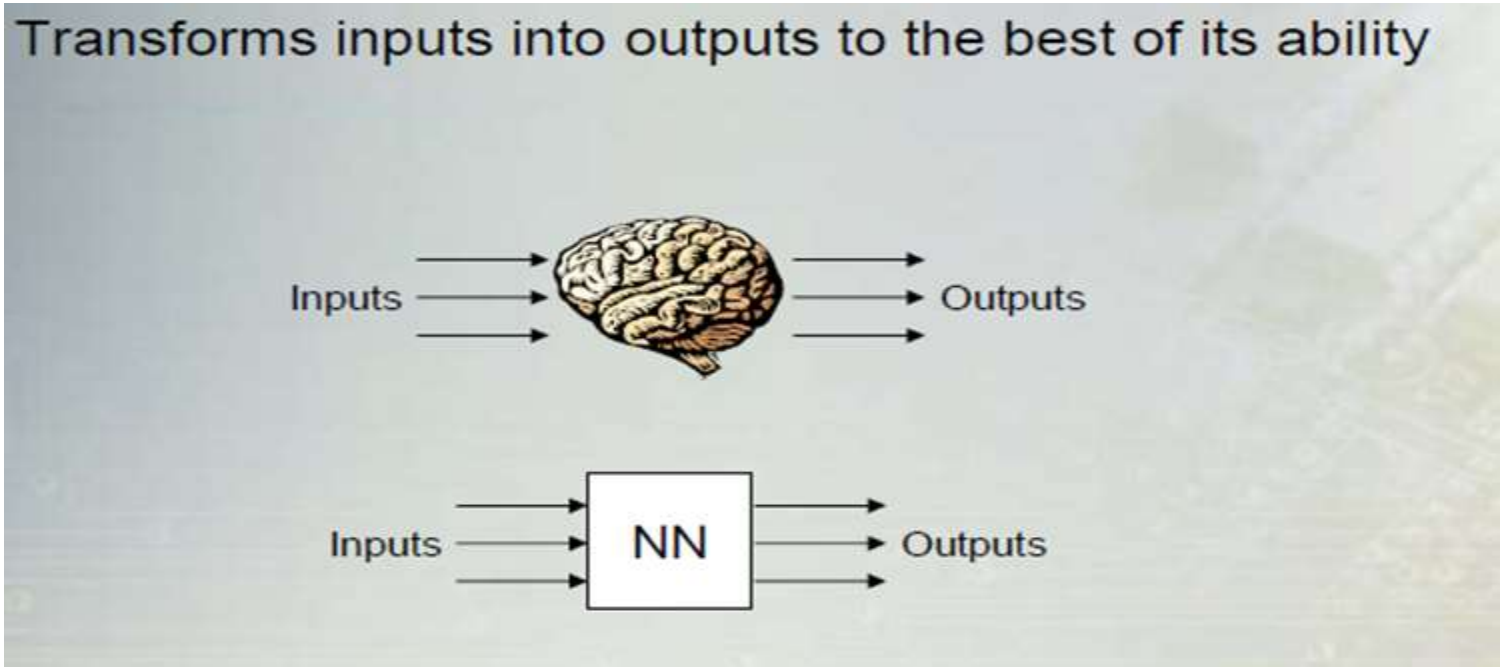
Artificial Neural Networks (ANN)

Neural Networks -Origin

- Inspired by the way biological nervous systems, such as the brain, process information.
- Key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems.



Best learning system known to us?

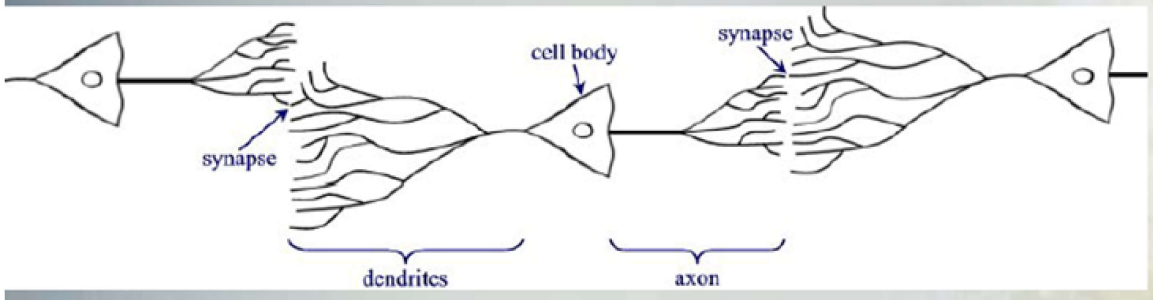
Biological system

?

Artificial system

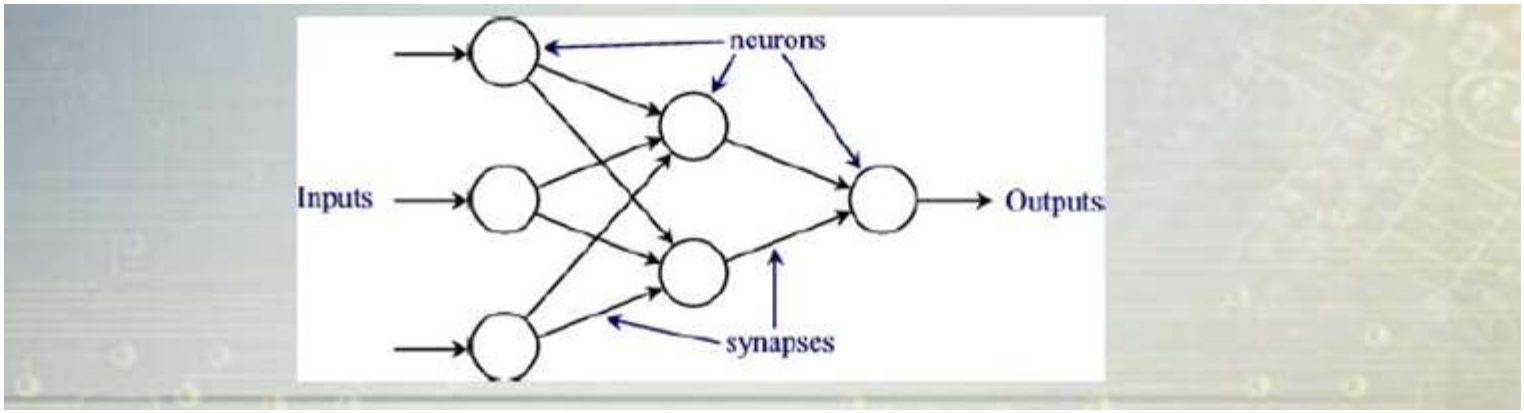
How does the brain work?

