Imperial College London

Digital Image Processing

Huffman Coding

DR TANIA STATHAKI
READER (ASSOCIATE PROFESSOR) IN SIGNAL PROCESSING
IMPERIAL COLLEGE LONDON

Elements from Information Theory

- Any information generating process can be viewed as a source that emits a sequence of symbols chosen from a finite alphabet.
  - ASCII symbols (text)
  - n –bit image values (2^n symbols)

- The simplest form of an information source is the so called Discrete Memoryless Source (DMS). Successive symbols produced by such a source are statistically independent.

- A DMS is completely specified by the source alphabet $S = \{s_1, s_2, \dots, s_n\}$ and the associated probabilities $P = \{p_1, p_2, \dots, p_n\}$

Elements from Information Theory

- The **Self-Information** of a symbol s_i with probability p_i is defined as:

$$I(s_i) = \log_2 \frac{1}{p_i} = -\log_2 p_i$$

  - The occurrence of a less probable event provides more information.
  - The information of a sequence of independent events taken as a single event equals the sum of their individual information.
  - An event can be the occurence of a symbol.

- The **Average Information per Symbol** or **Entropy** of a DMS is:

$$H(S) = \sum_{i=1}^{n} p_i I(s_i) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

- The Entropy is measured in bits/symbol.

What is actually entropy?

- Interpretation of entropy:
  - By definition it is the average amount of information that symbols of a specific source carry.
  - Entropy is also a measure of the "disorder" (uncertainty) of a system.

- When we design a coding scheme the average number of bits per symbol we can achieve is always greater that the entropy. **Therefore, the entropy is the best we can do in term of bits/symbol!**

Extension of a DMS

- Given a DMS of size n, it might be beneficial to group the original symbols of the source into blocks of N symbols. Each block can now be considered as a single source symbol generated by a source S^N which has n^N symbols.

- In this case the entropy of the new source is

$$H(S^N) = N \times H(s)$$

- We observe that when the source is extended, the entropy increases, however, the symbols increase in length. The entropy per original symbol remains the same.

Noiseless source coding theorem

- Let S be a source with alphabet size n and entropy $H(s)$.

- Consider coding blocks of N source symbols into binary codewords. For any $\delta > 0$ (with δ a small number), it is possible by choosing N large enough to construct a code in such a way that the average number of bits per original source symbol l_{avg} satisfies the following:

$$H(S) \leq l_{avg} < H(s) + \delta$$

- We observe that for δ small enough, the average number of bits per symbols converges to the entropy of the source. This is the best coding we can achieve.

- The above is not realistic since the alphabet size increases too much with N.

Examples of possible codes for a 4-symbol source

- In the table below we see four different codes for a four-symbol alphabet. The entropy of the source is 1.75bits/symbol.

Symbols	Probability	Code 1	Code 2	Code 3	Code 4
s_1	1/2	0	0	0	0
s_2	1/4	0	1	10	01
s_3	1/8	1	00	110	011
s_4	1/8	10	11	111	0111
Average length		1.125	1.25	1.75	1.875

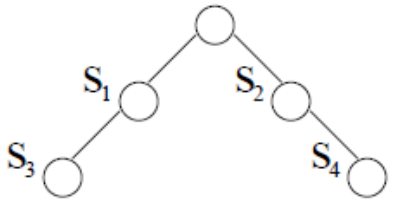
- The **Average Length** of a code is

$$l_{avg} = \sum_i l_i p_i$$

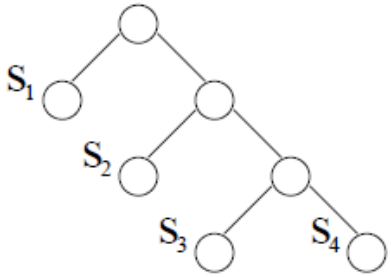
- l_i is the length of the codeword in bits which corresponds the symbol s_i.

