Neural Networks: Lecture 1

Dr. Marwan Torki

2022

جامعة الإسكندرية
ALEXANDRIA
UNIVERSITY

Instructor

- Dr. Marwan Torki

  Associate Professor

  Computer and Systems Engineering Department

  Faculty of Engineering

  Alexandria University

  Contact : mtorki@alexu.edu.eg

Grades

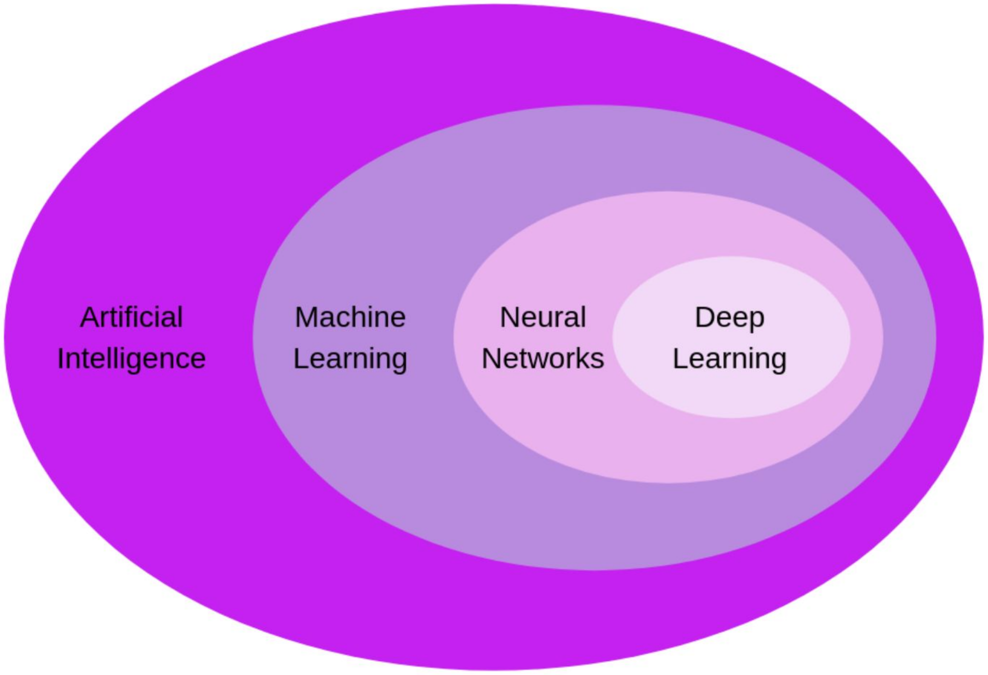
- Coursework %
  - Midterm %
  - Projects %
- Final %

Course Content

- The course covers the most common models in artificial neural networks with a focus on the multi-layer perceptron.

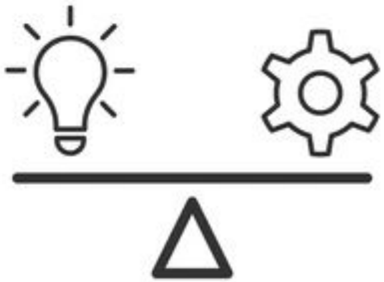
- The course also provides an introduction to deep learning.

Courses Overview



Lectures format

- The lectures will balance between theory and real life practices.



Tools

- Python
- Jupyter Notebooks
- Google Colab
- Tensorflow

Brief History of Neural Nets

- What was an active topic in 1987?

    back propagation, neural networks

- What happened the past 40 years?
    - Computers got faster
    - Data sets got larger
    - Software tools improved (TensorFlow, Theano, Keras)

Data

Data can be

Structured Data

X coordinate	Y coordinate	Class Name
10	5	A
15	10	B

Unstructured Data

- Images
- Videos
- Text
- Audio

ALEXANDRIA UNIVERSITY

Datasets

MNIST

ImageNet

MS COCO

CIFAR10

IMDB Reviews

Free Music Archive

kaggle

ALEXANDRIA UNIVERSITY
جامعة الإسكندرية

Notations

- vector: $\boldsymbol{x}$ $\boldsymbol{y}$
- vector element: x_i y_j
- matrix: $\boldsymbol{W}$
- matrix row: $\boldsymbol{W}_i$
- matrix element: w_{ij}

Supervised Learning

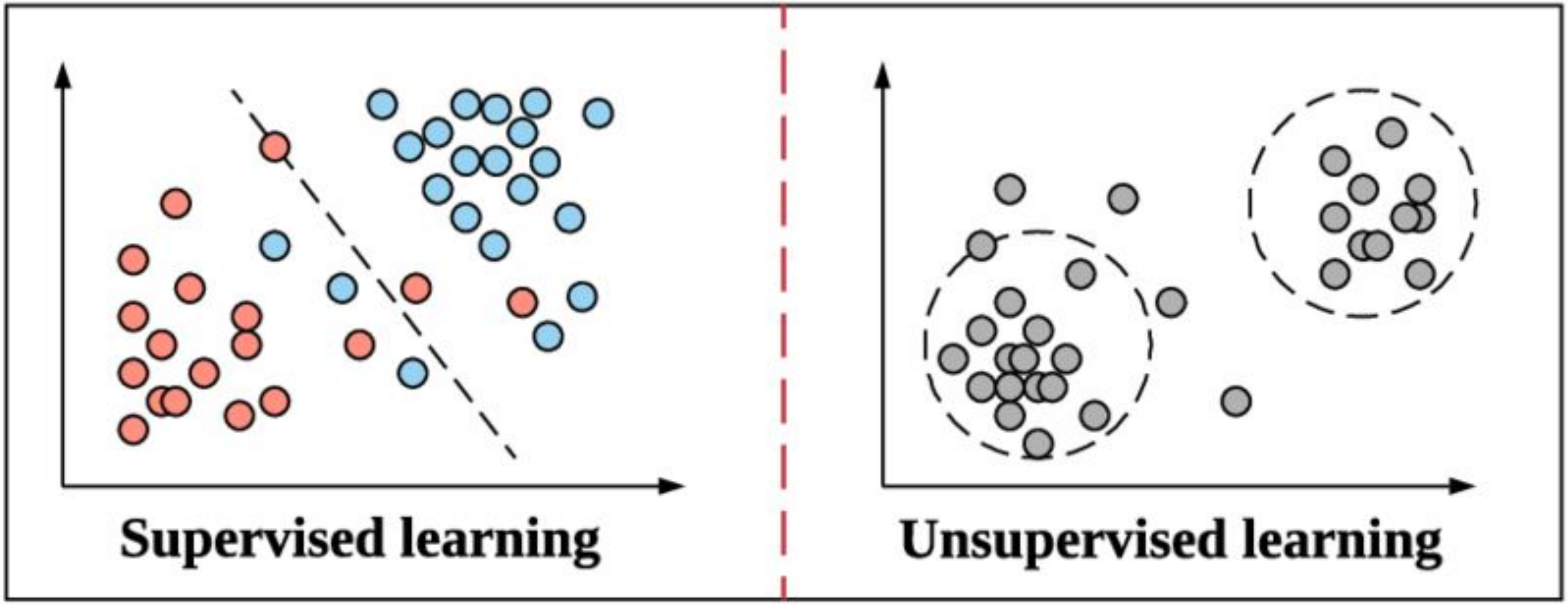
It is based on labeled training data.

$$X \in \mathbb{R}^n \qquad Y \in \{0, 1\}$$

input

Unsupervised Learning

It is based on unlabeled training data.

