Machine Learning from Data

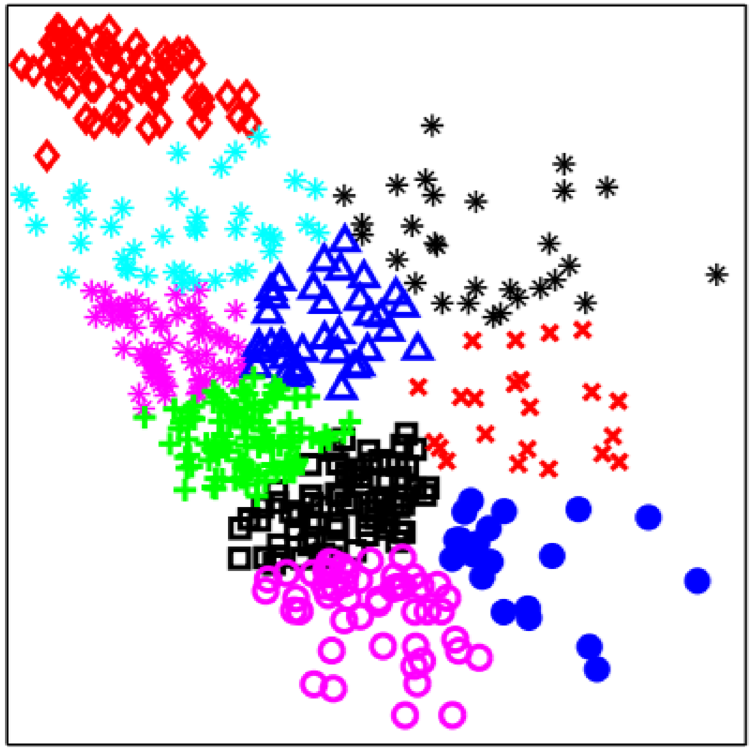
Lecture 20: Spring 2021

Today’s Lecture

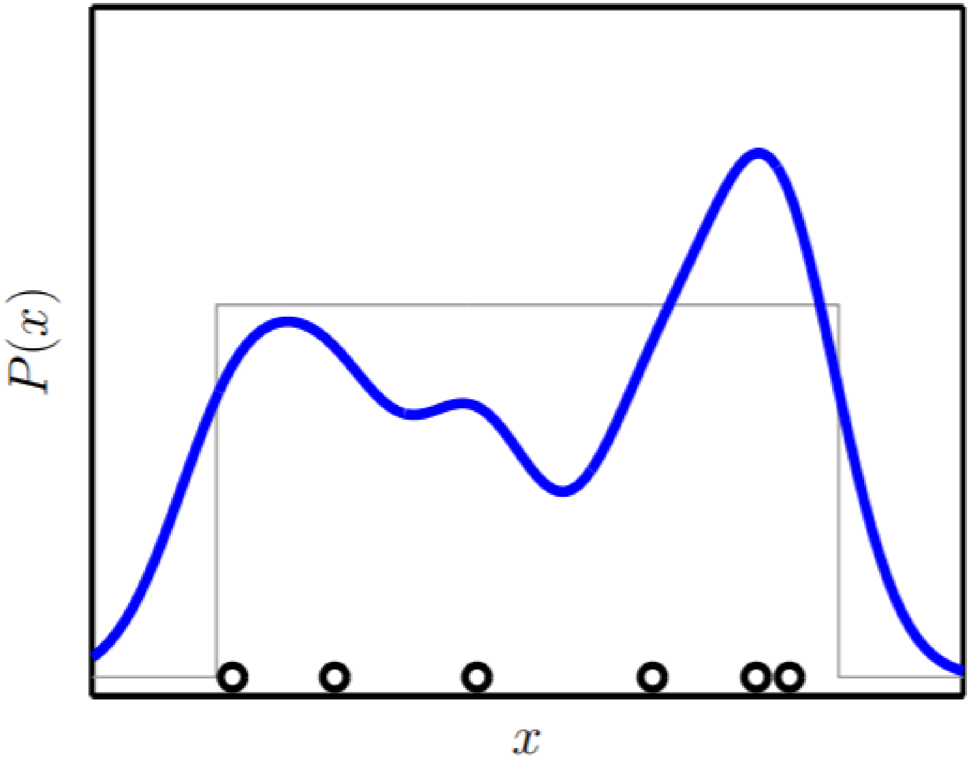
- Multilayer Perceptron
  - Multiple Layers
  - Approximation
  - Neural Network

