MALIS Project 2

The Perceptron

PAN Qizhi **BROWN Joel**

20. December 2024

1. Introduction

1.1. Overview

In this project, we developed a Perceptron model from scratch to classify image data of digits 0 through 9. Recognizing that the Perceptron is inherently a binary classifier, we trained ten separate models using a one-vs-rest approach. By integrating these classifiers, our system achieved an overall accuracy of 87.78% and an F1-score of 0.82.

1.2. ChatGPT Usage

We used ChatGPT to accelerate our coding, complete and revise our code. We originally conceived of the solution to the 10-target classification problem and used ChatGPT to help us write the model.

1.3. Contribution

Qizhi: Perception algorithm implement, model construction, training, paragraphs and images in report

Joel: Model analysis, improvement, paragraphs and sheets in report

2. Dataset Examination

The digits dataset from the sklearn.datasets module was used to explore the characteristics of handwritten digit images and their corresponding labels. This dataset contains 1,797 samples, each represented by an 8x8 grayscale image of a digit (0-9). The feature matrix X has a shape of (1797, 64), where each row represents a flattened version of the 8x8 image, while the target array y contains corresponding digit labels, with 10 unique classes (digits 0 through 9).

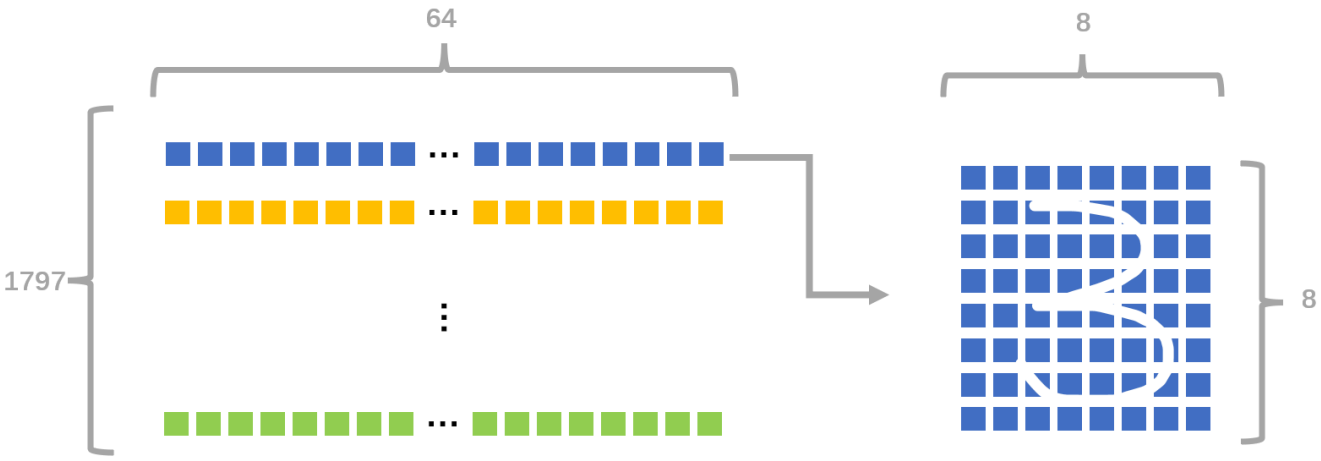

Figure 1: Illustration of digits dataset

Figure 2: Example images of digits

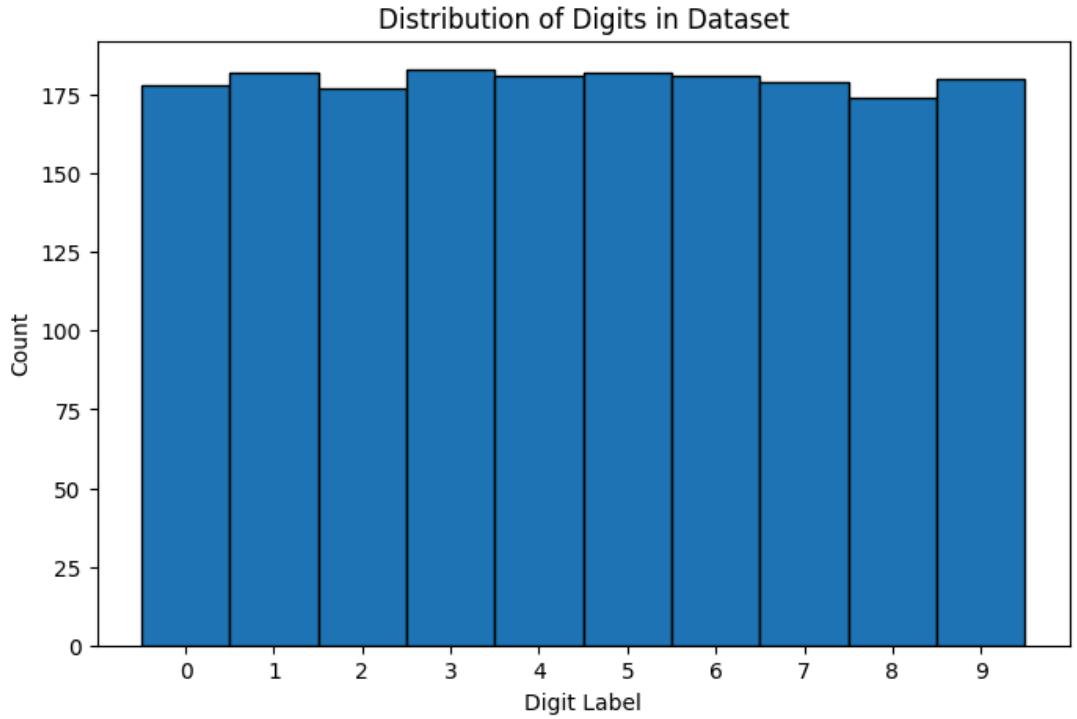

Figure 3: Distribution of labels

Here we display the example images of digits along with their respective labels, showcasing the diversity of handwriting styles. The distribution of target labels revealed a balanced dataset, with all digits well-represented.

