where $p_i = P(X_i = 1)$ is not constant but a function of i, chosen carefully so that the limit in (4.10) does not exist. For example, let

$$p_i = \begin{cases} 0.5 & \text{if } 2k < \log\log i \le 2k+1, \\ 0 & \text{if } 2k+1 < \log\log i \le 2k+2 \end{cases} \tag{4.13}$$

for $k = 0, 1, 2, \ldots$.
Then there are arbitrarily long stretches where $H(X_i) = 1$, followed by exponentially longer segments where $H(X_i) = 0$. Hence, the running average of the $H(X_i)$ will oscillate between 0 and 1 and will not have a limit. Thus, $H(\mathcal{X})$ is not defined for this process.

We can also define a related quantity for entropy rate:

$$H'(\mathcal{X}) = \lim_{n\to\infty} H(X_n|X_{n-1}, X_{n-2}, \ldots, X_1) \tag{4.14}$$

when the limit exists.

The two quantities $H(\mathcal{X})$ and $H'(\mathcal{X})$ correspond to two different notions of entropy rate. The first is the per symbol entropy of the n random variables, and the second is the conditional entropy of the last random variable given the past. We now prove the important result that for stationary processes both limits exist and are equal.

Theorem 4.2.1 *For a stationary stochastic process, the limits in (4.10) and (4.14) exist and are equal:*

$$H(\mathcal{X}) = H'(\mathcal{X}). \tag{4.15}$$

We first prove that $\lim H(X_n|X_{n-1}, \ldots, X_1)$ exists.

Theorem 4.2.2 *For a stationary stochastic process, $H(X_n|X_{n-1}, \ldots, X_1)$ is nonincreasing in n and has a limit $H'(\mathcal{X})$.*

Proof

$$H(X_{n+1}|X_1, X_2, \ldots, X_n) \le H(X_{n+1}|X_n, \ldots, X_2) \tag{4.16}$$

$$= H(X_n|X_{n-1}, \ldots, X_1), \tag{4.17}$$

where the inequality follows from the fact that conditioning reduces entropy and the equality follows from the stationarity of the process. Since $H(X_n|X_{n-1}, \ldots, X_1)$ is a decreasing sequence of nonnegative numbers, it has a limit, $H'(\mathcal{X})$. □

We now use the following simple result from analysis.

Theorem 4.2.3 (*Cesáro mean*) *If* $a_n \to a$ *and* $b_n = \frac{1}{n}\sum_{i=1}^{n} a_i$, *then* $b_n \to a$.

Proof: (*Informal outline*). Since most of the terms in the sequence $\{a_k\}$ are eventually close to a, then b_n, which is the average of the first n terms, is also eventually close to a.

Formal Proof: Let $\epsilon > 0$. Since $a_n \to a$, there exists a number $N(\epsilon)$ such that $|a_n - a| \leq \epsilon$ for all $n \geq N(\epsilon)$. Hence,

$$|b_n - a| = \left| \frac{1}{n}\sum_{i=1}^{n}(a_i - a) \right| \tag{4.18}$$

$$\leq \frac{1}{n}\sum_{i=1}^{n} |(a_i - a)| \tag{4.19}$$

$$\leq \frac{1}{n}\sum_{i=1}^{N(\epsilon)} |a_i - a| + \frac{n - N(\epsilon)}{n}\epsilon \tag{4.20}$$

$$\leq \frac{1}{n}\sum_{i=1}^{N(\epsilon)} |a_i - a| + \epsilon \tag{4.21}$$

for all $n \geq N(\epsilon)$. Since the first term goes to 0 as $n \to \infty$, we can make $|b_n - a| \leq 2\epsilon$ by taking n large enough. Hence, $b_n \to a$ as $n \to \infty$. □

Proof of Theorem 4.2.1: By the chain rule,

$$\frac{H(X_1, X_2, \ldots, X_n)}{n} = \frac{1}{n}\sum_{i=1}^{n} H(X_i | X_{i-1}, \ldots, X_1), \tag{4.22}$$

that is, the entropy rate is the time average of the conditional entropies. But we know that the conditional entropies tend to a limit H'. Hence, by Theorem 4.2.3, their running average has a limit, which is equal to the limit H' of the terms. Thus, by Theorem 4.2.2,

$$H(\mathcal{X}) = \lim \frac{H(X_1, X_2, \ldots, X_n)}{n} = \lim H(X_n | X_{n-1}, \ldots, X_1)$$
$$= H'(\mathcal{X}). \qquad \square \tag{4.23}$$

The significance of the entropy rate of a stochastic process arises from the AEP for a stationary ergodic process. We prove the general AEP in Section 16.8, where we show that for any stationary ergodic process,

$$-\frac{1}{n}\log p(X_1, X_2, \ldots, X_n) \to H(\mathcal{X}) \tag{4.24}$$

with probability 1. Using this, the theorems of Chapter 3 can easily be extended to a general stationary ergodic process. We can define a typical set in the same way as we did for the i.i.d. case in Chapter 3. By the same arguments, we can show that the typical set has a probability close to 1 and that there are about $2^{nH(\mathcal{X})}$ typical sequences of length n, each with probability about $2^{-nH(\mathcal{X})}$. We can therefore represent the typical sequences of length n using approximately $nH(\mathcal{X})$ bits. This shows the significance of the entropy rate as the average description length for a stationary ergodic process.

The entropy rate is well defined for all stationary processes. The entropy rate is particularly easy to calculate for Markov chains.

Markov Chains. For a stationary Markov chain, the entropy rate is given by

$$\begin{aligned} H(\mathcal{X}) &= H'(\mathcal{X}) = \lim H(X_n|X_{n-1}, \ldots, X_1) = \lim H(X_n|X_{n-1}) \\ &= H(X_2|X_1), \end{aligned} \tag{4.25}$$

where the conditional entropy is calculated using the given stationary distribution. Recall that the stationary distribution μ is the solution of the equations

$$\mu_j = \sum_i \mu_i P_{ij} \quad \text{for all } j. \tag{4.26}$$

We express the conditional entropy explicitly in the following theorem.

Theorem 4.2.4 *Let $\{X_i\}$ be a stationary Markov chain with stationary distribution μ and transition matrix P. Let $X_1 \sim \mu$. Then the entropy rate is*

$$H(\mathcal{X}) = -\sum_{ij} \mu_i P_{ij} \log P_{ij}. \tag{4.27}$$

Proof: $H(\mathcal{X}) = H(X_2|X_1) = \sum_i \mu_i \left(\sum_j -P_{ij} \log P_{ij} \right)$. □

Example 4.2.1 (*Two-state Markov chain*) The entropy rate of the two-state Markov chain in Figure 4.1 is

$$H(\mathcal{X}) = H(X_2|X_1) = \frac{\beta}{\alpha+\beta}H(\alpha) + \frac{\alpha}{\alpha+\beta}H(\beta). \tag{4.28}$$

Remark If the Markov chain is irreducible and aperiodic, it has a unique stationary distribution on the states, and any initial distribution tends to the stationary distribution as $n \to \infty$. In this case, even though the initial distribution is not the stationary distribution, the entropy rate, which is defined in terms of long-term behavior, is $H(\mathcal{X})$, as defined in (4.25) and (4.27).

4.3 EXAMPLE: ENTROPY RATE OF A RANDOM WALK ON A WEIGHTED GRAPH

As an example of a stochastic process, let us consider a random walk on a connected graph (Figure 4.2). Consider a graph with m nodes labeled $\{1, 2, \ldots, m\}$, with weight $W_{ij} \geq 0$ on the edge joining node i to node j. (The graph is assumed to be undirected, so that $W_{ij} = W_{ji}$. We set $W_{ij} = 0$ if there is no edge joining nodes i and j.)

A particle walks randomly from node to node in this graph. The random walk $\{X_n\}$, $X_n \in \{1, 2, \ldots, m\}$, is a sequence of vertices of the graph. Given $X_n = i$, the next vertex j is chosen from among the nodes connected to node i with a probability proportional to the weight of the edge connecting i to j. Thus, $P_{ij} = W_{ij}/\sum_k W_{ik}$.

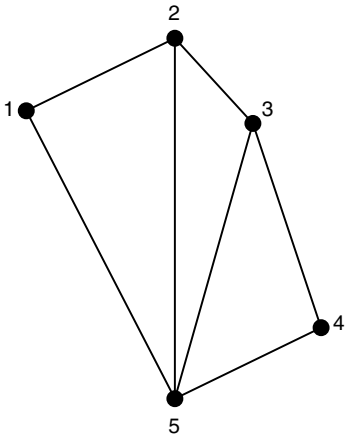


FIGURE 4.2. Random walk on a graph.

In this case, the stationary distribution has a surprisingly simple form, which we will guess and verify. The stationary distribution for this Markov chain assigns probability to node i proportional to the total weight of the edges emanating from node i. Let

$$W_i = \sum_j W_{ij} \tag{4.29}$$

be the total weight of edges emanating from node i, and let

$$W = \sum_{i,j:j>i} W_{ij} \tag{4.30}$$

be the sum of the weights of all the edges. Then $\sum_i W_i = 2W$.

We now guess that the stationary distribution is

$$\mu_i = \frac{W_i}{2W}. \tag{4.31}$$

We verify that this is the stationary distribution by checking that $\mu P = \mu$. Here

$$\sum_i \mu_i P_{ij} = \sum_i \frac{W_i}{2W} \frac{W_{ij}}{W_i} \tag{4.32}$$

$$= \sum_i \frac{1}{2W} W_{ij} \tag{4.33}$$

$$= \frac{W_j}{2W} \tag{4.34}$$

$$= \mu_j. \tag{4.35}$$

Thus, the stationary probability of state i is proportional to the weight of edges emanating from node i. This stationary distribution has an interesting property of locality: It depends only on the total weight and the weight of edges connected to the node and hence does not change if the weights in some other part of the graph are changed while keeping the total weight constant. We can now calculate the entropy rate as

$$H(\mathcal{X}) = H(X_2|X_1) \tag{4.36}$$

$$= -\sum_i \mu_i \sum_j P_{ij} \log P_{ij} \tag{4.37}$$

$$= -\sum_i \frac{W_i}{2W} \sum_j \frac{W_{ij}}{W_i} \log \frac{W_{ij}}{W_i} \tag{4.38}$$

$$= -\sum_i \sum_j \frac{W_{ij}}{2W} \log \frac{W_{ij}}{W_i} \tag{4.39}$$

$$= -\sum_i \sum_j \frac{W_{ij}}{2W} \log \frac{W_{ij}}{2W} + \sum_i \sum_j \frac{W_{ij}}{2W} \log \frac{W_i}{2W} \tag{4.40}$$

$$= H\left(\ldots, \frac{W_{ij}}{2W}, \ldots\right) - H\left(\ldots, \frac{W_i}{2W}, \ldots\right). \tag{4.41}$$

If all the edges have equal weight, the stationary distribution puts weight $E_i/2E$ on node i, where E_i is the number of edges emanating from node i and E is the total number of edges in the graph. In this case, the entropy rate of the random walk is

$$H(\mathcal{X}) = \log(2E) - H\left(\frac{E_1}{2E}, \frac{E_2}{2E}, \ldots, \frac{E_m}{2E}\right). \tag{4.42}$$

This answer for the entropy rate is so simple that it is almost misleading. Apparently, the entropy rate, which is the average transition entropy, depends only on the entropy of the stationary distribution and the total number of edges.

Example 4.3.1 (*Random walk on a chessboard*) Let a king move at random on an 8×8 chessboard. The king has eight moves in the interior, five moves at the edges, and three moves at the corners. Using this and the preceding results, the stationary probabilities are, respectively, $\frac{8}{420}$, $\frac{5}{420}$, and $\frac{3}{420}$, and the entropy rate is $0.92 \log 8$. The factor of 0.92 is due to edge effects; we would have an entropy rate of $\log 8$ on an infinite chessboard.

Similarly, we can find the entropy rate of rooks ($\log 14$ bits, since the rook always has 14 possible moves), bishops, and queens. The queen combines the moves of a rook and a bishop. Does the queen have more or less freedom than the pair?

Remark It is easy to see that a stationary random walk on a graph is *time-reversible*; that is, the probability of any sequence of states is the

same forward or backward:

$$\begin{aligned}&\Pr(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)\\ &\quad = \Pr(X_n = x_1, X_{n-1} = x_2, \ldots, X_1 = x_n).\end{aligned} \tag{4.43}$$

Rather surprisingly, the converse is also true; that is, any time-reversible Markov chain can be represented as a random walk on an undirected weighted graph.

4.4 SECOND LAW OF THERMODYNAMICS

One of the basic laws of physics, the second law of thermodynamics, states that the entropy of an isolated system is nondecreasing. We now explore the relationship between the second law and the entropy function that we defined earlier in this chapter.

In statistical thermodynamics, entropy is often defined as the log of the number of microstates in the system. This corresponds exactly to our notion of entropy if all the states are equally likely. But why does entropy increase?

We model the isolated system as a Markov chain with transitions obeying the physical laws governing the system. Implicit in this assumption is the notion of an overall state of the system and the fact that knowing the present state, the future of the system is independent of the past. In such a system we can find four different interpretations of the second law. It may come as a shock to find that the entropy does not always increase. However, *relative* entropy always decreases.

1. *Relative entropy $D(\mu_n||\mu'_n)$ decreases with n.* Let μ_n and μ'_n be two probability distributions on the state space of a Markov chain at time n, and let μ_{n+1} and μ'_{n+1} be the corresponding distributions at time $n+1$. Let the corresponding joint mass functions be denoted by p and q. Thus, $p(x_n, x_{n+1}) = p(x_n)r(x_{n+1}|x_n)$ and $q(x_n, x_{n+1}) = q(x_n)r(x_{n+1}|x_n)$, where $r(\cdot|\cdot)$ is the probability transition function for the Markov chain. Then by the chain rule for relative entropy, we have two expansions:

$$\begin{aligned}D(p(x_n, x_{n+1})||q(x_n, x_{n+1})) &= D(p(x_n)||q(x_n))\\ &\quad + D(p(x_{n+1}|x_n)||q(x_{n+1}|x_n))\end{aligned}$$

$$= D(p(x_{n+1})||q(x_{n+1})) + D(p(x_n|x_{n+1})||q(x_n|x_{n+1})).$$

Since both p and q are derived from the Markov chain, the conditional probability mass functions $p(x_{n+1}|x_n)$ and $q(x_{n+1}|x_n)$ are both equal to $r(x_{n+1}|x_n)$, and hence $D(p(x_{n+1}|x_n)||q(x_{n+1}|x_n)) = 0$. Now using the nonnegativity of $D(p(x_n|x_{n+1})||q(x_n|x_{n+1}))$ (Corollary to Theorem 2.6.3), we have

$$D(p(x_n)||q(x_n)) \geq D(p(x_{n+1})||q(x_{n+1})) \tag{4.44}$$

or

$$D(\mu_n||\mu'_n) \geq D(\mu_{n+1}||\mu'_{n+1}). \tag{4.45}$$

Consequently, the distance between the probability mass functions is decreasing with time n for any Markov chain.

An example of one interpretation of the preceding inequality is to suppose that the tax system for the redistribution of wealth is the same in Canada and in England. Then if μ_n and μ'_n represent the distributions of wealth among people in the two countries, this inequality shows that the relative entropy distance between the two distributions decreases with time. The wealth distributions in Canada and England become more similar.

2. *Relative entropy $D(\mu_n||\mu)$ between a distribution μ_n on the states at time n and a stationary distribution μ decreases with n.* In (4.45), μ'_n is any distribution on the states at time n. If we let μ'_n be any stationary distribution μ, the distribution μ'_{n+1} at the next time is also equal to μ. Hence,

$$D(\mu_n||\mu) \geq D(\mu_{n+1}||\mu), \tag{4.46}$$

which implies that any state distribution gets closer and closer to each stationary distribution as time passes. The sequence $D(\mu_n||\mu)$ is a monotonically nonincreasing nonnegative sequence and must therefore have a limit. The limit is zero if the stationary distribution is unique, but this is more difficult to prove.

3. *Entropy increases if the stationary distribution is uniform.* In general, the fact that the relative entropy decreases does not imply that the entropy increases. A simple counterexample is provided by any Markov chain with a nonuniform stationary distribution. If we start

this Markov chain from the uniform distribution, which already is the maximum entropy distribution, the distribution will tend to the stationary distribution, which has a lower entropy than the uniform. Here, the entropy decreases with time.
If, however, the stationary distribution is the uniform distribution, we can express the relative entropy as

$$D(\mu_n||\mu) = \log|\mathcal{X}| - H(\mu_n) = \log|\mathcal{X}| - H(X_n). \tag{4.47}$$

In this case the monotonic decrease in relative entropy implies a monotonic increase in entropy. This is the explanation that ties in most closely with statistical thermodynamics, where all the microstates are equally likely. We now characterize processes having a uniform stationary distribution.

Definition A probability transition matrix $[P_{ij}]$, $P_{ij} = \Pr\{X_{n+1} = j|X_n = i\}$, is called *doubly stochastic* if

$$\sum_i P_{ij} = 1, \qquad j = 1, 2, \ldots \tag{4.48}$$

and

$$\sum_j P_{ij} = 1, \qquad i = 1, 2, \ldots. \tag{4.49}$$

Remark The uniform distribution is a stationary distribution of P if and only if the probability transition matrix is doubly stochastic (see Problem 4.1).

4. *The conditional entropy $H(X_n|X_1)$ increases with n for a stationary Markov process*. If the Markov process is stationary, $H(X_n)$ is constant. So the entropy is nonincreasing. However, we will prove that $H(X_n|X_1)$ increases with n. Thus, the conditional uncertainty of the future increases. We give two alternative proofs of this result. First, we use the properties of entropy,

$$H(X_n|X_1) \geq H(X_n|X_1, X_2) \quad \text{(conditioning reduces entropy)} \tag{4.50}$$

$$= H(X_n|X_2) \quad \text{(by Markovity)} \tag{4.51}$$

$$= H(X_{n-1}|X_1) \quad \text{(by stationarity)}. \tag{4.52}$$

Alternatively, by an application of the data-processing inequality to the Markov chain $X_1 \to X_{n-1} \to X_n$, we have

$$I(X_1; X_{n-1}) \geq I(X_1; X_n). \tag{4.53}$$

Expanding the mutual informations in terms of entropies, we have

$$H(X_{n-1}) - H(X_{n-1}|X_1) \geq H(X_n) - H(X_n|X_1). \tag{4.54}$$

By stationarity, $H(X_{n-1}) = H(X_n)$, and hence we have

$$H(X_{n-1}|X_1) \leq H(X_n|X_1). \tag{4.55}$$

[These techniques can also be used to show that $H(X_0|X_n)$ is increasing in n for any Markov chain.]

5. *Shuffles increase entropy*. If T is a shuffle (permutation) of a deck of cards and X is the initial (random) position of the cards in the deck, and if the choice of the shuffle T is independent of X, then

$$H(TX) \geq H(X), \tag{4.56}$$

where TX is the permutation of the deck induced by the shuffle T on the initial permutation X. Problem 4.3 outlines a proof.

4.5 FUNCTIONS OF MARKOV CHAINS

Here is an example that can be very difficult if done the wrong way. It illustrates the power of the techniques developed so far. Let $X_1, X_2, \ldots, X_n, \ldots$ be a stationary Markov chain, and let $Y_i = \phi(X_i)$ be a process each term of which is a function of the corresponding state in the Markov chain. What is the entropy rate $H(\mathcal{Y})$? Such functions of Markov chains occur often in practice. In many situations, one has only partial information about the state of the system. It would simplify matters greatly if $Y_1, Y_2, \ldots, Y_n$ also formed a Markov chain, but in many cases, this is not true. Since the Markov chain is stationary, so is $Y_1, Y_2, \ldots, Y_n$, and the entropy rate is well defined. However, if we wish to compute $H(\mathcal{Y})$, we might compute $H(Y_n|Y_{n-1}, \ldots, Y_1)$ for each n and find the limit. Since the convergence can be arbitrarily slow, we will never know how close we are to the limit. (We can't look at the change between the values at n and $n+1$, since this difference may be small even when we are far away from the limit—consider, for example, $\sum \frac{1}{n}$.)

It would be useful computationally to have upper and lower bounds converging to the limit from above and below. We can halt the computation when the difference between upper and lower bounds is small, and we will then have a good estimate of the limit.

We already know that $H(Y_n|Y_{n-1}, \ldots, Y_1)$ converges monotonically to $H(\mathcal{Y})$ from above. For a lower bound, we will use $H(Y_n|Y_{n-1}, \ldots, Y_1, X_1)$. This is a neat trick based on the idea that X_1 contains as much information about Y_n as $Y_1, Y_0, Y_{-1}, \ldots$.

Lemma 4.5.1

$$H(Y_n|Y_{n-1}, \ldots, Y_2, X_1) \leq H(\mathcal{Y}). \tag{4.57}$$

Proof: We have for $k = 1, 2, \ldots$,

$$H(Y_n|Y_{n-1}, \ldots, Y_2, X_1) \stackrel{(a)}{=} H(Y_n|Y_{n-1}, \ldots, Y_2, Y_1, X_1) \tag{4.58}$$

$$\stackrel{(b)}{=} H(Y_n|Y_{n-1}, \ldots, Y_1, X_1, X_0, X_{-1}, \ldots, X_{-k}) \tag{4.59}$$

$$\stackrel{(c)}{=} H(Y_n|Y_{n-1}, \ldots, Y_1, X_1, X_0, X_{-1}, \ldots, X_{-k}, Y_0, \ldots, Y_{-k}) \tag{4.60}$$

$$\stackrel{(d)}{\leq} H(Y_n|Y_{n-1}, \ldots, Y_1, Y_0, \ldots, Y_{-k}) \tag{4.61}$$

$$\stackrel{(e)}{=} H(Y_{n+k+1}|Y_{n+k}, \ldots, Y_1), \tag{4.62}$$

where (a) follows from that fact that Y_1 is a function of X_1, and (b) follows from the Markovity of X, (c) follows from the fact that Y_i is a function of X_i, (d) follows from the fact that conditioning reduces entropy, and (e) follows by stationarity. Since the inequality is true for all k, it is true in the limit. Thus,

$$H(Y_n|Y_{n-1}, \ldots, Y_1, X_1) \leq \lim_k H(Y_{n+k+1}|Y_{n+k}, \ldots, Y_1) \tag{4.63}$$

$$= H(\mathcal{Y}). \qquad \square \tag{4.64}$$

The next lemma shows that the interval between the upper and the lower bounds decreases in length.

Lemma 4.5.2

$$H(Y_n|Y_{n-1}, \ldots, Y_1) - H(Y_n|Y_{n-1}, \ldots, Y_1, X_1) \to 0. \tag{4.65}$$

Proof: The interval length can be rewritten as

$$H(Y_n|Y_{n-1}, \ldots, Y_1) - H(Y_n|Y_{n-1}, \ldots, Y_1, X_1)$$
$$= I(X_1; Y_n|Y_{n-1}, \ldots, Y_1). \tag{4.66}$$

By the properties of mutual information,

$$I(X_1; Y_1, Y_2, \ldots, Y_n) \leq H(X_1), \tag{4.67}$$

and $I(X_1; Y_1, Y_2, \ldots, Y_n)$ increases with n. Thus, $\lim I(X_1; Y_1, Y_2, \ldots, Y_n)$ exists and

$$\lim_{n\to\infty} I(X_1; Y_1, Y_2, \ldots, Y_n) \leq H(X_1). \tag{4.68}$$

By the chain rule,

$$H(X) \geq \lim_{n\to\infty} I(X_1; Y_1, Y_2, \ldots, Y_n) \tag{4.69}$$

$$= \lim_{n\to\infty} \sum_{i=1}^{n} I(X_1; Y_i|Y_{i-1}, \ldots, Y_1) \tag{4.70}$$

$$= \sum_{i=1}^{\infty} I(X_1; Y_i|Y_{i-1}, \ldots, Y_1). \tag{4.71}$$

Since this infinite sum is finite and the terms are nonnegative, the terms must tend to 0; that is,

$$\lim I(X_1; Y_n|Y_{n-1}, \ldots, Y_1) = 0, \tag{4.72}$$

which proves the lemma. □

Combining Lemmas 4.5.1 and 4.5.2, we have the following theorem.

Theorem 4.5.1 *If $X_1, X_2, \ldots, X_n$ form a stationary Markov chain, and $Y_i = \phi(X_i)$, then*

$$H(Y_n|Y_{n-1}, \ldots, Y_1, X_1) \leq H(\mathcal{Y}) \leq H(Y_n|Y_{n-1}, \ldots, Y_1) \tag{4.73}$$

and

$$\lim H(Y_n|Y_{n-1}, \ldots, Y_1, X_1) = H(\mathcal{Y}) = \lim H(Y_n|Y_{n-1}, \ldots, Y_1). \tag{4.74}$$

In general, we could also consider the case where Y_i is a stochastic function (as opposed to a deterministic function) of X_i. Consider a Markov

process $X_1, X_2, \ldots, X_n$, and define a new process $Y_1, Y_2, \ldots, Y_n$, where each Y_i is drawn according to $p(y_i|x_i)$, conditionally independent of all the other X_j, $j \neq i$; that is,

$$p(x^n, y^n) = p(x_1) \prod_{i=1}^{n-1} p(x_{i+1}|x_i) \prod_{i=1}^{n} p(y_i|x_i). \tag{4.75}$$

Such a process, called a *hidden Markov model* (HMM), is used extensively in speech recognition, handwriting recognition, and so on. The same argument as that used above for functions of a Markov chain carry over to hidden Markov models, and we can lower bound the entropy rate of a hidden Markov model by conditioning it on the underlying Markov state. The details of the argument are left to the reader.