- Composed of many "neurons" that co-operate to perform the desired function



In brain a neuron has three principal components:

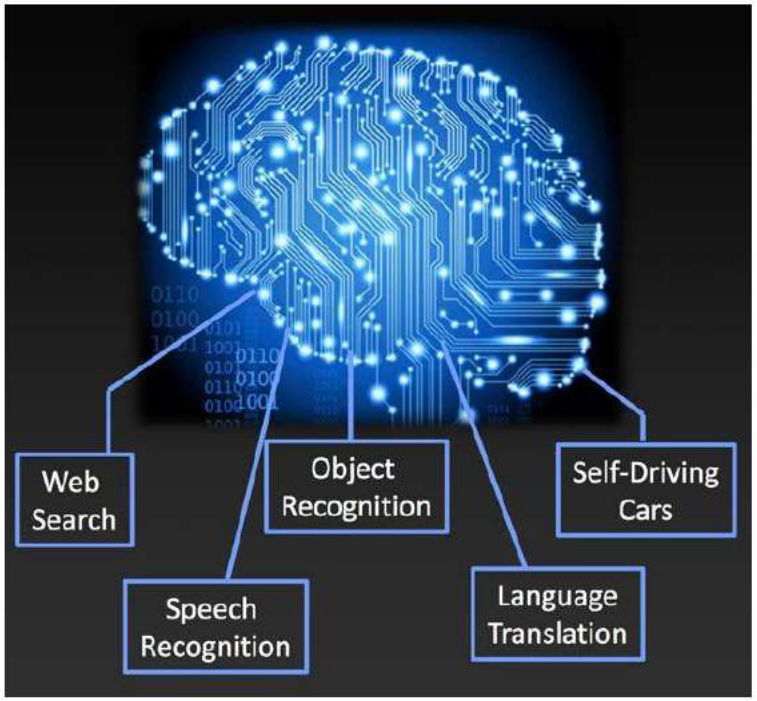
1. **Dendrites:-** that carry electrical signals into the cell body.
2. **Cell Body:-** effectively sums and thresholds these incoming signals.
3. **Axon:-** is a single long fiber that carries the signal from the cell body out to other neurons.
4. The point of contact between an axon of one cell and a dendrite of another cell is called a '**synapse'**



Background: ANN Vs Brain

ANN	Brain
It is simple (few neuron in connection)	It is complex (10^{11} Neurons and 10^{15} connections)
It is dedicated for specific purpose	It is generalized for all purpose
Response time is fast (it may be in Nanosecond)	Response time is slow (it may be in millisecond)
Design is regular	Design is arbitrary
Activities are synchronous	Activities are asynchronous

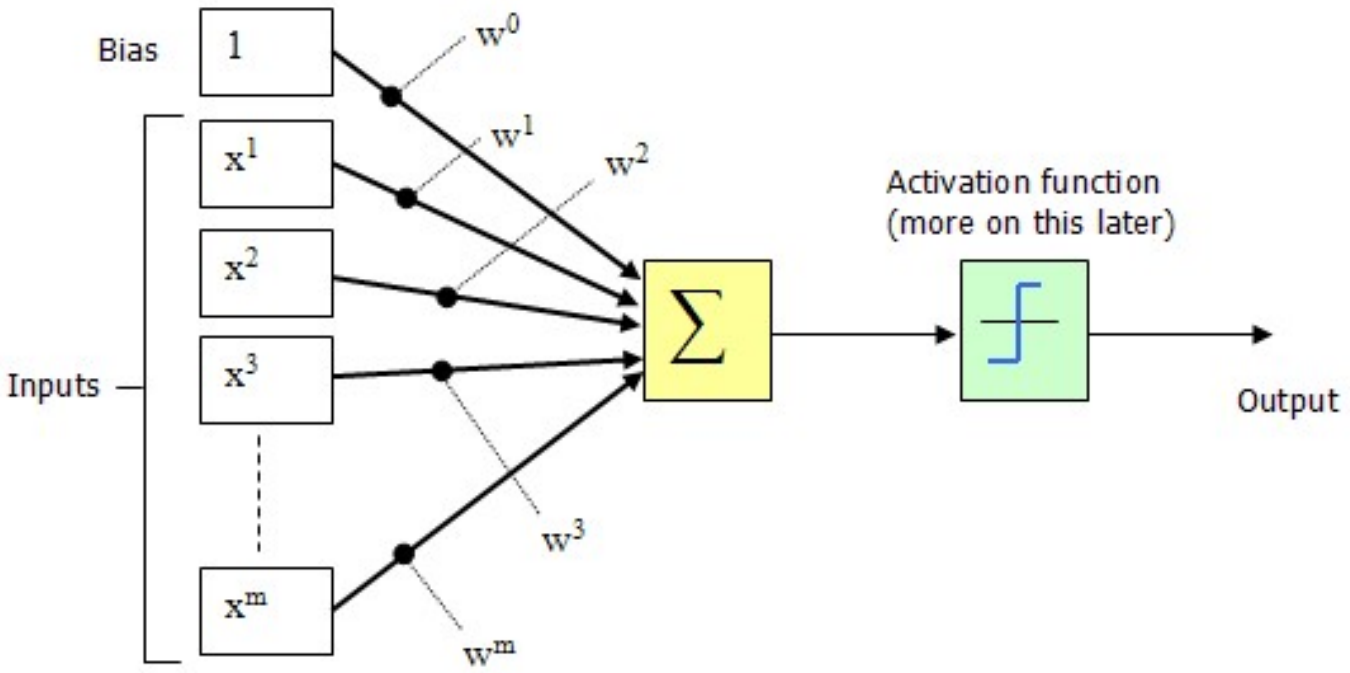
What is a neural network?



- A neural network is a "connectionist" computational system. The computational systems we write are procedural; a program starts at the first line of code, executes it, and goes on to the next, following instructions in a linear fashion. A true neural network does not follow a linear path. Rather, information is processed collectively, in parallel throughout a network of nodes (the nodes, in this case, being neurons).

- One of the key elements of a neural network is its ability to *learn*. A neural network is not just a complex system, but a complex ***adaptive*** system, meaning it can change its internal structure based on the information flowing through it. Typically, this is achieved through the adjusting of *weights*.

Perceptron



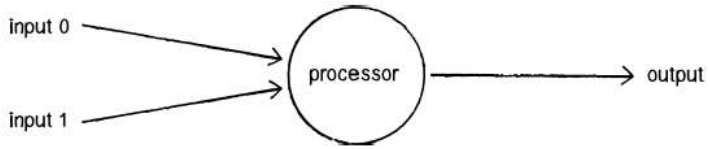
Perceptrons can only model linearly separable functions.

