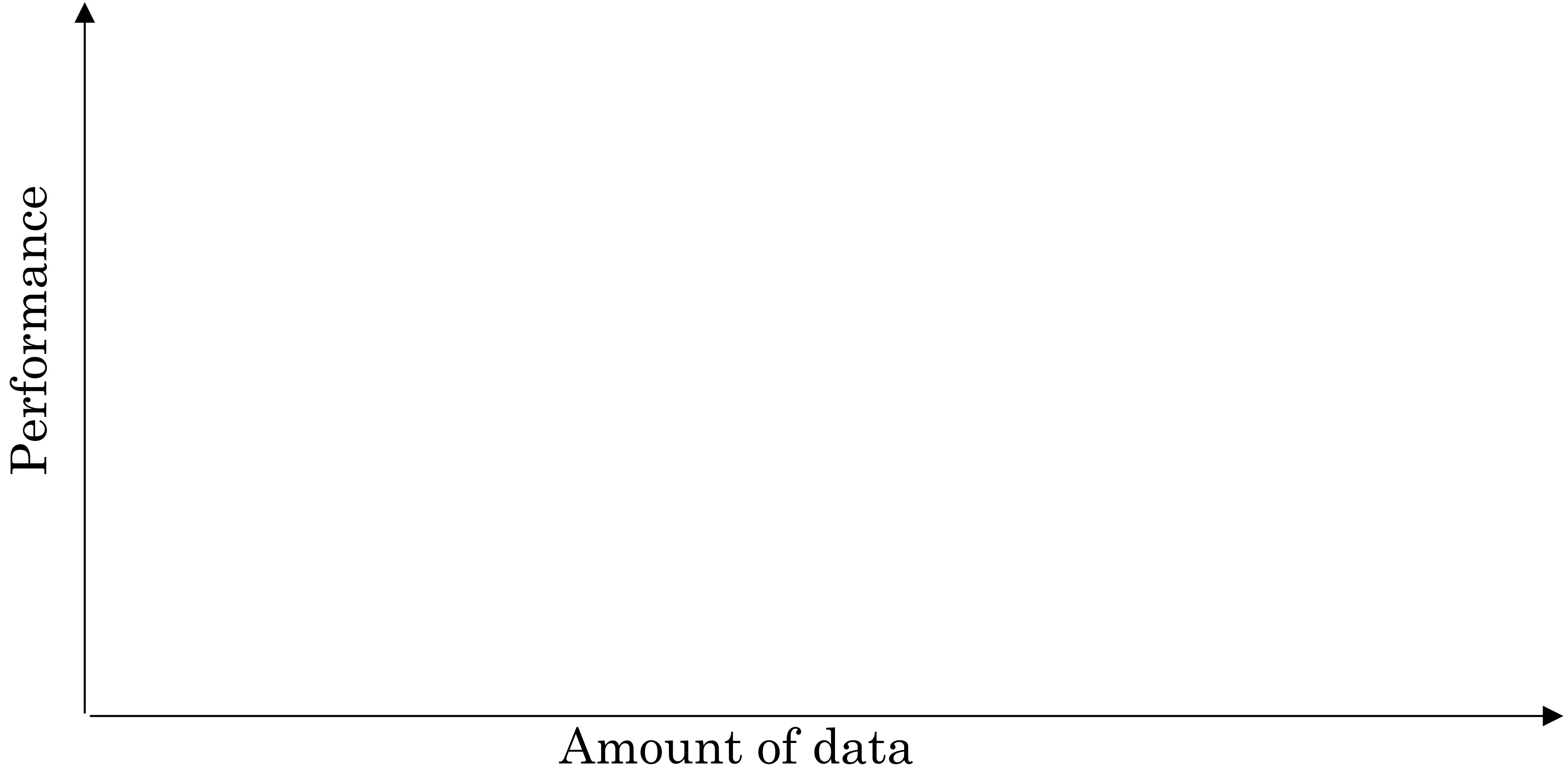
DATA ANALYSIS AND VISUALIZATION

INSTRUCTOR: UMME AMMARAH

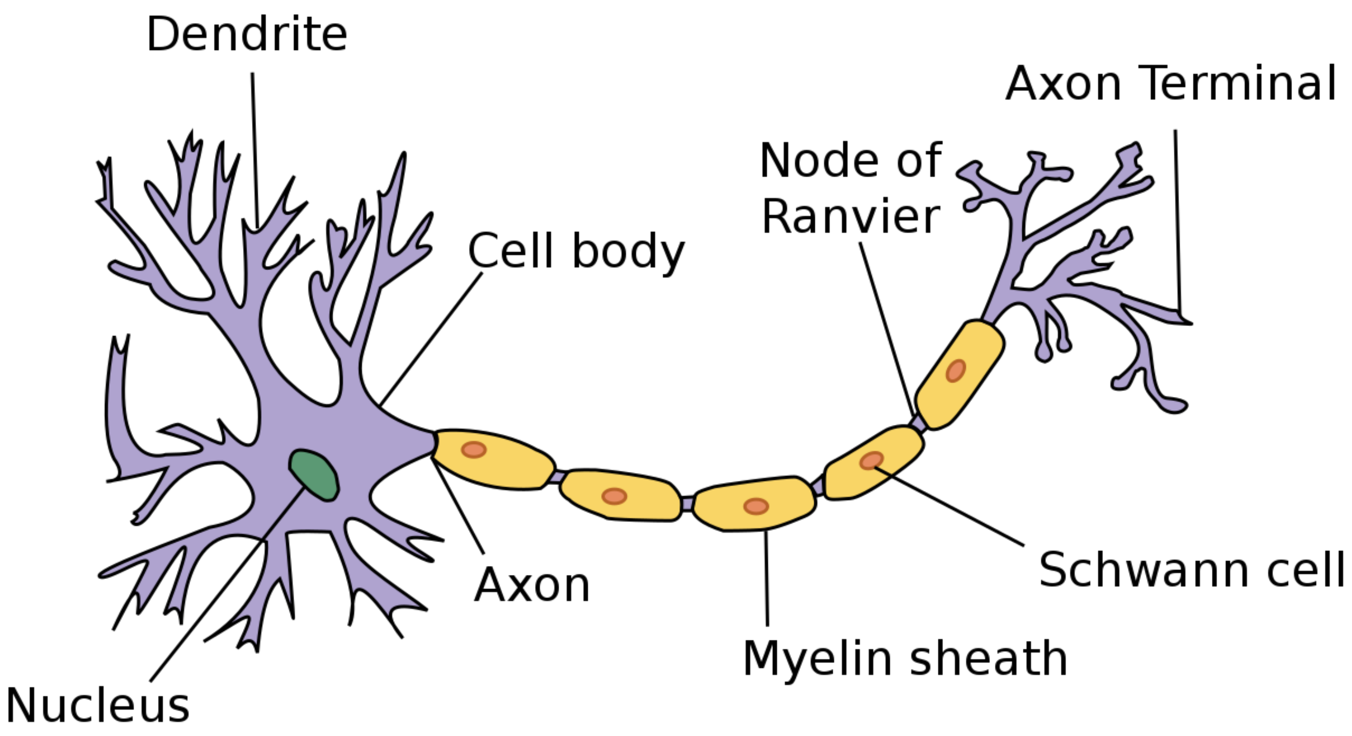
ARTIFICIAL NEURAL NETWORK

WHY DEEP LEARNING?

