Exercise Solutions - Neural Networks

1 Single Layer Perceptron

1.a

In the perceptron below, what will the output be when the input is (0, 0)? What about inputs (0, 1), (1, 1) and (1, 0)? What if we change the bias weight to -0.5?

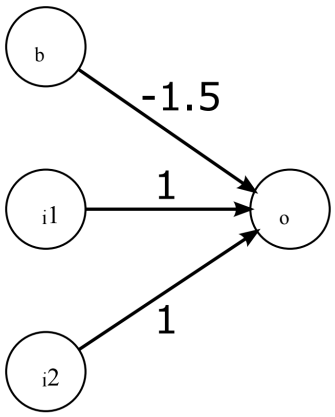


Figure 1: Single Layer Perceptron. b = 1

Answer:

Bias = −1.5			Bias = −0.5		
Input	Weighted sum	Output	Input	Weighted sum	Output
$(0, 0)$	-1.5	0	$(0, 0)$	-0.5	0
$(0, 1)$	-0.5	0	$(0, 1)$	0.5	1
$(1, 0)$	-0.5	0	$(1, 0)$	0.5	1
$(1, 1)$	0.5	1	$(1, 1)$	1.5	1

1.b

Starting with random weights, how do you proceed in order to train the perceptron above to perform any given binary operation?

Answer:
See the source code for the next task.

ℙ 1.c Implement a perceptron

Implement the perceptron, and train it to perform the logical functions NOT (use only one of the inputs), NAND, and NOR. What happens when you try to train it do the XOR function?

Source code:
The source code is available online on GitHub:
https://github.com/gsiolas/NN-exercises/blob/master/material/week5/bool_perceptron.py

2 Multi Layer Perceptron (MLP)

The figure below shows a multilayer perceptron that constructs the XOR function. How would you rewrite it to construct the binary equivalence function (i.e. the output is above threshold when both inputs are either 0 or 1)? Can you construct it so that it will detect equivalence for any combination of integer inputs?

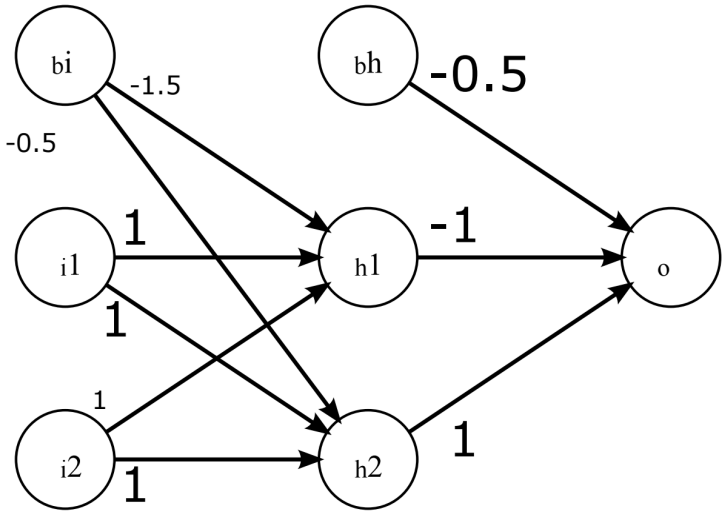
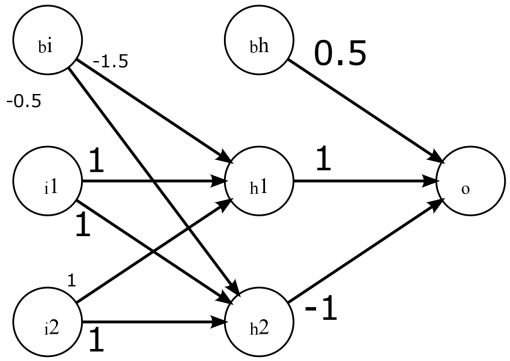


Figure 2: An illustrated example of a multi layer perceptron

Answer:
If we just want it to work in the binary case, we can simply switch signs on all the output weights to create the opposite function:



If we want it to work for all integers, we have to change the weights in the first layer. First, note that the output layer encodes the function $h_2 - h_1 - 0.5 > 0$ which is equivalent to $h_2 \wedge \neg h_1$ since h_1 and h_2 are either 0 or 1. If we equate true with 1 and false with 0 we can write h_1: i_1+i_2-1.5>0 and h_2: i_1+i_2-0.5>0, so the output is

$$o: \ i_1 + i_2 > 0.5 \ \wedge \ i_1 + i_2 \leq 1.5 \quad \Leftrightarrow \quad o: \ 0.5 < i_1 + i_2 \leq 1.5$$

Limiting i_1 and i_2 to integers gives us $o: \ i_1 + i_2 = 1$. So the output is 1 if and only if the sum of the inputs are 1. We can get the equivalence function by changing this to $o: \ i_1 - i_2 = 0$. Going back a step or two we see that we can achieve this by making the output

$$o: \ i_1 - i_2 > -0.5 \ \wedge \ i_1 - i_2 \leq 0.5$$

or something similar. This has the same logical form that we already have, so we don't need to change the weights of the output layer, but only those of the hidden layer.
Tracing back, we see that we would need $h_1: \ i_1 - i_2 - 0.5 > 0$ and $h_2: \ i_1 - i_2 + 0.5 > 0$ as the hidden layer outputs. This implies the following weights: