

Artificial Neural Network

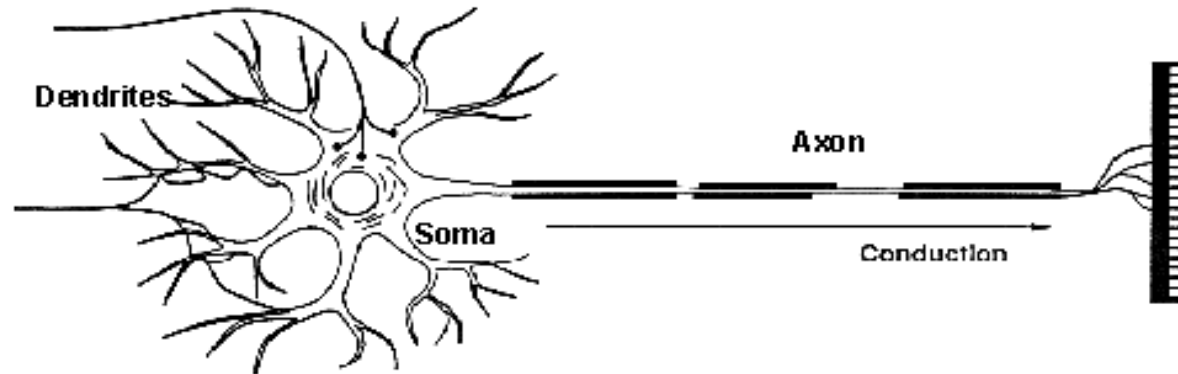
LEKSHMI R CHANDRAN

ASSISTANT PROFESSOR

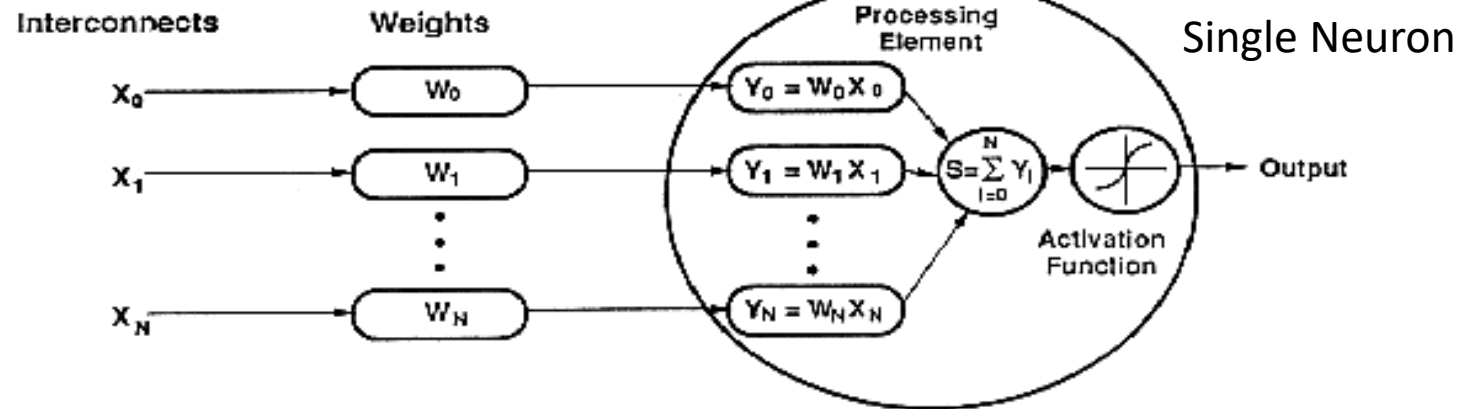
DEPT. OF EEE

How do ANNs work?

Biological Neuron



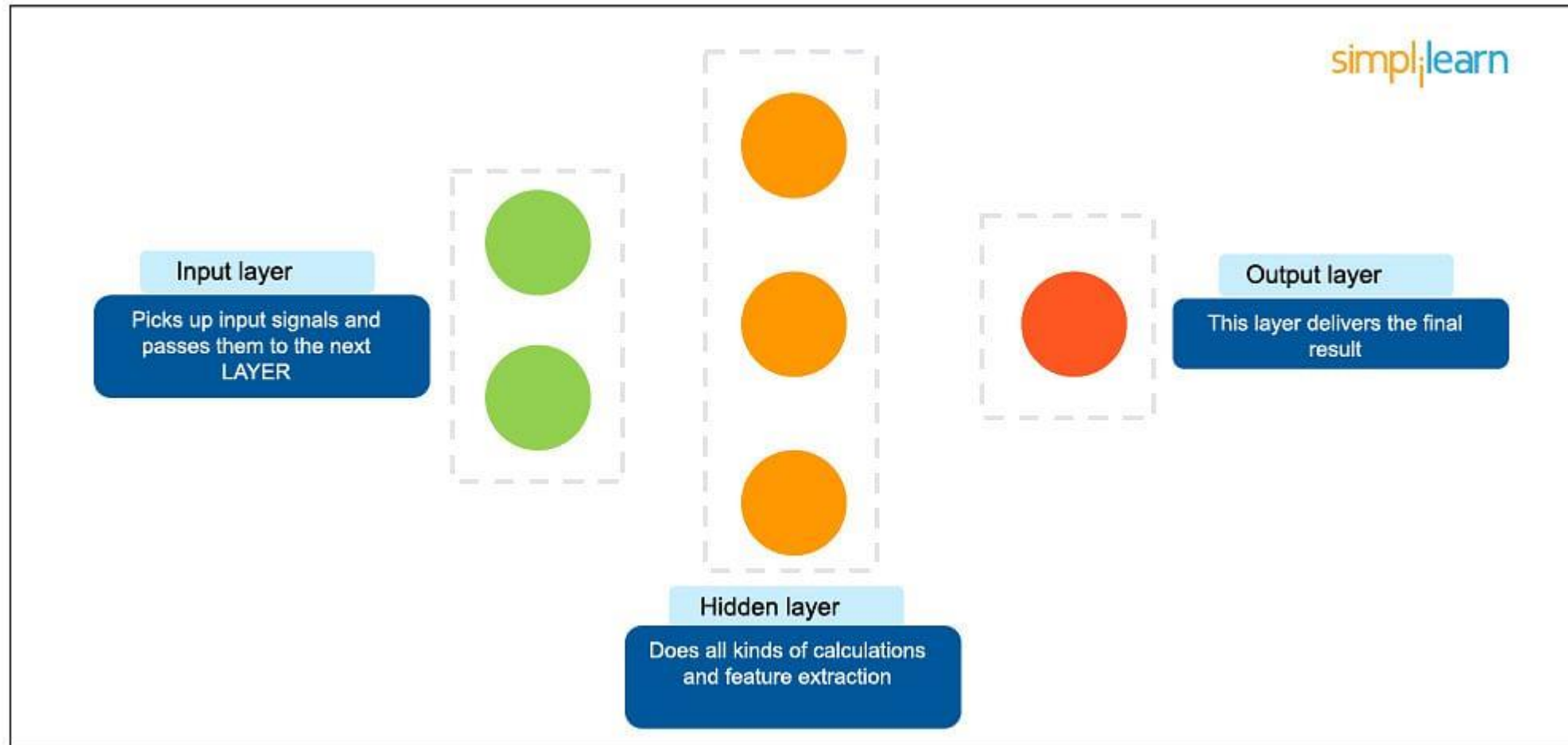
Artificial Neuron



Biological Neuron	Artificial Neuron
Cell Nucleus (Soma)	Node
Dendrites	Input
Synapse	Weights or interconnections
Axon	Output

An artificial neuron is an imitation of a human neuron

Artitifical Neural Network Architecture



Artificial Neural Network Architecture

Input Nodes (input layer): No computation is done here within this layer, they just pass the information to the next layer (hidden layer most of the time). A block of nodes is also called **layer**.