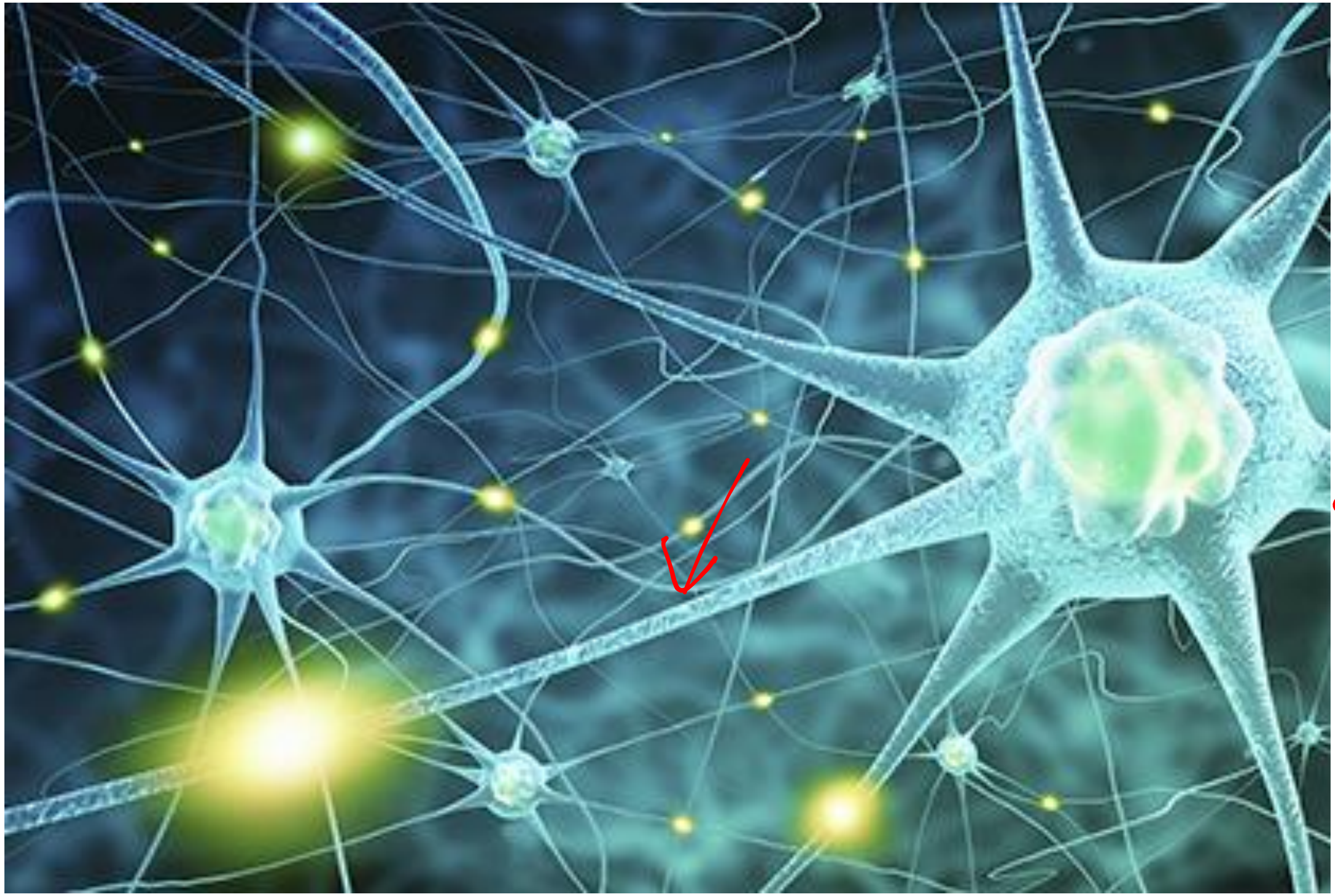
RECAP: **Unsupervised Learning**

k-Means Clustering ✓

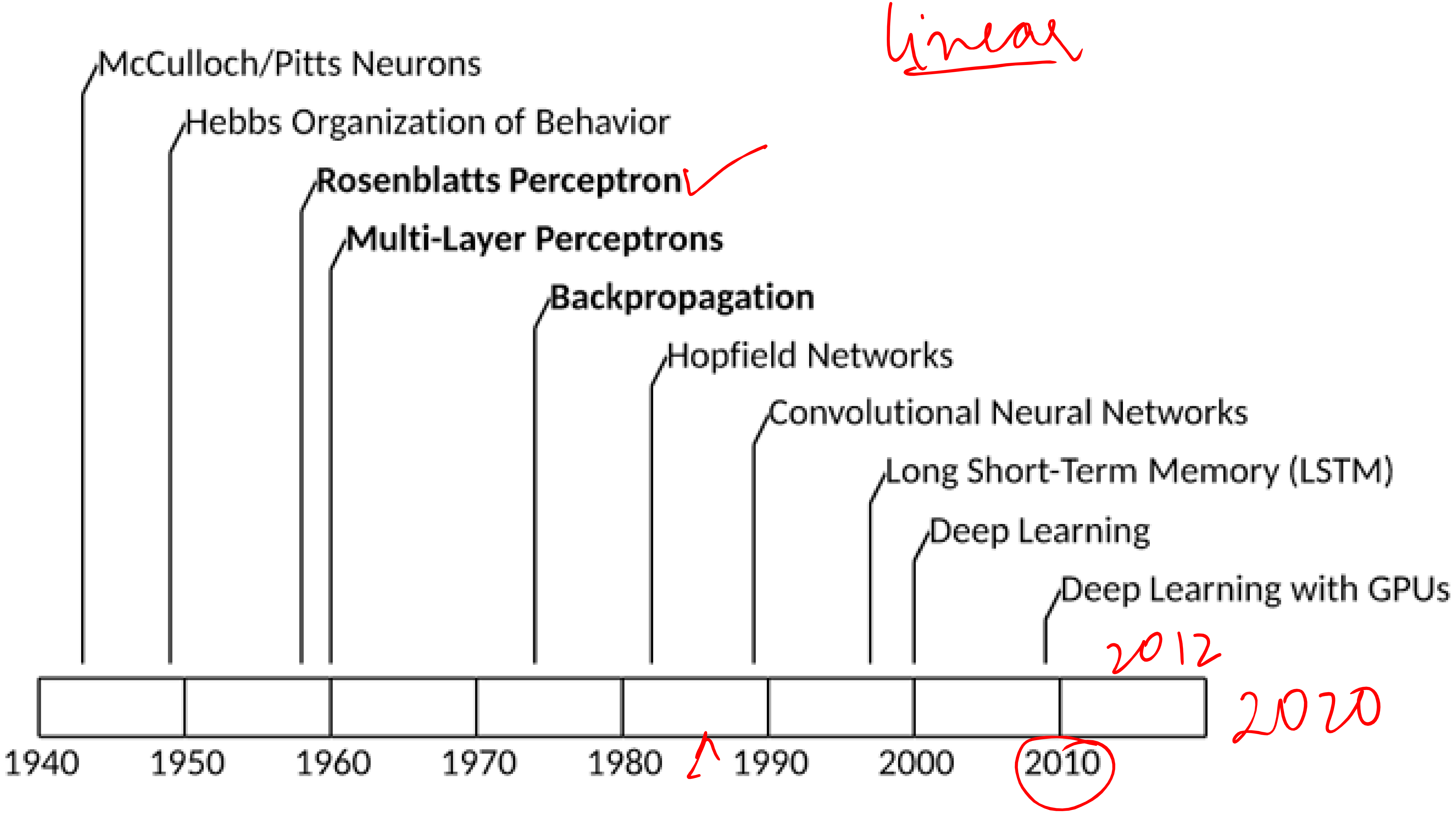


Gaussian Mixture Model ✓

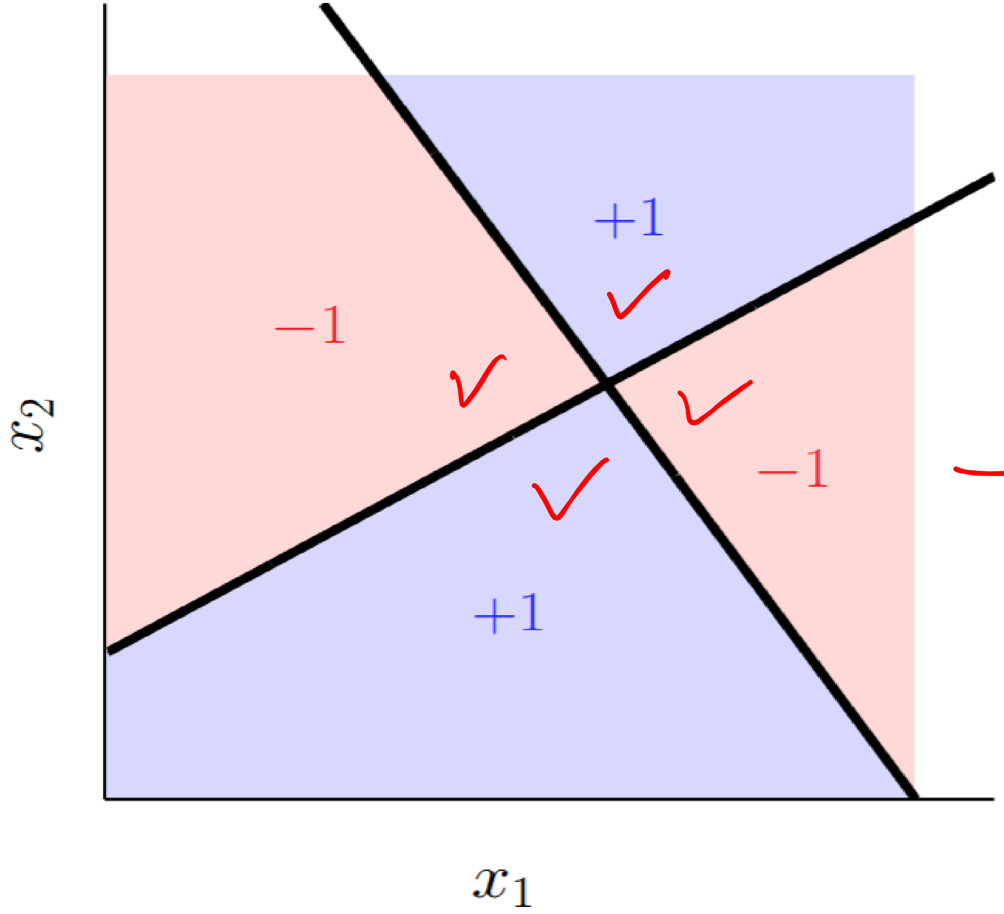




Neural Network – Biologically Inspired


linear
McCulloch/Pitts Neurons
Hebbs Organization of Behavior
Rosenblatts Perceptron
Multi-Layer Perceptrons
Backpropagation
Hopfield Networks
Convolutional Neural Networks
