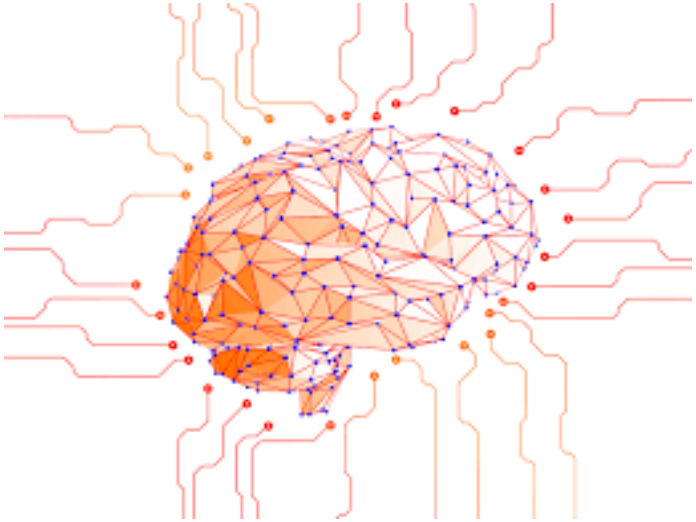
Introduction to Neural Networks and Perceptron

Machine intelligence

Dr. Ahmad Al-Mahasneh



Review of the last lecture

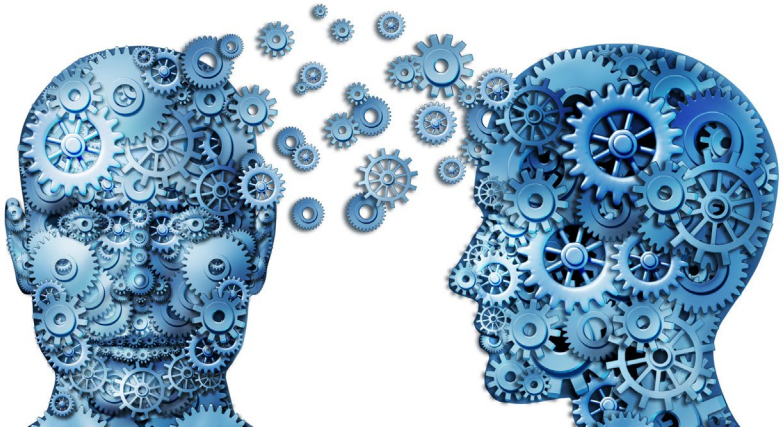
- AI methods and techniques
- AI Applications

Outline

- Human brain and neural networks
- Artificial neural networks properties, elements, activation functions and architecture.
- Learning types and tasks.
- Perceptron, OR, AND and XOR problems

Human Brain

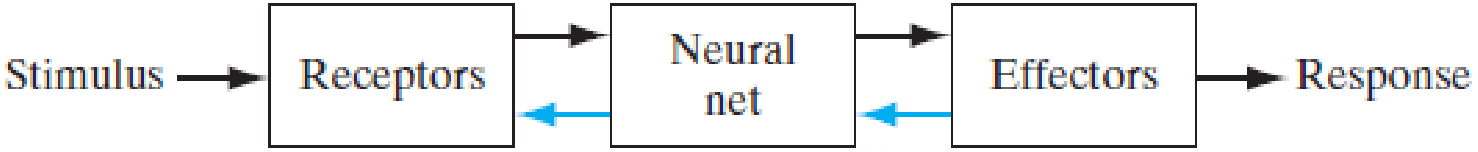
- The brain is a highly complex, nonlinear, parallel, and fast processor.

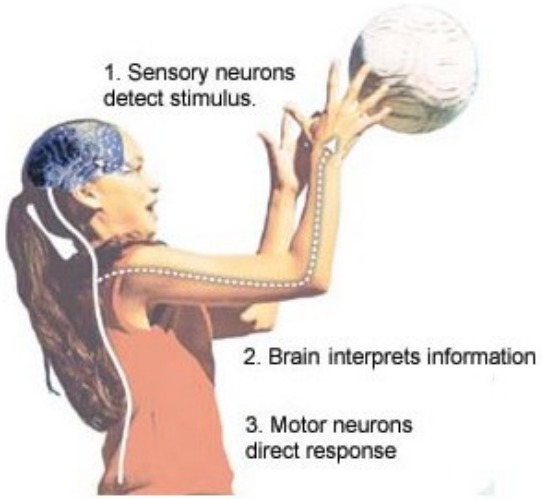


- How do humans learn ?
  - Through Experience over time by observing the environment and interacting with it (**reinforcement learning**) and learning from their parents (**supervised**)

Human Brain

It performs certain computations (e.g., pattern recognition, perception, and motor control) many times faster than the fastest digital computer in existence today.





