Back propagation Neural Network (BPNN)

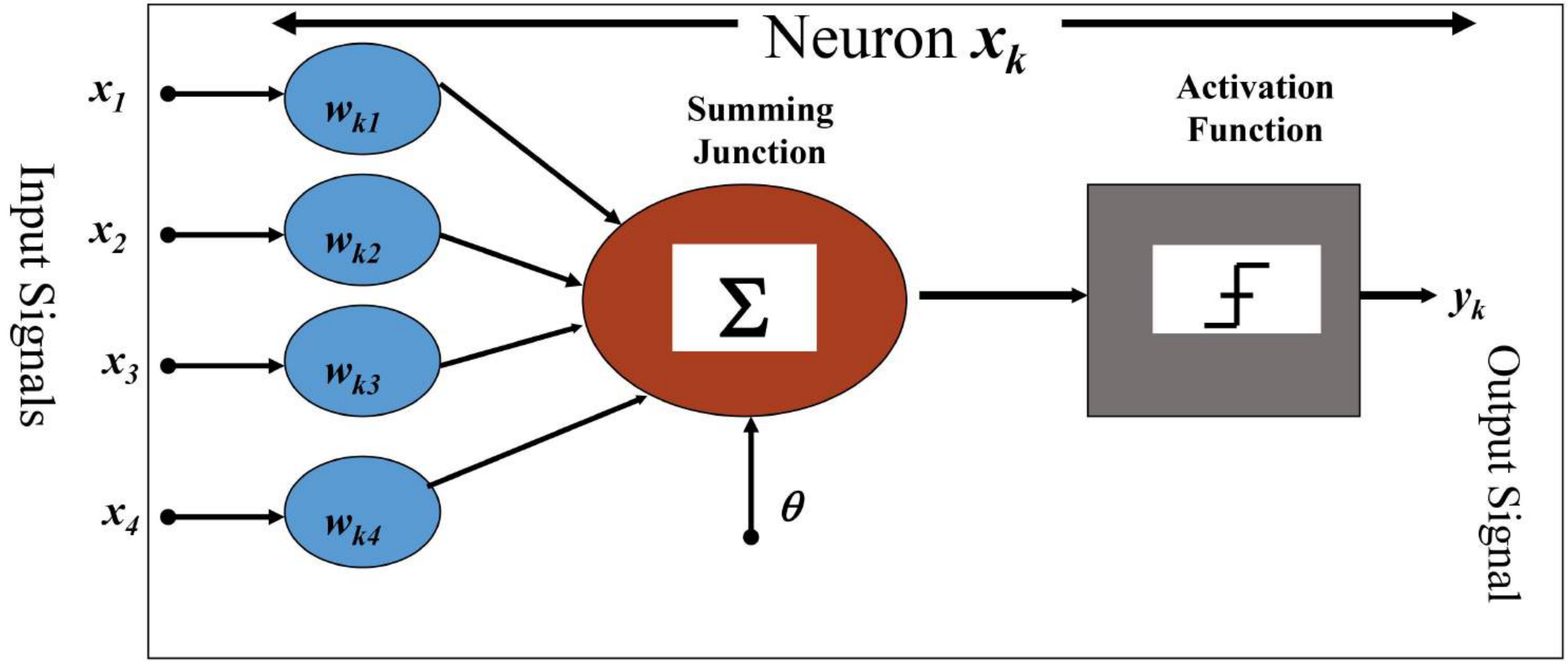
Dr. Nasima Begum
Associate Professor
Dept. of CSE, UAP