SUMMARY

Entropy rate. Two definitions of entropy rate for a stochastic process are

$$H(\mathcal{X}) = \lim_{n\to\infty} \frac{1}{n} H(X_1, X_2, \ldots, X_n), \tag{4.76}$$

$$H'(\mathcal{X}) = \lim_{n\to\infty} H(X_n|X_{n-1}, X_{n-2}, \ldots, X_1). \tag{4.77}$$

For a stationary stochastic process,

$$H(\mathcal{X}) = H'(\mathcal{X}). \tag{4.78}$$

Entropy rate of a stationary Markov chain

$$H(\mathcal{X}) = -\sum_{ij} \mu_i P_{ij} \log P_{ij}. \tag{4.79}$$

Second law of thermodynamics. For a Markov chain:

1. Relative entropy $D(\mu_n||\mu'_n)$ decreases with time
2. Relative entropy $D(\mu_n||\mu)$ between a distribution and the stationary distribution decreases with time.
3. Entropy $H(X_n)$ increases if the stationary distribution is uniform.

4. The conditional entropy $H(X_n|X_1)$ increases with time for a stationary Markov chain.
5. The conditional entropy $H(X_0|X_n)$ of the initial condition X_0 increases for any Markov chain.

Functions of a Markov chain. If $X_1, X_2, \ldots, X_n$ form a stationary Markov chain and $Y_i = \phi(X_i)$, then

$$H(Y_n|Y_{n-1}, \ldots, Y_1, X_1) \leq H(\mathcal{Y}) \leq H(Y_n|Y_{n-1}, \ldots, Y_1) \tag{4.80}$$

and

$$\lim_{n\to\infty} H(Y_n|Y_{n-1}, \ldots, Y_1, X_1) = H(\mathcal{Y}) = \lim_{n\to\infty} H(Y_n|Y_{n-1}, \ldots, Y_1). \tag{4.81}$$

PROBLEMS

4.1 *Doubly stochastic matrices*. An $n \times n$ matrix $P = [P_{ij}]$ is said to be *doubly stochastic* if $P_{ij} \geq 0$ and $\sum_j P_{ij} = 1$ for all i and $\sum_i P_{ij} = 1$ for all j. An $n \times n$ matrix P is said to be a *permutation* matrix if it is doubly stochastic and there is precisely one $P_{ij} = 1$ in each row and each column. It can be shown that every doubly stochastic matrix can be written as the convex combination of permutation matrices.

(a) Let $\mathbf{a}^t = (a_1, a_2, \ldots, a_n)$, $a_i \geq 0$, $\sum a_i = 1$, be a probability vector. Let $\mathbf{b} = \mathbf{a}P$, where P is doubly stochastic. Show that $\mathbf{b}$ is a probability vector and that $H(b_1, b_2, \ldots, b_n) \geq H(a_1, a_2, \ldots, a_n)$. Thus, stochastic mixing increases entropy.

(b) Show that a stationary distribution μ for a doubly stochastic matrix P is the uniform distribution.

(c) Conversely, prove that if the uniform distribution is a stationary distribution for a Markov transition matrix P, then P is doubly stochastic.

4.2 *Time's arrow*. Let $\{X_i\}_{i=-\infty}^{\infty}$ be a stationary stochastic process. Prove that

$$H(X_0|X_{-1}, X_{-2}, \ldots, X_{-n}) = H(X_0|X_1, X_2, \ldots, X_n).$$

In other words, the present has a conditional entropy given the past equal to the conditional entropy given the future. This is true even though it is quite easy to concoct stationary random processes for which the flow into the future looks quite different from the flow into the past. That is, one can determine the direction of time by looking at a sample function of the process. Nonetheless, given the present state, the conditional uncertainty of the next symbol in the future is equal to the conditional uncertainty of the previous symbol in the past.

4.3 *Shuffles increase entropy*. Argue that for any distribution on shuffles T and any distribution on card positions X that

$$H(TX) \geq H(TX|T) \tag{4.82}$$

$$= H(T^{-1}TX|T) \tag{4.83}$$

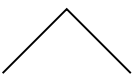
$$= H(X|T) \tag{4.84}$$

$$= H(X) \tag{4.85}$$

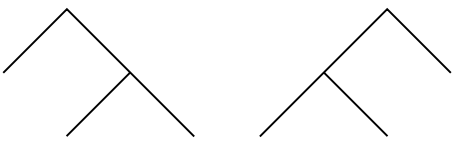
if X and T are independent.

4.4 *Second law of thermodynamics*. Let $X_1, X_2, X_3, \ldots$ be a stationary first-order Markov chain. In Section 4.4 it was shown that $H(X_n \mid X_1) \geq H(X_{n-1} \mid X_1)$ for $n = 2, 3, \ldots$. Thus, conditional uncertainty about the future grows with time. This is true although the unconditional uncertainty $H(X_n)$ remains constant. However, show by example that $H(X_n|X_1 = x_1)$ does not necessarily grow with n for every x_1.

4.5 *Entropy of a random tree*. Consider the following method of generating a random tree with n nodes. First expand the root node:

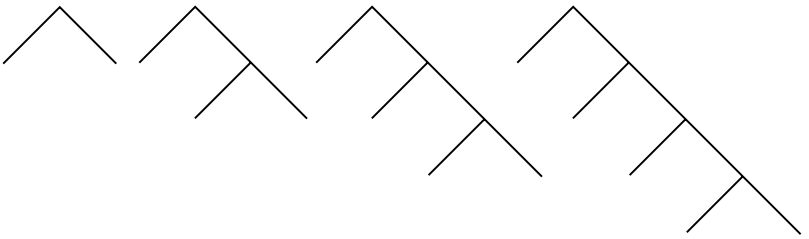


Then expand one of the two terminal nodes at random:

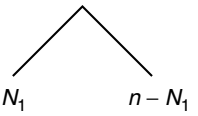


At time k, choose one of the $k-1$ terminal nodes according to a uniform distribution and expand it. Continue until n terminal nodes

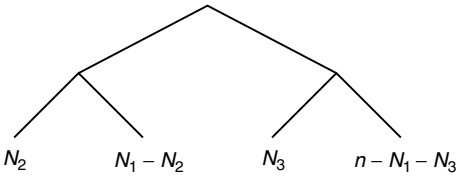
have been generated. Thus, a sequence leading to a five-node tree might look like this:



Surprisingly, the following method of generating random trees yields the same probability distribution on trees with n terminal nodes. First choose an integer N_1 uniformly distributed on $\{1, 2, \ldots, n-1\}$. We then have the picture



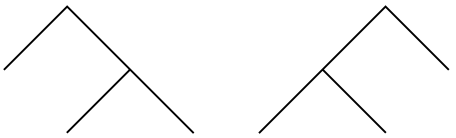
Then choose an integer N_2 uniformly distributed over $\{1, 2, \ldots, N_1 - 1\}$, and independently choose another integer N_3 uniformly over $\{1, 2, \ldots, (n - N_1) - 1\}$. The picture is now



Continue the process until no further subdivision can be made. (The equivalence of these two tree generation schemes follows, for example, from Polya's urn model.)

Now let T_n denote a random n-node tree generated as described. The probability distribution on such trees seems difficult to describe, but we can find the entropy of this distribution in recursive form.

First some examples. For $n = 2$, we have only one tree. Thus, $H(T_2) = 0$. For $n = 3$, we have two equally probable trees:



Thus, $H(T_3) = \log 2$. For $n = 4$, we have five possible trees, with probabilities $\frac{1}{3}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}$.

Now for the recurrence relation. Let $N_1(T_n)$ denote the number of terminal nodes of T_n in the right half of the tree. Justify each of the steps in the following:

$$H(T_n) \overset{\text{(a)}}{=} H(N_1, T_n) \tag{4.86}$$

$$\overset{\text{(b)}}{=} H(N_1) + H(T_n|N_1) \tag{4.87}$$

$$\overset{\text{(c)}}{=} \log(n-1) + H(T_n|N_1) \tag{4.88}$$

$$\overset{\text{(d)}}{=} \log(n-1) + \frac{1}{n-1}\sum_{k=1}^{n-1}(H(T_k) + H(T_{n-k})) \tag{4.89}$$

$$\overset{\text{(e)}}{=} \log(n-1) + \frac{2}{n-1}\sum_{k=1}^{n-1} H(T_k) \tag{4.90}$$

$$= \log(n-1) + \frac{2}{n-1}\sum_{k=1}^{n-1} H_k. \tag{4.91}$$

(f) Use this to show that

$$(n-1)H_n = nH_{n-1} + (n-1)\log(n-1) - (n-2)\log(n-2) \tag{4.92}$$

or

$$\frac{H_n}{n} = \frac{H_{n-1}}{n-1} + c_n \tag{4.93}$$

for appropriately defined c_n. Since $\sum c_n = c < \infty$, you have proved that $\frac{1}{n}H(T_n)$ converges to a constant. Thus, the expected number of bits necessary to describe the random tree T_n grows linearly with n.

4.6 *Monotonicity of entropy per element.* For a stationary stochastic process $X_1, X_2, \ldots, X_n$, show that

(a)

$$\frac{H(X_1, X_2, \ldots, X_n)}{n} \leq \frac{H(X_1, X_2, \ldots, X_{n-1})}{n-1}. \tag{4.94}$$

(b)

$$\frac{H(X_1, X_2, \ldots, X_n)}{n} \geq H(X_n|X_{n-1}, \ldots, X_1). \tag{4.95}$$

4.7 *Entropy rates of Markov chains*

(a) Find the entropy rate of the two-state Markov chain with transition matrix

$$P = \begin{bmatrix} 1 - p_{01} & p_{01} \\ p_{10} & 1 - p_{10} \end{bmatrix}.$$

(b) What values of p_{01}, p_{10} maximize the entropy rate?

(c) Find the entropy rate of the two-state Markov chain with transition matrix

$$P = \begin{bmatrix} 1 - p & p \\ 1 & 0 \end{bmatrix}.$$

(d) Find the maximum value of the entropy rate of the Markov chain of part (c). We expect that the maximizing value of p should be less than $\frac{1}{2}$, since the 0 state permits more information to be generated than the 1 state.

(e) Let $N(t)$ be the number of allowable state sequences of length t for the Markov chain of part (c). Find $N(t)$ and calculate

$$H_0 = \lim_{t \to \infty} \frac{1}{t} \log N(t).$$

[*Hint*: Find a linear recurrence that expresses $N(t)$ in terms of $N(t-1)$ and $N(t-2)$. Why is H_0 an upper bound on the entropy rate of the Markov chain? Compare H_0 with the maximum entropy found in part (d).]

4.8 *Maximum entropy process*. A discrete memoryless source has the alphabet $\{1, 2\}$, where the symbol 1 has duration 1 and the symbol 2 has duration 2. The probabilities of 1 and 2 are p_1 and p_2, respectively. Find the value of p_1 that maximizes the source entropy per unit time $H(\mathcal{X}) = \frac{H(X)}{ET}$. What is the maximum value $H(\mathcal{X})$?

4.9 *Initial conditions*. Show, for a Markov chain, that

$$H(X_0|X_n) \geq H(X_0|X_{n-1}).$$

Thus, initial conditions X_0 become more difficult to recover as the future X_n unfolds.

4.10 *Pairwise independence*. Let $X_1, X_2, \ldots, X_{n-1}$ be i.i.d. random variables taking values in $\{0, 1\}$, with $\Pr\{X_i = 1\} = \frac{1}{2}$. Let $X_n = 1$ if $\sum_{i=1}^{n-1} X_i$ is odd and $X_n = 0$ otherwise. Let $n \geq 3$.

(a) Show that X_i and X_j are independent for $i \neq j$, $i, j \in \{1, 2, \ldots, n\}$.

(b) Find $H(X_i, X_j)$ for $i \neq j$.

(c) Find $H(X_1, X_2, \ldots, X_n)$. Is this equal to $nH(X_1)$?

4.11 *Stationary processes.* Let $\ldots, X_{-1}, X_0, X_1, \ldots$ be a stationary (not necessarily Markov) stochastic process. Which of the following statements are true? Prove or provide a counterexample.

(a) $H(X_n|X_0) = H(X_{-n}|X_0)$.

(b) $H(X_n|X_0) \geq H(X_{n-1}|X_0)$.

(c) $H(X_n|X_1, X_2, \ldots, X_{n-1}, X_{n+1})$ is nonincreasing in n.

(d) $H(X_n|X_1, \ldots, X_{n-1}, X_{n+1}, \ldots, X_{2n})$ is nonincreasing in n.

4.12 *Entropy rate of a dog looking for a bone.* A dog walks on the integers, possibly reversing direction at each step with probability $p = 0.1$. Let $X_0 = 0$. The first step is equally likely to be positive or negative. A typical walk might look like this:

$$(X_0, X_1, \ldots) = (0, -1, -2, -3, -4, -3, -2, -1, 0, 1, \ldots).$$

(a) Find $H(X_1, X_2, \ldots, X_n)$.

(b) Find the entropy rate of the dog.

(c) What is the expected number of steps that the dog takes before reversing direction?

4.13 *The past has little to say about the future.* For a stationary stochastic process $X_1, X_2, \ldots, X_n, \ldots$, show that

$$\lim_{n\to\infty} \frac{1}{2n} I(X_1, X_2, \ldots, X_n; X_{n+1}, X_{n+2}, \ldots, X_{2n}) = 0. \quad (4.96)$$

Thus, the dependence between adjacent n-blocks of a stationary process does not grow linearly with n.

4.14 *Functions of a stochastic process*

(a) Consider a stationary stochastic process $X_1, X_2, \ldots, X_n$, and let $Y_1, Y_2, \ldots, Y_n$ be defined by

$$Y_i = \phi(X_i), \quad i = 1, 2, \ldots \quad (4.97)$$

for some function ϕ. Prove that

$$H(\mathcal{Y}) \leq H(\mathcal{X}). \quad (4.98)$$

(b) What is the relationship between the entropy rates $H(\mathcal{Z})$ and $H(\mathcal{X})$ if

$$Z_i = \psi(X_i, X_{i+1}), \quad i = 1, 2, \ldots \tag{4.99}$$

for some function ψ?

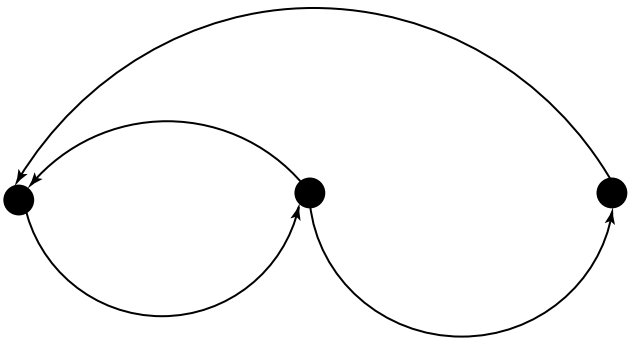
4.15 *Entropy rate*. Let $\{X_i\}$ be a discrete stationary stochastic process with entropy rate $H(\mathcal{X})$. Show that

$$\frac{1}{n} H(X_n, \ldots, X_1 \mid X_0, X_{-1}, \ldots, X_{-k}) \to H(\mathcal{X}) \tag{4.100}$$

for $k = 1, 2, \ldots$.

4.16 *Entropy rate of constrained sequences*. In magnetic recording, the mechanism of recording and reading the bits imposes constraints on the sequences of bits that can be recorded. For example, to ensure proper synchronization, it is often necessary to limit the length of runs of 0's between two 1's. Also, to reduce intersymbol interference, it may be necessary to require at least one 0 between any two 1's. We consider a simple example of such a constraint. Suppose that we are required to have at least one 0 and at most two 0's between any pair of 1's in a sequences. Thus, sequences like 101001 and 0101001 are valid sequences, but 0110010 and 0000101 are not. We wish to calculate the number of valid sequences of length n.

(a) Show that the set of constrained sequences is the same as the set of allowed paths on the following state diagram:



(b) Let $X_i(n)$ be the number of valid paths of length n ending at state i. Argue that $\mathbf{X}(n) = [X_1(n)\ X_2(n)\ X_3(n)]^t$ satisfies the

following recursion:

$$\begin{bmatrix} X_1(n) \\ X_2(n) \\ X_3(n) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_1(n-1) \\ X_2(n-1) \\ X_3(n-1) \end{bmatrix}, \tag{4.101}$$

with initial conditions $\mathbf{X}(1) = [1\ 1\ 0]^t$.

(c) Let

$$A = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{4.102}$$

Then we have by induction

$$\mathbf{X}(n) = A\mathbf{X}(n-1) = A^2\mathbf{X}(n-2) = \cdots = A^{n-1}\mathbf{X}(1). \tag{4.103}$$

Using the eigenvalue decomposition of A for the case of distinct eigenvalues, we can write $A = U^{-1}\Lambda U$, where Λ is the diagonal matrix of eigenvalues. Then $A^{n-1} = U^{-1}\Lambda^{n-1}U$. Show that we can write

$$\mathbf{X}(n) = \lambda_1^{n-1}\mathbf{Y}_1 + \lambda_2^{n-1}\mathbf{Y}_2 + \lambda_3^{n-1}\mathbf{Y}_3, \tag{4.104}$$

where $\mathbf{Y}_1, \mathbf{Y}_2, \mathbf{Y}_3$ do not depend on n. For large n, this sum is dominated by the largest term. Therefore, argue that for $i = 1, 2, 3$, we have

$$\frac{1}{n}\log X_i(n) \to \log \lambda, \tag{4.105}$$

where λ is the largest (positive) eigenvalue. Thus, the number of sequences of length n grows as λ^n for large n. Calculate λ for the matrix A above. (The case when the eigenvalues are not distinct can be handled in a similar manner.)

(d) We now take a different approach. Consider a Markov chain whose state diagram is the one given in part (a), but with arbitrary transition probabilities. Therefore, the probability transition matrix of this Markov chain is

$$P = \begin{bmatrix} 0 & 1 & 0 \\ \alpha & 0 & 1-\alpha \\ 1 & 0 & 0 \end{bmatrix}. \tag{4.106}$$

Show that the stationary distribution of this Markov chain is

$$\mu = \left[\frac{1}{3-\alpha}, \frac{1}{3-\alpha}, \frac{1-\alpha}{3-\alpha}\right]. \tag{4.107}$$

(e) Maximize the entropy rate of the Markov chain over choices of α. What is the maximum entropy rate of the chain?

(f) Compare the maximum entropy rate in part (e) with $\log \lambda$ in part (c). Why are the two answers the same?

4.17 *Recurrence times are insensitive to distributions.* Let $X_0, X_1, X_2, \ldots$ be drawn i.i.d. $\sim p(x), x \in \mathcal{X} = \{1, 2, \ldots, m\}$, and let N be the waiting time to the next occurrence of X_0. Thus $N = \min_n\{X_n = X_0\}$.

(a) Show that $EN = m$.

(b) Show that $E \log N \leq H(X)$.

(c) (*Optional*) Prove part (a) for $\{X_i\}$ stationary and ergodic.

4.18 *Stationary but not ergodic process.* A bin has two biased coins, one with probability of heads p and the other with probability of heads $1 - p$. One of these coins is chosen at random (i.e., with probability $\frac{1}{2}$) and is then tossed n times. Let X denote the identity of the coin that is picked, and let Y_1 and Y_2 denote the results of the first two tosses.

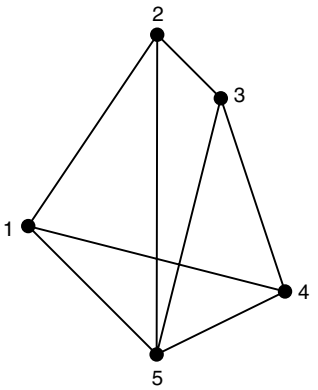
(a) Calculate $I(Y_1; Y_2|X)$.

(b) Calculate $I(X; Y_1, Y_2)$.

(c) Let $\mathcal{H}(\mathcal{Y})$ be the entropy rate of the Y process (the sequence of coin tosses). Calculate $\mathcal{H}(\mathcal{Y})$. [*Hint:* Relate this to $\lim \frac{1}{n} H(X, Y_1, Y_2, \ldots, Y_n)$.]

You can check the answer by considering the behavior as $p \to \frac{1}{2}$.

4.19 *Random walk on graph.* Consider a random walk on the following graph:



(a) Calculate the stationary distribution.

(b) What is the entropy rate?

(c) Find the mutual information $I(X_{n+1}; X_n)$ assuming that the process is stationary.

4.20 *Random walk on chessboard*. Find the entropy rate of the Markov chain associated with a random walk of a king on the 3×3 chessboard

1	2	3
4	5	6
7	8	9

What about the entropy rate of rooks, bishops, and queens? There are two types of bishops.

4.21 *Maximal entropy graphs*. Consider a random walk on a connected graph with four edges.

(a) Which graph has the highest entropy rate?

(b) Which graph has the lowest?

4.22 *Three-dimensional maze*. A bird is lost in a $3 \times 3 \times 3$ cubical maze. The bird flies from room to room going to adjoining rooms with equal probability through each of the walls. For example, the corner rooms have three exits.

(a) What is the stationary distribution?

(b) What is the entropy rate of this random walk?

4.23 *Entropy rate*. Let $\{X_i\}$ be a stationary stochastic process with entropy rate $H(\mathcal{X})$.

(a) Argue that $H(\mathcal{X}) \leq H(X_1)$.

(b) What are the conditions for equality?

4.24 *Entropy rates*. Let $\{X_i\}$ be a stationary process. Let $Y_i = (X_i, X_{i+1})$. Let $Z_i = (X_{2i}, X_{2i+1})$. Let $V_i = X_{2i}$. Consider the entropy rates $H(\mathcal{X})$, $H(\mathcal{Y})$, $H(\mathcal{Z})$, and $H(\mathcal{V})$ of the processes $\{X_i\}$,$\{Y_i\}$, $\{Z_i\}$, and $\{V_i\}$. What is the inequality relationship $\leq$, $=$, or $\geq$ between each of the pairs listed below?

(a) $H(\mathcal{X}) \gtreqless H(\mathcal{Y})$.

(b) $H(\mathcal{X}) \gtreqless H(\mathcal{Z})$.

(c) $H(\mathcal{X}) \gtreqless H(\mathcal{V})$.

(d) $H(\mathcal{Z}) \gtreqless H(\mathcal{X})$.

4.25 *Monotonicity*

(a) Show that $I(X; Y_1, Y_2, \ldots, Y_n)$ is nondecreasing in n.

(b) Under what conditions is the mutual information constant for all n?

4.26 *Transitions in Markov chains.* Suppose that $\{X_i\}$ forms an irreducible Markov chain with transition matrix P and stationary distribution μ. Form the associated "edge process" $\{Y_i\}$ by keeping track only of the transitions. Thus, the new process $\{Y_i\}$ takes values in $\mathcal{X} \times \mathcal{X}$, and $Y_i = (X_{i-1}, X_i)$. For example,

$$X^n = 3, 2, 8, 5, 7, \ldots$$

becomes

$$Y^n = (\emptyset, 3), (3, 2), (2, 8), (8, 5), (5, 7), \ldots.$$

Find the entropy rate of the edge process $\{Y_i\}$.

4.27 *Entropy rate.* Let $\{X_i\}$ be a stationary $\{0, 1\}$-valued stochastic process obeying

$$X_{k+1} = X_k \oplus X_{k-1} \oplus Z_{k+1},$$

where $\{Z_i\}$ is Bernoulli(p)and $\oplus$ denotes mod 2 addition. What is the entropy rate $H(\mathcal{X})$?

4.28 *Mixture of processes.* Suppose that we observe one of two stochastic processes but don't know which. What is the entropy rate? Specifically, let $X_{11}, X_{12}, X_{13}, \ldots$ be a Bernoulli process with parameter p_1, and let $X_{21}, X_{22}, X_{23}, \ldots$ be Bernoulli(p_2). Let

$$\theta = \begin{cases} 1 & \text{with probability } \frac{1}{2} \\ 2 & \text{with probability } \frac{1}{2} \end{cases}$$

and let $Y_i = X_{\theta i}, i = 1, 2, \ldots$, be the stochastic process observed. Thus, Y observes the process $\{X_{1i}\}$ or $\{X_{2i}\}$. Eventually, Y will know which.

(a) Is $\{Y_i\}$ stationary?

(b) Is $\{Y_i\}$ an i.i.d. process?

(c) What is the entropy rate H of $\{Y_i\}$?

(d) Does

$$-\frac{1}{n} \log p(Y_1, Y_2, \ldots Y_n) \longrightarrow H?$$

(e) Is there a code that achieves an expected per-symbol description length $\frac{1}{n} EL_n \longrightarrow H$?

Now let θ_i be Bern($\frac{1}{2}$). Observe that

$$Z_i = X_{\theta_i i}, \qquad i = 1, 2, \ldots.$$

Thus, θ is not fixed for all time, as it was in the first part, but is chosen i.i.d. each time. Answer parts (a), (b), (c), (d), (e) for the process $\{Z_i\}$, labeling the answers (a′), (b′), (c′), (d′), (e′).

