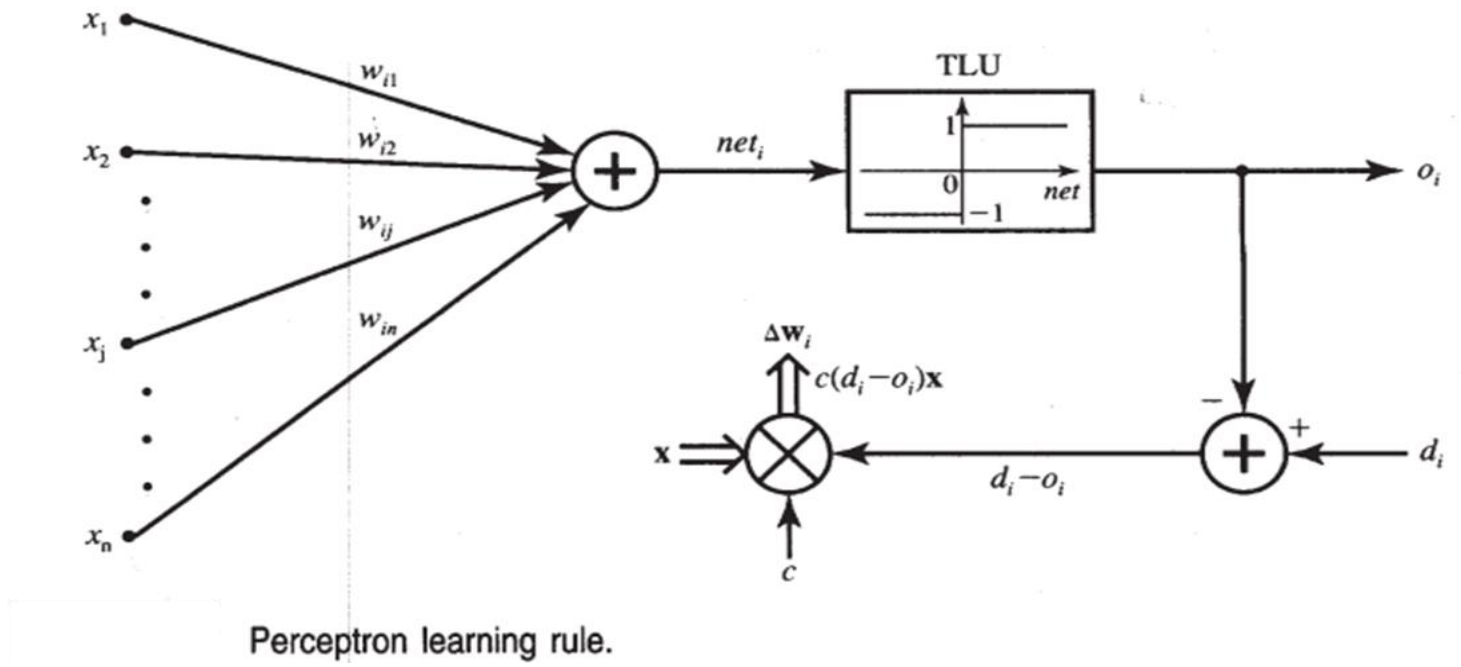
Perceptron Learning Rule

Perceptron Learning Rule



Perceptron learning rule.

A neuron with 3 inputs has the weight vector w=[0.1,0.3,−0.2] The input vector is x=[0.8,0.6,0.4]. The activation function is binary sigmoidal activation. Find the output of the neuron.

$$\text{Binary sigmoid output: } O = \frac{1}{1+e^{-z}}$$

(0.18, 0.5448788923735801)

