

Chapter 2

A Measure of Information

2.1 Encoding of Information

In this lecture we will think of a message as a sequence of symbols produced by an information source. The message will appear as a signal, which is a function of one or more independent variables. The signal may be a sound wave, an image, or any of a myriad of physical forms.

We will find in a future lecture that the message may be based on phenomena other than language. A language may be thought of as organized combinations of symbols that express and communicate thoughts and feelings. A message may be an expression in a language, and that is how the term is ordinarily used. However, we want to use the term in a more general form so we can model more than human communication. If we think of a message from a machine or from nature, it is difficult to associate it with thoughts or feelings. We can generalize by noting that we can usually model things, at least conceptually, in terms of their state. A state space model is a quite general structure. We will therefore say that a message is an expression of the state of a system¹.

Suppose that we have made some observations and have deduced the state of a system. We may want to record the state so that it can be reset to that value at another time. This is essential, for example, in simulation. To record the state we need to construct a signal that will drive a recording

¹More precisely, it is the mathematical model of a system that has states. A good model will behave like the natural system. The behavior of the model is determined by its state. The current state is deduced from observations.

apparatus. A little thought will convince you that recording the state of a system or recording a message are equivalent tasks. Moreover, recording and communication are equivalent from our conceptual viewpoint.

A recording is done through a signal that is suited to the recording medium. That signal may be produced by a sequence of symbols from an alphabet that we may choose— it is not determined by the system under observation. We simply require that each potential state (or each potential message) be uniquely represented. If there are N different states or messages, then there must be at least N unique arrangements of symbols available to us. In the last lecture we saw that any alphabet with two or more symbols can be used. Moreover, it is not the appearance of the symbols, but their number, that is important in determining their enumeration properties. We will find that at times one can gain some efficiency by properly choosing the size of the recording alphabet, but that is usually a small gain.

The important determiner of the efficiency of a recording code is the manner in which sequences of symbols are associated with the different states. It is the structure of the code, and not the identity of the symbols, that is important. A good code can greatly reduce the amount of data needed to record the state of the model. This is, in fact, the goal of image compression algorithms. A compression system contains an algorithm, which is a model of images, and a method to record the data needed to reproduce the image. The best compression system is the one which reproduces the image with acceptable fidelity with the minimum data.

The amount of information needed to record the state of a system depends upon the probabilities of the various states. It turns out that if we know the state probabilities we can calculate the minimum amount of data that is required on the average. This minimum is given by the entropy of the system, which is closely related to the concept of entropy in physical systems.

A code is an alphabet of symbols and a set of rules that associate events with patterns of symbols from the alphabet. An event is anything that one wishes to express in terms of the code, such as the state of a system. The development of codes depends only upon the abstract notion of events and their associated probabilities. The association with state models is accomplished by interpreting the meaning of event as a particular state.

The ingredients of a code are

1. an alphabet A of symbols. We don't care about the specific representation of objects in A , only that they be distinguishable and there be a

...nite number of them. We will typically represent the alphabet in the form $A_r = \{a_1, a_2, \dots, a_r\}$. We only care about the size r . The smallest useful alphabet is A_2 , whose symbols are customarily represented as $A_2 = \{0, 1\}$.

2. a set of events $E = \{e_1, e_2, \dots\}$. A set of ...nite size may be indicated as E_n and one of infinite size by E_∞ . The specific physical characteristics of the events is of no interest as far as the code is concerned.
3. a rule R that maps events onto codewords, which are combinations of symbols from A .
4. a set of probabilities over E . That is, one can compute the probability of any set B of events from E . If $B \subseteq E$ then one has a probability measure $p(B)$ which satisfies all the rules for probability measures.

A code C can be represented by the above items, and denoted by $C(A, E, R, p)$.

We require that R be such that each event is represented by a unique codeword. Let w_i be the codeword for an event e_i , and let n_i be the number of symbols (or length) of w_i . The set W of codewords is represented by the rule $R: E \rightarrow W$. For unique decoding we require a rule $R^{-1}: W \rightarrow E$.

Let $n(e)$ be the length, or number of symbols, of a word $w(e)$ that represents an event e . The average number of symbols used to represent an event, and the average number of symbols per codeword, is

$$\bar{n} = \sum_{e \in \mathcal{E}} n(e)p(e) \quad (2.1)$$

A good code is one which has a small value for \bar{n} . Here we seek to know how to find the minimum value of \bar{n} for a given event space, described by (E, p) over all possible code rules for a given alphabet A_r . We would also like to have guidance on how to find R for a given (E, p, A_r) such that the code is efficient and both R and R^{-1} are practical to implement. Information theory has made considerable progress on all these tasks.

Example 1 Random Number Generator

A random number generator can produce positive integers from the set $\{e_1 = 1, e_2 = 2, \dots, e_n = n\}$. This machine has a ...nite event space. The generator output is to be represented by words whose letters are from a binary

code alphabet, $A = \{A, B\}$. We are interested in a code that can uniquely represent a sequence of numbers produced by the source.