Long Short-Term Memory (LSTM)
Deep Learning
Deep Learning with GPUs
2012
2020
1940
1950
1960
1970
1980
1990
2000
2010

XOR Function: Limitation of the Linear Model

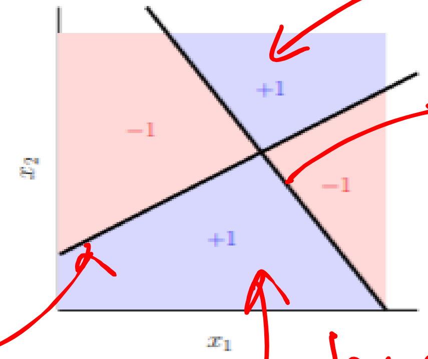


$f(x)$

→ transform

Decomposing XOR

f



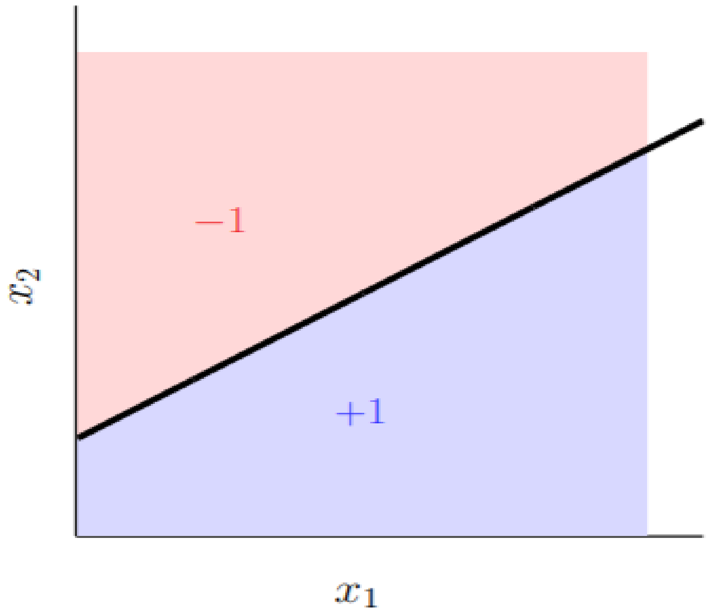
$h_2 = +1$
$h_1 = -1$

$h_1 = +1$ $h_2 =$ not +ve

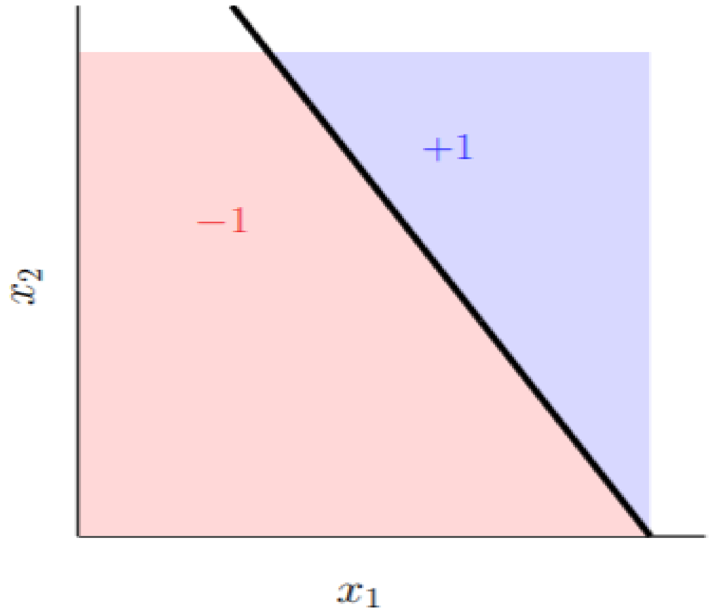
$$f = h_1\overline{h_2} + \overline{h_1}h_2$$

