Chapter 2: Source coding

Vahid Meghdadi

meghdadi@ensil.unilim.fr
University of Limoges
Markov source

Entropy of Markov Source

Source coding

Definitions
Shannon-Fano algorithm
Huffman coding
Lempel-Ziv coding
Markov model for information sources

Given the present, the future is independent of the past. This fact can be presented as follows. For a given source producing $X_1$, $X_2$ and $X_3$, we can always write:

$$P(X_1, X_2, X_3) = P(X_1)P(X_2|X_1)P(X_3|X_1, X_2)$$

However for a Markov source, the history of the source is saved in its present state:

$$P(X_k = s_q|X_1, X_2, ..., X_{k-1}) = P(X_k = s_q|S_k)$$

It means that the state contains all the past of the source.
Markov source modeling

- The source can be in one of $n$ possible states.
- At each symbol generation, the source changes its state from $i$ to $j$.
- This state change is done with the probability $p_{ij}$ which depends only on the initial state $i$ and the final state $j$ and remains constant over the time.
- The generated symbol depends only on the initial state and the next state ($i$ and $j$).
- The transition matrix $P$ is defined where $[p_{ij}]$ is the probability of transmission from state $i$ to state $j$. 
Markov source, example

- A symbol is sent out at each transition.
- What is the probability of message "A"?
- What is the probability of message "B"?
- What is the probability of message "AB"?
- Give the transition matrix.
Entropy of a Markov source

Each state can be considered as a source. For this source the entropy can be calculated. We average then these entropies over all the states to find the entropy of the whole source, i.e.:

$$H = \sum_{i=1}^{n} P_i H_i$$

Where \( n \) is the number of states and :

$$H_i = \sum_{j=1}^{n} P_{ij} \log_2(1/P_{ij})$$

The information rate will be then:

$$R = r_s H$$
Example

For the following example calculate

▶ the entropy of the source
▶ the information per symbol that is contained in the messages of size 1, 2 and 3 symbols
▶ is it true that the information for longer messages is less?
Suppose the messages $m_i$ of size $N$ at the output of a Markov source. Because the information content of sequence $m_i$ is $-\log_2 p(m_i)$, the mean of information per symbol is:

$$G_N = -\frac{1}{N} \sum_i p(m_i) \log_2 p(m_i)$$

It can be shown that $G_N$ is a monotone decreasing function of $N$, and:

$$\lim_{N \to \infty} G_N = H \text{ bits/symbol}$$
Application for compression.

- Suppose that there is a source modeled by Markov model.
- If we find the statistic for the sequences of one symbol, the correlation between the consecutive symbols is not exploited.
- So the information seems to be more than reality.
- If we consider the sequences of $n$ symbols, there is more correlation taken into account, so entropy seems to be lessen and nearer to real source entropy.

**Conclusion:** to better compress, we should code the longer sequences.
Two types of compression can be considered:

- **Lossless coding**: the information can be reconstructed exactly from coded sequence (Shannon, Huffman, LZW, PKZIP, GIF, TIFF,...)

- **Lossy coding**: Information loss happens in coding process (JPEG, MPEG, Wavelet, transform coding, sub band coding,...)

In this course we only consider the first one. We suppose furthermore that the sequences at the output of encoder are binary.
Definition
A source code $C$ for a random variable $X$ is a mapping from $\mathcal{X}$, the range of $X$, to $\mathcal{D}$, the set of finite length strings of symbols from a $D$-ary alphabet. $C(x)$ denotes the codeword corresponding to $x$ and $l(x)$ denotes the length of $C(x)$.

Example
If you toss a coin, $\mathcal{X} = \{\text{tail, head}\}$, $C(\text{head}) = 0$, $C(\text{tail}) = 11$, $l(\text{head}) = 1$, $l(\text{tail}) = 2$. 
Definition

The expected (average) length of a code $C(x)$ for a random variable $X$ with probability mass function $p(x)$ is given by

$$L(C) = \sum_{x \in \mathcal{X}} p(x)l(x)$$

For example for the above example the expected length of the code is

$$L(C) = \frac{1}{2} \times 2 + \frac{1}{2} \times 1 = 1.5$$
Definition
The code is singular if $C(x_1) = C(x_2)$ and $x_1 \neq x_2$.

