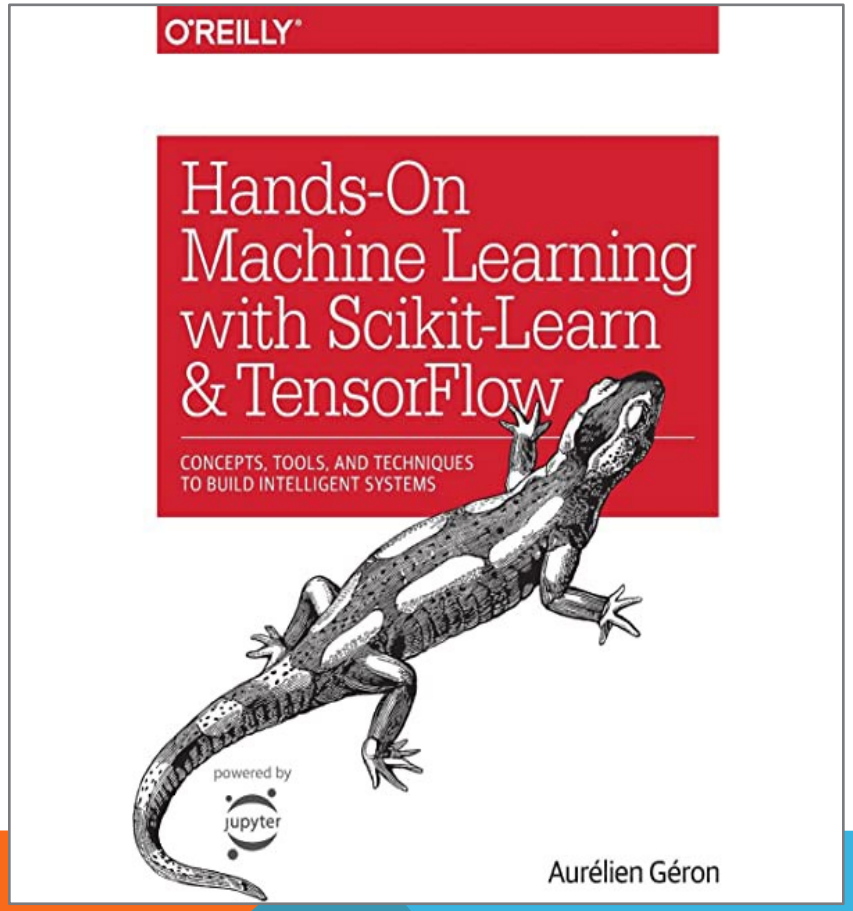
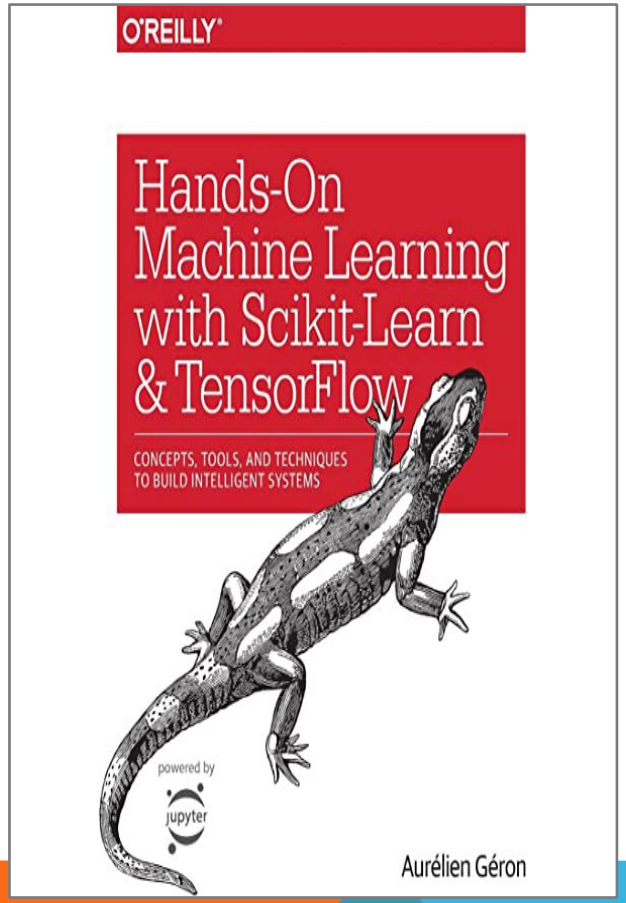
Machine Learning



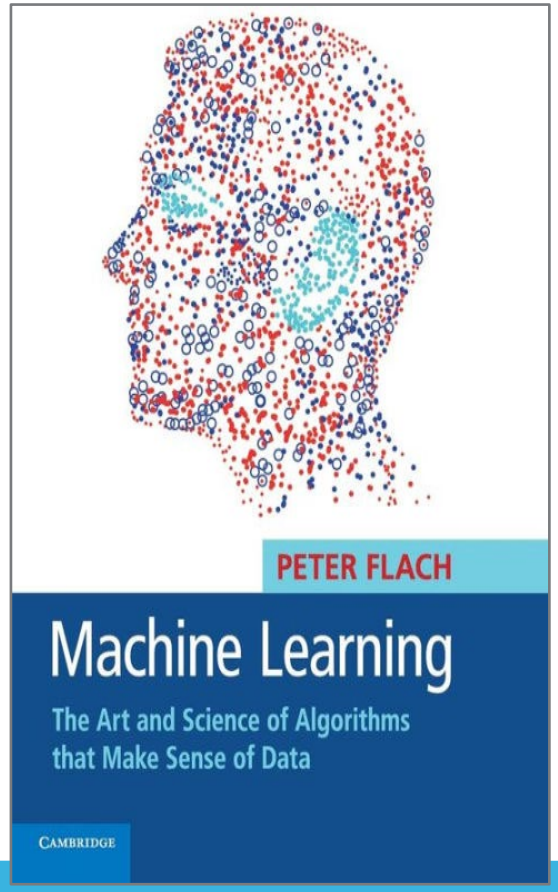
Introduced by

Dr. Ebtsam Adel

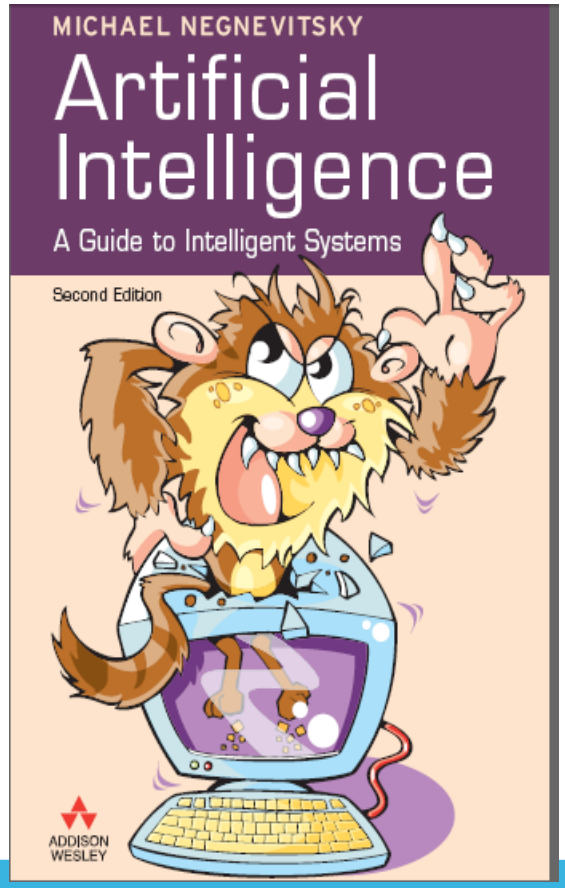
Materials



Aurélien Géron



PETER FLACH



Michael Negnevitskyt

Introduced by

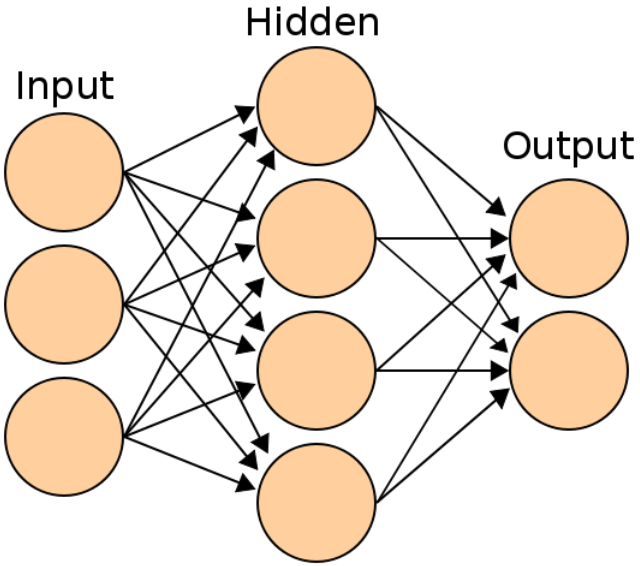
Dr. Ebtsam Adel

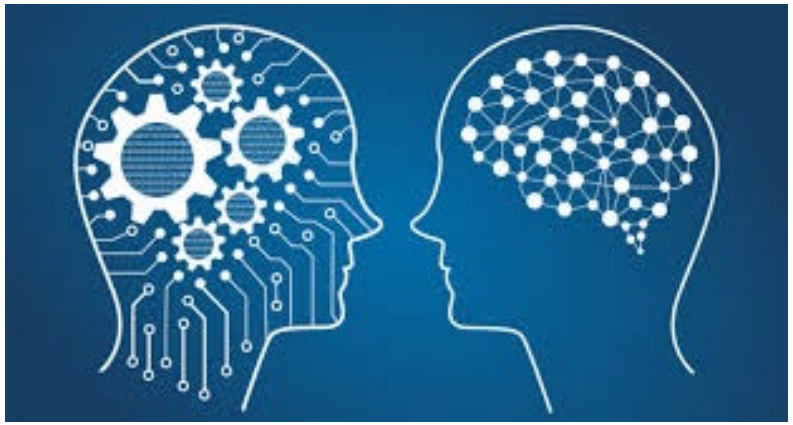

QUIZ!

