ECE 154C
Spring 2010

Lecture Notes #1 (Source Coding)
* Note that the **source encoder** converts all types of information to a stream of binary digits.

* Sometimes the output of the **source decoder** must be an exact replica of the information (e.g., computer data) - called **noiseless coding**.

* Other times the output of the **source decoder** can be approximately equal to the information (e.g., music, TV, speech) - called **coding with distortion**.
WHAT WE WILL COVER IN ECE 154C
(REFERENCE: CHAPTER 10 ZIEMER & TRAUTER)

I SOURCE CODING - NOISELESS CODES

- Basic idea is to use as few binary digits as possible and still be able to recover the information exactly.

- Topics include:
  - Huffman codes
  - Shannon-Fano codes
  - Ternary codes
  - Entropy of source
  - Lempel-Ziv codes

II SOURCE CODING WITH DISTORTION

- Again idea is to use the minimum number of binary digits for a given value of distortion.

- Topics include:
  - Gaussian source
  - Optimal quantizing

III CHANNEL CAPACITY OF A NOISY CHANNEL

- Even if channel is noisy, messages can be sent essentially error free if extra digits are transmitted.
- Basic idea is to use as few extra digits as possible

- Topics covered
  - Channel capacity
  - Mutual information
  - Some examples

II Channel Coding

- Basic idea - detect errors that occurred on channel and then correct them.

- Topics covered
  - Hamming code
  - General theory of block codes
    (Parity check matrix, generator matrix, minimum distance, etc.)
  - LDPC codes
  - Turbo Codes
  - Code performance
Basic concepts came from one man!

Claude Shannon

Shannon used simple models that captured the essence of the problem.

Example 1 - Simple model of a source (called a Discrete Memoryless Source or DMS)

\[
\begin{array}{c}
\text{INFO} \\
\text{SOURCE}
\end{array}
\cdots \text{ACBADAABB} \cdots
\]

- 4 letters \( (A, B, C, D) \)
- i.i.d. \( (\text{independent and identically distributed}) \)
- \( P(A) = p_1, P(B) = p_2, P(C) = p_3, P(D) = p_4 \)
- \[
\frac{1}{2} \sum_{a=1}^{4} p_a = 1
\]
- Simplest code:
  - \( A \rightarrow 00 \)
  - \( B \rightarrow 01 \)
  - \( C \rightarrow 10 \)
  - \( D \rightarrow 11 \)

Question: Can we use fewer than 2 binary digits per source letter and still recover information from the binary sequence? Answer: Depends on values of \( (p_1, p_2, p_3, p_4) \)
EXAMPLE - 8 letters

Source letters Σ a, b, c, d, e, f, g, h

Probabilities Σ p_a p_b p_c p_d p_e p_f p_g p_h

<table>
<thead>
<tr>
<th>Code</th>
<th>Code 1</th>
<th>Code 2</th>
<th>Code 3</th>
<th>Code 4</th>
<th>Code 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>01</td>
<td>10</td>
<td>10</td>
<td>01</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>10</td>
<td>110</td>
<td>110</td>
<td>011</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
<td>11</td>
<td>1110</td>
<td>1110</td>
<td>0111</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>000</td>
<td>1110</td>
<td>1110</td>
<td>0111</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>001</td>
<td>11110</td>
<td>11110</td>
<td>01111</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>010</td>
<td>11110</td>
<td>11110</td>
<td>01111</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>011</td>
<td>111110</td>
<td>111110</td>
<td>011111</td>
</tr>
</tbody>
</table>

U.D. Yes No. Yes No. Yes No. Yes No.
Instr. Yes No. Yes No. Yes No. Some 5

Best Code $\rho$

\[
\bar{E} = \frac{1}{2} \times 1 + \frac{1}{4} \times 2 + \frac{1}{8} \times 3 + \frac{1}{16} \times 4 + \frac{1}{32} \times 6
\]

\[
= 0.5 + 0.5 + 0.375 + 0.25 + 0.375 = 2
\]

$H_2(8) = 2$...
\[
\begin{align*}
L &= (0.1 + 0.1 + 0.2 + 0.2 + 0.3 + 0.5 + 1) \\
&= 2.4
\end{align*}
\]

But can do better by encoding 2 or more at a time !
Pictorial Diagram of Simplest Code

INFO SOURCE \[\text{ABBC...}\] SOURCE ENCODER \[\text{O0010110...}\] TO COMM. CHANNEL

CODE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Code Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>00</td>
</tr>
<tr>
<td>B</td>
<td>01</td>
</tr>
<tr>
<td>C</td>
<td>10</td>
</tr>
<tr>
<td>D</td>
<td>11</td>
</tr>
</tbody>
</table>

CODE BOOK

O0010110... FROM COMM. CHANNEL (ERROR FREE)

SOURCE DECODER \[\text{ABBC...}\] INFO SINK

Average length of code words \(\bar{L} = 2(p_1 p_2 + p_2 p_3) = 2\)
EXAMPLE 2: SIMPLE MODEL FOR NOISY CHANNEL

Model from 15413

If \( s_0(1) = -s_1(1) \) equally likely signals,

\[
p_{\text{error}} = Q\left(\frac{2E}{N_0}\right) = \rho
\]

SHANNON MODEL

\[ P_{Y|X}(y|x) = \begin{cases} 1-p & y = x \\ p & y \neq x \end{cases} \]

This model is called a binary symmetric channel (or BSC) for short.

Question: Can we send information "error-free" over such a channel even though \( p \neq 0 \).

Answer: The answer is essentially yes if rate of the code (to be defined) is less than the capacity of the channel (to be defined) and e e
BACK TO SIMPLE EXAMPLE OF A SOURCE

\((p_1, p_2, p_3, p_4) = (1/2, 1/4, 1/2, 1/8)\)

\[
\begin{array}{c}
\text{Source} \\
\text{Encoder}
\end{array}
\]

\[
\begin{array}{c|cccccccccc}
\text{IN} \rightarrow & A & A & B & C & D & A & B & C & A & D & \ldots & \text{Source} \\
\text{Encoder} & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & \ldots
\end{array}
\]

\[
\begin{array}{c|c}
p(A) & 1/2 \\
p(B) & 1/4 \\
p(C) & 1/8 \\
p(D) & 1/8
\end{array}
\]

ASSUMPTIONS

1. ONE MUST BE ABLE TO RECOVER THE SOURCE SEQUENCE FROM THE BINARY SEQUENCE
2. ONE KNOWS THE "START" OF THE BINARY SEQUENCE AT THE RECEIVER.
3. ONE WOULD LIKE TO MINIMIZE THE AVERAGE NUMBER OF Binary DIGITS PER SOURCE LETTER.

EXAMPLES OF CODES

1. 

\[
\begin{array}{c|c}
\text{A} & 00 \\
\text{B} & 01 \\
\text{C} & 10 \\
\text{D} & 11
\end{array}
\]

\[
\begin{array}{c}
\text{ABAC} \\
0010010 \\
\rightarrow \text{ABAC}
\end{array}
\]

\[
\begin{array}{c}
\text{AV} = 2
\end{array}
\]

2. 

\[
\begin{array}{c|c}
\text{A} & 0 \\
\text{B} & 1 \\
\text{C} & 00 \\
\text{D} & 10
\end{array}
\]

\[
\begin{array}{c}
\text{AABD} \\
00110
\end{array}
\]

\[
\begin{array}{c}
\text{CBD} \\
\text{CAN'T USE THIS CODE}!!!
\end{array}
\]

\[
\begin{array}{c}
\text{VIOLATES ASSUMPTION 2}
\end{array}
\]
\( (\frac{1}{3})A \rightarrow 0 \)
\( (\frac{1}{4})B \rightarrow 10 \)
\( (\frac{1}{6})C \rightarrow 110 \)
\( (\frac{1}{8})D \rightarrow 111 \)

**Decoding Rule**

1. Follow binary sequence until you reach a code word.
2. Continue.

\[
I = \frac{1}{2}I_1 + \frac{1}{4}I_2 + \frac{1}{6}I_3 + \frac{1}{8}I_4 = 1 \frac{3}{4}
\]

No code that satisfies assumptions can have smaller average (\( I \) of binary digits per info symbol). We will find out why this is true later.

