Digit Classification Using the Pocket Perceptron Learning Algorithm

Name: Serge Alhalbi

Institution: The Ohio State University

Course: Advanced Machine Learning for Remote Sensing – ECE 7402

Project: 1

Instructor: Dr. Rongjun Quin

Date: 03/18/2025

ABSTRACT

This project explores the classification of handwritten digits using the Perceptron Learning Algorithm enhanced with the Pocket strategy for improved convergence and generalization. Two distinct feature extraction methods are investigated: a low-dimensional combination of symmetry and intensity, and a high-dimensional representation using Histogram of Oriented Gradients (HOG). To handle multi-class classification, both One-vs-All and One-vs-One strategies are implemented, leveraging binary perceptrons for pairwise or individual class discrimination. The system is evaluated on a dataset of digit images, with performance metrics including overall accuracy, class-specific accuracy, producer's and user's accuracy, and the Kappa coefficient. Experimental results demonstrate the comparative effectiveness of the feature sets and classification strategies, while visualizations of decision boundaries and confusion matrices provide insights into classifier behavior. The modular implementation facilitates extensibility for testing additional feature sets and classification schemes, highlighting the flexibility and effectiveness of the Pocket Perceptron in multi-class digit recognition tasks.

Methodology

The Perceptron Learning Algorithm was implemented using the Pocket strategy, as described in the following pseudocode:

```
Initialize:
$w \leftarrow random\ vector\ in\ \mathbb{R}^{d+1}$
$w_{pocket} \leftarrow w$
$A_{best} \leftarrow 0$

Repeat for $t = 1$ to $T_{max}$:

1. Compute predictions:
$\hat{y}_i = sign(w^T x_i)$

2. Find misclassified samples:
$M = \{i | \hat{y}_i \neq y_i\}$

3. If $M = \emptyset$, terminate.

4. Choose a random misclassified point:
$i^* \sim Uniform(M)$

5. Update weights:
$w \leftarrow w + \eta y_i x_i$

6. Evaluate accuracy with updated weights:
$A = \frac{1}{n} \sum_{i=1}^{n} 1[sign(w^T x_i) = y_i]$

7. If $A > A_{best}$:
$w_{pocket} \leftarrow w, A_{best} \leftarrow A$

Return $w_{pocket}$
```

