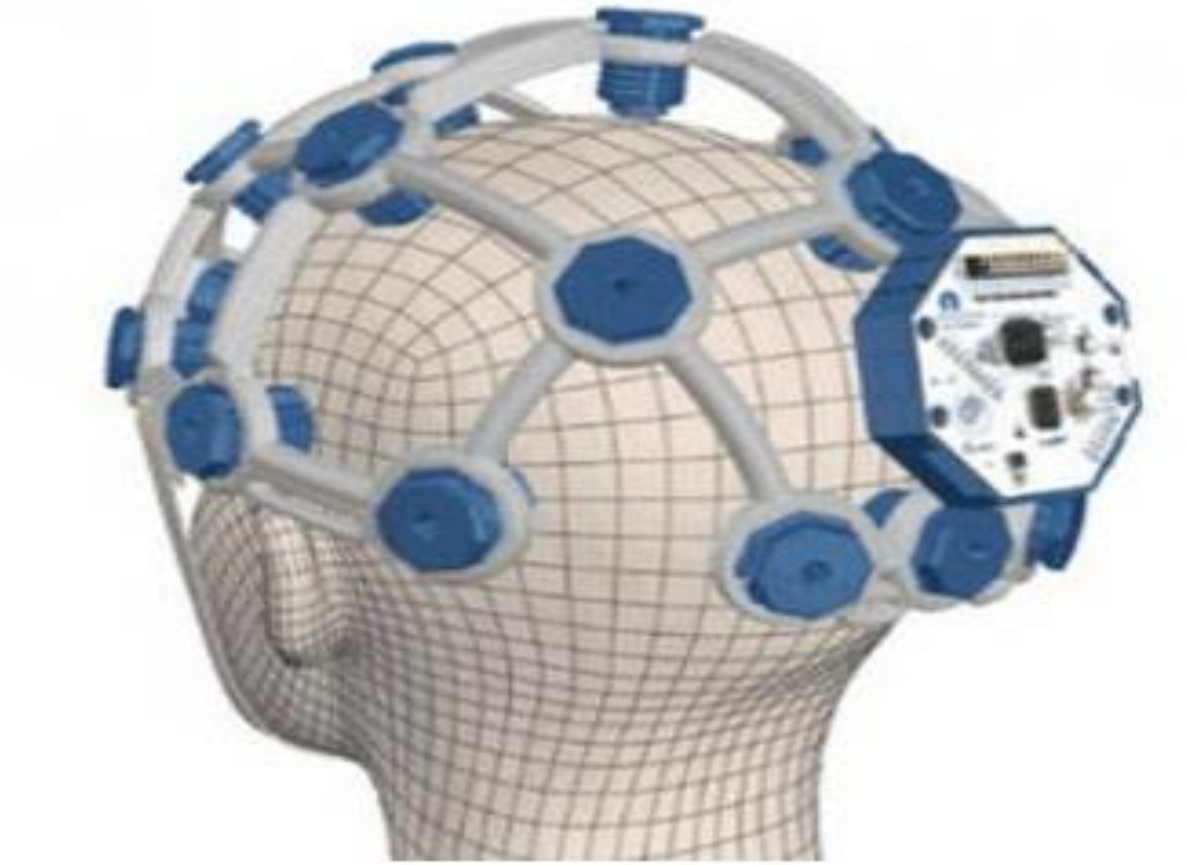
NEURAL NETWORK



www.chandanverma.com

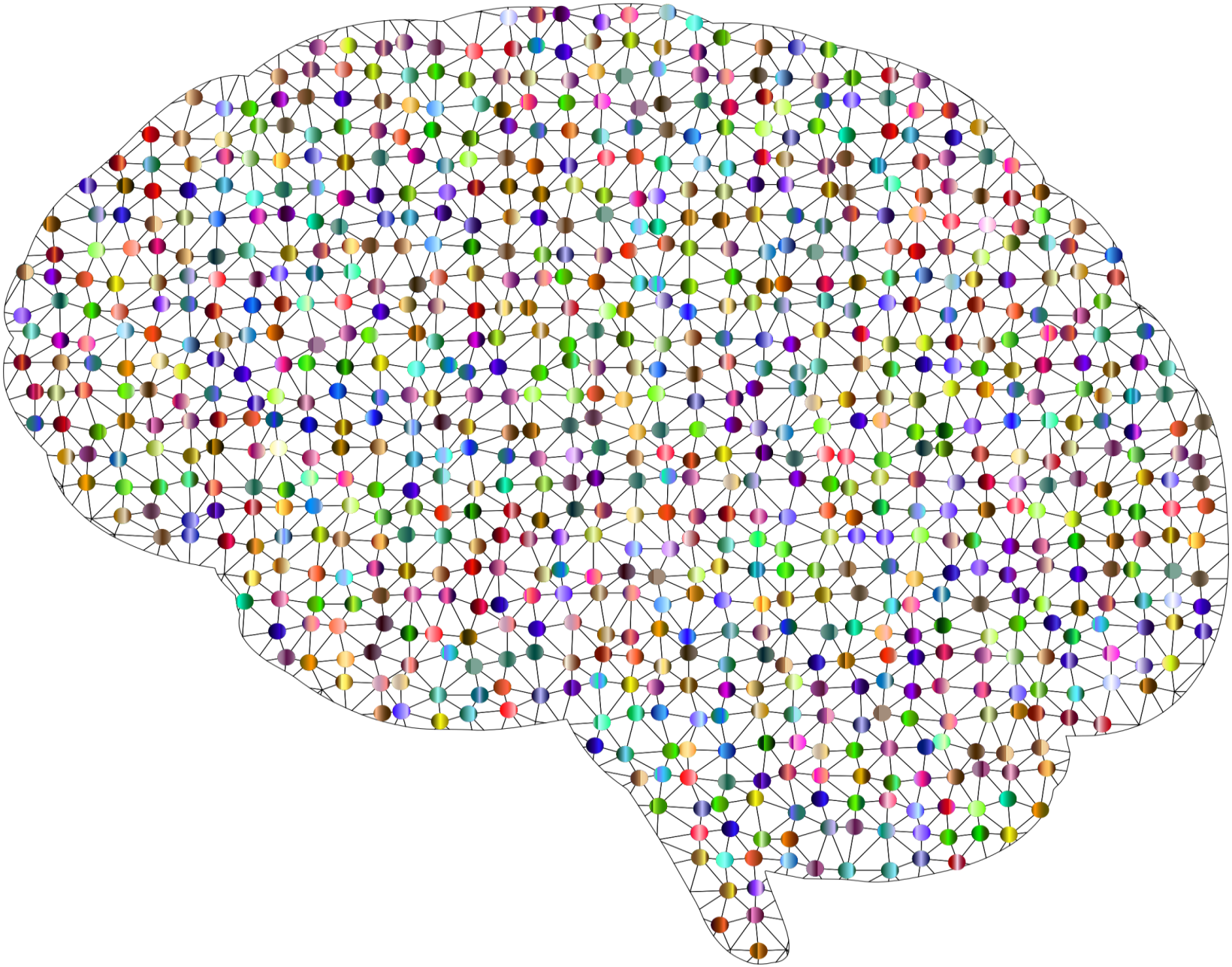
BRAIN

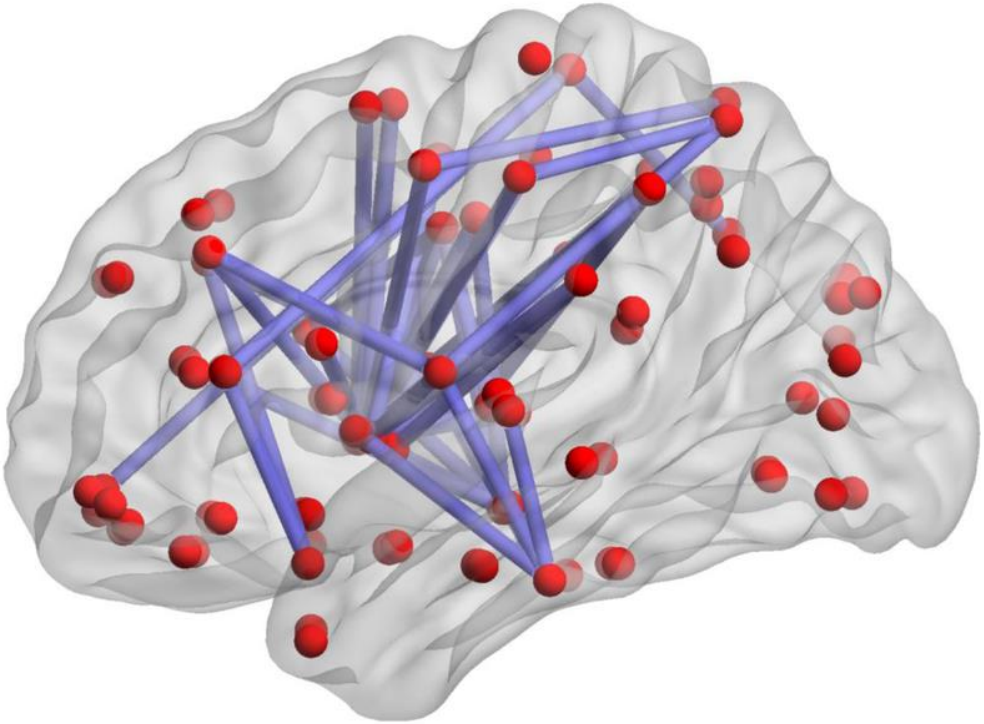
A typical brain contains something like 100 billion miniscule cells called **neurons.**

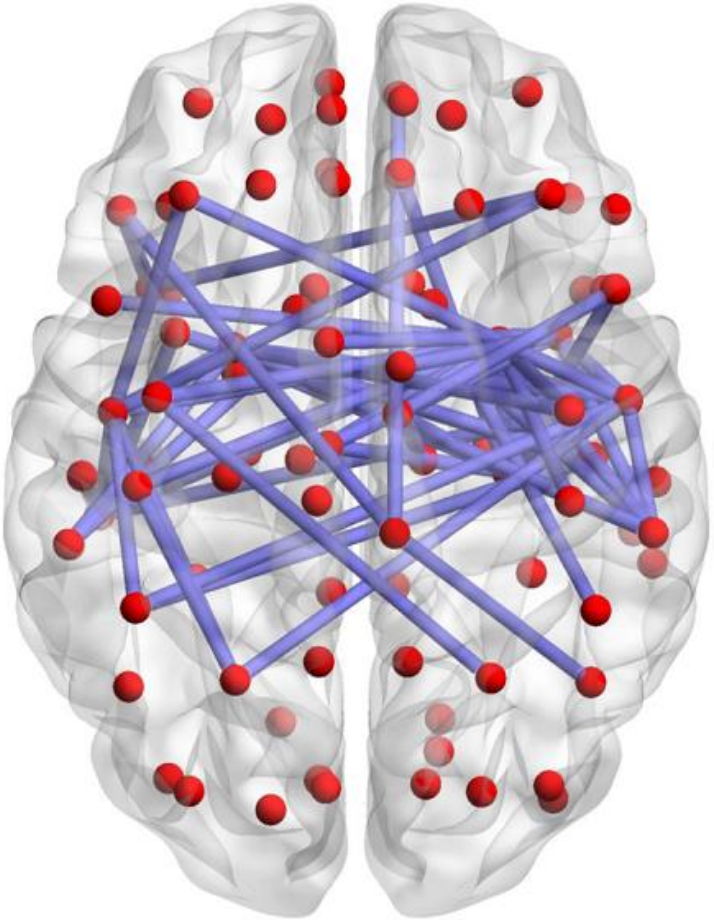
Each neuron is made up of a **cell body** (the central mass of the cell) with a number of connections coming off it.

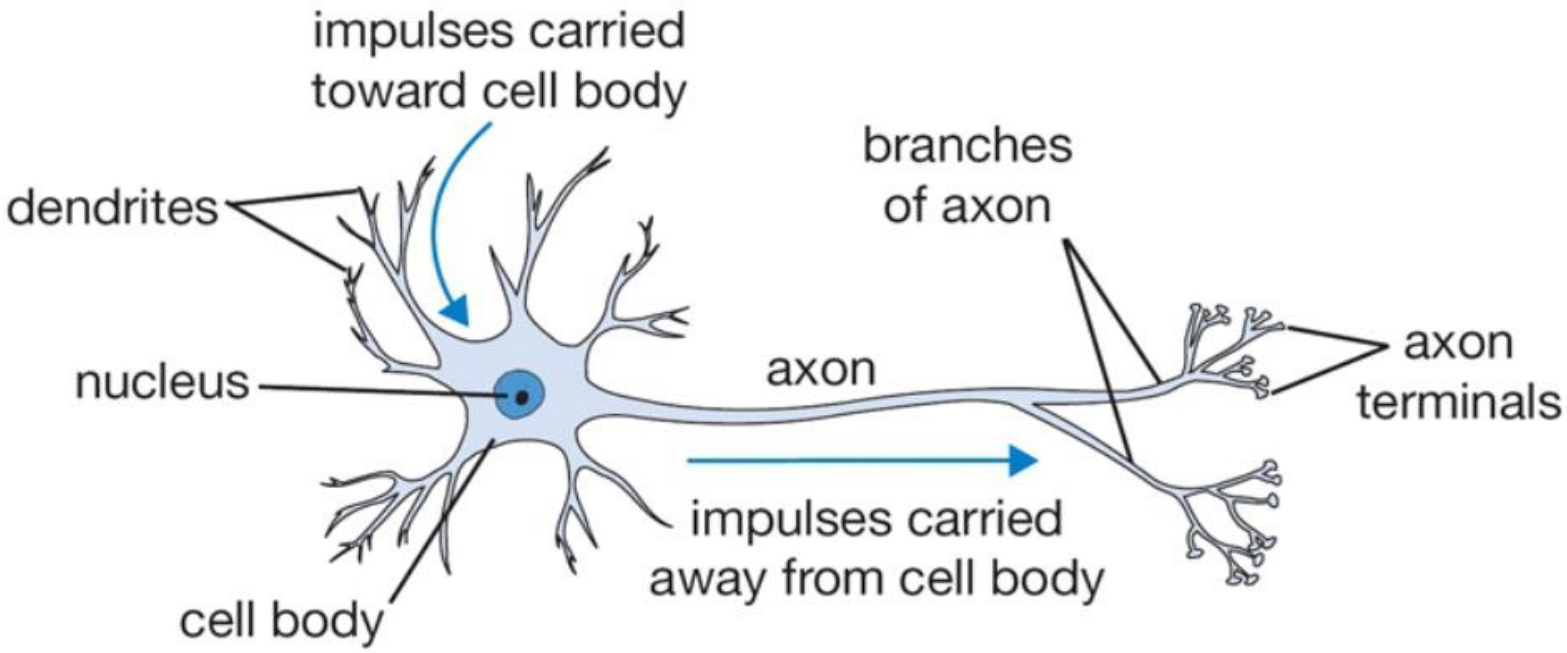
The cell body contains the nucleus, the **storehouse of genetic** information, and gives rise to two types of cell processes, axons and dendrites

www.chandanverma.com









Biological neuron

numerous **dendrites** (the cell's inputs—carrying information toward the cell body) and a single **axon** (the cell's output—carrying information away).

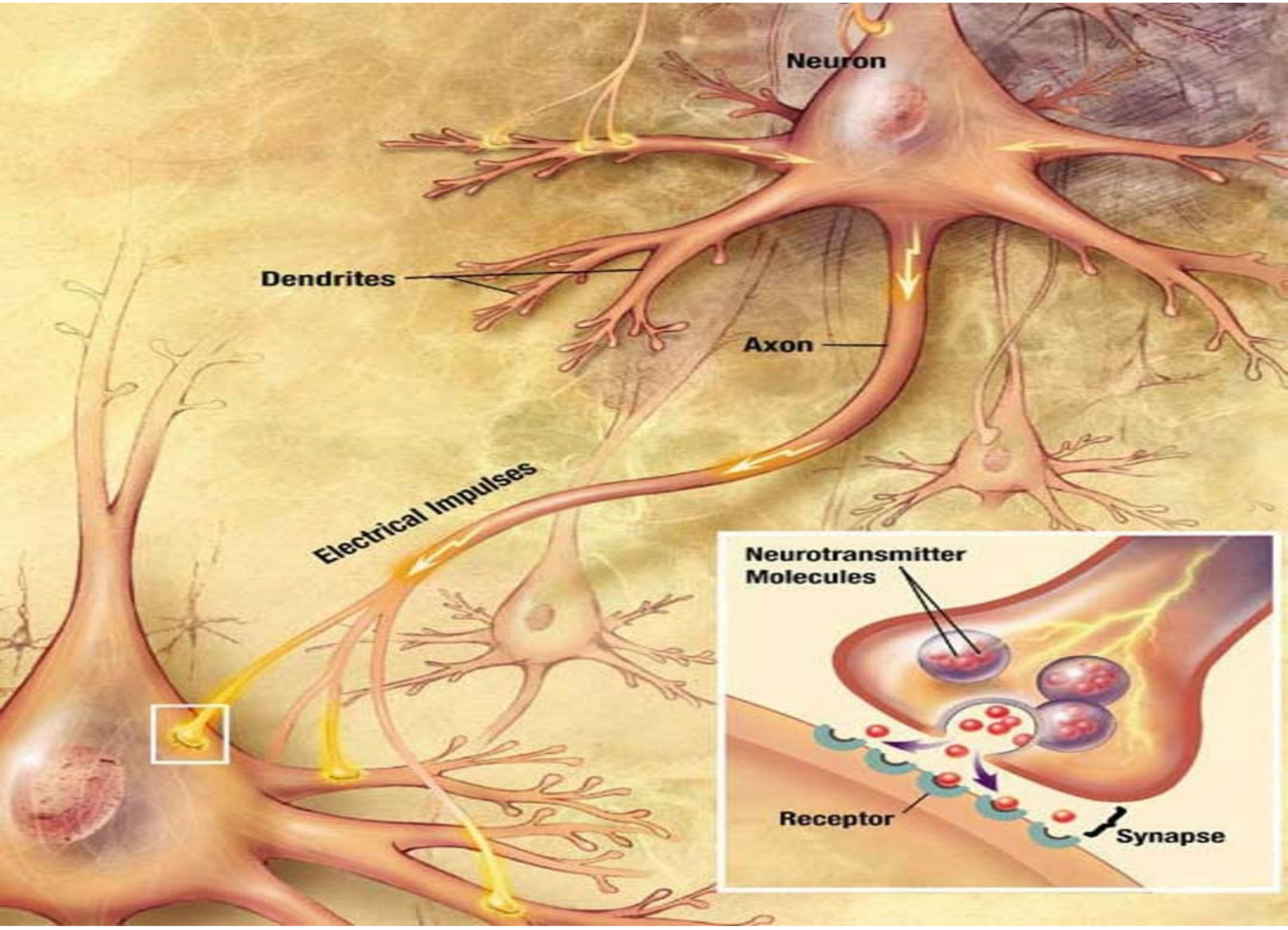
www.chandanverma.com

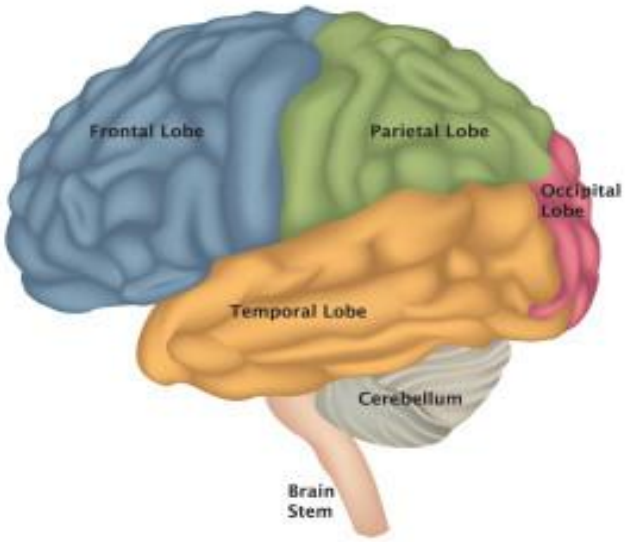
Synapses

The connections between one neuron and another are called synapses. Information always leaves a neuron via its axon , and is then transmitted across a synapse to the receiving neuron.

Neuron Firing

Neurons only fire when input is bigger than some threshold. It should, however, be noted that firing doesn't get bigger as the stimulus increases, its an all or nothing arrangement.