If $n = 2$, an obvious code rule is to assign $e_1 = 1 \rightarrow A$, $e_2 = 2 \rightarrow B$. There will then be a one-to-one relationship between any sequence from the source and a corresponding sequence of letters from the alphabet.

If $n = 4$ then it is necessary to construct four codewords from an alphabet of two letters. We could use $e_1 \rightarrow AA$, $e_2 \rightarrow AB$, $e_3 \rightarrow BA$, $e_4 \rightarrow BB$.

By extension, we can construct a code whose words are of length $\log_2 n$ when n is any power of 2. A n that is not a power of 2 can be accommodated by using codewords whose length is the next integer above $\log_2 n$.

Codes that are constructed in this manner are uniquely decodable. However, they may not be as efficient as possible. The efficiency cannot be determined without knowing the statistics of the source events.

A definition of unique decodability is given below.

2.1.1 Uniquely Decodable Codes

Suppose that one wants to record a sequence of events, which would then produce a sequence of codewords. We would like to decode any sequence of symbols produced by concatenating the codewords produced by the sequence of events. We want to be able to do this without introducing another symbol to serve as a separator, or comma, between codewords in the sequence. Codes with this property are called uniquely decodable.

2.1.2 Instantaneous Codes

We would like to be able to decode each of the codewords in the sequence independently. This imposes a stronger requirement than that of being uniquely decodable. We want to be able to recognize each codeword as soon as it is seen, reading left to right, independent of surrounding codewords. This stronger requirement could impose conditions that make the code longer.

A necessary and sufficient condition for a code to be instantaneous is that no complete codeword be a prefix of some other codeword.

Example 2 A code that is not instantaneous.

The code with words $a = 1, b = 10, c = 100, d = 1000$ is uniquely decodable because each symbol is determined by the number of zeros it contains.

e_i	w_i	n_i
e_1	1	1
e_2	01	2
e_3	001	3
e_4	000	3

Table 2.1: A non-instantaneous code for an alphabet of four symbols.

The sequence $bdac$ is encoded as 1010001100. One cannot decode a sequence of digits until one is sure all the zeros have been received. Hence, the sequence 10 may be the symbol b or the beginning of c or d . It cannot be decoded until more digits are seen. Hence, this code is not instantaneous.

To construct an instantaneous code we can use the simple expedient of not using any previously defined codeword as a prefix when defining a new codeword.

Example 3 A non-instantaneous code

The code $\{a = 1, b = 01, c = 001, d = 000\}$ is instantaneous. The decoding rule is to recognize a symbol as soon as its codeword is complete. The sequence $bdac$ encodes as 010001001.

Clearly, one can have an instantaneous code that is uniquely decodable. The question is whether there is any advantage in not imposing the constraint of unique decodability. It turns out that there is no advantage, as will be shown by the analysis developed below. We first state and prove Kraft's inequality and then use it in Millan's inequality. This establishes the above assertion.

2.1.3 Kraft's Inequality

The Kraft inequality provides the necessary and sufficient conditions for a code to be instantaneous. Suppose that a system has q recognizable events in a set $E_q = \{e_1, e_2, \dots, e_q\}$ which are to be recorded or communicated using an alphabet $A_r = \{a_1, a_2, \dots, a_r\}$. The events can be any system observables or combination of observables that are of interest. Here we are interested in the number q . A code C is now defined which maps each event onto a codeword

$w_i = R(e_i)$ where w_i is constructed of symbols from A_r and is of length n_i . An example of a code with symbols from A_2 and codeword lengths $\{1, 2, 3, 3\}$ is shown in Table 2.1.

Theorem 4 A sufficient condition for the existence of an instantaneous code with word lengths $\{n_1, n_2, \dots, n_q\}$ is that

$$\sum_{i=1}^q r^{-n_i} \leq 1 \quad (2.2)$$

For a binary code, which uses the alphabet A_2 , the Kraft inequality is

$$\sum_{i=1}^q 2^{-n_i} \leq 1 \quad (2.3)$$

Example 5 The binary codeword lengths in Example 3 are $\{1, 2, 3, 3\}$. The Kraft inequality is

$$2^{-1} + 2^{-2} + 2^{-3} + 2^{-3} = 1$$

Hence, an instantaneous code exists, as we have seen by demonstration above.

2.1.4 Proof of the Kraft Inequality

To prove the sufficiency, we will provide a method to actually construct a code with word lengths $\{n_1, n_2, \dots, n_q\}$ when these lengths satisfy (2.2). Let c_i be the number of codewords of length i . In the above example $c_1 = 1$, $c_2 = 1$, $c_3 = 2$. If the maximum codeword length is n , then the counts must satisfy

$$\sum_{i=1}^n c_i = q \quad (2.4)$$

since there must be a codeword for each source event

Using the above definitions, we can write the Kraft inequality as

$$\sum_{i=1}^n c_i r^{-i} \leq 1 \quad (2.5)$$

Upon multiplying by r^n we have

$$\sum_{i=1}^n c_i r^{n-i} \leq r^n \quad (2.6)$$

This inequality may be written out and rearranged to obtain

$$c_n \cdot r^n \leq c_1 r^{n-1} + c_2 r^{n-2} + \dots + c_{n-1} r$$

The term on the right of the inequality must be nonnegative. If we divide it by r and rearrange, we obtain