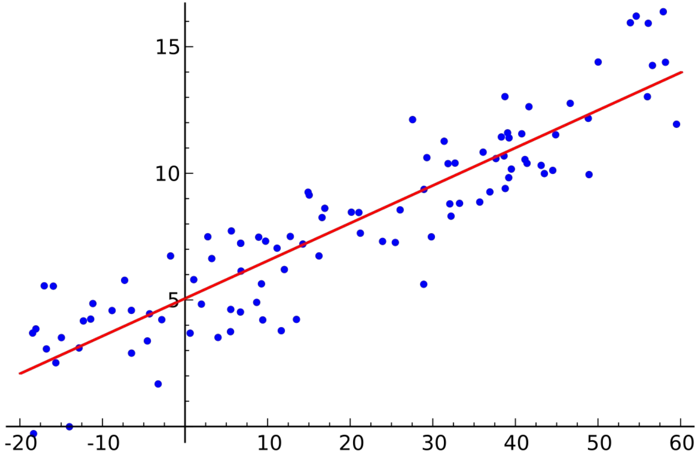

Supervised learning
Unsupervised learning

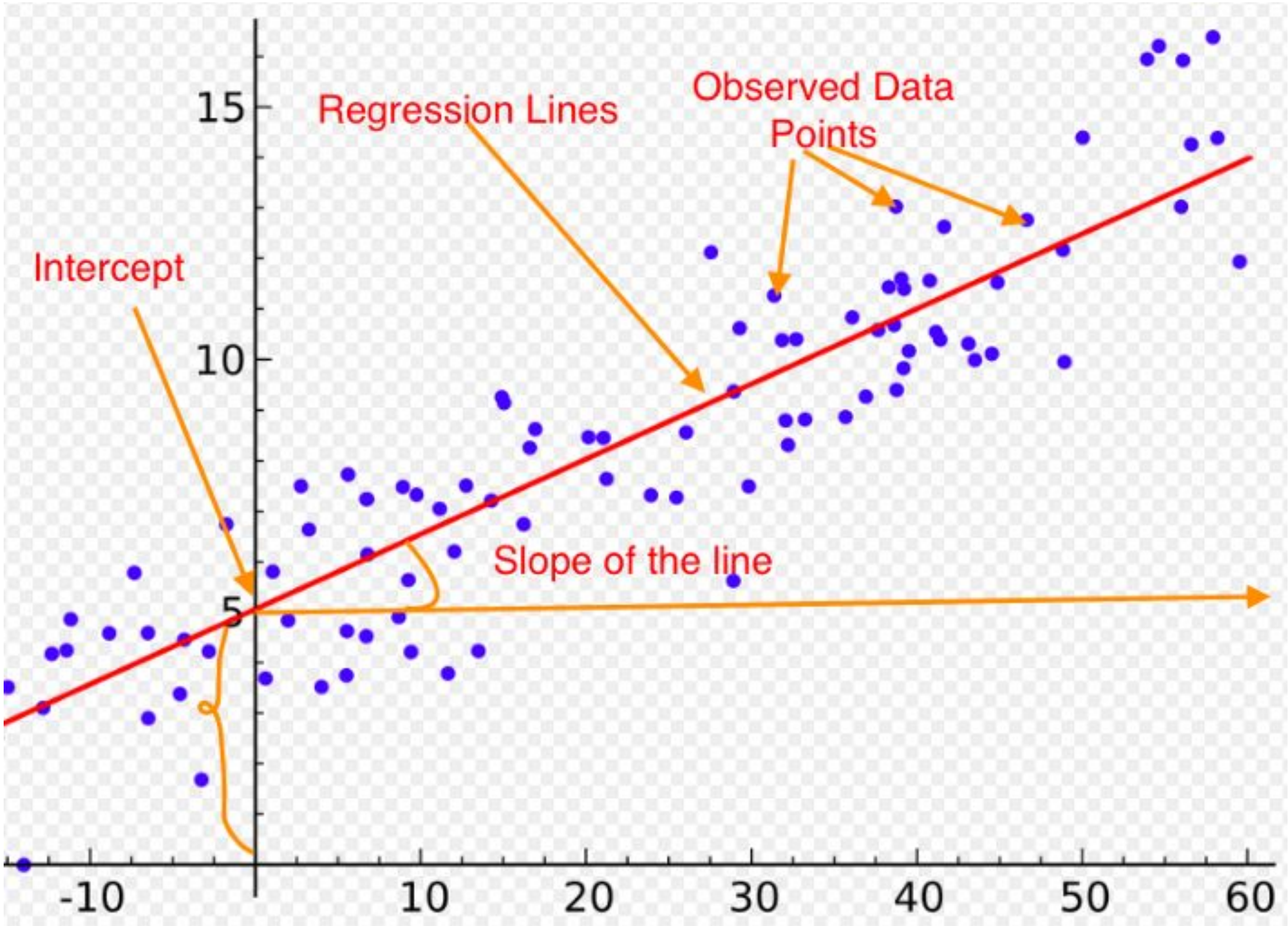
Linear Classifiers

Linear Regression

- Best fit line for a set of points

$$\hat{y} = \boldsymbol{w}^\top \boldsymbol{x} + b,$$





Regression Lines
Observed Data Points
Intercept
Slope of the line
15
10
5
-10
10
20
30
40
50
60

Linear Regression

- Error
  - Linear regression most often use

    Mean Square Error (MSE)

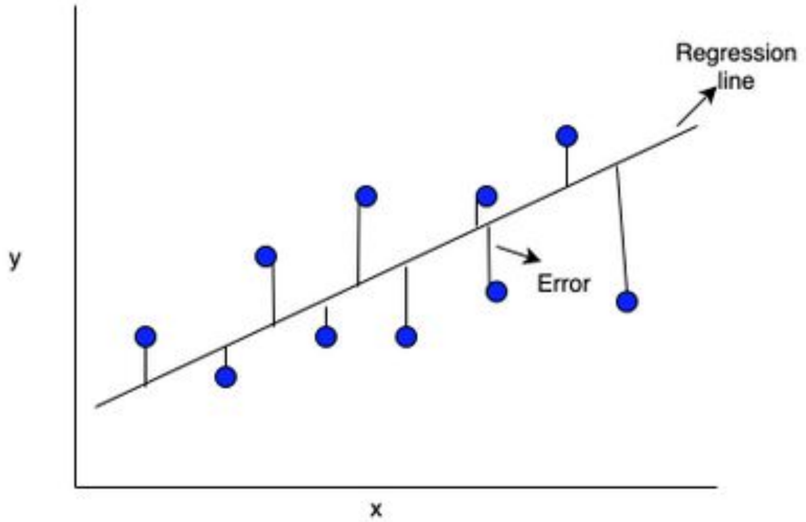
$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

