

Experiment 4

NAME	Shreya Shetty
UID	2019140059
CLASS	TE IT
BATCH	B
ACADEMIC YEAR	2021-22
SUBJECT	SC (Soft Computing)
COURSE CODE	IT312
EXPERIMENT NO.	3

Aim:

Write a program to design a perceptron to recognize these patterns for the problem statement in experiment No.3.(Use any Open source tools)

Problem Statement:

Problem To Distinguish Between Apples And Oranges:

A produce dealer has a warehouse that store a variety of fruits & vegetables. When fruit is brought to the warehouse , a various types of fruits may be mixed together. The dealer wants a machine that will sort the fruit according to type . There is a conveyer belt on which the fruit is loaded .This conveyer passes through a set of sensors, which measure three properties of fruits : shape , texture and weight.

Bias= < Any Value>

Type of sensor	Output of sensor	Condition
Shape sensor	1	if fruit is approx. round
	0	if fruit is elliptical.
Texture Sensor	1	If surface is smooth
	0	If surface is rough
Fruit sensor	1	Apple
	0	Orange

Design a perceptron to recognize these patterns using Joone Editor.

Experiment 4

Theory:

Joone Editor

Joone is a FREE Neural Network framework to create, train and test artificial neural networks. The aim is to create a powerful environment both for enthusiastic and professional users, based on the newest Java technologies. Joone's neural networks can be built on a local machine, be trained on a distributed environment and run on whatever device.

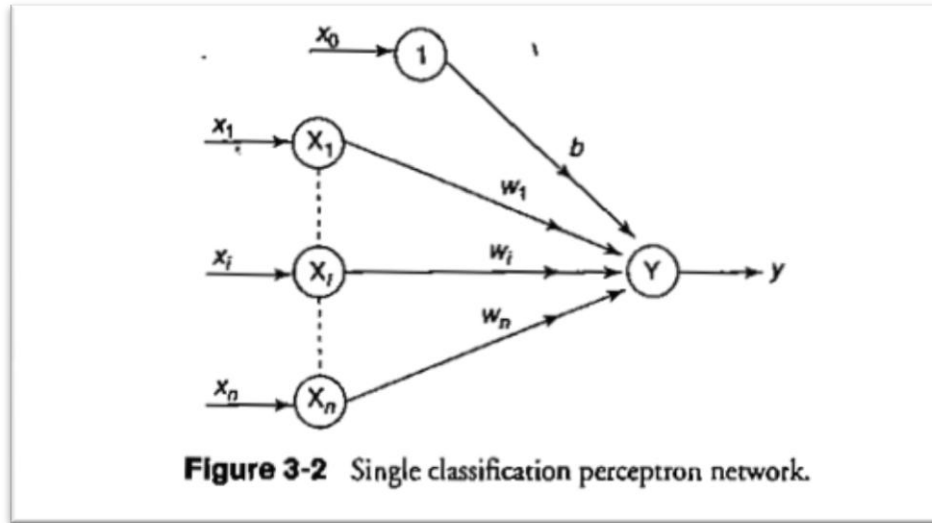
Perceptron Model

- The Perceptron networks come under single-layer feed-forward networks and are also called simple perceptron's.
- Various types of perceptron's were designed by Rosenblatt (1962) and Minsky-Papert (1969, 1988). However, a simple perceptron network was discovered by Block in 1962.
- The perceptron network consists of three units, namely, the sensory unit (input unit), associator unit (hidden unit), and response unit (output unit).
- The sensory units are connected to associator units with fixed weights having values 1, 0 or -1, which are assigned randomly.
- The binary activation function is used in the sensory unit and associator unit.
- The response unit has an activation of 1, 0 or -1. The binary step with a fixed threshold θ is used as activation for the associator. The output signals that are sent from the associator unit to the response unit are only binary.
- The output of the perceptron network is given by $y_{out} = f(y_{in})$ where y_{out} is activation function and is defined as

$$y_{out} = \begin{cases} 1, & \text{if } y_{in} > \theta \\ 0, & \text{if } -\theta \leq y_{in} \leq \theta \\ -1, & \text{if } y_{in} < -\theta \end{cases}$$

