Neural Networks

Exercise 1 : Simple Perceptron

We consider a case with two input variables $x_1, x_2 \in \{0,1\}$ and a single perceptron with activation function a. We multiply these inputs respectively with weights w_1 and w_2. The perceptron is activated if the sum of these is greater than some threshold w_0.

$$a(x) = \begin{cases} 1 & \text{if } w_1x_1 + w_2x_2 - w_0 > 0 \\ 0 & \text{otherwise.} \end{cases}$$

1. Provide the weights w_0, w_1 and w_2 so that the perceptron computes the logical `AND` function.
2. Do the same for the logical `OR` function.
3. Show that one single perceptron is not able to compute the logical `XOR` function.
4. Build a neural *network* with a single output that computes `XOR`.

x_1	x_2	x_1 `XOR` x_2
1	1	0
1	0	1
0	1	1
0	0	0

Exercise 2 : Update rule for multilayer neural networks

We saw in class how to update the weights for a single unit using gradient descent. The goal of this exercise is to verify the update rules for the multilayer neural networks.

Notation :

— x_{ji} is the i^{th} input to unit j
— w_{ji} is the weight associated with the i^{th} input to unit j
— $net_j = \sum_i w_{ji}x_{ji}$ is the weighted sum of the inputs for unit j
— o_j is the output computed by unit j
— t_j is the target output for unit j
— σ is the sigmoid function
— out denotes the set of units in the final layer of the network
— $downstream(j)$ the set of units whose immediate inputs include the output of unit j.

We measure the error of a training example d using this formula :

$$E_d(\overrightarrow{w}) = \frac{1}{2}\sum_{k \in out}(t_k - o_k)^2$$

1. What is the derivative of the sigmoid function $\sigma(x) = \dfrac{1}{1+e^{-x}}$?
2. Write the update rules
   (a) for the units in the final layer of the network
   (b) for the units in the hidden layers