Information Theory and Coding
4.1 Introduction

Huffman Codes

Variable-Length Codes:

Chapter 4
The code is not uniquely decodable. Although the property of

\[ \{3, 1\} \cup \{2, 1\} = 1100 \]

The particular received message 1101 could be one of these two:

- 00 = 3
- 11 = 1
- 10 = 2
- 0 = 1

This is not a uniquely decodable system. The code is not uniquely decodable—there are two different strings that have the same code.

Consider the following code:

### 4.3.1. Invariant Codes

- 011 = 3
- 010 = 3
- 010 = 3
- 010 = 3

Exercises

4.3.1. Determine whether the code is uniquely decodable.

### 4.2.2. Invariant Codes

Consider the following code:

- 011 = 3
- 010 = 3
- 010 = 3
- 010 = 3

This is a uniquely decodable system. The code is uniquely decodable—there is only one string that has the same code.

To make our thoughts clear, we make a formal definition of a uniquely decodable code.

- **Definition:** The code is uniquely decodable if and only if the following condition holds:
  - For any two distinct symbols of the code, there are no prefixes of the symbols that are equal.

In summary, the code is uniquely decodable if and only if the code is uniquely decodable.
To get an instantaneous (coming) code (Figure 4.1-1):

\[
\begin{align*}
011 &= \varepsilon_s \\
010 &= \varepsilon_s \\
01 &= \varepsilon_s \\
0 &= \varepsilon_s
\end{align*}
\]

Given that we are to construct a code with the symbols 0, 1, and 2. We can see the trouble—some code words are prefixed or other words; that is, they are the same.

111 = \varepsilon_s
110 = \varepsilon_s
10 = \varepsilon_s
0 = \varepsilon_s

Let’s understand the following: the above code is the bottom part of the diagram. It is clear that all uniquely decodable codes, the instantaneous codes.

4. Constitution of instantaneous Codes

It is clear that all uniquely decodable codes, the instantaneous codes.

Another possibility is to have a code word that is longer than the other.

In the case of the binary I denote the code word that is received. This is known as the occurrence of the code word in the code.

In this example the occurrence is inappropriate since the binary is not one of the symbols.

If we refer to the binary tree diagram, the symbols are 1, 0, and \( \varepsilon_s \).

That the terminal states of this tree are the four source symbols.

No code can be a prefix of any other code.

Of course, each is symbol and then return control to the initial state.

![Decoding tree diagram](image)
4.5 The Kraft Inequality

The Kraft Inequality is a necessary and sufficient condition for the existence of a prefix code. It states that the sum of the probabilities of all codewords in a prefix code must be less than or equal to 1.

\[ \sum_{i=1}^{n} \frac{1}{2^i} \leq 1 \]

To prove this, consider the following steps:

1. **Base Case**: For a prefix code of length 1, the inequality holds because the probability of each symbol is less than or equal to 1/2, and the sum of probabilities for one symbol is 1/2.
2. **Inductive Step**: Assume the inequality holds for prefix codes of length \( k \). We want to show it holds for codes of length \( k+1 \).
3. **Construction**: Given a prefix code of length \( k+1 \), we can construct a prefix code of length \( k \) by removing the last symbol from each codeword.
4. **Application**: By the inductive hypothesis, the sum of the probabilities of the codewords in the \( k \) prefix code is less than or equal to 1. Since we removed the last symbol from each codeword, the sum of the probabilities of the codewords in the \( k+1 \) prefix code is also less than or equal to 1.

Thus, the Kraft Inequality is satisfied for all prefix codes.

**Exercises**

4.4.1 Devise a similar example using six words.

**Figure 4.4.2 Decoding Tree**

The decoding tree is used to decode a prefix code. Each path from the root to a leaf represents a codeword. The decoding process involves following the path that matches the received sequence of symbols.

**Figure 4.4.3 Decoding Tree**

In this construction, the use of the 0 for the first symbol reduces the number of possible codewords. Instead of this, let us use the number of possibilities available below. Which of these two codes is better (more efficient)? This depends on how the messages are encoded.
have the inequality $\leq L \leq f' + f$.

One way to prove the Kraft inequality is by considering the expected number of codewords for each symbol. Let $L_i$ be the length of the codeword for symbol $i$. Then the expected number of codewords is $\sum_{i} 2^{-L_i}$. By the Kraft inequality, this sum is less than or equal to 1.

