Digital Communications
SSY125, Lectures 2, 3, and 4

A Measure of Information, Source Compression (Chapters 2 and 3)

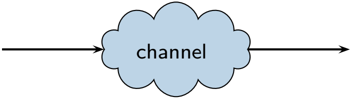
Alexandre Graell i Amat
alexandre.graell@chalmers.se
https://sites.google.com/site/agraellamat

November 6 and 11, 2019

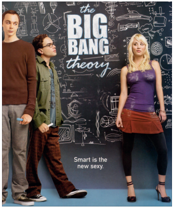
CHALMERS

movie, audio

100110101



satellite link,
fiber, telephone line,
hard-disk drive, DVD



100011101

In This Lectures...

- How do we measure information
- How can we mathematically describe an information source
- Data compression

Discrete Information Source

Discrete Information Source

The information source generates a sequence of symbols that take values on a finite alphabet $\mathcal{X}$.

A Mathematical Model

The output of an information source is a sequence of random variables $X^{(1)}X^{(2)}X^{(3)}\ldots$, at times $t = 1, 2, 3, \ldots$,

- $X^{(i)} \in \mathcal{X} = \{x_1, x_2, \ldots, x_M\}$, $|\mathcal{X}| = M$
- Probability mass function (PMF) $P_X(x)$, $\Pr(X = x_i) = P(x_i) = p_i$,

$$p_i \geq 0 \ \forall i, \quad \text{and} \quad \sum_{x_i \in \mathcal{X}} p_i = 1.$$

The source is completely described by $\mathcal{X}$ and $P_X(x)$.

Discrete Information Source

In This Course...

Discrete memoryless sources, i.e., sources where the random variables $X^{(i)}$ are independent of each other.

Some Notation...

- PMF of random variable X: $P_X(x)$
- Conditional PMF: $P_{Y|X}(y|x)$

We will write $P(x)$ for $P_X(X = x)$ and $P(y|x)$ for $P_{Y|X}(Y = y|X = x)$.

A Measure of Information

- How can we measure the information content of a particular outcome $X = x_i$ from the source?
- How much information does a source contain?

A Measure of Information

Example: Consider the following two claims (about Göteborg's weather)

- This weekend it will rain
- This weekend there will be around 30°C

Which sentence conveys more information?

Discrete Information Source

$\mathcal{X} = \{x_1, x_2, \ldots, x_M\}$, with $p_1 \geq p_2 \ldots \geq p_M$.
Which output conveys more information, x_1 or x_M?

The information associated to a particular message x is related to its uncertainty: The more uncertain the message is, the more information it conveys!

A Measure of Information

A good measure of information associated to a symbol x, $\mathsf{i}(x)$, should:

P1 Be a decreasing function of the probability of the symbol, i.e.,

$$\mathsf{i}(x_i) > \mathsf{i}(x_j) \quad \text{if} \quad p_i < p_j.$$

A Measure of Information

Example: Consider the following three claims

- This weekend it will rain
- This weekend there will be 30°C
- This weekend there will be 30.2°C

Does claim 3 convey much more information than claim 2? No, roughly the same!

Discrete Information Source

$\mathcal{X} = \{x_1, x_2, \ldots, x_M\}$, with $p_1 \geq p_2 \ldots \geq p_M$, and $p_i = 0.25$ and $p_j = 0.25001$.
We would expect that $\mathsf{i}(x_i) \approx \mathsf{i}(x_j)$.

- A slight change in the probability of a symbol should only cause a slight change in the information it conveys $\longrightarrow$ A measure of information should be a continuous function of the probability.
- Two equiprobable symbols carry the same amount of information $\longrightarrow$ A measure of information should depend only on the probability of occurrence.

A Measure of Information

A Good measure of information associated to a symbol x, $\mathsf{i}(x)$, should:

P1 Not depend on the symbol itself, but only on its probability,

$$\mathsf{i}(x_i) = \mathsf{i}(x_j) \quad \text{if} \quad p_i = p_j.$$

P2 Be a continuous, decreasing function of the probability of the symbol,

$$\mathsf{i}(x_i) > \mathsf{i}(x_j) \quad \text{if} \quad p_i < p_j.$$

A Measure of Information

Example: Rolling a dice

Information conveyed by the outcome of rolling a dice once: i_1.
How much information is conveyed by the outcome of rolling a dice twice?
$\mathsf{i}_2 = 2\mathsf{i}_1$!

