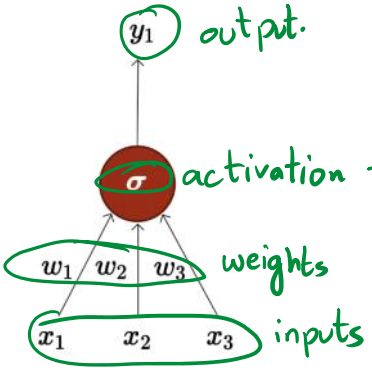
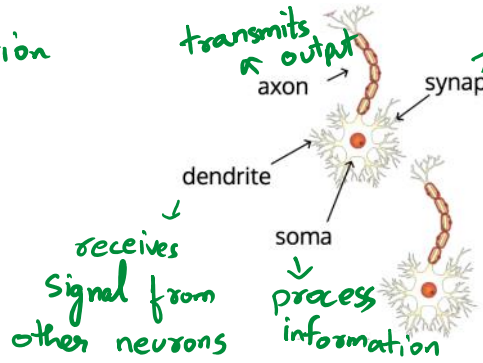
Neurons

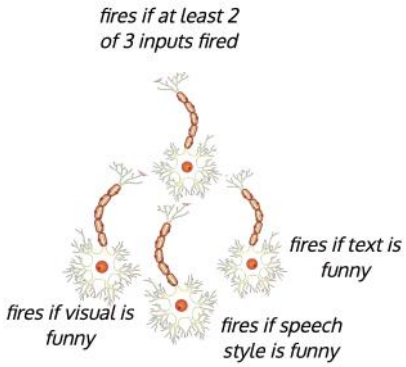
The most fundamental building block of a deep learning neural network is a neuron.



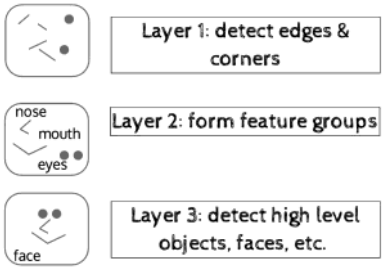
Inspiration: biological neurons



Working: Sense organs → (information) neurons → (fire/activate) response



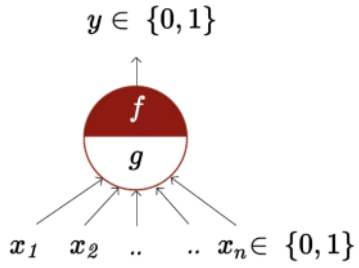
There are multiple layers of neurons which form a heirarchy



McCulloch Pitts Neuron

neuroscientist — logician

- Came up with a highly simplified computational model of a neuron.



- based on binary inputs : $x_1, x_2, \cdots, x_n$
- g aggregates the input
- f takes a binary decision based on g

Inputs can excitatory or inhibitory

Inhibitory : If x_i is on (or 1) then $y = 0$ regardless of other inputs.

eg- If I have high temperature then regardless of weather, movie prices, mood, I'll not go to watch a movie.

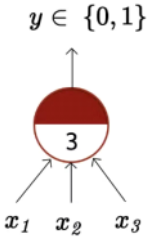
$y = 0$ if any x_i is inhibitory, else

$$g(x_1, x_2, \cdots, x_n) = g(x) = \sum_{i=1}^{n} x_i$$

$$y = f(g(x)) = 1 \quad \text{if} \quad g(x) \geq \theta \quad (\theta: \text{threshold parameter})$$

$$= 0 \quad \text{if} \quad g(x) < \theta$$

Boolean Functions



AND function

x_1	x_2	x_3	$g(x)$	$f(g(x))$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	2	0
1	0	0	1	0
1	0	1	2	0

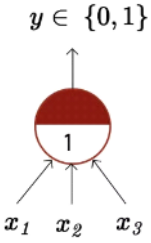
$f(g(x)) = 1$ if $g(x) \geq 3$

AND function

$f(g(x)) = 1$ if $g(x) \geq 3$

$= 0$ else

1	0	0	1	0
1	0	1	2	0
1	1	0	2	0
1	1	1	3	1

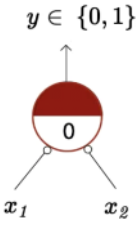


OR function

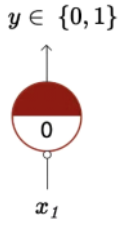
$f(g(x)) = 1$ if $g(x) \geq 1$

$= 0$ else

x_1	x_2	x_3	$g(x)$	$f(g(x))$
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	2	1
1	0	0	1	1
1	0	1	2	1
1	1	0	2	1
1	1	1	3	1