- The perceptron learning rule is used in the weight updation between the associant unit and the response unit. For each training input, the net will calculate the response and it will determine whether or not an error has occurred.
- The error calculation is based on the comparison of the values with those of the calculated outputs.
- The weights on the connections from the units that send the non zero signal will get adjusted suitably. The weights will be adjusted on the basis of the learning rule if an error has occurred for a particular training pattern
- If no error occurs, there is no weight update and hence the training process may be stopped.
- The below figure shows a simple classification perceptron network :

Experiment 4



The notable points regarding Procedure Perceptron-Learning are given below:

1. The input vectors are allowed to be either binary or bipolar. However, the outputs must be in bipolar form.
2. The bias w_0 is adjustable but the threshold q used in the activation function is fixed.
3. Learning takes place only when the computed output does not match the target output. Moreover, as $\Delta w_i = n \times t \times x_i$ the weight adjustment is 0 if $x_i = 0$. Hence, no learning takes place if either the input is 0, or the computed output matches the target output. Consequently, as training proceeds, more and more training patterns yield correct results and less learning is required by the net.
4. The threshold q of the activation function may be interpreted as a separating band of width $2q$ between the region of positive response and negative response. The band itself is 'undecided' in the sense that if the net input falls within the range $[-q, q]$, the activation is neither positive, nor negative. Moreover, changing the value of q would change the width of the undecided region, along with the position of the separating lines. Therefore, for Perceptrons, the bias and the threshold are no longer interchangeable.
5. The band separating the regions of positive response from that of the negative response is defined by the pair of lines

$$w_0 x_0 + w_1 x_1 + w_2 x_2 = \Theta$$

$$w_0 x_0 + w_1 x_1 + w_2 x_2 = -\Theta$$

Procedure Perceptron-Learning:

Step 1. Initialize all weights, w_0, \dots, w_m .

Step 2. Set learning rate h such that $0 < h \leq 1$, and threshold q .

Step 3. For each training pair $s : t$ do Steps 4–8.

Experiment 4

Step 4. Activate the input units, $x_i = s_i$, for $i = 0, \dots, m$.

Step 5. Compute the net input to the output unit

$$y_in = \sum_{i=0}^m w_i x_i$$

Step 6. Compute the activation of the output unit using the function

$$y_out = \begin{cases} 1, & \text{if } y_in > \theta \\ 0, & \text{if } -\theta \leq y_in \leq \theta \\ -1, & \text{if } y_in < -\theta \end{cases}$$

Step 7. If there is an error, i.e., $y_out \neq t$, then adjust the weights as follows


$$w_i \text{ (new)} = w_i \text{ (old)} + \eta \times t \times x_i$$

Step 8. If there were no error, i.e., $y_out = t$, for the entire set of training pairs, then stop. Otherwise go to Step 3.

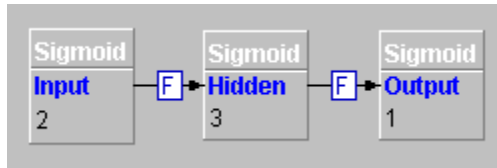
Procedure:




1. Install Java virtual Machine
2. Install Joone Editor from <https://www.jooneworld.com/download.html>
3. Implement Basic Neural Network Learning rules from -
<https://www.jooneworld.com/docs/sampleEditor.html>

Run the editor, and execute these following steps:

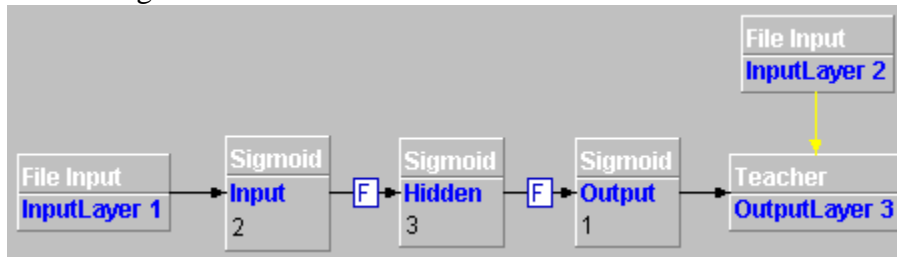
1. Add a new sigmoid layer  and set its layerName to 'Input' and the rows parameter to 2
 2. Add a new sigmoid layer, and set its layerName to 'Hidden' and the rows parameter to 3
 3. Add a new sigmoid layer, set its layerName to 'Output', leaving the rows parameter to 1
 4. Connect the input layer to the hidden layer dragging a line from the little circle on the right hand side of the input layer, releasing the mouse button when the arrow is on the hidden layer
 5. Repeat the above step connecting the hidden layer to the output layer
- At this stage the screen should look similar to this:

Experiment 4



6. Insert a File Input layer  to the left of the input layer, then click on it to display the properties window:
 - Set the AdvancedColumnSelector parameter to "1,2"
 - Enter c:\joone\xor.txt in the fileName parameter
 - Leave the firstRow as 1 and the lastRow as 0 so that the input layer will read all the rows in the file
7. Connect the File Input to the input layer
8. Insert a Teacher layer  to the right of the output layer
9. Connect the output layer to the Teacher layer
Now we must provide to the teacher the desired data (the last column of the file xor.txt) to train the net:
10. Insert a File Input layer  on top of the Teacher layer, then click on it to display the properties window:
 - Set the AdvancedColumnSelector parameter to 3
 - Enter c:\joone\xor.txt in the fileName parameter
 - Leave the firstRow as 1 and the lastRow as 0 so that the input layer will read all the rows in the file
11. Connect the Teacher layer to that last File Input layer dragging a line from the little red box on the top side of the Teacher layer, releasing the mouse button when the yellow arrow is on the last inserted File Input layer.

At this stage the screen should look similar to this:



12. Click on the 'Net->Control Panel' menu item to display the control panel. Insert the following:
 - Set the epochs parameter to 10,000. This will process the file 10,000 times
 - Set the training patterns parameter to 4. This sets the number of example rows to read
 - Set the learningRate parameter to 0.8 and the momentum parameter to 0.3
 - Set the learning parameter to TRUE, as the net must be trained
13. Click the **START** button, and you'll see the training process starting

The Control Panel shows the cycles remaining and the current error of the net. At the end of the last cycle, the error would be very small (less than 0.1), otherwise click on the 'Net->Randomize' menu item (to add a noise to the weights of the net) and click again on the START button.

Experiment 4

If you want, you can save the net with the 'File->Save as' menu item, so you can use again the net later loading it from the file system.

Testing the XOR example

To test the trained net:

1. Add an Output File layer on the right of the output layer, click on it and insert into the properties window:
 - "#topofpage" on the fileName parameter.
2. Connect the output layer to the File Output layer
3. Select the line that connects the output layer to the Teacher layer and click on the 'Edit->Delete' to disconnect the Teacher from the neural net
4. On the Control Panel change the following:
 - Set the epochs parameter to 1. This will process the input file once
 - Set the learning parameter to FALSE, as the net is not being trained
5. Click on the **START** button
6. Open the xorout.txt file with an editor, and you'll see a result like this (the values can change from a run to another, depending on the initial random values of the weights):