- Can you name four common unsupervised tasks?

- If your model performs great on the training data but generalizes poorly to new instances, what is happening? Can you name three possible solutions?

Artificial Neural Networks (P1)





Introduced by

Dr.Ebtsam Adel

AGENDA

Biological Background
Artificial Neurons
Activation Functions
A Simple Perceptron
Neural Network Characteristics

AGENDA

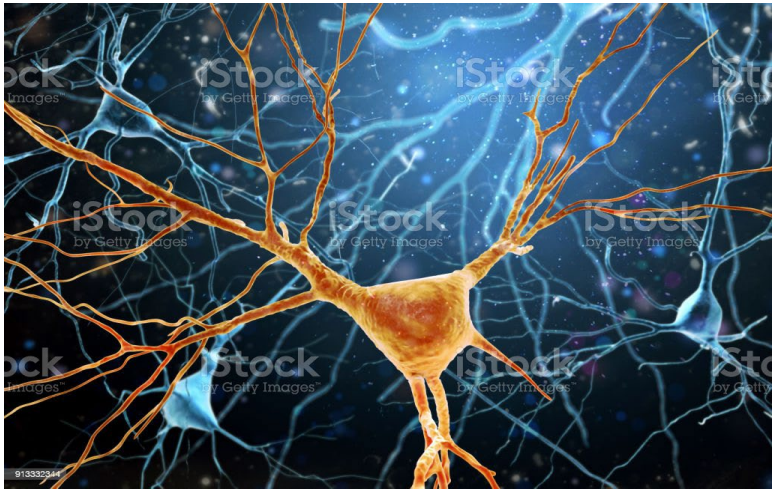
Biological Background
Artificial Neurons
Activation Functions
A Simple Perceptron
Neural Network Characteristics

BIOLOGICAL BACKGROUND

Neuron

- The basic element of the brain is a natural neuron. a neuron is the basic building block for all types of neural networks.

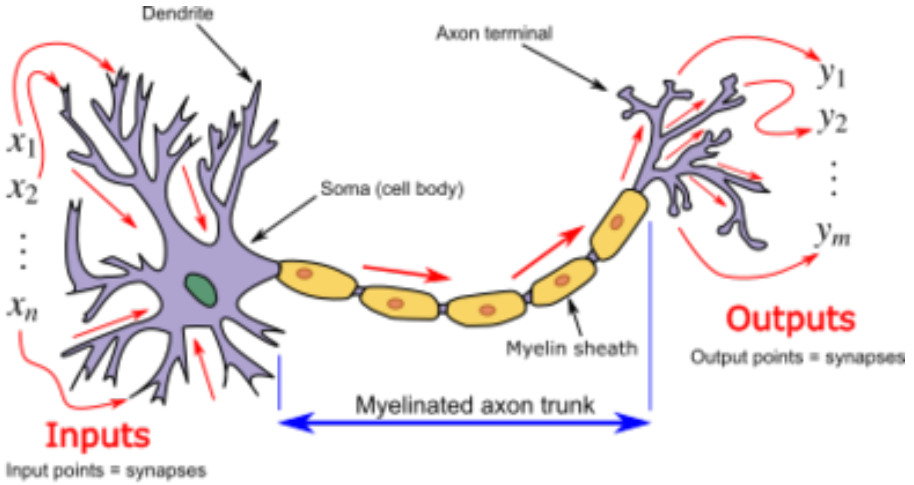
- The human brain incorporates nearly **10 billion** neurons and 60 trillion connections, **synapses**, between them.



BIOLOGICAL BACKGROUND

❖ **Neuron consists of:**

- ❑ Dendrites: Carry signals in.
- ❑ Cell body: processing.
- ❑ Axon: carry signals away
- ❑ Synapses: Size changes in response to learning. The synapse resistance to the incoming signal can be changed during a "learning" process.



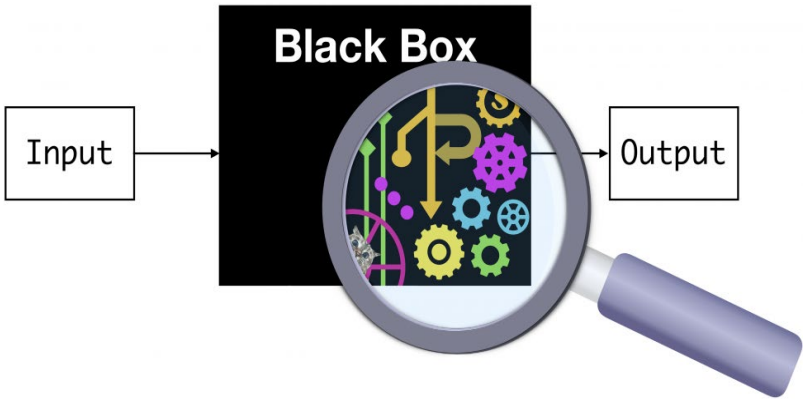
Each neuron receives a number of input signals through its connections; and always produce single output.

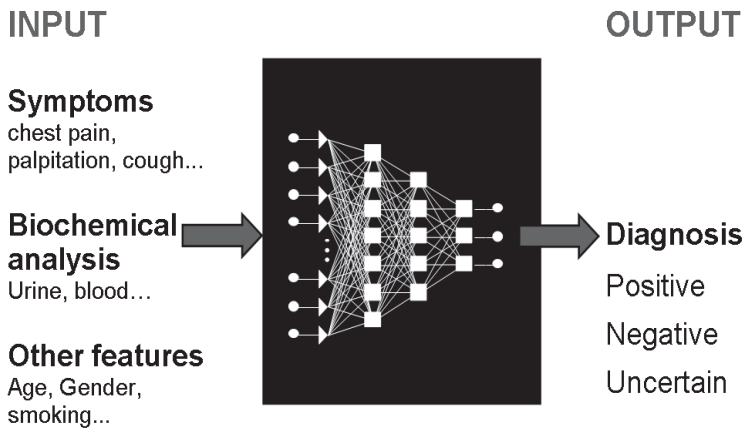
AGENDA

Biological Background
Artificial Neurons
Activation Functions
A Simple Perceptron
Neural Network Characteristics

NEURAL NETWORKS INTRO.

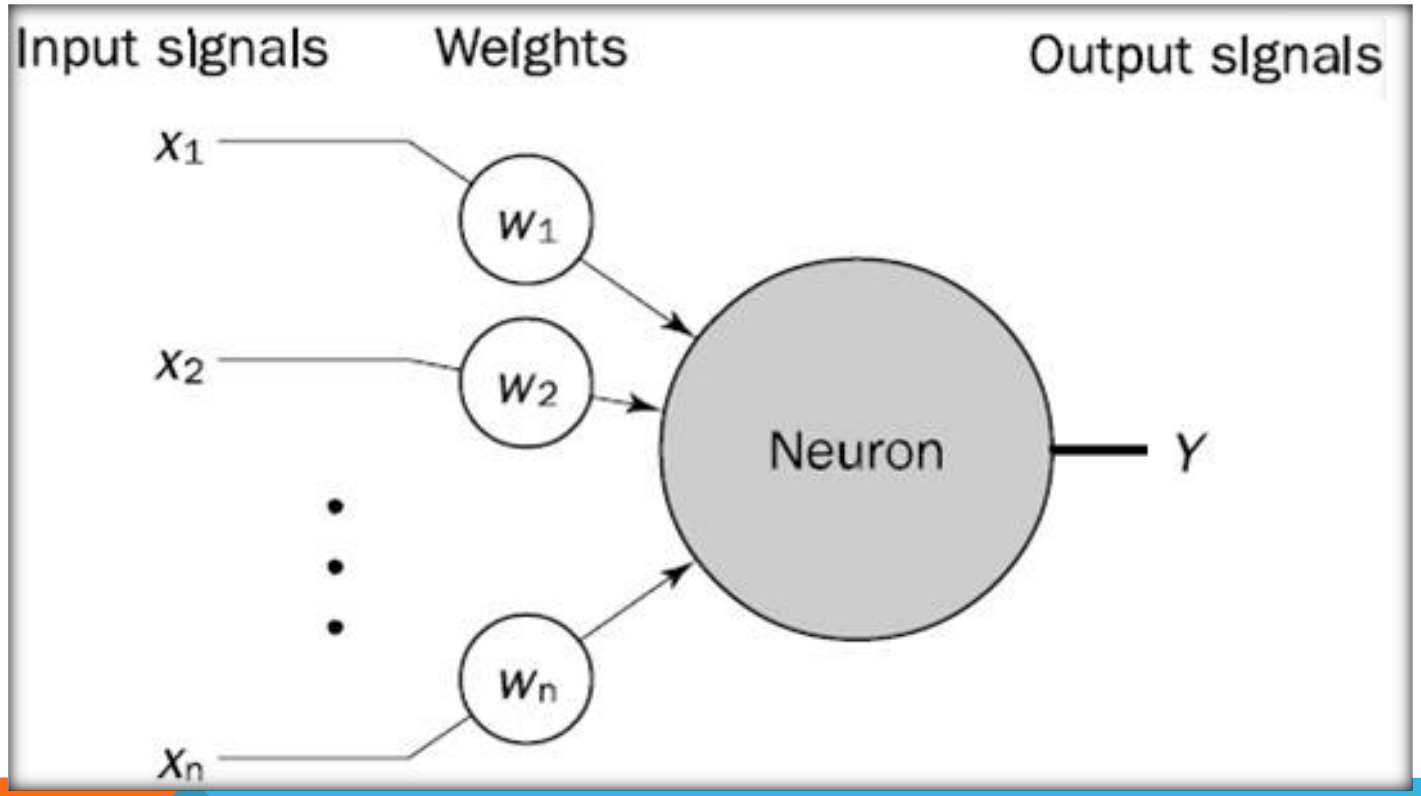
- We can think of a neuron as a sort of black box, receiving input vector x then producing a scalar output y. The same output value *y* can be sent out through multiple edges emerging from the neuron.