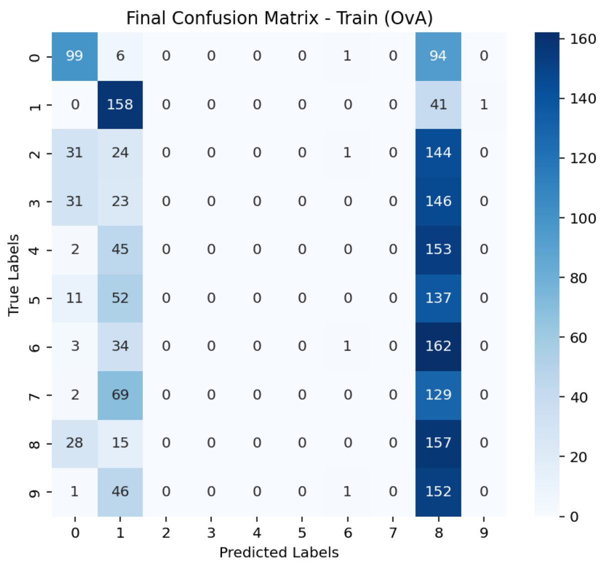
RESULTS

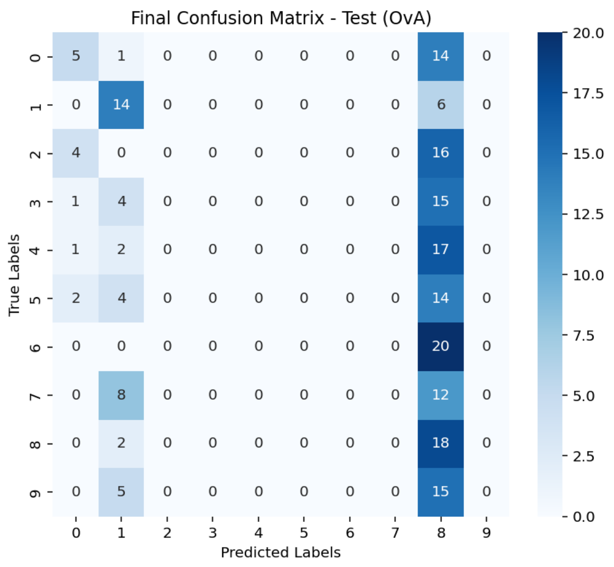
Symmetry and Intensity Results, One vs All:

Table:

Metric		Symmetry and Intensity, One vs All
Training	Accuracy	0.2075
	Class Accuracy	[0.495 0.79 0. 0. 0. 0. 0.005 0. 0.785 0.]
	Recall	[0.495 0.79 0. 0. 0. 0. 0.005 0. 0.785 0.]
	Precision	[0.47596154 0.33474576 0. 0. 0. 0. 0.25 0. 0.11939163 0.]
	Kappa	0.1194
Testing	Accuracy	0.185
	Class Accuracy	[0.25 0.7 0. 0. 0. 0. 0. 0. 0.9 0.]
	Recall	[0.25 0.7 0. 0. 0. 0. 0. 0. 0.9 0.]
	Precision	[0.38461538 0.35 0. 0. 0. 0. 0. 0. 0.12244898 0.]
	Kappa	0.0944

Confusion Matrix:





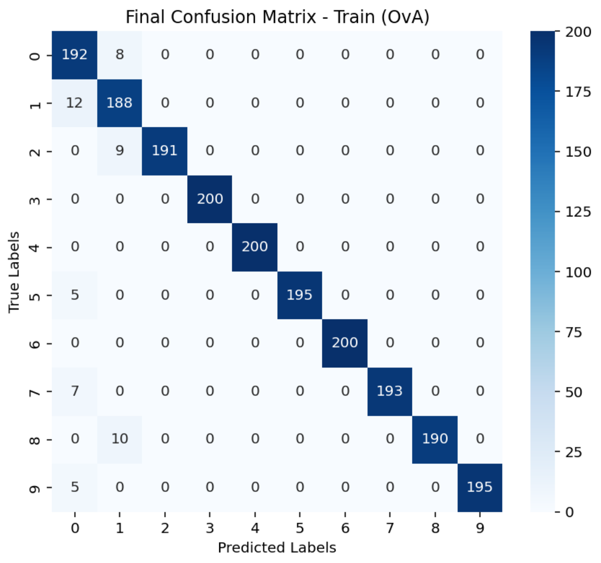
Testing2 folder results:

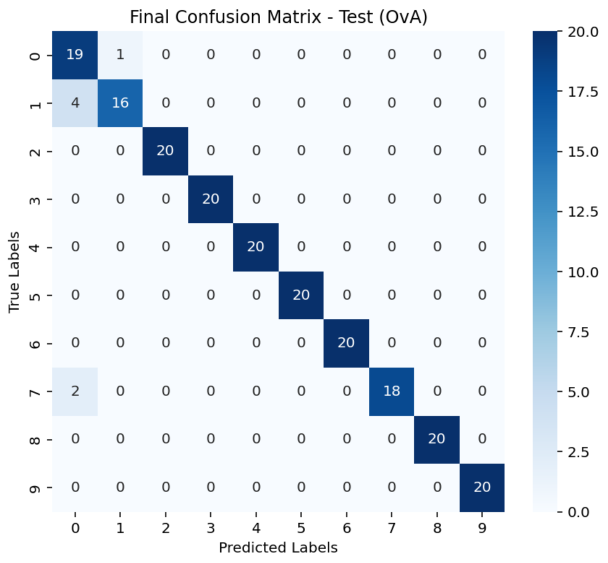
Accuracy: 0.1943

Symmetry and Intensity Results, One vs One:
Table:

