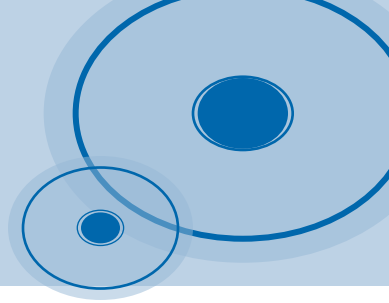




# **Artificial Intelligence**

Artificial Neural Network


# Artificial neural networks



- Also known as
  - Neural Networks
  - Neural Net
  - ANN

# Aims



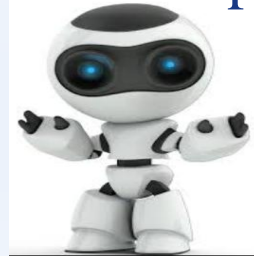
- Introduction, or how the brain works
  - The neuron as a simple computing element
  - The perceptron
  - Multilayer neural networks
- 

# Introduction, or how the brain works

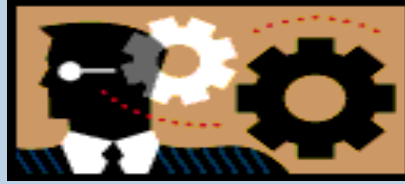
- Machine learning involves adaptive mechanisms that enable computers to:  
learn by data example  
learn by data analogy



learn from data experience



# Brain

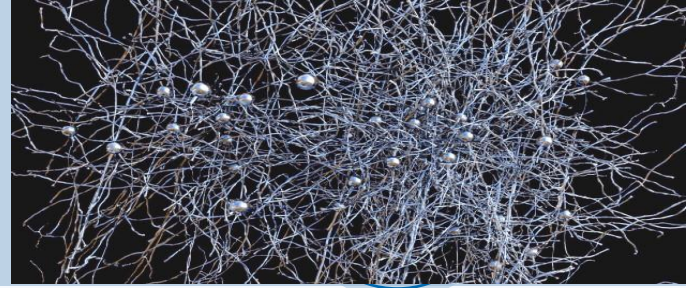


- Reasoning based on the human brain. The brain consists of **a densely interconnected set of neurons**.



MakeAGIF.com

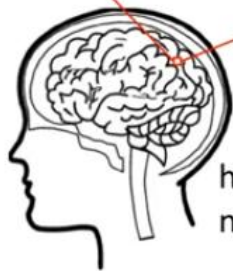
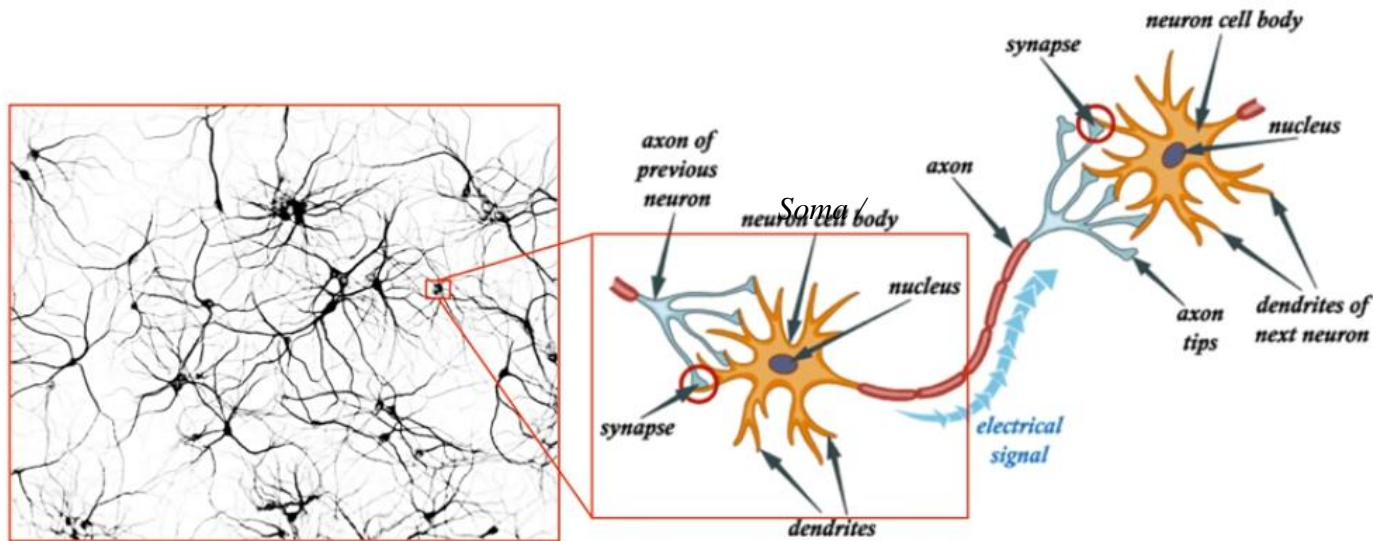
# Neural network



- The human brain incorporates nearly 100 billion neurons and 60 trillion connections, synapses, between them.