Neural Networks

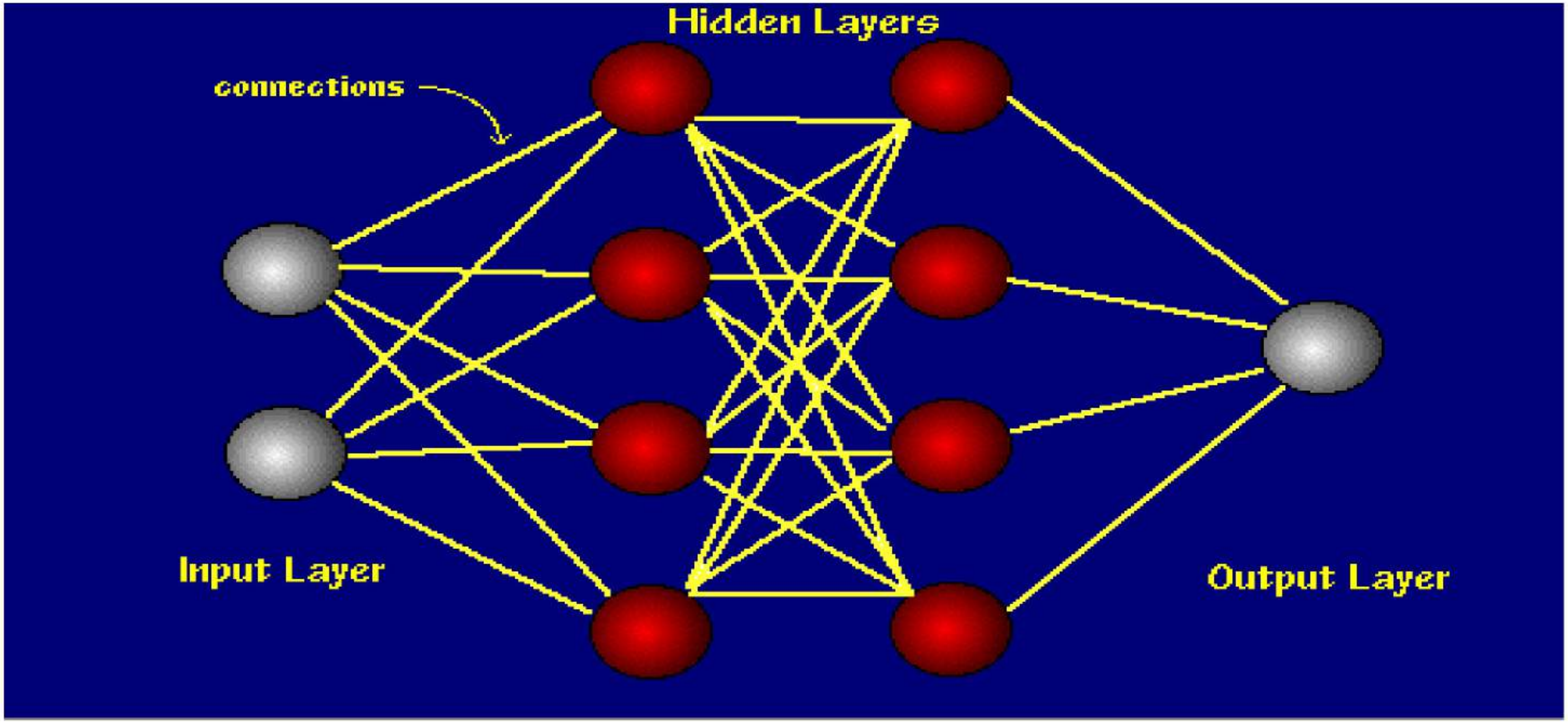
Outline:

1. Introduction to Neural Networks
2. Perceptron
3. **Introduction to Backpropagation NN**
4. Associative Memory: Hopfield Nets
5. Summary