For two independent random variables X and Y the information we gain when we learn X and Y should equal the sum of the information gained if we learn X and Y separately. $\longrightarrow$ The information should be additive.

A Measure of Information

A Good measure of information associated to a symbol x, $\mathsf{i}(x)$, should satisfy the following three properties:

P1 Not depend on the symbol itself, but only on its probability,

$$\mathsf{i}(x_i) = \mathsf{i}(x_j) \quad \text{if} \quad p_i = p_j.$$

P2 Be a continuous, decreasing function of the probability of the symbol,

$$\mathsf{i}(x_i) > \mathsf{i}(x_j) \quad \text{if} \quad p_i < p_j.$$

P3 For two independent symbols x_i and x_j,

$$\mathsf{i}(x_i, x_j) = \mathsf{i}(x_i) + \mathsf{i}(x_j).$$

Shannon's Information Measure

Definition (Shannon's Information Content)

The Shannon information content of the outcome $X = x_i$ is

$$\mathsf{i}(x_i) = \log_a \frac{1}{p_i}.$$

The base a of the logarithm determines the unit of information. If $a = 2$, the unit of information is the bit. If the base is e, then the information unit is called nat.

Properties

1. ✓
2. ✓
3. For two independent symbols x and y,

$$\begin{aligned}\mathsf{i}(x,y) = \log\frac{1}{P(x,y)} = \log\frac{1}{P(x)P(y)} &= \log\frac{1}{P(x)} + \log\frac{1}{P(y)} \\ &= \mathsf{i}(x) + \mathsf{i}(y).\end{aligned}$$

Average Information Content of a Source

A complete characterization of the source can then be obtained by defining the average information content,

$$\mathsf{H}(X) = \sum_{i=1}^{M} p_i \mathsf{i}(x_i) = \sum_{i=1}^{M} p_i \log \frac{1}{p_i}.$$

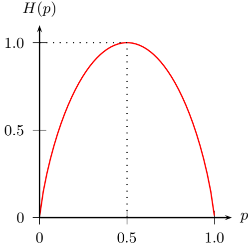
Definition (Entropy)

The entropy of a random variable (random symbol) X that takes values on the alphabet $\mathcal{X} = \{x_1, x_2, \ldots, x_M\}$ with probabilities $p_1, p_2, \ldots, p_M$ is defined as

$$\mathsf{H}(X) = \sum_{x \in \mathcal{X}} P(x) \log \frac{1}{P(x)} = \sum_{i=1}^{M} p_i \log \frac{1}{p_i}.$$

The entropy measures the average information content (or *uncertainty*) of X.

Average Information Content of a Source



Definition (The Binary Entropy Function)

Let X be a binary source with two possible values $\mathcal{X} = \{x_1, x_2\}$ such that $P(x_1) = p$ and $P(x_2) = 1 - p$. Then

$$\mathsf{H}(X) = \mathsf{H_b}(p),$$

where $\mathsf{H_b}(p)$ is called the binary entropy function,

$$\mathsf{H_b}(p) \triangleq p \log \frac{1}{p} + (1-p) \log \frac{1}{1-p}.$$

Average Information Content of a Source

Lemma

The entropy of a random variable X that takes values on the alphabet $\mathcal{X} = \{x_1, \ldots, x_M\}$, $|\mathcal{X}| = M$, is bounded by

$$0 \leq \mathsf{H}(X) \leq \log M \quad \textit{bits},$$

where

$$\mathsf{H}(X) = 0 \qquad \textit{if and only if} \;\; p_i = 1 \;\; \textit{for some} \;\; i,$$

$$\mathsf{H}(X) = \log M \qquad \textit{if and only if} \;\; p_i = \frac{1}{M} \;\; \forall i.$$

Data Compression

Wh_ c_n y_u r_co_er this q_est_on?

Most information sources contain redundancy!

Data (Source) Compression

Find an efficient representation of the output of the source which results in zero or little redundancy $\longrightarrow$ Encode (map) the symbols (or messages) at the output of the source to a sequence of symbols, called codeword, that is shorter in average.