We need to use multi-layer perceptron to tackle non-linear problems.

Perceptron

A Perceptron is a mathematical model of a biological neuron

1. Each input gets scaled up or down by multiplying with weights
2. All signals are summed up
3. Activation



Invented in 1957 by Frank Rosenblatt at the Cornell Aeronautical Laboratory, a perceptron is the simplest neural network possible: a computational model of a single neuron. A perceptron consists of one or more inputs, a processor, and a single output.

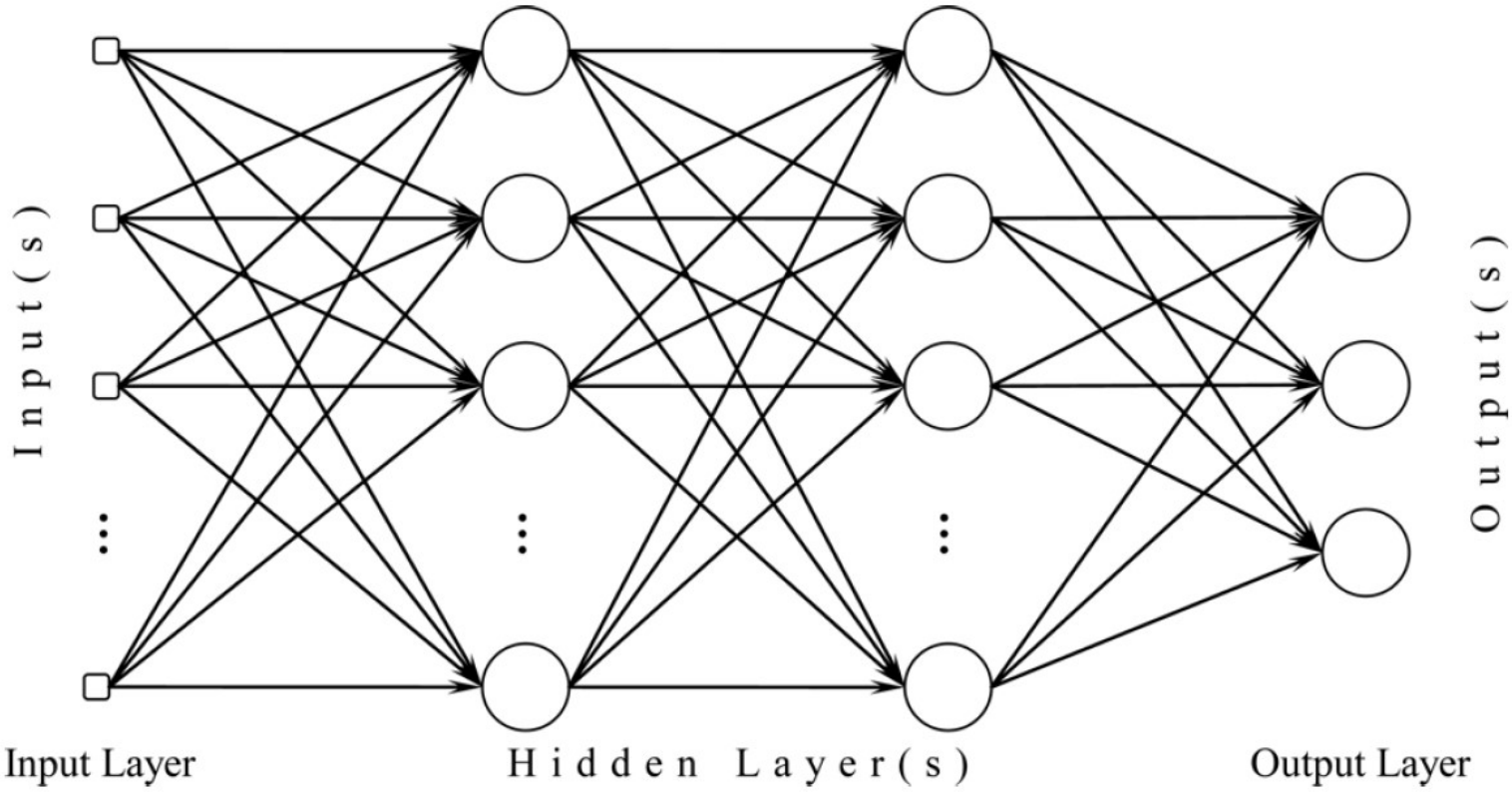
A perceptron follows the "feed-forward" model, meaning inputs are sent into the neuron, are processed, and result in an output. In the diagram above, this means the network (one neuron) reads from left to right: inputs come in, output goes out.

Activation Functions

Activation functions also known as transfer function is used to map input nodes to output nodes in certain fashion.

- There are many activation functions
- Sigmoid, tanh , relu, softmax are the most