Metric		Symmetry and Intensity, One vs One
Training	Accuracy	0.972
	Class Accuracy	[0.96 0.94 0.955 1. 1. 0.975 1. 0.965 0.95 0.975]
	Recall	[0.96 0.94 0.955 1. 1. 0.975 1. 0.965 0.95 0.975]
	Precision	[0.86877828 0.8744186 1. 1. 1. 1. 1. 1. 1. 1.]
	Kappa	0.9689
Testing	Accuracy	0.965
	Class Accuracy	[0.95 0.8 1. 1. 1. 1. 1. 0.9 1. 1.]
	Recall	[0.95 0.8 1. 1. 1. 1. 1. 0.9 1. 1.]
	Precision	[0.76 0.94117647 1 1. 1. 1. 1. 1. 1. 1. 1.]
	Kappa	0.9611

Confusion Matrix:





Testing2 folder results:

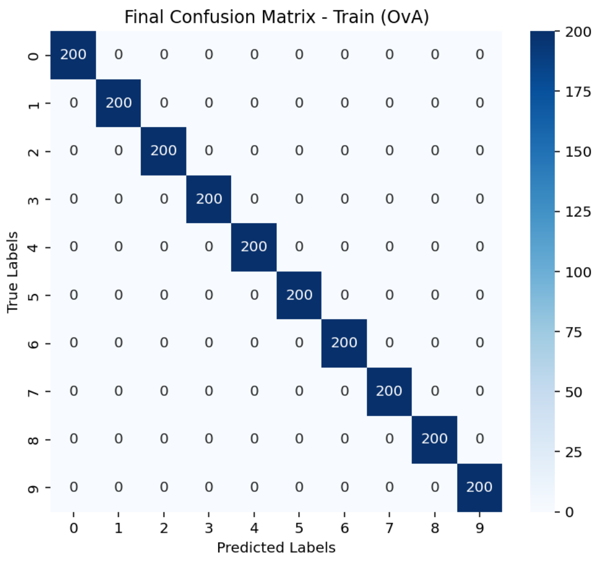
Accuracy: 0.9714

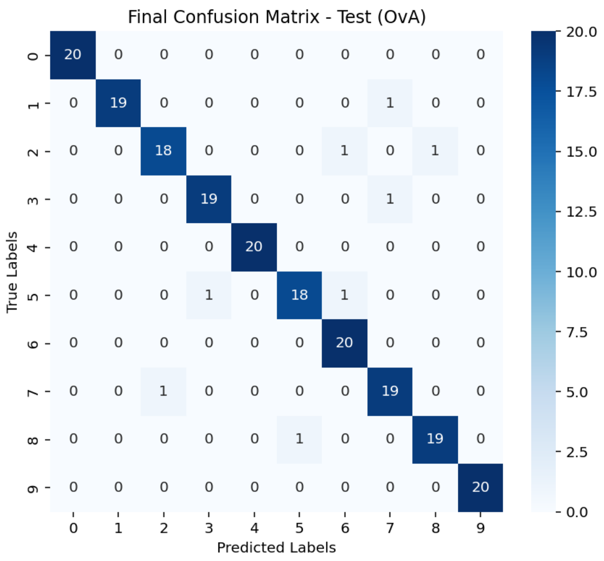
HOG, One vs All:

Table:

Metric		HOG, One vs All
Training	Accuracy	1
	Class Accuracy	[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
	Recall	[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
	Precision	[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
	Kappa	1
Testing	Accuracy	0.96
	Class Accuracy	[1. 0.95 0.9 0.95 1. 0.9 1. 0.95 0.95 1.]
	Recall	[1. 0.95 0.9 0.95 1. 0.9 1. 0.95 0.95 1.]
	Precision	[1. 1. 0.94736842 0.95 1. 0.94736842. 0.90909091 0.9047619 0.95 1.]
	Kappa	0.9556