Code characteristics

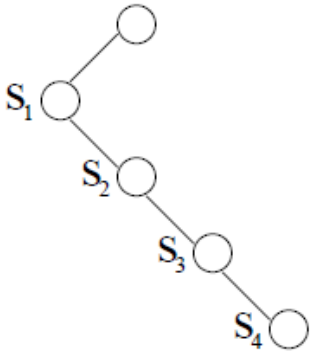
- It is desirable for a code to exhibit Unique Decodability.
- **Prefix Codes**: no codeword is a prefix of another codeword.
- A prefix code is always uniquely decodable. The reverse is not true.
- A code can be possible depicted as a binary tree where the symbols are the nodes and the branches are 0 or 1. The codeword of a symbold can be found if we concatenate the 0s and 1s that we have to scan until we reach that symbol, starting from the “root “ of the tree.
- In a prefix code the codewords are associated only with the external nodes.



Code 2



Code 3



Code 4

Huffman coding (1952)

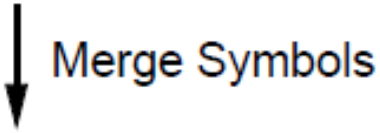
- Huffman coding is a very popular coding method.

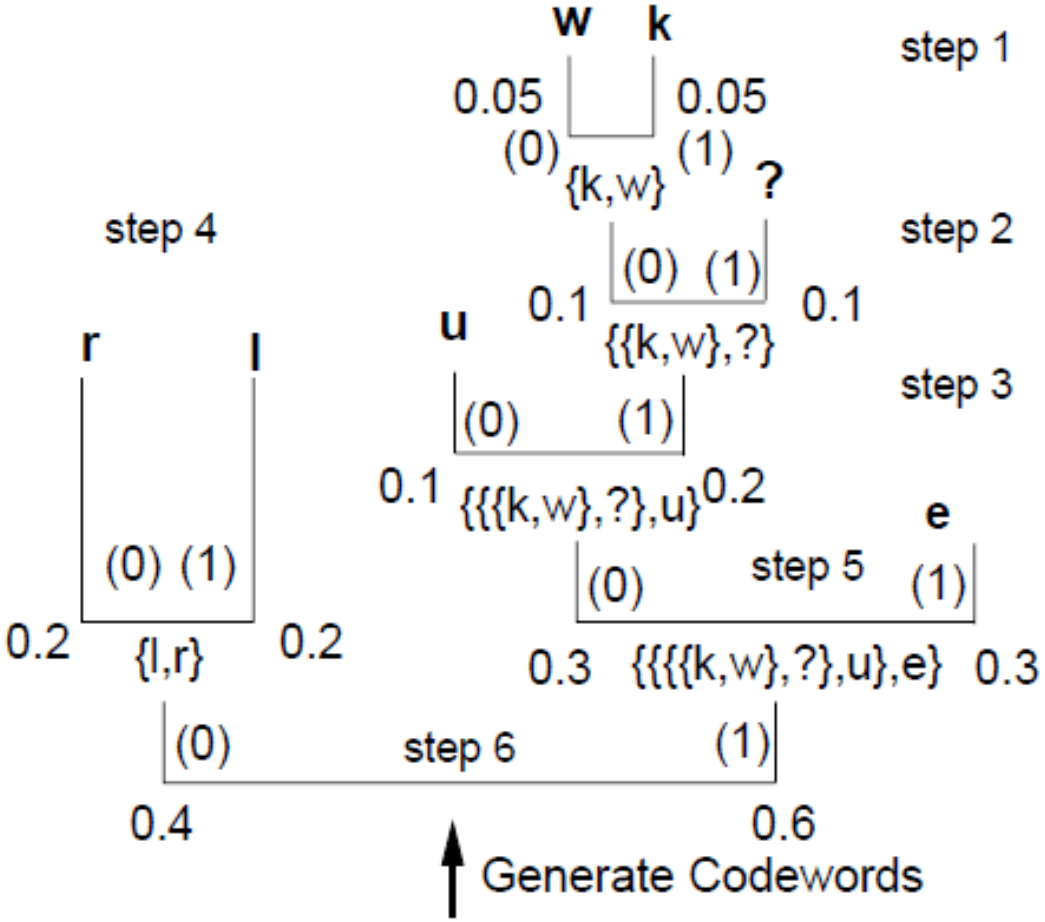
- Huffman codes are:
  - Prefix codes.
  - Optimal for a given model (set of probabilities).