ARTIFICIAL NEURONS- FROM BIOLOGY TO SIMULATION

Neuron as a simple computing element:



ARTIFICIAL NN VS. BIOLOGICAL NN

- Mapping between artificial neural network and biological neural network

Biological Neural Network	*Artificial Neural Network*
Soma	Neuron
Dendrite	Input
Axon	Output
Synapse	Weight

ARTIFICIAL NEURONS- FROM BIOLOGY TO SIMULATION

- A **neural network** (NN) is an abstract computer model of the human brain. The human brain has an estimated 10^{11} tiny units called **neurons**.
- network is viewed as a graph, neurons can be represented as nodes (or vertices), and interconnections as edges.
- In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase.

ARTIFICIAL NEURONS

- The neural network of the brain is considered to be the fundamental functional source of intelligence, which includes perception, cognition, and learning for humans as well as other living creatures.

- We have inputs x_1, x_2, ..., x_m coming into the neuron. These inputs are the stimulation levels of a natural neuron. Each input *xi* is multiplied by its corresponding weight w_i, then the product $x_i w_i$ is fed into the body of the neuron.

- The weights represent the biological synaptic strengths in a natural neuron.

ARTIFICIAL NEURONS- FROM BIOLOGY TO SIMULATION

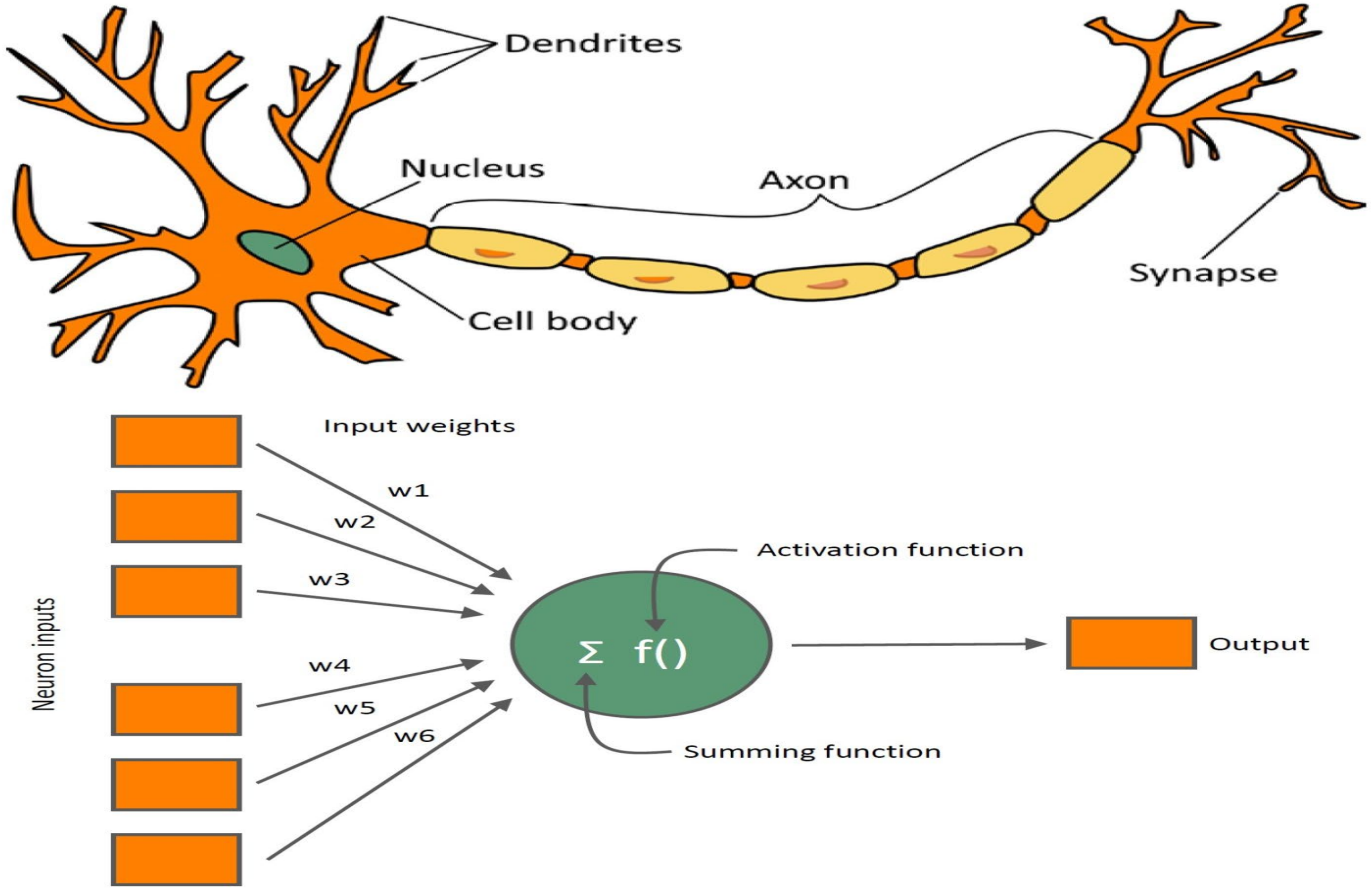
- The neuron computes the **weighted sum** of the input signals and compares the result with a **threshold** value.

$$X = \sum_{i=1}^{n} x_i w_i$$

- The **activation function** determine the response for the output of the neuron.

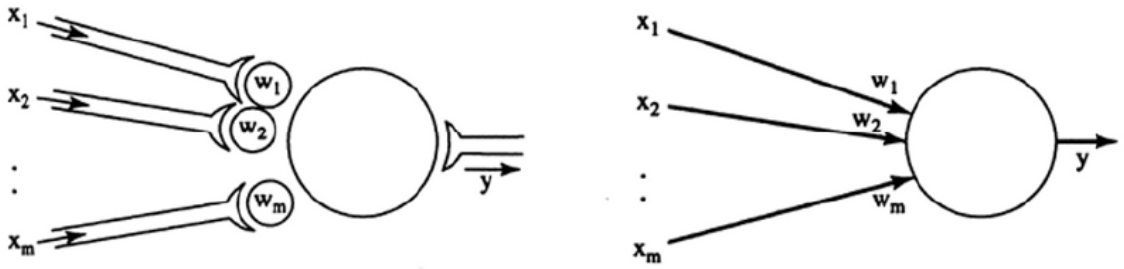
$Y = f(X)$

ARTIFICIAL NEURONS



ARTIFICIAL NEURONS

- The neuron adds up all the products for $i = 1, m$. The weighted sum of the products is usually denoted as ***net***.



- The weighted sum of the products is usually denoted as net. Therefore, the neuron evaluates:

$$net = x_1w_1 + x_2w_2 + \ldots\ldots\ldots\ldots + x_mw_m.$$

ARTIFICIAL NEURONS

In mathematical terms,

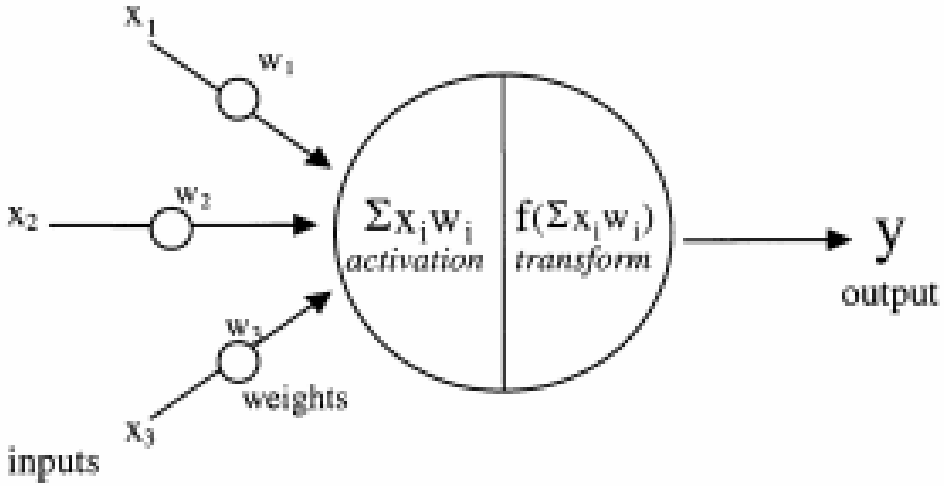
- Given two vectors x = (x1; x2;..... ; xm) and w = (w1;w2;.....;wm).
- **net = $x_1w_1 + x_2w_2 + ::: + x_mw_m$..**
- Finally, the neuron computes its output y as a certain function of net, i.e., y = f (net). This function is called the activation transfer function.