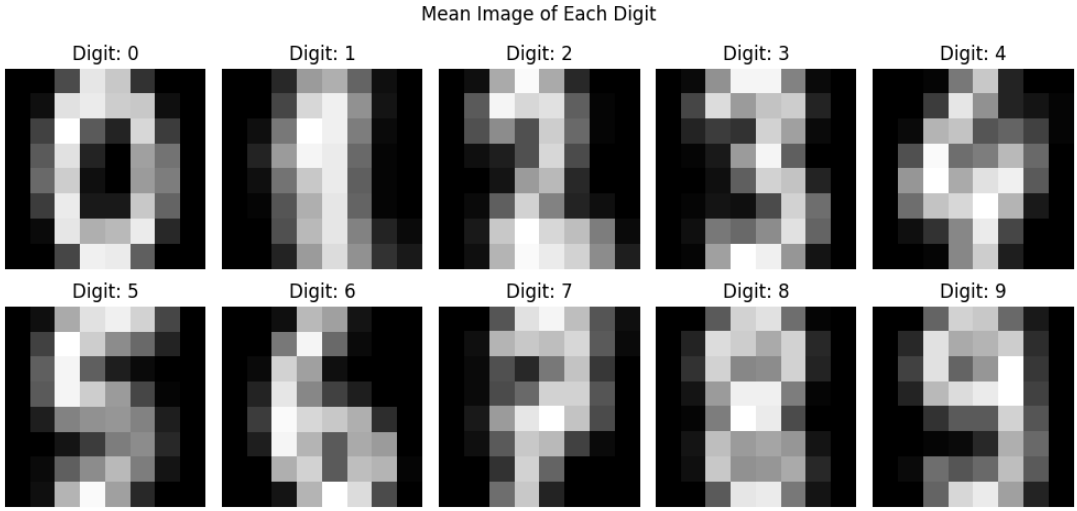

Figure 4: Mean images

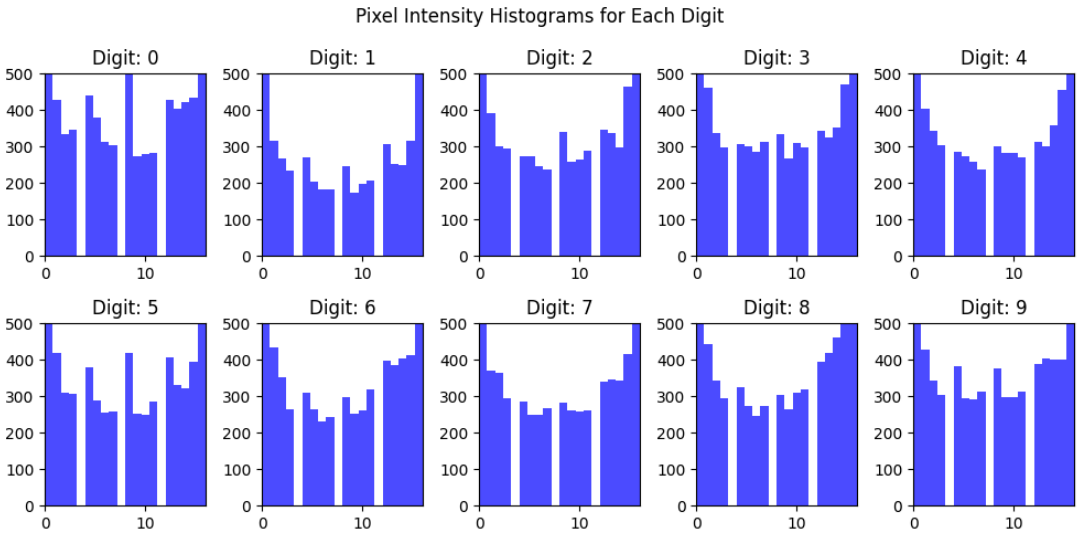

Figure 5: Distribution of pixels in pach label

To further understand the data, mean images of each digit were computed and visualized, highlighting distinctive pixel intensity patterns across different digits. Additionally, histograms of pixel intensities were plotted for each digit, illustrating the variation in brightness and structural differences among samples.

This comprehensive examination underscores the dataset's utility for tasks such as classification and pattern recognition.

3. Solution

3.1. Method

Perceptron is a linear model that determines a decision plane in R^N to classify data into two categories. To extend this binary classifier for a 10-class classification task, we adopt a one-vs-rest approach. Specifically, we train 10 separate models, where each model is tasked with distinguishing one digit (target k) from all the others. The outputs from these 10 binary classifiers are then combined to achieve a final prediction for 10-class classification.

Here's the detailed solution:

1. Transform Labels: For each target class k (k$\in\{0,1,...,9\}$), we convert the original labels into binary labels for a one-vs-rest classification task.
   - Set the target k to 1.
   - Set all other targets to -1.
2. Train Classifier $model_k$:Use the transformed labels in training and validation set to train a perceptron model.
3. Combine and test: Initialize the prediction set(y_Test) to all -1, then collect predictions(y_pred_k) from each classifier and assign values in a certain strategy, which is, for each test sample where y_pred_k is 1 (i.e., the sample belongs to class k) and the sample has not yet been assigned a class in y_Pred (value is still -1), update y_Pred to k.

It's vital to mention that, if we only consider the distribution of these 10 targets, we can say the data is balanced, because each target has similar amount. While in training different classifier, the distributions of each 2 targets(1 and -1) are approximately 1:9, which are totally unbalanced. Simply using accuracy will cause huge problems. Since in our task, telling a target from the rest is more important, we should implement precision, recall and F1-score to measure our model.

