Lecture-2

Perceptron

A **perceptron** is the simplest kind of artificial neural network model, introduced by Frank Rosenblatt (1958). It's basically a linear classifier: it takes inputs, applies weights, sums them up, passes through an activation function, and produces an output (decision).

Mathematical Formulation

Suppose we have an input vector:

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]$$

and corresponding weights:

$$\mathbf{w} = [w_1, w_2, \ldots, w_n]$$

plus a bias term b.

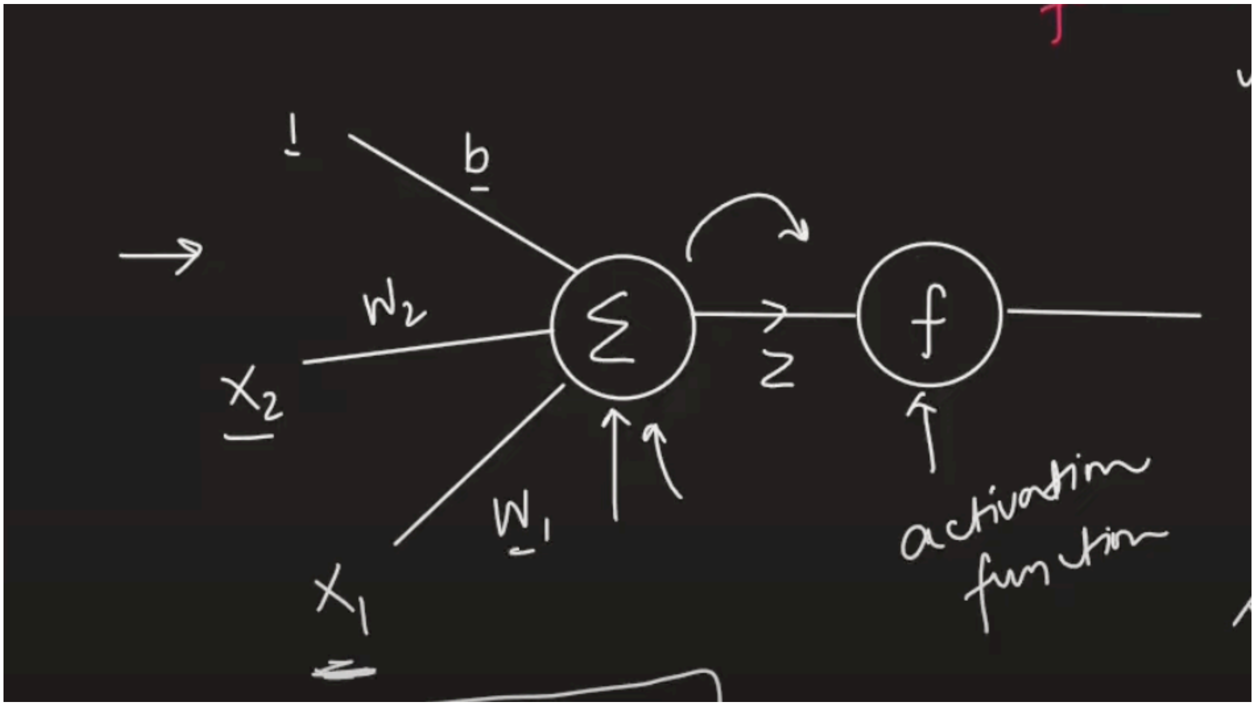
The perceptron computes a **weighted sum**:

$$z = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^{n} w_i x_i + b$$

Then it applies an **activation function** (in the original perceptron, a step function):

$$y = f(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

So the perceptron makes a binary decision: output is either 1 (positive class) or 0 (negative class).



Geometric Meaning

- The decision boundary is the hyperplane defined by:

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

- The perceptron classifies input vectors into two classes depending on which side of the hyperplane they fall.

Learning Rule (Weight Update)

The perceptron updates its weights when it makes a mistake:

$$w_i^{(new)} = w_i^{(old)} + \eta(y_{true} - y_{pred})x_i$$

$$b^{(new)} = b^{(old)} + \eta(y_{true} - y_{pred})$$

where:

- η = learning rate
- y_{true} = actual label
- y_{pred} = perceptron output

A perceptron is a **linear classifier** defined by y=f(w·x+b)y = f(\mathbf{w}\cdot\mathbf{x} + b)y=f(w·x+b), where fff is usually a step function.