**Tree Description of Code**
**Some Definitions**

**Block Code** - Each source symbol is represented by some sequence of code symbols called code words.

**Non-Singular Code** - Code words are distinct.

**U.D. (Uniquely Decodable) Code** - Every concatenation of m code words is distinct for every finite m.

**Instantaneous Code** - A U.D. code where we can decode each code word without seeing subsequent code words.

**Examples of Block Codes (Binary Case)**

<table>
<thead>
<tr>
<th>Source Symbols</th>
<th>Code 1</th>
<th>Code 2</th>
<th>Code 3</th>
<th>Code 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>00</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>01</td>
<td>10</td>
<td>01</td>
</tr>
<tr>
<td>C</td>
<td>00</td>
<td>11</td>
<td>110</td>
<td>011</td>
</tr>
<tr>
<td>D</td>
<td>01</td>
<td>11</td>
<td>110</td>
<td>111</td>
</tr>
</tbody>
</table>


Instant, Instant, Not Instant.

A necessary and sufficient condition for a code to be instantaneous is that no code word be a prefix of any other code word.
Coding several source symbols at a time

<table>
<thead>
<tr>
<th>Source Symbols</th>
<th>Prob.</th>
<th>U.D. Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>.5</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>.35</td>
<td>10</td>
</tr>
<tr>
<td>C</td>
<td>.25</td>
<td>11</td>
</tr>
</tbody>
</table>

\[ z = 1.5 \] (Binary digits / source symbol)

<table>
<thead>
<tr>
<th>2 Symbols</th>
<th>Prob.</th>
<th>U.D. Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA</td>
<td>.25</td>
<td>01</td>
</tr>
<tr>
<td>AB</td>
<td>.175</td>
<td>11</td>
</tr>
<tr>
<td>AC</td>
<td>.075</td>
<td>0010</td>
</tr>
<tr>
<td>BA</td>
<td>.175</td>
<td>000</td>
</tr>
<tr>
<td>BB</td>
<td>.125</td>
<td>101</td>
</tr>
<tr>
<td>BC</td>
<td>.0525</td>
<td>1001</td>
</tr>
<tr>
<td>CA</td>
<td>.075</td>
<td>0011</td>
</tr>
<tr>
<td>CB</td>
<td>.0525</td>
<td>10000</td>
</tr>
<tr>
<td>CC</td>
<td>.0225</td>
<td>10001</td>
</tr>
</tbody>
</table>

\[ L = 2.9275 \] (Binary digits / 2 source symbols)

\[ \frac{L}{2} = 1.46375 \] (Binary digits / source symbol)

Notes

1. It is more efficient to build a code for 2 source symbols.

2. The codes given above are Huffman codes. The procedure for making Huffman codes will be described next.
Binary Huffman Codes

1. Order probabilities - Highest to Lowest
2. Add two lowest probabilities
3. Reorder probabilities
4. Break ties in any way you want

Example:

1. \( 1, 2, 0.15, 3, 0.25 \rightarrow 0.3 \)
   \( 0.25 \)
   \( 0.2 \)
   \( 0.15 \)
   \( 0.1 \)

2. \( 0.3 \)
   \( 0.25 \)
   \( 0.2 \)
   \( 0.15 \)
   \( 0.1 \)

3, 4. Get either

\( 0.3 \rightarrow 0.3 \)
\( 0.25 \rightarrow 0.25 \)
\( 0.2 \rightarrow 0.2 \)
\( 0.15 \rightarrow 0.15 \)
\( 0.1 \rightarrow 0.1 \)

5. Assign 0 to top branch and 1 to bottom branch (or vice versa)

6. Continue until we have only one probability equal to 1.

7. Average length = \( \bar{L} = \sum \text{of probabilities of combined nodes (i.e., the circled ones)} \)
EXAMPLE CONTINUED

In this case the two ways of breaking the tie led to two different codes with the same set of code lengths. This is not always the case — sometimes we get different codes with different code lengths. (See next example.)
1. BINARY HUFFMAN CODE WILL HAVE THE SHORTEST AVERAGE LENGTH AS COMPARED WITH ANY V.D. CODE FOR THAT SET OF PROBABILITIES. (NO V.D. WILL HAVE A SHORTER AVERAGE LENGTH).

2. THE HUFFMAN CODE IS NOT UNIQUE. BREAKING TIES IN DIFFERENT WAYS CAN RESULT IN VERY DIFFERENT CODES. THE AVERAGE LENGTH, HOWEVER, WILL BE THE SAME FOR ALL OF THESE CODES.

**Example**

![Binary Huffman Code Diagram]
ON OPTIMALITY OF BINARY HUFFMAN CODES

The proof that a binary Huffman code is optimal — that is, has the shortest average code word length as compared with any V.D. code for that same set of probabilities — is omitted.

However, it is based on the fact that in the process of constructing the Huffman code for that set of probabilities other codes are formed for other sets of probabilities, all of which are optimal.

Example: \((P_1, P_2, P_3, P_4) = (\frac{1}{4}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8})\)

```
0  .5
10 .25
110 .125
111 .125

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>110</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>111</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
```

Code for \((.5, .5)\):
- .5 0
- .5 1

Code for \((.5, .25, .25)\):
- .5 0
- .25 10
- .25 11
**Binary Source** \( \{A, B\} \) \( (p_1, p_2) = (.9, .1) \)

We now construct a series of Huffman codes, by encoding **N** source symbols at a time for \( N = 1, 2, 3, 4 \).

\[ N = 1 \]

\[
\begin{array}{c|c}
A & .9 \\
B & .1 \\
\end{array}
\]

**Code Book**

\[
\begin{array}{c|c}
A & \leftrightarrow 0 \\
B & \leftrightarrow 1 \\
\end{array}
\]

\[ \bar{L}_1 = 1 \]

\[ N = 2 \]

\[
\begin{array}{c|c}
A & .81 \\
B & .19 \\
\end{array}
\]

**Code Book**

\[
\begin{array}{c|c}
0 & \leftrightarrow AA \\
1 & \leftrightarrow AB \\
10 & \leftrightarrow BA \\
101 & \leftrightarrow BB \\
\end{array}
\]

\[ \bar{L}_2 = 1.28 \]

\[ \frac{\bar{L}_2}{2} = .645 \]
\[ d = 3 \quad \{ \text{AAA, AAB, ABA, BAA, ABB, BAA, BBA, BBB} \} \]

\[ (p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8) \]

\[ (0.729, 0.031, 0.031, 0.009, 0.009, 0.009, 0.009, 0.009) \]

\[ L_3 = 1 + 0.272 + 0.162 + 0.109 + 0.028 + 0.018 + 0.01 = 1.598 \]

\[ L_{3/3} = 0.533 \]
Huffman Code

\[ \text{n} = 4 \quad P[C] = 0.9 \quad P[e] = 0.1 \]

\[ \frac{L}{4} = 0.49 \]
Another Binary Coding Technique for U.D. Codes

Shannon-Fano Codes - This technique is not necessarily optimum.

1) Order probabilities in decreasing order.
2) Partition into 2 sets that are as close to equally probable as possible.
3) Continue using step 2 over and over. (Label top set with a "0" and bottom set with a "1").

Same example:

a) \( m=1 \)
   \[
   \begin{array}{c}
   \frac{9}{10} - 0 \\
   \frac{1}{10} - 1 \\
   \end{array}
   \]

b) \( m=2 \)
   \[
   \begin{array}{c}
   \frac{81}{99} - 0 \\
   \frac{90}{99} - 1 \\
   \frac{1}{99} - 0 \\
   \end{array}
   \]
   \[ L_2 = 1.81 \times 10^{-3} + 2 \times 0.9 + 3 \times (0.09 + 0.01) = 1.29 \]
   \[ \frac{L_2}{2} = 0.65 \]

C) \( m=3 \)
   \[
   \begin{array}{c}
   \frac{729}{900} - 0 \\
   \frac{81}{900} - 0 \\
   \frac{9}{900} - 0 \\
   \frac{1}{900} - 1 \\
   \frac{1}{900} - 0 \\
   \frac{1}{900} - 0 \\
   \frac{1}{900} - 1 \\
   \frac{1}{900} - 1 \\
   \end{array}
   \]

   \[ L_3 = 1 \times 1.729 + 3 \times (0.081 + 0.09 + 0.01) + 5 \times (0.009 + 0.009 + 0.009 + 0.001) = 1.598 \]

   \[ \frac{L_3}{3} = 0.5327 \]

Same as Huffman in all of these cases.

