Run-Length Coding

- Lots of 0’s and not too many 1’s.
  - Fax of letters
  - Graphics
- Simple run-length code
  - 000001000000000000000000001001001…..
  - 6 9 10 3 2 …
  - Code the bits as a sequence of integers

Golomb Code of Order m

- Let \( n = qm + r \) where \( 0 < r < m \).
  - Divide \( m \) into \( n \) to get the quotient \( q \) and remainder \( r \).
- Code for \( n \) has two parts:
  1. \( q \) is coded in unary
  2. \( r \) is coded as a fixed prefix code

Example:

```
\[ n = qm + r \] is represented by:
\[ \overline{9} \quad \overline{11} \overline{10} r \]
```

- \( \overline{r} \) is the fixed prefix code for \( r \)
- Example:
  2 6 9 10 27
  0 1 0 1 1 1 1 1 0 0 0 1
  0 1 0 1 1 1 1 1 0 0 0 1

Alternative Explanation Golomb Code

Run Length Example: \( m = 5 \)

```
input    output
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 1 0 1 0 1 1 1 1 1 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1
```

In this example we coded 17 bit in only 9 bits.
Choosing $m$

- Suppose that 0 has the probability $p$ and 1 has probability $1-p$.
- The probability of $0^n1$ is $p^n(1-p)$. The Golomb code of order $m = \lceil -\frac{1}{\log_2 p} \rceil$ is optimal.
- Example: $p = 127/128$.

\[
m = \lceil -\frac{1}{\log_2 (127/128)} \rceil = 89
\]

Average Bit Rate for Golomb Code

Average Bit Rate = \frac{\text{Average output code length}}{\text{Average input code length}}

- $m = 4$ as an example. With $p$ as the probability of 0.

\[
\text{ABR} = 4p^4 + 4p^3(1-p) + 3p^2(1-p) + 2p(1-p) + (1-p)
\]

Comparison of GC with Entropy

![Graph comparing GC with entropy](image)

Notes on Golomb codes

- Useful for binary compression when one symbol is much more likely than another.
  - binary images
  - fax documents
  - bit planes for wavelet image compression
- Need a parameter (the order)
  - training
  - adaptively learn the right parameter
- Variable-to-variable length code
- Last symbol needs to be a 1
  - coder always adds a 1
  - decoder always removes a 1

Tunstall Codes

- Variable-to-fixed length code
- Example

<table>
<thead>
<tr>
<th>input</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>000</td>
</tr>
<tr>
<td>b</td>
<td>001</td>
</tr>
<tr>
<td>ca</td>
<td>010</td>
</tr>
<tr>
<td>cb</td>
<td>011</td>
</tr>
<tr>
<td>cca</td>
<td>100</td>
</tr>
<tr>
<td>ccb</td>
<td>101</td>
</tr>
<tr>
<td>ccc</td>
<td>110</td>
</tr>
</tbody>
</table>

Tunstall code Properties

1. No input code is a prefix of another to assure unique encodability.
2. Minimize the number of bits per symbol.
Prefix Code Property

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>000</td>
</tr>
<tr>
<td>b</td>
<td>001</td>
</tr>
<tr>
<td>ca</td>
<td>010</td>
</tr>
<tr>
<td>db</td>
<td>011</td>
</tr>
<tr>
<td>rca</td>
<td>100</td>
</tr>
<tr>
<td>tcb</td>
<td>101</td>
</tr>
<tr>
<td>rcc</td>
<td>110</td>
</tr>
</tbody>
</table>

Unused output code is 111.

Use for unused code

- Consider the string “cc”. It does not have a code.
- Send the unused code and some fixed code for the cc.
- Generally, if there are k internal nodes in the prefix tree then there is a need for k-1 fixed codes.

Designing a Tunstall Code

- Suppose there are m initial symbols.
- Choose a target output length n where $2^n > m$.

1. Form a tree with a root and m children with edges labeled with the symbols.
2. If the number of leaves is $> 2^n - m$ then halt.*
3. Find the leaf with highest probability and expand it to have m children.** Go to 2.

* In the next step we will add m-1 more leaves.
** The probability is the product of the probabilities of the symbols on the root to leaf path.

Example

- $P(a) = 0.7$, $P(b) = 0.2$, $P(c) = 0.1$
- $n = 3$

Example

- $P(a) = 0.7$, $P(b) = 0.2$, $P(c) = 0.1$
- $n = 3$
**Bit Rate of Tunstall**

- The length of the output code divided by the average length of the input code.
- Let $p_i$ be the probability of and $r_i$ the length of input code $i$ ($1 \leq i \leq s$) and let $n$ be the length of the output code.

$$\text{Average bit rate} = \frac{n}{\sum_{i=1}^{s} p_i r_i}$$

**Example**

- Huffman

```
  \begin{array}{ccc}
    a & b & c \\
    .343 & .098 & .049 \\
  \end{array}
  \begin{array}{ccc}
    a & b & c \\
    .14 & .07 & .1 \\
  \end{array}
```

- ABR: $3(0.343 + 0.098 + 0.049) + 2(0.14 + 0.07) + 0.2 + 0.1 = 1.37$ bits per symbol
- Entropy: $1.16$ bits per symbol

**Notes on Tunstall Codes**

- Variable-to-fixed length code
- Error resilient
  - A flipped bit will introduce just one error in the output
  - Huffman is not error resilient. A single bit flip can destroy the code.