Example: AND Gate with a Perceptron

We want a perceptron to learn the AND logic gate.
The truth table is:

x_1	x_2	**Output** (y)
0	0	0
0	1	0
1	0	0
1	1	1

Step 1: Initialize

- Inputs: $\mathbf{x} = [x_1, x_2]$
- Weights: $\mathbf{w} = [0, 0]$ (start at 0)
- Bias: $b = 0$
- Learning rate: $\eta = 1$

Step 2: Perceptron Rule

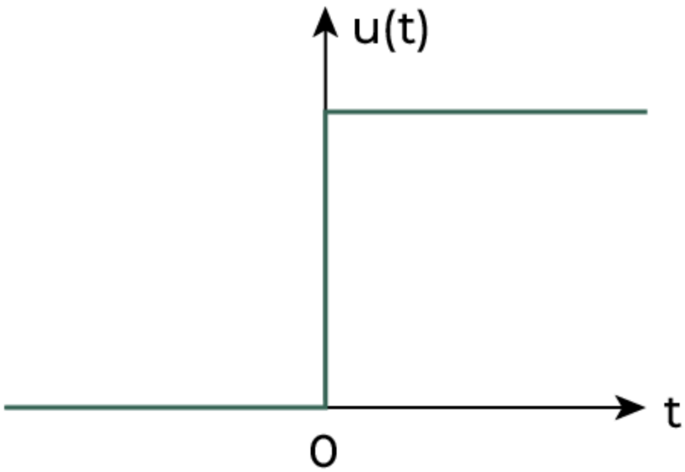
$$z = w_1x_1 + w_2x_2 + b$$

$$y = f(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}$$

Update rule if misclassified:

$$w_i \leftarrow w_i + \eta(y_{true} - y_{pred})x_i$$

$$b \leftarrow b + \eta(y_{true} - y_{pred})$$



$$u(t) = \begin{cases} 1 & t > 0 \\ 0 & t > 0 \end{cases}$$

Epoch 1 (first pass through all data)

Example 1: (0,0), $y_{true} = 0$

- $z = 0 \cdot 0 + 0 \cdot 0 + 0 = 0$ → predict 1 (since $z \geq 0$) ❌ wrong
- Update:
  - $w_1 = 0 + (0 - 1) \cdot 0 = 0$
  - $w_2 = 0 + (0 - 1) \cdot 0 = 0$
  - $b = 0 + (0 - 1) = -1$
- New parameters: $w_1 = 0, w_2 = 0, b = -1$

Example 2: (0,1), $y_{true} = 0$

- $z = 0 \cdot 0 + 0 \cdot 1 - 1 = -1$ → predict 0 ✅ correct
- No update
- Parameters unchanged: $w_1 = 0, w_2 = 0, b = -1$

Example 3: (1,0), $y_{true} = 0$

- $z = 0 \cdot 1 + 0 \cdot 0 - 1 = -1$ → predict 0 ✅ correct
- No update
- Parameters unchanged: $w_1 = 0, w_2 = 0, b = -1$

Example 4: (1,1), $y_{true} = 1$

- $z = 0 \cdot 1 + 0 \cdot 1 - 1 = -1$ → predict 0 ❌ wrong
- Update:
  - $w_1 = 0 + (1 - 0) \cdot 1 = 1$
  - $w_2 = 0 + (1 - 0) \cdot 1 = 1$
  - $b = -1 + (1 - 0) = 0$
- New parameters: $w_1 = 1, w_2 = 1, b = 0$

Epoch 2

Example 1: (0,0), $y_{true} = 0$

- $z = 1 \cdot 0 + 1 \cdot 0 + 0 = 0$ → predict 1 ❌ wrong
- Update:
  - $w_1 = 1 + (0 - 1) \cdot 0 = 1$
  - $w_2 = 1 + (0 - 1) \cdot 0 = 1$
  - $b = 0 + (0 - 1) = -1$
- Parameters: $w_1 = 1, w_2 = 1, b = -1$