4.29 *Waiting times*. Let X be the waiting time for the first heads to appear in successive flips of a fair coin. For example, $\Pr\{X = 3\} = (\frac{1}{2})^3$. Let S_n be the waiting time for the nth head to appear. Thus,

$$S_0 = 0$$

$$S_{n+1} = S_n + X_{n+1},$$

where $X_1, X_2, X_3, \ldots$ are i.i.d according to the distribution above.

(a) Is the process $\{S_n\}$ stationary?

(b) Calculate $H(S_1, S_2, \ldots, S_n)$.

(c) Does the process $\{S_n\}$ have an entropy rate? If so, what is it? If not, why not?

(d) What is the expected number of fair coin flips required to generate a random variable having the same distribution as S_n?

4.30 *Markov chain transitions*

$$P = [P_{ij}] = \begin{bmatrix} \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \end{bmatrix}.$$

Let X_1 be distributed uniformly over the states $\{0, 1, 2\}$. Let $\{X_i\}_1^\infty$ be a Markov chain with transition matrix P; thus, $P(X_{n+1} = j | X_n = i) = P_{ij}, i, j \in \{0, 1, 2\}$.

(a) Is $\{X_n\}$ stationary?

(b) Find $\lim_{n\to\infty} \frac{1}{n} H(X_1, \ldots, X_n)$.

Now consider the derived process $Z_1, Z_2, \ldots, Z_n$, where

$$Z_1 = X_1$$

$$Z_i = X_i - X_{i-1} \pmod 3, \qquad i = 2, \ldots, n.$$

Thus, Z^n encodes the transitions, not the states.

(c) Find $H(Z_1, Z_2, \ldots, Z_n)$.

(d) Find $H(Z_n)$ and $H(X_n)$ for $n \geq 2$.

(e) Find $H(Z_n|Z_{n-1})$ for $n \geq 2$.

(f) Are Z_{n-1} and Z_n independent for $n \geq 2$?

4.31 *Markov*. Let $\{X_i\} \sim \text{Bernoulli}(p)$. Consider the associated Markov chain $\{Y_i\}_{i=1}^n$, where
$Y_i = (\text{the number of 1's in the current run of 1's})$. For example, if $X^n = 101110\ldots$, we have $Y^n = 101230\ldots$.

(a) Find the entropy rate of X^n.

(b) Find the entropy rate of Y^n.

4.32 *Time symmetry*. Let $\{X_n\}$ be a stationary Markov process. We condition on (X_0, X_1) and look into the past and future. For what index k is

$$H(X_{-n}|X_0, X_1) = H(X_k|X_0, X_1)?$$

Give the argument.

4.33 *Chain inequality*. Let $X_1 \to X_2 \to X_3 \to X_4$ form a Markov chain. Show that

$$I(X_1; X_3) + I(X_2; X_4) \leq I(X_1; X_4) + I(X_2; X_3). \tag{4.108}$$

4.34 *Broadcast channel*. Let $X \to Y \to (Z, W)$ form a Markov chain [i.e., $p(x, y, z, w) = p(x)p(y|x)p(z, w|y)$ for all x, y, z, w]. Show that

$$I(X; Z) + I(X; W) \leq I(X; Y) + I(Z; W). \tag{4.109}$$

4.35 *Concavity of second law*. Let $\{X_n\}_{-\infty}^{\infty}$ be a stationary Markov process. Show that $H(X_n|X_0)$ is concave in n. Specifically, show that

$$\begin{aligned} &H(X_n|X_0) - H(X_{n-1}|X_0) - (H(X_{n-1}|X_0) - H(X_{n-2}|X_0)) \\ &\quad = -I(X_1; X_{n-1}|X_0, X_n) \leq 0. \end{aligned} \tag{4.110}$$

Thus, the second difference is negative, establishing that $H(X_n|X_0)$ is a concave function of n.

HISTORICAL NOTES

The entropy rate of a stochastic process was introduced by Shannon [472], who also explored some of the connections between the entropy rate of the process and the number of possible sequences generated by the process. Since Shannon, there have been a number of results extending the basic

theorems of information theory to general stochastic processes. The AEP for a general stationary stochastic process is proved in Chapter 16.

Hidden Markov models are used for a number of applications, such as speech recognition [432]. The calculation of the entropy rate for constrained sequences was introduced by Shannon [472]. These sequences are used for coding for magnetic and optical channels [288].

CHAPTER 5

DATA COMPRESSION

We now put content in the definition of entropy by establishing the fundamental limit for the compression of information. Data compression can be achieved by assigning short descriptions to the most frequent outcomes of the data source, and necessarily longer descriptions to the less frequent outcomes. For example, in Morse code, the most frequent symbol is represented by a single dot. In this chapter we find the shortest average description length of a random variable.

We first define the notion of an instantaneous code and then prove the important Kraft inequality, which asserts that the exponentiated codeword length assignments must look like a probability mass function. Elementary calculus then shows that the expected description length must be greater than or equal to the entropy, the first main result. Then Shannon's simple construction shows that the expected description length can achieve this bound asymptotically for repeated descriptions. This establishes the entropy as a natural measure of efficient description length. The famous Huffman coding procedure for finding minimum expected description length assignments is provided. Finally, we show that Huffman codes are competitively optimal and that it requires roughly H fair coin flips to generate a sample of a random variable having entropy H. Thus, the entropy is the data compression limit as well as the number of bits needed in random number generation, and codes achieving H turn out to be optimal from many points of view.

5.1 EXAMPLES OF CODES

Definition A *source code* C for a random variable X is a mapping from $\mathcal{X}$, the range of X, to $\mathcal{D}^*$, the set of finite-length strings of symbols from a D-ary alphabet. Let $C(x)$ denote the codeword corresponding to x and let $l(x)$ denote the length of $C(x)$.

Elements of Information Theory, Second Edition, By Thomas M. Cover and Joy A. Thomas

For example, $C(\text{red}) = 00$, $C(\text{blue}) = 11$ is a source code for $\mathcal{X} = \{\text{red}, \text{blue}\}$ with alphabet $\mathcal{D} = \{0, 1\}$.

Definition The *expected length* $L(C)$ of a source code $C(x)$ for a random variable X with probability mass function $p(x)$ is given by

$$L(C) = \sum_{x \in \mathcal{X}} p(x) l(x), \tag{5.1}$$

where $l(x)$ is the length of the codeword associated with x.

Without loss of generality, we can assume that the D-ary alphabet is $\mathcal{D} = \{0, 1, \ldots, D-1\}$.

Some examples of codes follow.

Example 5.1.1 Let X be a random variable with the following distribution and codeword assignment:

$$\begin{aligned} &\Pr(X = 1) = \tfrac{1}{2}, \quad \text{codeword } C(1) = 0 \\ &\Pr(X = 2) = \tfrac{1}{4}, \quad \text{codeword } C(2) = 10 \\ &\Pr(X = 3) = \tfrac{1}{8}, \quad \text{codeword } C(3) = 110 \\ &\Pr(X = 4) = \tfrac{1}{8}, \quad \text{codeword } C(4) = 111. \end{aligned} \tag{5.2}$$

The entropy $H(X)$ of X is 1.75 bits, and the expected length $L(C) = El(X)$ of this code is also 1.75 bits. Here we have a code that has the same average length as the entropy. We note that any sequence of bits can be uniquely decoded into a sequence of symbols of X. For example, the bit string 0110111100110 is decoded as 134213.

Example 5.1.2 Consider another simple example of a code for a random variable:

$$\begin{aligned} &\Pr(X = 1) = \tfrac{1}{3}, \quad \text{codeword } C(1) = 0 \\ &\Pr(X = 2) = \tfrac{1}{3}, \quad \text{codeword } C(2) = 10 \\ &\Pr(X = 3) = \tfrac{1}{3}, \quad \text{codeword } C(3) = 11. \end{aligned} \tag{5.3}$$

Just as in Example 5.1.1, the code is uniquely decodable. However, in this case the entropy is $\log 3 = 1.58$ bits and the average length of the encoding is 1.66 bits. Here $El(X) > H(X)$.

Example 5.1.3 (*Morse code*) The Morse code is a reasonably efficient code for the English alphabet using an alphabet of four symbols: a dot,

a dash, a letter space, and a word space. Short sequences represent frequent letters (e.g., a single dot represents E) and long sequences represent infrequent letters (e.g., Q is represented by "dash,dash,dot,dash"). This is not the optimal representation for the alphabet in four symbols—in fact, many possible codewords are not utilized because the codewords for letters do not contain spaces except for a letter space at the end of every codeword, and no space can follow another space. It is an interesting problem to calculate the number of sequences that can be constructed under these constraints. The problem was solved by Shannon in his original 1948 paper. The problem is also related to coding for magnetic recording, where long strings of 0's are prohibited [5], [370].

We now define increasingly more stringent conditions on codes. Let x^n denote $(x_1, x_2, \ldots, x_n)$.

Definition A code is said to be *nonsingular* if every element of the range of X maps into a different string in $\mathcal{D}^*$; that is,

$$x \neq x' \Rightarrow C(x) \neq C(x'). \tag{5.4}$$

Nonsingularity suffices for an unambiguous description of a single value of X. But we usually wish to send a sequence of values of X. In such cases we can ensure decodability by adding a special symbol (a "comma") between any two codewords. But this is an inefficient use of the special symbol; we can do better by developing the idea of self-punctuating or instantaneous codes. Motivated by the necessity to send sequences of symbols X, we define the extension of a code as follows:

Definition The *extension* C^* of a code C is the mapping from finite-length strings of $\mathcal{X}$ to finite-length strings of $\mathcal{D}$, defined by

$$C(x_1 x_2 \cdots x_n) = C(x_1)C(x_2)\cdots C(x_n), \tag{5.5}$$

where $C(x_1)C(x_2)\cdots C(x_n)$ indicates concatenation of the corresponding codewords.

Example 5.1.4 If $C(x_1) = 00$ and $C(x_2) = 11$, then $C(x_1x_2) = 0011$.

Definition A code is called *uniquely decodable* if its extension is nonsingular.

In other words, any encoded string in a uniquely decodable code has only one possible source string producing it. However, one may have to look at the entire string to determine even the first symbol in the corresponding source string.

Definition A code is called a *prefix code* or an *instantaneous code* if no codeword is a prefix of any other codeword.

An instantaneous code can be decoded without reference to future codewords since the end of a codeword is immediately recognizable. Hence, for an instantaneous code, the symbol x_i can be decoded as soon as we come to the end of the codeword corresponding to it. We need not wait to see the codewords that come later. An instantaneous code is a *self-punctuating code*; we can look down the sequence of code symbols and add the commas to separate the codewords without looking at later symbols. For example, the binary string 01011111010 produced by the code of Example 5.1.1 is parsed as 0,10,111,110,10.

The nesting of these definitions is shown in Figure 5.1. To illustrate the differences between the various kinds of codes, consider the examples of codeword assignments $C(x)$ to $x \in \mathcal{X}$ in Table 5.1. For the nonsingular code, the code string 010 has three possible source sequences: 2 or 14 or 31, and hence the code is not uniquely decodable. The uniquely decodable code is not prefix-free and hence is not instantaneous. To see that it is uniquely decodable, take any code string and start from the beginning. If the first two bits are 00 or 10, they can be decoded immediately. If

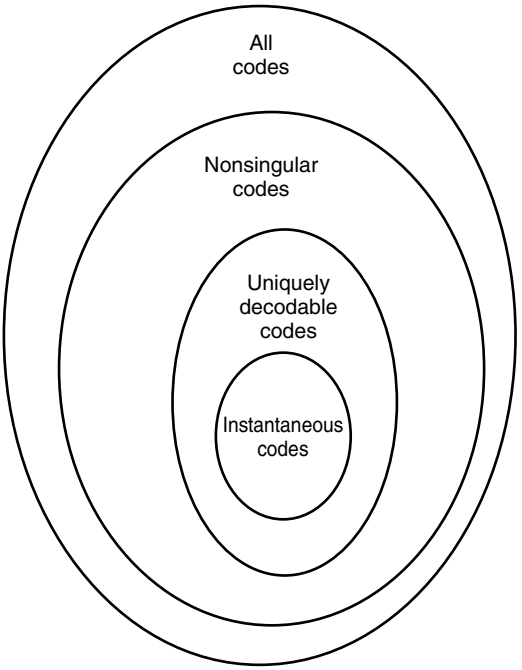


FIGURE 5.1. Classes of codes.

TABLE 5.1 Classes of Codes

X	Singular	Nonsingular, But Not Uniquely Decodable	Uniquely Decodable, But Not Instantaneous	Instantaneous
1	0	0	10	0
2	0	010	00	10
3	0	01	11	110
4	0	10	110	111

the first two bits are 11, we must look at the following bits. If the next bit is a 1, the first source symbol is a 3. If the length of the string of 0's immediately following the 11 is odd, the first codeword must be 110 and the first source symbol must be 4; if the length of the string of 0's is even, the first source symbol is a 3. By repeating this argument, we can see that this code is uniquely decodable. Sardinas and Patterson [455] have devised a finite test for unique decodability, which involves forming sets of possible suffixes to the codewords and eliminating them systematically. The test is described more fully in Problem 5.5.27. The fact that the last code in Table 5.1 is instantaneous is obvious since no codeword is a prefix of any other.

5.2 KRAFT INEQUALITY

We wish to construct instantaneous codes of minimum expected length to describe a given source. It is clear that we cannot assign short codewords to all source symbols and still be prefix-free. The set of codeword lengths possible for instantaneous codes is limited by the following inequality.

Theorem 5.2.1 (*Kraft inequality*) *For any instantaneous code (prefix code) over an alphabet of size D, the codeword lengths* $l_1, l_2, \ldots, l_m$ *must satisfy the inequality*

$$\sum_i D^{-l_i} \leq 1. \tag{5.6}$$

Conversely, given a set of codeword lengths that satisfy this inequality, there exists an instantaneous code with these word lengths.

Proof: Consider a D-ary tree in which each node has D children. Let the branches of the tree represent the symbols of the codeword. For example, the D branches arising from the root node represent the D possible values of the first symbol of the codeword. Then each codeword is represented

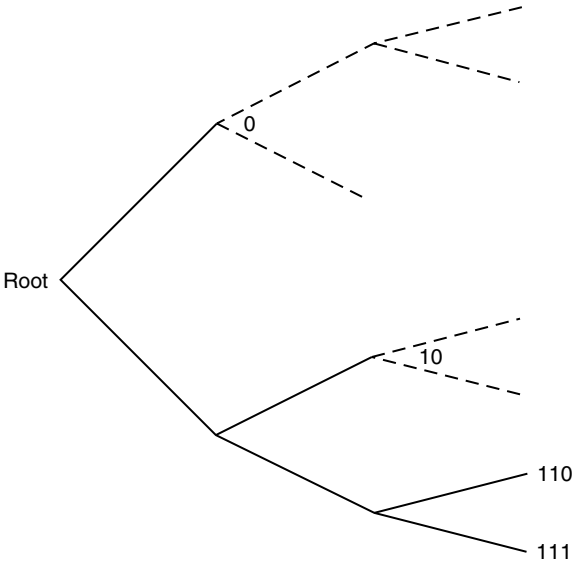


FIGURE 5.2. Code tree for the Kraft inequality.

by a leaf on the tree. The path from the root traces out the symbols of the codeword. A binary example of such a tree is shown in Figure 5.2. The prefix condition on the codewords implies that no codeword is an ancestor of any other codeword on the tree. Hence, each codeword eliminates its descendants as possible codewords.

Let $l_{\max}$ be the length of the longest codeword of the set of codewords. Consider all nodes of the tree at level $l_{\max}$. Some of them are codewords, some are descendants of codewords, and some are neither. A codeword at level l_i has $D^{l_{\max}-l_i}$ descendants at level $l_{\max}$. Each of these descendant sets must be disjoint. Also, the total number of nodes in these sets must be less than or equal to $D^{l_{\max}}$. Hence, summing over all the codewords, we have

$$\sum D^{l_{\max}-l_i} \leq D^{l_{\max}} \tag{5.7}$$

or

$$\sum D^{-l_i} \leq 1, \tag{5.8}$$

which is the Kraft inequality.

Conversely, given any set of codeword lengths $l_1, l_2, \ldots, l_m$ that satisfy the Kraft inequality, we can always construct a tree like the one in

Figure 5.2. Label the first node (lexicographically) of depth l_1 as codeword 1, and remove its descendants from the tree. Then label the first remaining node of depth l_2 as codeword 2, and so on. Proceeding this way, we construct a prefix code with the specified $l_1, l_2, \ldots, l_m$. □

We now show that an infinite prefix code also satisfies the Kraft inequality.

Theorem 5.2.2 (*Extended Kraft Inequality*) *For any countably infinite set of codewords that form a prefix code, the codeword lengths satisfy the extended Kraft inequality,*

$$\sum_{i=1}^{\infty} D^{-l_i} \leq 1. \tag{5.9}$$

Conversely, given any $l_1, l_2, \ldots$ satisfying the extended Kraft inequality, we can construct a prefix code with these codeword lengths.

Proof: Let the D-ary alphabet be $\{0, 1, \ldots, D-1\}$. Consider the ith codeword $y_1 y_2 \cdots y_{l_i}$. Let $0.y_1 y_2 \cdots y_{l_i}$ be the real number given by the D-ary expansion

$$0.y_1 y_2 \cdots y_{l_i} = \sum_{j=1}^{l_i} y_j D^{-j}. \tag{5.10}$$

This codeword corresponds to the interval

$$\left[0.y_1 y_2 \cdots y_{l_i}, 0.y_1 y_2 \cdots y_{l_i} + \frac{1}{D^{l_i}}\right), \tag{5.11}$$

the set of all real numbers whose D-ary expansion begins with $0.y_1 y_2 \cdots y_{l_i}$. This is a subinterval of the unit interval [0, 1]. By the prefix condition, these intervals are disjoint. Hence, the sum of their lengths has to be less than or equal to 1. This proves that

$$\sum_{i=1}^{\infty} D^{-l_i} \leq 1. \tag{5.12}$$

Just as in the finite case, we can reverse the proof to construct the code for a given $l_1, l_2, \ldots$ that satisfies the Kraft inequality. First, reorder the indexing so that $l_1 \leq l_2 \leq \ldots$. Then simply assign the intervals in

order from the low end of the unit interval. For example, if we wish to construct a binary code with $l_1 = 1, l_2 = 2, \ldots$, we assign the intervals $[0, \frac{1}{2}), [\frac{1}{2}, \frac{1}{4}), \ldots$ to the symbols, with corresponding codewords 0, 10, ... □

In Section 5.5 we show that the lengths of codewords for a uniquely decodable code also satisfy the Kraft inequality. Before we do that, we consider the problem of finding the shortest instantaneous code.

5.3 OPTIMAL CODES

In Section 5.2 we proved that any codeword set that satisfies the prefix condition has to satisfy the Kraft inequality and that the Kraft inequality is a sufficient condition for the existence of a codeword set with the specified set of codeword lengths. We now consider the problem of finding the prefix code with the minimum expected length. From the results of Section 5.2, this is equivalent to finding the set of lengths $l_1, l_2, \ldots, l_m$ satisfying the Kraft inequality and whose expected length $L = \sum p_i l_i$ is less than the expected length of any other prefix code. This is a standard optimization problem: Minimize

$$L = \sum p_i l_i \tag{5.13}$$

over all integers $l_1, l_2, \ldots, l_m$ satisfying

$$\sum D^{-l_i} \leq 1. \tag{5.14}$$

A simple analysis by calculus suggests the form of the minimizing l_i^*. We neglect the integer constraint on l_i and assume equality in the constraint. Hence, we can write the constrained minimization using Lagrange multipliers as the minimization of

$$J = \sum p_i l_i + \lambda \left(\sum D^{-l_i} \right). \tag{5.15}$$

Differentiating with respect to l_i, we obtain

$$\frac{\partial J}{\partial l_i} = p_i - \lambda D^{-l_i} \log_e D. \tag{5.16}$$

Setting the derivative to 0, we obtain

$$D^{-l_i} = \frac{p_i}{\lambda \log_e D}. \tag{5.17}$$

Substituting this in the constraint to find λ, we find $\lambda = 1/\log_e D$, and hence

$$p_i = D^{-l_i}, \tag{5.18}$$

yielding optimal code lengths,

$$l_i^* = -\log_D p_i. \tag{5.19}$$

This noninteger choice of codeword lengths yields expected codeword length

$$L^* = \sum p_i l_i^* = -\sum p_i \log_D p_i = H_D(X). \tag{5.20}$$

But since the l_i must be integers, we will not always be able to set the codeword lengths as in (5.19). Instead, we should choose a set of codeword lengths l_i "close" to the optimal set. Rather than demonstrate by calculus that $l_i^* = -\log_D p_i$ is a global minimum, we verify optimality directly in the proof of the following theorem.

Theorem 5.3.1 *The expected length L of any instantaneous D-ary code for a random variable X is greater than or equal to the entropy $H_D(X)$; that is,*

$$L \geq H_D(X), \tag{5.21}$$

with equality if and only if $D^{-l_i} = p_i$.

Proof: We can write the difference between the expected length and the entropy as

$$L - H_D(X) = \sum p_i l_i - \sum p_i \log_D \frac{1}{p_i} \tag{5.22}$$

$$= -\sum p_i \log_D D^{-l_i} + \sum p_i \log_D p_i. \tag{5.23}$$

Letting $r_i = D^{-l_i} / \sum_j D^{-l_j}$ and $c = \sum D^{-l_i}$, we obtain

$$L - H = \sum p_i \log_D \frac{p_i}{r_i} - \log_D c \tag{5.24}$$

$$= D(\mathbf{p}||\mathbf{r}) + \log_D \frac{1}{c} \tag{5.25}$$

$$\geq 0 \tag{5.26}$$

by the nonnegativity of relative entropy and the fact (Kraft inequality) that $c \leq 1$. Hence, $L \geq H$ with equality if and only if $p_i = D^{-l_i}$ (i.e., if and only if $-\log_D p_i$ is an integer for all i). □

Definition A probability distribution is called *D-adic* if each of the probabilities is equal to D^{-n} for some n. Thus, we have equality in the theorem if and only if the distribution of X is D-adic.

The preceding proof also indicates a procedure for finding an optimal code: Find the D-adic distribution that is closest (in the relative entropy sense) to the distribution of X. This distribution provides the set of codeword lengths. Construct the code by choosing the first available node as in the proof of the Kraft inequality. We then have an optimal code for X.

However, this procedure is not easy, since the search for the closest D-adic distribution is not obvious. In the next section we give a good suboptimal procedure (Shannon–Fano coding). In Section 5.6 we describe a simple procedure (Huffman coding) for actually finding the optimal code.

5.4 BOUNDS ON THE OPTIMAL CODE LENGTH

We now demonstrate a code that achieves an expected description length L within 1 bit of the lower bound; that is,

$$H(X) \leq L < H(X) + 1. \tag{5.27}$$

Recall the setup of Section 5.3: We wish to minimize $L = \sum p_i l_i$ subject to the constraint that $l_1, l_2, \ldots, l_m$ are integers and $\sum D^{-l_i} \leq 1$. We proved that the optimal codeword lengths can be found by finding the D-adic probability distribution closest to the distribution of X in relative entropy, that is, by finding the D-adic $\mathbf{r}$ ($r_i = D^{-l_i} / \sum_j D^{-l_j}$) minimizing

$$L - H_D = D(\mathbf{p}||\mathbf{r}) - \log\left(\sum D^{-l_i}\right) \geq 0. \tag{5.28}$$

The choice of word lengths $l_i = \log_D \frac{1}{p_i}$ yields $L = H$. Since $\log_D \frac{1}{p_i}$ may not equal an integer, we round it up to give integer word-length assignments,

$$l_i = \left\lceil \log_D \frac{1}{p_i} \right\rceil, \tag{5.29}$$

where $\lceil x \rceil$ is the smallest integer $\geq x$. These lengths satisfy the Kraft inequality since

$$\sum D^{-\lceil \log \frac{1}{p_i} \rceil} \leq \sum D^{-\log \frac{1}{p_i}} = \sum p_i = 1. \tag{5.30}$$

This choice of codeword lengths satisfies

$$\log_D \frac{1}{p_i} \leq l_i < \log_D \frac{1}{p_i} + 1. \tag{5.31}$$

Multiplying by p_i and summing over i, we obtain

$$H_D(X) \leq L < H_D(X) + 1. \tag{5.32}$$

Since an optimal code can only be better than this code, we have the following theorem.