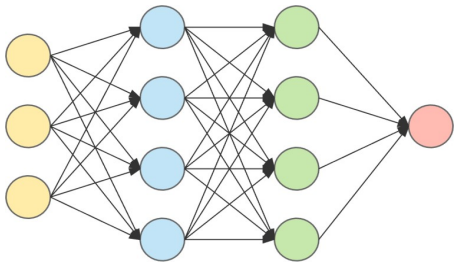
Artificial Neural Network(ANN)

- ANN is a massively parallel distributed processor made up of simple processing units that has the capability to store experiential knowledge and make it available for later use.

ANN resemble human brain in:

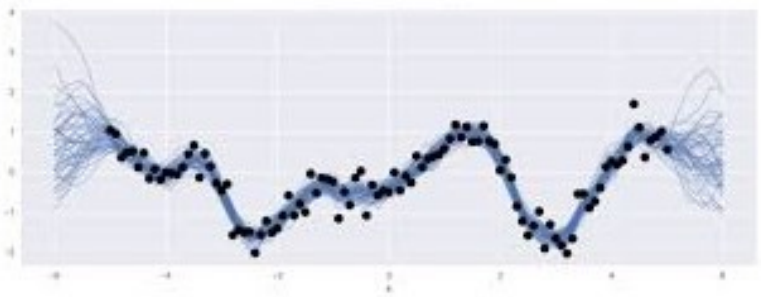
- 1. Knowledge is acquired by the network from its environment through a learning process.

- 2. Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.

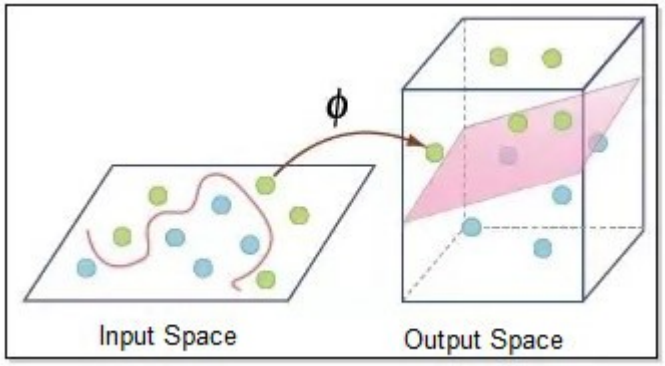


Neural Network Properties

- Nonlinearity: as it is nonlinear, it enable us to capture nonlinearities in the training data.



- Input-Output Mapping: building a relationship between the inputs and outputs.

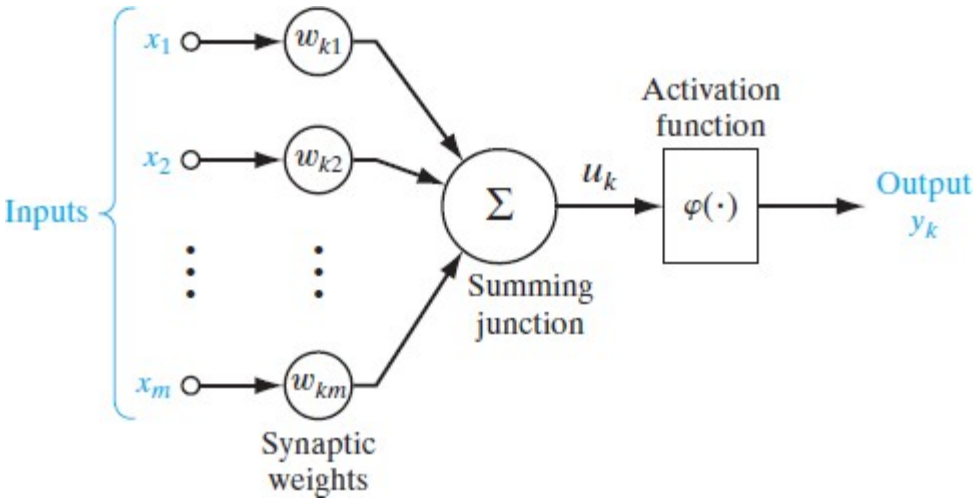


- Adaptivity: the ability to learn(adapt their parameters).