Multi Layer Perceptron



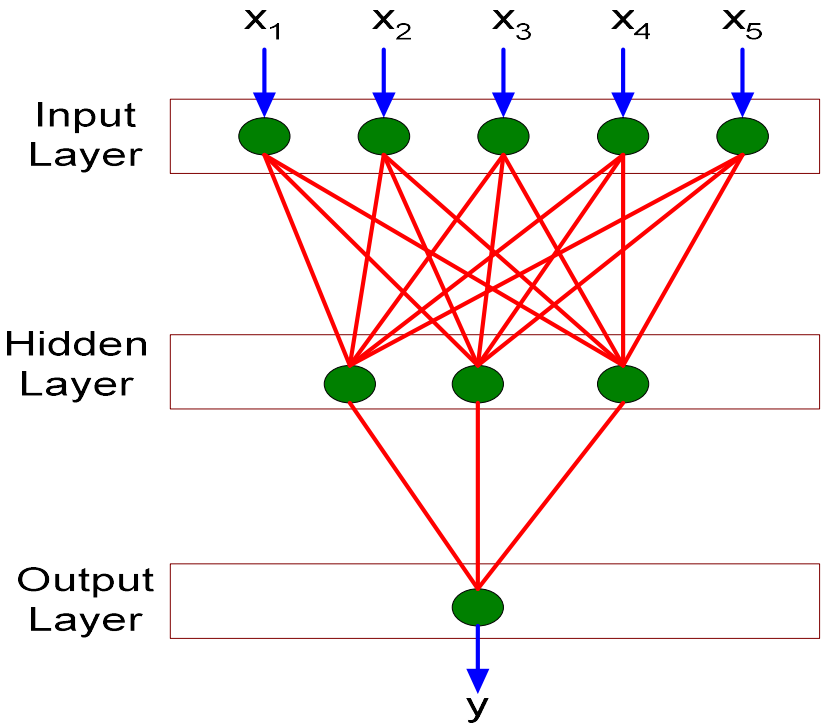
Multi Layer Perceptron

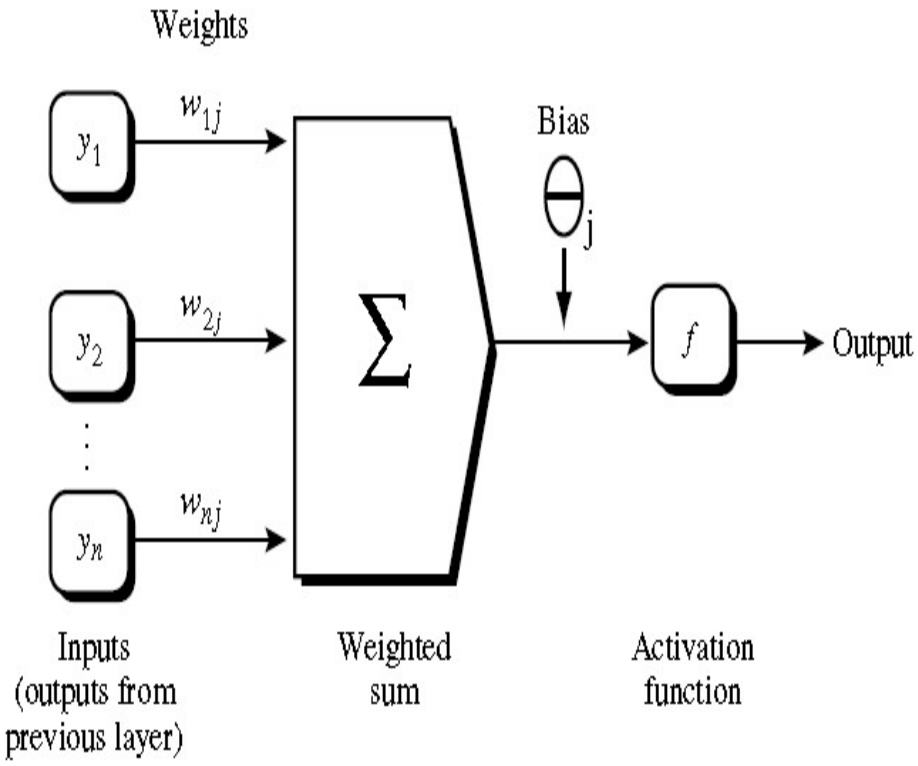
Input Layer
- Visible layer or the left most layer in the network
- Number of nodes in input layer is equal to number of variables in input data
- No activation function

Hidden Layer
- Layers between input and output layers
- Adds non linearity to the network
- Activation/Squashing function

Output Layer
- Right most layer in the network
- Fires the output
- Activation function

General Structure of ANN

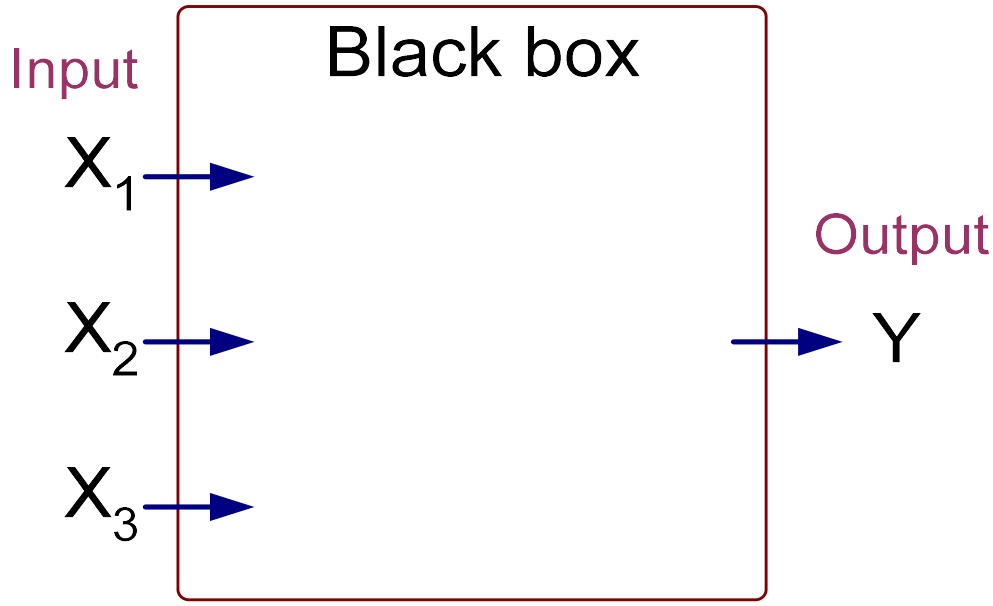




θ_j is the bias of the unit. The bias acts as a threshold, which is used to adjust the output along with the weighted sum of the inputs to the neuron. Therefore bias is a constant which helps the model in a way that it can fit best for the given data..

ANN

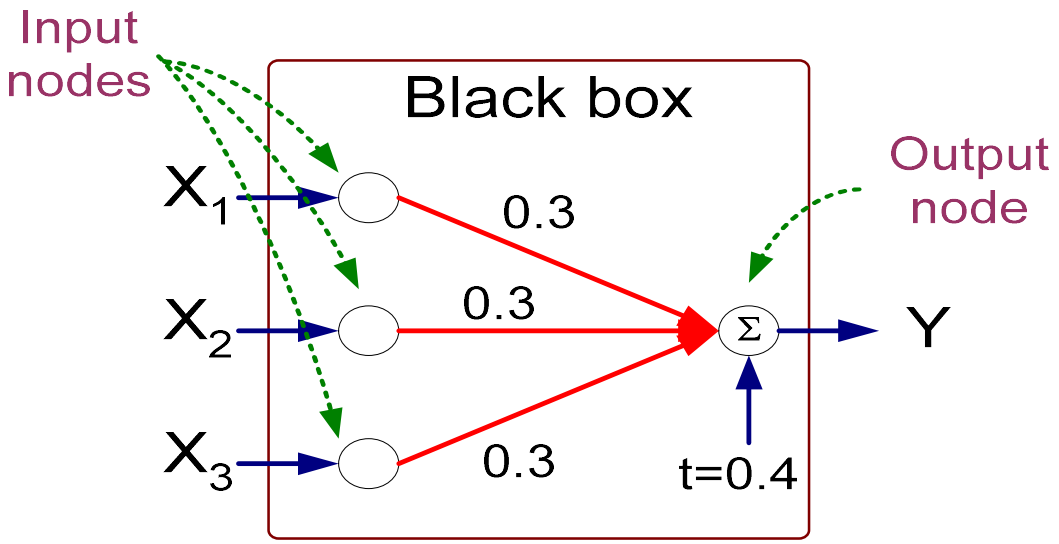
X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



Output Y is 1 if at least two of the three inputs are equal to 1.