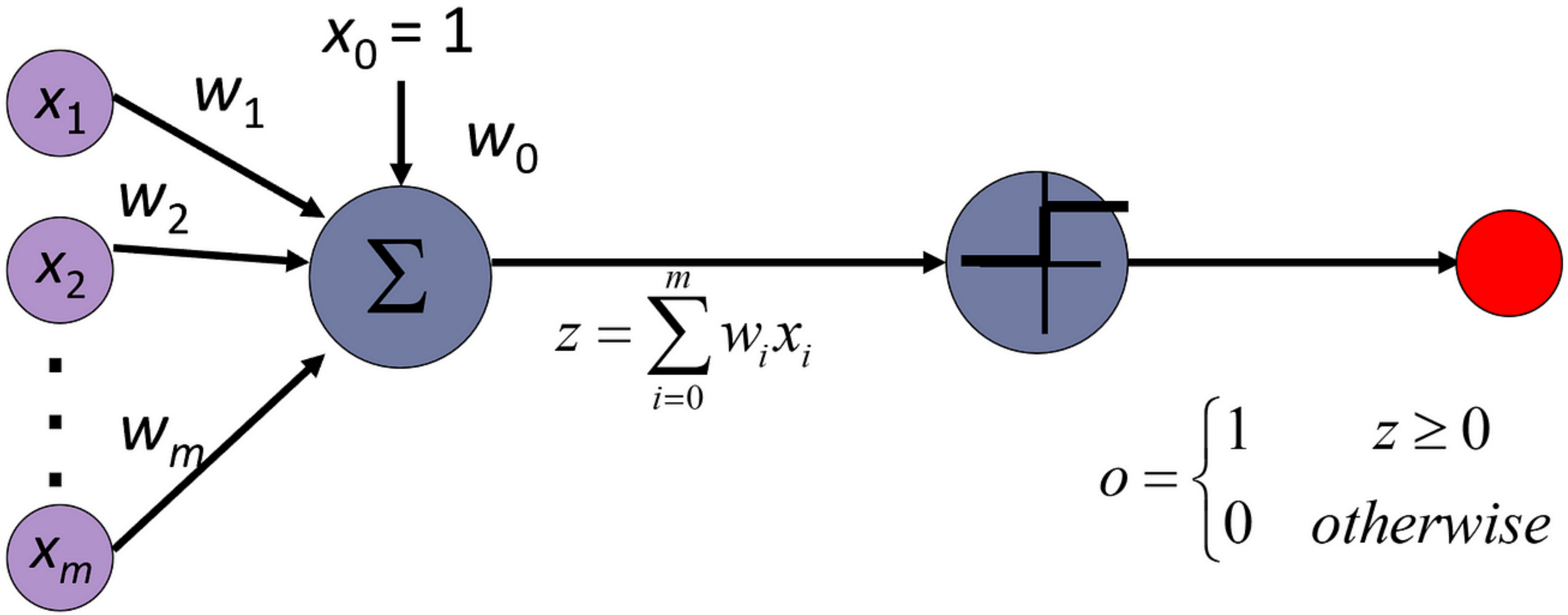


WHAT IS NEURAL NETWORK

- It is a powerful learning algorithm inspired by how the brain works.



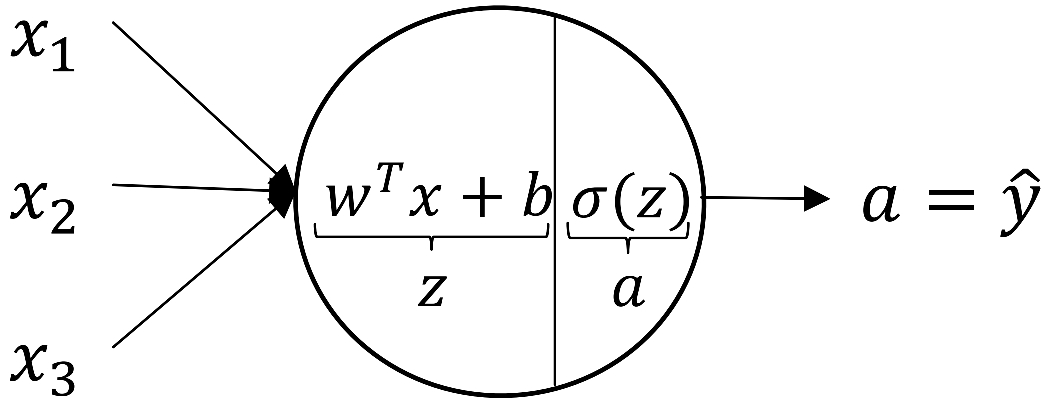
PERCEPTRON



$x_0 = 1$
w_1
w_0
w_2
x_1
x_2
Σ
w_m
x_m
$z = \sum_{i=0}^{m} w_i x_i$
$o = \begin{cases} 1 & z \geq 0 \\ 0 & otherwise \end{cases}$

ADDING SIGMOID UNIT (ACTIVATION FUNCTION)

Perceptron with a smooth activation function is called a neuron.



$$z = w^T x + b$$

$$a = \sigma(z)$$

EXAMPLE

w = [0.2, 0.3, 0.9]

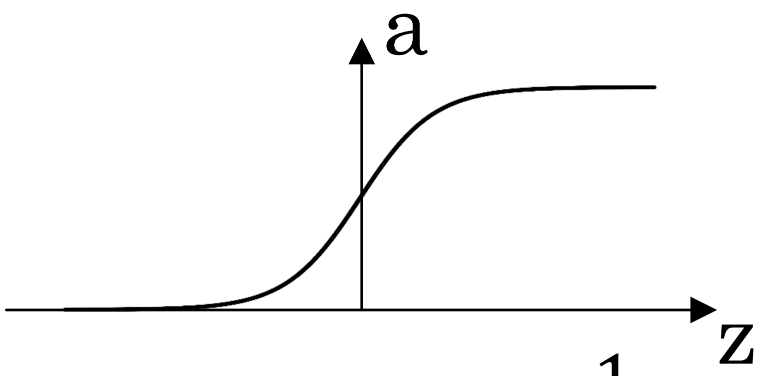
b = 0.5

x = [0.5, 0.6, 0.1]

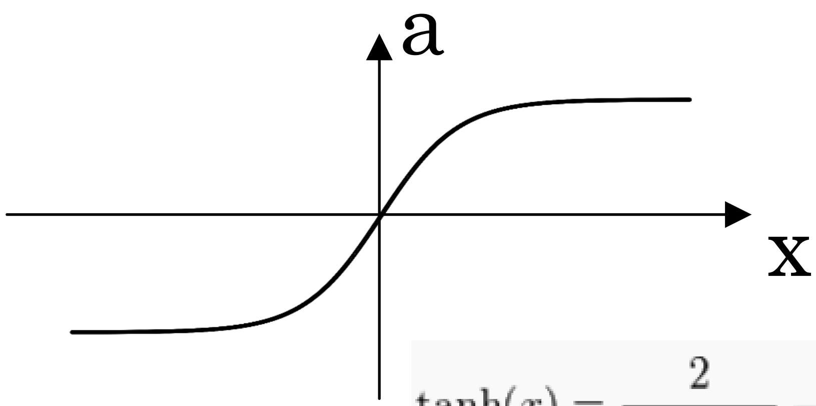
$$\text{sigmoid:}\quad a = \frac{1}{1+e^{-z}}$$