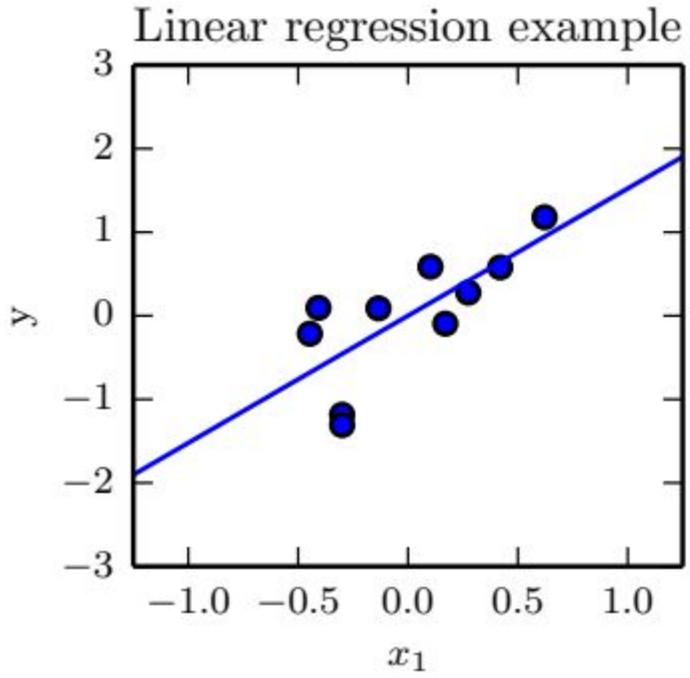
MSE = mean squared error

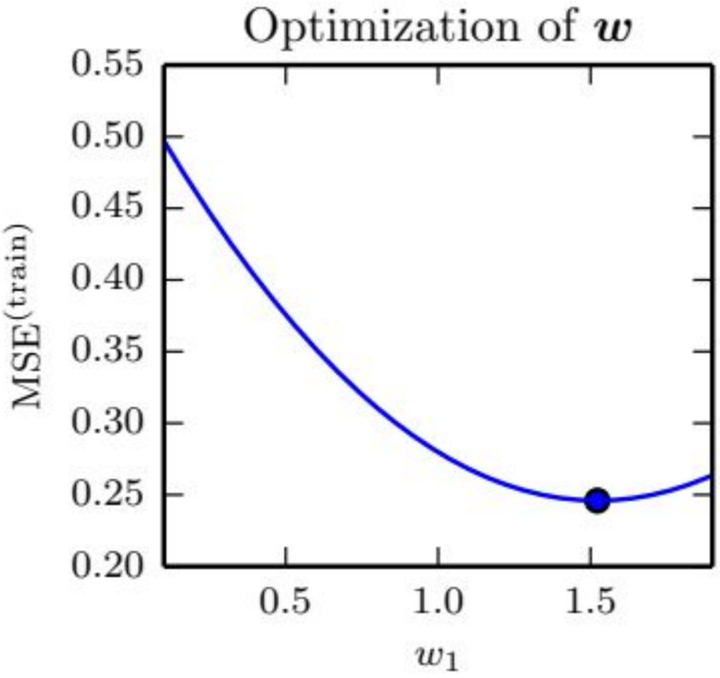
n = number of data points

Y_i = observed values

$\hat{Y}_i$ = predicted values

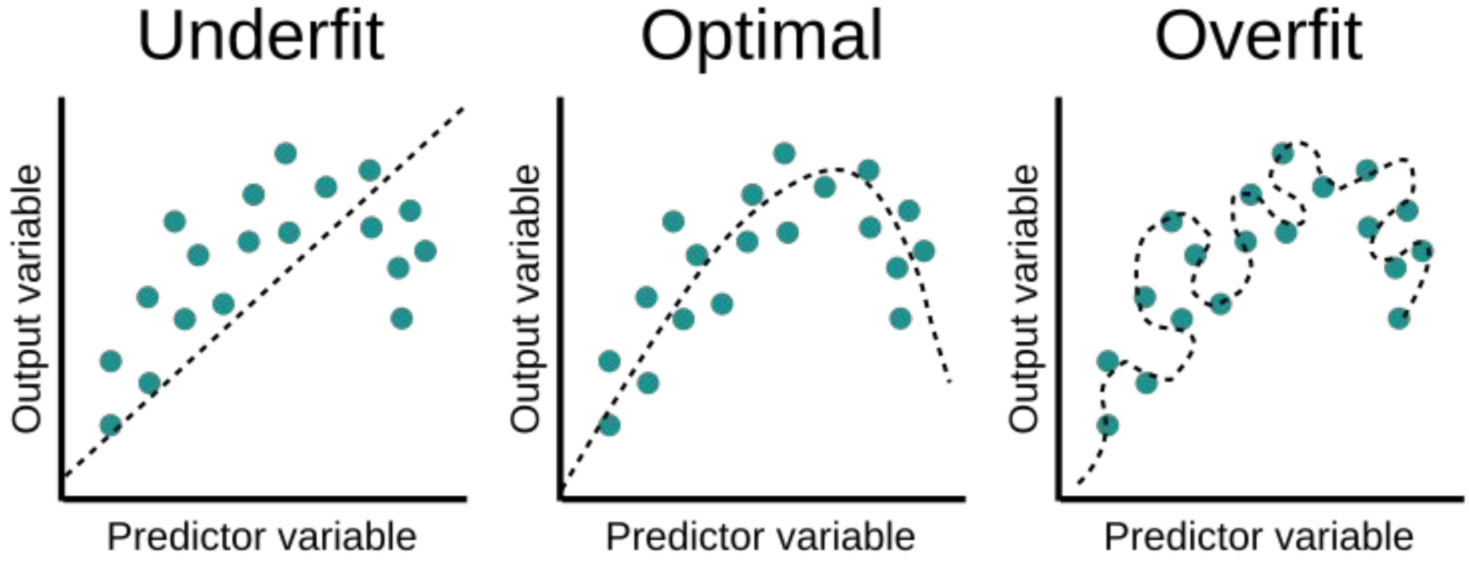






ALEXANDRIA UNIVERSITY

Overfitting and Underfitting



Neural Networks: Lecture 2

Dr. Marwan Torki

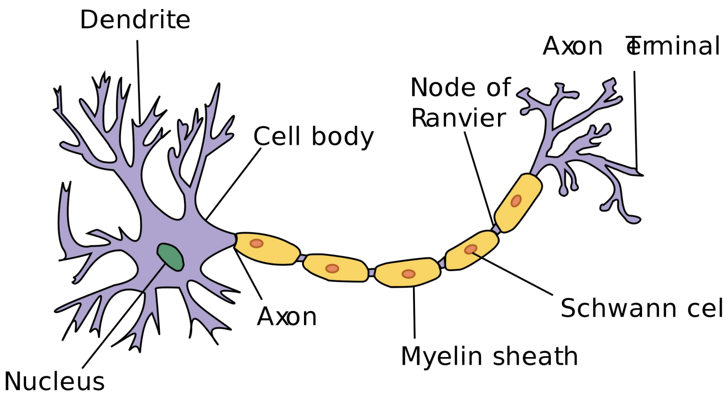
2022

جامعة الإسكندرية
ALEXANDRIA
UNIVERSITY

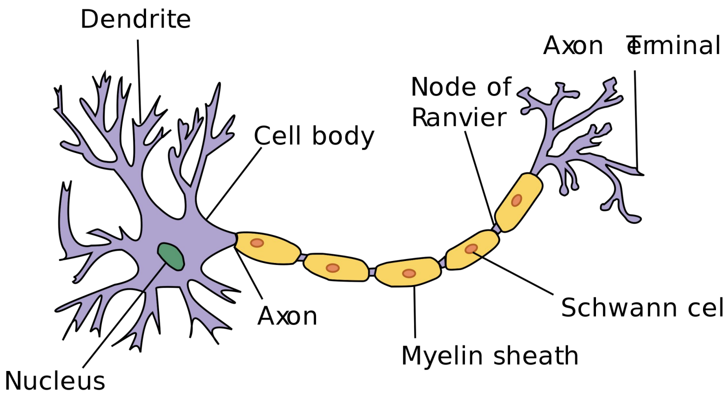
Inspiration from our Brains!

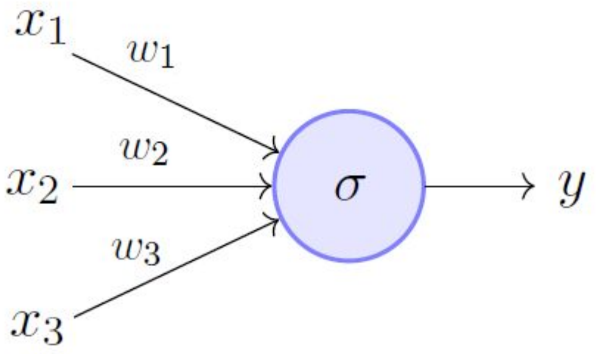
Biological Neurons

- Dendrite receives, integrates signals from other neurons.
- Neuron cell body “decides”.
- Axons communicate decision to other neurons.



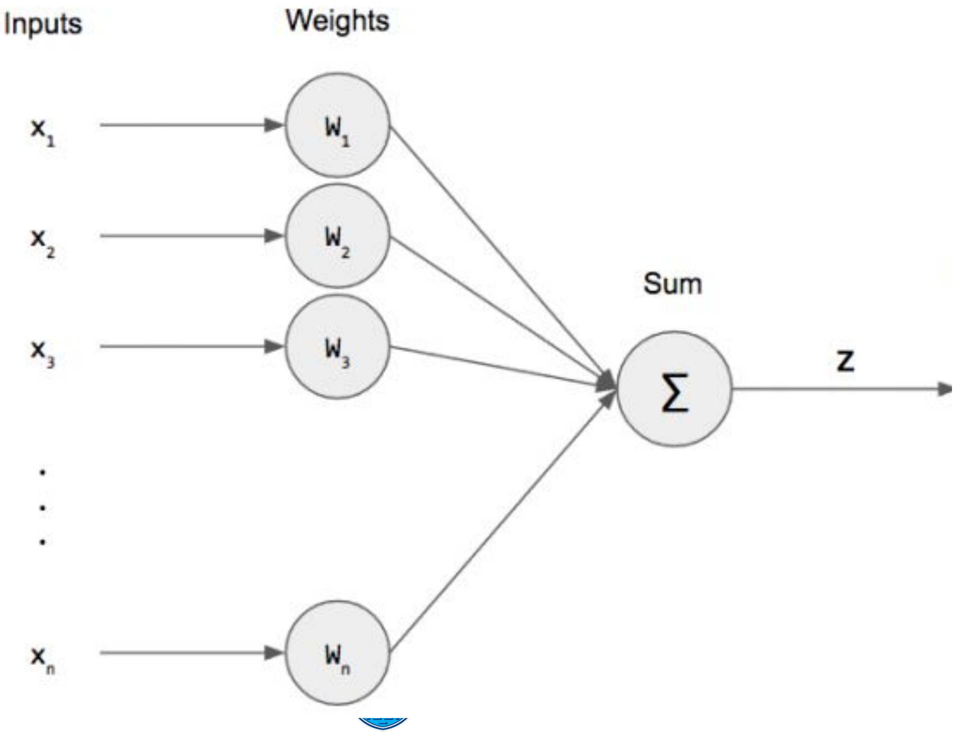
Modeling Individual Neurons (Perceptron)





Artificial Neuron

Modeling Individual Neurons (Perceptron)



Linear Models Require Bias or Intercept

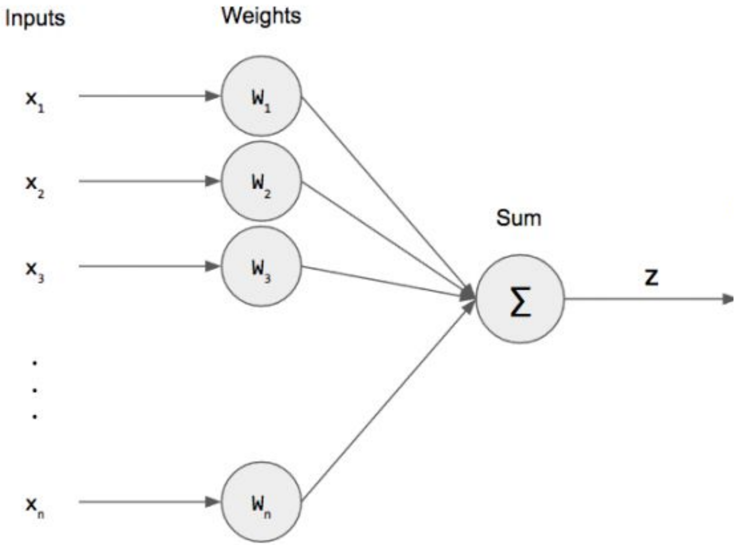
- W are weights and b is called a bias (also called an offset or intercept).