# Neurons and the brain

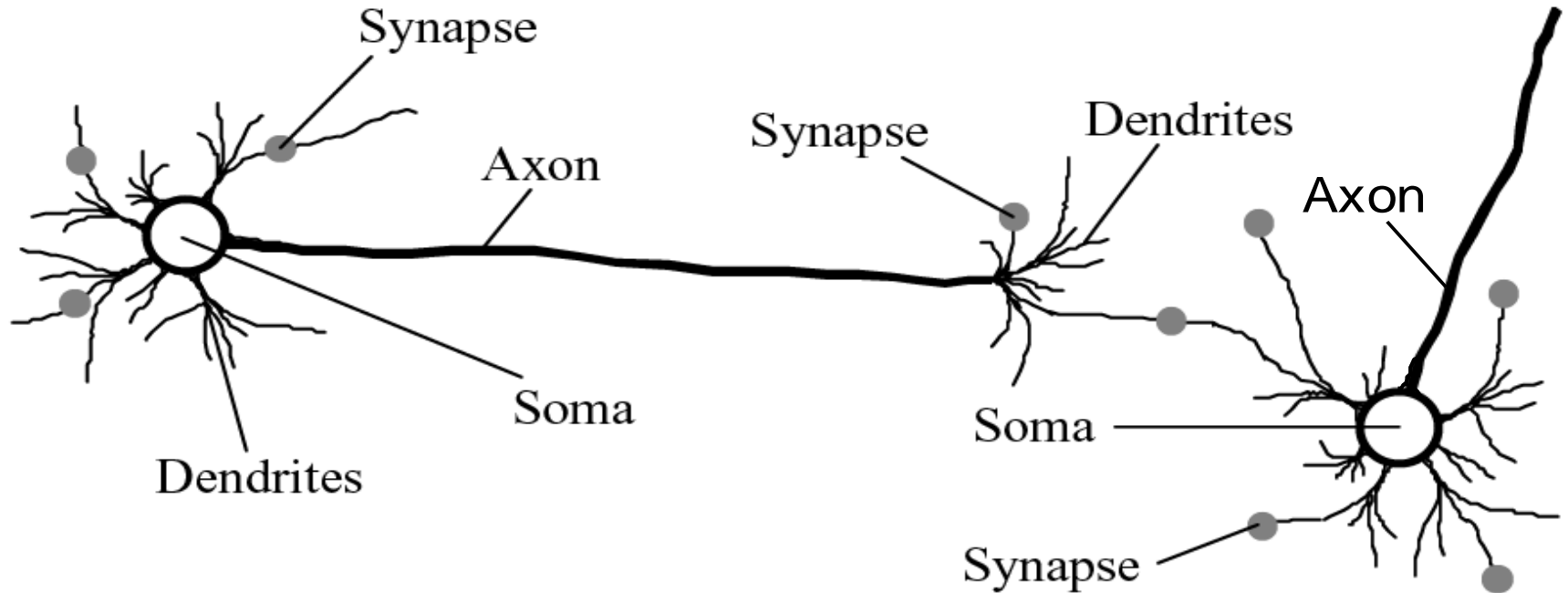


humans don't  
need features





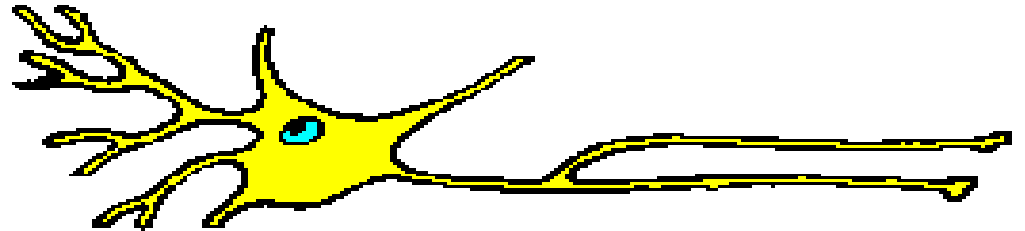
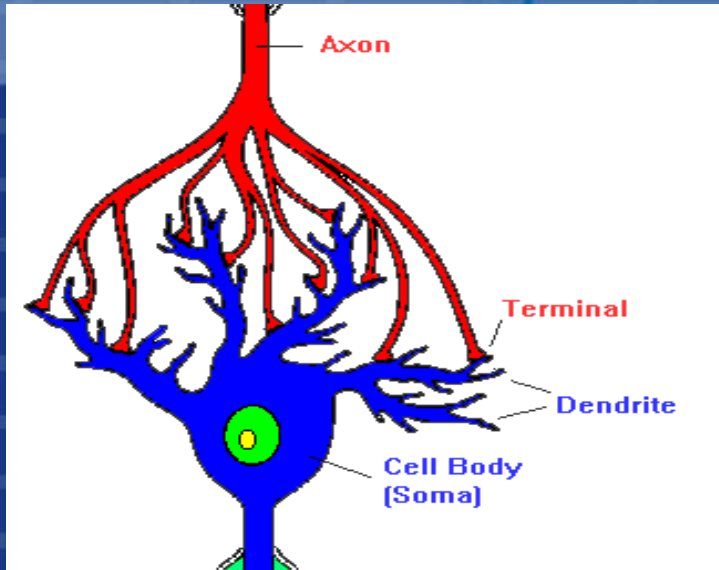
# Biological neural network



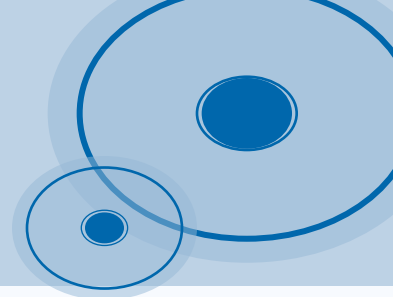


# Communication between neurons

when an electrical impulse travels down an axon to the synaptic terminal.



# BNNs Functions



Soma	The basic cell body
Dendrite	Receive signals from axon
Axon	Propagate signals from one neuron to another
Synapse	Release chemical substance to cause the change in the electrical potential of the cell body

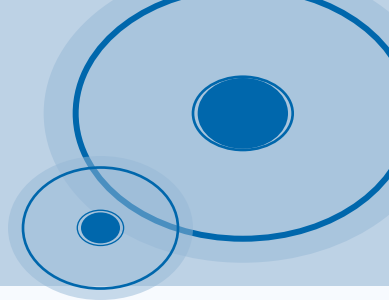
# Synapse



The University of California, San Francisco's Roger Anguera created this gorgeous visualization showing how brain signals travel through and around the brain. Each color represents source power and connectivity in a different frequency band, i.e. theta, alpha, beta, and gamma waves

