Unidade Curricular

“Informação e Codificação”

António José Ribeiro Neves

an@ua.pt

https://www.ua.pt/pt/uc/15264

Outline

Exercises

Given an alphabet $\mathcal{A} = \{a_1, a_2, a_3, a_4\}$, find the first-order entropy in the following cases:

(a) $P(a_1) = P(a_2) = P(a_3) = P(a_4) = \frac{1}{4}$.

(b) $P(a_1) = \frac{1}{2}$, $P(a_2) = \frac{1}{4}$, $P(a_3) = P(a_4) = \frac{1}{8}$.

(c) $P(a_1) = 0.505$, $P(a_2) = \frac{1}{4}$, $P(a_3) = \frac{1}{8}$, and $P(a_4) = 0.12$.

Given the following sequence:

$$ATGCTTAAGCTGCTTAACCTGAAGCTTCCGCTGAAGAACCTG$$

$$CTGAACCCGCTTAAGCTGAACCTTCTGAAGCTTAACCTGCTT$$

(a) Estimating the probabilities from the sequence, compute the first, second, third, and fourth order entropy for this sequence.

(b) Based on these entropies, can you infer how this sequence is structured?

Challenges

Alice is tossing a coin repeatedly. She wants to send the outcomes to Bob. To send the message, Alice is only able to use a flashlight, by turning it on and off once per second.

1. *Suppose that Alice and Bob agreed on the size of the message, n, beforehand, i.e., on how many coin tossing outcomes will be sent in each message. Propose a procedure for this transmission scheme.*
2. *Suppose now that Alice intends to send several messages of lengths, n1 , n2 , . . . , not known in advance by Bob. Can you suggest a transmission procedure that works in this case?*
3. *Alice wants to send a letter to Bob, using the same transmission scheme. Consider an alphabet of 27 symbols (26 letters plus the space). How can this be done?*

Challenges

Now Alice wants to play a game with Bob (they are finally face to face!). She thinks of a number between 1 *and* 100*. Bob is allowed to ask every question he wants, but only gets a yes or no answer.*

1. *What is the minimum number of questions that Bob needs to sequentially pose to Alice (the next question is posed only after knowing the answer to the previous question), in order to discover the number Alice thought of? What should those questions be?*
2. *Now Bob is only allowed to pose all the questions at once. Is the minimum number of required questions the same? In this case, what should be the questions?*

Alice has a deck of playing cards (52 cards). After shuffling the deck, Alice wants to send enough information to Bob in order for him to put his deck in the same order. How many bits does Alice need to send to Bob to communicate this information? Can you propose a method for doing it?

Challenges

Using the flashlight communication system, Alice wants to communicate to Bob the results of throwing a pair of dice (i.e., a number from 2 to 12). However, Alice is running out of batteries for the flashlight and, therefore, she wants to preserve them as much as possible. So, she wonders if, in this case, it is possible to use, on average, less than log2 11 ≈ 3.46 *bits for sending to Bob the outcome of each trial. . . Can you help her solving this problem?*

The Morse code is composed of sequences of dots and dashes of various lengths, according to the convention shown in Fig. 4.1. In this code, the characters of a word need to be separated by additional space (three units of time, where one unit of time is equivalente to the length of a dot). For example, the letters "ET" originate the sequence "· —", to distinguish from the letter "A" that is encoded as "·—". Do you think this explicit separation could be avoided? If you think so, how should the code be constructed?

Some questions…

What is Variable Length Coding? Why this is usefull in compression?

How Do We Assign Code Lengths in Variable Length Coding?

How Does Huffman Coding Work?