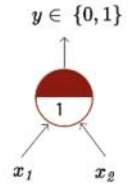
NOR function



NOT function

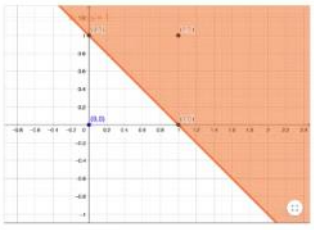
circle at the end represents inhibitory input.

Geometric Representation



OR function

$x_1 + x_2 = \sum_{i=1}^{2} x_i \geq 1$

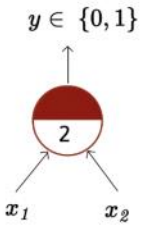


Equation of line: $x_1 + x_2 = 1$

$\theta = 1$ OR

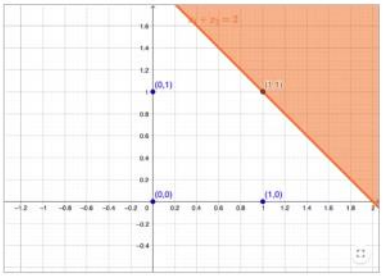
$$\sum_{i=1}^{n} x_i - \theta = 0$$

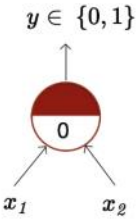
Inputs which produce an output 0 will be on one side $\left(\sum_{i=1}^{n} x_i < \theta\right)$ of the line & the ones that produce output 1 will be on the other side $\left(\sum_{i=1}^{n} x_i \geq \theta\right)$ of the line



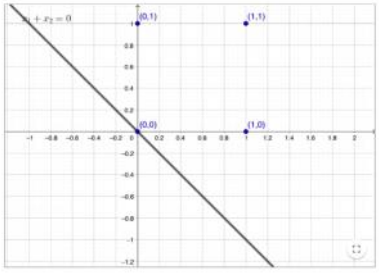
AND function

$x_1 + x_2 = \sum_{i=1}^{2} x_i \geq 2$





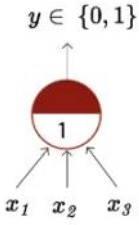
Tautology (always ON)



What if we have more than 2 inputs?

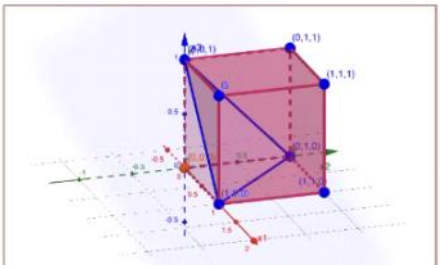
Then instead of a line we'll have a plane.

For the OR function we want a plane such that point (0,0,0) lies on one side & remaining 7 points lie on the other side.

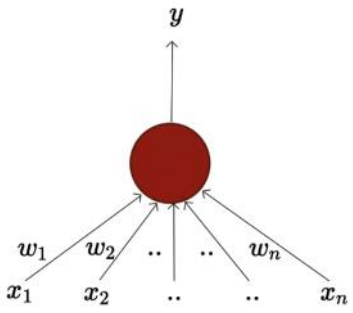


OR

$x_1 + x_2 + x_3 = \theta = 1$



Perceptrons



A more general computational model than McCulloch-Pitts neurons

Difference: Weights

Inputs are no longer boolean

$y = 1$ if $\sum_{i=1}^{n} w_i * x_i \geq \theta$

$= 0$ if $\sum_{i=1}^{n} w_i * x_i < \theta$

$y = 1$ if $\sum_{i=1}^{n} w_i * x_i - \theta \geq 0$

$= 0$ if $\sum_{i=1}^{n} w_i * x_i - \theta < 0$

$$y = 1 \text{ if } \sum_{i=0}^{n} w_i * x_i \geq 0$$

$$= 0 \text{ if } \sum_{i=0}^{n} w_i * x_i < 0$$

$x_0 = 1$

$w_0 = -\theta$

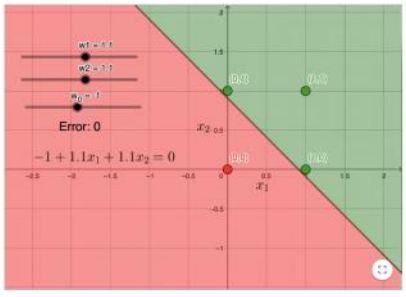
↓

bias as it represents the prior (prejudice)