$$c_{n-1} \cdot r^{n-1} \leq c_1 r^{n-2} + c_2 r^{n-3} + \dots + c_{n-2} r$$

Once again, the term on the right must be nonnegative. We can continue in this fashion to obtain the following sequence of inequalities:

$$c_n \cdot r^n \leq c_1 r^{n-1} + c_2 r^{n-2} + \dots + c_{n-1} r \quad (2.7)$$

$$c_{n-1} \cdot r^{n-1} \leq c_1 r^{n-2} + c_2 r^{n-3} + \dots + c_{n-2} r \quad (2.8)$$

$$\vdots \quad (2.9)$$

$$c_3 \cdot r^3 \leq c_1 r^2 + c_2 r \quad (2.10)$$

$$c_2 \cdot r^2 \leq c_1 r \quad (2.11)$$

$$c_1 \cdot r \leq r \quad (2.12)$$

The above inequalities now show that we can construct the required code words. We are required to form c_1 words of length 1, leaving $r - c_1$ symbols that can serve as the beginning of longer codewords. By adding a new symbol to the end of each of these prefixes we can construct as many as $r(r - c_1) = r^2 - c_1 r$ words of length 2. We are assured by (2.11) that there will be enough words available. From the unused two-symbol prefixes, there will remain $r^2 - c_1 r - c_2$ which can be used to construct $r^3 - c_1 r^2 - c_2 r$ three-symbol words. Comparing with (2.10) we see that this is a sufficient number. Continuing in this manner, we will find that it is possible to construct the necessary number of words of each length. We have thus demonstrated that (2.2) or (2.5) are sufficient to guarantee the existence of an instantaneous code.

2.1.5 Millar's Inequality

Although the above proof shows that satisfaction of the Kraft inequality guarantees the existence of an instantaneous (and therefore a uniquely decodable) code, it does not show that the condition is necessary. Millar

there be uniquely decodable codes that are more efficient? The analysis below shows that the requirement is also necessary.

Consider raising the summation in the Kraft inequality to a power. Then

$$\sum_{i=1}^m x^{-n_i} = x^{-n_1} + x^{-n_2} + \dots + x^{-n_q} \quad (2.13)$$

When this is multiplied out and the terms with equal exponents are gathered together, the expression will be of the form

$$\sum_{i=1}^m x^{-n_i} = \sum_{k=m}^n N_k x^{-k} \quad (2.14)$$

where n is the length of the longest codeword and N_k is the number of terms of the form x^{-k} . N_k is also the number of distinct strings of codewords that can be constructed by stringing together m codewords such that the string length is exactly k symbols. If the code is to be uniquely decodable, then this number must be less than the number of possible strings of length k , which is x^k . Therefore, if we substitute x^k for N_k in place of N_k , the above equation becomes an inequality:

$$\sum_{i=1}^m x^{-n_i} \leq \sum_{k=m}^n x^k x^{-k} \\ \leq mn + m + 1 \\ \leq mn$$

Let x stand for the sum on the left. For any $x > 1$, $x^m > mn$ for any n if m is sufficiently large. This would violate the above inequality. Hence, a uniquely decodable code requires $x \leq 1$, which is identical to the Kraft inequality (2.2).

Motzkin's inequality shows that there is no advantage in not using an instantaneous code. That is, for any uniquely decodable code there is an equivalent instantaneous code.

2.1.6 Efficient Codes

We would like to construct instantaneous codes which use the minimum average number of symbols per event. This will enable us to describe an

event in the most efficient manner possible, a fact that is important for storage and transmission of information. The average number of alphabet symbols per codeword is given by (2.28)

$$\bar{n} = \sum_{i=1}^q n_i p_i$$

where q is the number of distinct events in E . We want to choose the n_i to satisfy the Kraft inequality and minimize \bar{n} . This is not a trivial matter, but it has been solved by the Huffman coding procedure for the case of statistically independent events. We will examine the Huffman procedure later. For now we will consider the problem of finding a lower bound on \bar{n} .

In our analysis we need to make use of the following theorem. We will find this theorem useful on other occasions.

Theorem 6 Let x_1, x_2, \dots, x_q and y_1, y_2, \dots, y_q be any two sets of numbers with $x_i > 0$ and $y_i > 0$ for $1 \leq i \leq q$ with

$$\sum_{i=1}^q x_i = 1 \quad \text{and} \quad \sum_{i=1}^q y_i = 1$$

Then

$$\sum_{i=1}^q x_i \log_a \frac{1}{x_i} \geq \sum_{i=1}^q x_i \log_a \frac{1}{y_i} \quad (2.15)$$

To prove this we note first that $\ln x \leq x - 1$ with equality only at $x = 1$. This can be shown with a graph of the two functions. Then

$$\begin{aligned} \sum_{i=1}^q x_i \ln \frac{y_i}{x_i} &\leq \sum_{i=1}^q x_i \left(\frac{y_i}{x_i} - 1 \right) \\ &= \sum_{i=1}^q y_i - \sum_{i=1}^q x_i \\ &= 0 \end{aligned}$$

with equality if and only if $x_i = y_i$ for all i . By expanding the logarithm and dividing through by $\ln a$ the theorem is established for any logarithmic base a .