  As we learned in linear regression to represent the data correctly we need to add bias to our equations.

- The weights determine the influence of each feature on our prediction and the bias just says what value the predicted output should take when all of the features take value 0.

- Even if we will never see any outputs with zero value, we still need the bias or else we will limit the expressivity of our model.

Modeling Individual Neurons (Perceptron)



$$z = \left(\sum_{i=1}^{n} x_i \times w_i \right) + b$$

Initialize Weight

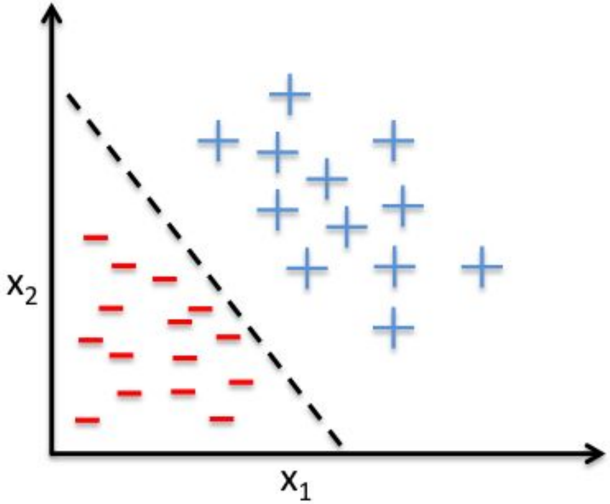
Weight initialization is an important design choice when working with neurons

W could be set to :

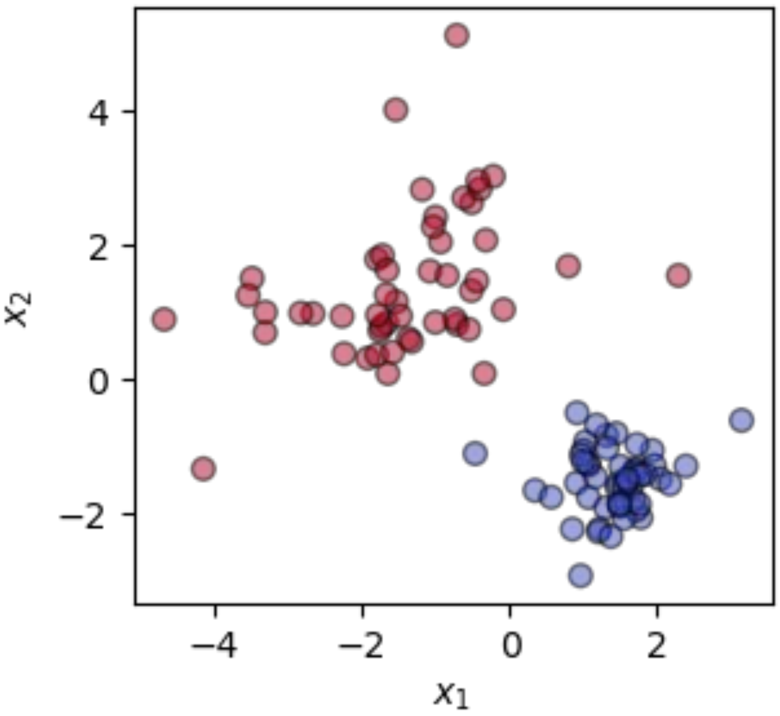
- W = 0 or 1 or -1 etc..
- W = Random Value
- W = Initializer Function eg. (Xavier and He Initializers) (Later)

Condition for Perceptron Success

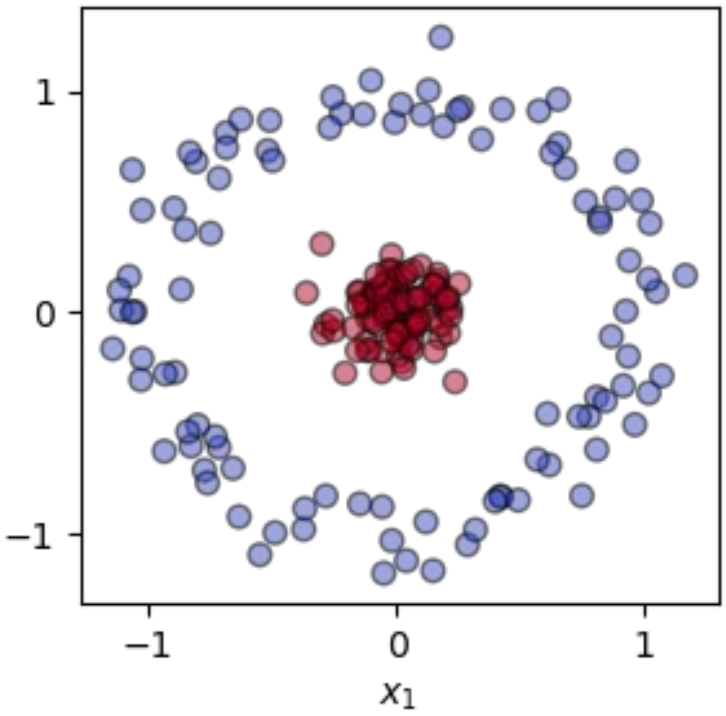
- The Perceptron learning rule is guaranteed to succeed if the data
    - Is linearly separable.
    - A hyperplane must exist that can separate positive and negative examples



Example of a linear decision boundary for binary classification.



(a) Linearly separable classes



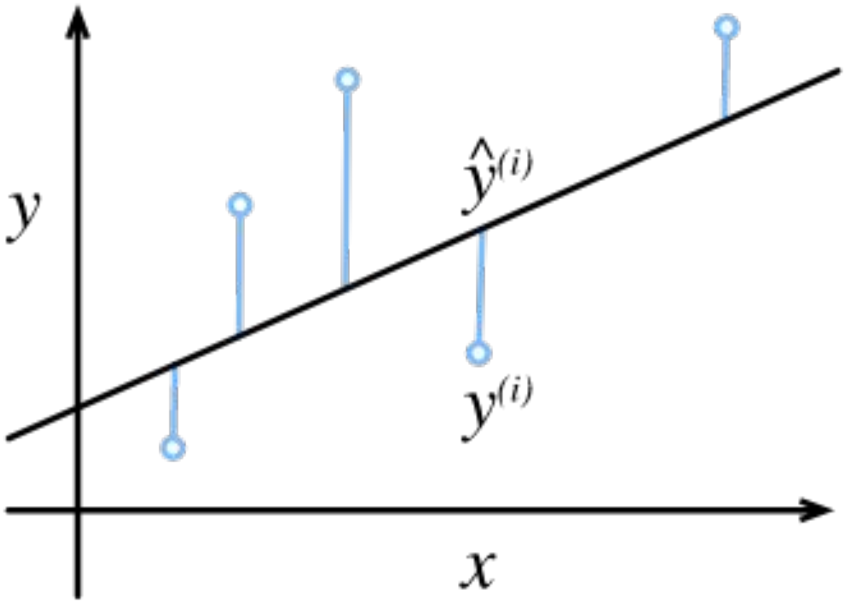
(b) Nonlinearly separable classes

How to measure the “Fitness” of our neurons?

- The loss function quantifies the distance between the real and predicted value of the target.
- The loss will usually be a non-negative number where smaller values are better and perfect predictions incur a loss of 0.
- The most popular loss function as mentioned in last lecture is the mean squared error.

$$\mathrm{MSE} = \underbrace{\boxed{\frac{1}{n}\sum_{i=1}^{n}}}_{\text{Mean}}\underbrace{\boxed{(Y_i - \hat{Y}_i)}}_{\text{Error}}\underbrace{\boxed{^2}}_{\text{Squared}}$$

How to measure the “Fitness” of our neurons?



$$\mathrm{MSE} = \underbrace{\boxed{\frac{1}{n}\sum_{i=1}^{n}}}_{\text{Mean}} (\underbrace{\boxed{Y_i - \hat{Y}_i}}_{\text{Error}})^{\boxed{2}_{\text{Squared}}}$$

ALEXANDRIA UNIVERSITY

In a Perfect World :

- We want to find parameters (w∗ ,b∗) that minimize the total loss across all our data.