AND

OR



$h_1(\mathbf{x}) = \text{sign}(\mathbf{w}_1^{\mathrm{T}}\mathbf{x})$



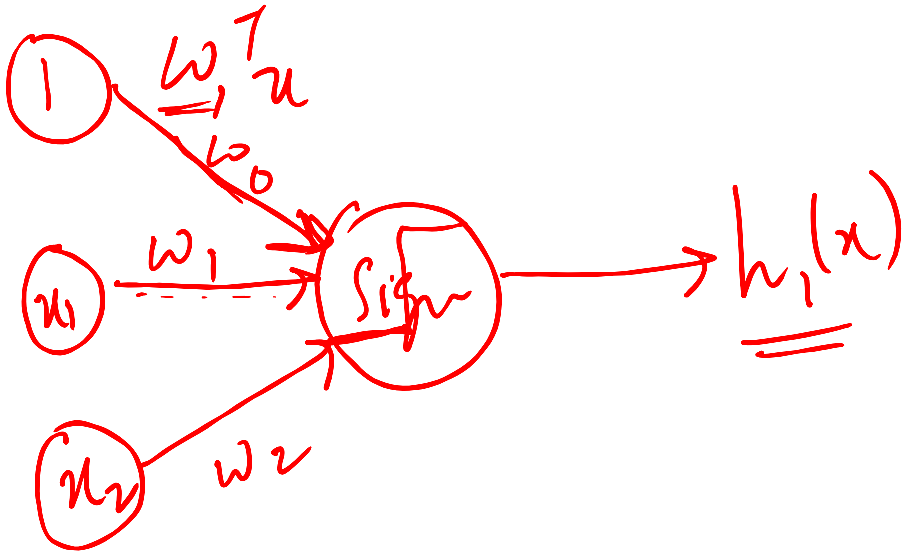
$h_2(\mathbf{x}) = \text{sign}(\mathbf{w}_2^{\mathrm{T}}\mathbf{x})$

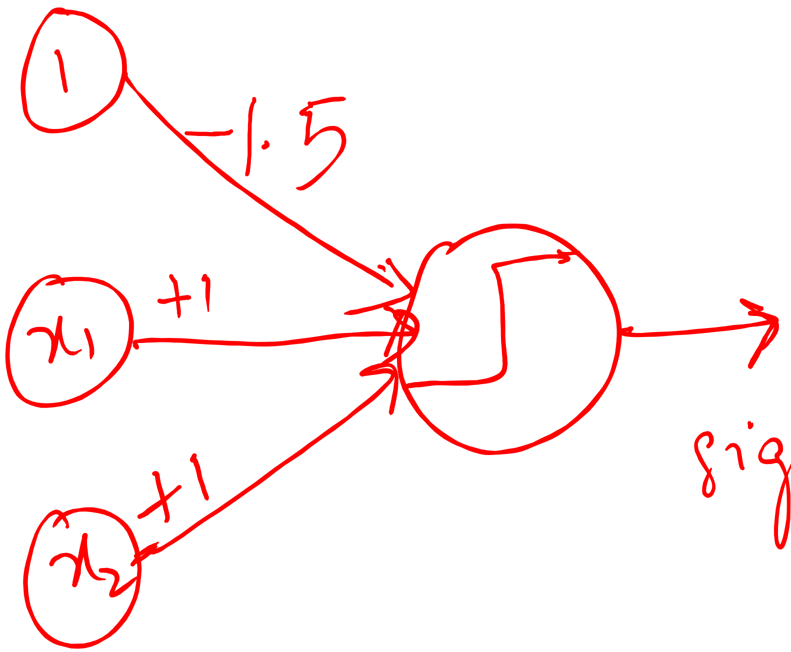
$$AND(x_1, x_2 \dots x_d) = \begin{cases} +1 & \text{if } x_1 = x_2 = \dots = x_d = +1 \\ -1 & \text{o.w.} \end{cases}$$

$$OR(x_1, x_2 \dots x_d) = \begin{cases} +1 & \text{o.w} \\ -1 & \text{if } x_1 = x_2 = \dots x_d = -1 \end{cases}$$

$$h_1(x) = \underline{\text{sign}}\,(w_1^T x)$$

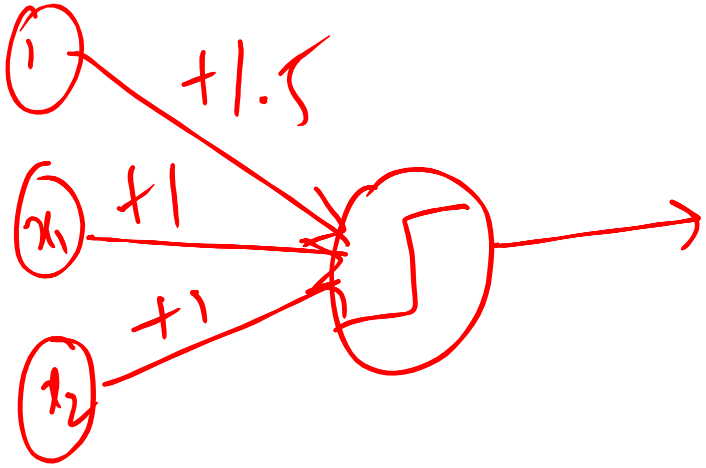
$$f(x) = \underbrace{\underbrace{h_1 \bar{h}_2}_{\text{AND}} \underset{\text{OR}}{+} \bar{h}_1 h_2}$$





