# CSE 490 GZ
## Introduction to Data Compression
Winter 2004

Golomb Codes

Tunstall Codes

---

## Run-Length Coding

- Lots of 0's and not too many 1's.
  - Fax of letters
  - Graphics
- Simple run-length code
  - Input
    000000100000000010000000000010001001.....
  - Symbols
    6 9 10 3 2 ...
  - Code the bits as a sequence of integers
  - Problem: How long should the integers be?

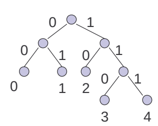


---

## Golomb Code of Order m
## Variable Length Code for Integers

- Let n = qm + r where $0 \le r < m$.
  - 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: m = 5




code for r



---

## Example

- n = qm + r is represented by:

$$\overbrace{11\cdots1}^{q}0\hat{r}$$

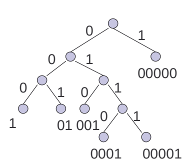
  - where $\hat{r}$ is the fixed prefix code for r
- Example (m = 5):

| 2 | 6 | 9 | 10 | 27 |
|---|---|---|---|---|
| 010 | 1001 | 10111 | 11000 | 11111010 |



---

## Alternative Explanation
## Golomb Code of order 5




| input | output |
|---|---|
| 00000 | 1 |
| 00001 | 0111 |
| 0001 | 0110 |
| 001 | 010 |
| 01 | 001 |
| 1 | 000 |

Variable length to variable length code.



---

## Run Length Example: m = 5

$\boxed{00000}$0100000000010000000000010001001.....
1
00000$\boxed{01}$00000000010000000000010001001.....
001
0000001$\boxed{00000}$00001000000000010001001.....
1
000000100000$\boxed{00001}$000000000010001001.....
0111

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

$$m = \left\lceil {-1}/{\log_2 p} \right\rceil$$

is optimal.
- Example: p = 127/128.

$$m = \left\lceil {-1}/{\log_2 (127/128)} \right\rceil = 89$$



## Average Bit Rate for Golomb Code

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

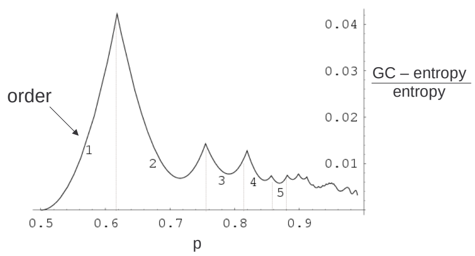
- m = 4 as an example. With p as the probability of 0.

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

| | | | | | |
|---|---|---|---|---|---|
| ouput | 1 | 011 | 010 | 001 | 000 |
| input | 0000 | 0001 | 001 | 01 | 1 |
| weight | $p^4$ | $p^3$(1-p) | $p^2$(1-p) | p(1-p) | 1-p |



## Comparison of GC with Entropy






## 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

| input | output |
|---|---|
| a | 000 |
| b | 001 |
| ca | 010 |
| cb | 011 |
| cca | 100 |
| ccb | 101 |
| ccc | 110 |

a b cca cb ccc ...
000 001 110 011 110 ...

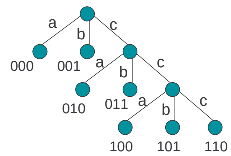


## Tunstall code Properties

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

## Prefix Code Property

| | |
|---|---|
| a | 000 |
| b | 001 |
| ca | 010 |
| cb | 011 |
| cca | 100 |
| ccb | 101 |
| ccc | 110 |




Unused output code is 111.



## Use for unused code

- Consider the string “cc”, if it occurs at the end of the data. 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 $2^n > m$.

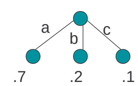
1. Form a tree with a root and m children with edges labeled with the symbols.
2. If the number of leaves is $> 2^n - m$ then halt.*
3. Find the leaf with highest probability and expand it to have m children.** Go to 2.

* 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

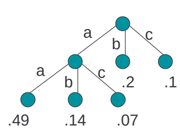
- P(a) = .7, P(b) = .2, P(c) = .1
- n = 3






## Example

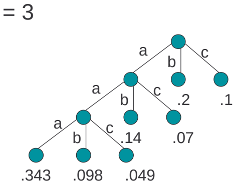
- P(a) = .7, P(b) = .2, P(c) = .1
- n = 3






## Example

- P(a) = .7, P(b) = .2, P(c) = .1
- n = 3




| | |
|---|---|
| aaa | 000 |
| aab | 001 |
| aac | 010 |
| ab | 011 |
| ac | 100 |
| b | 101 |
| c | 110 |

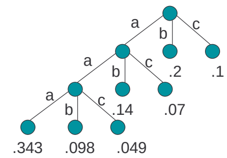
# Bit Rate of Tunstall

- 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




| | | |
|---|---|---|
| aaa | .343 | 000 |
| aab | .098 | 001 |
| aac | .049 | 010 |
| ab | .14 | 011 |
| ac | .07 | 100 |
| b | .2 | 101 |
| c | .1 | 110 |

ABR = 3/[3 (.343 + .098 + .049) + 2 (.14 + .07) + .2 + .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.