Theorem 5.4.1 *Let $l_1^*, l_2^*, \ldots, l_m^*$ be optimal codeword lengths for a source distribution* $\mathbf{p}$ *and a D-ary alphabet, and let L^* be the associated expected length of an optimal code ($L^* = \sum p_i l_i^*$). Then*

$$H_D(X) \leq L^* < H_D(X) + 1. \tag{5.33}$$

Proof: Let $l_i = \lceil \log_D \frac{1}{p_i} \rceil$. Then l_i satisfies the Kraft inequality and from (5.32) we have

$$H_D(X) \leq L = \sum p_i l_i < H_D(X) + 1. \tag{5.34}$$

But since L^*, the expected length of the optimal code, is less than $L = \sum p_i l_i$, and since $L^* \geq H_D$ from Theorem 5.3.1, we have the theorem. □

In Theorem 5.4.1 there is an overhead that is at most 1 bit, due to the fact that $\log \frac{1}{p_i}$ is not always an integer. We can reduce the overhead per symbol by spreading it out over many symbols. With this in mind, let us consider a system in which we send a sequence of n symbols from X. The symbols are assumed to be drawn i.i.d. according to $p(x)$. We can consider these n symbols to be a supersymbol from the alphabet $\mathcal{X}^n$.

Define L_n to be the expected codeword length per input symbol, that is, if $l(x_1, x_2, \ldots, x_n)$ is the length of the binary codeword associated

with $(x_1, x_2, \ldots, x_n)$ (for the rest of this section, we assume that $D = 2$, for simplicity), then

$$L_n = \frac{1}{n} \sum p(x_1, x_2, \ldots, x_n) l(x_1, x_2, \ldots, x_n) = \frac{1}{n} El(X_1, X_2, \ldots, X_n). \tag{5.35}$$

We can now apply the bounds derived above to the code:

$$H(X_1, X_2, \ldots, X_n) \leq El(X_1, X_2, \ldots, X_n) < H(X_1, X_2, \ldots, X_n) + 1. \tag{5.36}$$

Since $X_1, X_2, \ldots, X_n$ are i.i.d., $H(X_1, X_2, \ldots, X_n) = \sum H(X_i) = nH(X)$. Dividing (5.36) by n, we obtain

$$H(X) \leq L_n < H(X) + \frac{1}{n}. \tag{5.37}$$

Hence, by using large block lengths we can achieve an expected codelength per symbol arbitrarily close to the entropy.

We can use the same argument for a sequence of symbols from a stochastic process that is not necessarily i.i.d. In this case, we still have the bound

$$H(X_1, X_2, \ldots, X_n) \leq El(X_1, X_2, \ldots, X_n) < H(X_1, X_2, \ldots, X_n) + 1. \tag{5.38}$$

Dividing by n again and defining L_n to be the expected description length per symbol, we obtain

$$\frac{H(X_1, X_2, \ldots, X_n)}{n} \leq L_n < \frac{H(X_1, X_2, \ldots, X_n)}{n} + \frac{1}{n}. \tag{5.39}$$

If the stochastic process is stationary, then $H(X_1, X_2, \ldots, X_n)/n \to H(\mathcal{X})$, and the expected description length tends to the entropy rate as $n \to \infty$. Thus, we have the following theorem:

Theorem 5.4.2 *The minimum expected codeword length per symbol satisfies*

$$\frac{H(X_1, X_2, \ldots, X_n)}{n} \leq L_n^* < \frac{H(X_1, X_2, \ldots, X_n)}{n} + \frac{1}{n}. \tag{5.40}$$

Moreover, if $X_1, X_2, \ldots, X_n$ is a stationary stochastic process,

$$L_n^* \to H(\mathcal{X}), \tag{5.41}$$

where $H(\mathcal{X})$ is the entropy rate of the process.

This theorem provides another justification for the definition of entropy rate—it is the expected number of bits per symbol required to describe the process.

Finally, we ask what happens to the expected description length if the code is designed for the wrong distribution. For example, the wrong distribution may be the best estimate that we can make of the unknown true distribution. We consider the Shannon code assignment $l(x) = \left\lceil \log \frac{1}{q(x)} \right\rceil$ designed for the probability mass function $q(x)$. Suppose that the true probability mass function is $p(x)$. Thus, we will not achieve expected length $L \approx H(p) = -\sum p(x) \log p(x)$. We now show that the increase in expected description length due to the incorrect distribution is the relative entropy $D(p||q)$. Thus, $D(p||q)$ has a concrete interpretation as the increase in descriptive complexity due to incorrect information.

Theorem 5.4.3 (*Wrong code*) *The expected length under $p(x)$ of the code assignment $l(x) = \left\lceil \log \frac{1}{q(x)} \right\rceil$ satisfies*

$$H(p) + D(p||q) \leq E_p l(X) < H(p) + D(p||q) + 1. \tag{5.42}$$

Proof: The expected codelength is

$$\begin{aligned} El(X) &= \sum_x p(x) \left\lceil \log \frac{1}{q(x)} \right\rceil && (5.43)\\ &< \sum_x p(x) \left(\log \frac{1}{q(x)} + 1 \right) && (5.44)\\ &= \sum_x p(x) \log \frac{p(x)}{q(x)} \frac{1}{p(x)} + 1 && (5.45)\\ &= \sum_x p(x) \log \frac{p(x)}{q(x)} + \sum_x p(x) \log \frac{1}{p(x)} + 1 && (5.46)\\ &= D(p||q) + H(p) + 1. && (5.47) \end{aligned}$$

The lower bound can be derived similarly. □

Thus, believing that the distribution is $q(x)$ when the true distribution is $p(x)$ incurs a penalty of $D(p||q)$ in the average description length.

5.5 KRAFT INEQUALITY FOR UNIQUELY DECODABLE CODES

We have proved that any instantaneous code must satisfy the Kraft inequality. The class of uniquely decodable codes is larger than the class of

instantaneous codes, so one expects to achieve a lower expected codeword length if L is minimized over all uniquely decodable codes. In this section we prove that the class of uniquely decodable codes does not offer any further possibilities for the set of codeword lengths than do instantaneous codes. We now give Karush's elegant proof of the following theorem.

Theorem 5.5.1 (*McMillan*) *The codeword lengths of any uniquely decodable D-ary code must satisfy the Kraft inequality*

$$\sum D^{-l_i} \leq 1. \tag{5.48}$$

Conversely, given a set of codeword lengths that satisfy this inequality, it is possible to construct a uniquely decodable code with these codeword lengths.

Proof: Consider C^k, the kth extension of the code (i.e., the code formed by the concatenation of k repetitions of the given uniquely decodable code C). By the definition of unique decodability, the kth extension of the code is nonsingular. Since there are only D^n different D-ary strings of length n, unique decodability implies that the number of code sequences of length n in the kth extension of the code must be no greater than D^n. We now use this observation to prove the Kraft inequality.

Let the codeword lengths of the symbols $x \in \mathcal{X}$ be denoted by $l(x)$. For the extension code, the length of the code sequence is

$$l(x_1, x_2, \ldots, x_k) = \sum_{i=1}^{k} l(x_i). \tag{5.49}$$

The inequality that we wish to prove is

$$\sum_{x \in \mathcal{X}} D^{-l(x)} \leq 1. \tag{5.50}$$

The trick is to consider the kth power of this quantity. Thus,

$$\left(\sum_{x \in \mathcal{X}} D^{-l(x)} \right)^k = \sum_{x_1 \in \mathcal{X}} \sum_{x_2 \in \mathcal{X}} \cdots \sum_{x_k \in \mathcal{X}} D^{-l(x_1)} D^{-l(x_2)} \cdots D^{-l(x_k)} \tag{5.51}$$

$$= \sum_{x_1, x_2, \ldots, x_k \in \mathcal{X}^k} D^{-l(x_1)} D^{-l(x_2)} \cdots D^{-l(x_k)} \tag{5.52}$$

$$= \sum_{x^k \in \mathcal{X}^k} D^{-l(x^k)}, \tag{5.53}$$

by (5.49). We now gather the terms by word lengths to obtain

$$\sum_{x^k \in \mathcal{X}^k} D^{-l(x^k)} = \sum_{m=1}^{kl_{\max}} a(m) D^{-m}, \tag{5.54}$$

where $l_{\max}$ is the maximum codeword length and $a(m)$ is the number of source sequences x^k mapping into codewords of length m. But the code is uniquely decodable, so there is at most one sequence mapping into each code m-sequence and there are at most D^m code m-sequences. Thus, $a(m) \leq D^m$, and we have

$$\left(\sum_{x \in \mathcal{X}} D^{-l(x)} \right)^k = \sum_{m=1}^{kl_{\max}} a(m) D^{-m} \tag{5.55}$$

$$\leq \sum_{m=1}^{kl_{\max}} D^m D^{-m} \tag{5.56}$$

$$= kl_{\max} \tag{5.57}$$

and hence

$$\sum_j D^{-l_j} \leq (kl_{\max})^{1/k}. \tag{5.58}$$

Since this inequality is true for all k, it is true in the limit as $k \to \infty$. Since $(kl_{\max})^{1/k} \to 1$, we have

$$\sum_j D^{-l_j} \leq 1, \tag{5.59}$$

which is the Kraft inequality.

Conversely, given any set of $l_1, l_2, \ldots, l_m$ satisfying the Kraft inequality, we can construct an instantaneous code as proved in Section 5.2. Since every instantaneous code is uniquely decodable, we have also constructed a uniquely decodable code. □

Corollary *A uniquely decodable code for an infinite source alphabet $\mathcal{X}$ also satisfies the Kraft inequality.*

Proof: The point at which the preceding proof breaks down for infinite $|\mathcal{X}|$ is at (5.58), since for an infinite code $l_{\max}$ is infinite. But there is a

simple fix to the proof. Any subset of a uniquely decodable code is also uniquely decodable; thus, any finite subset of the infinite set of codewords satisfies the Kraft inequality. Hence,

$$\sum_{i=1}^{\infty} D^{-l_i} = \lim_{N\to\infty} \sum_{i=1}^{N} D^{-l_i} \leq 1. \tag{5.60}$$

Given a set of word lengths $l_1, l_2, \ldots$ that satisfy the Kraft inequality, we can construct an instantaneous code as in Section 5.4. Since instantaneous codes are uniquely decodable, we have constructed a uniquely decodable code with an infinite number of codewords. So the McMillan theorem also applies to infinite alphabets. □

The theorem implies a rather surprising result—that the class of uniquely decodable codes does not offer any further choices for the set of codeword lengths than the class of prefix codes. The set of achievable codeword lengths is the same for uniquely decodable and instantaneous codes. Hence, the bounds derived on the optimal codeword lengths continue to hold even when we expand the class of allowed codes to the class of all uniquely decodable codes.

5.6 HUFFMAN CODES

An optimal (shortest expected length) prefix code for a given distribution can be constructed by a simple algorithm discovered by Huffman [283]. We will prove that any other code for the same alphabet cannot have a lower expected length than the code constructed by the algorithm. Before we give any formal proofs, let us introduce Huffman codes with some examples.

Example 5.6.1 Consider a random variable X taking values in the set $\mathcal{X} = \{1, 2, 3, 4, 5\}$ with probabilities 0.25, 0.25, 0.2, 0.15, 0.15, respectively. We expect the optimal binary code for X to have the longest codewords assigned to the symbols 4 and 5. These two lengths must be equal, since otherwise we can delete a bit from the longer codeword and still have a prefix code, but with a shorter expected length. In general, we can construct a code in which the two longest codewords differ only in the last bit. For this code, we can combine the symbols 4 and 5 into a single source symbol, with a probability assignment 0.30. Proceeding this way, combining the two least likely symbols into one symbol until we are finally left with only one symbol, and then assigning codewords to the symbols, we obtain the following table:

Codeword Length	Codeword	X	Probability
2	01	1	0.25 0.3 0.45 0.55 1
2	10	2	0.25 0.25 0.3 0.45
2	11	3	0.2 0.25 0.25
3	000	4	0.15 0.2
3	001	5	0.15

This code has average length 2.3 bits.

Example 5.6.2 Consider a ternary code for the same random variable. Now we combine the three least likely symbols into one supersymbol and obtain the following table:

Codeword	X	Probability
1	1	0.25 0.5 1
2	2	0.25 0.25
00	3	0.2 0.25
01	4	0.15
02	5	0.15

This code has an average length of 1.5 ternary digits.

Example 5.6.3 If $D \geq 3$, we may not have a sufficient number of symbols so that we can combine them D at a time. In such a case, we add dummy symbols to the end of the set of symbols. The dummy symbols have probability 0 and are inserted to fill the tree. Since at each stage of the reduction, the number of symbols is reduced by $D - 1$, we want the total number of symbols to be $1 + k(D - 1)$, where k is the number of merges. Hence, we add enough dummy symbols so that the total number of symbols is of this form. For example:

Codeword	X	Probability
1	1	0.25 0.25 0.5 1.0
2	2	0.25 0.25 0.25
01	3	0.2 0.2
02	4	0.1 0.2
000	5	0.1 0.1
001	6	0.1
002	Dummy	0.0

This code has an average length of 1.7 ternary digits.

A proof of the optimality of Huffman coding is given in Section 5.8.

5.7 SOME COMMENTS ON HUFFMAN CODES

1. *Equivalence of source coding and 20 questions*. We now digress to show the equivalence of coding and the game "20 questions". Suppose that we wish to find the most efficient series of yes–no questions to determine an object from a class of objects. Assuming that we know the probability distribution on the objects, can we find the most efficient sequence of questions? (To determine an object, we need to ensure that the responses to the sequence of questions uniquely identifies the object from the set of possible objects; it is not necessary that the last question have a "yes" answer.)

   We first show that a sequence of questions is equivalent to a code for the object. Any question depends only on the answers to the questions before it. Since the sequence of answers uniquely determines the object, each object has a different sequence of answers, and if we represent the yes–no answers by 0's and 1's, we have a binary code for the set of objects. The average length of this code is the average number of questions for the questioning scheme.

   Also, from a binary code for the set of objects, we can find a sequence of questions that correspond to the code, with the average number of questions equal to the expected codeword length of the code. The first question in this scheme becomes: Is the first bit equal to 1 in the object's codeword?

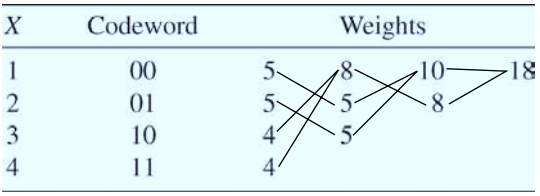
   Since the Huffman code is the best source code for a random variable, the optimal series of questions is that determined by the Huffman code. In Example 5.6.1 the optimal first question is: Is X equal to 2 or 3? The answer to this determines the first bit of the Huffman code. Assuming that the answer to the first question is "yes," the next question should be "Is X equal to 3?", which determines the second bit. However, we need not wait for the answer to the first question to ask the second. We can ask as our second question "Is X equal to 1 or 3?", determining the second bit of the Huffman code independent of the first.

   The expected number of questions EQ in this optimal scheme satisfies

$$H(X) \leq EQ < H(X) + 1. \tag{5.61}$$

2. *Huffman coding for weighted codewords*. Huffman's algorithm for minimizing $\sum p_i l_i$ can be applied to any set of numbers $p_i \geq 0$, regardless of $\sum p_i$. In this case, the Huffman code minimizes the sum of weighted code lengths $\sum w_i l_i$ rather than the average code length.

Example 5.7.1 We perform the weighted minimization using the same algorithm.



In this case the code minimizes the weighted sum of the codeword lengths, and the minimum weighted sum is 36.

3. *Huffman coding and "slice" questions* (*Alphabetic codes*). We have described the equivalence of source coding with the game of 20 questions. The optimal sequence of questions corresponds to an optimal source code for the random variable. However, Huffman codes ask arbitrary questions of the form "Is $X \in A$?" for any set $A \subseteq \{1, 2, \ldots, m\}$.

Now we consider the game "20 questions" with a restricted set of questions. Specifically, we assume that the elements of $\mathcal{X} = \{1, 2, \ldots, m\}$ are ordered so that $p_1 \geq p_2 \geq \cdots \geq p_m$ and that the only questions allowed are of the form "Is $X > a$?" for some a. The Huffman code constructed by the Huffman algorithm may not correspond to *slices* (sets of the form $\{x : x < a\}$). If we take the codeword lengths ($l_1 \leq l_2 \leq \cdots \leq l_m$, by Lemma 5.8.1) derived from the Huffman code and use them to assign the symbols to the code tree by taking the first available node at the corresponding level, we will construct another optimal code. However, unlike the Huffman code itself, this code is a *slice code*, since each question (each bit of the code) splits the tree into sets of the form $\{x : x > a\}$ and $\{x : x < a\}$.

We illustrate this with an example.

Example 5.7.2 Consider the first example of Section 5.6. The code that was constructed by the Huffman coding procedure is not a

slice code. But using the codeword lengths from the Huffman procedure, namely, {2, 2, 2, 3, 3}, and assigning the symbols to the first available node on the tree, we obtain the following code for this random variable:

$$1 \to 00, \quad 2 \to 01, \quad 3 \to 10, \quad 4 \to 110, \quad 5 \to 111$$

It can be verified that this code is a slice code, codes known as *alphabetic codes* because the codewords are ordered alphabetically.

4. *Huffman codes and Shannon codes*. Using codeword lengths of $\lceil \log \frac{1}{p_i} \rceil$ (which is called *Shannon coding*) may be much worse than the optimal code for some particular symbol. For example, consider two symbols, one of which occurs with probability 0.9999 and the other with probability 0.0001. Then using codeword lengths of $\lceil \log \frac{1}{p_i} \rceil$ gives codeword lengths of 1 bit and 14 bits, respectively. The optimal codeword length is obviously 1 bit for both symbols. Hence, the codeword for the infrequent symbol is much longer in the Shannon code than in the optimal code.

   Is it true that the codeword lengths for an optimal code are always less than $\lceil \log \frac{1}{p_i} \rceil$? The following example illustrates that this is not always true.

Example 5.7.3 Consider a random variable X with a distribution $\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{4}, \frac{1}{12}\right)$. The Huffman coding procedure results in codeword lengths of $(2, 2, 2, 2)$ or $(1, 2, 3, 3)$ [depending on where one puts the merged probabilities, as the reader can verify (Problem 5.5.12)]. Both these codes achieve the same expected codeword length. In the second code, the third symbol has length 3, which is greater than $\lceil \log \frac{1}{p_3} \rceil$. Thus, the codeword length for a Shannon code could be less than the codeword length of the corresponding symbol of an optimal (Huffman) code. This example also illustrates the fact that the set of codeword lengths for an optimal code is not unique (there may be more than one set of lengths with the same expected value).

Although either the Shannon code or the Huffman code can be shorter for individual symbols, the Huffman code is shorter on average. Also, the Shannon code and the Huffman code differ by less than 1 bit in expected codelength (since both lie between H and $H + 1$.)

5. *Fano codes*. Fano proposed a suboptimal procedure for constructing a source code, which is similar to the idea of slice codes. In his method we first order the probabilities in decreasing order. Then we choose k such that $\left|\sum_{i=1}^{k} p_i - \sum_{i=k+1}^{m} p_i\right|$ is minimized. This point divides the source symbols into two sets of almost equal probability. Assign 0 for the first bit of the upper set and 1 for the lower set. Repeat this process for each subset. By this recursive procedure, we obtain a code for each source symbol. This scheme, although not optimal in general, achieves $L(C) \leq H(X) + 2$. (See [282].)

5.8 OPTIMALITY OF HUFFMAN CODES

We prove by induction that the binary Huffman code is optimal. It is important to remember that there are many optimal codes: inverting all the bits or exchanging two codewords of the same length will give another optimal code. The Huffman procedure constructs one such optimal code. To prove the optimality of Huffman codes, we first prove some properties of a particular optimal code.

Without loss of generality, we will assume that the probability masses are ordered, so that $p_1 \geq p_2 \geq \cdots \geq p_m$. Recall that a code is optimal if $\sum p_i l_i$ is minimal.

Lemma 5.8.1 *For any distribution, there exists an optimal instantaneous code (with minimum expected length) that satisfies the following properties:*

1. *The lengths are ordered inversely with the probabilities (i.e., if $p_j > p_k$, then $l_j \leq l_k$).*
2. *The two longest codewords have the same length.*
3. *Two of the longest codewords differ only in the last bit and correspond to the two least likely symbols.*

Proof: The proof amounts to swapping, trimming, and rearranging, as shown in Figure 5.3. Consider an optimal code C_m:

- *If $p_j > p_k$, then $l_j \leq l_k$.* Here we swap codewords.
  Consider C'_m, with the codewords j and k of C_m interchanged. Then

$$L(C'_m) - L(C_m) = \sum p_i l_i{}' - \sum p_i l_i \tag{5.62}$$

$$= p_j l_k + p_k l_j - p_j l_j - p_k l_k \tag{5.63}$$

$$= (p_j - p_k)(l_k - l_j). \tag{5.64}$$

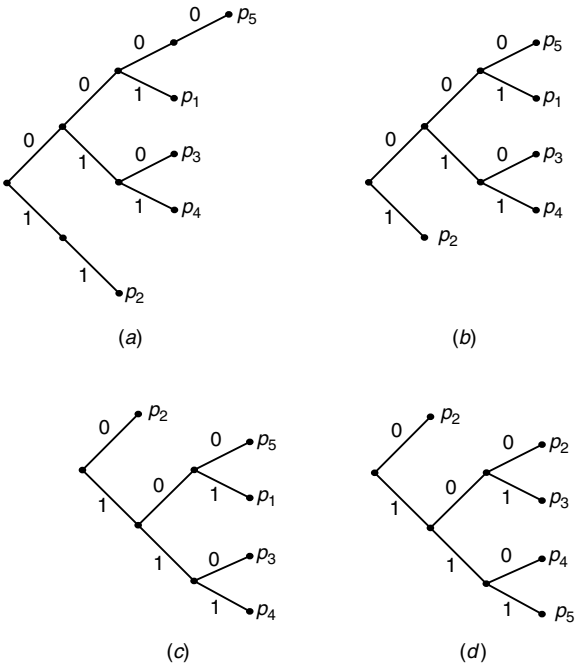


FIGURE 5.3. Properties of optimal codes. We assume that $p_1 \geq p_2 \geq \cdots \geq p_m$. A possible instantaneous code is given in (a). By trimming branches without siblings, we improve the code to (b). We now rearrange the tree as shown in (c), so that the word lengths are ordered by increasing length from top to bottom. Finally, we swap probability assignments to improve the expected depth of the tree, as shown in (d). Every optimal code can be rearranged and swapped into canonical form as in (d), where $l_1 \leq l_2 \leq \cdots \leq l_m$ and $l_{m-1} = l_m$, and the last two codewords differ only in the last bit.

But $p_j - p_k > 0$, and since C_m is optimal, $L(C'_m) - L(C_m) \geq 0$. Hence, we must have $l_k \geq l_j$. Thus, C_m itself satisfies property 1.

- *The two longest codewords are of the same length.* Here we trim the codewords. If the two longest codewords are not of the same length, one can delete the last bit of the longer one, preserving the prefix property and achieving lower expected codeword length. Hence, the two longest codewords must have the same length. By property 1, the longest codewords must belong to the least probable source symbols.
- *The two longest codewords differ only in the last bit and correspond to the two least likely symbols.* Not all optimal codes satisfy this property, but by rearranging, we can find an optimal code that does. If there is a maximal-length codeword without a sibling, we can delete the last bit of the codeword and still satisfy the prefix property. This reduces the average codeword length and contradicts the optimality

of the code. Hence, every maximal-length codeword in any optimal code has a sibling. Now we can exchange the longest codewords so that the two lowest-probability source symbols are associated with two siblings on the tree. This does not change the expected length, $\sum p_i l_i$. Thus, the codewords for the two lowest-probability source symbols have maximal length and agree in all but the last bit.

Summarizing, we have shown that if $p_1 \geq p_2 \geq \cdots \geq p_m$, there exists an optimal code with $l_1 \leq l_2 \leq \cdots \leq l_{m-1} = l_m$, and codewords $C(x_{m-1})$ and $C(x_m)$ that differ only in the last bit. □

Thus, we have shown that there exists an optimal code satisfying the properties of the lemma. We call such codes *canonical* codes. For any probability mass function for an alphabet of size m, $\mathbf{p} = (p_1, p_2, \ldots, p_m)$ with $p_1 \geq p_2 \geq \cdots \geq p_m$, we define the Huffman reduction $\mathbf{p}' = (p_1, p_2, \ldots, p_{m-2}, p_{m-1} + p_m)$ over an alphabet of size $m - 1$ (Figure 5.4). Let $C^*_{m-1}(\mathbf{p}')$ be an optimal code for $\mathbf{p}'$, and let $C^*_m(\mathbf{p})$ be the canonical optimal code for $\mathbf{p}$.

The proof of optimality will follow from two constructions: First, we expand an optimal code for $\mathbf{p}'$ to construct a code for $\mathbf{p}$, and then we

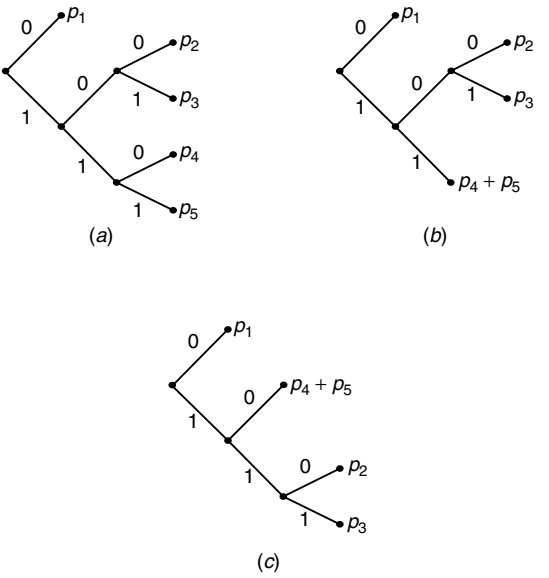


FIGURE 5.4. Induction step for Huffman coding. Let $p_1 \geq p_2 \geq \cdots \geq p_5$. A canonical optimal code is illustrated in (*a*). Combining the two lowest probabilities, we obtain the code in (*b*). Rearranging the probabilities in decreasing order, we obtain the canonical code in (*c*) for $m - 1$ symbols.

condense an optimal canonical code for $\mathbf{p}$ to construct a code for the Huffman reduction $\mathbf{p}'$. Comparing the average codeword lengths for the two codes establishes that the optimal code for $\mathbf{p}$ can be obtained by extending the optimal code for $\mathbf{p}'$.