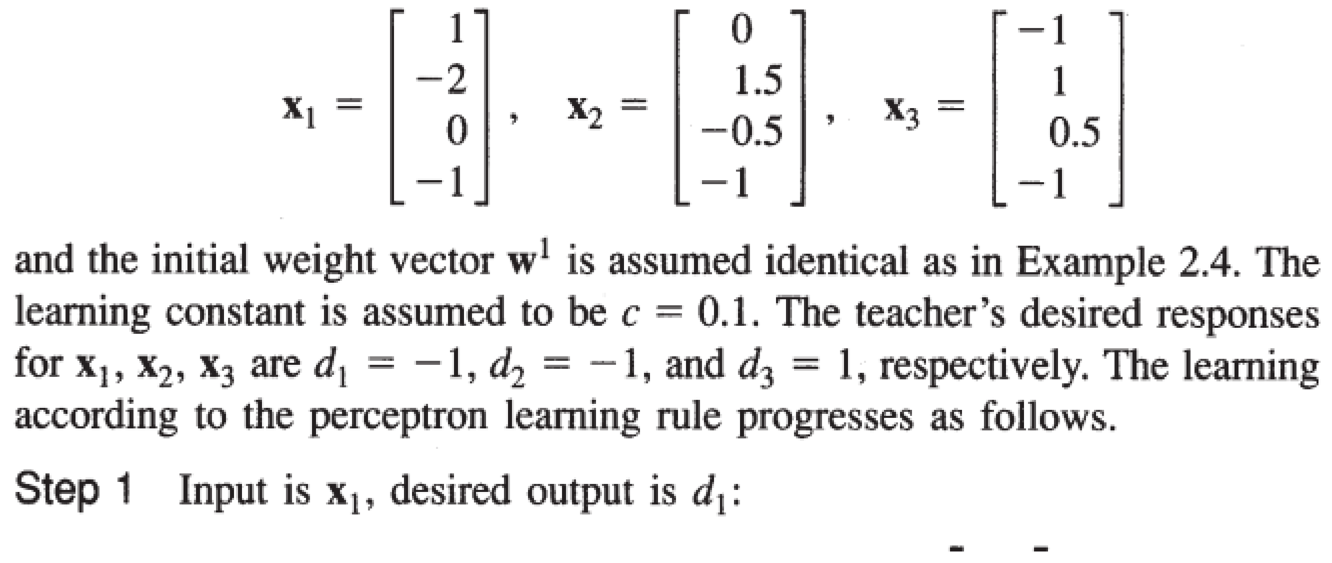
Perceptron Learning Rule

```
# Given values
weights = [0.1, 0.3, -0.2]
inputs = [0.8, 0.6, 0.4]

# Step 1: Compute the weighted sum (z)

# Step 2: Apply the binary sigmoid activation function
```

Result

Z= 0.18, O=0.544)

Perceptron learning Rule



$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ -2 \\ 0 \\ -1 \end{bmatrix}, \quad \mathbf{x}_2 = \begin{bmatrix} 0 \\ 1.5 \\ -0.5 \\ -1 \end{bmatrix}, \quad \mathbf{x}_3 = \begin{bmatrix} -1 \\ 1 \\ 0.5 \\ -1 \end{bmatrix}$$

and the initial weight vector $\mathbf{w}^1$ is assumed identical as in Example 2.4. The learning constant is assumed to be $c = 0.1$. The teacher's desired responses for $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$ are $d_1 = -1$, $d_2 = -1$, and $d_3 = 1$, respectively. The learning according to the perceptron learning rule progresses as follows.

Step 1 Input is $\mathbf{x}_1$, desired output is d_1:

Sums on delta rule

- **Learning rate (η)**: 0.1
- **Initial weights (w1,w2)**: [0.5, -0.5]
- **Input vector (x)**: [1, 2]
- **Bias (b)**: 0.1
- **Desired output (d)**: 0.8
- **Activation function**: Linear (y=net}

Sums on delta rule

$$\text{net}_1 = (0.5 \cdot 1) + (-0.5 \cdot 2) + 0.1 = 0.5 - 1.0 + 0.1 = -0.4$$

2. **Output** (y_1):

$$y_1 = \text{net}_1 = -0.4$$

3. **Error** ($d - y_1$):

$$e_1 = 0.8 - (-0.4) = 1.2$$

4. **Update weights**:

$$\Delta w_1 = 0.1 \cdot 1.2 \cdot 1 = 0.12, \quad \Delta w_2 = 0.1 \cdot 1.2 \cdot 2 = 0.24$$

   New weights:

$$w_1 = 0.5 + 0.12 = 0.62, \quad w_2 = -0.5 + 0.24 = -0.26$$

5. **Update bias**:

$$\Delta b = 0.1 \cdot 1.2 = 0.12$$

   New bias:

$$b = 0.1 + 0.12 = 0.22$$

$$\text{net}_2 = (0.62 \cdot 1) + (-0.26 \cdot 2) + 0.22 = 0.62 - 0.52 + 0.22 = 0.32$$

2. **Output** (y_2):

$$y_2 = \text{net}_2 = 0.32$$

3. **Error** ($d - y_2$):

$$e_2 = 0.8 - 0.32 = 0.48$$

4. **Update weights**:

$$\Delta w_1 = 0.1 \cdot 0.48 \cdot 1 = 0.048, \quad \Delta w_2 = 0.1 \cdot 0.48 \cdot 2 = 0.096$$

