20CYS304 AI & NN
Deep Learning

Unit 3

Perceptron vs neuron

- In deep learning, the terms “perceptron” and “neuron” are related but have distinct meanings, and they are not exactly the same.

- While both concepts are fundamental building blocks of neural networks, they differ in their historical origins and functionality in modern deep learning models.

- A perceptron is a type of artificial neuron that serves as the building block for more complex neural networks.
- A **perceptron** is the earliest form of a neural network unit, introduced by **Frank Rosenblatt** in 1958.
- It is a **binary classifier** that makes predictions based on a linear combination of input features. The perceptron algorithm was one of the first algorithms used to implement a simple neural network.
- It performs binary classification using a linear decision boundary.
- **Perceptrons are supervised learning algorithms and is a type of an ANN**

Key differences: Perceptron and Neuron

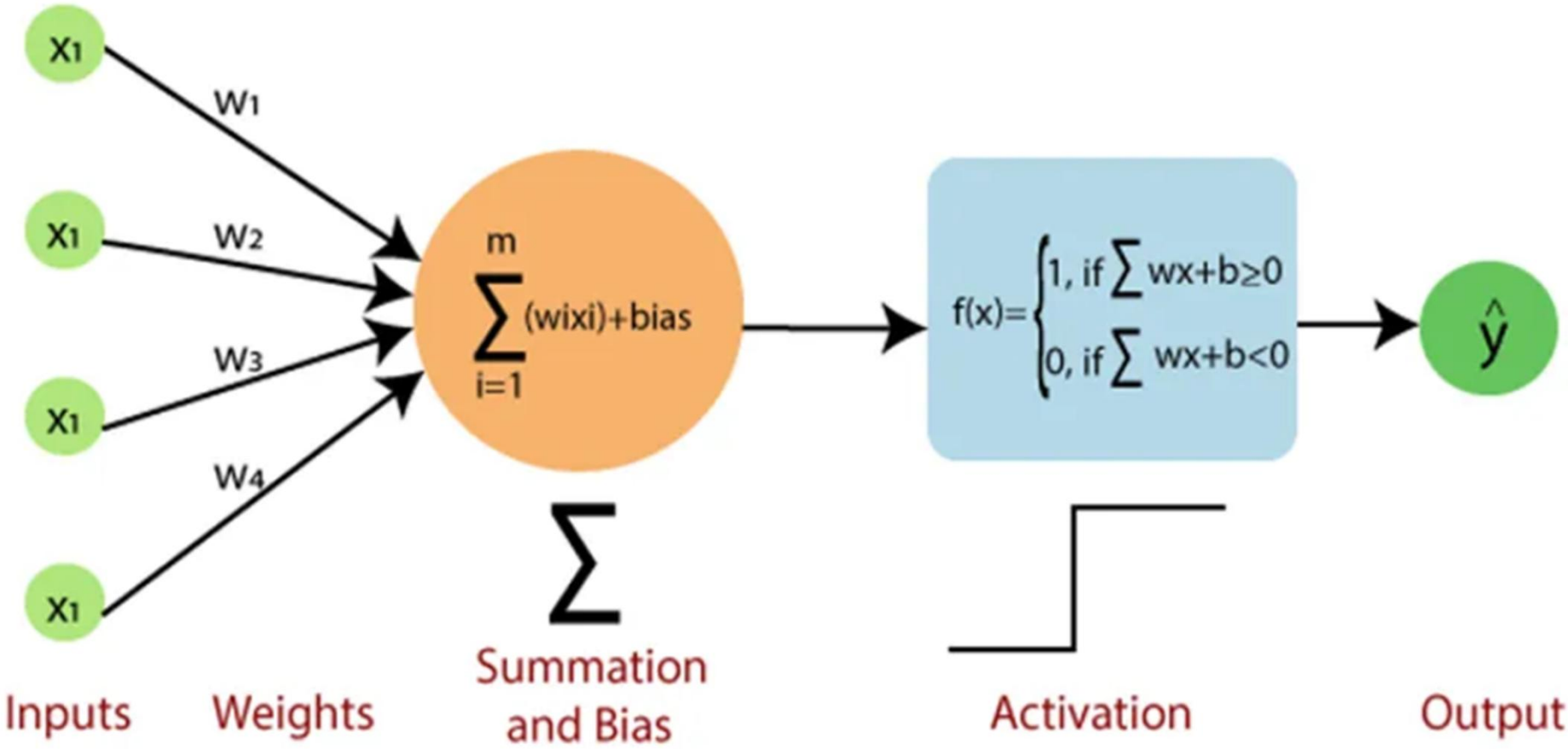
Aspect	Perceptron	Neuron in Deep Learning
Output	Binary (0 or 1)	Continuous or non-linear (depends on the activation function)
Activation Function	Step function	Sigmoid, ReLU, Tanh, Softmax, etc.
Complexity	Simple, handles linearly separable problems	Complex, handles non-linear, multi-dimensiona problems
Layers	Single layer (original perceptron)	Multi-layer (input, hidden, and output layers)
Learning	Basic learning rule for linear problems	Gradient descent with backpropagation
Use Case	Binary classification	Classification, regression, and various deep learning tasks