But it is not the same for \( m=4 \)!! It is worse.
$M = 4$  Shannon - Fano
Binary Huffman Codes

\[ L = 1 \times 5 + 2 \times 2 + 3 \times 15 + 3 \times 15 = \frac{165}{16} \]

\[ L = 1 \times 4 + 2 \times 3 + 3 \times 1 + 5 \times \left( 0.05 + 0.05 + 0.05 + 0.05 \right) \\
= 1.4 + 6 + 3 + 1 = 2.3 \]
\[ L = 1 \times 0.4 + 2 \times 0.3 + 4 \times 0.1 + 5 \times (0.5 + 0.5) + 4 \times (0.5 + 0.5) \]
\[ = 1 \times 0.4 + 6 \times 0.3 + 4 \times 0.5 + 4 \times 0.5 = 2.3 \]

*Has different length distribution but same average length*
Binary Shannon-Fano Codes

1. \[ \begin{align*}
  \text{0.5} & & \text{0} \\
  \text{0.2} & & \text{1} \\
  \text{0.15} & & \text{1} \\
  \text{0.15} & & \text{1} \\
\end{align*} \]

2. \[ \begin{align*}
  \text{0.4} & & \text{0} \\
  \text{0.3} & & \text{1} \\
  \text{0.1} & & \text{1} \\
  \text{0.05} & & \text{1} \\
  \text{0.05} & & \text{1} \\
  \text{0.05} & & \text{1} \\
\end{align*} \]

\[ \overline{L} = 1 \times 0.4 + 2 \times 0.3 + 4 \times 0.1 + 7 \times 0.05 + 4 \times 0.05 + 5 \times 0.05 + 5 \times 0.05 \]

\[ = 0.4 + 0.6 + 0.4 + 0.2 + 0.2 + 0.25 + 0.25 \]

\[ \overline{L} = 2.3 \quad \text{(Same as Huffman code)} \]
Shannon–Fano

\[
\begin{array}{cccc}
2 \times 0.25 & 0 & 0 & \\
2 \times 0.20 & 0 & 1 & \\
3 \times 0.15 & 1 & 0 & 0 \\
7 \times 0.10 & 1 & 0 & 1 \\
18 \times 0.05 & 1 & 0 & 1 & 0 \\
185 \times 0.05 & 1 & 0 & 1 & 1 \\
9 \times 0.05 & 1 & 1 & 0 & \\
5 \times 0.04 & 1 & 1 & 1 & 0 \\
6 \times 0.03 & 1 & 1 & 1 & 1 & 0 \\
6 \times 0.03 & 1 & 1 & 1 & 1 & 1 & \\
\end{array}
\]

\[L = 3.11\]

\[
\begin{align*}
L &= 1 + 0.59 + 0.41 + 0.34 + 0.11 + 0.19 \\
   &= 1 + 0.11 + 0.1 + 0.09 + 0.06 = 3.1
\end{align*}
\]
ENTROPY OF AN I.I.D. SOURCE

Let an I.I.D. source have M source letters that occur with probabilities $p_1, p_2, \ldots, p_M$. Of course $\sum_{i=1}^{M} p_i = 1$.

The entropy of the source $S$ is denoted $H(S)$ and is defined as

$$H_c(S) = \sum_{i=1}^{M} p_i \log_2 \frac{1}{p_i} = -\sum_{i=1}^{M} p_i \log_2 p_i$$

$$H_a(S) = E \left[ \log_a \frac{1}{p_i} \right]$$

The base of the logarithms is usually taken to be equal to 2. In that case $H_a(S)$ is written as $H_2(S)$ and is measured in units of "bits".

Other bases can be used. Since

$$\log_a x = \left(\log_b x\right) \left(\log_a b\right) = \frac{\log_b x}{\log_b a}$$

then

$$H_a(S) = H_b(S) \cdot \log_a b = \frac{H_b(S)}{\log_b a}$$
A USEFUL THEOREM

Let $P_1, P_2, \ldots, P_M$ be one set of probabilities
and let $P'_1, P'_2, \ldots, P'_M$ be another set of
probabilities. (Note $\sum_{i=1}^M P_i = 1$ and $\sum_{i=1}^M P'_i = 1$)

THEOREM

$$\sum_{i=1}^M P_i \log \frac{1}{P_i} \leq \sum_{i=1}^M P'_i \log \frac{1}{P'_i} \quad \text{with equality if } P_i = P'_i$$

for $i = 1, 2, \ldots, M$

Proof

First note that $\ln x \leq x - 1$ with equality if $x = 1$

$$\sum_{i=1}^M P_i \log \frac{P'_i}{P_i} = \left(\sum_{i=1}^M P_i \ln \frac{P'_i}{P_i}\right) \log e$$

$$\leq (\log e) \sum_{i=1}^M P_i \left(\frac{P'_i}{P_i} - 1\right)$$

$$= (\log e) \left(\sum_{i=1}^M P'_i - \sum_{i=1}^M P_i\right) = 0$$

$$\therefore \sum_{i=1}^M P_i \log \frac{1}{P_i} - \sum_{i=1}^M P'_i \log \frac{1}{P'_i} \leq 0$$

$$\sum_{i=1}^M P'_i \log \frac{1}{P'_i} \leq \sum_{i=1}^M P_i \log \frac{1}{P_i}$$
MORE ON Entropy

1. For an equally likely i.i.d. source with $M$ source letters

$$H_2(s) = \log_2 M \quad (\text{or } H_0(s) = \log_2 M, \text{ any } a_i)$$

2. For any i.i.d. source with $M$ source letters

$$0 \leq H_2(s) \leq \log_2 M \quad \text{Follows from previous theorem with } \rho_i = \frac{1}{M} \text{ for all } i$$

3. Consider an i.i.d. source with $M$ source letters, $S$. If we consider encoding $m$ source letters at a time, this is an i.i.d. source with $M^n$ source letters. Call this the $m$th extension of the source and denote it by $S^m$. Then

$$H_2(S^m) = m \cdot H_2(S) \quad (\text{or } H_0(S^m) = m \cdot H_0(S), \text{ any } a_i)$$

The proofs are omitted but are easy.
COMPUTATION OF ENTROPY (base 2)

**Example 2**

\[ M = 2 \quad (p_1, p_2) = (0.9, 0.1) \]

\[ H_2(s) = 0.9 \log_2 0.9 + 0.1 \log_2 0.1 = 0.469 \text{ bits} \]

From before we gave Huffman codes for this source and extensions of this source for which

\[ \overline{L}_1 = 1 \]
\[ \overline{L}_{2/2} = 0.645 \]
\[ \overline{L}_{3/3} = 0.533 \]
\[ \overline{L}_{4/4} = 0.49 \]

Note that \[ \overline{L}_m / m \geq H_2(s) \]

but as \( m \) gets larger \( \overline{L}_m / m \) is getting closer to \( H_2(s) \).

