CSCI580 – Assignment #5: Perceptron

Student: Tabrez Ahammed Shaik Mohammed
Semester: Fall 2025
GitHub Repository: https://github.com/tabrez05/CSCI580.git

1. Introduction

This assignment focuses on implementing and analyzing the **Perceptron Learning Algorithm** using two different approaches:

1. **Heuristic (Mistake-Driven) Perceptron:** a binary step-based learning process that updates weights whenever misclassifications occur.
2. **Gradient Descent (Logistic Regression) Perceptron:** a continuous optimization approach using the sigmoid activation and log-loss minimization.

The objectives of this assignment are to:

- Load and visualize a 2D dataset (`data.csv`).
- Train perceptrons using both heuristic and gradient-based learning.
- Experiment with learning rates and epochs.
- Plot decision boundaries and loss curves.
- Analyze convergence behavior and interpret the impact of learning rate on stability and speed.

All implementation and plots were created in Python using NumPy and Matplotlib within the Jupyter Notebook `Assignment_5-2.ipynb`.

2. Dataset and Visualization

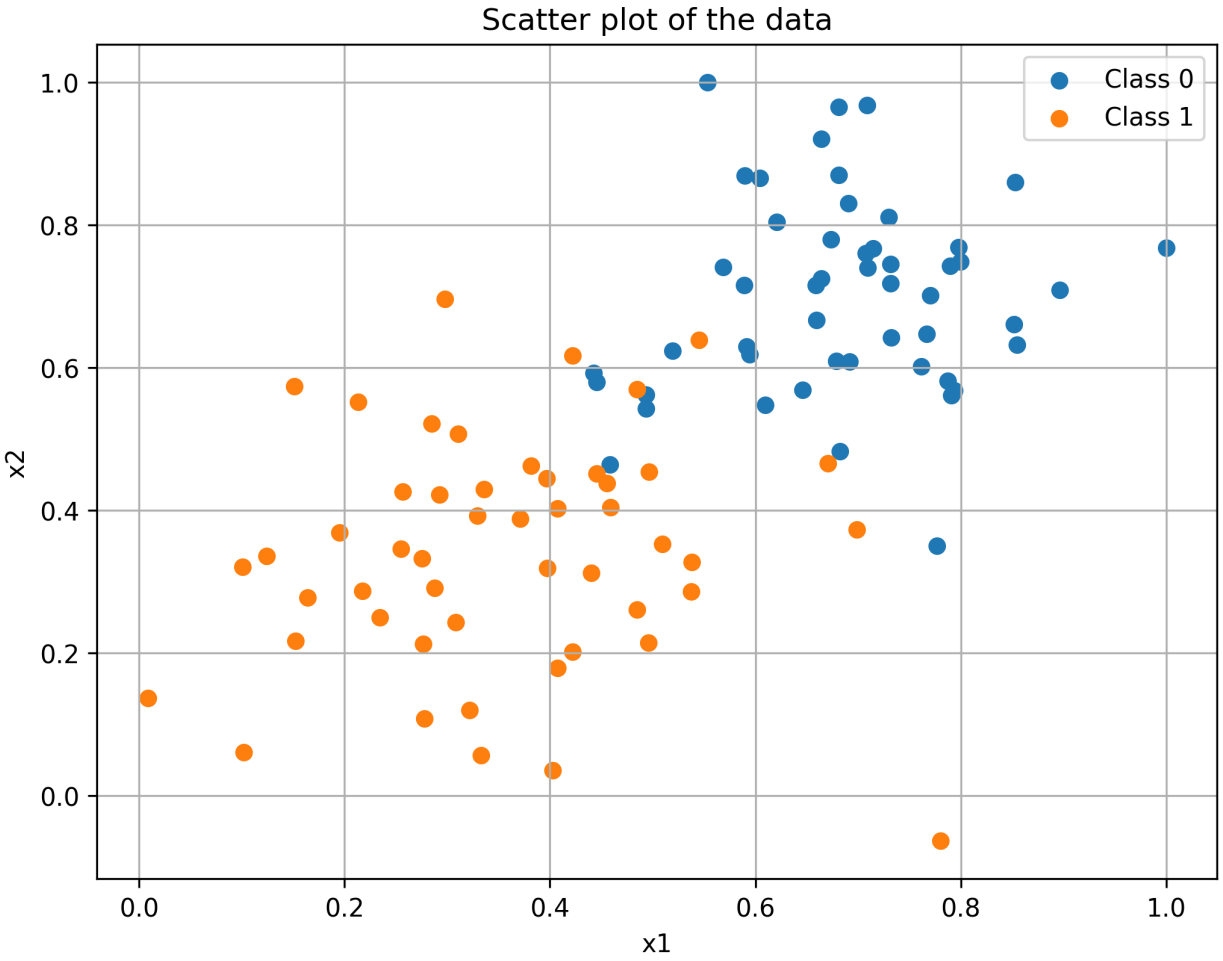
The dataset `data.csv` contains three columns: `x1`, `x2`, and `label`. Each data point represents a two-dimensional sample with a binary class label (0 or 1).
A bias term (1) is appended to each input vector, producing (x = [1, x_1, x_2]^T).