Source Encoder

- Can encode source symbols separately or groups of symbols.
- Typically requires knowledge of the statistics of the source
- $\longrightarrow$ We can achieve compression on average by assigning shorter codewords to more probable symbols and longer codewords to less likely ones. (Variable-length encoder).

Data Compression

- How much can we compress such that we lose no information?
- Are there efficient methods to assign codewords to source symbols?
- How can we make sure that the source code is easy to decode?

Data Compression

symbol	probability	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$
a	$1/2$	000	00	00	0	0	0
b	$1/4$	001	00	01	01	01	10
c	$1/8$	101	10	10	001	011	110
d	$1/8$	111	11	11	111	111	111

Example: Alphabet $\{a, b, c, d\}$ with probabilities $\{1/2, 1/4, 1/8, 1/8\}$

We would like to design a good (binary) source code to compress the language produced by this alphabet.

- $\mathcal{C}_1$ Not efficient
- $\mathcal{C}_2$ Not uniquely decodable
- $\mathcal{C}_3$ Uniquely decodable, not efficient
- $\mathcal{C}_4$ Not uniquely decodable
- $\mathcal{C}_5$ Uniquely decodable, not instantaneous
- $\mathcal{C}_6$ Uniquely decodable, instantaneous (prefix-free)

Example: The Morse Code

The Morse code assigns the most frequent letters to shorter codewords, and less frequent letters to longer codewords:

- "e" $\longrightarrow$ "$\cdot$"
- "q" $\longrightarrow$ "$- - \cdot -$"

Data Compression

A source code should satisfy the following properties:

- P1 The code must be uniquely decodable, i.e., the symbols generated by the source must be uniquely retrieved from the encoded string of bits.
- P2 The code should be easy to decode.
- P3 The code should compress the source as much as possible.

Data Compression

Definition

Given a set Σ, Σ^+ denotes the set of all strings over Σ of any (nonzero) finite length.

Example

If $\Sigma = \{a, b, c\}$ then a, ab, aac and $bbac$ are strings over Σ of lengths one, two, three and four respectively.

Data Compression

Definition (Binary Source Code)

Consider a random variable X which takes values on $\mathcal{X} = \{x_1, \ldots, x_M\}$.

- A binary encoder $\mathcal{E}$ is a function $\mathcal{E} : \mathcal{X} \to \{0,1\}^+$ that maps each source symbol $x_i \in \mathcal{X}$ to a binary sequence $\boldsymbol{c}_i \in \{0,1\}^+$ (codeword). The number of bits in $\boldsymbol{c}_i$ is called its length and is denoted by ℓ_i.
- The binary source code $\mathcal{C}$ is the set of all codewords, i.e., $\mathcal{C} = \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_M\}$.

Example: Code $\mathcal{C}_6$

x_i		$\boldsymbol{c}_i$	ℓ_i
a	$\to$	0	1
b	$\to$	10	2
c	$\to$	110	3
d	$\to$	111	3

Definition (Extended Code)

The extended code $\mathcal{C}^+$ is the set of binary sequences resulting from the mapping $\mathcal{E}^+ : \mathcal{X}^+ \to \{0,1\}^+$ from a sequence of symbols in the alphabet $\mathcal{X}$ to a binary sequence obtained by concatenating the corresponding codewords, such that, for a sequence of symbols of length n, at times $t = 1, \ldots, n$,

$$\boldsymbol{c}^+(x^{(1)}x^{(2)} \ldots x^{(n)}) = \boldsymbol{c}(x^{(1)})\boldsymbol{c}(x^{(2)}) \ldots \boldsymbol{c}(x^{(n)}),$$

Example: Extended Code $\mathcal{C}_6^+$

$$\begin{array}{ccc} aab & \to & 0010 \\ bca & \to & 101100 \end{array}$$

Uniquely Decodable Code

P1 The code must be uniquely decodable, i.e., the symbols generated by the source must be uniquely retrieved from the encoded string of bits.