Components of a Perceptron:

- **Inputs**: The perceptron takes several inputs (x1,x2,…,xn)
- **Weights**: Each input is associated with a weight (w1,w2,…,wn).
- **Bias**: A bias term (b) is added to shift the decision boundary.
- **Activation Function**: The perceptron uses a **step function** (a simple thresholding function) to determine whether the weighted sum of inputs plus the bias is above or below a certain threshold.
- The mathematical representation is:

$$y = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i x_i + b > 0 \\ 0 & \text{if } \sum_{i=1}^{n} w_i x_i + b \leq 0 \end{cases}$$

- **Binary Output**: The output of a perceptron is binary (1 or 0), making it suitable for linearly separable classification problems.


x1
x1
x1
x1
w1
w2
w3
w4
m
∑ (wixi)+bias
i=1
f(x)= 1, if ∑ wx+b≥0
0, if ∑ wx+b<0
ŷ
Inputs
Weights
Summation and Bias
Activation
Output

Eg. 1: Perceptron training rule for OR gate

- ✓ **In OR gate, we find two inputs A, B and one output Y**
- ✓ **We have to assign weight value to each input**
- ✓ **lets assume, w1=0.6, w2=0.6, Threshold=1 Learning rate n=0.5**

A	B	Y=A+B
0	0	0
0	1	1
1	0	1
1	1	1

w1 = 0.6, w2 = 0.6 Threshold = 1 and Learning Rate n = 0.5

A	B	Y=A+B
0	0	0
0	1	1
1	0	1
1	1	1

1. A=0, B=0 and Target = 0
   - $wi.xi = 0*0.6 + 0*0.6 = 0$
   - This is not greater than the threshold of 1, so the output = 0

2. A=0, B=1 and Target = 1
   - $wi.xi = 0*0.6 + 1*0.6 = 0.6$
   - This is not greater than the threshold of 1, so the output = 0

In case 2, the actual output is not matched with target. Hence we have to update the weights using perceptron training rule.

$$wi = wi + n(t - o)xi$$

n =Learning rate
t=target
o=Actual output
xi=input associated with ith weight

2. A=0, B=1 and Target = 1

- $wi.xi = 0*0.6 + 1*0.6 = 0.6$
- This is not greater than the threshold of 1, so the output = 0

$$wi = wi + n(t - o)xi$$

$$w1 = 0.6 + 0.5(1 - 0)0 = 0.6$$

$$w2 = 0.6 + 0.5(1 - 0)1 = 1.1$$

Updated w1 and w2

We need to verify all 4 cases with a new weights

1. A=0, B=0 and Target = 0

- $wi.xi = 0*0.6 + 0*1.1 = 0$
- This is not greater than the threshold of 1, so the output = 0

matched

Lets check with remaining cases

2. A=0, B=1 and Target = 1

- $wi.xi = 0*0.6 + 1*1.1 = 1.1$
- This is greater than the threshold of 1, so the output = 1

matched

3. A=1, B=0 and Target = 1

- $wi.xi = 1*0.6 + 0*1.1 = 0.6$
- This is not greater than the threshold of 1, so the output = 0