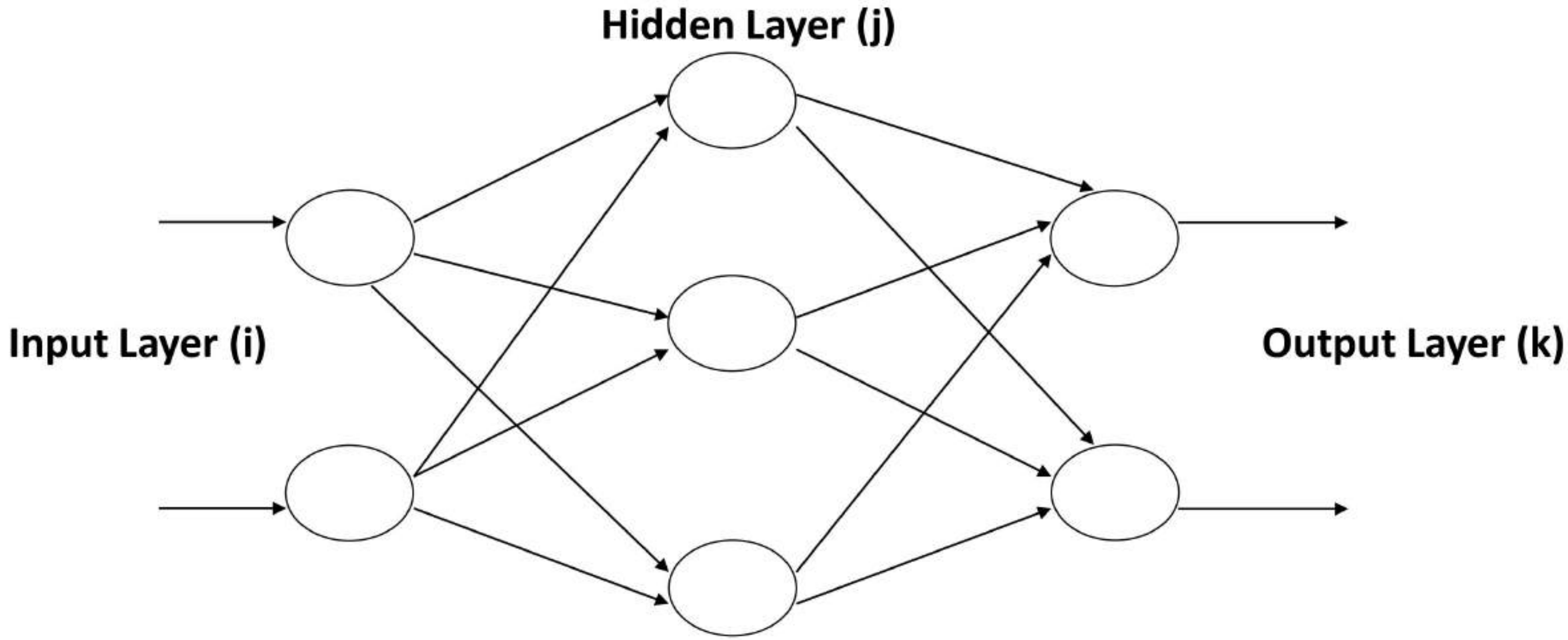
ANN's: A Schematic View



Backpropagation Neural Networks



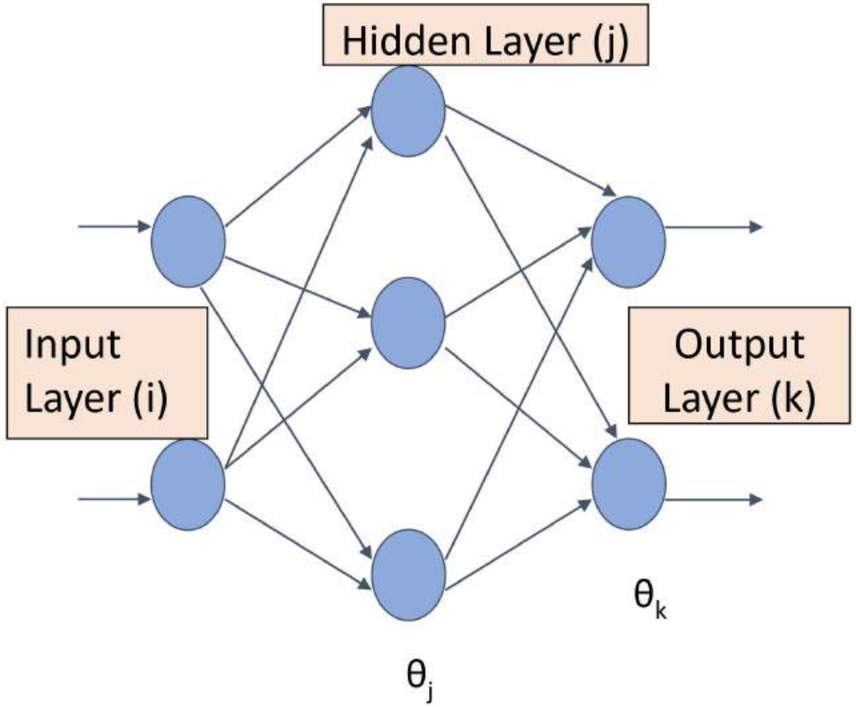
What is a Backpropagation NN?