$$y = a = f(z)$$

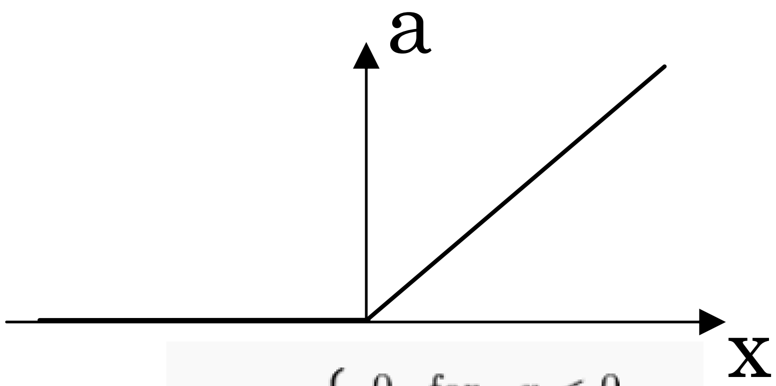
ACTIVATION FUNCTIONS



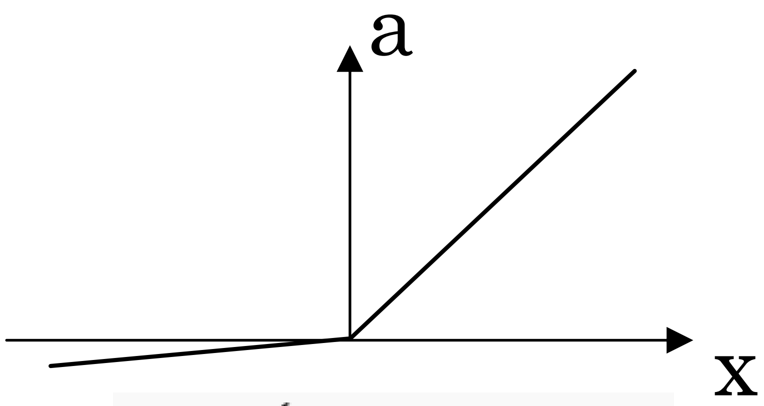
sigmoid: $a = \dfrac{1}{1 + e^{-z}}$



$$\tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$



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



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

BOOLEAN GATES

AND

x_1	x_2	t
0	0	0
0	1	0
1	0	0
1	1	1

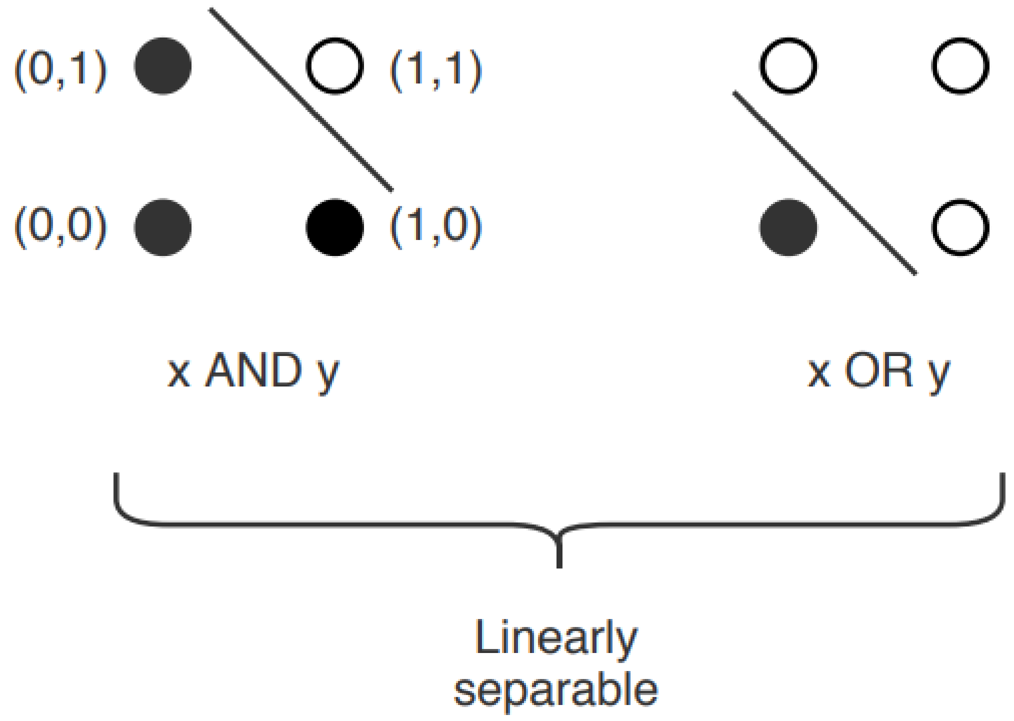
$$f(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & \sum w_i x_i + b \geq 0 \\ 0 & \sum w_i x_i + b < 0 \end{cases}$$

OR

x_1	x_2	t
0	0	0
0	1	1
1	0	1
1	1	1

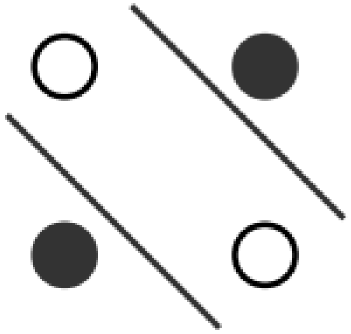
PERCEPTRON

- A perceptron is actually a linear classifier.
- It's weights wi and b represent a line that divides input space into 2 regions.



XOR PROBLEM

- A perceptron cannot model the XOR problem because XOR is not a linear classification problem. No single line can separate the 0s (black) from the 1s (white).
- But combination of two lines can.

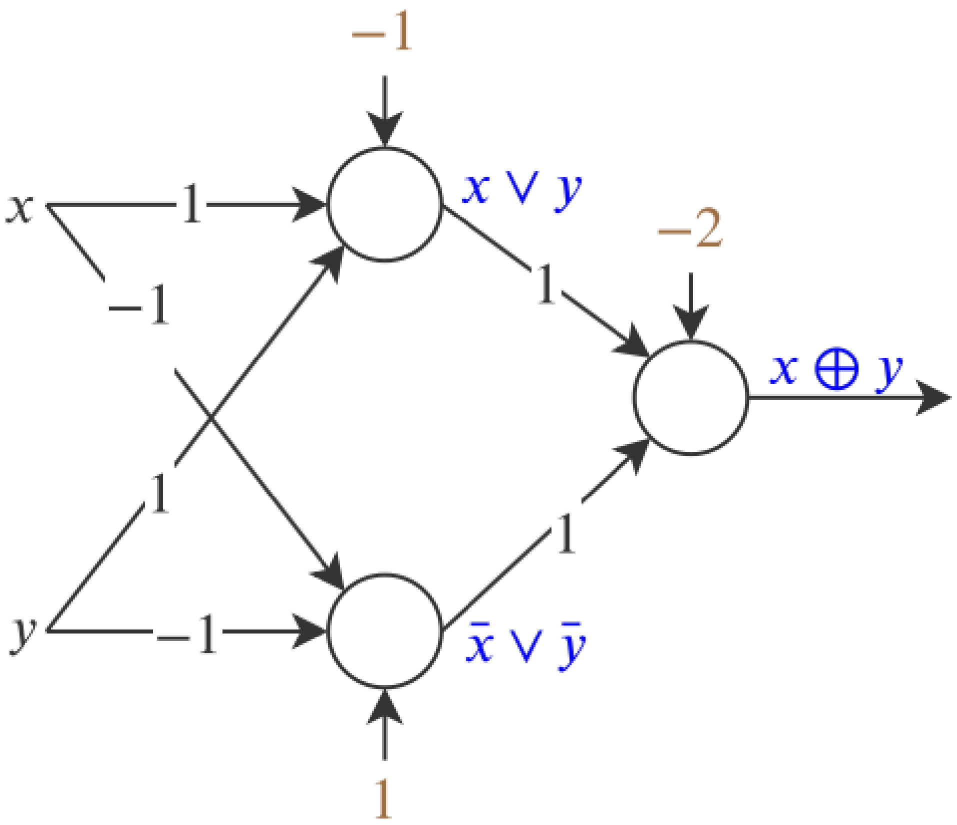


x XOR y

Not linearly separable

MULTILAYER PERCEPTRONS

- Combination of perceptrons can solve the XOR problem.
- Two lines can divide the XOR space into 0-region and 1-region
- 3 perceptrons can model XOR. Two perceptrons for the two lines and a final perceptron to combine them into a final decision.



A *network* of 3 neurons is more powerful than 1 neuron.
Just like the brain!

PERCEPTRON

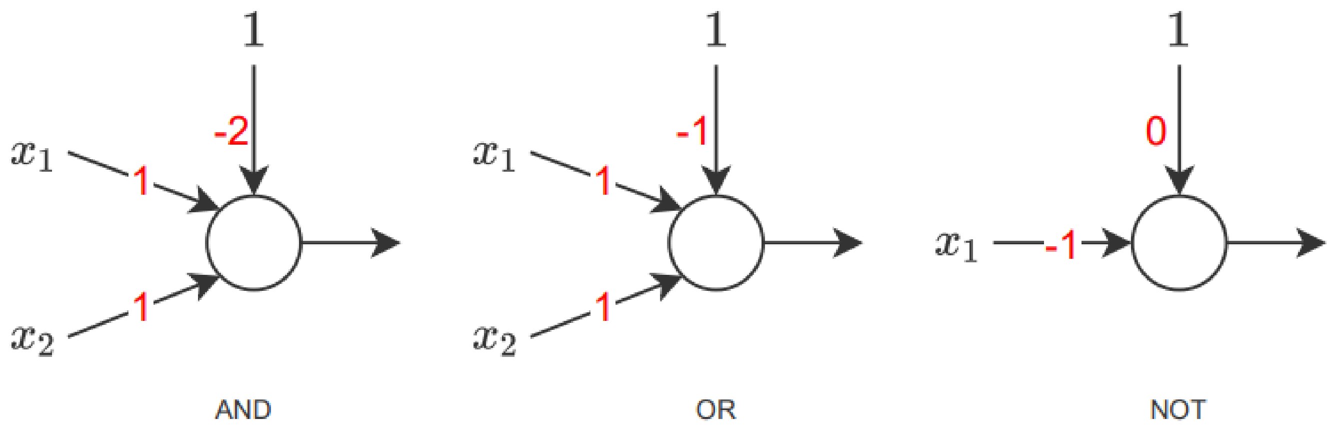
- *Importantly*, the weights of a perceptron can be learned.
- Perceptron learning rule:

$$w_i \leftarrow w_i + \eta(y - t)x_i$$

if output y and desired target t are different.