$\text{AND}(x_1, x_2)$

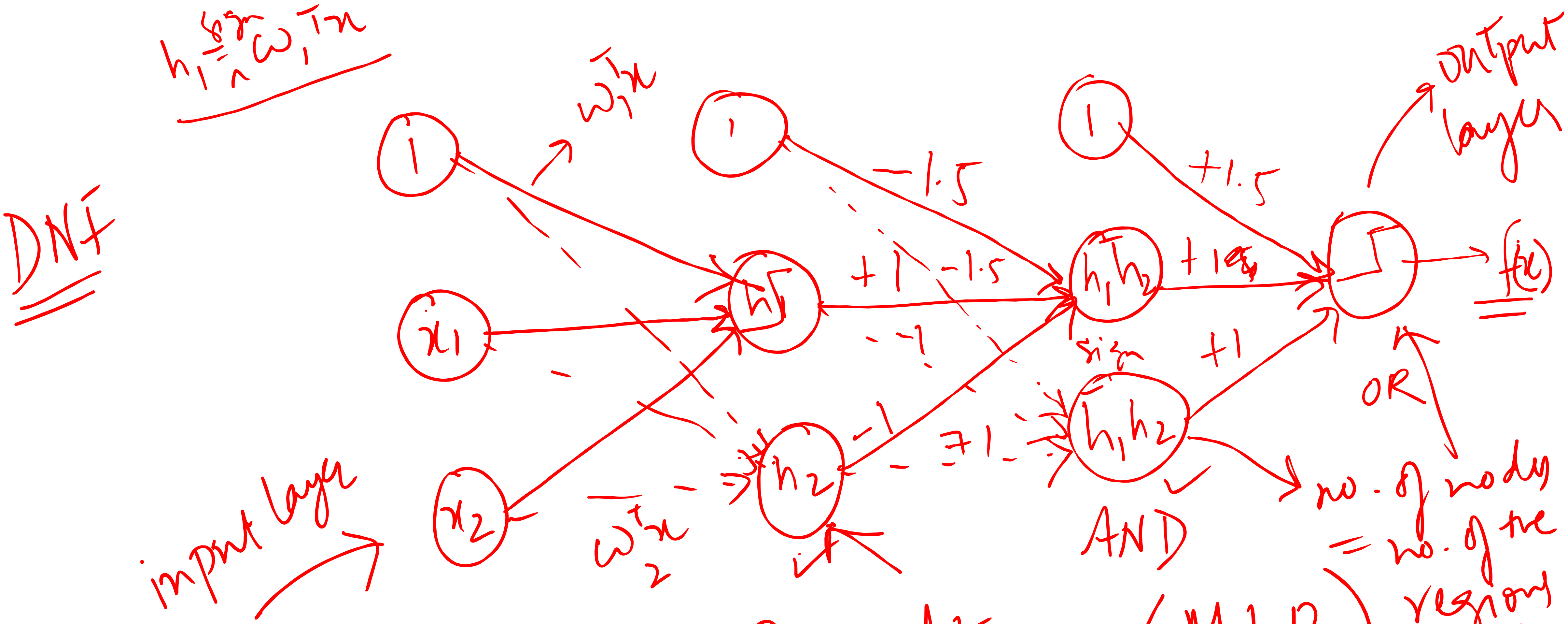
$\text{sign}(-1.5 + x_1 + x_2)$



$\text{OR}(x_1, x_2)$

$\downarrow$

$\text{sign}(1.5 + x_1 + x_2)$



Multilayer Perceptron (MLP)

The Multilayer Perceptron (MLP)