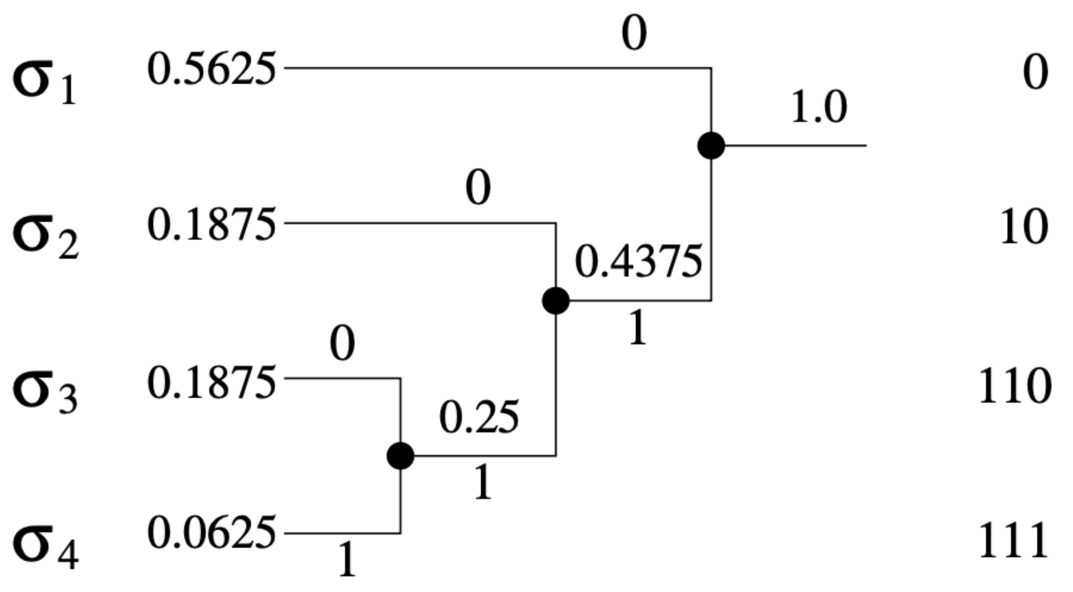
What is Shannon-Fano Coding?

What is a Comma Code, and How Does It Work?

What are Golomb Codes? Where they can be used?

Huffman codes

- Variable-length codes developed by David A. Huffman (1925–1999)
- These codes are optimum, in the sense that they minimize the average length of the encoded messages, among all possible variable-length codes.
- They are prefix-free codes, i.e., none of the codewords has a prefix made of a shorter codeword.
- For constructing the Huffman codes:
  1. the symbols (tree nodes) with the smallest probabilities are combined first
  2. the new node gets the sum of the probabilities of the two combined nodes - this procedure is repeated until having a single node with a probability equal to one
  3. finally, the codewords are obtained by associating 0's and 1's to the branches of the tree.



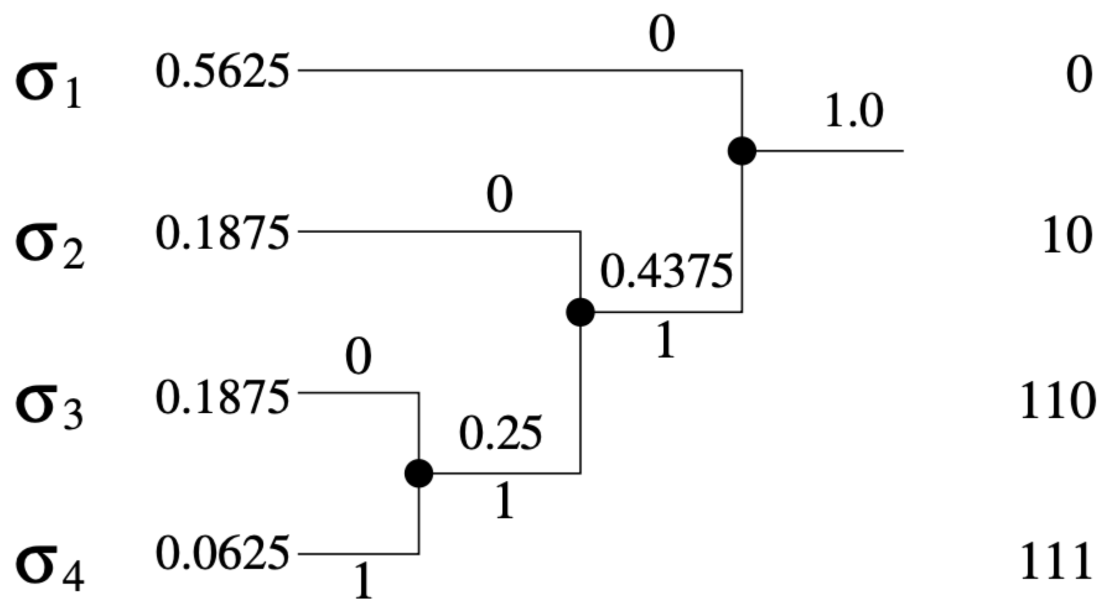
Example of tree associated with a 4-symbol Hu

Problem

- What is the mean number of bits/symbol in this exemple?