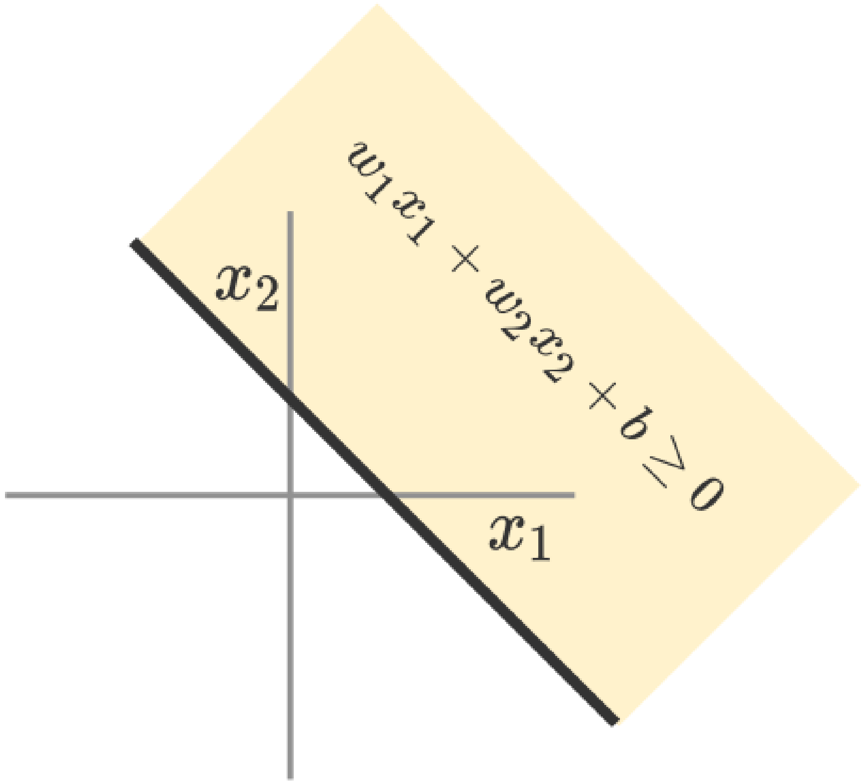
MULTI-LAYER PERCEPTRON

- A single perceptron can model the basis set {AND, OR, NOT} of logic gates.



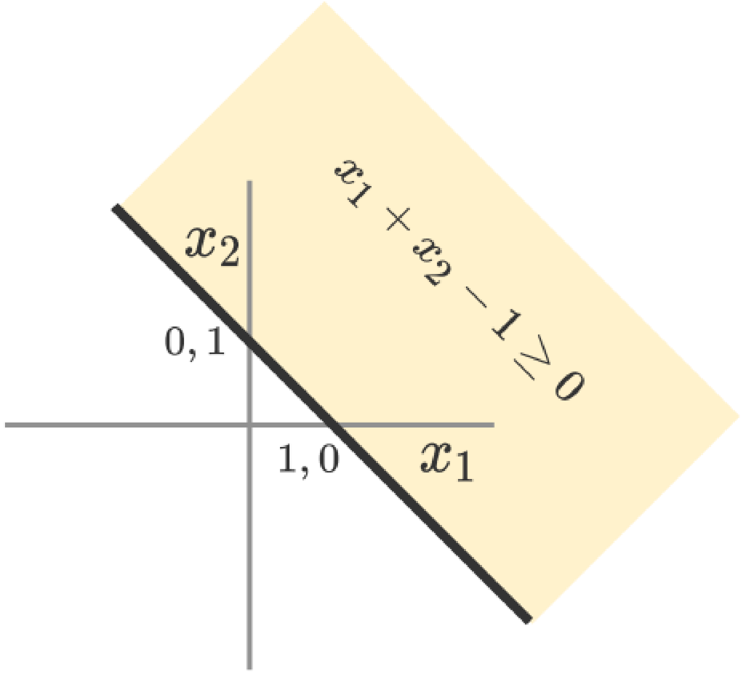
- All Boolean functions can be written using combinations of these basic gates.
- Therefore, combinations of perceptrons (MLPs) can model all Boolean functions.

MLPS AND CLASSIFICATION BOUNDARIES

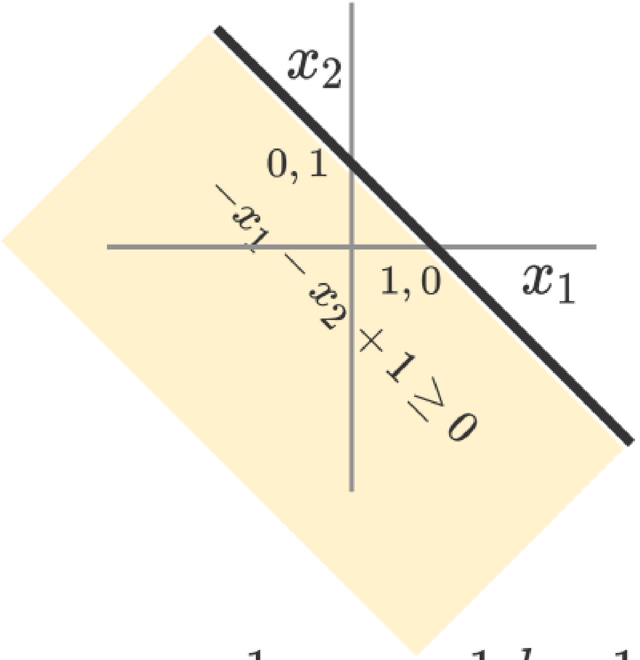


A perceptron divides input space into 2 regions. Dividing boundary is a line.