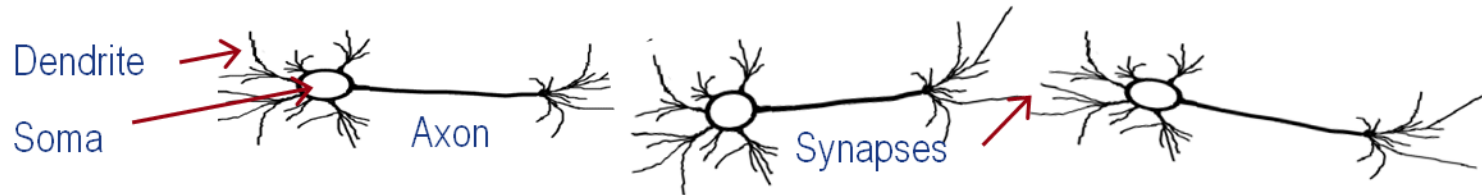
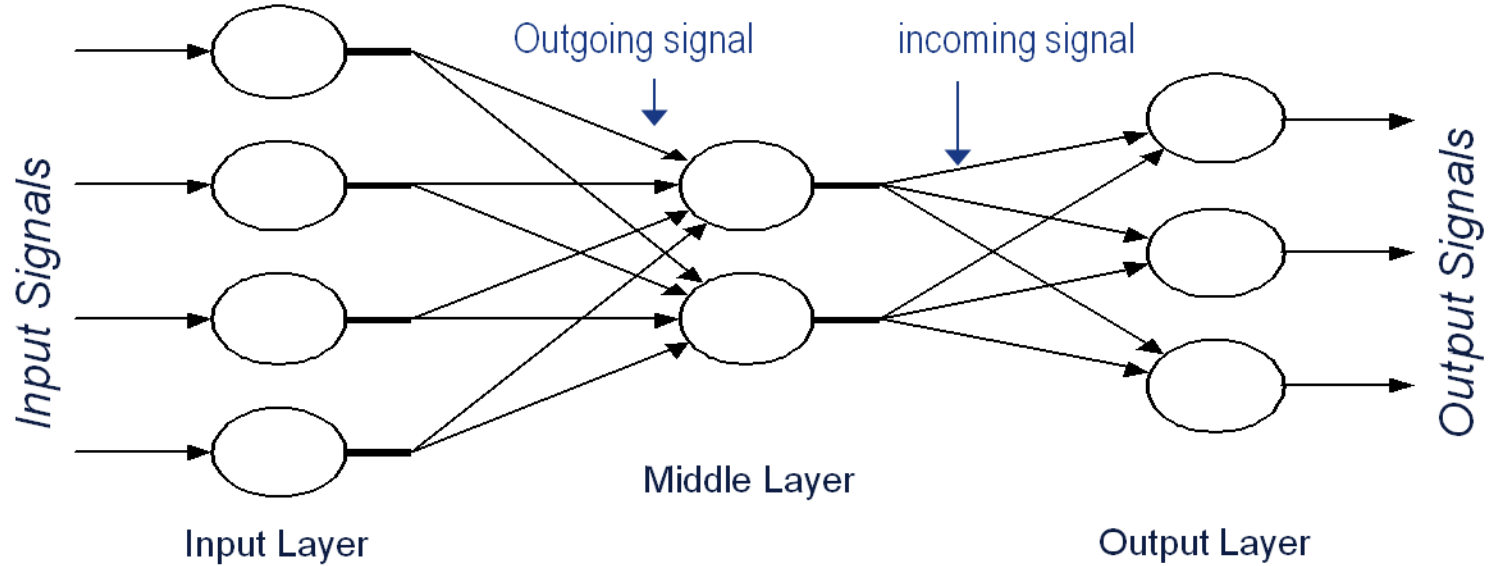


# Artificial Neural Network

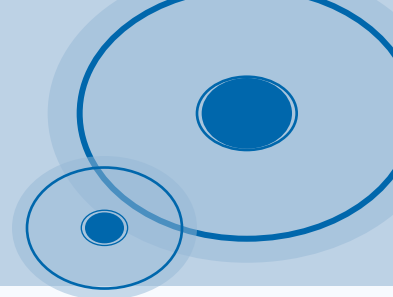


- An artificial neural network consists of a number of very simple processors, also called **neurons**, which are analogous to the biological neurons in the brain.
- The neurons are connected by **weighted links** passing signals from one neuron to another.

