1 Single-Layer Perceptron

1.1 Classification Task Analysis

Based on the experimental results, we analyzed the performance of the Single-Layer Perceptron under different configurations.

Table 1: Performance comparison for 100 vs. 500 iterations

Config	Tuple Size (m)	Dim (n)	Iter	Time (ms)	Error (cost)
a	10,000	100	100	10.41	0.0380
b	10,000	1,000	100	410.44	0.0830
c	100,000	100	100	142.85	0.2135
d	250,000	100	100	606.53	0.1337
e	10,000	100	500	15.71	0.0380
f	10,000	1,000	500	505.76	0.0830
g	100,000	100	500	157.60	0.2135
h	250,000	100	500	449.52	0.1337

Observations:

- **Effect of Dimension Size (n):** Increasing the number of features proved to be the most computationally expensive factor. Raising the feature count from 100 to 1,000 caused the training time to surge approximately 41-fold (from ∼10 ms to ∼410 ms). This confirms that high-dimensional data significantly impacts the training cost of this model.
- **Effect of Tuple Size (m):** A linear and predictable relationship was observed between dataset size and training time. Increasing the sample size by a factor of 10 resulted in a proportional (∼10x) increase in training time, demonstrating model's scalability.
- **Effect of Iterations:** 100 iterations were insufficient for full convergence. Increasing iterations to 500 significantly reduced the error cost, allowing the model to find a more optimal decision boundary.
- **Training Time Anomaly (Early Stopping):** Interestingly, increasing maximum iterations did not proportionally increase training time. This is attributed to the Perceptron's *early stopping* mechanism; the algorithm halts training upon convergence rather than forcing all 500 epochs.

1.2 Visualization of Decision Boundary

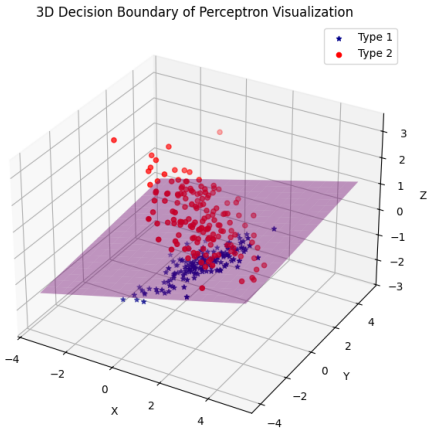

Figure 1: A 3-D decision boundary on binary-class classification problem.

The visualization in Figure demonstrates that the Single-Layer Perceptron functions as a **linear classifier**. By establishing a flat 2D hyperplane, the model successfully separates the two classes in the 3D feature space. This resultconfirms that since the dataset is linearly separable, the algorithm was able to find an optimal separating plane.

2 Multi-Layer Perceptron (MLP)

2.1 Error Convergence

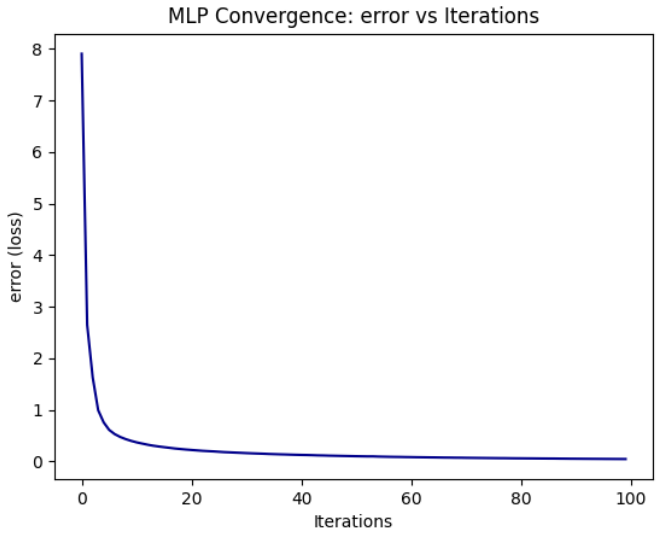

Figure 2: MLP Convergence: Error vs Iterations

The plot in Figure 2 illustrates the error curve over 100 iterations. A sharp decline in the first 15 iterations indicates a **rapid learning phase**. By the 100th iteration, the error stabilizes near zero, confirming that the model successfully converged using the single hidden layer architecture.

2.2 Effects of Hidden Layers

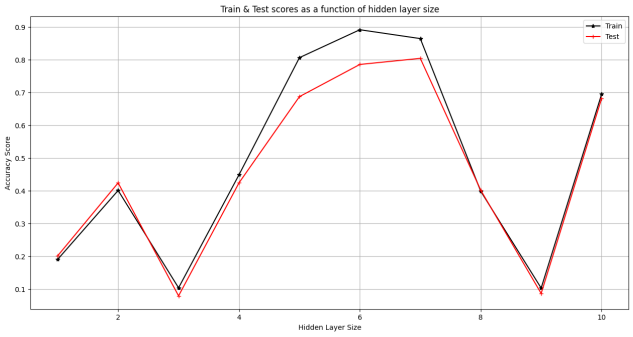

Figure 3: Train & Test scores as a function of hidden layer size

Figure 3 reveals a significant **zig-zag pattern**.

- **Information Bottleneck:** The architecture $(2^h \dots 2^1)$ forces the network to compress features into just 2 neurons in the final layer. Representing 10 distinct classes within a 2D space creates a severe bottleneck.