- Fault-tolerance: even if parts of the network failed, it will still produce a reasonable output.

Neuron Elements

- Synaptic weights
- Summing junctions (adders)
- Activation Functions

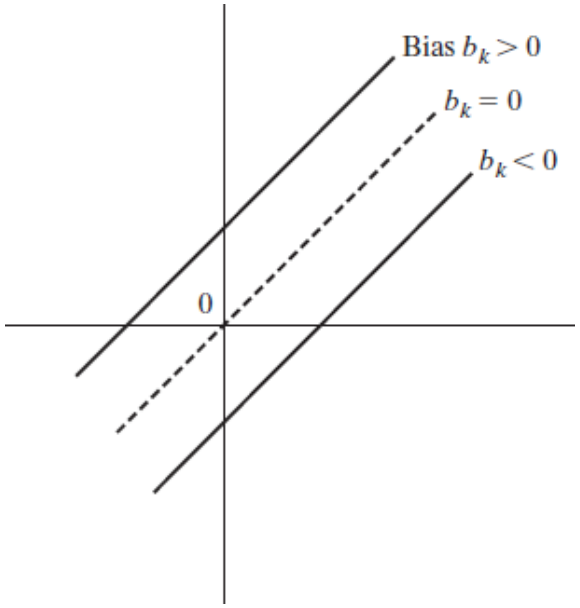


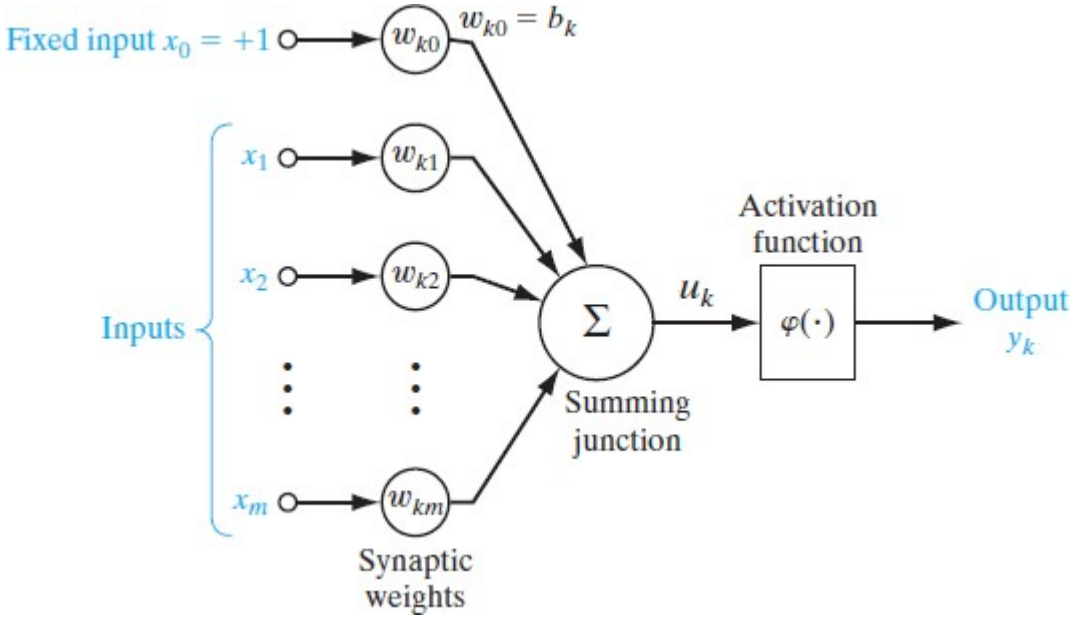
$$u_k = \sum_{j=1}^{m} w_{kj} x_j$$

$$y_k = \varphi(u_k)$$

Neuron Elements

- Weights
- Summing junction
- Activation Function

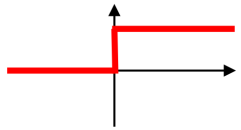
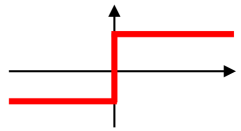
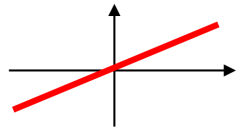
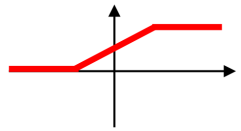
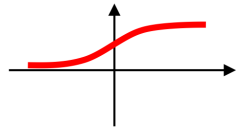
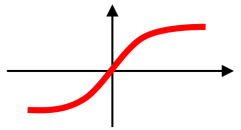
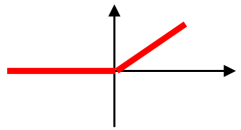
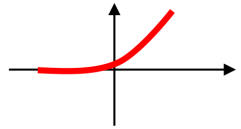