For two symbols, we have $L_1 \leq 1$ and $L_2 \leq 1$.

If $L_1 = L_2 = 1$, then $L = 2$.

If $L_1 = 1$ and $L_2 = 2$, then $L = 3$.

If $L_1 = 2$ and $L_2 = 1$, then $L = 3$.

If $L_1 = 2$ and $L_2 = 2$, then $L = 4$.

For three symbols, we can show that $L \leq 4$.

If $L_1 = L_2 = L_3 = 1$, then $L = 3$.

If $L_1 = L_2 = 1$ and $L_3 = 2$, then $L = 4$.

If $L_1 = L_2 = 2$ and $L_3 = 1$, then $L = 4$.

If $L_1 = 1$, $L_2 = 2$, and $L_3 = 2$, then $L = 5$.

For any number of symbols, the Kraft inequality is still valid, but the proof becomes more complex.
Symbols. Of the eight binary symbols, there are seven that can happen, consider the case of five symbols. 111, 011, 010, 001, 110, 101, and 100. Suppose that we do not have 7 bits to represent each symbol. But suppose that we could use the seven symbols in a unary system (as in a prefix system), we could use 7-bit code words in a binary system to represent each symbol. The reason for the moment in the earlier block codes, if we had exactly 7-bit code words in a binary system, we would exactly 4.6. Shortened block codes.

4.5.3. Carry out the computation of $\lambda_j$ for a comma code.

4.5.2. Generalize Exercise 4.5.1 to radix $r$.

4.5.1. Does the inequality $\ell(x) \leq 1 + \log_r 3$ satisfy the Kraft inequality? (Use $\lambda_j = 1/\ell(x)$, $\ell(x) = \log_r 3$.)

Exercises

Figure 4.6.2 Decoding tree.

Figure 4.6.1 Decoding tree.
$$d \geq \cdots \geq d \geq d \geq d$$

Both if the entries $l_i$ are not in the opposite order, then we do not have

$$\frac{d}{d_n} \leq \frac{d}{d_n} \leq \frac{d}{d_n} \leq \frac{d}{d_n}$$

If the entries $l_i$ are not in the opposite order, then we do not have

$$\frac{d}{d_n} \leq \frac{d}{d_n} \leq \frac{d}{d_n} \leq \frac{d}{d_n}$$

Now the rest we will make use of the propositions of the various

4.8 Huffman Codes

Huffman codes

Example 4.6.2 Discuss the case of five symbols in the ternary base.

The Huffman codes will be binary codes which are special cases of uniquely decodable codes.

4.6.1 Discuss the case of two symbols in a ternary base.

Exercises

Exhibit 4.6.2

are essentially block codes with small modifications.

and therefore $K = 1$. We will call these shortened block codes. They

(See Figure 4.6.2.) In both cases there are no unused terminals

are essentially block codes with small modifications.

and therefore $K = 1$. We will call these shortened block codes. They

(See Figure 4.6.2.) In both cases there are no unused terminals

are essentially block codes with small modifications.

and therefore $K = 1$. We will call these shortened block codes. They

(See Figure 4.6.2.) In both cases there are no unused terminals

are essentially block codes with small modifications.

and therefore $K = 1$. We will call these shortened block codes. They

(See Figure 4.6.2.) In both cases there are no unused terminals

are essentially block codes with small modifications.

and therefore $K = 1$. We will call these shortened block codes. They

(See Figure 4.6.2.) In both cases there are no unused terminals

are essentially block codes with small modifications.

and therefore $K = 1$. We will call these shortened block codes. They

(See Figure 4.6.2.) In both cases there are no unused terminals

are essentially block codes with small modifications.
The proof of the encoding properties is well explained at the end of Section 1. We simplify our notation at each stage in a shorter code. We simplify encoding, a reduction in each stage to a shorter code. We explain how to achieve this, and because of the encoding inequalities, they must have the same length. And because the encoding inequalities they must have the same length. We thus use two unequal codes.