Holds in general.

\[ \Rightarrow \text{One can prove that for a binary Huffman code} \]
\[ H_2(s) \leq \overline{L}_m / m < H_2(s) + \frac{1}{m} \]
**Computation of Entropy (Base 2)**

**Example 2**

\[ M = 3 \quad (p_1, p_2, p_3) = (0.5, 0.35, 0.15) \]

\[ H_2(S) = 0.5 \log_2 0.5 + 0.35 \log_2 0.35 + 0.15 \log_2 0.15 = 1.44 \text{ bits} \]

But from before we gave codes for this source such that:

1 symbol at a time \[ \overline{L}_1 = 1.5 \]

2 symbols at a time \[ \overline{L}_2/2 = 1.46 \]
COMPUTATION OF ENTROPY (BASE 2)

Example 2

M=4 \quad (P_1, P_2, P_3, P_4) = (\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8})

\begin{align*}
H_2(s) &= \frac{1}{2} \log_2 \frac{1}{\frac{1}{2}} + \frac{1}{4} \log_2 \frac{1}{\frac{1}{4}} + \frac{1}{8} \log_2 \frac{1}{\frac{1}{8}} + \frac{1}{8} \log_2 \frac{1}{\frac{1}{8}} \\
&= 1.75 \text{ BITS}
\end{align*}

But from before we gave the code

\begin{align*}
\frac{1}{2} &= 0 \\
\frac{1}{4} &= 10 \\
\frac{1}{8} &= 110 \\
\frac{1}{8} &= 111
\end{align*}

\[ \bar{L}_1 = 1.75 \]

In this special case, \( \bar{L}_1 = H_2(s) \). Thus one cannot improve on the efficiency of the code by encoding several letters at a time.
Significance of Entropy (Base 2)

For any U.D. code corresponding to the $N$-th extension of the IID source $S$, for each $N=1,2,...$

$$\overline{L}_N / N \geq H(S)$$

For a binary Huffman code corresponding to the $N$-th extension of the IID source $S$

$$\frac{\overline{L}_N}{N} \geq H_2(S) \quad \text{and} \quad \frac{\overline{L}_N}{N} < H_2(S) + \frac{1}{N}$$

But this implies

$$\lim_{N \to \infty} \frac{\overline{L}_N}{N} \to H(S)$$
NON-BINARY CODE WORDS

THE CODE SYMBOLS THAT MAKE UP THE CODEWORDS CAN BE FROM A HIGHER ORDER ALPHABET THAN 2.

EXAMPLE

1,1,0 SOURCE \{A, B, C, D, E\}

<table>
<thead>
<tr>
<th>SOURCE SYMBOLS</th>
<th>TERNARY CODE</th>
<th>QUATERNARY CODE</th>
<th>5-LEVEL CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>20</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>21</td>
<td>30</td>
<td>3</td>
</tr>
<tr>
<td>E</td>
<td>22</td>
<td>31</td>
<td>4</td>
</tr>
</tbody>
</table>

Again we are only interested in U.D. codes (where each concatenation of code words can be decoded in only one way).

A lower bound to the average code length of any U.D. code with \(2\)-levels is \(H_2(s)\) where

\[
H_2(s) = \frac{1}{M} \sum_{i=1}^{M} \rho_i \log_2 \frac{1}{\rho_i}
\]

For example, for a TERNARY code, the average length (per source letter), \(\overline{L}/M\), is no less than \(H_3(s)\).
KRAFT INEQUALITY

A necessary and sufficient condition for the construction of an instantaneous code with \( M \) code words of lengths \( l_1, l_2, \ldots, l_M \) where the code symbols take on \( n \) different values is that

\[
\sum_{i=1}^{M} r_i^{-l_i} \leq 1
\]

Proof of Sufficiency. We construct an instantaneous code with these code word lengths. Let there be \( M_j \) code words of length \( j \) for \( j = 1, 2, \ldots, l_* = \max l_i \).

Then\[
\sum_{i=1}^{M_j} r_i^{-l_i} = \sum_{j=1}^{g^*} m_j \cdot r_i^{-l_i}
\]

Assume that \( \sum_{i=1}^{M} r_i^{-l_i} = \sum_{j=1}^{g^*} m_j \cdot r_i^{-l_i} \leq 1 \). Then

\[
\frac{l_*}{\sum_{j=1}^{g^*} m_j} \cdot r_i^{-l_i} \leq r_i^{-x*}
\]

or

\[M_{x*} + M_{x-1} \cdot r + M_{x-2} \cdot r^2 + \ldots + M_1 \cdot r^{l_*-1} \leq r^{x*}
\]

But since \( M_{x*} > 0 \) we then have

\[0 \leq M_{x*} \leq r^{x*} - M_1 \cdot r - M_2 \cdot r^2 - \ldots - M_{x-1} \cdot r^{x-1}
\]
But dividing by $n$ and noting that $M_{x-2} > 0$ we have

$$0 \leq \frac{M_{x-1}}{n} - \frac{M_{x-2}}{n} - \cdots - \frac{M_{x-2}}{n}$$

Continuing we get

$$0 \leq M_3 \leq \frac{\sqrt{3}}{n} - \frac{M_1}{n} - \frac{M_2}{n}$$
$$0 \leq M_2 \leq \frac{\sqrt{2}}{n} - \frac{M_2}{n}$$
$$M_1 \leq \frac{\sqrt{1}}{n}$$

Note that if $\sum_{i=1}^{M} \frac{a_i}{\sqrt{i}} < 1$, then the $a_i$ satisfy the above equations. Note that $M_1 \leq r$. If $M_1 < r$ we have $(r-M_1)$ unused prefixes to form code words of length $2$, for which the code words of length $1$ are not prefixes. But $M_2 \leq \sqrt{2} - M_2$. If $M_2 < \sqrt{2} - M_2$, there are $(\sqrt{2} - M_2, r - M_2)$ code words of length $3$ which satisfy the prefix condition. ETC.

Proof of necessity - Follows from McMillan inequality.
A NECESSARY AND SUFFICIENT CONDITION FOR THE EXISTENCE OF AN U.O.D. CODE WITH M CODE WORDS OF LENGTH $l_1, l_2, \ldots, l_M$ WHERE THE CODE SYMBOLS TAKE ON "$a$" DIFFERENT VALUES IS

$$\sum_{i=1}^{M} x_i^a = 1$$

SKETCH OF PROOF OF NECESSITY

1. Assume $\sum_{i=1}^{M} x_i^a > 1$ and a U.O.D. code exists

2. If $(\sum_{i=1}^{M} x_i^a > 1)$, then $\left[\sum_{i=1}^{M} x_i^{-l_i}\right]^N \geq \varepsilon^N$

3. $(\sum_{i=1}^{M} x_i^{-l_i})^N = \sum_{k=m}^{m_0} N_k x^{-k}$ WHERE $N_k = \# s_k$

4. STRINGS $s$ OF $N$ CODE WORDS THAT HAVE A LENGTH EXACTLY "$k$".

4. If the code is U.O.D. $N_k \leq 2^k$. But then

for a U.O.D. code

$$(\sum_{k=m}^{m_0} N_k x^{-k}) \leq \sum_{k=m}^{m_0} 1 = M q^* - N + 1$$

which grows linearly with $N$, not exponentially with $N$. Q.E.D.
Lower Bound for $I$ for a U.D. code

$\tilde{I} \geq H_n(s)$ where $\tilde{I}$ = average length of U.D. code and $s = \# of symbols in code alphabet. \tilde{I} = H_n(s)$ iff $\rho_i = r^{-l_i}$.

Proof: let $\rho_i' = \frac{r^{-l_i}}{\sum_{j=1}^{M} r^{-l_j}}$. Note $\rho_i' > 0$ and $\sum_{i=1}^{M} \rho_i' = 1$.

From before

$H_n(s) = \sum_{i=1}^{M} \rho_i \log_2 \frac{1}{\rho_i} \leq \sum_{i=1}^{M} \rho_i \log_2 \frac{1}{\rho_i'}$

Then

$H_n(s) \leq \sum_{i=1}^{M} \rho_i l_i + \sum_{i=1}^{M} \rho_i \log_2 \left( \sum_{j=1}^{M} r^{-l_j} \right)$

But for a U.D. code $\sum_{j=1}^{M} r^{-l_j} \leq 1$ so $\log_2 \left( \sum_{j=1}^{M} r^{-l_j} \right) \leq 0$

$\therefore H_n(s) \leq \tilde{I}$

Equality occurs iff $\sum_{j=1}^{M} r^{-l_j} = 1$ and $\rho_i = \rho_i'$

But both of these conditions hold if $\rho_i = r^{-l_i}$, $l_i$ an integer.
HUFFMAN Code AND ENTROPY

A U.D. coding scheme exists for a source with probabilities $P_1, P_2, \ldots, P_N$ with average length $\bar{L}$ and $N$ code symbols such that

$$H_2(S) = \bar{L} < H_2(S) + 1$$

**Proof**

Choose $L_k$ as the unique integer in the range

$$\log_2 \frac{1}{P_k} \leq L_k < \log_2 \frac{1}{P_k} + 1; \text{ i.e., } L_k = \lceil \log_2 \frac{1}{P_k} \rceil$$

