Lossless Compression

Multimedia Systems (Module 2)

- Lesson 1:
  - Minimum Redundancy Coding based on Information Theory:
    - Shannon-Fano Coding
    - Huffman Coding
- Lesson 2:
  - Adaptive Coding based on Statistical Modeling:
    - Adaptive Huffman
    - Arithmetic coding
- Lesson 3:
  - Dictionary-based Coding
    - LZW

Lossless Compression

Multimedia Systems (Module 2 Lesson 1)

Summary:

- Compression
  - With loss
  - Without loss
- Shannon: Information Theory
- Shannon-Fano Coding Algorithm
- Huffman Coding Algorithm

Sources:

- *The Data Compression Book,* 2nd Ed., Mark Nelson and Jean-Loup Gailly.
- Dr. Ze-Nian Li's course material

Compression

Why Compression?

All media, be it text, audio, graphics or video has "redundancy".

Compression attempts to eliminate this redundancy.

What is Redundancy?

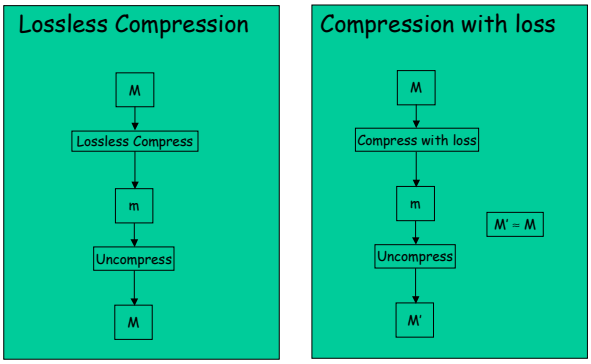
- If one representation of a media content, M, takes X bytes and another takes Y bytes(Y< X), then X is a redundant representation relative to Y.
- Other forms of Redundancy

  If the representation of a media captures content that is not perceivable by humans, then removing such content will not affect the quality of the content.
  - For example, capturing audio frequencies outside the human hearing range can be avoided without any harm to the audio's quality.

Is there a representation with an optimal size Z that cannot be improved upon?

This question is tackled by information theory.

Compression



Information Theory

According to Shannon, the **entropy**[@] of an information source S is defined as:

$$H(S) = \sum_i (p_i \log_2 (1/p_i))$$

- $\log_2 (1/p_i)$ indicates the amount of information contained in *symbol* S_i, i.e., the number of bits needed to code *symbol* S_i.
- For example, in an image with uniform distribution of gray-level intensity, i.e. $p_i = 1/256$, with the number of bits needed to code each gray level being 8 bits. The entropy of the image is 8.
- Q: What is the entropy of a source with M symbols where each symbol is equally likely?
  - *Entropy,* $H(S) = \log_2 M$
- Q: How about an image in which half of the pixels are white and half are black?
  - *Entropy,* $H(S) = 1$

[@] *Here is an excellent primer by Dr. Schnieder on this subject*

Information Theory

Discussion:

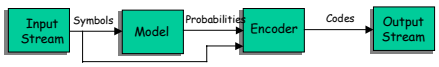
- Entropy is a measure of how much information is encoded in a message. Higher the entropy, higher the information content.
  - We could also say entropy is a measure of uncertainty in a message. Information and Uncertainty are equivalent concepts.
- The units (in coding theory) of entropy are *bits per symbol*.
  - It is determined by the base of the logarithm:
    2: binary (bit);
    10: decimal (digit).
- Entropy gives the actual number of bits of information contained in a message source.
  - **Example:** If the probability of the character 'e' appearing in this slide is 1/16, then the information content of this character is 4 bits. So, the character string "eeeee" has a total content of 20 bits (contrast this to using an 8-bit ASCII coding that could result in 40 bits to represent "eeeee").

Data Compression = Modeling + Coding

Data Compression consists of taking a stream of symbols and transforming them into codes.

- The model is a collection of data and rules used to process input symbols and determine their probabilities.
- A coder uses a model (probabilities) to spit out codes when its given input symbols

Let's take Huffman coding to demonstrate the distinction:



- The output of the Huffman encoder is determined by the Model (probabilities). *Higher the probability shorter the code.*
  - Model A could determine raw probabilities of each symbol occurring anywhere in the input stream. (p_i = *# of occurrences of* S_i */ Total number of Symbols*)
  - Model B could determine prob. based on the last 10 symbols in the i/p stream. (*continuously re-computes the probabilities*)

The Shannon-Fano Encoding Algorithm

1. Calculate the frequencies of the list of symbols (organize as a list).
2. Sort the list in (decreasing) order of frequencies.
3. Divide list into two halfs, with the total freq. *Counts of each half being as close as possible to the other.*
4. The upper half is assigned a code of 0 and lower a code of 1.
5. Recursively apply steps 3 and 4 to each of the halves, until each symbol has become a corresponding code leaf on a tree.

Example

Symbol	A	B	C	D	E
Count	15	7	6	6	5
	0	0	1	1	1
	0	1	0	1	1
				0	1

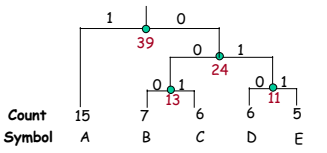
Symbol	Count		Info. $-\log_2(p_i)$	Code	Subtotal # of Bits
A	15	x	1.38	00	30
B	7	x	2.48	01	14
C	6	x	2.70	10	12
D	6	x	2.70	110	18
E	5	x	2.96	111	15
	85.25				89

It takes a total of 89 bits to encode 85.25 bits of information (Pretty good huh!)

The Huffman Algorithm

1. Initialization: Put all nodes in an OPEN list *L*, keep it sorted at all times (e.g., ABCDE).
2. Repeat the following steps until the list *L* has only one node left:
   1. From *L* pick two nodes having the lowest frequencies, create a parent node of them.
   2. Assign the sum of the children's frequencies to the parent node and insert it into L.
   3. Assign code 0, 1 to the two branches of the tree, and delete the children from L.

Example



Symbol	Count		Info. $-\log_2(p_i)$	Code	Subtotal # of Bits
A	15	x	1.38	1	15
B	7	x	2.48	000	21
C	6	x	2.70	001	18
D	6	x	2.70	010	18
E	5	x	2.96	011	15
	85.25				87

Huffman Alg.: Discussion

Decoding for the above two algorithms is trivial as long as the coding table (the statistics) is sent before the data. There is an overhead for sending this, negligible if the data file is big.

Unique Prefix Property: no code is a prefix to any other code (all symbols are at the leaf nodes)

--> great for decoder, unambiguous; **unique Decipherability?**

If prior statistics are available and accurate, then Huffman coding is very good.

Number of bits (per symbol) needed for Huffman Coding is:

87 / 39 = 2.23

Number of bits (per symbol)needed for Shannon-Fano Coding is:

89 / 39 = 2.28