**Figure 4.8-1 Reduction process**

<table>
<thead>
<tr>
<th>Original</th>
<th>First Reduction</th>
<th>Second Reduction</th>
<th>Third Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

We begin our examination of Huffman codes with encoding into a small, alphabet into which we reduce symbols, like the number of bits in the alphabet. We will use a source symbol for the input 1's and 0's, and code alphabet for the output 1's and 0's. We begin with a source symbol for the input 1's and 0's, and code alphabet for the output 1's and 0's. In Section 4.11 we look at the case of a palindrome, to see how the encoding into Huffman codes works. We use the fact that any code equals the sum of the corresponding bit strings in a palindrome. We will do this for our code.
Length is 2 when (Figure 4.b-3, we will get lengths (1, 2, 3, 4) and the average

\[ Z' = (3)(0.10) + (3)(0.10) + (2)(0.20) + (2)(0.20) + (1)(0.20) + (1)(0.10) = 7 \]

On the other hand, if we push the median states up as high as possible, then in

\[ Z' = (3)(0.10) + (3)(0.10) + (2)(0.20) + (2)(0.20) + (1)(0.20) + (1)(0.10) = 7 \]

Figure 4.b-1, we get lengths (1, 2, 3, 4) and the average length is

If at each stage we put the median states as low as possible, then in

\[ Z = 2 \]

\[ y = d \]

\[ y = d \]

\[ y = d \]

\[ 4 \cdot 0 = d \]

The encoding process is not unique in several respects. First, the

Sec. 4.b. Huffman codes

Huffman codes are the shortest possible code.

For the other it must be less than, which is impossible. Therefore,

Therefore, the Huffman code length is 1.

We apply this 30 of the encoding needs we are computing. Since

Therefore, in either code we shorten the code and decrease the average

\[ 1 - b_d + b_d \]

by the amount
decodability of each bit after the reduction, we have shortened the code

can this the two equal probabilities symbols so that they are both

codewords without changing the average code length. Thus, this we

the following proposition: Symbols having the same length may be in

If there are more than two symbols of the maximum length, we can use

\[ 1 - b_d + b_d \]

so that the code length is reduced by

\[(1 - b_d + b_d)(1 - b) \]

and after the reduction they contribute

\[(1 - b_d + b_d) \]

continue

two equal probabilities symbols. Here we reduce a code, the two symbols

must have the same decision made in common, and they must be the

If there are only two symbols of maximum length in a tree, they

end.

In the Huffman algorithm, the decision is made to put the code in the left.

Compute the two equal probabilities symbols. Here we reduce a code, the two symbols

If there are only two symbols of maximum length in a tree, they

end.

In the Huffman algorithm, the decision is made to put the code in the left.

Compute the two equal probabilities symbols. Here we reduce a code, the two symbols

If there are only two symbols of maximum length in a tree, they

end.

In the Huffman algorithm, the decision is made to put the code in the left.

Compute the two equal probabilities symbols. Here we reduce a code, the two symbols

If there are only two symbols of maximum length in a tree, they

end.

In the Huffman algorithm, the decision is made to put the code in the left.

Compute the two equal probabilities symbols. Here we reduce a code, the two symbols

If there are only two symbols of maximum length in a tree, they

end.

In the Huffman algorithm, the decision is made to put the code in the left.

Compute the two equal probabilities symbols. Here we reduce a code, the two symbols

If there are only two symbols of maximum length in a tree, they

end.

In the Huffman algorithm, the decision is made to put the code in the left.

Compute the two equal probabilities symbols. Here we reduce a code, the two symbols

If there are only two symbols of maximum length in a tree, they

end.

In the Huffman algorithm, the decision is made to put the code in the left.

Compute the two equal probabilities symbols. Here we reduce a code, the two symbols

If there are only two symbols of maximum length in a tree, they

end.

In the Huffman algorithm, the decision is made to put the code in the left.

Compute the two equal probabilities symbols. Here we reduce a code, the two symbols

If there are only two symbols of maximum length in a tree, they

end.

In the Huffman algorithm, the decision is made to put the code in the left.

Compute the two equal probabilities symbols. Here we reduce a code, the two symbols

If there are only two symbols of maximum length in a tree, they

end.

In the Huffman algorithm, the decision is made to put the code in the left.

