BAUHAUS UNIVERSITÄT WEIMAR

INTRODUCTION TO MACHINE LEARNING

Bauhaus-Universität Weimar

ASSIGNMENT 4

SUBMITTED BY

Anjana MuraleedharanNair(125512)

Isabel Maria Binu (125514)

Vishal Sanjay Shivam (125353)

Sharat Anand (125404)

Exercise 1 : Gradient Descent

(a) Name one difference between the perceptron training rule and the gradient descent method.

One key difference between the perceptron training rule and the gradient descent method lies in their approach to updating the weights during the learning process.

The perceptron training rule updates weights incrementally based on the error in individual predictions, adjusting them proportionally to the misclassification. In contrast, gradient descent minimises a global cost function by iteratively updating weights in the direction of steepest decrease, using the gradient of the cost function with respect to the weights and a learning rate.

(b)Name the difference in the algorithm between batch gradient descent and incremental gradient descent.

Batch Gradient Descent updates weights based on the average gradient of the entire dataset, providing stable but computationally expensive updates. In contrast, Incremental (Stochastic) Gradient Descent updates weights after processing each individual training instance, offering computational efficiency but with more frequent, potentially noisy updates.

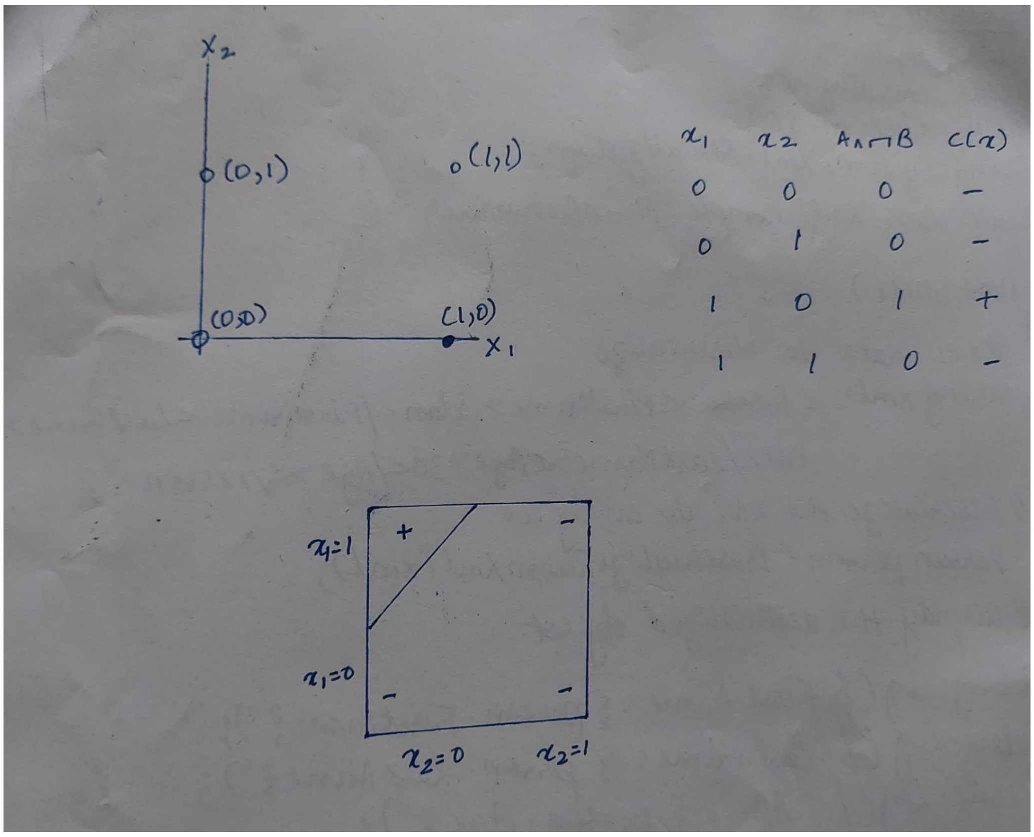
Exercise 2 : Perceptron Learning

Given two perceptrons y0() and y1() defined by heaviside(Pp j=0 wjxj) as usual with identical weights except that w0 = 0 for y0() but w0 = 1 for y1(). Is one of y0() and y1() more general than the other, and if yes, which one (or both)? Explain your answer.

y(1), with (w0 = 1, is more general than y(0), where w0 = 0 , as it considers the constant term in the input, allowing it to model a broader range of decision boundaries. y(0), on the other hand, is limited to decision boundaries passing through the origin.