Neuron
Dendrites
Axon
Electrical Impulses
Neurotransmitter Molecules
Receptor
Synapse

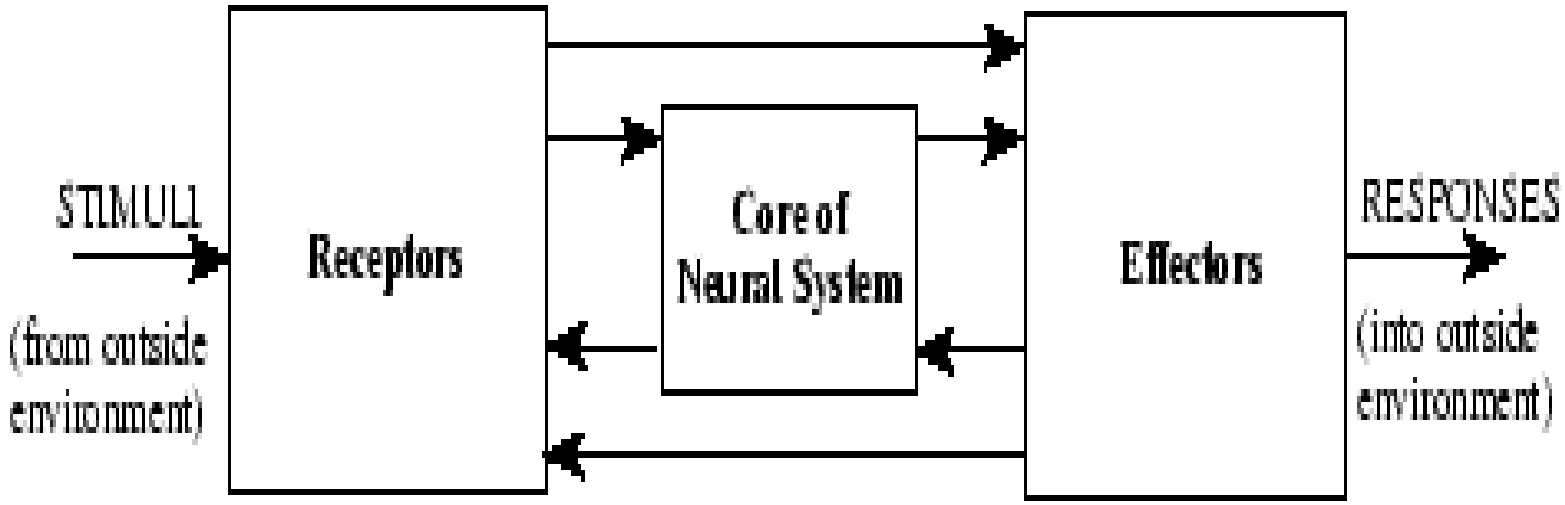


Brain Structure is Parallel and Distributed

The human brain consists of billions of neurons which are interconnected by synapses. If “enough” synaptic inputs to the neuron fires, then the neuron will also fire. This process is called ***thinking.***

Number of Neurons

Animal	Number of Neurons
Common Jellyfish	5,600
Ant	250,000
Frog	16,000,000
Cat	760,000,000
Humans	86,000,000,000
African Elephant	257,000,000,000



The brain receives the stimulus from the outside world, does the processing on the input, and then generates the output.

www.chandanverma.com

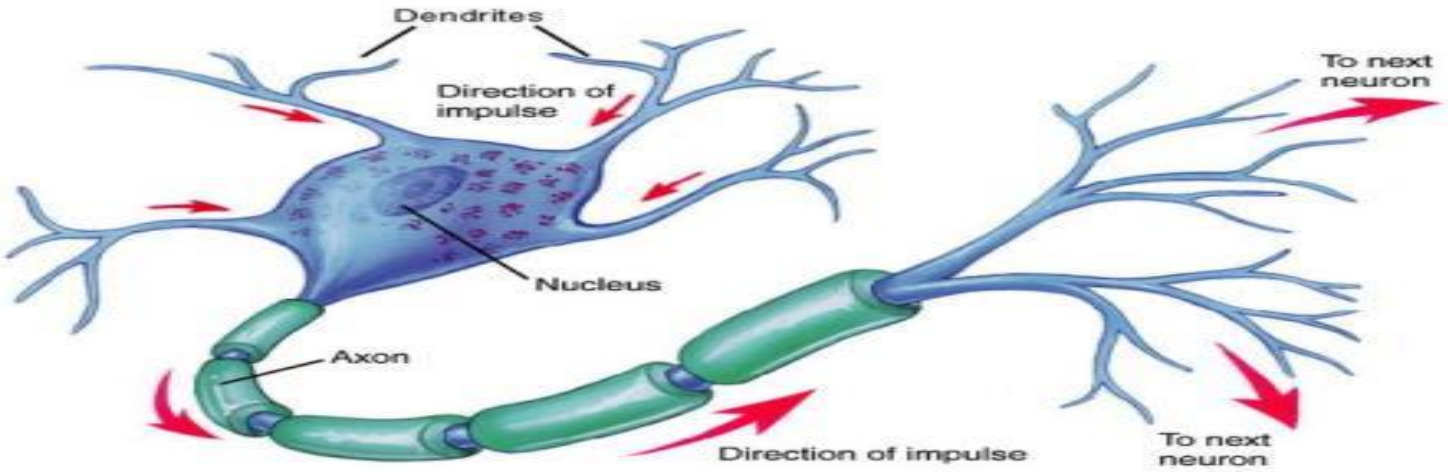
The neural system of the human body consists of three stages: **receptors**, **a neural network**, and **effectors**.

The receptors receive the stimuli either internally or from the external world, then pass the information into the neurons in a form of electrical impulses.

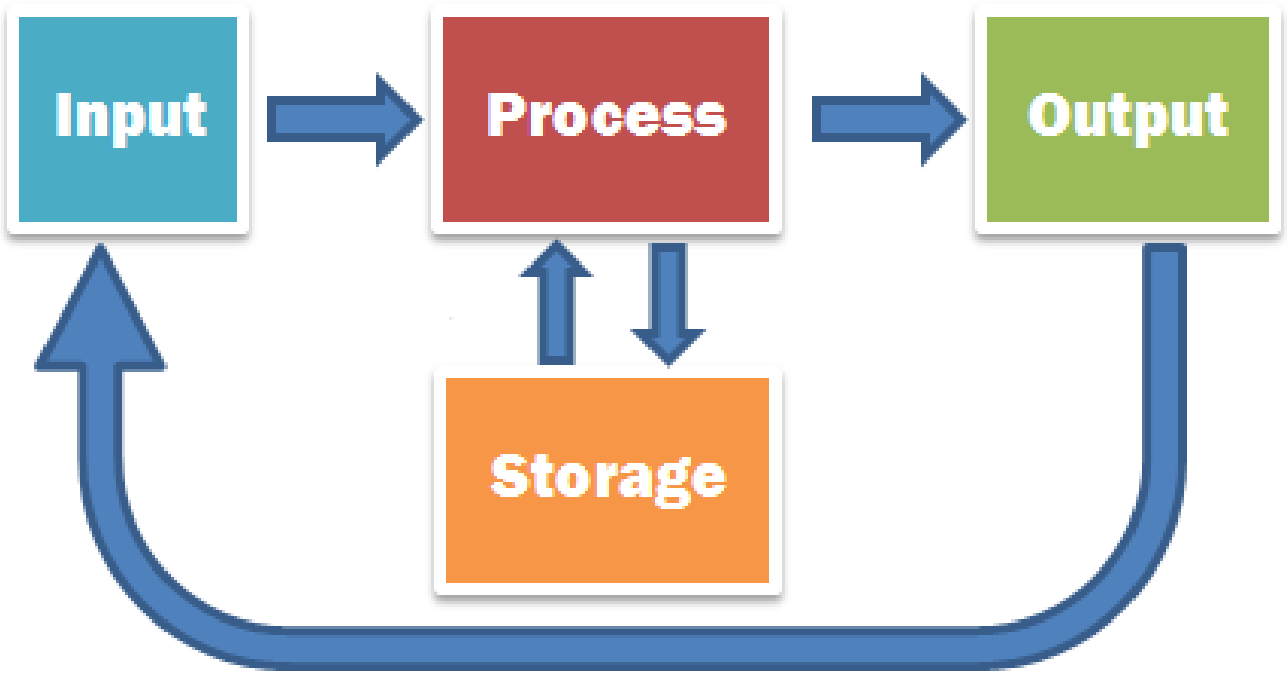
The neural network then processes the inputs then makes proper decision of outputs.

Finally, the effectors translate electrical impulses from the neural network into responses to the outside environment.

www.chandanverma.com

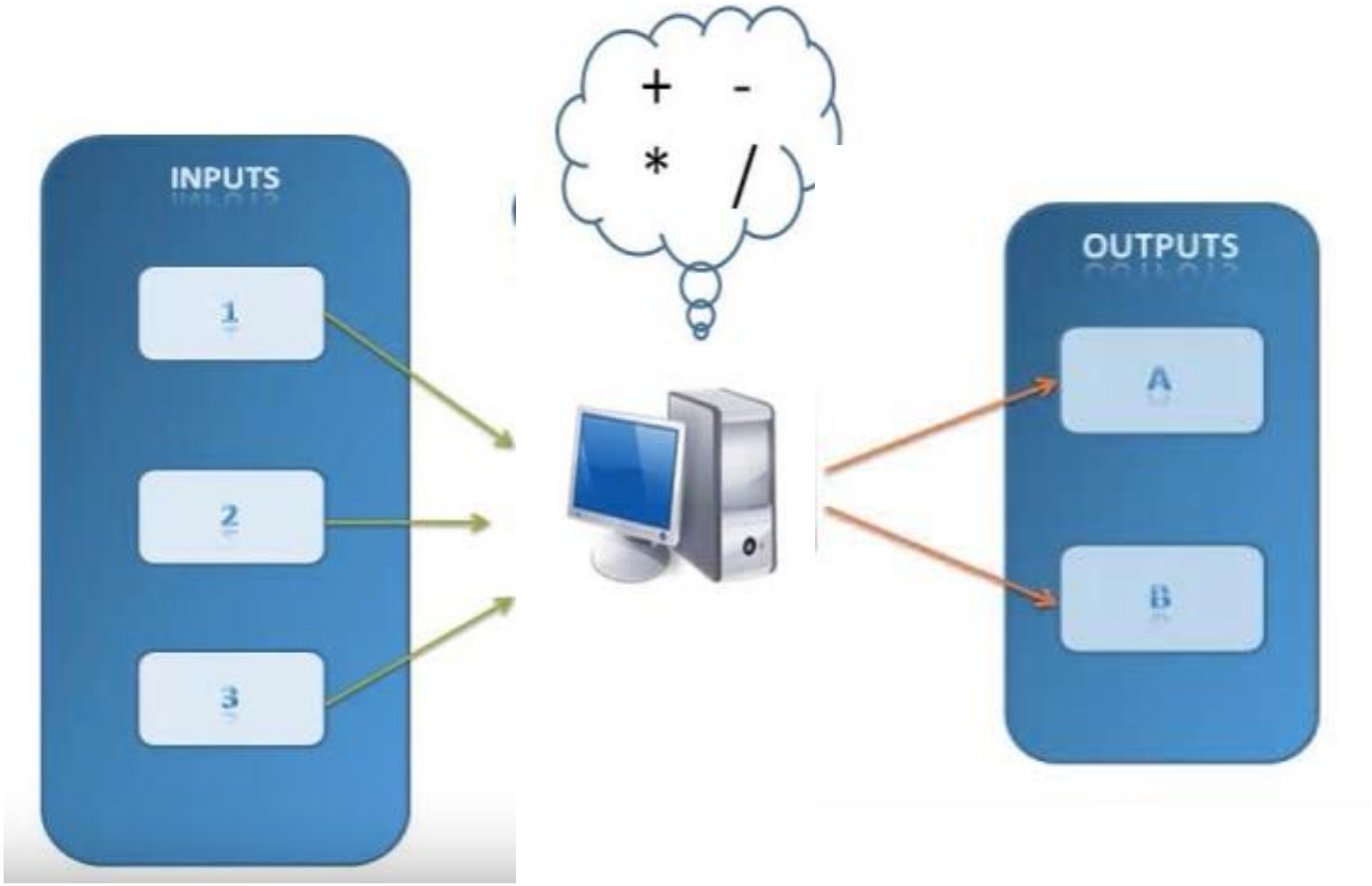

Dendrites
Direction of impulse
To next neuron
Nucleus
Axon
Direction of impulse
To next neuron