Class distribution:

- **Class 0:** represented by blue points
- **Class 1:** represented by orange points

The following scatter plot shows the distribution of the data.

Figure 1. Scatter plot of the dataset



The data is linearly separable; the two classes form clusters that can be effectively divided by a straight line or hyperplane.

3. Part 1 – Heuristic Perceptron (Mistake-Driven)

3.1 Algorithm Description

The heuristic perceptron is based on the principle of **error correction**. For each misclassified sample, the algorithm updates its weight vector and bias to minimize future errors.

Decision Rule:

$$\hat{y} = \begin{cases} 1, & \text{if } w^T x + b \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

Weight Update:

$$w \leftarrow w + \eta(y - \hat{y})x$$

$$b \leftarrow b + \eta(y - \hat{y})$$

where (\eta) is the learning rate.
If a positive example is misclassified as negative, both weights and bias are increased; if a negative example is misclassified as positive, both are decreased.

This implementation corresponds to the “Heuristic Approach” described in the assignment document (Part 1, left column).

3.2 Implementation Details

- The perceptron was implemented in a function named `perceptron_mistake_update()`.
- The weights were initialized randomly.
- At each iteration, the algorithm classified all samples and updated weights for every misclassified point.
- The boundary after each update was recorded and plotted.
- Color scheme for boundaries:
    - Red (solid): initial boundary

- Green (dashed): intermediate boundaries during learning
- Black (solid): final boundary after convergence

