1. Why is a perceptron (which uses a **sigmoid** activation function) equivalent to *logistic regression*?

- Has a weight associated with each input (attribute)
- Output is acquired by applying an activation function
  - If use sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$ to linear comb of inputs $\theta_0 + \theta_1 x_1 + \cdots \Rightarrow \sigma(\underset{\sim}{\theta}^T \underset{\sim}{x})$: logistic regression

2. Consider the following training set:

(x_1, x_2)	y
(0,0)	0
(0,1)	1
(1,1)	1

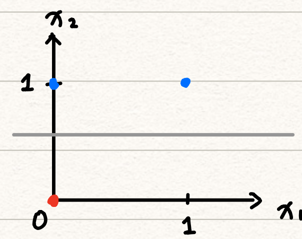
With the bias value of 1, the initial weight function of $\theta = \{\theta_0, \theta_1, \theta_2\}$ = {0.2, -0.4, 0.1} and learning rate of $\eta = 0.2$.

Consider the activation function of the perceptron as the step function $f = \begin{cases} 1 & if\ \Sigma > 0 \\ 0 & otherwise \end{cases}$

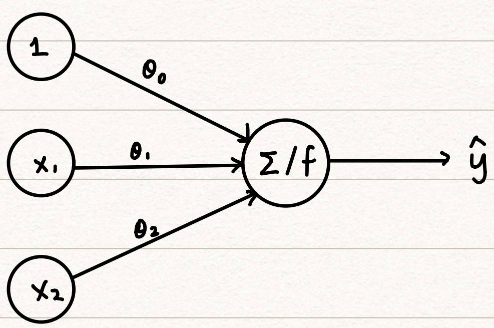
a) Can the perceptron learn a perfect solution for this data set?

Linearly separable ?

Y $\Rightarrow$ Can learn perfect solution.



b) Draw the perceptron graph and calculate the accuracy of the perceptron on the training data before training?



Acc : $\theta = \{0.2, -0.4, 0.1\}$

$f = \begin{cases} 1 & if\ \Sigma > 0 \\ 0 & otherwise \end{cases}$

(x_1, x_2)	$\Sigma = \theta_0 + \theta_1 x_1 + \theta_2 x_2$	$\hat{y} = f(\Sigma)$	y
(0,0)	0.2	1	0
(0,1)	0.2 + 0.1 = 0.3	1	1
(1,1)	0.2 - 0.4 + 0.1 = -0.1	0	1

Acc : $\frac{1}{3}$

c) Using the perceptron *learning rule* and the learning rate of $\eta = 0.2$. Train the perceptron **for one epoch**. What are the weights after the training?

Perceptron weight training rule : (for each instance) (online algorithm, contrast to batch algorithm e.g. NB, LR with GD)

Can be more efficient for large dataset

$$\theta_j^{(t)} \leftarrow \theta_j^{(t-1)} + \eta\,(y^i - \hat{y}^{i,(t)})\,x_j^i$$

lr $\eta = 0.2$

$\theta = \{0.2, -0.4, 0.1\}$

(x_1, x_2)	$\Sigma = \theta_0 + \theta_1 x_1 + \theta_2 x_2$	$\hat{y} = f(\Sigma)$	y	
$(0,0)$	0.2	1	0	✗
	Update θ: $\theta_0^{(1)} = \theta_0^{(0)} + \eta(y^1 - \hat{y}^1)\,x_0^1$ $= 0.2 + 0.2(0-1)\cdot 1 = 0$ $\theta_1^{(1)} = \theta_1^{(0)} + \eta(y^1 - \hat{y}^1)\,x_1^1 = -0.4$ $\theta_2^{(1)} = \theta_2^{(0)} + \eta(y^1 - \hat{y}^1)\,x_2^1 = 0.1$			
$(0,1)$	$0 - 0.4 \times 0 + 0.1 = 0.1$ Correct $\Rightarrow$ No update	1	1	✓
$(1,1)$	$0 - 0.4 \times 1 + 0.1 = -0.3$	0	1	✗
	Update θ: instance 3 $\theta_0^{(1)} = \theta_0^{(0)} + \eta(y^3 - \hat{y}^3)\,x_0^3$ $= 0 + 0.2(1-0)\cdot 1 = 0.2$ $\theta_1^{(1)} = \theta_1^{(0)} + \eta(y^3 - \hat{y}^3)\,x_1^3$ $= -0.4 + 0.2(1-0)\cdot 1 = -0.2$ $\theta_2^{(1)} = \theta_2^{(0)} + \eta(y^3 - \hat{y}^3)\,x^3$ $= 0.1 + 0.2(1-0)\cdot 1 = 0.3$			

$\underset{\sim}{\theta}^{(1)} = \{0.2, -0.2, 0.3\}$

d) What is the accuracy of the perceptron on the training data after training for one epoch? Did the accuracy improve?