   New weights:

$$w_1 = 0.62 + 0.048 = 0.668, \quad w_2 = -0.26 + 0.096 = -0.164$$

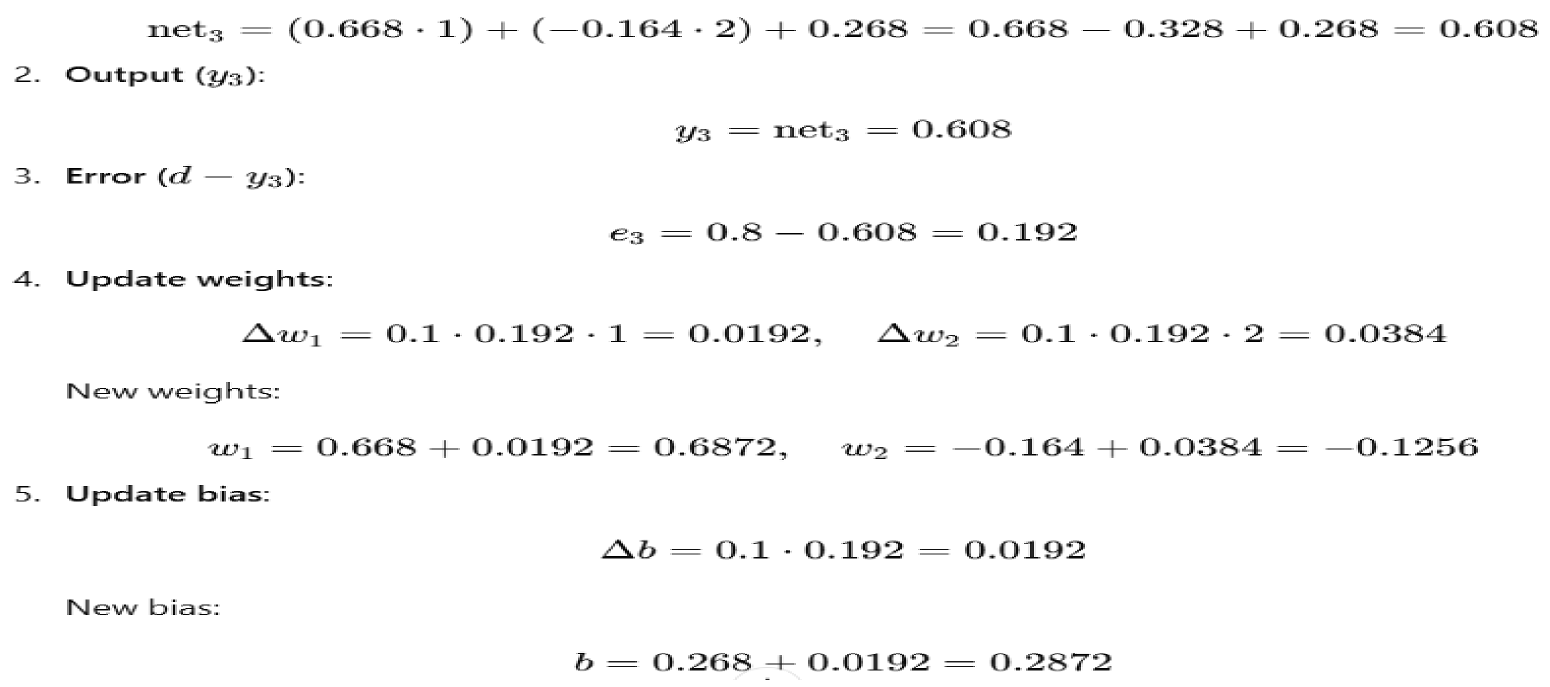
5. **Update bias**:

$$\Delta b = 0.1 \cdot 0.48 = 0.048$$

   New bias:

$$b = 0.22 + 0.048 = 0.268$$

Sums on delta rule



$$\text{net}_3 = (0.668 \cdot 1) + (-0.164 \cdot 2) + 0.268 = 0.668 - 0.328 + 0.268 = 0.608$$

2. **Output (y_3):**

$$y_3 = \text{net}_3 = 0.608$$

3. **Error ($d - y_3$):**

$$e_3 = 0.8 - 0.608 = 0.192$$

4. **Update weights:**

$$\Delta w_1 = 0.1 \cdot 0.192 \cdot 1 = 0.0192, \quad \Delta w_2 = 0.1 \cdot 0.192 \cdot 2 = 0.0384$$