From the optimal code for $\mathbf{p}'$, we construct an extension code for m elements as follows: Take the codeword in C^*_{m-1} corresponding to weight $p_{m-1} + p_m$ and extend it by adding a 0 to form a codeword for symbol $m-1$ and by adding 1 to form a codeword for symbol m. The code construction is illustrated as follows:

$$\begin{array}{cccll} & C^*_{m-1}(\mathbf{p}') & & C_m(\mathbf{p}) & \\ p_1 & w'_1 & l'_1 & w_1 = w'_1 & l_1 = l'_1 \\ p_2 & w'_2 & l'_2 & w_2 = w'_2 & l_2 = l'_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ p_{m-2} & w'_{m-2} & l'_{m-2} & w_{m-2} = w'_{m-2} & l_{m-2} = l'_{m-2} \\ p_{m-1} + p_m & w'_{m-1} & l'_{m-1} & w_{m-1} = w'_{m-1}0 & l_{m-1} = l'_{m-1} + 1 \\ & & & w_m = w'_{m-1}1 & l_m = l'_{m-1} + 1 \end{array} \tag{5.65}$$

Calculation of the average length $\sum_i p'_i l'_i$ shows that

$$L(\mathbf{p}) = L^*(\mathbf{p}') + p_{m-1} + p_m. \tag{5.66}$$

Similarly, from the canonical code for $\mathbf{p}$, we construct a code for $\mathbf{p}'$ by merging the codewords for the two lowest-probability symbols $m-1$ and m with probabilities p_{m-1} and p_m, which are siblings by the properties of the canonical code. The new code for $\mathbf{p}'$ has average length

$$L(\mathbf{p}') = \sum_{i=1}^{m-2} p_i l_i + p_{m-1}(l_{m-1} - 1) + p_m(l_m - 1) \tag{5.67}$$

$$= \sum_{i=1}^{m} p_i l_i - p_{m-1} - p_m \tag{5.68}$$

$$= L^*(\mathbf{p}) - p_{m-1} - p_m. \tag{5.69}$$

Adding (5.66) and (5.69) together, we obtain

$$L(\mathbf{p}') + L(\mathbf{p}) = L^*(\mathbf{p}') + L^*(\mathbf{p}) \tag{5.70}$$

or

$$(L(\mathbf{p}') - L^*(\mathbf{p}')) + (L(\mathbf{p}) - L^*(\mathbf{p})) = 0. \tag{5.71}$$

Now examine the two terms in (5.71). By assumption, since $L^*(\mathbf{p}')$ is the optimal length for $\mathbf{p}'$, we have $L(\mathbf{p}') - L^*(\mathbf{p}') \geq 0$. Similarly, the length of the extension of the optimal code for $\mathbf{p}'$ has to have an average length at least as large as the optimal code for $\mathbf{p}$ [i.e., $L(\mathbf{p}) - L^*(\mathbf{p}) \geq 0$]. But the sum of two nonnegative terms can only be 0 if both of them are 0, which implies that $L(\mathbf{p}) = L^*(\mathbf{p})$ (i.e., the extension of the optimal code for $\mathbf{p}'$ is optimal for $\mathbf{p}$).

Consequently, if we start with an optimal code for $\mathbf{p}'$ with $m-1$ symbols and construct a code for m symbols by extending the codeword corresponding to $p_{m-1} + p_m$, the new code is also optimal. Starting with a code for two elements, in which case the optimal code is obvious, we can by induction extend this result to prove the following theorem.

Theorem 5.8.1 *Huffman coding is optimal; that is, if C^* is a Huffman code and C' is any other uniquely decodable code, $L(C^*) \leq L(C')$.*

Although we have proved the theorem for a binary alphabet, the proof can be extended to establishing optimality of the Huffman coding algorithm for a D-ary alphabet as well. Incidentally, we should remark that Huffman coding is a "greedy" algorithm in that it coalesces the two least likely symbols at each stage. The proof above shows that this local optimality ensures global optimality of the final code.

5.9 SHANNON–FANO–ELIAS CODING

In Section 5.4 we showed that the codeword lengths $l(x) = \left\lceil \log \frac{1}{p(x)} \right\rceil$ satisfy the Kraft inequality and can therefore be used to construct a uniquely decodable code for the source. In this section we describe a simple constructive procedure that uses the cumulative distribution function to allot codewords.

Without loss of generality, we can take $\mathcal{X} = \{1, 2, \ldots, m\}$. Assume that $p(x) > 0$ for all x. The cumulative distribution function $F(x)$ is defined as

$$F(x) = \sum_{a \leq x} p(a). \tag{5.72}$$

This function is illustrated in Figure 5.5. Consider the modified cumulative distribution function

$$\overline{F}(x) = \sum_{a < x} p(a) + \frac{1}{2} p(x), \tag{5.73}$$

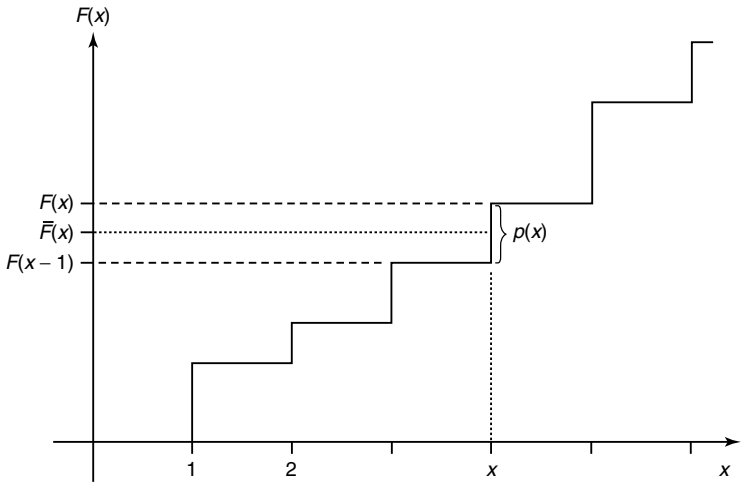


FIGURE 5.5. Cumulative distribution function and Shannon–Fano–Elias coding.

where $\overline{F}(x)$ denotes the sum of the probabilities of all symbols less than x plus half the probability of the symbol x. Since the random variable is discrete, the cumulative distribution function consists of steps of size $p(x)$. The value of the function $\overline{F}(x)$ is the midpoint of the step corresponding to x.

Since all the probabilities are positive, $\overline{F}(a) \neq \overline{F}(b)$ if $a \neq b$, and hence we can determine x if we know $\overline{F}(x)$. Merely look at the graph of the cumulative distribution function and find the corresponding x. Thus, the value of $\overline{F}(x)$ can be used as a code for x.

But, in general, $\overline{F}(x)$ is a real number expressible only by an infinite number of bits. So it is not efficient to use the exact value of $\overline{F}(x)$ as a code for x. If we use an approximate value, what is the required accuracy?

Assume that we truncate $\overline{F}(x)$ to $l(x)$ bits (denoted by $\lfloor \overline{F}(x) \rfloor_{l(x)}$). Thus, we use the first $l(x)$ bits of $\overline{F}(x)$ as a code for x. By definition of rounding off, we have

$$\overline{F}(x) - \lfloor \overline{F}(x) \rfloor_{l(x)} < \frac{1}{2^{l(x)}}. \tag{5.74}$$

If $l(x) = \left\lceil \log \frac{1}{p(x)} \right\rceil + 1$, then

$$\frac{1}{2^{l(x)}} < \frac{p(x)}{2} = \overline{F}(x) - F(x-1), \tag{5.75}$$

and therefore $\lfloor \overline{F}(x) \rfloor_{l(x)}$ lies within the step corresponding to x. Thus, $l(x)$ bits suffice to describe x.

In addition to requiring that the codeword identify the corresponding symbol, we also require the set of codewords to be prefix-free. To check whether the code is prefix-free, we consider each codeword $z_1 z_2 \cdots z_l$ to represent not a point but the interval $\left[0.z_1 z_2 \cdots z_l, 0.z_1 z_2 \cdots z_l + \frac{1}{2^l}\right)$. The code is prefix-free if and only if the intervals corresponding to codewords are disjoint.

We now verify that the code above is prefix-free. The interval corresponding to any codeword has length $2^{-l(x)}$, which is less than half the height of the step corresponding to x by (5.75). The lower end of the interval is in the lower half of the step. Thus, the upper end of the interval lies below the top of the step, and the interval corresponding to any codeword lies entirely within the step corresponding to that symbol in the cumulative distribution function. Therefore, the intervals corresponding to different codewords are disjoint and the code is prefix-free. Note that this procedure does not require the symbols to be ordered in terms of probability. Another procedure that uses the ordered probabilities is described in Problem 5.5.28.

Since we use $l(x) = \left\lceil \log \frac{1}{p(x)} \right\rceil + 1$ bits to represent x, the expected length of this code is

$$L = \sum_x p(x) l(x) = \sum_x p(x) \left(\left\lceil \log \frac{1}{p(x)} \right\rceil + 1 \right) < H(X) + 2. \quad (5.76)$$

Thus, this coding scheme achieves an average codeword length that is within 2 bits of the entropy.

Example 5.9.1 We first consider an example where all the probabilities are dyadic. We construct the code in the following table:

x	$p(x)$	$F(x)$	$\overline{F}(x)$	$\overline{F}(x)$ in Binary	$l(x) = \left\lceil \log \frac{1}{p(x)} \right\rceil + 1$	Codeword
1	0.25	0.25	0.125	0.001	3	001
2	0.5	0.75	0.5	0.10	2	10
3	0.125	0.875	0.8125	0.1101	4	1101
4	0.125	1.0	0.9375	0.1111	4	1111

In this case, the average codeword length is 2.75 bits and the entropy is 1.75 bits. The Huffman code for this case achieves the entropy bound. Looking at the codewords, it is obvious that there is some inefficiency—for example, the last bit of the last two codewords can be omitted. But if we remove the last bit from all the codewords, the code is no longer prefix-free.

Example 5.9.2 We now give another example for construction of the Shannon–Fano–Elias code. In this case, since the distribution is not dyadic, the representation of $F(x)$ in binary may have an infinite number of bits. We denote 0.01010101 ... by $0.\overline{01}$. We construct the code in the following table:

x	$p(x)$	$F(x)$	$\overline{F}(x)$	$\overline{F}(x)$ in Binary	$l(x) = \left\lceil \log \frac{1}{p(x)} \right\rceil + 1$	Codeword
1	0.25	0.25	0.125	0.001	3	001
2	0.25	0.5	0.375	0.011	3	011
3	0.2	0.7	0.6	$0.1\overline{0011}$	4	1001
4	0.15	0.85	0.775	$0.110\overline{0011}$	4	1100
5	0.15	1.0	0.925	$0.111\overline{0110}$	4	1110

The above code is 1.2 bits longer on the average than the Huffman code for this source (Example 5.6.1).

The Shannon–Fano–Elias coding procedure can also be applied to sequences of random variables. The key idea is to use the cumulative distribution function of the sequence, expressed to the appropriate accuracy, as a code for the sequence. Direct application of the method to blocks of length n would require calculation of the probabilities and cumulative distribution function for all sequences of length n, a calculation that would grow exponentially with the block length. But a simple trick ensures that we can calculate both the probability and the cumulative density function sequentially as we see each symbol in the block, ensuring that the calculation grows only linearly with the block length. Direct application of Shannon–Fano–Elias coding would also need arithmetic whose precision grows with the block size, which is not practical when we deal with long blocks. In Chapter 13 we describe arithmetic coding, which is an extension of the Shannon–Fano–Elias method to sequences of random variables that encodes using fixed-precision arithmetic with a complexity that is linear in the length of the sequence. This method is the basis of many practical compression schemes such as those used in the JPEG and FAX compression standards.

5.10 COMPETITIVE OPTIMALITY OF THE SHANNON CODE

We have shown that Huffman coding is optimal in that it has minimum expected length. But what does that say about its performance on any particular sequence? For example, is it always better than any other code for all sequences? Obviously not, since there are codes that assign short

codewords to infrequent source symbols. Such codes will be better than the Huffman code on those source symbols.

To formalize the question of competitive optimality, consider the following two-person zero-sum game: Two people are given a probability distribution and are asked to design an instantaneous code for the distribution. Then a source symbol is drawn from this distribution, and the payoff to player A is 1 or -1, depending on whether the codeword of player A is shorter or longer than the codeword of player B. The payoff is 0 for ties.

Dealing with Huffman code lengths is difficult, since there is no explicit expression for the codeword lengths. Instead, we consider the Shannon code with codeword lengths $l(x) = \left\lceil \log \frac{1}{p(x)} \right\rceil$. In this case, we have the following theorem.

Theorem 5.10.1 *Let $l(x)$ be the codeword lengths associated with the Shannon code, and let $l'(x)$ be the codeword lengths associated with any other uniquely decodable code. Then*

$$\Pr\left(l(X) \geq l'(X) + c\right) \leq \frac{1}{2^{c-1}}. \tag{5.77}$$

For example, the probability that $l'(X)$ is 5 or more bits shorter than $l(X)$ is less than $\frac{1}{16}$.

Proof

$$\Pr\left(l(X) \geq l'(X) + c\right) = \Pr\left(\left\lceil \log \frac{1}{p(X)} \right\rceil \geq l'(X) + c\right) \tag{5.78}$$

$$\leq \Pr\left(\log \frac{1}{p(X)} \geq l'(X) + c - 1\right) \tag{5.79}$$

$$= \Pr\left(p(X) \leq 2^{-l'(X)-c+1}\right) \tag{5.80}$$

$$= \sum_{x:\, p(x) \leq 2^{-l'(x)-c+1}} p(x) \tag{5.81}$$

$$\leq \sum_{x:\, p(x) \leq 2^{-l'(x)-c+1}} 2^{-l'(x)-(c-1)} \tag{5.82}$$

$$\leq \sum_{x} 2^{-l'(x)} 2^{-(c-1)} \tag{5.83}$$

$$\leq 2^{-(c-1)} \tag{5.84}$$

since $\sum 2^{-l'(x)} \leq 1$ by the Kraft inequality. □

Hence, no other code can do much better than the Shannon code most of the time. We now strengthen this result. In a game-theoretic setting, one would like to ensure that $l(x) < l'(x)$ more often than $l(x) > l'(x)$. The fact that $l(x) \leq l'(x) + 1$ with probability $\geq \frac{1}{2}$ does not ensure this. We now show that even under this stricter criterion, Shannon coding is optimal. Recall that the probability mass function $p(x)$ is dyadic if $\log \frac{1}{p(x)}$ is an integer for all x.

Theorem 5.10.2 *For a dyadic probability mass function $p(x)$, let $l(x) = \log \frac{1}{p(x)}$ be the word lengths of the binary Shannon code for the source, and let $l'(x)$ be the lengths of any other uniquely decodable binary code for the source. Then*

$$\Pr(l(X) < l'(X)) \geq \Pr(l(X) > l'(X)), \tag{5.85}$$

with equality if and only if $l'(x) = l(x)$ for all x. Thus, the code length assignment $l(x) = \log \frac{1}{p(x)}$ is uniquely competitively optimal.

Proof: Define the function $\text{sgn}(t)$ as follows:

$$\text{sgn}(t) = \begin{cases} 1 & \text{if } t > 0 \\ 0 & \text{if } t = 0 \\ -1 & \text{if } t < 0 \end{cases} \tag{5.86}$$

Then it is easy to see from Figure 5.6 that

$$\text{sgn}(t) \leq 2^t - 1 \quad \text{for } t = 0, \pm 1, \pm 2, \ldots. \tag{5.87}$$

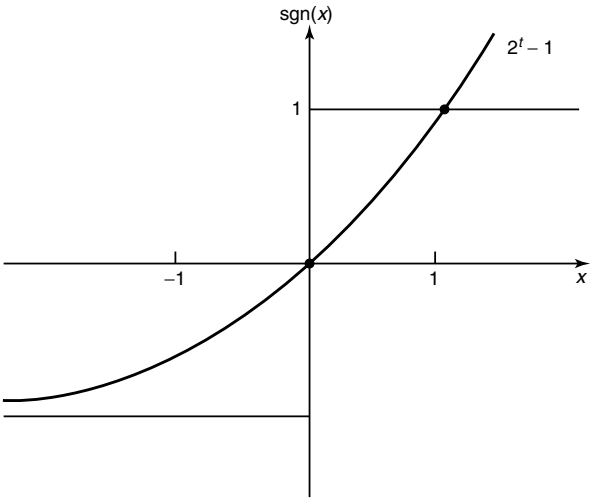


FIGURE 5.6. Sgn function and a bound.

Note that though this inequality is not satisfied for all t, it is satisfied at all integer values of t. We can now write

$$\Pr(l'(X) < l(X)) - \Pr(l'(X) > l(X)) = \sum_{x:l'(x)<l(x)} p(x) - \sum_{x:l'(x)>l(x)} p(x) \tag{5.88}$$

$$= \sum_x p(x)\text{sgn}(l(x) - l'(x)) \tag{5.89}$$

$$= E\ \text{sgn}\left(l(X) - l'(X)\right) \tag{5.90}$$

$$\overset{(a)}{\leq} \sum_x p(x)\left(2^{l(x)-l'(x)} - 1\right) \tag{5.91}$$

$$= \sum_x 2^{-l(x)}\left(2^{l(x)-l'(x)} - 1\right) \tag{5.92}$$

$$= \sum_x 2^{-l'(x)} - \sum_x 2^{-l(x)} \tag{5.93}$$

$$= \sum_x 2^{-l'(x)} - 1 \tag{5.94}$$

$$\overset{(b)}{\leq} 1 - 1 \tag{5.95}$$

$$= 0, \tag{5.96}$$

where (a) follows from the bound on $\text{sgn}(x)$ and (b) follows from the fact that $l'(x)$ satisfies the Kraft inequality.

We have equality in the above chain only if we have equality in (a) and (b). We have equality in the bound for $\text{sgn}(t)$ only if t is 0 or 1 [i.e., $l(x) = l'(x)$ or $l(x) = l'(x) + 1$]. Equality in (b) implies that $l'(x)$ satisfies the Kraft inequality with equality. Combining these two facts implies that $l'(x) = l(x)$ for all x. □

Corollary *For nondyadic probability mass functions,*

$$E\ sgn(l(X) - l'(X) - 1) \leq 0, \tag{5.97}$$

where $l(x) = \left\lceil \log \frac{1}{p(x)} \right\rceil$ *and* $l'(x)$ *is any other code for the source.*

Proof: Along the same lines as the preceding proof. □

Hence we have shown that Shannon coding $l(x) = \left\lceil \log \frac{1}{p(x)} \right\rceil$ is optimal under a variety of criteria; it is robust with respect to the payoff function. In particular, for dyadic p, $E(l - l') \leq 0$, $E \operatorname{sgn}(l - l') \leq 0$, and by use of inequality (5.87), $Ef(l - l') \leq 0$ for any function f satisfying $f(t) \leq 2^t - 1$, $t = 0, \pm 1, \pm 2, \ldots$.

5.11 GENERATION OF DISCRETE DISTRIBUTIONS FROM FAIR COINS

In the early sections of this chapter we considered the problem of representing a random variable by a sequence of bits such that the expected length of the representation was minimized. It can be argued (Problem 5.5.29) that the encoded sequence is essentially incompressible and therefore has an entropy rate close to 1 bit per symbol. Therefore, the bits of the encoded sequence are essentially fair coin flips.

In this section we take a slight detour from our discussion of source coding and consider the dual question. How many fair coin flips does it take to generate a random variable X drawn according to a specified probability mass function $\mathbf{p}$? We first consider a simple example.

Example 5.11.1 Given a sequence of fair coin tosses (fair bits), suppose that we wish to generate a random variable X with distribution

$$X = \begin{cases} a & \text{with probability } \frac{1}{2}, \\ b & \text{with probability } \frac{1}{4}, \\ c & \text{with probability } \frac{1}{4}. \end{cases} \tag{5.98}$$

It is easy to guess the answer. If the first bit is 0, we let $X = a$. If the first two bits are 10, we let $X = b$. If we see 11, we let $X = c$. It is clear that X has the desired distribution.

We calculate the average number of fair bits required for generating the random variable, in this case as $\frac{1}{2}(1) + \frac{1}{4}(2) + \frac{1}{4}(2) = 1.5$ bits. This is also the entropy of the distribution. Is this unusual? No, as the results of this section indicate.

The general problem can now be formulated as follows. We are given a sequence of fair coin tosses $Z_1, Z_2, \ldots$, and we wish to generate a discrete random variable $X \in \mathcal{X} = \{1, 2, \ldots, m\}$ with probability mass function

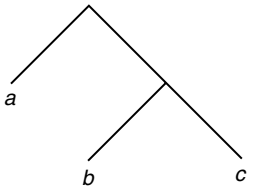


FIGURE 5.7. Tree for generation of the distribution $(\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$.

$\mathbf{p} = (p_1, p_2, \ldots, p_m)$. Let the random variable T denote the number of coin flips used in the algorithm.

We can describe the algorithm mapping strings of bits $Z_1, Z_2, \ldots,$ to possible outcomes X by a binary tree. The leaves of the tree are marked by output symbols X, and the path to the leaves is given by the sequence of bits produced by the fair coin. For example, the tree for the distribution $(\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$ is shown in Figure 5.7.

The tree representing the algorithm must satisfy certain properties:

1. The tree should be complete (i.e., every node is either a leaf or has two descendants in the tree). The tree may be infinite, as we will see in some examples.
2. The probability of a leaf at depth k is 2^{-k}. Many leaves may be labeled with the same output symbol—the total probability of all these leaves should equal the desired probability of the output symbol.
3. The expected number of fair bits ET required to generate X is equal to the expected depth of this tree.

There are many possible algorithms that generate the same output distribution. For example, the mapping $00 \to a, 01 \to b, 10 \to c, 11 \to a$ also yields the distribution $(\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$. However, this algorithm uses two fair bits to generate each sample and is therefore not as efficient as the mapping given earlier, which used only 1.5 bits per sample. This brings up the question: What is the most efficient algorithm to generate a given distribution, and how is this related to the entropy of the distribution?

We expect that we need at least as much randomness in the fair bits as we produce in the output samples. Since entropy is a measure of randomness, and each fair bit has an entropy of 1 bit, we expect that the number of fair bits used will be at least equal to the entropy of the output. This is proved in the following theorem. We will need a simple lemma about trees in the proof of the theorem. Let $\mathcal{Y}$ denote the set of leaves of a complete tree. Consider a distribution on the leaves such that the probability

of a leaf at depth k on the tree is 2^{-k}. Let Y be a random variable with this distribution. Then we have the following lemma.

Lemma 5.11.1 *For any complete tree, consider a probability distribution on the leaves such that the probability of a leaf at depth k is 2^{-k}. Then the expected depth of the tree is equal to the entropy of this distribution.*

Proof: The expected depth of the tree

$$ET = \sum_{y \in \mathcal{Y}} k(y) 2^{-k(y)} \tag{5.99}$$

and the entropy of the distribution of Y is

$$H(Y) = -\sum_{y \in \mathcal{Y}} \frac{1}{2^{k(y)}} \log \frac{1}{2^{k(y)}} \tag{5.100}$$

$$= \sum_{y \in \mathcal{Y}} k(y) 2^{-k(y)}, \tag{5.101}$$

where $k(y)$ denotes the depth of leaf y. Thus,

$$H(Y) = ET. \quad \square \tag{5.102}$$

Theorem 5.11.1 *For any algorithm generating X, the expected number of fair bits used is greater than the entropy $H(X)$, that is,*

$$ET \geq H(X). \tag{5.103}$$

Proof: Any algorithm generating X from fair bits can be represented by a complete binary tree. Label all the leaves of this tree by distinct symbols $y \in \mathcal{Y} = \{1, 2, \ldots\}$. If the tree is infinite, the alphabet $\mathcal{Y}$ is also infinite.

Now consider the random variable Y defined on the leaves of the tree, such that for any leaf y at depth k, the probability that $Y = y$ is 2^{-k}. By Lemma 5.11.1, the expected depth of this tree is equal to the entropy of Y:

$$ET = H(Y). \tag{5.104}$$

Now the random variable X is a function of Y (one or more leaves map onto an output symbol), and hence by the result of Problem 2.4, we have

$$H(X) \leq H(Y). \tag{5.105}$$

Thus, for any algorithm generating the random variable X, we have

$$H(X) \leq ET. \quad \square \tag{5.106}$$

The same argument answers the question of optimality for a dyadic distribution.