# Architecture of a typical artificial neural network



# Analogy between biological and artificial neural networks

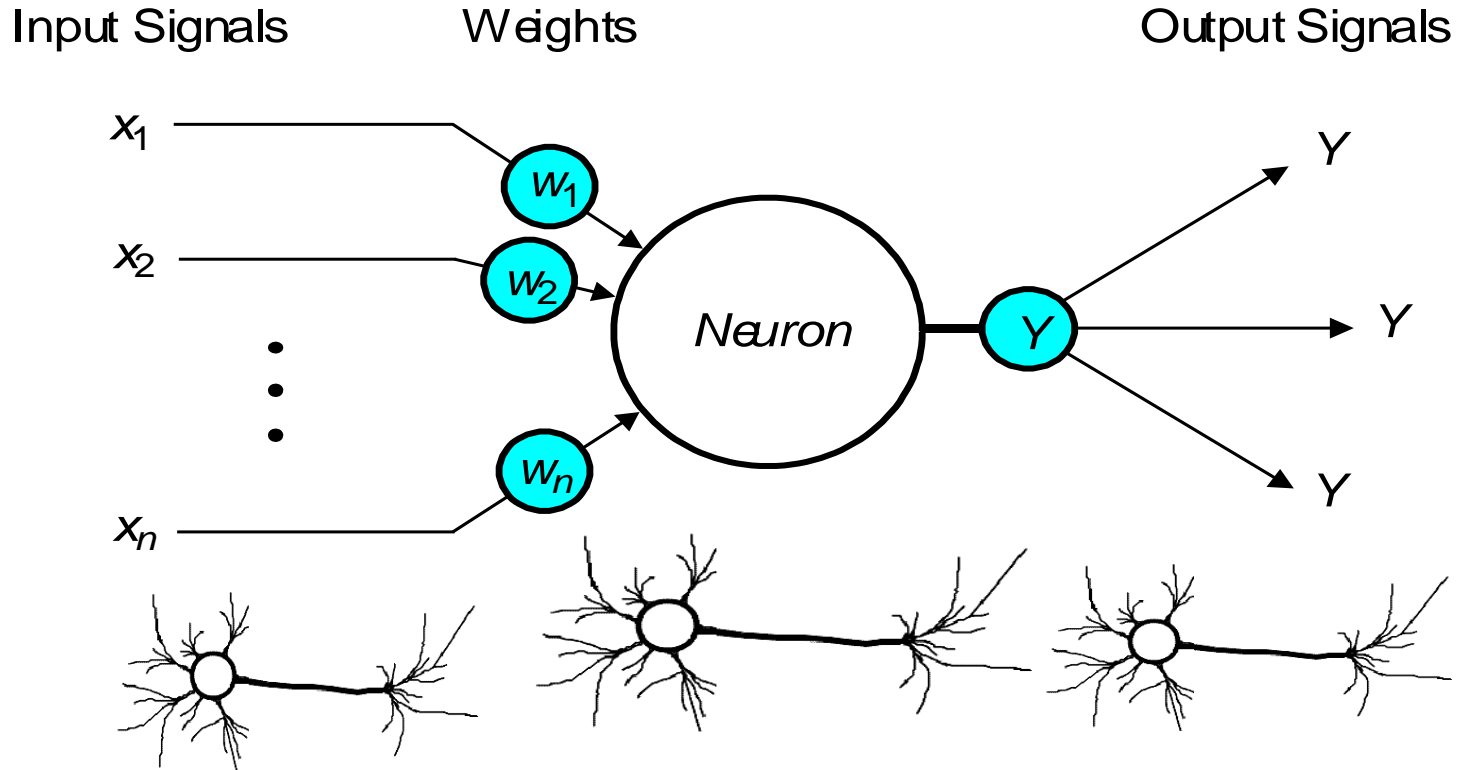


BNN	ANN	Functions
Soma	Neuron	Information processing
Dendrite	Input	Receive data
Axon	Output	Transmit data to another neuron
Synapse	Weight	To express the strength or the importance of neuron input



# The neuron as a simple computing element

## Diagram of a neuron



# Neuron: How it works?

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

$$X = \sum_{i=1}^n x_i w_i$$
$$Y = \begin{cases} +1, & \text{if } X \geq \theta \\ -1, & \text{if } X < \theta \end{cases}$$

input      weight      output      threshold

A weight represents the strength of the connection between units.

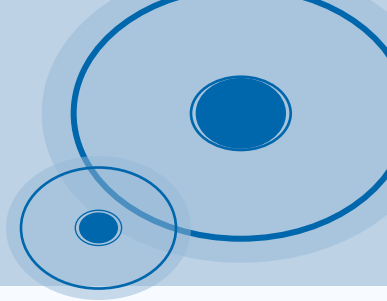
# Activation function

- The neuron uses the following transfer or **activation function**:
- This type of activation function is called a **sign function (hard limiter)**.

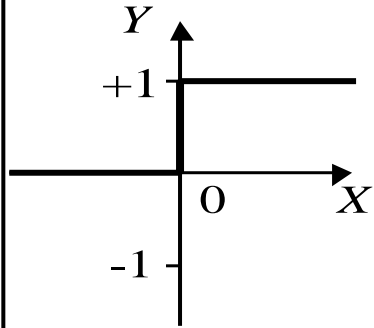
$$X = \sum_{i=1}^n x_i w_i \qquad Y = \begin{cases} +1, & \text{if } X \geq \theta \\ -1, & \text{if } X < \theta \end{cases}$$

Squashes the values into a smaller range

# Activation functions of a neuron

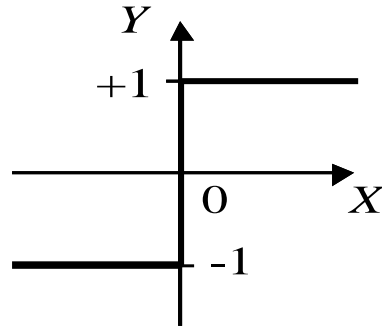


*Step function*



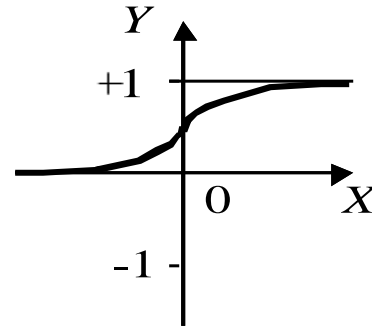
$$Y^{step} = \begin{cases} 1, & \text{if } X \geq 0 \\ 0, & \text{if } X < 0 \end{cases}$$

*Sign function*



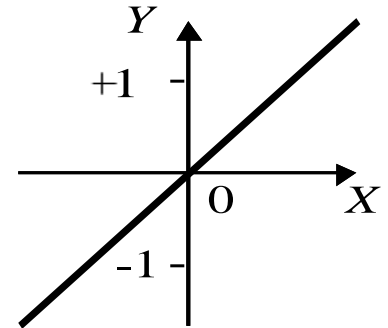
$$Y^{sign} = \begin{cases} +1, & \text{if } X \geq 0 \\ -1, & \text{if } X < 0 \end{cases}$$