Example 2: (0,1), $y_{true} = 0$

- $z = 1 \cdot 0 + 1 \cdot 1 - 1 = 0$ → predict 1 ❌ wrong
- Update:
  - $w_1 = 1 + (0 - 1) \cdot 0 = 1$
  - $w_2 = 1 + (0 - 1) \cdot 1 = 0$
  - $b = -1 + (0 - 1) = -2$
- Parameters: $w_1 = 1, w_2 = 0, b = -2$

Example 3: (1,0), $y_{true} = 0$

- $z = 1 \cdot 1 + 0 \cdot 0 - 2 = -1$ → predict 0 ✅ correct
- No update
- Parameters: $w_1 = 1, w_2 = 0, b = -2$

Example 4: (1,1), $y_{true} = 1$

- $z = 1 \cdot 1 + 0 \cdot 1 - 2 = -1$ → predict 0 ❌ wrong
- Update:
    - $w_1 = 1 + (1 - 0) \cdot 1 = 2$
    - $w_2 = 0 + (1 - 0) \cdot 1 = 1$
    - $b = -2 + (1 - 0) = -1$
- Parameters: $w_1 = 2, w_2 = 1, b = -1$

Epoch 3

Let's check predictions now with $w_1 = 2, w_2 = 1, b = -1$:

- (0,0): $z = -1$ → predict 0 ✅
- (0,1): $z = 0$ → predict 1 ❌ wrong (should be 0)
- (1,0): $z = 1$ → predict 1 ❌ wrong (should be 0)
- (1,1): $z = 2$ → predict 1 ✅

Still not fully correct. Updates will continue until convergence.

👉 After a few more epochs, the perceptron **converges** to something like:

$$w_1 = 1,\ w_2 = 1,\ b = -1.5$$

Final Model

Decision rule:

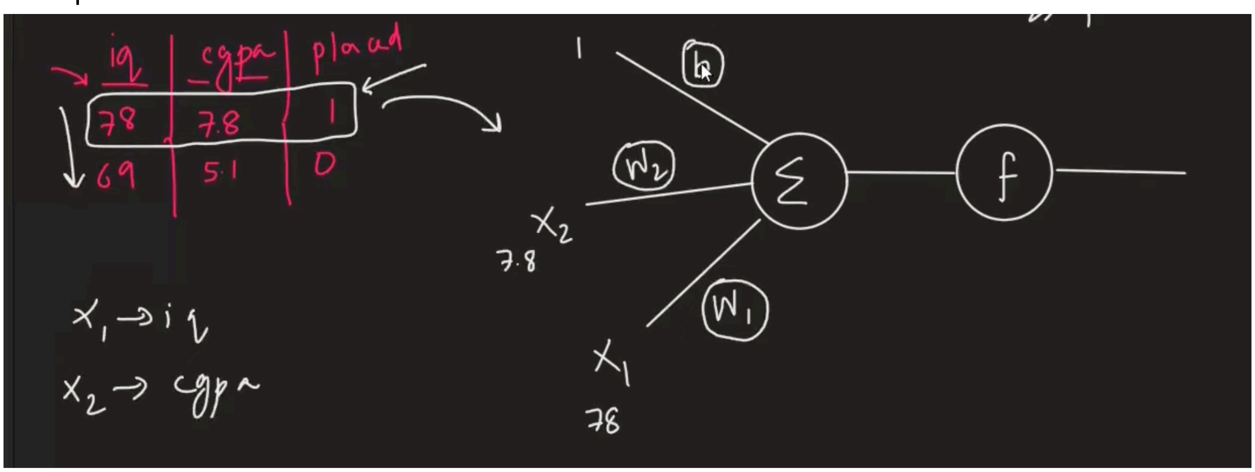
$$y = f(x_1 + x_2 - 1.5)$$

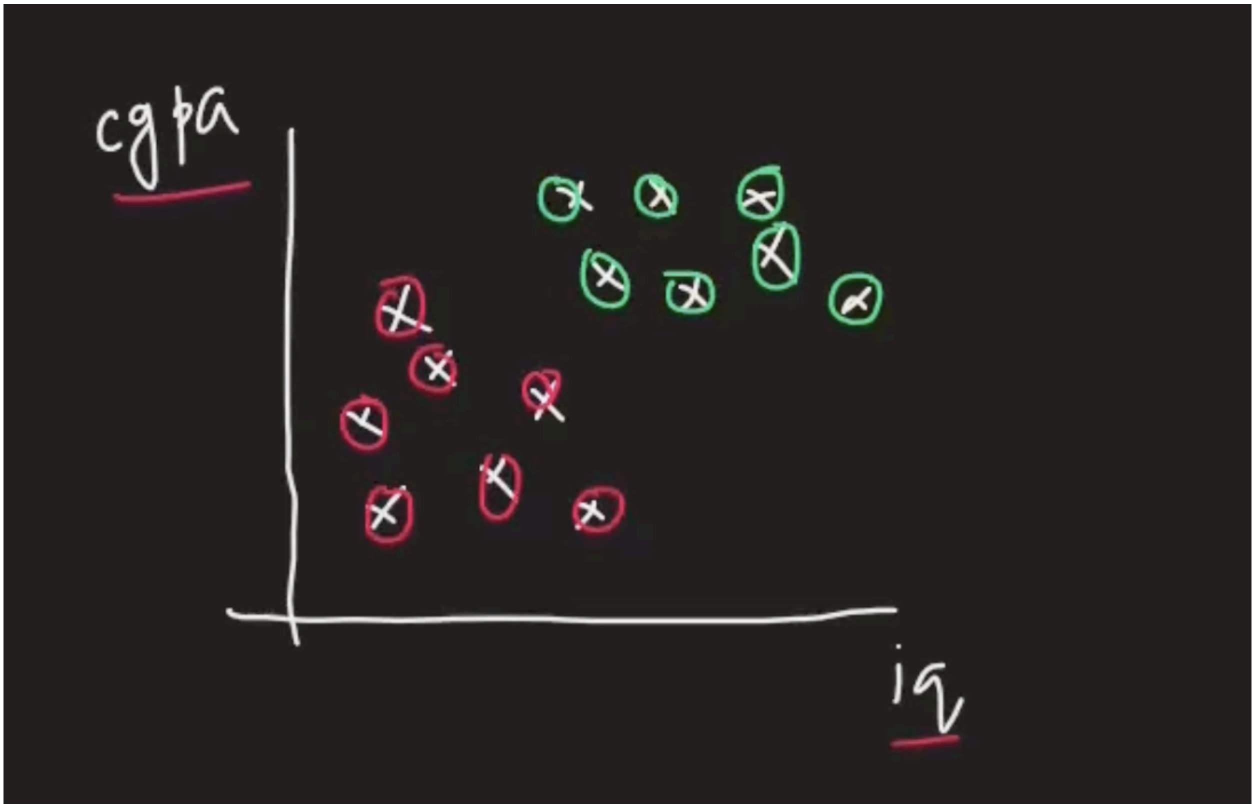
This classifies correctly:

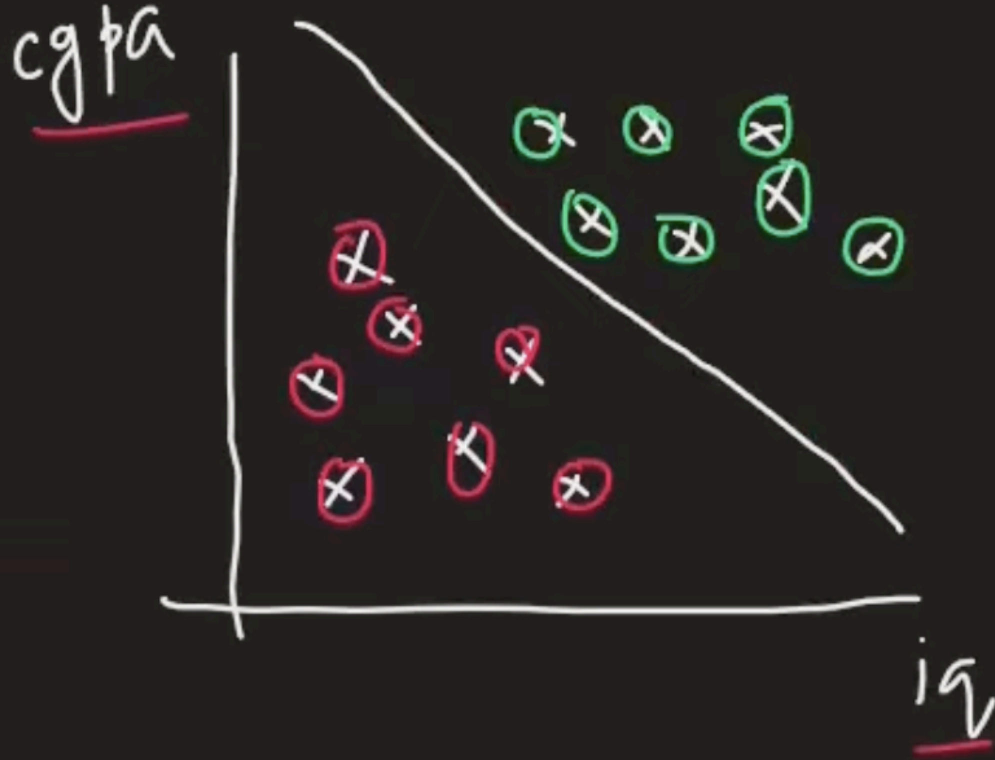
- (0,0): -1.5 → 0 ✅
- (0,1): -0.5 → 0 ✅
- (1,0): -0.5 → 0 ✅
- (1,1): 0.5 → 1 ✅