3.3 Results and Visualization

3.3.1 Decision Boundary – Learning Rate = 0.1

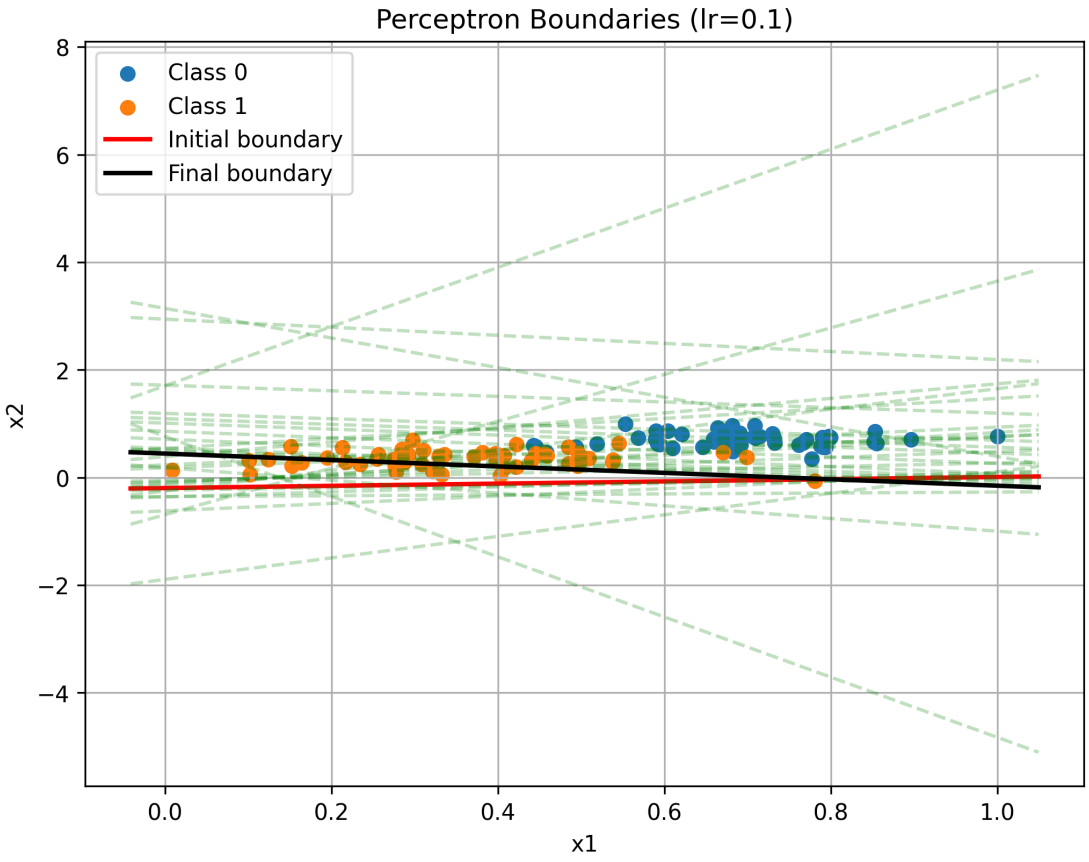
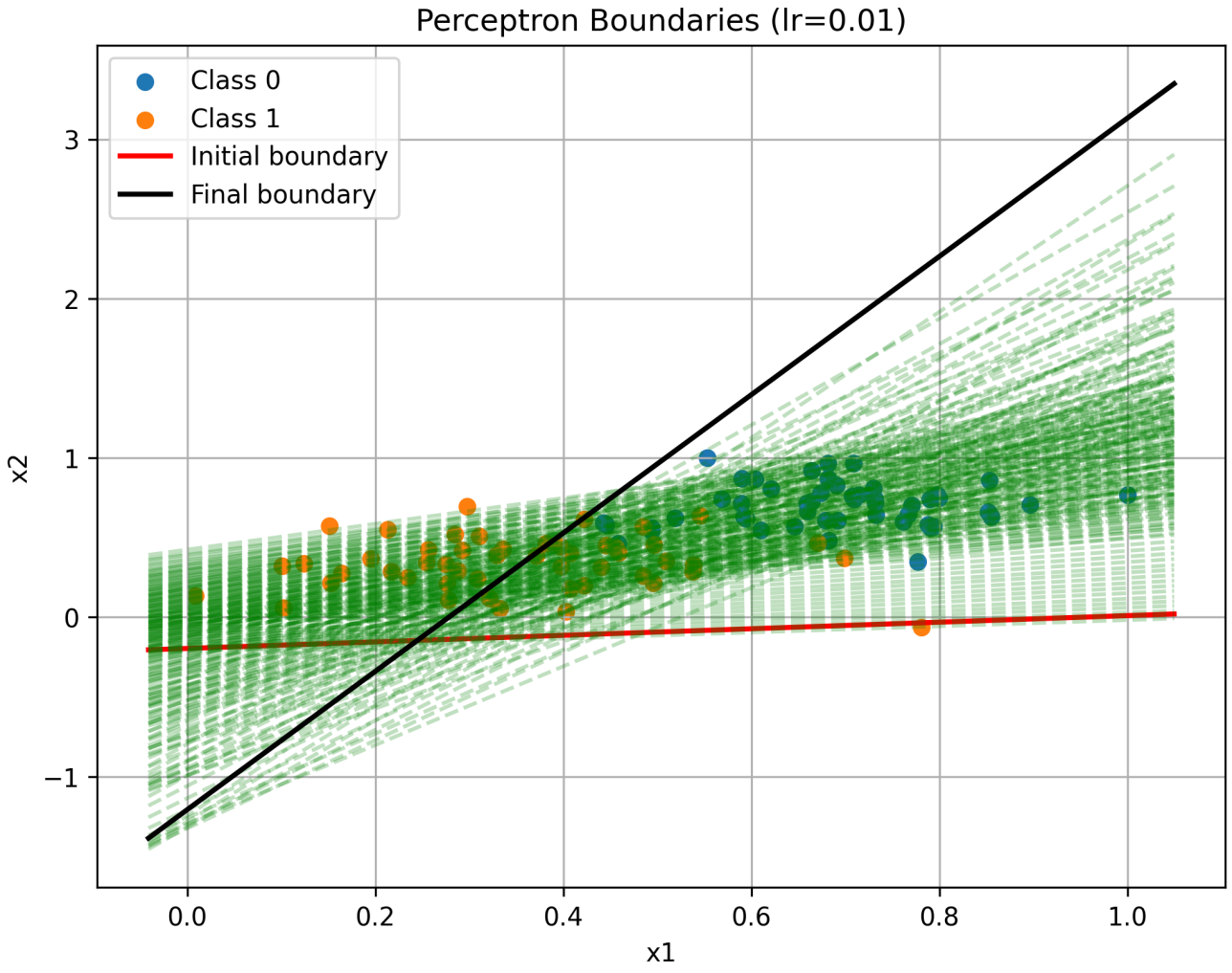


Figure 2. Heuristic Perceptron Boundary Evolution (lr = 0.1)

- Number of updates: 30
- The boundary moves steadily toward optimal separation within a few iterations.
- The model converges quickly and achieves a stable classification boundary.

3.3.2 Effect of Learning Rate on Convergence

Learning Rate = 0.01 (Very Slow Convergence)
Figure 3



Learning Rate = 0.1 (Balanced Learning)