$$u_k = \sum_{j=0}^{m} w_{kj} x_j$$

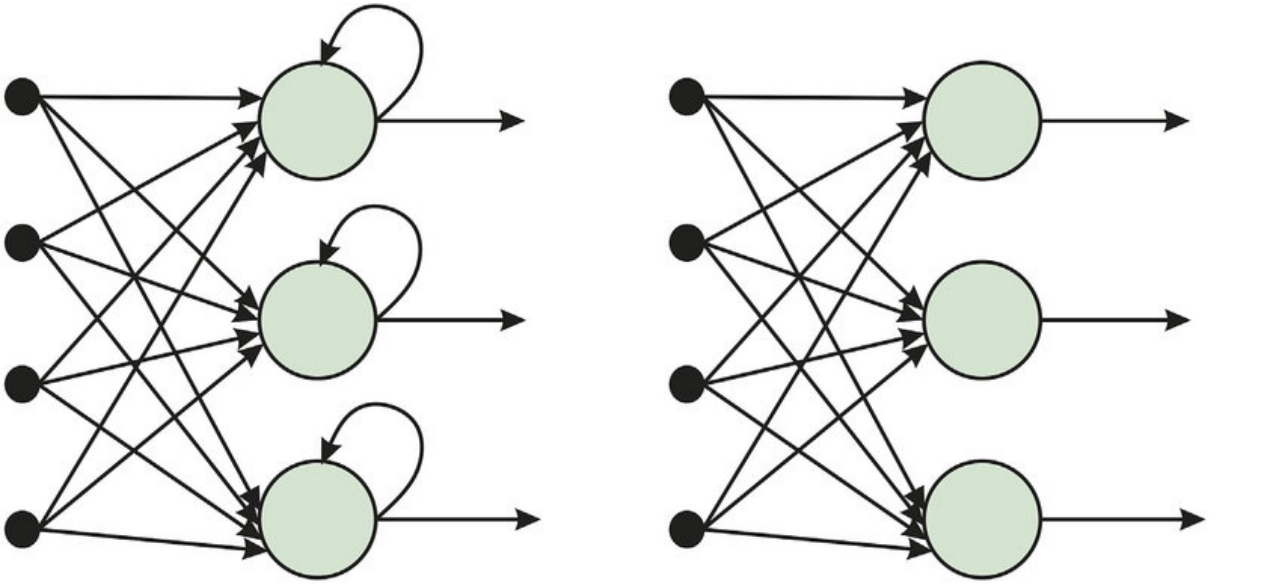
$$y_k = \varphi(u_k)$$

Activation Functions

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^{z})$	Multi-layer Neural Networks	

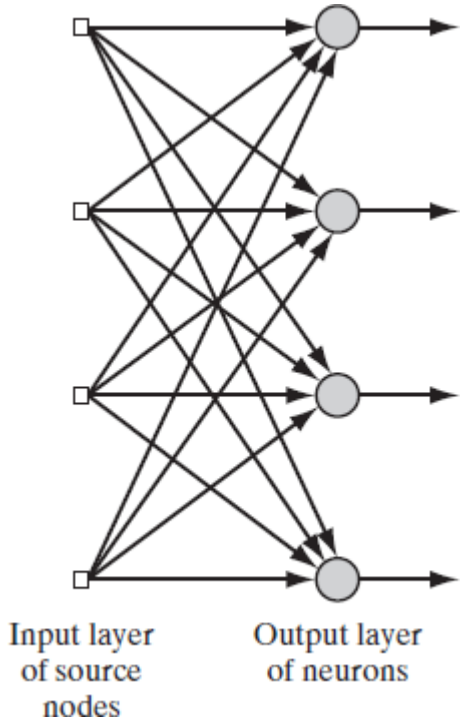
Network Architectures

- Sing layer neural networks
- Multi-layer feedforward neural networks
- Recurrent neural networks

