Learning and Adaptation

As stated earlier, ANN is completely inspired by the way biological nervous system, i.e. the human brain works. The most impressive characteristic of the human brain is to learn, hence the same feature is acquired by ANN.

What Is Learning in ANN?

Basically, learning means to do and adapt the change in itself as and when there is a change in environment. ANN is a complex system or more precisely we can say that it is a complex adaptive system, which can change its internal structure based on the information passing through it.

Why Is It important?

Being a complex adaptive system, learning in ANN implies that a processing unit is capable of changing its input/output behavior due to the change in environment. The importance of learning in ANN increases because of the fixed activation function as well as the input/output vector, when a particular network is constructed. Now to change the input/output behavior, we need to adjust the weights.

Classification

It may be defined as the process of learning to distinguish the data of samples into different classes by finding common features between the samples of the same classes. For example, to perform training of ANN, we have some training samples with unique features, and to perform its testing we have some testing samples with other unique features. Classification is an example of supervised learning.

Neural Network Learning Rules

We know that, during ANN learning, to change the input/output behavior, we need to adjust the weights. Hence, a method is required with the help of which the weights can be modified. These methods are called Learning rules, which are simply algorithms or equations. Following are some learning rules for the neural network −

Hebbian Learning Rule

This rule, one of the oldest and simplest, was introduced by Donald Hebb in his book *The Organization of Behavior* in 1949. It is a kind of feed-forward, unsupervised learning.

Basic Concept − This rule is based on a proposal given by Hebb, who wrote −

“When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A’s efficiency, as one of the cells firing B, is increased.”

From the above postulate, we can conclude that the connections between two neurons might be strengthened if the neurons fire at the same time and might weaken if they fire at different times.

Mathematical Formulation − According to Hebbian learning rule, following is the formula to increase the weight of connection at every time step.

$$\Delta w_{ji}(t)\ =\ \alpha x_i(t).\, y_j(t)$$

Here, $\Delta w_{ji}(t)$ = increment by which the weight of connection increases at time step **t**

α = the positive and constant learning rate

$x_i(t)$ = the input value from pre-synaptic neuron at time step **t**

$y_i(t)$ = the output of pre-synaptic neuron at same time step **t**

Perceptron Learning Rule

This rule is an error correcting the supervised learning algorithm of single layer feedforward networks with linear activation function, introduced by Rosenblatt.

Basic Concept − As being supervised in nature, to calculate the error, there would be a comparison between the desired/target output and the actual output. If there is any difference found, then a change must be made to the weights of connection.

Mathematical Formulation − To explain its mathematical formulation, suppose we have 'n' number of finite input vectors, x n , along with its desired/target output vector t n , where n = 1 to N.

Now the output 'y' can be calculated, as explained earlier on the basis of the net input, and activation function being applied over that net input can be expressed as follows −

$$y\ =\ f(y_{in})\ =\ \begin{cases}1, & y_{in}\ >\ \theta \\ 0, & y_{in}\ \leqslant\ \theta\end{cases}$$

Where **θ** is threshold.

The updating of weight can be done in the following two cases −

Case I − when **t ≠ y**, then

$$w(new)\ =\ w(old)\ +\ tx$$

Case II − when **t = y**, then

No change in weight

Delta Learning Rule $Widrow - HoffRule$

It is introduced by Bernard Widrow and Marcian Hoff, also called Least Mean Square LMS method, to minimize the error over all training patterns. It is kind of supervised learning algorithm with having continuous activation function.

Basic Concept − The base of this rule is gradient-descent approach, which continues forever. Delta rule updates the synaptic weights so as to minimize the net input to the output unit and the target value.

Mathematical Formulation − To update the synaptic weights, delta rule is given by

$$\Delta w_i \ = \ \alpha \, . x_i . e_j$$

Here Δw_i = weight change for ith pattern;

α = the positive and constant learning rate;

x_i = the input value from pre-synaptic neuron;

e_j = $(t \ - \ y_{in})$, the difference between the desired/target output and the actual output y_{in}

The above delta rule is for a single output unit only.