MLPS AND CLASSIFICATION BOUNDARIES

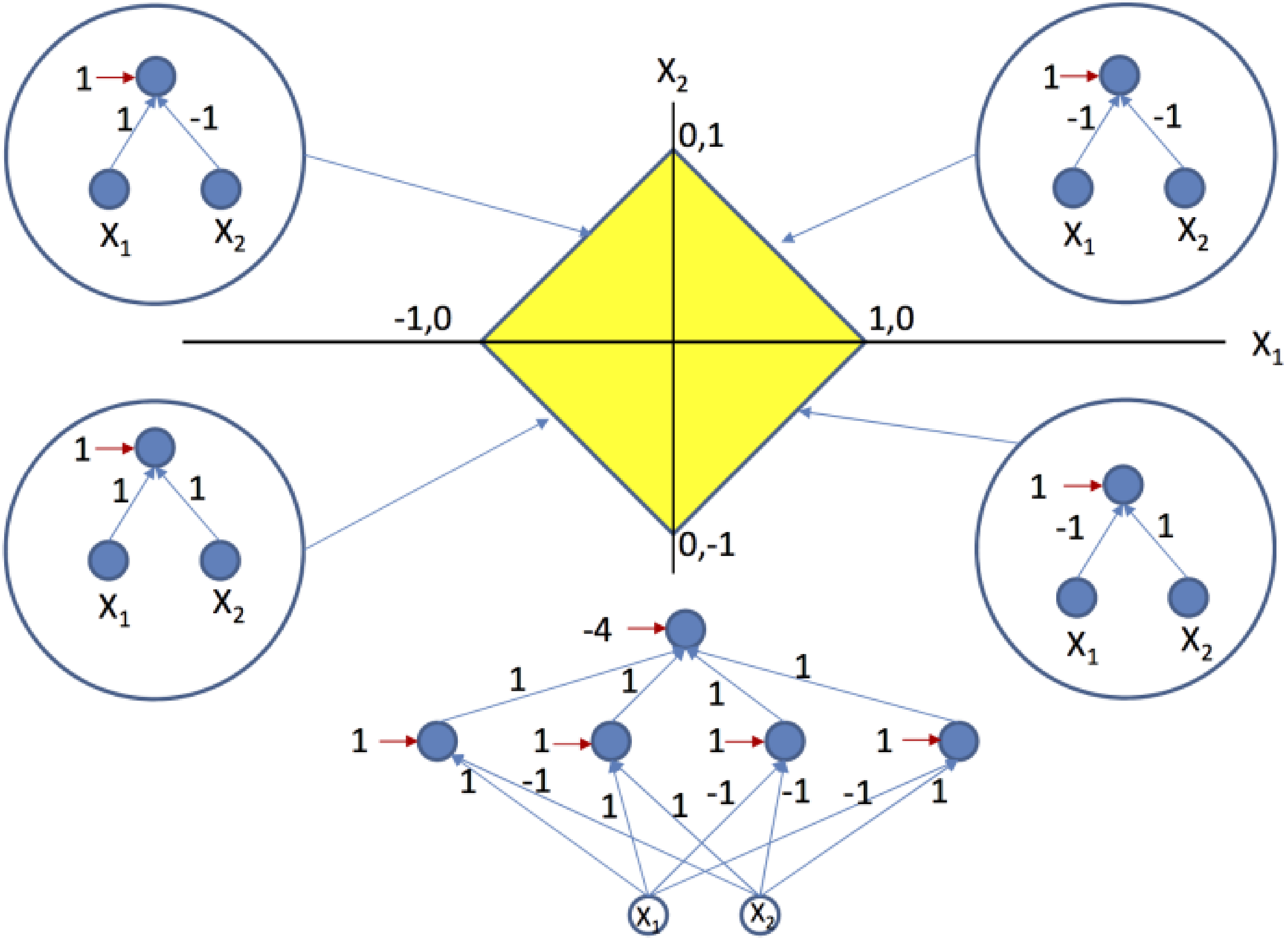


$w_1 = 1, w_2 = 1, b = -1$



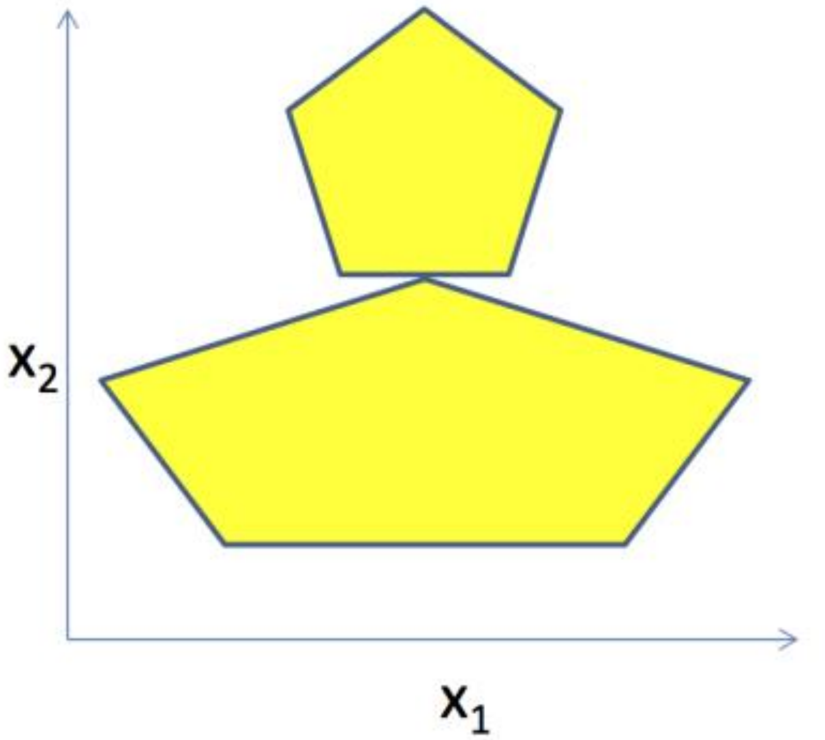
$w_1 = -1, w_2 = -1, b = 1$

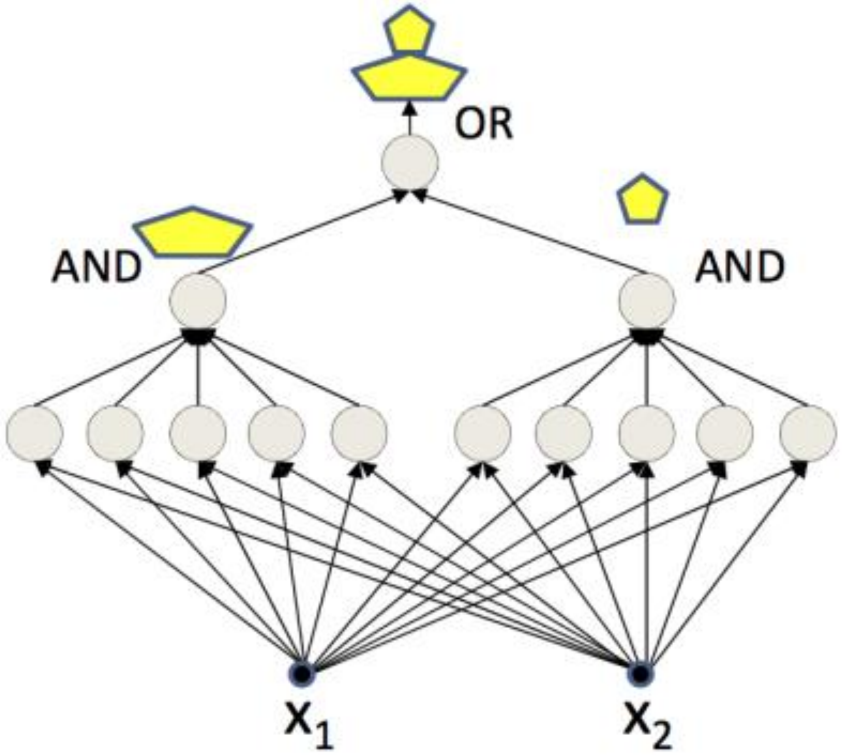
Weights determine the linear boundary and classification into region 1 and region 2.



Yellow region modelled by ANDing 4 linear classifiers (perceptrons). First layer contains 4 perceptrons for modelling 4 lines and second layer contains a perceptron for modelling an AND gate