How Computer Work



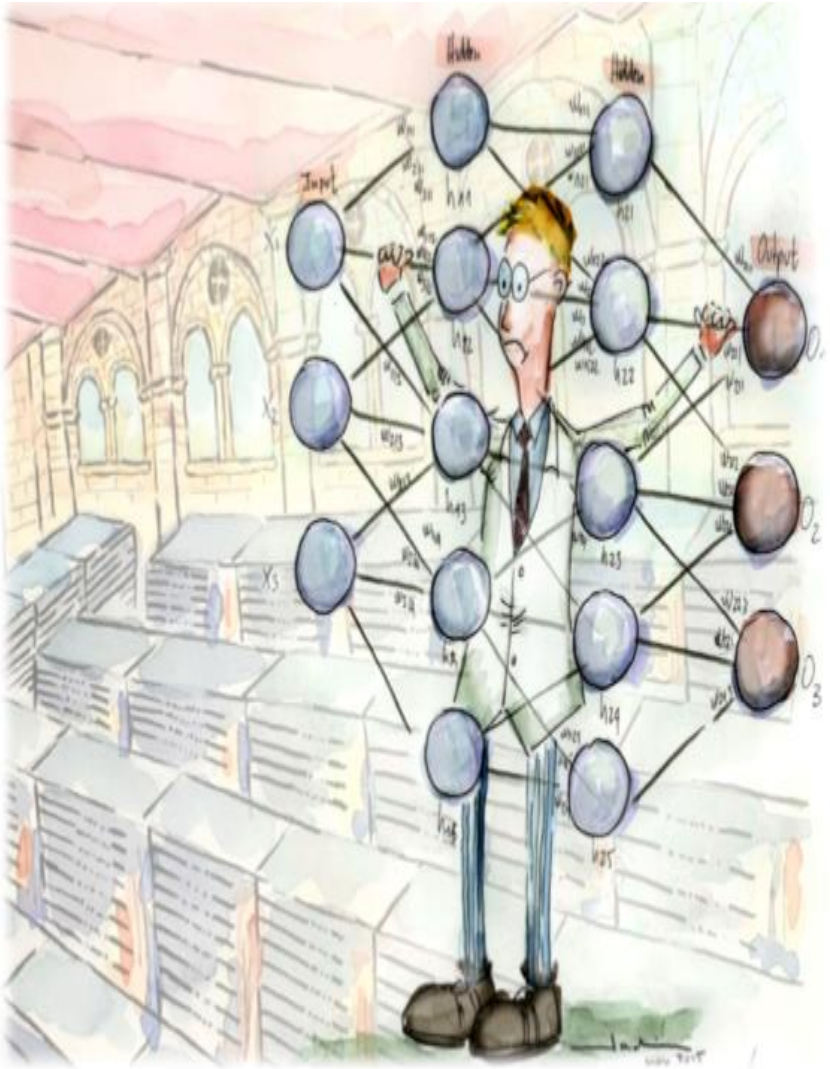
Feedback

www.chandanverma.com



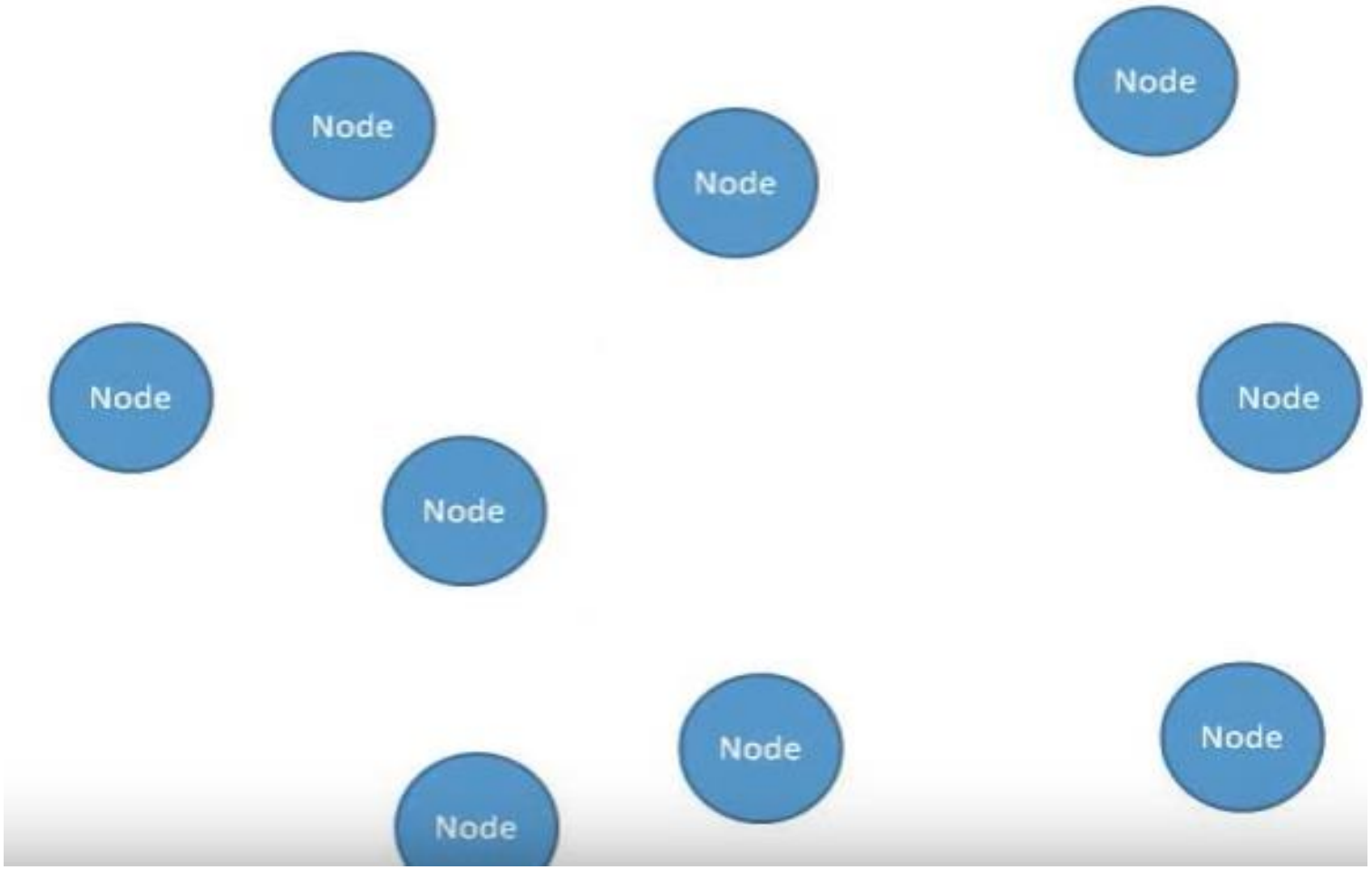
www.chandanverma.com

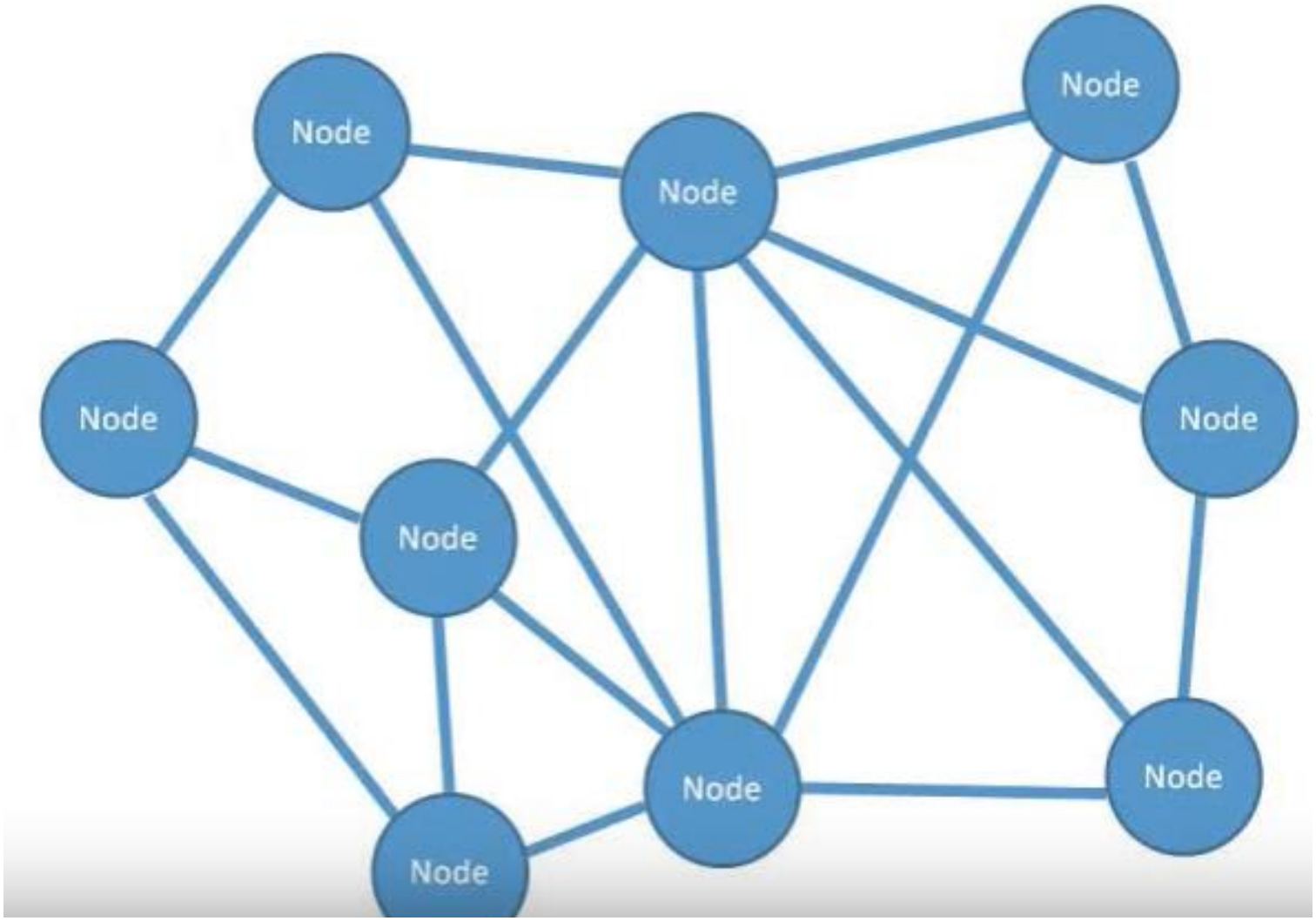
New type of Computing is required



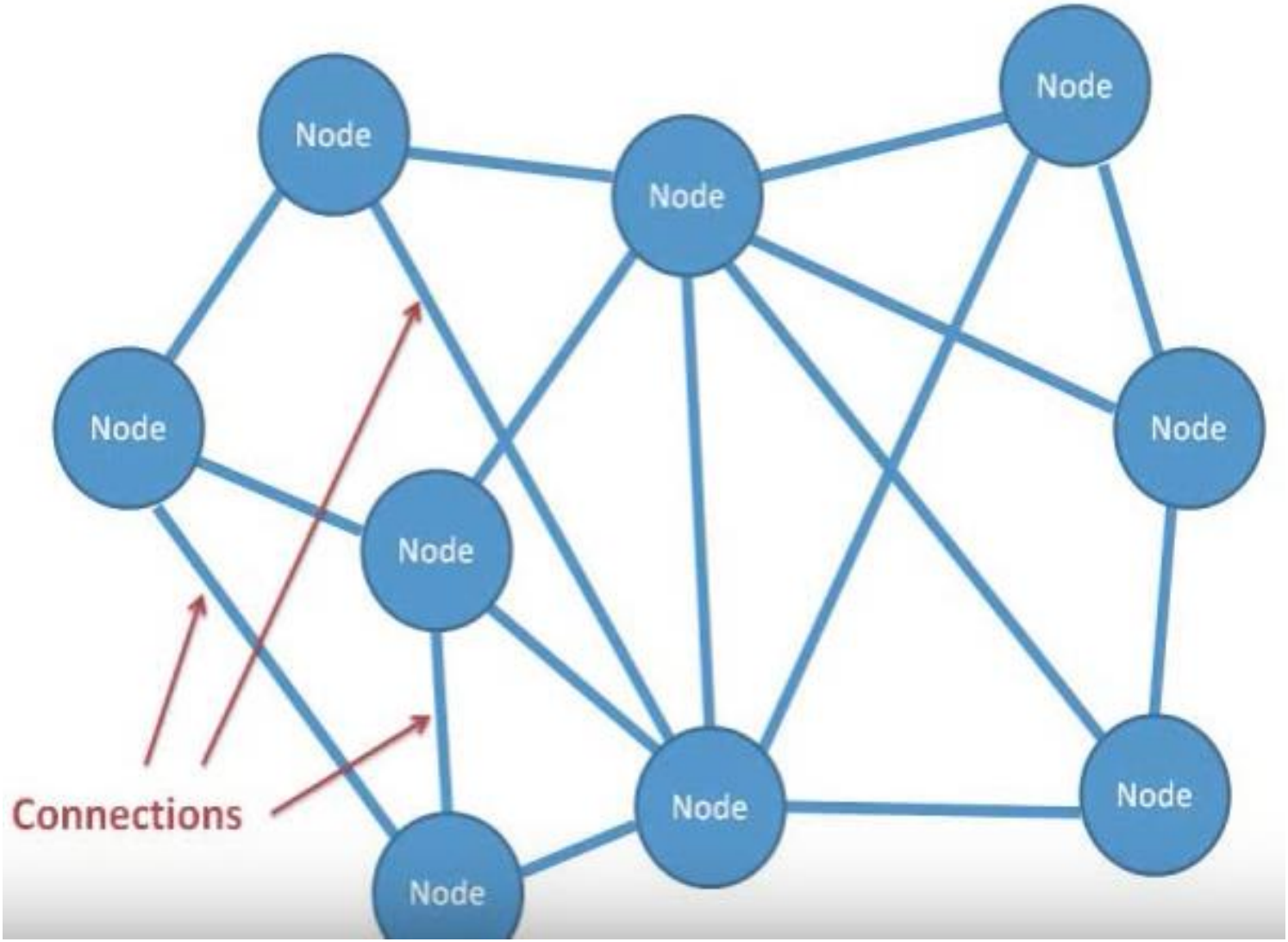
Giving Computers a greater ability to understand information and to learn, to Reason , and act upon it

Neural Network



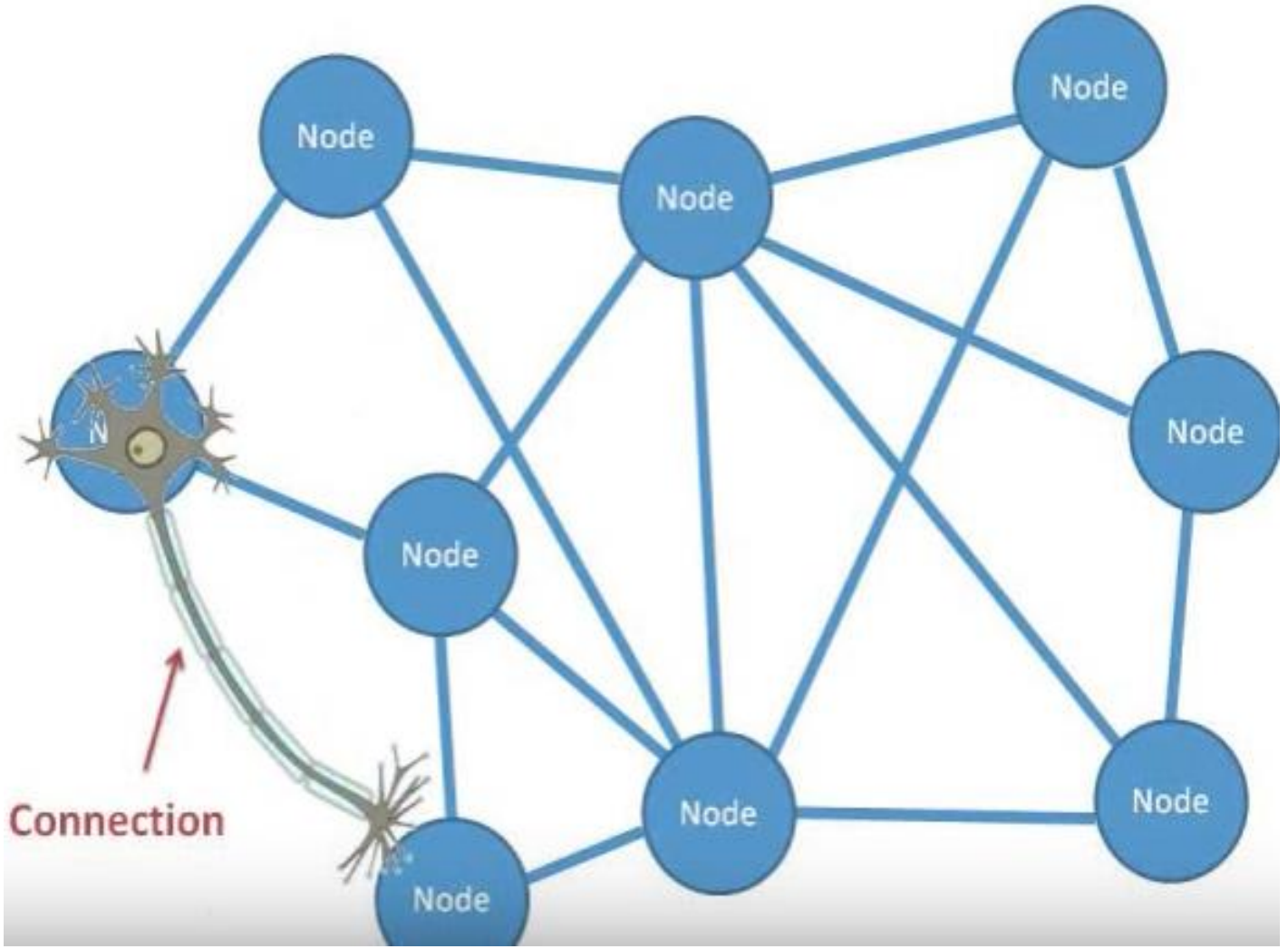


www.chandanverma.com

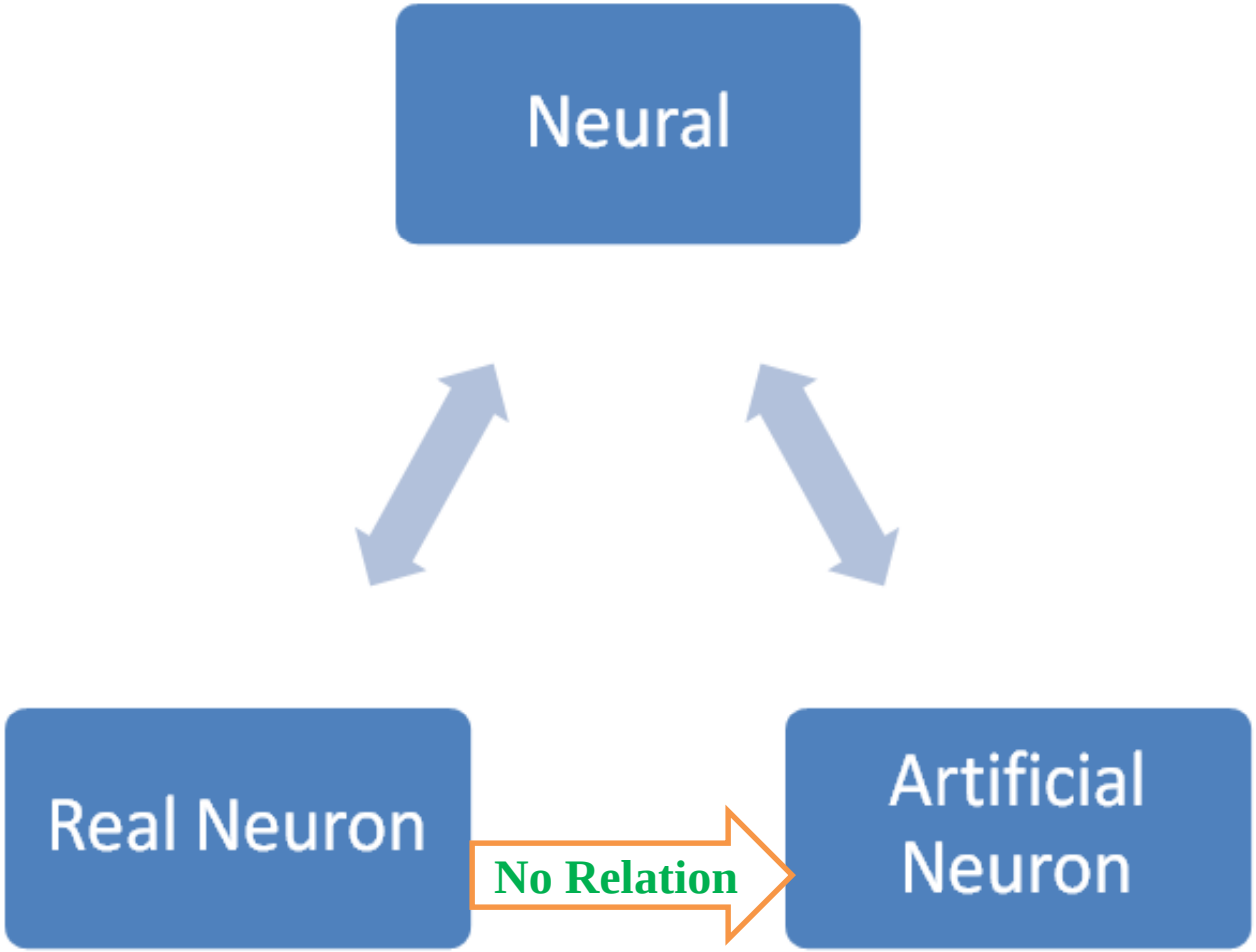


www.chandanverma.com

Modeling of Computer similar as Brain is.