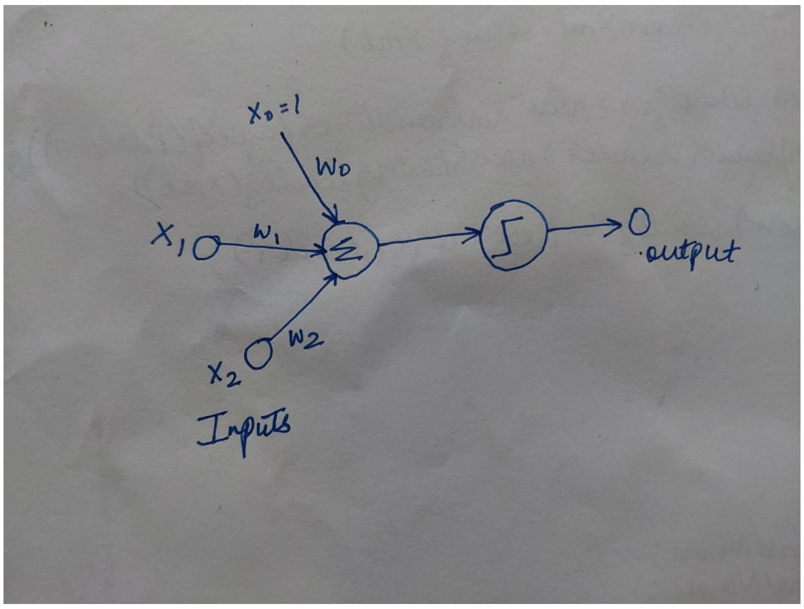
Exercise 3 : Perceptron Learning

(a)



x_1	x_2	$A \wedge \neg B$	$c(x)$
0	0	0	−
0	1	0	−
1	0	1	+
1	1	0	−

(b)



(c) The perceptron's output is calculated as y(x1,x2) = wo. xo+ w1. x1 + w2. x2,
where, wo (x0 =1) is the bias term, x1 is the input A, and x2 is the input ¬B
The boolean formula A⋀¬B is true when A is true (1) and ¬B is true (i.e. B is false or 0). Weight for A (w1): Since we want A to be true (1), set w1 to 1. Weight for ¬B (w2): Since we want ¬B to be true (0 for B being false), set w2 to 1.
So, the suitable weights are w = (0, 1, 1). With these weights, the perceptron will correctly implement the boolean formula A⋀¬B.
The decision boundary is w=w0+x1*w1+x2*w2= 0+1*1+0*(1) = 1

(d) w0 = −0.5 and w1 = w2 = 0.5 η=0.3

x_1	x_2	c
0	0	0
0	1	0
1	0	1
1	1	0

For x1=0, x2=0, c=0

y= (w0x0 + w1x1 + w2x2)
=(-0.5 x1+ 0.5x0 +0.5 x0) = 0.5<0
y= 0

Updating weights
w0 = w0+η (c-y).x0 = -0.5+0.3(0-0).1 = -0.5
w1= w1+η (c-y). x1 = 0.5+ 0.3 (0-0).0 = 0.5
w2=w2+η (c-y).x2 = 0.5+0.3(0-0).0 = 0.5

For x1=0, x2=1, c=0
y= (0.5x1 + 0.5x0 + 0.5x1) =0= 0, y=1
w0= -0.5+0.3(0-1).1 = -0.8
w1= 0.5+ 0.3 (0-1).0 = 0.5
w2= 0.5+0.3(0-1).1 = 0.2

For x1=1, x2=0, c=1
y= (-0.8x1 + 0.5x1 + 0) =0.3<0 , y=0
w0= -0.8+0.3(1-0).1 = -0.5
w1= 0.5+ 0.3 (1-0).1 = 0.8
w2= 0.2+0.3 (1-0).0 = 0.2

For x1=1, x2=1, c=0
y=(-0.5x1 + 0.8x1 +0.2) =0.5>0 , y=1
w0= -0.5+0.3(0-1).1 = -0.8
w1= 0.8+ 0.3 (0-1).1 = 0.5
w2= 0.2+0.3 (0-1).1 = -0.1