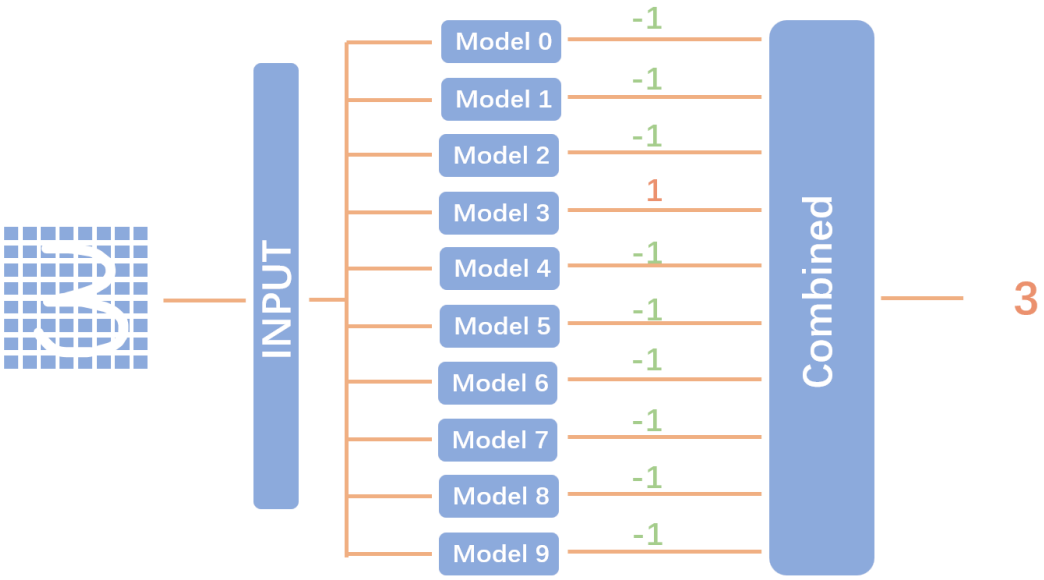

Figure 6: Illustration of our model for the solution

3.2. Results and Analysis

3.2.1. Individual Perceptron Performance

We trained 10 separate binary Perceptron classifiers, each responsible for identifying one digit (0–9) against all others using a one-vs-rest strategy. The performance metrics for each perceptron classifier(trained in $\alpha = 0.01$, epoch $= 50$) are summarized as table 1 and table 2.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-score(%)
0	99.72	100.00	97.22	0.99
1	98.33	94.29	89.19	0.92
2	99.17	92.31	100.00	0.96
3	98.06	89.47	91.89	0.91
4	99.44	97.22	97.22	0.97
5	97.78	85.00	94.44	0.89
6	99.72	100.00	97.22	0.99
7	98.61	89.74	97.22	0.93
8	95.83	77.78	80.00	0.79
9	96.67	78.57	91.67	0.85

Table 1: Performance Metrics for Individual Binary Classifiers in **Validate Set**

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-score(%)
0	99.44	97.22	97.22	0.97
1	96.94	83.78	86.11	0.85
2	99.72	97.22	100.00	0.99
3	98.06	89.47	91.89	0.91
4	98.61	87.80	100.00	0.94
5	99.72	97.37	100.00	0.99
6	99.44	97.22	97.22	0.97
7	99.17	92.31	100.00	0.96
8	93.89	70.97	62.86	0.67
9	97.22	84.21	88.89	0.86

Table 2: Performance Metrics for Individual Binary Classifiers in **Testing Set**

3.2.2. Key Observations

High Performance: Models for digits 0, 2, 4, 5, 6, and 7 had very high accuracy and F1-scores (≥ 0.94), indicating effective classification.

Challenges: Models for digits 1, 8 and 9 showed lower recall and F1-scores, especially digit 8 (0.63 for recall, 0.67 for F1-score), suggesting difficulties in correctly identifying this digit.

Average Metrics: The individual classifiers achieved an average accuracy of 98.22% and an average F1-score of 0.91, demonstrating strong overall performance despite some class-specific challenges.