- Huffman coding is based on the following two observations:
  - Symbols that occur more frequently will have shorter codewords than symbols that occur less frequently.
  - The two symbols that occur less frequently will have the same length.

Huffman coding

Symbol	Probability	Codeword
k	0.05	10101
l	0.2	01
u	0.1	100
w	0.05	10100
e	0.3	11
r	0.2	00
?	0.1	1011





Imperial College London

Huffman coding

		Step 1	Step 2	Step 3	Step 4	Step 5	Step 6
k	1/20	e 0.3	e 0.3	e 0.3	e 0.3	{l,r} 0.4	{{{{k,w},?},u},e} 0.6
l	0.2	l 0.2	l 0.2	l 0.2	{{{k,w},?},u} 0.3	e 0.3	{l,r} 0.4
u	0.1	r 0.2	r 0.2	r 0.2	l 0.2	{{{k,w},?},u} 0.3	
w	1/20	u 0.1	u 0.1	{{k,w},?} 0.2	r 0.2		
e	0.3	? 0.1	? 0.1	u 0.1			
r	0.2	k 0.05	{k,w} 0.01				
?	0.1	w 0.05					

Properties of Huffman Codes

- $H(S) \leq l_{avg} \leq H(S) + 1$
- If $p_{max} < 0.5$ then $l_{avg} \leq H(S) + p_{max}$
- If $p_{max} \geq 0.5$ then $l_{avg} \leq H(S) + p_{max} + 0.086$
- $H(S) = l_{avg}$ if the probabilities of the symbols are of the form 2^k, with k a negative integer.
- For an N −th extension of a DMS we have $H(S) \leq l_{avg} \leq H(S) + \frac{1}{N}$
- The complement of a Huffman code is also a valid Huffman code.
- A minimum variance Huffman code is obtained by placing the combined letter in the sorted list as high as possible.
- The code efficiency is defined as $H(S)/l_{avg}$
- The code redundancy is defined as $l_{avg} - H(S)$

Huffman Decoding: Bit-Serial Decoding

- This method is a fixed-input bit rate but variable-output symbol rate scheme. It consists of the following steps:
  1. Read the input compressed stream bit by bit and traverse the tree until a leaf node is reached.
  2. As each bit in the input stream is used, it is discarded. When the leaf node is reached, the Huffman decoder outputs the symbol at the leaf node. This completes the decoding for this symbol.

- We repeat these steps until all of the input is consumed. Since the codewords are not of the same length, the decoding bit rate is not the same for all symbols. Hence, this scheme has a fixed input bit rate but a variable output symbol rate.

Huffman Decoding: Lookup-Table-Based Decoding

- Lookup-table-based methods have a variable input bit rate and constant decoding symbol rate.
- We have to construct the so called **Lookup Table** using the symbol-to-codeword mapping table (Huffman code). If the longest codeword in this table is L bits, then the lookup table will have 2^L rows.
- Let c_i be the codeword that corresponds to symbol s_i. Assume that c_i has l_i bits. In this method we associate c_i not with a single codeword but with 2^{L-l_i} codewords. These are all the codewords where the first l_i bits are the codeword c_i and the last $L - l_i$ bits can be all possible binary numbers with $L - l_i$ bits. These are 2^{L-l_i} on total. Therefore each symbol s_i is associated with 2^{L-l_i} codewords of fixed length L.
- The 2^{L-l_i} pairs $\left(s_i, \text{codeword}_{ij}\right), j = 1, \dots, 2^{L-l_i}$ are stored in into the Lookup Table.

Huffman Decoding: Lookup-Table-Based Decoding (cont.)

- When we receive the bit stream for decoding we read the first L bits.
- By checking the Lookup Table we find the symbol s_i which has the read L-bit word as one if its possible codewords.
- When we find this symbol we know that the “true” codeword for that symbol is formed by the first l_i bits only of the read L-bit word.
- The first l_i bits are discarded from the buffer.
- The next l_i bits are appended to the buffer so that the next L-bit word for investigation is formed.
- We carry on this procedure until the entire bit stream is examined.