- A back-propagation neural network is a multi-layer network and the layers are fully connected, that is, **every neuron in each layer is connected to every other neuron in the adjacent forward layer.**

Backpropagation NN…

- In a back-propagation neural network, the learning algorithm has two phases (forward propagation and backward propagation).
- First, a training input pattern/feature vector is feeded to the network input layer.
- The network then propagates the input pattern from layer to layer until the output pattern is generated by the output layer.
- 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 updated as the error is propagated.



- To derive the back-propagation learning law, let us consider the three-layer network shown in **Fig.** The indices, ***i,j,k***, here refer to neurons in the input, hidden and output layers, respectively.

Backpropagation NN…

- Input signals are propagated through the network from left to right, and error signals propagated from right to left.
- The symbol $\mathbf{w_{ij}}$ denotes the weight for the connection between neuron $\boldsymbol{i}$ in the input layer and neuron $\boldsymbol{j}$ in the hidden layer.
- The symbol $\boldsymbol{w_{jk}}$ denotes the weight between *neuron* $\boldsymbol{j}$ in the hidden layer and neuron $\boldsymbol{k}$ in the output layer.
- To propagate error signals, we start at the output layer and work backward to the hidden layer. The error signal at the output of neuron k at iteration p is defined by:

$$e_k(p) = y_{dk}(p) - y_k(p)$$

Backpropagation NN…

- Error Gradient:

  - The rate of change of activation function used, multiplied by the error at iteration "p". It is represented by δ(p).

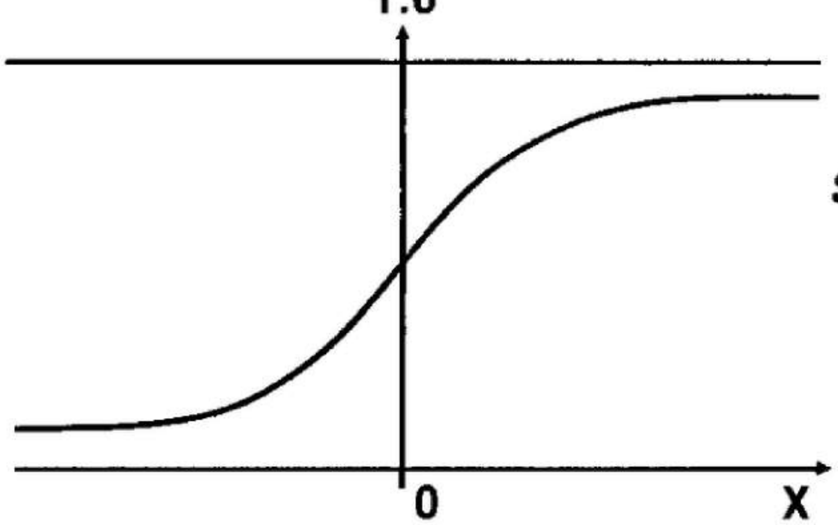
  - If Y_k(p) is the activation function, then $\delta_k(p) = \frac{\partial Y_k}{\partial X_k} \times e_k(p)$

  - After derivation, $\boldsymbol{\delta_k(p) = y_k(p) * [1 - y_k(p)] * e_k\,(p)}$ ……………….(1)

Activation Function

- The sigmoid activation function transforms the input, which can have any value between + - infinity, into a reasonable value in the range between 0 and 1.

- The input value is passed through the sigmoid activation function:



Sigmoid

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

Step 1: **Initialization**

Initialize the threshold values θ_j , θ_k, learning rate α, initial weights w_1, w_2, ..w_n with random number within the range [-2.4/F_i, 2.4/F_i], where F_i= Maximum no. of inputs connected to the single neuron.