Theorem 5.11.2 *Let the random variable X have a dyadic distribution. The optimal algorithm to generate X from fair coin flips requires an expected number of coin tosses precisely equal to the entropy:*

$$ET = H(X). \tag{5.107}$$

Proof: Theorem 5.11.1 shows that we need at least $H(X)$ bits to generate X. For the constructive part, we use the Huffman code tree for X as the tree to generate the random variable. For a dyadic distribution, the Huffman code is the same as the Shannon code and achieves the entropy bound. For any $x \in \mathcal{X}$, the depth of the leaf in the code tree corresponding to x is the length of the corresponding codeword, which is $\log \frac{1}{p(x)}$. Hence, when this code tree is used to generate X, the leaf will have a probability $2^{-\log \frac{1}{p(x)}} = p(x)$. The expected number of coin flips is the expected depth of the tree, which is equal to the entropy (because the distribution is dyadic). Hence, for a dyadic distribution, the optimal generating algorithm achieves

$$ET = H(X). \quad \square \tag{5.108}$$

What if the distribution is not dyadic? In this case we cannot use the same idea, since the code tree for the Huffman code will generate a dyadic distribution on the leaves, not the distribution with which we started. Since all the leaves of the tree have probabilities of the form 2^{-k}, it follows that we should split any probability p_i that is not of this form into atoms of this form. We can then allot these atoms to leaves on the tree. For example, if one of the outcomes x has probability $p(x) = \frac{1}{4}$, we need only one atom (leaf of the tree at level 2), but if $p(x) = \frac{7}{8} = \frac{1}{2} + \frac{1}{4} + \frac{1}{8}$, we need three atoms, one each at levels 1, 2, and 3 of the tree.

To minimize the expected depth of the tree, we should use atoms with as large a probability as possible. So given a probability p_i, we find the largest atom of the form 2^{-k} that is less than p_i, and allot this atom to the tree. Then we calculate the remainder and find that largest atom that will fit in the remainder. Continuing this process, we can split all the

probabilities into dyadic atoms. This process is equivalent to finding the binary expansions of the probabilities. Let the binary expansion of the probability p_i be

$$p_i = \sum_{j \geq 1} p_i^{(j)}, \tag{5.109}$$

where $p_i^{(j)} = 2^{-j}$ or 0. Then the atoms of the expansion are the $\{p_i^{(j)} : i = 1, 2, \ldots, m, \ j \geq 1\}$.

Since $\sum_i p_i = 1$, the sum of the probabilities of these atoms is 1. We will allot an atom of probability 2^{-j} to a leaf at depth j on the tree. The depths of the atoms satisfy the Kraft inequality, and hence by Theorem 5.2.1, we can always construct such a tree with all the atoms at the right depths. We illustrate this procedure with an example.

Example 5.11.2 Let X have the distribution

$$X = \begin{cases} a & \text{with probability } \frac{2}{3}, \\ b & \text{with probability } \frac{1}{3}. \end{cases} \tag{5.110}$$

We find the binary expansions of these probabilities:

$$\frac{2}{3} = 0.10101010\ldots_2 \tag{5.111}$$

$$\frac{1}{3} = 0.01010101\ldots_2 . \tag{5.112}$$

Hence, the atoms for the expansion are

$$\frac{2}{3} \to \left(\frac{1}{2}, \frac{1}{8}, \frac{1}{32}, \ldots\right) \tag{5.113}$$

$$\frac{1}{3} \to \left(\frac{1}{4}, \frac{1}{16}, \frac{1}{64}, \ldots\right). \tag{5.114}$$

These can be allotted to a tree as shown in Figure 5.8.

This procedure yields a tree that generates the random variable X. We have argued that this procedure is optimal (gives a tree of minimum expected depth), but we will not give a formal proof. Instead, we bound the expected depth of the tree generated by this procedure.

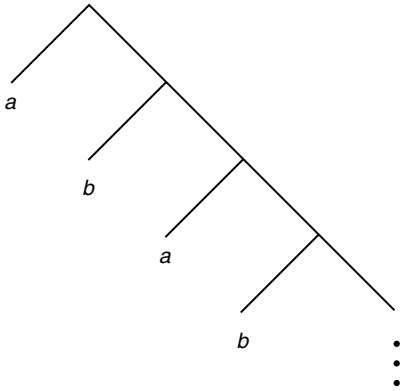


FIGURE 5.8. Tree to generate a $(\frac{2}{3}, \frac{1}{3})$ distribution.

Theorem 5.11.3 *The expected number of fair bits required by the optimal algorithm to generate a random variable X lies between $H(X)$ and $H(X) + 2$:*

$$H(X) \leq ET < H(X) + 2. \tag{5.115}$$

Proof: The lower bound on the expected number of coin tosses is proved in Theorem 5.11.1. For the upper bound, we write down an explicit expression for the expected number of coin tosses required for the procedure described above. We split all the probabilities $(p_1, p_2, \ldots, p_m)$ into dyadic atoms, for example,

$$p_1 \to \left(p_1^{(1)}, p_1^{(2)}, \ldots\right), \tag{5.116}$$

and so on. Using these atoms (which form a dyadic distribution), we construct a tree with leaves corresponding to each of these atoms. The number of coin tosses required to generate each atom is its depth in the tree, and therefore the expected number of coin tosses is the expected depth of the tree, which is equal to the entropy of the dyadic distribution of the atoms. Hence,

$$ET = H(Y), \tag{5.117}$$

where Y has the distribution, $(p_1^{(1)}, p_1^{(2)}, \ldots, p_2^{(1)}, p_2^{(2)}, \ldots, p_m^{(1)}, p_m^{(2)}, \ldots)$. Now since X is a function of Y, we have

$$H(Y) = H(Y, X) = H(X) + H(Y|X), \tag{5.118}$$

and our objective is to show that $H(Y|X) < 2$. We now give an algebraic proof of this result. Expanding the entropy of Y, we have

$$H(Y) = -\sum_{i=1}^{m} \sum_{j \geq 1} p_i^{(j)} \log p_i^{(j)} \tag{5.119}$$

$$= \sum_{i=1}^{m} \sum_{j: p_i^{(j)} > 0} j 2^{-j}, \tag{5.120}$$

since each of the atoms is either 0 or 2^{-k} for some k. Now consider the term in the expansion corresponding to each i, which we shall call T_i:

$$T_i = \sum_{j: p_i^{(j)} > 0} j 2^{-j}. \tag{5.121}$$

We can find an n such that $2^{-(n-1)} > p_i \geq 2^{-n}$, or

$$n - 1 < -\log p_i \leq n. \tag{5.122}$$

Then it follows that $p_i^{(j)} > 0$ only if $j \geq n$, so that we can rewrite (5.121) as

$$T_i = \sum_{j: j \geq n, p_i^{(j)} > 0} j 2^{-j}. \tag{5.123}$$

We use the definition of the atom to write p_i as

$$p_i = \sum_{j: j \geq n, p_i^{(j)} > 0} 2^{-j}. \tag{5.124}$$

To prove the upper bound, we first show that $T_i < -p_i \log p_i + 2p_i$. Consider the difference

$$T_i + p_i \log p_i - 2p_i \overset{(a)}{<} T_i - p_i(n-1) - 2p_i \tag{5.125}$$

$$= T_i - (n - 1 + 2) p_i \tag{5.126}$$

$$= \sum_{j: j \geq n, p_i^{(j)} > 0} j 2^{-j} - (n+1) \sum_{j: j \geq n, p_i^{(j)} > 0} 2^{-j} \tag{5.127}$$

$$= \sum_{j: j \geq n, p_i^{(j)} > 0} (j - n - 1) 2^{-j} \tag{5.128}$$

$$= -2^{-n} + 0 + \sum_{j:j\geq n+2, p_i^{(j)}>0} (j-n-1)2^{-j} \tag{5.129}$$

$$\stackrel{(b)}{=} -2^{-n} + \sum_{k:k\geq 1, p_i^{(k+n+1)}>0} k2^{-(k+n+1)} \tag{5.130}$$

$$\stackrel{(c)}{\leq} -2^{-n} + \sum_{k:k\geq 1} k2^{-(k+n+1)} \tag{5.131}$$

$$= -2^{-n} + 2^{-(n+1)}2 \tag{5.132}$$

$$= 0, \tag{5.133}$$

where (a) follows from (5.122), (b) follows from a change of variables for the summation, and (c) follows from increasing the range of the summation. Hence, we have shown that

$$T_i < -p_i \log p_i + 2p_i. \tag{5.134}$$

Since $ET = \sum_i T_i$, it follows immediately that

$$ET < -\sum_i p_i \log p_i + 2\sum_i p_i = H(X) + 2, \tag{5.135}$$

completing the proof of the theorem.

□

Thus, an average of $H(X) + 2$ coin flips suffice to simulate a random variable X.

SUMMARY

Kraft inequality. Instantaneous codes $\Leftrightarrow \sum D^{-l_i} \leq 1$.

McMillan inequality. Uniquely decodable codes $\Leftrightarrow \sum D^{-l_i} \leq 1$.

Entropy bound on data compression

$$L \stackrel{\triangle}{=} \sum p_i l_i \geq H_D(X). \tag{5.136}$$

Shannon code

$$l_i = \left\lceil \log_D \frac{1}{p_i} \right\rceil \tag{5.137}$$

$$H_D(X) \le L < H_D(X) + 1. \tag{5.138}$$

Huffman code

$$L^* = \min_{\sum D^{-l_i} \le 1} \sum p_i l_i \tag{5.139}$$

$$H_D(X) \le L^* < H_D(X) + 1. \tag{5.140}$$

Wrong code. $X \sim p(x)$, $l(x) = \left\lceil \log \frac{1}{q(x)} \right\rceil$, $L = \sum p(x) l(x)$:

$$H(p) + D(p||q) \le L < H(p) + D(p||q) + 1. \tag{5.141}$$

Stochastic processes

$$\frac{H(X_1, X_2, \ldots, X_n)}{n} \le L_n < \frac{H(X_1, X_2, \ldots, X_n)}{n} + \frac{1}{n}. \tag{5.142}$$

Stationary processes

$$L_n \to H(\mathcal{X}). \tag{5.143}$$

Competitive optimality. Shannon code $l(x) = \left\lceil \log \frac{1}{p(x)} \right\rceil$ versus any other code $l'(x)$:

$$\Pr\left(l(X) \ge l'(X) + c\right) \le \frac{1}{2^{c-1}}. \tag{5.144}$$

PROBLEMS

5.1 *Uniquely decodable and instantaneous codes*. Let $L = \sum_{i=1}^{m} p_i l_i^{100}$ be the expected value of the 100th power of the word lengths associated with an encoding of the random variable X. Let $L_1 = \min L$ over all instantaneous codes; and let $L_2 = \min L$ over all uniquely decodable codes. What inequality relationship exists between L_1 and L_2?

5.2 *How many fingers has a Martian?* Let

$$S = \begin{pmatrix} S_1, \ldots, S_m \\ p_1, \ldots, p_m \end{pmatrix}.$$

The S_i's are encoded into strings from a D-symbol output alphabet in a uniquely decodable manner. If $m = 6$ and the codeword lengths are $(l_1, l_2, \ldots, l_6) = (1, 1, 2, 3, 2, 3)$, find a good lower bound on D. You may wish to explain the title of the problem.

5.3 *Slackness in the Kraft inequality*. An instantaneous code has word lengths $l_1, l_2, \ldots, l_m$, which satisfy the strict inequality

$$\sum_{i=1}^{m} D^{-l_i} < 1.$$

The code alphabet is $\mathcal{D} = \{0, 1, 2, \ldots, D-1\}$. Show that there exist arbitrarily long sequences of code symbols in $\mathcal{D}^*$ which cannot be decoded into sequences of codewords.

5.4 *Huffman coding*. Consider the random variable

$$X = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 \\ 0.49 & 0.26 & 0.12 & 0.04 & 0.04 & 0.03 & 0.02 \end{pmatrix}.$$

(a) Find a binary Huffman code for X.

(b) Find the expected code length for this encoding.

(c) Find a ternary Huffman code for X.

5.5 *More Huffman codes*. Find the binary Huffman code for the source with probabilities $(\frac{1}{3}, \frac{1}{5}, \frac{1}{5}, \frac{2}{15}, \frac{2}{15})$. Argue that this code is also optimal for the source with probabilities $(\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5})$.

5.6 *Bad codes*. Which of these codes cannot be Huffman codes for any probability assignment?

(a) $\{0, 10, 11\}$

(b) $\{00, 01, 10, 110\}$

(c) $\{01, 10\}$

5.7 *Huffman 20 questions*. Consider a set of n objects. Let $X_i = 1$ or 0 accordingly as the ith object is good or defective. Let $X_1, X_2, \ldots, X_n$ be independent with $\Pr\{X_i = 1\} = p_i$; and $p_1 > p_2 > \cdots > p_n > \frac{1}{2}$. We are asked to determine the set of all defective objects. Any yes–no question you can think of is admissible.

(a) Give a good lower bound on the minimum average number of questions required.

(b) If the longest sequence of questions is required by nature's answers to our questions, what (in words) is the last question we should ask? What two sets are we distinguishing with this question? Assume a compact (minimum average length) sequence of questions.

(c) Give an upper bound (within one question) on the minimum average number of questions required.

5.8 *Simple optimum compression of a Markov source.* Consider the three-state Markov process $U_1, U_2, \ldots$ having transition matrix

U_{n-1} \ U_n	S_1	S_2	S_3
S_1	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$
S_2	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{4}$
S_3	0	$\frac{1}{2}$	$\frac{1}{2}$

Thus, the probability that S_1 follows S_3 is equal to zero. Design three codes C_1, C_2, C_3 (one for each state 1, 2 and 3, each code mapping elements of the set of S_i's into sequences of 0's and 1's, such that this Markov process can be sent with maximal compression by the following scheme:

(a) Note the present symbol $X_n = i$.

(b) Select code C_i.

(c) Note the next symbol $X_{n+1} = j$ and send the codeword in C_i corresponding to j.

(d) Repeat for the next symbol. What is the average message length of the next symbol conditioned on the previous state $X_n = i$ using this coding scheme? What is the unconditional average number of bits per source symbol? Relate this to the entropy rate $H(\mathcal{U})$ of the Markov chain.

5.9 *Optimal code lengths that require one bit above entropy.* The source coding theorem shows that the optimal code for a random variable X has an expected length less than $H(X) + 1$. Give an example of a random variable for which the expected length of the optimal code is close to $H(X) + 1$ [i.e., for any $\epsilon > 0$, construct a distribution for which the optimal code has $L > H(X) + 1 - \epsilon$].

5.10 *Ternary codes that achieve the entropy bound*. A random variable X takes on m values and has entropy $H(X)$. An instantaneous ternary code is found for this source, with average length

$$L = \frac{H(X)}{\log_2 3} = H_3(X). \tag{5.145}$$

(a) Show that each symbol of X has a probability of the form 3^{-i} for some i.

(b) Show that m is odd.

5.11 *Suffix condition*. Consider codes that satisfy the suffix condition, which says that no codeword is a suffix of any other codeword. Show that a suffix condition code is uniquely decodable, and show that the minimum average length over all codes satisfying the suffix condition is the same as the average length of the Huffman code for that random variable.

5.12 *Shannon codes and Huffman codes*. Consider a random variable X that takes on four values with probabilities $(\frac{1}{3}, \frac{1}{3}, \frac{1}{4}, \frac{1}{12})$.

(a) Construct a Huffman code for this random variable.

(b) Show that there exist two different sets of optimal lengths for the codewords; namely, show that codeword length assignments $(1, 2, 3, 3)$ and $(2, 2, 2, 2)$ are both optimal.

(c) Conclude that there are optimal codes with codeword lengths for some symbols that exceed the Shannon code length $\left\lceil \log \frac{1}{p(x)} \right\rceil$.

5.13 *Twenty questions*. Player A chooses some object in the universe, and player B attempts to identify the object with a series of yes–no questions. Suppose that player B is clever enough to use the code achieving the minimal expected length with respect to player A's distribution. We observe that player B requires an average of 38.5 questions to determine the object. Find a rough lower bound to the number of objects in the universe.

5.14 *Huffman code*. Find the **(a)** *binary* and **(b)** *ternary* Huffman codes for the random variable X with probabilities

$$p = \left(\frac{1}{21}, \frac{2}{21}, \frac{3}{21}, \frac{4}{21}, \frac{5}{21}, \frac{6}{21}\right).$$

(c) Calculate $L = \sum p_i l_i$ in each case.

5.15 *Huffman codes*

(a) Construct a binary Huffman code for the following distribution on five symbols: $\mathbf{p} = (0.3, 0.3, 0.2, 0.1, 0.1)$. What is the average length of this code?

(b) Construct a probability distribution $\mathbf{p}'$ on five symbols for which the code that you constructed in part (a) has an average length (under $\mathbf{p}'$) equal to its entropy $H(\mathbf{p}')$.

5.16 *Huffman codes*. Consider a random variable X that takes six values $\{A, B, C, D, E, F\}$ with probabilities 0.5, 0.25, 0.1, 0.05, 0.05, and 0.05, respectively.

(a) Construct a binary Huffman code for this random variable. What is its average length?

(b) Construct a quaternary Huffman code for this random variable [i.e., a code over an alphabet of four symbols (call them a, b, c and d)]. What is the average length of this code?

(c) One way to construct a binary code for the random variable is to start with a quaternary code and convert the symbols into binary using the mapping $a \rightarrow 00$, $b \rightarrow 01$, $c \rightarrow 10$, and $d \rightarrow 11$. What is the average length of the binary code for the random variable above constructed by this process?

(d) For any random variable X, let L_H be the average length of the binary Huffman code for the random variable, and let L_{QB} be the average length code constructed by first building a quaternary Huffman code and converting it to binary. Show that

$$L_H \leq L_{QB} < L_H + 2. \tag{5.146}$$

(e) The lower bound in the example is tight. Give an example where the code constructed by converting an optimal quaternary code is also the optimal binary code.

(f) The upper bound (i.e., $L_{QB} < L_H + 2$) is not tight. In fact, a better bound is $L_{QB} \leq L_H + 1$. Prove this bound, and provide an example where this bound is tight.

5.17 *Data compression*. Find an optimal set of binary codeword lengths $l_1, l_2, \ldots$ (minimizing $\sum p_i l_i$) for an instantaneous code for each of the following probability mass functions:

(a) $\mathbf{p} = (\frac{10}{41}, \frac{9}{41}, \frac{8}{41}, \frac{7}{41}, \frac{7}{41})$

(b) $\mathbf{p} = (\frac{9}{10}, (\frac{9}{10})(\frac{1}{10}), (\frac{9}{10})(\frac{1}{10})^2, (\frac{9}{10})(\frac{1}{10})^3, \ldots)$

5.18 *Classes of codes*. Consider the code $\{0, 01\}$.

(a) Is it instantaneous?

(b) Is it uniquely decodable?

(c) Is it nonsingular?

5.19 *The game of Hi-Lo*

(a) A computer generates a number X according to a known probability mass function $p(x), x \in \{1, 2, \ldots, 100\}$. The player asks a question, "Is $X = i$?" and is told "Yes," "You're too high," or "You're too low." He continues for a total of six questions. If he is right (i.e., he receives the answer "Yes") during this sequence, he receives a prize of value $v(X)$. How should the player proceed to maximize his expected winnings?

(b) Part (a) doesn't have much to do with information theory. Consider the following variation: $X \sim p(x)$, prize $= v(x)$, $p(x)$ known, as before. But *arbitrary* yes–no questions are asked sequentially until X is determined. ("Determined" doesn't mean that a "Yes" answer is received.) Questions cost 1 unit each. How should the player proceed? What is the expected payoff?

(c) Continuing part (b), what if $v(x)$ is fixed but $p(x)$ can be chosen by the computer (and then announced to the player)? The computer wishes to minimize the player's expected return. What should $p(x)$ be? What is the expected return to the player?

5.20 *Huffman codes with costs*. Words such as "Run!", "Help!", and "Fire!" are short, not because they are used frequently, but perhaps because time is precious in the situations in which these words are required. Suppose that $X = i$ with probability $p_i, i = 1, 2, \ldots, m$. Let l_i be the number of binary symbols in the codeword associated with $X = i$, and let c_i denote the cost per letter of the codeword when $X = i$. Thus, the average cost C of the description of X is $C = \sum_{i=1}^{m} p_i c_i l_i$.

(a) Minimize C over all $l_1, l_2, \ldots, l_m$ such that $\sum 2^{-l_i} \leq 1$. Ignore any implied integer constraints on l_i. Exhibit the minimizing $l_1^*, l_2^*, \ldots, l_m^*$ and the associated minimum value C^*.

(b) How would you use the Huffman code procedure to minimize C over all uniquely decodable codes? Let C_{Huffman} denote this minimum.

(c) Can you show that

$$C^* \le C_{\text{Huffman}} \le C^* + \sum_{i=1}^{m} p_i c_i ?$$

5.21 *Conditions for unique decodability.* Prove that a code C is uniquely decodable if (and only if) the extension

$$C^k(x_1, x_2, \ldots, x_k) = C(x_1)C(x_2)\cdots C(x_k)$$

is a one-to-one mapping from $\mathcal{X}^k$ to D^* for every $k \ge 1$. (The "only if" part is obvious.)

5.22 *Average length of an optimal code.* Prove that $L(p_1, \ldots, p_m)$, the average codeword length for an optimal D-ary prefix code for probabilities $\{p_1, \ldots, p_m\}$, is a continuous function of $p_1, \ldots, p_m$. This is true even though the optimal code changes discontinuously as the probabilities vary.

5.23 *Unused code sequences.* Let C be a variable-length code that satisfies the Kraft inequality with an equality but does *not* satisfy the prefix condition.

(a) Prove that some finite sequence of code alphabet symbols is not the prefix of any sequence of codewords.

(b) (*Optional*) Prove or disprove: C has infinite decoding delay.

5.24 *Optimal codes for uniform distributions.* Consider a random variable with m equiprobable outcomes. The entropy of this information source is obviously $\log_2 m$ bits.

(a) Describe the optimal instantaneous binary code for this source and compute the average codeword length L_m.

(b) For what values of m does the average codeword length L_m equal the entropy $H = \log_2 m$?

(c) We know that $L < H + 1$ for any probability distribution. The *redundancy* of a variable-length code is defined to be $\rho = L - H$. For what value(s) of m, where $2^k \le m \le 2^{k+1}$, is the redundancy of the code maximized? What is the limiting value of this worst-case redundancy as $m \to \infty$?

5.25 *Optimal codeword lengths.* Although the codeword lengths of an optimal variable-length code are complicated functions of the message probabilities $\{p_1, p_2, \ldots, p_m\}$, it can be said that less probable

symbols are encoded into longer codewords. Suppose that the message probabilities are given in decreasing order, $p_1 > p_2 \geq \cdots \geq p_m$.

(a) Prove that for any binary Huffman code, if the most probable message symbol has probability $p_1 > \frac{2}{5}$, that symbol must be assigned a codeword of length 1.

(b) Prove that for any binary Huffman code, if the most probable message symbol has probability $p_1 < \frac{1}{3}$, that symbol must be assigned a codeword of length ≥ 2.

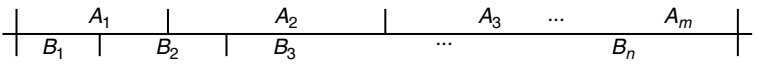
5.26 *Merges*. Companies with values $W_1, W_2, \ldots, W_m$ are merged as follows. The two least valuable companies are merged, thus forming a list of $m-1$ companies. The *value of the merge* is the sum of the values of the two merged companies. This continues until one supercompany remains. Let V equal the sum of the values of the merges. Thus, V represents the total reported dollar volume of the merges. For example, if $\mathbf{W} = (3, 3, 2, 2)$, the merges yield $(3, 3, 2, 2) \rightarrow (4, 3, 3) \rightarrow (6, 4) \rightarrow (10)$ and $V = 4 + 6 + 10 = 20$.

(a) Argue that V is the minimum volume achievable by sequences of pairwise merges terminating in one supercompany. (*Hint:* Compare to Huffman coding.)

(b) Let $W = \sum W_i, \quad \tilde{W}_i = W_i / W$, and show that the minimum merge volume V satisfies

$$W H(\tilde{\mathbf{W}}) \leq V \leq W H(\tilde{\mathbf{W}}) + W. \tag{5.147}$$

5.27 *Sardinas–Patterson test for unique decodability*. A code is not uniquely decodable if and only if there exists a finite sequence of code symbols which can be resolved into sequences of codewords in two different ways. That is, a situation such as



must occur where each A_i and each B_i is a codeword. Note that B_1 must be a prefix of A_1 with some resulting "dangling suffix." Each dangling suffix must in turn be either a prefix of a codeword or have another codeword as its prefix, resulting in another dangling suffix. Finally, the last dangling suffix in the sequence must also be a codeword. Thus, one can set up a test for unique decodability (which is essentially the Sardinas–Patterson test [456]) in

the following way: Construct a set S of all possible dangling suffixes. The code is uniquely decodable if and only if S contains no codeword.

(a) State the precise rules for building the set S.

(b) Suppose that the codeword lengths are l_i, $i = 1, 2, \ldots, m$. Find a good upper bound on the number of elements in the set S.