ANN

X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0

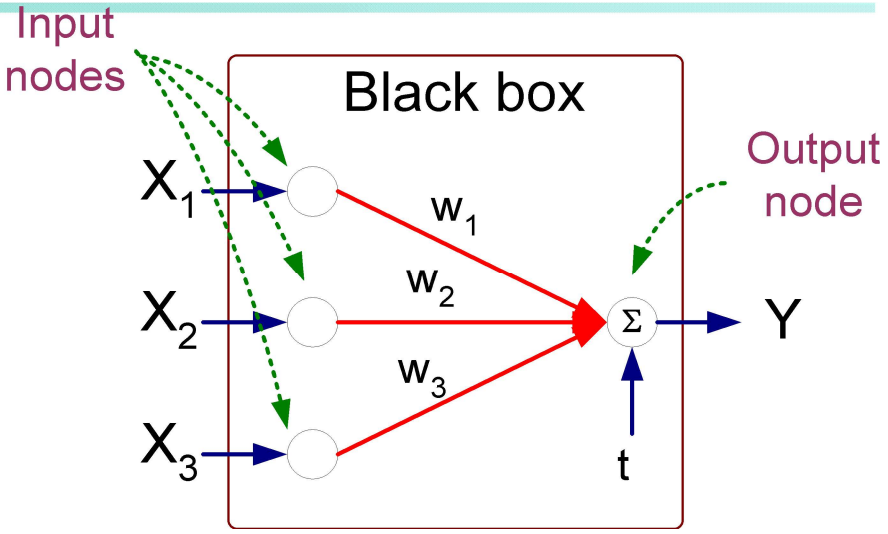


$$Y = I(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4 > 0)$$

$$\text{where } I(z) = \begin{cases} 1 & \text{if } z \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

Artificial Neural Networks

- Model is an assembly of inter-connected nodes and weighted links
- Output node sums up each of its input value according to the weights of its links
- Compare output node against some threshold t



Perceptron Model

$$Y = I(\sum_i w_i X_i - t)$$

Given the net input *Ij to unit j, then Oj, the output of unit j, is computed as,*

$$O_j = \frac{1}{1 + e^{-I_j}}.$$

This function is also referred to as a *squashing function, because it maps a large input* domain onto the smaller range of 0 to 1.

Where Do The Weights Come From?

- The weights in a neural network are the most important factor in determining its function
- Training is the act of presenting the network with some sample data and modifying the weights to better approximate the desired function
- There are two main types of training
  - Supervised Training
    - Supplies the neural network with inputs and the desired outputs
    - Response of the network to the inputs is measured
      - The weights are modified to reduce the difference between the actual and desired outputs

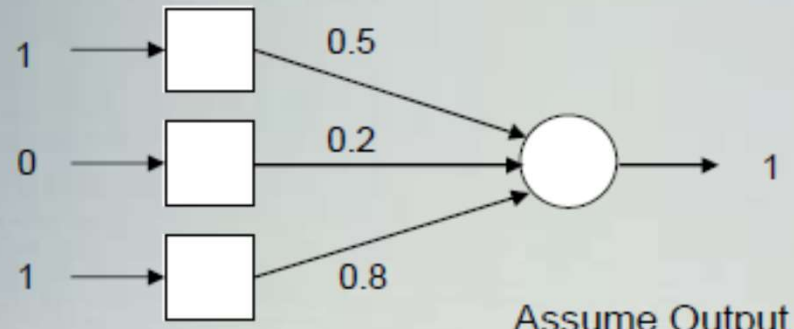
Where Do The Weights Come From?

- Unsupervised Training
  - Only supplies inputs
  - The neural network adjusts its own weights so that similar inputs cause similar outputs
    - The network identifies the patterns and differences in the inputs without any external assistance

How Do Perceptrons Learn?

- Uses supervised training
- If the output is not correct, the weights are adjusted according to the formula:
  - $W_{new} = W_{old} + \alpha(\text{desired} - \text{output})*\text{input}$