Not matched

$$wi = wi + n(t - o)xi$$

$$w1 = 0.6 + 0.5(1 - 0)1 = 1.1$$

$$w2 = 1.1 + 0.5(1 - 0)0 = 1.1$$

Modified weight values

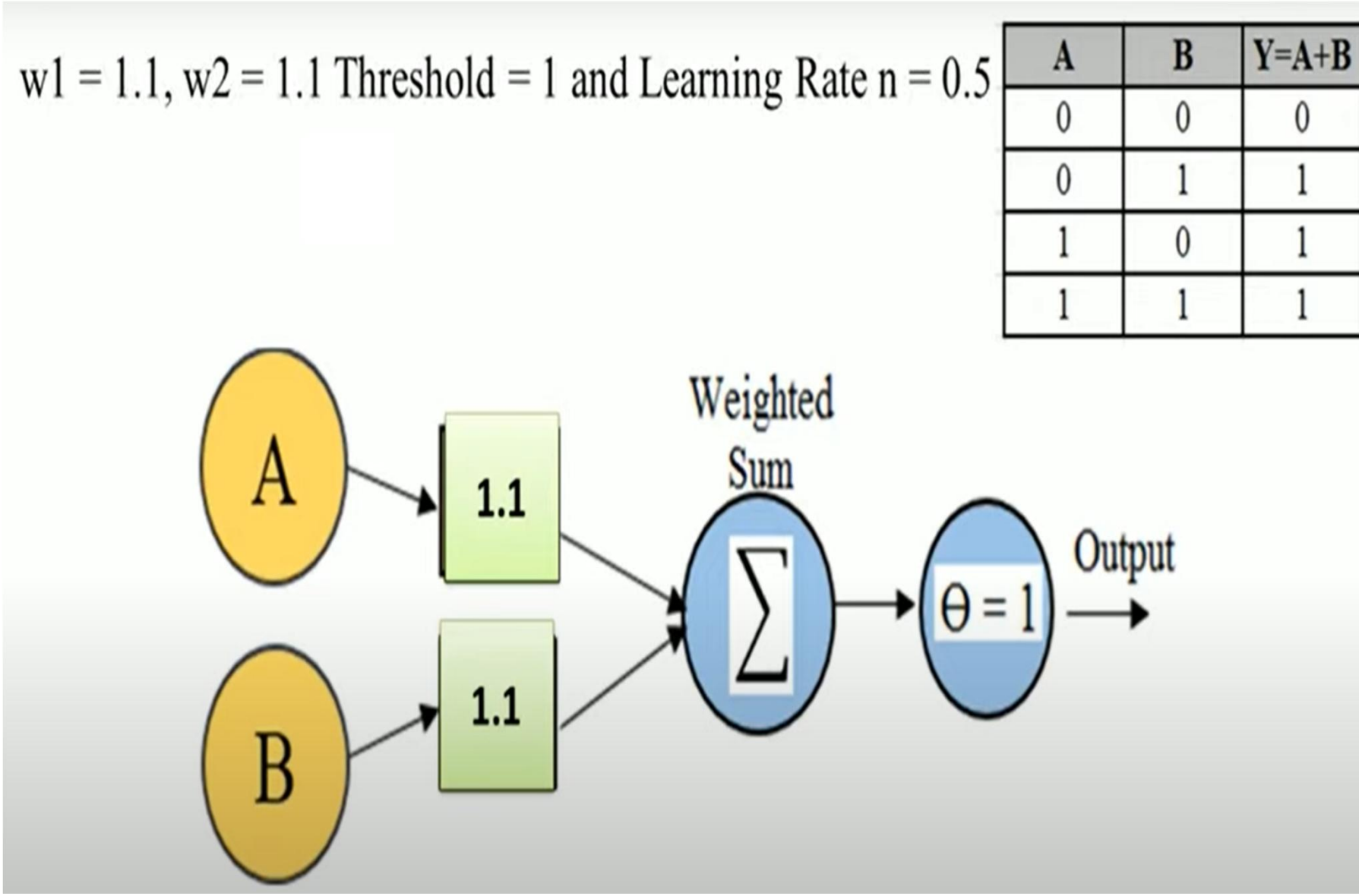
Apply the Modified weight values and check with all 4 cases

1. A=0, B=0 and Target = 0
   - $wi.xi = 0*1.1 + 0*1.1 = 0$
   - This is not greater than the threshold of 1, so the output = 0
2. A=0, B=1 and Target = 1
   - $wi.xi = 0*1.1 + 1*1.1 = 1.1$
   - This is greater than the threshold of 1, so the output = 1
3. A=1, B=0 and Target = 1
   - $wi.xi = 1*1.1 + 0*1.1 = 1.1$
   - This is greater than the threshold of 1, so the output = 1
4. A=1, B=1 and Target = 1
   - $wi.xi = 1*1.1 + 1*1.1 = 2.2$
   - This is greater than the threshold of 1, so the output = 1

