COMSATS Institute of Information Technology

ECI750 Multimedia Data Compression

Lecture 4

Huffman Coding – I

Dr. Shadan Khattak

Department of Electrical Engineering

COMSATS Institute of Information Technology - Abbottabad

Huffman Coding

- In this lecture, we will study
  - How can we generate Huffman codes when the probability model of the source is known?
  - Different types of Huffman codes, including
    - Minimum Variance Huffman Codes,
    - Extended Huffman Codes, and
    - Non-binary Huffman Codes

Huffman Coding

- The Huffman Coding Algorithm
  - Developed by David Huffman as part of a class assignment
  - Huffman codes are prefix codes and are optimum for a given model (set of probabilities)
  - It is based on two ideas:
    - In an optimum code, symbols that occur more frequently (have a higher probability of occurrence) will have shorter codewords than symbols that occur less frequently.
    - In an optimum code, the two symbols that occur least frequently will have the same length.

Huffman Coding

- Huffman Coding Principle:
  - Starting with two least probable symbols, $\gamma\ and\ \delta$, of an alphabet A, if the codeword for γ is $[m]0$, the codeword for δ should be $[m]1$, where m is a string of $1s$ and 0s.
  - Now the two symbols can be combined into a group, which represents a new symbol ψ in the alphabet set.
    - The symbol ψ has the probability $P(\gamma) + P(\delta)$.
  - Recursively determine the bit pattern $[m]$ using the new alphabet set.

Dr. Shadan Khattak
Department of Electrical Engineering
CIIT - Abbottabad

Huffman Coding

- Example:

$$A = \{a_1, a_2, a_3, a_4, a_5\}$$
$$P(a_1) = P(a_3) = 0.2, P(a_2) = 0.4, P(a_4) = P(a_5) = 0.1$$
$$H = 2.122\ bits/symbols$$

Symbol	Step 1	Step 2	Step 3	Step 4	Codeword
a_2	0.4	0.4	0.4	0.6 0	1
a_1	0.2	0.2	0.4 0	0.4 1	01
a_3	0.2	0.2 0	0.2 1		000
a_4	0.1 0	0.2 1			0010
a_5	0.1 1				0011

Dr. Shadan Khattak
Department of Electrical Engineering
CIIT - Abbottabad

Huffman Coding

- Example:
  - Average codeword length:

  $l = 0.4 * 1 + 0.2 * 2 + 0.2 * 3 + 0.1 * 4 + 0.1 * 4 = 2.2$ bits/symbol

  - $Redundancy \ = \ Average\, Codeword\, Length - Entropy$

    $= 2.2 - 2.122$

    $= 0.078$ bits/symbol

  - For Huffman codes, the redundancy is zero when the probabilities are negative powers of 2.

Huffman Coding

- <u>**Minimum Variance Huffman Code**</u>
  - When more than two symbols in a Huffman tree have the same probability, different merge orders produce different Huffman codes.

Symbol	Step 1	Step 2	Step 3	Step 4	Codeword
a_2	0.4	0.4	0.4	0.6 0	00
a_1	0.2	0.2	0.4 0	0.4 1	10
a_3	0.2	0.2 0	0.2 1		11
a_4	0.1 0	0.2 1			010
a_5	0.1 1				011

The average codeword length is still 2.2 bits/symbol. But variances are different!

  - We prefer a code with smaller length-variance.
  - To create a minimum variance Huffman code, put the combined letter as high in the list as possible!

Dr. Shadan Khattak
Department of Electrical Engineering
CIIT - Abbottabad

Huffman Coding

- **<u>Minimum Variance Huffman Code</u>**
  - Why are we interested in minimum variance codes?
    - The greater the variance, the more difficult is the buffer design problem
  - Consider the following:
  - For both the codes considered in Slide 5 and Slide 7, the average codeword length is 2.2 bits/symbol.
  - If we have to transmit 10,000 symbols/sec, we need a transmission capacity of 10,000 x 2.2 = 22,000 bits/sec.
  - If we are transmitting only symbols a_4 and a_5 for a few seconds, then,
    - For the code on Slide 5: 10,000 x 4 = 40,000 bits/sec ⇒ 18,000 bits/sec buffering
    - For the code on Slide 7: 10,000 x 3 = 30,000 bits/sec ⇒ 8,000 bits/sec buffering

Huffman Coding

- **Length of the Huffman Code**
  - The Huffman coding procedure generates an optimum code.
  - The average codeword length $\hat{l}$ of an optimum code (and thus the Huffman code) is bounded below by the entropy of the source (S) and bounded above by the entropy of the source plus 1 bit i.e.,

$$H(S) \leq \hat{l} < H(S) + 1$$

  - Given a sequence of positive integers $\{l_1, l_2, \dots, l_k\}$ satisfies

$$\sum_{i=1}^{k} 2^{-l_i} \leq 1$$

There exists a uniquely decodable code whose codeword lengths are given by $\{l_1, l_2, \dots, l_k\}$

Huffman Coding

- **Extended Huffman Codes**
  - As P_{max} increases, the efficiency of Huffman coding decreases.

Huffman Coding

- **Extended Huffman Codes**
  - **Example 1:**
    - Consider a source that puts out iid letters from the alphabet $A = \{a_1, a_2, a_3\}$ with the probability model: $P\{a_1\} = 0.8, P\{a_2\} = 0.02, P\{a_3\} = 0.18$
    - Entropy = 0.816 bits/symbol
    - Huffman Code:

Letter	Codeword
a_1	0
a_2	11
a_3	10

    - Average codeword length: 1.2 bits/symbol
    - Redundancy: 0.384 bits/symbol (47% of the entropy)
    - ⇒ to code this sequence, we would need 47% more bits than the minimum required.

Huffman Coding

- **Extended Huffman Codes**
  - **Example 2:**
    - Average codeword length: 1.7228 bits/symbol
    - Average codeword length in terms of the original alphabet: 1.7228/2 = 0.8614 bits/symbol
    - Redundancy: 0.045 bits/symbol (5.5% of the entropy)
    - ⇒ to code this sequence, we would need 5.5% more bits than the minimum required.

Letter	Probability	Code
a_1a_1	0.64	0
a_1a_2	0.016	10101
a_1a_3	0.144	11
a_2a_1	0.016	101000
a_2a_2	0.0004	10100101
a_2a_3	0.0036	1010011
a_3a_1	0.1440	100
a_3a_2	0.0036	10100100
a_3a_3	0.0324	1011

Dr. Shadan Khattak
Department of Electrical Engineering
CIIT - Abbottabad

Huffman Coding

- **<u>Extended Huffman Codes</u>**
  - By coding blocks of symbols together, we can reduce the redundancy of Huffman codes.
  - Blocking two symbols together, the alphabet size grows from m to m^2
  - As we block more and more symbols together, the size of the alphabet grows exponentially, and the Huffman coding scheme becomes more impractical.
  - Under these conditions, a more practical coding technique is *arithmetic coding* which we are going to see in the next week.

Huffman Coding

- **<u>Non-Binary Huffman Codes</u>**
  - Huffman codes can be applied to n-ary code space. For example, codewords composed of {0,1,2} called ternary Huffman code.

Letter	Probability	Codeword
a_1	0.20	2
a_2	0.05	021
a_3	0.20	00
a_4	0.20	01
a_5	0.25	1
a_6	0.10	020