Definition
The extension of a code $C$ is a code obtained as:

$$C(x_1x_2 \ldots x_n) = C(x_1)C(x_2) \ldots C(x_n)$$

It means that a long message can be coded by concatenating the shorter message code words. For example if $C(x_1) = 11$ and $C(x_2) = 00$, then $C(x_1x_2) = 1100$. 
Definitions (4)

Definition
A code is uniquely decodable if its extension is non-singular. In other words, any encoded string has only one possible source string and there is no ambiguity.

Definition
A code is called a *prefix code* or an *instantaneous code* if no code word is a prefix of any other codeword.
Example (1)

Example
The following code is a prefix code:
\( C(x_1) = 1, C(x_2) = 01, C(x_3) = 001, C(X_4) = 000 \). Any encoded sequence is uniquely decodable and its corresponding source word can be obtained as soon as the code word is received. In other word, an instantaneous code can be decoded without reference to the future codewords since the end of a codeword is immediately recognizable. For example the sequence 001100000001 is decoded as \( x_3x_1x_4x_4x_2 \).
Example

The following code is not instantaneous code but uniquely decodable: $C(x_1) = 1$, $C(x_2) = 10$, $C(x_3) = 100$, $C(X_4) = 000$. Why? Here you should wait to receive a 1 to be able to decode. Note that if we look at the encoded sequence from right to left, it becomes instantaneous.
Kraft-McMillan Inequality

For any uniquely decodable code $C$: \[ \sum_{w \in C} D^{-l(w)} \leq 1 \]
where $w$ is a codeword in $C$ and $l(w)$ is its length, $D$ is the size of alphabet, it is 2 for binary sequences.

**Lemma**

*For any message set $X$ with a mass probability function and associated uniquely decodable code $C$

\[ H(X) \leq L(C) \]

The proof uses Jensen inequality for concave logarithmic function and Kraft-MacMillan inequality: \[ \sum_i p_i f(x_i) \leq f(\sum_i p_i x_i). \]
Kraft-McMillan Inequality example

Here $D = 2$. There are 4 codewords $A=1$, $B=01$, $C=001$, $D=000$, with $l_i = 1, 2, 3$ and 3. So $\sum_{w \in C} D^{-l(w)} = 1$. If all of the branches are used the code is complete and the equality is obtained, as above. But, if the tree is not complete, for example ”C” in the figure is not used in the code, then the equality cannot be obtained in Kraft inequality.
Results

- The average code length is lower bounded by the source entropy.
- It can be shown that there is an upper bound based on entropy for optimal prefix code:

\[ L(c) \leq H(X) + 1 \]
Shannon-Fano algorithm (1)

A systematic method to design the code

- The input of the encoder is one of the $q$ possible sequences of size $N$ symbols: $m_i$.
- The corresponding probabilities are $p_1, p_2, \ldots, p_q$.
- The encoder transforms $m_i$ to a binary sequence $c_i$ trying to minimize the average output bits $L(C)$.
- The average number of bits per symbol at the output can be calculated as: $L(C) = (1/N) \sum_{i=1}^{q} n_i p_i$ bits/symbol where $n_i$ is the number of bits in the coded sequence $c_i$.
- For a good encoder $L(C)$ should be very close to the input entropy: $G_N = (1/N) \sum_{i=1}^{q} p_i \log(1/p_i)$.
The idea is to assign shorter codes to more probable messages. It is a variable length code.

**Theorem**

*If C is an optimal prefix code for the probabilities \( \{p_1, p_2, \ldots, p_n\} \), then \( p_i > p_j \) implies that \( l(c_i) \leq l(c_j) \).*
Shannon-Fano algorithm(3)

1. Order the messages from the most probable to the least probable, from $m_1$ to $m_q$.
2. Put these messages in the first column of a table.
3. In the second column put $n_i$ such that
   \[ \log_2\left(\frac{1}{p_i}\right) \leq n_i < 1 + \log_2\left(\frac{1}{p_i}\right) \]
4. In the third column, write $F_i = \sum_{k=1}^{i-1} p_k$ with $F_1 = 0$.
5. In the forth column, write down the binary representation of $F_i$ for $n_i$ bits. This column gives directly the code corresponding to the messages of the first column.
Shannon-Fano algorithm properties

- This is a uniquely decodable prefix code.
- The more probable messages are coded with shorter codes.
- The codewords are distinct.
- The average number of bits per symbol at the output is
  \[ G_N \leq L(C) < G_N + \frac{1}{N} \]
  - When \( N \to \infty \), \( G_N \to L(C) \) and \( L(C) \to H \).
  - The code performance is quantified by \( e = H/L(C) \).
For the above Markov source, give the code for

1. the messages of size one symbol,
2. the messages of size two symbols,
3. the messages of size three symbols.