Then

$$\sum_{k=1}^{N} L_k - \log_2 \sum_{k=1}^{N} P_k = 1 \text{ is a U.D. code exists with these lengths.}$$

$$\sum_{k=1}^{N} P_k \log_2 \frac{1}{P_k} \leq \sum_{k=1}^{N} P_k L_k < \sum_{k=1}^{N} P_k \log_2 \frac{1}{P_k} + \sum_{k=1}^{N} P_k$$

$$H_2(S) = \bar{L} < H_2(S) + 1 \quad \text{Q.E.D}$$

Then for an a.a.a source $S$, a HUFFMAN code exists with code alphabet $\mathcal{C}$ and $\bar{L}$ for the $N$th extension of this source such that

$$H_2(S^N) < \frac{\bar{L}N}{m} < H_2(S^N) + 1$$

**Proof** From Theorem above, a U.D. code exists for the $N$th extension of the source such that

$$H_2(S^N) \leq \bar{L}N < H_2(S^N) + 1$$

But $H_2(S^N) = mH_2(S)$ and the Huffman code is at least as good as the U.D. code Q.E.D.
**Examples of Codes with** $H_a(\mathcal{S}) \leq \overline{L} < H_a(\mathcal{S}) + 1$

**Example 1**

- $r = 2$
  - $A$: Prob = 0.7, $\log_2 \frac{1}{0.7} = 2.896$, $l_a = \lceil \log_2 \frac{1}{0.7} \rceil = 3$
  - $B$: Prob = 0.09, $\log_2 \frac{1}{0.09} = 4.67$, $l_a = \lceil \log_2 \frac{1}{0.09} \rceil = 5$
  - $C$: Prob = 0.01, $\log_2 \frac{1}{0.01} = 7$, $l_a = \lceil \log_2 \frac{1}{0.01} \rceil = 7$

  $H_a(\mathcal{S}) = 0.516$

  Note that $H_a(\mathcal{S}) \leq \overline{L} < H_a(\mathcal{S}) + 1$

  $0.516 \leq 1.33 < 1.516$

  Better Code (Actually Huffman Code)

  - $A$: 0, $\overline{L} = 1.1$
  - $B$: 10
  - $C$: 11

**Example 2**

- $r = 2$
  - $A$: Prob = 0.19, $\log_2 \frac{1}{0.19} = 3.96$, $l_a = \lceil \log_2 \frac{1}{0.19} \rceil = 4$
  - $B$: Prob = 0.19, $\log_2 \frac{1}{0.19} = 3.96$, $l_a = \lceil \log_2 \frac{1}{0.19} \rceil = 4$
  - $C$: Prob = 0.19, $\log_2 \frac{1}{0.19} = 3.96$, $l_a = \lceil \log_2 \frac{1}{0.19} \rceil = 4$
  - $D$:Prob = 0.19, $\log_2 \frac{1}{0.19} = 3.96$, $l_a = \lceil \log_2 \frac{1}{0.19} \rceil = 4$
  - $E$: Prob = 0.19, $\log_2 \frac{1}{0.19} = 3.96$, $l_a = \lceil \log_2 \frac{1}{0.19} \rceil = 4$
  - $F$: Prob = 0.05, $\log_2 \frac{1}{0.05} = 5.322$, $l_a = \lceil \log_2 \frac{1}{0.05} \rceil = 5$

  $H_a(\mathcal{S}) = 2.492$

  Note that $H_a(\mathcal{S}) \leq \overline{L} < H_a(\mathcal{S}) + 1$

  $2.492 \leq 3.10 < 3.516$

  Huffman Code

  - $A$: 00
  - $B$: 01
  - $C$: 100
  - $D$: 101
  - $E$: 110
  - $F$: 111

  $\overline{L} = 2.62$
Ex 3 \( n = 3 \)

<table>
<thead>
<tr>
<th>Prob</th>
<th>( \log_3 \frac{1}{p} )</th>
<th>( 2^n = \left[ \log_3 \frac{1}{p} \right] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.19</td>
<td>1.51</td>
</tr>
<tr>
<td>B</td>
<td>0.19</td>
<td>1.51</td>
</tr>
<tr>
<td>C</td>
<td>0.19</td>
<td>1.51</td>
</tr>
<tr>
<td>D</td>
<td>0.19</td>
<td>1.51</td>
</tr>
<tr>
<td>E</td>
<td>0.19</td>
<td>1.51</td>
</tr>
<tr>
<td>F</td>
<td>0.05</td>
<td>2.78</td>
</tr>
</tbody>
</table>

\( H_3(3) = 1.57 \)

**Huffman Code**

<table>
<thead>
<tr>
<th>( p )</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>10</td>
</tr>
<tr>
<td>C</td>
<td>11</td>
</tr>
<tr>
<td>D</td>
<td>12</td>
</tr>
<tr>
<td>E</td>
<td>20</td>
</tr>
<tr>
<td>F</td>
<td>21</td>
</tr>
</tbody>
</table>

\( \bar{L} = 1.81 \)
Properties of an optimal (compact) binary code

We only need to consider instantaneous codes!!

1. If \( p_i < p_j \), then \( l_i > l_j \)

Proof: Otherwise switching code words will reduce \( I \)

2. There is no single code word of length \( l_i = \max l_i \)

Proof: If there were shorter it by one digit, it will still not be a prefix of any other code word and will shorten \( I \).

3. Of the code words of length \( l_i \), they occur in pairs in which the code words in each pair agree in all but the last digit.

Proof: If not, shorten the code word \( p_i \) which is not the case by one digit and it will not be the prefix of any other code word. This will shorten \( I \).
Proof of Optimality of Binary Huffman Codes.

\[ P_d = P_{d-1} + P_{d-2} \]

\[ \bar{L}_{d-1} = \bar{L}_d + P_{d-1} + P_{d-2} \] since code \( \mathcal{C}(y-1) \) is same as code \( \mathcal{C}(y) \) except for two words that have length one more.

We now show that if code \( \mathcal{C}_y \) is optimal then code \( \mathcal{C}_{y-1} \) must also be optimal.

*Proof* Suppose there were a better code at \( (y-1) \). Call it's average length \( \bar{L}_{y-1}' < \bar{L}_{y-1} \). But the two code words with probabilities \( P_{d-1} \) and \( P_{d-2} \) are identical in all but the last digit. Form a new code at \( y \) that has the identical prefix as the code word for \( P_{d-1} \). This code will have average length \( \bar{L}_y' = \bar{L}_{y-1}' + (P_{d-1} + P_{d-2}) \) so that \( \bar{L}_{y-1}' < \bar{L}_{y-1} \). But this can't be the case if \( \mathcal{C}_y \) was optimal. Q.E.D.
ECE 154 C
SPRING 2010

LECTURE NOTES # 2 (SOURCE CODING)
Non-Binary Huffman Codes

Sometimes one has to add phantom source symbols with 0 probability in order to make a non-binary Huffman code.

The basic strategy to create a Huffman code where the code words are from an alphabet with 8 letters is to:

(a) Order probabilities high to low (perhaps with extra symbols with probability 0)
(b) Combine at least likely probabilities. Add them and re-order
(c) End up with 8 symbols (i.e., probabilities)

To do this we may need to add phantom symbols.

Example: 2 = 3 \{A, B, C, D\} \{(p_1, p_2, p_3, p_4) = (0.5, 0.8, 0.1, 0.1)\}

Wrong

\[
\begin{array}{c|c|c|c}
0 & 0.5 & 10 & 1 \\
10 & 0.3 & 11 & 11 \\
11 & 0.1 & 12 & 12 \\
\end{array}
\]

\(L_1 = 1.5\)

Right

\[
\begin{array}{c|c|c|c}
0 & 0.5 & 1 & 1 \\
1 & 0.3 & 20 & 20 \\
20 & 0.1 & 31 & 31 \\
\end{array}
\]

\(L_1 = 1.2\)

No U.D. ternary code will have smaller \(L_1\)!!!

