EC5.102: Information and Communication

Module: Source coding

Arti D. Yardi

Email address: arti.yardi@iiit.ac.in

Introduction to source coding (Data compression)

Introduction to data compression

- Suppose you have been given a file with contents:

  a c b a a a d a a b b a a b c d

- To transmit this file, you need to assign a sequence of 0s & 1s to each alphabet.
  - For example, one possible assignment could be

    a = 0 0 b = 0 1 c = 1 0 d = 1 1

  - The file is then given by

    0 0, 1 0, 0 1, 0 0, 0 0, 0 0, 1 1, 0 0, 0 0, 0 1, 0 1, 0 0, 0 0, 0 1, 1 0 1 1

  - The file has 32 bits

- Can you represent this file with fewer number of bits? **Yes!**
- This is **data compression**!
- The process of converting alphabet of a file into bit-sequence is called as **"source encoding"**.

Definition: Source code

- Example of a source code:
  - File: a c b a a a d a a b b a a b c d
  - To transmit this file, we assign: a = 0 0 b = 0 1 c = 1 0 d = 1 1
  - 0 0 is called as the "codeword" of a.
- **Definition of a source code:**
  - Consider a r.v. X with support set $\mathcal{X}$.
  - Let $\mathcal{D}^*$ be the set of finite-length strings of symbols from a D-ary alphabet. We can assume that the D-ary alphabet $\mathcal{D} = \{0, 1, \ldots, D-1\}$.
  - A source code C is defined as a mapping from $\mathcal{X}$ to $\mathcal{D}^*$.
  - $C(x)$: Codeword of $x \in \mathcal{X}$.
  - $\ell(x)$: Length of $C(x)$
- Encoding and decoding

Expected length of a source code $C(x)$

- Example continued...
  - File: a c b a a a d a a b b a a b c d
  - Source code: a = 0 0 b = 0 1 c = 1 0 d = 1 1
  - What is pmf of X?
- The expected length $L(C)$ a source code $C(x)$ for a r.v. X with pmf $p(x)$ is defined as

$$L(C) = \mathbb{E}_X\Big[\ell(X)\Big] = \sum_{x\in\mathcal{X}} p(x)\ell(x)$$

- Rate = Number of bits after source encoding / Number of symbols = $L(C)$
- **For "good" compression we wish to have $L(C)$ as low as possible!**
  - Can we choose a source code s.t. length of any codeword is say 1?
  - Will it be a "good source code"?
  - How to define a "good"?

How to define a "good" source code?

- Let X be a discrete RV with support set $\mathcal{X}$ and pmf $\{p(x)\}$ where $x \in \mathcal{X}$.
- Consider a source code $C : \mathcal{X} \to \mathcal{D}^*$ with expected length $L(C)$.
- How to define "good" source code?
  - $L(C)$ should be low for good compression.
  - How much low? Hint: We should not loose "information" contained in rv X!!
- A source code is said to be "optimal" if $L(C) = H(X)$.
- Coding efficiency $\eta := H(X) \ / \ L(C)$.
- For "lossless" data compression, $L(C) \geq H(X)$: **Source coding theorem** (Note: This is NOT a formal statement!)

Optimal source code for our example

- File: a c b a a a d a a b b a a b c d
- pmf of X: $\mathbb{P}[X = a] = 1/2,\ \mathbb{P}[X = b] = 1/4,\ \mathbb{P}[X = c] = 1/8,\ \mathbb{P}[X = d] = 1/d$
- Can you construct an optimal source code?
- Consider the following source code:

  a $= 0$ b $= 1\ 0$ c $= 1\ 1\ 0$ d $= 1\ 1\ 1$

  - What is the expected length $L(C) = \sum_{x \in \mathcal{X}} p(x) l(x)$ of this source code?

$$L(C) = \tfrac{1}{2} \times 1 + \tfrac{1}{4} \times 2 + \tfrac{1}{8} \times 3 + \tfrac{1}{3} \times 3 = 1.75 \text{ bits}$$

  - What is entropy $H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x)$ this pmf?

$$H(X) = \tfrac{1}{2} \times 1 + \tfrac{1}{4} \times 2 + \tfrac{1}{8} \times 3 + \tfrac{1}{3} \times 3 = 1.75 \text{ bits}$$

  - This is an optimal code since $L(C) = H(X)$.
  - Note: This optimal code is a variable length code!

