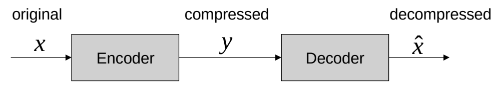
CSE 589
Applied Algorithms
Spring 1999

Data Compression
Information Theory

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

Lossless Compression

- Data is not lost - the original is really needed.
  - text compression
  - compression of computer binaries to fit on a floppy
- Compression ratio typically no better than 4:1 for lossless compression.
- Major techniques include
  - Huffman coding
  - Arithmetic coding
  - Dictionary techniques (Ziv,Lempel 1977,1978)
  - Sequitur (Nevill-Manning, Witten 1996)
  - Standards - Morse code, Braille, Unix compress, gzip, zip, GIF, JBIG, JPEG

Lossy Compression

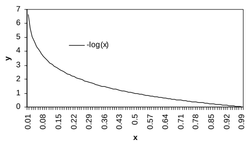
- Data is lost, but not too much.
  - audio
  - video
  - still images, medical images, photographs
- Compression ratios of 10:1 often yield quite high fidelity results.
- Major techniques include
  - Vector Quantization
  - Wavelets
  - Transforms
  - Standards - JPEG, MPEG

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
  - Suppose a "t" is received. Given English, the next symbol being a "z" has very low probability, the next symbol being a "h" has much higher probability. Receiving a "z" has much more information in it than receiving a "h". We already knew it was more likely we would receive an "h".

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.
  - $P(a_1) + P(a_2) + \ldots + P(a_m) = 1$
- $\inf(a_i) = -\log_2 P(a_i)$ bits is the information in bits of a_i.



Example

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

Entropy

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

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

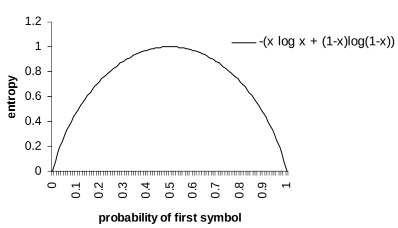
- H is the average number of bits required to code up a symbol, given all we know is the probability distribution of the symbols.
- H is the Shannon lower bound on the average number of bits to code a symbol in this source model.
- Stronger models of entropy include context.

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)
  - $H = -3^* (1/3)^*\log_2(1/3) = 1.6$ bits/symbol
- {a, b, c} with a 1, b 0, c 0 (best case)
  - $H = -1^*\log_2(1) = 0$
- Note that the standard coding of 3 symbols takes 2 bits.

Entropy Curve

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



First-Order Entropy of a String

- Suppose we are given a string $x_1x_2\ldots x_n$ in an alphabet $\{a_1,a_2,\ldots,a_m\}$ where $P(a_i)$ is the probability of symbol i.
- The first-order entropy of $x_1x_2\ldots x_n$ is

$$H(x_1x_2\cdots x_n) = \sum_{i=1}^{n} P(x_i)\inf(x_i) = -\sum_{i=1}^{n} P(x_i)\log_2(P(x_i))$$

- $H(x_1x_2\ldots x_n)$ is a lower bound on the number of bits to code the string $x_1x_2\ldots x_n$ given only the probabilities of the symbols. This is the Shannon lower bound.

Shannon Lower Bound

- Suppose we are given an algorithm that compresses a string x of length n and the algorithm only uses the frequencies of the symbols $\{a_1,a_2,...,a_m\}$ in the string as input.
- Let c(x) be the compressed result represented in bit.

$$|c(x)| \geq H(x) = nH$$

where $H = -\sum_{i=1}^{m} \frac{n_i}{n} \log_2(\frac{n_i}{n})$ and n_i is the frequency of a_i.

Example 1

- x =1 1 1 1 1 0 1 1 1 1 0 1
  - P(0) = 2/12 (from frequencies)
  - P(1) = 10/12 (from frequencies)
- H = -((2/12) $\log_2$(2/12) + (10/12) $\log_2$(10/12))= .65
- Lower bound of 12 x .65 = 7.8 bits
- Standard for a two symbol alphabet is 1 bits per symbol or 12 bits.
- There is a potential gain in some algorithm.

Example 2

- x =1 2 3 4 5 4 5 6 7 8 7 8
  - P(1) = P(2) = P(3) = P(6) = 1/12 (from frequencies)
  - P(4) = P(5) = P(7) = P(8) = 2/12 (from frequencies)
- H = -((4/12) $\log_2$(1/12) + (8/12) $\log_2$(2/12))= 2.92
- Lower bound of 12 x 2.92 = 35.02 bits
- Standard for an 8 symbol alphabet is 3 bits per symbol or 36 bits.
- No compression algorithm will give us much.

Example 2 with Context

- x =1 2 3 4 5 4 5 6 7 8 7 8
- define $x_{k+1} = x_k + r_k$
- r = 1 1 1 1 -1 1 1 1 1 -1 1 (residual)
- Compression Algorithm
  - represent x as x_1, r_1, r_2, ..., r_{11}
  - Compress this sequence.
  - 3 bits for x_1 and less than 11 bits for the rest, for less than 14 bits instead of 35.02 bits.
- This algorithm does not use just the frequencies of the symbols, but uses correlation between adjacent symbols.

Huffman Coding

- Huffman (1951)
- Uses frequencies of symbols in a string to build a variable rate prefix code.
  - Each symbol is mapped to a binary string.
  - More frequent symbols have shorter codes.
  - No code is a prefix of another.
- Example: a 0
b 100
c 101
d 11

Variable Rate Code Example

- Example: a 0, b 100, c 101, d 11
- Coding:
  - aabddcaa = 16 bits
  - 0 0 100 11 11 101 0 0= 14 bits
- Prefix code ensures unique decodability.
  - 00100111110100
  - a a b d d c a a
- Morse Code an example of variable rate code. E = . and Z = _ _ . .