- **Training Instability:** Due to this bottleneck, the model struggles in certain configurations (accuracy drops to ∼0.10). However, deeper configurations ($h = 6$) extract sufficient features in earlier layers to overcome this, achieving higher accuracy.

2.3 Effects of Noise

Table 2: Impact of Noise on Classification Performance

#	Noise rate ($P\%$)		**Error**
	Train	**Test**	
a.	0	25	0.1630
b.	0	50	0.2889
c.	0	75	0.4204
d.	25	0	0.0352
e.	50	0	0.0370
f.	75	0	0.0519
g.	25	25	0.1093
h.	50	50	0.1796
i.	75	75	0.2074
j.	denoised form of h.		0.1593

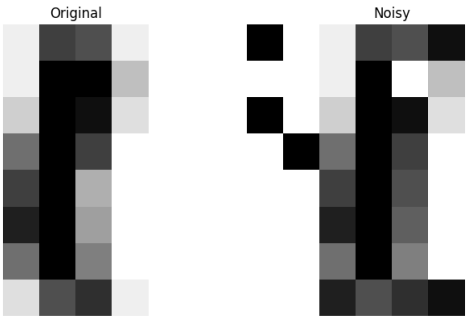

Figure 4: Comparison of Original and Noisy Digits

Observations:

- **Impact of Noise:** As test noise increases (a to c), error rises significantly, confirming MLP struggles with unseen corruption.
- **Data Augmentation Effect:** Training on noisy data (d to f) results in exceptional performance on clean data, suggesting noise acts as data augmentation.
- **Effect of noise on train and test:**Adding noise to the test set increases the error rate more significant compared to train set

3 Ensemble Learning

3.1 Bagging & Pasting

The experiment used a Bagging Classifier with 8 MLP estimators, each trained on 12.5% of the data.

334 out of 540 instances are correctly classified by learner #1
248 out of 540 instances are correctly classified by learner #2
283 out of 540 instances are correctly classified by learner #3
275 out of 540 instances are correctly classified by learner #4
288 out of 540 instances are correctly classified by learner #5
290 out of 540 instances are correctly classified by learner #6
332 out of 540 instances are correctly classified by learner #7
329 out of 540 instances are correctly classified by learner #8

467 out of 540 instances are correctly classified by bagging

Observation: Individual base learners acted as "weak learners" with high variance due to small data samples. The Bagging classifier significantly outperformed every single learner, demonstrating that aggregating high-variance models reduces overall error.

3.2 Boosting

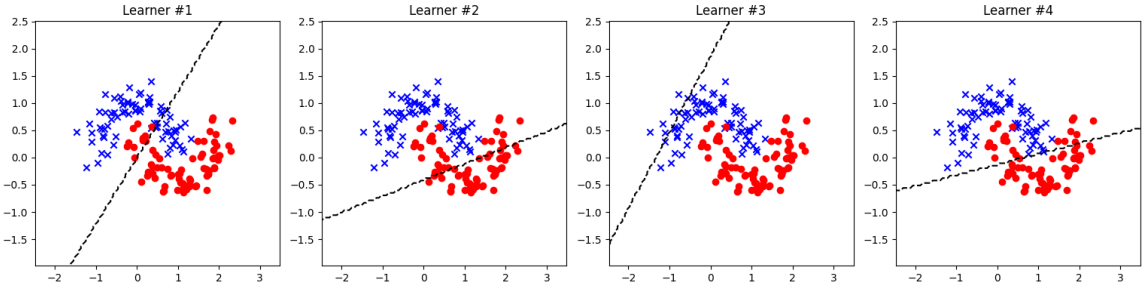

Figure 5: Decision boundaries on binary class classification

Analysis of Figure 5:

1. **Iterative Correction:** The shifting boundaries demonstrate Boosting's focus on previously misclassified instances.
2. **Linear Constraints:** Base learners (Logistic Regression) are linear, preventing perfect separation of the non-linear Moon dataset.
3. **Weakening Boundary:** The decision boundary in Learner 4 becomes flatter and less effective, showing that the linear model struggles to adapt to the increasingly complex, reweighted samples.

3.3 Different Learning Algorithms

Accuracy obtained by Logistic Regression	0.9543
Accuracy obtained by Decision Tree (DT)	0.9173
Accuracy obtained by K-Nearest Neighbors (KNN)	0.9279
Accuracy obtained by(Ensemble)	**0.9561**

3.4 Different Parameter Settings

```
Parameter setting:  l#1 Accuracy:  0.631578947368421
Parameter setting:  l#2 Accuracy:  0.631578947368421
Parameter setting:  l#3 Accuracy:  0.631578947368421
Parameter setting:  l#4 Accuracy:  0.631578947368421
Parameter setting:  l#5 Accuracy:  0.631578947368421
Parameter setting:  l#6 Accuracy:  0.631578947368421
Parameter setting:  l#7 Accuracy:  0.631578947368421
Parameter setting:  l#8 Accuracy:  0.631578947368421
Parameter setting:  l#9 Accuracy:  0.631578947368421
Parameter setting:  l#10 Accuracy:0.9298245614035088
--------------------------------------------------
Ensemble Learning Accuracy:  0.631578947368421
```

- **Bottleneck Failure:** Models 1#1 to 1#9 stuck at ∼63% accuracy (majority class baseline) due to the severe information bottleneck. Only model 1#10 succeeded (92%).
- **Ensemble Limitation:** Ensemble accuracy remained low at 63%. This highlights a critical flaw in Hard Voting: since 9 out of 10 voters were "weak", the single "strong" model was outvoted. For an ensemble to work, the majority must perform better than random guessing.