Compute the two equal probabilities symbols. Here we reduce a code, the two symbols

If there are only two symbols of maximum length in a tree, they

end.
4.9 Special Cases of Huffman Coding

4.9.1 Show two distinct decoding trees for the case \( \frac{d}{2} \), \( \frac{d}{2} \) and \( \frac{d}{2} \).

4.9.2 Give the Huffman code for the probabilities \( \frac{d}{2} \), \( \frac{d}{2} \) and \( \frac{d}{2} \).

4.9.3 Give the Huffman code for \( d \), \( d \), and \( d \).

4.9.4 Give the Huffman code for \( d \), \( d \), and \( d \).

4.9.5 Give the Huffman code for \( d \), \( d \), and \( d \).

4.9.6 Give the Huffman code for the probabilities \( \frac{d}{2} \), \( \frac{d}{2} \) and \( \frac{d}{2} \).

Exercises

Exercises 4.9

Exercises 4.9

Exercises 4.9

Exercises 4.9

Exercises 4.9

Exercises 4.9

Exercises 4.9

Exercises 4.9

Exercises 4.9

Exercises 4.9

Exercises 4.9

Exercises 4.9
Introduction

Chapter 6

First Theorem

Entropy and Shannon's

code compression can be found in Ref. [60]. The topic is not of current research interest to do more than point out the obvious fact that a number of methods of code compression that are used in transmission, so that both the sender and the receiver have the same encoding, correspond to code compression either from a single code or by keeping the current low or even high capacity, but it does catch spurious pressure. Thus, of course, we are not necessarily in a single code with transmission, so that both the sender and the receiver have the same encoding. Another way, when using serial transmission, is to embed the channel in a greater block with transmission, so that both the sender and the receiver have the same encoding. Another way, when using serial transmission, is to embed the channel in a greater block with transmission, so that both the sender and the receiver have the same encoding.
Let us examine and illustrate this additivity property. Let $a$ be a constant, and let $x$ be a random variable. Then, we have

$$E[X] = (d)E + (d)X = (d)E$$

for some base of the log system.

$$\frac{d}{d\log a} = (d)E$$

Informally, this assumption is similar to the constant $x$ as $1$, and we have

$$\frac{d}{d\log a} = (d)E$$

where $a$ is a constant greater than 0, $x$ is a random variable, and $E$ is the expected value of $x$. This assumption is satisfied in the log system. However, for some constants $x$ and some base of the log system, the first line

$$E[X] = (d)E$$

or after some further manipulation,

$$(w_{1})E = (d)E$$

and hence

$$w_{1}E = d \Rightarrow E = \frac{d}{w_{1}}$$

6.2 Information

Mathematical insights are covered in Chapter 9. The main theorem will appear in a later section—once a lot of preliminary work is done. When the theorem is proved, the proof of the noise and the proof of the theorem are understood. The proof of the noise is straightforward, as it identifies the problem of the noise. Whether or not this theorem is of any use, we assume in this section 1, that the theory is very similar. Yet we can see in Section 2, that the theory is very similar.
Entropy

6.3. Entropy

The entropy of a system is a measure of the number of microstates that correspond to a given macrostate. It is also a measure of the uncertainty in the system's state.

1. The two values of p that were considered, namely 0 and 1, and the solution is essentially unique. The continuity assumption covers $0 \leq p \leq 1$

2. $H = \sum p \log p$

Hence the right-hand side of the equation becomes $\log 2$

which is not 0.1. Now for any number $\beta$ there is only one such that (assuming $p_0$)

the choice makes the right-hand side of the equation above equal to

$\frac{(d/d) \log p}{(d/d) p} = \frac{c}{d}$

Next, pick $c = \frac{d}{d}$

$\left[ \frac{d}{d} \log c - (\log p) \right] u = \frac{d}{d} \log c - (\log p)$

Now take the difference of the two solutions and add to the constant

$(d/d) u = (d/d) c$

Hence we have the equation

$(d/d) u = (d/d) c$

which is, of course, the

the outcome of the de', then the derived formula of $d$ is

$\log \frac{1}{p} + \gamma = \gamma_{\log p} + \gamma p$
Although we spoke of the entropy function $H(S)$ as a function of

<table>
<thead>
<tr>
<th>Sum</th>
<th>1.8494</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>0.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>

and $p$ then from the data of Appendix B, column 3, we have,

The entropy function $H(S)$ is a function of $p$ and $d$ only.

The Shannon entropy is defined as the sum of the probability of each symbol, where the probability is $p_i$.

If you know the entropy of the distribution, you can use it to predict the probability of the next symbol.

But if you do not know the entropy of the distribution, you can use the next symbol to predict the probability of the next symbol.

If you do not know the entropy of the distribution, you can use the next symbol to predict the probability of the next symbol.

If you do not know the entropy of the distribution, you can use the next symbol to predict the probability of the next symbol.