Source coding: Questions of interest

- Is there an algorithm to design an optimal code systematically?
- Is optimal code unique?
- What if I don't know the pmf of X?
- What is the intuition behind the result that "for lossless source coding, the minimum value of expected length of source code $L(C)$ is equal to entropy $H(X)$"?
- What will happen if $L(C) < H(X)$?
- Is it desirable to design a source code with $L(C) < H(X)$?
- Can I design a "fixed-length" source code which is optimal?

- Types of source codes
  - Singular and nonsingular source code
  - Unique decodability of a source code
  - Prefix or instantaneous source code

Singular and nonsingular source code

- A code is said to be nonsingular if every element of $\mathcal{X}$ maps into a different string in $\mathcal{D}^*$, i.e.,

$$x \neq x' \Rightarrow C(x) \neq C(x') \text{ for any } x, x' \in \mathcal{X}$$

- Consider the source code for $\mathcal{X} = \{a\ b\ c\ d\}$ defined as follows.

$$C(a) = 0\ 0 \quad C(b) = 0\ 1 \quad C(c) = 1\ 0 \quad C(d) = 1\ 1$$

  This code is nonsingular.

- Consider the source code for $\mathcal{X} = \{a\ b\ c\ d\}$ defined as follows.

$$C(a) = 0 \quad C(b) = 0 \quad C(c) = 0\ 1 \quad C(d) = 1\ 1$$

  This code is singular.

- Why is it desirable to have nonsingular code?
  - Nonsingularity suffices for an unambiguous description of any $x \in X$.

Unique decodability of a source code

- Consider the following nonsingular source code for $\mathcal{X} = \{a\ b\ c\ d\}$ defined as follows.

$$C(a) = 0 \quad C(b) = 1 \quad C(c) = 0\ 1 \quad C(d) = 1\ 1$$

- File: a c b a a a d a a b b a a b c d
- Source coded file is given by

0, 0 1, 1, 0, 0, 0, 1 1, 0, 0, 1, 1, 0, 0, 1, 0 1, 1 1

- We have used "comma" to separate codewords and this ensures unique decodability.
- Without using comma, source coded file is given by

0 0 1 1 0 0 0 1 1 0 0 1 1 0 0 1 0 1 1 1 Wrong

0 0 1 1 0 0 0 1 1 0 0 1 1 0 0 1 0 1 1 1 Correct

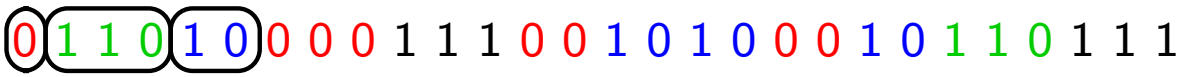
- Can we get rid of this "comma" and still ensure unique decodability? Yes!

Uniquely decodable source code: Example

- Consider the following nonsingular source code for $\mathcal{X} = \{a\ b\ c\ d\}$ defined as follows.

$$C(a) = 0 \quad C(b) = 1\ 0 \quad C(c) = 1\ 1\ 0 \quad C(d) = 1\ 1\ 1$$

- File: a c b a a a d a a b b a a b c d
- Source coded file is given by



- We can decode this file uniquely without using comma!
- We shall next define this formally.

Definition: Uniquely decodable source code

- Concatenation $C(x_1)C(x_2)$ of two codewords $C(x_1) = 0\ 0$ and $C(x_2) = 1\ 1$ is

$$C(x_1)C(x_2) = 0\ 0\ 1\ 1$$

- The extension C^* of a code C is the mapping from finite-length strings of $\mathcal{X}$ to to finite-length strings of $\mathcal{D}$, defined by

$$C(x_1, x_2, \ldots, x_n) = C(x_1)C(x_2)\ldots C(x_n)$$

- A code is called uniquely decodable if its extension is nonsingular.
- Any encoded string in a uniquely decodable code has only one possible source string producing it.
- However, one may have to look at the entire string to determine even the first symbol in the corresponding source string.

Unique decodability: Example

- Which of the following code is uniquely decodable?