Step 2: **Activation**

Activate the input layer and the desired output layer by applying inputs $x_1(p)$, $x_2(p)$, $x_3(p)$, ...$x_n(p)$ and desired output $Y_d(p)$ respectively. Then calculating the actual output of output layer.

a) Calculate the actual output of the neuron in the hidden layers:

$$y_j(p) = Sigmoid\left[\sum_{i=1}^{n} x_i(p)w_{ij}(p) - \theta_j\right]$$

b) Calculate the output of the neuron of the output layers:

$$y_k(p) = Sigmoid\left[\sum_{j=1}^{m} y_j(p)w_{jk}(p) - \theta_k\right]$$

c) Calculate the error of the neuron at the output layer: $e_k(p) = y_{dk}(p) - y_k(p)$

Step 3: **Weight Training**

Update the weights related to hidden and output layers.

a) Update the weights of the neuron in the output layers:

$$w_{jk}(p+1) = w_{jk}(p) + \Delta\, w_{jk}(p)$$

$$\Delta\, w_{jk}(p) = \alpha * y_j(p) * \delta_k\,(p)$$

$$\delta_k(p) = y_k(p) * [1 - y_k(p)] * e_k\,(p)$$

According to definition of error gradient

b) Update the weights of the neuron in the hidden layers:

$$w_{ij}(p+1) = w_{ij}(p) + \Delta\, w_{ij}(p)$$

$$\Delta\, w_{ij}(p) = \alpha * x_i(p) * \delta_j\,(p)$$

$$\delta_j(p) = y_j(p) * \left[1 - y_j(p)\right] * \sum_{k=1}^{l} \delta_k(p)\, w_{jk}\,(p)$$

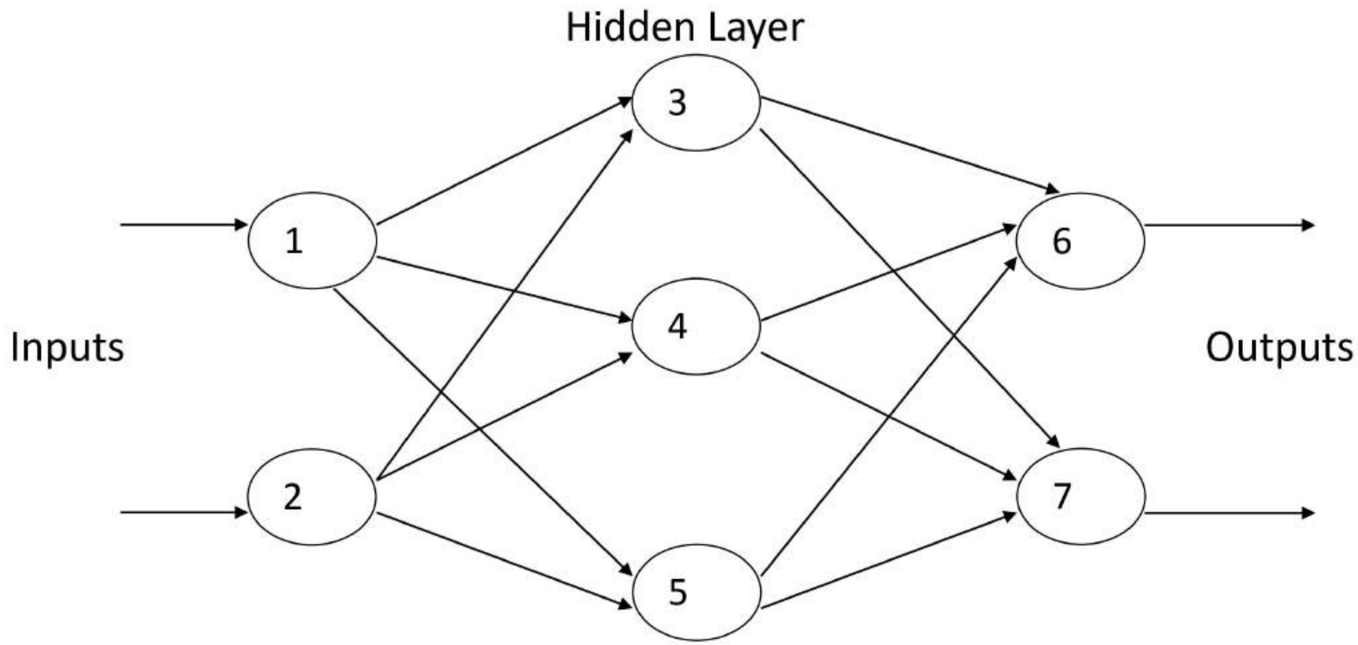
The definition of error gradient says that the error will be multiplied. But at the hidden layer, no error is calculated. So, the error gradient of output layer will be multiplied with the associated weights and summed-up.

