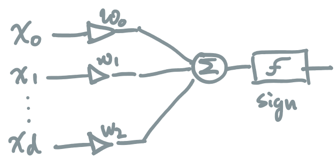
Neural Networks

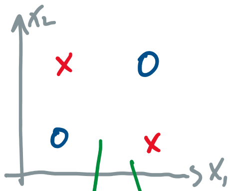
Tuesday, February 4, 2020 12:55

▷ allows non linear models

Recall PERCEPTRON Model:

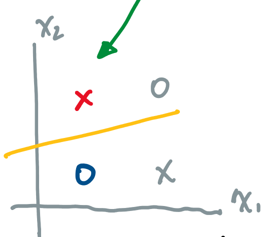


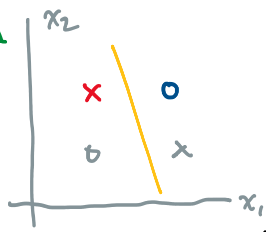
What if we have:



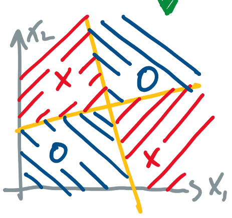
Let's split into two perceptrons

$\Rightarrow$ two different weights & hypothesis





$h_1(\underline{x}) = \text{sign}(\underline{w}_1^T \underline{x})$ $\quad$ $h_2(\underline{x}) = \text{sign}(\underline{w}_2^T \underline{x})$



The "togetherness" of the two becomes a binary logic:

TRUTH TABLE:

h_1	h_2	y
−1	−1	−1
−1	+1	+1
+1	−1	+1
+1	+1	−1

← this is an XOR gate

also $XOR(a,b) = (\neg a \wedge b) \vee (a \wedge \neg b)$

So the function we want to classify the problem is an XOR-function of h_1 and h_2:

$$f = OR(AND(\neg a, b), AND(a, \neg b))$$

$f = OR(\, AND(\neg a, b),\ AND(\, a, \neg b))$

ⓘ WE CAN IMPLEMENT **AND** AND **OR** LOGIC USING PERCEPTRONS

$AND(x_1, x_2) = sign(\underline{x_1 + x_2 - 1.5})$

x_1	x_2	AND
−1	−1	−1
−1	+1	−1
+1	−1	−1
+1	+1	+1

NOTE: this looks like a perceptron $w^T x$ where $\boxed{w_0 = -1.5,\ w_1 = 1,\ w_2 = 1}$

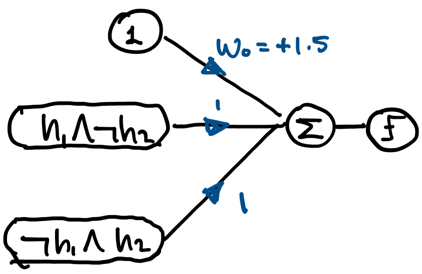
$OR(x_1, x_2) = (\, x_1 + x_2 + 1.5)$

x_1	x_2	OR
−1	−1	−1
−1	+1	+1
+1	−1	+1
+1	+1	+1

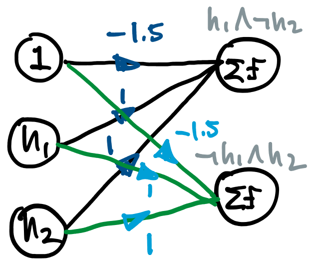
$\underline{w} = \begin{bmatrix} w_0 = +1.5 \\ w_1 = 1 \\ w_2 = 1 \end{bmatrix}$

So back to the problem, let's implement the outer OR gate first:

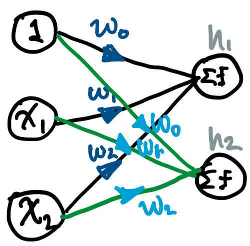
OUTER OR:



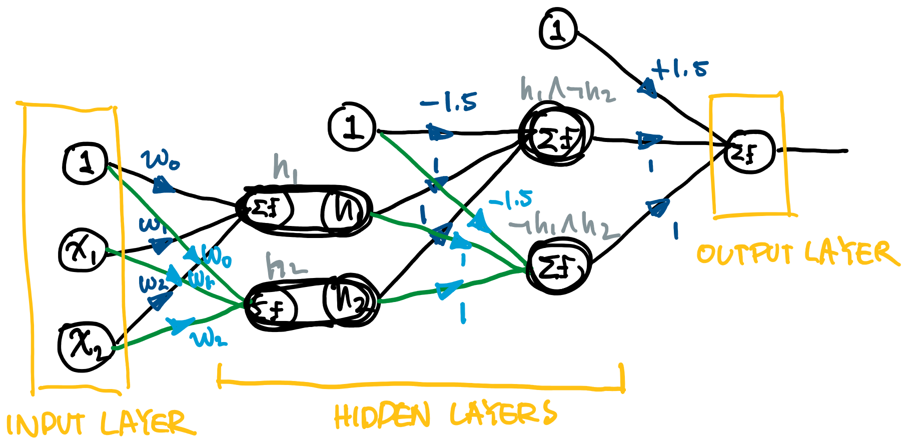
BREAK IT DOWN FURTHER:



EVEN FURTHER:



PUTTING IT ALL TOGETHER: **MULTILAYER PERCEPTRON**



UNIVERSAL APPROXIMATION THEOREM

- Large MLP with 2 hidden layers can approximate smooth target functions arbitraily well.