Appendix: Perceptron

Perceptron Algorithm

Input: A set of examples, $(x_1, y_1), \cdots, (x_n, y_n)$
Output: A perceptron defined by $(w_0, w_1, \cdots, w_d)$

Begin

2. Initialize the weights w_j to 0 $\forall j \in \{0, \cdots, d\}$
3. Repeat until convergence
    4. For each example x_i $\forall i \in \{1, \cdots, n\}$
    5. if $y_i f(x_i) \leq 0$ #an error?
    6. update all w_j with $w_j := w_j + \alpha y_i x_{ij}$ #adjust the weights

End

Perceptron

Some observations:

- The weights $w_1, \ldots, w_d$ determine the slope of the decision boundary.
- w_0 determines the offset of the decision boundary (sometimes noted b).
- Line 6 corresponds to:
  Mistake on positive: add x to weight vector.
  Mistake on negative: substract x from weight vector.
  Some other variants of the algorithm add or subtract 1.
- α is the learning rate.
- Convergence happen when the weights do not change anymore (difference between the last two weight vectors is 0).

Perceptron

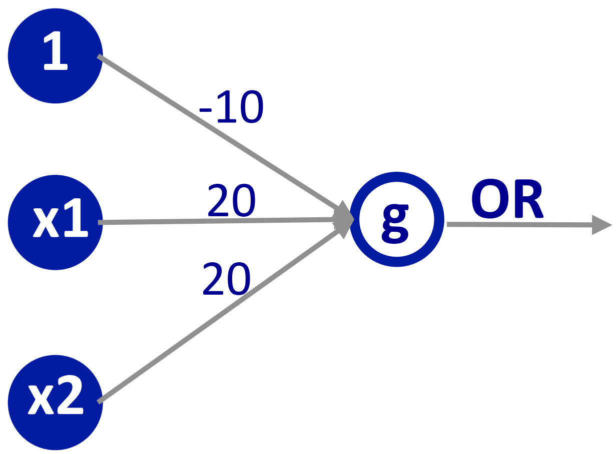
- The w_j determine the contribution of x_j to the label.
- $-w_0$ is a quantity that $\sum_{j=1}^{d} w_j x_{ij}$ needs to exceed for the perceptron to output 1.
- Can be used to represent many Boolean functions: AND, OR, NAND, NOR, NOT but not all of them (e.g., XOR).

The XOR example

First what is the perceptron of the OR?

The XOR example

x_1	x_2	x_1 OR x_2	$g(z)$
0	0	0	$g(w_0 + w_1 x_1 + w_2 x_2) = g(-10)$
0	1	1	$g(10)$
1	0	1	$g(10)$
1	1	1	$g(30)$

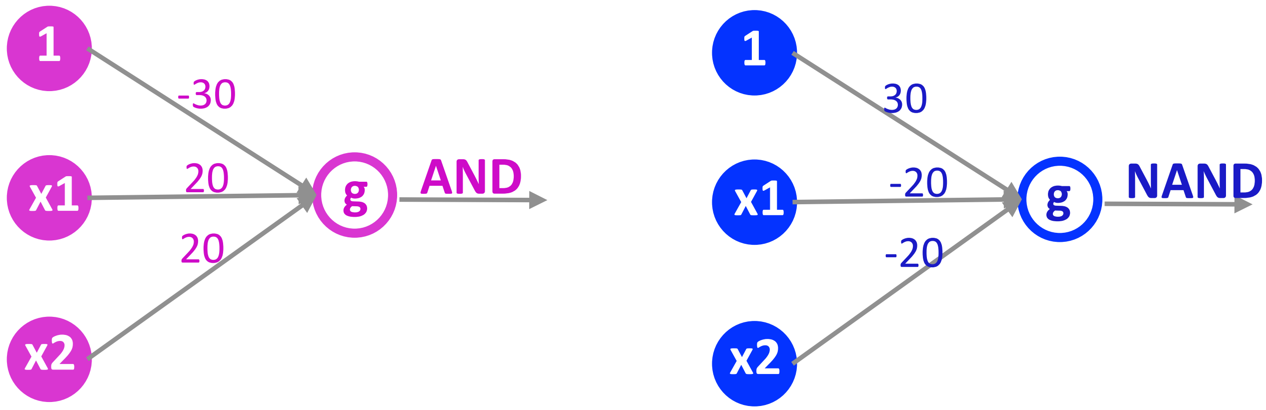


The XOR example

Similarly, we obtain the perceptrons for the AND and NAND:

The XOR example

Similarly, we obtain the perceptrons for the AND and NAND:



Note: how the weights in the NAND are the inverse weights of the AND.

The XOR example

Let's try to create a MLP for the XOR function using elementary perceptrons.

x_1	x_2	x_1 **XOR** x_2	$(x_1$ **OR** $x_2)$ **AND** $(x_1$ **NAND** $x_2)$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0