Step 4: **Iteration**

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

Example

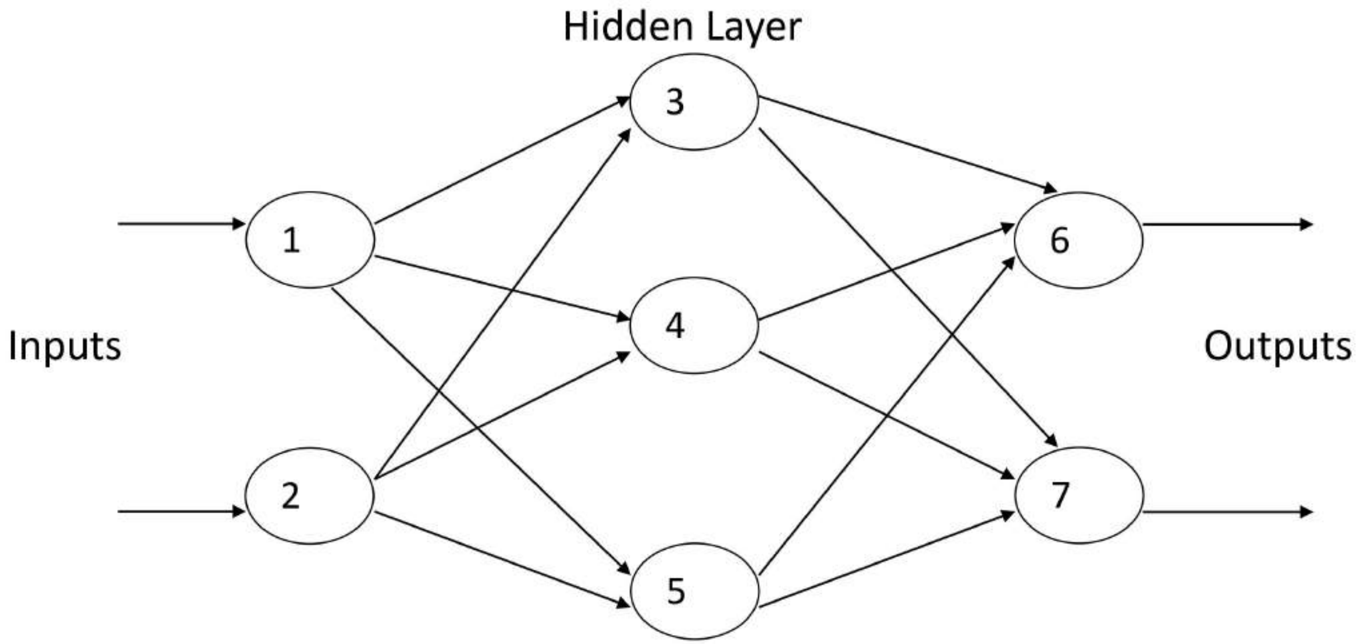
For the following NN, assume that W_{13}=0.3, W_{14}= - 0.4, W_{15}=0.6, W_{23}= - 0.9, W_{24}= - 0.2, W_{25}= - 0.3, W_{36}= - 0.4, W_{37}= - 0.1, W_{46}=0.3, W_{47}= - 0.3, W_{56}= 0.4, W_{57}=0.8, learning rate = 0.2.
If X=[1, 0] and Y=[0, 1], find the updated weights for output layer and hidden layer after one iteration. Assume that the threshold values are $\theta_j = \theta_k = 0$.



Example

Solution??

Ans: See the BPNN Math Example File



Application of BPNN

- The neural network is trained to enunciate each letter of a word and a sentence
- It is used in the field of speech recognition
- It is used in the field of character recognition
- It is used in the field of face recognition

Acknowledgement

- AIMA = Artificial Intelligence: A Modern Approach by Stuart Russell and Peter Norving (3rd edition)
- UC Berkeley
- U of toronto
- Other online resources

Thank You