Perceptron network – OR gate

w1 = 1.1, w2 = 1.1 Threshold = 1 and Learning Rate n = 0.5

A	B	Y=A+B
0	0	0
0	1	1
1	0	1
1	1	1



Homework

- Design a AND gate using single layer perceptron network.
- Assign random weight, T=1, n= 0.5

Limitations of Linear Nets and Perceptrons (1/2)

- **Linear separability**: The perceptron can only solve problems that are linearly separable (i.e., it can only classify data that can be separated by a straight line or hyperplane). It cannot solve more complex problems like XOR.
- **Single-layer model limitation**: The original perceptron is a single-layer model and does not have hidden layers, limiting its expressiveness.

Limitations of Linear Nets and Perceptrons (2/2)

- **Only Two-Class Output**:
  - Basic perceptrons are designed to handle binary classification. For multi-class problems, multiple perceptrons or extensions like one-vs-all must be employed, complicating the model.
- **Sensitivity to Noisy Data**:
  - Perceptrons can be sensitive to outliers and noisy data, which can have a significant impact on the learned weights, leading to poor model performance.

Multi-Layer Perceptrons (MLP)

- (MLP) overcome many limitations of basic perceptrons by introducing one or more hidden layers that allow for the modeling of more complex function mappings.

Architecture of MLP

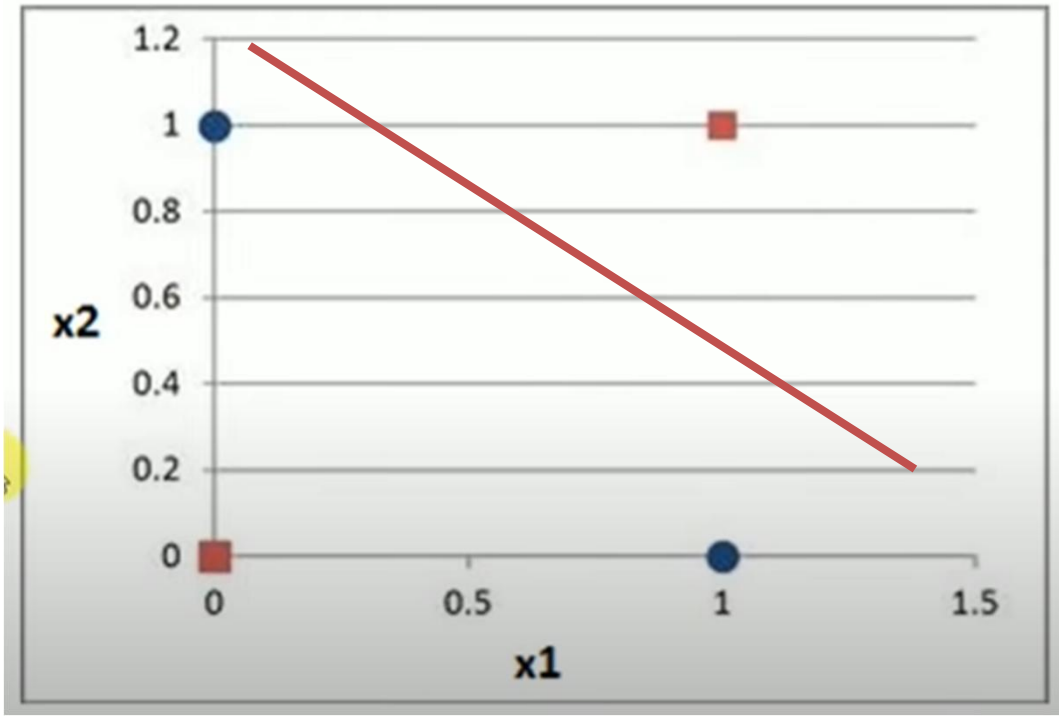
- **Input Layer**: Same as a perceptron, receives input features.
- **Hidden Layers**: One or more layers where each neuron is connected to the previous layer. Activation functions (e.g., ReLU, sigmoid, tanh) introduce non-linearity, allowing the network to model complex relationships.
- **Output Layer**: Provides predictions, which can accommodate multiple classes for multi-class classification tasks.

