Assignment - 1

Deepak Yadav

Q4) ans)

```
function Perceptron()
  clear all, close all;
  DataPoints = [-10 -10;-8 -2; -6 -12; -4 -4; 10 10;8 2; 6 12; 4 4;];
  DataPoints = DataPoints';
  DataLabel = [-1 -1 -1 -1 1 1 1 1];
  class1Pts = DataPoints(:, DataLabel == -1);
  class2Pts = DataPoints(:, DataLabel == 1);
  plot (class1Pts(1, :), class2Pts (2, :), 'bo', class2Pts (1, :), class2Pts (2, :), 'rx');
  W = findSepLine(class1Pts, class2Pts);
end
function W = findSepLine (class1Pts, class2Pts)
  class1Pts = [1,1,1,1;class1Pts];
  class2Pts = [1,1,1,1;class2Pts];
  W = [0 1 -1];   %x = y line
  xMin = min(class1Pts(2,:)); %x range for plotting the line
  xMax = max(class2Pts(2,:));
  xrange = xMin:0.01:xMax;
  alpha = 0.005;
  fprintf('Alpha : %f\n',alpha)
  for i = 1:30
     % ax + by + c = 0
     a = W(2);
     b = W(3);
     c = W(1);
     y = (-a*xrange -c)/b; %Generate y points on the boundary
     clf;
     plot (class1Pts(2,:), class1Pts(3,:), 'bo', class2Pts(2,:), class2Pts(3,:), 'rx');
     hold on;
     plot (xrange, y,'r'); pause(5); %Plot the boundary
     %Pick misclassified samples from negative class
     predClss1 = class1Pts'*W'; %Predict class for negative class samples
```

```
	pickdmcSac1 = find(predClss1>0); %pick those which are wrongly predicted
as positive
	sumdxX1 = sum(pickdmcSac1, 2); %compute gradient contribution from
negative sample
	%Pick misclassified samples from positive class
	predClss2 = class2Pts'*W'; %Predict class for positive class samples
	pickdmcSac2 = find(predClss2<=0); %pick those which are predicted as
from negative class
	sumdxX2 = sum(-1*pickdmcSac2, 2); %compute gradient contribution from
negative samples
	errCst = sumdxX1 + sumdxX2 %compute gradient for both negative and
positive class
	W = W - alpha*errCst %apply gradient descent
	if (isempty(pickdmcSac1) && isempty(pickdmcSac2))
	   %if no misclassified samples
	   return;
	end
  end
end
```

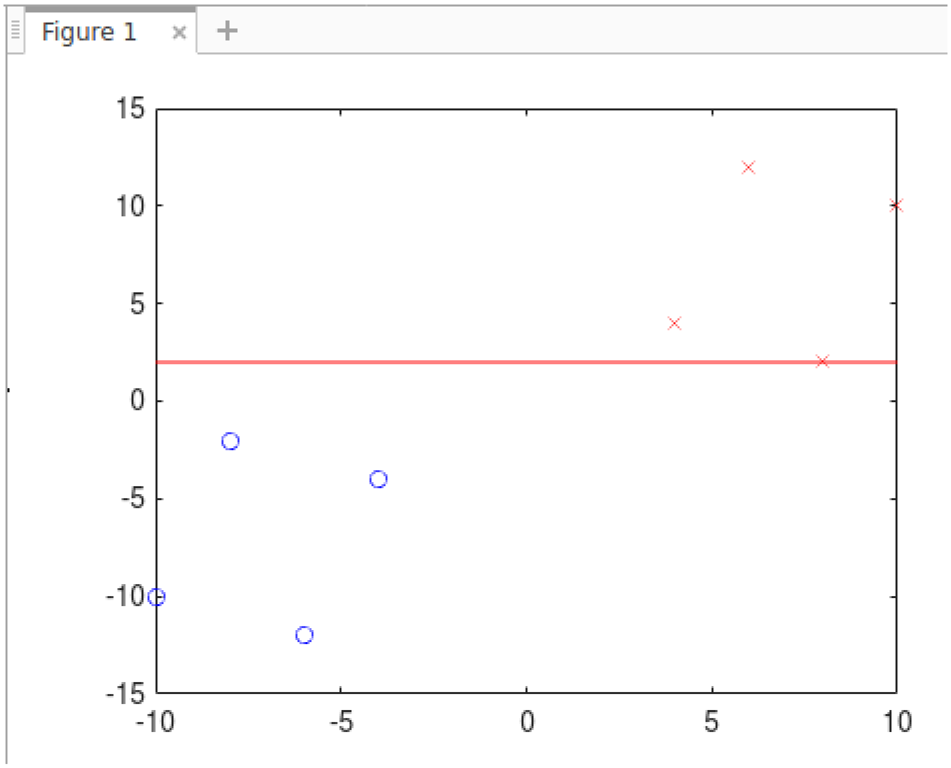


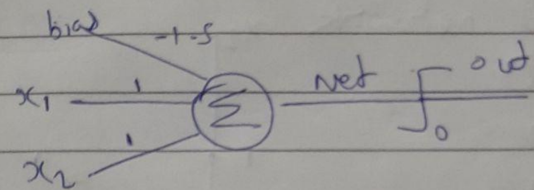
Figure 1
15
10
5
0
-5
-10
-15
-10
-5
0
5
10