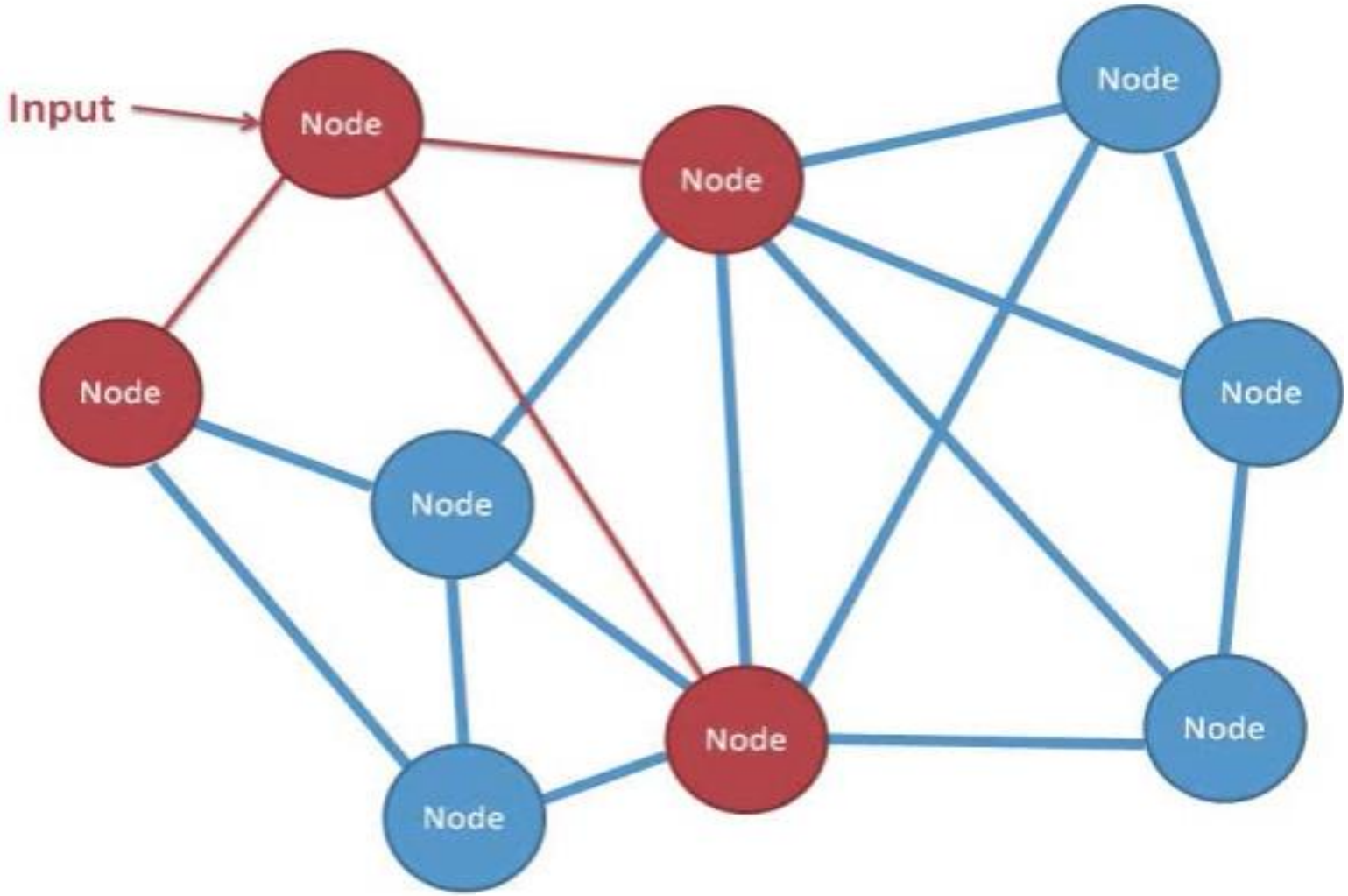


www.chandanverma.com

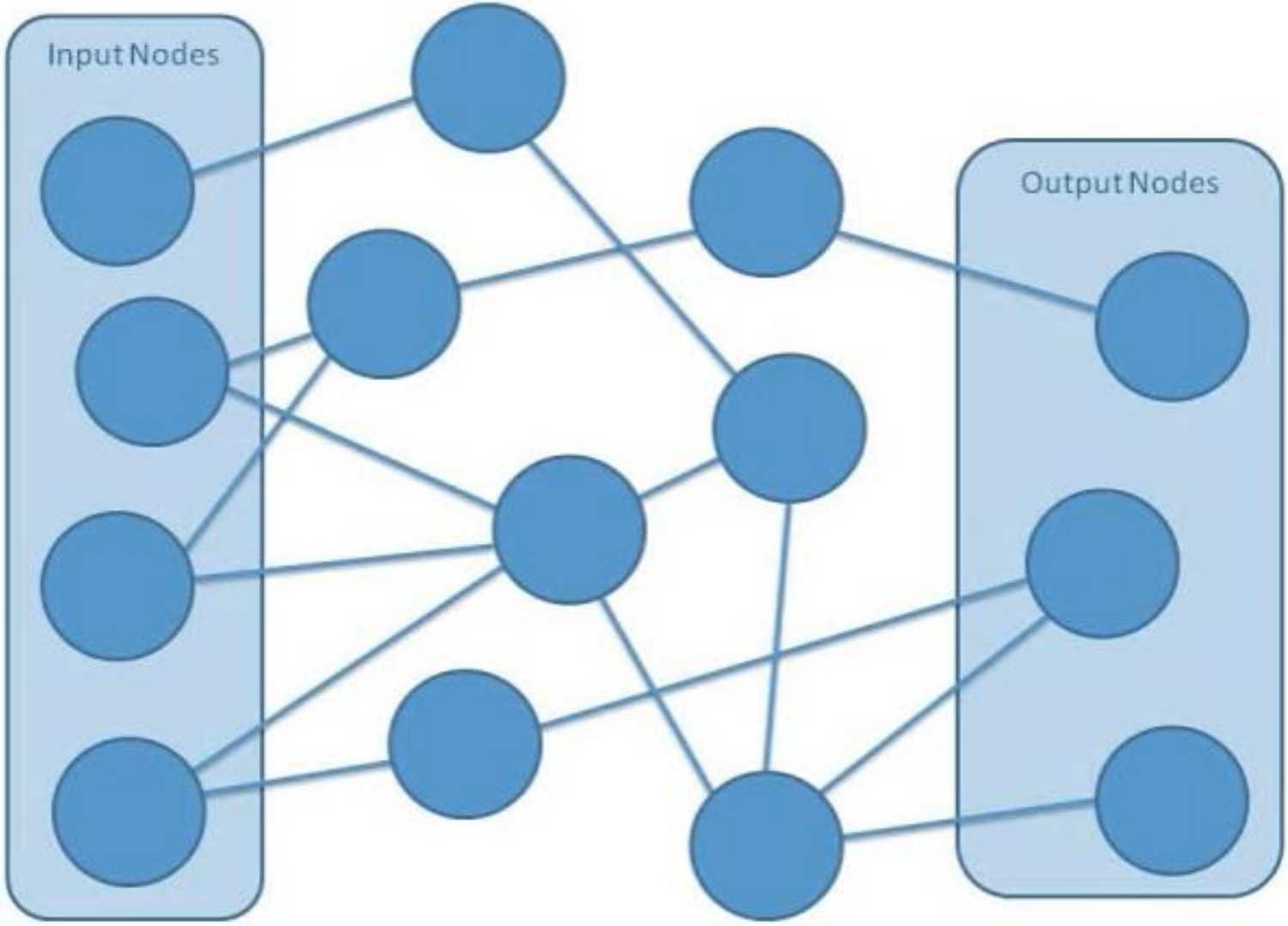


www.chandanverma.com

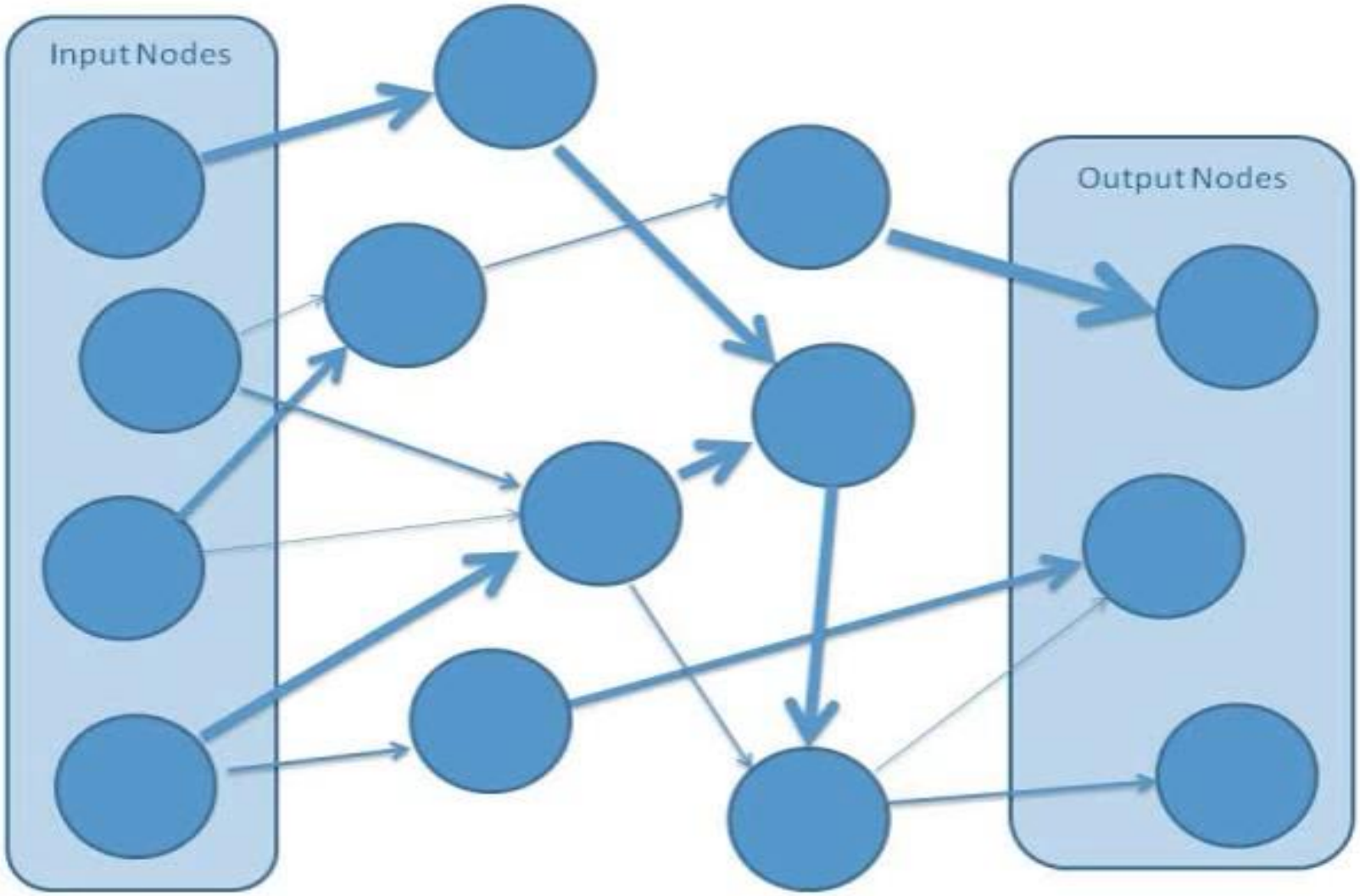
Neural Network Modeling



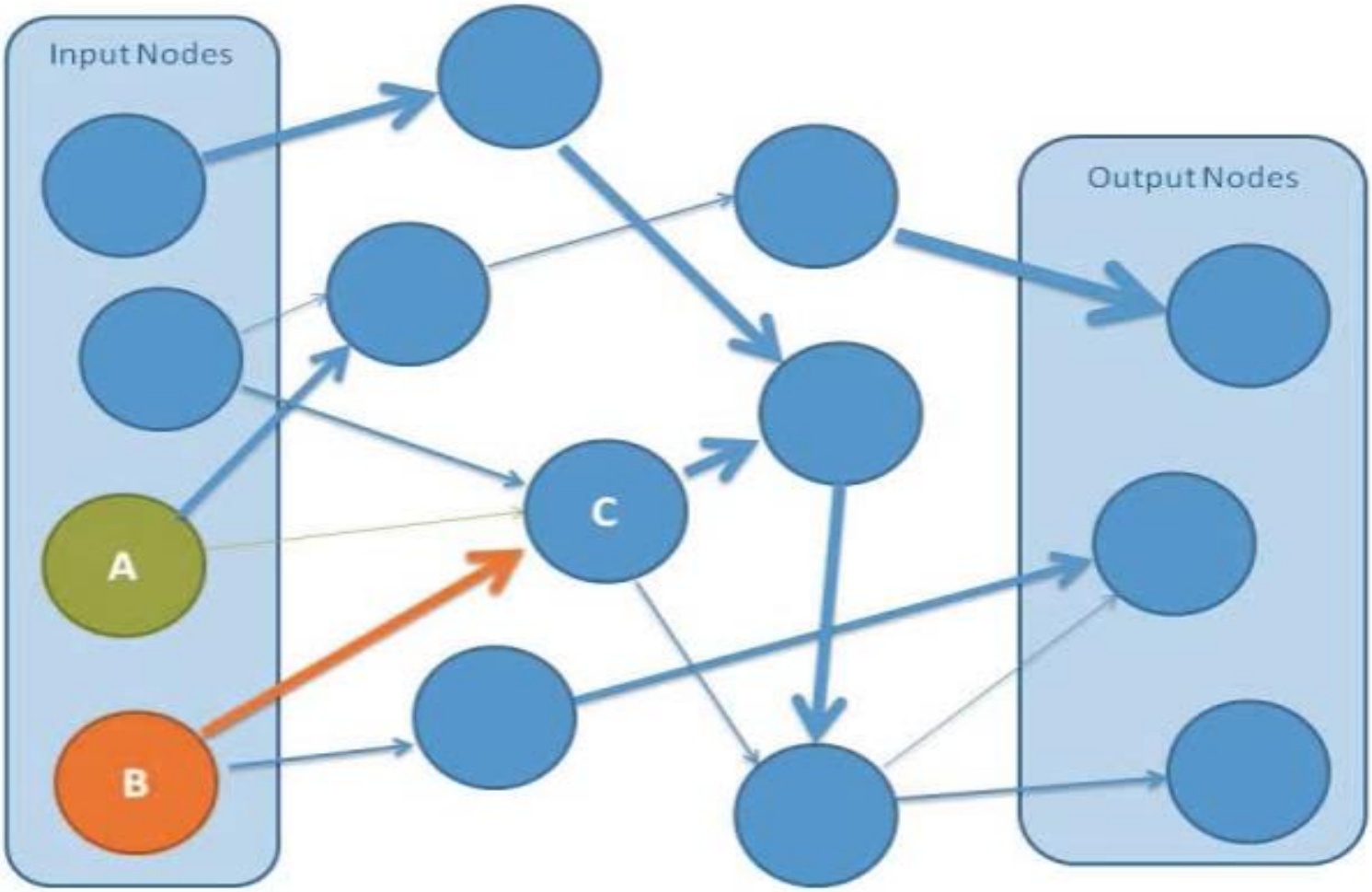
www.chandanverma.com



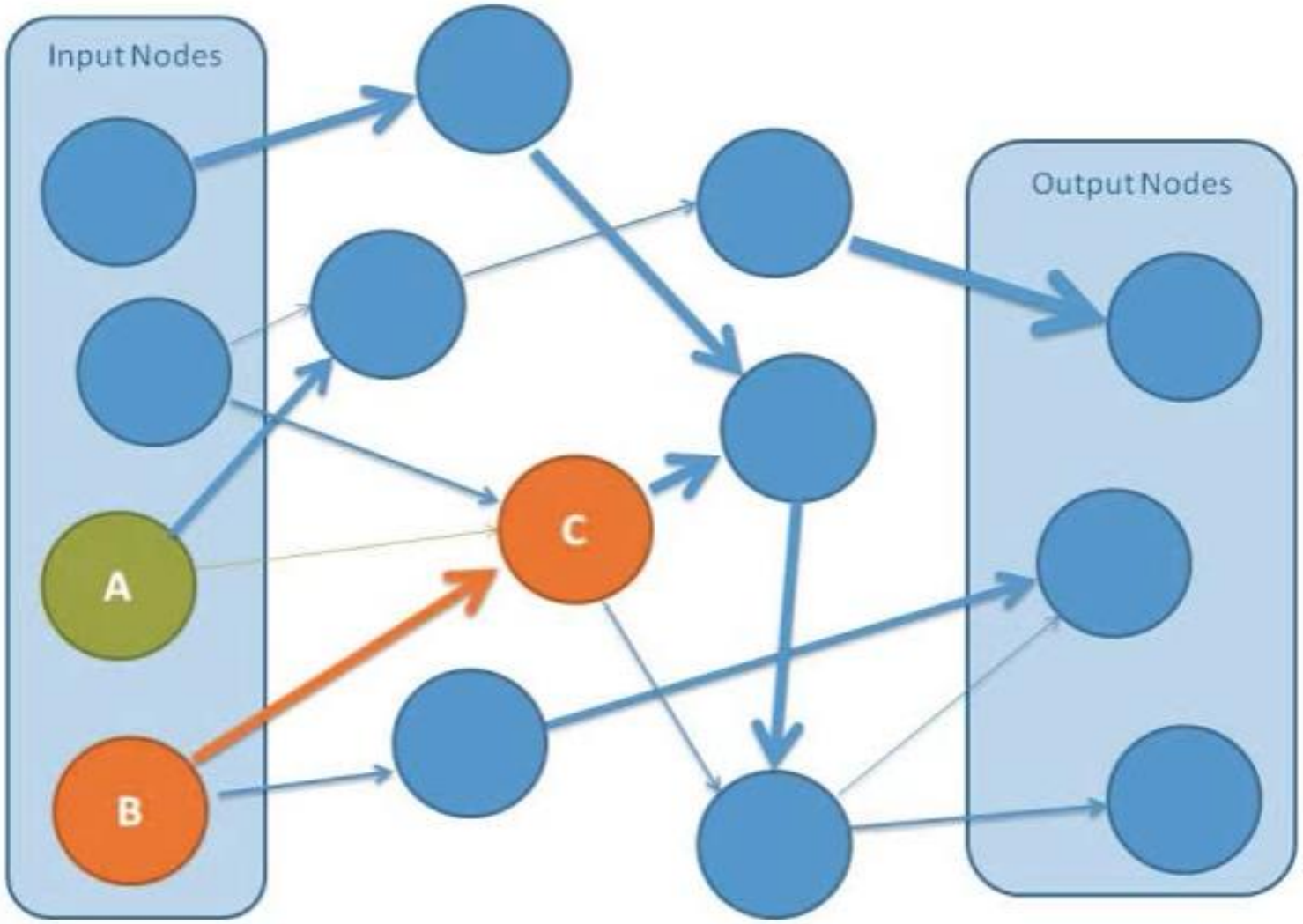
Connection Values Are Called Weights



Different Weights

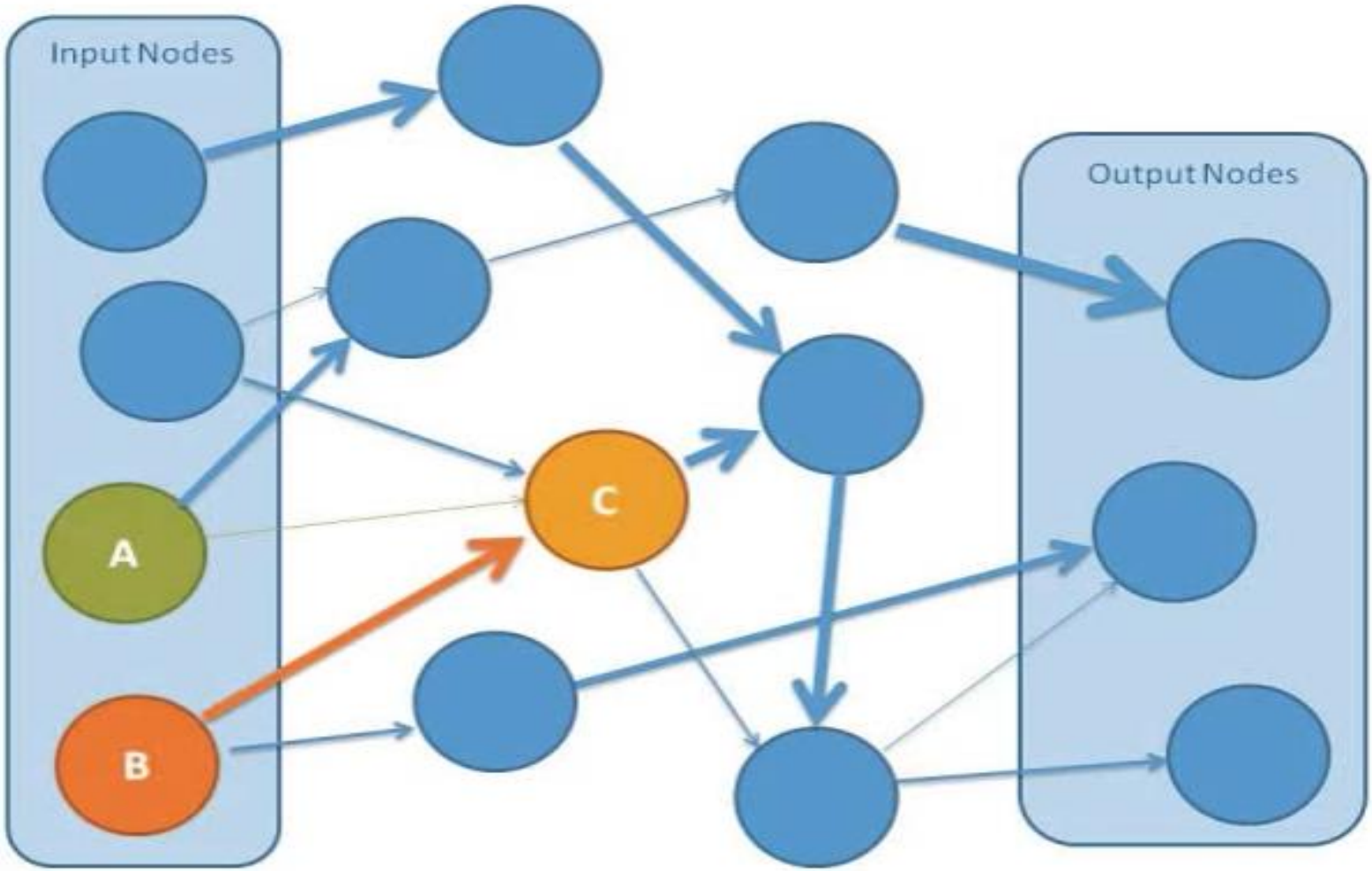


www.chandanverma.com

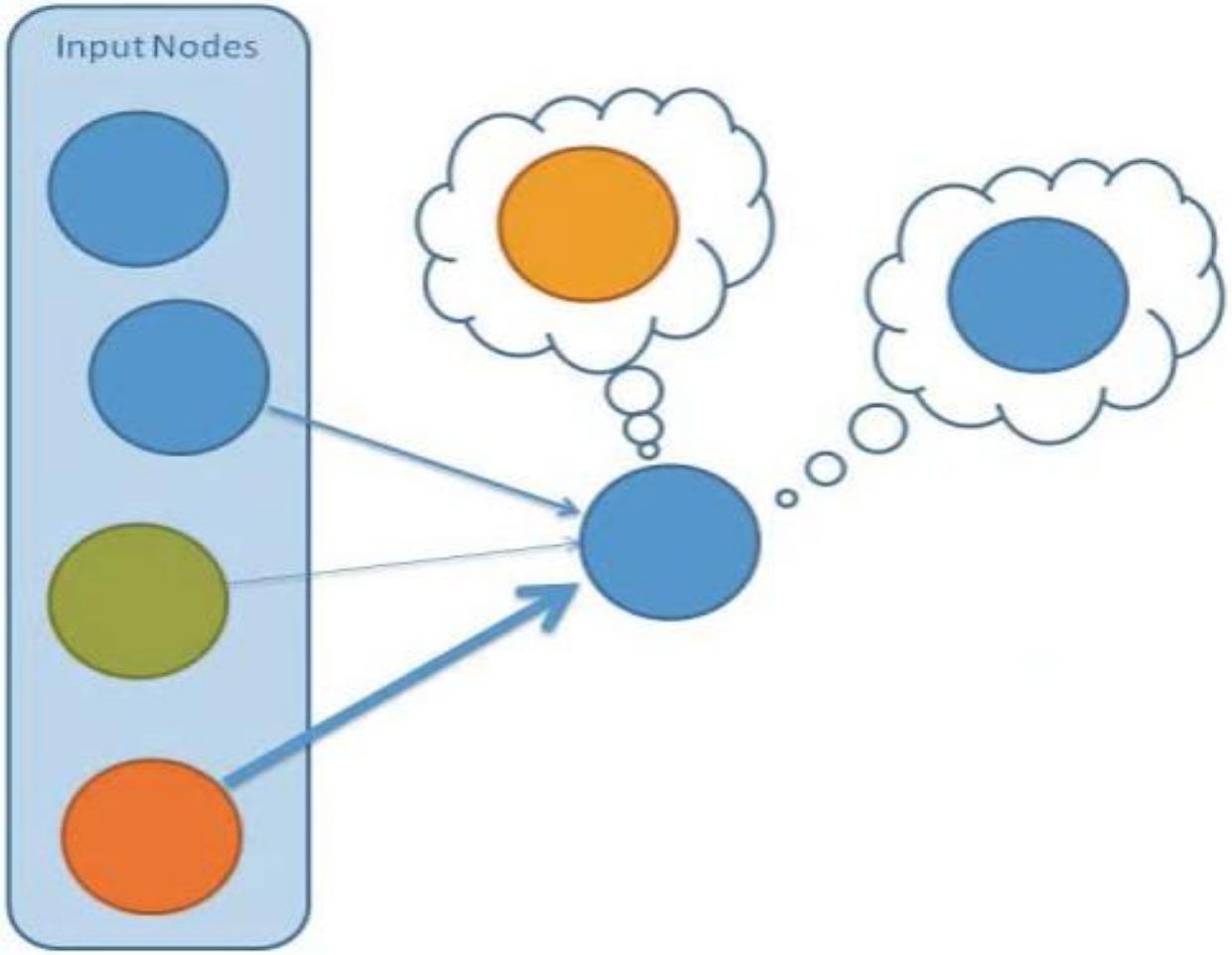


More often Node value design on the basis of

SUM and **Average**

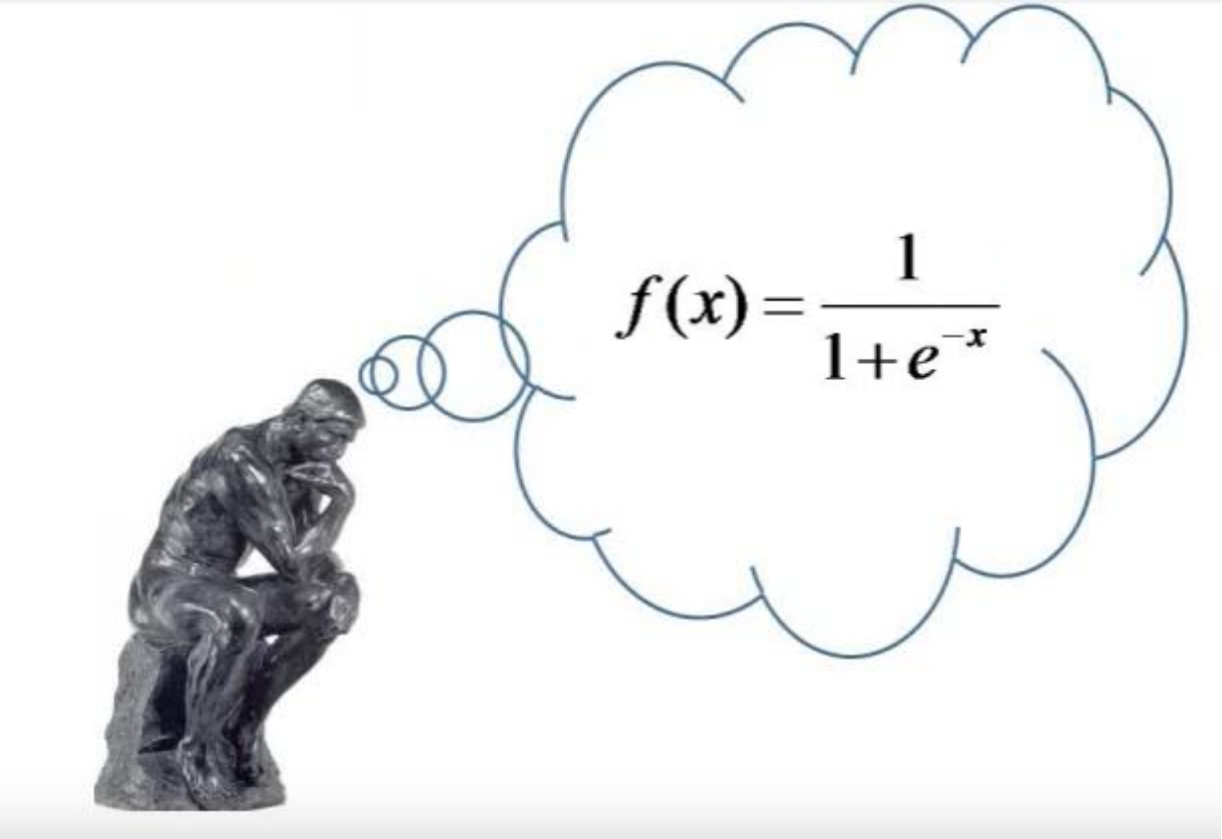


www.chandanverma.com

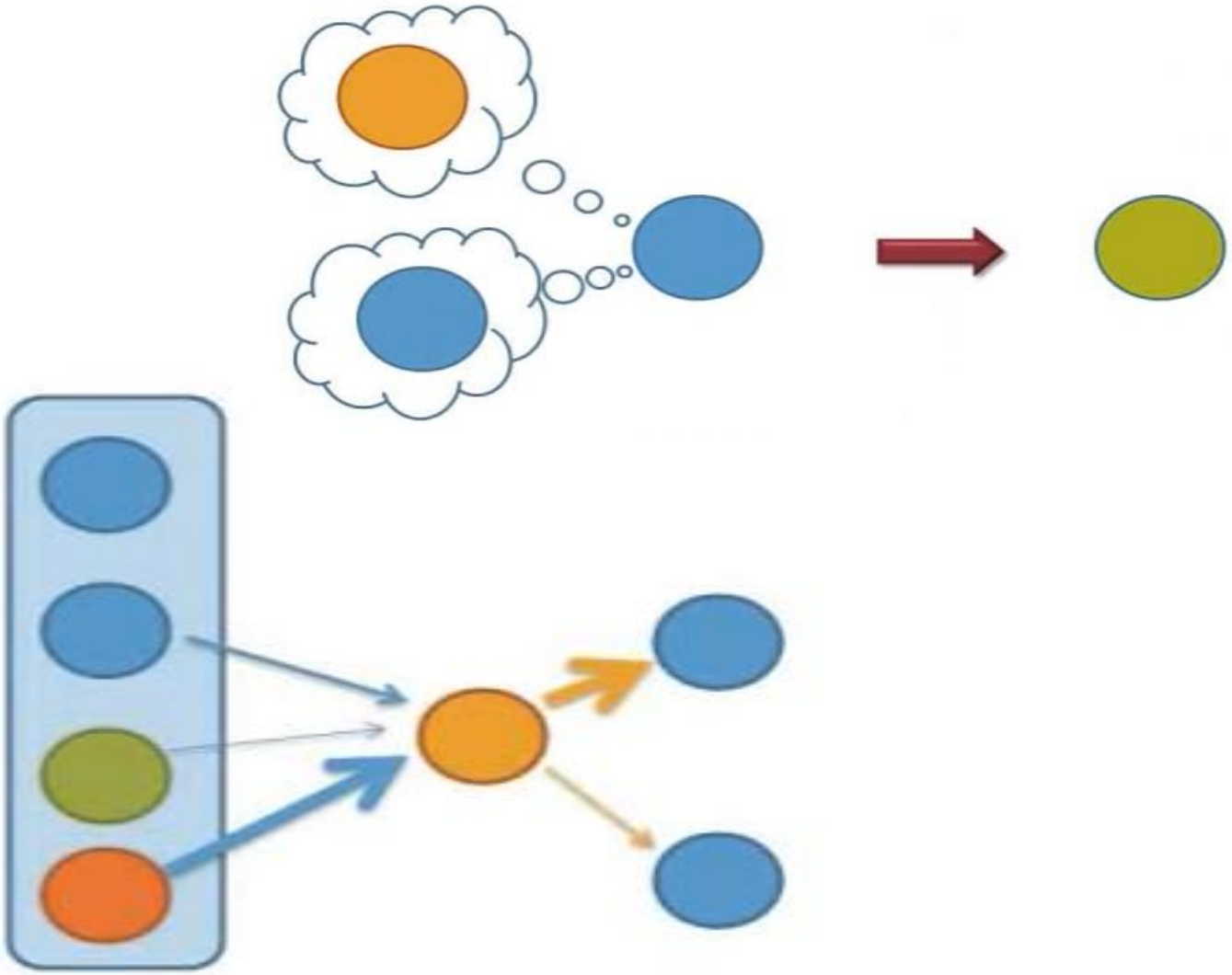


www.chandanverma.com

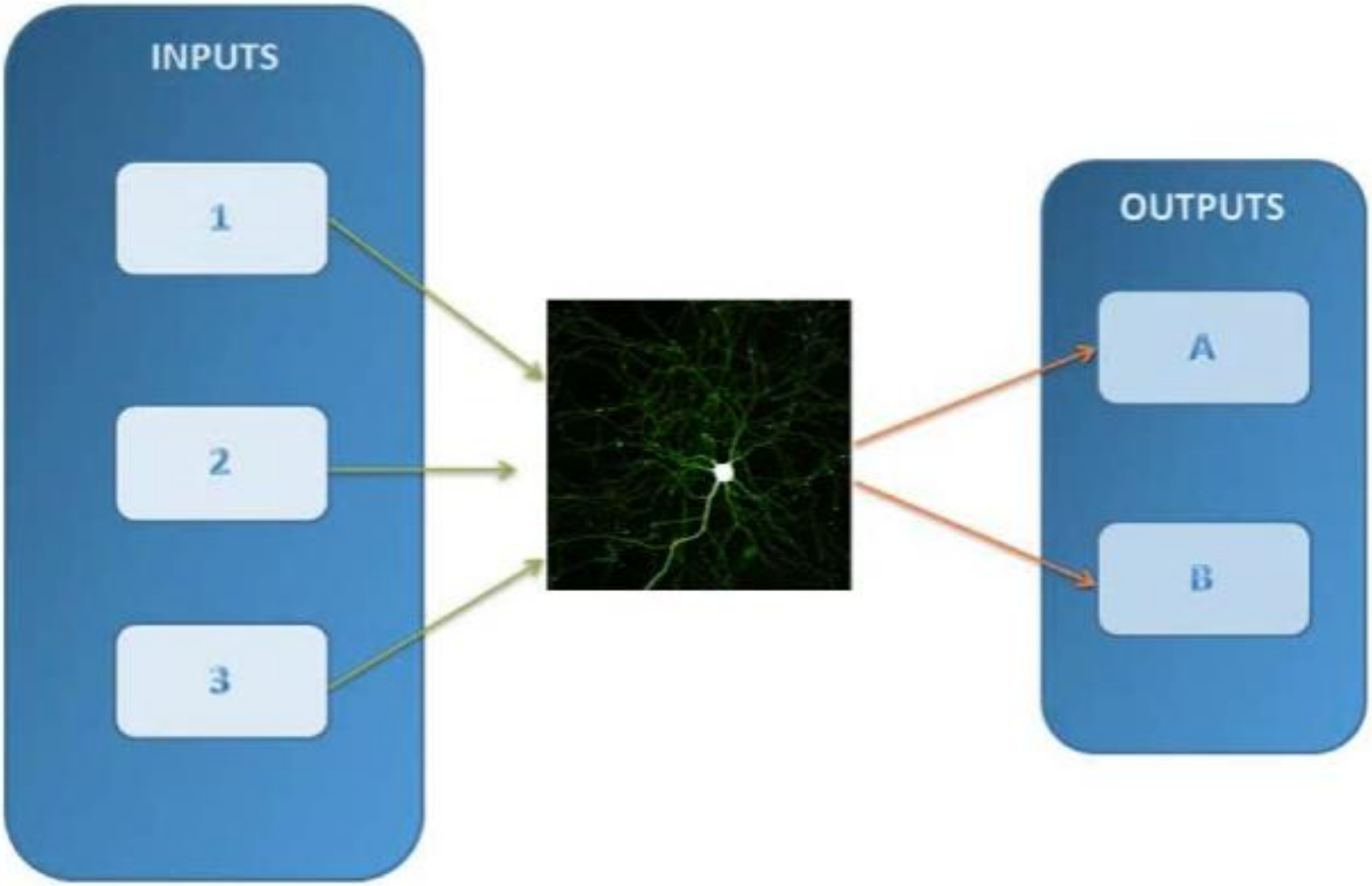
Decision



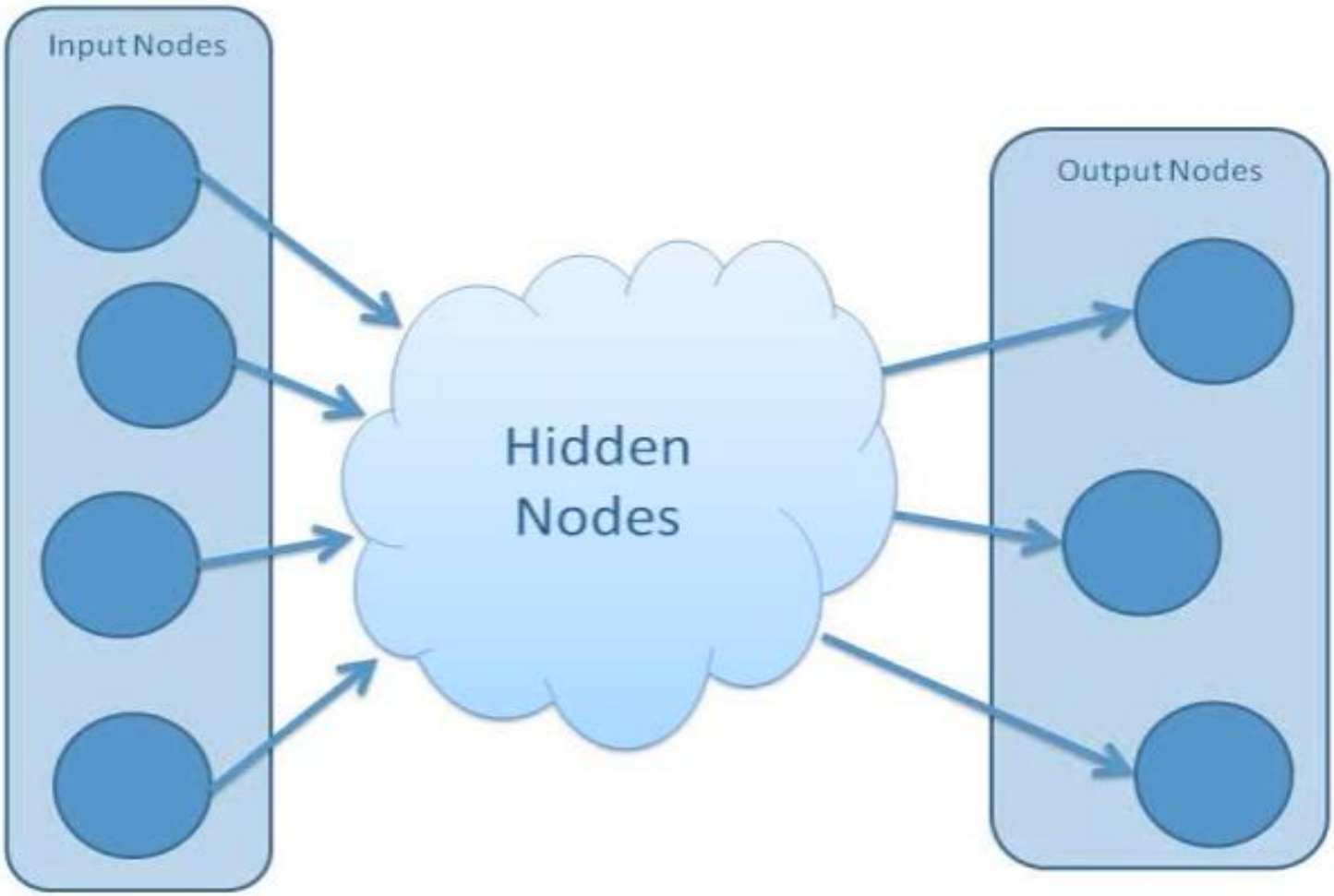
www.chandanverma.com



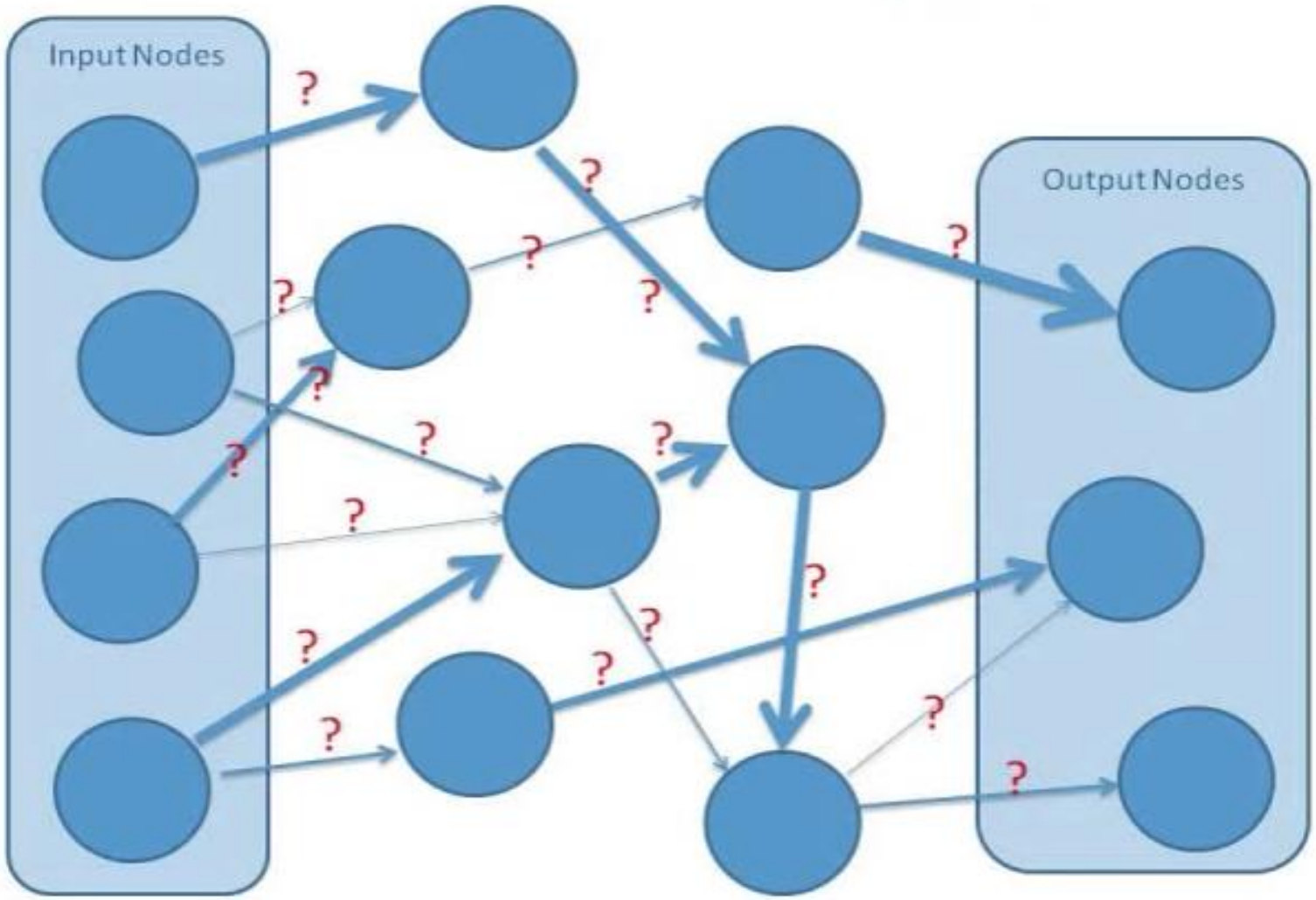
Using Connection Weight and Transfer Neural Network takes Input And Produce Output.