1. $\{0, 10, 11\}$
2. $\{0, 01, 11\}$
3. $\{0, 01, 10\}$
4. $\{0, 01\}$
5. $\{0, 01, 10, 11\}$
6. $\{110, 11, 10\}$
7. $\{110, 11, 100, 00, 10\}$

1. Yes
2. Yes
3. No
4. Yes
5. No
6. Yes
7. Yes

Prefix code or instantaneous source code

- A code is called a prefix code or instantaneous code if no codeword is a prefix of any other codeword.
- For example, the following code is a prefix code.

$$C(a) = 0 \quad C(b) = 1\ 0 \quad C(c) = 1\ 1\ 0 \quad C(d) = 1\ 1\ 1$$

- Example:

0 1 0 1 1 1 1 1 0 1 0

0, 1 0 1 1 1 1 1 0 1 0

0, 1 0, 1 1 1 1 1 0 1 0

0, 1 0, 1 1 1, 1 1 0 1 0

0, 1 0, 1 1 1, 1 1 0, 1 0

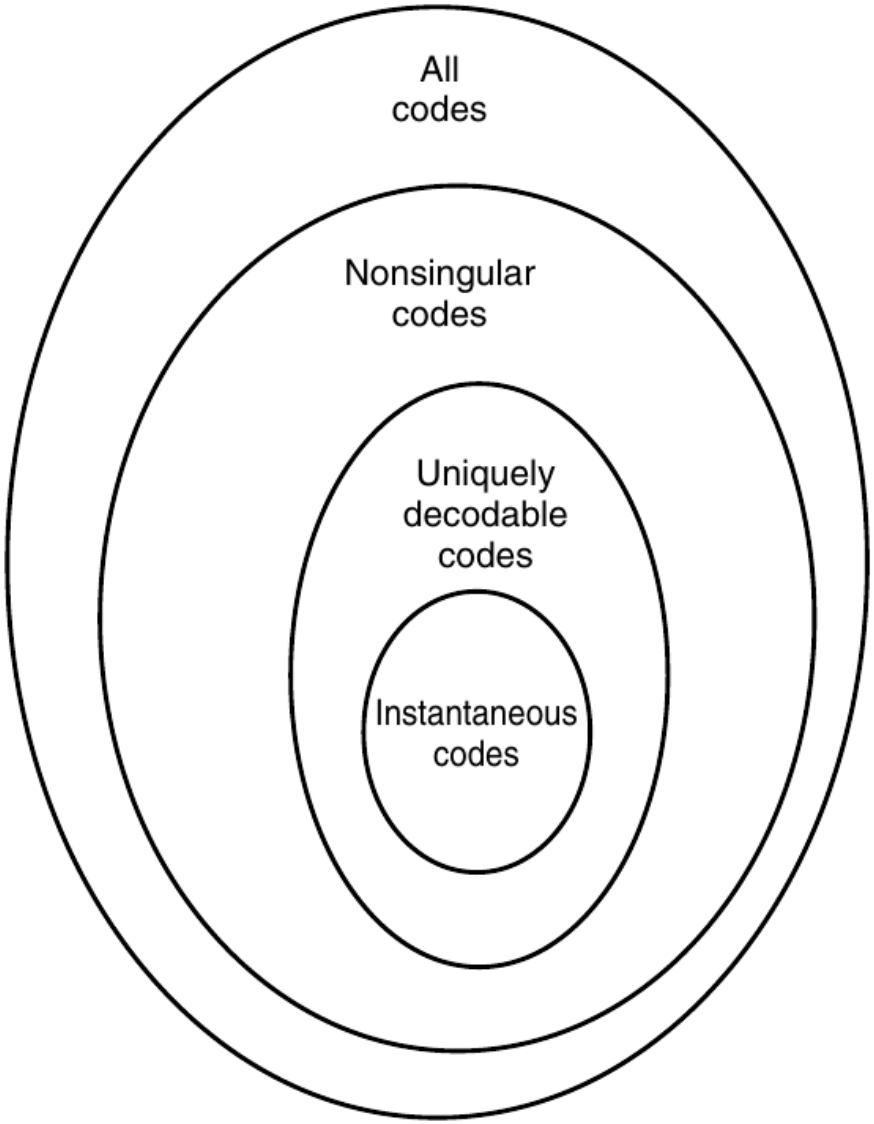
- Instantaneous code: We can look down the sequence of code symbols and add commas to separate the codewords without looking at later symbols.
- It is a self-punctuating code!

