UNIVERSITÉ DE GENÈVE

ANALYSE ET TRAITEMENT DE L'INFORMATION
14X026

TP 1: Linear Algebra

Author: Joao Costa

E-mail: Joao.Costa@etu.unige.ch

December 2022

UNIVERSITÉ
DE GENÈVE
FACULTÉ DES SCIENCES
Département d'informatique

Exercise 1. Quantifiers of information

1. Compute: H(u), H(v), H(w)
   $H(X) = -\sum_{x \in X} P_x(x) \cdot log_2(P_x(x))$

   (a) $H(u) = P_u(0) \cdot log_2(P_u(0)) + P_u(1) \cdot log_2(P_u(1)) = 0,9544$
   where: $P_u(0) = \frac{1}{4} * 2 + \frac{1}{8} = 0,625$
   and $P_u(1) = \frac{1}{8} + \frac{1}{4} = 0,375$

   (b) $H(v) = P_v(0) \cdot log_2(P_v(0)) + P_v(1) \cdot log_2(P_v(1)) = 0,9544$
   where: $P_v(0) = \frac{1}{4} + \frac{1}{8} = 0,375$
   and $P_v(1) = \frac{1}{8} + \frac{1}{4} + \frac{1}{4} = 0,625$

   (c) $H(w) = P_w(0) \cdot log_2(P_w(0)) + P_w(1) \cdot log_2(P_w(1)) = 1$
   where: $P_w(0) = \frac{1}{4} * 2 = 0,5$
   and $P_w(1) = \frac{1}{8} * 2 + \frac{1}{4} + \frac{1}{4} = 0,5$

2. Compute: H(u|v), H(v|u), H(w|u)
   $H(X|Y) = -\sum_{x \in X, y \in Y} P_{x,y}(x,y) \cdot log_2(P_{x,y}(x|y))$
   $P(X|Y) = \frac{P(x,Y)}{P(Y)}$

   (a) $H(u|v) = -[P(0,0) \cdot log_2(P(0,0)) + P(0,1) \cdot log_2(P(0,1)) + P(1,0) \cdot log_2(P(1,0)) + P(1,1) \cdot log_2(P(1,1))] = 0,9512$
   where: $P(0,0) = \frac{1}{4} + 0 = 0,25$
   and $P(1,0) = \frac{1}{8} + 0 = 0,125$
   and $P(0,1) = \frac{1}{8} + \frac{1}{4} = 0,375$
   and $P(1,1) = \frac{1}{4} + 0 = 0,25$

   (b) $H(v|u) = -[P(0,0) \cdot log_2(P(0,0)) + P(0,1) \cdot log_2(P(0,1)) + P(1,0) \cdot log_2(P(1,0)) + P(1,1) \cdot log_2(P(1,1))] = 0,9512$
   where: $P(0,0) = \frac{1}{4} + 0 = 0,25$
   and $P(1,0) = \frac{1}{8} + 0 = 0,125$
   and $P(0,1) = \frac{1}{8} + \frac{1}{4} = 0,375$
   and $P(1,1) = \frac{1}{4} + 0 = 0,25$

   (c) $H(w|u) = -[P(0,0) \cdot log_2(P(0,0)) + P(0,1) \cdot log_2(P(0,1)) + P(1,0) \cdot log_2(P(1,0)) + P(1,1) \cdot log_2(P(1,1))] = 0,4512$
   where: $P(0,0) = 0,5$
   and $P(1,0) = 0,125$
   and $P(0,1) =$
   and $P(1,1) = 0,375$

3. Compute: I(u,v), I(u,w), I(u,v,w)
   $I(X,Y) = \sum_{x,y} P(x,y) \cdot log(\frac{P(x,y)}{P(x) \cdot P(y)}) = H(X) - H(X|Y)$

   (a) $I(u,v) = H(u) - H(u|v) = 0,0035$

   (b) $I(u,w) = H(u) - H(u|w) = 0,5487$

   (c) $I(u,v,w) = I(v,w) - I(v,w|u) = 0,0581$
   where: $I(v,w|u) = 0,1068$
   and $I(v,w) = 0,0487$

4. Compute: H(u,v,w)

   (a) $H(u,v,w) = H(u) + H(v|u) + H(w|u,v) = 2,25$

Exercise 2. Source coding

See file exo_2.ipynb for code

First of we generate the alphabet, and the probability for each symbol of the alphabet, with $p_1 = 0.1$, the symbol 00000 is the most probable with probability $(\frac{9}{10})^5$, and the less likely being 11111 with $(\frac{1}{10})^5$ probability.

Then we generate a bit sequence of length 10000, which means there are 2000 syombols, each length 5, (with $p_1 = 0.1$)

Now we simply count each occurence of each symbol of the alphabet, the results are as expected: (below an image of a print that shows the occurrence of each symbol)

```
['00000' '00001' '00010' '00011' '00100' '00101' '00110' '00111' '01000'
 '01001' '01010' '01011' '01100' '01101' '01110' '01111' '10000' '10001'
 '10010' '10011' '10100' '10101' '11000' '11001' '11010' '11100']
[1227  124  121   12  128   14   21    2  127   13   12    2   15    3
    2    1  112   16   16    3   13    2    9    1    3    1]
```