✔️ The perceptron has **learned the AND gate**.

Example




cgpa
iq



$$\boxed{Ax + By + C \geq 0}$$

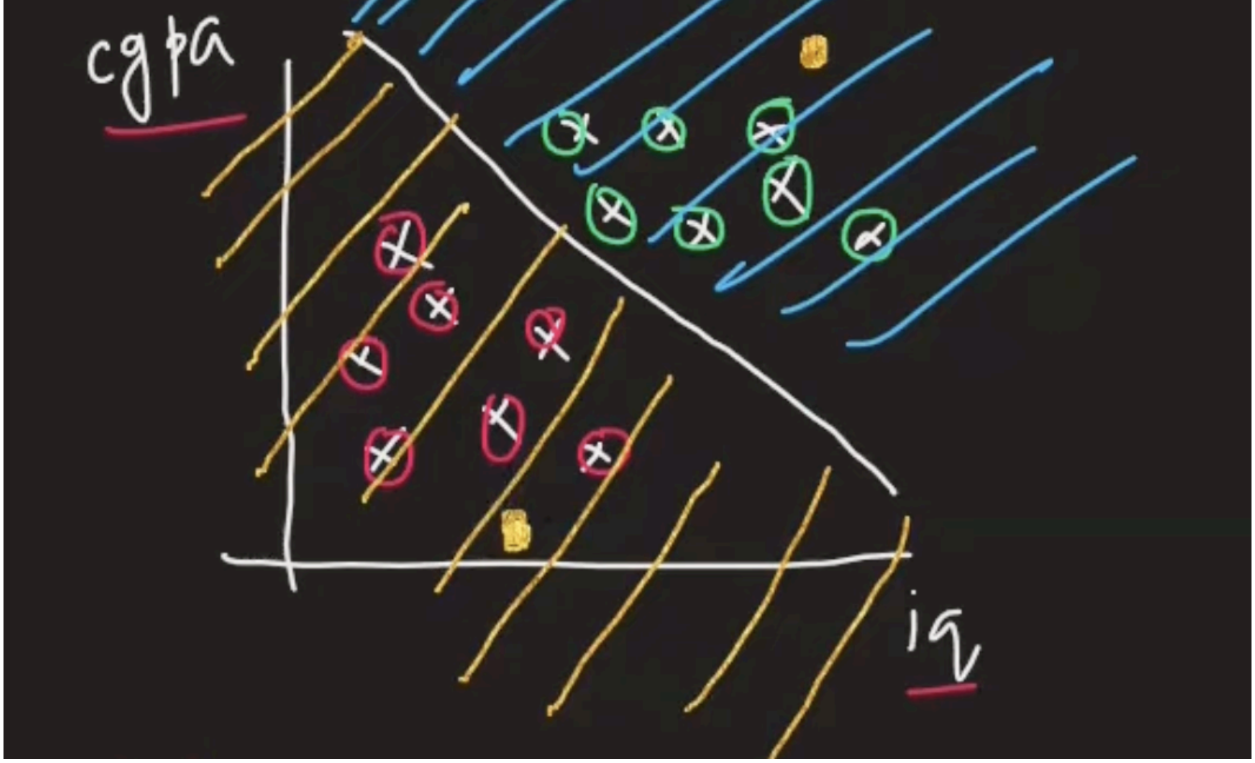
$$Ax + By + C < 0$$

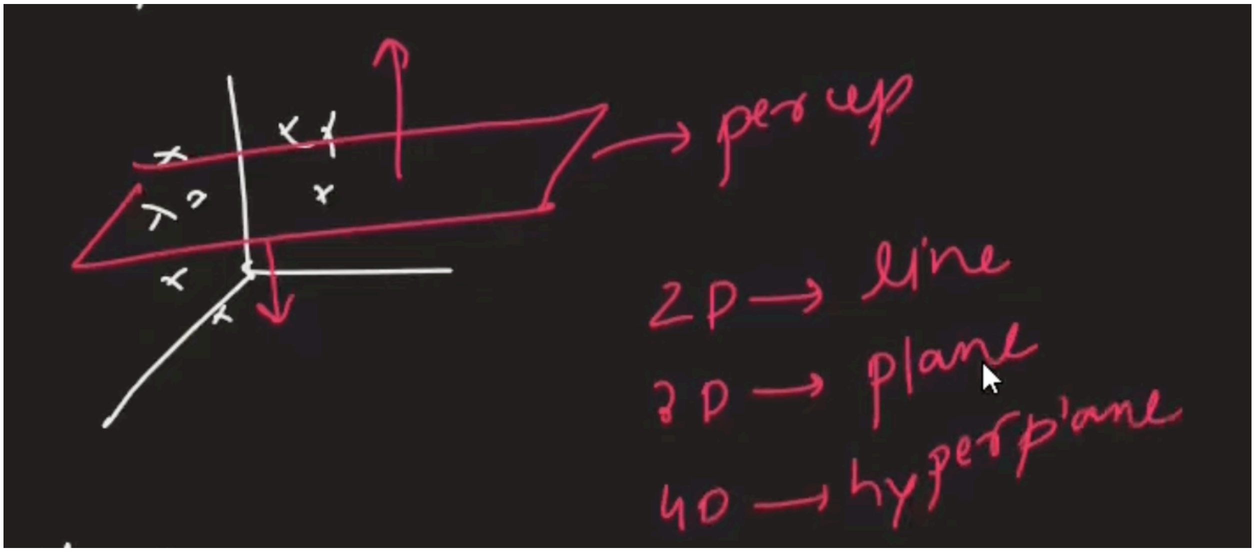
$$\boxed{Ax + By + C \geq 0}$$

$Ax + By + C < 0$ region

$Ax + By + C \geq 0 \leftarrow$ region

iq | cgpa | placed





Divide data into two classes-> binary classifier

Limitations

1. Linearly Separable Only

- A perceptron can only classify datasets that are **linearly separable**.
- Example:
  - AND, OR → can be solved ✅
  - XOR → **cannot** be solved ❌ because no straight line (hyperplane) can separate the two classes.

Mathematically, the decision boundary is always:

$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

which is just a **line (2D)**, **plane (3D)**, or **hyperplane (nD)**.

---

2. No Hidden Layers

- A single perceptron has only **input and output**, no hidden layers.
- This means it cannot model **complex functions** or **non-linear relationships**.

---

3. Step Activation Function

- The perceptron uses a **binary step function**:

$$f(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}$$

- This is **non-differentiable**, so gradient-based optimization (like backpropagation) cannot be used.
- Training is less efficient and limited.

4. Limited Output

- Outputs are **only binary (0 or 1)**.
- Cannot handle **probabilities** or **multi-class problems** easily.

---

5. Convergence Issues

- Perceptron learning rule guarantees convergence only if data is **linearly separable**.
- If not, the perceptron **never converges**.