AGENDA

Biological Background
Artificial Neurons
Activation Functions
A Simple Perceptron
Neural Network Characteristics

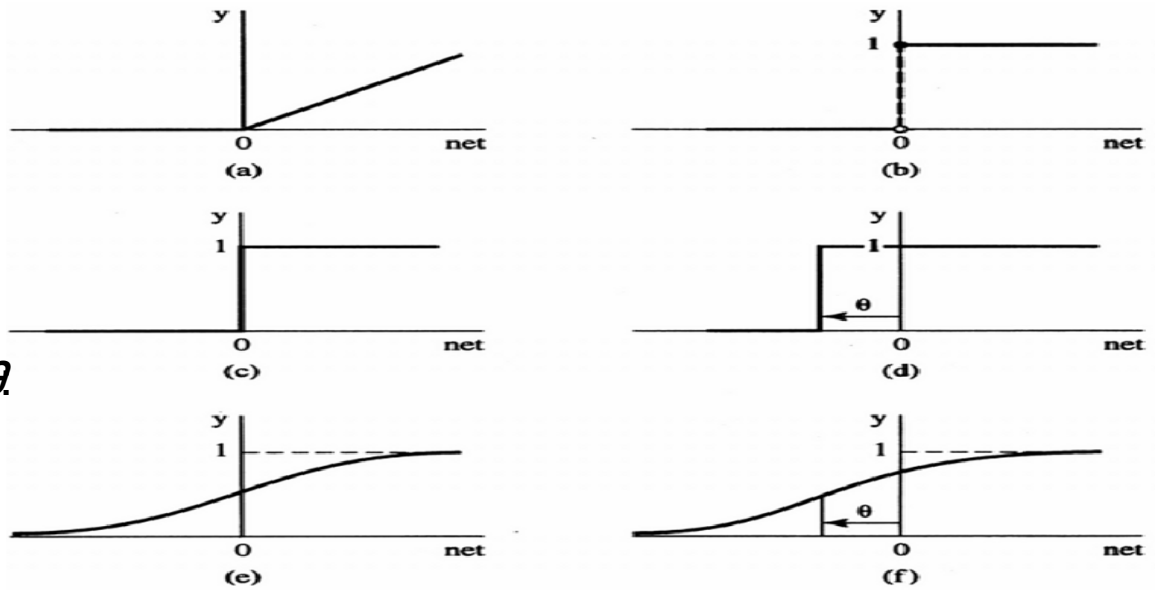
ACTIVATION (TRANSFER)) FUNCTIONS

- Activation functions are functions used in neural networks to computes the weighted sum of input and biases, of which is used to decide if a neuron can be fired or not.
- Activation functions of linear, step, sigmoid, etc.
- Various forms of activation functions can be defined depending on the characteristics of applications.



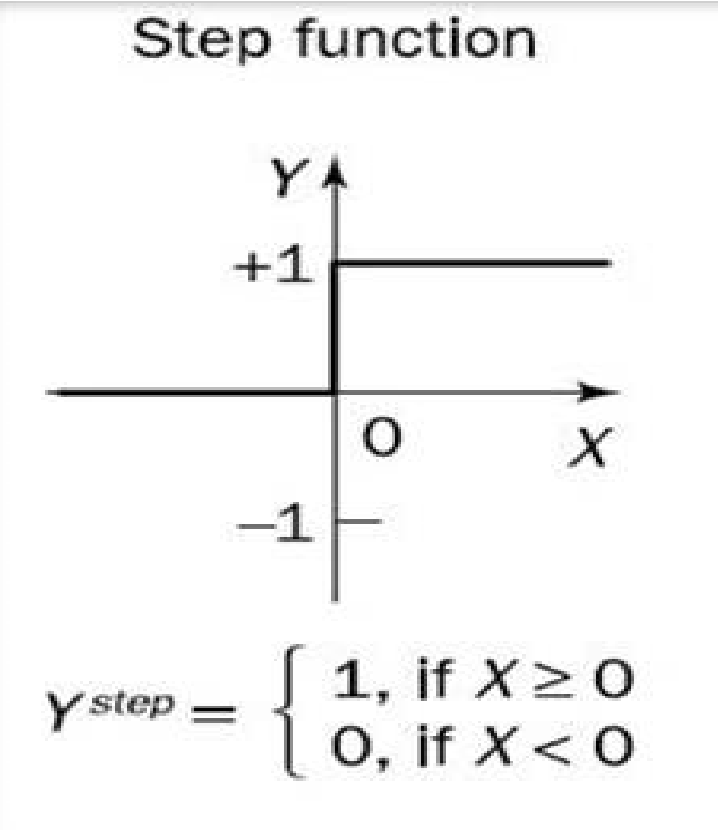
ACTIVATION (TRANSFER)) FUNCTIONS

(a) linear function,
(b) step function,
(c) step function,
(d) step function with threshold θ,
(e) sigmoid function,
(f) sigmoid function with threshold θ.



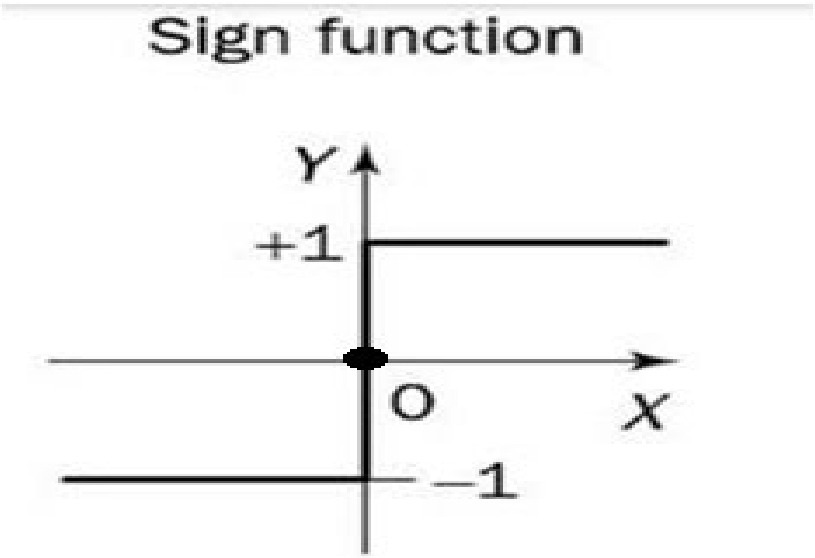
ACTIVATION (TRANSFER)) FUNCTIONS

Step Function



ACTIVATION (TRANSFER)) FUNCTIONS

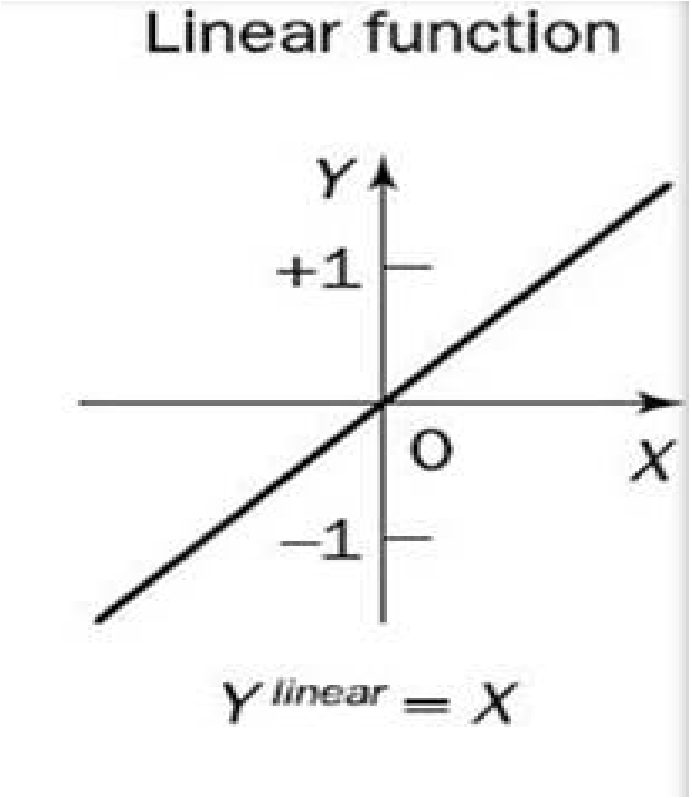
Sign Function



$$Y^{sign} = \begin{cases} +1, \text{ if } X > 0 \\ 0, \quad \text{if } X=0 \\ -1, \text{ if } X < 0 \end{cases}$$