- How do you decode the following binary string:

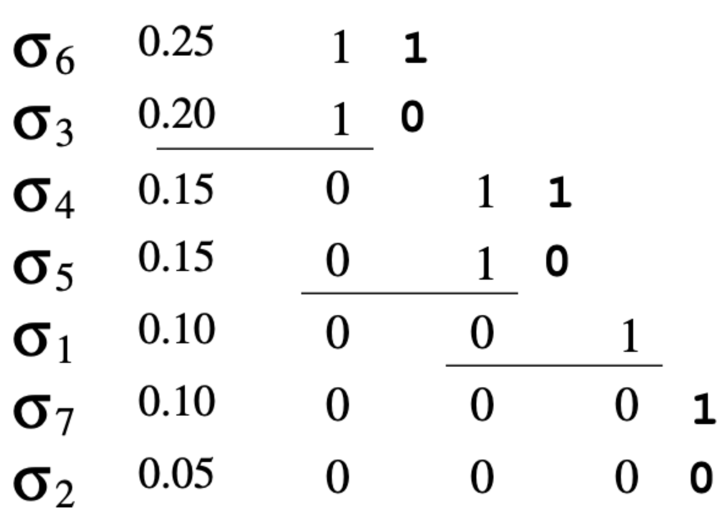
  0100111101100



Example of tree associated with a 4-symbol Hu

Shannon-Fano codes

- Shannon-Fano coding was proposed before Huffman coding and, although not optimum as Huffman coding, it nevertheless provides good variable-size codes.
- Similar to Huffman codes, in the sense that they also are variable-length and immediately decodable codes.
- For constructing these codes
  - the symbols are first ordered by increasing or decreasing probability
  - the symbols are then split into two sets having as much as possible similar total probabilities
  - One of the sets gets a “0” label, whereas the other gets a “1” label.
  - This is repeated until having sets with only one or two symbols.



σ_6	0.25	1	**1**		
σ_3	0.20	1	**0**		
σ_4	0.15	0	1	**1**	
σ_5	0.15	0	1	**0**	
σ_1	0.10	0	0	1	
σ_7	0.10	0	0	0	**1**
σ_2	0.05	0	0	0	**0**