α is the learning rate



$1 * 0.5 + 0 * 0.2 + 1 * 0.8 = 1.3$

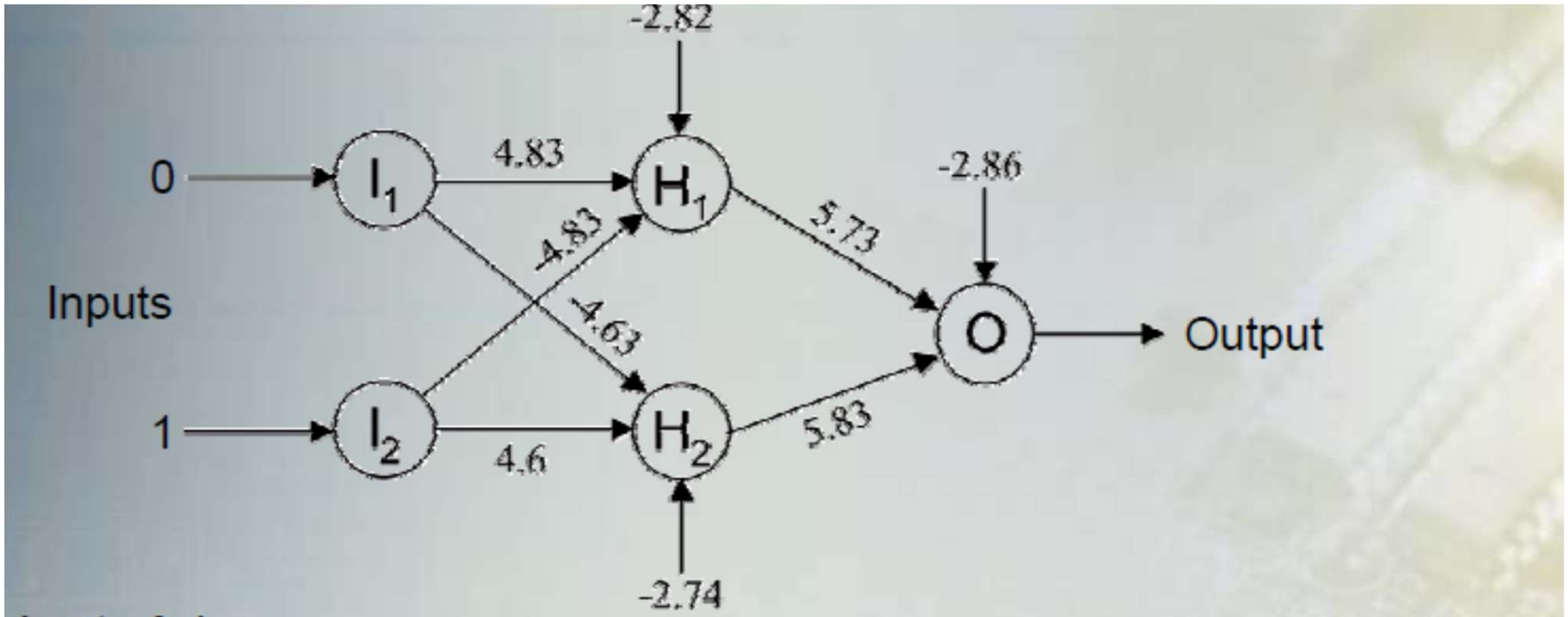
Assuming Output Threshold = 1.2

$1.3 > 1.2$

Assume Output was supposed to be 0
→ update the weights

Assume $\alpha = 1$

$W_{1new} = 0.5 + 1*(0-1)*1 = -0.5$
$W_{2new} = 0.2 + 1*(0-1)*0 = 0.2$
$W_{3new} = 0.8 + 1*(0-1)*1 = -0.2$



Inputs: 0, 1

H_1: Net = 0(4.83) + 1(-4.83) – 2.82 = -7.65
Output = $1 / (1 + e^{7.65}) = 4.758 \times 10^{-4}$

H_2: Net = 0(-4.63) + 1(4.6) – 2.74 = 1.86
Output = $1 / (1 + e^{-1.86}) = 0.8652$

O: Net = $4.758 \times 10^{-4}(5.73) + 0.8652(5.83) – 2.86 = 2.187$
Output = $1 / (1 + e^{-2.187}) = 0.8991 \equiv$ "1"

Learning Algorithms:
Back propagation for classification

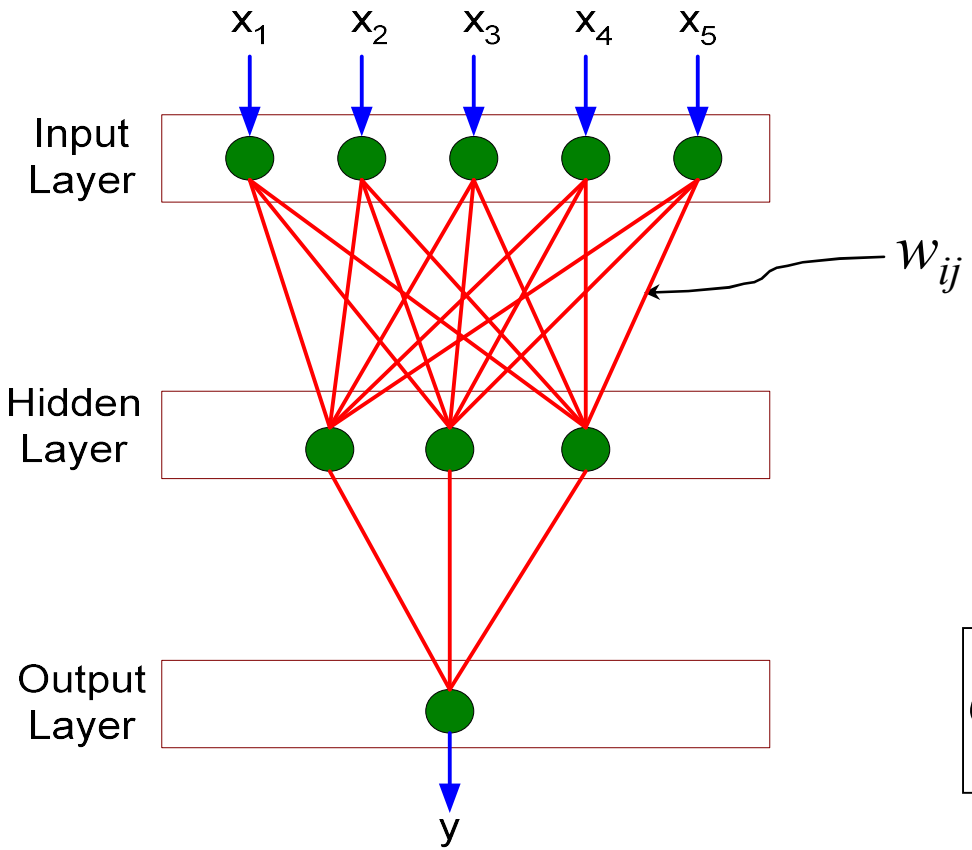
What is backpropagation

- Backpropagation is a neural network learning algorithm.
- There are many different kinds of neural networks and neural network algorithms.
- The most popular neural network algorithm is *backpropagation, which gained repute* in the 1980s.
- Multilayer feed-forward networks is type of neural network on which the backpropagation algorithm performs.
- Backpropagation learns for a set of weights that fits the training data so as to minimize the mean squared distance between the network's class prediction and the known target value of the tuples.

Major Steps for Back Propagation Network

- Constructing a network
  - input data representation
  - selection of number of layers, number of nodes in each layer.
- Training the network using training data
- Pruning the network
- Interpret the results

A Multi-Layer Feed-Forward Neural Network



$$I_j = \sum_i w_{ij} O_i + \theta_j$$

$$O_j = \frac{1}{1 + e^{-I_j}}$$

How A Multi-Layer Neural Network Works?

- The **inputs** to the network correspond to the attributes measured for each training tuple
- Inputs are fed simultaneously into the units making up the **input layer**
- They are then weighted and fed simultaneously to a **hidden layer**
- The number of hidden layers is arbitrary, although usually only one
- The weighted outputs of the last hidden layer are input to units making up the **output layer**, which gives out the network's prediction
- The network is **feed-forward** in that none of the weights cycles back to an input unit or to an output unit of a previous layer

Defining a Network Topology

- First decide the **network topology:** # of units in the *input layer*, # of *hidden layers* (if > 1), # of units in *each hidden layer*, and # of units in the *output layer*
- Normalizing the input values for each attribute measured in the training tuples to [0.0—1.0]
- One **input** unit per domain value
- **Output**, if for classification and more than two classes, one output unit per class is used
- Once a network has been trained and its accuracy is **unacceptable**, repeat the training process with a *different network topology* or a *different set of initial weights*

Backpropagation

- Iteratively process a set of training tuples & compare the network's prediction with the actual known target value
- For each training tuple, the weights are modified to **minimize the mean squared error** between the network's prediction and the actual target value
- Modifications are made in the "**backwards**" direction: from the output layer, through each hidden layer down to the first hidden layer, hence "**backpropagation**"
- Steps
  - Initialize weights and biases in the network
  - Propagate the inputs forward (by applying activation function)
  - Backpropagate the error (by updating weights and biases)
  - Terminating condition (when error is very small, etc.)

