Lab 06: Deep Learning (Neural Networks with Python)

1. Theoretical Part

1.1 Artificial Neural Networks (ANN)

Artificial Neural Networks are computational models inspired by the human brain. They consist of interconnected neurons organized into layers:

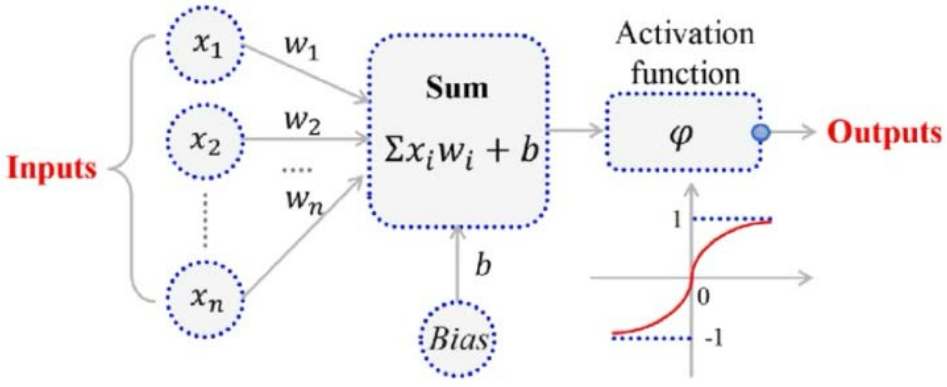
- Input Layer
- Hidden Layer(s)
- Output Layer

Each neuron computes:

$$y = f\left(\sum w_i x_i + b\right)$$

Where:

- w_i: weights
- x_i: inputs
- b: bias
- f: activation function



(b) Artificial neuron

1.2 Perceptron Model

1.2.1 Single-Layer Perceptron

- Structure: Input → Output
- Can only solve **linear problems**

1.2.2 Multilayer Perceptron (MLP)

- Structure: Input → Hidden → Output
- Introduces **non-linearity**
- Can solve complex problems

1.3 Activation Functions

Common functions:

- Sigmoid:

$$f(x) = \frac{1}{1 + e^{-x}}$$

- ReLU:

$$f(x) = \max(0, x)$$

- Tanh:

$$f(x) = \tanh(x)$$

1.4 Error Function (Loss Function)

Quadratic error (MSE):

$$E = \frac{1}{2}(y_d - y)^2$$

Where:

- y_d: desired output
- y: predicted output

1.5 Backpropagation Algorithm

Steps:

1. **Forward propagation**
2. Compute error
3. **Backward propagation (gradients)**
4. Update weights:

$$W_{new} = W_{old} - \eta \frac{\partial E}{\partial W}$$

Where:

- η: learning rate

2. Practical Part: MLP Training

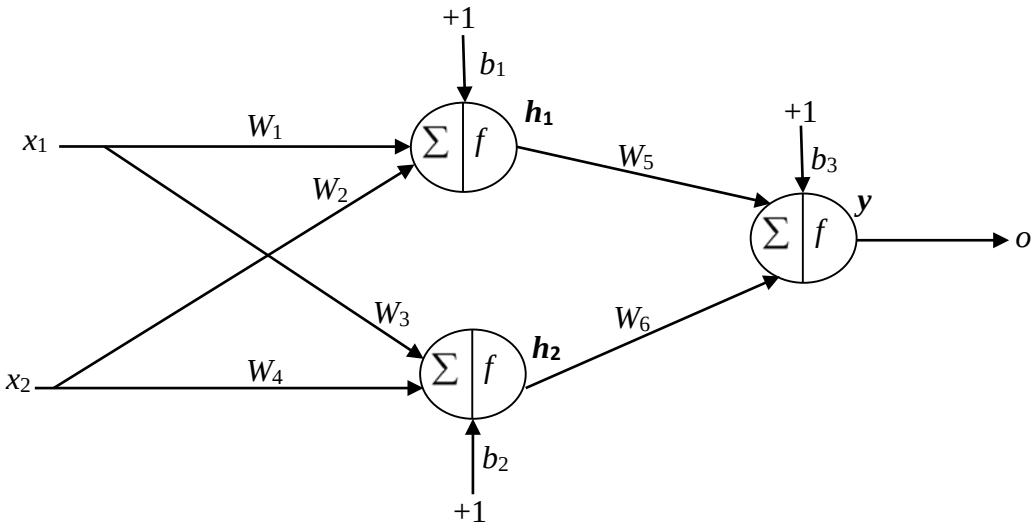
2.1 Problem Definition

We consider the neural network from Chapter 06:

- Inputs: x_1, x_2
- Hidden layer: h_1, h_2
- Output: y

Dataset:

x1	x2	yd
0.1	0.3	0.03



2.2 Initial Parameters

$$W1 = 0.5,\ W2 = 0.1,\ W3 = 0.62,\ W4 = 0.2$$

$$W5 = -0.2,\ W6 = 0.3$$

$$b1 = 0.4,\ b2 = -0.1,\ b3 = 1.83$$

Learning rate:

$$\eta = 0.01$$

❓ Questions

🔷 1. Network Representation

1. Draw the architecture of the neural network (2–2–1).
2. Identify all weights and biases on the diagram.

🔷 2. Forward Propagation

1. Compute:
   - z_1, z_2
   - h_1, h_2 using the sigmoid function
2. Compute:
   - z_3
   - Output y

🔷 3. Loss Computation

1. Write the expression of the Mean Squared Error (MSE).
2. Compute the error E for the obtained output.

🔷 4. Backpropagation

1. Compute the output layer error:

$$\delta_3$$

2. Compute hidden layer errors:

$$\delta_1,\ \delta_2$$

🔷 5. Weights Update

1. Update:
   - W_5, W_6
2. Update:
   - W_1, W_2, W_3, W_4
3. Update all biases.

🔷 6. Iterative Training

1. Repeat the process for several epochs. epochs: 500, 1000,100000
2. Observe the evolution of the error.

🔷 7. NumPy Implementation

1. Implement the neural network using NumPy.
2. Perform:
   - Forward propagation
   - Backpropagation
   - Weight updates
3. Plot the loss function versus epochs.

🔷 8. PyTorch Implementation

1. Implement the same network using PyTorch.
2. Use:
   - `nn.Linear`
   - `Sigmoid`
   - `MSELoss`
3. Train the model and compare results with NumPy.

🔷 9. Analysis

1. Compare manual and PyTorch results.
2. Explain the role of:
   - Learning rate
   - Number of epochs
3. Discuss convergence behavior.

Bonus Question (Optional)

Extend the model to learn the **AND gate** using 4 input samples.