Comparison Table

Aspect	Single-Layer Perceptron (SLP)	Multi-Layer Perceptron (MLP)
Architecture	One input layer, one output layer (no hidden layers)	Input, hidden, and output layers
Problem Solvability	Solves only linearly separable problems (e.g., AND)	Solves both linear and non-linear problems (e.g., XOR)
Activation Function	Step function (binary output)	Non-linear functions (e.g., sigmoid, ReLU, tanh)
Learning Algorithm	Perceptron learning rule (no backpropagation)	Trained using backpropagation and gradient descent
Output	Binary (0 or 1)	Continuous (for regression) or multi-class (for classification)
Applications	Simple binary classification	Complex tasks (classification, regression, image recognition)
Complexity	Simple and limited	Complex and powerful

Eg. MLP: Designing XOR gate

Consider the truth table for XOR function



x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Not linearly separable.XOR is represented as

$$y = x_1\overline{x_2} + \overline{x_1}x_2$$

$$y = x_1\overline{x_2} + \overline{x_1}x_2$$

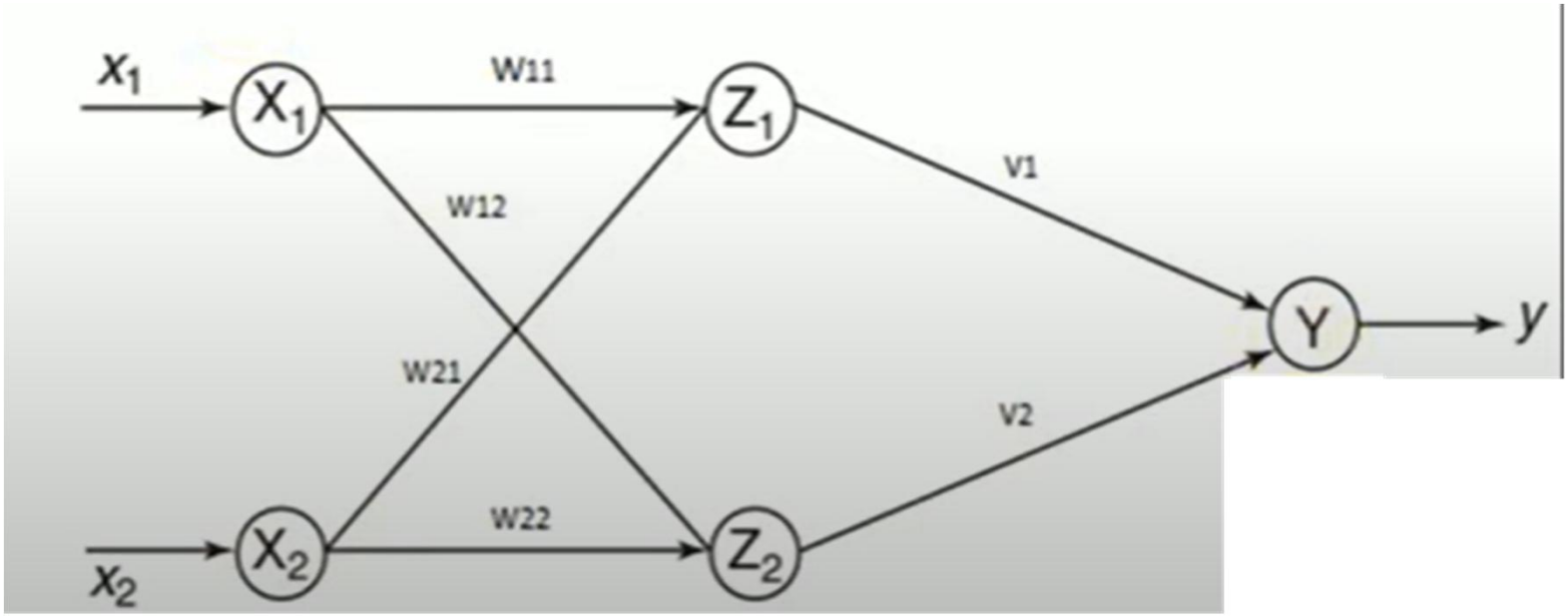
$$y = z_1 + z_2$$

where

$$z_1 = x_1\overline{x_2} \qquad \text{(function 1)}$$

$$z_2 = \overline{x_1}x_2 \qquad \text{(function 2)}$$

$$y = z_1 \text{ (OR) } z_2 \qquad \text{(function 3)}$$



- First function $z_1 = x_1 \bar{x}_2$
- Assume the Initial weights are $w_{11} = w_{21} = 1$
- Threshold = 1 and learning rate = 1.5
- $(0,0) \rightarrow Z_{1in} = w_{ij} * x_i = 1 * 0 + 1 * 0 = 0 \ (out = 0)$
- $(0,1) \rightarrow Z_{1in} = w_{ij} * x_i = 1 * 0 + 1 * 1 = 1 \ (out = 1)$
  - $w_{ij} = w_{ij} + n(t - o)x_i$
  - $w_{11} = 1 + 1.5 * (0 - 1) * 0 = 0$
