CSE 490 GZ
Introduction to Data Compression
Winter 2004

Course Policies
Introduction to Data Compression
Entropy
Prefix Codes

Instructors

- Instructor
  - Richard Ladner
  - ladner@cs.washington.edu
  - 206 543-9347
  - office hours: Tue and Thurs 11 to noon
- TA
  - Neva Cherniavsky
  - nchernia@cs.washington.edu
  - office hours - Mon and Wed at 9:30 to 10:30

Prerequisites

- CSE 142, 143
- CSE 326 or CSE 373
- Reason for the prerequisites:
  - Data compression has many algorithms
  - Some of the algorithms require complex data structures

Resources

- Text Book
  - Khalid Sayood, Introduction to Data Compression, Second Edition, Morgan Kaufmann Publishers, 2000.
- 490gz Course Web Page
  - http://www.cs.washington.edu/490gz/
- Papers and Sections from Books
- E-mail list
  - For dissemination of information by instructor and TA
  - Please sign up

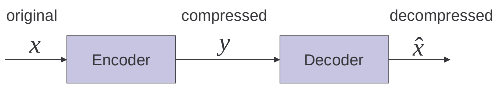
Engagement by Students

- Weekly Assignments
  - Understand compression methodology
  - Due in class on Fridays (except midterm Friday)
  - No late assignments accepted except with prior approval
- Programming Projects
  - Bi-level arithmetic coder and decoder.
  - Image coder and decoder.
  - Build code and experiment

Final Exam and Grading

- Final Exam - 8:30-10:20 a.m. Tuesday, March 16, 2004
- Midterm Exam – Friday, February 6, 2004
- Percentages
  - Weekly assignments (25%)
  - Midterm exam (15%)
  - Projects (25%)
  - Final exam (35%)

Basic Data Compression Concepts



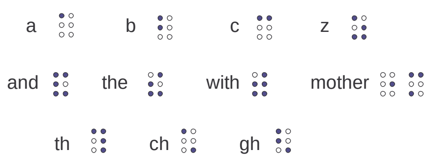
- Lossless compression $x = \hat{x}$
  - Also called entropy coding, reversible coding.
- Lossy compression $x \neq \hat{x}$
  - Also called irreversible coding.
- Compression ratio = $|x|/|y|$
  - $|x|$ is number of bits in x.

Why Compress

- Conserve storage space
- Reduce time for transmission
  - Faster to encode, send, then decode than to send the original
- Progressive transmission
  - Some compression techniques allow us to send the most important bits first so we can get a low resolution version of some data before getting the high fidelity version
- Reduce computation
  - Use less data to achieve an approximate answer

Braille

- System to read text by feeling raised dots on paper (or on electronic displays). Invented in 1820s by Louis Braille, a French blind man.



Braille Example

Clear text:

Call me Ishmael. Some years ago -- never mind how long precisely -- having \\ little or no money in my purse, and nothing particular to interest me on shore, \\ I thought I would sail about a little and see the watery part of the world. (238 characters)

Grade 2 Braille in ASCII.

,call me ,i\%mael4 ,``s ye$>$s ago -- n``e m9d h[l;g precisely -- hav+ \\ ll or no m``oy 9 my purse1 \& no?+ ``picul$>$ 6 9t]e/ me on \%ore1 \\ ,i $?$``$|$ ,i wd sail ab a ll \& see ! wat]y ``p (! _w4 (203 characters)

Compression ratio = 238/203 = 1.17

Lossless Compression

- Data is not lost - the original is really needed.
  - text compression
  - compression of computer binary files
- Compression ratio typically no better than 4:1 for lossless compression on many kinds of files.
- Statistical Techniques
  - Huffman coding
  - Arithmetic coding
  - Golomb coding
- Dictionary techniques
  - LZW, LZ77
  - Sequitur
  - Burrows-Wheeler Method
- Standards - Morse code, Braille, Unix compress, gzip, zip, bzip, GIF, JBIG, Lossless JPEG

Lossy Compression

- Data is lost, but not too much.
  - audio
  - video
  - still images, medical images, photographs
- Compression ratios of 10:1 often yield quite high fidelity results.
- Major techniques include
  - Vector Quantization
  - Wavelets
  - Block transforms
  - Standards - JPEG, MPEG

Why is Data Compression Possible

- Most data from nature has redundancy
  - There is more data than the actual information contained in the data.
  - Squeezing out the excess data amounts to compression.
  - However, unsqueezing is necessary to be able to figure out what the data means.
- Information theory is needed to understand the limits of compression and give clues on how to compress well.

What is Information

- Analog data
  - Also called continuous data
  - Represented by real numbers (or complex numbers)
- Digital data
  - Finite set of symbols $\{a_1, a_2, \ldots, a_m\}$
  - All data represented as sequences (strings) in the symbol set.
  - Example: {a,b,c,d,r} abracadabra
  - Digital data can be an approximation to analog data

Symbols

- Roman alphabet plus punctuation
- ASCII - 256 symbols
- Binary - {0,1}
  - 0 and 1 are called bits
  - All digital information can be represented efficiently in binary
  - {a,b,c,d} fixed length representation

symbol	a	b	c	d
binary	00	01	10	11

  - 2 bits per symbol

Exercise - How Many Bits Per Symbol?

- Suppose we have n symbols. How many bits (as a function of n) are needed in to represent a symbol in binary?
  - First try n a power of 2.

