Perceptron :

- Data we need to work with:

  linearly seperable.

  yes → 1

  no → 0

- artificial neuron.

* simplest neural network.

* neurons of brain :

receive
$\boxed{\text{input}}$ — $\boxed{\text{output}}$ based on previous inputs.
store info

* weights - represent the importance of each input.

* sum / aggregate of all these values must be greater than a threshold to make the decision. (weighted sum)

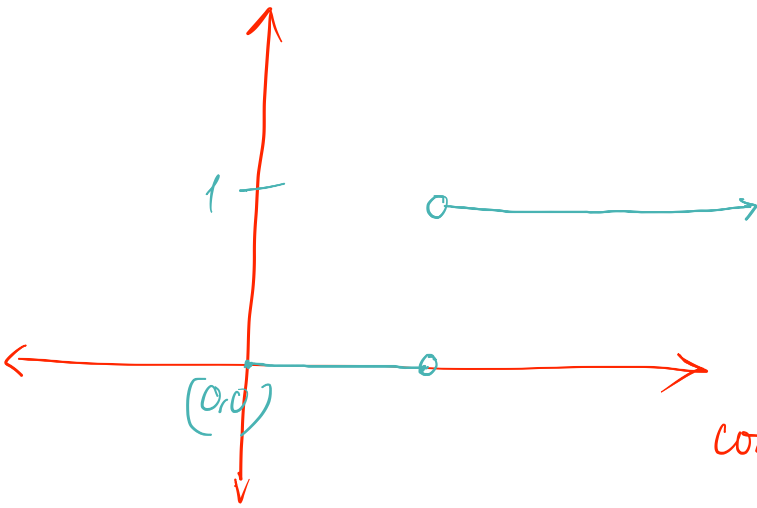
Ex: if a student gets 3 out of 5 points, the student's pass (1) the test that person's fail (0). So, the threshold here is 3.

* aggregated sum is like calculating the marks of student based on each of their responses (inputs) using marks (weights).

→ activation function introduces non-linearity so that we can get outputs other just 0 or 1.

if sum is > threshold → output is 1 (fires)

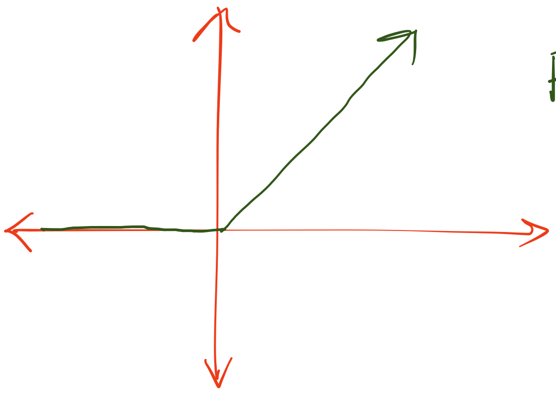
else output is 0 (doesn't fire).



STEP FUNCTION

* use for binary classifications.
* Lacks non-linearity. (so can't learn/work with complex patterns).

Other activation functions:



Relu - Rectified Linear Unit. → sets -ve values to 0.

if sum < 0 → output is 0.

else output is +ve number.

$$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$

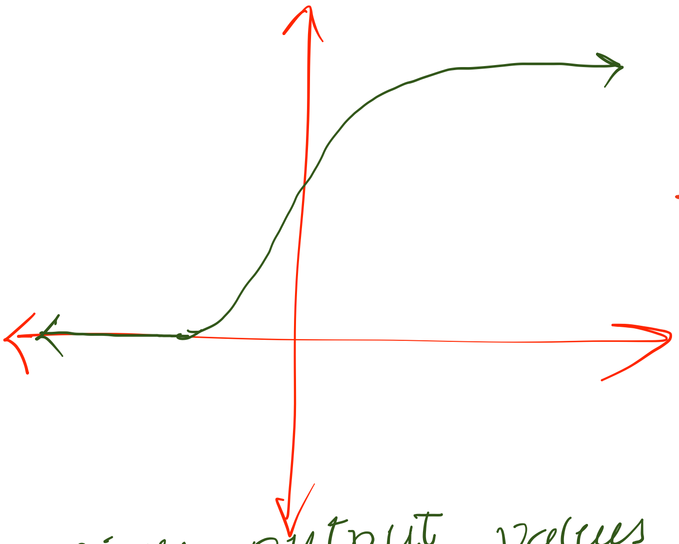
* non-linearity ✓.
* avoids vanishing gradient problems (unlike sigmoid or Tanh).
* Faster training.