x_1	x_2	OR	
0	0	0	$w_0 + \sum_{i=1}^{2} w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$
1	1	1	$w_0 + \sum_{i=1}^{2} w_i x_i \geq 0$

$$w_0 + w_1.0 + w_2.0 < 0 \implies w_0 < 0$$
$$w_0 + w_1.0 + w_2.1 \geq 0 \implies w_2 \geq -w_0$$
$$w_0 + w_1.1 + w_2.0 \geq 0 \implies w_1 \geq -w_0$$
$$w_0 + w_1.1 + w_2.1 \geq 0 \implies w_1 + w_2 \geq -w_0$$

One possible solution to this set of inequalities is $w_0 = -1, w_1 = 1.1, w_2 = 1.1$ (and various other solutions are possible)

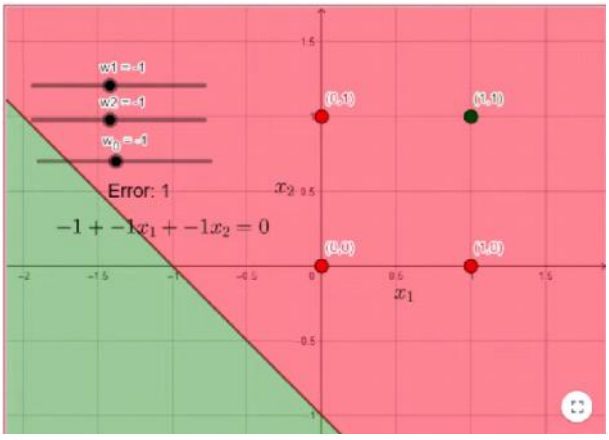


Note that we can come up with a similar set of inequalities and find the value of θ for a McCulloch Pitts neuron also (Try it!)

Errors & Error Surfaces

Let us fix the threshold ($w_0 = -1$) & try different values of w_1 & w_2

Let $w_1 = -1$, $w_2 = -1$. No. of errors = 1



Similarly,

w_1	w_2	errors
-1	-1	1
1.5	0	1
10	-10	2

we want the values of w_1 & w_2 where errors is 0

Perceptron Learning Algorithm

```
P ← inputs with label 1
N ← inputs with label 0
Initialize w randomly;
while !convergence do
    Pick random x ∈ P∪N
    if x ∈ P & Σ_{i=0}^{n} w_i x_i > 0 then
        w = w + x
    end
    if x ∈ N & Σ_{i=0}^{n} w_i x_i ≥ 0 then
        w = w - x
    end
end
//the algorithm will run until convergence [when all points have been correctly classified]
```

convergence

$P \rightarrow \sum_{i=0}^{n} w_i x_i > 0$

$N \rightarrow \sum_{i=0}^{n} w_i x_i < 0$

data has been clearly separated!

$x = [x_0\ x_1\ \cdots\ x_n]$

$w = [w_0\ w_1\ \cdots\ w_n]$

Why does it work?

Consider the two vectors w & x

$$w = [w_0, w_1, \cdots, w_n], \quad x = [1, x_0, x_1, \cdots, x_n]$$

$$w \cdot x = w^T x = \sum_{i=0}^{n} w_i x_i$$

we can rewrite the perceptron rule as:

$y = 1$ if $w^T x \geq 0$

$= 0$ if $w^T x < 0$

We are interested in finding the line $w^T x = 0$ which divides the input space into two halves

Every point (x) that lies on line satisfies the eqn

Every point (x) that lies on line satisfies the eqn

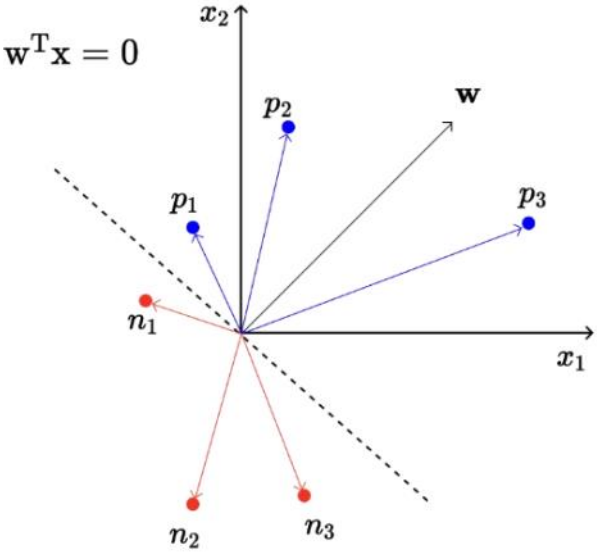
$w^T x = 0$

What can we say about the angle (α) b/w w & any point (x) which lies on this line?