www.chandanverma.com

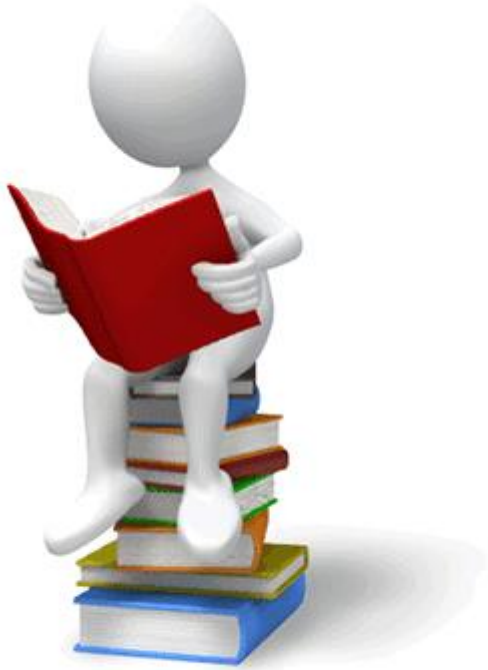


They Do most of work and get least Credit



How Connection Weight Determent????

How We Solve Any Problem



Training By Example

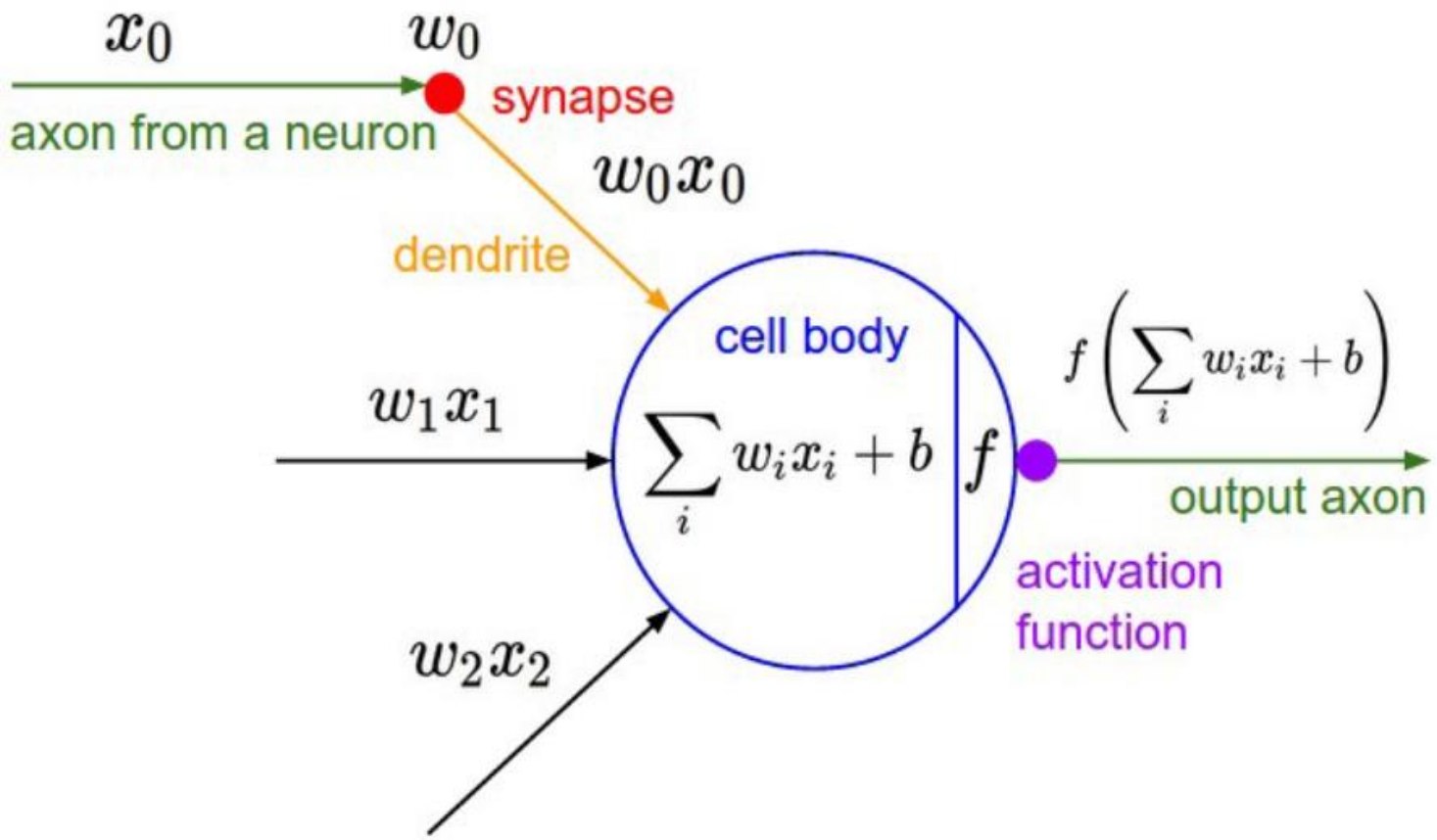
NEURAL NETWORK

A Neural Network is a massively parallel distributed Processor that has a natural Propensity for **Storing Experimental Knowledge** and making it available for Use.

ANN-Training is Done by Example.

A neural network, is a collection of layers that transform the input in some way to produce an output

Mathematical model of a neuron



Data, represented by **x0,** travels through the connections between the neurons.
The strength of the connections are represented by their **weights** (**w0x0, w1x1**, etc).
If the signal is strong enough, it fires the neuron via its "**activation function**" and makes the neuron "**active**."