Discussion: Non-Powers of Two

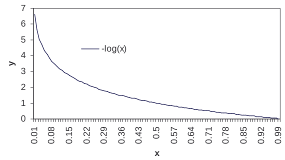
- Can we do better than a fixed length representation for non-powers of two?

Information Theory

- Developed by Shannon in the 1940's and 50's
- Attempts to explain the limits of communication using probability theory.
- Example: Suppose English text is being sent
  - It is much more likely to receive an "e" than a "z".
  - In some sense "z" has more information than "e".

First-order Information

- Suppose we are given symbols $\{a_1, a_2, \ldots, a_m\}$.
- $P(a_i)$ = probability of symbol a_i occurring in the absence of any other information.
  - $P(a_1) + P(a_2) + \ldots + P(a_m) = 1$
- $inf(a_i) = \log_2(1/P(a_i))$ bits is the information of a_i in bits.



Example

- {a, b, c} with P(a) = 1/8, P(b) = 1/4, P(c) = 5/8
  - $inf(a) = \log_2(8) = 3$
  - $inf(b) = \log_2(4) = 2$
  - $inf(c) = \log_2(8/5) = .678$
- Receiving an “a” has more information than receiving a “b” or “c”.

First Order Entropy

- The first order entropy is defined for a probability distribution over symbols $\{a_1, a_2, \ldots, a_m\}$.

$$H = \sum_{i=1}^{m} P(a_i)\log_2\left(\frac{1}{P(a_i)}\right)$$

- H is the average number of bits required to code up a symbol, given all we know is the probability distribution of the symbols.
- H is the Shannon lower bound on the average number of bits to code a symbol in this “source model”.
- Stronger models of entropy include context.

Entropy Examples

- {a, b, c} with a 1/8, b 1/4, c 5/8.
  - H = 1/8 *3 + 1/4 *2 + 5/8* .678 = 1.3 bits/symbol
- {a, b, c} with a 1/3, b 1/3, c 1/3. (worst case)
  - $H = 3^* (1/3)^*\log_2(3) = 1.6$ bits/symbol
- Note that a standard code takes 2 bits per symbol

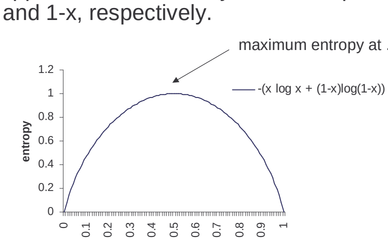
symbol	a	b	c
binary code	00	01	10

An Extreme Case

- {a, b, c} with a 1, b 0, c 0
  - H = ?

Entropy Curve

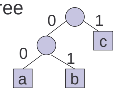
- Suppose we have two symbols with probabilities x and 1-x, respectively.



A Simple Prefix Code

- {a, b, c} with a 1/8, b 1/4, c 5/8.
- A prefix code is defined by a binary tree
- Prefix code property
  - no output is a prefix of another

binary tree

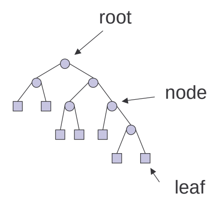


code

input	output
a	00
b	01
c	1

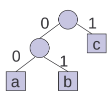
ccabccbccc
1 1 00 01 1 1 01 1 1 1

Binary Tree Terminology



1. Each node, except the root, has a unique parent.
2. Each internal node has exactly two children.

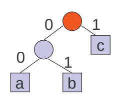
Decoding a Prefix Code




```
repeat
start at root of tree
  repeat
    if read bit = 1 then go right
    else go left
  until node is a leaf
  report leaf
until end of the code
```

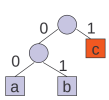
11000111100

Decoding a Prefix Code



11000111100

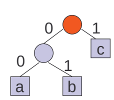
Decoding a Prefix Code



11000111100

c

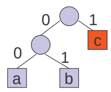
Decoding a Prefix Code



11000111100

c

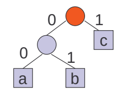
Decoding a Prefix Code



11000111100

cc

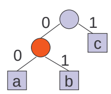
Decoding a Prefix Code



11000111100

cc

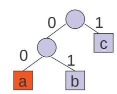
Decoding a Prefix Code



11000111100

cc

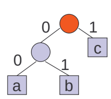
Decoding a Prefix Code



11000111100

cca

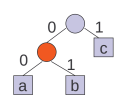
Decoding a Prefix Code



11000111100

cca

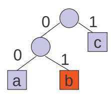
Decoding a Prefix Code



11000111100

cca

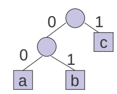
Decoding a Prefix Code



11000111100

ccab

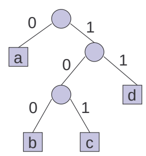
Decoding a Prefix Code



11000111100

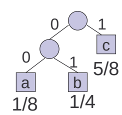
ccabccca

Exercise Encode/Decode



- Player 1: Encode a symbol string
- Player 2: Decode the string
- Check for equality

How Good is the Code



bit rate = (1/8)2 + (1/4)2 + (5/8)1 = 11/8 = 1.375 bps
Entropy = 1.3 bps
Standard code = 2 bps

(bps = bits per symbol)

Design a Prefix Code 1

- abracadabra
- Design a prefix code for the 5 symbols {a,b,r,c,d} which compresses this string the most.

Design a Prefix Code 2

- Suppose we have n symbols each with probability 1/n. Design a prefix code with minimum average bit rate.
- Consider n = 2,3,4,5,6 first.