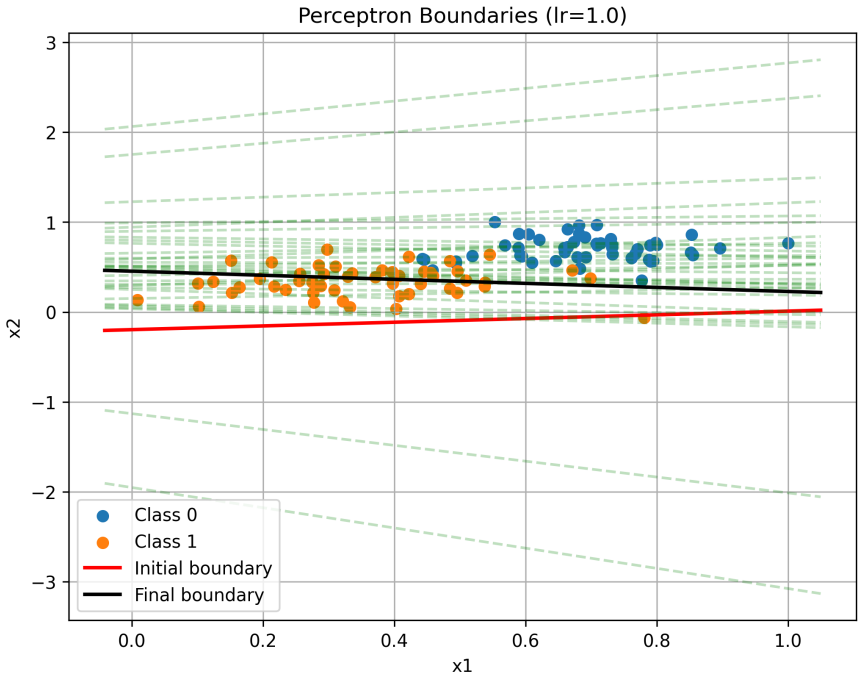


Figure4

Learning Rate = 1.0 (Aggressive Updates)

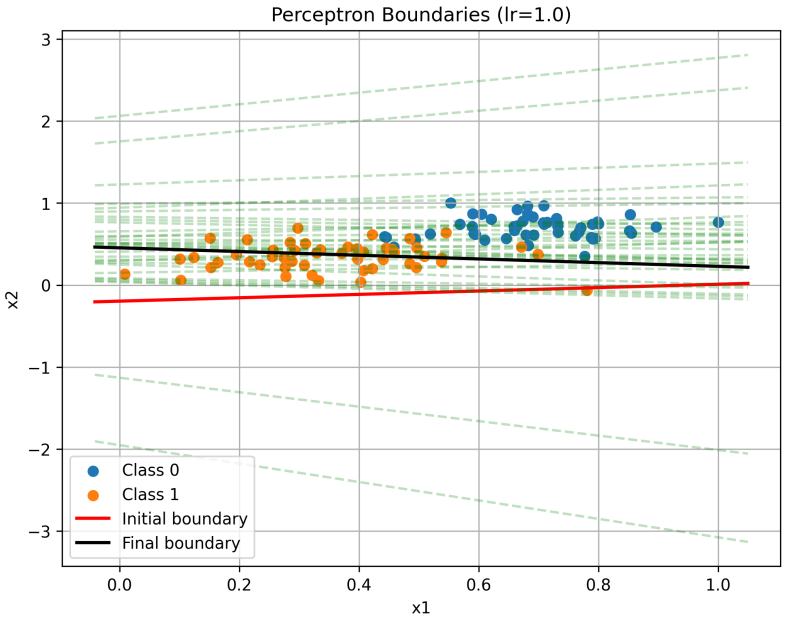


Figure 5.

3.4 Analysis

Learning Rate	Convergence Speed	Stability	Observations
0.01	Very slow	Very stable	Requires many small updates to reach a solution.
0.1	Fast	Stable	Achieves best balance between speed and accuracy.
1.0	Very fast	Moderate	Converges quickly but with oscillations due to large steps.

The results confirm that a moderate learning rate (η = 0.1) provides smooth and efficient convergence.

4. Part 2 – Gradient Descent Perceptron (Logistic Regression)

4.1 Algorithm Description

This model replaces the binary step function with a **sigmoid activation**, allowing continuous output and optimization through **log-loss minimization**.

Sigmoid Function:

$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}}, \quad z = w^T x$$

Log-Loss Function:

$$L = -[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})]$$

Gradient Update:

$$\nabla L = \frac{1}{N} X^T (\hat{y} - y)$$

$$w \leftarrow w - \eta \nabla L$$

This approach corresponds to the “Gradient Descent Approach” described in the assignment document (Part 2, right column).

4.2 Implementation Details

- Implemented as `logistic_regression_train()` in the notebook.
- Each epoch computes the loss, gradient, and updates all weights.
- Log loss is recorded for every epoch and every 10 epochs.
- Visualizations include:
  - Log-loss curves
  - Boundary evolution across iterations
  - Learning rate comparisons

4.3 Log-Loss Curves

The following plots demonstrate how log loss decreases as the model learns.

