VIT BHOPAL
www.vitbhopal.ac.in

Supervised Learning
(Understanding a Neural Network to Predict a Test Score Based on Defined Features)

Dr. Virendra Singh Kushwah

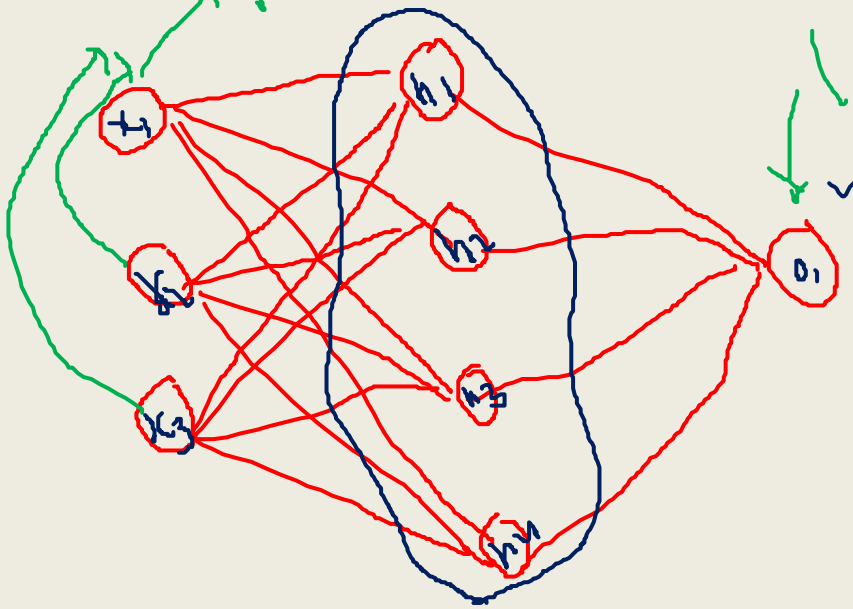
Assistant Professor Grade-II

School of Computing Science and Engineering

Virendra.Kushwah@vitbhopal.ac.in

7415869616

- With approximately 100 billion neurons, the human brain processes data at speeds as fast as 268 mph (miles per hour). In essence, a neural network is a collection of neurons connected by synapses. This collection is organized into three main layers: the input layer, the hidden layer, and the output layer. You can have many hidden layers, which is where the term deep learning comes into play. In an artificial neural network, there are several inputs, which are called features, and produce a single output, which is called a label.



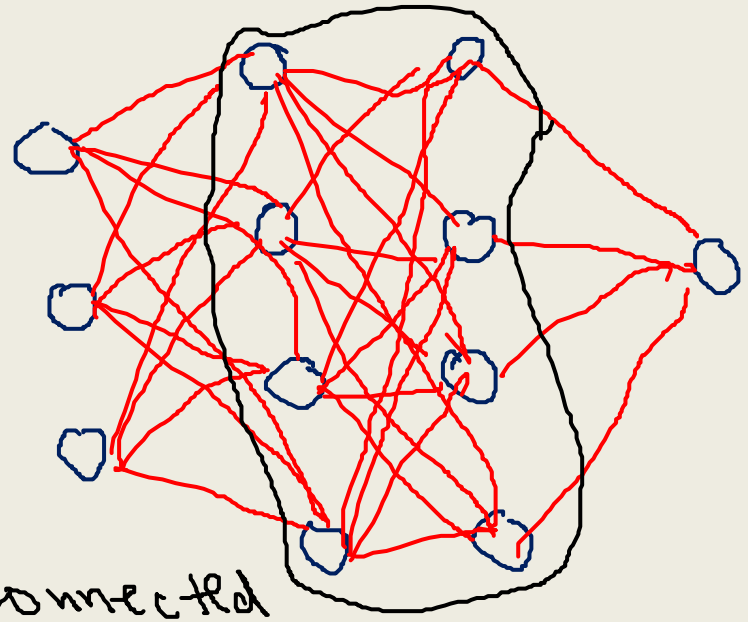
1 OL

1 HL

Fully feed forward Connected

Single layer NN

A perceptron



Fully connected feed forward

multilayer NN

multilayer perceptron or multiple perceptron

Hours Studied, Hours Slept (input)	Test Score (output)
2, 9	92
1, 5	86
3, 6	89
4, 8	?