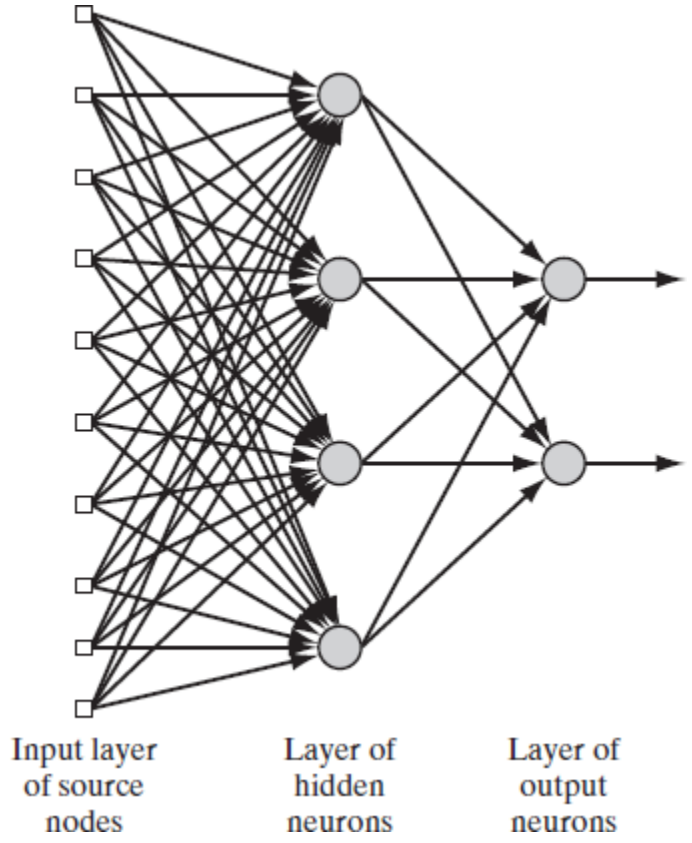


(a) Recurrent Neural Network (b) Feed-Forward Neural Network

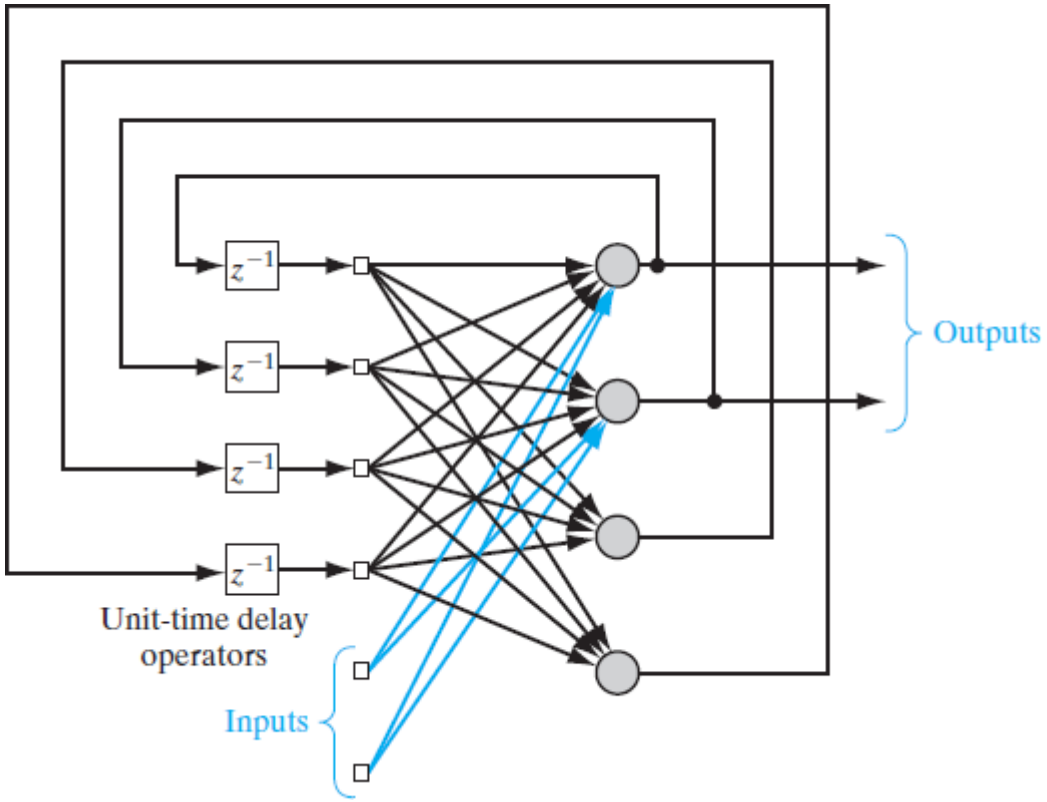
