Department of Electronics Engineering

Experiment No. : 04
ADALINE Network

Aim: To develop a SCILAB program for OR function with bipolar inputs and targets using ADALINE network.

Apparatus: SCILAB

Theory: Widrow and Hoff [1960] developed the learning rule that is very closely related to the perceptron learning rule. The rule, called the Delta rule, adjusts the weights to reduce the difference between the net input to the output unit, and the desired output which results in a least mean squared error (LMS error). Adaline (Adaptive Linear Neuron) and Madaline (Multi-layered Adaline) networks use the LMS learning rule.

Adaline uses bipolar activations for its input signals and target output. The weights and bias of the adaline network are adjustable. The adaline network has only one output unit which receives input from several units and bias (whose activation is always +1). The link between the input and the output neuron possess weighted interconnections, which can be changed as the training progresses.

The initial weights are set to small random values and not to zero as in case of perceptron as it influences the error factor. The net input is calculated based on the training input patterns and the weights. By applying the delta learning rule, the weight updation is carried out. The training process continues until the error, which is the difference between the target and the net input becomes minimum.

Training Algorithm :

Step 1 : Initialize all the weights and the bias to small random values other than zero.

Step 2 : While stopping condition is false, perform steps 3 – 7.

Step 3 : Set activations for the input units with input vector.

$$x_i = s_i \quad (i = 1 \text{ to } n)$$

Step 4 : Set activation for the output unit with output neuron.

$$y = t$$

Step 5 : Compute net input to output unit

$$y_{in} = b + \sum_{i=1}^{n} x_i w_i$$

Step 6 : Update the weights and the bias,

$$w_i(new) = w_i(old) + \alpha(t - y_{in})x_i \; \dots \; for \; i = 1 \; to \; n$$

$$b(new) = b(old) + (t - y_{in})$$

Step 7 : Test for stopping condition.
The stopping condition may be when the weight change reaches small level or number of iterations attained.

Procedure:

1. To implement the function using SCILAB / Python code and check the resultant weights and bias values for each epoch.
2. Calculate the mean error per epoch and stop the program when the error remains constant.
3. Display the final weights and bias values.

Department of Electronics Engineering

Problem : Implement OR function using ADALINE networks for bipolar inputs and targets. (learning rate, α = 0.1).

The algorithm should stop if the total error between 2 epochs is less than 0.1.

Display the final weights and total number of epochs.

Results :

1. *Solve and upload the given problem for 2 epochs.*

Learning rate α = 0.1

Inputs			Target	Net input		Weight changes			Weights			Error	Total Error
x1	x2	b	t	y_{in}	$t-y_{in}$	Δw1	Δw2	Δb	w1	w2	b	$(t-y_{in})$^2	E
Initial condition									**0.1**	**0.1**	**0.1**		
1	1	1	1	0.3	0.7	0.07	0.07	0.07	0.17	0.17	0.17	0.49	3.0210875
1	-1	1	1	0.17	0.83	0.083	-0.083	0.083	0.253	0.087	0.253	0.6889	
-1	1	1	1	0.087	0.913	-0.091	0.0913	0.0913	0.1617	0.178	0.344	0.83357	
-1	-1	1	-1	0.0043	-1.0043	0.1004	0.1004	-0.1	0.2617	0.2787	0.2439	1.00862	
Epoch 2													
1	1	1	1	0.7847	0.2153	0.0215	0.0215	0.0215	0.2837	0.302	0.2654	0.04635	1.9328506
1	-1	1	1	0.2488	0.7512	0.07512	-0.0751	0.07512	0.3588	0.2688	0.3405	0.5643	
-1	1	1	1	0.2085	0.7915	-0.0792	0.07915	0.07915	0.2796	0.3059	0.4196	0.62647	
-1	-1	1	-1	-0.1659	-0.8341	0.08341	0.08341	-0.0834	0.363	0.3893	0.3362	0.69572	