Example of construction of a Shannon-Fano

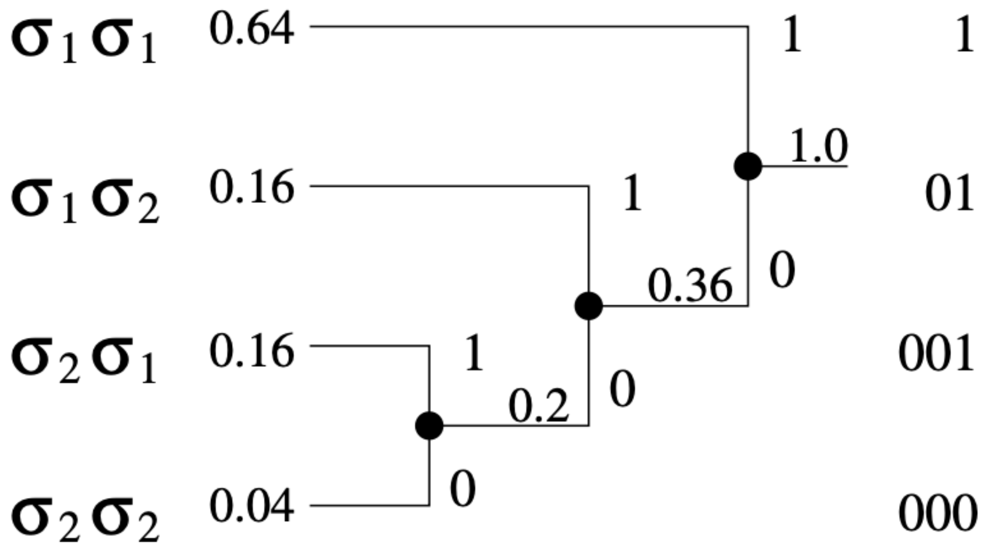
Alphabet extension

- *Consider two symbols,* σ1 *and* σ2*. Varying the values of* p1 *and* p2*, we get the following entropy values:*

p_1	0.9	0.8	0.7	0.6	0.5
p_2	0.1	0.2	0.3	0.4	0.5
H	0.47	0.72	0.88	0.97	1.00

- *As expected, the larger the difference between these two probabilities, less bits, on average, are in theory needed for representing a message.*
- *Consider, for example, the case where* p1 = 0.8 *and* p2 = 0.2*:*
    - *Using Huffman coding or Shannon-Fano coding, we get an average code length of one bit per symbol.*
    - *This is 28% longer than what is theoretically necessary.*

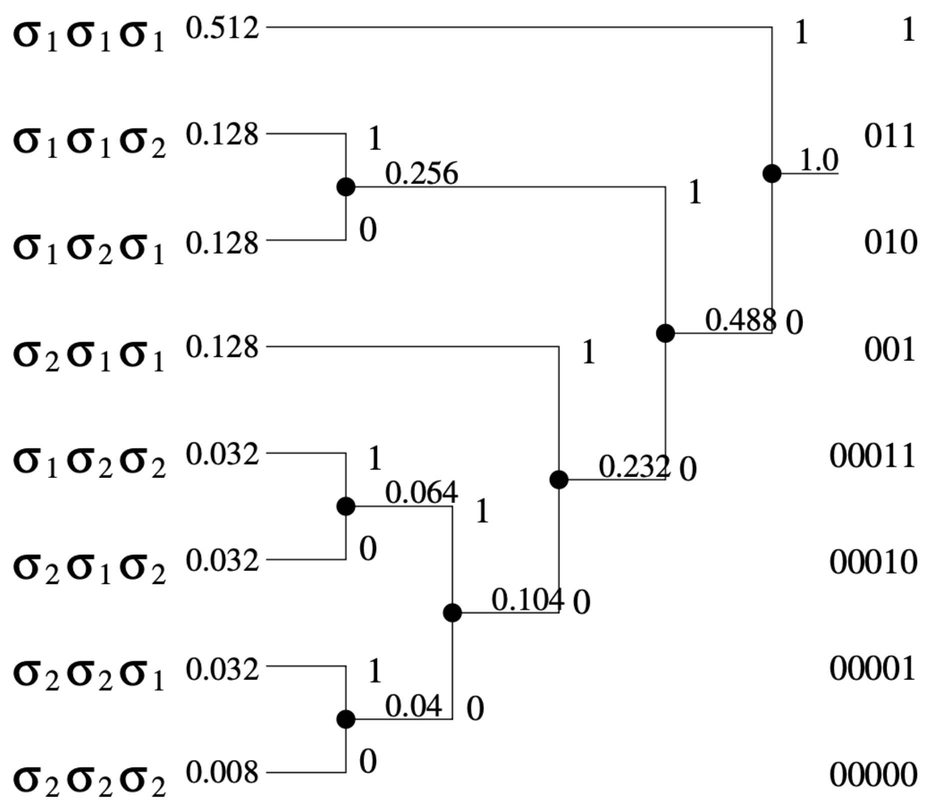
Pair	Probability
$\sigma_1\sigma_1$	$0.8 \times 0.8 = 0.64$
$\sigma_1\sigma_2$	$0.8 \times 0.2 = 0.16$
$\sigma_2\sigma_1$	$0.2 \times 0.8 = 0.16$
$\sigma_2\sigma_2$	$0.2 \times 0.2 = 0.04$