ACTIVATION (TRANSFER)) FUNCTIONS

Linear Function

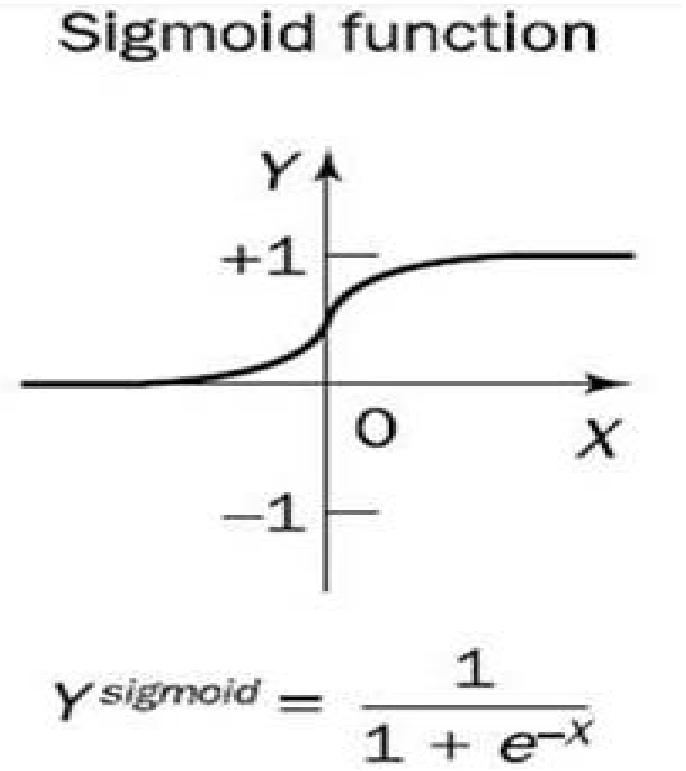


ACTIVATION (TRANSFER)) FUNCTIONS

Sigmoid Function

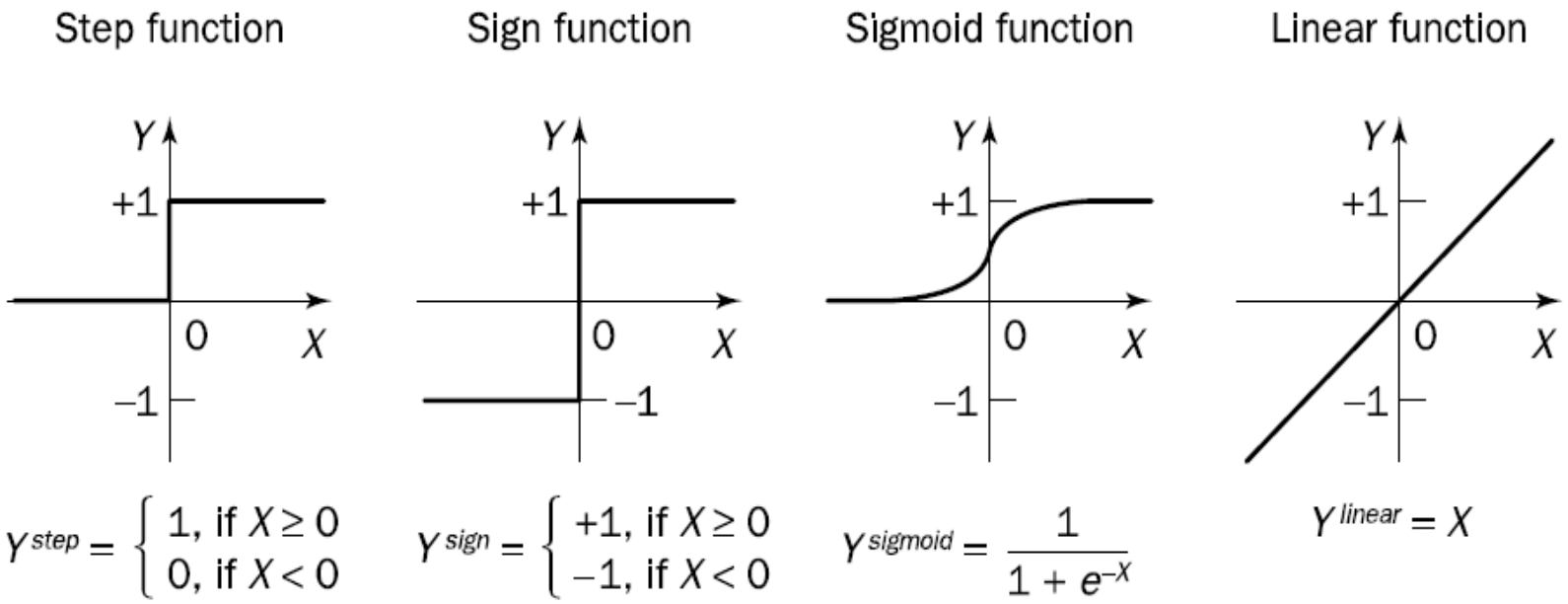
$$f(x) = \left(\frac{1}{(1 + exp^{-x})} \right)$$

e (exponential function)=2.7



$$Y_{sigmoid} = \frac{1}{1 + e^{-x}}$$

ACTIVATION (TRANSFER)) FUNCTIONS



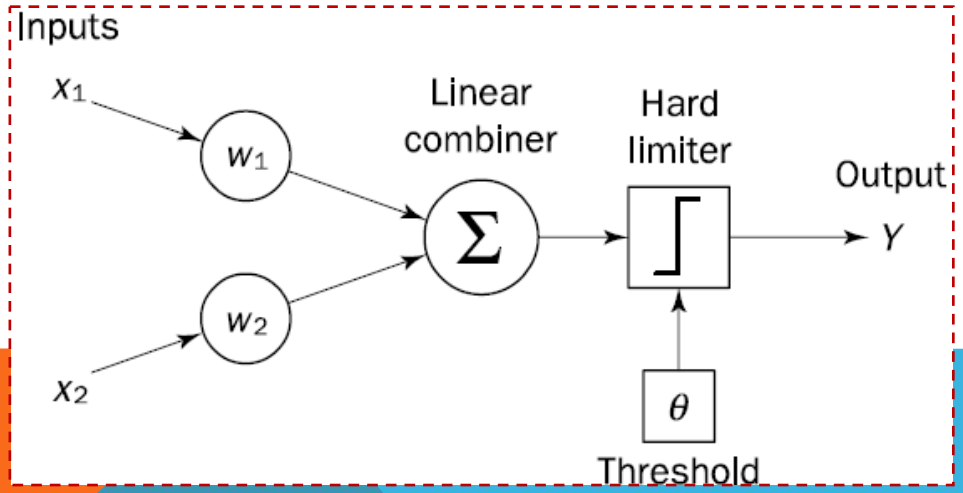
- The step and sign activation functions, also called hard limit functions, are often used in decision-making neurons for classification and pattern recognition tasks.

AGENDA

Biological Background
Artificial Neurons
Activation Functions
A Simple Perceptron
Neural Network Characteristics

A SIMPLE PERCEPTRON

- The operation of Rosenblatt's perceptron is based on the McCulloch and Pitts neuron model in 1958.
- The model consists of a linear combiner followed by a hard limiter.
- The weighted sum of the inputs is applied to the hard limiter, which produces an output equal to +1 if its input is positive and -1 if it is negative.
- The hyperplane is defined by the **linearly separable** function.



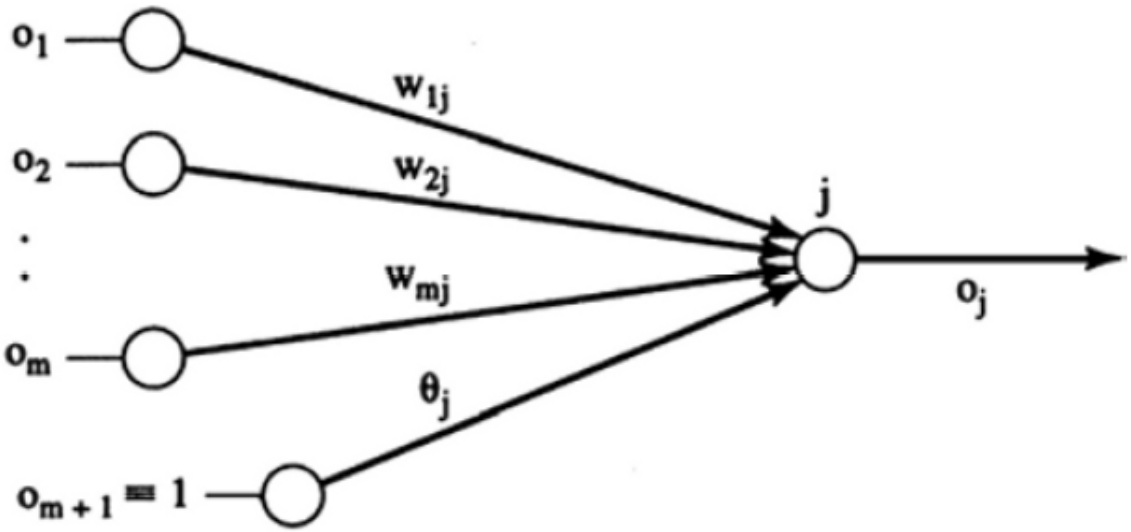
$$\sum_{i=1}^{n} x_i w_i - \theta = 0$$

A SIMPLE PERCEPTRON

- Only an input layer and an output layer (0-hidden layers), is called a simple perceptron.