Hidden nodes (hidden layer): In Hidden layers is where intermediate processing or computation is done, they perform computations and then transfer the weights (signals or information) from the input layer to the following layer (another hidden layer or to the output layer). It is possible to have a neural network without a hidden layer and I'll come later to explain this.

Output Nodes (output layer): Here we finally use an activation function that maps to the desired output format (e.g. softmax for classification).

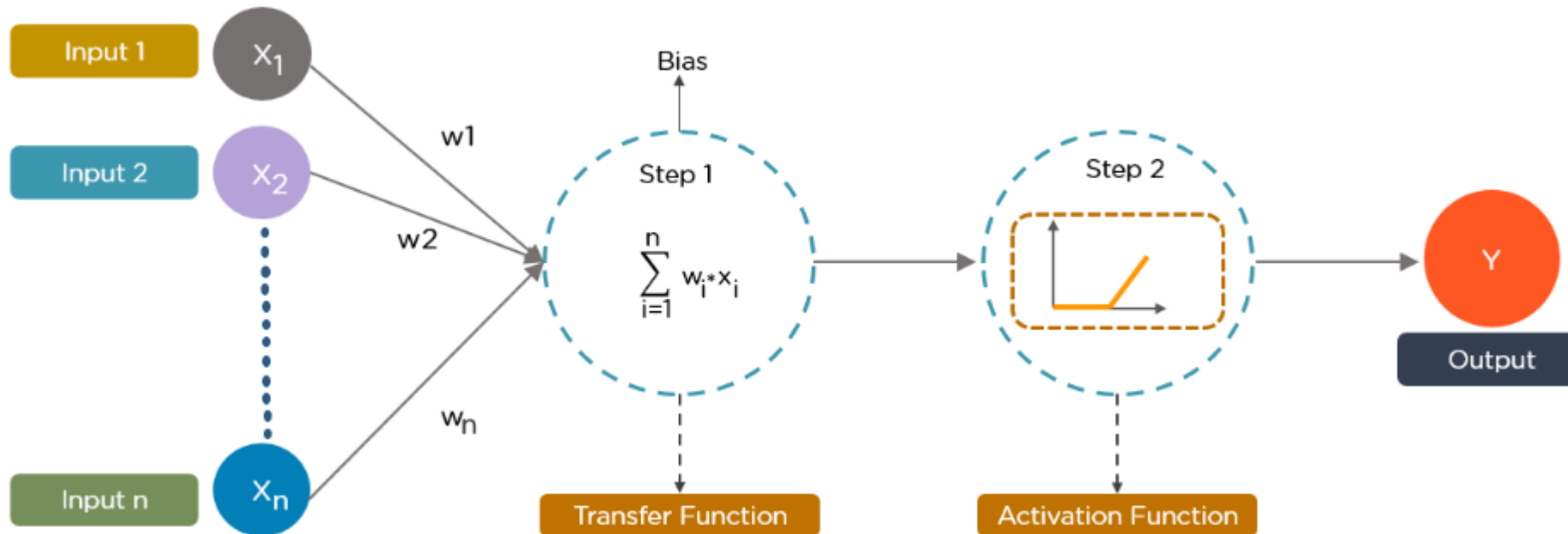
Weights: determines which signals get passed along or not, or to what extent a signal gets passed along. weights are what you will adjust through the process of learning.

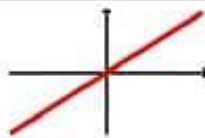
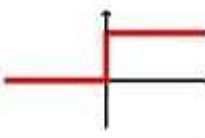
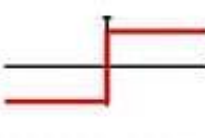


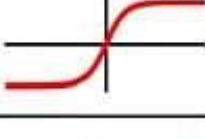

Bias: bias helps in controlling the value at which activation function will trigger, like the intercept added in a linear equation → helps the model in a way that it can fit best for the given data.

Activation function: the **activation function** of a node defines the output of that node given an input or set of inputs. In artificial neural networks this function is also called the transfer function. *Firing rate* of the neuron with an **activation function** (e.x *sigmoid function*), represents the frequency of the spikes along the axon.

Learning rule: The *learning rule* is a rule or an algorithm which modifies the parameters of the neural network, in order for a given input to the network to produce a favored output. This *learning* process typically amounts to modifying the weights and thresholds.