Average length: 1.56 bits (0.78 *bits/original symbol.*)

Alphabet extension

- The average length of the corresponding Huffman code is 2.184 bits, i.e., 0.728 bits/original symbol



$\sigma_1\sigma_1\sigma_1$ 0.512
$\sigma_1\sigma_1\sigma_2$ 0.128
$\sigma_1\sigma_2\sigma_1$ 0.128
$\sigma_2\sigma_1\sigma_1$ 0.128
$\sigma_1\sigma_2\sigma_2$ 0.032
$\sigma_2\sigma_1\sigma_2$ 0.032
$\sigma_2\sigma_2\sigma_1$ 0.032
$\sigma_2\sigma_2\sigma_2$ 0.008
0.256
0.064
0.04
0.104
0.232
0.488
1.0
1
011
010
001
00011
00010
00001
00000

Group	Probability
$\sigma_1\sigma_1\sigma_1$	$0.8 \times 0.8 \times 0.8 = 0.512$
$\sigma_1\sigma_1\sigma_2$	$0.8 \times 0.8 \times 0.2 = 0.128$
$\sigma_1\sigma_2\sigma_1$	$0.8 \times 0.2 \times 0.8 = 0.128$
$\sigma_1\sigma_2\sigma_2$	$0.8 \times 0.2 \times 0.2 = 0.032$
$\sigma_2\sigma_1\sigma_1$	$0.2 \times 0.8 \times 0.8 = 0.128$
$\sigma_2\sigma_1\sigma_2$	$0.2 \times 0.8 \times 0.2 = 0.032$
$\sigma_2\sigma_2\sigma_1$	$0.2 \times 0.2 \times 0.8 = 0.032$
$\sigma_2\sigma_2\sigma_2$	$0.2 \times 0.2 \times 0.2 = 0.008$

Comma code (also known as unary code)

- Sometimes, it is necessary to design codes for representing, efficiently, integer numbers whose probabilities decrease with their values.
- On the other hand, these are "open" codes, i.e., they are able to accommodate rare, but possible, (large) numbers.
- The comma code (also known as the unary code) is the simplest variable-length code. Each codeword is built using a string of 0's (1's), terminated by one 1 (0).

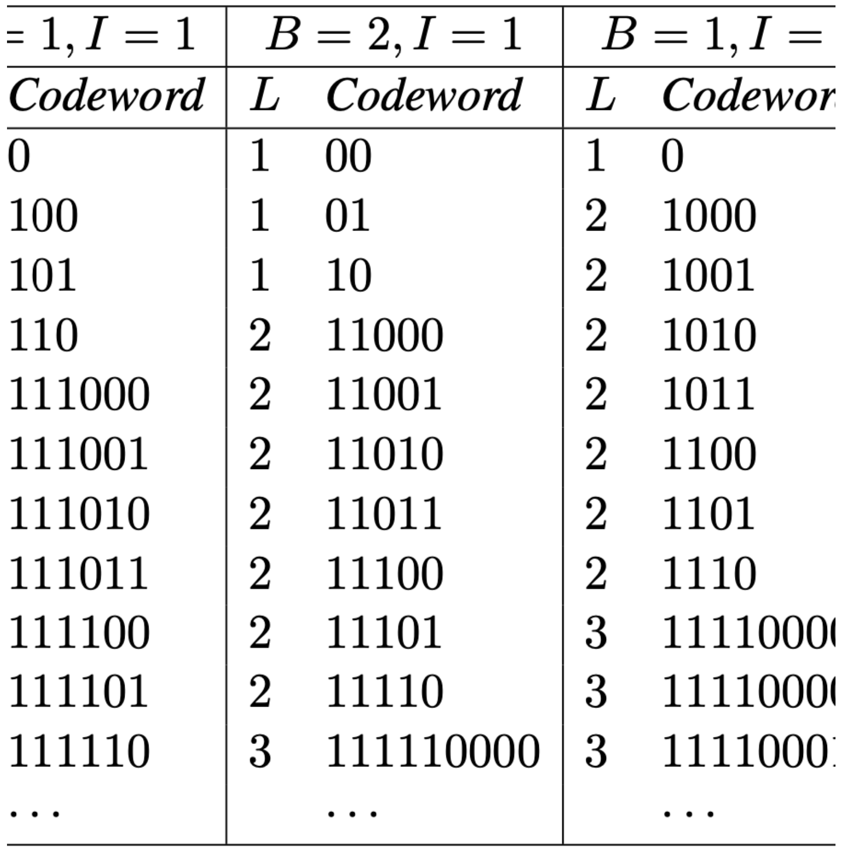
- Shift-codes are generalizations of the comma code.
- These codes are organized by levels, based on a set of 2B codewords of length B.
- One of those codewords is used to indicate the shift to the next level.