Perception (brain)

1. Input
2. Processing
3. output



Example: attendance a training course

1. Trainer
2. Time
3. cost

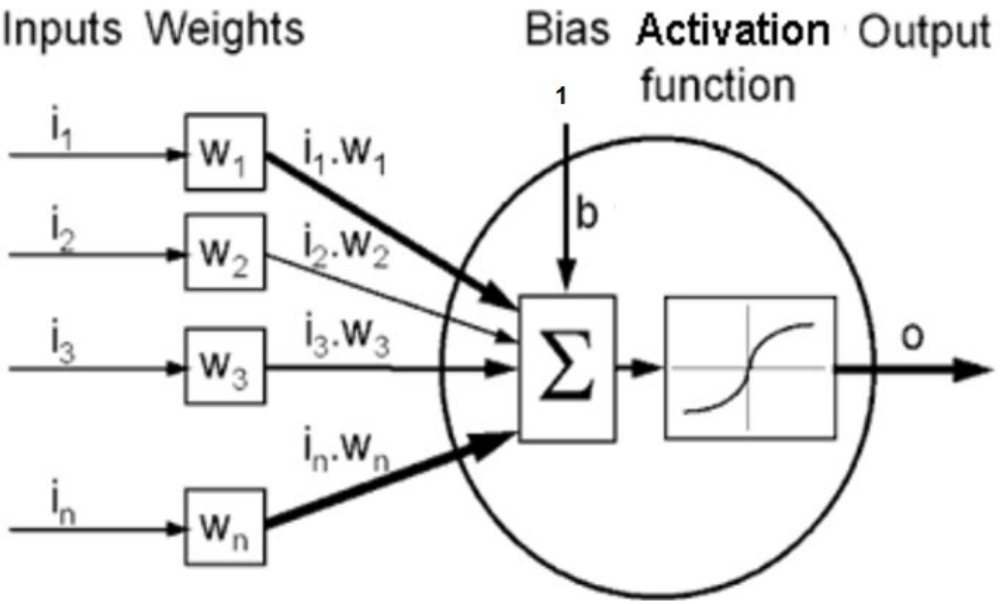
A SIMPLE PERCEPTRON

- **Representation** refers to whether a neural network is able to produce a particular function by **assigning appropriate weights**.

- Perceptron can learn anything it can represent.

- The activation function is a step function with threshold.

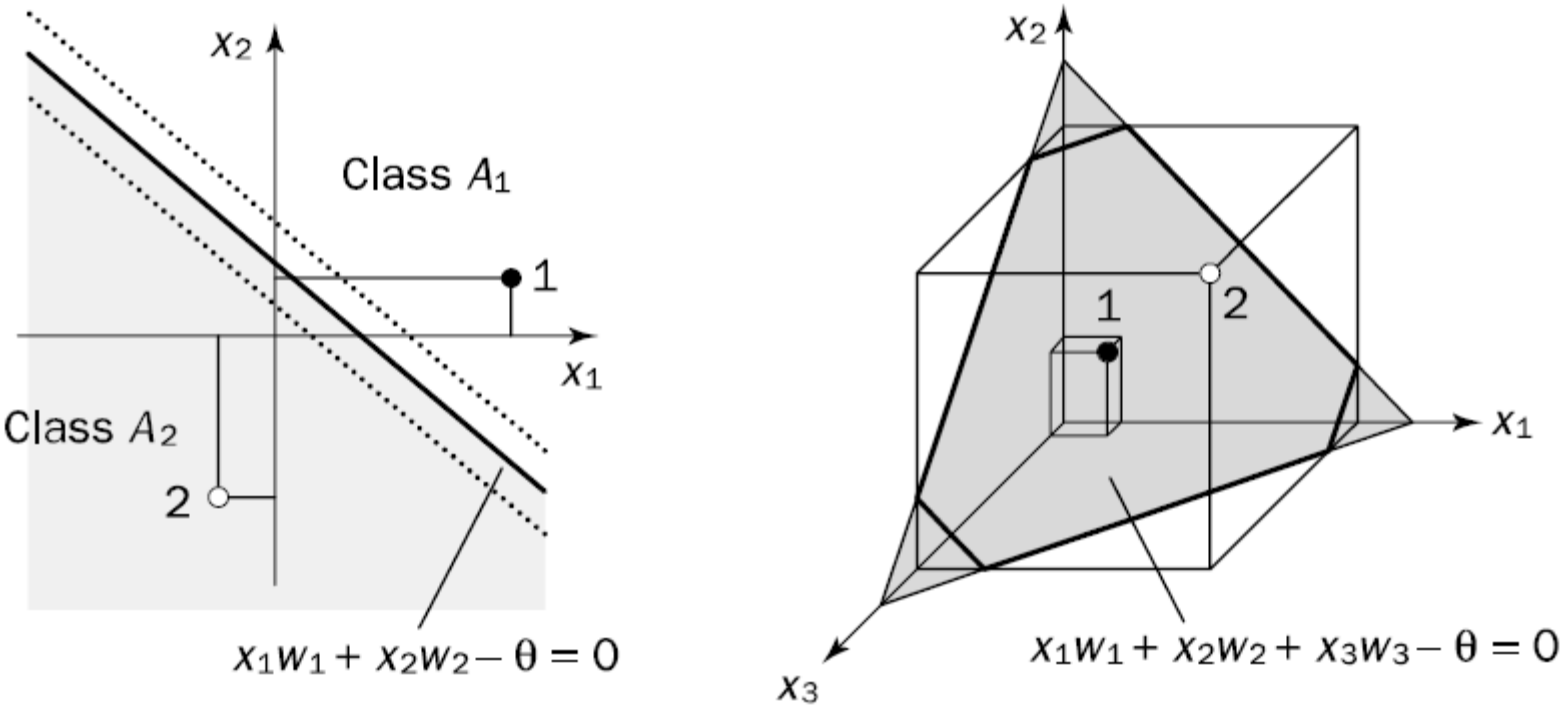
A SIMPLE PERCEPTRON



$$X = b + \sum_{i=1}^{n} \mathbf{I_i}\, w_i$$

$$Y^{sign} = \begin{cases} +1, \text{ if } X > 0 \\ \mathbf{0, \quad if\ X=0} \\ -1, \text{ if } X < 0 \end{cases}$$

Linear separability



Linear separability in the perceptron's

HOW DOES THE PERCEPTRON LEARN ITS CLASSIFICATION TASKS?

- This is done by making small adjustments in the weights to reduce the difference between the actual and desired outputs of the perceptron.
- The initial weights are randomly assigned, usually in the range [-0.5; 0.5], and then updated to obtain the output consistent with the training examples.
- If at iteration p, the actual output is Y(p) and the desired output is $Y_d(p)$, then the **error** is given by:

HOW DOES THE PERCEPTRON LEARN ITS CLASSIFICATION TASKS?

- If at iteration p, the **actual output** is Y (p) and the **desired output** is $Y_d(p)$, then the **error** is given by:

$$e(p) = Y_d(p) - Y(p) \qquad \text{where } p = 1, 2, 3, \ldots$$

Iteration p here refers to the pth training example presented to the perceptron.

If the error, $e(p)$, is positive, we need to increase perceptron output $Y(p)$, but if it is negative, we need to decrease $Y(p)$. Taking into account that each perceptron input contributes $x_i(p) \times w_i(p)$ to the total input $X(p)$, we find that if input value $x_i(p)$ is positive, an increase in its weight $w_i(p)$ tends to increase perceptron output $Y(p)$, whereas if $x_i(p)$ is negative, an increase in $w_i(p)$ tends to decrease $Y(p)$. Thus, the following **perceptron learning rule** can be established:

$$w_i(p+1) = w_i(p) + \alpha \times x_i(p) \times e(p),$$

where α is the **learning rate**, a positive constant less than unity.

PERCEPTRON'S LEARNING ALGORITHM

Step 1: ***Initialisation***

Set initial weights $w_1, w_2, \ldots, w_n$ and threshold θ to random numbers in the range $[-0.5, 0.5]$.

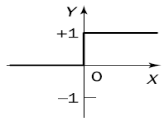
Step 2: ***Activation***

Activate the perceptron by applying inputs $x_1(p), x_2(p), \ldots, x_n(p)$ and desired output $Y_d(p)$. Calculate the actual output at iteration $p = 1$

$$Y(p) = step\left[\sum_{i=1}^{n} x_i(p)w_i(p) - \theta\right], \qquad (6.6)$$

where n is the number of the perceptron inputs, and *step* is a step activation function.

Step function



$$Y^{step} = \begin{cases} 1, \text{ if } X \geq 0 \\ 0, \text{ if } X < 0 \end{cases}$$

PERCEPTRON'S LEARNING ALGORITHM CONT.

Step 3: ***Weight training***

Update the weights of the perceptron

$$w_i(p+1) = w_i(p) + \Delta w_i(p), \tag{6.7}$$

where $\Delta w_i(p)$ is the weight correction at iteration p.

The weight correction is computed by the **delta rule**:

$$\Delta w_i(p) = \alpha \times x_i(p) \times e(p) \tag{6.8}$$

Step 4: ***Iteration***

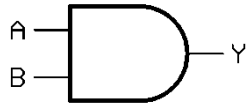
Increase iteration p by one, go back to Step 2 and repeat the process until convergence.

Example

EXAMPLE. X1 AND X2 (BOOLEAN AND)

This example illustrates that representation of the AND function is possible. The AND function, $y = x_1$ AND x_2 should produce the following:

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

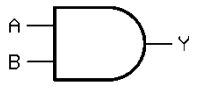


CAN WE TRAIN A PERCEPTRON TO PERFORM BASIC LOGICAL OPERATIONS

Epoch	Inputs x_1	Inputs x_2	Desired output Y_d	Initial weights w_1	Initial weights w_2	Actual output Y	Error e	Final weights w_1	Final weights w_2
1	0	0	0	0.3	−0.1	0	0	0.3	−0.1
	0	1	0	0.3	−0.1	0	0	0.3	−0.1
	1	0	0	0.3	−0.1	1	−1	0.2	−0.1
	1	1	1	0.2	−0.1	0	1	0.3	0.0

Threshold: $\theta = 0.2$; learning rate: $\alpha = 0.1$.