Mathematical Model of a Neuron

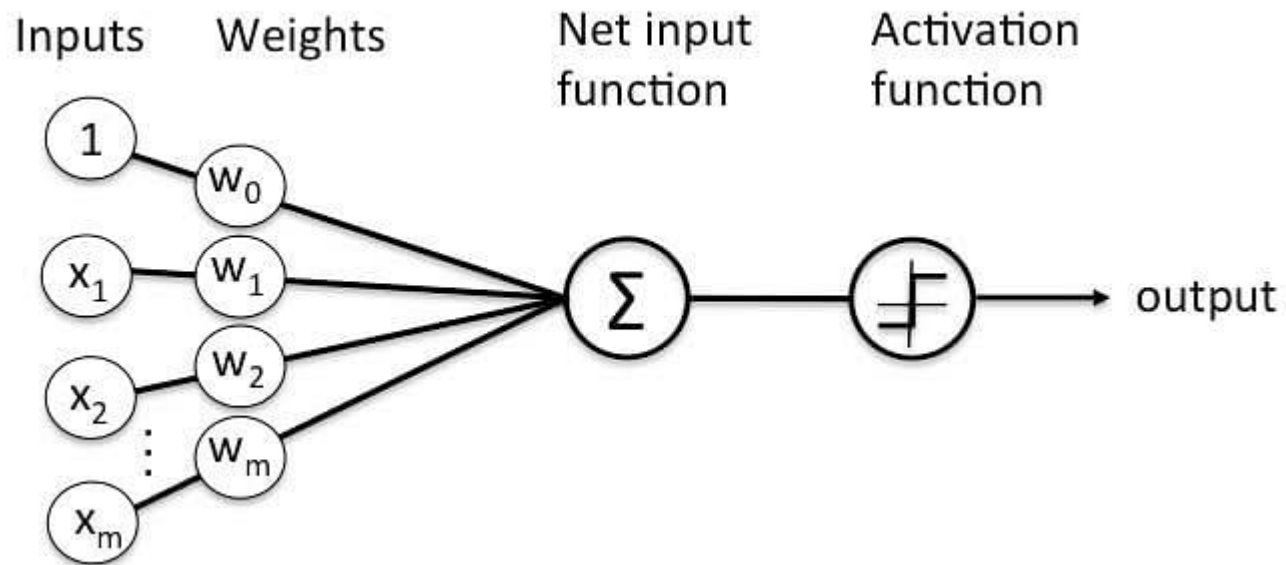


Activation Function	Equation	Example	1D Graph
Linear	$\phi(z) = z$	Adaline, linear regression	
Unit Step (Heaviside Function)	$\phi(z) = \begin{cases} 0 & z < 0 \\ 0.5 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Sign (signum)	$\phi(z) = \begin{cases} -1 & z < 0 \\ 0 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Piece-wise Linear	$\phi(z) = \begin{cases} 0 & z \leq -\frac{1}{2} \\ z + \frac{1}{2} & -\frac{1}{2} \leq z \leq \frac{1}{2} \\ 1 & z \geq \frac{1}{2} \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multilayer NN	
Hyperbolic Tangent (tanh)	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multilayer NN, RNNs	
ReLU	$\phi(z) = \begin{cases} 0 & z < 0 \\ z & z > 0 \end{cases}$	Multilayer NN, CNNs	

Perceptron

A Perceptron is an algorithm for supervised learning of binary classifiers.

→ Algorithm enables neurons to learn and processes elements in the training set one at a time



Example 1

The input to a single-input neuron is 2.0, its weight is 2.3 and its bias is -3.

- What is the net input to the transfer function?
- What is the neuron output if the transfer function is Log-sigmoid?

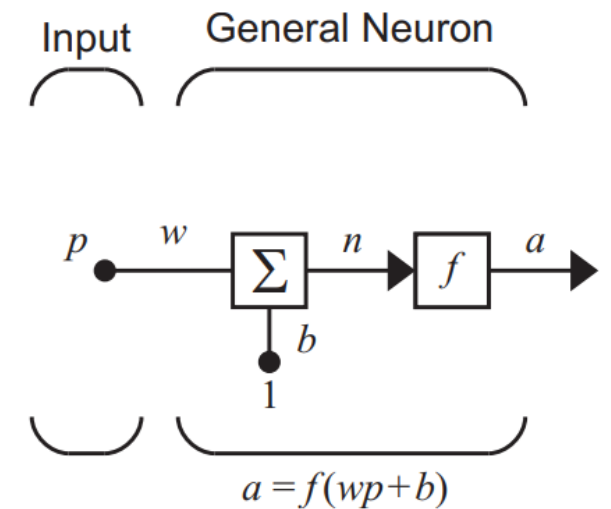
Solution

Net input to the transfer function

$$n = wp + b = (2.3)(2) + (-3) = 1.6$$

Neuron output if the transfer function is Log-sigmoid

$$= \frac{1}{1 + e^{-1.6}} = 0.8320$$



Example 2

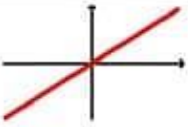
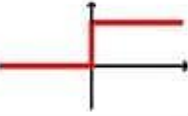
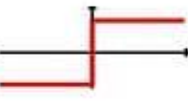



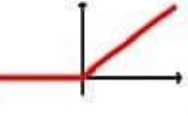
The input to a single-input neuron is 2.0, its weight is 2.3 and its bias is -3.

- What is the net input to the transfer function?
- What is the neuron output if the transfer function is Signum?

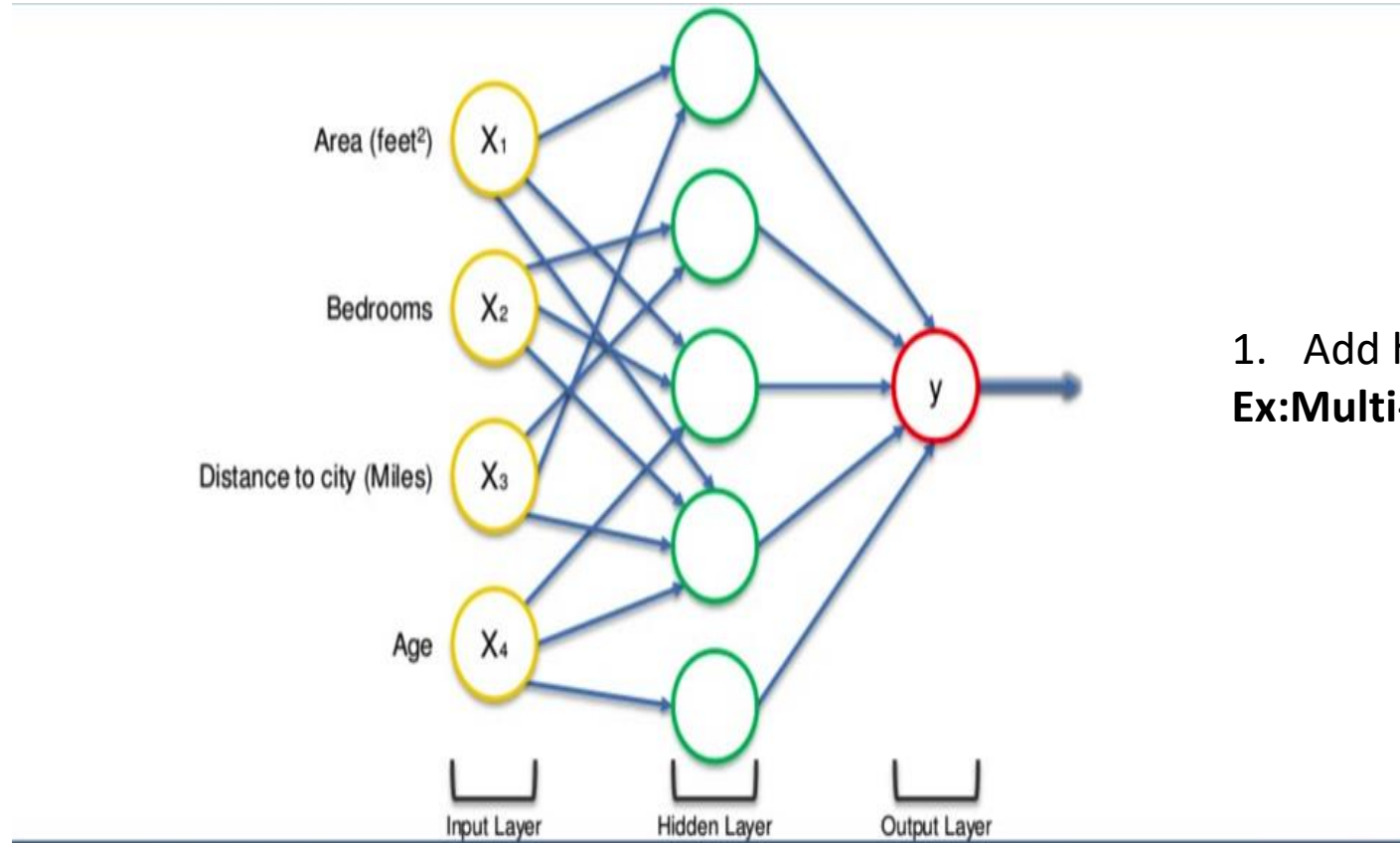
Example 3

The input to a single-input neuron is 2.0, its weight is 2.3 and its bias is -3.