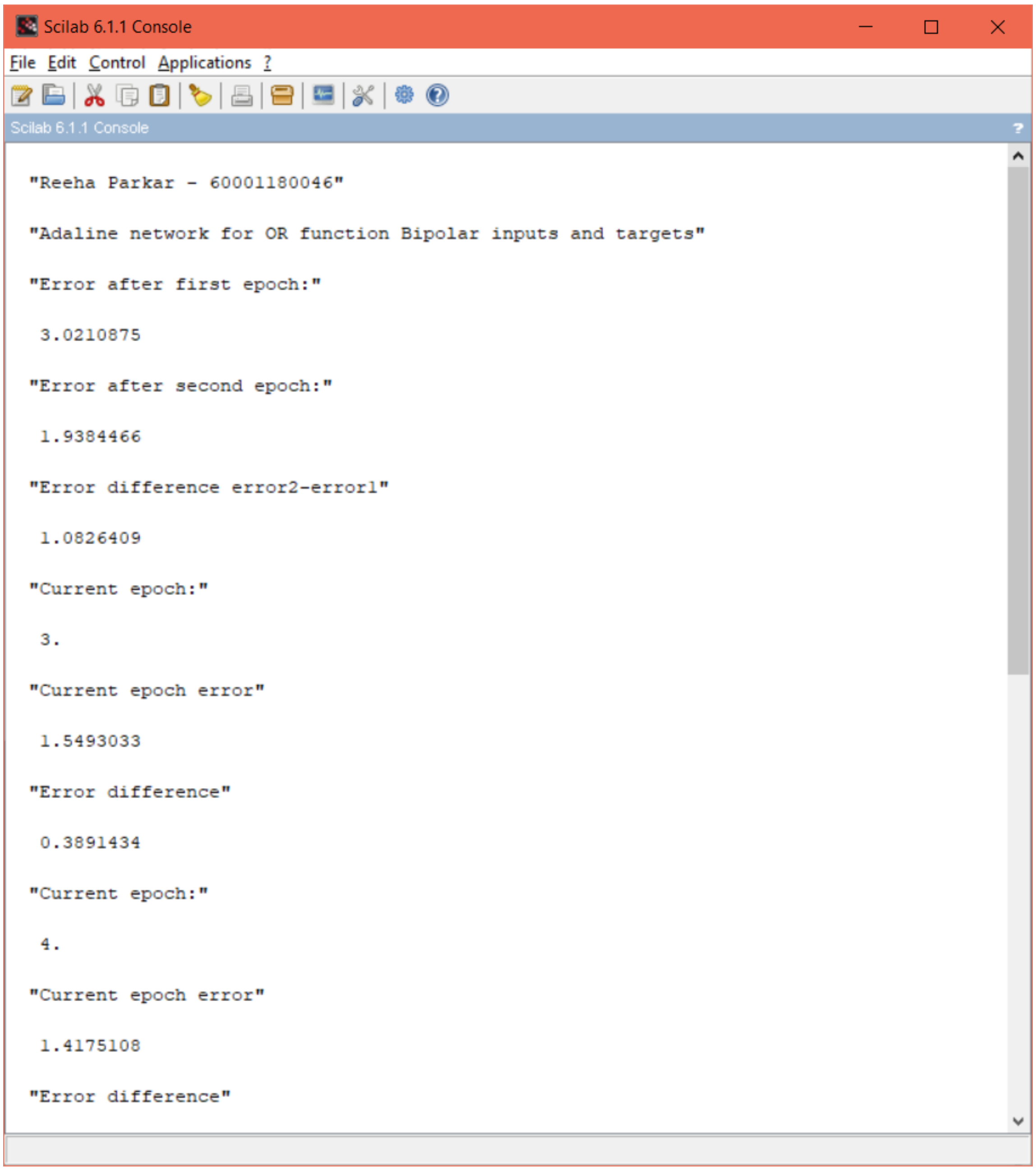
2. *(Code and outputs)*

Code:

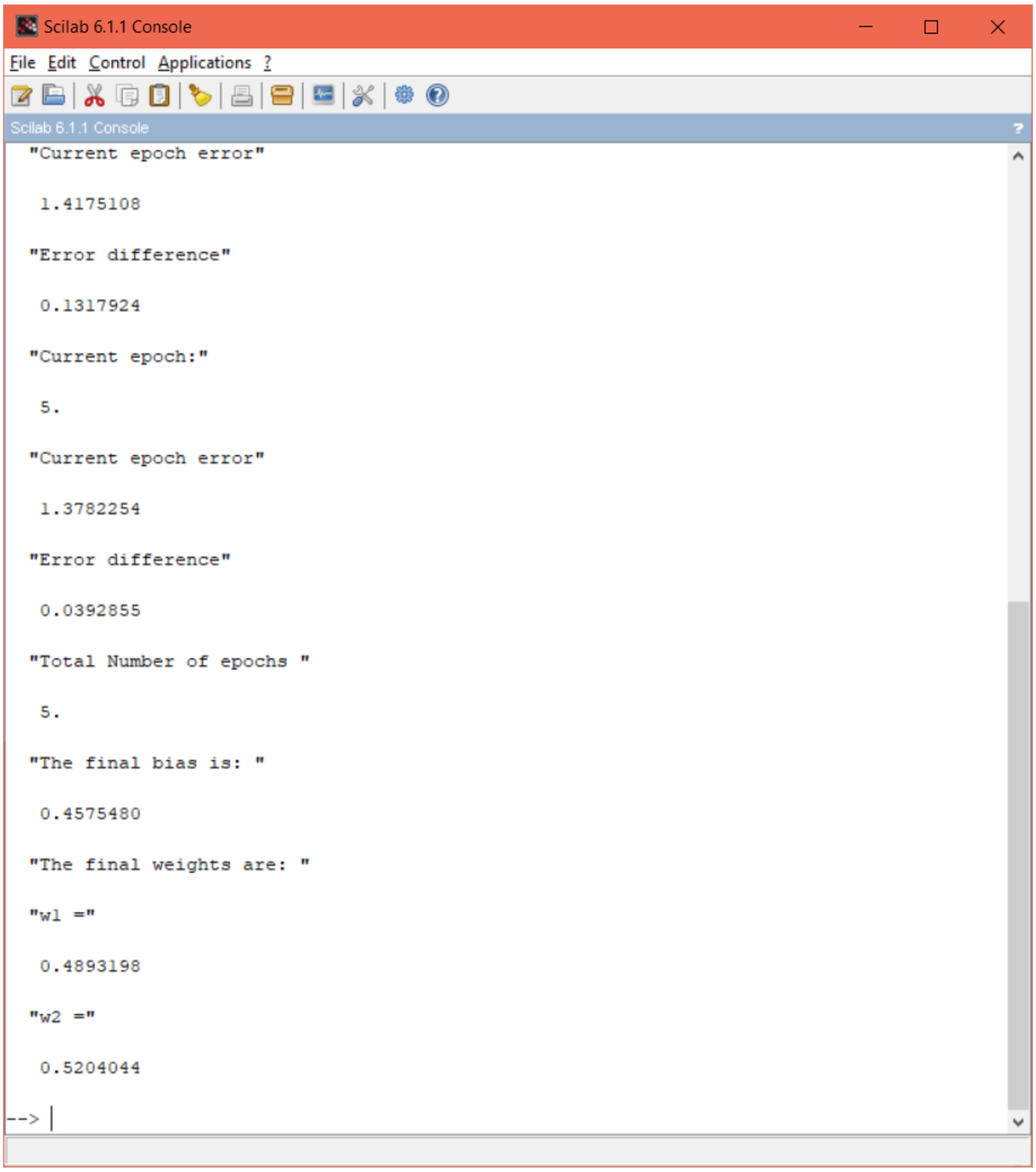
```
//truth table
//x1 x2  b  t
//1   1   1  1
//1  -1   1  1
//-1  1   1  1
//-1 -1   1 -1

clc ;
clear ;
disp("Reeha Parkar - 60001180046");
disp ('Adaline network for OR function Bipolar inputs and targets') ;

// inputs
x1 =[1 1 -1 -1];
x2 =[1 -1 1 -1];

// bias
x3 =[1 1 1 1];

// target
t =[1 1 1 -1];

// weights and bias
w1 =0.1;
w2 =0.1;
b =0.1;

//learning rate
alpha =0.1;
```

```
// error:
e =0;
e1=0;

delw1 =0; delw2 =0; delb =0;
epoch =1;

//1st epoch
for i =1:4
    nety(i) = w1*x1(i) + w2*x2(i) + b ;

    nt =[ nety(i) t(i)];
    delw1 = alpha*(t(i)-nety(i)) * x1(i) ;
    delw2 = alpha*(t(i)-nety(i)) * x2(i) ;
    delb = alpha*(t(i)-nety(i)) * x3(i) ;
    // weight changes
    wc =[ delw1 delw2 delb ]
    // update weights
    w1 = w1 + delw1 ;
    w2 = w2 + delw2 ;
    b = b + delb ;
    //new weights
    w =[ w1 w2 b ];
    // input current
    x =[ x1(i) x2(i) x3(i) ];

end

for i =1:4
e = e + (t(i)-nety(i))^2;
end;

//Error prints:
disp("Error after first epoch:");
disp(e);

//2nd epoch
for i =1:4
    nety(i) = w1*x1(i) + w2*x2(i) + b ;

    nt =[ nety(i) t(i) ];
    delw1 = alpha*(t(i)-nety(i)) * x1(i) ;
    delw2 = alpha*(t(i)-nety(i)) * x2(i) ;
    delb = alpha*(t(i)-nety(i)) * x3(i) ;
    // weight changes
    wc =[ delw1 delw2 delb ]
