Assignment 1

Submission Deadline: 17th September 2025

Programming Task 1

Based on two input features design a binary classifier. For each task, plot the training data and decision boundary separating the two binary classes, also test the model using a few examples and attach the output. Initialize weights randomly and then apply perceptron learning algorithm for optimal weight update. From programming task 4 to 13, implement any two.

1. Implement AND, OR, NAND, and NOR logical gates using single layer perceptron, plot the decision boundary.
2. Implement XOR logical gate by combining single layer perceptrons, visualize the outcomes by plotting decision boundaries.
3. Based on the data of 10-person height and weight, classify whether the person is healthy (1) or overweight (-1).
4. Classify emails as spam (1) or non-spam (-1) based on the frequency of specific keywords and email length.
5. Detect credit card fraud by classifying transactions as fraudulent (1) or non-fraudulent (-1) using transaction amount and frequency.
6. Predict customer churn by classifying customers as likely to churn (1) or not (-1) based on usage duration and customer complaints.
7. Classify text reviews as positive (1) or negative (-1) based on the frequency of positive and negative words.
8. Diagnose patients as having a disease (1) or not (-1) based on blood pressure and cholesterol levels.
9. Predict loan approval by classifying applications as approved (1) or denied (-1) based on applicant's income and credit score.
10. Predict online purchase intent by classifying users as likely to make a purchase (1) or not (-1) based on browsing time and viewed products.
11. Detect malware by classifying computer files as malicious (1) or non-malicious (-1) based on file size and file type.
12. Detect faults in industrial processes by classifying instances as faulty (1) or non-faulty (-1) based on temperature and pressure.
13. Classify days as rainy (1) or non-rainy (-1) based on humidity and atmospheric pressure.

Programming Task 2

Binary Classification of Iris Dataset

- **Description:** The Iris dataset is a popular dataset in machine learning consisting of 150 samples of iris flowers, each with four features (sepal length, sepal width, petal length, and petal width) and a target label indicating the species of iris (setosa, versicolor, or virginica).
- For this task, consider only two classes: setosa and versicolor. Use a single-layer perceptron to classify whether an iris flower belongs to the setosa or versicolor species based on its sepal length and sepal width features.
  1. Load the Iris dataset from a library like scikit-learn or any other source.
  2. Preprocess the dataset to extract only the samples corresponding to the setosa and versicolor species.
  3. Split the dataset into training and testing sets.
  4. Implement a single-layer perceptron class that supports training using gradient descent.
  5. Train the perceptron model on the training set using gradient descent. Plot the decision boundary for each weight update iteration.
  6. Experiment with different hyperparameters such as learning rate and number of epochs to observe their impact on the model's performance. Plot decision boundary for all experiments.
  7. Plot the final decision boundary with different marker/color.
  8. Evaluate the accuracy, precision, recall, and F1-score of the trained model on the testing set.
  9. Report the performance of the trained model and any insights gained like which hyperparameters worked optimally than others.

Programming Task 3

Breast Cancer Diagnosis

- **Description:** Build a binary classifier to predict whether a tumor is malignant or benign based on features extracted from breast cancer biopsies. The dataset contains features computed from digitized images of breast mass, and the target variable indicates whether the tumor is malignant (1) or benign (-1).
  1. Obtain a dataset of breast cancer biopsies with labeled samples indicating malignant or benign tumors (you can use datasets available in libraries like scikit-learn or from sources like the UCI Machine Learning Repository).
  2. Preprocess the dataset by scaling the features to have zero mean and unit variance.
  3. Split the dataset into training and testing sets.
  4. Implement a single-layer perceptron class that supports training using gradient descent.
  5. Train the perceptron model on the training set using gradient descent. Plot the decision boundary for each weight update iteration.
  6. Experiment with different hyperparameters, such as learning rate and number of epochs, to optimize the model's performance. Plot decision boundary for all experiments.
  7. Plot the final decision boundary with different marker/color.
  8. Evaluate the accuracy, precision, recall, and F1-score of the trained model on the testing set.
  9. Report the performance of the trained model and any insights gained like which hyperparameters worked optimally than others.

Programming Task 4

Implement backpropagation for Iris Classification

1. **Data Preparation:** Obtain the Iris dataset and split it into training and testing sets.
2. **Network Architecture:**
    - Define a feedforward neural network with appropriate input, hidden, and output layers.
    - Choose activation functions for each layer.
3. **Initialization:** Initialize the weights and biases of the network with small random values.
4. **Forward Propagation:**
    - Pass the input data through the network to compute the output.
    - Apply activation functions at each neuron.
5. **Loss Calculation:** Compute the loss function (mean squared error) using the predicted outputs and true labels.
6. **Backpropagation:**
    - Compute the gradients of the loss function with respect to the weights and biases using backpropagation.
    - Update the weights and biases using gradient descent.
7. **Training:**
    - Iterate through the training data for multiple epochs, repeating the forward and backward passes.
    - Plot the decision boundary for each weight update iteration.
    - Plot the final decision boundary with different marker/color.
8. **Iterative Improvement:**
    - Experiment with different network architectures (number of hidden layers), activation functions (tanh, logistic sigmoid), and hyperparameters (learning rate, number of epochs, batch size) to improve performance.
    - Plot decision boundary for all experiments.
    - Plot the final decision boundary with different marker/color.
9. **Testing and Evaluation:**
    - Evaluate the trained model on the testing set.
    - Compute metrics (confusion matrix) such as accuracy, precision, recall, and F1-score.
10. **Documentation and Reporting:**
    - Document the implementation details and results.
    - Report the performance of the trained model and any insights gained like which hyperparameters worked optimally than others.