Features

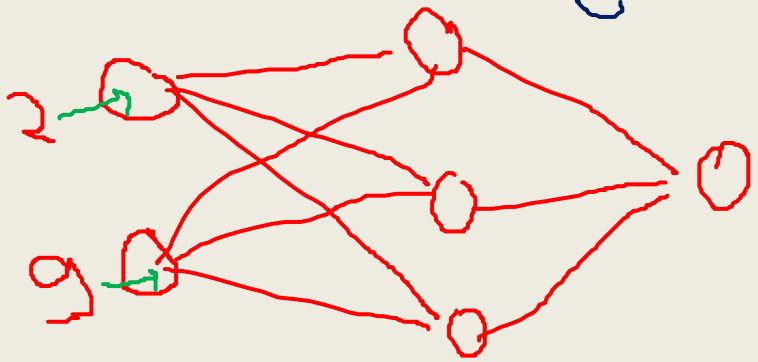
Actual

O & 1

0.92

Features = 2

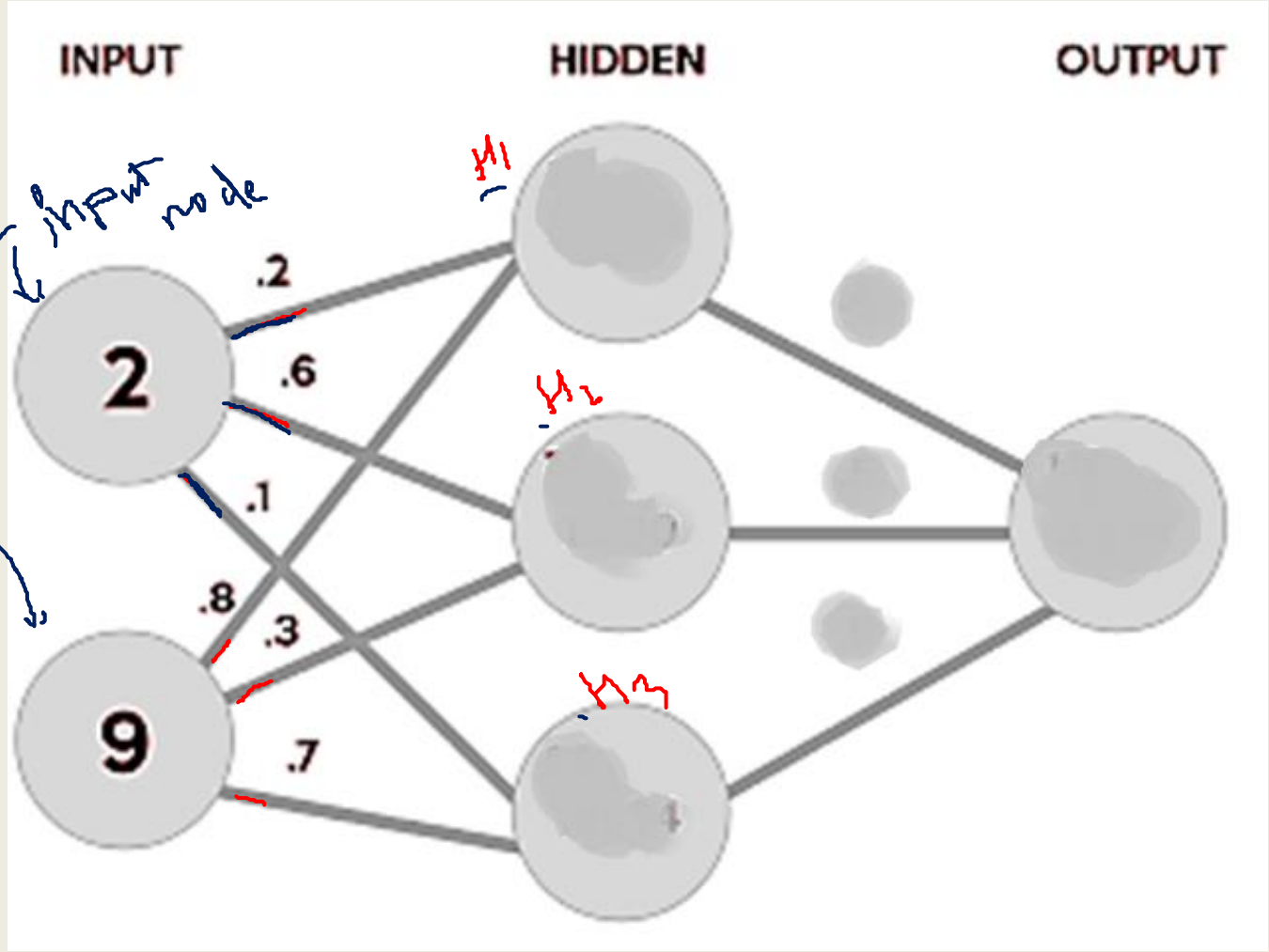
Today no. of input nodes @ IL



It will decide the no. of nodes based on no. of weights @ each input node

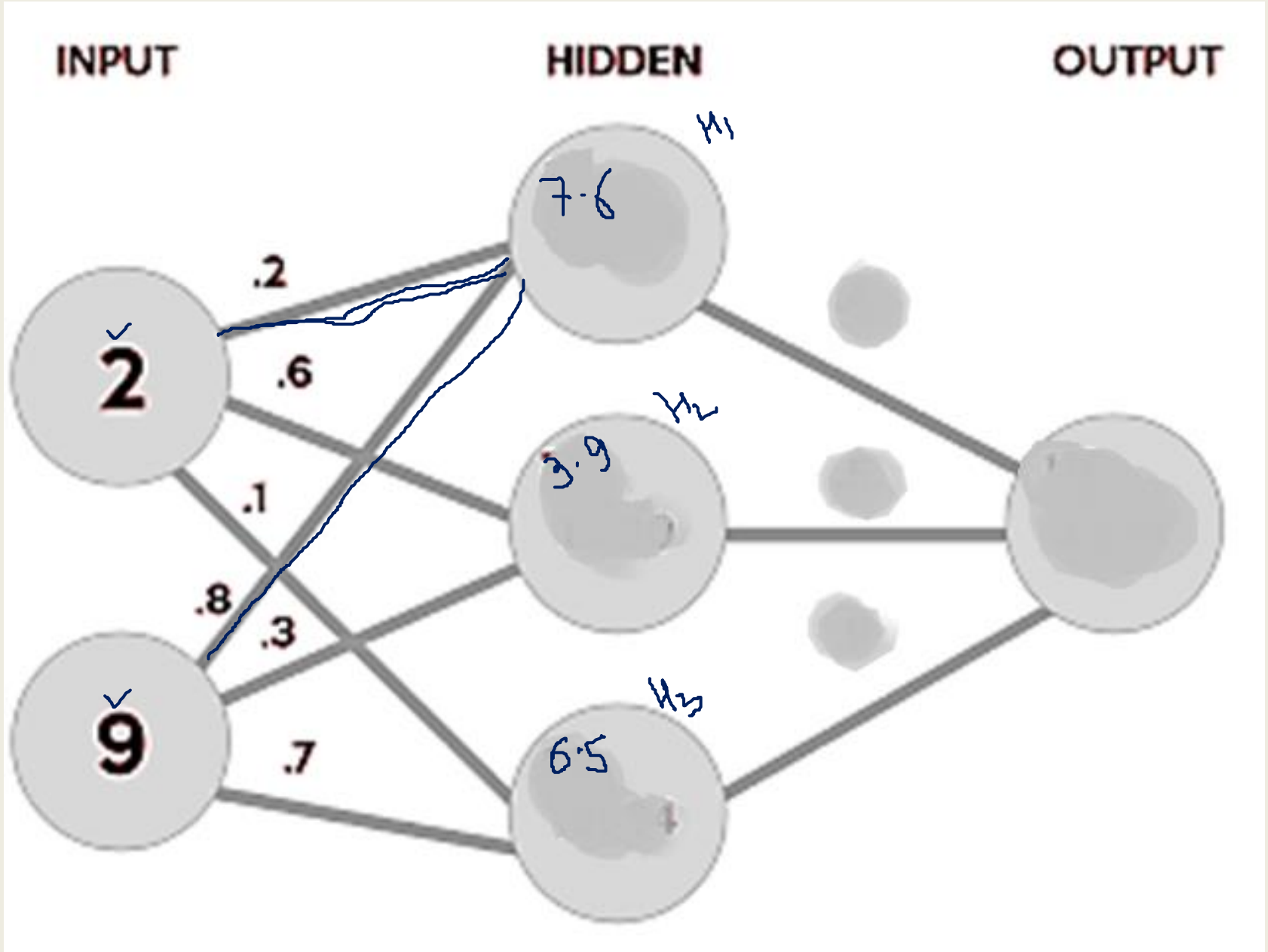
The calculations behind our network

- In the data set, our input data, X , is a 3x2 matrix (means 3 weights for 2 input). Our output data, y , is a 3x1 matrix (means 3 weights for one output). Each element in matrix X needs to be multiplied by a corresponding weight and then added together with all the other results for each neuron in the hidden layer. Here's how the first input data element (2 hours studying and 9 hours sleeping) would calculate an output in the network:



- This image breaks down what our neural network actually does to produce an output. First, the products of the random generated weights (.2, .6, .1, .8, .3, .7) on each synapse and the corresponding inputs are summed to arrive as the first values of the hidden layer. These sums are in a smaller font as they are not the final values for the hidden layer.

- To get the final value for the hidden layer, we need to apply the activation function. The role of an activation function is to introduce nonlinearity. An advantage of this is that the output is mapped from a range of 0 and 1, making it easier to alter weights in the future.