    // updating of weights
    w1 = w1 + delw1 ;
    w2 = w2 + delw2 ;
    b = b + delb ;
    //new weights
    w =[ w1 w2 b ];
    // input pattern
    x =[ x1(i) x2(i) x3(i) ];

end

for i =1:4
    e1 = e1 + (t(i)-nety(i))^2;
end;

//Error prints:
disp("Error after second epoch:");
disp(e1);
disp("Error difference error2-error1");
disp(e-e1);

epoch = epoch + 1;

while(e - e1) > 0.1
    epoch = epoch +1;
    e = e1;
    e1 = 0;
    for i =1:4
        nety(i) = w1*x1(i) + w2*x2(i) + b ;

        nt =[nety(i) t(i)];
        delw1 = alpha*(t(i)-nety(i)) * x1(i);
        delw2 = alpha*(t(i)-nety(i)) * x2(i);
        delb = alpha*(t(i)-nety(i)) * x3(i);
        // weight changes
        wc =[ delw1 delw2 delb ]
        // update weights
        w1 = w1 + delw1 ;
        w2 = w2 + delw2 ;
        b = b + delb ;
        //weights
        w =[ w1 w2 b ];
        // input
        x =[ x1(i) x2(i) x3(i) ];
    end

    //printing the error difference
    for i =1:4
        e1 = e1 + (t(i)-nety(i))^2;
    end

    disp("Current epoch:");
    disp(epoch);
    disp("Current epoch error") ;
    disp(e1);
    disp("Error difference");
    disp(e-e1);
end

disp("Total Number of epochs ");
disp(epoch);
disp("The final bias is: ");
disp(b);
disp("The final weights are: ");
disp("w1 =");
disp(w1);
disp("w2 =");
      disp(w2);
```