---

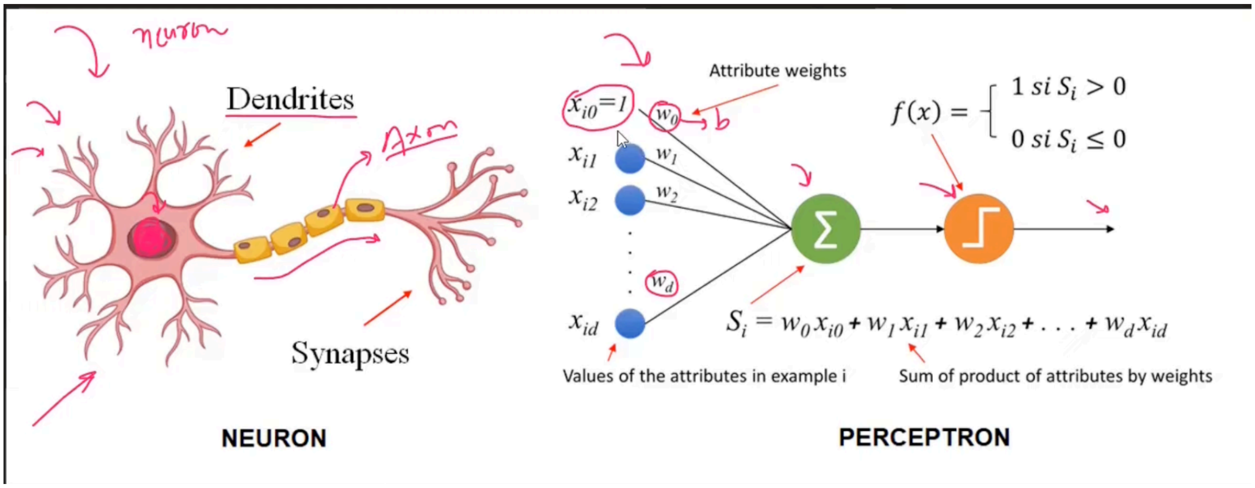
6. Not Suitable for Complex Tasks

- Cannot do image recognition, speech recognition, NLP, etc.
- Too simplistic compared to multi-layer neural networks.

Summary (Limitations of Perceptron)

- ❌ Only works for linearly separable problems
- ❌ Cannot solve XOR problem
- ❌ No hidden layers → very limited expressiveness
- ❌ Step activation function → non-differentiable
- ❌ Binary output only (no probabilities)
- ❌ No convergence if data not separable

Neuron vs Perceptron



Code

Questions

1. What type of function does a perceptron compute?
a) Non-linear function
b) Linear threshold function
c) Exponential function
d) Quadratic function

---

2. Which of the following is TRUE about a perceptron?
a) It can solve any classification problem
b) It can only solve linearly separable problems
c) It requires multiple hidden layers to work
d) It always uses a sigmoid activation

---

3. If input x=[1,0] weights w=[0.5,−0.3]w = [0.5, -0.3]w=[0.5,−0.3], and bias b=0.2b = 0.2b=0.2, what is the perceptron net input zzz?

a) 0.5
b) 0.7
c) 0.2
d) -0.3

---

4. What is the role of the bias in a perceptron?
a) To speed up training
b) To shift the decision boundary
c) To reduce overfitting
d) To increase the number of features

---

5. The perceptron learning rule updates weights as:

a) $w_{new} = w_{old} + \eta(y - \hat{y})x$
b) $w_{new} = w_{old} - \eta(y + \hat{y})x$
c) $w_{new} = w_{old} + \eta(y \cdot \hat{y})$
d) $w_{new} = w_{old} - \eta(y - \hat{y})$

---

6. Why can't a single-layer perceptron solve the XOR problem?
a) Because XOR is not linearly separable
b) Because XOR requires infinite training
c) Because XOR has too many inputs
d) Because perceptron doesn't support bias

---

7. Which activation function was originally used in perceptrons?
a) Sigmoid
b) Step function
c) ReLU
d) Tanh

---

8. The perceptron convergence theorem states that:
a) The perceptron always converges to zero error
b) The perceptron converges if the data is linearly separable
c) The perceptron never converges
d) The perceptron only converges with sigmoid activation

---

9. Which of the following best describes the limitation of perceptrons?
a) They can only classify data in 2D
b) They cannot model non-linear decision boundaries
c) They always overfit
d) They require GPUs to train

---

10. What major advancement overcame the perceptron's limitations and led to deep learning?
a) Support Vector Machines
b) Multi-layer perceptrons with backpropagation
c) Random Forests
d) Gradient Boosting

✅ Answer Key

1 → b
2 → b
3 → b
4 → b
5 → a
6 → a
7 → b
8 → b
9 → b
10 → b