Classes of source codes

X	Singular	Nonsingular, But Not Uniquely Decodable	Uniquely Decodable, But Not Instantaneous	Instantaneous
1	0	0	10	0
2	0	010	00	10
3	0	01	11	110
4	0	10	110	111

- Verify each property: Homework

Classes of source codes



Criteria for Optimal Code Design

- Code must be uniquely decodable.
- Code should be a prefix or instantaneous.
  - Prefix Condition: A code is said to satisfy prefix condition if no code word is prefix to another code word.
  - Prefix condition is necessary and sufficient condition for a code to be uniquely decodable and instantaneous.
- Need smaller average codeword length among all prefix codes.
- Huffman codes are uniquely decodable, instantaneous codes with minimum average codeword length. In this sense, they are optimal.

Summary

- Introduction to source coding
  - Data compression
  - Definition of source code
  - Optimal source code
  - Singular and nonsingular source code
  - Unique decodability of a source code
  - Prefix or instantaneous source code
  - Criteria for optimal code design

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
- Towards source coding theorem...
  - Can you construct a uniquely decodable source code C for a file with alphabets a, b, c, d and pmf $\left[\frac{1}{2} \ \frac{1}{4} \ \frac{1}{8} \ \frac{1}{8}\right]$ s.t. $L(C) < H(X)$?
  - For "lossless source coding" we need: $L(C) \geq H(X)$
  - If for a code $L(C) < H(X)$, then we cannot have zero decoding error using this code.
  - We shall discuss this in detail today.

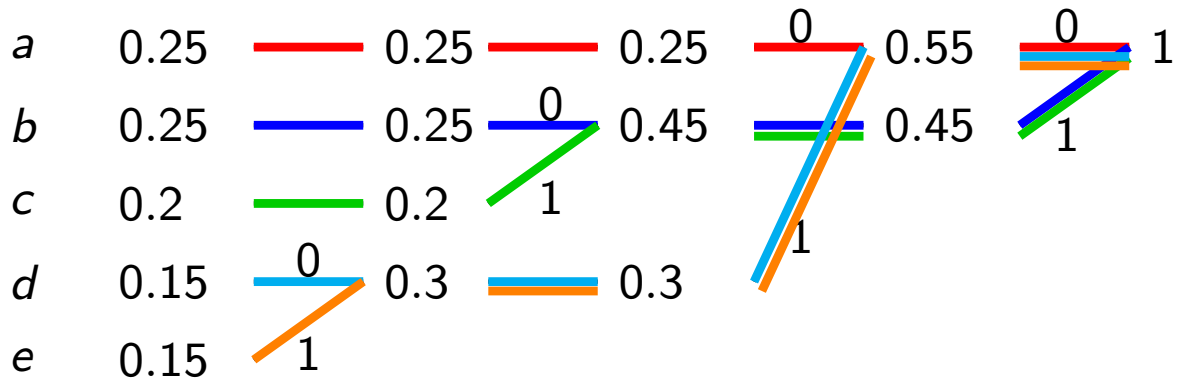
Huffman coding

Introduction to Huffman coding

- In Huffman coding, fixed length blocks of the source output are mapped to variable length binary blocks. This is called fixed-to-variable length coding.
- The idea here is to map the more frequently occurring fixed length sequences to shorter binary sequences and the less frequently occurring ones to longer binary sequences, thus achieving good coding efficiency.
- An prefix code for a given distribution can be constructed by a simple algorithm discovered by Huffman.
- Huffman code is a uniquely decodable code.

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.
- Why this code is a prefix code? (Homework)

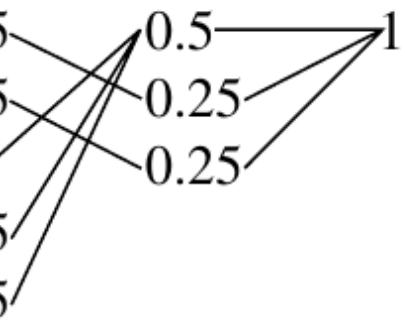
Huffman code: Example 2

- Construct a binary Huffman code for the following example.
- $\mathcal{X} = \{a, b, c, d\}$
- $p(a) = 1/2, p(b) = 1/4, p(c) = 1/8, p(d) = 1/8$
- In class

Huffman code: Example 3

- 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 4 (Homework)

- 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 code: Example 5

- Determine the Huffman code for a coin such that $p(\text{head}) = 0.8$. Compare this with the entropy and determine the efficiency of your code.
- Answer: The outcome heads is most likely but it is not possible to encode the outcome using less than one bit. Although the entropy is 0.72, the number of bits needed is 1 so the efficiency is only 0.72.

