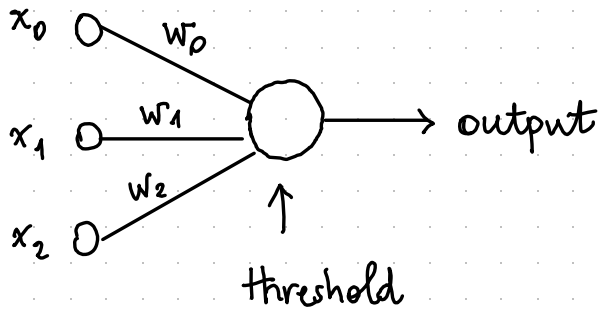
THE PERCEPTRON



The perceptron is like an artificial neuron.

Idea: $x_0, x_1, \ldots, x_{M_{in}-1}$ are signals in input. $w_0, w_1, \ldots, w_{M_{in}-1}$ are weights anociated to each input.

The weighted input in the neuron is

$$\sum_{j=0}^{M_{in}-1} x_j w_j = x \; w$$

A threshold is inside the neuron.

If the weighted input is above the threshold, the neuron emits an output signal (1), otherwise it does not (0).

$$\text{output} = \begin{cases} 1 & \text{if } xw > \text{threshold}, \\ 0 & \text{if } xw \leq \text{threshold}. \end{cases}$$

So a perceptron is like a unit that is processing a decision based on inputs.

Instead of using the threshold:

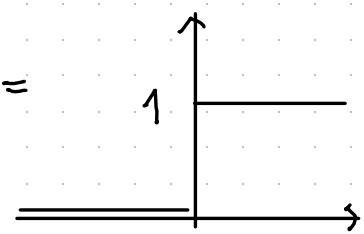
$$\text{output} = \begin{cases} 1 & \text{if } xw + b > 0 \\ 0 & \text{if } xw + b \leq 0 \end{cases} = H(xw+b)$$

↑

a bias is a measure of how easy it is to get the perceptron to fire

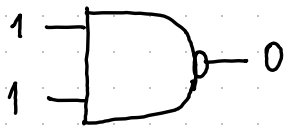
(the larger the bias, the easier it is that the output is 1)

$H(t)$ = Heaviside function =

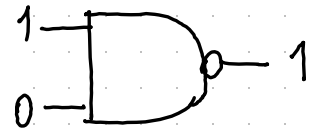


An interesting observation is that a perceptron can implement logical functions.

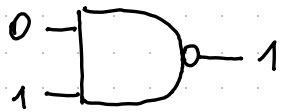
Example: Consider a NAND (= NOT AND) gate.



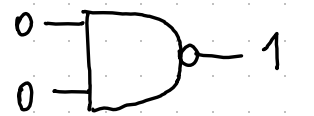
1 NAND 1 = 0



1 NAND 0 = 1

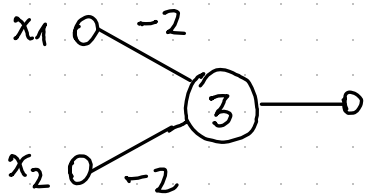


0 NAND 1 = 1



0 NAND 0 = 1

We can build this operation using a perceptron.



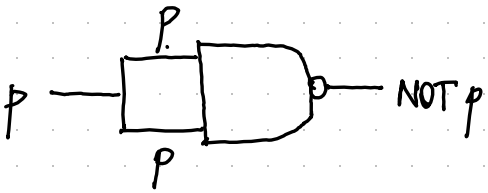
$-2x_1 - 2x_2 + 3$

- $-2 \cdot 1 - 2 \cdot 1 + 3 = -4 + 3 = -1 < 0$
  output = 0
- $-2 \cdot 0 - 2 \cdot 1 + 3 = 1 > 0$ output = 1
- $-2 \cdot 0 - 2 \cdot 0 + 3 = 3 > 0$ output = 1

Why the example on the NAND gate?
Because with NAND we can build "many" logical functions (see below).

Example: Building NOT using NAND

NOT 1 = 0
NOT 0 = 1



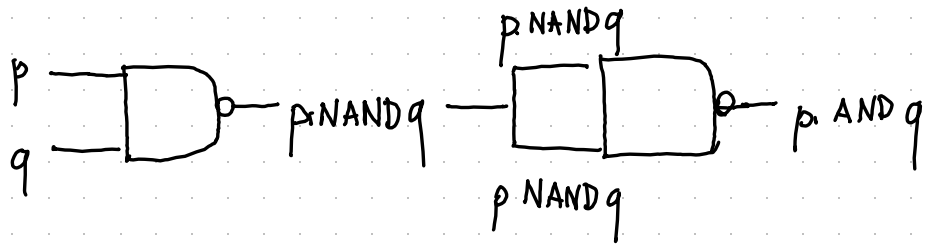
1 NAND 1 = 0 = NOT 1
0 NAND 0 = 1 = NOT 0