Confusion Matrix:





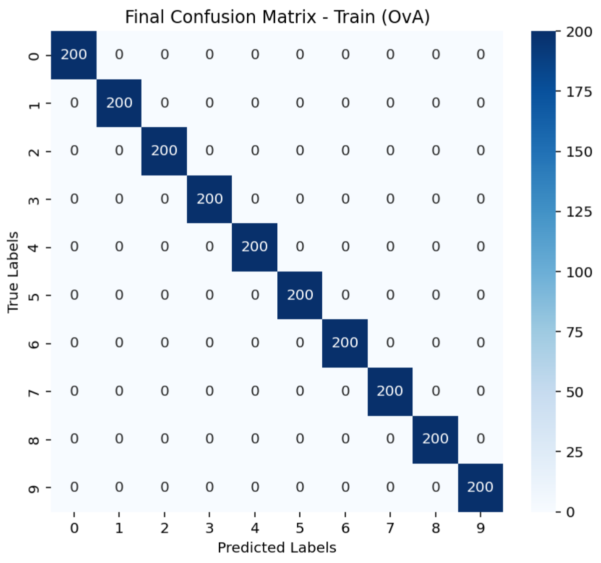
Testing2 folder results:

Accuracy: 0.9714

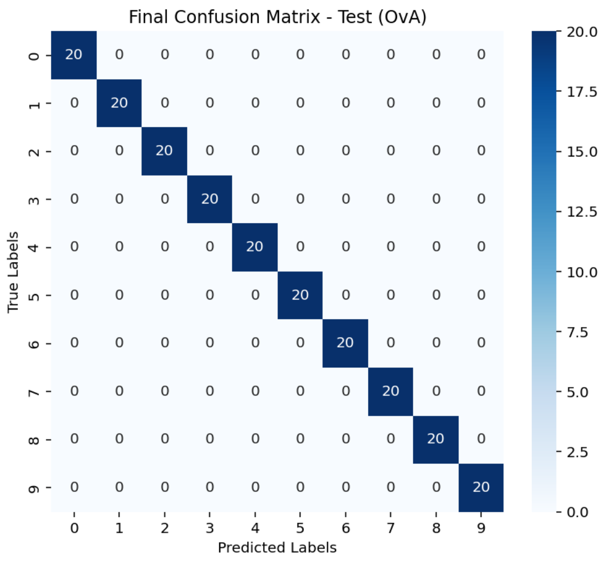
HOG, One vs One:
Table:

Metric		**HOG, One vs One**
Training	**Accuracy**	1
	Class Accuracy	[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
	Recall	[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
	Precision	[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
	Kappa	1
Testing	**Accuracy**	1
	Class Accuracy	[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
	Recall	[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
	Precision	[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
	Kappa	1

Confusion Matrix:



True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	200	0	0	0	0	0	0	0	0	0
1	0	200	0	0	0	0	0	0	0	0
2	0	0	200	0	0	0	0	0	0	0
3	0	0	0	200	0	0	0	0	0	0
4	0	0	0	0	200	0	0	0	0	0
5	0	0	0	0	0	200	0	0	0	0
6	0	0	0	0	0	0	200	0	0	0
7	0	0	0	0	0	0	0	200	0	0
8	0	0	0	0	0	0	0	0	200	0
9	0	0	0	0	0	0	0	0	0	200



True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	20	0	0	0	0	0	0	0	0	0
1	0	20	0	0	0	0	0	0	0	0
2	0	0	20	0	0	0	0	0	0	0
3	0	0	0	20	0	0	0	0	0	0
4	0	0	0	0	20	0	0	0	0	0
5	0	0	0	0	0	20	0	0	0	0
6	0	0	0	0	0	0	20	0	0	0
7	0	0	0	0	0	0	0	20	0	0
8	0	0	0	0	0	0	0	0	20	0
9	0	0	0	0	0	0	0	0	0	20