*Sigmoid function*



$$Y^{sigmoid} = \frac{1}{1 + e^{-X}}$$

*Linear function*



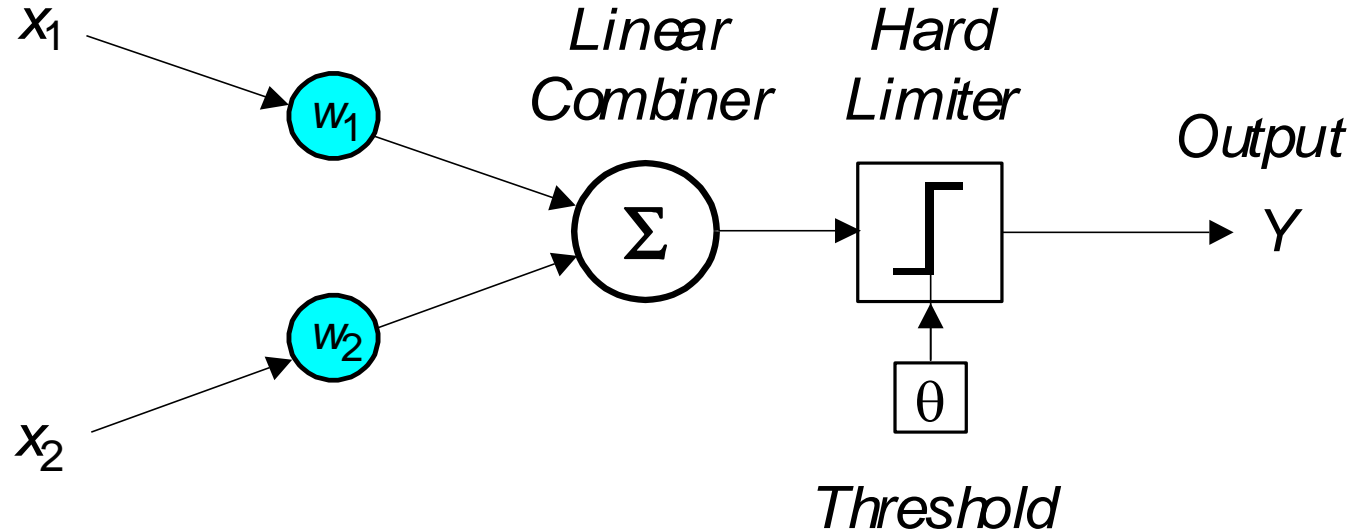
$$Y^{linear} = X$$

# single neuron (simplest form): **Perceptron**

- Frank Rosenblatt, 1958
- based on the McCulloch and Pitts neuron model
- It consists of
  - A single neuron
  - adjustable synaptic weights
  - a hard limiter (sign function).

# Single-layer perceptron (with 2 inputs)

*Inputs*



# Perceptron's training algorithm

- **Step 1: Initialization**

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



# Perceptron's training algorithm

- **Step 2: Activation**

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

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

- where  $n$  is the number of the perceptron inputs, and  $\text{step}$  is a step activation function

# Perceptron's training algorithm

- **Step 3: Weight training**

Update the weights of the perceptron

$$w_i(p+1) = w_i(p) + \Delta w_i(p)$$

- where  $\Delta w_i(p)$  is the weight correction at iteration  $p$ .
- The weight correction is computed by the delta rule:

$$\Delta w_i(p) = \alpha \cdot x_i(p) \cdot e(p)$$

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

## $\alpha$ and $e(p)$

- learning rate,  $\alpha = 0.1$
- Error,  $e(p) = Yd(p) - Y(p)$

# Perceptron's training algorithm

- **Step 4: Iteration**

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

# Example of perceptron learning: the logical operation AND

First round  
of training  
(Epoch 1)

Epoch	Inputs		Desired output $Y_d$	Initial weights		Predicted Output	Error $e$	Final weights	
	$x_1$	$x_2$		$w_1$	$w_2$			$w_1$	$w_2$
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			1	-1	0.2	-0.1
	1	1	1			0	1	0.3	0.0

Random  
[-0.5, 0.5]

$$\theta = 0.2$$

$$\alpha = 0.1$$

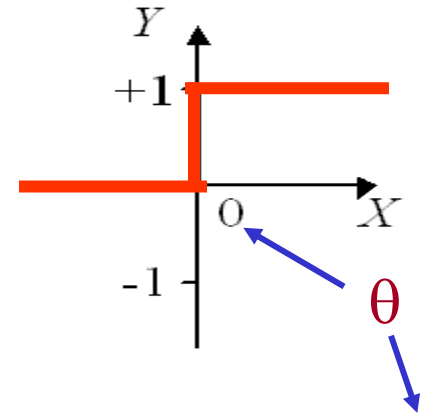
**Step 1**  $x = 0 \times 0.3 + 0 \times (-0.1) = 0$

**Step 2**  $x < \theta \quad 0 < 0.2 \rightarrow Y = 0$

**Step 3**  $w_i(p+1) = w_i(p) + \Delta w_i(p)$

where  $\Delta w_i(p) = \alpha \cdot x_i(p) \cdot e(p)$

*Step function*



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

Epoch	Inputs		Desired output $Y_d$	Initial weights		Predicted Output $\hat{Y}$	Error $e$	Final weights	
	$x_1$	$x_2$		$w_1$	$w_2$			$w_1$	$w_2$
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:  $\alpha = 0.1$

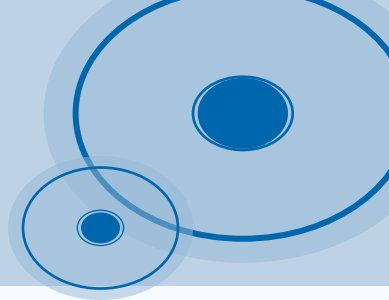
The weights are the same compared to previous round

# ANN algorithm

- Supervised learning
  - Single layer NN
    - Perceptron
  - Multilayer NN
    - Back-propagation
    - Accelerated learning
    - The Hopfield network
    - Bidirectional Associative memory
- Unsupervised learning
  - Hebbian Learning
  - Competitive Learning