Example: Building AND using NAND

1 AND 1 = 1
1 AND 0 = 0
0 AND 0 = 0

$$p \text{ AND } q = \text{NOT}(p \text{ NAND } q) =$$

$$= (p \text{ NAND } q) \text{ NAND } (p \text{ NAND } q)$$



Example: Building OR with NAND

1 OR 1 = 1

1 OR 0 = 1

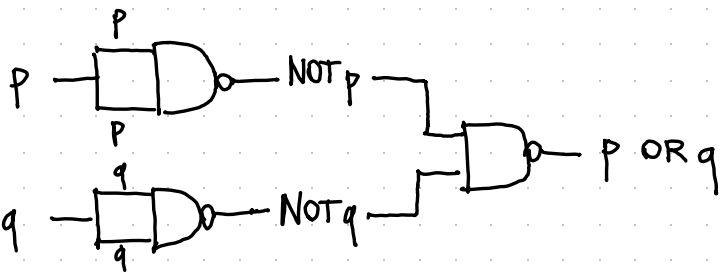
0 OR 1 = 1

0 OR 0 = 0

$$p \text{ OR } q = \text{NOT}((\text{NOT } p) \text{ AND } (\text{NOT } q)) =$$

$$= (\text{NOT } p) \text{ NAND } (\text{NOT } q)$$

$$= (p \text{ NAND } p) \text{ NAND } (q \text{ NAND } q)$$



Example: $p \Rightarrow q$

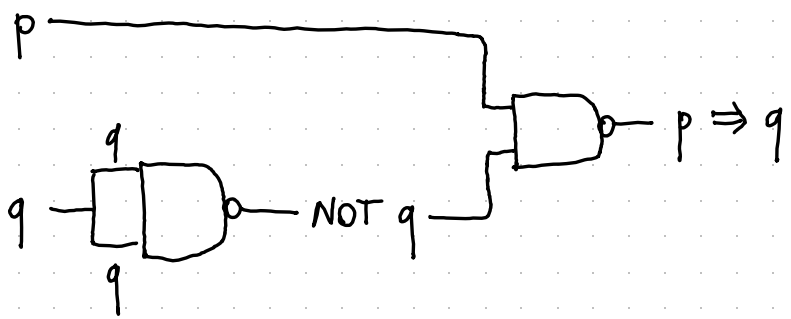
p	q	$p \Rightarrow q$
1	1	1
1	0	0
0	1	1
0	0	1

$(p \Rightarrow q) = (\text{NOT } p) \text{ OR } q =$

$= \text{NOT } ((\text{NOT } (\text{NOT } p)) \text{ AND } (\text{NOT } q))$

$= p \text{ NAND } (\text{NOT } q) =$

$= p \text{ NAND } (q \text{ NAND } q)$



Theorem: NAND gate is a universal logic gate, i.e., for any function $f: \{0,1\}^n \longrightarrow \{0,1\}$, $n \geq 1$, its truth table can be reconstructed by suitably applying consecutive NAND gates to the inputs.

Proof: The proof is constructive:

e.g., $n=2$ $f(P_1, P_2)$ has a truth table:

P_1	P_2	$f(P_1,P_2)$
1	1	$f(1,1)$
1	0	$f(1,0)$
0	1	$f(0,1)$
0	0	$f(0,0)$

Recipe:

- step 1: Look at the values $f(P_1,P_2) = 1$. These can be expressed using AND on the inputs and NOTing some of them in such a way that it works only for that row

- Step 2: combine the results with OR.

Example:

p_1	p_2	$f(p_1, p_2)$	
1	1	0	
1	0	1	p_1 AND (NOT p_2)
0	1	1	(NOT p_1) AND p_2
0	0	0	

$f(p_1, p_2) =$ (p_1 AND (NOT p_2)) OR ((NOT p_1) AND p_2)

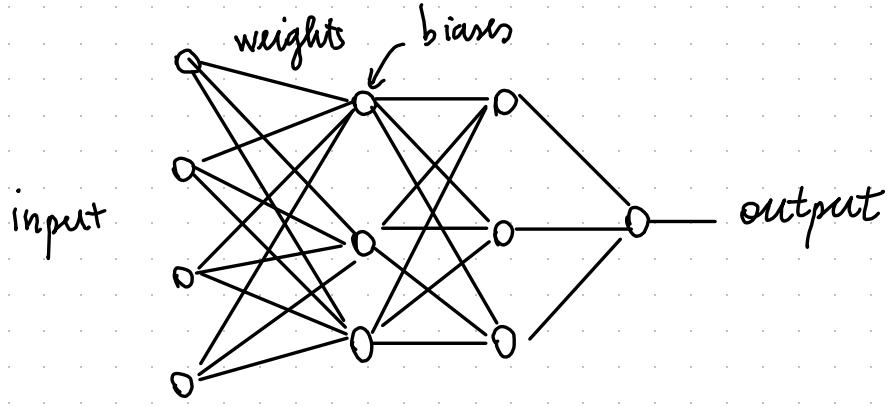
This works regardless of the number of inputs: we can always write, using AND and NOT an operation that returns 1 only on a selected row. Then the results are all combined.

To conclude, we showed that {NAND} generates {AND, NOT, OR}.

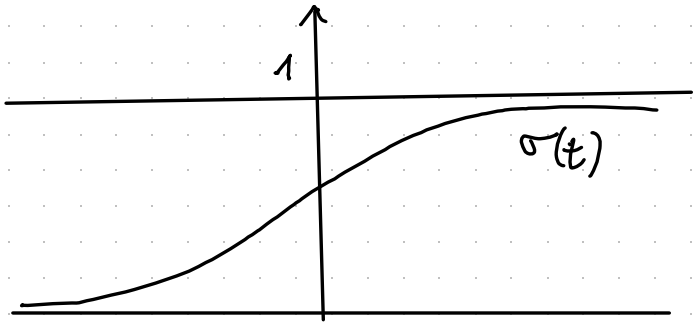
□

This theoretical results shows that an entire processor can be created using only NAND gates.
This is the first signal of the potential of perceptrons.

A first idea of artificial neural network is to put together perceptrons.



In fact, we will not use the perceptron. We will approximate it with sigmoid functions.



The sigmoid function $\sigma(t) = \frac{1}{1+e^{-t}}$ gives a smooth approximation of a perceptron via

$$\sigma(xw+b).$$