Scaling

- Scaling:
  - By scaling we can keep L and R in a reasonable range of values so that \( W = R - L \) does not underflow.
  - The code can be produced progressively, not at the end.
  - Complicates decoding some.

Scaling Principle

Lower half
If \([L, R)\) is contained in \([0, .5)\) then
  - \( L := 2L \)
  - \( R := 2R \)
  - output 0, followed by \( C \)'s
  - \( C := 0 \)

Upper half
If \([L, R)\) is contained in \([.5, 1)\) then
  - \( L := 2L - 1 \)
  - \( R := 2R - 1 \)
  - output 1, followed by \( C \)'s
  - \( C := 0 \)

Middle half
If \([L, R)\) is contained in \([.25, .75)\) then
  - \( L := 2L - .5 \)
  - \( R := 2R - .5 \)
  - \( C := C + 1 \)

Example

- baa

Scale middle half

C = 0
L = 1/3 \( \Rightarrow R = 3/3 \)
L = 3/9 \( \Rightarrow R = 5/9 \)

- baa

C = 1
L = 3/9 \( \Rightarrow R = 5/9 \)
L = 3/18 \( \Rightarrow R = 11/18 \)
Example

- `baa`

Scale lower half

\[
\begin{align*}
C &= 1 \\
L &= \frac{3}{18} \quad R = \frac{11}{18} \\
L &= \frac{9}{54} \quad R = \frac{17}{54}
\end{align*}
\]

Example

- `baa 01`

\[
\begin{align*}
C &= 0 \\
L &= \frac{9}{54} \quad R = \frac{17}{54} \\
L &= \frac{18}{54} \quad R = \frac{34}{54}
\end{align*}
\]

Example

- `baa 01`

In end \(L < \frac{1}{2} < R\), choose tag to be \(\frac{1}{2}\)

\[
\begin{align*}
C &= 0 \\
L &= \frac{9}{54} \quad R = \frac{17}{54} \\
L &= \frac{18}{54} \quad R = \frac{34}{54}
\end{align*}
\]

Example with Scaling

- `acc`

\[
\begin{align*}
\text{Code} &= 0101 \\
\text{Equally Likely model}
\end{align*}
\]

Integer Implementation

- \(m\) bit integers
  - Represent 0 with 000...0 (\(m\) times)
  - Represent 1 with 111...1 (\(m\) times)
- Probabilities represented by frequencies
  - \(n_i\) is the number of times that symbol \(a\) occurs
  - \(C_i = n_1 + n_2 + \ldots + n_i\)
  - \(N = n_1 + n_2 + \ldots + n_m\)
  - \(W = R - L + 1\)

Context

- Consider 1 symbol context.
- Example: 3 contexts.
Arithmetic Coding with Context

• Maintain the probabilities for each context.
• For the first symbol use the equal probability model
• For each successive symbol use the model for the previous symbol.

Adaptation

• Simple solution – Equally Probable Model.
  – Initially all symbols have frequency 1.
  – After symbol x is coded, increment its frequency by 1
  – Use the new model for coding the next symbol
• Example in alphabet a,b,c,d

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Frequency</th>
<th>After aabaac is encoded</th>
<th>The probability model</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>2</td>
<td>a 5/10</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>2</td>
<td>b 2/10</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td>2</td>
<td>c 2/10</td>
</tr>
<tr>
<td>d</td>
<td>1</td>
<td>2</td>
<td>d 1/10</td>
</tr>
</tbody>
</table>

Zero Frequency Problem

• How do we weight symbols that have not occurred yet.
  – Equal weights? Not so good with many symbols
  – Escape symbol, but what should its weight be?
  – When a new symbol is encountered send the <esc>, followed by the symbol in the equally probable model. (Both encoded arithmetically.)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Frequency</th>
<th>After aabaac is encoded</th>
<th>The probability model</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td>2</td>
<td>a 4/7</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>2</td>
<td>b 1/7</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td>2</td>
<td>c 1/7</td>
</tr>
<tr>
<td>d</td>
<td>1</td>
<td>2</td>
<td>d 0</td>
</tr>
<tr>
<td>&lt;esc&gt;</td>
<td>1</td>
<td>2</td>
<td>&lt;esc&gt; 1/7</td>
</tr>
</tbody>
</table>

PPM

• Prediction with Partial Matching
  – Cleary and Witten (1984)
• State of the art arithmetic coder
  – Arbitrary order context
  – The context chosen is one that does a good prediction given the past
  – Adaptive
• Example
  – Context “the” does not predict the next symbol “a” well. Move to the context “he” which does.

Arithmetic vs. Huffman

• Both compress very well. For m symbol grouping,
  – Huffman is within 1/m of entropy.
  – Arithmetic is within 2/m of entropy.
• Context
  – Huffman needs a tree for every context.
  – Arithmetic needs a small table of frequencies for every context.
• Adaptation
  – Huffman has an elaborate adaptive algorithm
  – Arithmetic has a simple adaptive mechanism.
• Bottom Line – Arithmetic is more flexible than Huffman.