Assignment 1 – Deep Neural Networks (DNN)

Programming Assignment: Comparing Linear Models and MLPs, Max Marks: 10
Course: Deep Neural Networks, Type: Individual Programming
Submission: .ipynb + HTML version (Download HTML version of the ipynb file)

1. Overview

What You Will Do - Select a real-world dataset (≥ 500 samples, ≥ 10 <50 features), implement a baseline linear model from scratch (no sklearn models), implement an MLP from scratch, compare performance on test data, analyze results.

Learning Objectives - Understand regression/classification fundamentals, implement gradient descent, build neural networks without high-level libraries, evaluate and compare models, analyze computational trade-offs.

2. Dataset Requirements

Source: UCI ML Repository, Kaggle, OpenML, Data.gov
Samples: ≥ 500
Features: ≥ 10 (excluding target)
Data type: Numeric and/or categorical
Task: **Regression or Classification (Multiclass)**

Not Allowed Datasets- Image datasets (MNIST, CIFAR), text/NLP datasets, audio datasets, preprocessed datasets from Papers with Code, toy datasets with < 500 samples (Iris, Wine, Diabetes).

3. Implementation Requirements

Must Implement From Scratch

Regression: Linear Regression + MLP (with backprop)
Binary Classification: Logistic Regression + MLP
Multi-class: Softmax Regression + MLP

Allowed Libraries - NumPy (arrays), Pandas (data handling), Matplotlib/Seaborn (plots), sklearn ONLY for train_test_split, StandardScaler/MinMaxScaler, LabelEncoder/OneHotEncoder.

Prohibited - sklearn.linear_model, sklearn.neural_network, TensorFlow, Keras, PyTorch, JAX, any high-level ML library.

Penalty for using prohibited libraries: –5 marks.

4. Assignment Structure

4.1 Dataset Selection (1 Mark) - Load dataset, describe it, state regression/classification, justify primary evaluation metric important for your dataset.

4.2 Data Preprocessing (1 Mark) -Train-test split (70-30/80–20/ 90–10), handle missing values, encode categorical variables, scale features, etc.

4.3 Baseline Model (2 Marks) - Implement from scratch, initialize weights, forward pass, compute loss, compute gradients, update weights (w = w – lr × grad), store loss at every iteration (loss_history required), implement predict() function which will return prediction.

4.4 Multi-Layer Perceptron (4 Marks) - At least 1 hidden layer, ReLU for hidden layers, appropriate output activation, full forward propagation, backpropagation using chain rule, gradient descent updates, loss_history required, architecture format: [input, hidden1, hidden2, …, output].
Mandatory functions inside MLP class:

- init() ← for architecture initialisation
- initialize_parameters() ← (initialize W and b for each layer)
- forward_propagation() ← # Compute a c t i v a t i o n s through all layers
- backward_propagation() ← # Compute gradients using chain rule
- fit() ← # Training loop with forward / backward passes
- predict() ← # Return p r e d i c t i o n s

Implement function get_assignment_results().
Return: dataset_name, n_samples, n_features, problem_type, primary_metric, baseline_model metrics (including test performance, training time), mlp_model metrics (including architecture, test performance, training time).

Missing function: –2 marks.

4.5 Evaluation & Comparison (2 Marks)

Regression: MSE, RMSE, MAE, R^2
Classification: Accuracy, Precision, Recall, F1, etc

Required plots: training loss curves (baseline vs MLP), performance comparison bar chart, optional domain-specific plots.

Analysis (< 200 words): which model performed better, by how much, why, computational cost difference, challenges/surprises.

Debugging Checklist
If model not working:
1. Check data shapes at each step, 2. Verify gradient computation
3. Check for NaN/Inf values, 4. Try learning rates: [0.001, 0.01, 0.1]
5. Ensure loss computed correctly, 6. Verify update: w = w - lr * grad
7. Check activation functions, 8. Print intermediate values

5. Evaluation Criteria (10 Marks)

Dataset: 1, Preprocessing:1, Baseline (correct implementation, decreasing loss): 2
MLP (valid architecture, forward/backprop, decreasing loss): 4
Comparison (metrics, plots, < 200 words analysis): 2

Deductions: Dataset too small: –1, Loss not decreasing: –1 per model, Analysis ≥ 200 words: –0.5, Prohibited libraries: –5

6. Submission Instructions

File name format: YourStudentID_assignment1.ipynb,
Example: 2025AA05005_assignment1.ipynb. Upload to LMS, verify successful submission.

7. Academic Integrity

Allowed: Clarifications, conceptual help, debugging assistance.
Not allowed: Sharing code, copying implementations, using AI for full solutions, collaboration.
Individual assignment.

Start early, scale features, track loss carefully, use vectorized NumPy, test frequently, ensure both models converge.