Compare with entropy (base 3)

\[H_3(S) = 0.5 \log_3 0.5 + 0.3 \log_3 0.3 + 0.1 \log_3 0.1 + 0.1 \log_3 0.1\]

\[= 1.063\]
If one starts with \( M \) source symbols and one combines the \( r \) least likely into one symbol, one is left with \( M - (r-1) \) symbols.

After doing this \( r \) times, one is left with \( M - r \) symbols.

But at the end we must be left with \( r \) symbols.

Thus we must add \( "D" \) phantom symbols to ensure that \( M + D = \alpha(2r-1) = r \) or \( (M+D) = \alpha'(2r-1) + 1 \).

**Example**: \( r = 3 \)

\[
\begin{array}{cccc}
\frac{M}{3} & 0 & \frac{D}{1} & 0 \\
1 & 1 & 0 & 0 \\
5 & 0 & 1 & 0 \\
6 & 0 & 1 & 0 \\
7 & 0 & 1 & 0 \\
8 & 1 & 0 & 0 \\
9 & 0 & 0 & 0 \\
10 & 1 & 0 & 0 \\
\end{array}
\]

**Example**: \( r = 4 \)

\[
\begin{array}{cccc}
\frac{M}{4} & 0 & \frac{D}{2} & 0 \\
1 & 2 & 1 & 0 \\
5 & 0 & 1 & 0 \\
6 & 0 & 1 & 0 \\
7 & 0 & 1 & 0 \\
8 & 1 & 0 & 0 \\
9 & 0 & 0 & 0 \\
10 & 1 & 0 & 0 \\
\end{array}
\]

**Example**: \( r = 5 \)

\[
\begin{array}{cccc}
\frac{M}{5} & 0 & \frac{D}{3} & 0 \\
1 & 3 & 2 & 2 \\
5 & 0 & 1 & 0 \\
6 & 0 & 1 & 0 \\
7 & 0 & 1 & 0 \\
8 & 1 & 0 & 0 \\
9 & 0 & 0 & 0 \\
10 & 1 & 0 & 0 \\
\end{array}
\]

**Example**: \( r = 6 \)

\[
\begin{array}{cccc}
\frac{M}{6} & 0 & \frac{D}{4} & 0 \\
1 & 4 & 3 & 3 \\
5 & 0 & 2 & 2 \\
6 & 0 & 2 & 2 \\
7 & 0 & 2 & 2 \\
8 & 1 & 0 & 0 \\
9 & 0 & 0 & 0 \\
10 & 1 & 0 & 0 \\
11 & 2 & 1 & 0 \\
12 & 2 & 1 & 0 \\
13 & 3 & 1 & 0 \\
14 & 3 & 1 & 0 \\
15 & 3 & 1 & 0 \\
\end{array}
\]
Run Length Codes for Fax (Black/White)

... www BB www www wBBBa wwwwww .......

...
Encoding of

VARIABLE LENGTH SOURCE SEQUENCES

Previously we only considered the situation where we encoded \( N \) source symbols into variable length code sequences for a fixed value of \( N \).

We could call this "fixed length to variable length" encoding.

But another possibility exists. We could encode variable length source sequences into fixed or code words.

**Example**: \( \{A, B\} \) and source \( (P_A, P_B) = (0.9, 0.1) \)

**Code Book**

<table>
<thead>
<tr>
<th>Source Sequences</th>
<th>Codewords</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>AB</td>
<td>01</td>
</tr>
<tr>
<td>AAB</td>
<td>10</td>
</tr>
<tr>
<td>AAA</td>
<td>11</td>
</tr>
</tbody>
</table>

Average length of source phrase: \( 1 \times 0.1 + 2 \times 0.9 + 3 \times (0.1) + 7 \cdot 0.01 = 2.71 \)

Average \# of code symbols/source symbol: \( \frac{2}{2.71} = 0.739 \)
Tunstall Codes - U.D. Variable to Fixed Encoding - Binary Code Words

Basic Idea - To encode into binary code words of fixed length \( L \), make 2^L source phrases that are as nearly equally probable as we can.

We do this by making the source phrases as leaves of a tree and always splitting the leaf with the highest probability.

Example: \( (A, B, C, D) \) \( (p_1, p_2, p_3, p_4) = (0.5, 0.3, 0.1, 0.1) \)

Source phrases: \{D, C, BB, BC, BD, BAA, BAB, BAC, BAD, AB, AC, AD, AAA, AAB, AAC, AAD\}
<table>
<thead>
<tr>
<th>Source Phrases</th>
<th>Codebook</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>0000</td>
</tr>
<tr>
<td>C</td>
<td>0001</td>
</tr>
<tr>
<td>BB</td>
<td>0010</td>
</tr>
<tr>
<td>BC</td>
<td>0011</td>
</tr>
<tr>
<td>BD</td>
<td>0100</td>
</tr>
<tr>
<td>BAA</td>
<td>0101</td>
</tr>
<tr>
<td>BAB</td>
<td>0110</td>
</tr>
<tr>
<td>BAC</td>
<td>0111</td>
</tr>
<tr>
<td>BAD</td>
<td>1000</td>
</tr>
<tr>
<td>AB</td>
<td>1001</td>
</tr>
<tr>
<td>AC</td>
<td>1010</td>
</tr>
<tr>
<td>AD</td>
<td>1011</td>
</tr>
<tr>
<td>AAA</td>
<td>1100</td>
</tr>
<tr>
<td>AAB</td>
<td>1101</td>
</tr>
<tr>
<td>AAC</td>
<td>1110</td>
</tr>
<tr>
<td>AAD</td>
<td>1111</td>
</tr>
</tbody>
</table>

Average length of source phrase = sum of probabilities of internal nodes

\[
= 1 + .5 + .3 + .25 + .15 = 2.2
\]

Average number of code symbols/source symbol

\[
= 4/2.2 = 1.82
\]
IMPROVED TUNSTALL CODING

Since the phrases are not equally probable, one can use a Huffman code on the phrases.

The result is encoding a variable number of source symbols into a variable number of code symbols.

**Example** \((P_A, P_B) = (.9, .1)\)

![Code Book Diagram]

**Code Book**

<table>
<thead>
<tr>
<th>Source Phrases</th>
<th>Codewords</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>11</td>
</tr>
<tr>
<td>AB</td>
<td>110</td>
</tr>
<tr>
<td>AAA</td>
<td>0</td>
</tr>
<tr>
<td>AAB</td>
<td>111</td>
</tr>
</tbody>
</table>

\[
Au \# of code symbols \\
\frac{469}{532} = 0.881 \approx 88.1\%
\]

\[
\text{Eff} = \frac{469}{2.71} = 1.442 \\
\frac{1.442}{2.71} = .532
\]
Summary of Results for \((A, B) = (0.9, 0.1)\)

All of the following use 4 code words in coding table

1. Huffman Code
   
   \[
   \begin{align*}
   AA & \rightarrow 0 \\
   AB & \rightarrow 11 \\
   BA & \rightarrow 100 \\
   BB & \rightarrow 101
   \end{align*}
   \]
   
   Efficiency \( \geq 72.7\% \)

2. Shannon-Fano Code
   
   Efficiency \( > 72.7\% \)

3. Ternary Code
   
   \[
   \begin{align*}
   B & \rightarrow 00 \\
   AB & \rightarrow 01 \\
   AA & \rightarrow 10 \\
   AAA & \rightarrow 11
   \end{align*}
   \]
   
   ESF \( \geq 63.5\% \)

4. Ternary / Huffman
   
   \[
   \begin{align*}
   B & \rightarrow 11 \\
   AB & \rightarrow 100 \\
   AAA & \rightarrow 0 \\
   AAB & \rightarrow 101
   \end{align*}
   \]
   
   ESF \( = 85.1\% \)
LEMPEL-ZIV SOURCE CODING

The basic idea is that if we have a dictionary of $2^n$ source phrases (available at both the encoder and the decoder) in order to encode one of these phrases one needs only "$n" binary digits.

Normally a computer stores each symbol as an ASCII character of 8 binary digits, (actually only 7 are needed).

Using L-Z encoding, far less than 7 binary digits per symbol are needed. Typically the compression is about 2:1 or 3:1.