Show that the code performance is:

\[ e_1 = 40.56\%, \quad e_2 = 56.34\% \quad \text{and} \quad e_3 = 62.40\% \]
Huffman coding constructs the coding tree with a systematic method. Suppose that the messages are ordered with their probabilities, \( m_1 \) is the most probable and \( m_q \) the least one.

We consider the two last messages and we sum up their probabilities to obtain a new message. We construct at the same time the tree. Then, with new messages we continue as before.
Huffman algorithm example

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0.25</td>
</tr>
<tr>
<td>b</td>
<td>0.2</td>
</tr>
<tr>
<td>c</td>
<td>0.15</td>
</tr>
<tr>
<td>d</td>
<td>0.12</td>
</tr>
<tr>
<td>e</td>
<td>0.1</td>
</tr>
<tr>
<td>f</td>
<td>0.1</td>
</tr>
<tr>
<td>g</td>
<td>0.08</td>
</tr>
</tbody>
</table>

Huffman encoding:
- a: 01
- b: 11
- c: 001
- d: 101
- e: 100
- f: 0001
- g: 0000

Huffman tree:
```
          1
         /|
       0.58 1
      /    /
   0.33 0.42
 /     /   /
 a:1   c:0.25  d:0.22
 /     /    /
 g:0.08 f:0.1 e:0.1 d:0.12 b:1
```

Vahid Meghdadi
Chapter 2: Source coding
Lempel-Ziv algorithm

- Huffman encoding is optimal (average block length is minimal).
- Huffman coding requires the probability distribution of source.
- In Huffman coding, the input sequence is fixed length but the output is variable length.
- In LZ coding, the knowledge of probabilities is not required.
- LZ coding therefore belongs to the class of universal source coding.
- In LZ coding, the input sequence can be variable length but the output is fixed length.
Dictionary construction

In this algorithm we construct "on the fly" a dictionary. That’s why the code is called dictionary coding. Suppose the binary message 1010110100100111010100011001110101100011011... to be coded.

There is no entry in our table yet. The encoder constructs the first entry of its table with the first letter in the sequence, 1, with the location in the table 1.

The second letter, which is a 0, does not belong to the table, so it is added with the location 2.

The next letter is a 1 which is already in the table, so the encoder continues to receive the next letter. Now 10 is not in the table, that will be added with the index 3.
For the example given the dictionary for the sequence
1010110100101101010001100110101100011011 will be:

1, 0, 10, 11, 01, 00, 100, 111, 010, 1000, 011, 001, 110, 101, 10001, 1011

Note that each phrase is a concatenation of a previous phrase in the table with the new letter to be appended.
To encode the sequence, the codewords are the position of the phrase in the dictionary with the new letter appended to it. Initially, the location 0000 is used to encode a phrase that has not appeared previously.
Encoding

Assuming the codewords of length 5, the constructed dictionary together with the codewords are presented in the following table:

<table>
<thead>
<tr>
<th>location</th>
<th>content</th>
<th>codeword</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0001</td>
<td>00001</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
<td>00000</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
<td>00010</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
<td>00011</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
<td>00101</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
<td>00100</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
<td>00110</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>15</td>
<td>1111</td>
<td>10101</td>
</tr>
</tbody>
</table>
In decoding the table must be constructed also "on the fly". For example for the previous sequence, the receiver must know that the codeword length is 5. It receives a 00001. So the entry of the table is the 1. This is the decoded sequence and it puts in its table the 1 in the first location. Then it receives a 00000 which says that the second entry of the table is a 0 and the decoded phrase will be 0. The third word is 00010 which means that 0001 is appended to 0. The decoded phrase will be the phrase corresponding to the first row, 1 with a 0 at the end: 10, and so on ...