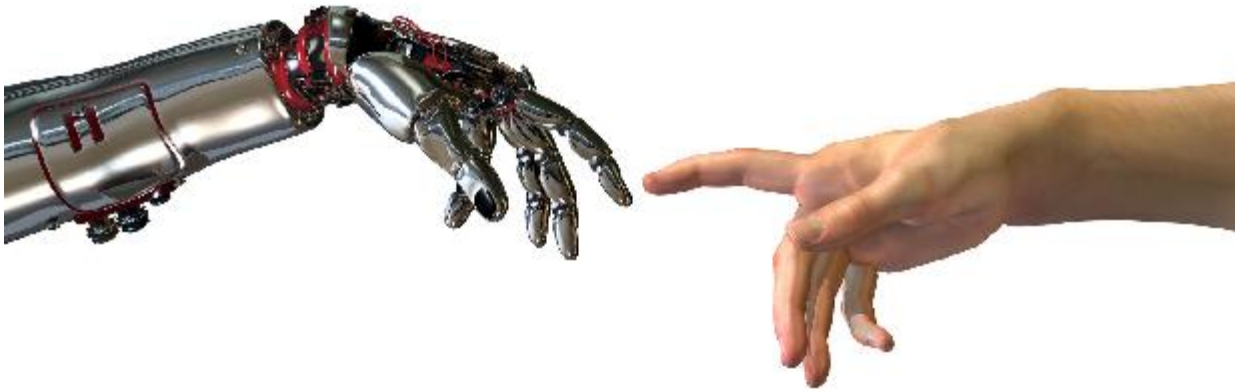
INTELLIGENCE

- Perception (Concept or Senses)
- Problem Solving
- Reasoning
- Learning
- Control Uncertainties

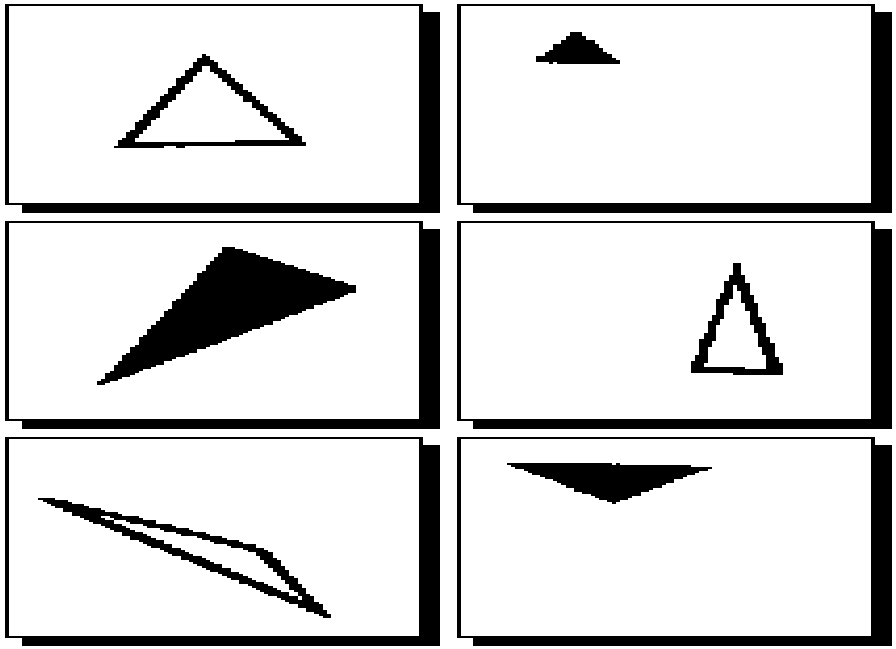
Intelligence is defined as general cognitive problem-solving skills.

Artificial Intelligence

- Artificial Intelligence is a attribute of machines which gives it ability to mimic the Human thought process.



Brain	**10^4 Synapses**	**10-6 size**	**30W**	**100HZ**	**LEARN YES**
Computer	10^8 Transistor	10-6 size	30W	10^9 HZ	LITTLE (NO)



www.chandanverma.com

Perceptrons

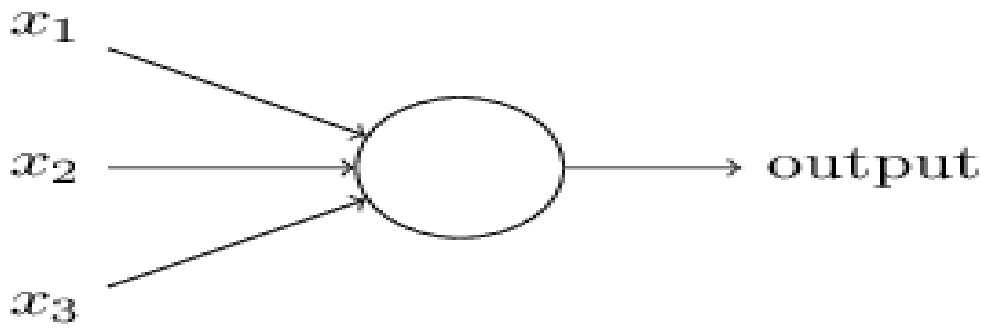
A simple neural network architecture is called perceptron.

The perceptron consists of two types of nodes: input nodes, which are used to represent the input attributes, and an output node, which is used to represent the model output.

In a perceptron, each input node is connected via a weighted link to the output Node.

This is one layer Network and use in The case of liner separable

This is a type of artificial neuron



$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$

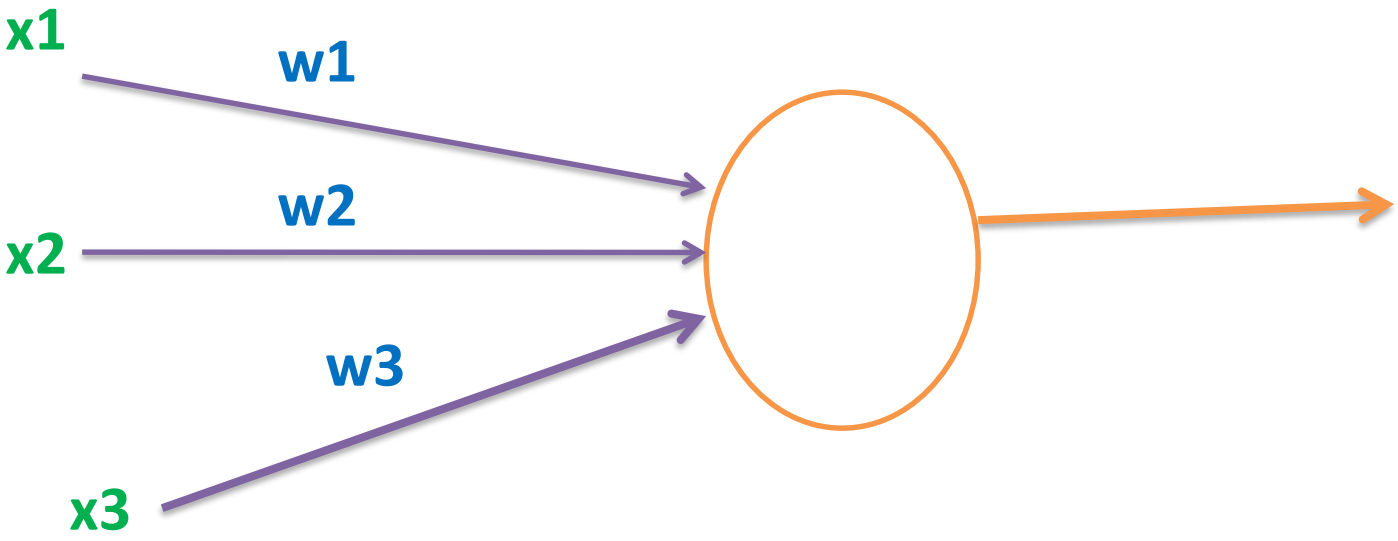
www.chandanverma.com

Example

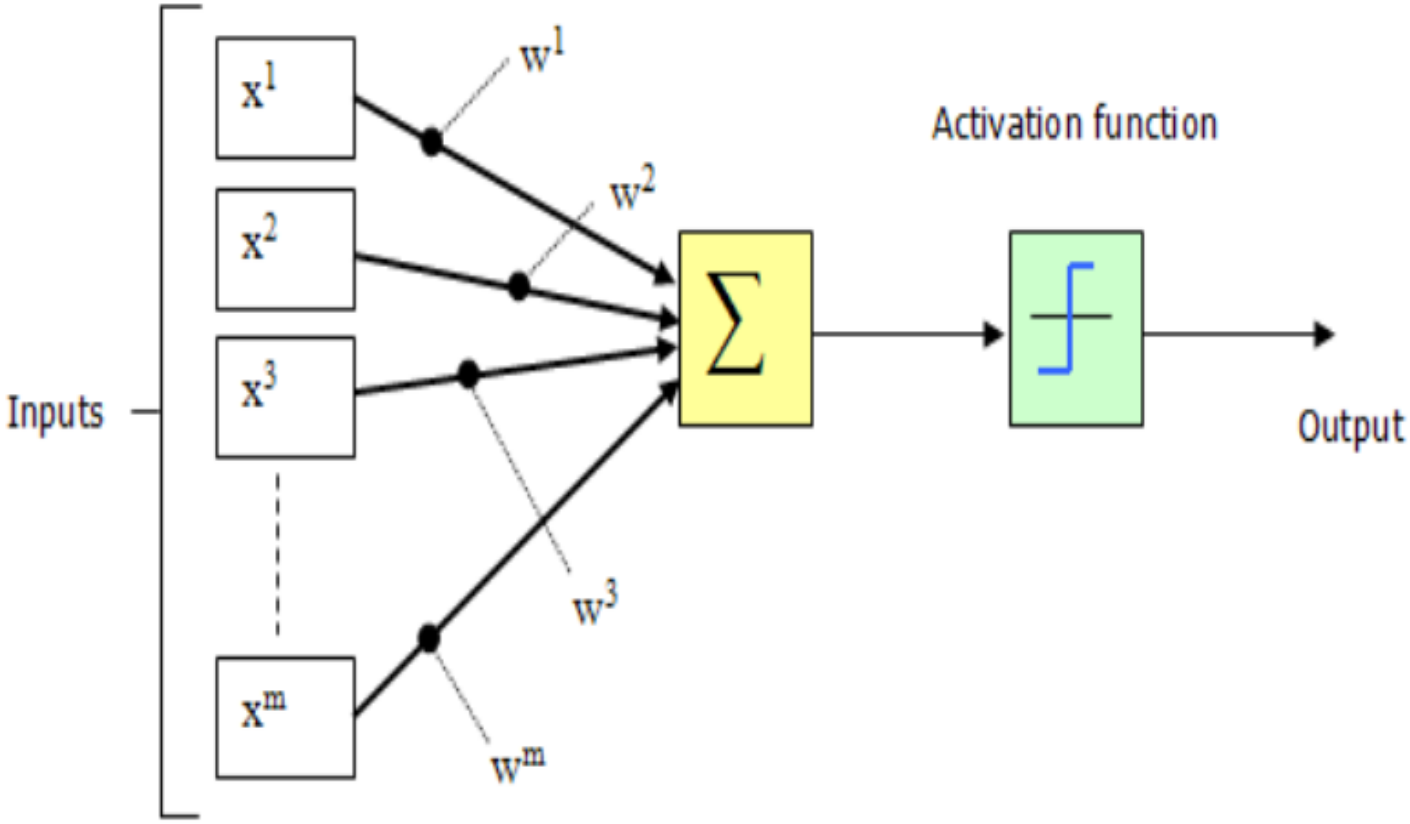
Suppose the weekend is coming up, and you've heard that there's going to be a cheese festival in your city. You like cheese, and are trying to decide whether or not to go to the festival.

1.Is the weather good?
2.Does your boyfriend or girlfriend want to accompany you?
3.Is the festival near public transit? (You don't own a car).

www.chandanverma.com



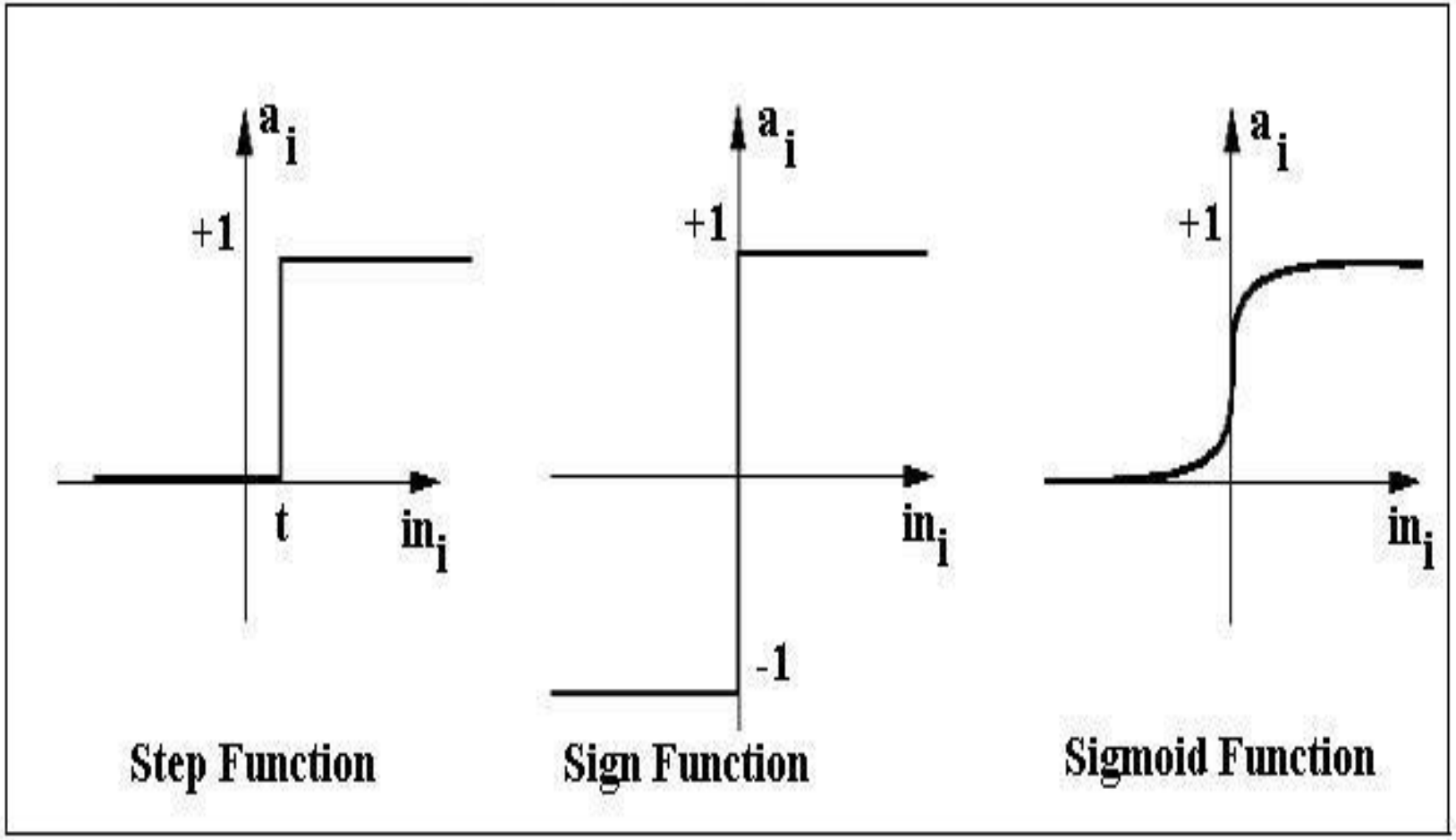
x1	Weather	X2	Friend	X3	Transit
0	Not Good	0	Not GO	0	No
1	Good	1	Go	1	yes


Inputs
x1
x2
x3
xm
w1
w2
w3
wm
Activation function
Output

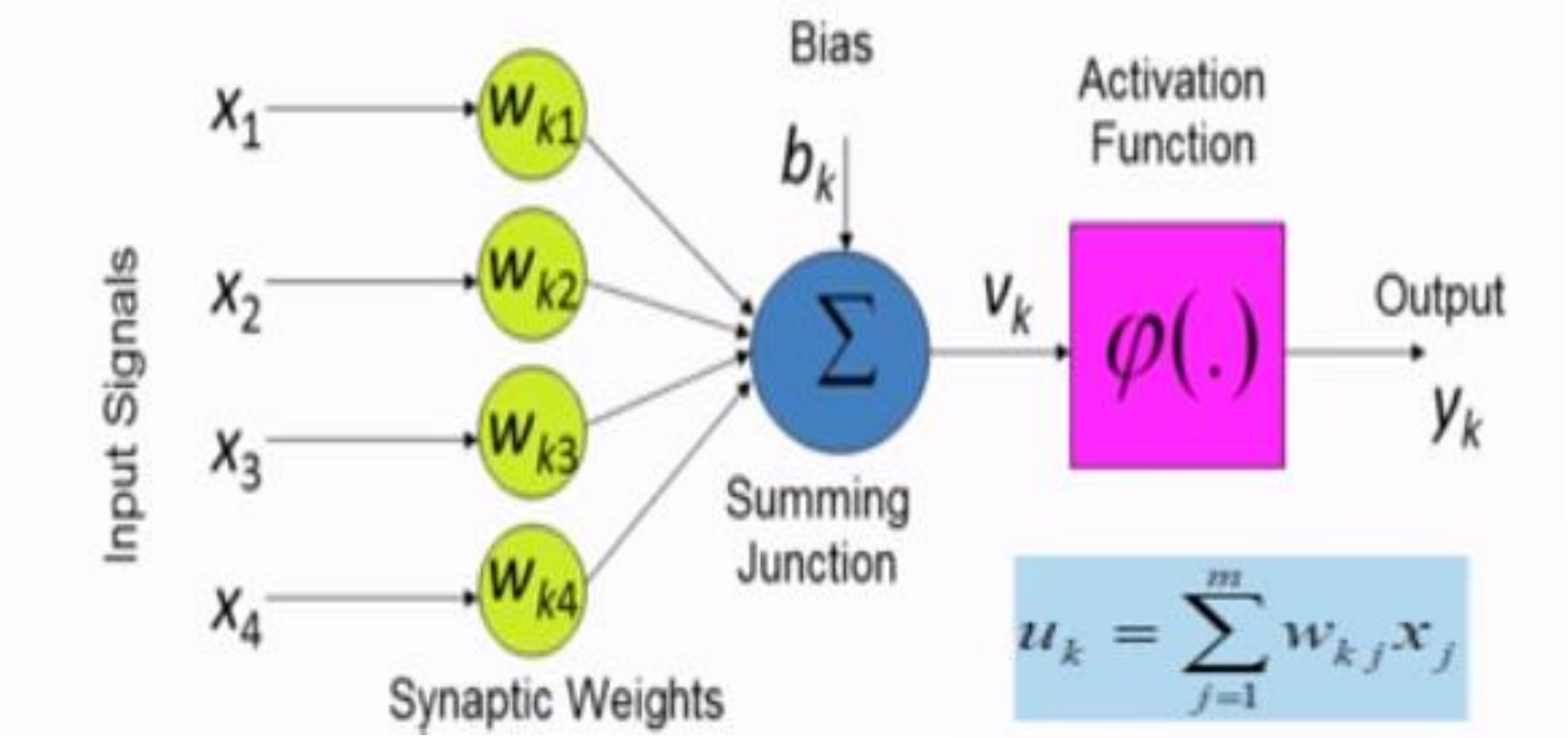
Basic Element Of ANN

- Input and Output Nodes
- Weights
- Activation Function

- Type of Neural Network
- Type of Activation Function
- Type of Learning Rule or Training

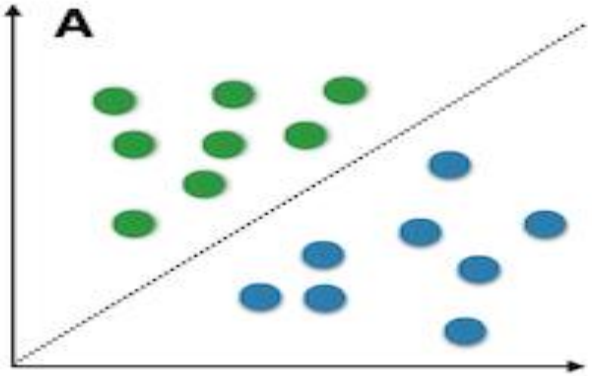

a i
+1
t
in i
Step Function
a i
+1
in i
-1
Sign Function
a i
+1
in i
Sigmoid Function

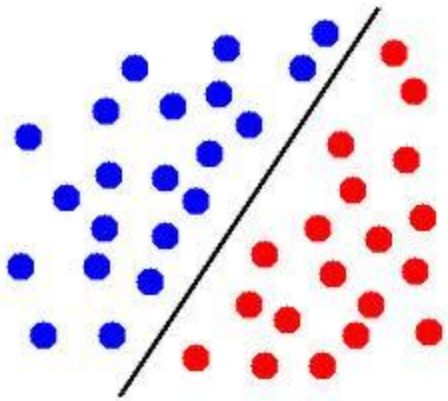
SINGLE-LAYER NEURAL NETWORKS (PERCEPTRONS)



Input 1 (x_1) = 0.6
Input 2 (x_2) = 1.0
Weight 1 (w_1) = 0.5
Weight 2 (w_2) = 0.8
Threshold = 1.0

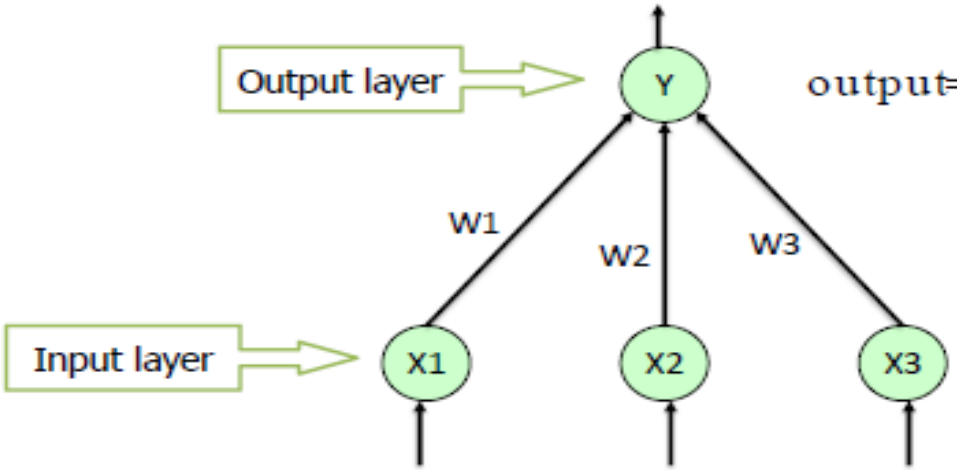
$$x_1w_1 + x_2w_2 = (0.6 \times 0.5) + (1 \times 0.8) = 1.1$$





Linearly separable

www.chandanverma.com



$$\text{output} = \begin{cases} 1 & \text{if } \sum w_i x_i > \theta \\ 0 & \text{otherwise} \end{cases}$$

The single layer perceptron does not have a priori knowledge, so the initial weights are assigned randomly. SLP sums all the weighted inputs and if the sum is above the threshold (some predetermined value), SLP is said to be activated (output=1).