Backpropagation: Algorithm

Algorithm: Backpropagation. Neural network learning for classification or prediction, using the backpropagation algorithm.

Input:

- D, a data set consisting of the training tuples and their associated target values;
- l, the learning rate;
- *network*, a multilayer feed-forward network.

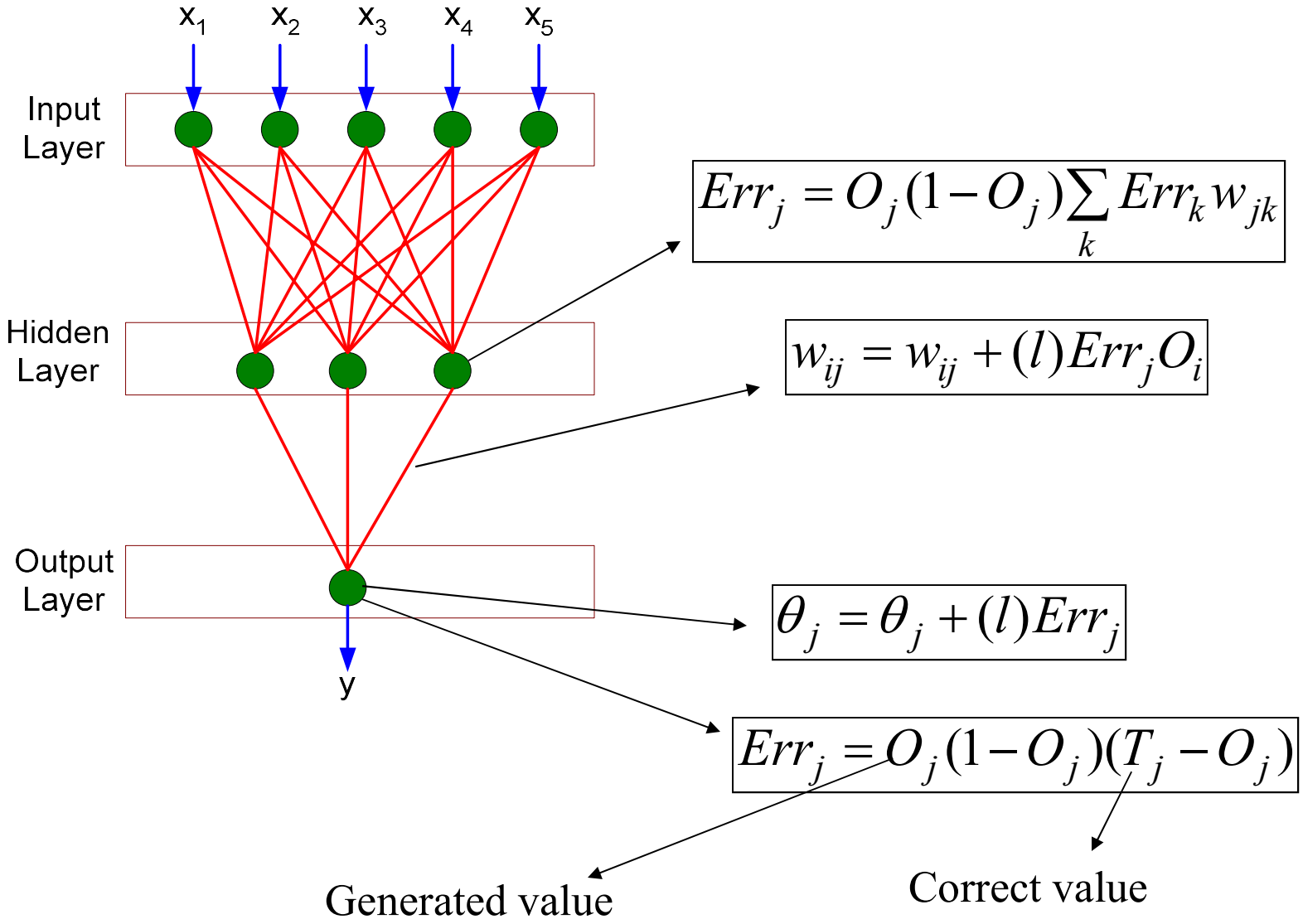
Output: A trained neural network.

Method:

(1) Initialize all weights and biases in *network*;
(2) **while** terminating condition is not satisfied {
(3) **for** each training tuple $\boldsymbol{X}$ in D {
(4) // Propagate the inputs forward:
(5) **for** each input layer unit j {
(6) $O_j = I_j$; // output of an input unit is its actual input value
(7) **for** each hidden or output layer unit j {
(8) $I_j = \sum_i w_{ij} O_i + \theta_j$; //compute the net input of unit j with respect to the previous layer, i
(9) $O_j = \frac{1}{1+e^{-I_j}}$; } // compute the output of each unit j
(10) // Backpropagate the errors:
(11) **for** each unit j in the output layer
(12) $Err_j = O_j(1 - O_j)(T_j - O_j)$; // compute the error
(13) **for** each unit j in the hidden layers, from the last to the first hidden layer
(14) $Err_j = O_j(1 - O_j) \sum_k Err_k w_{jk}$; // compute the error with respect to the next higher layer, k
(15) **for** each weight w_{ij} in *network* {
(16) $\Delta w_{ij} = (l) Err_j O_i$; // weight increment
(17) $w_{ij} = w_{ij} + \Delta w_{ij}$; } // weight update
(18) **for** each bias θ_j in *network* {
(19) $\Delta \theta_j = (l) Err_j$; // bias increment
(20) $\theta_j = \theta_j + \Delta \theta_j$; } // bias update
(21) } }

Backpropagation algorithm.