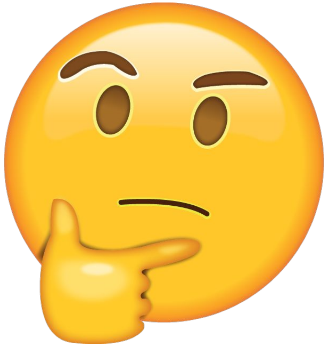
$$\mathbf{w}^*, b^* = \underset{\mathbf{w}, b}{\operatorname{argmin}}\ L(\mathbf{w}, b).$$

Open Question:

How to represent non-linear data?

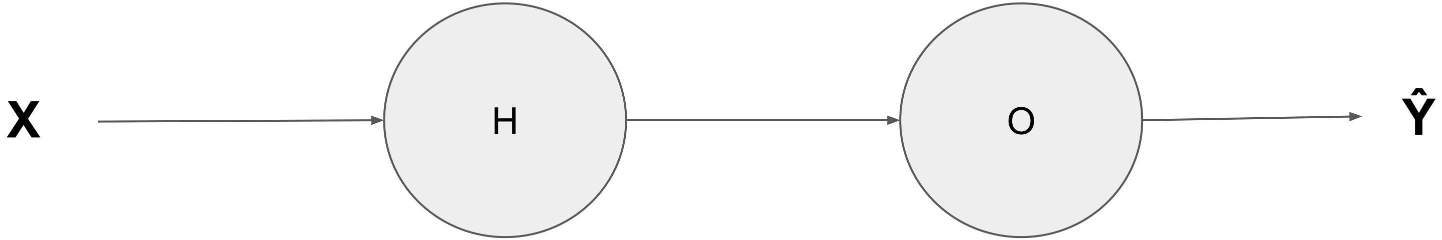
How to represent non-linear data?

Maybe add more *Neurons* ?



2 Neurons Model

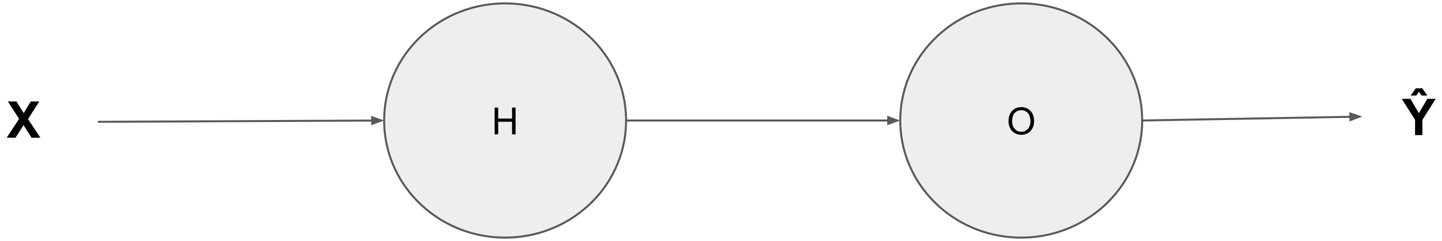
- Let's assume we have 2 Neurons, H and O.



2 Neurons Model

- Let's assume we have 2 Neurons, H and O.
- With representing equations :

$$\mathbf{H} = \mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)},$$
$$\mathbf{O} = \mathbf{H}\mathbf{W}^{(2)} + \mathbf{b}^{(2)}.$$



ALEXANDRIA UNIVERSITY

2 Neurons Model (Cont'd)

Calculate Final Output :

$$\mathbf{H} = \mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)},$$
$$\mathbf{O} = \mathbf{H}\mathbf{W}^{(2)} + \mathbf{b}^{(2)}.$$

2 Neurons Model (Cont'd)

Calculate Final Output :

$$\mathbf{H} = \mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)},$$
$$\mathbf{O} = \mathbf{H}\mathbf{W}^{(2)} + \mathbf{b}^{(2)}.$$

$$\mathbf{O} = (\mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)})\mathbf{W}^{(2)} + \mathbf{b}^{(2)}$$

2 Neurons Model (Cont'd)

Calculate Final Output :

$$\mathbf{H} = \mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)},$$
$$\mathbf{O} = \mathbf{H}\mathbf{W}^{(2)} + \mathbf{b}^{(2)}.$$

$$\mathbf{O} = (\mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)})\mathbf{W}^{(2)} + \mathbf{b}^{(2)}$$
$$= \mathbf{X}\mathbf{W}^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(2)}$$

2 Neurons Model (Cont'd)

Calculate Final Output :

$$\mathbf{H} = \mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)},$$
$$\mathbf{O} = \mathbf{H}\mathbf{W}^{(2)} + \mathbf{b}^{(2)}.$$

$$\begin{aligned}\mathbf{O} &= (\mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)})\mathbf{W}^{(2)} + \mathbf{b}^{(2)} \\ &= \mathbf{X}\mathbf{W}^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(2)} \\ &= \mathbf{X}\mathbf{W} + \mathbf{b}.\end{aligned}$$

2 Neurons Model (Cont'd)

Calculate Final Output :

$$\mathbf{H} = \mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)},$$
$$\mathbf{O} = \mathbf{H}\mathbf{W}^{(2)} + \mathbf{b}^{(2)}.$$

$$\begin{aligned}\mathbf{O} &= (\mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)})\mathbf{W}^{(2)} + \mathbf{b}^{(2)} \\ &= \mathbf{X}\mathbf{W}^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(1)}\mathbf{W}^{(2)} + \mathbf{b}^{(2)} \\ &= \boxed{\mathbf{X}\mathbf{W} + \mathbf{b}.}\end{aligned}$$

Just another Linear Model !!

Add Non-Linearity to Neurons

- In order to realize the potential of multi-neuron (multilayer) architectures, we need one more key ingredient:

  a nonlinear “activation” function with symbol $\boldsymbol{\sigma}$ to be applied to each output of a neuron

Add Non-Linearity to Neurons

- In general, with activation functions in place, it is no longer possible to collapse our 2 neurons for example into a linear model:

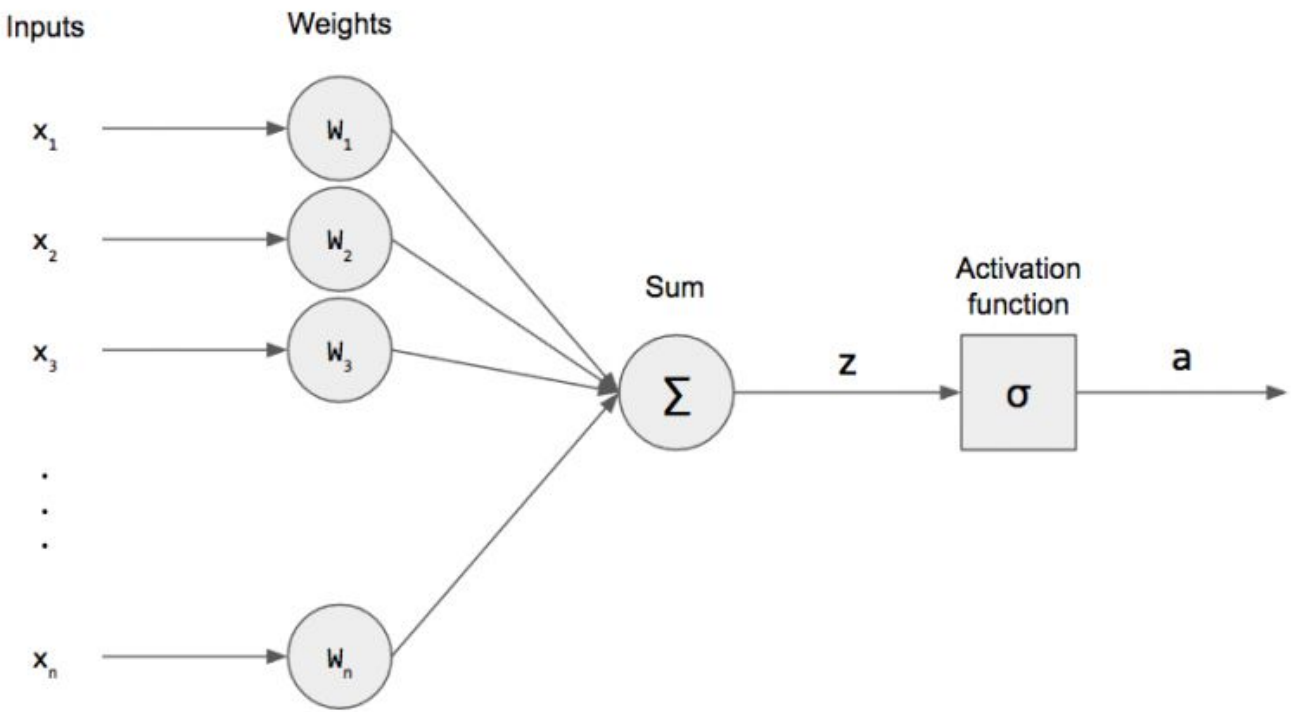
$$\mathbf{H} = \sigma(\mathbf{X}\mathbf{W}^{(1)} + \mathbf{b}^{(1)}),$$
$$\mathbf{O} = \mathbf{H}\mathbf{W}^{(2)} + \mathbf{b}^{(2)}.$$

Note : O doesn't have activation function just for simplification but generally each output should have activation applied

Activation Functions

- So generally, Activation functions decide whether a neuron should be activated or not by calculating the weighted sum and further adding bias with it.
- They are also differentiable operators. (Important for next lecture!)

