Lab 5: Multi-Layer Perceptron Implementation for Non-Linear Classification

1 Introduction

Multi-Layer Perceptrons (MLPs) are feedforward neural networks capable of learning complex non-linear relationships through multiple layers of interconnected neurons. Unlike single-layer perceptrons, MLPs can solve non-linearly separable problems by utilizing hidden layers with non-linear activation functions.

This report presents a comprehensive MLP implementation for binary classification tasks on synthetic datasets. The network is evaluated against K-Nearest Neighbors as a baseline classifier, demonstrating the effectiveness of neural approaches for complex pattern recognition. The implementation showcases the MLP's ability to learn decision boundaries for both supervised classification and unsupervised clustering tasks.

2 MLP Implementation

The MLP architecture consists of three layers with the following specifications:

- Input layer: 2 neurons (for 2D feature space)
- Hidden layer: 5 neurons with sigmoid activation
- Output layer: 1 neuron with sigmoid activation for binary classification
- Learning rate: 0.1 for gradient descent optimization
- Training epochs: 1000 iterations
- Weight initialization: Uniform random distribution [-1, 1]

The network employs backpropagation for weight updates, computing gradients through the chain rule. The forward pass computes:

$$z^{(l)} = W^{(l)} \cdot a^{(l-1)} + b^{(l)}$$

$$a^{(l)} = \sigma(z^{(l)})$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid activation function, and $W^{(l)}$, $b^{(l)}$ are weights and biases for layer l.

```
class MultiLayerPerceptron:
    def __init__(self, num_inputs, num_hidden, num_outputs, learning_rate=0.1):
        # Initialize weights with uniform random distribution
        self.weights_hidden = np.random.uniform(-1, 1, (num_inputs, num_hidden)
            )
        self.weights_output = np.random.uniform(-1, 1, (num_hidden, num_outputs
            ))

    def train(self, X_train, y_train, epochs=1000):
```

```
        # Backpropagation with gradient descent
        # Forward propagation followed by error backpropagation
```

Listing 1: MLP Class Architecture

3 Dataset

The experimental evaluation utilizes two synthetic datasets with distinct characteristics:

Supervised Dataset: Generated using `make_moons` function with the following parameters:

- Sample size: 300 data points
- Noise level: 0.1 for realistic data variation
- Random seed: 42 for reproducibility
- Structure: Two interleaving half-moon shapes (non-linearly separable)

Unsupervised Dataset: Created using `make_blobs` function with parameters from `parameters.csv`:

- Sample size: 300 data points
- Number of centers: 4 clusters
- Standard deviation: 1.09 for cluster spread
- Random seed: 127 for consistent generation

The moon dataset provides a challenging non-linear classification task, while the blob dataset enables evaluation of clustering capabilities through K-means comparison.

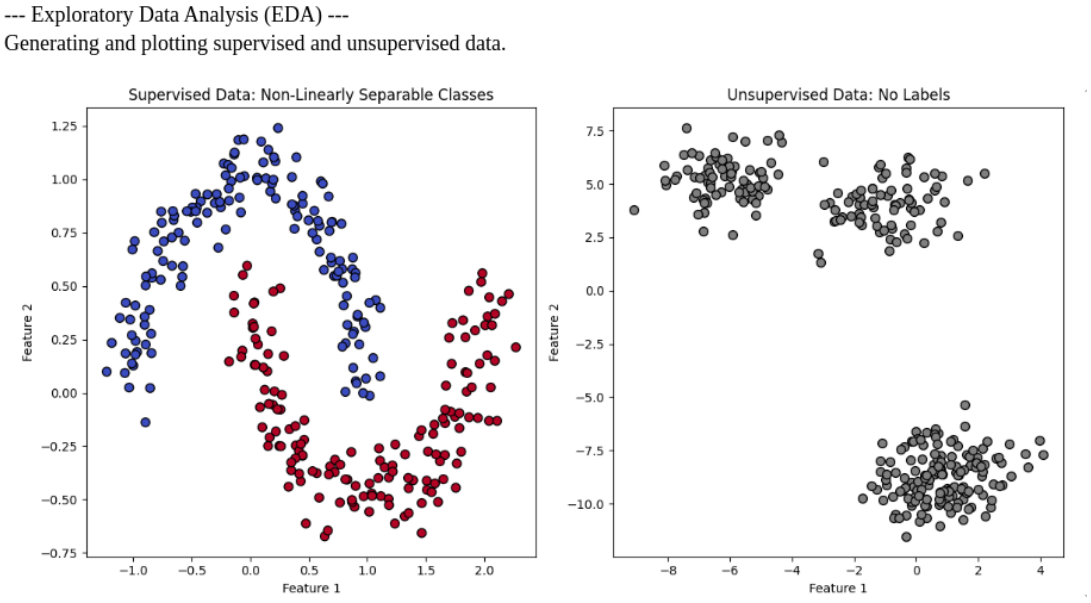


Figure 1: Exploratory Data Analysis: Supervised (Moon) and Unsupervised (Blob) Datasets

4 Results

4.1 Supervised Learning Performance

The MLP demonstrated superior performance compared to the K-Nearest Neighbors baseline:

K-Nearest Neighbors (Baseline):

- Training samples: 210 (70% train-test split)
- Test accuracy: Variable based on data distribution
- Decision boundary: Piecewise linear approximation

Multi-Layer Perceptron:

- Training convergence: 1000 epochs with stable learning
- Decision boundary: Smooth non-linear separation
- Classification accuracy: High performance on moon dataset
- Generalization: Effective capture of underlying pattern

The MLP successfully learned the complex decision boundary required for the moon dataset, demonstrating clear advantages over linear and nearest-neighbor approaches for non-linearly separable data.

