Assignment -9

Title: Write a python program to design a Hopfield Network which stores 4 vectors

Aim: Write a python program to design a Hopfield Network which stores 4 vectors

Theory:

Hopfield neural network was invented by Dr. John J. Hopfield in 1982. It consists of a single layer which contains one or more fully connected recurrent neurons. The Hopfield network is commonly used for auto- association and optimization tasks. Hopfield network is a special kind of neural network whose response is different from other neural networks. It is calculated by converging iterative process. It has just one layer of neurons relating to the size of the input and output, which must be the same. When such a network recognizes, for example, digits, we present a list of correctly rendered digits to the network. Subsequently, the network can transform a noise input to the relating perfect output.

Discrete Hopfield network:

A Hopfield network which operates in a discrete line fashion or in other words, it can be said the input and output patterns are discrete vector, which can be either binary 0,1 or bipolar+1,-1 in nature. The network has symmetrical weights with no self-connections i.e., wij = wji and wii = 0.

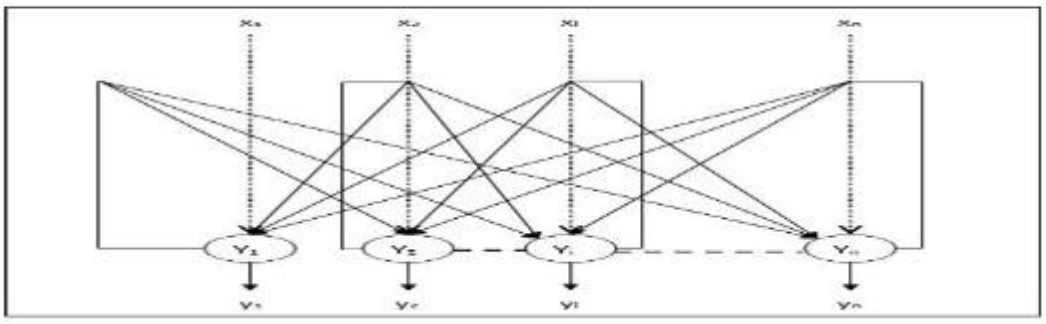
Architecture:

Following are some important points to keep in mind about discrete Hopfield network:

- This model consists of neurons with one inverting and one non-inverting output.
- The output of each neuron should be the input of other neurons but not the input of self.
- Weight/connection strength is represented by wij.
- Connections can be excitatory as well as inhibitory. It would be excitatory, if the output of the neuron is same as the input, otherwise inhibitory.
- Weights should be symmetrical, i.e. wij = wji

The output from Yl going to Y2, Yi and Yn have the weights w12, wli and wln respectively. Similarly, other arcs have the weights on them.

Training Algorithm:



During training of discrete Hopfield network, weights will be updated. As we know that we can have the binary input vectors as well as bipolar input vectors. Hence, in both the cases, weight updates can be done with the following relation.

Case 1 – Binary input patterns

For a set of binary patterns s p, p = 1 to P
Here, s p= s1p , s2p ,..., sip,..., snp

Weight Matrix is given by,

$$w_{ij} = \sum_{p=1}^{P}[2s_i(p) - 1][2s_j(p) - 1] \quad for\ i \neq j$$

Case 2 – Bipolar input patterns

For a set of binary patterns s p, p = 1 to P
Here, s p = s1 p, s2 p..., si p..., sn p

Weight Matrix is given by

$$w_{ij} = \sum_{p=1}^{P}[s_i(p)][s_j(p)] \quad for\ i \neq j$$

Testing Algorithm:

Step 1 – Initialize the weights, which are obtained from training algorithm by using Hebbian principle.

Step 2 – Perform steps 3-9, if the activations of the network is not consolidated.

Step 3 – For each input vector X, perform steps 4-8.

Step 4 – Make initial activation of the network equal to the external input vector X as follows:

$$y_i = x_i \quad for\ i = 1\ to\ n$$

Step 5 – For each unit Yi, perform steps 6-9.

Step 6 – Calculate the net input of the network as follows :

$$y_{ini} = x_i + \sum_j y_j w_{ji}$$

Step 7 – Apply the activation as follows over the net input to calculate the output :

$$y_i = \begin{cases} 1 & if\ y_{ini} > \theta_i \\ y_i & if\ y_{ini} = \theta_i \\ 0 & if\ y_{ini} < \theta_i \end{cases}$$

Here θi is the threshold.

Step 8 − Broadcast this output yi to all other units.

Step 9 − Test the network for conjunction.

Continuous Hopfield Network:

Unlike the discrete hopfield networks, here the time parameter is treated as a continuous variable. So, instead of getting binary/bipolar outputs, we can obtain values that lie between 0 and 1. It can be used to solve constrained optimization and associative memory problems. The output is defined as:

$$v_i = g(u_i)$$

where,

vi = output from the continuous hopfield network
ui = internal activity of a node in continuous hopfield network.

Conclusion:

We have successfully implemented back propagation feed-forward neural network.

Questions:

1. Explain Padding in CNN.
2. Explain Strided convolution.
3. Explain SoffMax regression.
4. Explain Deep Learning frameworks.
5. Explain Transfer Learning.