The updating of weight can be done in the following two cases −

Case-I − when **t ≠ y**, then

$$w(new) \ = \ w(old) \ + \ \Delta w$$

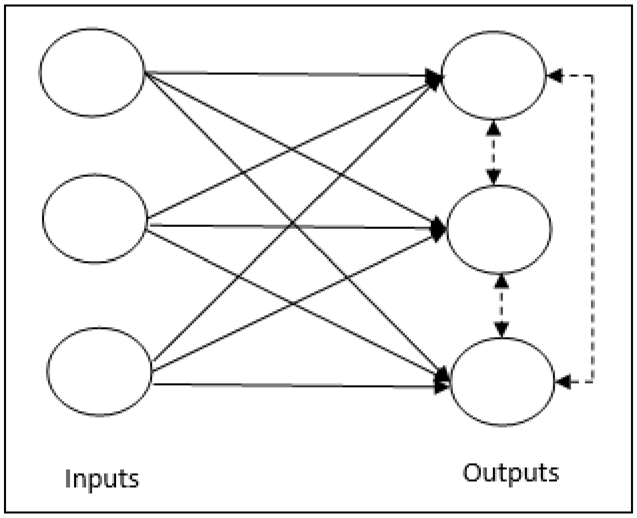
Case-II − when **t = y**, then

No change in weight

Competitive Learning Rule $Winner - takes - all$

It is concerned with unsupervised training in which the output nodes try to compete with each other to represent the input pattern. To understand this learning rule, we must understand the competitive network which is given as follows −

Basic Concept of Competitive Network − This network is just like a single layer feedforward network with feedback connection between outputs. The connections between outputs are inhibitory type, shown by dotted lines, which means the competitors never support themselves.



Basic Concept of Competitive Learning Rule − As said earlier, there will be a competition among the output nodes. Hence, the main concept is that during training, the output unit with the highest activation to a given input pattern, will be declared the winner. This rule is also called Winner-takes-all because only the winning neuron is updated and the rest of the neurons are left unchanged.

Mathematical formulation − Following are the three important factors for mathematical formulation of this learning rule −

- **Condition to be a winner** − Suppose if a neuron y_k wants to be the winner then there would be the following condition −

$$y_k = \begin{cases} 1 & if\, v_k > v_j\, for\, all\, j,\, j \neq k \\ 0 & otherwise \end{cases}$$

It means that if any neuron, say y_k , wants to win, then its induced local field $theoutputofsummationunit$, say v_k , must be the largest among all the other neurons in the network.

- **Condition of sum total of weight** − Another constraint over the competitive learning rule is, the sum total of weights to a particular output neuron is going to be 1. For example, if we consider neuron **k** then −

$$\sum_j w_{kj} = 1 \qquad for\ all\ k$$

- **Change of weight for winner** − If a neuron does not respond to the input pattern, then no learning takes place in that neuron. However, if a particular neuron wins, then the corresponding weights are adjusted as follows

$$\Delta w_{kj} = \begin{cases} -\alpha(x_j - w_{kj}), & if\, neuron\, k\, wins \\ 0, & if\, neuron\, k\, losses \end{cases}$$

Here α is the learning rate.

This clearly shows that we are favoring the winning neuron by adjusting its weight and if there is a neuron loss, then we need not bother to re-adjust its weight.

Outstar Learning Rule

This rule, introduced by Grossberg, is concerned with supervised learning because the desired outputs are known. It is also called Grossberg learning.

Basic Concept − This rule is applied over the neurons arranged in a layer. It is specially designed to produce a desired output **d** of the layer of **p** neurons.

Mathematical Formulation − The weight adjustments in this rule are computed as follows

$$\Delta w_j = \alpha\,(d - w_j)$$

Here **d** is the desired neuron output and α is the learning rate.