Now by letting $x_i = p(e_i)$ and $y_i = 1/q$, $i = 1, 2, \dots, q$ we have (with $a = q$)

$$\sum_{i=1}^q p_i \log_q \frac{1}{p_i} \cdot \sum_{i=1}^q p_i \log_q q$$

or

$$\sum_{i=1}^q p_i \log_q \frac{1}{p_i} \cdot 1 \quad (2.16)$$

with equality if and only if $p_i = 1/q$ for all i . It is common to write the above expression in terms of \log_2 , which yields

$$\sum_{i=1}^q p_i \log_2 \frac{1}{p_i} \cdot \log_2 q \quad (2.17)$$

The above sum is known as the entropy of a source in which the events are statistically independent

$$H(E) = \sum_{i=1}^q p_i \log_2 \frac{1}{p_i} \text{ bits/event} \quad (2.18)$$

The entropy, which will be discussed in considerable detail in the next lecture, measures the average amount of uncertainty that an observer has about the next source event. The uncertainty is maximum when all states are equally likely. That is, from (2.18)

$$H(E) = \log_2(q) \quad (2.19)$$

where q is the size of the event space, with equality if and only if all events are equally likely.

For now, the entropy is just an interesting quantity that enables us to put a lower bound on the average codeword length. Let y_1, y_2, \dots, y_q be any numerical quantities that satisfy $y_i > 0$ for all i such that $\sum_{i=1}^q y_i = 1$. Then by Theorem 6 we know that

$$\sum_{i=1}^q p_i \log \frac{1}{p_i} \geq \sum_{i=1}^q p_i \log \frac{1}{y_i} \quad (2.20)$$

with equality if and only if $y_i = p_i$ for all i . Now let us choose the y_i to be

$$y_i = \frac{r^{-n_i}}{\sum_{j=1}^{\infty} r^{-n_j}} \tag{2.21}$$

Then

$$\begin{aligned} H(\mathbf{E}) &= \sum_{i=1}^{\infty} p_i (\log_2 r^{-n_i}) + \sum_{j=1}^{\infty} p_j \log_2 r^{-n_j} \\ &= \log_2 r \sum_{i=1}^{\infty} p_i n_i + \log_2 r \sum_{j=1}^{\infty} p_j n_j \\ &= \bar{n} \log_2 r + \log_2 r \sum_{j=1}^{\infty} p_j n_j \end{aligned} \tag{2.22}$$

The sum in the last term must satisfy the Kraft inequality if the code is to be uniquely decodable. The logarithm of this term is therefore non-positive. Therefore, we can write the above inequality as

$$\bar{n} \log_2 r \geq H(\mathbf{E})$$

or

$$\bar{n} \geq \frac{H(\mathbf{E})}{\log_2 r} \tag{2.23}$$

The efficiency of a code is defined as

$$\eta = \frac{H(\mathbf{E})}{\bar{n} \log_2 r} \tag{2.24}$$

A code will have an efficiency of one when the word lengths from the alphabet A_r are chosen to match the entropy. This issue is explored further in the homework.

A code that is close to the lower bound for the minimum average length can be constructed by a simple method. Choose the codeword lengths to satisfy

$$\log_r \frac{1}{p_i} \cdot n_i < \log_r \frac{1}{p_i} + 1 \tag{2.25}$$

If $p_i = r^{-k_i}$ where k_i is an integer, then choose $n_i = k_i$, otherwise choose n_i to be the...rst integer greater than $\log_r p_i$. The average codeword length can be found by multiplying through the above equation and summing over all the source events.

$$\sum_{i=1}^{\infty} p_i \log_r \frac{1}{p_i} \cdot \sum_{i=1}^{\infty} n_i p_i < \sum_{i=1}^{\infty} p_i \log_r \frac{1}{p_i} + \sum_{i=1}^{\infty} p_i \quad (2.26)$$

$$\frac{H(E)}{\log_2 r} \cdot \bar{n} < \frac{H(E)}{\log_2 r} + 1 \quad (2.27)$$

Example 7 An image coding problem. A 1024×1024 image has eight grey levels. A histogram analysis has found that the levels have the probabilities listed in the table below. Also shown are two instantaneous binary codes that could be used to represent the image. The word lengths of Code B satisfy (2.25) while those of Code A were constructed by the Huffman coding procedure to be discussed in Lecture 3. It is readily verified that both codes satisfy the Kraft inequality and are instantaneous.

p_i	$\log_2 p_i$	Code A	Code B	n_{A_i}	n_{B_i}
0.25	2.	00	00	2	2
0.2	2.32	10	100	2	3
0.17	2.56	010	010	3	3
0.15	2.74	110	110	3	3
0.1	3.32	111	1110	3	4
0.08	3.64	0111	0111	4	4
0.03	5.06	01100	011000	5	6
0.02	5.64	01101	011010	5	6

The average number of digits per codeword are

$$n_A = \sum_{i=1}^{\infty} p_i n_{A_i} = 2.73$$

$$n_B = \sum_{i=1}^{\infty} p_i n_{B_i} = 3.08$$

Clearly, Code B is no prize since it requires more digits per pixel than would be required by using a straight three digit binary code. However, Code A uses an average of less than three digits per pixel. The entropy of the image, assuming the pixels are statistically independent, is $H = 2.7$ bits per pixel. Hence, Code A is actually quite efficient.