* Dying Relu problem:

if many units become inactive and output 0 (for -ve inputs or 0), they wouldn't contribute to learning. This results in dead neurons.

Sigmoid function:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



gives output values b/w $[0, 1]$.

* used when u need a probability output [cuz output is in b/w 0 and 1 incl.].
* continuous & differentiable. (useful in gradient descent / backpropagation).
* vanishing gradient problem (for large +ve/-ve values gives very small gradients → slows down learning or causes ~~the~~ models to get stuck).
* Not 0 centered (always +ve or always -ve) (not just for sigmoid this).

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

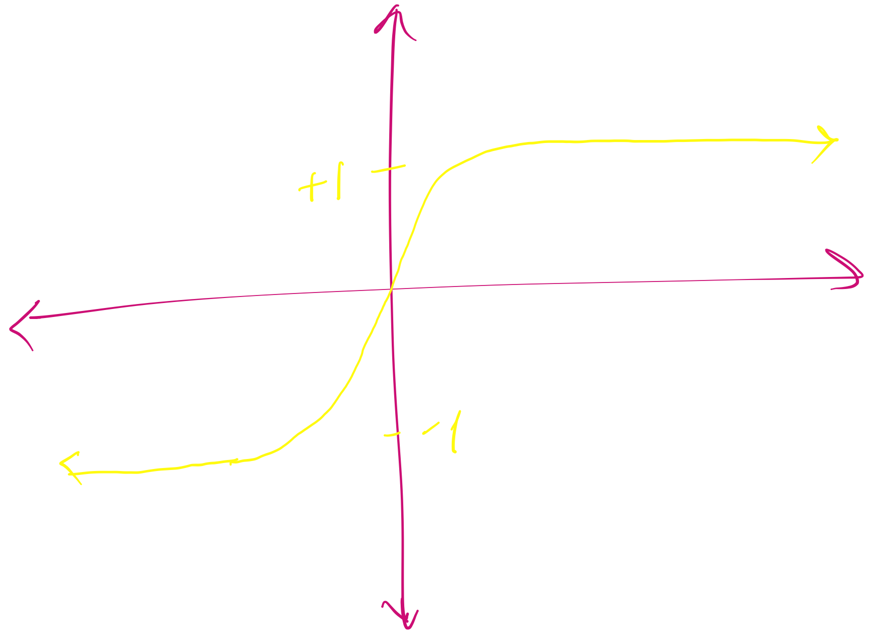
$$\lim_{x \to -\infty} \frac{e^x - e^{-x}}{e^x + e^{-x}} < 0 \sim -1.$$

$$\lim_{x \to +\infty} \frac{e^x - e^{-x}}{e^x + e^{-x}} > 0 \sim 1.$$

Range → $[-1, 1]$

* 0-centered. (improves convergence).
* Gradient based optimization → continuous and differentiable.
* vanishing gradient problem. $\left(\frac{\partial L}{\partial w}\right)$
* can lead to slower training due to complex calculation.

$$\frac{e^{2x}-1}{e^{2x}+1} \text{ or } \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



Loss: Difference b/w predicted vs true value.

* calculated using mean-squared error or cross-entropy. Also called cost function.

Back propagation: $\frac{\partial L}{\partial W}$.

Gradients & weight updates:

sigmoid derivative: $\frac{d}{dx}\sigma(x) = \sigma(x)\left[1-\sigma(x)\right]$.

$$\frac{d}{dx}(\tanh(x)) = 1 - \tanh^2(x).$$

---

weight updates:

$$w_{new} = w_{old} + \eta \cdot \nabla L(w).$$

sign doesn't matter cuz if

$\nabla L(w)$ is <0 / should be reduced,

that effect shows up.