Architecture: Single Layer



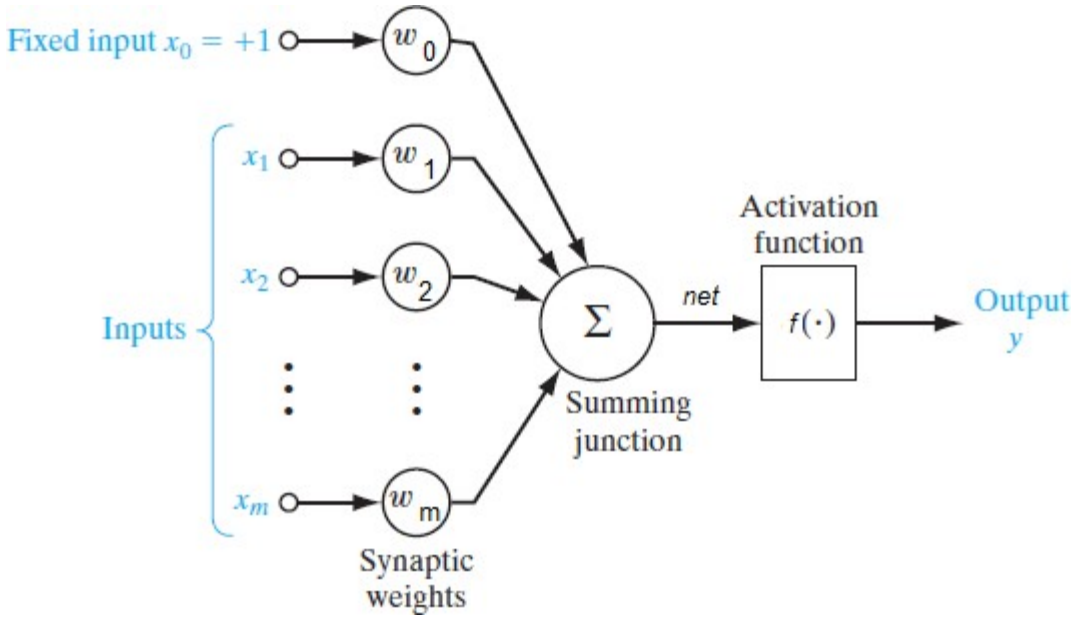
Architecture: Multilayer Feedforward Network