Testing2 folder results:

Accuracy: 1

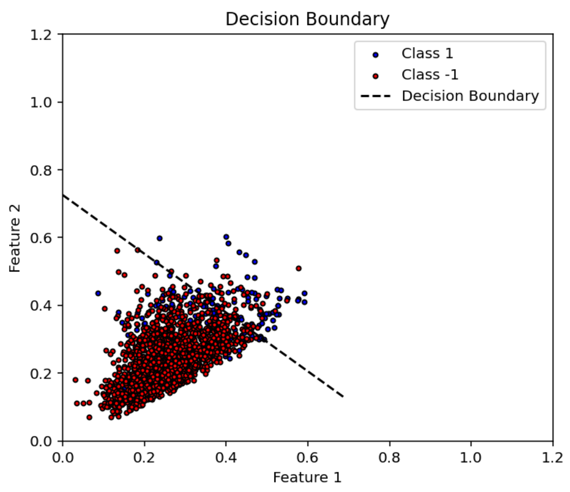
Analysis

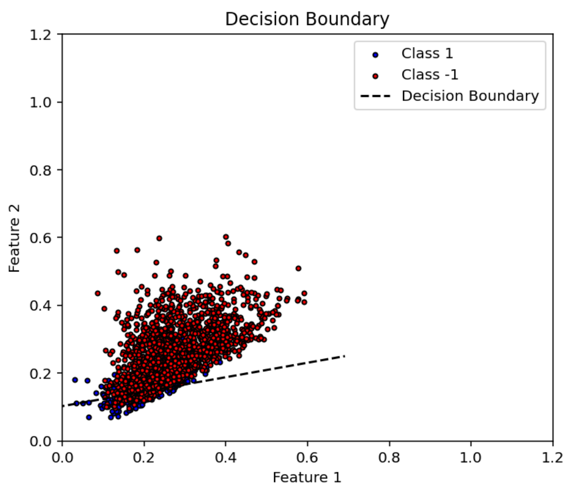
Symmetry and Intensity Analysis, One vs All:

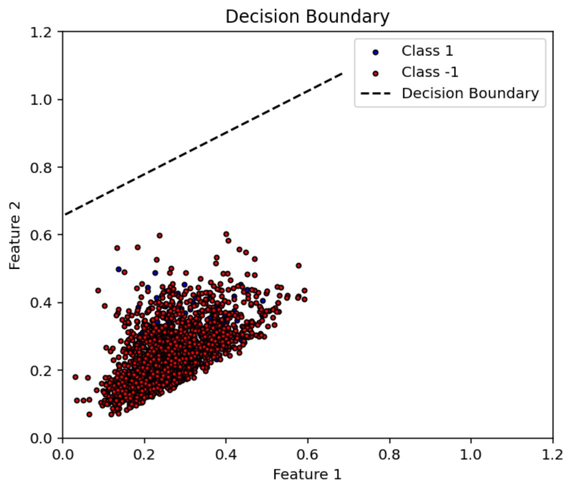
We observe that although the accuracy appears very high for each classifier during training in the one-vs-all setup (not during final evaluation), the overall accuracy and other evaluation metrics are poor. There are two straightforward explanations for this.

