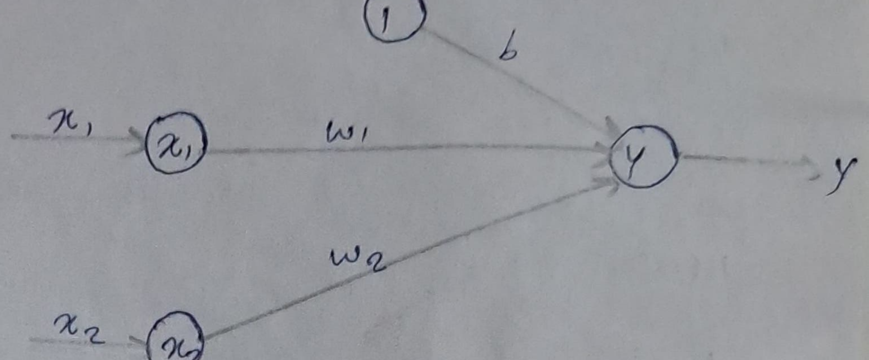
a) Implement OR function using the perceptron training algorithm with binary inputs and bipolar targets.

Ans) Truth table (bipolar form)

x_1	x_2	t
1	1	1
1	-1	1
-1	1	1
-1	-1	-1



Initial weights, $w_1 = w_2 = 0$

Bias $b = 0$ $\quad \alpha = 1$

$$y = \begin{cases} 1 & \text{if } y_{in} > 0 \\ 0 & \text{if } y_{in} = 0 \\ -1 & \text{if } y_{in} < 0 \end{cases}$$

> EPOCH - 1

where

$y_{in} = x_1 w_1 + x_2 w_2 + b$

Pattern 1: $(x_1 = 1, x_2 = 1, t = 1)$

$y_{in} = (1 \times 0) + (1 \times 0) + 0 = 0$

activation: $y_{in} = 0 \Rightarrow y = 0$

$t = 1, y = 0 \rightarrow$ error $\rightarrow$ update required $(t \neq y)$

$w_i(new) = w_i(old) + \alpha t x_i$

$w_1(new) = 0 + 1 \times 1 \times 1 = 1$

$w_2(new) = 0 + (1 \times 1 \times 1) = 1$ $\quad b(new) = b(old) + \alpha t$

$b(new) = 0 + 1 \times 1 = 1$

Pattern 2 ($x_1 = 1$, $x_2 = -1$, $t = 1$)

$y_{in} = (1 \times 1) + (-1 \times 1) + 1 = 1$

$y_{in} > 0 \Rightarrow y = 1$

$t = 1$, $y = 1$ $t = y$ so no update
$w_1 = 1$, $w_2 = 1$, $b = 1$

Pattern 3 ($x_1 = -1$, $x_2 = 1$, $t = 1$)

$y_{in} = (-1 \times 1) + (1 \times 1) + 1 = 1$

$y = 1$ $t = 1$, $t = y$ so no update.

Pattern 4 ($x_1 = -1$, $x_2 = -1$, $t = -1$)

$y_{in} = (-1 \times 1) + (-1 \times 1) + 1 = -1$

$y_{in} < 0 \Rightarrow y = -1$

$t = -1$, $y = -1$ $t = y$ so no update.

> EPOCH - 2

All outputs are correct. So no weight change.
training stops.

final weights, $w_1 = 1$, $w_2 = 1$, $b = 1$

b): Explain the training algorithm used for a perceptron network with single output classes.

Ans): The perceptron training Algorithm proposed by frank Rosenblatt, is a supervised learning algorithm used to train a single layered perceptron.

Steps in training Algorithm.

> step 0: Initialization

- Initialize all weights $w_1, w_2 \cdots w_n$ to small random values or zero.
- Initialize bias b. and also choose learning rate, α $(0<\alpha\leq 1)$

> step 1: • Perform step 2-6 until the final stopping condition is false

> step 2: Perform steps 3-5 for each training pair indicated by $s:t$.

> step 3: The ~~o/p~~ i/p layer containing input unit is applied with identity activation functions. $x_i = s_i$

> step 4: Calculate the o/p of network

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

$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} \geq \theta \\ 0 & \text{if } -\theta \leq y_{in} \leq \theta \\ -1 & \text{if } y_{in} < -\theta \end{cases}$$

> step 5: weight and bias adjustment.

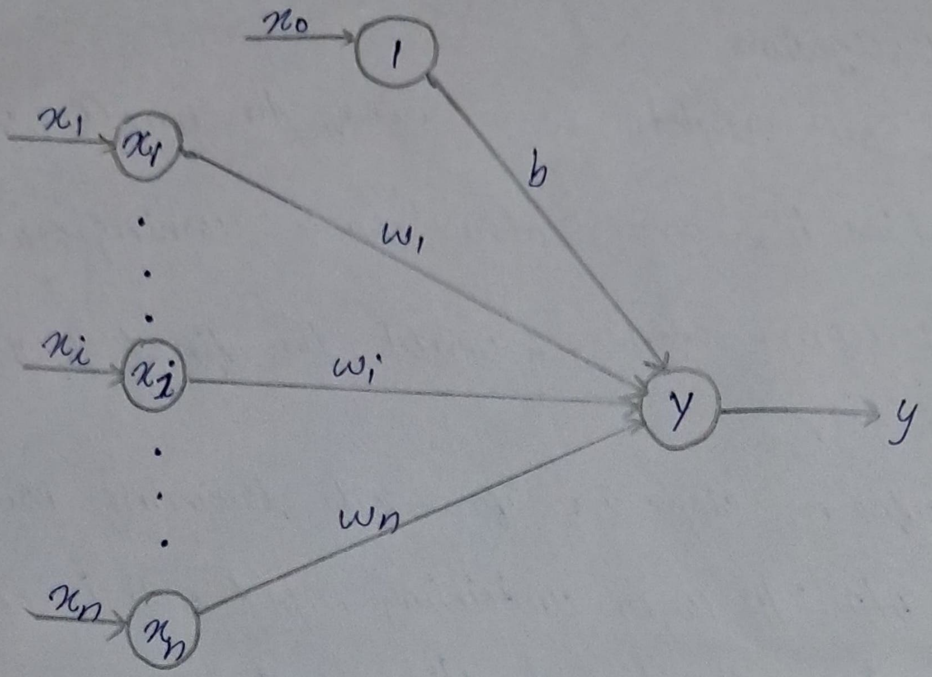
if $y \neq t$ then $w_i(new) = w_i(old) + \alpha t x_i$

$b(new) = b(old) + \alpha t$

else, we have $w(new) = w(old)$

$b(new) = b(old)$

> step 6: Train the network until there is no weight change otherwise, start again from step 2.



x0
1
x1
x1
b
w1
xi
xi
wi
y
y
wn
xn
xn