There are two versions of L-Z codes. We will only discuss the "window" version. In this version, symbols that have already been encoded are stored in a window. The encoder then looks at the next symbols to be encoded to find the longest string that is in the window that matches the source symbols to be encoded.
If it can't find the next symbol in the window, it sends a "0" followed by the 8 (or 9) bits of the ASCII character.

If it finds a sequence of one or more symbols in the window, it sends a "1" followed by the bit position of the first symbol in the match followed by the length of the match. These latter two quantities are encoded into binary.

Then the sequence that was just encoded is put into the window.

**Example**

```
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
THE - T H R E E - A R E - I N - T H E - C
```

"THE " is encoded as (1, "15", "4")

4 bits 2 bits

And then the window contains

```
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
T H R E E - A R E - I N - T H E - C...
```
186
sum of hrs

<table>
<thead>
<tr>
<th>Hour</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>53</td>
</tr>
</tbody>
</table>

Example of Lernell Ziv's work:

<table>
<thead>
<tr>
<th>Step</th>
<th>Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 = 8 + 1</td>
<td>(h, q)</td>
</tr>
<tr>
<td>5 = 1 + h + 1</td>
<td>(g, r)</td>
</tr>
<tr>
<td>4 = 8 + 1</td>
<td>(g, q)</td>
</tr>
<tr>
<td>3 = 1 + h + 1</td>
<td>(g, r)</td>
</tr>
<tr>
<td>2 = 8 + 1</td>
<td>(g, q)</td>
</tr>
<tr>
<td>1 = 0 + h + 1</td>
<td>(g, r)</td>
</tr>
<tr>
<td>0 = 0 + h + 1</td>
<td>(g, r)</td>
</tr>
</tbody>
</table>

Note: The table continues with similar calculations.
EXAMPLE: ENCODE THE TEXT
"MY_MY_MY_WHAT_A_HAT_IS_THAT"

16 BIT WINDOW
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

MY
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
MY-
Coding with Distortion

\[ \begin{align*}
  \text{INFO SOURCE} & \rightarrow \text{SOURCE ENCODER} & \rightarrow \text{TO NOISY CHANNEL (USING ERROR CORRECTING CODE)} \\
  X(t) & \rightarrow \text{D/A CONVERTER} & \rightarrow \text{SOURCE DECODER} & \rightarrow \text{INFO SINK}
\end{align*} \]

\[ E^2 = \lim_{T \to 0} \frac{1}{T} \int_{-T/2}^{T/2} E(X(t) - \hat{X}(t))^2 \, dt = \text{M.S.E.} \]

If signals are bandlimited, one can sample at Nyquist rate and convert continuous-time problem to discrete-time problem. This sampling is part of the A/D converter.

\[ \begin{align*}
  X_1, X_2, \ldots, X_N & \rightarrow \text{SOURCE ENCODER} & \rightarrow 1011010 \\
  1011010 & \rightarrow \text{SOURCE DECODER} & \rightarrow \frac{1}{m} \sum_{i=1}^{m} (X_i - \hat{X}_i)^2 \\
  E^2 & = \lim_{m \to \infty} \frac{1}{m} \sum_{i=1}^{m} (X_i - \hat{X}_i)^2
\end{align*} \]
A/D Conversion and D/A Conversion

Assume a random variable $X$ which falls into the range $(X_{min}, X_{max})$ to be converted into $k$ binary digits. Let $M = 2^k$. The usual A/D converter first subdivide the interval $(X_{min}, X_{max})$ into $M$ equal sub-intervals of width $\Delta = (X_{max} - X_{min}) / M$ as shown below.

For the case of $k = 3$ and $M = 8$, we call the $i^{th}$ sub-interval, $R_i$.

\[ R_1, R_2, R_3, R_4, R_5, R_6, R_7, R_8 \]

\[ X_{min} \quad | \quad \Delta \quad | \quad \Delta \quad | \quad \Delta \quad | \quad \Delta \quad | \quad \Delta \quad | \quad \Delta \quad | \quad X_{max} \]

Assume that if $X$ falls in the region $R_i$ ($X \in R_i$), then the D/A converter uses as an estimate of $X$, the value $\hat{X}_i$ which is the center of the $i^{th}$ region. Then the mean-squared error between $X$ and $\hat{X}$ is

\[ E = E[(X - \hat{X})^2] = \int (X - \hat{X})^2 f_X(x) \, dx \]

where $f_X(x)$ is the probability density function of the random variable $X$.

Let $f_{X | R_i}(x)$ be the conditional density function of $X$ given that $X$ falls in the region $R_i$. Then

\[ E = \sum_{i=1}^{M} p[X \in R_i] \int (x - \hat{X}_i)^2 f_{X | R_i}(x) \, dx \]

Note that

\[ \sum_{i=1}^{M} p[X \in R_i] = 1 \]

and

\[ \int f_{X | R_i}(x) \, dx = 1 \quad \text{for} \quad i = 1, 2, \ldots, M \]
Now make the further assumption that \( b \) is large enough so that 
\[
\int_{x \in R_i} x \, f(x) \, dx \text{ is a constant over the region } R_i.
\]
Then \( f(x) = \frac{1}{\Delta} \) for all \( i \), and
\[
\int_{x \in R_i} (x-y_i)^2 f(x) \, dx = \frac{1}{\Delta} \int_{a}^{b} (x-(\frac{b-a}{2}))^2 \, dx = \frac{1}{\Delta} \int_{-\frac{\Delta}{2}}^{\frac{\Delta}{2}} (x-0)^2 \, dx
\]
\[
= \frac{1}{\Delta} \frac{2}{3} \left( \frac{\Delta}{2} \right)^3 = \frac{\Delta^2}{12}
\]
Then \( \mathbb{E}^2 = \sum_{i=1}^{M} p[x \in R_i] \cdot \frac{\Delta^2}{12} = \frac{\Delta^2}{12} \).

If \( X \) has variance \( \sigma_x^2 \), the signal-to-noise ratio of the A to D(\& DtoA) converter is often defined as \( \left( \frac{\sigma_x^2}{\Delta^2} \right) \).

If \( X_{\min} \) is equal to \(-\infty\) and/or \( X_{\max} = +\infty \), then the least and least intervals can be infinite in extent. However \( f(x) \) is usually small enough in those intervals so that the result is still approximately the same.
Scalar Quantization of (Gaussian) Samples.

Usual Scalar Quantization (3-Binary Digits/Sample)

**Encoder**

<table>
<thead>
<tr>
<th>$x$</th>
<th>Encoder Code</th>
<th>$a &lt; x &lt; b$</th>
<th>$b &lt; x &lt; 2b$</th>
<th>$2b &lt; x &lt; 3b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$-3b &lt; x$</td>
<td>000</td>
<td>100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$-2b &lt; x &lt; -b$</td>
<td>001</td>
<td>101</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$-b &lt; x &lt; 0$</td>
<td>010</td>
<td>110</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 &lt; $x$ &lt; $b$</td>
<td>011</td>
<td>111</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Decoder**

<table>
<thead>
<tr>
<th>$x$</th>
<th>Decoder Code</th>
<th>$a &lt; x &lt; b$</th>
<th>$b &lt; x &lt; 2b$</th>
<th>$2b &lt; x &lt; 3b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3.5b</td>
<td>000</td>
<td>100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-2.5b</td>
<td>001</td>
<td>101</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-1.5b</td>
<td>010</td>
<td>110</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.5b</td>
<td>011</td>
<td>111</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5b</td>
<td>110</td>
<td>110</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.5b</td>
<td>111</td>
<td>111</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.5b</td>
<td>111</td>
<td>111</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.5b</td>
<td>111</td>
<td>111</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
OPTIMUM SCALAR QUANTIZER

\[ b_{i-1} \leq x < b_i \rightarrow x = a_i, \quad i = 1, 2, \ldots, M \]
\[ b_0 = -\infty, \quad b_M = +\infty \]

\[ a_1, a_2, a_3, \ldots, a_M \]

\[ x \]
\[ b_0 = -\infty, \quad b_1, \quad b_2, \quad b_3, \quad \ldots, \quad b_{M-1}, \quad b_M = +\infty \]