(c) Determine which of the following codes is uniquely decodable:

(i) {0, 10, 11}
(ii) {0, 01, 11}
(iii) {0, 01, 10}
(iv) {0, 01}
(v) {00, 01, 10, 11}
(vi) {110, 11, 10}
(vii) {110, 11, 100, 00, 10}

(d) For each uniquely decodable code in part (c), construct, if possible, an infinite encoded sequence with a known starting point such that it can be resolved into codewords in two different ways. (This illustrates that unique decodability does not imply finite decodability.) Prove that such a sequence cannot arise in a prefix code.

5.28 *Shannon code*. Consider the following method for generating a code for a random variable X that takes on m values $\{1, 2, \ldots, m\}$ with probabilities $p_1, p_2, \ldots, p_m$. Assume that the probabilities are ordered so that $p_1 \geq p_2 \geq \cdots \geq p_m$. Define

$$F_i = \sum_{k=1}^{i-1} p_k, \tag{5.148}$$

the sum of the probabilities of all symbols less than i. Then the codeword for i is the number $F_i \in [0, 1]$ rounded off to l_i bits, where $l_i = \lceil \log \frac{1}{p_i} \rceil$.

(a) Show that the code constructed by this process is prefix-free and that the average length satisfies

$$H(X) \leq L < H(X) + 1. \tag{5.149}$$

(b) Construct the code for the probability distribution (0.5, 0.25, 0.125, 0.125).

5.29 *Optimal codes for dyadic distributions*. For a Huffman code tree, define the probability of a node as the sum of the probabilities of all the leaves under that node. Let the random variable X be drawn from a dyadic distribution [i.e., $p(x) = 2^{-i}$, for some i, for all $x \in \mathcal{X}$]. Now consider a binary Huffman code for this distribution.

(a) Argue that for any node in the tree, the probability of the left child is equal to the probability of the right child.

(b) Let $X_1, X_2, \ldots, X_n$ be drawn i.i.d. $\sim p(x)$. Using the Huffman code for $p(x)$, we map $X_1, X_2, \ldots, X_n$ to a sequence of bits $Y_1, Y_2, \ldots, Y_{k(X_1, X_2, \ldots, X_n)}$. (The length of this sequence will depend on the outcome $X_1, X_2, \ldots, X_n$.) Use part (a) to argue that the sequence $Y_1, Y_2, \ldots$ forms a sequence of fair coin flips [i.e., that $\Pr\{Y_i = 0\} = \Pr\{Y_i = 1\} = \frac{1}{2}$, independent of $Y_1, Y_2, \ldots, Y_{i-1}$]. Thus, the entropy rate of the coded sequence is 1 bit per symbol.

(c) Give a heuristic argument why the encoded sequence of bits for any code that achieves the entropy bound cannot be compressible and therefore should have an entropy rate of 1 bit per symbol.

5.30 *Relative entropy is cost of miscoding*. Let the random variable X have five possible outcomes $\{1, 2, 3, 4, 5\}$. Consider two distributions $p(x)$ and $q(x)$ on this random variable.

Symbol	$p(x)$	$q(x)$	$C_1(x)$	$C_2(x)$
1	$\frac{1}{2}$	$\frac{1}{2}$	0	0
2	$\frac{1}{4}$	$\frac{1}{8}$	10	100
3	$\frac{1}{8}$	$\frac{1}{8}$	110	101
4	$\frac{1}{16}$	$\frac{1}{8}$	1110	110
5	$\frac{1}{16}$	$\frac{1}{8}$	1111	111

(a) Calculate $H(p)$, $H(q)$, $D(p||q)$, and $D(q||p)$.

(b) The last two columns represent codes for the random variable. Verify that the average length of C_1 under p is equal to the entropy $H(p)$. Thus, C_1 is optimal for p. Verify that C_2 is optimal for q.

(c) Now assume that we use code C_2 when the distribution is p. What is the average length of the codewords. By how much does it exceed the entropy p?

(d) What is the loss if we use code C_1 when the distribution is q?

5.31 *Nonsingular codes*. The discussion in the text focused on instantaneous codes, with extensions to uniquely decodable codes. Both these are required in cases when the code is to be used repeatedly to encode a sequence of outcomes of a random variable. But if we need to encode only one outcome and we know when we have reached the end of a codeword, we do not need unique decodability—the fact that the code is nonsingular would suffice. For example, if a random variable X takes on three values, a, b, and c, we could encode them by 0, 1, and 00. Such a code is nonsingular but not uniquely decodable.

In the following, assume that we have a random variable X which takes on m values with probabilities $p_1, p_2, \ldots, p_m$ and that the probabilities are ordered so that $p_1 \geq p_2 \geq \cdots \geq p_m$.

(a) By viewing the nonsingular binary code as a ternary code with three symbols, 0, 1, and "STOP," show that the expected length of a nonsingular code $L_{1:1}$ for a random variable X satisfies the following inequality:

$$L_{1:1} \geq \frac{H_2(X)}{\log_2 3} - 1, \tag{5.150}$$

where $H_2(X)$ is the entropy of X in bits. Thus, the average length of a nonsingular code is at least a constant fraction of the average length of an instantaneous code.

(b) Let L_{INST} be the expected length of the best instantaneous code and $L^*_{1:1}$ be the expected length of the best nonsingular code for X. Argue that $L^*_{1:1} \leq L^*_{\text{INST}} \leq H(X) + 1$.

(c) Give a simple example where the average length of the nonsingular code is less than the entropy.

(d) The set of codewords available for a nonsingular code is {0, 1, 00, 01, 10, 11, 000, ...}. Since $L_{1:1} = \sum_{i=1}^{m} p_i l_i$, show that this is minimized if we allot the shortest codewords to the most probable symbols. Thus, $l_1 = l_2 = 1, l_3 = l_4 = l_5 = l_6 = 2$, etc. Show that in general $l_i = \left\lceil \log\left(\frac{i}{2} + 1\right) \right\rceil$, and therefore $L^*_{1:1} = \sum_{i=1}^{m} p_i \left\lceil \log\left(\frac{i}{2} + 1\right) \right\rceil$.

(e) Part (d) shows that it is easy to find the optimal nonsingular code for a distribution. However, it is a little more tricky to deal with the average length of this code. We now bound this average length. It follows from part (d) that $L^*_{1:1} \geq$

$\tilde{L} \stackrel{\triangle}{=} \sum_{i=1}^{m} p_i \log\left(\frac{i}{2}+1\right)$. Consider the difference

$$F(\mathbf{p}) = H(X) - \tilde{L} = -\sum_{i=1}^{m} p_i \log p_i - \sum_{i=1}^{m} p_i \log\left(\frac{i}{2}+1\right). \tag{5.151}$$

Prove by the method of Lagrange multipliers that the maximum of $F(\mathbf{p})$ occurs when $p_i = c/(i+2)$, where $c = 1/(H_{m+2} - H_2)$ and H_k is the sum of the harmonic series:

$$H_k \stackrel{\triangle}{=} \sum_{i=1}^{k} \frac{1}{i}. \tag{5.152}$$

(This can also be done using the nonnegativity of relative entropy.)

(f) Complete the arguments for

$$H(X) - L_{1:1}^* \le H(X) - \tilde{L} \tag{5.153}$$

$$\le \log(2(H_{m+2} - H_2)). \tag{5.154}$$

Now it is well known (see, e.g., Knuth [315]) that $H_k \approx \ln k$ (more precisely, $H_k = \ln k + \gamma + \frac{1}{2k} - \frac{1}{12k^2} + \frac{1}{120k^4} - \epsilon$, where $0 < \epsilon < 1/252n^6$, and $\gamma =$ Euler's constant $= 0.577\ldots$). Using either this or a simple approximation that $H_k \le \ln k + 1$, which can be proved by integration of $\frac{1}{x}$, it can be shown that $H(X) - L_{1:1}^* < \log\log m + 2$. Thus, we have

$$H(X) - \log\log|\mathcal{X}| - 2 \le L_{1:1}^* \le H(X) + 1. \tag{5.155}$$

A nonsingular code cannot do much better than an instantaneous code!

5.32 *Bad wine*. One is given six bottles of wine. It is known that precisely one bottle has gone bad (tastes terrible). From inspection of the bottles it is determined that the probability p_i that the ith bottle is bad is given by $(p_1, p_2, \ldots, p_6) = (\frac{8}{23}, \frac{6}{23}, \frac{4}{23}, \frac{2}{23}, \frac{2}{23}, \frac{1}{23})$. Tasting will determine the bad wine. Suppose that you taste the wines one at a time. Choose the order of tasting to minimize the

expected number of tastings required to determine the bad bottle. Remember, if the first five wines pass the test, you don't have to taste the last.

(a) What is the expected number of tastings required?

(b) Which bottle should be tasted first?

Now you get smart. For the first sample, you mix some of the wines in a fresh glass and sample the mixture. You proceed, mixing and tasting, stopping when the bad bottle has been determined.

(a) What is the minimum expected number of tastings required to determine the bad wine?

(b) What mixture should be tasted first?

5.33 *Huffman vs. Shannon*. A random variable X takes on three values with probabilities 0.6, 0.3, and 0.1.

(a) What are the lengths of the binary Huffman codewords for X? What are the lengths of the binary Shannon codewords $\left(l(x) = \left\lceil \log\left(\frac{1}{p(x)}\right)\right\rceil\right)$ for X?

(b) What is the smallest integer D such that the expected Shannon codeword length with a D-ary alphabet equals the expected Huffman codeword length with a D-ary alphabet?

5.34 *Huffman algorithm for tree construction*. Consider the following problem: m binary signals $S_1, S_2, \ldots, S_m$ are available at times $T_1 \leq T_2 \leq \cdots \leq T_m$, and we would like to find their sum $S_1 \oplus S_2 \oplus \cdots \oplus S_m$ using two-input gates, each gate with one time unit delay, so that the final result is available as quickly as possible. A simple greedy algorithm is to combine the earliest two results, forming the partial result at time $\max(T_1, T_2) + 1$. We now have a new problem with $S_1 \oplus S_2, S_3, \ldots, S_m$, available at times $\max(T_1, T_2) + 1, T_3, \ldots, T_m$. We can now sort this list of T's and apply the same merging step again, repeating this until we have the final result.

(a) Argue that the foregoing procedure is optimal, in that it constructs a circuit for which the final result is available as quickly as possible.

(b) Show that this procedure finds the tree that minimizes

$$C(T) = \max_i (T_i + l_i), \tag{5.156}$$

where T_i is the time at which the result allotted to the ith leaf is available and l_i is the length of the path from the ith leaf to the root.

(c) Show that

$$C(T) \geq \log_2 \left(\sum_i 2^{T_i} \right) \tag{5.157}$$

for any tree T.

(d) Show that there exists a tree such that

$$C(T) \leq \log_2 \left(\sum_i 2^{T_i} \right) + 1. \tag{5.158}$$

Thus, $\log_2 \left(\sum_i 2^{T_i} \right)$ is the analog of entropy for this problem.

5.35 *Generating random variables*. One wishes to generate a random variable X

$$X = \begin{cases} 1 & \text{with probability } p \\ 0 & \text{with probability } 1 - p. \end{cases} \tag{5.159}$$

You are given fair coin flips $Z_1, Z_2, \ldots$. Let N be the (random) number of flips needed to generate X. Find a good way to use $Z_1, Z_2, \ldots$ to generate X. Show that $EN \leq 2$.

5.36 *Optimal word lengths*.

(a) Can $l = (1, 2, 2)$ be the word lengths of a binary Huffman code. What about (2,2,3,3)?

(b) What word lengths $l = (l_1, l_2, \ldots)$ can arise from binary Huffman codes?

5.37 *Codes*. Which of the following codes are

(a) Uniquely decodable?

(b) Instantaneous?

$$C_1 = \{00, 01, 0\}$$
$$C_2 = \{00, 01, 100, 101, 11\}$$
$$C_3 = \{0, 10, 110, 1110, \ldots\}$$
$$C_4 = \{0, 00, 000, 0000\}$$

5.38 *Huffman*. Find the Huffman D-ary code for $(p_1, p_2, p_3, p_4, p_5, p_6) = (\frac{6}{25}, \frac{6}{25}, \frac{4}{25}, \frac{4}{25}, \frac{3}{25}, \frac{2}{25})$ and the expected word length

(a) For $D = 2$.

(b) For $D = 4$.

5.39 *Entropy of encoded bits*. Let $C : X \longrightarrow \{0, 1\}^*$ be a nonsingular but nonuniquely decodable code. Let X have entropy $H(X)$.

(a) Compare $H(C(X))$ to $H(X)$.

(b) Compare $H(C(X^n))$ to $H(X^n)$.

5.40 *Code rate*. Let X be a random variable with alphabet $\{1, 2, 3\}$ and distribution

$$X = \begin{cases} 1 & \text{with probability } \frac{1}{2} \\ 2 & \text{with probability } \frac{1}{4} \\ 3 & \text{with probability } \frac{1}{4}. \end{cases}$$

The data compression code for X assigns codewords

$$C(x) = \begin{cases} 0 & \text{if } x = 1 \\ 10 & \text{if } x = 2 \\ 11 & \text{if } x = 3. \end{cases}$$

Let $X_1, X_2, \ldots$ be independent, identically distributed according to this distribution and let $Z_1 Z_2 Z_3 \cdots = C(X_1)C(X_2) \cdots$ be the string of binary symbols resulting from concatenating the corresponding codewords. For example, 122 becomes 01010.

(a) Find the entropy rate $H(\mathcal{X})$ and the entropy rate $H(\mathcal{Z})$ in bits per symbol. Note that Z is not compressible further.

(b) Now let the code be

$$C(x) = \begin{cases} 00 & \text{if } x = 1 \\ 10 & \text{if } x = 2 \\ 01 & \text{if } x = 3 \end{cases}$$

and find the entropy rate $H(\mathcal{Z})$.

(c) Finally, let the code be

$$C(x) = \begin{cases} 00 & \text{if } x = 1 \\ 1 & \text{if } x = 2 \\ 01 & \text{if } x = 3 \end{cases}$$

and find the entropy rate $H(\mathcal{Z})$.

5.41 *Optimal codes*. Let $l_1, l_2, \ldots, l_{10}$ be the binary Huffman codeword lengths for the probabilities $p_1 \geq p_2 \geq \cdots \geq p_{10}$. Suppose that we get a new distribution by splitting the last probability

mass. What can you say about the optimal binary codeword lengths $\tilde{l}_1, \tilde{l}_2, \ldots, \tilde{l}_{11}$ for the probabilities $p_1, p_2, \ldots, p_9, \alpha p_{10}, (1-\alpha)p_{10}$, where $0 \le \alpha \le 1$.

5.42 *Ternary codes*. Which of the following codeword lengths can be the word lengths of a 3-ary Huffman code, and which cannot?

(a) $(1, 2, 2, 2, 2)$

(b) $(2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3)$

5.43 *Piecewise Huffman*. Suppose the codeword that we use to describe a random variable $X \sim p(x)$ always starts with a symbol chosen from the set $\{A, B, C\}$, followed by binary digits $\{0, 1\}$. Thus, we have a ternary code for the first symbol and binary thereafter. Give the optimal uniquely decodable code (minimum expected number of symbols) for the probability distribution

$$p = \left(\frac{16}{69}, \frac{15}{69}, \frac{12}{69}, \frac{10}{69}, \frac{8}{69}, \frac{8}{69}\right). \tag{5.160}$$

5.44 *Huffman*. Find the word lengths of the optimal binary encoding of $p = \left(\frac{1}{100}, \frac{1}{100}, \ldots, \frac{1}{100}\right)$.

5.45 *Random 20 questions*. Let X be uniformly distributed over $\{1, 2, \ldots, m\}$. Assume that $m = 2^n$. We ask random questions: Is $X \in S_1$? Is $X \in S_2$?... until only one integer remains. All 2^m subsets S of $\{1, 2, \ldots, m\}$ are equally likely to be asked.

(a) Without loss of generality, suppose that $X = 1$ is the random object. What is the probability that object 2 yields the same answers for k questions as does object 1?

(b) What is the expected number of objects in $\{2, 3, \ldots, m\}$ that have the same answers to the questions as does the correct object 1?

(c) Suppose that we ask $n + \sqrt{n}$ random questions. What is the expected number of wrong objects agreeing with the answers?

(d) Use Markov's inequality $\Pr\{X \ge t\mu\} \le \frac{1}{t}$, to show that the probability of error (one or more wrong object remaining) goes to zero as $n \longrightarrow \infty$.

HISTORICAL NOTES

The foundations for the material in this chapter can be found in Shannon's original paper [469], in which Shannon stated the source coding

theorem and gave simple examples of codes. He described a simple code construction procedure (described in Problem 5.5.28), which he attributed to Fano. This method is now called the Shannon–Fano code construction procedure.

The Kraft inequality for uniquely decodable codes was first proved by McMillan [385]; the proof given here is due to Karush [306]. The Huffman coding procedure was first exhibited and proved to be optimal by Huffman [283].

In recent years, there has been considerable interest in designing source codes that are matched to particular applications, such as magnetic recording. In these cases, the objective is to design codes so that the output sequences satisfy certain properties. Some of the results for this problem are described by Franaszek [219], Adler et al. [5] and Marcus [370].

The arithmetic coding procedure has its roots in the Shannon–Fano code developed by Elias (unpublished), which was analyzed by Jelinek [297]. The procedure for the construction of a prefix-free code described in the text is due to Gilbert and Moore [249]. The extension of the Shannon–Fano–Elias method to sequences is based on the enumerative methods in Cover [120] and was described with finite-precision arithmetic by Pasco [414] and Rissanen [441]. The competitive optimality of Shannon codes was proved in Cover [125] and extended to Huffman codes by Feder [203]. Section 5.11 on the generation of discrete distributions from fair coin flips follows the work of Knuth and Yao[317].

CHAPTER 6

GAMBLING AND DATA COMPRESSION

At first sight, information theory and gambling seem to be unrelated. But as we shall see, there is strong duality between the growth rate of investment in a horse race and the entropy rate of the horse race. Indeed, the sum of the growth rate and the entropy rate is a constant. In the process of proving this, we shall argue that the financial value of side information is equal to the mutual information between the horse race and the side information. The horse race is a special case of investment in the stock market, studied in Chapter 16.

We also show how to use a pair of identical gamblers to compress a sequence of random variables by an amount equal to the growth rate of wealth on that sequence. Finally, we use these gambling techniques to estimate the entropy rate of English.

6.1 THE HORSE RACE

Assume that m horses run in a race. Let the ith horse win with probability p_i. If horse i wins, the payoff is o_i for 1 (i.e., an investment of 1 dollar on horse i results in o_i dollars if horse i wins and 0 dollars if horse i loses).

There are two ways of describing odds: a-for-1 and b-to-1. The first refers to an exchange that takes place before the race—the gambler puts down 1 dollar before the race and at a-for-1 odds will receive a dollars after the race if his horse wins, and will receive nothing otherwise. The second refers to an exchange after the race—at b-to-1 odds, the gambler will pay 1 dollar after the race if his horse loses and will pick up b dollars after the race if his horse wins. Thus, a bet at b-to-1 odds is equivalent to a bet at a-for-1 odds if $b = a - 1$. For example, fair odds on a coin flip would be 2-for-1 or 1-to-1, otherwise known as *even odds*.

We assume that the gambler distributes all of his wealth across the horses. Let b_i be the fraction of the gambler's wealth invested in horse i, where $b_i \geq 0$ and $\sum b_i = 1$. Then if horse i wins the race, the gambler will receive o_i times the amount of wealth bet on horse i. All the other bets are lost. Thus, at the end of the race, the gambler will have multiplied his wealth by a factor $b_i o_i$ if horse i wins, and this will happen with probability p_i. For notational convenience, we use $b(i)$ and b_i interchangeably throughout this chapter.

The wealth at the end of the race is a random variable, and the gambler wishes to "maximize" the value of this random variable. It is tempting to bet everything on the horse that has the maximum expected return (i.e., the one with the maximum $p_i o_i$). But this is clearly risky, since all the money could be lost.

Some clarity results from considering repeated gambles on this race. Now since the gambler can reinvest his money, his wealth is the product of the gains for each race. Let S_n be the gambler's wealth after n races. Then

$$S_n = \prod_{i=1}^{n} S(X_i), \tag{6.1}$$

where $S(X) = b(X)o(X)$ is the factor by which the gambler's wealth is multiplied when horse X wins.

Definition The *wealth relative* $S(X) = b(X)o(X)$ is the factor by which the gambler's wealth grows if horse X wins the race.

Definition The *doubling rate* of a horse race is

$$W(\mathbf{b}, \mathbf{p}) = E(\log S(X)) = \sum_{k=1}^{m} p_k \log b_k o_k. \tag{6.2}$$

The definition of doubling rate is justified by the following theorem.

Theorem 6.1.1 *Let the race outcomes $X_1, X_2, \ldots$ be i.i.d. $\sim p(x)$. Then the wealth of the gambler using betting strategy* $\mathbf{b}$ *grows exponentially at rate $W(\mathbf{b}, \mathbf{p})$; that is,*

$$S_n \doteq 2^{nW(\mathbf{b},\mathbf{p})}. \tag{6.3}$$

Proof: Functions of independent random variables are also independent, and hence $\log S(X_1), \log S(X_2), \ldots$ are i.i.d. Then, by the weak law of large numbers,

$$\frac{1}{n} \log S_n = \frac{1}{n} \sum_{i=1}^{n} \log S(X_i) \to E(\log S(X)) \quad \text{in probability.} \tag{6.4}$$

Thus,

$$S_n \doteq 2^{nW(\mathbf{b},\mathbf{p})}. \qquad \square \tag{6.5}$$

Now since the gambler's wealth grows as $2^{nW(\mathbf{b},\mathbf{p})}$, we seek to maximize the exponent $W(\mathbf{b}, \mathbf{p})$ over all choices of the portfolio $\mathbf{b}$.

Definition The *optimum doubling rate* $W^*(\mathbf{p})$ is the maximum doubling rate over all choices of the portfolio $\mathbf{b}$:

$$W^*(\mathbf{p}) = \max_{\mathbf{b}} W(\mathbf{b}, \mathbf{p}) = \max_{\mathbf{b}: b_i \geq 0,\ \sum_i b_i = 1} \sum_{i=1}^{m} p_i \log b_i o_i. \tag{6.6}$$

We maximize $W(\mathbf{b}, \mathbf{p})$ as a function of $\mathbf{b}$ subject to the constraint $\sum b_i = 1$. Writing the functional with a Lagrange multiplier and changing the base of the logarithm (which does not affect the maximizing $\mathbf{b}$), we have

$$J(\mathbf{b}) = \sum p_i \ln b_i o_i + \lambda \sum b_i. \tag{6.7}$$

Differentiating this with respect to b_i yields

$$\frac{\partial J}{\partial b_i} = \frac{p_i}{b_i} + \lambda, \quad i = 1, 2, \ldots, m. \tag{6.8}$$

Setting the partial derivative equal to 0 for a maximum, we have

$$b_i = -\frac{p_i}{\lambda}. \tag{6.9}$$

Substituting this in the constraint $\sum b_i = 1$ yields $\lambda = -1$ and $b_i = p_i$. Hence, we can conclude that $\mathbf{b} = \mathbf{p}$ is a stationary point of the function $J(\mathbf{b})$. To prove that this is actually a maximum is tedious if we take

second derivatives. Instead, we use a method that works for many such problems: Guess and verify. We verify that proportional gambling $\mathbf{b} = \mathbf{p}$ is optimal in the following theorem. Proportional gambling is known as *Kelly gambling* [308].

Theorem 6.1.2 (*Proportional gambling is log-optimal*) *The optimum doubling rate is given by*

$$W^*(\mathbf{p}) = \sum p_i \log o_i - H(\mathbf{p}) \tag{6.10}$$

and is achieved by the proportional gambling scheme $\mathbf{b}^* = \mathbf{p}$.

Proof: We rewrite the function $W(\mathbf{b}, \mathbf{p})$ in a form in which the maximum is obvious:

$$W(\mathbf{b}, \mathbf{p}) = \sum p_i \log b_i o_i \tag{6.11}$$

$$= \sum p_i \log \left(\frac{b_i}{p_i} p_i o_i \right) \tag{6.12}$$

$$= \sum p_i \log o_i - H(\mathbf{p}) - D(\mathbf{p}||\mathbf{b}) \tag{6.13}$$

$$\leq \sum p_i \log o_i - H(\mathbf{p}), \tag{6.14}$$

with equality iff $\mathbf{p} = \mathbf{b}$ (i.e., the gambler bets on each horse in proportion to its probability of winning). □

Example 6.1.1 Consider a case with two horses, where horse 1 wins with probability p_1 and horse 2 wins with probability p_2. Assume even odds (2-for-1 on both horses). Then the optimal bet is proportional betting (i.e., $b_1 = p_1$, $b_2 = p_2$). The optimal doubling rate is $W^*(\mathbf{p}) = \sum p_i \log o_i - H(\mathbf{p}) = 1 - H(\mathbf{p})$, and the resulting wealth grows to infinity at this rate:

$$S_n \doteq 2^{n(1-H(\mathbf{p}))}. \tag{6.15}$$

Thus, we have shown that proportional betting is growth rate optimal for a sequence of i.i.d. horse races if the gambler can reinvest his wealth and if there is no alternative of keeping some of the wealth in cash.

