Lab Task 1: Implementing a Simple Perceptron

Objective:
Implement a single-layer perceptron (SLP) to classify linearly separable data. Understand how the perceptron algorithm works and visualize its decision boundary.

Task:

1. **Dataset Creation:**
    - Create a synthetic dataset with two features (X1, X2) and two classes (0 and 1) that are linearly separable. For example, use the following points:

Class 0: [(1, 2), (2, 3), (3, 3), (4, 5)]

Class 1: [(6, 5), (7, 8), (8, 7), (9, 9)]

2. **Perceptron Model:**

- Implement a Perceptron algorithm using the following steps:
    - Initialize weights (w1, w2) and bias (b).
    - Define the activation function as a step function (0 if sum $\leq$ 0, 1 if sum > 0).
    - Update weights using the Perceptron learning rule:

      $w = w + \alpha * (y\_true - y\_pred) * X$

      $b = b + \alpha * (y\_true - y\_pred)$

    - Train the model for a fixed number of epochs (e.g., 10).

3. **Visualization:**
    - Plot the dataset points and the decision boundary learned by the Perceptron model.

Expected Output:

- A plot showing the dataset points and the decision boundary separating the two classes.
- The model should converge to an appropriate decision boundary after training.

Lab Task 2: Training a Multilayer Perceptron (MLP) for XOR Problem

1. **Objective:**
   Understand the capability of a multilayer perceptron (MLP) to solve non-linearly separable problems like XOR. Implement a simple MLP with one hidden layer.

Task:

1. **Dataset:**
   - Use the XOR problem dataset:

Inputs: [(0, 0), (0, 1), (1, 0), (1, 1)]

Targets: [0, 1, 1, 0]

2. **MLP Architecture:**

- Build an MLP model with one hidden layer of 2 neurons.
- Use the sigmoid activation function for both hidden and output layers.
- Implement backpropagation to train the model using the following error function (Mean Squared Error):

  $E = 1/2 * \Sigma (y\_true - y\_pred)^2$

3. **Training:**
   - Use stochastic gradient descent (SGD) with a learning rate ($\alpha = 0.1$) and 1000 epochs.
   - Train the model on the XOR dataset.
4. **Visualization:**
   - Plot the loss curve over epochs.
   - Show the final decision boundary of the MLP.

Expected Output:

- The model should converge and solve the XOR problem with a minimum loss.
- A plot showing the loss curve over epochs.
- A plot of the decision boundary showing the correct classification of the XOR problem.

Lab Task 3: Classifying MNIST Dataset with a Feedforward Neural Network

Objective:
Train a feedforward neural network (FNN) to classify handwritten digits from the MNIST dataset.

Task:

1. **Dataset:**
    - Load the MNIST dataset (digit images of size 28x28) using a library like `tensorflow` or `keras`.
2. **Network Architecture:**
    - Design an FNN with:
        - 2 hidden layers (e.g., 128 neurons in each).
        - ReLU activation for hidden layers.
        - Softmax activation for the output layer (since this is a multi-class classification problem).
3. **Training:**
    - Compile the model with a categorical cross-entropy loss function and the Adam optimizer.
    - Train the model for 10 epochs with a batch size of 32.
4. **Evaluation:**
    - Evaluate the model accuracy on the test dataset.
    - Visualize a few test images along with their predicted and true labels.

Expected Output:

- The model should achieve a classification accuracy of 98% or more on the MNIST test set.
- A few test images with predictions displayed.