  - $w_{21} = 1 + 1.5 * (0 - 1) * 1 = -0.5$

x_1	x_2	z_1
0	0	0
0	1	0
1	0	1
1	1	0

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

← Modify weight

- First function $z_1 = x_1 \overline{x_2}$
- Assume the Initial weights are $w_{11} = 1 \; and \; w_{21} = -0.5$
- Threshold = 1 and learning rate = 1.5
- $(0,0) \rightarrow Z_{1in} = w_{ij} * x_i = 1 * 0 + (-0.5) * 0 = 0 \quad (out = 0)$
- $(0,1) \rightarrow Z_{1in} = w_{ij} * x_i = 1 * 0 + (-0.5) * 1 = -0.5 \; (out = 0)$
- $(1,0) \rightarrow Z_{1in} = w_{ij} * x_i = 1 * 1 + (-0.5) * 0 = 1 \quad (out = 1)$
- $(1,1) \rightarrow Z_{1in} = w_{ij} * x_i = 1 * 1 + (-0.5) * 1 = 0.5 \quad (out = 0)$

Next Z2

$$z_2 = \overline{x_1} x_2$$

x_1	x_2	z_2
0	0	0
0	1	1
1	0	0
1	1	0

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

First function $z_2 = \overline{x_1} x_2$

Assume the Initial weights are $w_{12} = w_{22} = 1$

Threshold = 1 and learning rate = 1.5

$(0,0) \rightarrow Z_{2in} = w_{ij} * x_i = 1 * 0 + 1 * 0 = 0 \quad (out = 0)$

$(0,1) \rightarrow Z_{2in} = w_{ij} * x_i = 1 * 0 + 1 * 1 = 1 \quad (out = 1)$

$(1,0) \rightarrow Z_{2in} = w_{ij} * x_i = 1 * 1 + 1 * 0 = 1 \quad (out = 1)$

$- w_{ij} = w_{ij} + n(t - o)x_i$

$- w_{12} = 1 + 1.5 * (0 - 1) * 1 = -0.5$

$- w_{22} = 1 + 1.5 * (0 - 1) * 0 = 1$

$$z_2 = \overline{x_1} x_2$$

- Assume the Initial weights are $w_{12} = -0.5$ *and* $w_{22} = 1$
- Threshold = 1 and learning rate = 1.5
- $(0,0) \rightarrow Z_{2in} = w_{ij} * x_i = (-0.5) * 0 + 1 * 0 = 0 \quad (out = 0)$

(0,1) -> What is the answer?

(1,0) -> What is the answer?

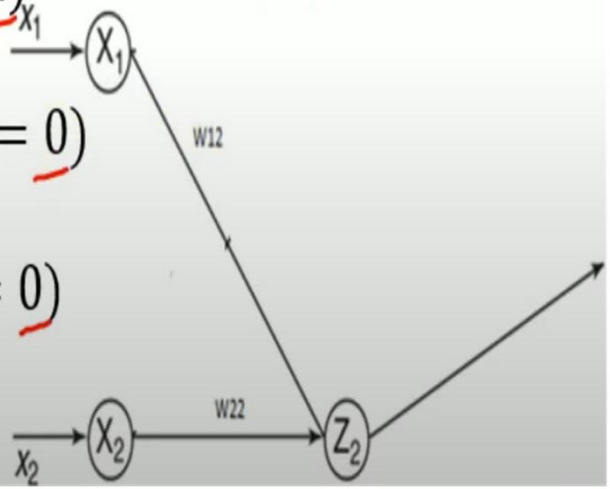
(1,1) -> What is the answer?