- What is the net input to the transfer function?
- What is the neuron output if the transfer function is Hyperbolic Tangent Function?

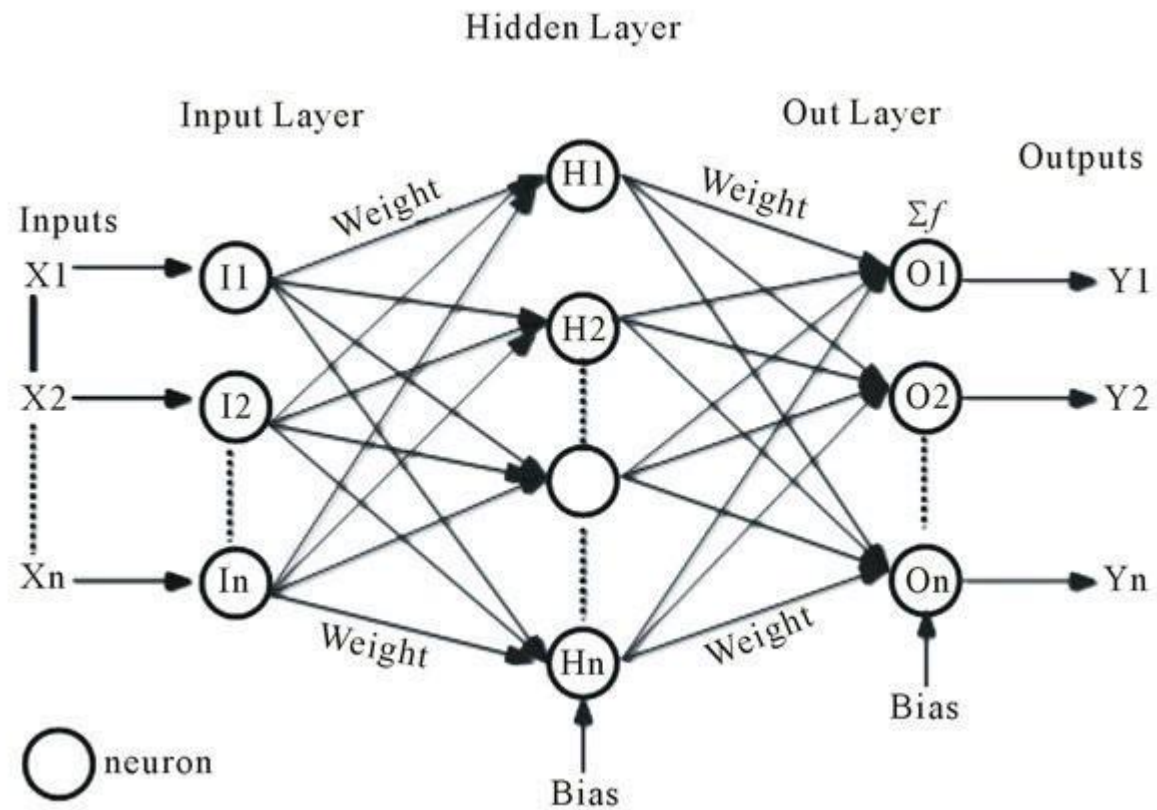
Activation Function	Equation	Example	1D Graph
Linear	$\phi(z) = z$	Adaline, linear regression	
Unit Step (Heaviside Function)	$\phi(z) = \begin{cases} 0 & z < 0 \\ 0.5 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Sign (signum)	$\phi(z) = \begin{cases} -1 & z < 0 \\ 0 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Piece-wise Linear	$\phi(z) = \begin{cases} 0 & z \leq -\frac{1}{2} \\ z + \frac{1}{2} & -\frac{1}{2} \leq z \leq \frac{1}{2} \\ 1 & z \geq \frac{1}{2} \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multilayer NN	
Hyperbolic Tangent (tanh)	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multilayer NN, RNNs	
ReLU	$\phi(z) = \begin{cases} 0 & z < 0 \\ z & z > 0 \end{cases}$	Multilayer NN, CNNs	

Is there a way to amplify the power of the Neural Network?