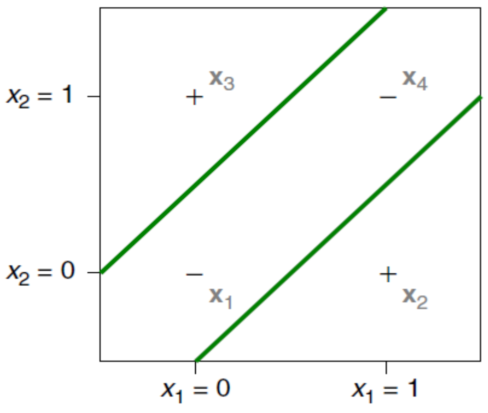
(e) The learning rate impacts how quickly a model converges to a solution. If it's too small, minimal weight changes per update necessitate more training epochs, resulting in a lengthier process. Conversely, a too-high learning rate may cause divergence, impeding convergence with rapid changes that require fewer epochs but may not yield a stable solution. Large learning rates risk unstable training due to swift weight updates and potential divergence, while small rates lead to slow convergence but generally maintain stability.

Exercise 4 : Perceptron Learning

Why can the boolean formula A XOR B not be learned by a single perceptron? Justify your answer with a drawing.

The boolean formula A XOR B cannot be learned by a single perceptron because it is not linearly separable. This indicates that the inputs cannot be efficiently divided into the proper output classes by a single perceptron, which has a linear decision boundary.

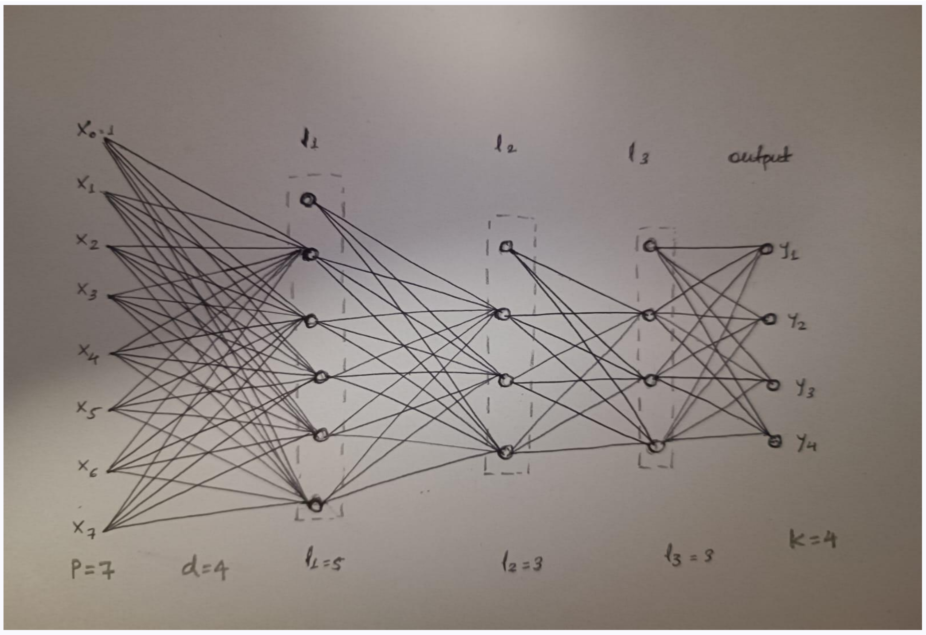
	x1	x2	XOR	c
X1	0	0	0	-
X2	1	0	1	+
X3	0	1	1	+
X4	1	1	0	-



Plotting this makes it clear that the two classes cannot be divided by a single straight line. This drawback emphasizes the requirement for multilayer perceptrons to represent non-linear decision boundaries.

Exercise 6 : Parameters of the Multilayer Perceptrons

(a)



(b) Number of weights = (p+1)*l1 + (l1+1)*l2 + (l2+1)*l3 + (l3+1)*k

= (7+1)5 + (5+1)3 + (3+1)3 +(3+1)4

= 40+18+12+16

= 86

(c) Doubling the units

l1=10,l2=6,l3=6

Number of weights = 80+66+42+28 = 216