Lab Class ML:V

Ankit Satpute, 120825 ;
Hsueh Wei, 120820;
Sagar Nagaraj Simha, 120797
(M.Sc. CS4DM) - Group 13

2a)

Training data

A	B	A ^ ~B	
0	1	0	F
0	0	0	F
1	1	1	T
1	0	0	F

y(xA, xB) = w0.x0 + w1.xA + w2.XB, where x0 =1

Diagram of the training data and the chosen decision boundary in coordinate system

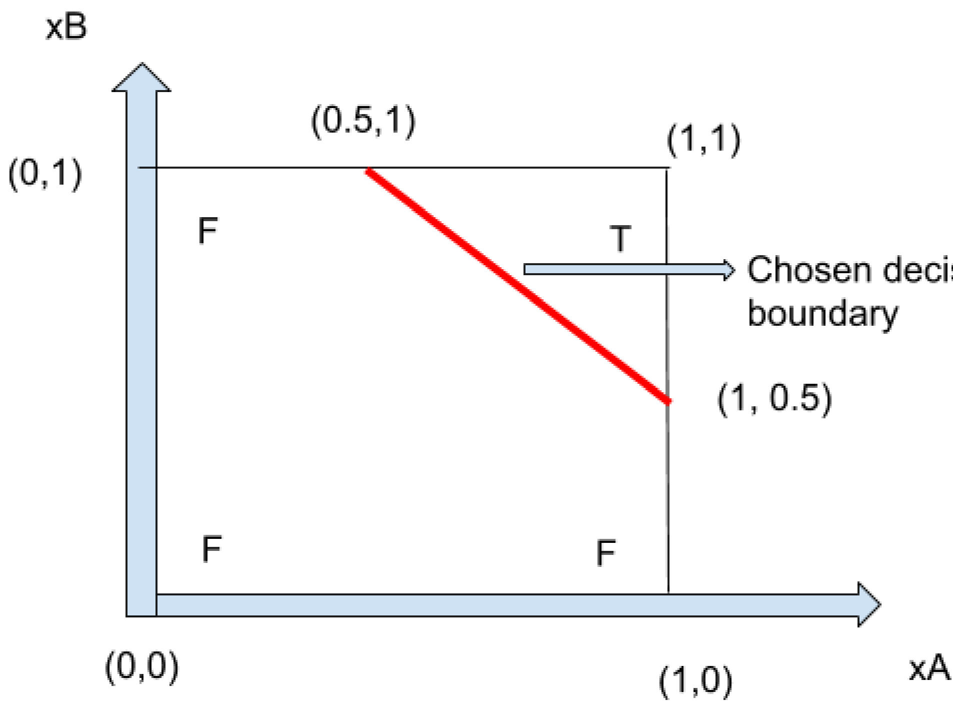


Figure1

Perceptron:

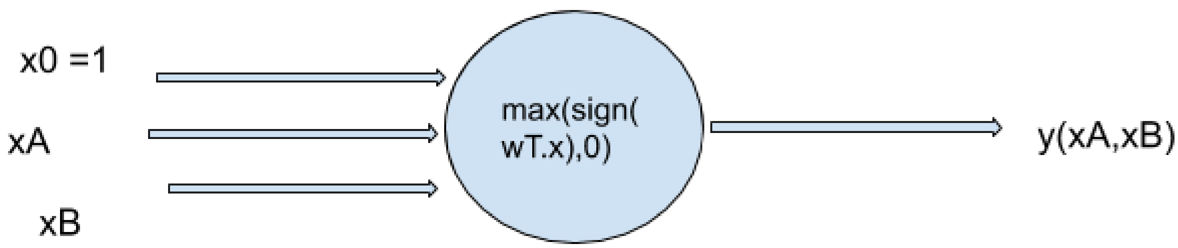


Figure2

Let the decision boundary be the line with two points (0.5,1) and (1,0.5) as shown by the red line in Figure 1 above.

Equation of the decision boundary line is xA+xB = 1.5, where xA+xB>1.5 classifies as T and xA+xB<1.5 classifies as F.

