|  |  |  |  |
| --- | --- | --- | --- |
| **Course Name:** | **ANNFS** | **Semester:** | **VI** |
| **Date of Performance:** | **08/02/2022** | **Batch No:** | **B2** |
| **Faculty Name:** | **Dr. Jagannath Nirmal** | **Roll No:** | **1912052** |
| **Faculty Sign & Date:** |  | **Grade/Marks:** |  |

**Experiment No: 2**

**Title: HEBBS rule**

|  |
| --- |
| **Aim and Objective of the Experiment:** |
| Write a program to understand the hebbian learning rule that can be applied for supervised learning of neural network |

|  |
| --- |
| **COs to be achieved:** |
| **CO1:** Understand the role of neural networks in engineering, artificial intelligence, and cognitive modelling.  **CO3:** Develop concepts and techniques of neural networks through the study of the most important supervised and unsupervised neural network models. |

**Apparatus / Software tools used: MATLAB/Python/C/C++…**

|  |
| --- |
| **Theory:** |
| 1. **HEBBIAN learning rule:** 2. The learning signal is equal to neuron s output r=f(w,ntx) the increment of Wi of weight vector becomes 4. This simple weight is adapted using following increment   the learning rule requires the weight initialization of small random values around prior to the learning Hebbian rule represent purely feed forward learning algorithm of HEBB network Initialize all weight and bias the zero for i=0, where n=no of input neuron For each input training vector and output pattern following |

|  |
| --- |
| **Block Diagram/Program flowchart:** |
| Hebbian Learning - an overview | ScienceDirect Topics |

|  |
| --- |
| **Algorithm:** |
| 1. **Initialization**   Set initial synaptic weights wij and thresholds to small random values, say in an interval [0, 1]. Also assign small positive values to the learning rate parameter.   1. **Activation**   Compute the neuron output at iteration p   1. **Learning**   Update the weightsin the network:   1. Wi(new)=b(old)+y Adjust Bias: B=b(old)+y 2. **Iteration**   Increase iteration p by one, go back to Step 2 and 3 and continue until the synaptic weights reach their steady-state values |
| clc;  clear all;  close all;  %Enter weights, bias and threshold value  disp('Enter initial weights');  W1o = input('W1 = ');  W2o = input('W2 = ');  disp('Enter initial bias');  bo = input('b = ');  disp('Enter Threshold value');  theta = input('theta = ');  %Defining AND gate parameters  X1 = [1 1 -1 -1];  X2 = [1 -1 1 -1];  b = [1 1 1 1];  Z = [0 0 0 0];  Y = [1 -1 -1 -1];  %Training weights and bias  for i=1:4  W1n = W1o + X1(i)\*Y(i);  W1o = W1n;  W2n = W2o + X2(i)\*Y(i);  W2o = W2n;  bn = bo + Y(i);  bo = bn;  end  %Finding output and comparing with target output  con = 1;  while con  Yin = X1\*W1n + X2\*W2n + bn;  for i=1:4  if Yin(i) >= theta  Z(i) = 1;  else  Z(i) = -1;  end  end  disp('Output of Net');  disp(Z);  if Z==Y  con = 0;  else  disp('Network is not learning');  return  end  end  %Displaying the correct weights and threshold value  disp('HEBBS Model for AND function');  disp('Weights of Neuron');  disp(W1n);  disp(W2n);  disp('Weights of Bias');  disp(bn);  disp('Threshold value');  disp(theta);  %plotting the trained network  hold on;  scatter(X1(Z>=1),X2(Z>=1),'r','filled','linewidth',10);  scatter(X1(Z<1),X2(Z<1),'b','filled','linewidth',10);  X1 = [-0.75 0 0.75 1.5];  X2 = [1.5 0.75 0 -0.75];  plot(X1,X2,'g','Linewidth',2);  hold off;  xlim([-1.2 1.2]);  ylim([-1.2 1.2]);  title('HEBBS Model for AND function');  xlabel('X1');  ylabel('X2'); |

|  |
| --- |
| **Observation:** |
| Attach program and its results after execution     * **Command Window** |

|  |
| --- |
| **Post Lab Subjective/Objective type Questions:** |
| 1. Why Hebb rule is more suitable for bipolar data than binary data?   According to Hebb’s rule, the weights are found to increase proportionately to the product of input and output. It means that in a Hebb network if two neurons are interconnected then the weights associated with these neurons can be increased by changes in the synaptic gap.  This network is suitable for bipolar data. The Hebbian learning rule is generally applied to logic gates.  The weights are updated as:  W (new) = w (old) + x\*y   1. Write about Hebbian learning and its weight upgradation steps   It is one of the first and also easiest learning rules in the neural network. It is used for pattern classification. It is a single layer neural network, i.e. it has one input layer and one output layer. The input layer can have many units, say n. The output layer only has one unit. Hebbian rule works by updating the weights between neurons in the neural network for each training sample.  Hebbian Learning Rule Algorithm :   1. Set all weights to zero, wi = 0 for i=1 to n, and bias to zero. 2. For each input vector, S(input vector) : t(target output pair), repeat steps 3-5. 3. For each input vector, S(input vector) : t(target output pair), repeat steps 3-5. 4. For each input vector, S(input vector) : t(target output pair), repeat steps 3-5. 5. For each input vector, S(input vector) : t(target output pair), repeat steps 3-5. |

|  |
| --- |
| **Conclusion:** |
| In this experiment, we have successfully learned and implemented the program for Hebbian Learning rule using MATLAB and applied it for the AND gate, and verified the obtained results. |

|  |
| --- |
| **Signature of faculty in-charge with Date:** |