We now consider a special case when the odds are fair with respect to some distribution (i.e., there is no track take and $\sum \frac{1}{o_i} = 1$). In this case, we write $r_i = \frac{1}{o_i}$, where r_i can be interpreted as a probability mass function

over the horses. (This is the bookie's estimate of the win probabilities.) With this definition, we can write the doubling rate as

$$W(\mathbf{b}, \mathbf{p}) = \sum p_i \log b_i o_i \tag{6.16}$$

$$= \sum p_i \log \left(\frac{b_i}{p_i} \frac{p_i}{r_i} \right) \tag{6.17}$$

$$= D(\mathbf{p}||\mathbf{r}) - D(\mathbf{p}||\mathbf{b}). \tag{6.18}$$

This equation gives another interpretation for the relative entropy *distance*: The doubling rate is the difference between the distance of the bookie's estimate from the true distribution and the distance of the gambler's estimate from the true distribution. Hence, the gambler can make money only if his estimate (as expressed by $\mathbf{b}$) is better than the bookie's.

An even more special case is when the odds are m-for-1 on each horse. In this case, the odds are fair with respect to the uniform distribution and the optimum doubling rate is

$$W^*(\mathbf{p}) = D\left(\mathbf{p}||\frac{1}{m}\right) = \log m - H(\mathbf{p}). \tag{6.19}$$

In this case we can clearly see the duality between data compression and the doubling rate.

Theorem 6.1.3 (*Conservation theorem*) *For uniform fair odds,*

$$W^*(\mathbf{p}) + H(\mathbf{p}) = \log m. \tag{6.20}$$

Thus, the sum of the doubling rate and the entropy rate is a constant.

Every bit of entropy decrease doubles the gambler's wealth. Low entropy races are the most profitable.

In the analysis above, we assumed that the gambler was fully invested. In general, we should allow the gambler the option of retaining some of his wealth as cash. Let $b(0)$ be the proportion of wealth held out as cash, and $b(1), b(2), \ldots, b(m)$ be the proportions bet on the various horses. Then at the end of a race, the ratio of final wealth to initial wealth (the *wealth relative*) is

$$S(X) = b(0) + b(X)o(X). \tag{6.21}$$

Now the optimum strategy may depend on the odds and will not necessarily have the simple form of proportional gambling. We distinguish three subcases:

1. *Fair odds with respect to some distribution*: $\sum \frac{1}{o_i} = 1$. For fair odds, the option of withholding cash does not change the analysis. This is because we can get the effect of withholding cash by betting $b_i = \frac{1}{o_i}$ on the ith horse, $i = 1, 2, \ldots, m$. Then $S(X) = 1$ irrespective of which horse wins. Thus, whatever money the gambler keeps aside as cash can equally well be distributed over the horses, and the assumption that the gambler must invest all his money does not change the analysis. Proportional betting is optimal.
2. *Superfair odds*: $\sum \frac{1}{o_i} < 1$. In this case, the odds are even better than fair odds, so one would always want to put all one's wealth into the race rather than leave it as cash. In this race, too, the optimum strategy is proportional betting. However, it is possible to choose $\mathbf{b}$ so as to form a *Dutch book* by choosing $b_i = c\frac{1}{o_i}$, where $c = 1/\sum \frac{1}{c_i}$, to get $o_i b_i = c$, irrespective of which horse wins. With this allotment, one has wealth $S(X) = 1/\sum \frac{1}{o_i} > 1$ with probability 1 (i.e., no risk). Needless to say, one seldom finds such odds in real life. Incidentally, a Dutch book, although risk-free, does not optimize the doubling rate.
3. *Subfair odds*: $\sum \frac{1}{o_i} > 1$. This is more representative of real life. The organizers of the race track take a cut of all the bets. In this case it is optimal to bet only some of the money and leave the rest aside as cash. Proportional gambling is no longer log-optimal. A parametric form for the optimal strategy can be found using Kuhn–Tucker conditions (Problem 6.6.2); it has a simple "water-filling" interpretation.

6.2 GAMBLING AND SIDE INFORMATION

Suppose the gambler has some information that is relevant to the outcome of the gamble. For example, the gambler may have some information about the performance of the horses in previous races. What is the value of this side information?

One definition of the financial value of such information is the increase in wealth that results from that information. In the setting described in Section 6.1 the measure of the value of information is the increase in the doubling rate due to that information. We will now derive a connection between mutual information and the increase in the doubling rate.

To formalize the notion, let horse $X \in \{1, 2, \ldots, m\}$ win the race with probability $p(x)$ and pay odds of $o(x)$ for 1. Let (X, Y) have joint probability mass function $p(x, y)$. Let $b(x|y) \geq 0$, $\sum_x b(x|y) = 1$ be an arbitrary conditional betting strategy depending on the side information

Y, where $b(x|y)$ is the proportion of wealth bet on horse x when y is observed. As before, let $b(x) \geq 0$, $\sum b(x) = 1$ denote the unconditional betting scheme.

Let the unconditional and the conditional doubling rates be

$$W^*(X) = \max_{\mathbf{b}(x)} \sum_x p(x) \log b(x) o(x), \tag{6.22}$$

$$W^*(X|Y) = \max_{\mathbf{b}(x|y)} \sum_{x,y} p(x, y) \log b(x|y) o(x) \tag{6.23}$$

and let

$$\Delta W = W^*(X|Y) - W^*(X). \tag{6.24}$$

We observe that for (X_i, Y_i) i.i.d. horse races, wealth grows like $2^{nW^*(X|Y)}$ with side information and like $2^{nW^*(X)}$ without side information.

Theorem 6.2.1 *The increase ΔW in doubling rate due to side information Y for a horse race X is*

$$\Delta W = I(X; Y). \tag{6.25}$$

Proof: With side information, the maximum value of $W^*(X|Y)$ with side information Y is achieved by conditionally proportional gambling [i.e., $b^*(x|y) = p(x|y)$]. Thus,

$$W^*(X|Y) = \max_{\mathbf{b}(x|y)} E\big[\log S\big] = \max_{\mathbf{b}(x|y)} \sum p(x, y) \log o(x) b(x|y) \tag{6.26}$$

$$= \sum p(x, y) \log o(x) p(x|y) \tag{6.27}$$

$$= \sum p(x) \log o(x) - H(X|Y). \tag{6.28}$$

Without side information, the optimal doubling rate is

$$W^*(X) = \sum p(x) \log o(x) - H(X). \tag{6.29}$$

Thus, the increase in doubling rate due to the presence of side information Y is

$$\Delta W = W^*(X|Y) - W^*(X) = H(X) - H(X|Y) = I(X; Y). \quad \square \tag{6.30}$$

Hence, the increase in doubling rate is equal to the mutual information between the side information and the horse race. Not surprisingly, independent side information does not increase the doubling rate.

This relationship can also be extended to the general stock market (Chapter 16). In this case, however, one can only show the inequality $\Delta W \leq I$, with equality if and only if the market is a horse race.

6.3 DEPENDENT HORSE RACES AND ENTROPY RATE

The most common example of side information for a horse race is the past performance of the horses. If the horse races are independent, this information will be useless. If we assume that there is dependence among the races, we can calculate the effective doubling rate if we are allowed to use the results of previous races to determine the strategy for the next race.

Suppose that the sequence $\{X_k\}$ of horse race outcomes forms a stochastic process. Let the strategy for each race depend on the results of previous races. In this case, the optimal doubling rate for uniform fair odds is

$$\begin{aligned} & W^*(X_k|X_{k-1}, X_{k-2}, \ldots, X_1) \\ & \quad = E\left[\max_{\mathbf{b}(\cdot|X_{k-1}, X_{k-2}, \ldots, X_1)} E[\log S(X_k)|X_{k-1}, X_{k-2}, \ldots, X_1]\right] \\ & \quad = \log m - H(X_k|X_{k-1}, X_{k-2}, \ldots, X_1), \end{aligned} \tag{6.31}$$

which is achieved by $b^*(x_k|x_{k-1}, \ldots, x_1) = p(x_k|x_{k-1}, \ldots, x_1)$.

At the end of n races, the gambler's wealth is

$$S_n = \prod_{i=1}^{n} S(X_i), \tag{6.32}$$

and the exponent in the growth rate (assuming m for 1 odds) is

$$\frac{1}{n} E \log S_n = \frac{1}{n} \sum E \log S(X_i) \tag{6.33}$$

$$= \frac{1}{n} \sum (\log m - H(X_i|X_{i-1}, X_{i-2}, \ldots, X_1)) \tag{6.34}$$

$$= \log m - \frac{H(X_1, X_2, \ldots, X_n)}{n}. \tag{6.35}$$

The quantity $\frac{1}{n}H(X_1, X_2, \ldots, X_n)$ is the average entropy per race. For a stationary process with entropy rate $H(\mathcal{X})$, the limit in (6.35) yields

$$\lim_{n\to\infty} \frac{1}{n} E \log S_n + H(\mathcal{X}) = \log m. \tag{6.36}$$

Again, we have the result that the entropy rate plus the doubling rate is a constant.

The expectation in (6.36) can be removed if the process is ergodic. It will be shown in Chapter 16 that for an ergodic sequence of horse races,

$$S_n \doteq 2^{nW} \quad \text{with probability 1,} \tag{6.37}$$

where $W = \log m - H(\mathcal{X})$ and

$$H(\mathcal{X}) = \lim \frac{1}{n} H(X_1, X_2, \ldots, X_n). \tag{6.38}$$

Example 6.3.1 (*Red and black*) In this example, cards replace horses and the outcomes become more predictable as time goes on. Consider the case of betting on the color of the next card in a deck of 26 red and 26 black cards. Bets are placed on whether the next card will be red or black, as we go through the deck. We also assume that the game pays 2-for-1; that is, the gambler gets back twice what he bets on the right color. These are fair odds if red and black are equally probable.

We consider two alternative betting schemes:

1. If we bet sequentially, we can calculate the conditional probability of the next card and bet proportionally. Thus, we should bet $(\frac{1}{2}, \frac{1}{2})$ on (red, black) for the first card, $(\frac{26}{51}, \frac{25}{51})$ for the second card if the first card is black, and so on.
2. Alternatively, we can bet on the entire sequence of 52 cards at once. There are $\binom{52}{26}$ possible sequences of 26 red and 26 black cards, all of them equally likely. Thus, proportional betting implies that we put $1/\binom{52}{26}$ of our money on each of these sequences and let each bet "ride."

We will argue that these procedures are equivalent. For example, half the sequences of 52 cards start with red, and so the proportion of money bet on sequences that start with red in scheme 2 is also one-half, agreeing with the proportion used in the first scheme. In general, we can verify that betting $1/\binom{52}{26}$ of the money on each of the possible outcomes will at each

stage give bets that are proportional to the probability of red and black at that stage. Since we bet $1/\binom{52}{26}$ of the wealth on each possible output sequence, and a bet on a sequence increases wealth by a factor of 2^{52} on the sequence observed and 0 on all the others, the resulting wealth is

$$S_{52}^{*} = \frac{2^{52}}{\binom{52}{26}} = 9.08. \tag{6.39}$$

Rather interestingly, the return does not depend on the actual sequence. This is like the AEP in that the return is the same for all sequences. All sequences are typical in this sense.

6.4 THE ENTROPY OF ENGLISH

An important example of an information source is English text. It is not immediately obvious whether English is a stationary ergodic process. Probably not! Nonetheless, we will be interested in the entropy rate of English. We discuss various stochastic approximations to English. As we increase the complexity of the model, we can generate text that looks like English. The stochastic models can be used to compress English text. The better the stochastic approximation, the better the compression.

For the purposes of discussion, we assume that the alphabet of English consists of 26 letters and the space symbol. We therefore ignore punctuation and the difference between upper- and lowercase letters. We construct models for English using empirical distributions collected from samples of text. The frequency of letters in English is far from uniform. The most common letter, E, has a frequency of about 13%, and the least common letters, Q and Z, occur with a frequency of about 0.1%. The letter E is so common that it is rare to find a sentence of any length that does not contain the letter. [A surprising exception to this is the 267-page novel, *Gadsby*, by Ernest Vincent Wright (Lightyear Press, Boston, 1997; original publication in 1939), in which the author deliberately makes no use of the letter E.]

The frequency of pairs of letters is also far from uniform. For example, the letter Q is always followed by a U. The most frequent pair is TH, which occurs normally with a frequency of about 3.7%. We can use the frequency of the pairs to estimate the probability that a letter follows any other letter. Proceeding this way, we can also estimate higher-order conditional probabilities and build more complex models for the language. However, we soon run out of data. For example, to build a third-order Markov approximation, we must estimate the values of

$p(x_i|x_{i-1}, x_{i-2}, x_{i-3})$. There are $27^4 = 531,441$ entries in this table, and we would need to process millions of letters to make accurate estimates of these probabilities.

The conditional probability estimates can be used to generate random samples of letters drawn according to these distributions (using a random number generator). But there is a simpler method to simulate randomness using a sample of text (a book, say). For example, to construct the second-order model, open the book at random and choose a letter at random on the page. This will be the first letter. For the next letter, again open the book at random and starting at a random point, read until the first letter is encountered again. Then take the letter after that as the second letter. We repeat this process by opening to another page, searching for the second letter, and taking the letter after that as the third letter. Proceeding this way, we can generate text that simulates the second-order statistics of the English text.

Here are some examples of Markov approximations to English from Shannon's original paper [472]:

1. *Zero-order approximation*. (The symbols are independent and equiprobable.)

   XFOML RXKHRJFFJUJ ZLPWCFWKCYJ
   FFJEYVKCQSGXYD QPAAMKBZAACIBZLHJQD

2. *First-order approximation*. (The symbols are independent. The frequency of letters matches English text.)

   OCRO HLI RGWR NMIELWIS EU LL NBNESEBYA TH EEI
   ALHENHTTPA OOBTTVA NAH BRL

3. *Second-order approximation*. (The frequency of pairs of letters matches English text.)

   ON IE ANTSOUTINYS ARE T INCTORE ST BE S DEAMY
   ACHIN D ILONASIVE TUCOOWE AT TEASONARE FUSO
   TIZIN ANDY TOBE SEACE CTISBE

4. *Third-order approximation*. (The frequency of triplets of letters matches English text.)

   IN NO IST LAT WHEY CRATICT FROURE BERS GROCID
   PONDENOME OF DEMONSTURES OF THE REPTAGIN IS
   REGOACTIONA OF CRE

5. *Fourth-order approximation*. (The frequency of quadruplets of letters matches English text. Each letter depends on the previous three letters. This sentence is from Lucky's book, *Silicon Dreams* [366].)

   THE GENERATED JOB PROVIDUAL BETTER TRAND THE DISPLAYED CODE, ABOVERY UPONDULTS WELL THE CODERST IN THESTICAL IT DO HOCK BOTHE MERG. (INSTATES CONS ERATION. NEVER ANY OF PUBLE AND TO THEORY. EVENTIAL CALLEGAND TO ELAST BENERATED IN WITH PIES AS IS WITH THE)

   Instead of continuing with the letter models, we jump to word models.

6. *First-order word model*. (The words are chosen independently but with frequencies as in English.)

   REPRESENTING AND SPEEDILY IS AN GOOD APT OR COME CAN DIFFERENT NATURAL HERE HE THE A IN CAME THE TO OF TO EXPERT GRAY COME TO FURNISHES THE LINE MESSAGE HAD BE THESE.

7. *Second-order word model*. (The word transition probabilities match English text.)

   THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH WRITER THAT THE CHARACTER OF THIS POINT IS THEREFORE ANOTHER METHOD FOR THE LETTERS THAT THE TIME OF WHO EVER TOLD THE PROBLEM FOR AN UNEXPECTED

The approximations get closer and closer to resembling English. For example, long phrases of the last approximation could easily have occurred in a real English sentence. It appears that we could get a very good approximation by using a more complex model. These approximations could be used to estimate the entropy of English. For example, the entropy of the zeroth-order model is $\log 27 = 4.76$ bits per letter. As we increase the complexity of the model, we capture more of the structure of English, and the conditional uncertainty of the next letter is reduced. The first-order model gives an estimate of the entropy of 4.03 bits per letter, while the fourth-order model gives an estimate of 2.8 bits per letter. But even the fourth-order model does not capture all the structure of English. In Section 6.6 we describe alternative methods for estimating the entropy of English.

The distribution of English is useful in decoding encrypted English text. For example, a simple substitution cipher (where each letter is replaced

by some other letter) can be solved by looking for the most frequent letter and guessing that it is the substitute for E, and so on. The redundancy in English can be used to fill in some of the missing letters after the other letters are decrypted: for example,

TH_R_ _S _NLY _N_ W_Y T_ F_LL _N TH_ V_W_LS _N TH_S S_NT_NC_.

Some of the inspiration for Shannon's original work on information theory came out of his work in cryptography during World War II. The mathematical theory of cryptography and its relationship to the entropy of language is developed in Shannon [481].

Stochastic models of language also play a key role in some speech recognition systems. A commonly used model is the trigram (second-order Markov) word model, which estimates the probability of the next word given the preceding two words. The information from the speech signal is combined with the model to produce an estimate of the most likely word that could have produced the observed speech. Random models do surprisingly well in speech recognition, even when they do not explicitly incorporate the complex rules of grammar that govern natural languages such as English.

We can apply the techniques of this section to estimate the entropy rate of other information sources, such as speech and images. A fascinating nontechnical introduction to these issues may be found in the book by Lucky [366].

6.5 DATA COMPRESSION AND GAMBLING

We now show a direct connection between gambling and data compression, by showing that a good gambler is also a good data compressor. Any sequence on which a gambler makes a large amount of money is also a sequence that can be compressed by a large factor. The idea of using the gambler as a data compressor is based on the fact that the gambler's bets can be considered to be his estimate of the probability distribution of the data. A good gambler will make a good estimate of the probability distribution. We can use this estimate of the distribution to do arithmetic coding (Section 13.3). This is the essential idea of the scheme described below.

We assume that the gambler has a mechanically identical twin, who will be used for the data decompression. The identical twin will place the same bets on possible sequences of outcomes as the original gambler (and will therefore make the same amount of money). The cumulative amount

of money that the gambler would have made on all sequences that are lexicographically less than the given sequence will be used as a code for the sequence. The decoder will use the identical twin to gamble on all sequences, and look for the sequence for which the same cumulative amount of money is made. This sequence will be chosen as the decoded sequence.

Let $X_1, X_2, \ldots, X_n$ be a sequence of random variables that we wish to compress. Without loss of generality, we will assume that the random variables are binary. Gambling on this sequence will be defined by a sequence of bets

$$b(x_{k+1} \mid x_1, x_2, \ldots, x_k) \geq 0, \quad \sum_{x_{k+1}} b(x_{k+1} \mid x_1, x_2, \ldots, x_k) = 1, \tag{6.40}$$

where $b(x_{k+1} \mid x_1, x_2, \ldots, x_k)$ is the proportion of money bet at time k on the event that $X_{k+1} = x_{k+1}$ given the observed past $x_1, x_2, \ldots, x_k$. Bets are paid at uniform odds (2-for-1). Thus, the wealth S_n at the end of the sequence is given by

$$S_n = 2^n \prod_{k=1}^{n} b(x_k \mid x_1, \ldots, x_{k-1}) \tag{6.41}$$

$$= 2^n b(x_1, x_2, \ldots, x_n), \tag{6.42}$$

where

$$b(x_1, x_2, \ldots, x_n) = \prod_{k=1}^{n} b(x_k | x_{k-1}, \ldots, x_1). \tag{6.43}$$

So sequential gambling can also be considered as an assignment of probabilities (or bets) $b(x_1, x_2, \ldots, x_n) \geq 0$, $\sum_{x_1,\ldots,x_n} b(x_1, \ldots, x_n) = 1$, on the 2^n possible sequences.

This gambling elicits both an estimate of the true probability of the text sequence ($\hat{p}(x_1, \ldots, x_n) = S_n/2^n$) as well as an estimate of the entropy $\left[\hat{H} = -\frac{1}{n} \log \hat{p}\right]$ of the text from which the sequence was drawn. We now wish to show that high values of wealth S_n lead to high data compression. Specifically, we argue that if the text in question results in wealth S_n, then $\log S_n$ bits can be saved in a naturally associated deterministic data compression scheme. We further assert that if the gambling is log optimal, the data compression achieves the Shannon limit H.

Consider the following data compression algorithm that maps the text $\mathbf{x} = x_1 x_2 \cdots x_n \in \{0, 1\}^n$ into a code sequences $c_1 c_2 \cdots c_k$, $c_i \in \{0, 1\}$. Both the compressor and the decompressor know n. Let the 2^n text sequences be arranged in lexicographical order: for example, $0100101 < 0101101$. The encoder observes the sequence $x^n = (x_1, x_2, \ldots, x_n)$. He then calculates what his wealth $S_n(x'(n))$ would have been on all sequences $x'(n) \leq x(n)$ and calculates $F(x(n)) = \sum_{x'(n) \leq x(n)} 2^{-n} S_n(x'(n))$. Clearly, $F(x(n)) \in [0, 1]$. Let $k = \lceil n - \log S_n(x(n)) \rceil$. Now express $F(x(n))$ as a binary decimal to k-place accuracy: $\lfloor F(x(n)) \rfloor = .c_1 c_2 \cdots c_k$. The sequence $c(k) = (c_1, c_2, \ldots, c_k)$ is transmitted to the decoder.

The decoder twin can calculate the precise value $S(x'(n))$ associated with each of the 2^n sequences $x'(n)$. He thus knows the cumulative sum of $2^{-n} S(x'(n))$ up through any sequence $x(n)$. He tediously calculates this sum until it first exceeds $.c(k)$. The first sequence $x(n)$ such that the cumulative sum falls in the interval $[.c_1 \cdots c_k, .c_1 \ldots c_k + (1/2)^k]$ is defined uniquely, and the size of $S(x(n))/2^n$ guarantees that this sequence will be precisely the encoded $x(n)$.

Thus, the twin uniquely recovers $x(n)$. The number of bits required is $k = \lceil n - \log S(x(n)) \rceil$. The number of bits saved is $n - k = \lfloor \log S(x(n)) \rfloor$. For proportional gambling, $S(x(n)) = 2^n p(x(n))$. Thus, the expected number of bits is $Ek = \sum p(x(n)) \lceil -\log p(x(n)) \rceil \leq H(X_1, \ldots, X_n) + 1$.

We see that if the betting operation is deterministic and is known both to the encoder and the decoder, the number of bits necessary to encode $x_1, \ldots, x_n$ is less than $n - \log S_n + 1$. Moreover, if $p(x)$ is known, and if proportional gambling is used, the description length expected is $E(n - \log S_n) \leq H(X_1, \ldots, X_n) + 1$. Thus, the gambling results correspond precisely to the data compression that would have been achieved by the given human encoder–decoder identical twin pair.

The data compression scheme using a gambler is similar to the idea of arithmetic coding (Section 13.3) using a distribution $b(x_1, x_2, \ldots, x_n)$ rather than the true distribution. The procedure above brings out the duality between gambling and data compression. Both involve estimation of the true distribution. The better the estimate, the greater the growth rate of the gambler's wealth and the better the data compression.

6.6 GAMBLING ESTIMATE OF THE ENTROPY OF ENGLISH

We now estimate the entropy rate for English using a human gambler to estimate probabilities. We assume that English consists of 27 characters

(26 letters and a space symbol). We therefore ignore punctuation and case of letters. Two different approaches have been proposed to estimate the entropy of English.

1. *Shannon guessing game*. In this approach the human subject is given a sample of English text and asked to guess the next letter. An optimal subject will estimate the probabilities of the next letter and guess the most probable letter first, then the second most probable letter next, and so on. The experimenter records the number of guesses required to guess the next letter. The subject proceeds this way through a fairly large sample of text. We can then calculate the empirical frequency distribution of the number of guesses required to guess the next letter. Many of the letters will require only one guess; but a large number of guesses will usually be needed at the beginning of words or sentences.
   Now let us assume that the subject can be modeled as a computer making a deterministic choice of guesses given the past text. Then if we have the same machine and the sequence of guess numbers, we can reconstruct the English text. Just let the machine run, and if the number of guesses at any position is k, choose the kth guess of the machine as the next letter. Hence the amount of information in the sequence of guess numbers is the same as in the English text. The entropy of the guess sequence is the entropy of English text. We can bound the entropy of the guess sequence by assuming that the samples are independent. Hence, the entropy of the guess sequence is bounded above by the entropy of the histogram in the experiment. The experiment was conducted in 1950 by Shannon [482], who obtained a value of 1.3 bits per symbol for the entropy of English.
2. *Gambling estimate*. In this approach we let a human subject gamble on the next letter in a sample of English text. This allows finer gradations of judgment than does guessing. As in the case of a horse race, the optimal bet is proportional to the conditional probability of the next letter. The payoff is 27-for-1 on the correct letter.
   Since sequential betting is equivalent to betting on the entire sequence, we can write the payoff after n letters as

$$S_n = (27)^n b(X_1, X_2, \ldots, X_n). \tag{6.44}$$

   Thus, after n rounds of betting, the expected log wealth satisfies

$$E\frac{1}{n}\log S_n = \log 27 + \frac{1}{n}E\log b(X_1, X_2, \ldots, X_n) \tag{6.45}$$