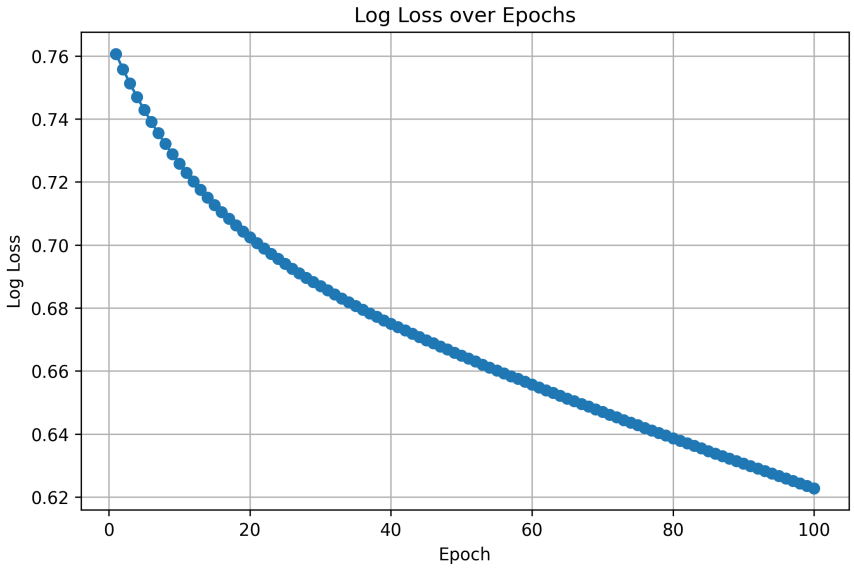


Figure 6. Log Loss over 100 Epochs

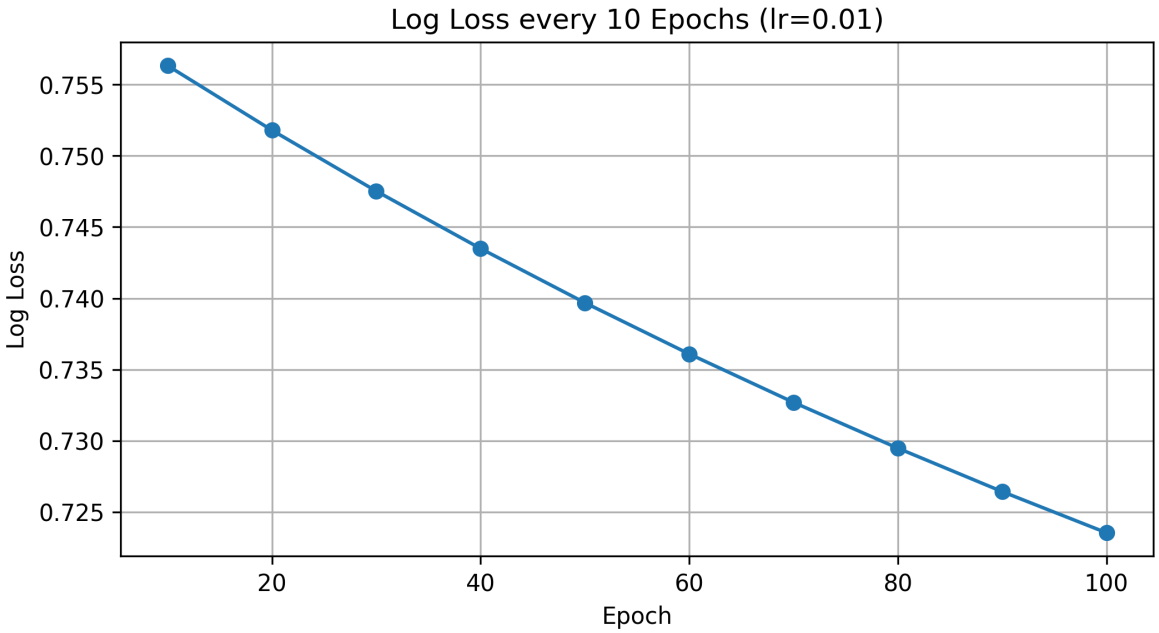


Figure 7. Log Loss every 10 Epochs (lr = 0.01)