-1.5+xA+xB = 0

The weight vectors are:
w0= -1.5
w1 = 1
w2 = 1

Check with the training data,

1) x = [1 0 1], c=0
   y = max(sign(-1.5 + -1.5*0 + 1*1), 0) = 0
2) x = [1,0,0], c=0
   y = max(sign(-1.5 + -1.5*0 + 1*0), 0) = 0
3) x = [1,1,1], c=1
   y = max(sign(-1.5 + 1*1 + 1*1), 0) = 1
4) x = [1,1,0], c=0
   y = max(sign(-1.5 + 1*1 + 1*0), 0) = 0

2b)

Batch gradient Descent , n (learning rate) = 0.1

Training data

x0	x1	x2	c(x)
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Initial weight vector, w = [-0.5, 0.5, 0.5]

1) Iteration 1:
   Delta_w = [0,0,0]
   a) For [x,c] = [(1,0,0), 0]
      error = 0 - max(sign(-0.5*1 + 0.5*0 +0.5*0)
      = 0-0 = 0
      Delta_w = [0 0 0] + 0.1*0 * [1 0 0]
      Delta_w = [0 0 0]
   b) For [x,c] = [(1,0,1), 0]
      error = 0 - max(sign(-0.5*1 + 0.5*0 +0.5*1)
      = 0-0 = 0
      Delta_w = [0 0 0] + 0.1*0 * [1 0 1]

Delta_w = [0 0 0]

c) For [x,c] = [(1,1,0), 1]

error = 1 - max(sign(-0.5*1 + 0.5*1 +0.5*0)

= 1-0 = 1

Delta_w = [0 0 0] + 0.1*1* [1 1 0]

Delta_w = [0.1 0.1 0]

d) For [x,c] = [(1,1,1), 0]

error = 0 - max(sign(-0.5*1 + 0.5*1 +0.5*1)

= 0-1 = -1

Delta_w = [0.1 0.1 0] + 0.1*-1* [1 1 1]

Delta_w = [0 0 -0.1]

w = w + Delta_w

w = [-0.5 0.5 0.5] + [0 0 -0.1]

w = [-0.5 0.5 0.4] after iteration 1

2) Iteration 2:

Delta_w = [0,0,0]

a) For [x,c] = [(1,0,0), 0]

error = 0 - max(sign(-0.5*1 + 0.5*0 +0.4*0)

= 0-0 = 0

Delta_w = [0 0 0] + 0.1*0 * [1 0 0]

Delta_w = [0 0 0]

b) For [x,c] = [(1,0,1), 0]

error = 0 - max(sign(-0.5*1 + 0.5*0 +0.4*1)

= 0-0 = 0

Delta_w = [0 0 0] + 0.1*0 * [1 0 1]

Delta_w = [0 0 0]

c) For [x,c] = [(1,1,0), 1]

error = 1 - max(sign(-0.5*1 + 0.5*1 +0.4*0)

= 1-0 = 1

Delta_w = [0 0 0] + 0.1*1* [1 1 0]

Delta_w = [0.1 0.1 0]

d) For [x,c] = [(1,1,1), 0]

error = 0 - max(sign(-0.5*1 + 0.5*1 +0.4*1)

= 0-1 = -1

Delta_w = [0.1 0.1 0] + 0.1*-1* [1 1 1]

Delta_w = [0 0 -0.1]

w = w + Delta_w
w = [-0.5 0.5 0.4] + [0 0 -0.1]
w = [-0.5 0.5 0.3] after iteration 2

Check:

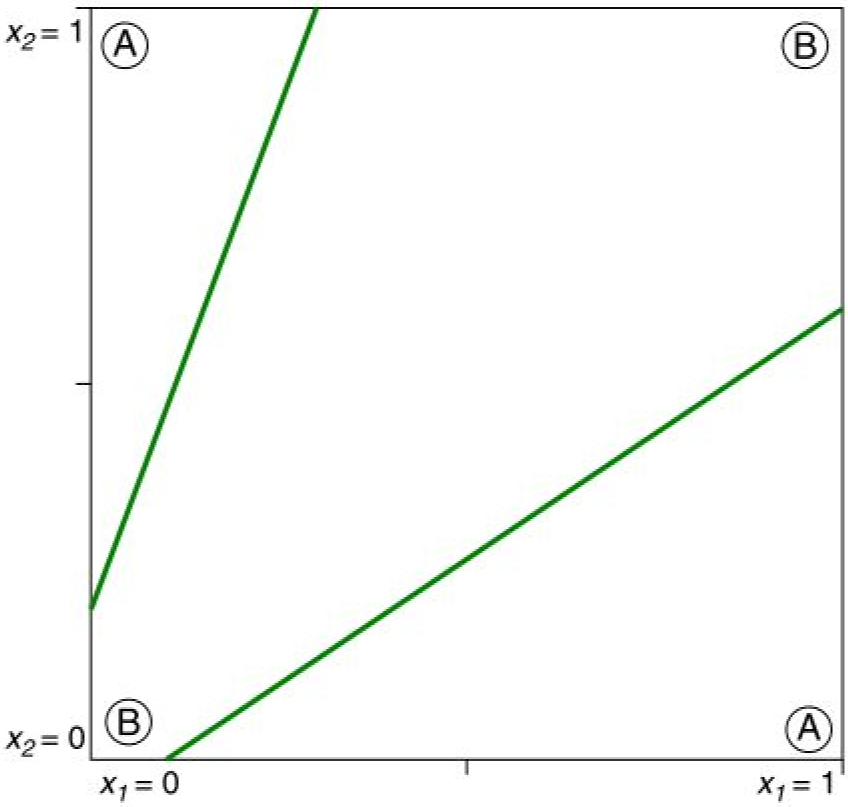
1) x = [1 0 0], c=0
   y = max(sign(-0.5*1 + -1.5*0 + 0.3*0), 0) = 0
2) x = [1,0,1], c=0
   y = max(sign(-0.5*1 + 0.5*0 + 0.3*1), 0) = 0
3) x = [1,1,0], c=1
   y = max(sign(-0.5*1 + 0.5*1 + 0.3*0), 0) = 0
4) x = [1,1,1], c=0
   y = max(sign(-0.5*1 + 0.5*1 + 0.3*1), 0) = 1