Modeling Individual Neurons with Activation

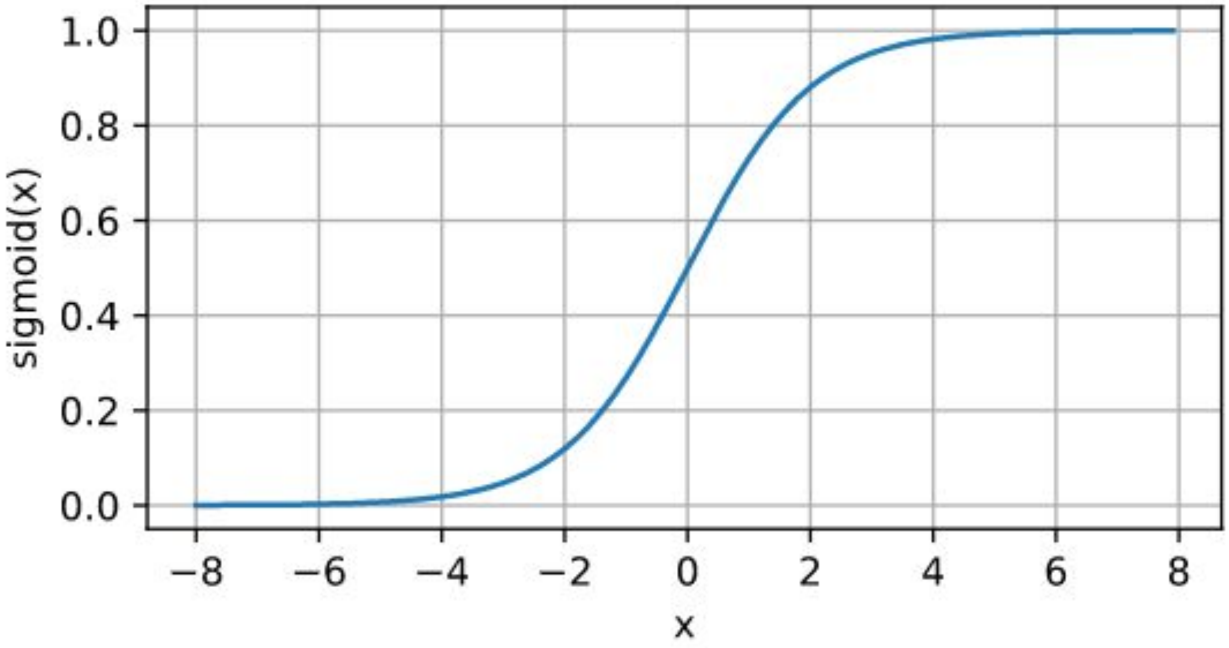


Activation Functions : Sigmoid Function

- The sigmoid function transforms its inputs, for which values lie in the domain R , to outputs that lie on the interval (0, 1).
- For that reason, the sigmoid is often called a squashing function: it squashes any input in the range (-inf, inf) to some value in the range (0, 1)

$$\operatorname{sigmoid}(x) = \frac{1}{1+\exp(-x)}.$$

Activation Functions : Sigmoid Function

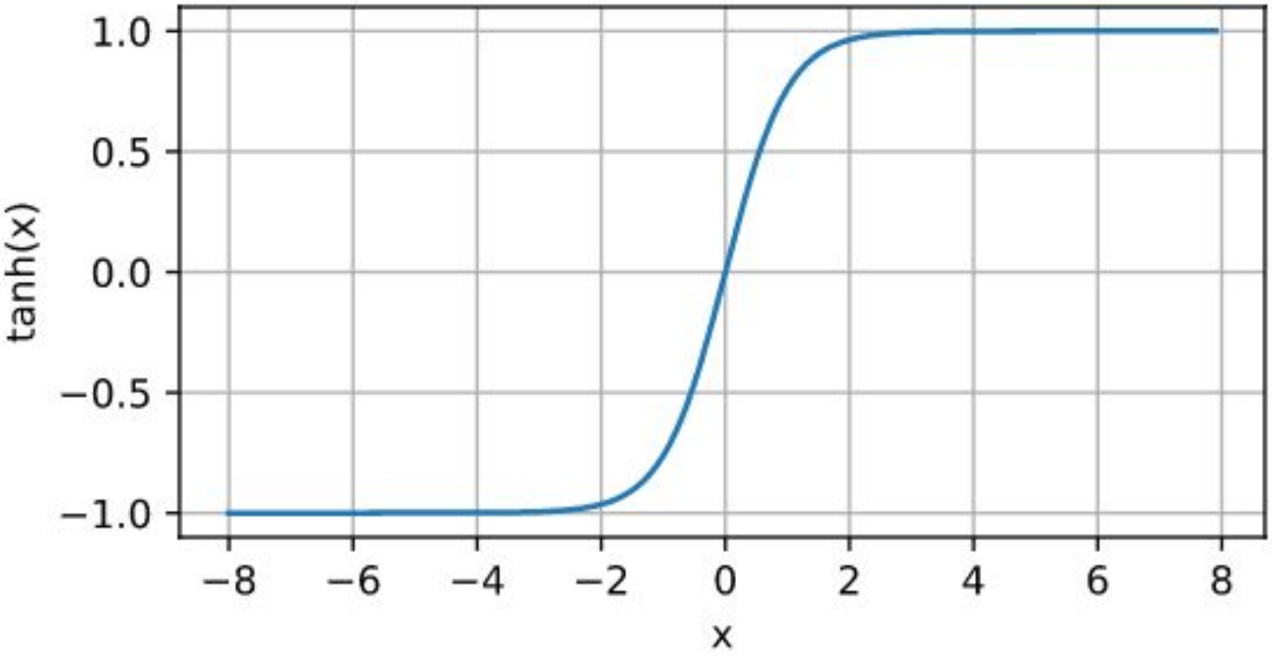


Activation Functions : Tanh Function

- Like the sigmoid function, the tanh (hyperbolic tangent) function also squashes its inputs, transforming them into elements on the interval between -1 and 1

$$\tanh(x) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)}.$$

Activation Functions : Tanh Function

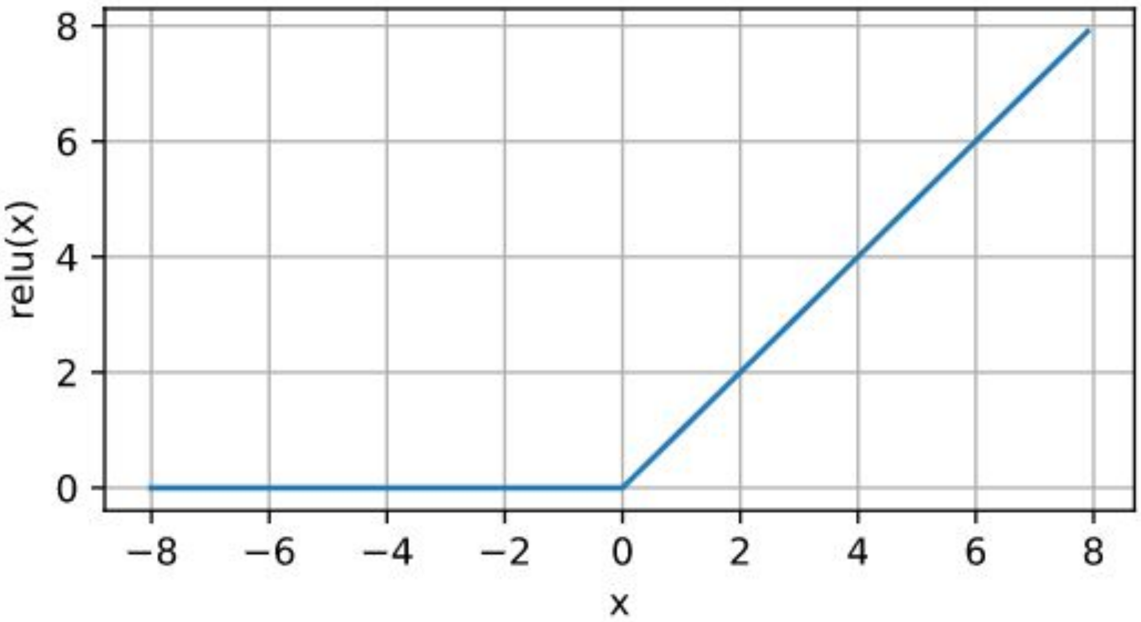


Activation Functions : ReLU Function

- The most popular choice, due to both simplicity of implementation and its good performance on a variety of predictive tasks, is the rectified linear unit (ReLU).
- ReLU provides a very simple nonlinear transformation. Given an element x , the function is defined as the maximum of that element and 0