$$z_2 = \overline{x_1} x_2$$

- Assume the Initial weights are $w_{12} = -0.5$ *and* $w_{22} = 1$
- Threshold = 1 and learning rate = 1.5
- $(0,0) \rightarrow Z_{2in} = w_{ij} * x_i = (-0.5) * 0 + 1 * 0 = 0 \quad (out = 0)$
- $(0,1) \rightarrow Z_{2in} = w_{ij} * x_i = (-0.5) * 0 + 1 * 1 = 1 \quad (out = 1)$
- $(1,0) \rightarrow Z_{2in} = w_{ij} * x_i = (-0.5) * 1 + 1 * 0 = -0.5 \quad (out = 0)$
- $(1,1) \rightarrow Z_{2in} = w_{ij} * x_i = (-0.5) * 1 + 1 * 1 = 0.5 \quad (out = 0)$

X_1	X_2	Z_2
0	0	0
0	1	1
1	0	0
1	1	0

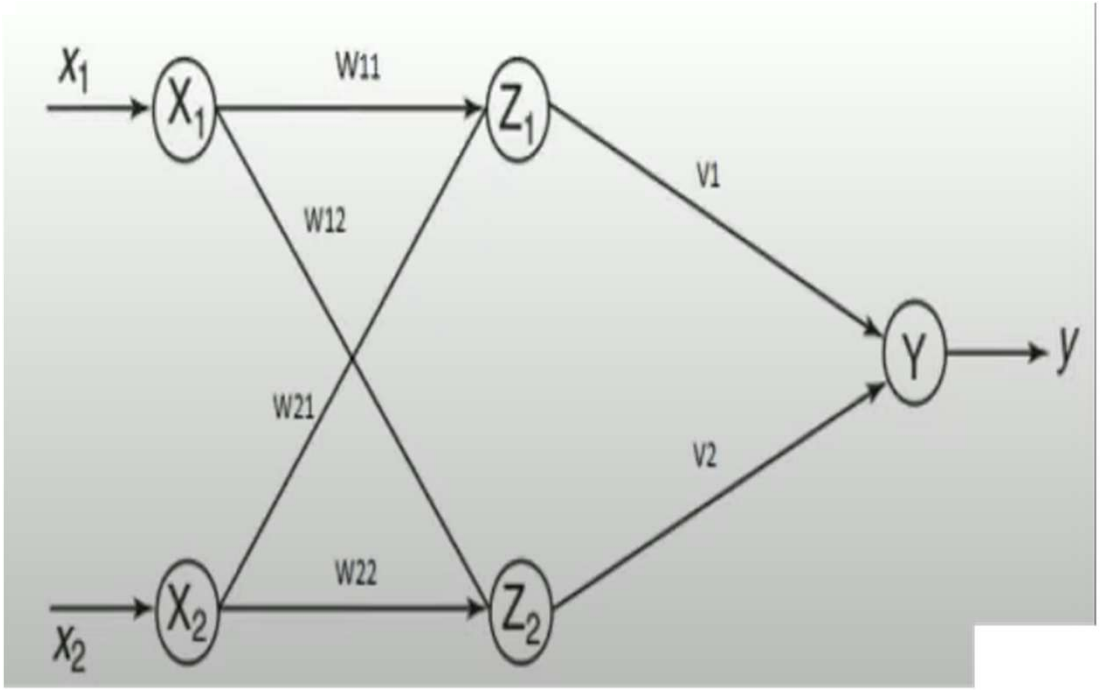
$$f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} \geq \theta \\ 0 & \text{if } y_{in} < \theta \end{cases}$$



Now, will continue with final function

$$y = z_1 \,(\mathrm{OR})\, z_2 \qquad y_{in} = z_1 v_1 + z_2 v_2$$

x_1	x_2	z_1	z_2	y
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	0	0	0



Apply the values and find Weighted Sum for “y”