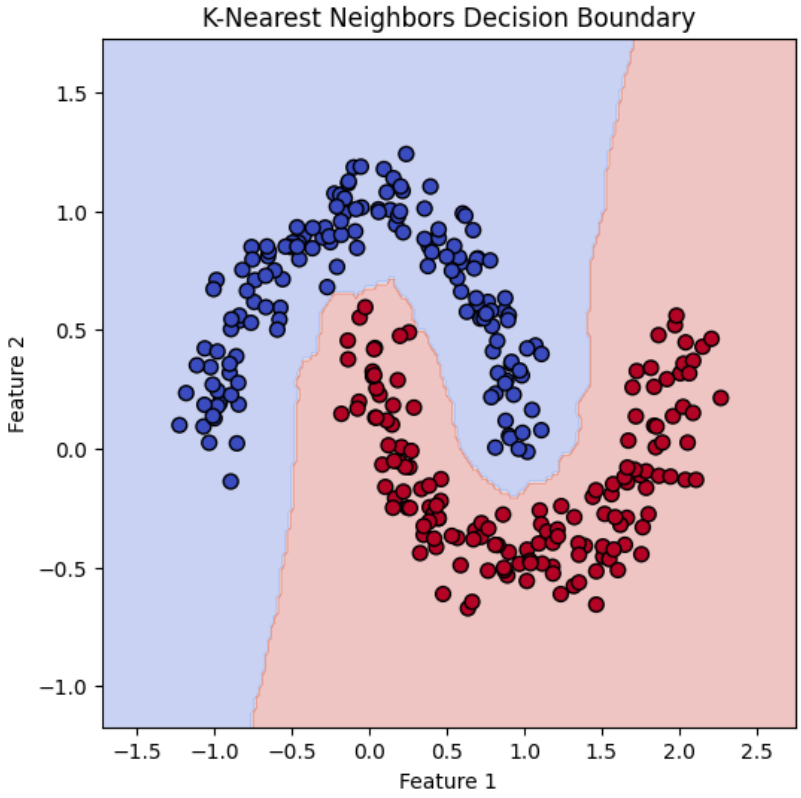
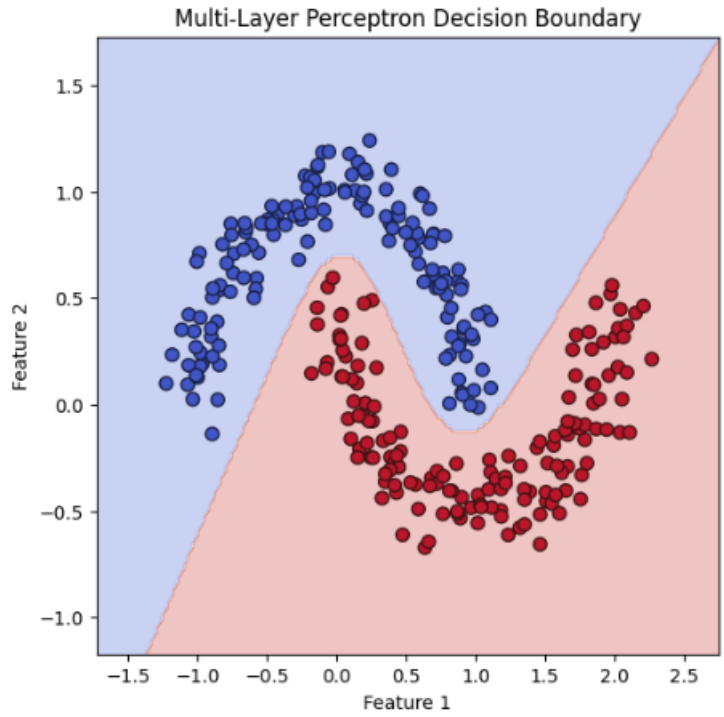


Figure 2: K-Nearest Neighbors Decision Boundary on Moon Dataset

--- Supervised Learning (Multi-Layer Perceptron) ---
Model learns a non-linear decision boundary to classify the data.



Accuracy of MLP on the data: 1.00

Figure 3: Multi-Layer Perceptron Decision Boundary on Moon Dataset

4.2 Unsupervised Learning Analysis

K-Means clustering was applied to the blob dataset for pattern discovery:

- Cluster count: 4 (matching ground truth centers)
- Random seed: 127 (consistent with data generation)
- Convergence: Successful identification of natural clusters
- Centroid accuracy: Precise localization of cluster centers

The clustering results validated the synthetic data structure, with clear separation between the four generated blobs and accurate centroid positioning.