The angle is $90°$ $\left[\cos\alpha = \frac{w^T x}{\|w\|\|x\|} = 0\right]$

Angle b/w points lying on the positive half space of the line & w?
Less than 90°

Angle b/w points lying on the negative half space of the line & w?
Greater than 90°



Coming back to the algorithm,

For $x \in P$ if $w^T x < 0$ then it means that the angle (α) b/w x & w is $>90°$ but we want it to be less than 90°

So when $w_{new} = w + x$,

$$\cos(\alpha_{new}) \propto (w_{new})^T x$$
$$\propto (w + x)^T x$$
$$\propto w^T x + x^T x$$
$$\propto \cos(\alpha) + x^T x$$

$\therefore \cos(\alpha_{new}) > \cos(\alpha)$

Thus α_{new} will become less than α & this is what we wanted

Similarly

Similarly,

For $x \in P$ if $w^T x < 0$ then it means that the angle (α) b/w x & w is $< 90°$ but we want it to be more than $90°$

So when $w_{new} = w - x$,

$$\cos(\alpha_{new}) \propto (w_{new})^T x$$
$$\propto (w - x)^T x$$
$$\propto w^T x - x^T x$$
$$\propto \cos(\alpha) - x^T x$$

$$\therefore \cos(\alpha_{new}) < \cos(\alpha)$$

Thus α_{new} will become more than α & this is what we wanted

Proof of Convergence

Theorem

Two sets P & N of points in an n-dimensional space are called linearly separable if $n+1$ real numbers $w_0, w_1, \cdots, w_n$ exist such that every point $(x_1, x_2, \cdots, x_n) \in P$ satisfies $\sum_{i=1}^{n} w_i * x_i \geq w_0$ & every point $(x_1, x_2, \cdots, x_n) \in N$ satisfies $\sum_{i=1}^{n} w_i * x_i < w_0$

Assumption

Data is linearly separable

Preposition

If the sets P & N are finite & linearly separable, the perceptron learning algorithm updates the weight vector w a finite number of times.

Proof

If $x \in N$ then $-x \in P$ $\left[\because w^T x < 0 \Rightarrow w^T(-x) \geq 0\right]$

We can consider a single set,

$P' = P \cup N^-$

& for every $p \in P'$ ensure that

$\boxed{w^T p \geq 0}$

Further, we normalize all the p's so that $\|p\| = 1$

Algorithm: Perceptron Learning Algorithm

$P \leftarrow$ *inputs with label* 1;
$N \leftarrow$ *inputs with label* 0;
$N^- \leftarrow$ *negations of all points in* N;
$P' \leftarrow P \cup N^-$
Initialize **w** randomly;
while !*convergence* **do**
  Pick random $\mathbf{p} \in P'$
  $\mathbf{p} \leftarrow \frac{\mathbf{p}}{\|p\|}$ (so now, $\| \mathbf{p} \| = 1$)
  if $\sum_{i=0}^{n} w_i * p_i < 0$ **then**
    $\mathbf{w} = \mathbf{w} + \mathbf{p}$;
  end
end
//the algorithm converges when all the inputs are classified correctly
//notice that we do not need the other **if** condition because by construction we want all points in P' to lie

Observations

w^* is the optimal solution which exists but we don't know what that is

Proof

Suppose at the time of step t, we inspected the point $P_i \in P'$ & found that $w^T P_i < 0$.