We will find that the simple Shannon coding technique of (2.25) can actually be quite efficient when groupings of pixels rather than single pixels are used. It is the theoretical basis of computations of asymptotic efficiency of coding methods.

2.2 A Measure of Information

In this lecture we will examine the relationships between events and observations. Suppose that there is a space X of events and a space Y of observations. How much information is provided by a particular observation $y \in Y$ about each possible event $x \in X$? This question can be applied to any situation in which there is a need to relate observations to underlying causes. Its roots were in the field of communication, where the space X is the set of symbols or messages and the space Y is the space of signals, but it has far wider application. In general, X and Y are collections of random events connected by some process, as shown in Figure 2.1.

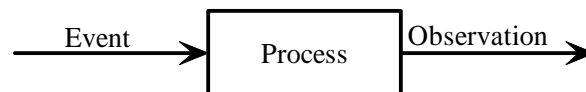


Figure 2.1: Events and observations may be related by any kind of process. The effect of the process is modeled by the conditional probability function $p(y|x)$.

It is presumed that the joint probability distribution $p(x, y)$ is available for any point (x, y) in the space $X \times Y$. All of our results will ultimately be related to this distribution. The theory will apply if and only if this function can be constructed when modeling a real problem.

The rules of probability can be used to relate the joint probability to the

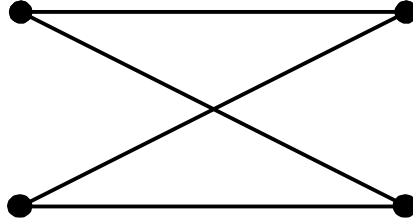


Figure 2.2: Input/output relationships for a binary symmetric channel with error probability 0.1.

marginal and conditional probabilities.

$$\begin{aligned}
 p(x) &= \sum_{y \in \mathcal{Y}} p(x, y) & p(y|x) &= \frac{p(x, y)}{p(x)} \\
 p(y) &= \sum_{x \in \mathcal{X}} p(x, y) & p(x|y) &= \frac{p(x, y)}{p(y)}
 \end{aligned}$$

It is common that the conditional relationship $p(y|x)$ is available for a process. This is the normal situation when modeling an observation system, communication channel, recording system, and the like. If one then models the distribution of input events by choosing $p(x)$ it is possible to calculate the joint probability distribution and all of the other probabilities above.

We are interested in the amount of information about the input that is provided by the observed output. Before the observation, a particular input had probability $p(x)$ and after the observation it has probability $p(x|y)$. Any observation of the output changes the probability distribution over the input. Some possible inputs will be made more likely and some less likely. In the extreme, one event may be made certain and the others made impossible. We would like to have a reasonable way to describe the amount of information that is provided about the input by any particular observation.

An example of the common digital communication channel called the binary symmetric channel (BSC) is shown in Figure 2.2. This model is used for many digital communication and recording systems. Both the input and output space contain two events. The probability of error is equal to the crossover probability. For this example, it is assumed that the input events are equally likely, which, because of the channel symmetry, causes the output

symbols to also be equally likely.

An observer, located at the output side, assumes a uniform probability distribution $p(x_1) = p(x_2) = 0.5$ before any observation is made. After an observation is made, say y_1 , the probabilities change to $p(x_1|y_1) = 0.9$ and $p(x_2|y_1) = 0.1$. The "evidence," although not completely certain, points to x_1 as the cause of the observation y_1 . How much "information" has the observer received about the input?

The measure of information is based on the following definition:

Definition 8 The amount of information provided by the occurrence of the event $y \in Y$ about the occurrence of the event $x \in X$ is defined as

$$I(x; y) = \log \frac{p(x|y)}{p(x)} \quad (2.28)$$

The base of the logarithm determines the units of information. When the base is 2, the unit is the bit². This definition provides a natural, intuitive interpretation of "information" that will be developed as we investigate its properties.

The information measure has the property that if the observation y increases the probability of x then it is positive. That is, if $p(x|y) > p(x)$ then $I(x; y) > 0$. If we expand (2.28) we have

$$I(x; y) = -\log p(x) + \log p(x|y) \quad (2.29)$$

This expression can be given an intuitive meaning by defining the uncertainty of an event

Definition 9 The uncertainty of an event x is $-\log p(x)$.

We see that the uncertainty is zero if $p(x) = 1$ and increases as the probability decreases.