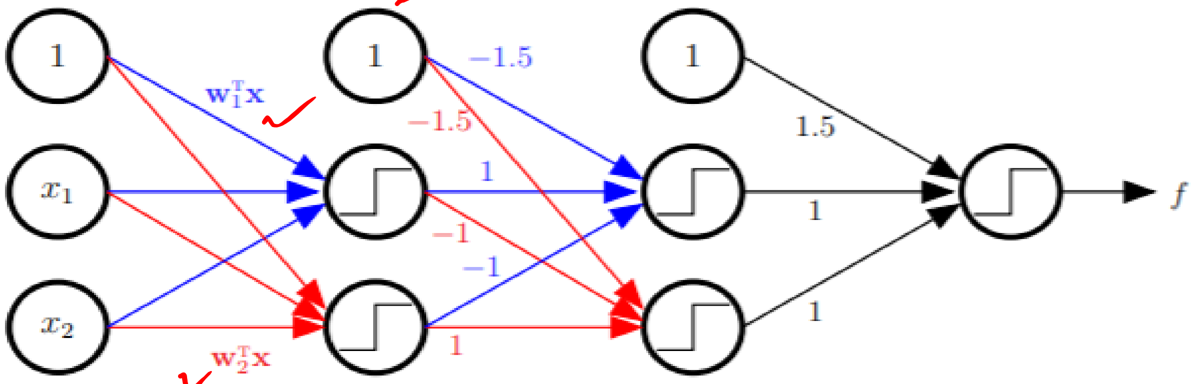
1st layer 16

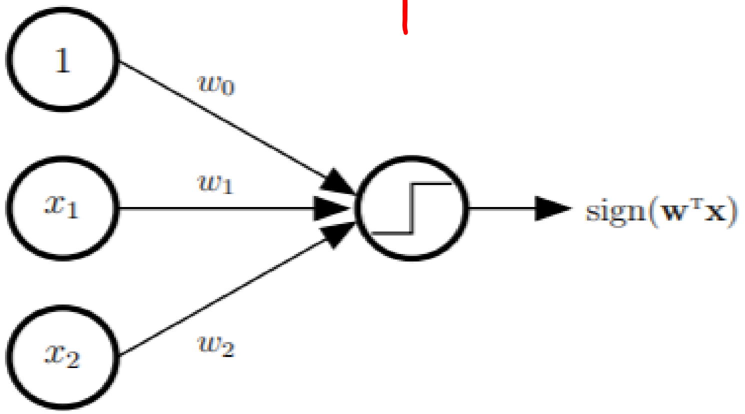
Deep

→ more layers

Wide

→ more nodes



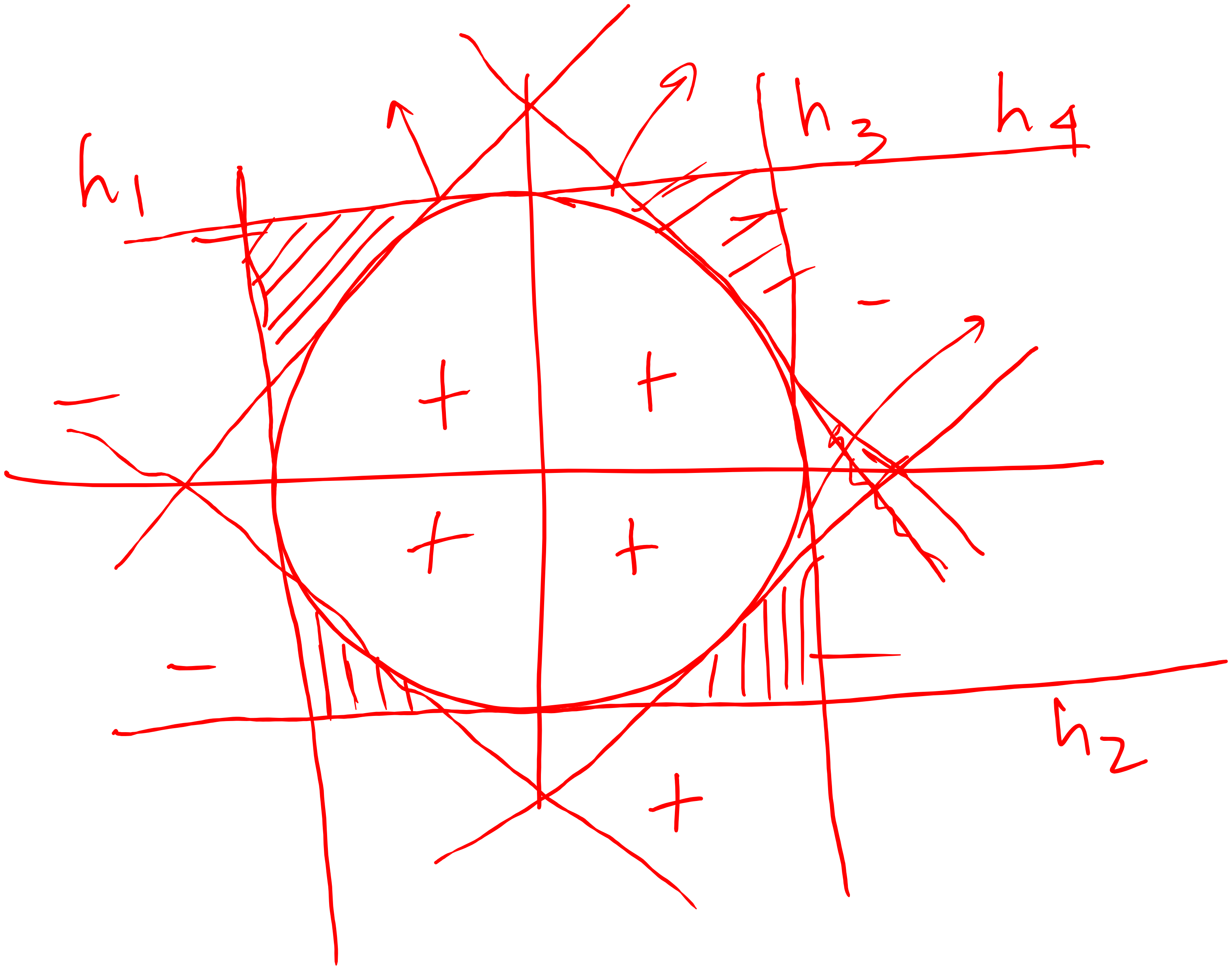


More layers allow us to implement f

These additional layers are called *hidden layers*

Universal Approximation

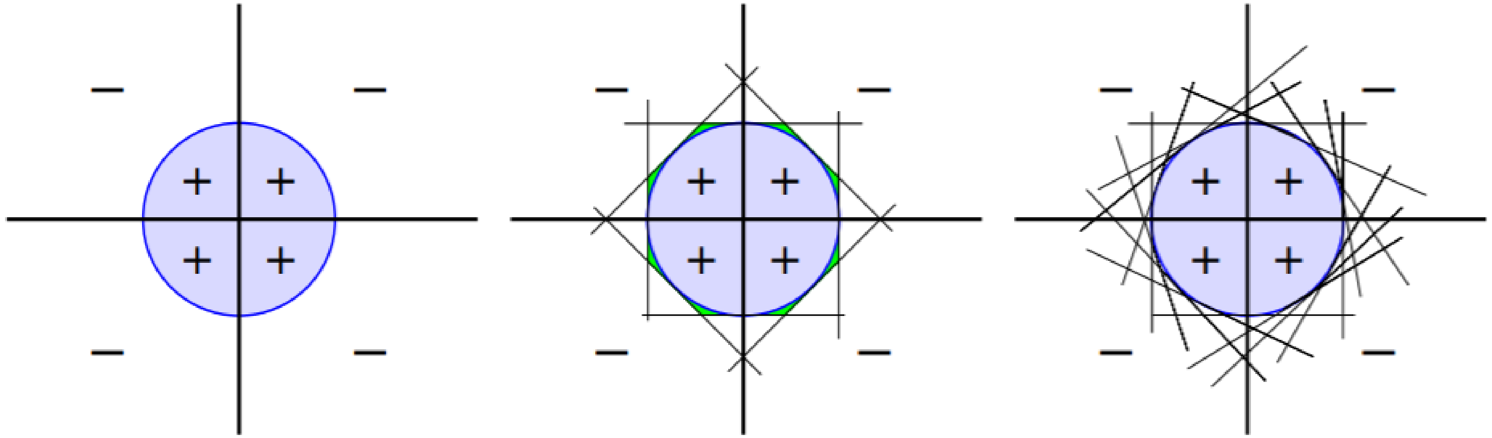
Any target function f that can be decomposed into linear separators can be implemented by a 3-layer MLP.



16 perceptions

Universal Approximation

A sufficiently smooth separator can "essentially" be decomposed into linear separators.



Target   8 perceptrons   16 perceptrons

Relax the ability to update in every layer $\longrightarrow$ good approximation power.

Generalization will/may suffer.