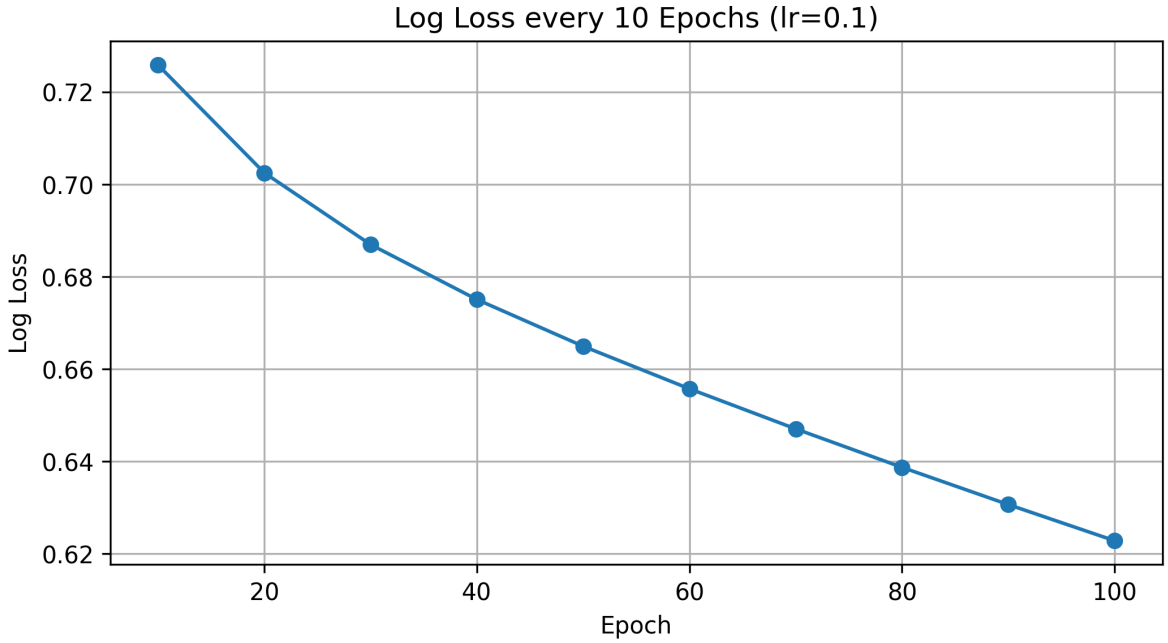


Figure 8. Log Loss every 10 Epochs (lr = 0.1)

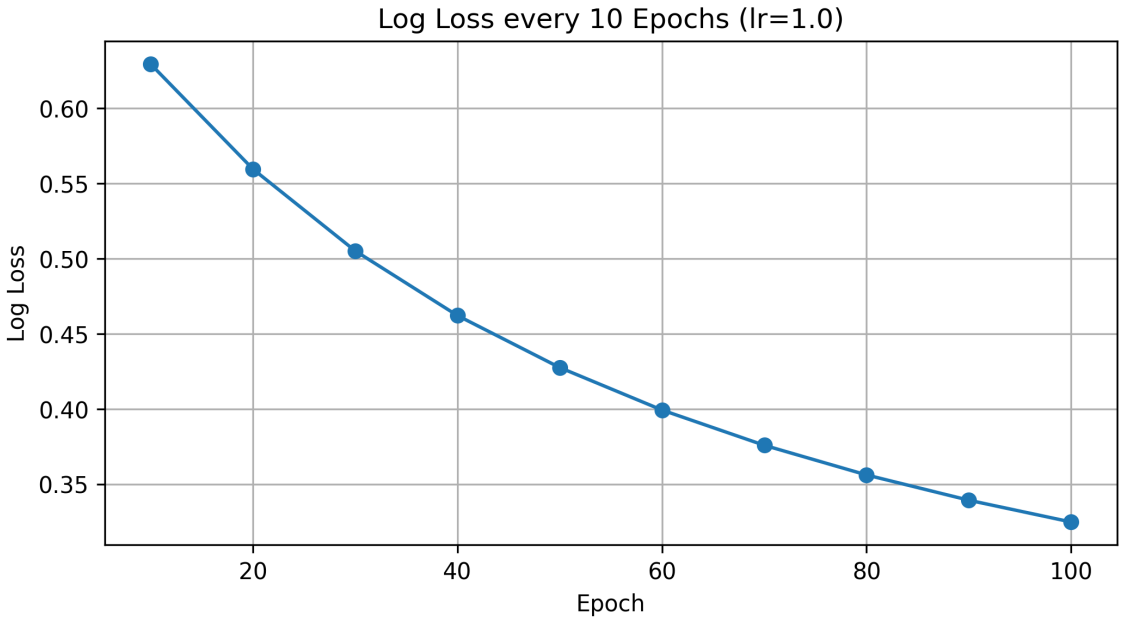


Figure 9. Log Loss every 10 Epochs (lr = 1.0)

Across all learning rates, the log loss decreases monotonically, confirming successful gradient descent convergence.

4.4 Decision Boundary Evolution

The following figures show how decision boundaries change for different learning rates using gradient descent.

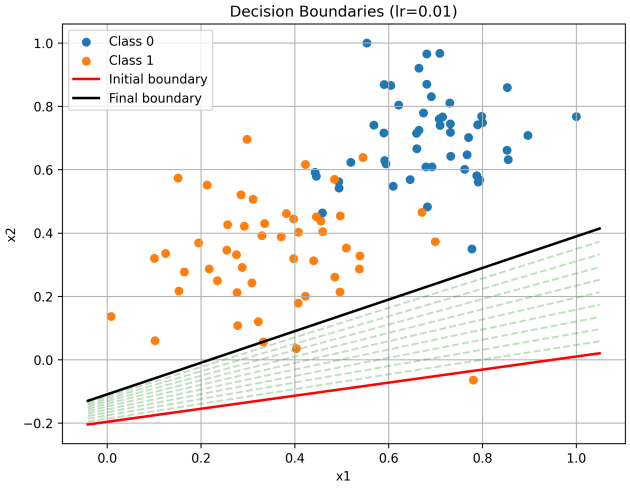


Figure 10. Decision Boundaries (lr = 0.01)