For the BSC of Figure 2.2 the uncertainty about both x_1 and x_2 is $-\log_2 0.5 = 1$ bit before the observation. This uncertainty can be removed by telling the observer which of two equally likely events has occurred. After observing y_1 the probability of x_1 has changed to $p(x_1|y_1) = 0.9$, and its

²The units "nat" and "Hartley" are often used with the bases e and 10 , respectively. The decimal unit is in honor of communication pioneer R.V.L. Hartley.

uncertainty now is $-\log_2 0.9 = 0.152$ bit. Its uncertainty has decreased because its probability has increased from its original value. The information received about the event x_1 is $1 - 0.152 = 0.848$ bit. At the same time, the probability of x_2 has changed to $p(x_2|y_1) = 0.1$. Its uncertainty now is $-\log_2 0.1 = 3.322$ bit. The information received about the event x_2 is -2.322 bit. If it later turns out that x_1 actually occurred then the information received gave a correct indication and is positive. However, if an error occurred then it is negative.

The information gained by the observation can then be expressed as

$$I(x; y) = \text{Initial uncertainty} - \text{Final Uncertainty} \quad (2.30)$$

The amount of information about x that is produced by the observation y equals the initial uncertainty about x minus the uncertainty of x conditioned on y .

The term "uncertainty" can be related to the feeling of surprise. There is little surprise in the occurrence of an event with probability close to one, but there is great surprise at the occurrence of a rare event.

2.2.1 Self Information

Suppose that an observation y removes all of the uncertainty about x . That is, $p(x|y) = 1$. This would be the case for an ideal error-free channel. Then, since $\log p(x|y) = 0$, the amount of information provided must equal the initial uncertainty about x . We call this the self information.

Definition 10 The self information associated with an event $x \in X$ is

$$I(x) = -\log p(x) \quad (2.31)$$

It is common to write $I(p)$ when one wants to focus on the probability $p(x)$ rather than on the event. A plot of $I(p)$ is shown in Figure 2.3. We see that an event has no uncertainty when $p = 1$ and infinite uncertainty when $p = 0$. $I(p)$ is a measure of our "surprise" when an event of probability p occurs. We are not very surprised if $p \approx 1$ and very surprised if $p \approx 0$.

We have applied the model of partial information gain to transmission through a binary communication channel. It can also be applied to code

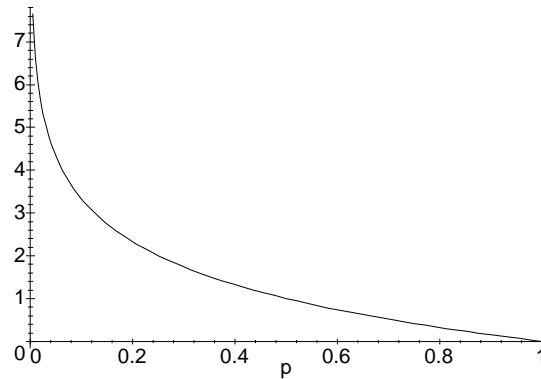


Figure 2.3: Uncertainty of an event as a function of its probability.

applications. We have seen that a set of events can be represented by code words. When a codeword is transmitted symbol-by-symbol, each symbol adds to the quantity of information carried by the word. As each symbol is received at the other end, it adds information about the event that was transmitted. Each symbol of the received codeword will make some events less likely and others more likely until the final digit makes one event certain and the others impossible. We will explore this when we examine decoding trees below.

2.2.2 Entropy as Average Self Information

Suppose we want to know the average amount of information that is provided by the source. This is the amount of information per input event that will have to be carried by the channel or stored in a recording system. We can find the answer by averaging over the self information of each $x \in X$.

The self information $I(x) = -\log p(x)$ is a random variable whose value is determined by the event x . Being a random variable, it is possible to compute its average value.

$$\begin{aligned}
 E[I] &= \sum_{x \in X} p(x) I(x) \\
 &= \sum_{x \in X} p(x) \log p(x) = H(X)
 \end{aligned} \tag{2.32}$$

This is recognized as the quantity we named entropy in Lecture 2. Therefore, entropy can be interpreted as the average information of the underlying event space. When the logarithm of base 2 is used, the units are bits per event. We found that the average codeword length \bar{n} cannot be less than H digits. Thus, each digit of a binary code cannot, on the average, carry more than one bit of information.

In Lecture 2 we showed that the entropy for an event space of size q is bounded by

$$0 \leq H(E) \leq \log_2 q \quad (2.33)$$

with $H(E) = 0$ if some $p(e_i) = 1$ and $H(E) = \log_2 q$ if all $p(e_i) = 1/q$. In the first case only one of the events can occur so there is no uncertainty and in the second any event can occur with equal probability so the uncertainty is maximum.

The BSC provides a useful example. Let the input probabilities be $p(x_1) = p$ and $p(x_2) = 1 - p$. The entropy associated with a binary event space is a function of the single variable, p , and can be plotted as a simple curve

$$H(p) = -p \log_2 p - (1 - p) \log_2 (1 - p) \quad (2.34)$$

As illustrated in Figure 2.4, this function is symmetric about $p = 0.5$, is maximum where the events are equally probable and is zero where either event is certain. This means that if $p \neq 0.5$ it is possible to record or communicate the information from this source at a rate of less than one bit per event. This can, in fact, be approximated closely by Huffman encoding of doublets or triplets from the source.