Output

$$w_1 x_1 + w_2 x_2 + \ldots + w_n x_n > \theta \Rightarrow 1$$

$$w_1 x_1 + w_2 x_2 + \ldots + w_n x_n \leq \theta \Rightarrow 0$$

The input values are presented to the perceptron, and if the predicted output is the same as the desired output, then the performance is considered satisfactory and no changes to the weights are made. However, if the output does not match the desired output, then the weights need to be changed to reduce the error.

Perceptron Weights Adjustment

$$\Delta w = \eta \times d \times x$$

$d \rightarrow$ Predicted output - Desired output

$\eta \rightarrow$ Learning rate, usually less than 1

$x \rightarrow$ Input data

www.chandanverma.com

The rule

The output node has a "threshold" t.

Rule: If summed input ≥ t, then it "fires“ (output y = 1).

Else (summed input < t) it doesn't fire (output y = 0).

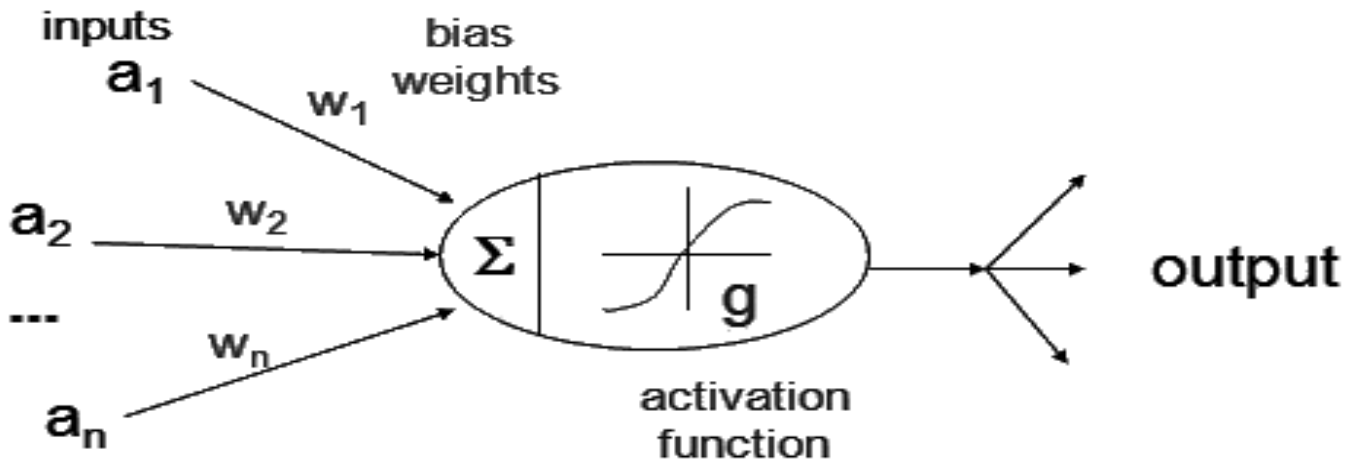
$$\text{if } \sum_i w_i I_i \geq t \text{ then } y = 1$$

$$\text{else } (\text{ if } \sum_i w_i I_i < t \ ) \text{ then } y = 0$$

www.chandanverma.com

Learning Rule

- Update The Weight and Bias to minimize the error.
- The learning Rate helps us control how much we change the weight and bias.



Let input x = (I_1, I_2, .., I_n) where each I_i = 0 or 1. And let output y = 0 or 1.

Given input x = (I_1, I_2, .., I_n). Perceptron produces output y. We are told correct output **O**.
If we had wrong answer, change w_i's and t, otherwise do nothing.

Given input x = (I_1, I_2, .., I_n). Perceptron produces output y. We are told correct output **O**.
For all i:

$$w_i := w_i + C\,(O-y)\,I_i$$

$$t := t - C\,(O-y)$$

www.chandanverma.com

Suppose you have the following classification problem and would like to solve it with a single vector input, two-element perceptron network.

$$\left\{\mathbf{p}_1 = \begin{bmatrix}2\\2\end{bmatrix}, t_1 = 0\right\}\left\{\mathbf{p}_2 = \begin{bmatrix}1\\-2\end{bmatrix}, t_2 = 1\right\}\left\{\mathbf{p}_3 = \begin{bmatrix}-2\\2\end{bmatrix}, t_3 = 0\right\}\left\{\mathbf{p}_4 = \begin{bmatrix}-1\\1\end{bmatrix}, t_4 = 1\right\}$$

$\mathbf{W}(0)=[0 \quad 0] \quad b(0)=0$

Start by calculating the perceptron's output a for the first input vector $\mathbf{p}_1$, using the initial weights and bias.

$\alpha = hardlim(\mathbf{W}(0)\mathbf{p}1 + b(0))$

$= hardlim([0\ 0][2\ 2]' + 0)$

$= hardlim(0) = 1$

The output a does not equal the target value t_1, so use the perceptron rule to find the incremental changes to the weights and biases based on the error.

$e = t1 - \alpha = 0 - 1 = -1$

$\Delta \mathbf{W} = e\mathbf{p}^T 1 = (-1)[2\ 2] = [-2\ -2]$

$\Delta b = e = (-1) = -1$

new weights and bias using the perceptron update rules.

$\mathbf{W}new = \mathbf{W}old + \mathbf{ep}^T$

$= [0\ 0] + [-2\ -2] = [-2\ -2] = \mathbf{W}(1)$

$bnew = bold + e =$

$0 + (-1) = -1 = b(1)$

next input vector, $\mathbf{p}_2$. The output is calculated below.

$\alpha = hardlim(\mathbf{W}(1)\mathbf{p}2 + b(1))$

$= hardlim([-2\ -2][1\ -2]\ -1)$

$= hardlim(1) = 1$

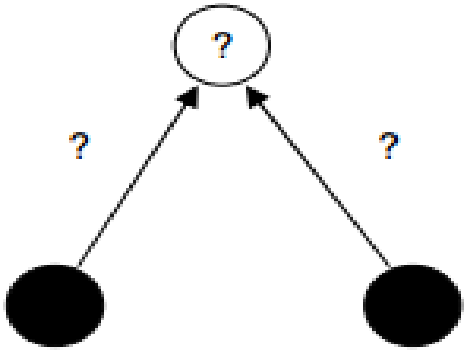
On this occasion, the target is 1, so the error is zero. Thus there are no changes in weights or bias, so $\mathbf{W}(2) = \mathbf{W}(1) = [-2\ -2]$ and $b(2) = b(1) = -1$.

IMPLEMENTATION OF LOGICAL NOT, AND, AND OR

NOT	
in	out
0	1
1	0

AND		
in_1	in_2	out
0	0	0
0	1	0
1	0	0
1	1	1

OR		
in_1	in_2	out
0	0	0
0	1	1
1	0	1
1	1	1



AND

	T	F
T	T	F
F	F	F

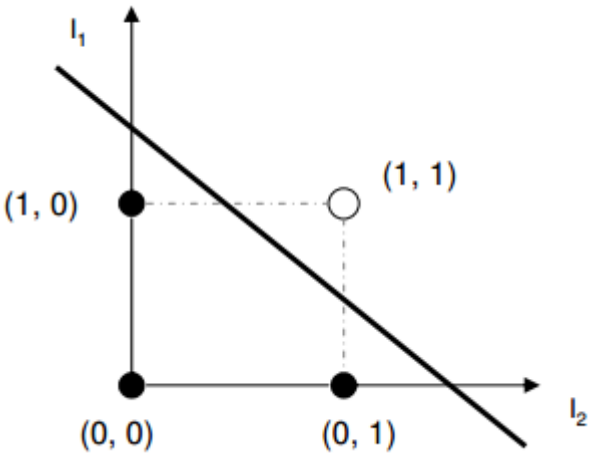
OR

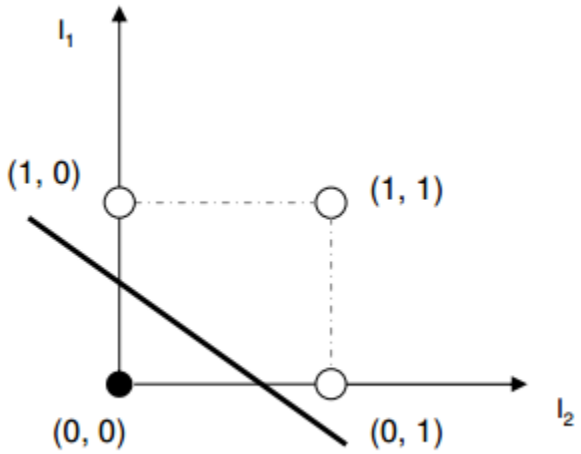
	T	F
T	T	T
F	T	F

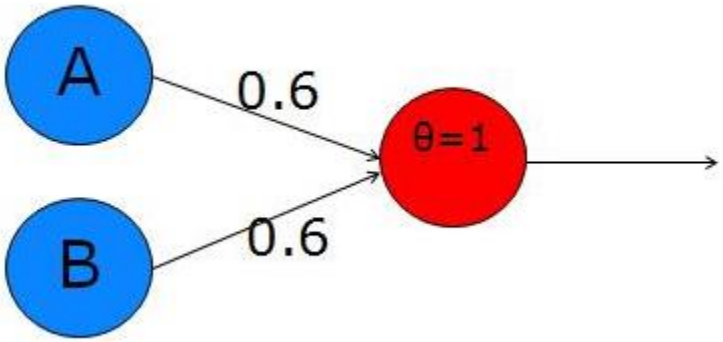
XOR

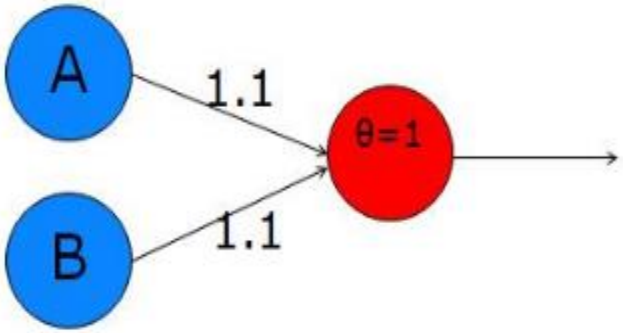
	T	F
T	F	T
F	T	F

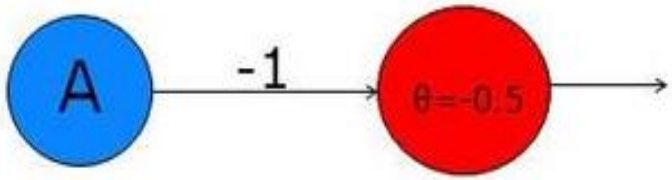
XOR* is the equivalent of *OR* and *NOT AND


I1
(1, 0)
(1, 1)
(0, 0)
(0, 1)
I2


I1
(1, 0)
(1, 1)
(0, 0)
(0, 1)
I2


A
B
0.6
0.6
θ=1

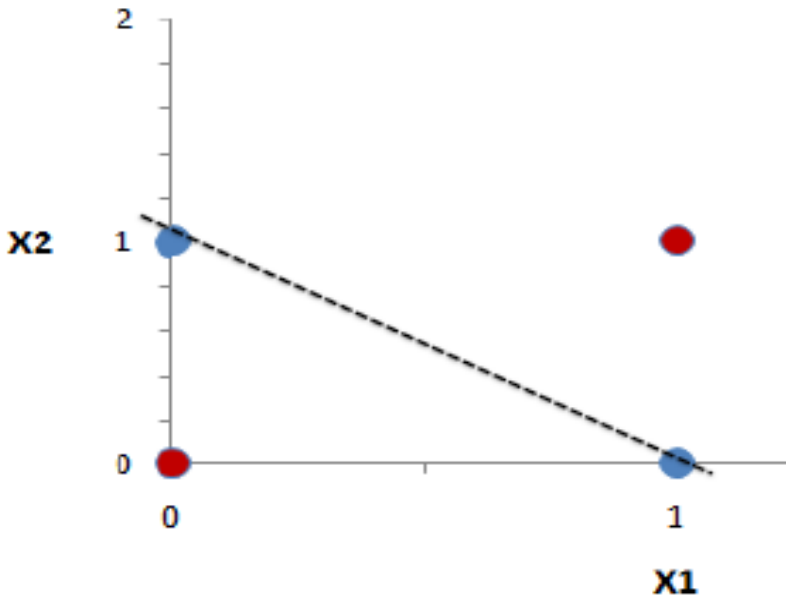

A
B
1.1
1.1
θ=1


A
-1
θ=-0.5

Xor Function

INPUT		OUTPUT
A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

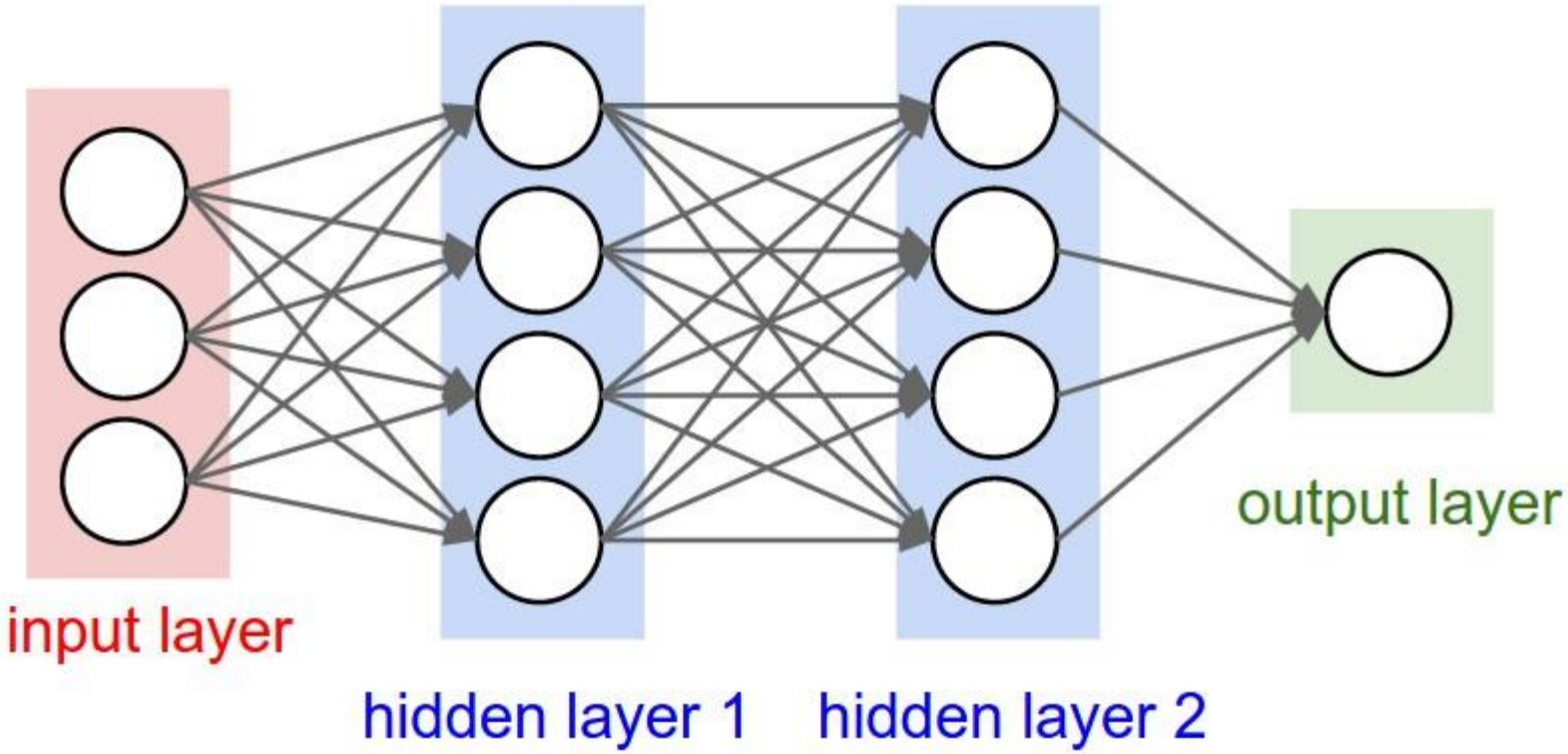
Truth Table



It is Not Linear separable.

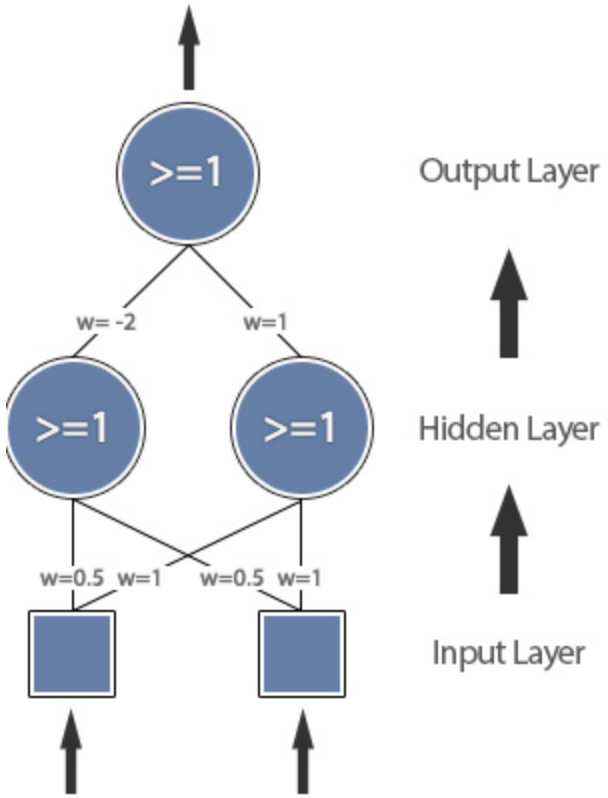
It can not model using single Perceptron .

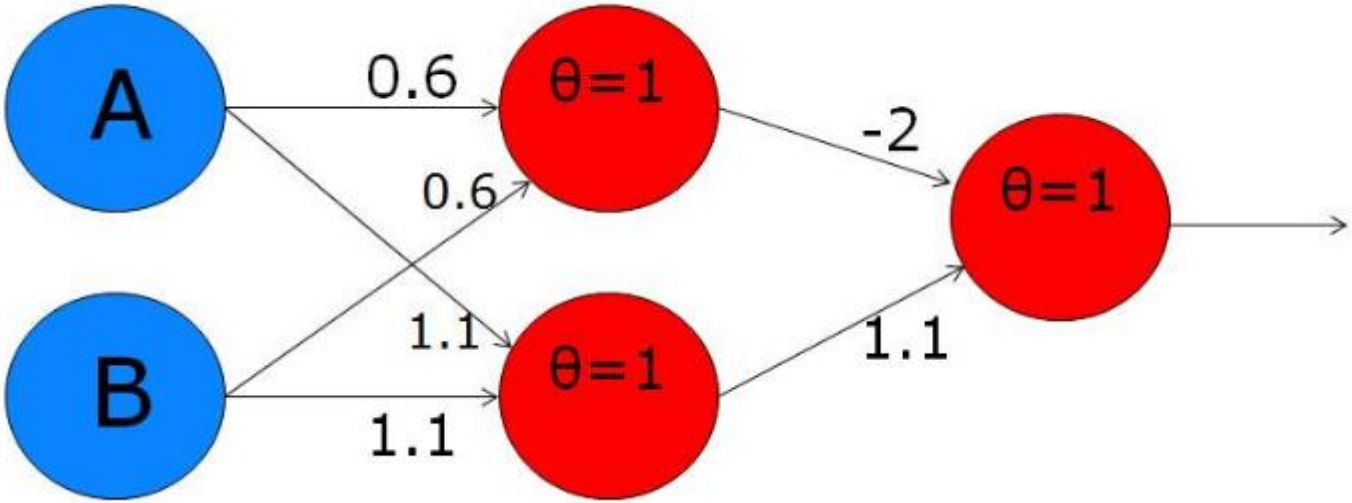
Multi-layer Perceptron



www.chandanverma.com

XOR

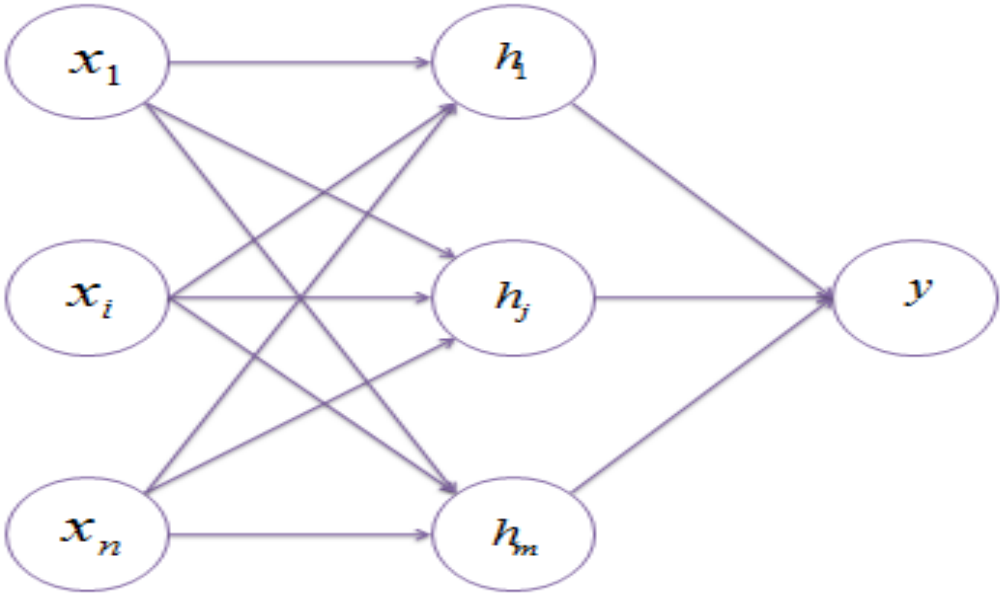

>=1
Output Layer
w= -2
w=1
>=1
>=1
Hidden Layer
w=0.5
w=1
w=0.5
w=1
Input Layer

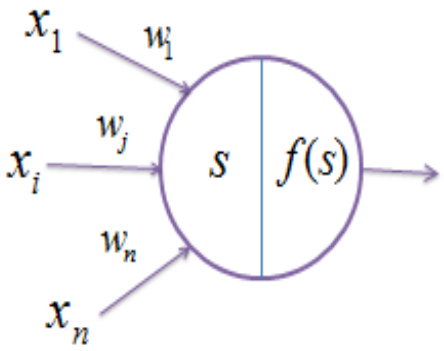

A
B
0.6
0.6
1.1
1.1
θ=1
θ=1
-2
θ=1
1.1

Multi-layer Perceptron

A multi-layer perceptron (**MLP**) has the same structure of a single layer perceptron with one or more hidden layers.
The backpropagation algorithm consists of two phases:
1.The forward phase where the activations are propagated from The input to the output layer.