Definition (Uniquely Decodable Code)

A code $\mathcal{C}$ is uniquely decodable if

$$\forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}^+, \boldsymbol{x} \neq \boldsymbol{y} \Longrightarrow \boldsymbol{c}^+(\boldsymbol{x}) \neq \boldsymbol{c}^+(\boldsymbol{y}).$$

Example

symbol	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$
a	0	0	0
b	01	01	10
c	001	011	110
d	111	111	111

$\mathcal{C}_4$ is not uniquely decodable: $ab \rightarrow 001$ and $c \rightarrow 001$.

Prefix-Free Code

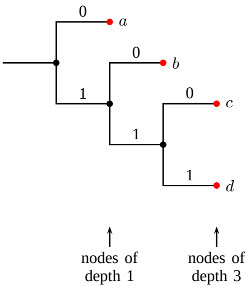
P2 The code should be easy to decode (decode each codeword immediately at the arrival of the last bit of that codeword) $\longrightarrow$ no codeword can be a prefix of another codeword.

Definition (Prefix-Free Code)

A code is called a prefix-free code if no codeword is the prefix of any other codeword.

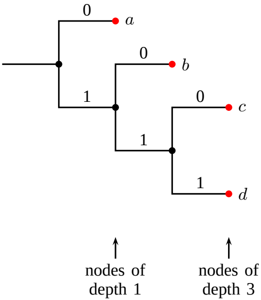
- We can decode codewords instantaneously, i.e., as soon as they are received $\longrightarrow$ instantaneous code
- Prefix-free code $\Longrightarrow$ Uniquely decodable

Binary Code Tree



- Each node has two descendants
- Each node in the tree represents the binary string corresponding to the branches from the root to the node
- Every codeword can be represented by a particular path through the tree

Binary Code Tree



symbol	$\mathcal{C}_6$
a	0
b	10
c	110
d	111

For prefix-free codes...

- Every codeword corresponds to a leaf
- Intermediate nodes correspond to prefixes of some codewords

The Kraft Inequality

Theorem (Kraft Inequality)

There exists a binary prefix-free code of cardinality M and codeword lengths $\ell_1, \ell_2, \ldots, \ell_M$ if and only if

$$\sum_{i=1}^{M} 2^{-\ell_i} \leq 1.$$

The Kraft Inequality

Proof (Necessity: Prefix-free $\Longrightarrow \sum_{i=1}^{M} 2^{-\ell_i} \leq 1$):

1. Prefix-free code $\longrightarrow$ can be described by a binary tree with codewords corresponding to leaves.
2. Let $\ell_{\mathsf{max}} = \max \{\ell_1, \ell_2, \ldots, \ell_M\}$, and expand the tree so that all branches have depth ℓ_{max}.
3. If expanded, a leaf (codeword) at depth ℓ_i would lead to $2^{\ell_{\mathsf{max}} - \ell_i}$ leaves at depth ℓ_{max}.
4. We must satisfy:
$$\sum_{i=1}^{M} 2^{\ell_{\mathsf{max}} - \ell_i} \leq 2^{\ell_{\mathsf{max}}}$$
5. Divide both sides by $2^{\ell_{\mathsf{max}}}$,
$$\sum_{i=1}^{M} 2^{-\ell_i} \leq 1.$$

The Kraft Inequality

Proof (Sufficiency: $\sum_{i=1}^{M} 2^{-\ell_i} \leq 1 \Longrightarrow$ Prefix-free):

Assume that the codeword lengths are sorted in increasing order, $\ell_1 \leq \ell_2 \leq \ldots \leq \ell_M$.
We can construct a prefix-free code as follows.

1. Start with a complete tree where all leaves are at depth ℓ_M. For each node at depth i we have two nodes at depth $i+1$.
2. Choose a free node at depth ℓ_1 for the first codeword, and remove all its descendants (this removes $2^{\ell_M - \ell_1}$ leaves at depth ℓ_M). Do the same for ℓ_2 and codeword 2. This removes $2^{\ell_M - \ell_2}$ leaves at depth ℓ_M. Continue for ℓ_3, etc., until we have placed all codewords.

The Kraft Inequality

Proof (Sufficiency: $\sum_{i=1}^{M} 2^{-\ell_i} \leq 1 \Longrightarrow$ Prefix-free):

We need to prove that at every step i there are free leaves to be assigned to a codeword.