Calculate input value at hidden for each hidden node - h_1, h_2, h_3

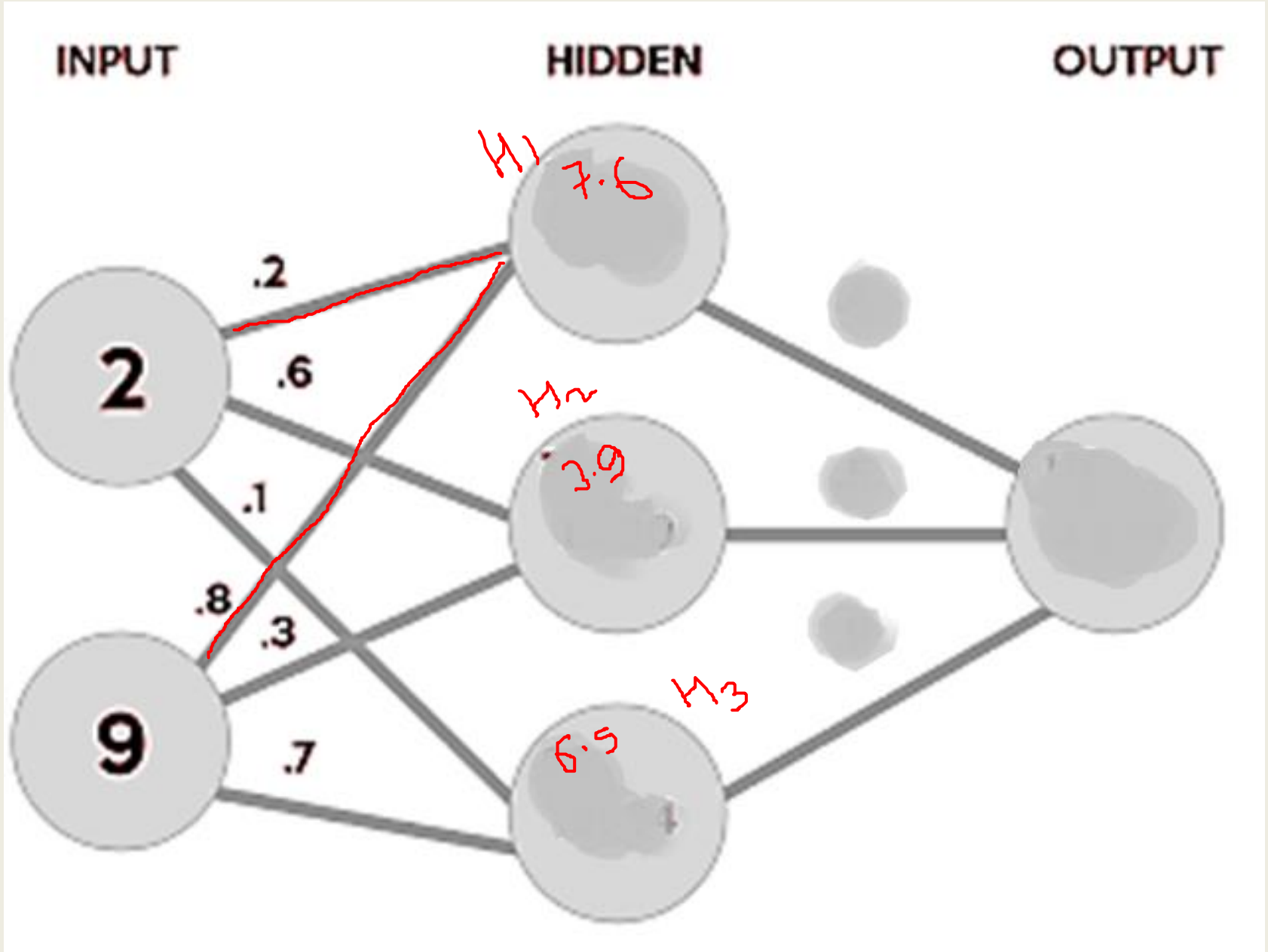
$$h_j = \sum_{i=1}^{n} x_i w_i$$

$$= 2 \times 0.2 + 9 \times 0.8$$

$$= 0.4 + 7.2 = 7.6$$

$h_2 = 3.9$

$h_3 = 6.5$



Input value for hidden layer

H_1, H_2, H_3 $\sum_{i=1}^{n} w_i x_i$

$H_1 =$

$2 \times 0.2 +$

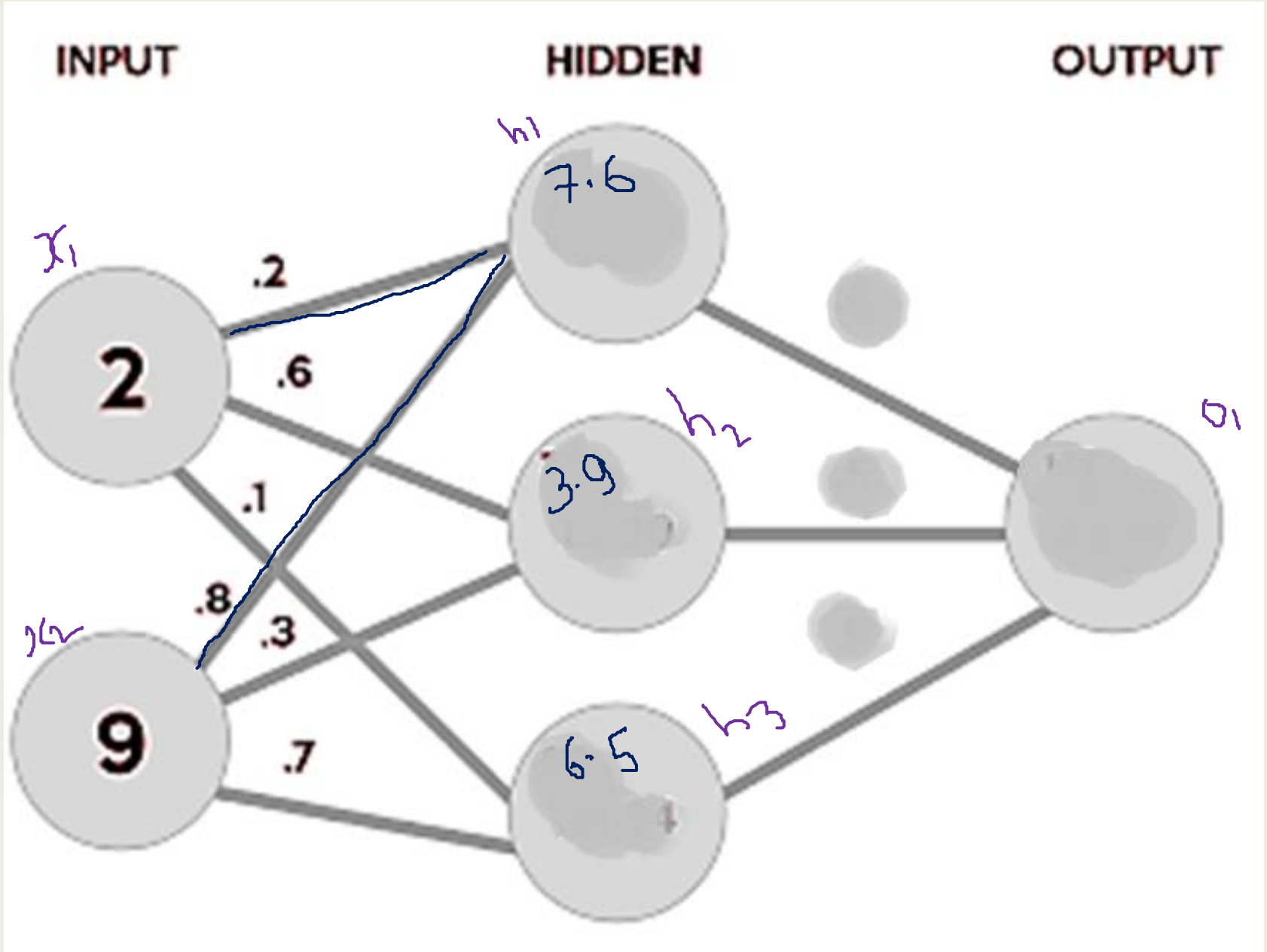
$9 \times 0.8 : 0.4$

$+7.2$

7.6

$H_2 = 3.9$

$H_3 = 6.5$



input value for h_1

$$\sum_{i=0} x_i w_i$$

$2 \times 0.2 = 0.4$

$9 \times 0.8 = 7.2$

7.6

$h_2 = 3.9$

$h_3 = 6.5$

Summation at Hidden layer from Input layer

VIT®
BHOPAL
www.vitbhopal.ac.in

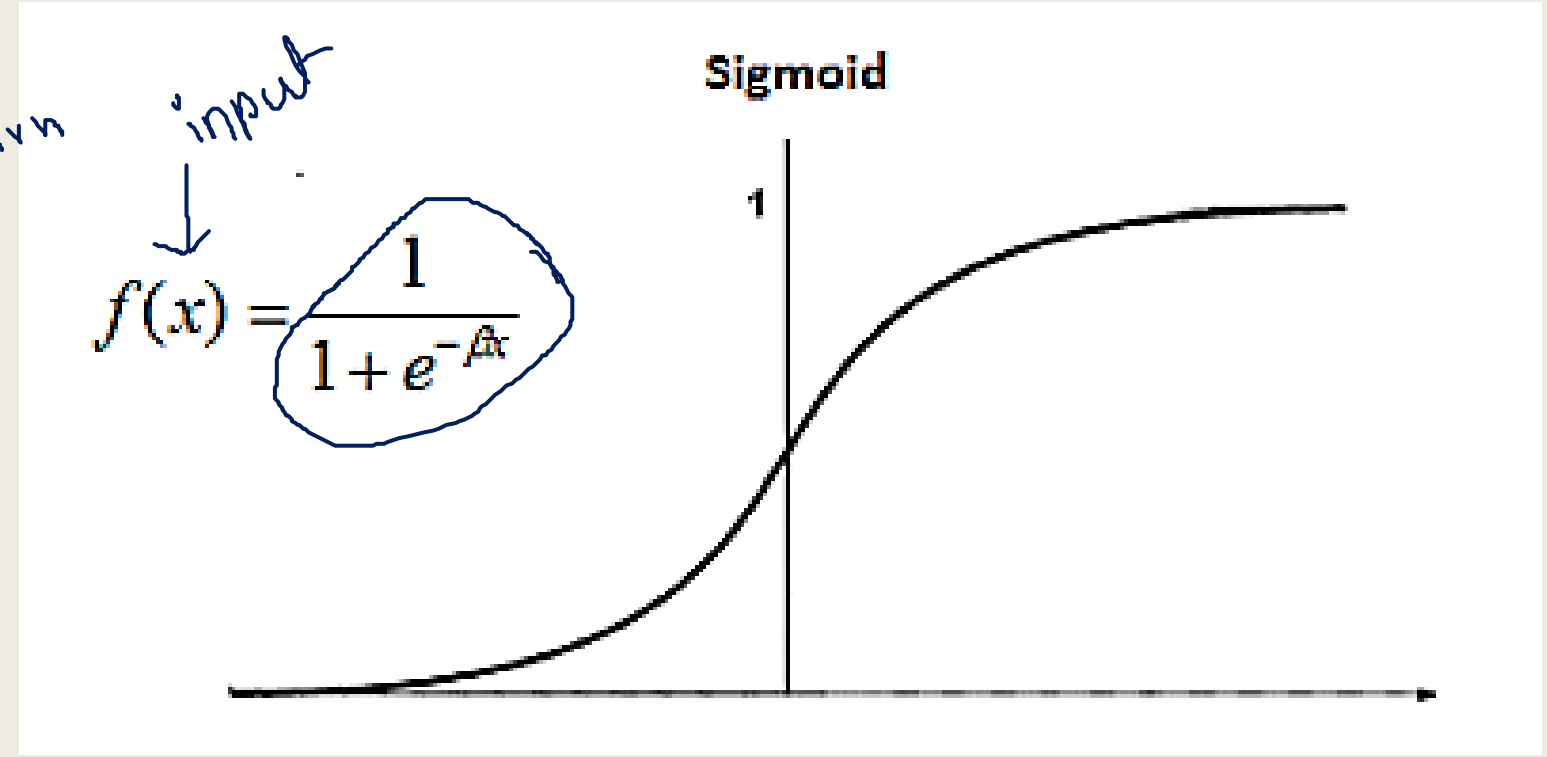
- H1 = (2 * .2) + (9 * .8) = 7.6
- H2 = (2 * .6) + (9 * .3) = 3.9
- H3 = (2 * .1) + (9 * .7) = 6.5

tanh, Sigmoid, ReLU, Softmax etc.