New weights:

$$w_1 = 0.668 + 0.0192 = 0.6872, \quad w_2 = -0.164 + 0.0384 = -0.1256$$

5. **Update bias:**

$$\Delta b = 0.1 \cdot 0.192 = 0.0192$$

New bias:

$$b = 0.268 + 0.0192 = 0.2872$$

Final Weights and Bias:

•w1=0.6872,w2=−0.1256,b=0.287

•After 3 iterations, the output (y) converges closer to the desired output (d).
•Further iterations will reduce the error further.

Three Classes of Deep Learning

- **Generative deep architectures**

It is a class of deep learning models designed to generate new data samples that resemble the training data. These architectures model the underlying data distribution and can create realistic data such as images, text, audio, or other complex data types.

- **Discriminative deep architectures**

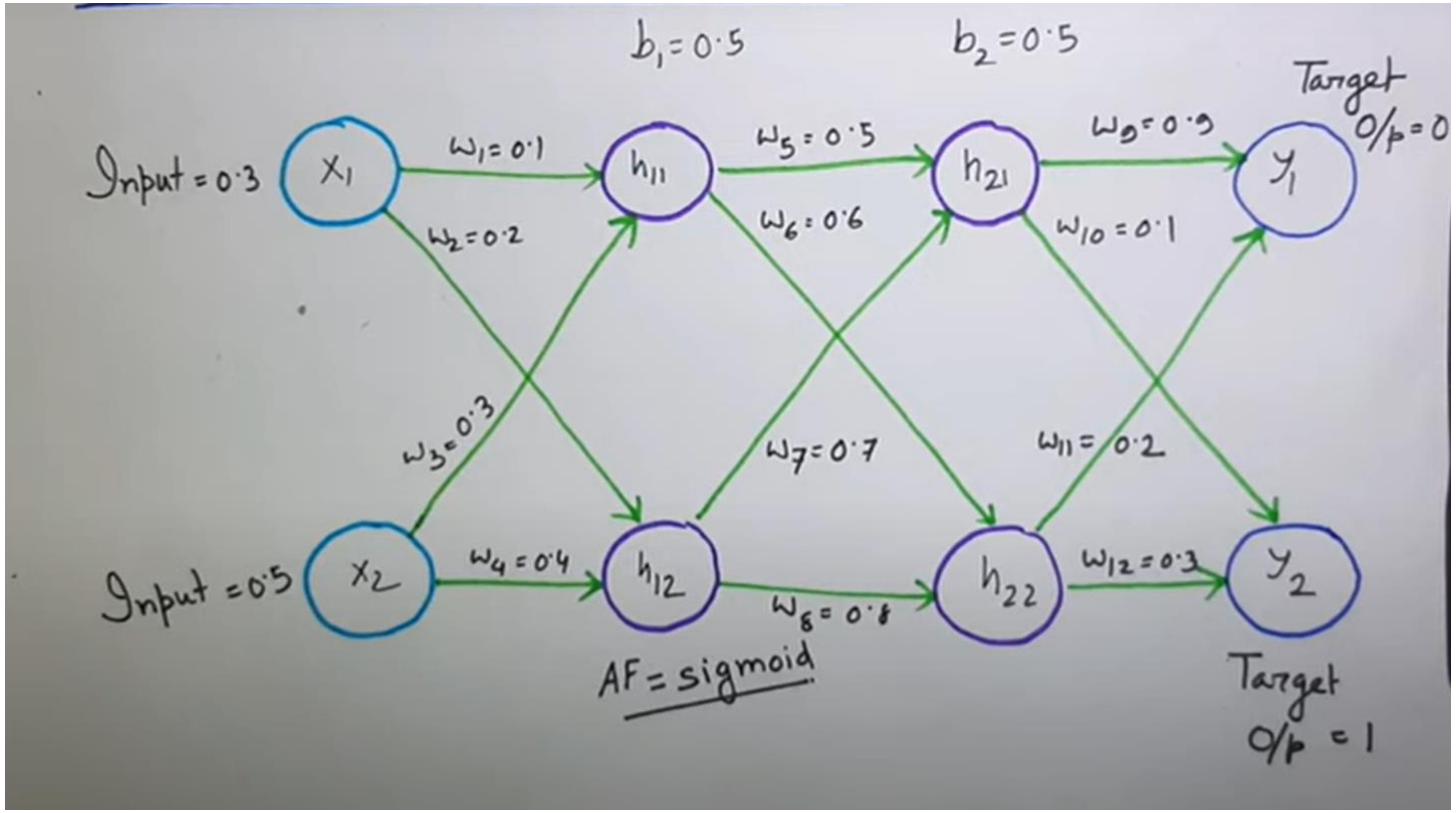
class of deep learning models designed to classify data or make predictions by learning the decision boundaries between different classes.