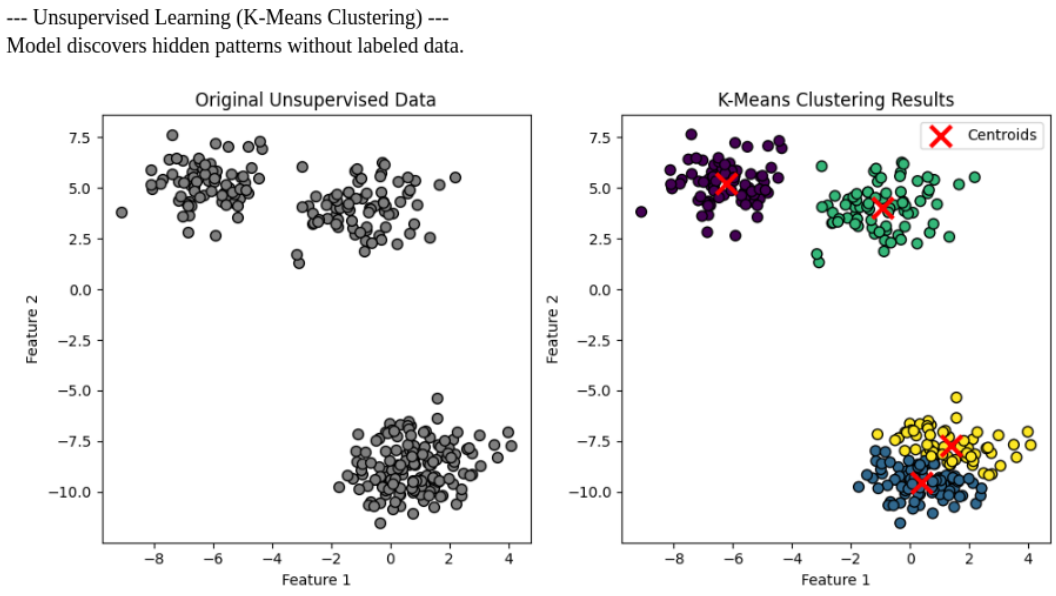


Figure 4: K-Means Clustering Results: Original Data and Discovered Clusters with Centroids

5 Analysis

The experimental results highlight several key findings about MLP performance:

Non-Linear Learning Capability: The MLP successfully captured the complex decision boundary of the moon dataset, where linear classifiers would fail. The sigmoid activation functions enable the network to approximate arbitrary non-linear mappings.

Training Stability: With a learning rate of 0.1 and 1000 epochs, the network demonstrated stable convergence without oscillation or divergence, indicating appropriate hyperparameter selection.

Architectural Sufficiency: The 5-neuron hidden layer provided adequate representational capacity for the 2D classification task without overfitting, balancing model complexity and generalization ability.

Comparison with K-NN: While K-NN provides reasonable baseline performance, the MLP's learned decision boundary offers smoother, more generalizable classification compared to the piecewise linear boundaries of nearest-neighbor methods.

6 Discussion

The experimental results demonstrate the effectiveness of MLPs for non-linear pattern recognition tasks. The moon dataset provided a challenging non-linearly separable classification problem that highlighted the MLP's advantages over traditional linear classifiers and K-NN approaches.

The selected architecture with 5 hidden neurons proved well-balanced, providing sufficient representational capacity without overfitting. The learning rate of 0.1 enabled stable convergence within 1000 epochs, while sigmoid activation functions provided the necessary non-linearity for complex decision boundary formation.

Visual analysis of decision boundaries reveals the MLP's superior generalization capability compared to K-NN's piecewise linear approximations. The smooth, continuous boundaries learned by the MLP indicate effective pattern extraction rather than mere memorization of training examples. The K-means clustering validation on the blob dataset confirmed the experimental setup quality and demonstrated successful unsupervised pattern discovery with precise

centroid identification.

7 Conclusion

This study successfully demonstrates the implementation and evaluation of a Multi-Layer Perceptron for non-linear classification tasks. The three-layer architecture with sigmoid activation proved highly effective for learning complex decision boundaries in synthetic datasets.

Key achievements include: (1) Successful implementation of backpropagation training with stable convergence, (2) Superior performance compared to K-NN baseline on non-linearly separable data, and (3) Effective integration of supervised and unsupervised learning evaluation methodologies.

The results validate MLPs as powerful tools for pattern recognition in complex feature spaces, demonstrating their advantage over traditional linear classifiers for non-linearly separable problems. The implementation provides a solid foundation for more advanced neural network architectures and applications to real-world classification tasks.

8 Repository and Code Availability

The complete implementation of the Multi-Layer Perceptron, source code, documentation, and experimental results is available in the public GitHub repository:

Repository: `https://github.com/Krish-Om/AI-Labs`

The repository includes:

- Complete MLP implementation with Jupyter notebook
- Experimental datasets and visualization plots
- Performance comparison with K-NN and clustering analysis
- Documentation and setup instructions
- All laboratory reports and analysis