The entropy now has a meaning in terms of the average quantity of information needed to record the output of a source. We have shown that the entropy of a source that generates statistically independent events can be calculated by (2.32). There are many sources that do not fit the independent event model, and more analysis will be needed to construct a method to compute it for such situations. For example, estimation of the entropy of English text is very difficult because one needs to take all of the constraints of spelling and grammar into account somehow. But we will find that however difficult it is to estimate H , it provides a lower bound on the amount of data needed to represent events. It is therefore a fundamental quantity in the design of efficient communication or recording systems.

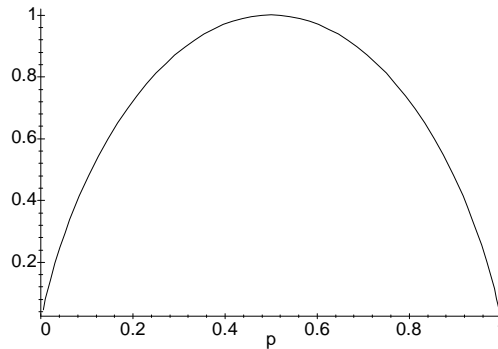


Figure 2.4: Entropy $H(p)$ as a function of p for a binary event space

It should be noted that entropy is associated with long sequences of events, not with individual events. It is an average over the information produced by many events. It is meaningless to speak of the entropy of an individual event.

2.2.3 Decoding Trees and Partial Information

Let us suppose that events are to be represented by codewords whose symbols are drawn from an alphabet of size r . In an efficient code, codeword length is related to event probability so that short codewords are associated with events of high probability. We will now see that codeword length is related to entropy.

The uncertainty of an event can be related to the length of its codeword. Events with high probability and, therefore, low uncertainty should get short codewords while those with low probability, and high uncertainty, should get long codewords. An efficient binary code has words of length $\lceil \log_2 p_i \rceil$. Thus, the binary codeword for an event e_i should have about the same number of digits as the uncertainty measured in bits. They cannot be equal unless the uncertainty is an integer, in which case $p_i = 2^{-n_i}$ and $\bar{n} = H(E)$. The code in Table 2.2 is an example.

Suppose now that someone begins to show you a codeword a digit at a time. The...rst digit gives you some information about the event; the second

i	p_i	j	$\log p_i$	w_i	n_i
1	1/2	1		0	1
2	1/4	2		10	2
3	1/8	3		110	3
4	1/8	3		111	4

Table 2.2: Example of a 100% efficient code

i	p_i	j	$\log p_i$	w_i	n_i
1	0.4	1.32		0	1
2	0.3	1.74		10	2
3	0.2	2.32		110	3
4	0.1	3.32		111	4

Table 2.3: Example of a code that is less than 100% efficient

gives you more, and so on. Finally, when you have seen all the digits in the codeword, you know which event has occurred. Each digit gives you information and reduces your uncertainty. That is why we use the symbol $I(e_i)$: it represents the information required to remove the uncertainty. We equate information to reduction in uncertainty and measure it in bits.

Because $\bar{n} \approx H(E)$, each digit can convey, on the average, at most one bit of information. We often talk about the symbols '0' and '1' as "bits" when they are really digits. A bit is a unit of information. A digit carries x bits of information, where x depends upon the situation.

A decoding tree for the code in Table 2.2 is shown in Figure 2.5. Each branch shows the digit related to it as well as a pair $(p, I(p))$. Suppose that you are given a codeword digit by digit. Each digit chooses between an upper or lower branch. The first symbol has probability $p = 0.5$ of being 1 or 0, so the first digit carries $\log_2 = 1$ bit of information. If the first symbol is '1' you go from node a to node b. A gain, either path is equally likely, and so on. At each step the new information removes a full bit of uncertainty. If $I(p)$ is summed along the path it equals the total uncertainty of the event, which, in this case, is an integer equal to the codeword length.

Suppose now that the probabilities are slightly different, so that $j \log p_i$ is not an integer. This situation is shown in Table 2.3. The code tree is shown