<u>Output:</u>



```
"Reeha Parkar - 60001180046"

"Adaline network for OR function Bipolar inputs and targets"

"Error after first epoch:"

 3.0210875

"Error after second epoch:"

 1.9384466

"Error difference error2-error1"

 1.0826409

"Current epoch:"

 3.

"Current epoch error"

 1.5493033

"Error difference"

 0.3891434

"Current epoch:"

 4.

"Current epoch error"

 1.4175108

"Error difference"
```



```
 "Current epoch error"

  1.4175108

 "Error difference"

  0.1317924

 "Current epoch:"

  5.

 "Current epoch error"

  1.3782254

 "Error difference"

  0.0392855

 "Total Number of epochs "

  5.

 "The final bias is: "

  0.4575480

 "The final weights are: "

 "w1 ="

  0.4893198

 "w2 ="

  0.5204044

--> 
```

Conclusion :
Hence, in this lab, we developed a SCILAB program for OR function with bipolar inputs and targets using ADALINE network. The solved values and the experiment values match which proves the accuracy of the model.

Department of Electronics Engineering

Review Questions

Answer the following questions on journal sheets and attach the images or scan a pdf for the same.

1. Define delta rule and state error function for delta rule.
2. What is the significance of Adaline network?
3. Explain stopping condition for Adaline network
4. How is a Madaline network formed?
5. State applications for Adaline and Madaline network

Reeha Parkar
60001180046.

NNFL Review questions
ADALINE NETWORK.

1 of 4

Q1. Define delta rule and state error function for delta rule.

Delta rule: the adjustment made to the synaptic weight of a neuron is proportional to the product of the error signal and the input signal of the synapse in ~~sit~~ question.

The update value of synaptic weight is:

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n).$$

in computational terms:

$$w_{kj}(n) = z^{-1}[w_{kj}(n+1)]$$

z^{-1} : storage element.

where: weight change is given as:

$$\Delta w_{kj}(n) = \eta e_k(n)\, x_j(n)$$

where $e_k(n)$ = error function and it is given as:

$$e_k(n) = [d_k(n) - o_k(n)]\, f'(net)$$

and

$f'(net)$ = derivative of the activation function

Q2. What is the significance of Adaline network?

- Adaptive linear neural element network makes use of supervised learning with linear activation function. It has only one output unit.
- It may be trained using delta rule.
- It helps minimize the mean squared error

Recha Parkar
60001180046

2 of 4

between the activation and target values
- It uses bipolar activation function.
- The weights and bias in the algorithm are adjustable
- It allows us to keep track of the error for each iteration, hence helps understand better, the effectiveness of the model and create a stopping condition
- the network training is contained until this error is minimized to a very small value.

Q3. Explain stopping condition for Adaline network:
Adaline : training algorithm.
(i) weights and bias are set to some random values but not zero. set the learning parameter α.
(ii) set activations for the input units $i=1$ to n.
(iii) Net input : $y_n = b + \sum_{i=1}^{n} x_i w_i$
(iv) Update weights and bias.
$$w_i(new) = w_i(old) + \alpha(t - y_{in})x_i$$
$$b(new) = b(old) + \alpha(t - y_{in})$$
(v) Once the weight change that occured during the training is smaller than a specified tolerance, then stop the training process, else continue. This is the test of stopping condition for this network.
(vi) Perform steps (ii) to (v) for each bipolar training, when stopping condition is false.

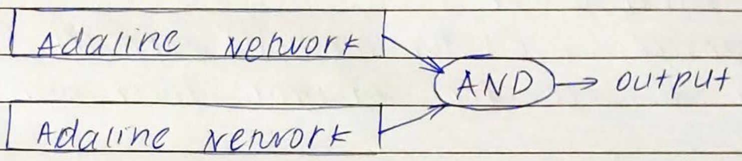
Q4. How is Madaline Network formed?

FOR EDUCATIONAL USE

Reecha Parkar
60001180046

3 of 4

- Madaline network is formed using many adaline networks in parallel with a single output whose value is based on certain selection values rules.
- It may use a majority rate rule.