0	–	0
1	–	10
2	–	110
3	–	1110
4	–	11110
...		

$B = 1$	$B = 2$	$B = 3$
0	00	000
10	01	001
110	10	010
1110	1100	011
11110	1101	100
111110	1110	101
1111110	111100	110
11111110	111101	111000
...	...	...

Shift Codes

- Another type of shift code uses, besides the base, B, an increment, I.
- The first level is formed by all codewords of length B, except for the shift codeword.
- The next level (level 2) is formed by all codewords of B + I bits, except for the shift codeword, concatenated with the (prefix) shift codeword of the previous level.
- Generalizing, level L consists of all B + I (L − 1) bit codewords, except for the shift sequence, concatenated with the prefix (shift codeword) of level L − 1.



$= 1, I = 1$	$B = 2, I = 1$		$B = 1, I =$	
Codeword	*L*	*Codeword*	*L*	*Codewor*
0	1	00	1	0
100	1	01	2	1000
101	1	10	2	1001
110	2	11000	2	1010
111000	2	11001	2	1011
111001	2	11010	2	1100
111010	2	11011	2	1101
111011	2	11100	2	1110
111100	2	11101	3	11110000
111101	2	11110	3	11110000
111110	3	111110000	3	1111000
...		...		...

Golomb code

- The Golomb code was proposed by Solomon W. Golomb (1932–2016)
- It relies on separating an integer into two parts:
  - One of those parts is represented by a unary code.
  - The other part is represented using a binary code.
- The Golomb code is a family of codes that depend on an integer

parameter,m > 0.

$$p_i = \alpha^i(1-\alpha), \quad i = 0, 1, 2, \ldots$$

- The Golomb code is optimum for an information source following a distribution:

$$m = \left\lceil -\frac{1}{\log \alpha} \right\rceil.$$

- An integer i ≥ 0 is represented by two numbers, q and r, where

$$q = \left\lfloor \frac{i}{m} \right\rfloor \qquad \text{and} \qquad r = i - qm$$

- The quotient, q, can have the values 0, 1, 2, . . . , and is represented by the corresponding unary code. The remainder of the division, r, can have the values 0, 1, 2, . . . , m−1, and is represented by the corresponding binary code.

Consider $i = 11$ *and* $m = 4$. *Then,*

$$q = \left\lfloor \frac{11}{4} \right\rfloor = 2, \quad r = 11 - 2 \times 4 = 3 \quad \rightarrow \quad 001\ 11$$

Golomb code

If m is not a power of 2, then the binary code (that represents the remainder of the division), is not efficient. In that case, the number of bits used can be reduced using a **truncated binary code**:

- First, define $b = \lceil \log m \rceil$.
- Encode the first $2^b - m$ values of r using the first $2^b - m$ binary codewords of $b - 1$ bits.
- Encode the remainder values of r by coding the number $r + 2^b - m$ in binary codewords of b bits.

Golomb code for $m = 5$ *and* $i = 0, 1, \ldots, 15$*:*

i	q	r	*Codeword*	i	q	r	*Codeword*
0	0	0	000	8	1	3	10110
1	0	1	001	9	1	4	10111
2	0	2	010	10	2	0	11000
3	0	3	0110	11	2	1	11001
4	0	4	0111	12	2	2	11010
5	1	0	1000	13	2	3	110110
6	1	1	1001	14	2	4	110111
7	1	2	1010	15	3	0	111000