For each symbol $s$, we will get

From this it follows that on the average, over the whole alphabet of symbols, $s$.

Thus we can define the information content of the symbol $s$ as

and the information per symbol of the alphabet $S$ is

Hence the information is


$$N[d_0 \cdots d_{(d-1)d}] =$$

$$N[d_0 \cdots d_{(d-1)d}] =$$

$$= d$$
\[
\lim_{x \to 0^+} \left( \frac{x/1}{x^2 \log x} \right) = 0
\]

To prove this, we write it as

\[
0 = \lim_{x \to 0^+} x^0 \log x
\]

Ignore the factor \(x^0\). We also need the property (since \(x^0 = 1\)) that the derivative is zero, we can set \(1/e\) where the derivative is zero.

We see both the infinite slope at \(x = 0\) and then the maximum occurs.

\[
\lim_{x \to 0^+} \left( \frac{d}{1} \right) \frac{d}{\log d} = \lim_{x \to 0^+} \left( \frac{d}{1} \right) \frac{d}{\log d} = \left( \frac{d}{1} \right) \frac{d}{\log d}
\]

The function of \(x\) is graphed in Figure 6.3-1. From the graph, we see that the only two points with probability we will write \(H(f)\), when there are only two values, with probabilities we will refer to the alphabet \(d\) and write \(H(f)\), although

---

**Figure 6.3-1**

\[
\begin{align*}
0 & \quad 0.05 \quad 0.1 \quad 0.2 \quad 0.4 \quad 1.0 \\
\log(d) & \quad -2 \quad -1 \quad -0.5 \quad 0 \quad 1
\end{align*}
\]
The entropy has properties of both the arithmetic mean (the average) distribution much as the variance in statistics summarizes a distribution.

The entropy function of a distribution summarizes one aspect of a

We have Table 6.2-1 (corresponding to Table 4.9-1).

The entropy, \( H \), is easily computed:

\[
N \log_2 \frac{N}{L} + \frac{N}{L} = \log_2 N
\]

Then

\[
\frac{N}{N}, \log_2 \frac{1}{N} = \frac{N}{L}
\]

\[
\frac{N}{L} + 1 - \frac{N}{L} = \frac{N}{L}
\]

\[
0 = \frac{N}{L}
\]

\[
1 = \frac{N}{L}
\]

Using the recursive definition:

\[
\frac{N}{N}, \log_2 \frac{1}{N} = \frac{N}{L}
\]

where

\[
N \log_2 \frac{N}{L} + \frac{N}{L} = \log_2 N
\]

\[
\left( \frac{N}{L} \log_2 \frac{N}{L} + \frac{N}{L} \right) \frac{N}{L} = \log_2 N
\]

\[
\left[ \left( \frac{N}{L} \log_2 \frac{N}{L} \right) \frac{N}{L} \right] \frac{N}{L} = \log_2 N
\]

\[
\left[ N \log_2 \frac{N}{L} - \left( \frac{N}{L} \right) \log_2 \frac{N}{L} \right] \frac{N}{L} = \log_2 N
\]

\[
\frac{1}{N} \log_2 N
\]

The entropy for the \( N \) items is

\[
\frac{N}{L} \log_2 \frac{N}{L} + \frac{N}{L} = \log_2 N
\]

where

\[
(N - \log_2 N + \frac{N}{L} = \frac{N}{L} = \log_2 N
\]

\[
\frac{N}{L} \log_2 \frac{N}{L} + \frac{N}{L} = \log_2 N
\]

ability.

with \( \frac{N}{L} \) proportional to \( \frac{N}{L} \). Thus for \( N \) items, we have the prob-

is the entropy of Zipf's law (Section 4.9), where the frequency of the

A final example of the computation of the entropy of a distribution

one event.

As we see from this example, whenever all the probabilities are equal,

the entropy over the alphabet is the same as the information for any

\[ H(S) = 2.5849 \ldots \text{bits of information} \]

\[ H(S) = \log_2 6 \]

Figure 6.2.2. Entropy function for two probability distributions.
Finding the tangent line at the point \((1, 0)\), we find that the slope of the tangent line is 1. A first property of the \(\log_b x\) function can be seen from Figure 6.4-1. The graph of the function has a number of mathematical properties that are very useful. We examine these properties deeper into the text.

