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 $C$ 1's
    - $C := 0$

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

- Middle Half
  - If $[L, R)$ is contained in $[.25, .75)$ then
    - $L := 2L - .5$; $R := 2R - .5$
    - $C := C + 1$

Example

- $baa$

  + $L = 1/3$; $R = 3/3$
  + $C = 0$

+ $L = 3/9$; $R = 11/18$
  + $C = 1$

  Scale middle half
Example

- baa

\[ C = 1 \]
\[ L = \frac{3}{18} R = \frac{11}{18} \]
\[ L = \frac{9}{54} R = \frac{17}{54} \]

Scale lower half

Example

- baa 01

\[ C = 0 \]
\[ L = \frac{9}{54} R = \frac{17}{54} \]
\[ L = \frac{18}{54} R = \frac{34}{54} \]

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

Example

- baa 011

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

\[ C = 0 \]
\[ L = \frac{9}{54} R = \frac{17}{54} \]
\[ L = \frac{18}{54} R = \frac{34}{54} \]

Context

- Consider 1 symbol context.
  - Example: 3 contexts.

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 \( a_i \) occurs
  - \( C_i = n_1 + n_2 + ... + n_{i-1} \)
  - \( N = n_1 + n_2 + ... + n_m \)

Coding the \( i \)-th symbol using integer calculations. Must use scaling!

Example with Scaling

- acc

Code = 0101

Equally Likely model
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>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>b</td>
<td>1 1 2 2 2</td>
</tr>
<tr>
<td>c</td>
<td>1 1 1 1 1</td>
</tr>
<tr>
<td>d</td>
<td>1 1 1 1 1</td>
</tr>
</tbody>
</table>

  After aabaac is encoded
  
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>4 5</td>
</tr>
<tr>
<td>b</td>
<td>2</td>
</tr>
<tr>
<td>c</td>
<td>2</td>
</tr>
<tr>
<td>d</td>
<td>1</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.)
    
    | Symbol | Frequency |
    |--------|-----------|
    | a      | 0 1 2 3 4 |
    | b      | 0 0 1 1 1 |
    | c      | 0 0 0 0 0 |
    | d      | 0 0 0 0 0 |
    | <esc>  | 1 1 1 1 1 |

  After aabaac is encoded
  
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>3/7</td>
</tr>
<tr>
<td>b</td>
<td>1/7</td>
</tr>
<tr>
<td>c</td>
<td>1/7</td>
</tr>
<tr>
<td>d</td>
<td>0</td>
</tr>
<tr>
<td>&lt;esc&gt;</td>
<td>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.