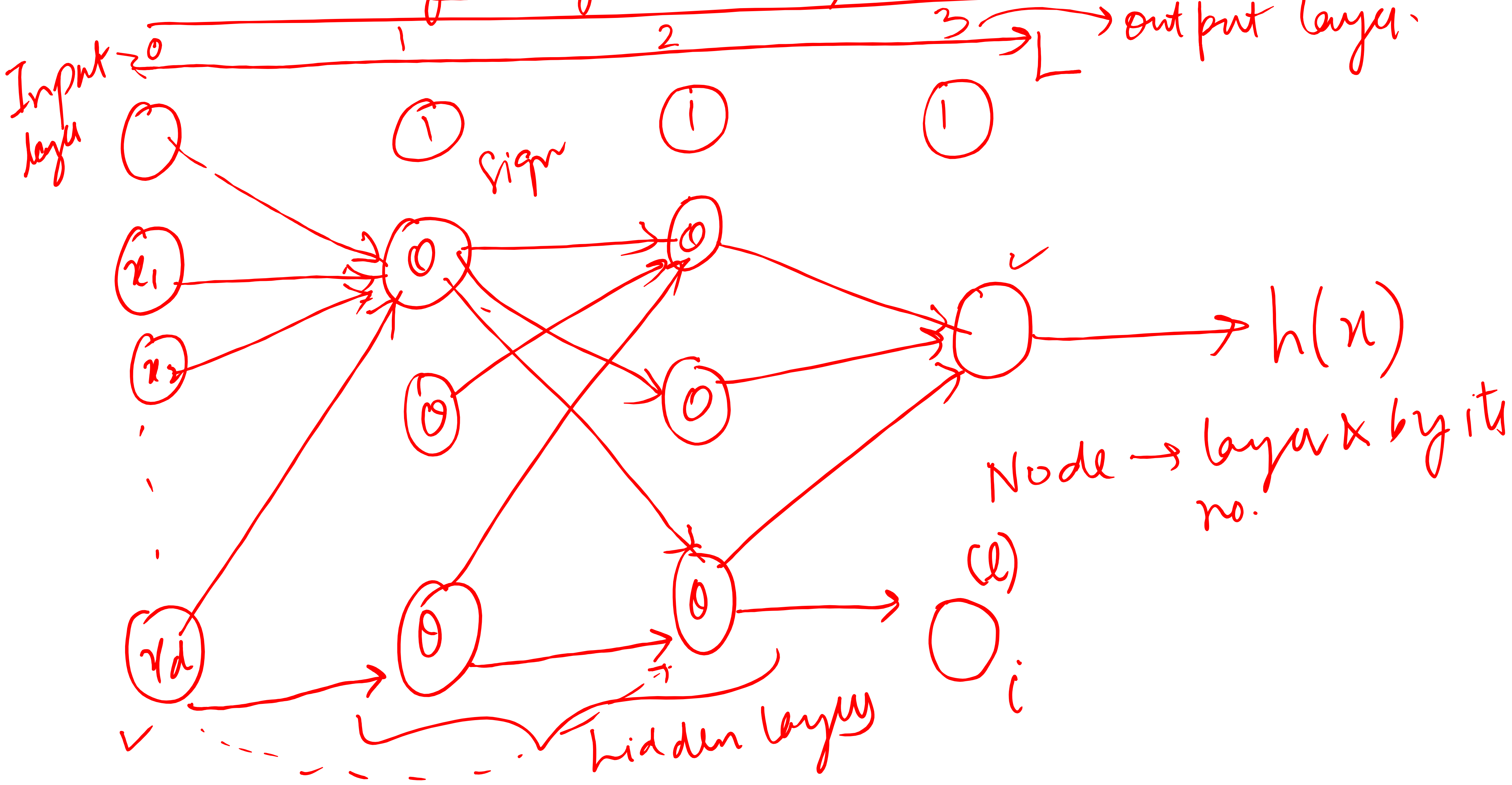
Approximation $\uparrow$ generalization $\downarrow$

Perceptron $\longrightarrow$ PLA / Pocket, Pseudo Inverse, Gradient Descent.

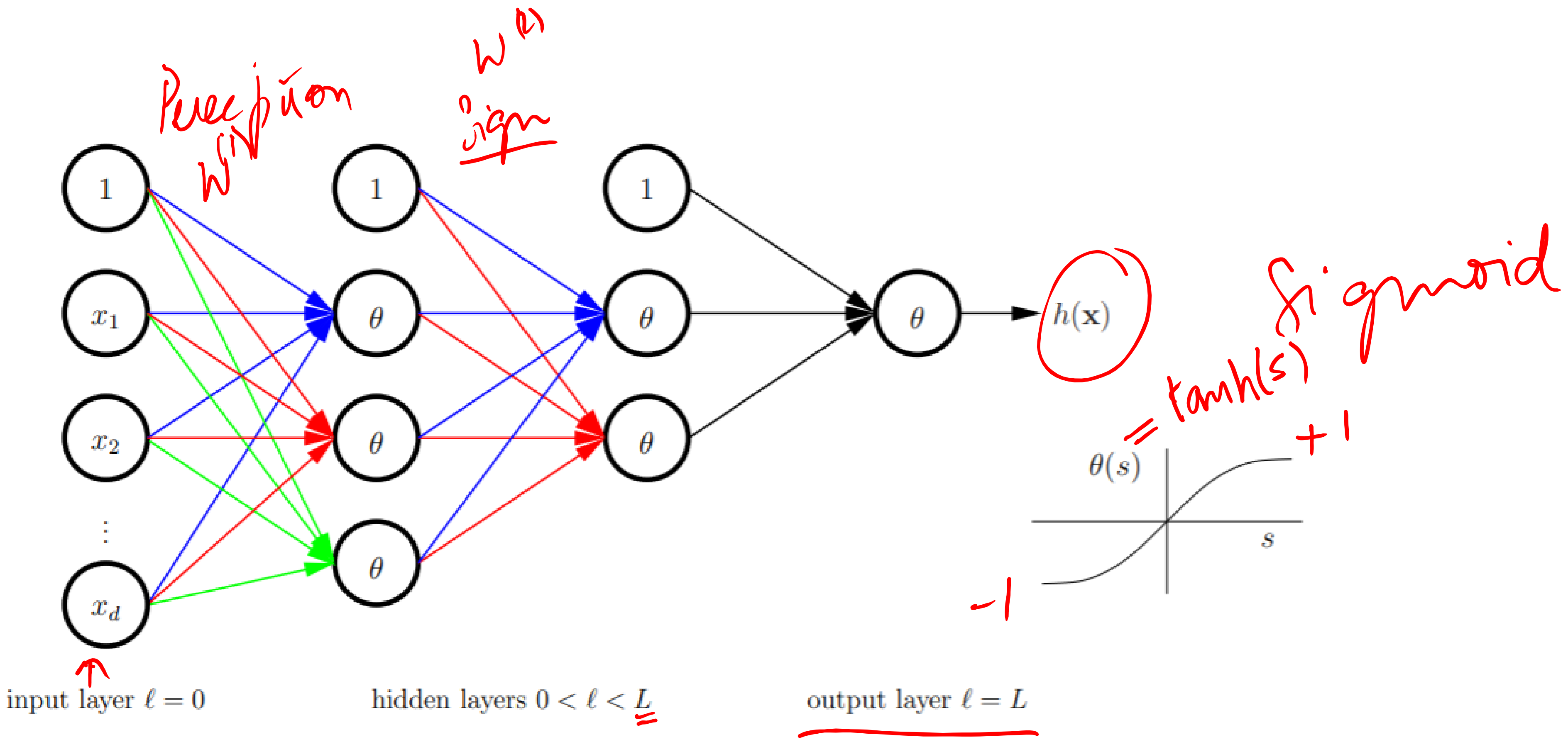
Sign function $\longrightarrow$ The Sigmoid (tanh)

MLP $\longrightarrow$ The Neural Network.