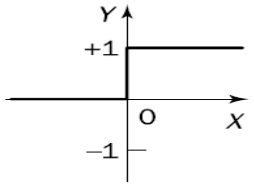
$$Y(p) = step\left[\sum_{i=1}^{n} x_i(p)w_i(p) - \theta\right]$$



$$e(p) = Y_d(p) - Y(p)$$

(0*0.3+0*-0.1)-0.2=-0.2==>Y=0
(0*0.3+1*-0.1)-0.2=-0.3==> Y=0
(1*0.3+0*-0.1)-0.2=0.1==>Y=1
(1*0.2+1*-0.1)-0.2=-0.1==>Y=0

Step function



$$Y^{step} = \begin{cases} 1, \text{ if } X \geq 0 \\ 0, \text{ if } X < 0 \end{cases}$$

- the sequence of four input patterns representing an **epoch**.

$$w_i(p+1) = w_i(p) + \alpha \times x_i(p) \times e(p)$$

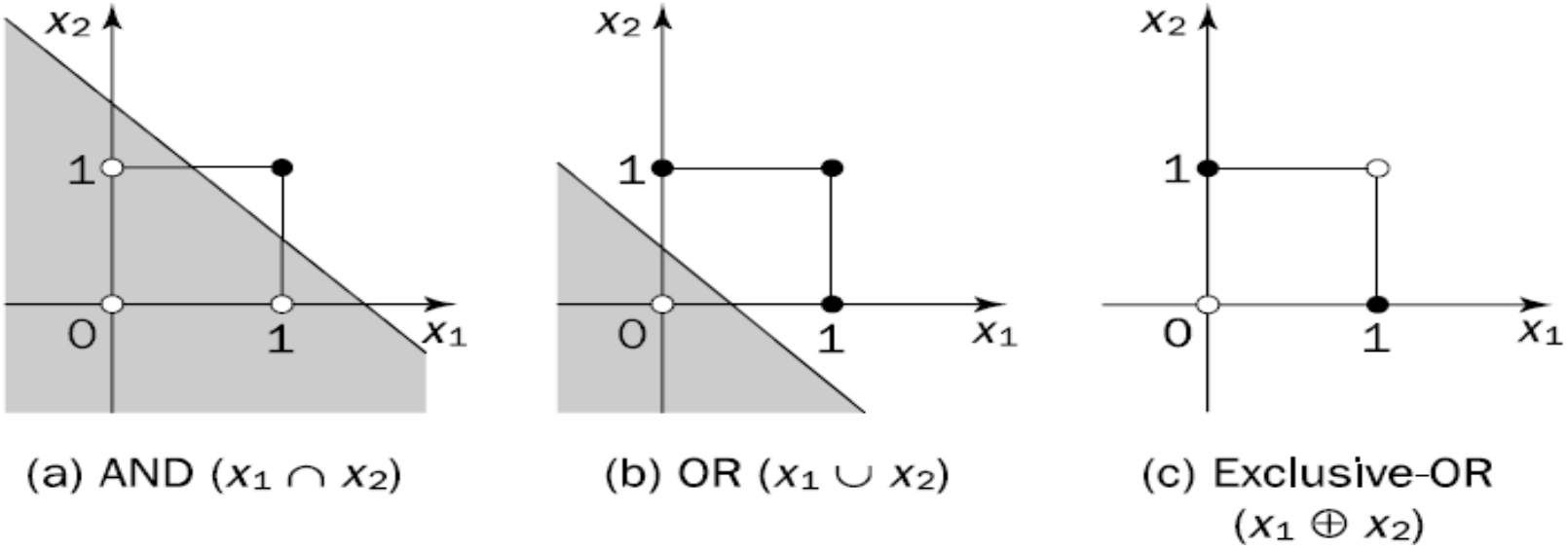
W1=0.3+0.1*-1*1=0.2
W2=w2=-0.1

CAN WE TRAIN A PERCEPTRON TO PERFORM BASIC LOGICAL OPERATIONS

Epoch	Inputs x_1	Inputs x_2	Desired output Y_d	Initial weights w_1	Initial weights w_2	Actual output Y	Error e	Final weights w_1	Final weights w_2
1	0	0	0	0.3	−0.1	0	0	0.3	−0.1
	0	1	0	0.3	−0.1	0	0	0.3	−0.1
	1	0	0	0.3	−0.1	1	−1	0.2	−0.1
	1	1	1	0.2	−0.1	0	1	0.3	0.0
2	0	0	0	0.3	0.0	0	0	0.3	0.0
	0	1	0	0.3	0.0	0	0	0.3	0.0
	1	0	0	0.3	0.0	1	−1	0.2	0.0
	1	1	1	0.2	0.0	1	0	0.2	0.0
3	0	0	0	0.2	0.0	0	0	0.2	0.0
	0	1	0	0.2	0.0	0	0	0.2	0.0
	1	0	0	0.2	0.0	1	−1	0.1	0.0
	1	1	1	0.1	0.0	0	1	0.2	0.1
4	0	0	0	0.2	0.1	0	0	0.2	0.1
	0	1	0	0.2	0.1	0	0	0.2	0.1
	1	0	0	0.2	0.1	1	−1	0.1	0.1
	1	1	1	0.1	0.1	1	0	0.1	0.1
5	0	0	0	0.1	0.1	0	0	0.1	0.1
	0	1	0	0.1	0.1	0	0	0.1	0.1
	1	0	0	0.1	0.1	0	0	0.1	0.1
	1	1	1	0.1	0.1	1	0	0.1	0.1

Threshold: $\theta = 0.2$; learning rate: $\alpha = 0.1$.

CAN WE TRAIN A PERCEPTRON TO PERFORM BASIC LOGICAL OPERATIONS



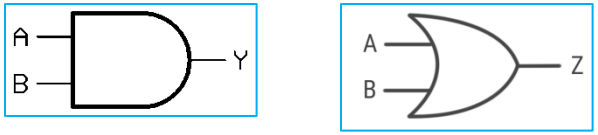
Two-dimensional plots of basic logical operations

- Points in the input space where the function output is 1 are indicated by black dots, and points where the output is 0 are indicated by white dots.

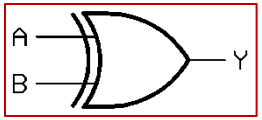
CAN WE TRAIN A PERCEPTRON TO PERFORM BASIC LOGICAL OPERATIONS

- the perceptron can learn the operation OR. However, a single-layer perceptron cannot be trained to perform the operation Exclusive OR.

- A perceptron is able to represent a function only if there is some line that separates all the black dots from all the white dots. Such functions are called linearly separable. Therefore, a perceptron can learn the operations AND and OR, but not Exclusive-OR.



linear

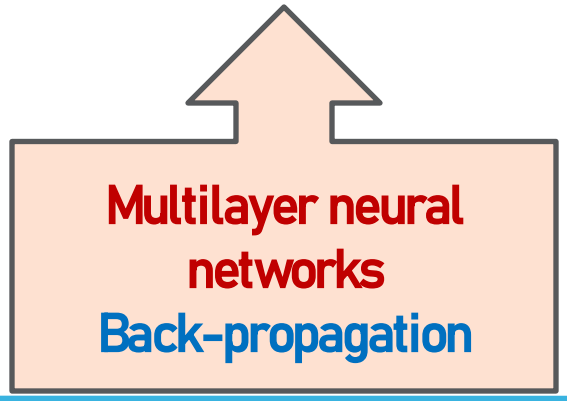


Non- linear

CAN WE TRAIN A PERCEPTRON TO PERFORM BASIC LOGICAL OPERATIONS

How do we cope with problems which are not linearly separable?

To cope with such problems we need multilayer neural networks. In fact, history has proved that the limitations of Rosenblatt's perceptron can be overcome by advanced forms of neural networks, for example multilayer perceptrons trained with the back-propagation algorithm.



AGENDA

Biological Background
Artificial Neurons
Activation Functions
A Simple Perceptron
Neural Network Characteristics

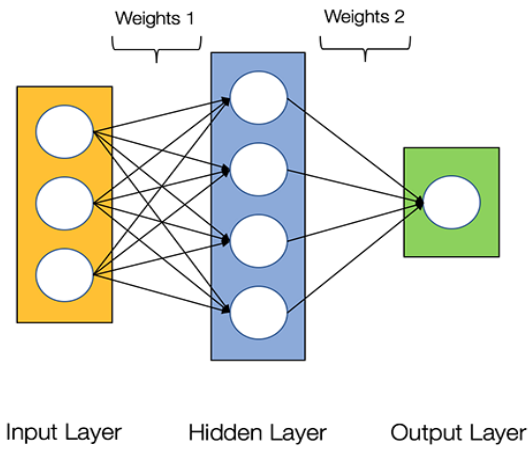
NEURAL NETWORK CHARACTERISTICS

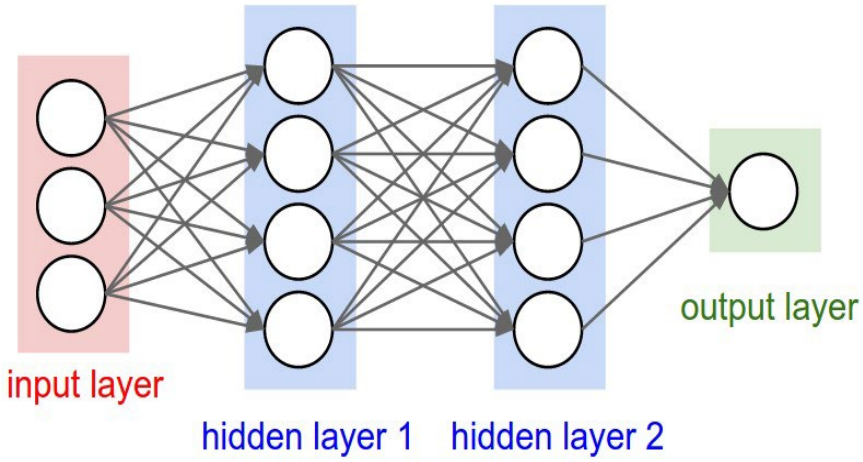
- There are various characteristics associated with the neural networks, and by selecting different characteristic values, we can come up many different models.