1. Add hidden layer
Ex: Multi-layer perceptron (MLP)

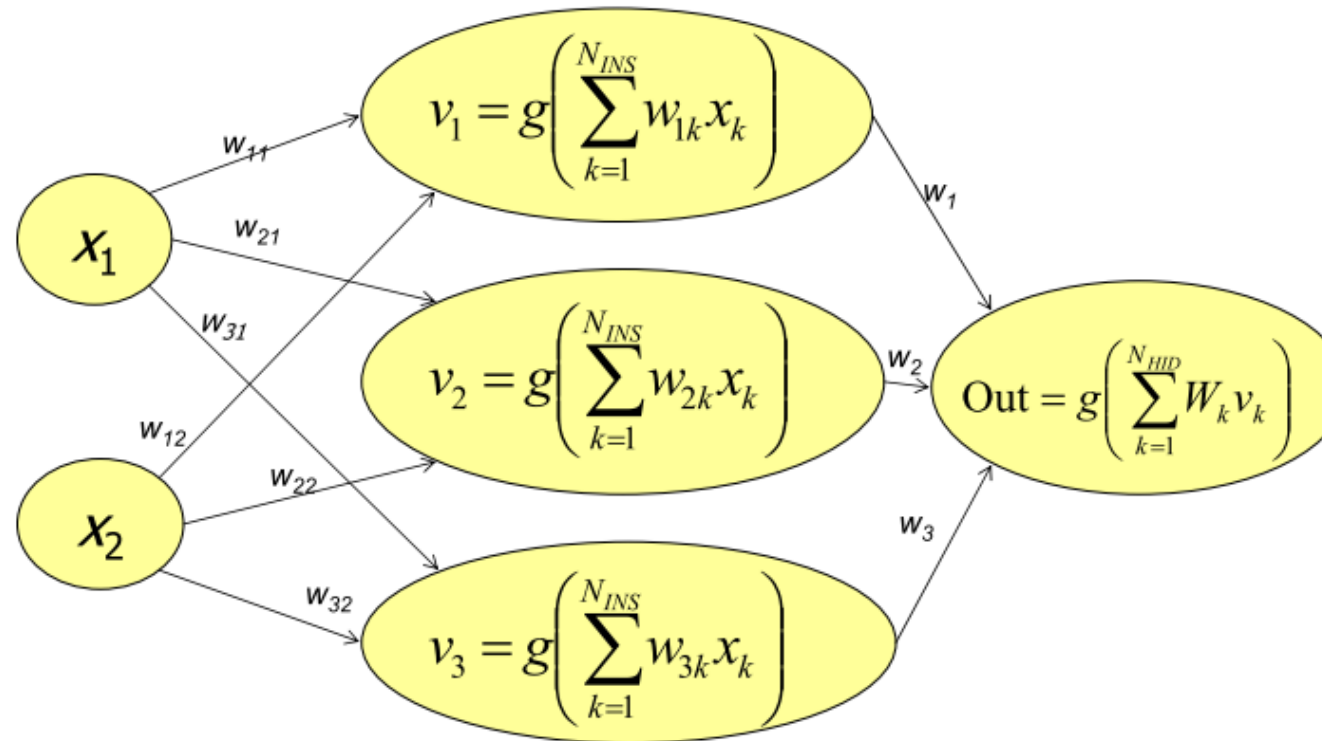
Multilayer perceptron



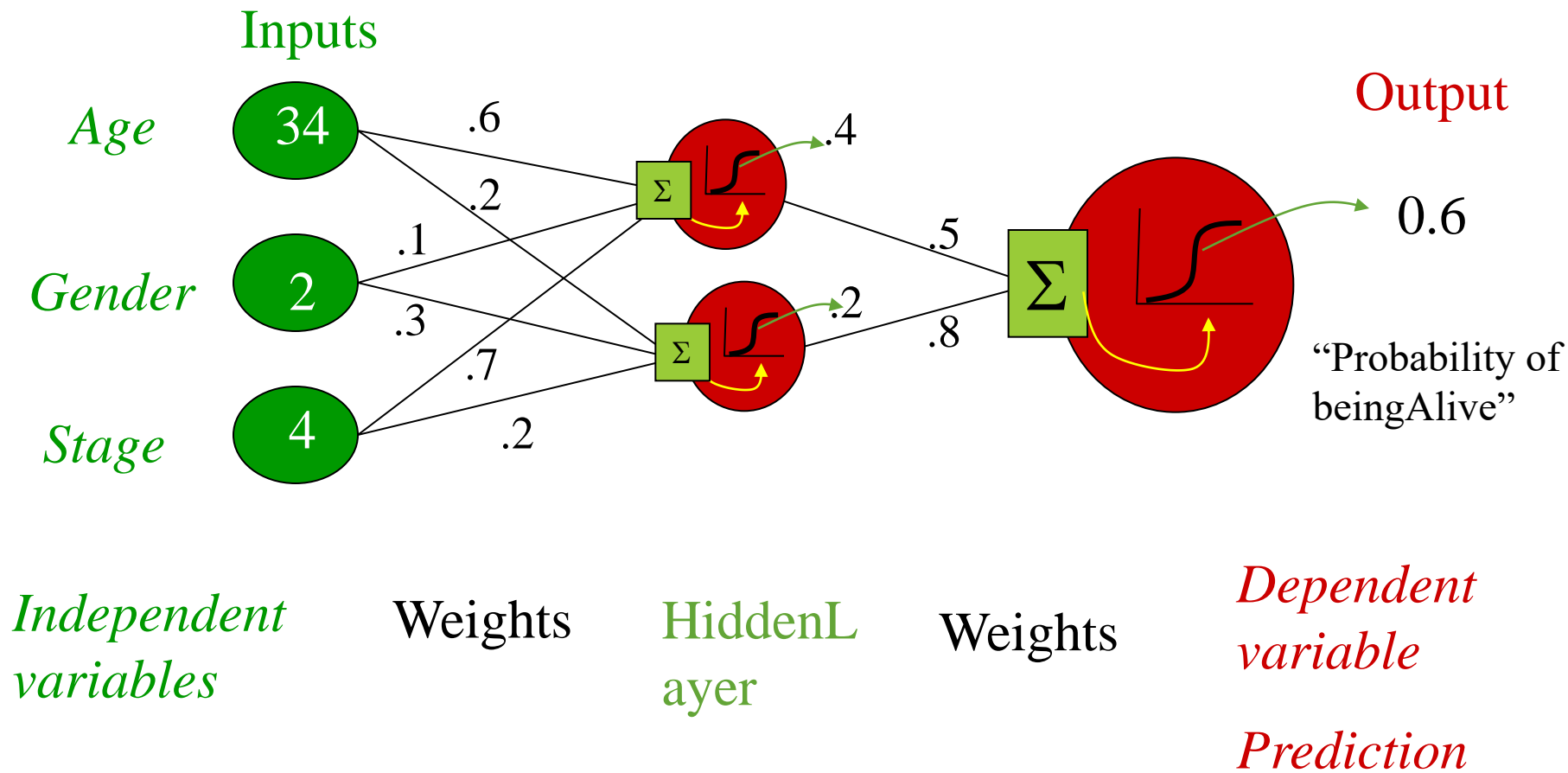
Multilayer perceptron - Example

$N_{\text{INPUTS}} = 2$

$N_{\text{HIDDEN}} = 3$



Numerical Example- Multilayer perceptron



ANN Learning

- 1. Forward Propagation** – Forward Propagation refers to the movement of the input through the hidden layers to the output layers.
- 2. Cost Function** – The cost or loss function tries to penalize the network when it makes errors. The learning process revolves around minimizing the cost.