1. After assigning a node at depth j to a codeword, the number of remaining leaves at depth M is

$$2^{\ell_M} - \sum_{i=1}^{j} 2^{\ell_M - \ell_i}$$

2. Now,

$$2^{\ell_M} - \sum_{i=1}^{j} 2^{\ell_M - \ell_i} = 2^{\ell_M} \left(1 - \sum_{i=1}^{j} 2^{-\ell_i}\right) > 2^{\ell_M} \left(1 - \sum_{i=1}^{M} 2^{-\ell_i}\right) \geq 0.$$

Kraft-McMillan's Inequality

Prefix-Free Codes

Very nice properties: Uniquely decodable and easy to decode.

However...is there a non prefix-free uniquely-decodable code that achieves lower average codeword length?

Theorem (Kraft-McMillan's Inequality)

A uniquely decodable code with codeword lengths $\ell_1, \dots, \ell_M$ exists if and only if

$$\sum_{i=1}^{M} 2^{-\ell_i} \leq 1.$$

$\Longrightarrow$ We can restrict ourselves to prefix-free codes with no loss in performance!

Efficient Codes

P3 The code should compress the source as much as possible.

Our goal will be to minimize the average number of code bits per source symbol (the average codeword length),

$$\bar{L} = \sum_{i=1}^{M} p_i \ell_i.$$

Example (Code $\mathcal{C}_6$)

Code $\mathcal{C}_6$ has average codeword length

$$\bar{L} = \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + 2 \cdot \frac{1}{8} \cdot 3 = 1.75 \text{ bits.}$$

The Limits of Source Coding

- What is the best achievable compression?

Theorem

The average codeword length of a uniquely decodable code is lower bounded by

$$\bar{L} \geq \mathsf{H}(X).$$

Proof: We will prove that $\mathsf{H}(X) - \bar{L} \leq 0$.

$$\begin{aligned}
\mathsf{H}(X) - \bar{L} &= \sum_{i=1}^{M} p_i \log \frac{1}{p_i} - \sum_{i=1}^{M} p_i \ell_i \\
&= \sum_{i=1}^{M} p_i \left[\log \frac{1}{p_i} - \log 2^{\ell_i} \right] = \sum_{i=1}^{M} p_i \left[\log \frac{1}{p_i} + \log 2^{-\ell_i} \right] \\
&= \sum_{i=1}^{M} p_i \log \frac{2^{-\ell_i}}{p_i}.
\end{aligned}$$

Proof (Cont'd):

$$\mathsf{H}(X) - \bar{L} = \sum_{i=1}^{M} p_i \log \frac{2^{-\ell_i}}{p_i} = \frac{1}{\ln 2} \sum_{i=1}^{M} p_i \ln \frac{2^{-\ell_i}}{p_i}$$

$$\overset{(\ln x \leq x-1)}{\leq} \frac{1}{\ln 2} \sum_{i=1}^{M} p_i \left(\frac{2^{-\ell_i}}{p_i} - 1 \right)$$

$$= \frac{1}{\ln 2} \left(\sum_{i=1}^{M} 2^{-\ell_i} - \underbrace{\sum_{i=1}^{M} p_i}_{=1} \right) = \frac{1}{\ln 2} \left(\sum_{i=1}^{M} 2^{-\ell_i} - 1 \right) \leq 0.$$

- $\bar{L} = \mathsf{H}(X)$ if and only if the codeword lengths satisfy $\ell_i = \log \frac{1}{p_i}$ $\forall i$.

The Limits of Source Coding

- How close to the entropy can we expect to compress?

Theorem (Source Coding Theorem for a Single Random Symbol)

Let X be a random variable generated by a discrete memoryless source with entropy $\mathsf{H}(X)$. There exists a prefix-free code $\mathcal{C}$ with average codeword length satisfying

$$\mathsf{H}(X) \leq \bar{L}(\mathcal{C}) < \mathsf{H}(X) + 1.$$

The Limits of Source Coding

Proof:

1. Set the codeword lengths to

$$\ell_i = \left\lceil \log \frac{1}{p_i} \right\rceil.$$

2. Check that there is a prefix-free code with these lengths,

$$\sum_{i=1}^{M} 2^{-\ell_i} = \sum_{i=1}^{M} 2^{-\left\lceil \log \frac{1}{p_i} \right\rceil} \leq \sum_{i=1}^{M} 2^{-\log \frac{1}{p_i}} = \sum_{i=1}^{M} p_i = 1.$$