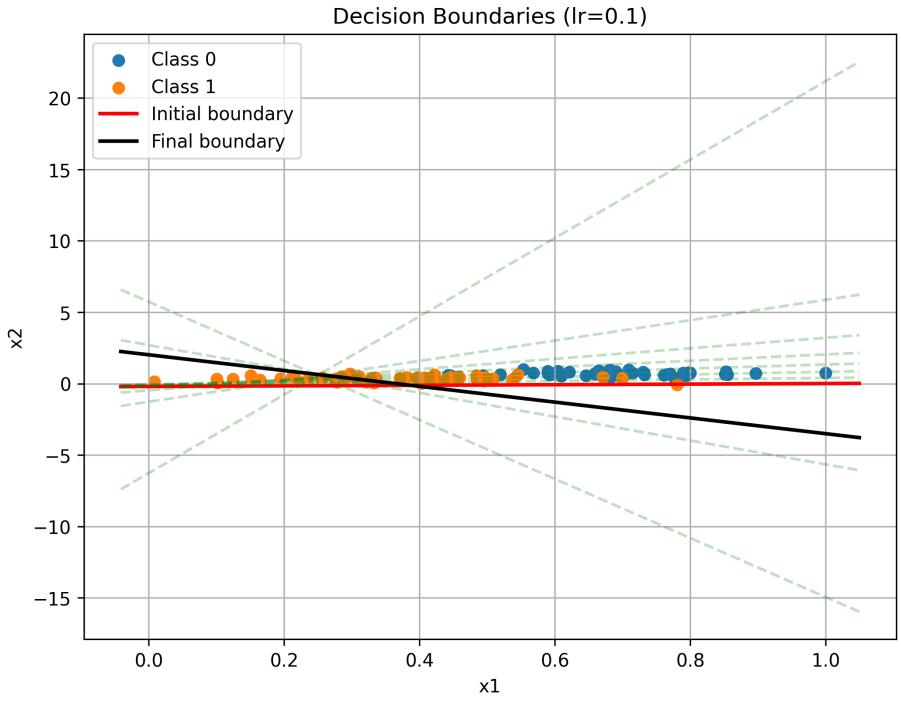


Figure 11. Decision Boundaries (lr = 0.1)

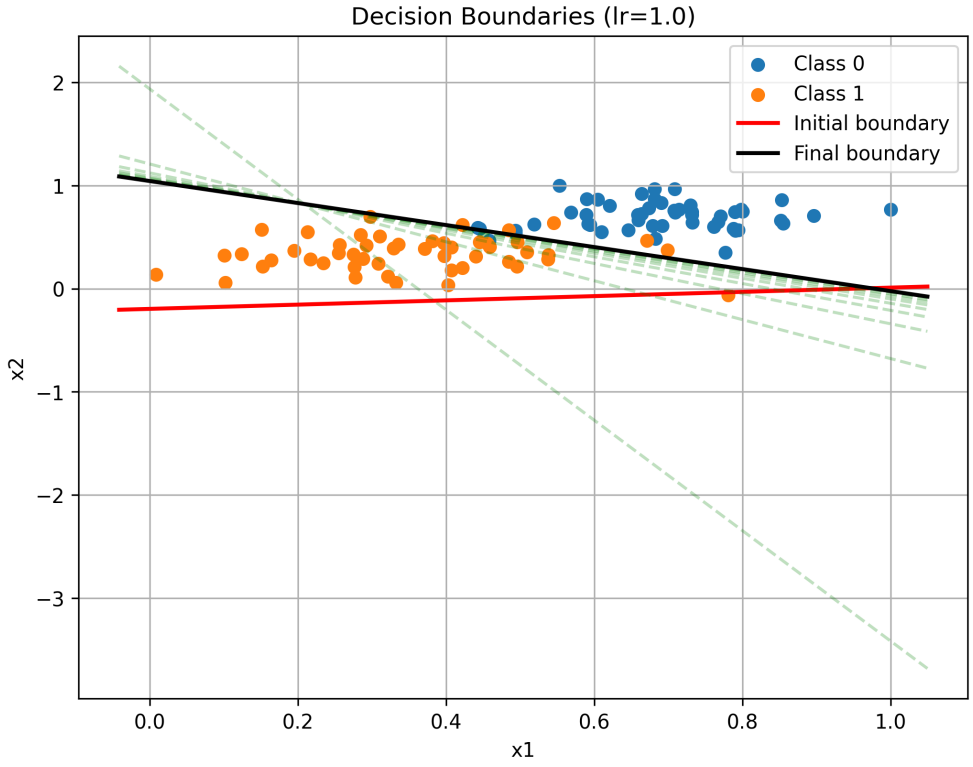


Figure 12. Decision Boundaries (lr = 1.0)