Backpropagation



Example - Sample calculations for learning by the backpropagation algorithm

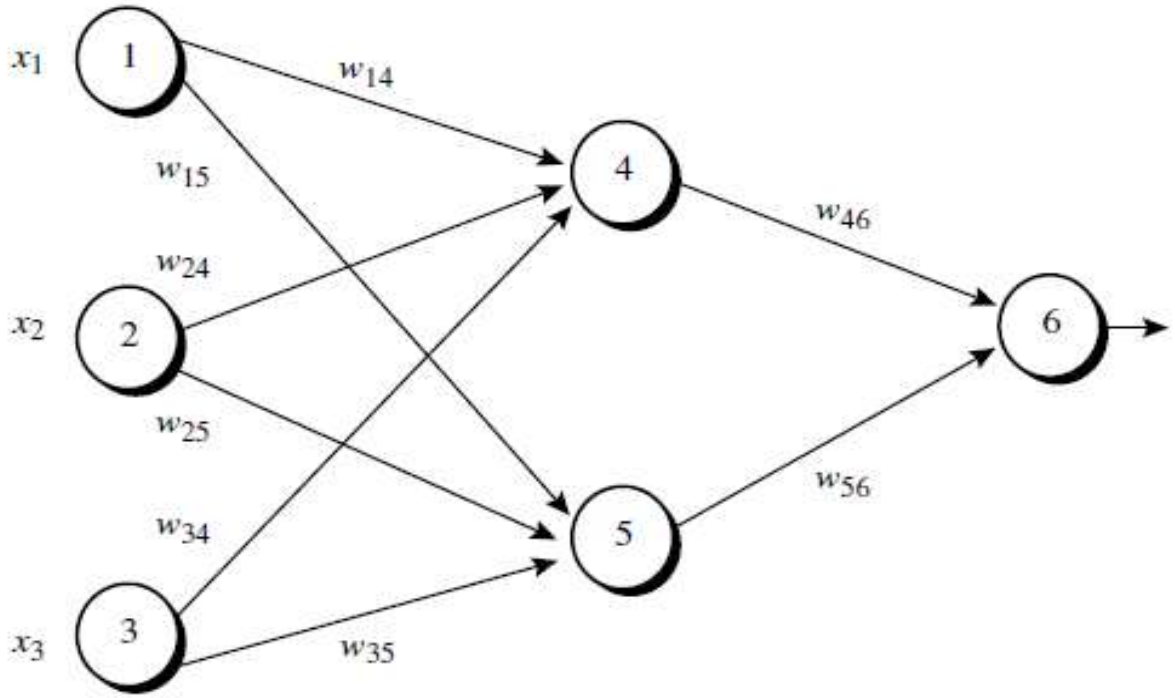
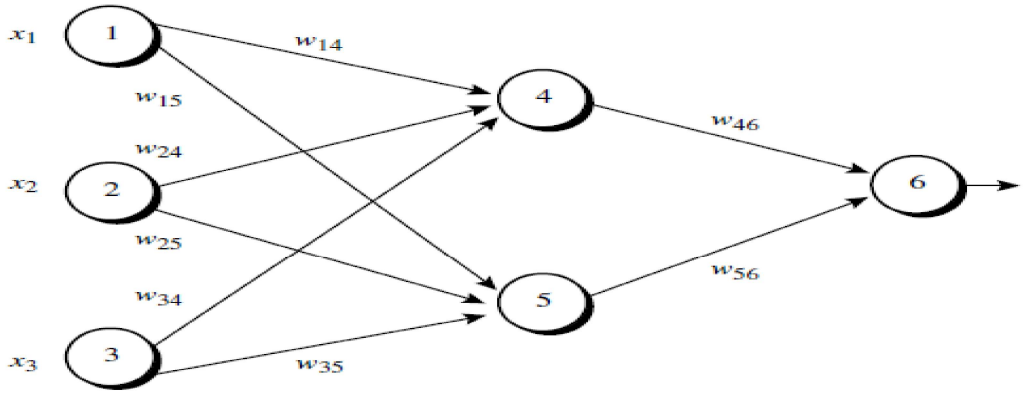


Figure shows : a multilayer feed-forward neural network.
Let the learning rate be 0.9.
The first training tuple, ***X = (1, 0, 1), whose class label is 1.***



Initial input, weight, and bias values.

x_1	x_2	x_3	w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{56}	θ_4	θ_5	θ_6
1	0	1	0.2	−0.3	0.4	0.1	−0.5	0.2	−0.3	−0.2	−0.4	0.2	0.1

The net input and output calculations.

Unit j	*Net input,* I_j	*Output,* O_j
4	$0.2+0-0.5-0.4=-0.7$	$1/(1+e^{0.7})=0.332$
5	$-0.3+0+0.2+0.2=0.1$	$1/(1+e^{-0.1})=0.525$
6	$(-0.3)(0.332)-(0.2)(0.525)+0.1=-0.105$	$1/(1+e^{0.105})=0.474$

Calculation of the error at each node.

Unit j	Err_j
6	$(0.474)(1-0.474)(1-0.474)=0.1311$
5	$(0.525)(1-0.525)(0.1311)(-0.2)=-0.0065$
4	$(0.332)(1-0.332)(0.1311)(-0.3)=-0.0087$

for each unit j in the output layer
    $Err_j = O_j(1-O_j)(T_j-O_j)$; // compute the error
for each unit j in the hidden layers, from the last to the first hidden layer
    $Err_j = O_j(1-O_j)\sum_k Err_k w_{jk}$; // compute the error with respect to the next higher layer, k

```
for each weight w_ij in network {
    Δw_ij = (l)Err_j O_i; // weight incremen
    w_ij = w_ij + Δw_ij; } // weight update
for each bias θ_j in network {
    Δθ_j = (l)Err_j; // bias increment
    θ_j = θ_j + Δθ_j; } // bias update
```

Calculations for weight and bias updating.

Weight or bias	*New value*
w_{46}	$-0.3 + (0.9)(0.1311)(0.332) = -0.261$
w_{56}	$-0.2 + (0.9)(0.1311)(0.525) = -0.138$
w_{14}	$0.2 + (0.9)(-0.0087)(1) = 0.192$
w_{15}	$-0.3 + (0.9)(-0.0065)(1) = -0.306$
w_{24}	$0.4 + (0.9)(-0.0087)(0) = 0.4$
w_{25}	$0.1 + (0.9)(-0.0065)(0) = 0.1$
w_{34}	$-0.5 + (0.9)(-0.0087)(1) = -0.508$
w_{35}	$0.2 + (0.9)(-0.0065)(1) = 0.194$
θ_6	$0.1 + (0.9)(0.1311) = 0.218$
θ_5	$0.2 + (0.9)(-0.0065) = 0.194$
θ_4	$-0.4 + (0.9)(-0.0087) = -0.408$

Neural Network as a Classifier

- Weakness
  - Long training time
  - Require a number of parameters, e.g., the network topology or ``structure."
  - Poor interpretability: Difficult to interpret the symbolic meaning behind the learned weights and of ``hidden units" in the network

- Strength
  - High tolerance to noisy data as well as their ability to classify patterns on which they have not been trained.
  - They are well-suited for continuous-valued inputs *and outputs*, unlike most decision tree algorithms.
  - They have been successful on a wide array of real-world data, including handwritten character recognition, pathology and laboratory medicine, and training a computer to pronounce English text.
  - Neural network algorithms are inherently parallel; parallelization techniques can be used to speed up the computation process.

These above factors contribute toward the usefulness of neural networks for classification and prediction in machine learning.