- There are many activation functions out there. In this case, we'll stick to one of the more popular ones - the sigmoid function.

always return 0 & 1

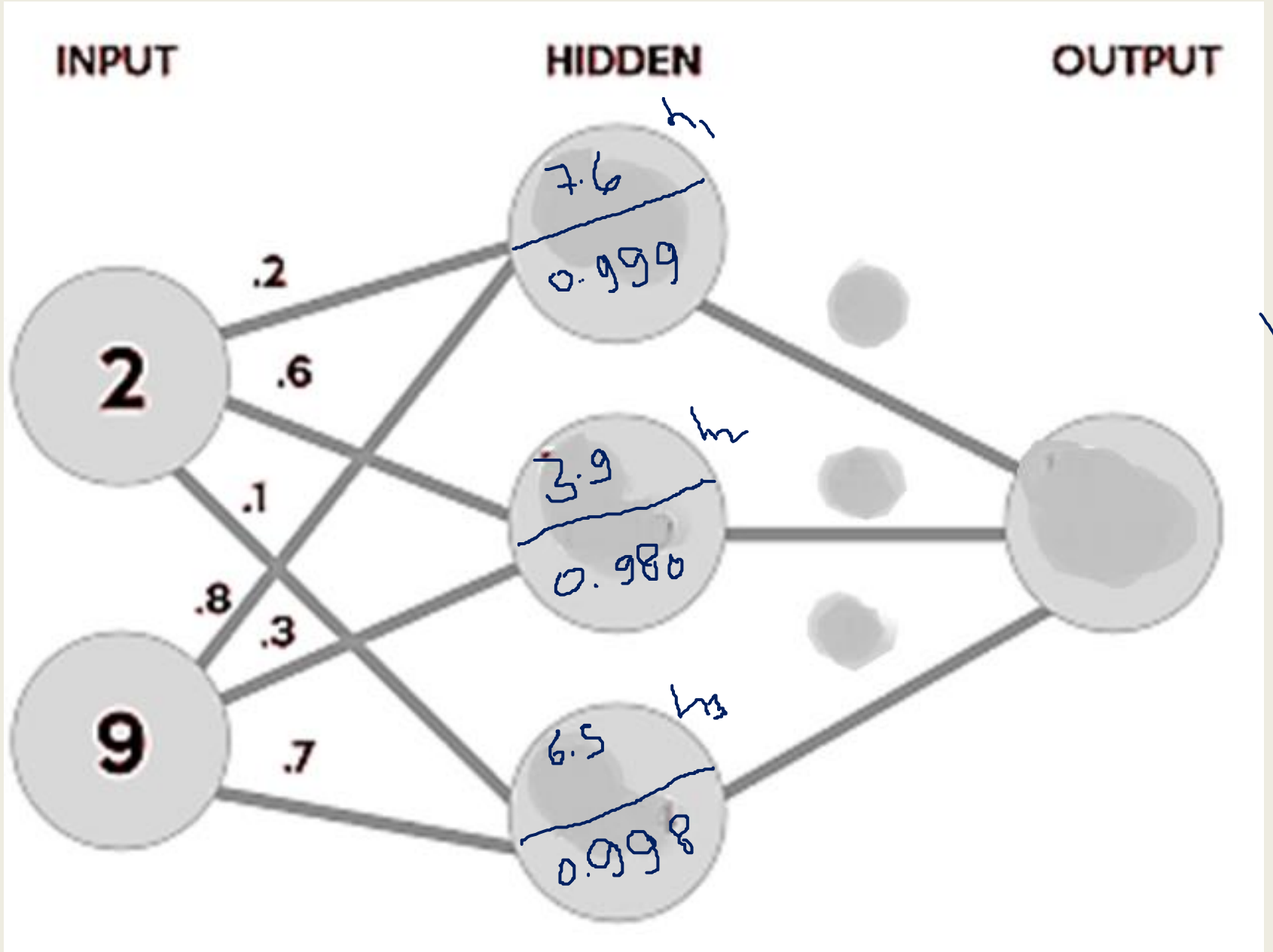
'input'



$\frac{1}{1+e^{-x}}$

or

$\frac{e^{x}}{e^{x}+1}$



Activate a node by using activation fun.