The entropy function is a measure of the amount of uncertainty or unpredictability in a set of events that can occur. Thus, in designing an experiment, we usually want to select those experiments in which the occurrence of the outcomes is as independent as possible. The maximum entropy function is a measure of the amount of uncertainty in a set of events. From this, we can define the amount of uncertainty in a set of events. To do this, we need to know the amount of uncertainty in a set of events. To do this, we need to know the amount of uncertainty in a set of events.

The entropy function measures the average amount of uncertainty, and

\[ H(X) = -\sum_{i} p(x_i) \log p(x_i) \]

6.4 Mathematical Properties of the Entropy Function

\[ \log 0.3 + \frac{1}{2} \log 0.7 = \log (0.3 \times 0.7^{1/2}) = \log 0.49 \]

6.3-2 For \( \text{a draw of a card from a deck of 52 cards} \), how many bits of information do we get from one draw of a card?

\[ H(X) = -\sum_{i} p(x_i) \log p(x_i) \]

Exercise

Which is a weighted geometric mean?

\[ \frac{d}{I} \begin{cases} \log 2 = \frac{d}{I} \\ \log 3 = \frac{d}{I} \end{cases} \]

and the geometric mean. We have

\[ H(X) = -\sum_{i} p(x_i) \log p(x_i) \]

<table>
<thead>
<tr>
<th>(X)</th>
<th>(p(x))</th>
<th>(\log p(x))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11.5</td>
<td>1.115</td>
</tr>
<tr>
<td>6</td>
<td>1.5</td>
<td>0.229</td>
</tr>
<tr>
<td>6</td>
<td>2.5</td>
<td>0.429</td>
</tr>
<tr>
<td>9</td>
<td>3.5</td>
<td>0.531</td>
</tr>
<tr>
<td>8</td>
<td>4.5</td>
<td>0.698</td>
</tr>
<tr>
<td>2</td>
<td>5.5</td>
<td>0.796</td>
</tr>
<tr>
<td>2</td>
<td>6.5</td>
<td>0.894</td>
</tr>
<tr>
<td>4</td>
<td>7.5</td>
<td>0.973</td>
</tr>
<tr>
<td>8</td>
<td>8.5</td>
<td>1.115</td>
</tr>
</tbody>
</table>

Table 6.3-1: Entropy of the flip's law
\[ b \cdot \cdots \cdot z \cdot 1 = f \]  
\[ 0 = x + \left[ I - \frac{\langle I \rangle^2 \log}{I} \right] z \cdot \frac{\log}{I} = \frac{\langle d \rangle}{f} \]

\( (1 - \langle d \rangle) a + \left( \frac{\langle d \rangle}{I} \right)^2 \log \left[ \frac{\log}{I} \right] = \langle d \cdot \cdots \cdot d \cdot \cdot d \rangle f \)

Let us derive the function.

For a more detailed derivation of the important result, we use the

**Equation (6.4–3)**

\[ b \cdot \log = (s)^2 H \]

Therefore,

\[ 0 \leq b \cdot \log - (s)^2 H \]

Using the Gibbs inequality (6.4–2) with \( b/I = \langle x \rangle \), we have

\[ \left( \frac{\langle d \rangle}{I} \right) ^2 \log \left[ \frac{\log}{I} \right] = \]

\[ \langle d \rangle \cdot \log - \left( \frac{\langle d \rangle}{I} \right) ^2 \log \left[ \frac{\log}{I} \right] = \langle d \rangle \cdot \log - (s)^2 H \]

We begin by considering the quantity

\[ 1 = \langle d \rangle ^2 \]  
with \[ \langle d \rangle ^2 \log \left[ \frac{\log}{I} \right] = (s)^2 H \]

one does not = 1 and all the others. \( b/I = \langle x \rangle \) and the second probability distribution be

The second result we need is the fundamental relation between

\[ (1 - x) \leq x \cdot \log \]  

Thus we have for all \( x \geq 0 \), the useful inequality

\[ 1 - x = \lambda \]

or

\[ (I - x) I = 0 - \lambda \]

so that the incorrect one is

\[ I = \frac{1 - x}{(x \cdot \log x)} \]

Therefore, back to logs base 2, we have the fundamental Gibbs in-
Upon expanding the log term into a sum of logs, we notice that one

\[ 0 = \left( \frac{d}{d\Omega} \right)^2 \log d \frac{1}{2} \leq \Omega \]

can use the fundamental theorem in inequality (6-2.6)

\[ \text{where, of course,} \]

\[ \frac{\eta}{\eta_d} = \Omega \]