We make the correction: $w_{t+1} = w_t + P_i$

Let β be the angle b/w w^* & w_{t+1}

$$\cos\beta = \frac{w^* \cdot w_{t+1}}{\|w_{t+1}\|}$$

Numerator: $w^* \cdot w_{t+1}$

$= w^* \cdot (w_t + P_i)$

$= w^* w_t + w^* P_i$

$w^* w_t + w^* P_i \geq w^* w_t + \delta$

$\geq w^* \cdot (w_{t-1} + P_j) + \delta$

$\geq w^* \cdot w_{t-1} + w^* P_j + \delta$

$\geq w^* \cdot w_{t-1} + 2\delta$

Numerator $\geq w^* \cdot w_0 + (k)\delta$

Denominator^2 $= \|w_{t+1}\|^2$

$= (w_t + P_i)(w_t + P_i)$

$= \|w_t\|^2 + 2 w_t \cdot P_i + \|P_i\|^2$

$\leq \|w_t\|^2 + \|P_i\|^2$

$\leq \|w_{t-1}\|^2 + \|P_j\|^2 + \|P_i\|^2$

$\leq \|w_0\|^2 + k\|P_i\|^2$

For all points belonging to P' let's multiply w^*

$w^* \cdot P_1$, $w^* \cdot P_2$, ..., $w^* \cdot P_m$

The products will be some scalar $\in \mathbb{R}$. Let's call the minimum of these dot products as 'δ'

$\delta = \min(w^* \cdot P_i \mid \forall i)$

Now $w^* \cdot P_1, w^* \cdot P_2, \ldots, w^* P_m \leq \delta$ as it is the minimum value

$k \leq t$ as the k is the no. of updates of weight vector & t is the no. of steps

The correction is made only if $w^* \cdot P_i \leq 0$ at the step & we get a δ

$\therefore w_t \cdot P_i \leq 0$

$$= \|w_t\| + \|p_i\|$$

$$\leq \|w_t\|^2 + 1$$

$$\leq \left(\|w_{t-1}\|^2 + 1\right) + 1$$

$$\leq \|w_{t-1}\|^2 + 2$$

Denominator

$$\leq \|w_0\|^2 + k$$

$$\therefore \cos\beta \geq \frac{w^* w_0 + k\delta}{\sqrt{\|w_0\|^2 + k}}$$

$\because w_t \cdot p_i \leq 0$

$\because \|p_i\|^2 = 1$

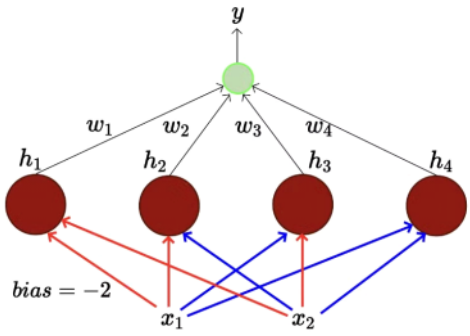
By the same observation that we made about δ

$\cos\beta$ grows proportionally to $\sqrt{k}$

As k (no. of corrections) increase, $\cos\beta$ will become large

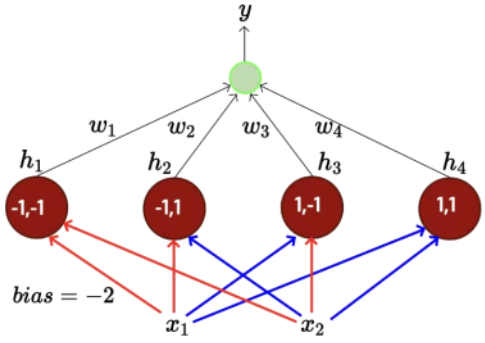
But since $\cos\beta \leq 1$, k must be bounded by a maximum no.

Thus, there can only be a finite no. of updates/corrections & the algorithm will converge!



red edge indicates w = -1

blue edge indicates w = +1



red edge indicates w = -1

blue edge indicates w = +1

Terminology:

This network contains 3 layers

The layer containing the inputs (x_1, x_2) is called the input layer

The middle layer containing the 4 perceptrons is called the hidden layer

The final layer containing one output neuron is called the output layer

The outputs of the 4 perceptrons in the hidden layer are denoted by h_1, h_2, h_3, h_4

The red and blue edges are called layer 1 weights

w_1, w_2, w_3, w_4 are called layer 2 weights

We claim that this network can be used to implement any boolean function (linearly separable or not)!

In other words, we can find w_1, w_2, w_3, w_4 such that the truth table of any boolean function can be represented by this network

Astonishing claim! Well, not really, if you understand what is going on

Each perceptron in the middle layer fires only for a specific input (and no two perceptrons fire for the same input)

the fourth perceptron fires for {1,1}