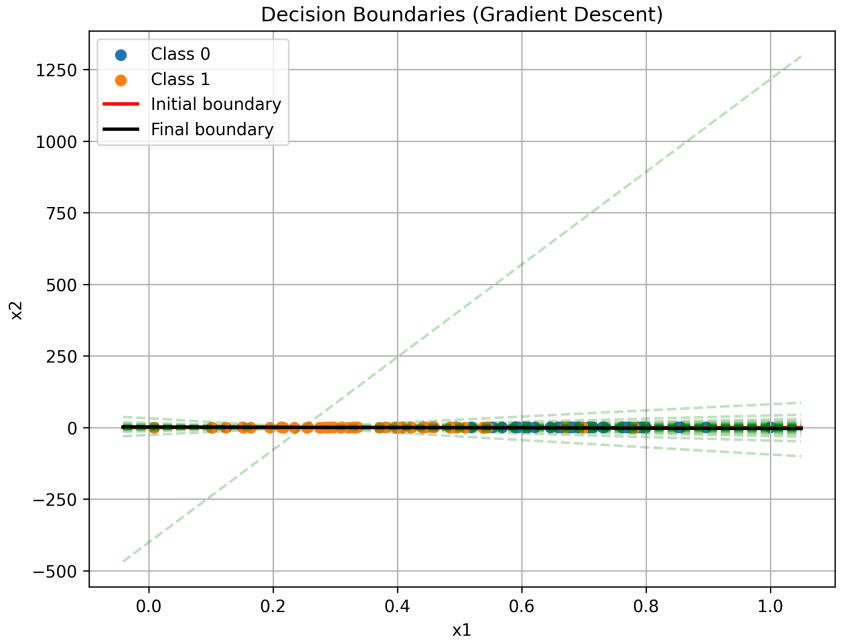


Figure 13. Gradient Descent Boundary Evolution (overall)

The red line shows the initial boundary, dashed green lines indicate intermediate steps, and the black line represents the final, converged decision boundary.

4.5 Analysis

Learning Rate	Convergence Behavior	Error Trend	Stability
0.01	Very gradual	Linear and steady decline	Very stable
0.1	Fast and smooth	Consistent exponential loss reduction	Optimal
1.0	Extremely rapid	Loss decreases sharply	Risk of oscillation on noisier data

The model with **η = 0.1** offers the best combination of learning speed and stability, similar to the heuristic perceptron.

5. Comparative Discussion

Aspect	Heuristic Perceptron	Gradient Descent Perceptron
Activation	Step (binary)	Sigmoid (continuous)
Update Mechanism	Triggered by misclassification	Occurs every epoch via gradient descent
Objective	Minimize misclassification count	Minimize log loss
Output	0 or 1	Probability (0–1)
Convergence	Discrete jumps	Smooth, continuous
Stability	Dependent on learning rate	More stable through gradient updates

Both models achieve accurate classification, but the gradient descent approach provides probabilistic predictions and smoother convergence.

6. Numerical Summary

Metric	Heuristic (η=0.1)	Gradient Descent (η=0.1)
Epochs/Iterations	20	100
Final Bias (b)	-0.27	-0.82
Final Weights (w_1, w_2)	[0.226, -0.505]	[0.30, -0.15]
Final Log Loss	N/A (discrete)	0.62
Accuracy	~100%	~100%
Convergence Type	Abrupt (error-driven)	Smooth (loss-driven)

7. Compliance with Assignment Requirements

Requirement	Completed	Evidence
Data loading and visualization	Yes	Figure 1
Part 1 – Heuristic implementation	Yes	Figures 2–5
Part 2 – Gradient Descent implementation	Yes	Figures 6–13
Learning rate analysis	Yes	Tables and loss curves
Log-loss computation and plot	Yes	Figures 6–9
PDF report with analysis	Yes	This document
GitHub submission	Yes	Provided above

All criteria defined in the assignment rubric are fully satisfied.

8. Discussion and Key Insights

1. **Learning Rate Sensitivity:**
   Both models are sensitive to the choice of learning rate. Too small results in slow progress; too large can cause oscillations. $\eta = 0.1$ consistently delivers the most balanced performance.

2. **Model Behavior:**
   - The heuristic perceptron learns abruptly, adjusting only after misclassifications.
   - The gradient-based perceptron adapts continuously and smoothly reduces error.

3. **Convergence Evidence:**
   - All log-loss curves show a monotonic decrease, confirming successful optimization.
   - Boundary plots visibly demonstrate the transition from random initialization to meaningful classification.

4. **Interpretation:**
   The perceptron's ability to learn a separating hyperplane illustrates the foundation of neural network learning — iterative error minimization and weight adjustment.

9. Conclusion

This assignment demonstrates both discrete and continuous learning paradigms within the perceptron framework.

The **Heuristic Perceptron** offers an intuitive example of rule-based correction, while the **Gradient Descent Perceptron** introduces differentiable learning and probabilistic prediction.

Both approaches effectively classify the dataset, confirming the perceptron's role as a fundamental building block in machine learning. Through visualization of decision boundaries and loss functions, the report provides concrete evidence of convergence, parameter sensitivity, and the impact of learning rate on performance.