(6-2.6)

We now define the numbers \( \Omega \) (pseudo probabilities):

\[ I = \left( \frac{\eta}{\eta_0} \right)^{1/2} = \Omega \]

(6-2.7)

The Kittle inequality (4.1.2), we have

the error inequality (4.1.2), we have

From some definitions of the numbers is represented in some tables. From

the error inequality (4.1.2), we have

This is a slightly more general definition of the error of the

Given any distribution \( H(x) \),

We now prove a fundamental relationship between the average code

6.5 Entropy and Coding

\[ b^u | f = \left( \frac{b}{I}, \cdots, \frac{b}{I} \right)^{H} \]

namely,

\[ 0 = \left( 0, \cdots, 0, \log b \right)^{H} \]

For any other distribution \( \eta \), the equality holds.

\[ b^u | f = \left( \frac{\eta}{\eta_0} \right)^{H} \]

since everything in this equation is a constant, it follows that each

For any other distribution \( \eta \), the equality holds.

\[ b^u | f = \left( \frac{\eta}{\eta_0} \right)^{H} \]

Since everything in this equation is a constant, it follows that each
In terms of the average length $L$ of the code (6.5-4), we have
\[
1 + (\frac{d}{I})^2 > 1 \Rightarrow (\frac{d}{I})^2 \leq \frac{1}{\log d}
\]

Therefore, there is an unambiguous

\[
\frac{d}{I} < \frac{1}{\log d}
\]

Since $d'$ is the unique when we sum this inequality, we get
\[
\frac{d}{I} < \frac{1}{\log d}
\]

Take the reciprocal of each term, we obtain
\[
\frac{1}{d'} > \frac{d}{I} \leq \frac{1}{\log d}
\]

Since the two extreme values just span a unit length, removing the logarithm, we get
\[
1 + (\frac{d}{I})^2 \log d > 1 \Rightarrow (\frac{d}{I})^2 \log d
\]

In section 6-5, we assumed that the code word lengths $l_i$ were given.

Shannon-Fano Coding

6.6 Shannon-Fano Coding

Table 6-3 shows how Huffman coding approaches the entropy for the

\[
\eta = \log d
\]

where $d$ is the average code word length.

Shannon-Fano Coding is less efficient than Huffman coding but when you can go directly from the probability $p$ to the code word length $l$, Shannon-Fano coding is less efficient than Huffman coding, but not when you have to go through the whole set of probabilities.

The whole set of probabilities; then for each $p_i$ there is a corresponding probability $p_i^*$ such that $\sum p_i^* = 1$. Given the source symbol $s_i$ and the code word length $l_i$, the source symbol $s_i$ is produced by the probability $p_i$ to the nearest approximating length $l_i^*$. The problem is to minimize the average length $L$. Suppose we are asked to provide a code word for each symbol. Huffman Coding is an example where the code word lengths are given. Huffman Coding is a technique that allows for the coding of binary digital data to achieve a minimum average code word length. It is based on the principle of assigning shorter code words to more frequent symbols and longer code words to less frequent symbols.

Shannon-Fano Coding is a technique that allows for the coding of binary digital data to achieve a minimum average code word length. It is based on the principle of assigning shorter code words to more frequent symbols and longer code words to less frequent symbols. In Section 6-5, we assumed that the code word lengths $l_i$ were given.
We have the table:

\[
\begin{array}{c|c}
\frac{2^2}{1 - \frac{2}{r}} + 1 = \frac{2^2}{r} + \frac{2^2}{r - 1} & \text{for } r = 2, 3, 4, 5, 6, 7, 8
\end{array}
\]

For Shannon–Fano, we have

\[
\frac{2^2}{r} + \frac{2^2}{r - 1} = \text{as} \quad I = \frac{2^2}{r} \quad \text{and } \frac{2^2}{r - 1}
\]

We then assign

<table>
<thead>
<tr>
<th>3</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td>0</td>
</tr>
<tr>
<td>1 1</td>
<td>1</td>
</tr>
</tbody>
</table>

In order: For example, from the probabilities

\[
\begin{align*}
\text{Shannon–Fano:} & \quad 3 & \quad 2 & \quad 1 & \quad 0
\end{align*}
\]

Thus for Shannon–Fano coding we again have the entropy as a lower bound.