Now we use the Hoffman code to compute a code for each symbol, it gives a smaller code for the symbols that occour more often:

```
Bits Code          Value Symbol
   3 000               0 '00100'
   6 001000            8 '01100'
   6 001001            9 '10001'
   6 001010           10 '10010'
   9 001011000        88 '01011'
   9 001011001        89 '01110'
   9 001011010        90 '10101'
  10 0010110110      182 '11001'
  10 0010110111      183 '11100'
   7 0010111          23 '11000'
   6 001100           12 '00110'
   7 0011010          26 '00011'
   7 0011011          27 '01010'
   7 0011100          28 '01001'
   9 001110100       116 '01101'
   9 001110101       117 '10011'
   9 001110110       118 '11010'
  11 00111011100     476 _EOF
  11 00111011101     477 '01111'
  10 0011101111      239 '00111'
   7 0011110          30 '10100'
   7 0011111          31 '00101'
   4 0100              4 '10000'
   4 0101              5 '00010'
   4 0110              6 '00001'
   4 0111              7 '01000'
   1 1                 1 '00000'
```

Finally we compute the entropy:

```
Length of the output of the encoder: 4656
Entropy of the sequence : 2.3449779679464062
Ratio len encoded / len original : 2.328
```

These results are decent, but Huffman encoding would work best if all the bits are iid, which is not the case here . However it still manages to create a smaller encoded singal!

Exercise 3. Moments

See file exo_3.ipynb for code

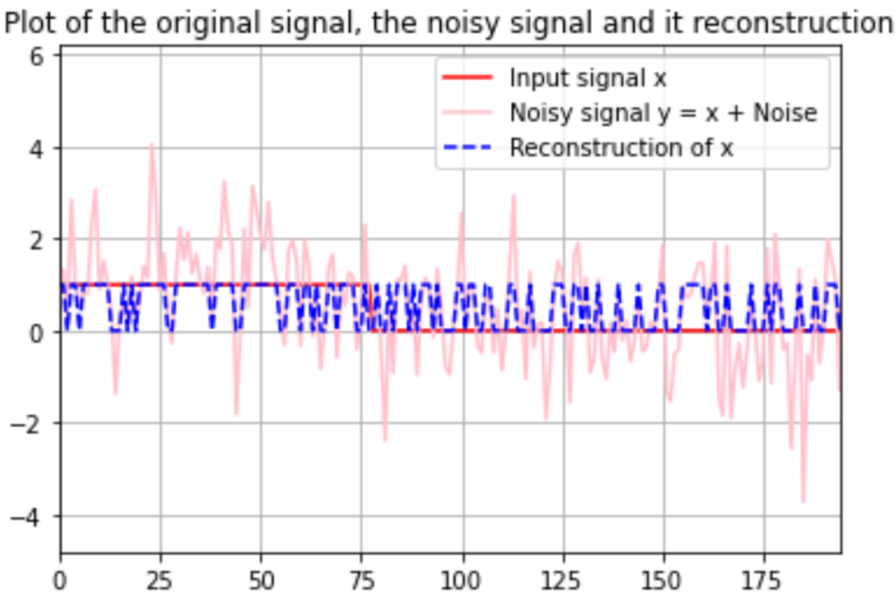
First we generate a signal b, and repeat it n times. The result is x

x is of length n*k, k is length of b, and n the number of repetitions.

Secondly we comput y = x + z, where z is a random noise, this noise simulates signal loss during transmission.

Then, from y we try to reconstruct x, this is done by setting a decision threshold, to act as a boundry, if y[i] is larger than a certain threshold, then we assum it is a 1, if it is smaller, we assume it was a 0.

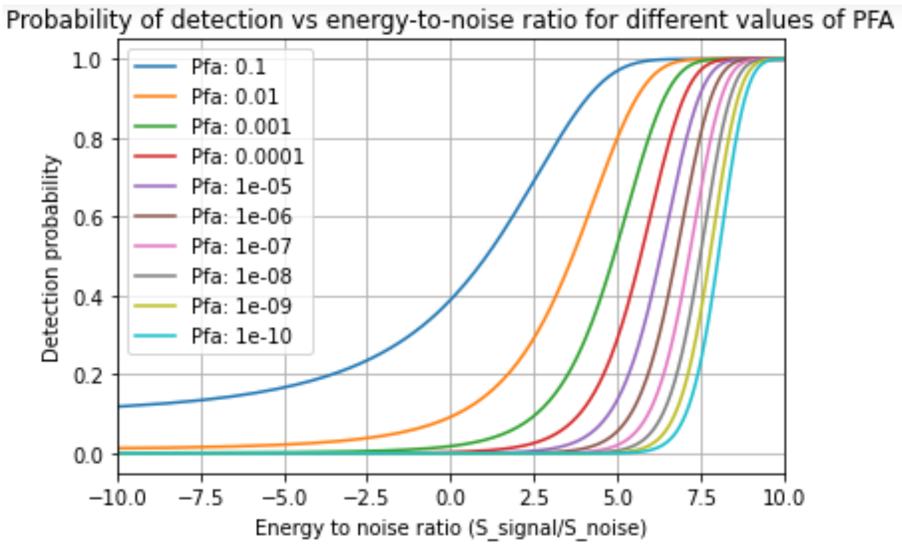
Here we can see an graph of all these values:



with the follwoing P_{fa} and Pm values :

```
Value of P_fa: 0.0003
Value of P_m: 0.0004
Value of P_e: 0.0007
```

we can see that the values are quite low, which means we are making less errors!



In the following graph we try to see how signal lenght impacts both probabilites of miss and false acceptance, the idea is simple, the noisier the input signal is, the longer it needs to be for a good signal reconstruction! Which is to be completely expected.