DEEPAK YADAV
AM.EN: U4AIE19024

Page No.:
Date:

Assignment - 1

1 ans) Training a neural network means identifying parameter of a neuron. Lets take eg., of a 'and' neural network

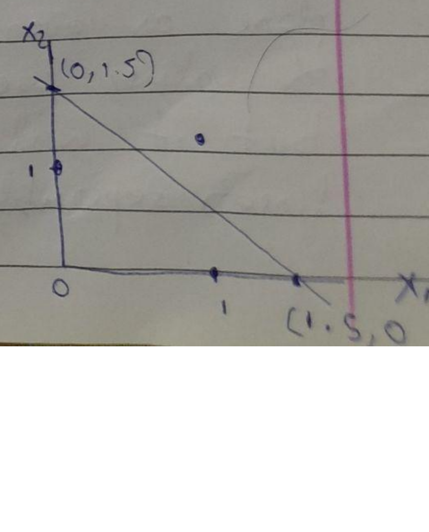


So from the diagram -1.5, 1, 1 are weights and threshold is also a parameter since we use it as a step function.

2 ans) A neuron can implement a line on a plane so it means its capacity is accordance to the data which is linearly seperable, then a perceptron or a single neuron can always find a decision boundary and will always converge.

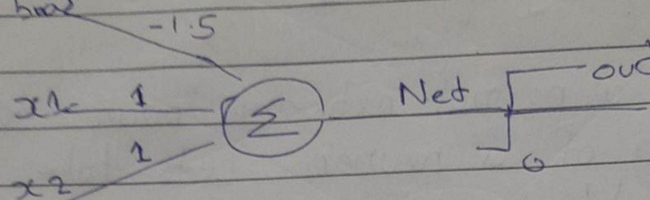
3 ans) AND neural network

X1	X2	O
0	0	0
0	1	0
1	0	0
1	1	1



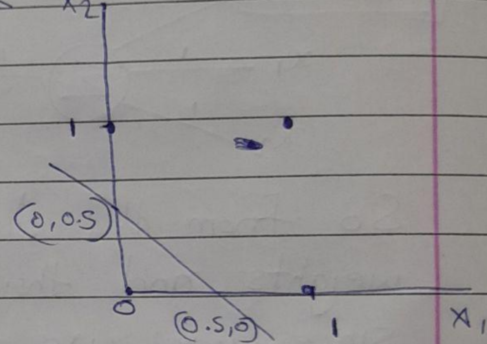
DEEPAK YADAV
AM.EN.U4AIE19024

So decision boundary will be $x1 + x2 - 1.5 = 0$

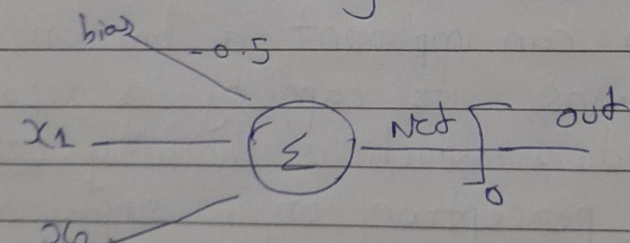


OR neural network

$x1$	$x2$	O
0	0	0
0	1	1
1	0	1
1	1	1



decision boundary will be $x1 + x2 - 0.5 = 0$



5 No, it will not give the best hyper plane as eventhough we change different α values, we will get different line but it will not be the best. But for linear seperable data it will always converge.

DEEPAK YADAV
4M.EN.U4AIE19024

The best hyper plane is when boundary which maximizes the margin and that is given by the support vector machine (SVM).

6 Since XOR is non-linear separable problem we would not be able to find $W^T X$ that separate training sample of two classes. So it will never converge.
Even if we change the activation function to more sophisticated sigmoid and neuron activation saturates at either close to 0 or 1, the graddient at these region is almost zero.

XOR neural network

$$A \oplus B = A\bar{B} + \bar{A}B$$
$$= \bar{A}B + A\bar{B} + A\bar{A} + B\bar{B}$$
$$= A(\bar{A}+\bar{B}) + B(\bar{A}+\bar{B})$$
$$= (A+B)(\bar{A}+\bar{B})$$
$$= (A+B)(\overline{AB})$$

So from here we can see that XOR gate consist of OR, NAND and AND gate
Decision boundary now consist of:

OR $\rightarrow$ ~~$2x_1 + 2x_2$~~ $2A + 2B = 1$

NAND $\rightarrow$ $A + B = 2$

AND $\rightarrow$ $A + B = 1$

DEEPAK YADAV
AM.EN.U4AIE19027