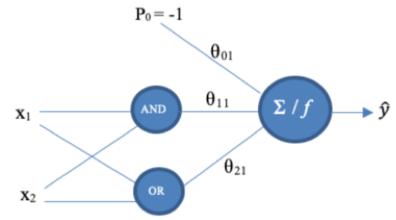
Acc : $\theta = \{0.2, -0.2, 0.3\}$

$f = \begin{cases} 1 & \text{if } \Sigma > 0 \\ 0 & \text{otherwise} \end{cases}$

(x_1, x_2)	$\Sigma = \theta_0 + \theta_1 x_1 + \theta_2 x_2$	$\hat{y} = f(\Sigma)$	y
(0,0)	0.2	1	0
(0,1)	0.2 + 0.3 = 0.5	1	1
(1,1)	0.2 - 0.2 + 0.3 = 0.3	1	1

Acc : $\frac{2}{3}$

3. Consider the two levels deep network illustrated below. It is composed of three perceptron. The two perceptron of the first level implement the AND and OR function, respectively.



Determine the weights θ_{11}, θ_{21} and bias θ_{01} such that the network implements the XOR function. The initial weights are set to zero, i.e., $\theta_{01} = \theta_{11} = \theta_{21} = 0$, and the learning rate η (eta) is set to 0.1.

Notes:

- The input function for the perceptron on level 2 is the weighted sum (Σ) of its input.
- The activation function f for the perceptron on level 2 is a *step function*:

$$f = \begin{cases} 1 & if \sum > 0 \\ 0 & otherwise \end{cases}$$

- Assume that the weights for the perceptron of the first level are given.

XOR: for each training example x

$$\left\{ \begin{array}{l} p \leftarrow (p_0, f_{AND}(x), f_{OR}(x)) \quad \text{(level 2)} \\ \hat{y} \leftarrow f(\Sigma_i \theta_i p_i) = \begin{cases} 1 & \text{if } \Sigma > 0 \\ 0 & \text{otherwise} \end{cases} \end{array} \right.$$

$y \leftarrow$ target of x (XOR)

train: For $i = 1 : n$

$$\left. \begin{array}{l} \Delta\theta_i \leftarrow \eta (y - \hat{y}) p_i \\ \theta_i \leftarrow \theta_i + \Delta\theta_i \end{array} \right\} \text{(update } \theta\text{) training}$$

Output of level 1:

instance	x_1	x_2	$f_{AND}(x)$ (p_1)	$f_{OR}(x)$ (p_2)	target y XOR
1	1	0	0	1	1
2	0	1	0	1	1
3	1	1	1	1	0
4	0	0	0	0	0

all possibilities

Inputs for level 2 perceptron : $\langle p_0, p_1, p_2 \rangle = \langle -1, p_1, p_2 \rangle$

Initial params $\theta : \langle \theta_0, \theta_1, \theta_2 \rangle = \langle 0, 0, 0 \rangle$

$\eta = 0.1$

For first epoch (training) :

	$\langle p_0, \overset{AND}{p_1}, \overset{OR}{p_2} \rangle$	$\Sigma = \theta_0 p_0 + \theta_1 p_1 + \theta_2 p_2$	$\hat{y} = f(\Sigma)$	$\overset{x_1 \text{ XOR } x_2}{y}$	$\Delta\theta_i = \eta(y - \hat{y}) p_i$
1	$\langle -1, 0, 1 \rangle$	0 update θ: $\theta \leftarrow \theta + \Delta\theta$ $\langle 0, 0, 0 \rangle + \langle -0.1, 0, 0.1 \rangle$ $= \langle -0.1, 0, 0.1 \rangle$	0	1	$0.1(1-0) \times \langle -1, 0, 1 \rangle$ $= \langle -0.1, 0, 0.1 \rangle$
2	$\langle -1, 0, 1 \rangle$	$-1(-0.1) + 0.1 = 0.2$	1	1	✗ (No update)
3	$\langle -1, 1, 1 \rangle$	$-0.1(-1) + 0.1 \times 1 = 0.2$ update θ: $\langle -0.1, 0, 0.1 \rangle + \langle 0.1, -0.1, -0.1 \rangle$ $= \langle 0, -0.1, 0 \rangle$	1	0	$0.1(0-1) \times \langle -1, 1, 1 \rangle$ $= \langle 0.1, -0.1, -0.1 \rangle$
4	$\langle -1, 0, 0 \rangle$	$-0.1 \times 0 = 0$	0	0	✗ (No update)
	...	...	...	...	...

No changes after 4 epochs $\longrightarrow$ Converge

Final $\theta = \langle 0, -0.2, 0.1 \rangle$