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

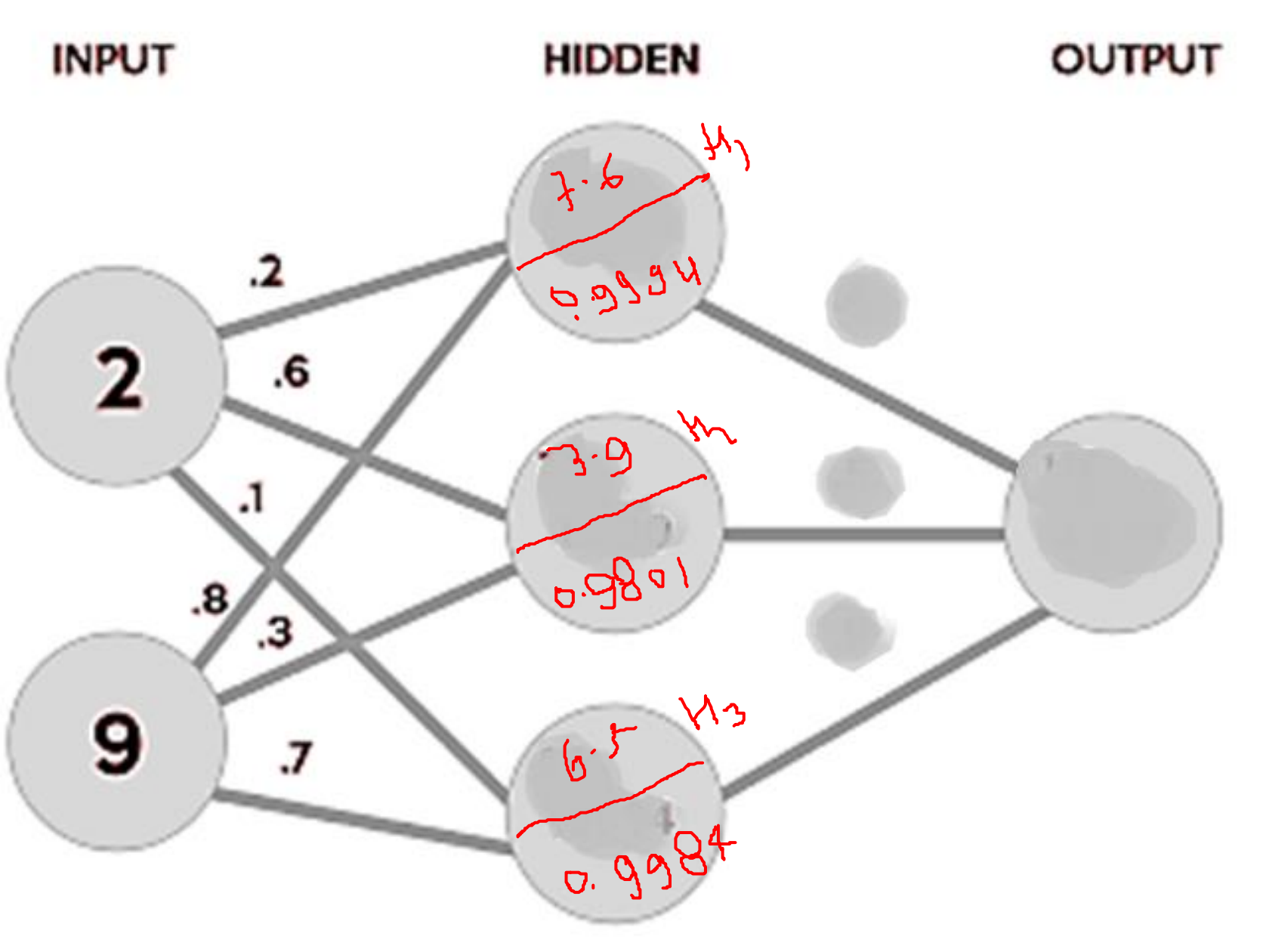
h_1

$e = 2.718$

$x = 7.6$

$$f(7.6) = \frac{e^{7.6}}{e^{7.6} + 1}$$

$$f(7.6) = 0.999$$



Activate node at hidden layer

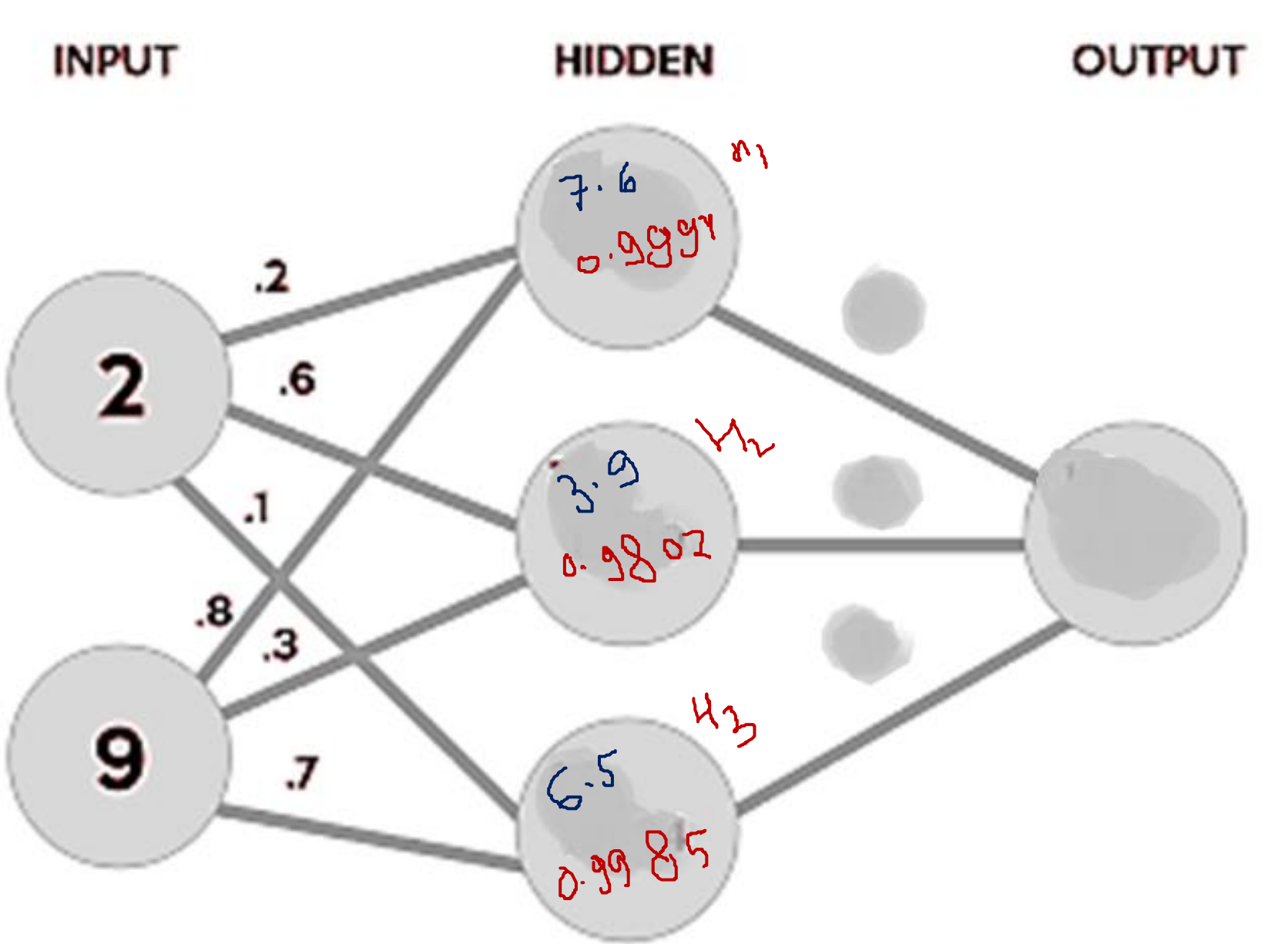
$$f(x) = \frac{e^x}{e^x + 1}$$

H_1

$$f(7.6) = \frac{e^{7.6}}{e^{7.6} + 1}$$

$e = 2.718$

$f(7.6) = 0.9994$



Apply activation function

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

$e = 2.718$

$x = 7.6$ for h_1

