Sécurité des systèmes d'information
Exercise sheet 2 : Entropy

02 Octobre 2019

Non-mandatory exercise sheet. Please upload your answers on Moodle before Monday **10/07/2019 17h15**.

All answers should be carefully justified.

Entropy theory

We consider a source of information represented by a random variable X on an alphabet of n symboles, each letter x_i having probability p_i. The entropy of this source, written $H(X)$, is defined as :

$$H(X) = -\sum_{i=1}^{n} p_i \log_2(p_i) = \sum_{i=1}^{n} p_i \log_2(\frac{1}{p_i})$$

This entropy is maximum when $p_i = \frac{1}{n}$ for all n. It corresponds to the maximal randomness of the source.

The joint entropy for two random variables X and Y, written $H(X, Y)$ is simply the entropy taken on every possible pairs (x,y) in the joint alphabet :

$$H(X, Y) = -\sum_{x \in X} \sum_{y \in Y} p(x, y) \log_2(p(x, y))$$

Finally, the conditional entropy of X given Y is given by :

$$H(X|Y) = -\sum_{y \in Y} \sum_{x \in X} p(y)p(x|y) \log_2(p(x|y))$$

The conditional entropy gives very insightful informations in Cryptography. For instance, the value $H(\text{Plaintext} \mid \text{Ciphertext})$ measures exactly the efficiency of a given encryption technique, by telling how hard it is to guess the original text given the encrypted version.

Huffman Coding

The Huffman coding is a way to modify the encoding of a given piece of information in order to use a lesser number of bits without loosing any information (Think for instance about the Morse code, which is a way to rewrite english language that takes into account the fact that 'e' is much more frequent than 'x').

We measure the efficiency of a coding procedure by the codeword length :

$$L(C) = \sum_{x \in X} p(x) \cdot \#\{\text{bits used to code the letter x}\}$$

The goal is to construct a binary tree in the following way : sort the variables by frequency of apparition, then merge the two less frequent variables in a new one. Repeat this process until you only have two variables left.

Example : Let A,B,C,D be the alphabet with probabilites 0.2, 0.5, 0.2, 0.1 respectively.

- We first sort the variables by frequency : B, A, C, D.
- We then merge the last two variables in a new one :
  $E \leftarrow C, D$; $P(E) = 0.3$.
- We sort again : B, E, A.
- We merge the last two variables :
  $F \leftarrow E, A$; $P(F) = 0.5$.
- Final sort : B, F.

We thus obtain the following code :

- $B = 0$, $F = 1$.
- We then split the merged variables : $F \rightarrow E, A$; $E = 10$, $A = 11$.
- And again : $E \rightarrow C, D$; $C = 100$, $D = 101$.
- We thus obtain $B = 0$, $A = 11$, $C = 100$, $D = 101$

The main motivation being to reduce the size of the most frequent elements. Note that a naive coding would lead us to write :

$$A = 00,\ B = 01,\ C = 10,\ D = 11$$

that gives a codeword length of 2 bits whereas :

$$L(\text{Huffman Coding}) = 0.5 \cdot 1 + 0.2 \cdot 2 + 0.2 \cdot 3 + 0.1 \cdot 3 = 1.8$$

A celebrated result of information theory gives a lower bound on the minimal codeword length theorically achievable. This is known as the Shannon's source coding theorem :

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

For every possible coding C.

In our example, $H(X) \approx 1.76$ is indeed a lower bound.

Finally, we mention a result saying that the Huffman coding is almost optimal :

$$H(X) \leq L(\text{Huffman coding}) \leq H(X) + 1$$

Exercise 1 : Entropy exploration

- Let X be a bernoulli random variable, taking values 0 and 1, with probability $\mathbb{P}(X = 1) = p$. We let p vary from 0 to 1. Draw the graph of the value $H(X)$ as a function of p.
- Let X be any random variable on a fixed alphabet. How do you interpret the result $H(X) = 0$? What is the maximum value that H(X) can take on this alphabet ?

Exercise 2 : Entropy computation

1. By hand :

   We consider the following simple symmetric cryptography system : The set of plaintexts $P = \{m_1, m_2, m_3\}$, the set of ciphertexts $C = \{1, 2, 3, 4, 5\}$, and the set of keys $K = \{k_1, k_2, k_3\}$, with the following encoding rule :

	m_1	m_2	m_3
k_1	3	2	1
k_2	4	5	2
k_3	1	4	3

   We suppose that we use each key with the same probability $\frac{1}{3}$. The plaintexts have the following probability of apparition :

   $$p(m_1) = \frac{1}{4} \quad p(m_2) = \frac{3}{20} \quad p(m_3) = \frac{6}{10}$$

   Compute the following entropies : $H(P)$, $H(K)$, $H(C)$, $H(P|C)$.

2. Programming part :

   Using the datas in the letters_frequency.py file, write a python code that computes the entropy of the english language.

Exercise 3 : Huffman coding

1. By hand :

   We consider a random letter generator that produces characters A and B (ASCII 8-bit characters) with probability 0.3 et 0.7 respectively. Let $S^2 = \{AA, AB, BA, BB\}$ be a source that generates the characters two at a time (drawing each of them independently according to the given probabilites).

   Compute :

   - $H(S)$.
   - $H(S^2)$.
   - Find the Huffman coding for S^2.
   - Compute the codeword length of the Huffman coding and compare it with Shannon bound.
   - What is the main interest in using this coding ?

2. Programming part :

   - Implement the Huffman coding to the english language, using the joint python file.
   - Implement a function that computes the codeword length of this code and compare with entropy.
   - (Bonus) Compare the obtained result with the traditional Morse code and comment.