Optimize \{b_i\} and \{a_i\} to minimize \( E^e \)

\[ E^e = \sum_{i=1}^{M} \int_{b_{i-1}}^{b_i} (x - a_i)^2 f_x(x) \, dx \]

\[ \frac{\partial E^e}{\partial a_j} = 0 \]
\[ \frac{\partial E^e}{\partial b_j} = 0 \]

Use Leibnitz's Rule

\[ \frac{\partial}{\partial t} \int_{a(t)}^{b(t)} f(x, t) \, dx = f(b(t), t) \frac{\partial b(t)}{\partial t} \]
\[ -f(a(t), t) \frac{\partial a(t)}{\partial t} \]
\[ + \int_{a(t)}^{b(t)} \frac{\partial}{\partial t} f(x, t) \, dt \]
\[
\frac{\partial}{\partial b_j} \left( \sum_{i=1}^{m} \int_{a_{b_j}}^{b_{j+1}} (x-a_i)^2 f_x(x) \, dx \right) = 0
\]

\[
\frac{\partial}{\partial b_j} \int_{a_{b_j}}^{b_j} (x-a_j)^2 f_x(x) \, dx + \frac{\partial}{\partial b_j} \int_{b_j}^{b_{j+1}} (x-a_{j+1})^2 f_x(x) \, dx = 0
\]

\[
\left. (b_j - a_j)^2 f_x(x) \right|_{x=b_j} - \left. (b_j - a_{j+1})^2 f_x(x) \right|_{x=b_j} = 0
\]

\[
b_j - 2a_j b_j + a_j^2 = b_j - 2b_j a_{j+1} + a_{j+1}^2
\]

\[
2b_j (a_{j+1} - a_j) = a_{j+1}^2 - a_j^2
\]

\[
b_j = \frac{a_{j+1} + a_j}{2} \quad (I)
\]

\[
\frac{\partial}{\partial a_j} \left( \sum_{i=1}^{m} \int_{b_{j-1}}^{b_j} (x-a_i)^2 f_x(x) \, dx \right) = -2 \int_{b_{j-1}}^{b_j} (x-a_j) f_x(x) \, dx = 0
\]

\[
a_j \int_{b_{j-1}}^{b_j} f_x(x) \, dx = \int_{b_{j-1}}^{b_j} f_x(x) \, dx
\]

\[
a_j = \frac{\int_{b_{j-1}}^{b_j} x f_x(x) \, dx}{\int_{b_{j-1}}^{b_j} f_x(x) \, dx} \quad (II)
\]
Note that the $b_{k,j}$ can be found from (I) once the $a_{x,j}$ are known. (The $b_{k,j}$ are the midpoints of the $a_{x,j}$.)

And the $a_{x,j}$ can be solved from (II) once the $b_{k,j}$ are known. (The $a_{x,j}$ are the centroids of the corresponding regions.)

Thus one can use a computer to iteratively solve for the $a_{x,j}$ and the $b_{k,j}$.

1. One starts with an initial guess for the $b_{k,j}$.
2. One uses (II) to solve for the $a_{x,j}$.
3. One uses (I) to solve for the $b_{k,j}$
4. One repeats steps 2 and 3 until the $a_{x,j}$ and the $b_{k,j}$ "stop changing."

Comments

1. This works for any $f_{x|x}$
2. If $f_{x|x}$ only has a finite support one adjusts $b_0$ and $b_m$ to be the limits of the support.
3. For a Gaussian, one needs to know $\int_0^b f_{x|x}(x)dx$ and $\int_0^b x f_{x|x}(x)dx$. (True for any $f_{x|x}$)
\[
\int_a^b \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2} x^2} \, dx = Q(b) - Q(a)
\]

\[
\int x \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2} x^2} \, dx = \ldots \quad \text{(integrate by parts)}
\]

or let \( y = x^2 \)

4. If \( M = 2^n \) one could use "a" binary digits to represent the quantized value. However, since the quantized values are not necessarily equally likely, one could use a Huffman code to use fewer binary digits (on the average).

5. After the \( \{a_i\} \) and \( \{b_i\} \) are known, one computes \( \varepsilon^2 \) from

\[
\varepsilon^2 = \frac{1}{M} \sum_{i=1}^{M} \int_{b_{i-1}}^{b_i} (x-a_i)^2 f_x(x) \, dx
\]

6. For \( M = 2 \) and \( f_x(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2} x^2 / \sigma^2} \) one can easily show that:

\( b_0 = -\infty \), \( b_1 = 0 \), \( b_2 = +\infty \),

\( a_0 = -a_1 = \sqrt{\frac{2\sigma}{\pi}} \)

\[
\varepsilon^2 = \left( 1 - \frac{2}{\pi} \right) \sigma^2 = .3694 \sigma^2
\]
Vector Quantization

One can achieve a smaller \( \varepsilon^2 \) by quantizing several samples at a time.

We would then use regions in a \( m \)-dimensional space.

The rate-distortion formula tells us how small \( \varepsilon^2 \) can be as \( n \to \infty \).

For a Gaussian with one binary digit per sample, \( \varepsilon^2 \geq \frac{\sigma^2}{4} = (0.25)\sigma^2 \)

This follows from the result on the next page.
Discrete-Time Gaussian Source

Let source produce i.i.d. Gaussian samples \( x_1, x_2, \ldots \)

where
\[
 f_X(x) = \frac{1}{\sqrt{2\pi}\sigma^2} e^{-\frac{1}{2} \left\frac{x^2}{\sigma^2} \right}\]

Let source encoder produce a sequence of binary digits at a rate of \( R \) binary digits/source symbol. In our previous terminology \( R = k \).

Let the source decoder produce the sequence \( \hat{x}_1, \hat{x}_2, \ldots, \hat{x}_k \) such that the mean-squared error between \( \{x_n\} \) and \( \{\hat{x}_n\} \) is \( E^2 \).

Then one can prove that for any such system
\[
 R \geq \frac{1}{2} \log_2 \left( \frac{\sigma^2}{E^2} \right) \quad \text{for} \quad E^2 \leq \sigma^2.
\]

\( R = 0 \) for \( E^2 \geq \sigma^2 \)

This is an example of "Rate-Distortion Theory."

Note that for \( R = k = 1 \),
\[
 1 \geq \frac{1}{2} \log_2 \left( \frac{\sigma^2}{E^2} \right) \\
 2 \geq \log_2 \left( \frac{\sigma^2}{E^2} \right) \\
 4 \geq \frac{\sigma^2}{E^2} \quad \text{or} \quad E^2 \geq \frac{\sigma^2}{4} \]
Reduced Fidelity Audio Compression

MP3 players use a form of audio compression called MPEG-1 AudioLayer 3. It takes advantage of a psycho-acoustic phenomenon whereby a loud tone at one frequency "masks" the presence of softer tones at neighboring frequencies. Thus, these softer neighboring tones need not be stored (or transmitted).

Compression efficiency of an audio compression scheme is normally described by the encoded bit rate (prior to the introduction of coding bits.) The CD has a bit rate of \((44.1 \times 10^3 \times 2 \times 16) = 1.41 \times 10^6\) bits/second. The term \(44.1 \times 10^3\) is the sampling rate which is approximately the Nyquist frequency of the audio to be compressed. The term \(2\) comes from the fact that there are two channels in a stereo audio system. The term \(16\) comes from the 16-bit (or \(2^{16} = 65,536\) levels) A-to-D converter. (A slightly higher sampling rate \(48 \times 10^3\) samples/second is used for a DAT recorder.)

Different standards are used in MP3 players. Several bit rates are specified in the MPEG-1 Layer 3 standard. These are 32, 40, 48, 56, 64, 80, 96, 112, 128, 144, 160, 192, 224, 256, and 320 kilobits/second. The sampling rates allowed are 32, 44.1, and 48 kilohertz but the sampling rate of \(44.1 \times 10^3\) Hz is almost always used.
The basic idea behind the scheme is as follows. A block of 576 time domain samples are converted into 576 frequency domain samples using a DFT. The coefficients are then modified using psycho-acoustic principles. The processed coefficients are then converted into a bit stream using various schemes including Huffman encoding. The process is reversed at the receiver: bits $\rightarrow$ frequency-domain coefficients $\rightarrow$ time-domain samples.