3.2.3. Combined Perception Model

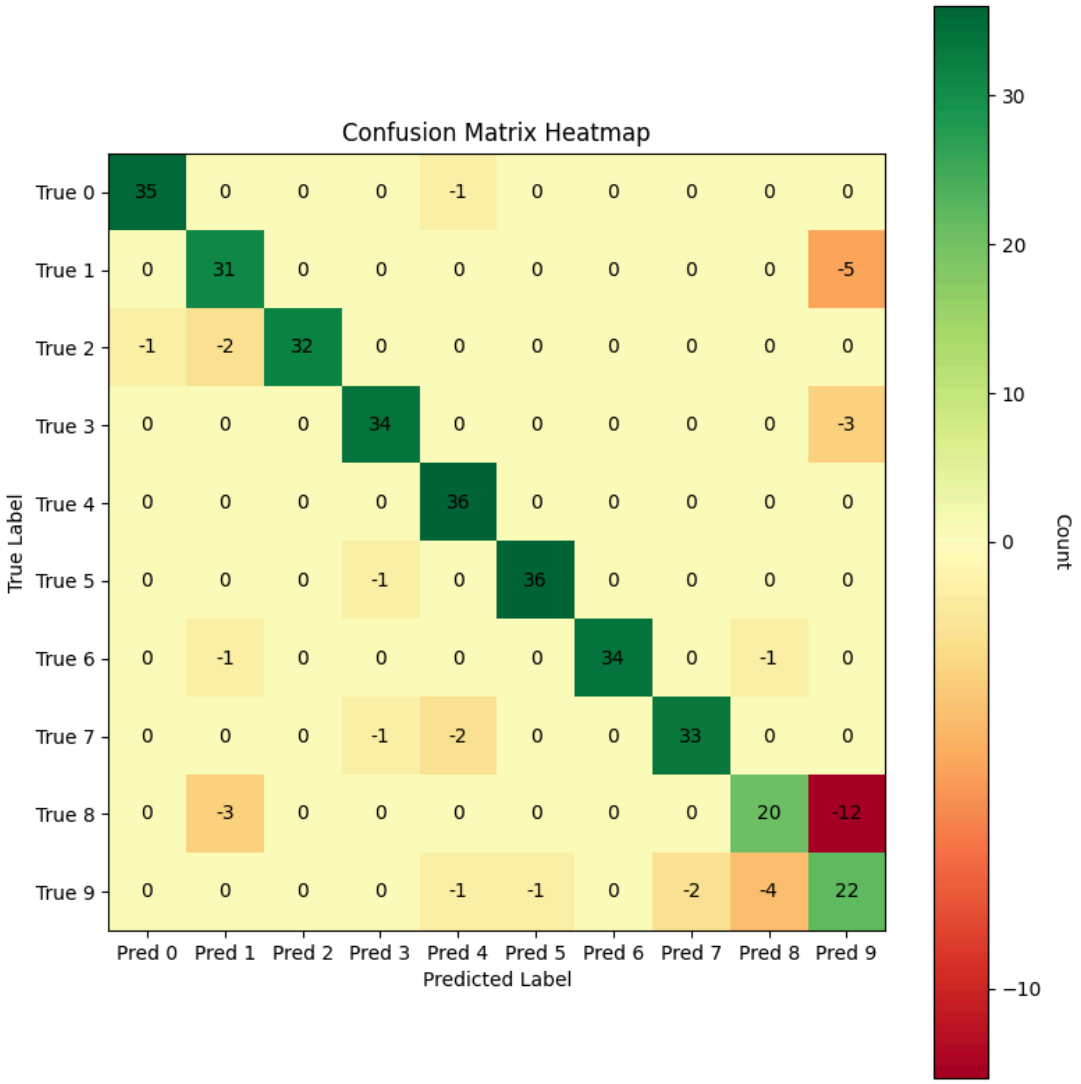

Figure 7: Combined Perception Model Results. Green areas mean the correct prediction and red areas mean the wrong prediction. The more counts, the deeper color. The dark red '−12' means there are 12 labels supposed to be 8 but predicted to be 9

Metric	Value
Testing Accuracy	87.78%
Combined F1-score	0.82

Table 3: Combined Multi-Class Model Performance

We obtained an overall accuracy of 87.78% and an F1-score of 0.82 on the test set. This demonstrates that the perceptron can capture and distinguish patterns in the digit images which enables the model to classify them but it's nothing near perfect.

