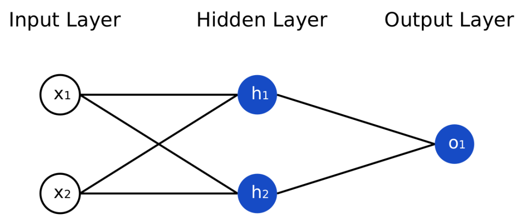
Information and Communication Technologies — S7 — Work n°13

Graphs — Part 2

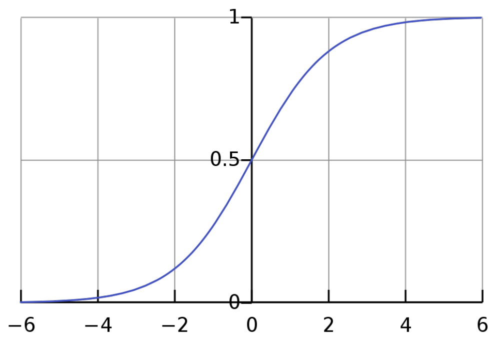
We will build our first neural network. A neural network takes some inputs, and computes some outputs. The outputs are usually numbers between 0 and 1, indicating a probability that the inputs have a characteristic. On the following picture, there are two inputs, from those inputs the network computes two intermediate values, and from those intermediate values, one single output is computed.



Let us take a very simple example of a neural network that takes as input the weight and the height of a person, and outputs the likelihood that this person is a female (thus, a value close to 0 will mean that the network guesses that the person is a male, a value close to 1 that it is a female). Based on the following data, let's build such a neural network.

Name	Weight (kg)	Height (m)	Gender
Alice	60	1.65	F
Bob	72.5	1.83	M
Charlie	69	1.78	M
Diana	54	1.52	F

A very common function to use is the "sigmoid" function. This function is called this way because it is shaped like an S, see the following drawing :



The basic sigmoid function is $f(x) = \dfrac{1}{1 + \mathrm{e}^{-x}}$.

1. What is the limit of f when $x \to -\infty$? When $x \to +\infty$?

This function thus assigns 0 to very low values of x and 1 to very high values of x. It is possible to shift the graph so that it's not centered in $x = 0$, and to change the appearance of the S. For instance, you can try to put different values of a and b :

$$f(x) = \frac{1}{1 + \mathrm{e}^{-(a+bx)}}$$

2. What happens when you change the value of b?

3. Can you tell where is centered the graph of f, depending of a and b?

This function is important because it allows to create thresholds to accept or reject some inputs based on the values.

4. How would you create a threshold to recognize if a person is a male or female, based on their height?

5. How would you create a threshold to recognize if a person is a male or female, based on their weight?

In a neural network, each artificial neuron will use data from the artificial neurons that are in the previous layer. It is thus a good idea to keep track of the ingoing neighborhood in the data structure, see Listing 1.

```python
class Neuron:
    def __init__(self, name):
        self.name = name
        self.value = 0
        self.in_neighbors = []
        self.weights = []
        self.a = 0
        self.b = 1

input1 = Neuron("weight")
input2 = Neuron("height")
h1 = Neuron("h1")
h2 = Neuron("h2")
output = Neuron("output")

# We must first define the nodes, before we can define the neighbors (which use
    the neurons!)
# WARNING: the in_neighbors list and the weights list must be of the same length.
output.in_neighbors = [h1, h2]
output.weights = [0.5, 0.5]
h1.in_neighbors = [input1, input2]
h1.weights = [0.5, 0.5]
h2.in_neighbors = [input1, input2]
h2.weights = [0.5, 0.5]

# Sigmoid function
def sigmoid(x, a, b):
    return 1/(1 + 2.17128128**(-(a + b*x)))

# Recursive function to compute the output value of a neuron based on the ingoing
    neighbors of this neuron, the weights associated to those neighbors, and the a
     and b value of the sigmoid function associated to this neuron.
def compute_value(n):
    if n.in_neighbors == []:
        return n.value
    in_values = []
    for neighbor in n.in_neighbors:
        in_values.append(compute_value(neighbor))
    value = 0
    for i in range(len(in_values)):
        value = value + in_values[i] * n.weights[i]
    return sigmoid(value, n.a, n.b)

# Put the values of Alice inside the network and computes the output value
input1.data = 60
input2.data = 1.65
print(compute_value(output))
```

Listing 1: The neural network data structure.

6. Test the python file.

7. Modify the values of the field “a” of the Neurons h1 and h2 to take into account the thresholds computed in the two previous questions.

8. Test the python file again. Change the test persons to see if our neural network still works.