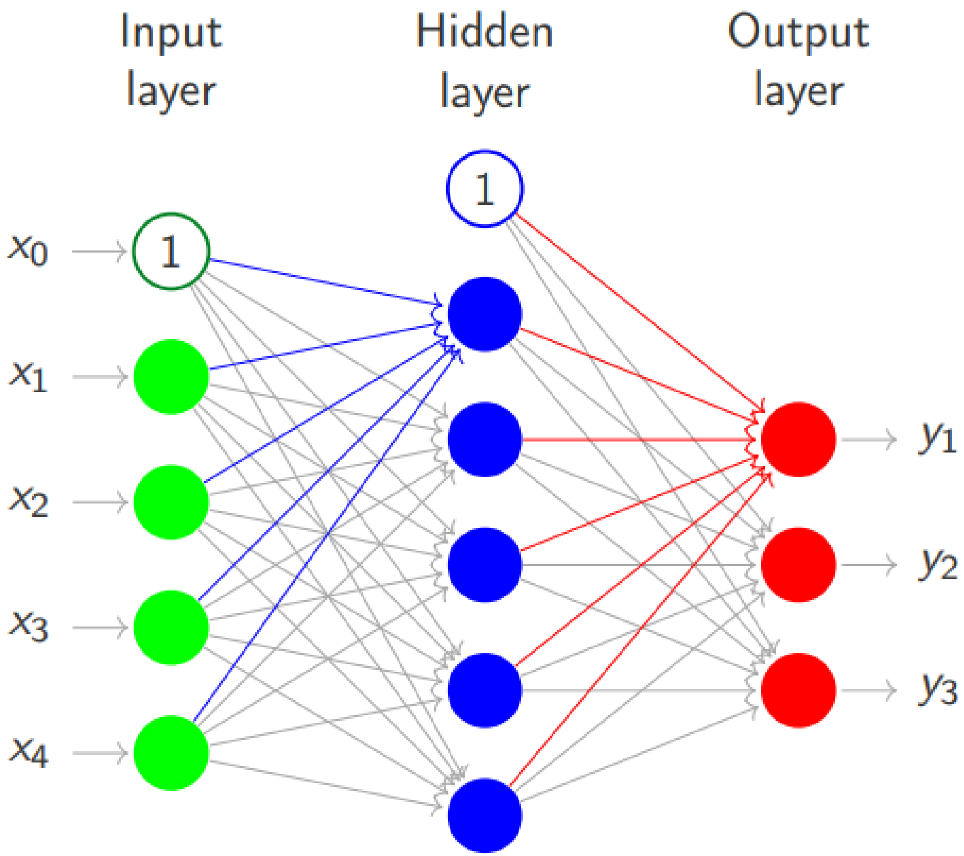
NON-CONTIGUOUS BOUNDARIES





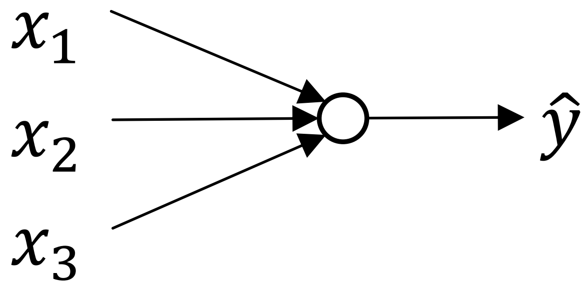
- Yellow region equals OR(polygon 1, polygon 2). Each polygon equals AND of some lines. Each line equals 1 perceptron.
- Since inputs and outputs are visible, all layers in-between are known as **hidden layers**

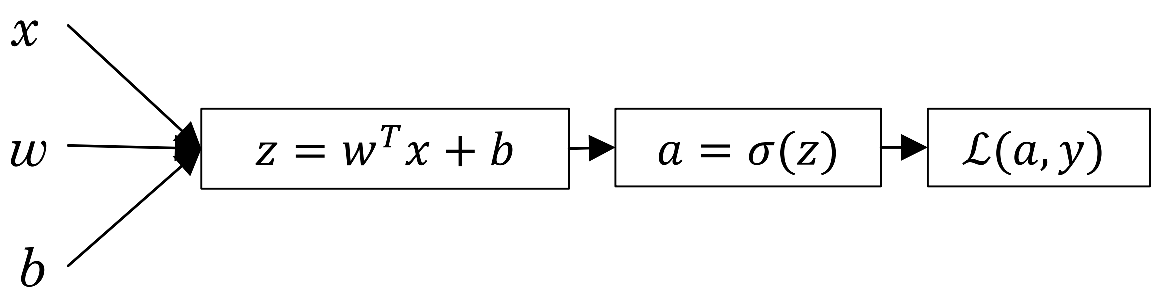
NEURAL NETWORK

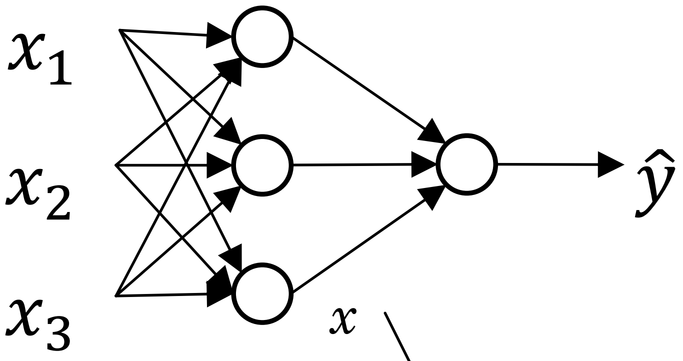


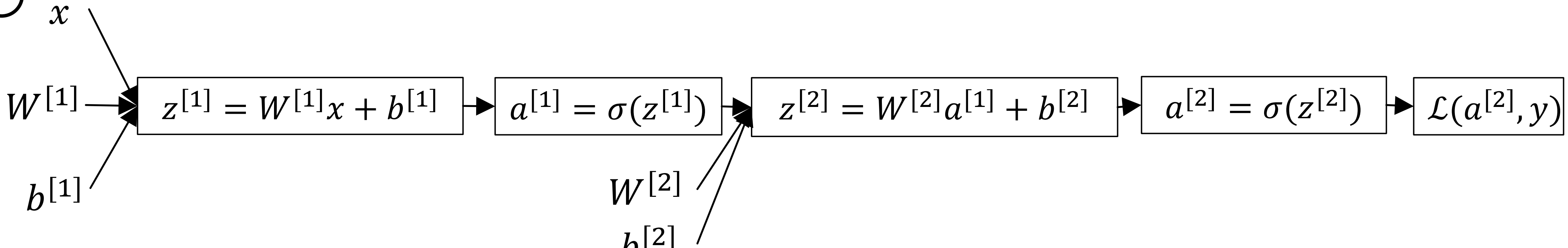
Output of a neural network can be visualised graphically as *forward propagation of information*.