Ex: Define the cost function to be the mean squared error, as –

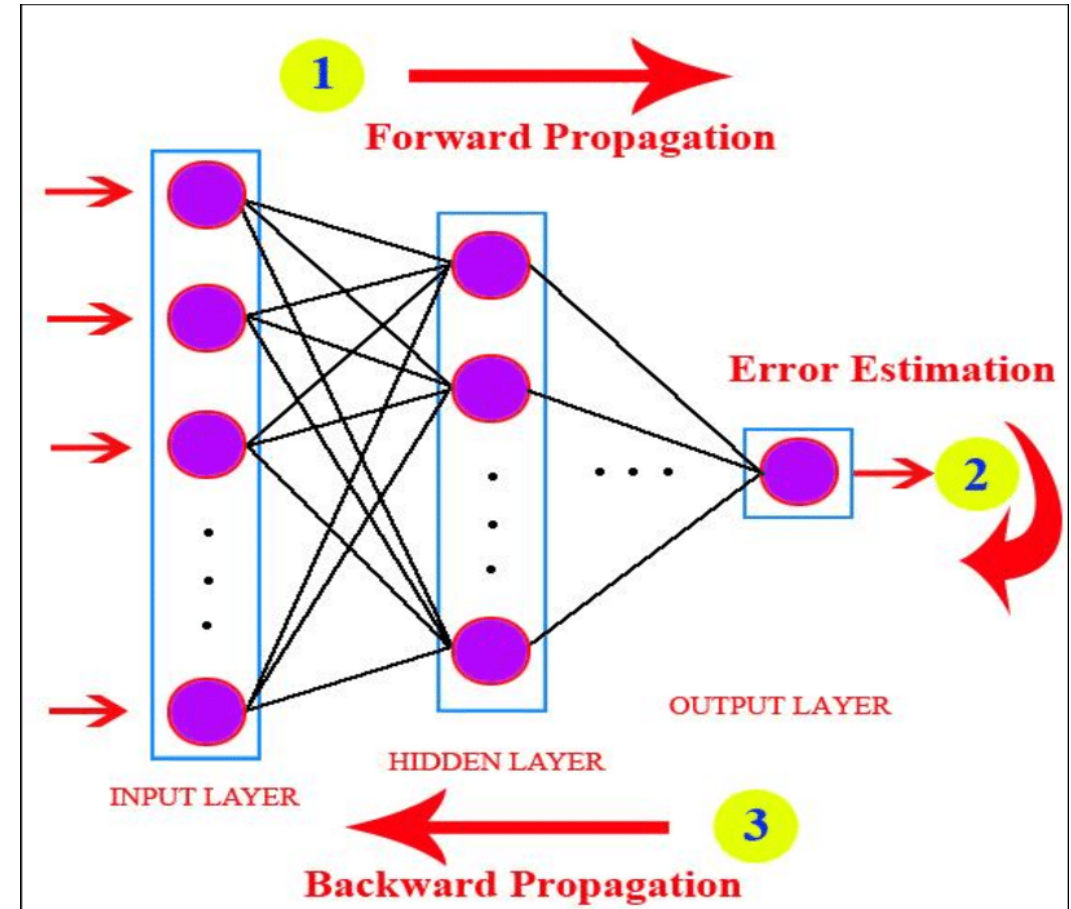
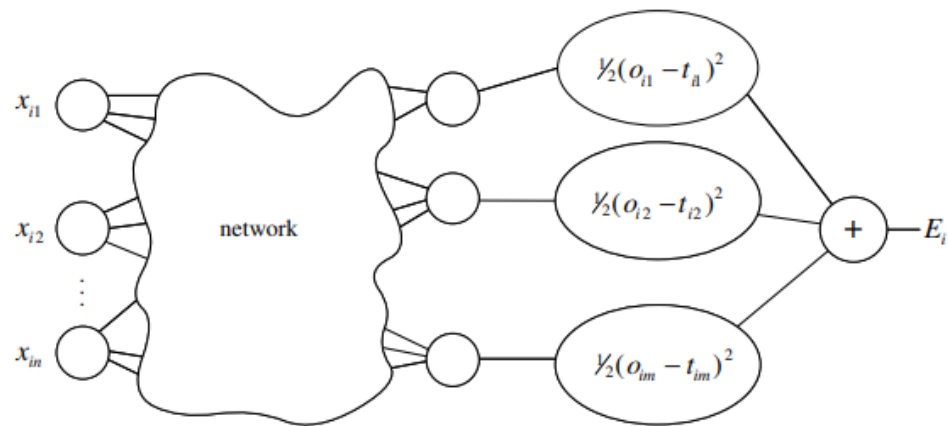
$$C = \sum (y - a)^2 / 2$$

a is the predicted value and y is the actual value of that particular example.

- 3. Gradient Descent** – Gradient descent is an optimization algorithm for minimizing the cost.
- 4. Learning Rate** – The learning rate is defined as the amount of minimization in the cost function in each iteration.
- 5. Back propagation** – Initially random weights and bias values are assigned to our nodes. After each forward propagation, calculate the error of the network. This error is then fed back to the network along with the gradient of the cost function to update the weights of the network. These weights are then updated so that the errors in the subsequent iterations is reduced. This updating of weights using the gradient of the cost function is known as back-propagation. In back-propagation the movement of the network is backwards, the error along with the gradient flows back from the out layer through the hidden layers and the weights are updated.

The weights and bias are updated as follows and the back propagation and gradient descent is repeated until convergence.

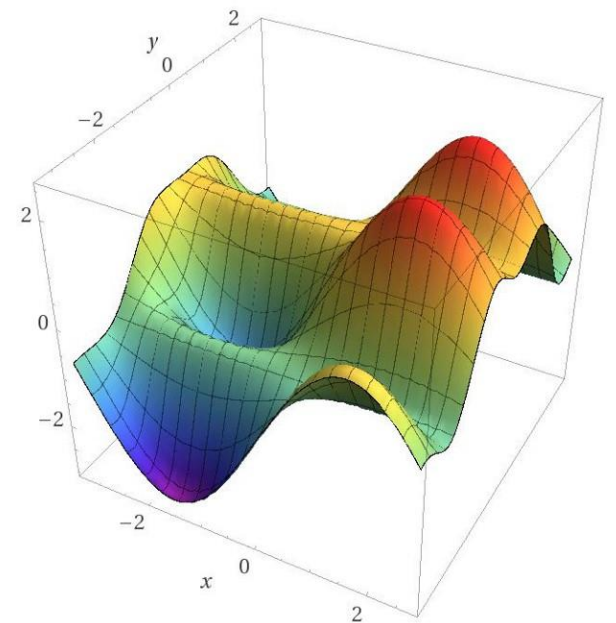
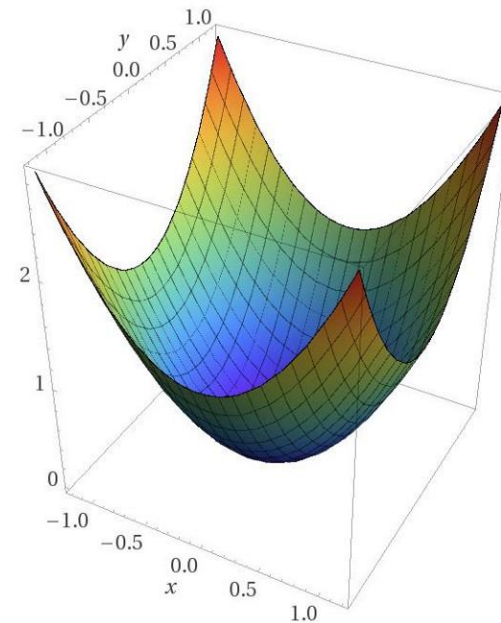
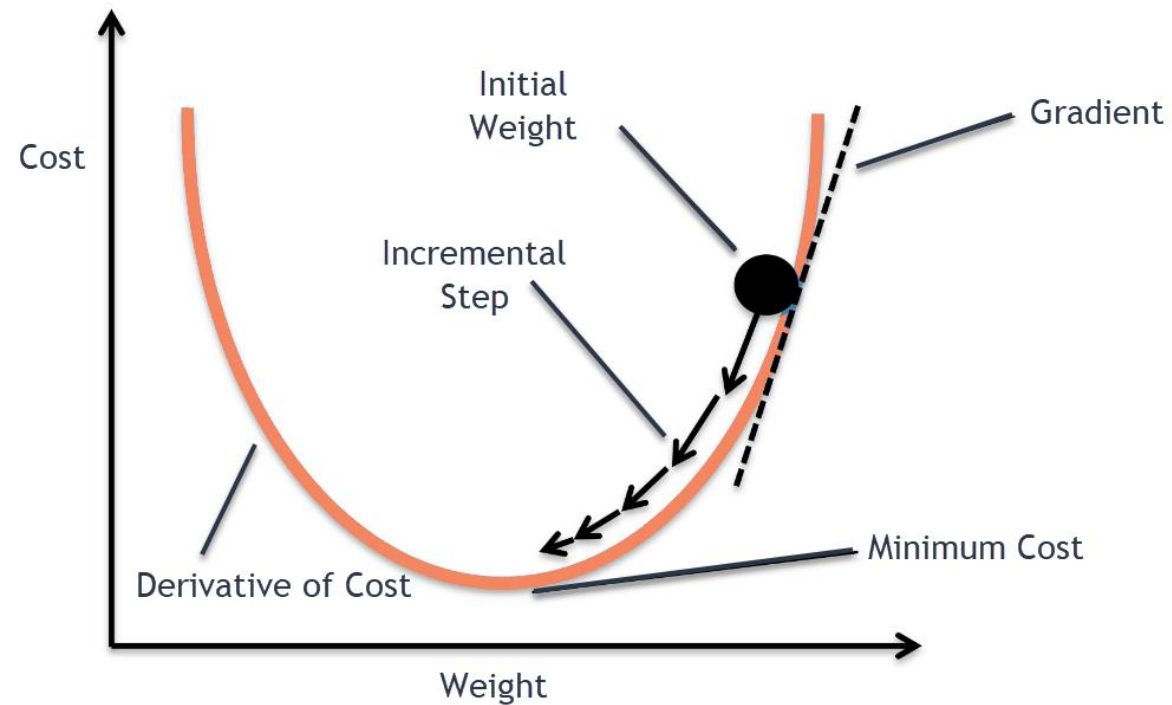
Concept of Forward propagation and Back Propagation in a Feed-forward Neural Network



