Inspired by the functionality of neural network, researchers tried make something similar and that is artificial neural network (ANN). The neurons takes some information as input via Dendrites, processes it and then sends to next neuron until it reaches destination. Similar to this ANN is shown in figure 3.4,

Figure 1: Basic Artificial Neural Network

Ann consists of few parts,

- Nodes which are shown as input, hidden and output layer.
- Edges which contains weights.
- Activation function. It defines whether a neuron will fire or not with a certain limitation.
- Bias input node that helps to find the solution faster and in other field as well. Except output layer, all layers contains bias node.

As shown in the figure, the input node is connected with all node of hidden layer except bias. The hidden layers are connected with all output nodes. The weights are initialized with random numbers. ANN works in two phases: forward phase and backward phase. The Input node takes the inputs, processes it by multiplying the weight corresponded to the edge and sends it to next layer. Similarly hidden layer takes the input that was given, processes it using an activation function, then similarly the output is multiplied with the corresponding weight and sends it to output layer. The output layer takes the inputs, processes it and gives a corresponding output. This is called forward phase. In backward phase the weights are updated according to the output. In forward phase the following equation is used as activation function:

$$\mathrm{g(x)} = \frac{1}{1+e^{-\beta}x}$$

Here, β is a random number.
The calculations for in each neuron k in hidden layer is given below:

$$\mathrm{h_k} = \sum_{n=0}^{L} x_\mathrm{i} v_\mathrm{ik}$$

$$\mathrm{a_k} = \mathrm{g(h_k)} = \frac{1}{1+exp(-\beta h_\mathrm{k})}$$

Here, L is the number of input nodes. The calculations for in each neuron j in output layer is given below:

$$\mathrm{h_j} = \sum_{j} a_\mathrm{k} w_\mathrm{jk}$$

$$\mathrm{y_j} = \mathrm{g(h_j)} = \frac{1}{1+exp(-\beta h_\mathrm{j})}$$

Here, N is the number of nodes in hidden layers.
The following equations are used for updating the weights in output layer:

$$\delta_\mathrm{o}(j) = (y_\mathrm{j} - t_\mathrm{j}) y_\mathrm{j} (1 - y_\mathrm{j})$$

$$\mathrm{w_{jk}} \Leftarrow w_\mathrm{jk} - \eta \delta_\mathrm{o}(j) a_\mathrm{k}$$

Here η is the learning rate. The following equations are used for updating the weights in hidden layer:

$$\delta_\mathrm{h}(\mathrm{k}) = \mathrm{a_k(1\text{-}a_k)} \sum_{j=1}^{N} w_\mathrm{jk} \delta_\mathrm{o}(\mathrm{j})$$

$$\mathrm{v_{ik}} = \mathrm{v_{ik}} \text{ - } \eta \delta_\mathrm{h}(k) x_\mathrm{i}$$

ANN works well with large data set. So, if the data set is too small then it will not perform well. The output often depends on number of iteration and number

1 Artificial Neural Network