$$\mathrm{ReLU}(x) = \max(x, 0).$$

Activation Functions : ReLU Function

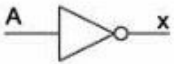


Bonus Quiz

Show that :

$$tanh(x) + 1 = 2 * sigmoid(2x)$$

Logic Gates

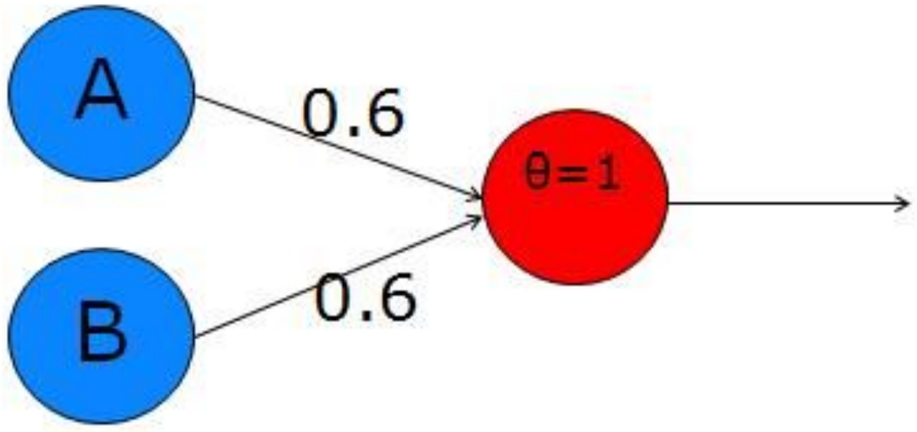
Name	NOT	AND	NAND	OR	NOR	XOR	XNOR
Alg. Expr.	$\overline{A}$	AB	$\overline{AB}$	$A+B$	$\overline{A+B}$	$A \oplus B$	$\overline{A \oplus B}$
Symbol							
Truth Table	A X 0 **1** 1 **0**	B A X 0 0 **0** 0 1 **0** 1 0 **0** 1 1 **1**	B A X 0 0 **1** 0 1 **1** 1 0 **1** 1 1 **0**	B A X 0 0 **0** 0 1 **1** 1 0 **1** 1 1 **1**	B A X 0 0 **1** 0 1 **0** 1 0 **0** 1 1 **0**	B A X 0 0 **0** 0 1 **1** 1 0 **1** 1 1 **0**	B A X 0 0 **1** 0 1 **0** 1 0 **0** 1 1 **1**

AND Gate

- Looking back at the logic table for the A^B, we can see that we only want the neuron to output a 1 when both inputs are activated. To do this, we want the sum of both inputs to be greater than the threshold, but each input alone must be lower than the threshold.

A	B	A^B
0	0	0
0	1	0
1	0	0
1	1	1

AND Gate

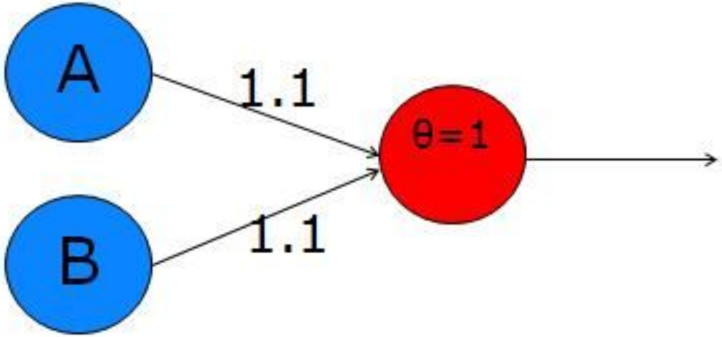


OR Gate

- In this case, we want the output to be 1 when either or both of the inputs, A and B, are active, but 0 when both of the inputs are 0.

A	B	AvB
0	0	0
0	1	1
1	0	1
1	1	1

OR Gate



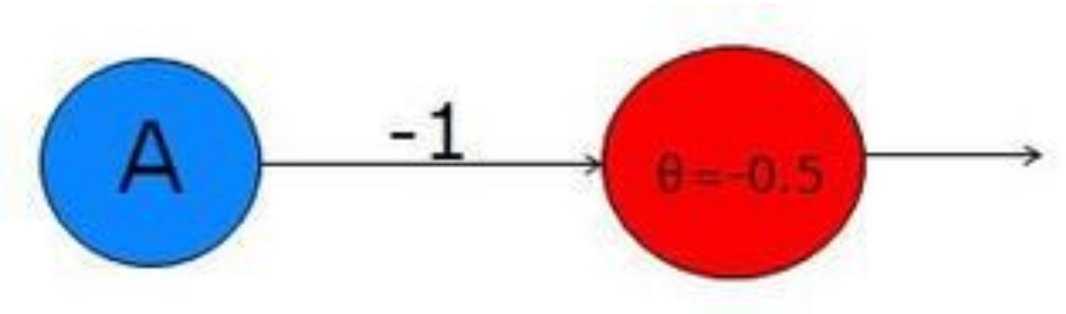
Not Gate

- In NOT Gate, We have to change a 1 to a 0 - this is easy, just make sure that the input doesn't exceed the threshold. However, we also have to change a 0 to a 1 - how can we do this?

A	¬A
1	0
0	1

Hint : Threshold can be negative

NOT Gate



ALEXANDRIA
UNIVERSITY

What is the least amount of neurons need to represent XOR gate?

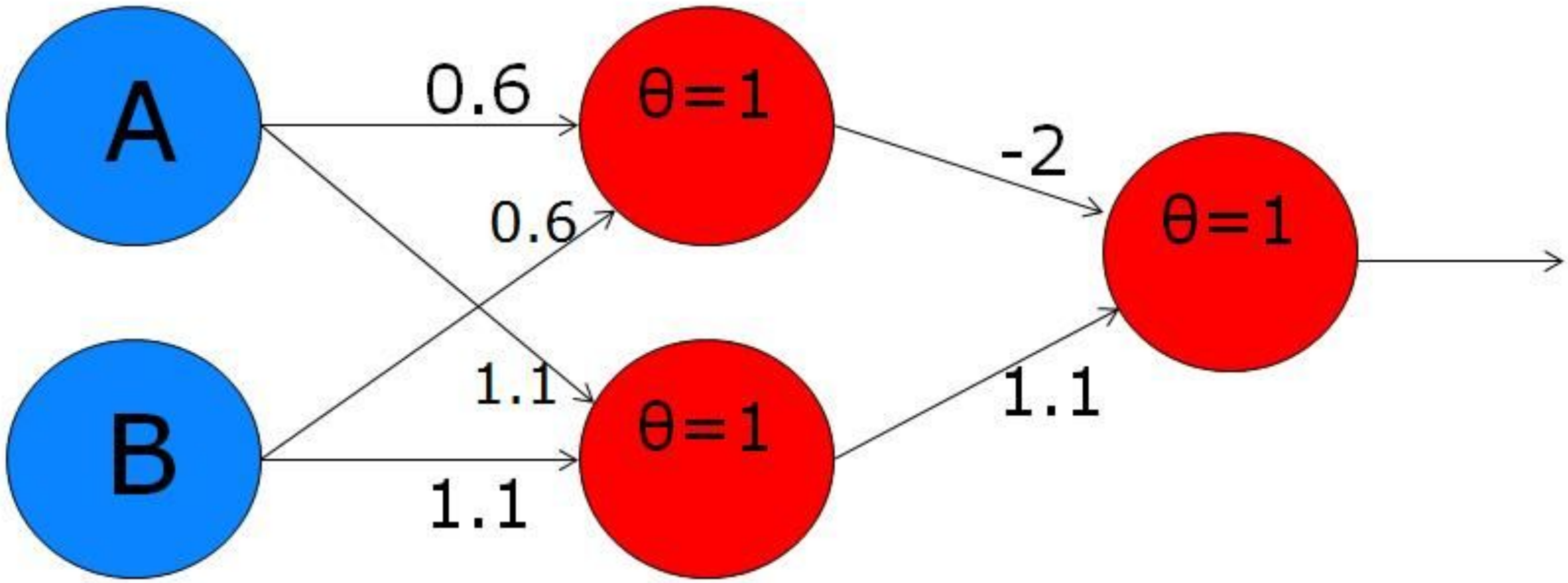
Truth Table of 2-input XOR Gate		
A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

جامعة الإسكندرية
ALEXANDRIA
UNIVERSITY

It is not possible to set up a single neuron to perform the XOR operation!