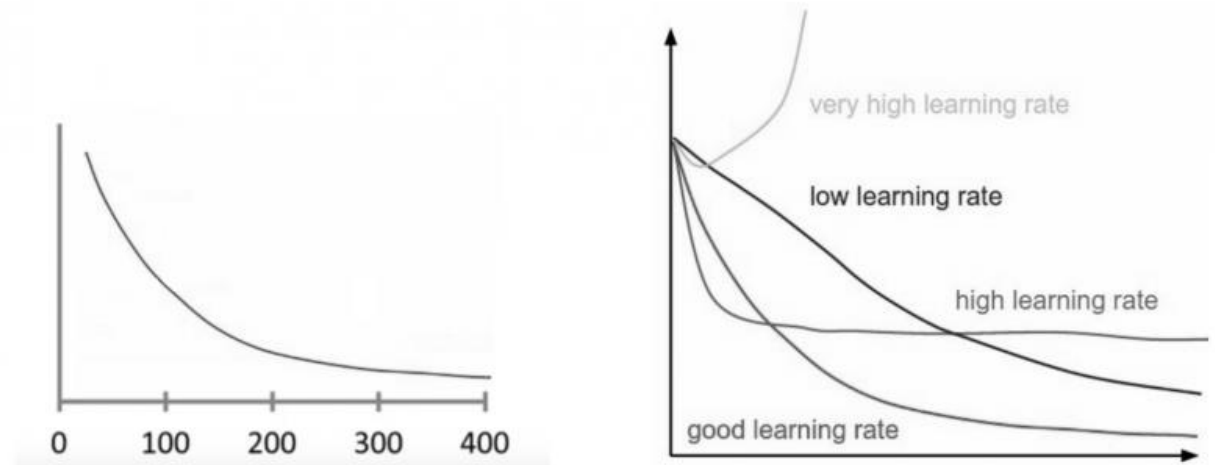
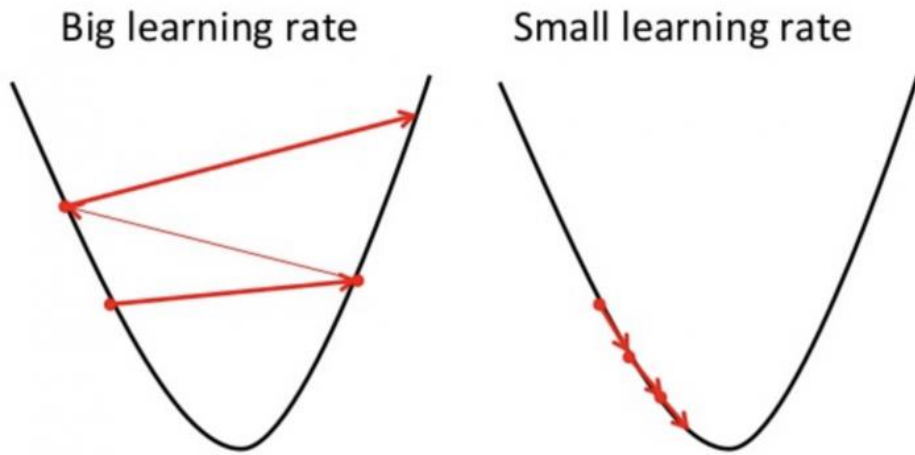
Gradient Descent

→ Optimization algorithm(optimizer)

→ minimize the error.