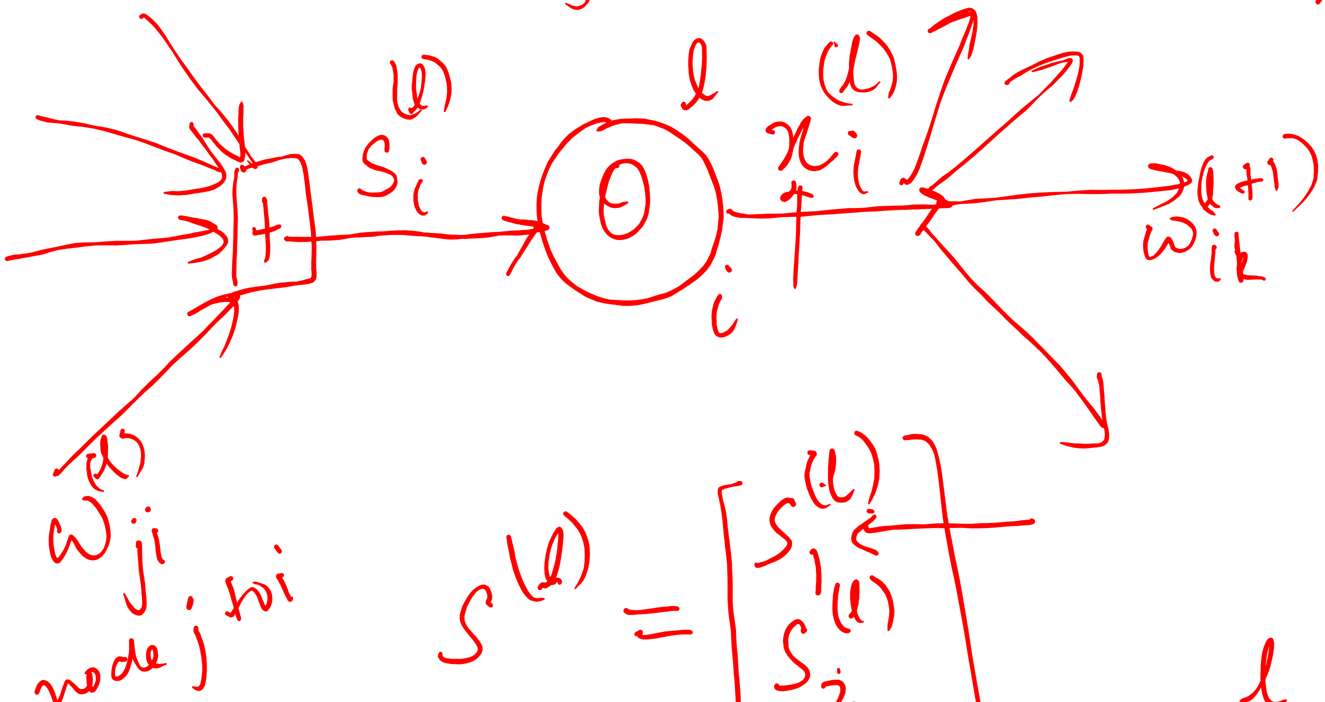
The (ML feed forward) Neural Network



The Neural Network



$s_3^{(2)}$ $\quad x_1^{(0)}$



$$x_i^{(l)} = \theta\left(s_i^{(l)}\right)$$

$$W^l = \begin{bmatrix} w_1^l & w_2^l & \cdots & w_{d^{(l)}}^l \\ | & | & & | \end{bmatrix}$$

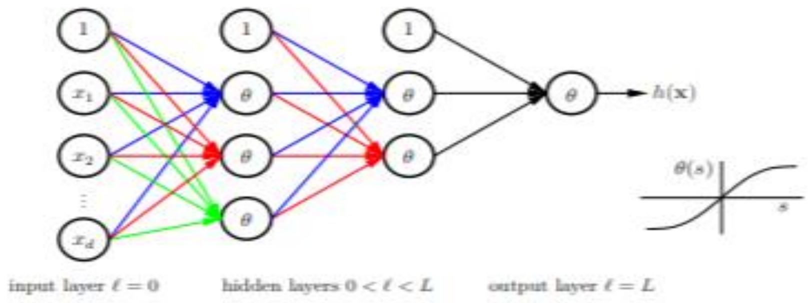
matrix

$$W^{(l+1)} = \begin{bmatrix} w_1^{l+1} & w_2^{l+1} & \cdots & w_{d^{(l+1)}}^{l} \\ | & | & & | \end{bmatrix}$$

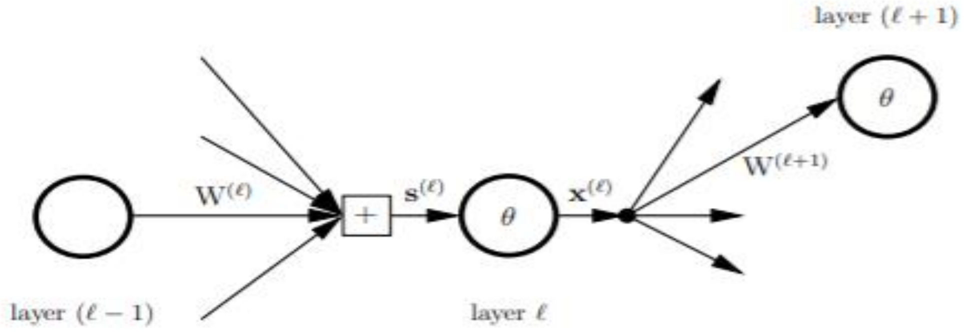
$$s^{(l)} = \begin{bmatrix} s_1^{(l)} \\ s_2^{(l)} \\ \vdots \\ s_d^{(l)} \end{bmatrix}$$

$$x^l = \begin{bmatrix} 1 \\ x_1^l \\ \vdots \\ x_d^l \end{bmatrix}$$

Zooming into a Hidden Node



MLP



layer ℓ parameters

signals in	$\mathbf{s}^{(\ell)}$	$d^{(\ell)}$ dimensional input vector
outputs	$\mathbf{x}^{(\ell)}$	$d^{(\ell)} + 1$ dimensional output vector
weights in	$\mathrm{W}^{(\ell)}$	$(d^{(\ell-1)} + 1) \times d^{(\ell)}$ dimensional matrix
weights out	$\mathrm{W}^{(\ell+1)}$	$(d^{(\ell)} + 1) \times d^{(\ell+1)}$ dimensional matrix

layers $\ell = 0, 1, 2, \ldots, L$
layer ℓ has "dimension" $d^{(\ell)} \implies d^{(\ell)} + 1$ nodes

$$\mathrm{W}^{(\ell)} = \begin{bmatrix} \mathbf{w}_1^{(\ell)} & \mathbf{w}_2^{(\ell)} & \cdots & \mathbf{w}_{d^{(\ell)}}^{(\ell)} \\ | & | & \vdots & | \end{bmatrix}$$

=

Thanks!