MATHEMATICAL REPRESENTATION

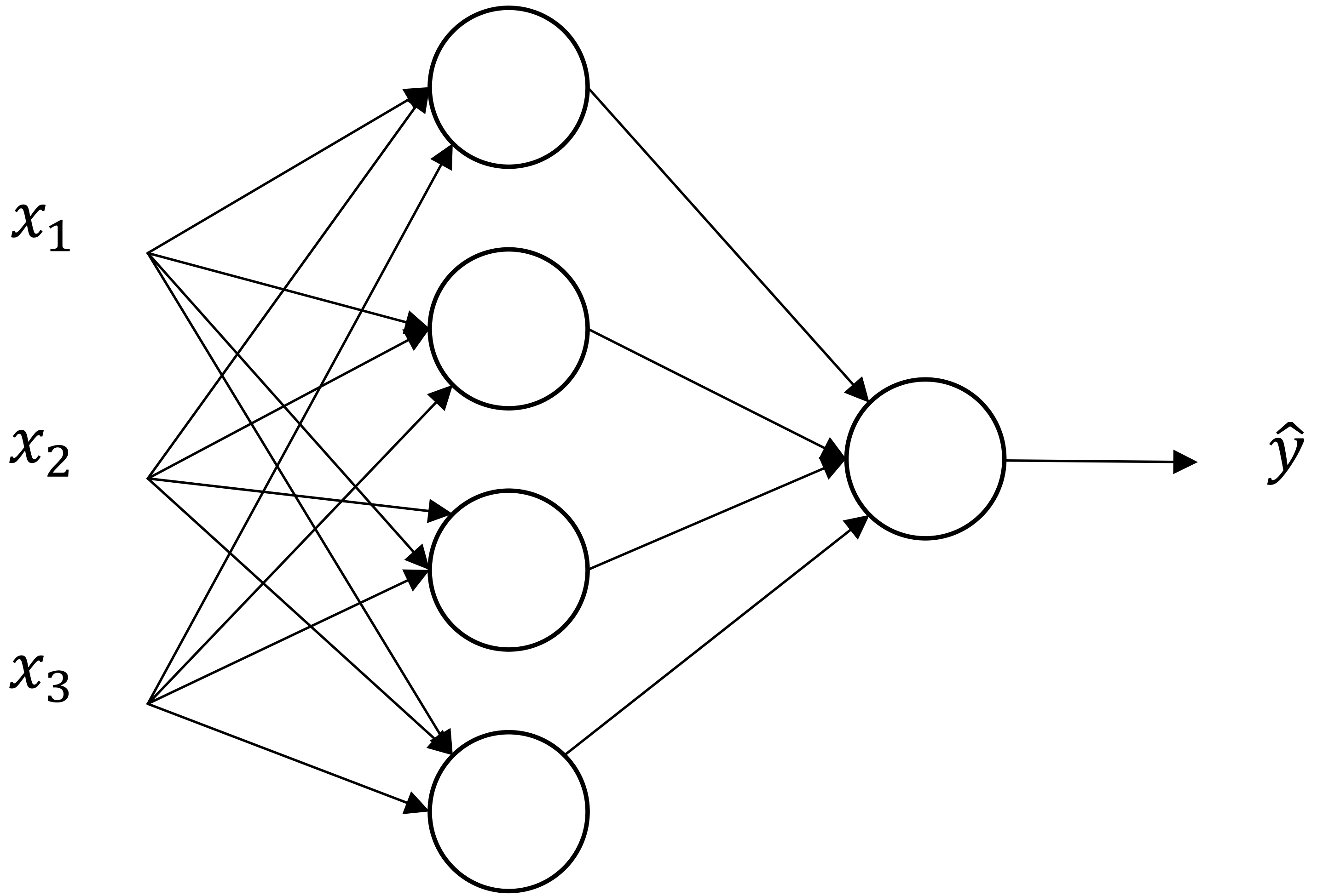




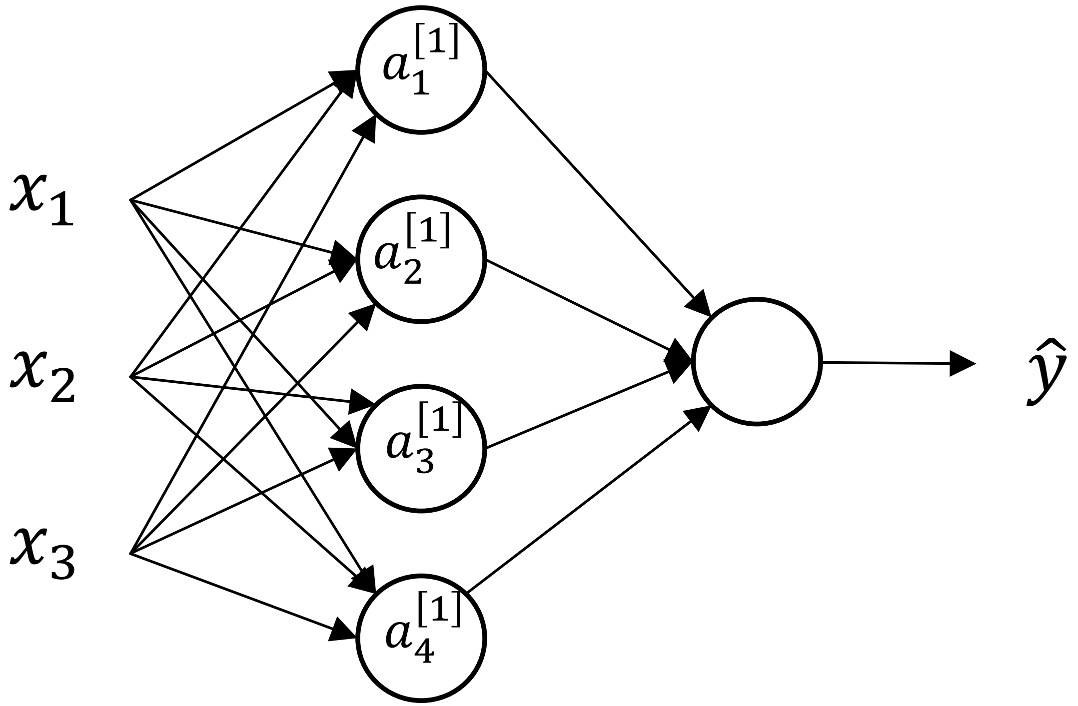




Neural Network Representation



Neural Network Representation



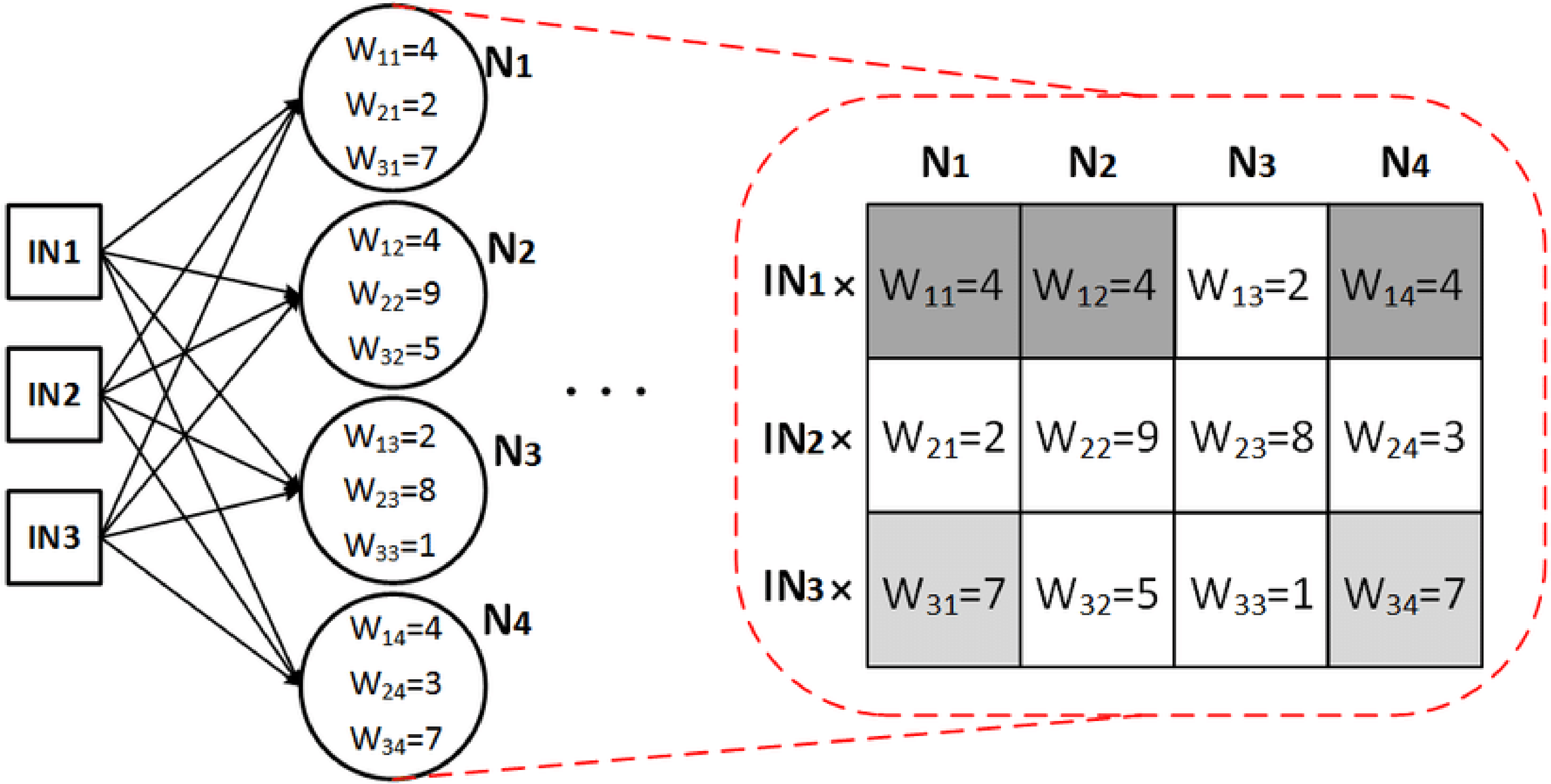
$$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]}, \ a_1^{[1]} = \sigma(z_1^{[1]})$$