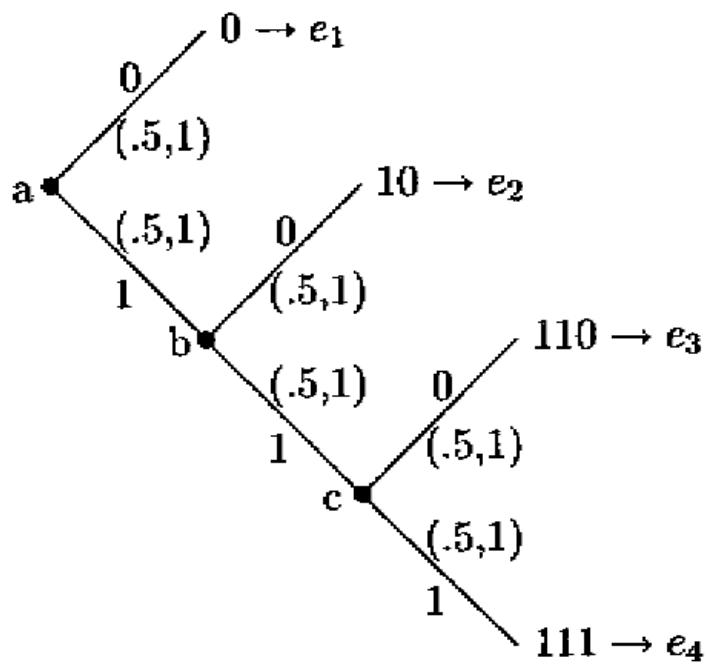


Figure 2.5: D ecoding tree for the code of T able 3.1.

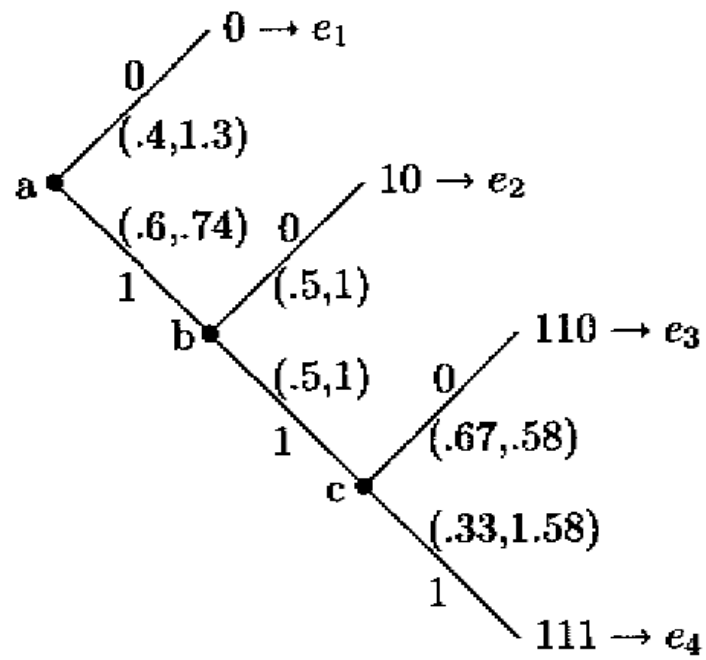


Figure 2.6 Decision tree for the code of Table 3.2.

in Figure 2.6 The probability p of taking a given branch from its node is now not always 0.5, so the uncertainty is not always 1 bit. The uncertainty of each branch is therefore not an integer. The sum of the uncertainties along each path is equal to the uncertainty of the event, and this is close to the codeword length. The average codeword length is now a number slightly greater than the entropy. If one multiplies the probabilities or adds the uncertainties along a path leading to an event, the result is the probability or the uncertainty of the event, shown in columns 2 and 3 of Table 2.3.

We now see that there is a close association between information and uncertainty. Each symbol in a codeword carries information that reduces the uncertainty about the event. Low probability events require more digits in an efficient code because they have more uncertainty.

Each digit of a codeword reduces the uncertainty but, unless it is the last one, does not eliminate it. To make this explicit, let a_{ij} be symbol j of codeword w_i associated with event e_i . Before any digit has been observed, the uncertainty about e_i is $-\log p(e_i)$. After symbol a_{11} is observed the probability of e_i has changed to $p(e_i|a_{11})$, which is the probability conditioned on the observed symbol. The uncertainty must now be $-\log p(e_i|a_{11})$. The difference in the uncertainty is the information provided by a_{11} .

The information about e_i provided by a_{11} is

$$I(e_i|a_{11}) = \log \frac{p(e_i|a_{11})}{p(e_i)}$$

Suppose that event space of Table 2.3 is used and that the event e_3 has occurred. The message is, therefore, $w_3 = (1, 1, 0)$. The original probability is $p(e_3) = 0.2$, which corresponds to 2.32 bits of uncertainty. After the first digit is received the probability is $p(e_3|1) = 1/3$ and the uncertainty has changed to $-\log(1/3) = 1.58$ bits. The information received was $2.32 - 1.58 = .74$ bits. Note that this is the quantity on that branch of Figure 2.6. When the second digit is received the probability is $p(e_3|11) = 2/3$ and the uncertainty is $-\log(2/3) = .58$ bits. The information received was $1.58 - .58 = 1$ bit, which is the quantity on that branch of the tree. The final digit increases the probability to $p(e_3|110) = 1$ so that the uncertainty is removed. This digit carried the remaining .58 bits of information.

This example shows that the amount of information carried by a digit depends upon the event and the preceding digits. Even if the events are independent, it is not necessarily true that the digits in a code sequence are statistically independent. However, we can say that in a 100% efficient

code each digit must carry a full bit of information and must be statistically independent of all other digits. To see this latter point, carry out an analysis of the code of Figure 2.5 in a manner that parallels the preceding paragraph.