Truth Table of 2-input XOR Gate		
A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

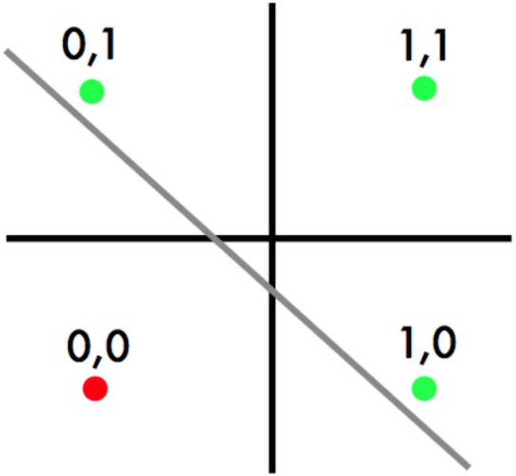
جامعة الإسكندرية
ALEXANDRIA
UNIVERSITY


A
B
0.6
0.6
1.1
1.1
θ=1
θ=1
-2
1.1
θ=1

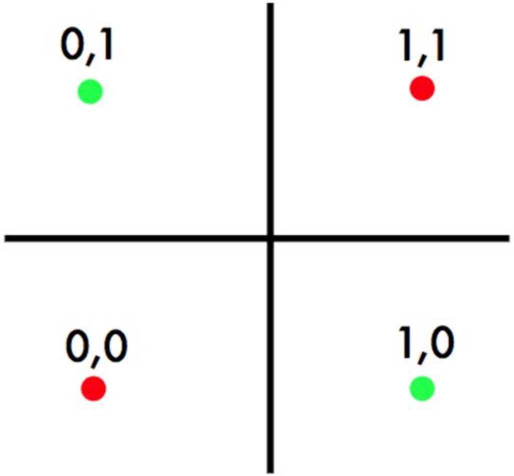
It is not possible to set up a single neuron to perform the XOR operation

<u>Why?</u>

The XO



OR

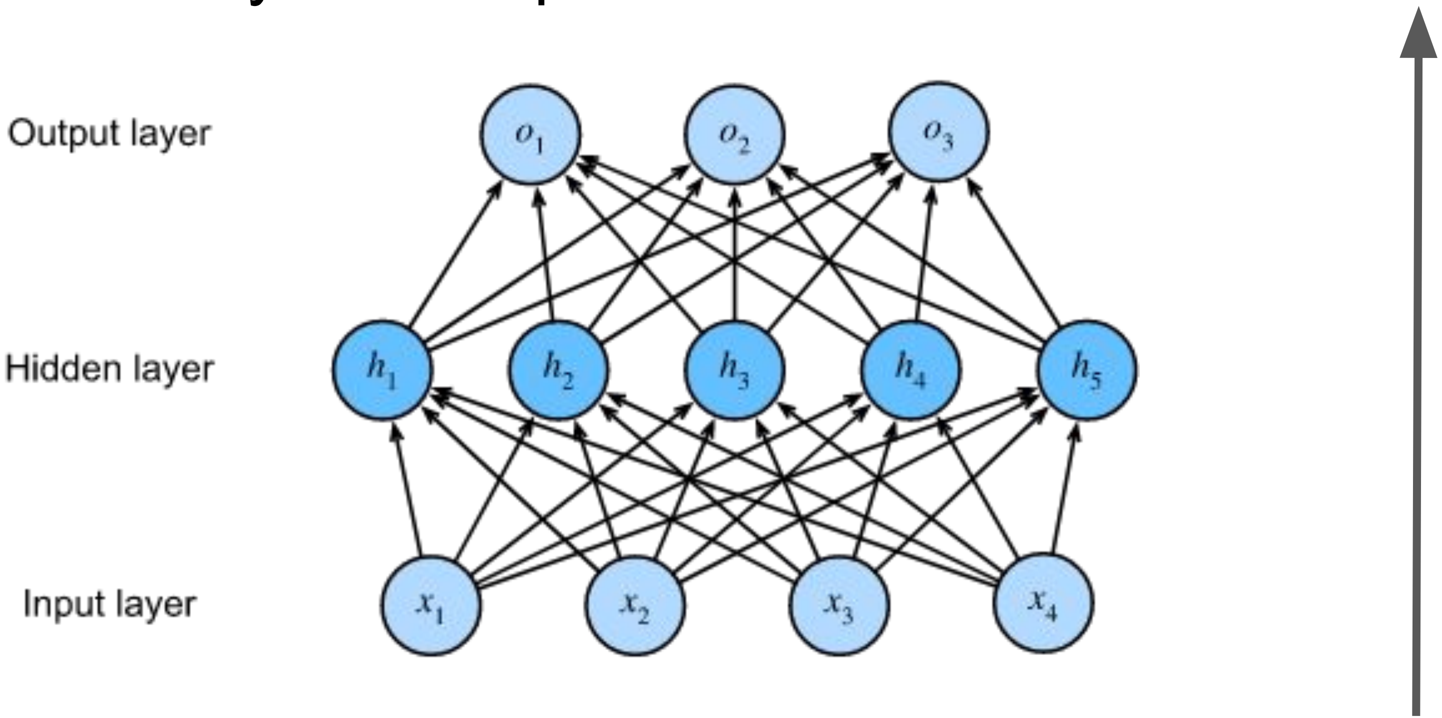


XOR

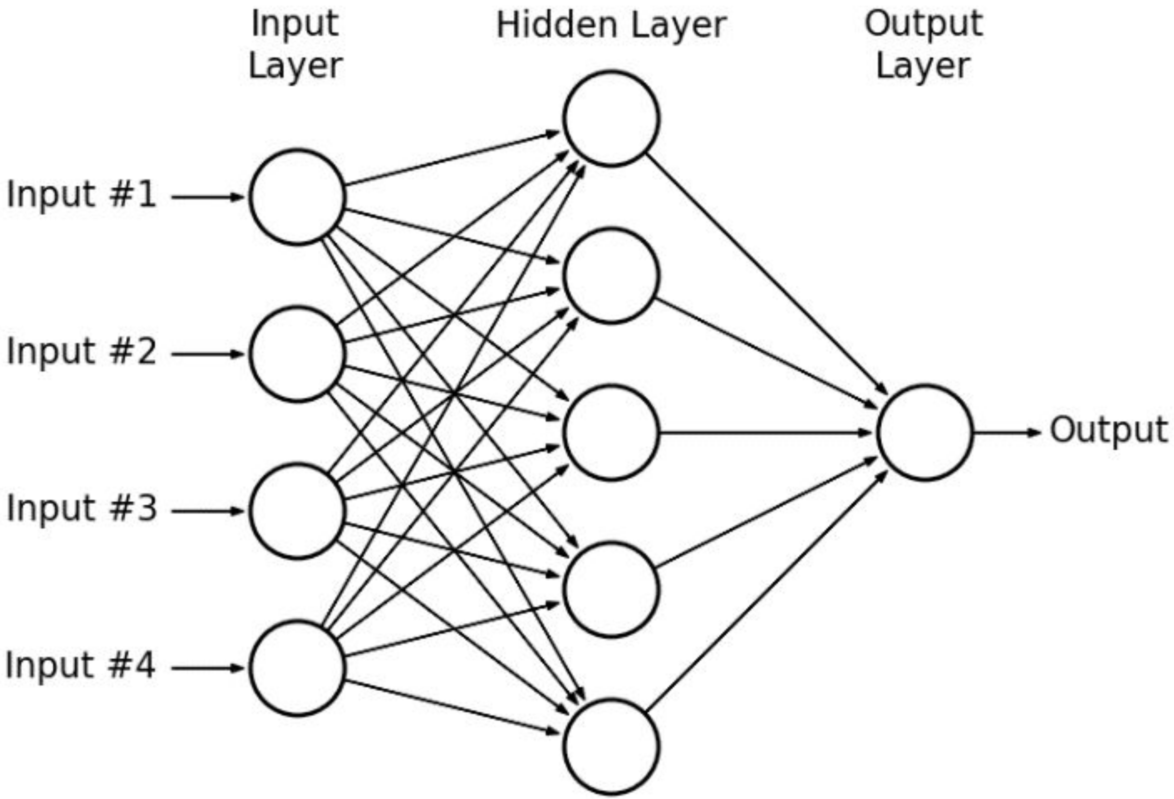
MLP : Multi Layer Perceptrons

- We can overcome the limitations of linear models and handle a more general class of functions by incorporating one or more hidden layers.
- The easiest way to do this is to stack many fully-connected layers on top of each other.
- Each layer feeds into the layer above it, until we generate outputs.
- We can think of the first $L-1$ layers as our representation and the final layer as our linear predictor.

MLP : Multi Layer Perceptron



MLP : Multi Layer Perceptron



Labs

- Continue : Python Libraries
- Implementation of neurons and activation functions in Python

جامعة الإسكندرية
ALEXANDRIA
UNIVERSITY