TOPOLOGY OF THE NETWORK ARCHITECTURE

❑ Multilayered

The backpropagation model has distinct layers such as input, hidden, and output. The neurons within each layer are connected with the neurons of the adjacent layers through directed edges. There are no connections among the neurons within the same layer.

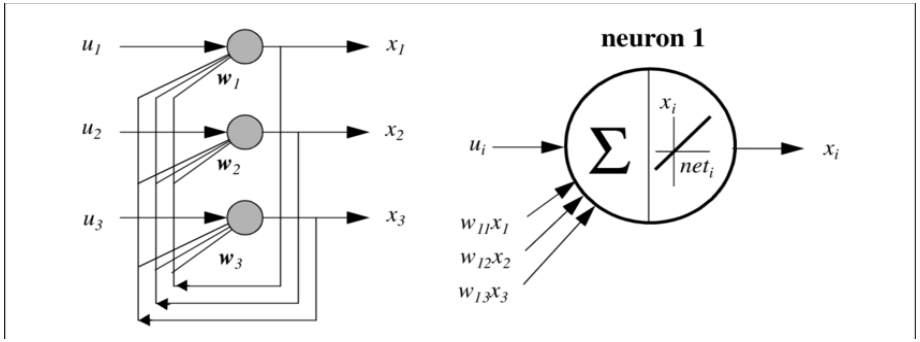




TOPOLOGY OF THE NETWORK ARCHITECTURE

Non-multilayered

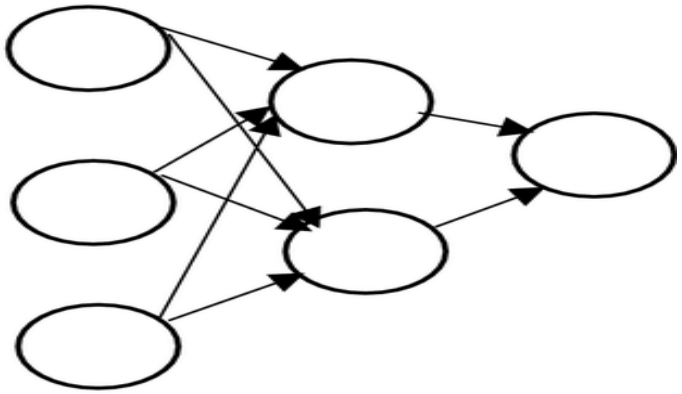
- We can also build neural network without such distinct layers as input, output, or hidden. Every neuron can be connected with every other neuron in the network through directed edges. Every neuron may input as well as output. A typical example is the Hopfield model.



DIRECTIONS OF OUTPUT

- **Non-recurrent (feedforward only)**

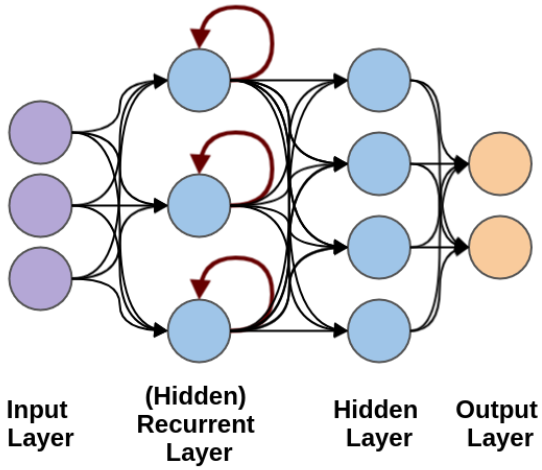
Neural network models with feedforward only are called non-recurrent. Incidentally, "backpropagation" in the backpropagation model should not be confused with feedbackward. The backpropagation is backward adjustments of the weights, not output movements from neurons.

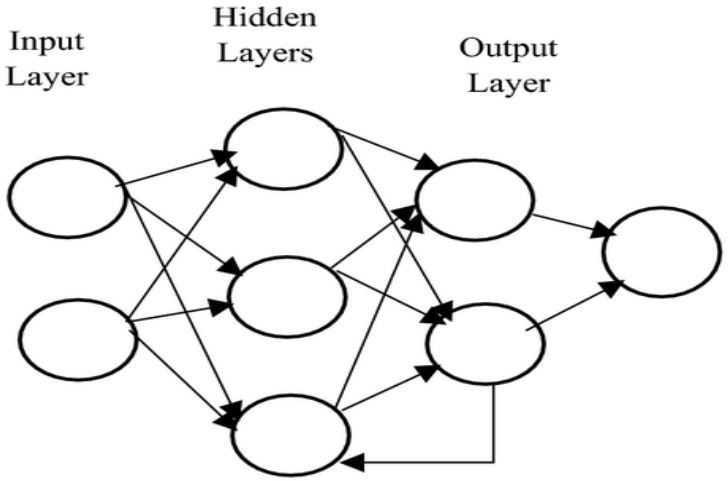


DIRECTIONS OF OUTPUT

- **Recurrent (feedforward and feedbackward)**

The outputs can also propagate backward (from right to left). This is called feedbackward. A neural network in which the outputs can propagate in both directions, forward and backward, is called a recurrent model.

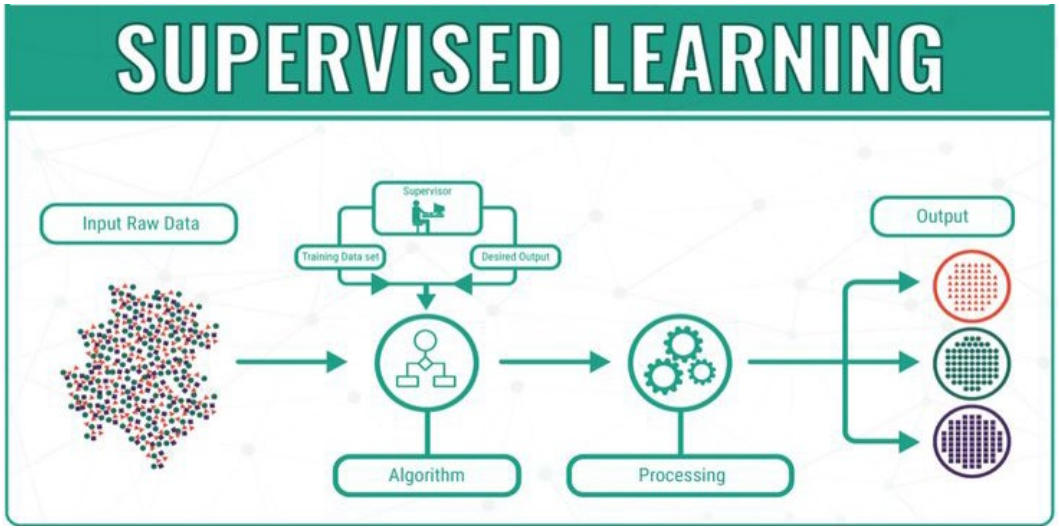




FORM OF LEARNING

❑ Supervised Learning

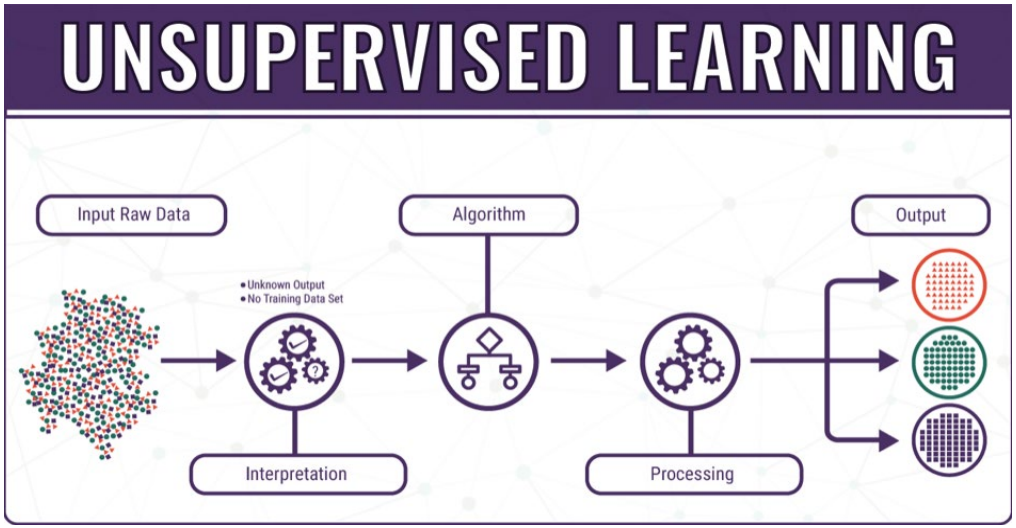
In supervised learning, a teacher knows what should be the correct output and this information is given to the neural network. The backpropagation model is such an example, assuming an existence of a teacher who knows what are correct patterns.



FORM OF LEARNING

❑ Unsupervised Learning

In some models, neural networks can learn by themselves after being given some form of general guidelines. The neural network adjusts by itself internally using certain criteria or algorithms.



TYPES OF INPUT VALUES

We can assume different types of input values. The most common types are:

- ✓ **Binary**: an input value is restricted to either 0 or 1.
- ✓ **Bipolar**: an input value is either -1 or 1.
- ✓ **Continuous**: Continuous real numbers in a certain range.


Thank You!
Any questions?