3. Upperbound om $\bar{L}(\mathcal{C})$,

$$\begin{aligned}\bar{L}(\mathcal{C}) &= \sum_{i=1}^{M} p_i \left\lceil \log \frac{1}{p_i} \right\rceil < \sum_{i=1}^{M} p_i \left(\log \frac{1}{p_i} + 1 \right) = \sum_{i=1}^{M} p_i \log \frac{1}{p_i} + \sum_{i=1}^{M} p_i \\ &= \mathsf{H}(X) + 1.\end{aligned}$$

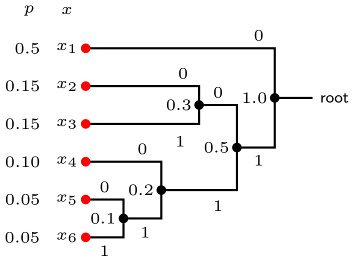
Optimal Source Coding: Huffman Coding

- Given a probability distribution P_X, how can we design an optimal prefix-free code? (i.e., with minimum $\bar{L}$)

The Huffman Coding Algorithm

1. Order the M source symbols in decreasing order of their probabilities
2. Create M leaves, one for each symbol
3. Combine the two leaves corresponding to the two least probable symbols into a new node. Assign to this node the sum of the probabilities of the two original nodes
4. Repeat Step 2 until only one node is free, and root it
5. Starting from the root of the obtained tree, assign the binary symbols 0 and 1 to each pair of branches that arise from each node. The codeword for each symbol is read as the binary sequence from the root to the leaf associated to the symbol

Optimal Source Coding: Huffman Coding



symbol	codeword
x_1	0
x_2	100
x_3	101
x_4	110
x_5	1110
x_6	1111

Example (Huffman Coding)

Coding of $\mathcal{X} = \{x_1, x_2, x_3, x_4, x_5, x_6\}$, $P_X = \{0.5, 0.15, 0.15, 0.10, 0.05, 0.05\}$.
$\mathrm{H}(X) = 2.08548$ bits. $\bar{L} = 2.1$ bits.

Encoding Blocks of Symbols

What We Have Achieved up to Now...

Source compression by encoding each output of the source independently using an optimal prefix-free (Huffman) code.

- Is there a more efficient approach?

Encoding Blocks of Symbols

We can do better than encoding each symbol separately! $\longrightarrow$ Encoding blocks of symbols.

Encoding Blocks of Symbols

Source $\longrightarrow X \in \mathcal{X} = \{x_1, x_2, \dots, x_M\}$

Source $\longrightarrow Y \in \mathcal{Y} = \{y_1, y_2, \dots, y_{M^\nu}\} \sim \mathcal{X}^\nu$

1. Combine ν consecutive symbols at the output of the source in a new symbol (message),

$$(x^{(1)}, x^{(2)}, \dots, x^{(\nu)}),$$

   where $x^{(j)} \in \mathcal{X}$.
2. Encode this message using a Huffman code.

We design a code for an equivalent source Y with alphabet $\mathcal{Y} = \{y_1, y_2, \dots, y_{M^\nu}\}$, $|\mathcal{Y}| = M^\nu$, consisting of all possible tuples $(x^{(1)}, x^{(2)}, \dots, x^{(\nu)})$ of length ν.
For a memoryless source, the probability of a message y_i is

$$p_i = \prod_{j=1}^{\nu} p_{i,j},$$

where $p_{i,j}$ is the probability of the jth symbol of y_i.

Encoding Blocks of Symbols

$$\boxed{\text{Source}} \longrightarrow Y \in \mathcal{Y} = \{y_1, y_2, \ldots, y_{M^\nu}\} \sim \mathcal{X}^\nu$$

From the Source Coding Theorem for a Single Random Symbol...

We can build a code for Y with average length $\bar{L}_\nu$,

$$\mathsf{H}(Y) \leq \bar{L}_\nu < \mathsf{H}(Y) + 1.$$

How do we compare $\bar{L}_\nu$ for different values of ν?

The average codeword length necessary to describe one source symbol, $\bar{L}_\nu/\nu$!