Architecture: Recurrent Network



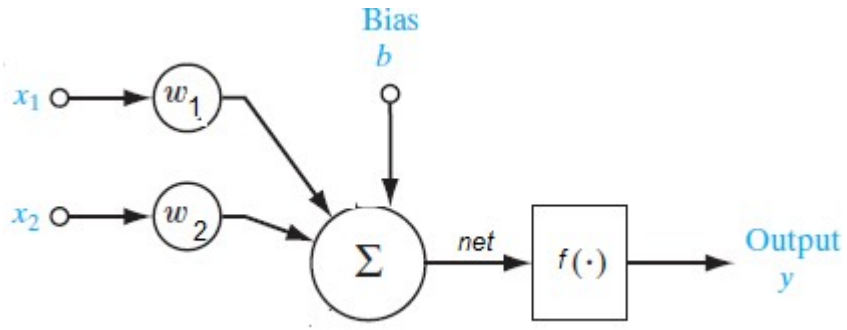
Perceptron



$$net = \sum_{\{m\}}^{\{i=1\}} x_i w_i$$

$$y = f(net)$$

Perceptron with two inputs

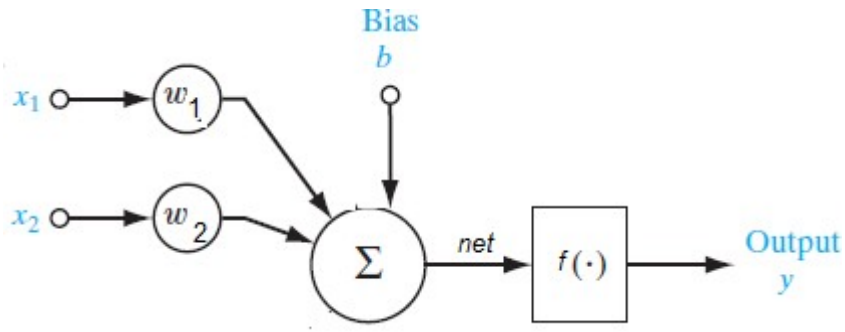


$$net = w_1 x_1 + x_2 w_2 + b$$

using a threshold activation function

$$y = f(net) = \begin{cases} 0 & if\ net < 0 \\ 1 & if\ net \geq 0 \end{cases}$$

Perceptron: OR Problem



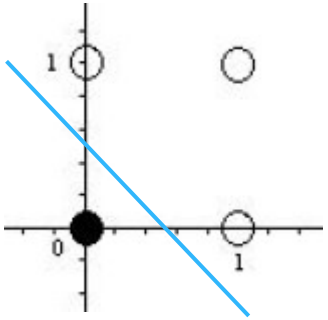
x2	x1	y
0	0	0
0	1	1
1	0	1
1	1	1

$$net = w_1x_1 + x_2w_2 + b$$

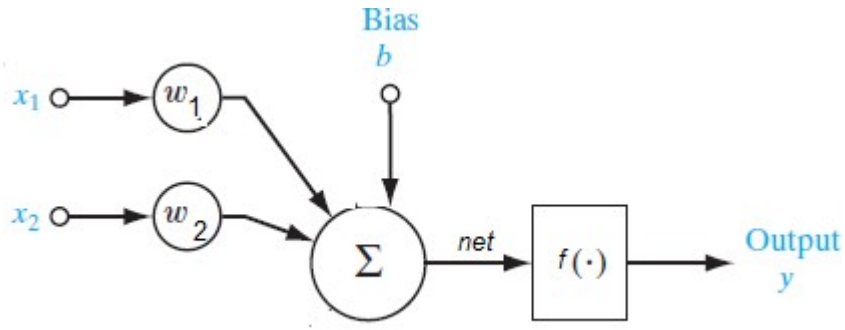
Let w1 = w2 = 1 and b = -0.5

$$\Rightarrow net = x_1 + x_2 - 0.5$$

$$y = \begin{cases} 0 & x_1 + x_2 - 0.5 < 0 \\ 1 & x_1 + x_2 - 0.5 \geq 0 \end{cases}$$



Perceptron: AND Problem



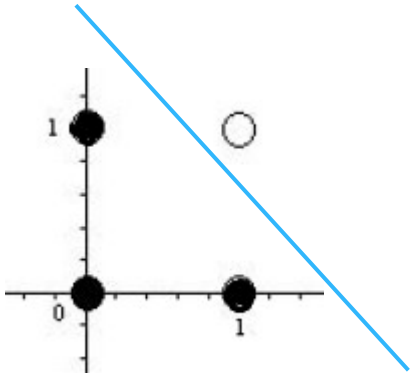
x2	x1	y
0	0	0
0	1	0
1	0	0
1	1	1

$$net = w_1 x_1 + x_2 w_2 + b$$

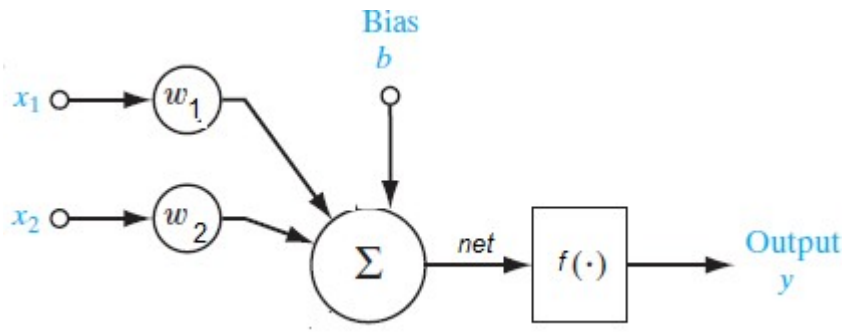
Let w1 = w2 = 1 and b = -1.5

$$\Rightarrow net = x_1 + x_2 - 1.5$$

$$y = \begin{cases} 0 & x_1 + x_2 - 1.5 < 0 \\ 1 & x_1 + x_2 - 1.5 \geq 0 \end{cases}$$