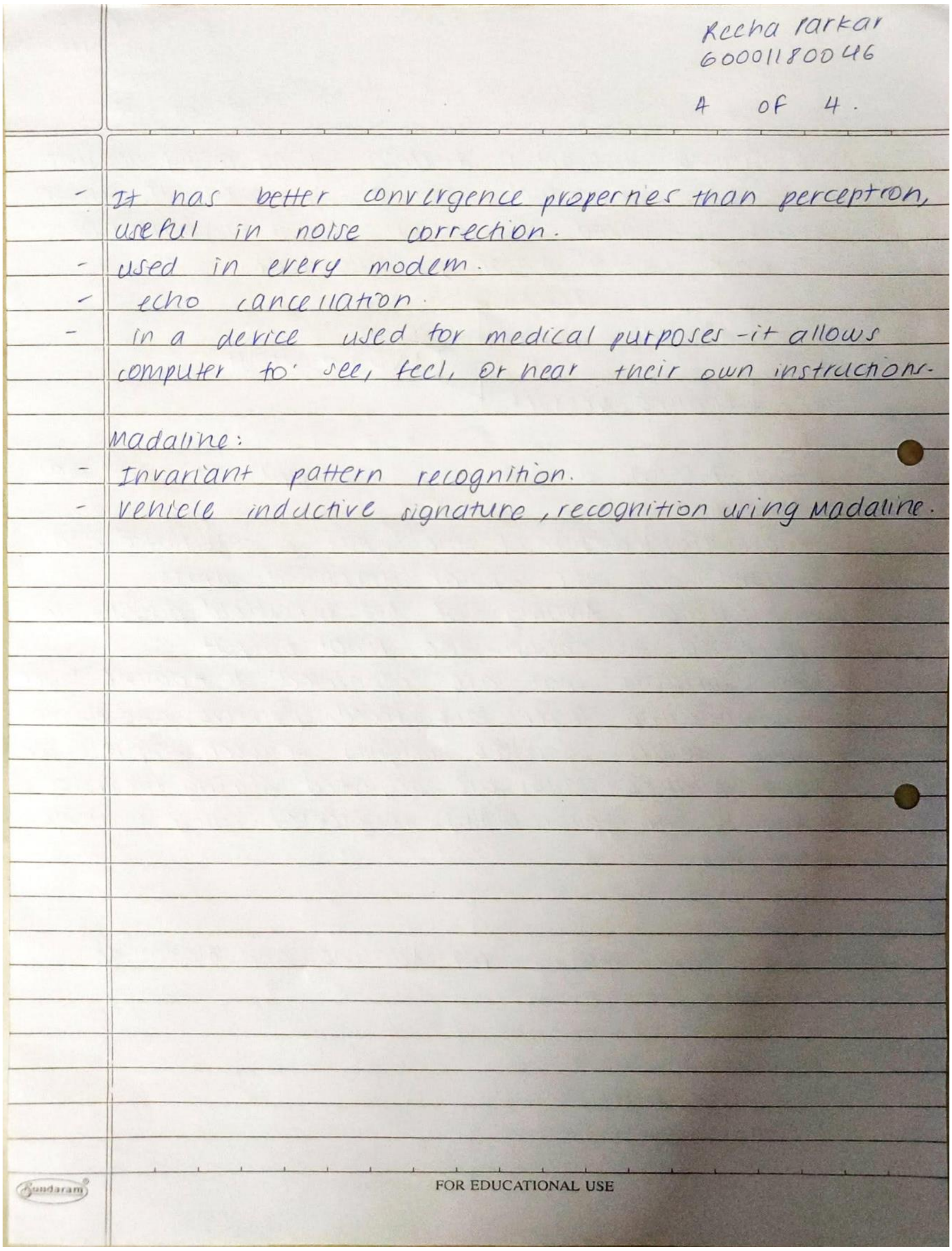


- Each Adaline unit receives the input bits and bias function.
- The weighted sum of the input is calculated and passed onto the bipolar threshold units.
- The logical ANDing of two threshold outputs are computed to obtain the final output.
- The weights that are connected to Adaline layer to Madaline layer are fixed, positive, and possess equal values., and the weights between input layer and Adaline layer are adjusted during the process.
- Non linear separability is tackled using multiple Adalines.

Q5. State applications for Adaline and Madaline network.

Adaline:
- Adaptive filtering
- pattern recognition

FOR EDUCATIONAL USE



Reeha Parkar
60001180046

4 of 4.

- It has better convergence properties than perceptron, useful in noise correction.
- used in every modem.
- echo cancellation.
- in a device used for medical purposes - it allows computer to see, feel, or hear their own instructions.

Madaline:
- Invariant pattern recognition.
- vehicle inductive signature, recognition using Madaline.