# Multilayer neural networks



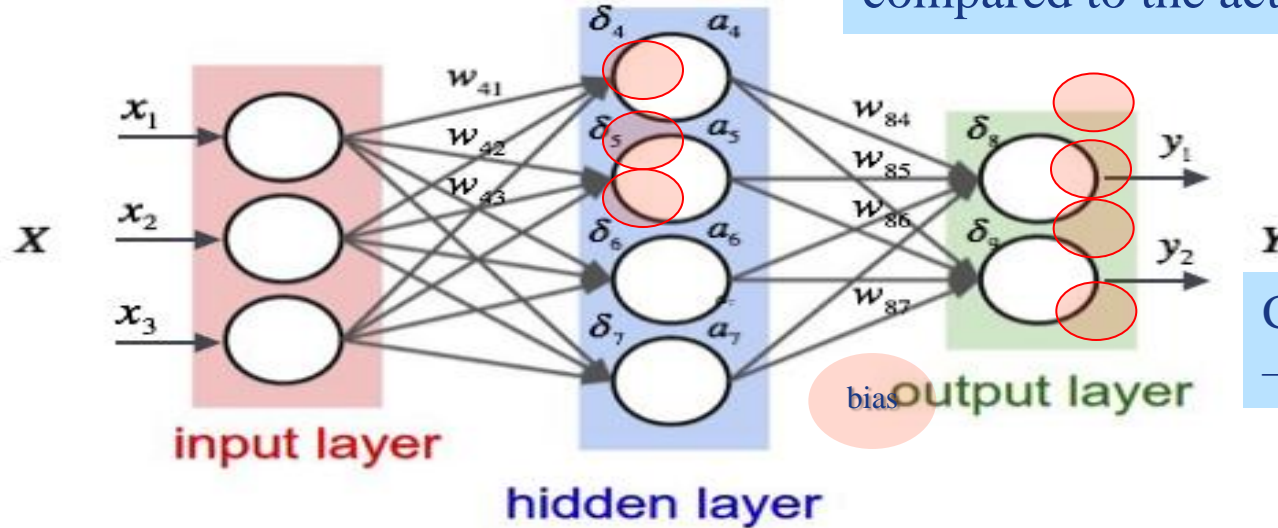
- A multilayer perceptron is a feedforward neural network with one or more hidden (middle) layers.
- The network consists of an input layer of source neurons, at least one middle or hidden layer of computational neurons, and an output layer of computational neurons.
- The input signals are propagated in a forward direction on a layer-by-layer basis.

# Neural Networks

Date	Precipitation [mm]	Snow depth [cm]	Precipitation type	Mean temperature	Minimum temperature	Maximum temperature	Average wind speed (main observations) [m/s]	Hours of sunshine	Mean relative humidity	Vapour pressure [hPa]
16519981	1	30	Rain	-0.1	-2.8	3	1.7	3.2	94	3.2
20519981	0	30	Not	-2.8	-4.7	3.2	1.6	3.5	90	2.6
80519981	0	30	Not	-5.1	-7.4	-3.8	4.3	4.8	57	2.5
40519981	0	30	Not	-6.6	-7	-4	2.3	0	73	2.8
50519981	0.7	11	Snow	-12.7	-14.5	-6.1	1.2	0	71	1.7
60519981	0	11	Not	-15.3	-17.3	-12.7	0.8	4.9	77	1.4
70519981	0	11	Not	-8	-15.8	-5.7	5.3	4.8	36	1.3
80519981	6.4	34	Snow	-1	-10	2.3	2.6	0	86	5.2
90519981	0.1	34	Rain	0.6	-2.4	3.5	0.3	1	84	5.8
130519981	0	34	Not	-5.1	-8.6	2.1	3.1	5.6	58	2.6
110519981	0	34	Not	-6.6	-8.4	-3.2	2.4	0.9	75	4.4
120519981	0	34	Not	0.9	-1	2.7	2.8	0	73	4.6
140519981	0	34	Snow	-5.1	-8.3	0.6	0.2	2.9	52	2.4
140519981	6.6	39	Snow	-3.8	-8.8	-2.8	2.2	0	94	4.5
150519981	5	36	Snow	-3.9	-6.1	-1.2	2.8	0	51	2.6
180519981	0	21	Not	-3.7	-7.6	2.3	6.7	0	35	2.3
170519981	0	21	Not	-5.1	-6.4	-1.9	4.3	3.6	48	2.3
180519981	0	21	Not	-9.9	-11.5	-5.5	3.3	0	64	1.9

Training to improve accuracy

To train the net, the outputs of the net are compared to the actual labels.

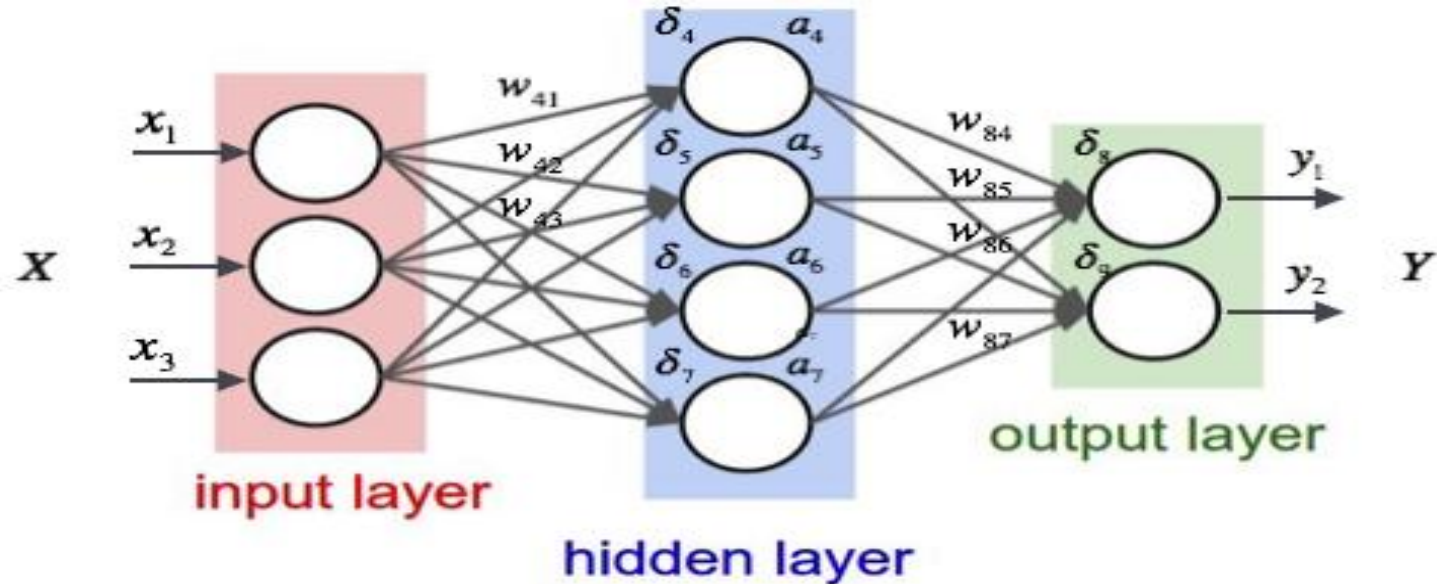


Cost/ error= desired output  
– actual output

Point of training? – To minimize the cost/ loss (error)