2. Backward phase, where the error between the observed actual and the requested nominal value in the output layer is propagated backwards in order to modify the weights and bias values.





Summation

$$s = \sum w \cdot x$$

Transformation

$$f(s) = \frac{1}{1+e^{-s}}$$

1. Error in any ***output*** neuron

$$d_o = y \times (1-y) \times (t-y)$$

2. Error in any ***hidden*** neuron

$$d_i = y_i \times (1-y_i) \times (w_i \times d_o)$$

3. Change the ***weights***

$$\Delta w = \eta \times d \times x$$

Calculate the output of a simple neuron

1. Define neuron parameters

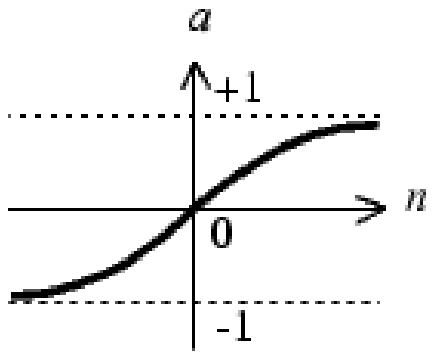
2. Define input vector

3. Calculate neuron output

4. Plot neuron output over the range of inputs

Transfer functions convert a neural network layer's net input into its net output.

tansig Symmetric sigmoid transfer function.
A = tansig(N) .
N:S-by-Q matrix of net input (column) vectors
returns A, the S-by-Q matrix of N's elements squashed into [-1 1].



$a = tansig(n)$

Tan-Sigmoid Transfer Function

f

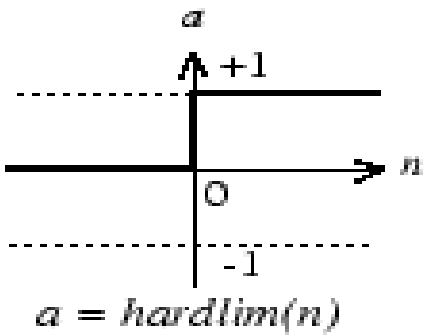
```
n = -5:0.1:5;
a = tansig(n);
plot(n,a)
```

hardlim:Hard-limit transfer function

A = hardlim(N)

N S-by-Q matrix of net input (column) vectors.

returns A, the S-by-Q Boolean matrix with 1s where N ≥ 0.



Hard-Limit Transfer Function

```
n = -5:0.1:5;
a = hardlim(n);
plot(n,a)
```

www.chandanverma.com

Meshgrid: Rectangular grid in 2-D and 3-D space

[X,Y] = meshgrid(1:3,10:14).

surf:3-D shaded surface plot

[X,Y] = meshgrid(-2:.2:2, -2:.2:2);

```
[X,Y] = meshgrid(-2:.2:2, -2:.2:2);
 Z = X .* exp(-X.^2 - Y.^2);
surf(X,Y,Z)
```

www.chandanverma.com

```
Define neuron parameters
close all, clear all, clc, format compact
w = [4 -2]
b = -3
func = 'tansig'
p=[2 3]
activation_potential = p*w'+b
neuron_output = feval(func, activation_potential)
```

Plot neuron output over the range of inputs

```
[p1,p2] = meshgrid(-10:.25:10);
z = feval(func, [p1(:) p2(:)]*w'+b );
z = reshape(z,length(p1),length(p2));
plot3(p1,p2,z)
grid on
xlabel('Input 1')
ylabel('Input 2')
zlabel('Neuron output')
```

www.chandanverma.com

Create custom neural network

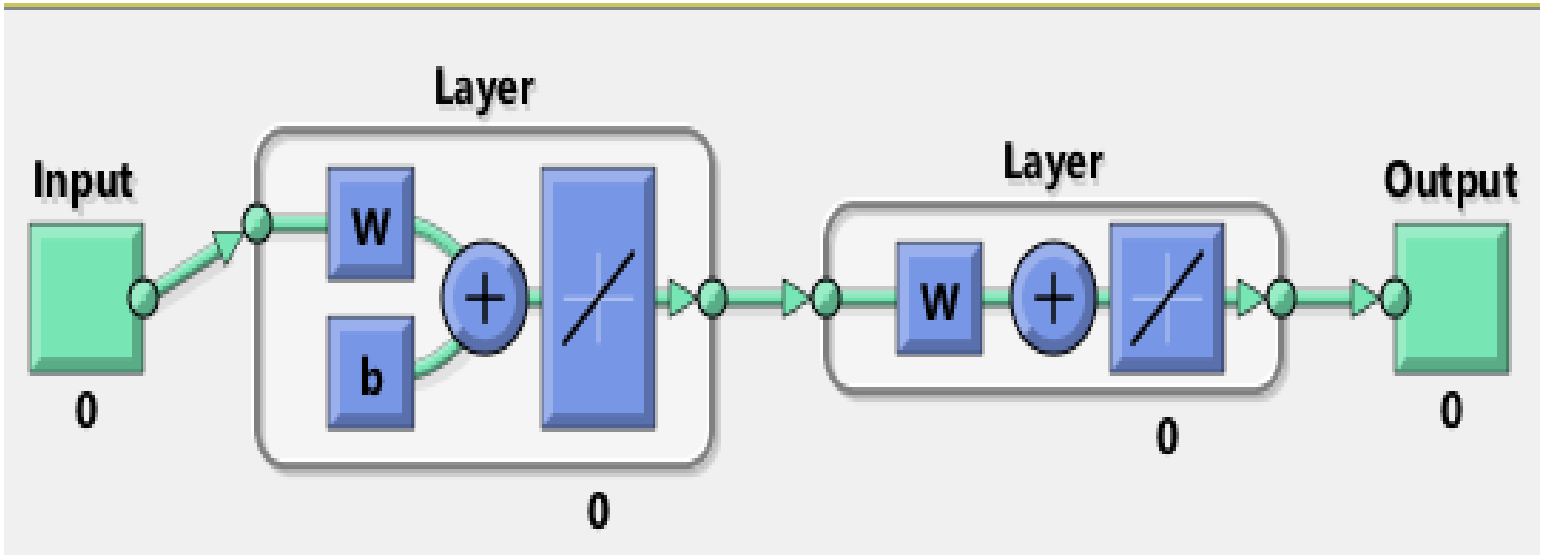
net = network

net =

network(numInputs,numLayers,biasConnect,inputConnect,layerConnect,outputConnect)

View
For displaying the network

```
inputs  = [1:6]' % input vector (6-dimensional pattern)
outputs = [1 2]' % corresponding target output vector
net = network( ...
1,          ... % numInputs,    number of inputs,
2,          ... % numLayers,    number of layers
[1; 0],     ... % biasConnect,  numLayers-by-1 Boolean vector,
[1; 0],     ... % inputConnect, numLayers-by-numInputs Boolean
    matrix,
[0 0; 1 0], ... % layerConnect, numLayers-by-numLayers Boolean
    matrix
[0 1]       ... % outputConnect, 1-by-numLayers Boolean vector
);
view(net)
```



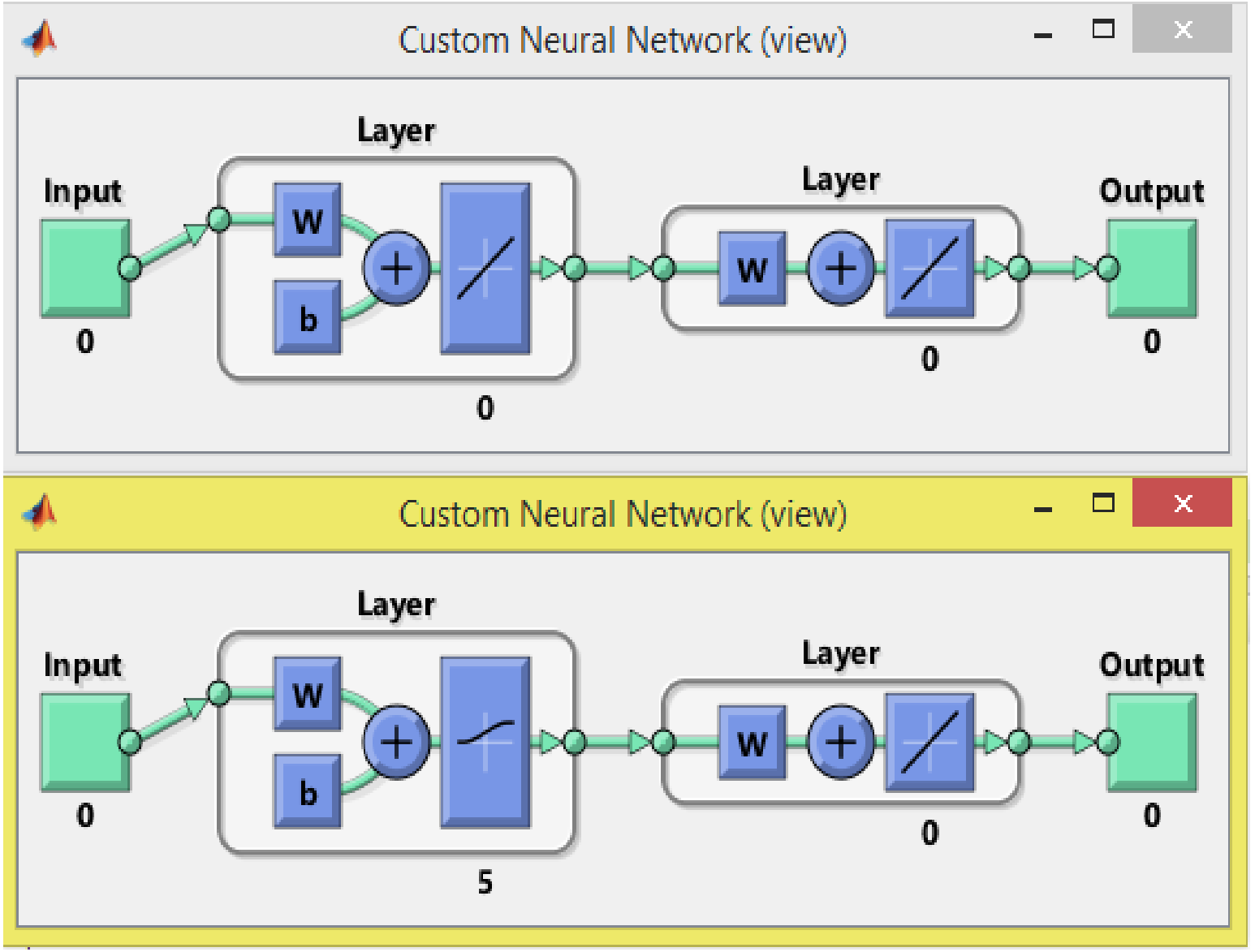
www.chandanverma.com

```
net.adaptFcn
net.initFcn
net.performFcn
```

```
net.inputs{1}
net.layers{1},
net.layers{2}
net.biases{1}
net.IW{1,1}, net.IW{2,1},
net.b{1}
```

```
net.layers{1}.transferFcn = 'tansig';
net.layers{2}.transferFcn = 'logsig';
```

```
net.layers{1}.size = 5;
% hidden layer transfer function
net.layers{1}.transferFcn = 'logsig';
view(net)
```


Custom Neural Network (view)
Input
0
Layer
W
b
0
Layer
W
0
Output
0
Custom Neural Network (view)
Input
0
Layer
W
b
5
Layer
W
0
Output
0

Train net and calculate neuron output

```
% initial network response without training
initial_output = net(inputs)
% network training
net.trainFcn = 'trainlm';
net.performFcn = 'mse';
net = train(net,inputs,outputs);
% network response after training
final_output = net(inputs)
```

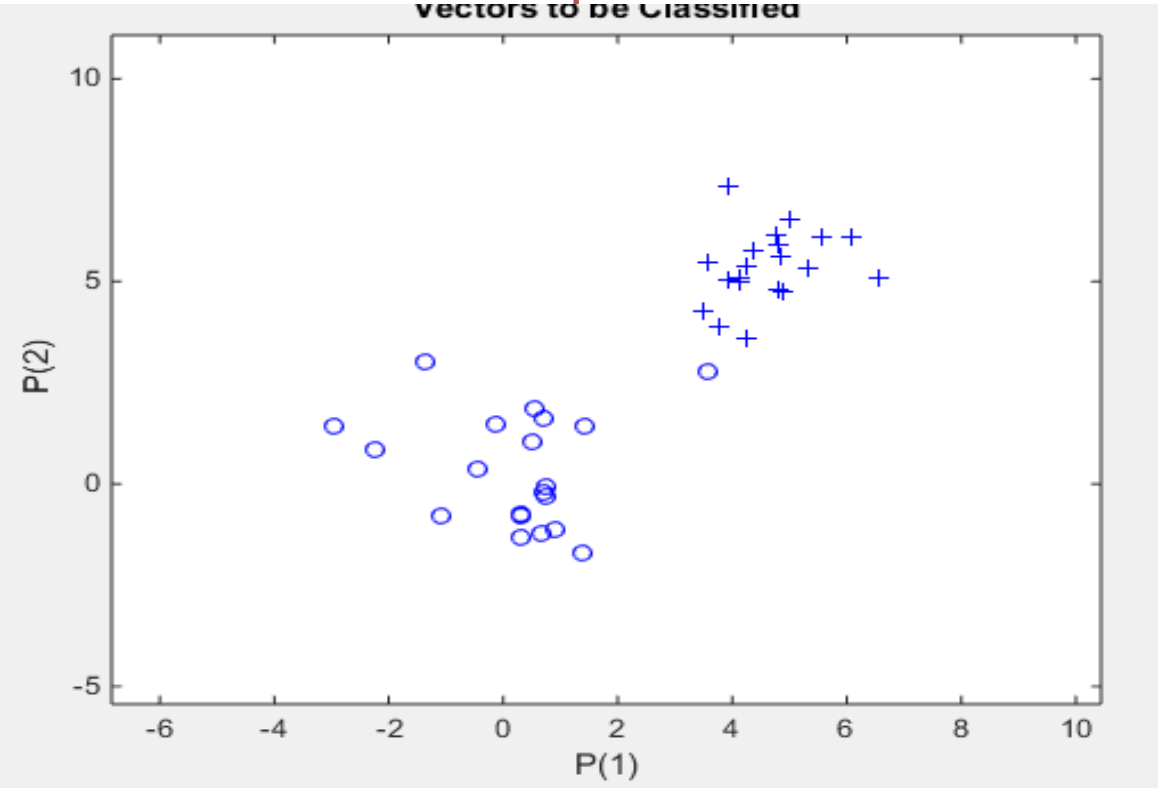
Classification of linearly separable data with a perceptron

Two clusters of data, belonging to two classes, are defined in a 2-dimensional input space. Classes are linearly separable. The task is to construct a Perceptron for the classification of data

1. Define input and output data
2. Create and train perceptron
3. Plot decision boundary

plotpv
Plot perceptron input/target vectors

```
close all, clear all, clc, format compact
% number of samples of each class
N = 20;
offset = 5; % offset for second class
x = [randn(2,N) randn(2,N)+offset]; % inputs
y = [zeros(1,N) ones(1,N)];        % outputs
figure(1)
plotpv(x,y);
```



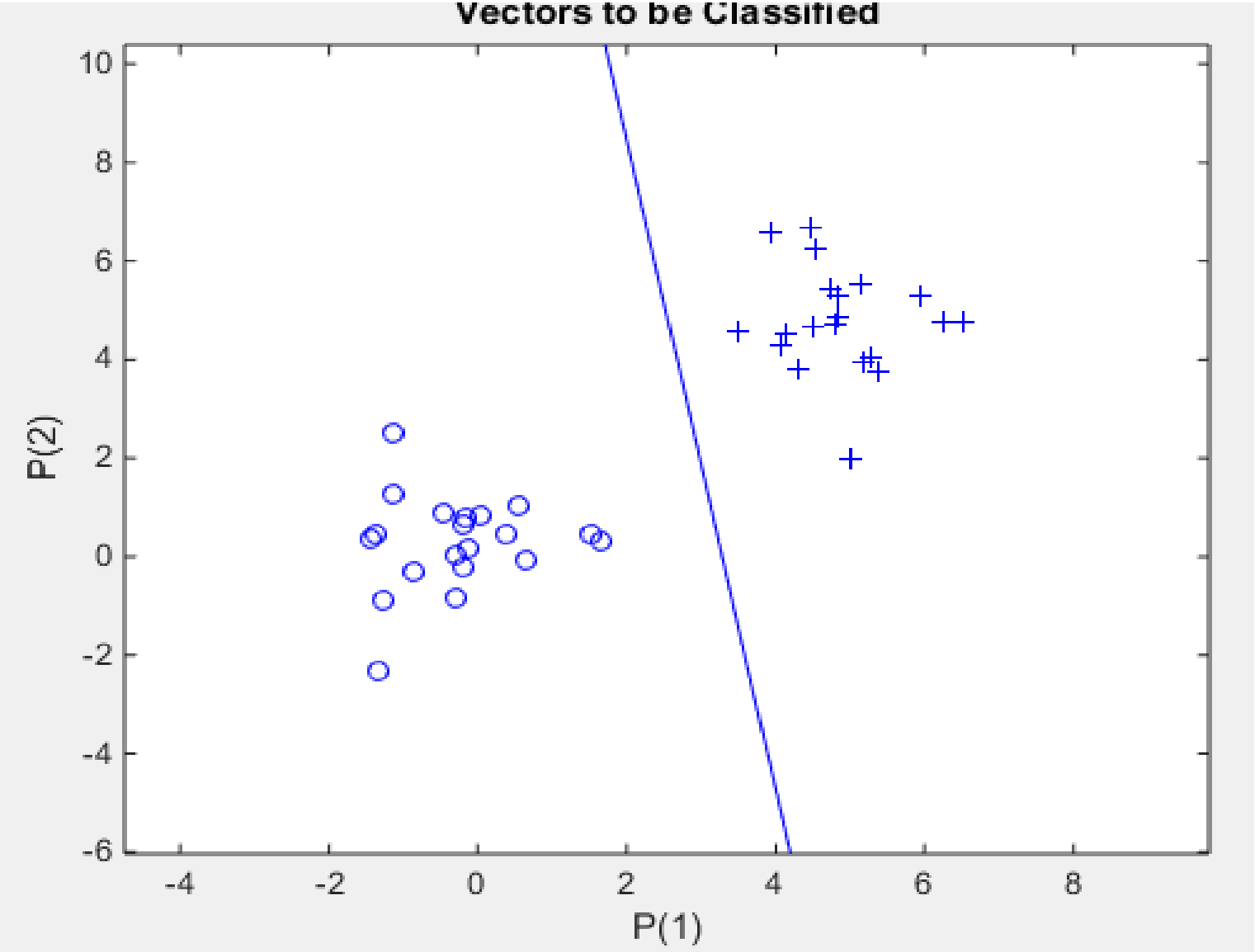
Create and train perceptron

perceptron(hardlimitTF,perceptronLF)

```
net = perceptron;
net = train(net,x,y);
view(net);
```

Plot decision boundary

```
figure(1)
plotpc(net.IW{1},net.b{1});
```


Vectors to be Classified
10
8
6
4
2
0
-2
-4
-6
P(2)
-4
-2
0
2
4
6
8
P(1)