$$f(7.6) = \frac{e^{7.6}}{e^{7.6} + 1}$$

$= 0.9994$

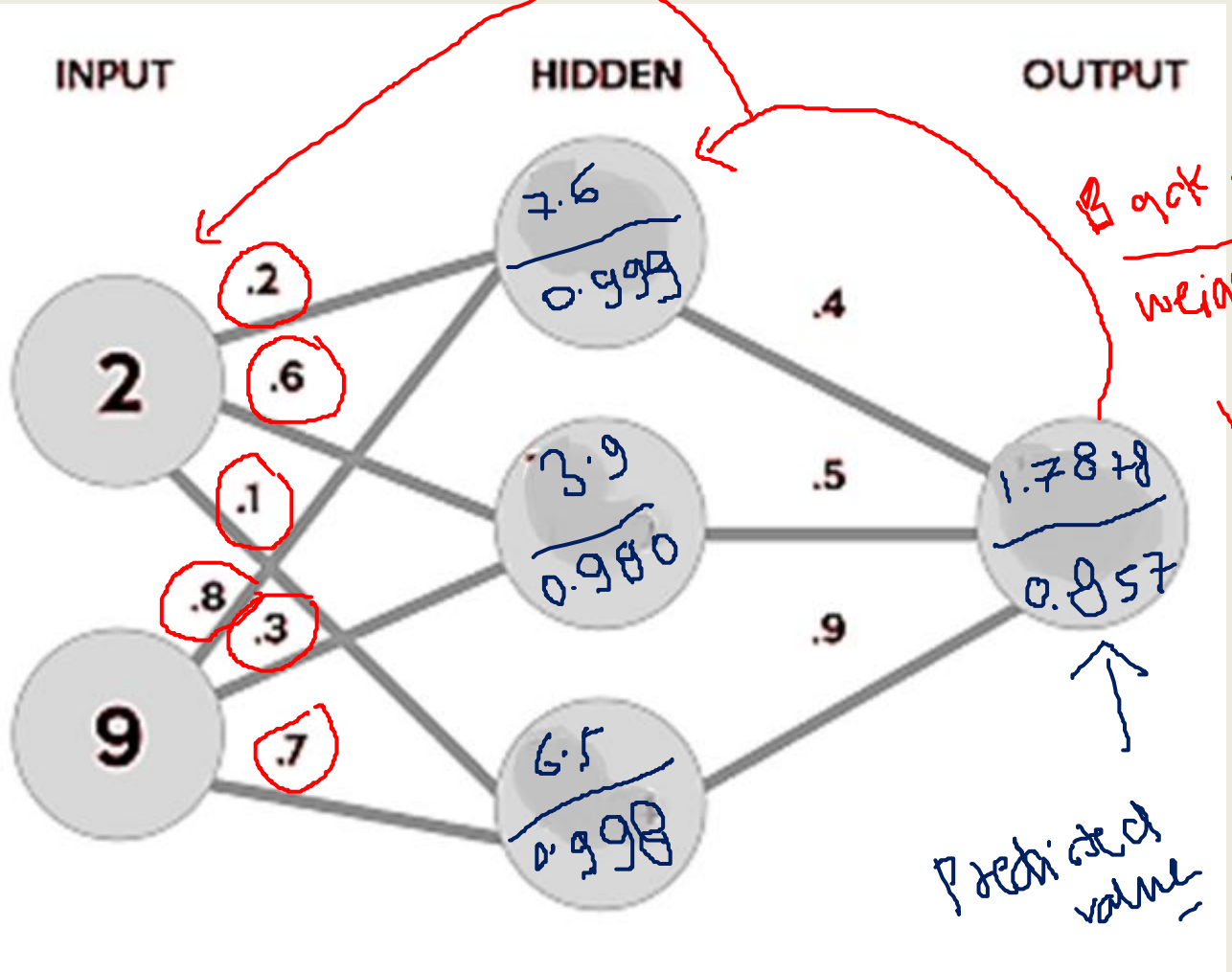
Values after apply Activation function

VIT®
BHOPAL
www.vitbhopal.ac.in

- By default e = 2.718

- S(7.6) = 0.9994
- S(3.9) = 0.9802
- S(6.5) = 0.9985

Now Calculate value for Output Layer



input

Back propagation weights updation

$= 0.999 \times 0.4 +$

$0.980 \times 0.5 +$

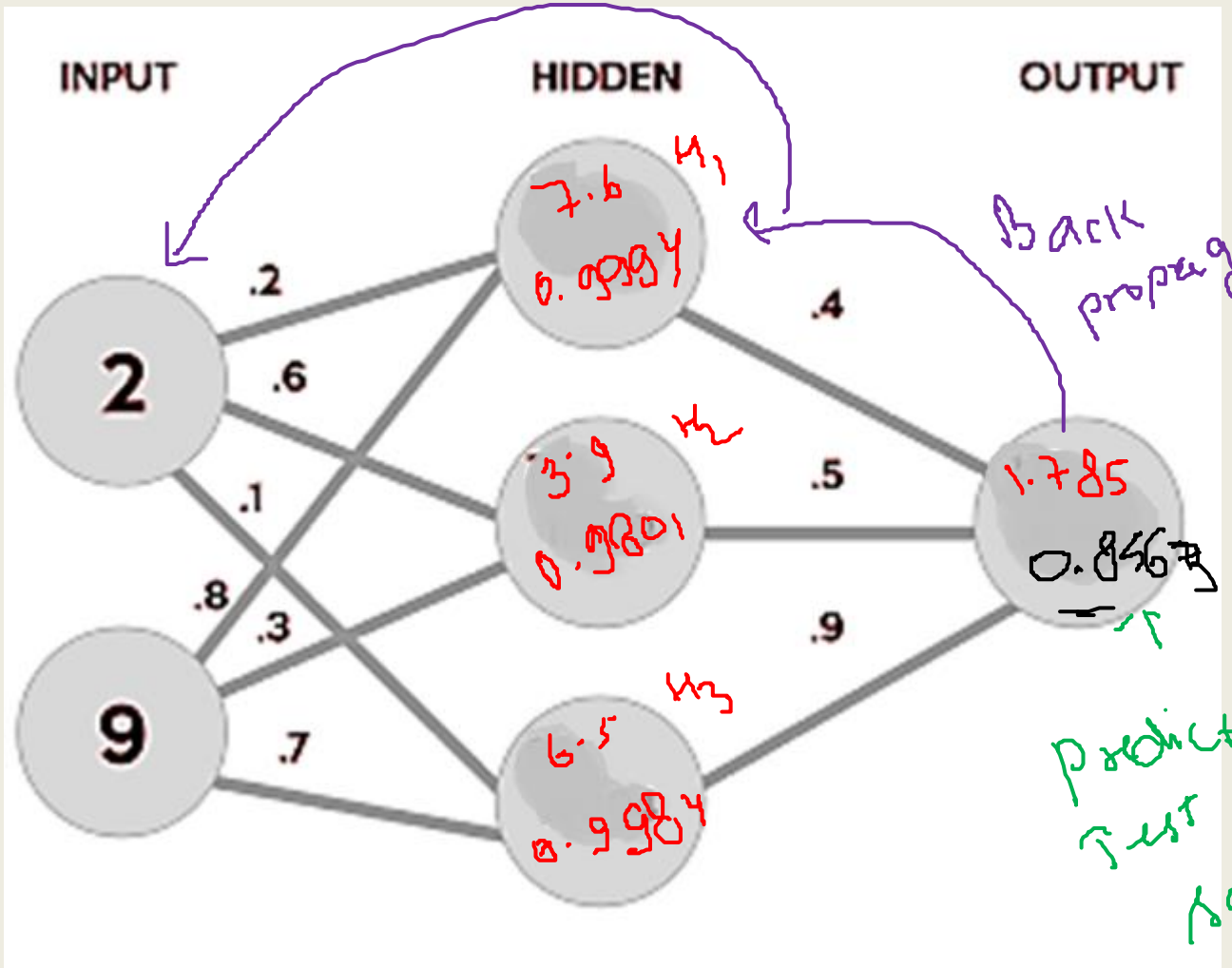
0.998×0.9

$= 1.7878$

output layer output node will be generated by a function (Sigmoid)

