EC5.102: Information and Communication

(Lec-3)

Source coding-3

(6-March-2025)

Arti D. Yardi

Email address: arti.yardi@iiit.ac.in
Office: A2-204, SPCRC, Vindhya A2, 1st floor

Summary of the last class

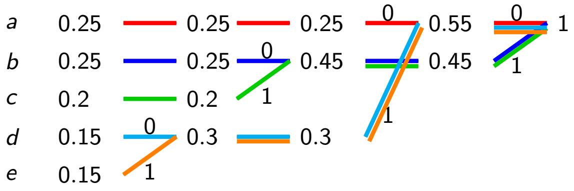
Recap

- Introduction to source coding: Definition, Examples, Expected length of code $L(C)$
- Types of source codes: Singular/Non-singular, Uniquely decodable, Prefix or instantaneous
- Examples of source codes:
  - ASCII: American Standard Code for Information Interchange
  - MORSE code
  - Huffman codes: Used for image compression, JPEG
  - Lempel-Ziv algorithm: ZIP, Gif

Huffman coding

Huffman code: Example 1

- $\mathcal{X} = \{a, b, c, d, e\}$
- $p(a) = 0.25, p(b) = 0.25, p(c) = 0.2, p(d) = 0.15, p(e) = 0.15$
- Huffman algorithm:



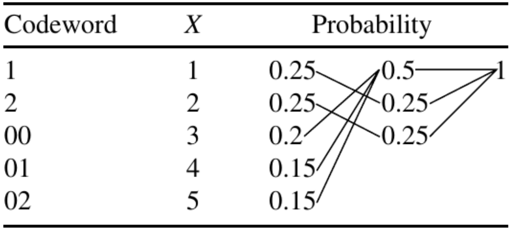
$C(a) = \{0\ 0\}$ $C(b) = \{1\ 0\}$ $C(c) = \{1\ 1\}$

$C(d) = \{0\ 1\ 0\}$ $C(e) = \{0\ 1\ 1\}$

- Compute $H(X)$ and expected length $L(C)$, and coding efficiency.

Huffman code: Example 2

- Construct a ternary code for the following example
- $\mathcal{X} = \{a, b, c, d, e\}$
- $p(a) = 0.25, p(b) = 0.25, p(c) = 0.2, p(d) = 0.15, p(e) = 0.15$
- Huffman algorithm:



Codeword	X	Probability		
1	1	0.25	0.5	1
2	2	0.25	0.25	
00	3	0.2	0.25	
01	4	0.15		
02	5	0.15		

Huffman code: Example 3 (Classwork)

- Construct a ternary and binary source code for the following example
- $\mathcal{X} = \{m_1, m_2, m_3, m_4, m_5, m_6\}$
- $p(m_1) = 0.3, p(m_2) = 0.25, p(m_3) = 0.15, p(m_4) = 0.12, p(m_5) = 0.1, p(m_6) = 0.08$
- Compare efficiencies of both the codes.

Huffman codes: Important notes

- Key idea: Map the more frequently occurring symbols to shorter binary sequences and the less frequently occurring ones to longer binary sequences, thus achieving good coding efficiency.
- For the given pmf, Huffman codes have the lowest expected length, among the class of uniquely decodable codes! (We won't prove this, see examples to justify this claim.)
- Huffman coding is fixed-to-variable length coding.
- Huffman codes are prefix codes. Why?
- **We will next see how to construct Huffman codes with $L(C) \to H(X)$!**

Construct Huffman codes with $L(C) \to H(X)$

Huffman code: Example

- A zero-memory source emits messages m_1 and m_2 with probabilities 0.8 and 0.2 respectively.
  - Find binary Huffman code.
  - Find Huffman code for its second and third order extensions.
  - Determine code efficiency in each case.
- Solution: In class

Huffman code: Solution

- Huffman code for $n = 1$ will be 0 and 1.
  - $L(C) = 1$ and $H(X) = -(0.8 \log_2(0.8) + 0.2 \log_2(0.2)) = 0.72$ bits
  - $\eta = H(X)/L(C) = 0.72$
- Huffman code for $n = 2$:

