PERCEPTRON (F. Rosenblatt 1957)

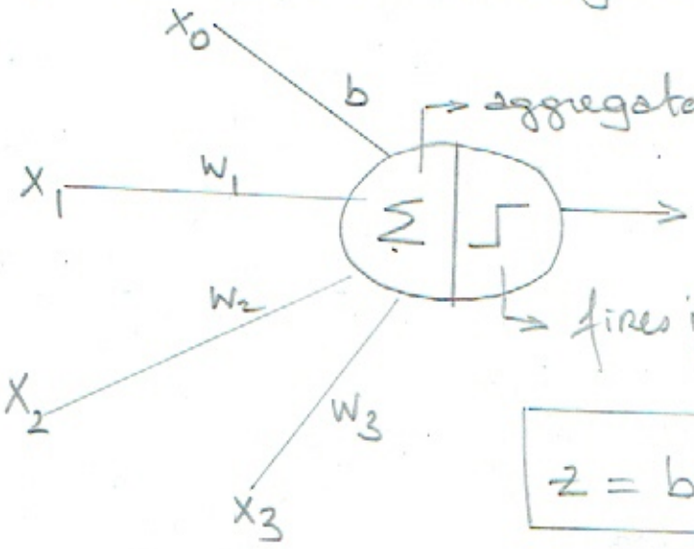
↳ LINEAR BINARY Classifier

↳ hyperplane H separates 2 classes ⊕ $y=1$ ⊖ $y=0$

↳ if there exists a hyperplane H that separates the 2 classes you will find it!

→ there ARE INFINITELY many!

→ PERCEPTRON Convergence THEORETM

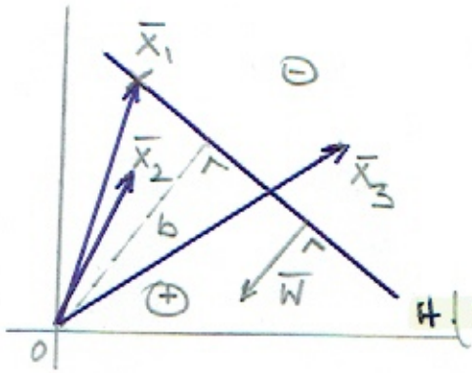


$$z = b + \sum_i w_i^T x_i$$

set $x_0 = 1$, $b = w_0$

↳ - threshold

$$z = \sum_i w_i^T x_i$$



↳ distance from ORIGIN.

↳ points to positive class by convention!

UNIT weight VECTOR

on H: $-w^T x_1 = b \rightarrow w^T x_1 + b = 0$ → PER CONVENTION belong to ⊖ class