$= e^{1.7878} / e^{1.7878} + 1 \Rightarrow 0.857$

Now Calculate value for Output Layer



input value of OL

$= 0.9994 \times 0.4 +$

$0.9801 \times 0.5 +$

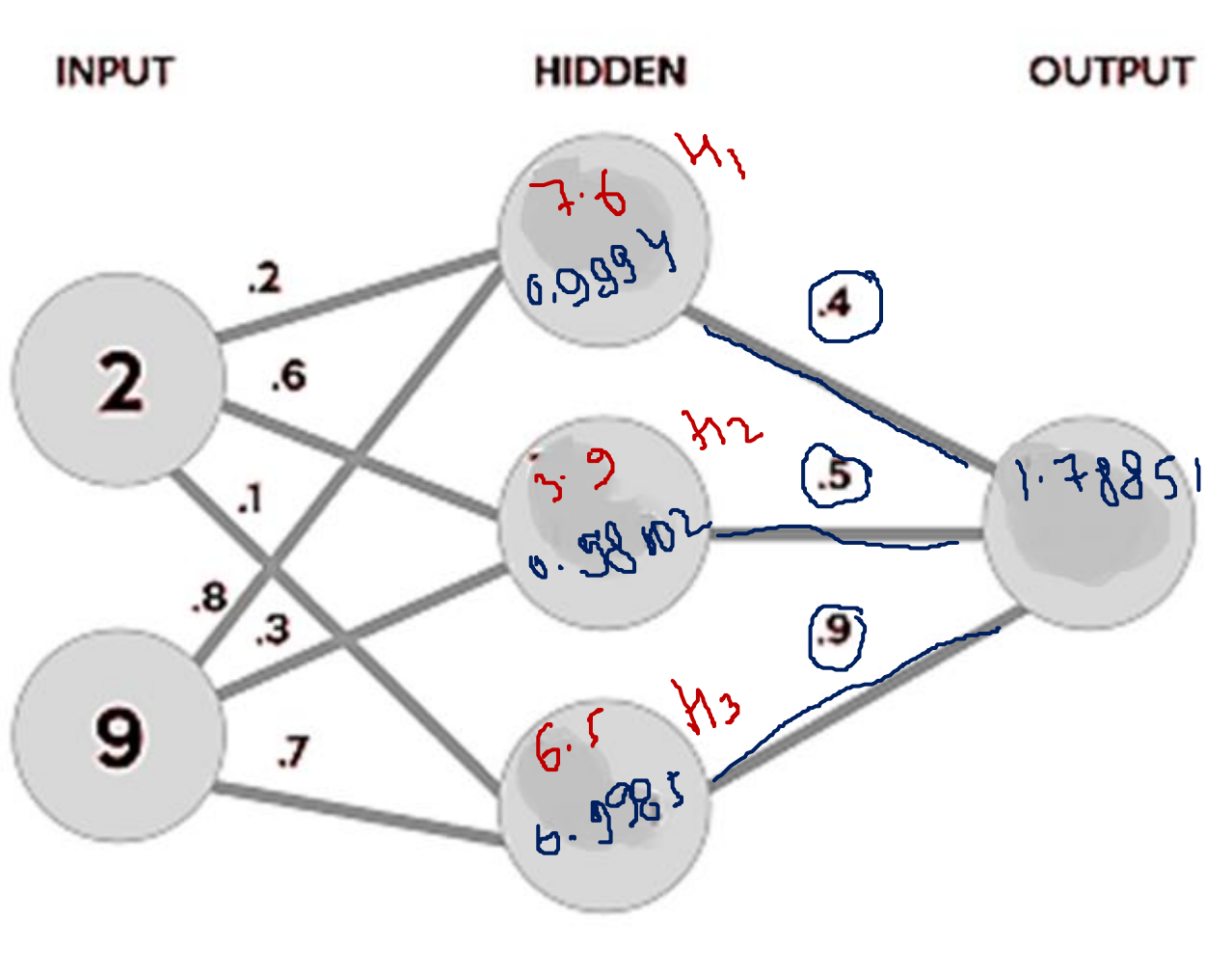
0.9984×0.9

$= 1.785$

output value of output node @ OL

$\Rightarrow \frac{e^{1.785}}{e^{1.785} + 1} \Rightarrow 0.85673$

Now Calculate value for Output Layer



Input value for output layer

$= 0.9994 \times 0.4$

$+ \quad 0.9802 \times 0.5$

$+ \quad 0.9985 \times 0.9$

$= 1.78851$

$=$

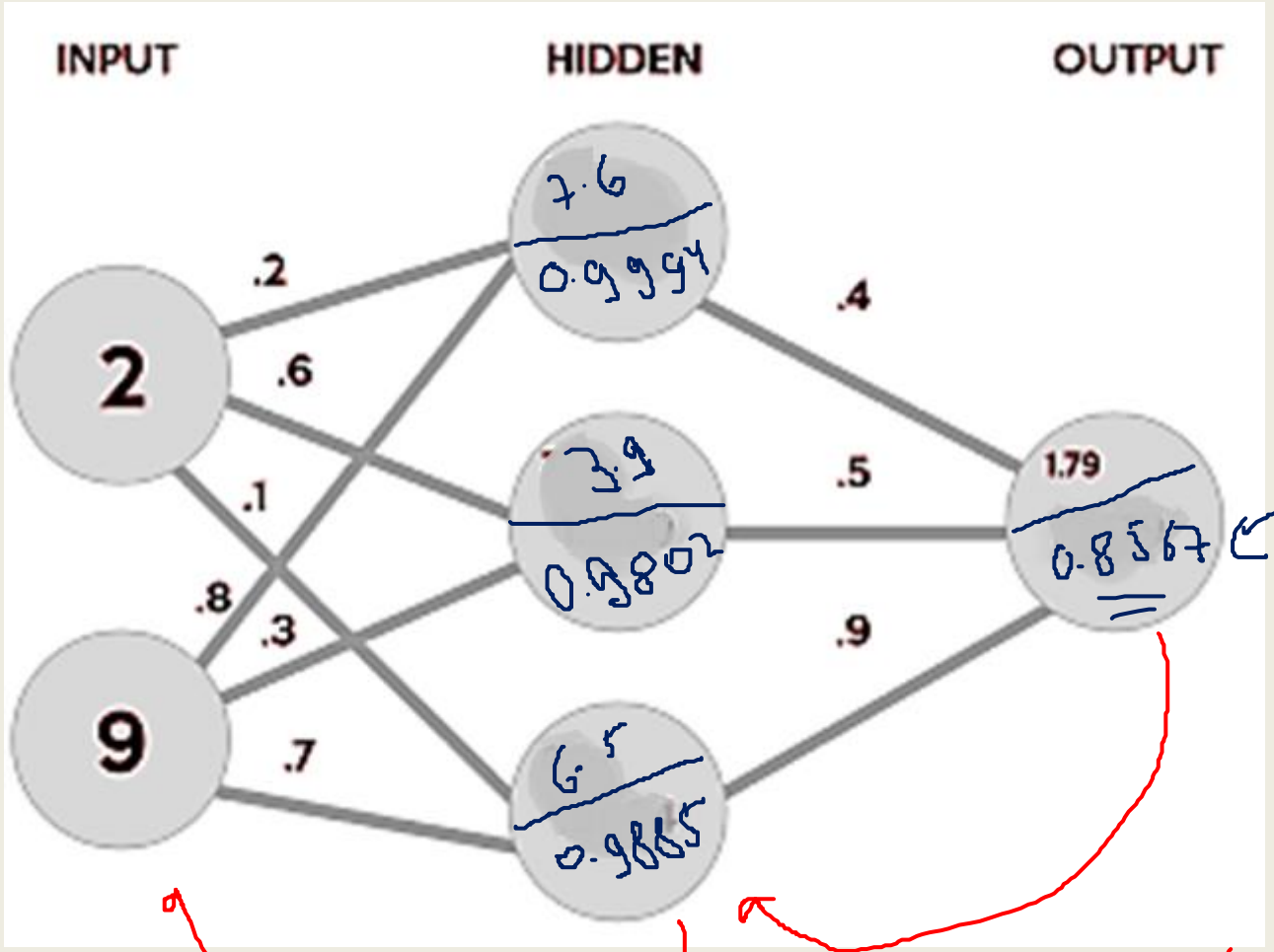
- =0.9994 x 0.4 = 0.39976
- =0.9802 x 0.5 = 0.49010
- =0.9985 x 0.9 = 0.89865

$\frac{e^x}{e^x+1} = \frac{e^{1.7885}}{e^{1.7885}+1}$

$= 0.8567$

- Get the summation of these values
- =1.78

- After apply activation function by using Sigmoid function = 0.85


INPUT
HIDDEN
OUTPUT
2
9
.2
.6
.1
.8
.3
.7
.4
.5
.9
7.6
0.9994
3.9
0.9802
6.5
0.9865
1.79
0.8567
Predicted value
Back propagation
update weights

- Theoretically, with those weights, our neural network will calculate .85 as our test score! However, our target was .92 . Our result wasn’t poor, it just isn’t the best it can be. We just got a little lucky when I chose the random weights for this example.

Calculating error

$0.5 \times (0.85 - 0.92)^2$

$0.5 \times (-0.7)^2$

$0.5 \times 0.49 = 0.245$

- One way of representing the loss function is by using the mean sum squared loss function:

$$\text{Loss} = \sum (0.5)(o-y)^2$$

$0.5 \times (0.85 - 0.92)^2$

$0.5 \times (-0.7)^2$

$= 0.5 \times 0.49$

$\Rightarrow 0.245$

- In this function, o is our predicted output, and y is our actual output. The mean sum squared loss function is the sum, over all the data points, of the square of the difference between the predicted and actual target variables, divided by the number of data points.
- Now that we have the loss function, our goal is to get it as close as we can to 0. That means we will need to have close to no loss at all. As we are training our network, all we are doing is minimizing the loss.

Back propagation

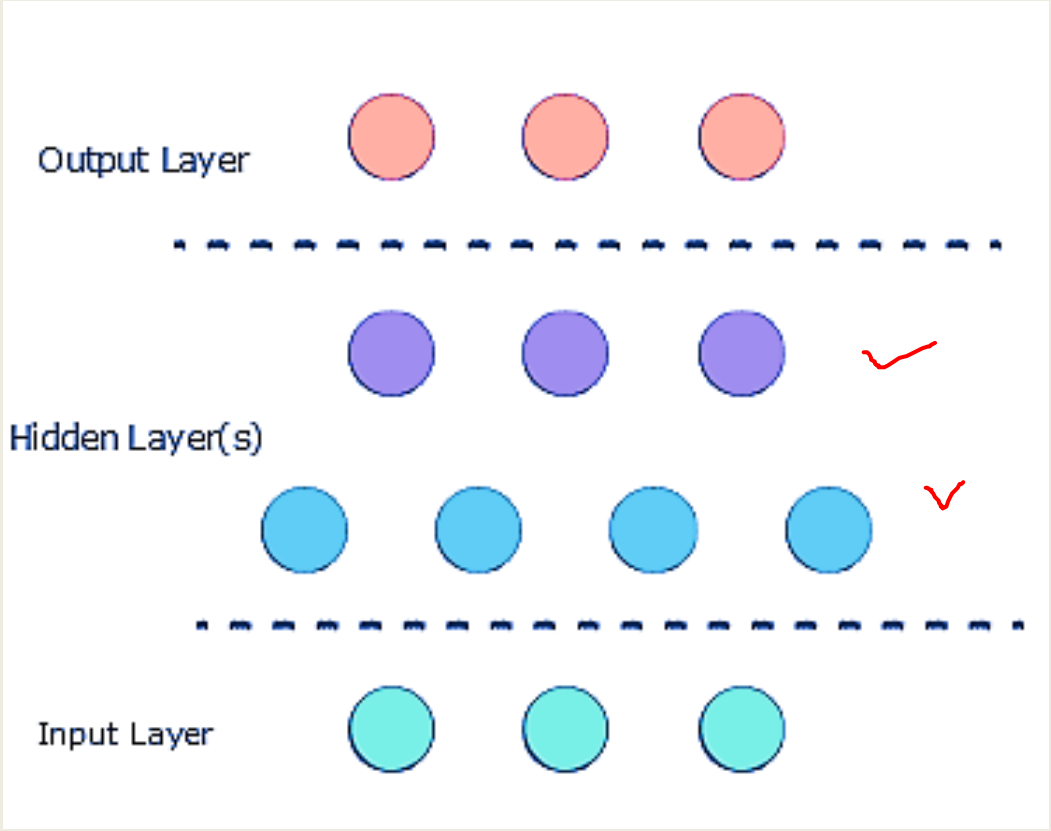
What is a multi-layered perceptron? Or Multiple Perceptron

- A multi-layered perceptron (MLP) is one of the most common neural network models used in the field of deep learning. Often referred to as a “vanilla” neural network, an MLP is simpler than the complex models of today’s era. However, the techniques it introduced have paved the way for further advanced neural networks.

- The multilayer perceptron (MLP) is used for a variety of tasks, such as stock analysis, image identification, spam detection, and election voting predictions.