- **Hybrid architectures:**

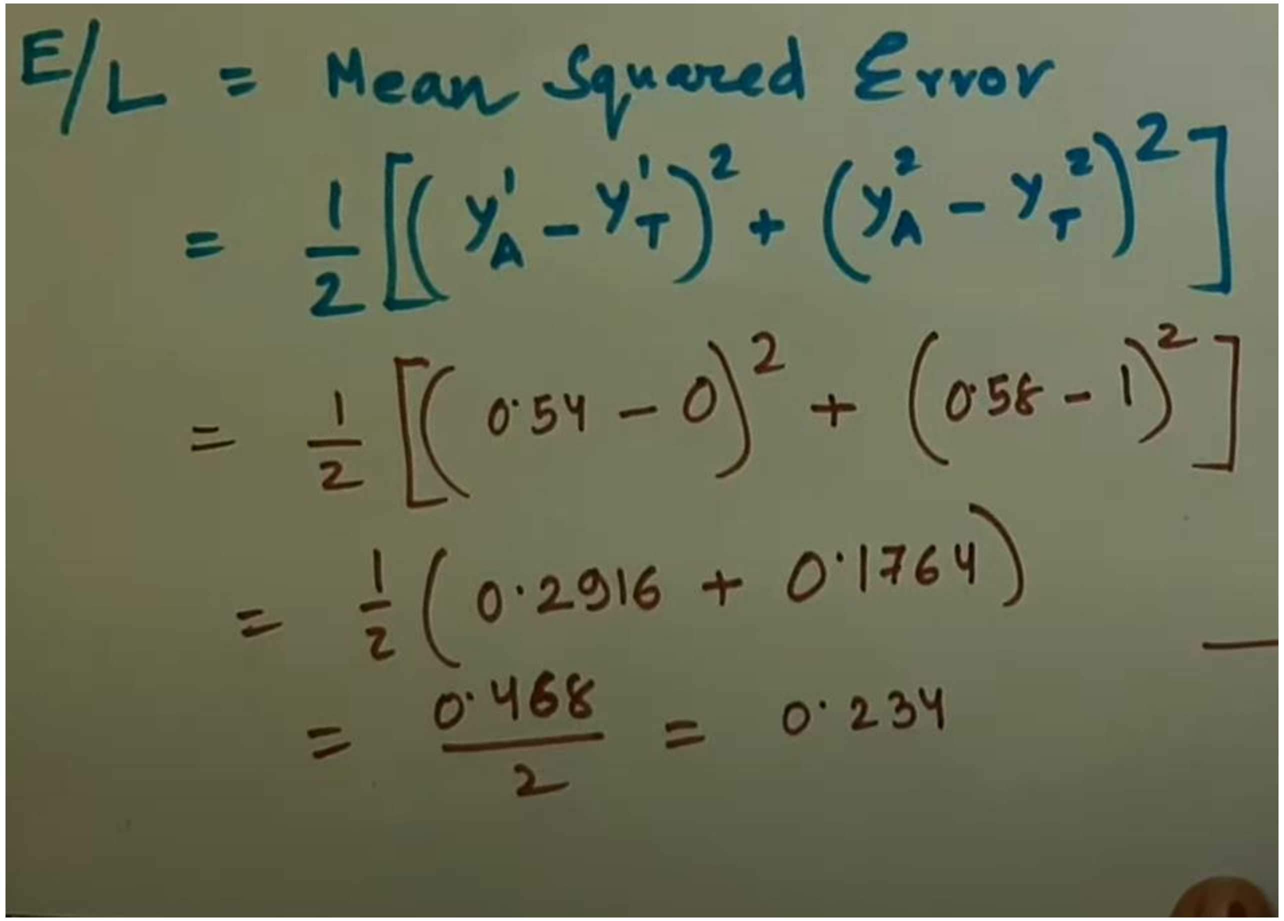
Combine **generative** and **discriminative** models to leverage the strengths of both. These architectures aim to balance the generative model's ability to model data distributions and the discriminative model's ability to classify or predict effectively.