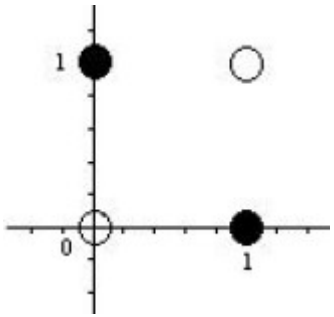


Perceptron: XOR Problem

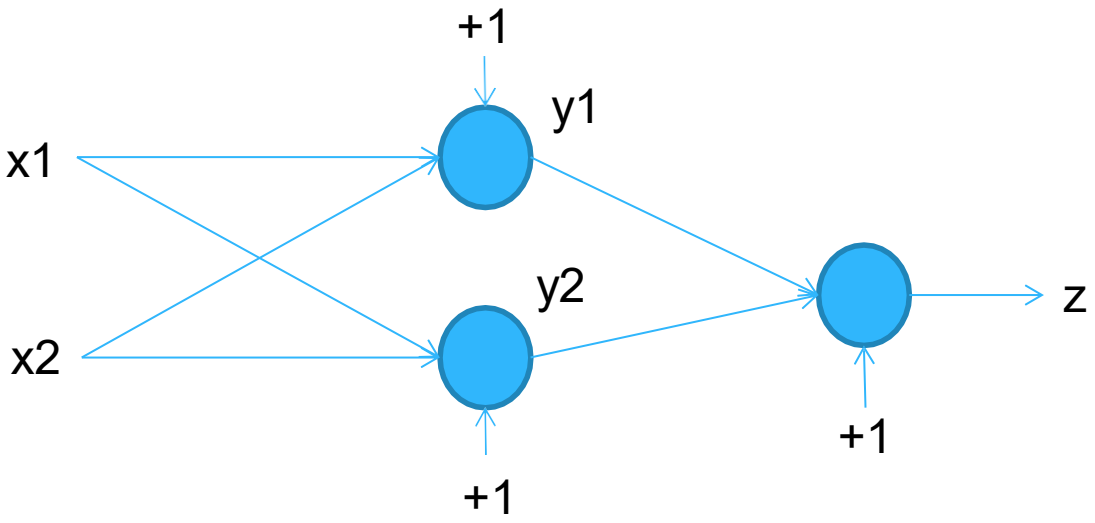


x2	x1	y
0	0	0
0	1	1
1	0	1
1	1	0

Can we separate the classes using one line ?



Perceptron: XOR Problem

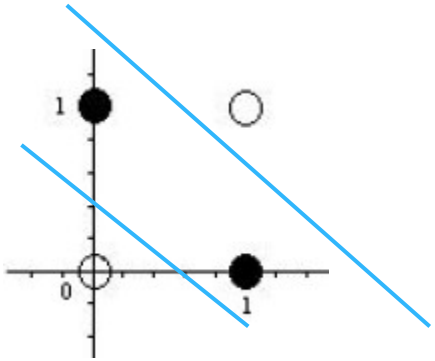


x1	x2	y1	y2	z
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

$$y_1 = \begin{cases} 0 & x_1 + x_2 - 0.5 < 0 \\ 1 & x_1 + x_2 - 0.5 \geq 0 \end{cases}$$

$$y_2 = \begin{cases} 0 & x_1 + x_2 - 1.5 < 0 \\ 1 & x_1 + x_2 - 1.5 \geq 0 \end{cases}$$

$$z = \begin{cases} 0 & y_1 - y_2 - 0.1 < 0 \\ 1 & y_1 - y_2 - 0.1 \geq 0 \end{cases}$$



Conclusions

- Neurons are composed of Weights, Summations and Activation Functions.
- Activation Functions can be Threshold, Sigmoidal, Hyperbolic Tangent and Linear.
- Network Architectures are Feedforward and Recurrent

Conclusions

- A Perceptron is a single neuron
- A Perceptron can be used to classify data if they can be separated by a single line (e.g. AND / OR problems)
- Hidden nodes are needed to classify the XOR problem
- More layers increase the computational power of neural networks.

Reference

1. Neural Networks and Learning Machine (Chapter One) by Simon Haykin 3rd Edition. Pearson 2009