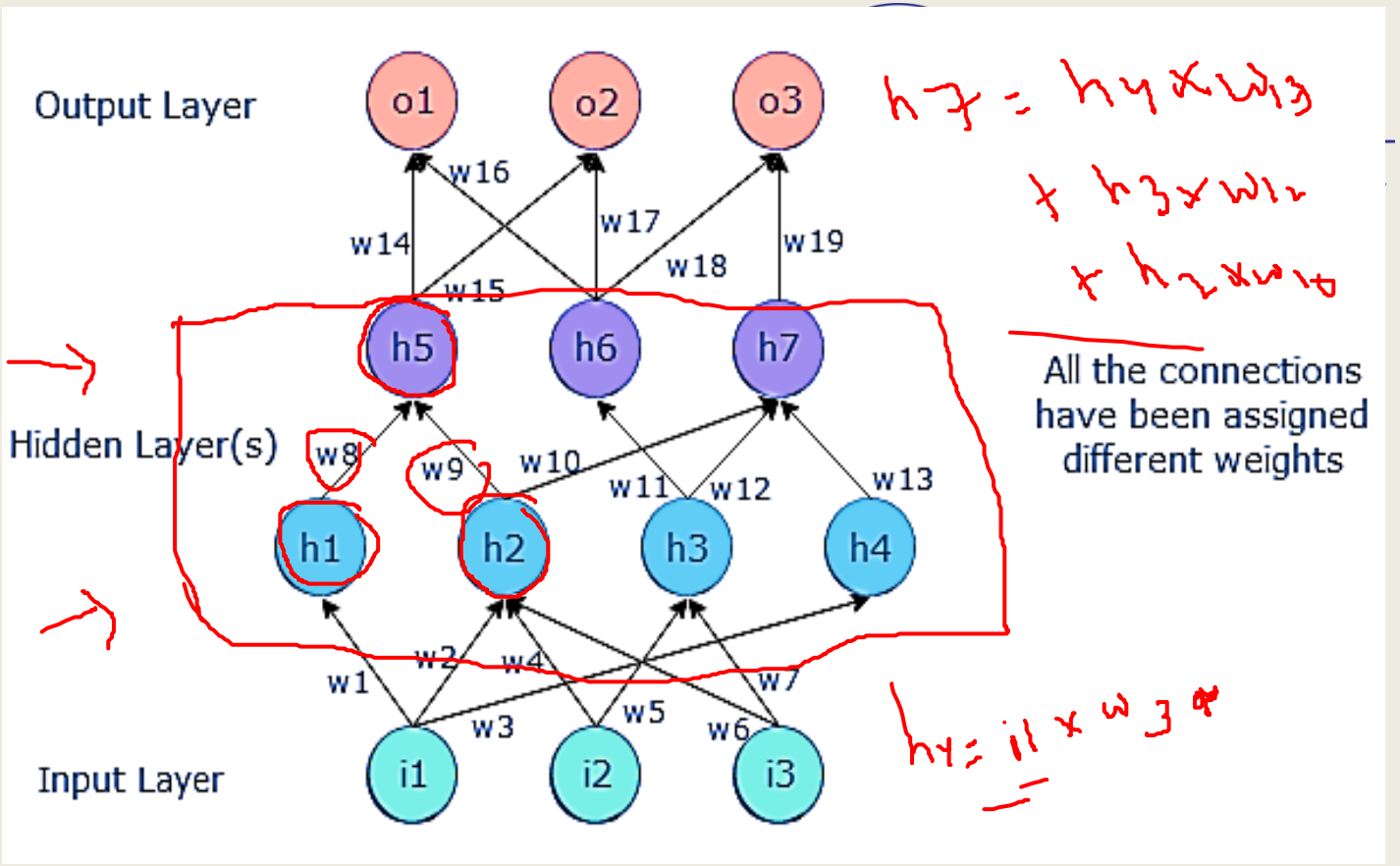
The Basic Structure

- A multi-layered perceptron consists of interconnected neurons transferring information to each other, much like the human brain. Each neuron is assigned a value. The network can be divided into three main layers.
- Input Layer - This is the initial layer of the network which takes in an input which will be used to produce an output.
- Hidden Layer(s) -The network needs to have at least one hidden layer. The hidden layer(s) perform computations and operations on the input data to produce something meaningful.
- Output Layer-The neurons in this layer display a meaningful output.



- The MLP is a feedforward neural network, which means that the data is transmitted from the input layer to the output layer in the forward direction.

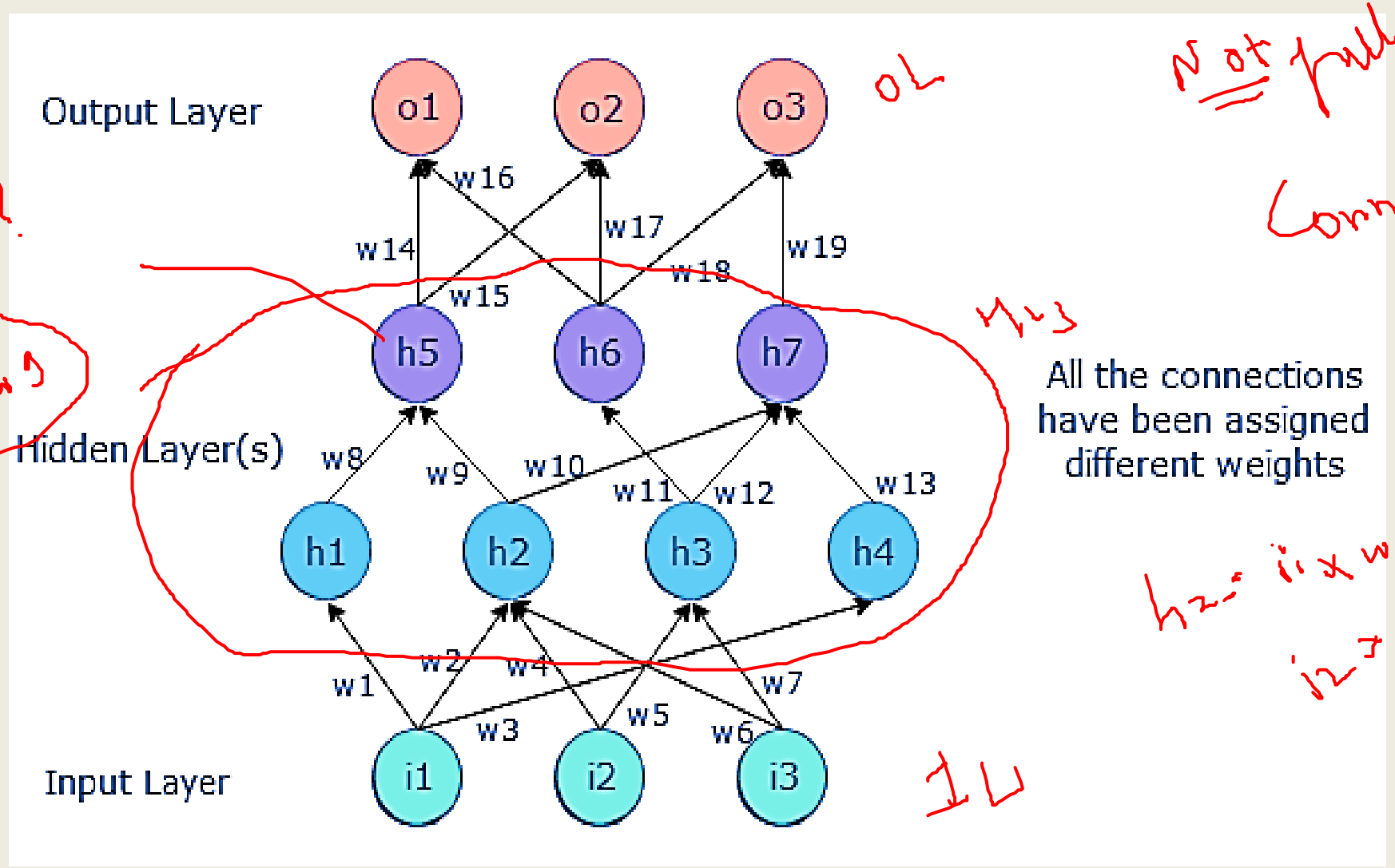
- The connections between the layers are assigned weights. The weight of a connection specifies its importance. This concept is the backbone of an MLP's learning process.



While the inputs take their values from the surroundings, the values of all the other neurons are calculated through a mathematical function involving the weights and values of the layer before it.

For example, the value of the h5 node could be:

h5=h1.w8+h2.w9


Output Layer
o1
o2
o3
w16
w17
w14
w19
w18
w15
h5
h6
h7
Hidden Layer(s)
w8
w9
w10
w11
w12
w13
h1
h2
h3
h4
w1
w2
w4
w7
w3
w5
w6
i1
i2
i3
Input Layer
All the connections have been assigned different weights

VIT®

BHOPAL

www.vitbhopal.ac.in