3) and 4) are misclassified by w=[-0.5 0.5 0.3] after 2 iterations

2c)

Truth table for XOR:

x1	x2	XOR	Class
0	0	0	B
1	0	1	A
0	1	1	A
1	1	0	B



[From ML slides]

A single perceptron only divides the plane into a linearly separable boundaries. But the XOR data as shown in the diagram through the truth table cannot be separated by a single hyperplane and requires two hyperplanes. And hence, XOR which represents a ‘non linearly separable sets’ requires more than one perceptron.

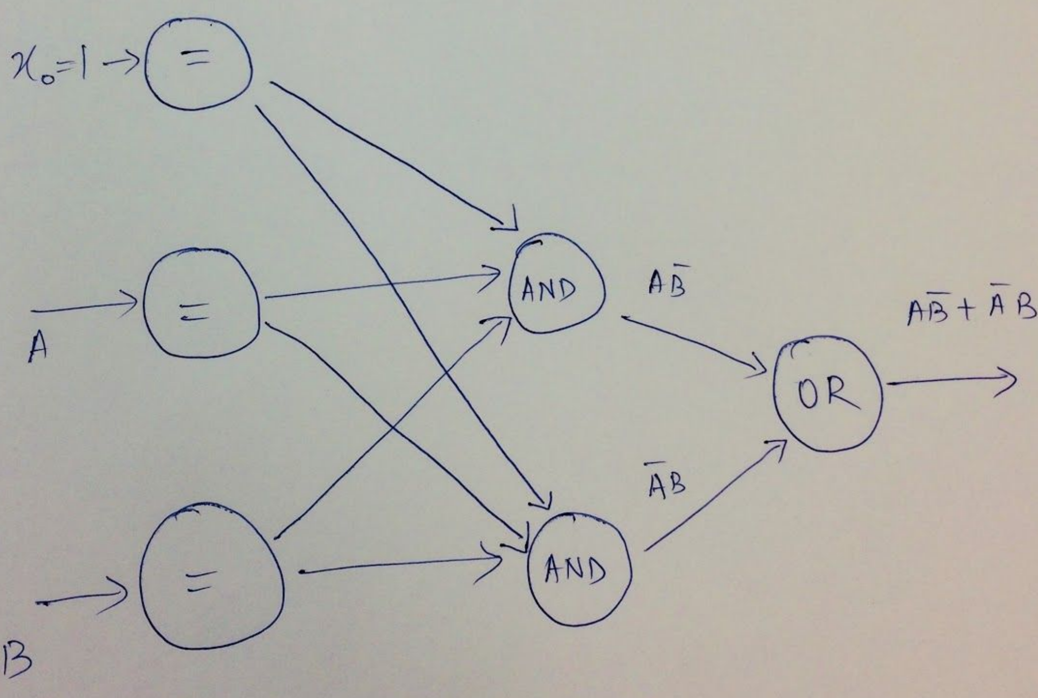
2d)

A XOR B = A. (~B) + (~A).B

The neurons would learn the logic of AND, AND and OR respectively when fed with the training data for XOR

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

$$A \oplus B = A\bar{B} + \bar{A}B$$



#References:
ML lecture slides