A very readable book on information theory and its applications in a number of fields is *Signals, Systems, and Noise—The Nature and Process of Communications*, by J.R. Pierce [167].
2. Another good introductory source for the material in this chapter is Chapter 6 of *Coding and Information Theory*, by R.W. Hamming [95].
3. Various models for text compression are described very nicely and in more detail in *Text Compression*, by T.C. Bell, J.G. Cleary, and I.H. Witten [20].
4. For a more thorough and detailed account of information theory, the following books are especially recommended (the first two are my personal favorites): *Information Theory*, by R.B. Ash [12]; *Transmission of Information*, by R.M. Fano [65]; *Information Theory and Reliable Communication*, by R.G. Gallagher [73]; *Entropy and Information Theory*, by R.M. Gray [93]; *Elements of Information Theory*, by T.M. Cover and J.A. Thomas [50]; and *The Theory of Information and Coding*, by R.J. McEliece [148].

2.5 Projects and Problems

1. Suppose X is a random variable that takes on values from an M-letter alphabet. Show that $0 \leq H(X) \leq \log_2 M$.
2. Show that for the case where the elements of an observed sequence are *iid*, the entropy is equal to the first-order entropy.
3. Given an alphabet $\mathcal{A} = \{a_1, a_2, a_3, a_4\}$, find the first-order entropy in the following cases:
   (a) $P(a_1) = P(a_2) = P(a_3) = P(a_4) = \frac{1}{4}$.
   (b) $P(a_1) = \frac{1}{2}$, $P(a_2) = \frac{1}{4}$, $P(a_3) = P(a_4) = \frac{1}{8}$.
   (c) $P(a_1) = 0.55$, $P(a_2) = \frac{1}{4}$, $P(a_3) = \frac{1}{8}$, and $P(a_4) = 0.12$.
4. Suppose we have a source with a probability model $P = \{p_0, p_1, \ldots, p_m\}$, and entropy H_P. Suppose we have another source with probability model $Q = \{q_0, q_1, \ldots, q_m\}$ and entropy H_Q, where
$$q_i = p_i \qquad i = 0, 1, \ldots, j-2, j+1, \ldots, m$$
and
$$q_j = q_{j-1} = \frac{p_j + p_{j-1}}{2}.$$
How is H_Q related to H_P (greater, equal, or less)? Prove your answer.
5. There are several image and speech files among the accompanying data sets. Write a program to compute the first-order entropy of some of the image and speech files. Pick one of the image files and compute its second-order entropy. Comment on the difference.
6. Conduct an experiment to see how well a model can describe a source.
   (a) Write a program that randomly selects letters from the 26-letter alphabet $\{a, b, \ldots, z\}$ and forms four-letter words. Form 100 such words and see how many of these words make sense.
   (b) Among the accompanying data sets is a file called `4letter.words`, which contains a list of four-letter words. Using this file, obtain a probability model for the alphabet. Now repeat part (a) generating the words using the probability model. To pick letters, construct the cumulative density function (*cdf*) $F_X(x)$ (see Appendix A for definition of *cdf*). Using a uniform pseudorandom number generator to generate a value r, where $0 \leq r < 1$, pick the letter x_k if $F_X(x_k - 1) \leq r < F_X(x_k)$. Compare your results with those of part (a).
   (c) Repeat (b) using a single-letter context.
   (d) Repeat (b) using a two-letter context.