$m_1\ m_1$	0.64	0
$m_1\ m_2$	0.16	11
$m_2\ m_1$	0.16	100
$m_2\ m_2$	0.04	101

  - $L_1(C) = 1.56$. But this word length for two messages of the original source.
  - Word length per message will be $L(C) = 1.56/2 = 0.78$
  - $\eta = 0.72/0.78 = 0.923$

Huffman code: Solution

- Huffman code for $N = 3$:

m_1 m_1 m_1	0.512	0
m_1 m_1 m_2	0.128	100
m_1 m_2 m_1	0.128	101
m_2 m_1 m_1	0.128	110
m_1 m_2 m_2	0.032	11100
m_2 m_1 m_2	0.032	11101
m_2 m_2 m_1	0.032	11110
m_2 m_2 m_2	0.008	11111

- ◦ $L_3(C) = 2.184$. But this word length for three messages of the original source.
- ◦ Word length per message will be $L(C) = 2.184/3 = 0.728$
- ◦ $\eta = 0.72/0.728 = 0.989$

Observations

- Summary:
  - For $n = 1$, $\eta = 0.72$
  - For $n = 2$, $\eta = 0.923$
  - For $n = 3$, $\eta = 0.989$
- Thus, as the block length increases, the coding efficiency improves and approaches to 1.
- As we use the Huffman coding algorithm over longer and longer blocks of symbols, the average number of bits required to encode each symbol approaches the entropy of the source.
- We will next see why is it so.

Observations

- We can show that the average codeword length of Huffman code satisfies the following inequality (proof skipped)

$$H(X) \leq L(C) \leq H(X) + 1$$

- If the Huffman code is designed for sequences of source letters of length n (the nth order extension of the source), we have

$$H(X^n) \leq L_n(C) \leq H(X^n) + 1$$

- Codeword length per message is $L(C) = L_n(C)/n$
- For memoryless source we have $H(X^n) = nH(X)$

$$H(X) \leq L(C) \leq H(X) + \frac{1}{n}$$

- If the source is memoryless, then for Huffman coding with sufficiently large block length (n), average number of bits required to encode each symbol approaches the entropy of the source.

Drawback of Huffman Coding

- The Huffman code is optimal in the sense that, for a given source, they provide a prefix code with minimum number of bits per message.
- It has two disadvantages:
    - It depends on the source probabilities; source statistics need to know in advance to design the algorithm
    - Complexity of the algorithm exponentially increases with the block length.
- Not a good choice for any practical source whose statistics are not known in advance.

Lempel Ziv coding

Lempel-Ziv (L-Z) Coding: Example 1

- Consider the sequence: 1 0 1 0 1 1 0 1 1 0 1 0 1 0 1 0 1 0
- Parsing: Identify phrases of the smallest length that haven't appeared before.
- After parsing the phrases are: 1, 0, 1 0, 1 1, 0 1, 1 0 1, 0 1 0, 1 0 1 0
- Notice: New phrase is concatenation of a previous phrase and a new source message.
- Encoding: Lexicographic (dictionary) ordering of the previous phrase and the new source message are concatenated.

In which dictionary location the longest prefix of the content appeared before

The last message of the content

Dictionary location	Contents	Codeword
1	1	(0,1)
2	0	(0,0)
3	10	(1,0)
4	11	(1,1)
5	01	(2,1)
6	101	(3,1)
7	010	(5,0)
8	1010	(6,0)

Lempel-Ziv (L-Z) Coding

The steps of L-Z algorithm are as follows:

- Any sequence of the source output is uniquely parsed into phrases of varying length and these phrases are encoded using codewords of equal length. (This is a variable-to-fixed length coding scheme.)
- Parsing is done by identifying phrases of the smallest length that have not appeared before.
- The new phrase is the concatenation of a previous phrase and a new source message.
- Encoding: Lexicographic (dictionary) ordering of the previous phrase and the new source message are concatenated.

Next class

- Kraft inequality
- Show that: For Prefix codes, $L(C) \geq H(X)$
- Statement of source coding theorem
- IMPORTANT: Not to miss the next class