Encoding Blocks of Symbols

$$\boxed{\text{Source}} \longrightarrow Y \in \mathcal{Y} = \{y_1, y_2, \ldots, y_{M^\nu}\} \sim \mathcal{X}^\nu$$

We can write

$$\frac{\mathsf{H}(Y)}{\nu} \leq \frac{\bar{L}_\nu}{\nu} < \frac{\mathsf{H}(Y)+1}{\nu}.$$

Since a message Y is built from ν independent symbols X,

$$\mathsf{H}(Y) = \nu \mathsf{H}(X).$$

Thus,

$$\mathsf{H}(X) \leq \frac{\bar{L}_\nu}{\nu} < \mathsf{H}(X) + \frac{1}{\nu}.$$

The Source Coding Theorem

Theorem (Source Coding Theorem)

Let X be a random variable generated by a discrete memoryless source with entropy $\mathsf{H}(X)$*. There exists a prefix-free code $\mathcal{C}$ that encodes messages of length ν symbols with average codeword length per source symbol $\frac{\bar{L}_\nu}{\nu}$ satisfying*

$$\mathsf{H}(X) \leq \frac{\bar{L}_\nu}{\nu} < \mathsf{H}(X) + \frac{1}{\nu}.$$

Choosing ν large enough we can approach the ultimate limit of compression, $\mathsf{H}(X)$, arbitrarily closely using Huffman codes!

However...encoding large blocks of symbols requires long codes $\longrightarrow$ difficult to construct, delay.

Efficiency of a Source Code

$$\eta = \frac{\nu \mathsf{H}(X)}{\bar{L}_\nu}, \quad 0 \leq \eta \leq 1.$$

The Source Coding Theorem

$\boxed{\text{Source}} \longrightarrow X \in \{x_1, x_2, x_3\}$ $\quad$ $\boxed{\text{Source}} \longrightarrow Y \in \{y_1, y_2, \ldots, y_9\} = \{x_1x_1, x_1x_2, \ldots, x_3x_3\}$

Example: Source X, $\mathcal{X} = \{x_1, x_2, x_3\}$, with probabilities $\{0.45, 0.35, 0.20\}$. $\mathsf{H}(X) = 1.513$ bits.

$\mathcal{C}_1$: Encode symbols separately using a Huffman code. $\bar{L} = 1.55$ bits, $\eta = 0.976$.

$\mathcal{C}_2$: Encode pairs of symbols, i.e., we have equivalent source $Y = (X^1, X^2)$, with $\mathcal{Y} = \{y_1, y_2, \ldots, y_9\} = \{x_1x_1, x_1x_2, x_1x_3, x_2x_1, x_2x_2, x_2x_3, x_3x_1, x_3x_2, x_3x_3\}$, $|\mathcal{Y}| = |\mathcal{X}|^2 = 9$, and encode messages Y using a Huffman code. $\frac{\bar{L}_\nu}{\nu} = 1.53375$ bits, $\eta = 0.989$.

The Source Coding Theorem

Table: Huffman code, encoding symbols separately, $\bar{L} = 1.55$, $\eta = 0.976$

symbol	probability	codeword
x_1	0.45	1
x_2	0.35	00
x_3	0.20	01

Table: Huffman code, encoding blocks of two symbols, $\bar{L}_\nu = 3.0675$, $\eta = 0.989$

symbol	probability	codeword
x_1x_1	0.2025	10
x_1x_2	0.1575	001
x_2x_1	0.1575	010
x_2x_2	0.1225	011
x_1x_3	0.09	111
x_3x_1	0.09	0000
x_2x_3	0.07	0001
x_3x_2	0.07	1100
x_3x_3	0.04	1101

Further Discussion

Source Compression using Huffman Codes

- Requires knowledge of the source statistics.
- Lossless: we can always recover the transmitted sequence of data symbols from the coded sequence with no loss of information.

Universal Source Compression

A source compression that achieves the ultimate limit regardless of the source: The Lempel-Ziv compression algorithm (gzip,GIF), proven asymptotically to compress down to the entropy of the source.

Lossy Source Compression

- Compression beyond the entropy of the source $\longrightarrow$ we cannot recover the original data identically after decompression.
- Typically closely related to the human perception.
- Used to compress sound, video or image (jpeg compression).