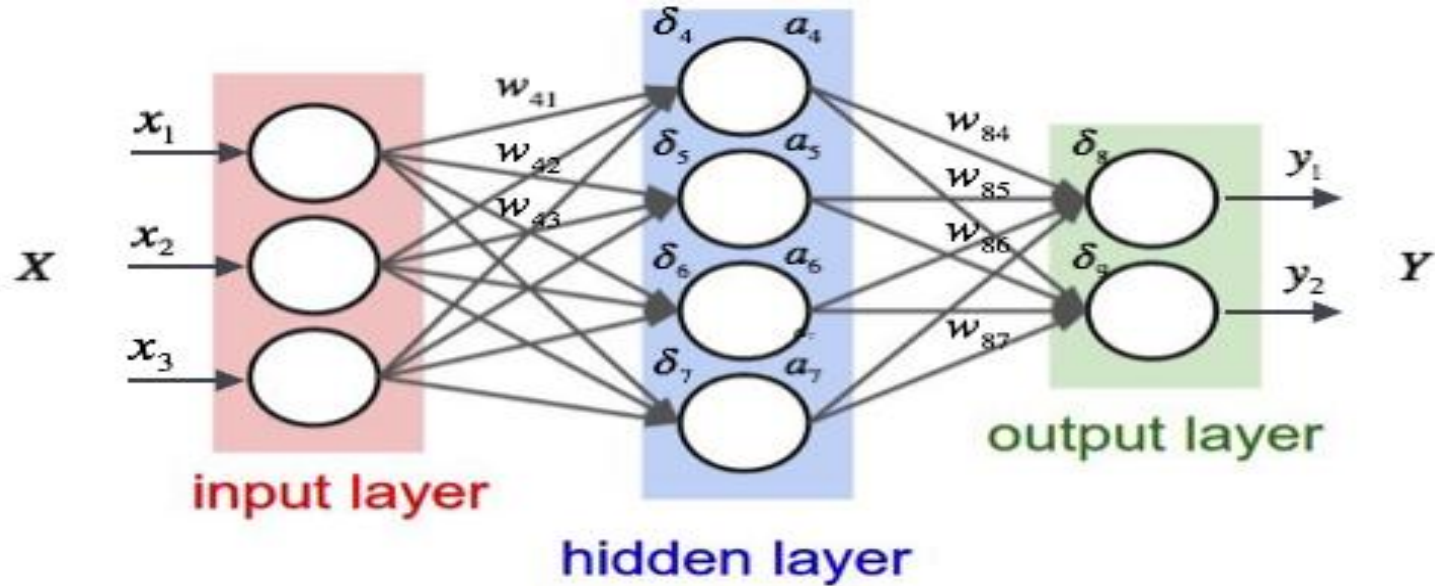
How?