Assume the initial weights are $v_1 = v_2 = 1$

Threshold = 1 and learning rate = 1.5

$$(0,0) \rightarrow y_{in} = v_i * x_i = 1 * 0 + 1 * 0 = 0 \quad (out = 0)$$

$$(0,1) \rightarrow y_{in} = v_i * x_i = 1 * 0 + 1 * 1 = 1 \quad (out = 1)$$

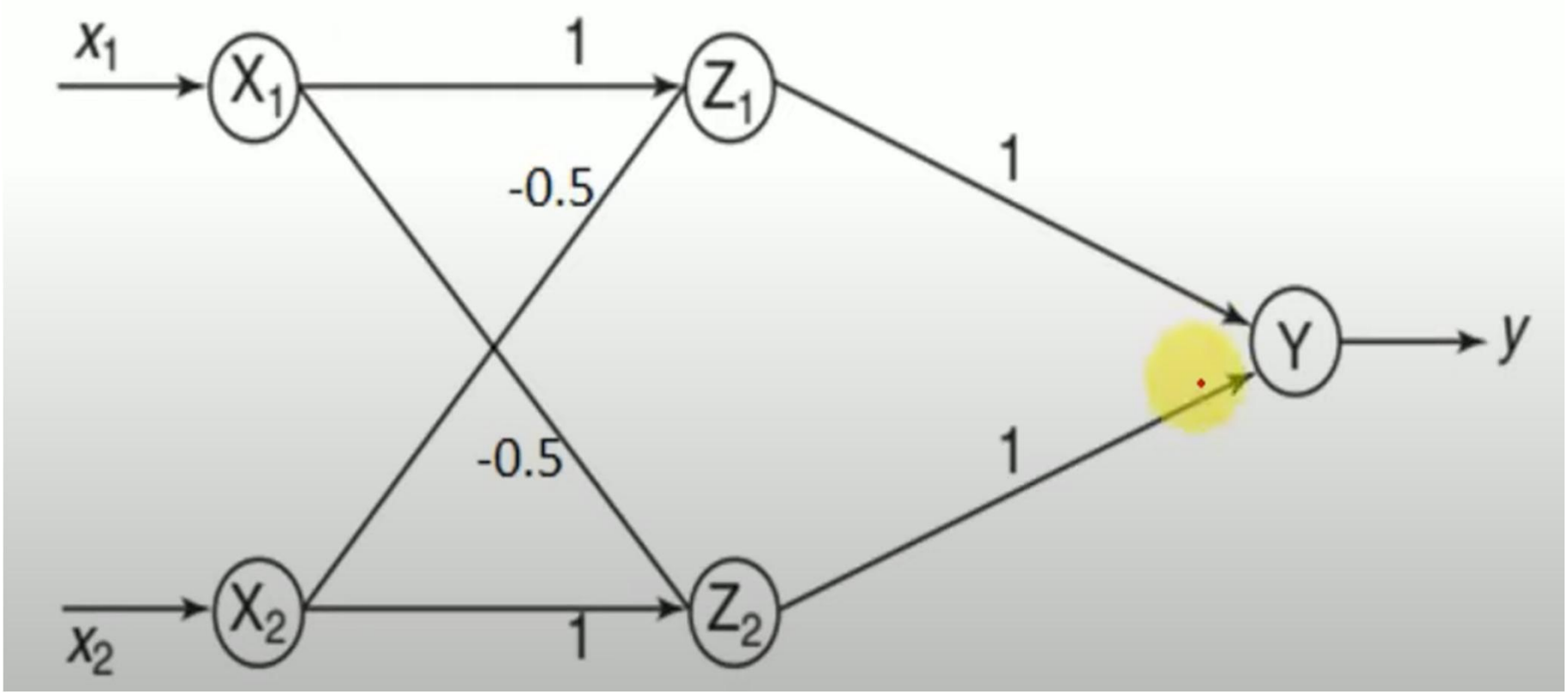
$$(1,0) \rightarrow y_{in} = v_i * x_i = 1 * 1 + 1 * 0 = 1 \quad (out = 1)$$

$$(0,0) \rightarrow y_{in} = v_i * x_i = 1 * 0 + 1 * 0 = 0 \quad (out = 0)$$

w11 = 1 and w21 = -0.5

w12 = -0.5 and w22 = 1

v1 = v2 = 1



Homework

- Construct MLP network for XAND Gate.

With initial weight values 1 , T= 1 and n=1.5

A	B	XAND Output
0	0	1
0	1	1
1	0	1
1	1	0