Huffman code: Example 6

- 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 of Example 6

- 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 of Example 6

- 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 from Example 6

- 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 from Example 6

- 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.
- There are many algorithms that overcome drawbacks of Huffman coding: Out of scope

Source coding theorem

Intuition behind source coding theorem (Example)

$N = 2$

$m_1\ m_1$	0.64
$m_1\ m_2$	0.16
$m_2\ m_1$	0.16
$m_2\ m_2$	0.04

$N = 3$

$m_1\ m_1\ m_1$	0.512
$m_1\ m_1\ m_2$	0.128
$m_1\ m_2\ m_1$	0.128
$m_2\ m_1\ m_1$	0.128
$m_1\ m_2\ m_2$	0.032
$m_2\ m_1\ m_2$	0.032
$m_2\ m_2\ m_1$	0.032
$m_2\ m_2\ m_2$	0.008

$N = 4$

$m_1\ m_1\ m_1\ m_1$	0.4096
$m_1\ m_1\ m_1\ m_2$	0.1024
$m_1\ m_1\ m_2\ m_1$	0.1024
$m_1\ m_2\ m_1\ m_1$	0.1024
$m_2\ m_1\ m_1\ m_1$	0.1024
$m_1\ m_1\ m_2\ m_2$	0.0256
$m_2\ m_1\ m_1\ m_2$	0.0256
$m_2\ m_2\ m_1\ m_1$	0.0256
$m_2\ m_1\ m_2\ m_1$	0.0256
$m_1\ m_2\ m_1\ m_2$	0.0256
$m_2\ m_2\ m_2\ m_1$	0.0064
$m_2\ m_2\ m_1\ m_2$	0.0064
$m_2\ m_1\ m_2\ m_2$	0.0064
$m_1\ m_2\ m_2\ m_2$	0.0064
$m_2\ m_2\ m_2\ m_2$	0.0016

Imagine the situation when $N = 10, 20, 100, 1000, \ldots$

Intuition behind source coding theorem (No proofs)

- Consider a discrete rv X with support set $\mathcal{X}$ and entropy $H(X)$.
- Theorem: The $\mathcal{X}^n$ set can be divided into two sets A and A^c as follows:
  - Typical set A: $P(A) \approx 1$ (as $n \to \infty$)
  - Non-typical set A^c: $P(A^c) \approx 0$ (as $n \to \infty$)
- Theorem: $|A| \approx 2^{nH(X)}$ and all elements in set A are equiprobable.
- What will an "optimal" source code for a uniform distribution over a set with 2^d elements? What will be its expected length?
- If we focus only on set A, what will be the expected length $L(C)$ of rate R of an optimal code?
- **Observe: We have a fixed-length optimal code!!**
- NOTE: Error will go to zero as $n \to \infty$

Source coding theorem

- Notation: $X, X^n, \mathcal{X}, \mathcal{X}^n, H(X)$
- Encoding: For $x^n \in \mathcal{X}^n$ codeword is $f^n(x^n)$
- Decoding: Codeword $f^n(x^n)$ is decoded as $g^n(f^n(x^n))$
- Statement of source coding theorem:
  - Achievability: For any rate $R > H(X)$, there exists a sequence of codes $\{f^{(n)}, g^{(n)}\}$ of rate R such that

$$P_e^{(n)} := \mathbb{P}\left[X^n \neq g^{(n)}\left(f^{(n)}(X^n)\right)\right] \to 0 \quad \text{as} \quad n \to \infty$$

  - Converse:

    Any rate $R < H(X)$ is not achievable.

Understanding source coding theorem

- Source coding theorem establishes a fundamental limit on the rate at which the output of an information source can be compressed without causing large error probability at the receiver.
- This is one of the fundamental theorems of information theory.
- It states that, a source with entropy rate H can be encoded with arbitrarily small error probability at any rate R (bits/source output) provided, $R > H$.
- If $R < H$, the error probability will be bounded away from zero, independent of the complexity of the encoder and decoder employed.