$-w^T x_2 < b \rightarrow w^T x_2 + b > 0$

$-w^T x_3 > b \rightarrow w^T x_3 + b < 0$

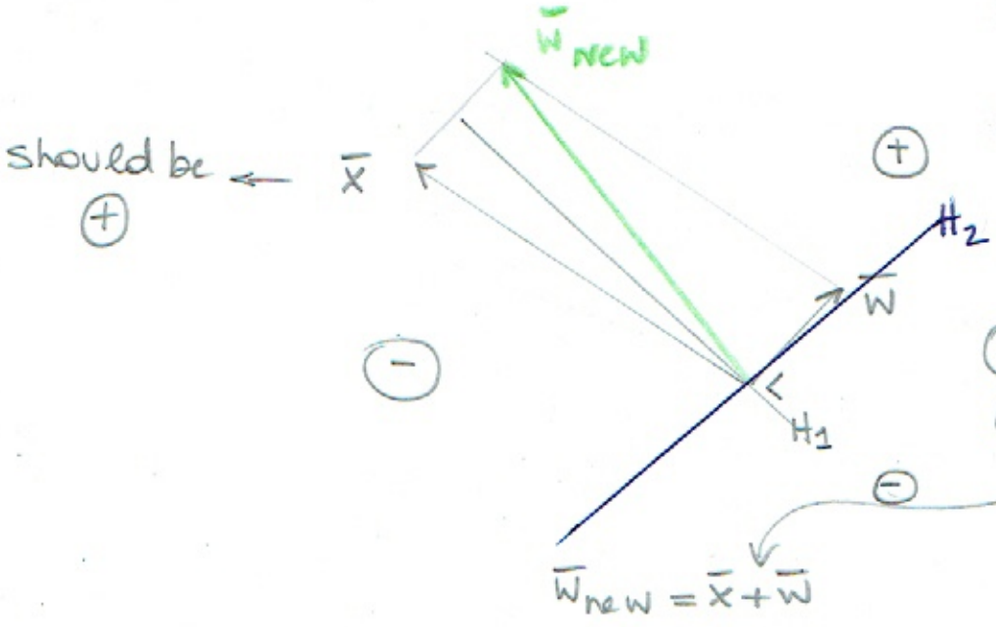
if H through origin

ON H: $w^T x = 0$

$w^T x > 0$ ⊕ class

$w^T x \leq 0$ ⊖ class

perceptron learning Algorithm



① start with random w

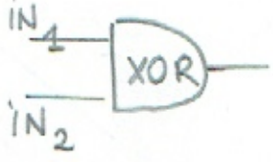
② Evaluate $\bar{x}$

③ if $\bar{x}$ wrongly classified

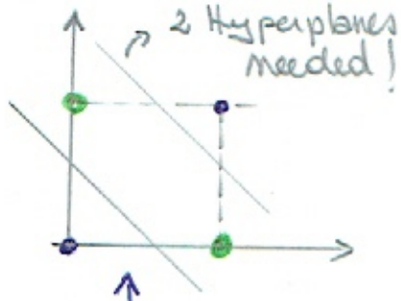
④ Repeat UNTIL all points are correctly classified

Limitations of perceptron → Minsky & Papart 1969

XOR problem XOR = exclusive OR port



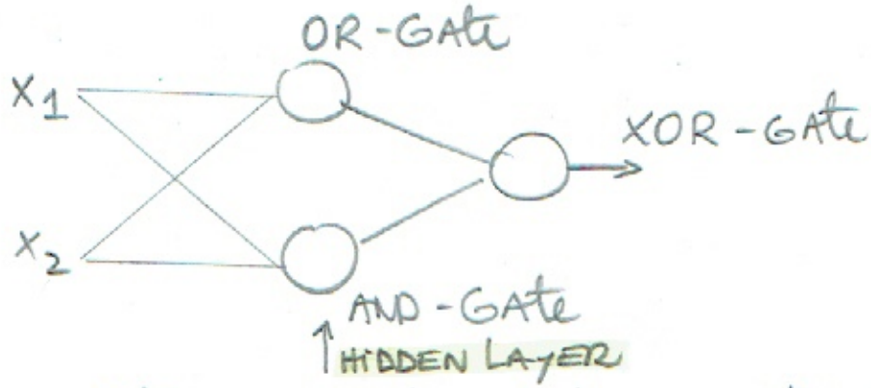
IN_1	IN_2	OUT
0	0	0
1	1	0
0	1	1
1	0	1



NEED FOR NON-LINEARITY ← YOU CANNOT separate classes WITH 1 LINEAR hyperplane!

Solution → **Multi-layer Perceptron**

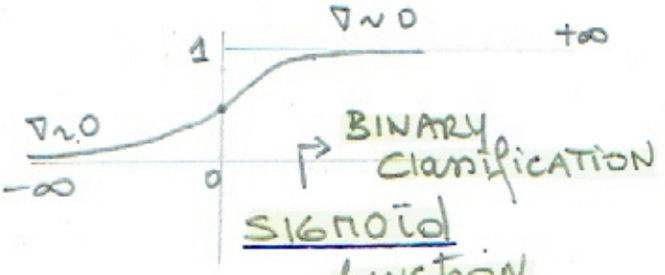
↳ ① provides non-linearity through INTRODUCTION of HIDDEN LAYERS + ACTIVATION FUNCTIONS



AND, OR Gates can be trained by perceptron learning rule!
HOWEVER whole network CANNOT!

Activation Function

① $$f(x) = \frac{1}{1+e^{-x}}$$



SIGMOID FUNCTION

→ smooth, continuous & differentiable!

↓
at output hidden layer
↓
NON-LINEARITY!

but: e^{-x} → expensive

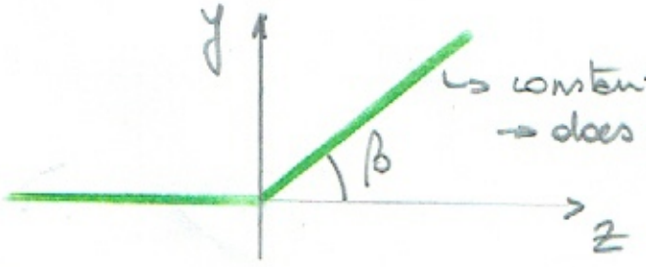
OUTPUTS close to 0 and 1 → kills gradient descent!
linear SCORE → probability!

② Softmax → multi-class classification

↳ VECTOR of K real NUMBERS → probability distribution of K probabilities

③ ReLu → rectified linear unit

VERY POPULAR! for hidden layers!



↳ constant ∇
→ does not SATURATE IN positive region!

$$y = \max(0, z) \quad \begin{cases} z < 0 \rightarrow 0 \\ z > 0 \rightarrow y \end{cases}$$

$$y = \max(0, w^T x + b)$$

less computationally expensive
Faster because constant ∇