While individual classifiers achieved high accuracies and F1-scores, the combined model's performance saw a significant drop. This maybe primarily due to the simplistic strategy of assigning a digit label based on the first positive prediction (Further discussed in 3.2).

4. Discussion

4.1. Model Metrics

The core principle of the perceptron is to find a separating hyperplane that distinguishes between two classes. However, since each individual model is only trained to separate one class from nine others, the problem becomes **imbalanced**: only about 1/10 of the samples are positive (the target digit) and 9/10 are negative (the rest of the digits). In this case, a high accuracy can be misleading because a classifier could achieve high accuracy by predominantly predicting the negative class.

Accuracy alone can be misleading in such imbalanced scenarios. For instance, a classifier might have a high accuracy by mostly predicting the negative class, yet fail to correctly identify the positive instances (digits 1 and 8 in our case). Therefore, we incorporated **precision, recall, and F1-score** to provide a better evaluation of each classifier's performance.

4.2. Model Efficacy

4.2.1. Individual Model Strengths and Weaknesses

4.2.1.1. High Accuracy and F1 scores

Most classifiers demonstrated outstanding performance, with accuracies and F1-scores close to 1. This indicates that the Perceptron effectively learned the decision boundaries for these digits, correctly distinguishing them from others.

4.2.1.2. Challenges with Specific Classes

However, The classifiers for digits 1, 8 and 9 had a lower recall indicates that the model often failed to recognize these digits when they were present. This could be due to:

- Feature Overlap: Digits 1, 8 and 9 might share more features with other digits, making them harder to classify with a linear model.
- Inherent Complexity: The variability in handwriting for these digits could introduce inconsistencies that the Perceptron couldn't fully capture.

4.2.2. Combined Model Strengths and Weaknesses

4.2.2.1. Computational Efficiency

The combined perceptron is simple making it computationally efficient and easy to implement. This makes it suitable for educational purposes and scenarios with linear separability while still yielding high performance results on classes with distinct features (e.g., 0, 4).

4.2.2.2. First Bias Limitation

Our simple approach was to combine the 10 binary models by checking which classifier strongly predicts a high enough certainty for a given sample. However, the first classifier to assert a sample as positive determines the final predicted digit. While this method worked to produce a fair enough multi-class output, it is quite simple and does not fully leverage more sophisticated rules. This simplistic combination strategy may assign a digit as soon as one classifier says "yes" which can potentially ignore other classifiers that might be more certain.

Similarly, more complex datasets like handwritten digits may not necessarily be linearly seperable and would fail using a perceptron model.

4.3. Suggestions

4.3.1. Use of more sophisticated models

While the Perceptron serves as an excellent introductory linear classifier, exploring more advanced machine learning models such as SVM's might improve classification accuracy and handle the complexities of multi-class problems more effectively.

4.3.2. Feature Engineering

Improving the quality and representativeness of the input features can lead to better model performance. Effective feature engineering can make the data more amenable to classification by highlighting the most distinct aspects of each digit.

4.3.3. Model Selection

Although we chose $\alpha = 0.01$, epoch $= 50$ to train our model for it showed a relatively high performance in our several attempts, we didn't do the model selection for the best model parameters. We could do the K-fold for cross validation to choose the best α, epoch and regularization in future improvement.

5. Conclusion

In this project, we successfully implemented a Perceptron model from scratch and applied it to the task of handwritten digit classification using the one-vs-rest strategy. The Perceptron demonstrated strong performance for most digit classes, achieving an average accuracy of 87.78% and an average F1-score of 0.82 across individual binary classifiers. These results highlight the Perceptron's effectiveness in handling binary classification tasks with distinct and separable features. We are more familiar with the classification problems and more skillful in implementing the machine learning algorithms.

A. Appendix

– Project Repo: https://github.com/RizPur/MALIS-Project/tree/main/