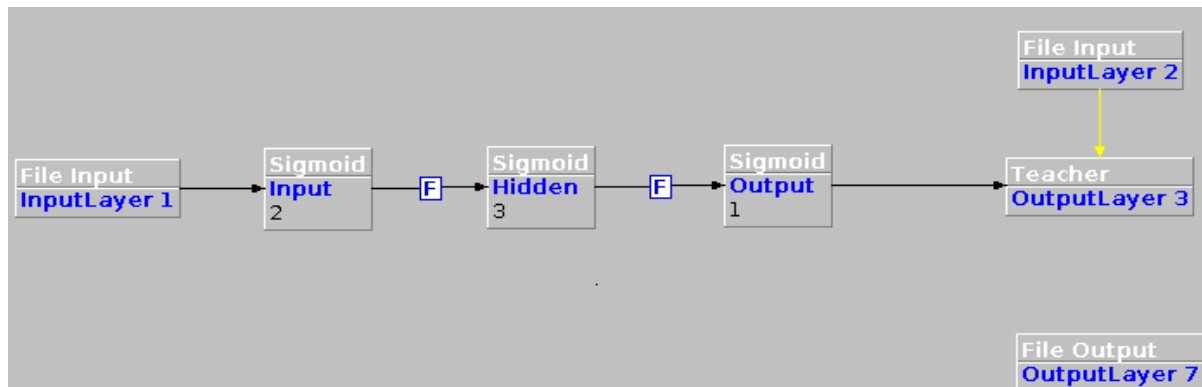
```
0.02592995527027603
0.9664584492704527
0.9648193164739925
0.03994103766843536
```

This result shows that the neural net has learned the XOR problem, providing the correct results:

- a value near zero when the input columns are equal to [0, 0] and [1, 1]
- a value near one when the input columns are equal to [0, 1] and [1, 0]

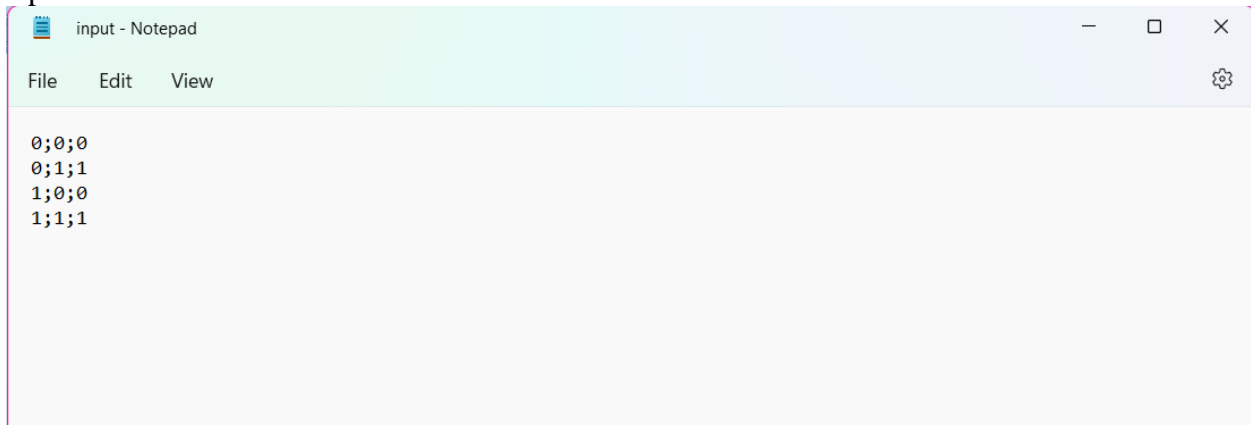
Output in Joone Editor:

Training the perceptron:



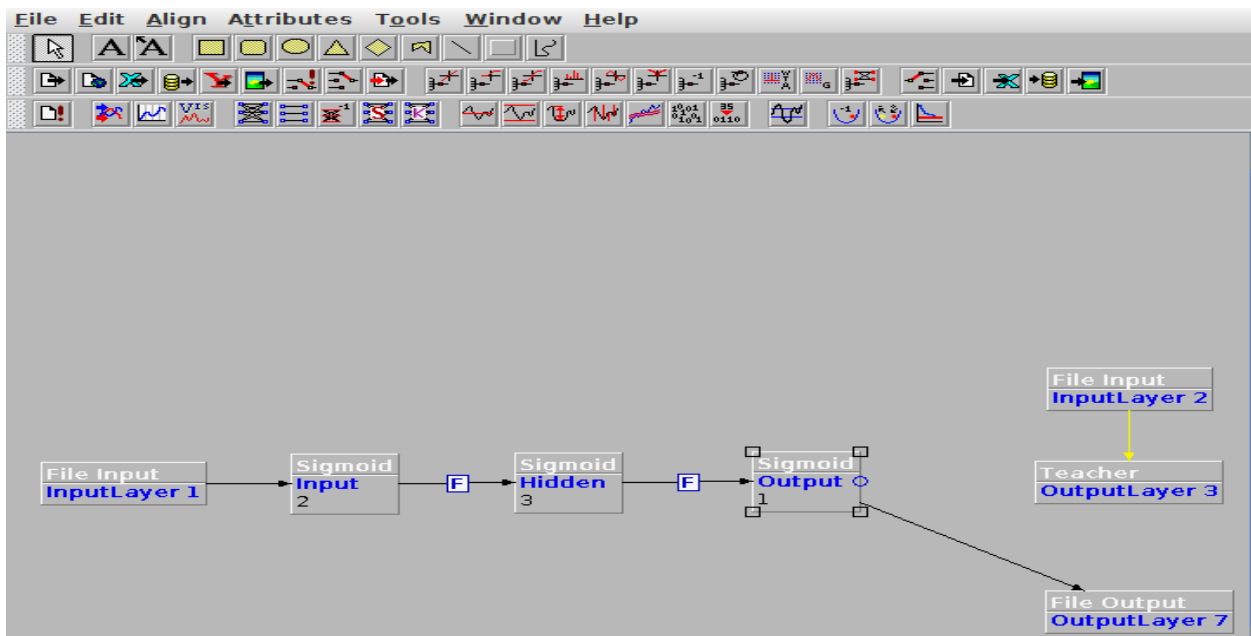
Experiment 4

Input File:

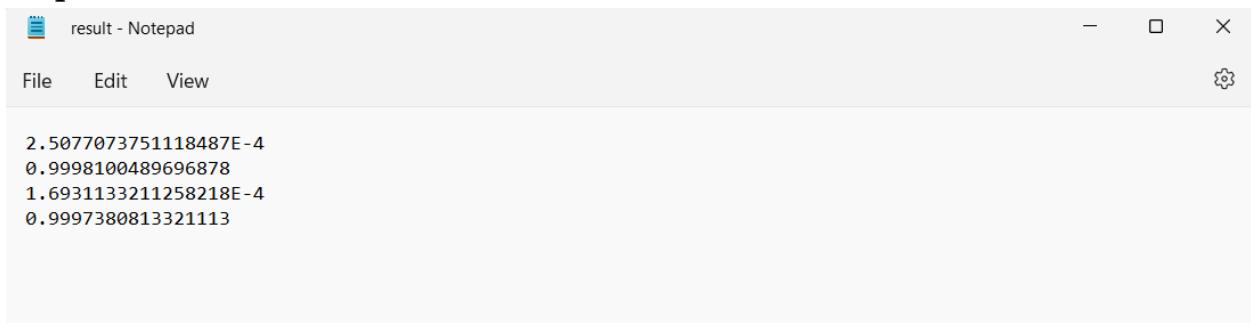


```
0;0;0
0;1;1
1;0;0
1;1;1
```

Output is stored in text file:



Output:



```
2.5077073751118487E-4
0.9998100489696878
1.6931133211258218E-4
0.9997380813321113
```

Experiment 4

Conclusion:

In this experiment, I learned to work with joone editor and I have successfully implemented a simple neural network using the Joone editor and also trained and tested the neural network with data.