# Neural Networks

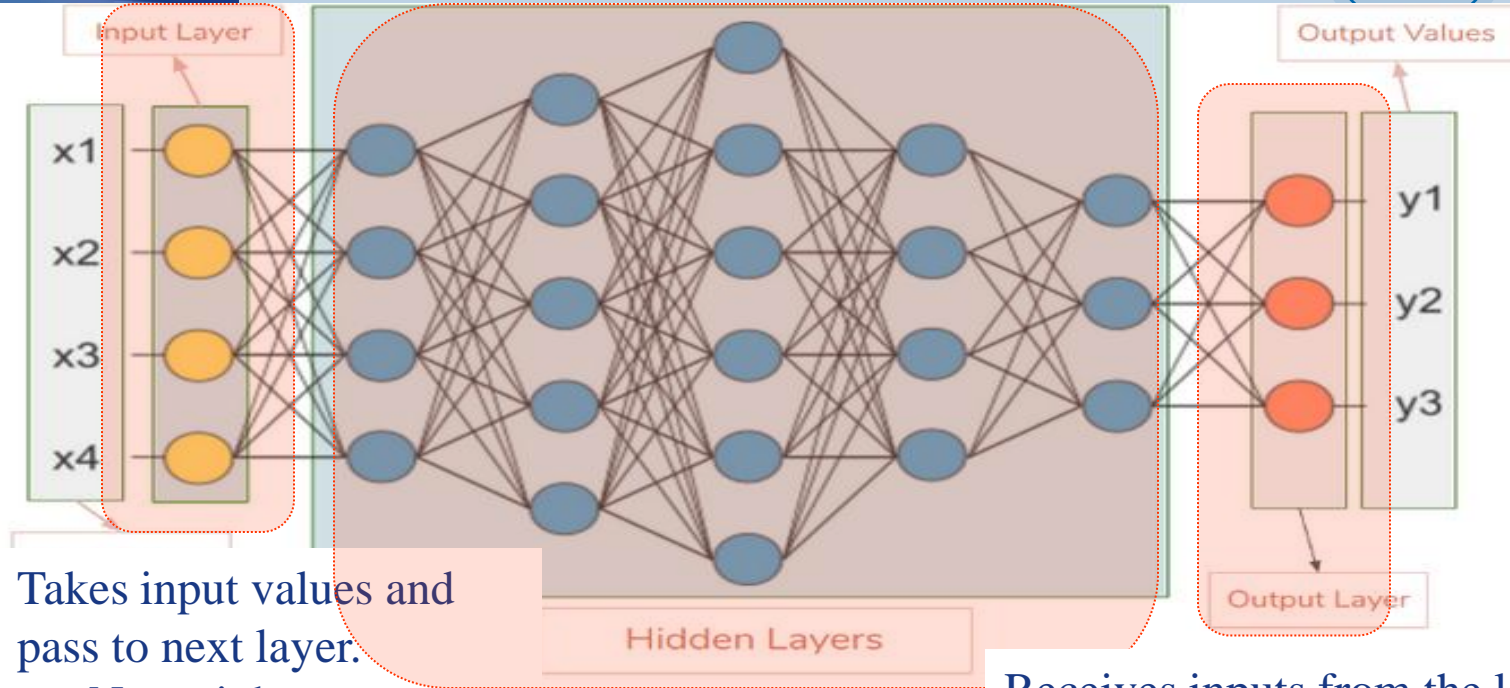


$$Y = \sum (\text{weight} * \text{input}) + \text{bias}$$

# Neural Networks



# Architecture of NN

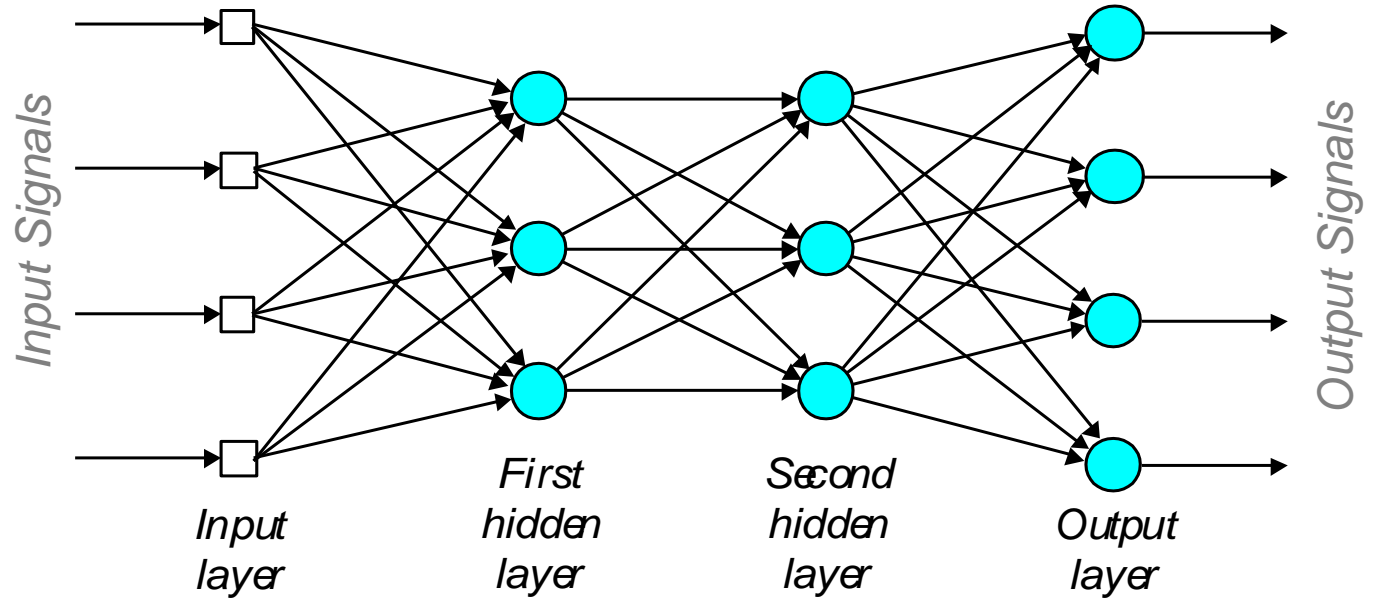


Takes input values and pass to next layer.

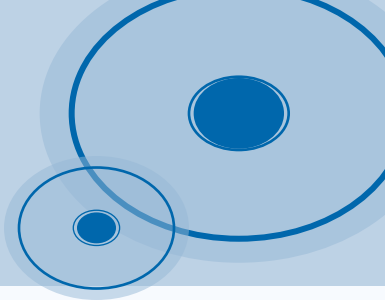
- No weight
- No bias

Receives inputs from the last hidden layer. Get desired values.

# Multilayer perceptron with two hidden layers



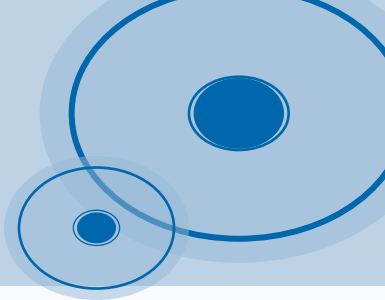
# What does the middle layer hide?



- A hidden layer “hides” its desired output.
- Neurons in the hidden layer cannot be observed through the input/output behavior of the network. There is no obvious way to know what the desired output of the hidden layer should be.

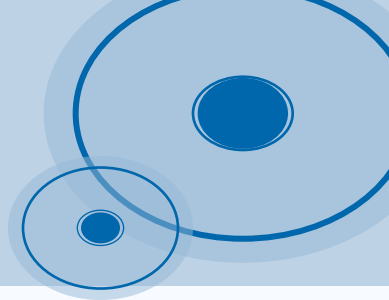


# What does the middle layer hide?



- Commercial ANNs incorporate three and sometimes four layers, including one or two hidden layers.
- Each layer can contain from 10 to 1000 neurons. Experimental neural networks may have five or even six layers, including three or four hidden layers, and utilize millions of neurons.

# Back-propagation neural network



- Learning in a multilayer network proceeds the same way as for a perceptron.
- A training set of input patterns is presented to the network.
- The network computes its output pattern, and if there is an error - or in other words a difference between actual and desired output patterns - the weights are adjusted to reduce this error.

# Back-propagation NN - 2 phases

- In a back-propagation neural network, the learning algorithm has **two phases**.
- First, a training input pattern is presented to the network input layer. The network propagates the input pattern from layer to layer until the output pattern is generated by the output layer.

# Back-propagation NN – 2 phases

- If this pattern is different from the desired output, an error is calculated and then propagated backwards through the network from the output layer to the input layer. The weights are modified as the error is propagated.

# Three-layer back-propagation neural network