$$z_2^{[1]} = w_2^{[1]T} x + b_2^{[1]}, \ a_2^{[1]} = \sigma(z_2^{[1]})$$

$$z_3^{[1]} = w_3^{[1]T} x + b_3^{[1]}, \ a_3^{[1]} = \sigma(z_3^{[1]})$$

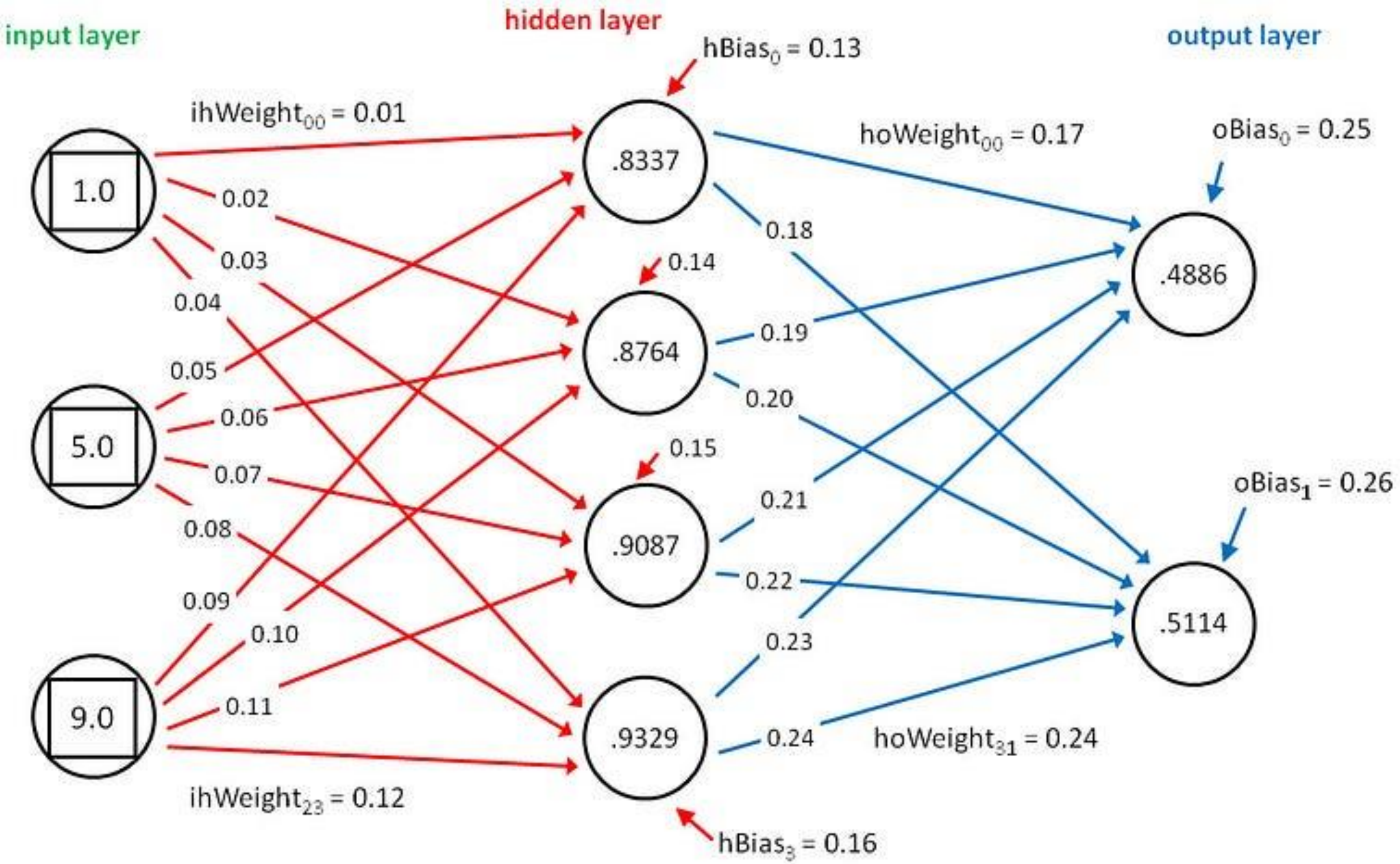
$$z_4^{[1]} = w_4^{[1]T} x + b_4^{[1]}, \ a_4^{[1]} = \sigma(z_4^{[1]})$$

WEIGHTS



IN1
IN2
IN3
W_{11}=4
W_{21}=2
W_{31}=7
N1
W_{12}=4
W_{22}=9
W_{32}=5
N2
W_{13}=2
W_{23}=8
W_{33}=1
N3
W_{14}=4
W_{24}=3
W_{34}=7
N4
. . .
N1 N2 N3 N4
IN1 × W_{11}=4 W_{12}=4 W_{13}=2 W_{14}=4
IN2 × W_{21}=2 W_{22}=9 W_{23}=8 W_{24}=3
IN3 × W_{31}=7 W_{32}=5 W_{33}=1 W_{34}=7

BIAS



input layer
hidden layer
output layer
hBias0 = 0.13
ihWeight00 = 0.01
hoWeight00 = 0.17
oBias0 = 0.25
1.0
5.0
9.0
.8337
.8764
.9087
.9329
.4886
.5114
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.10
0.11
0.14
0.15
0.18
0.19
0.20
0.21
0.22
0.23
0.24
oBias1 = 0.26
hoWeight31 = 0.24
ihWeight23 = 0.12
hBias3 = 0.16

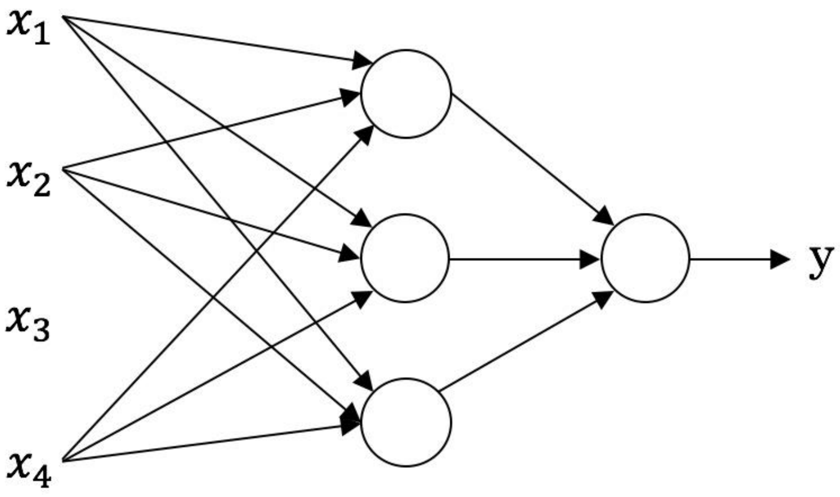
FORWARD PROPAGATION

Given x:

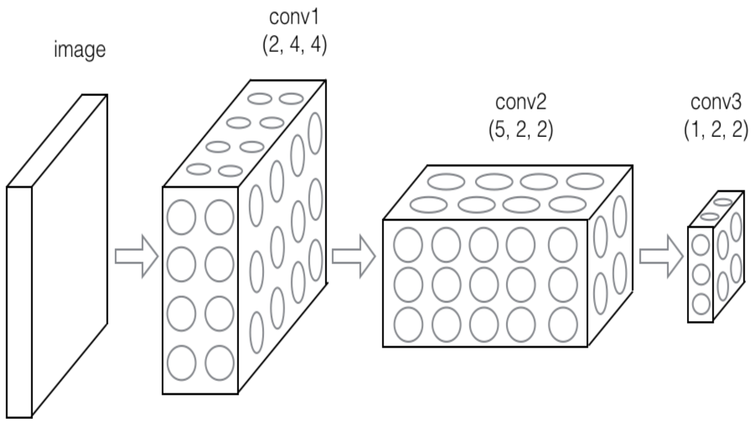
$$z^{[1]} = W^{[1]}x + b^{[1]}$$
$$a^{[1]} = \sigma(z^{[1]})$$
$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$
$$a^{[2]} = \sigma(z^{[2]})$$

$$\boxed{\begin{aligned} Z^{[1]} &= W^{[1]}X + b^{[1]} \\ A^{[1]} &= g^{[1]}(Z^{[1]}) \\ Z^{[2]} &= W^{[2]}A^{[1]} + b^{[2]} \\ A^{[2]} &= g^{[2]}(Z^{[2]}) \\ &\vdots \\ A^{[L]} &= g^{[L]}\left(Z^{[L]}\right) = \hat{Y} \end{aligned}}$$

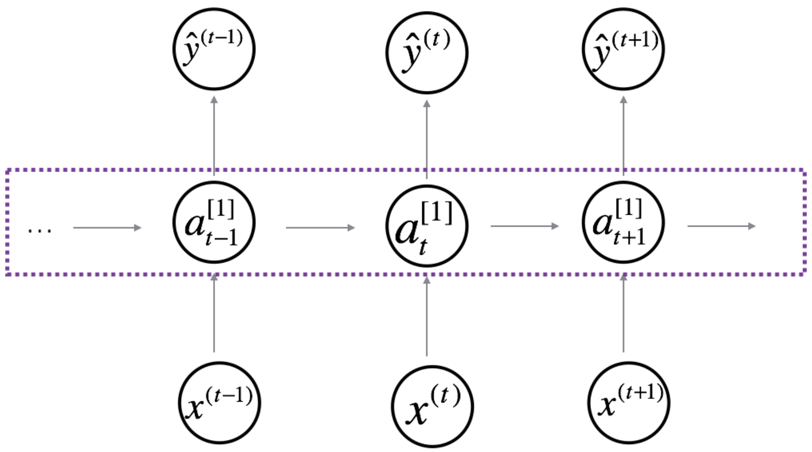
NEURAL NETWORK TYPES



Standard NN



Convolutional NN



Recurrent NN