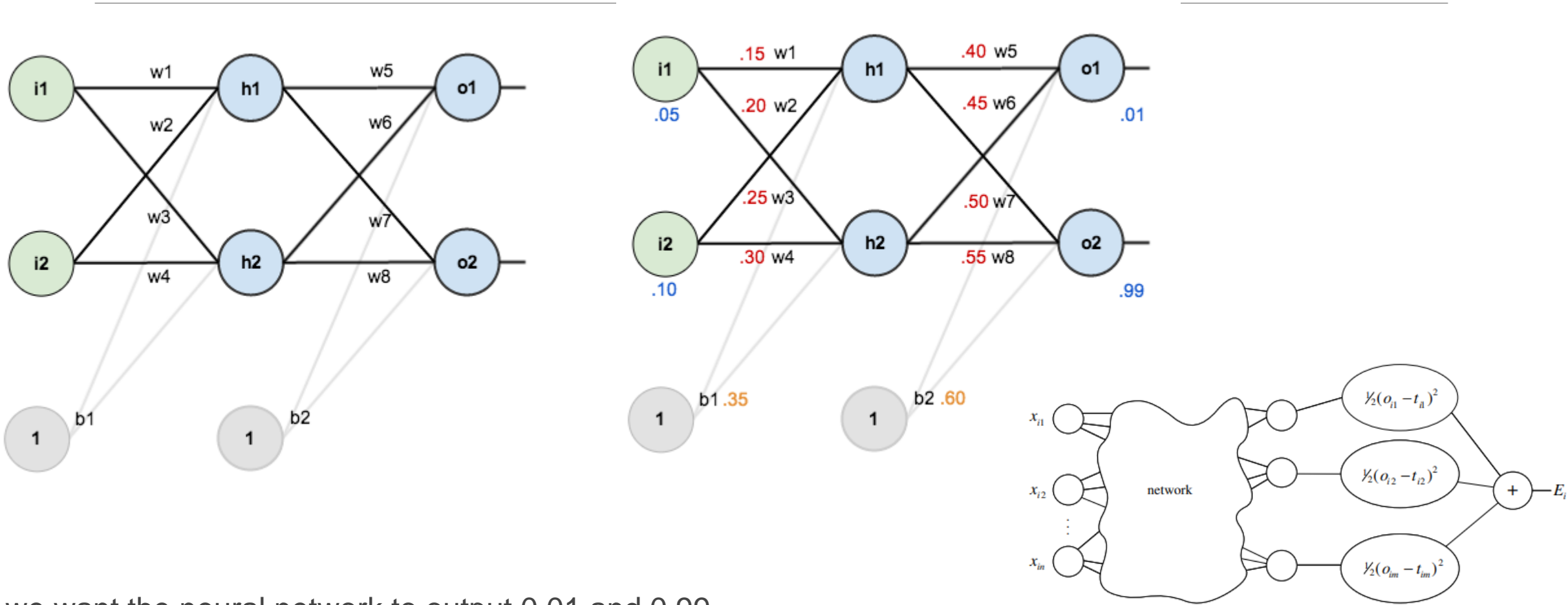


Learning rate and Gradient Descent



If gradient descent is working properly, the cost function should decrease after every iteration. For gradient descent to reach the local minimum, the learning rate should be set at an appropriate value, which is neither too low nor too high. This is important because if the steps it takes are too big, it may not reach the local minimum because it bounces back and forth between the convex function of gradient descent. If we set the learning rate to a very small value, gradient descent will eventually reach the local minimum but that may take a while.

Q1) Find the mean square error(MSE) of outputs and total error MSE after first forward propagation for given MLP? Consider sigmoid as activation function



we want the neural network to output 0.01 and 0.99.

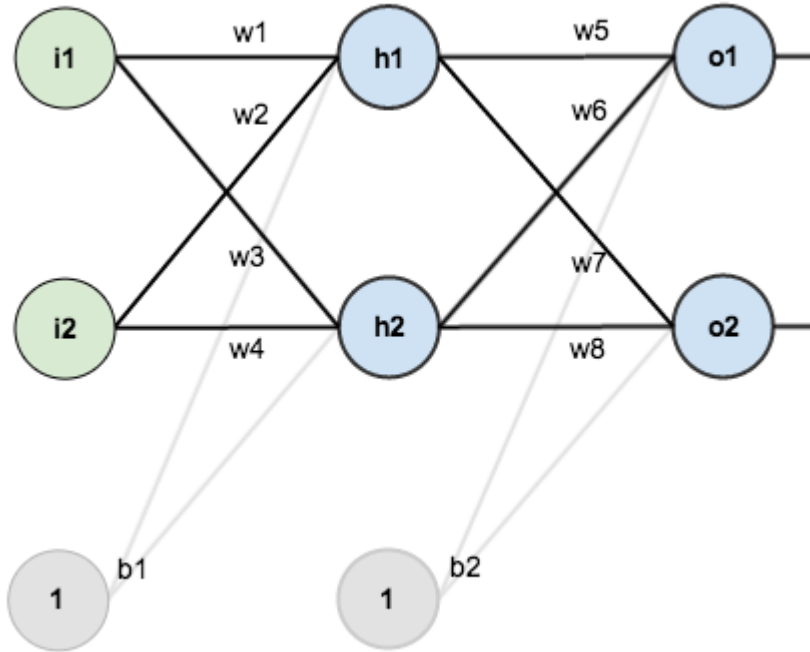
Hint; $E_{total} = 0.298$ (appr)

Apply Back Propagation to Q1

a) Find the updated weights of MLP?

b) Find the new predicted outputs with the updated weight?

c) Find the new cost function value after one iteration of back propagation.

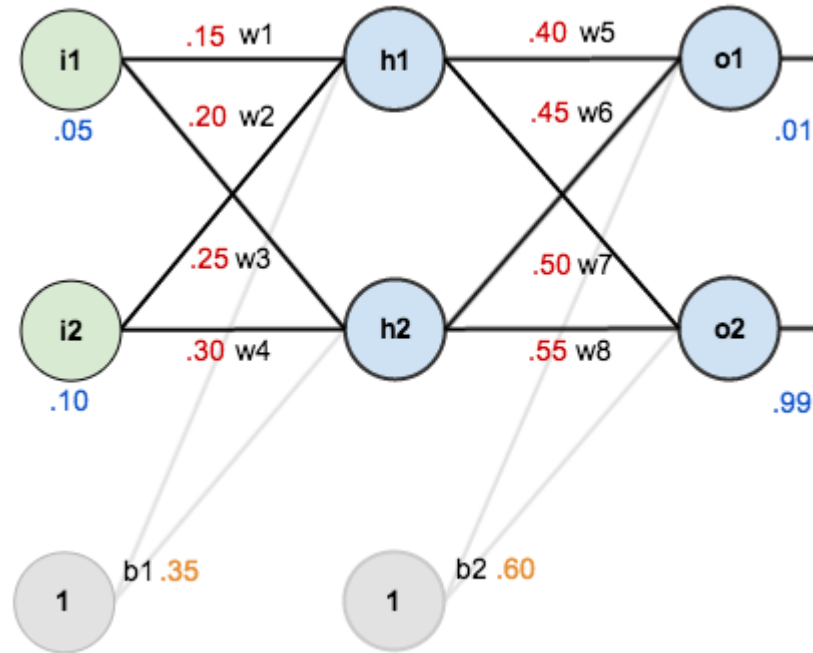


$w_{1\text{new}}=0.149780716$, $w_{2\text{new}}=0.19956143$,
 $w_{4\text{new}}=0.29950229$, **$w_{5\text{new}}=0.35891648$** ,

$w_{3\text{new}}=0.24975114$,
 $w_{6\text{new}}=0.408666186$,

$w_{7\text{new}}=0.511301270$,

$w_{8\text{new}}=0.561370121$



Hint; $E_{\text{total new}} = 0.291$ (appr)

THANK YOU 😊

A solid green horizontal bar at the bottom of the slide.