eg. SGAN (Semi-Supervised Learning with GANs)

Feed-Forward Network



Feed-Forward Network



E/L = Mean Squared Error

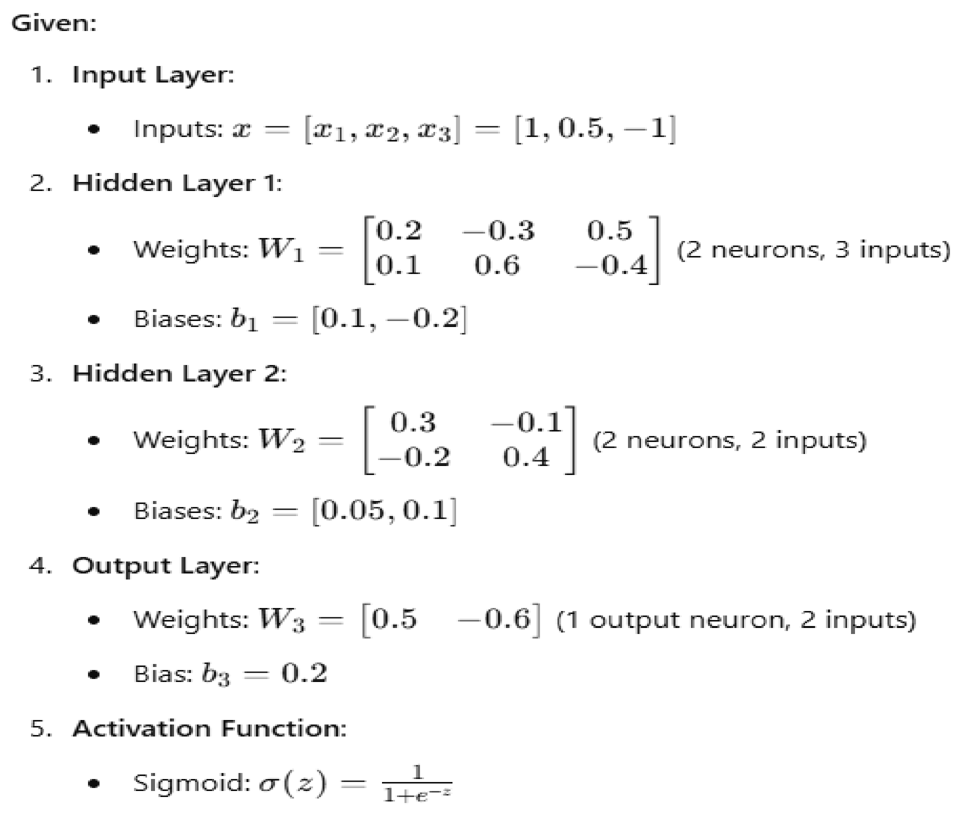
$$= \frac{1}{2}\left[\left(y_A^1 - y_T^1\right)^2 + \left(y_A^2 - y_T^2\right)^2\right]$$

$$= \frac{1}{2}\left[\left(0.54 - 0\right)^2 + \left(0.58 - 1\right)^2\right]$$

$$= \frac{1}{2}\left(0.2916 + 0.1764\right)$$

$$= \frac{0.468}{2} = 0.234$$

Feed Forward Network



Given:

1. **Input Layer:**
   - Inputs: $x = [x_1, x_2, x_3] = [1, 0.5, -1]$
2. **Hidden Layer 1:**
   - Weights: $W_1 = \begin{bmatrix} 0.2 & -0.3 & 0.5 \\ 0.1 & 0.6 & -0.4 \end{bmatrix}$ (2 neurons, 3 inputs)
   - Biases: $b_1 = [0.1, -0.2]$
3. **Hidden Layer 2:**
   - Weights: $W_2 = \begin{bmatrix} 0.3 & -0.1 \\ -0.2 & 0.4 \end{bmatrix}$ (2 neurons, 2 inputs)
   - Biases: $b_2 = [0.05, 0.1]$
4. **Output Layer:**
   - Weights: $W_3 = \begin{bmatrix} 0.5 & -0.6 \end{bmatrix}$ (1 output neuron, 2 inputs)
   - Bias: $b_3 = 0.2$
5. **Activation Function:**
   - Sigmoid: $\sigma(z) = \frac{1}{1+e^{-z}}$