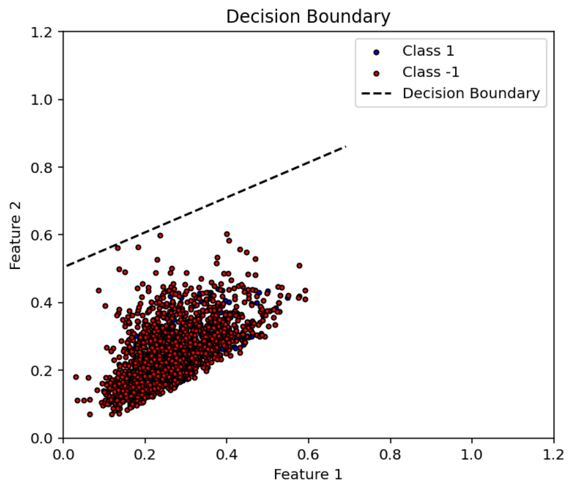
First, the data is imbalanced, with a 1:9 ratio, meaning the positive (blue) examples are nine times fewer than the negative examples. As a result, the decision boundary may appear reasonable in some cases (as shown in the first two images, which correspond to the classifiers for digits 0 and 1), but in reality, it tends to neglect the positive class entirely. The classifier often predicts everything as negative, placing nearly all data on one side of the boundary (as seen in the third and fourth images, which correspond to the classifiers for digits 2 and 3).

Second, the data in most of the one-vs-all classifiers is not linearly separable, which further limits the performance of the perceptron and leads to poor generalization.



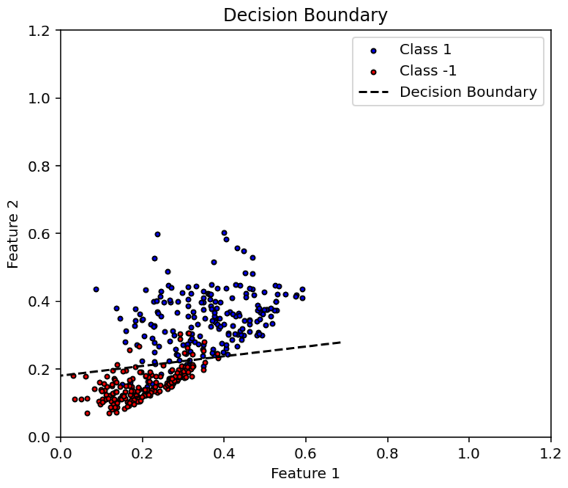


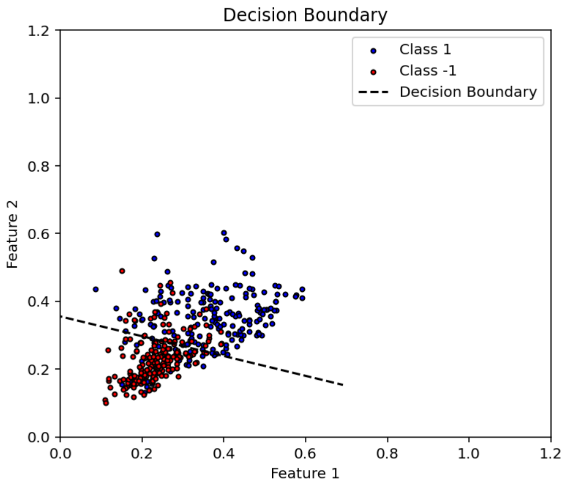




Symmetry and Intensity Analysis, One vs All:

For the same feature set, the one-vs-one strategy effectively addresses both the data imbalance and linear separability issues. This is evident in the images below, which show the classifiers for digits 0 vs 1 and 0 vs 4. These classifiers achieve very high accuracy and near-perfect evaluation metrics.





HOG Analysis:

Although we can no longer visualize the data or the decision boundary with the higher-dimensional feature set, the performance improves significantly, achieving excellent results for both the one-vs-all and one-vs-one strategies.

CONCLUSION

- Pocket Perceptron Stability:
    - The pocket strategy significantly improved the Perceptron's robustness by retaining the best-performing weight vector throughout training, especially helpful in non-linearly separable datasets.

- One-vs-All vs One-vs-One:
    - One-vs-All suffered in some cases due to class imbalance and poor separability between positive and negative classes.
    - One-vs-One generally outperformed One-vs-All by simplifying the classification task to distinguish between only two classes at a time.

- Feature Set Comparison:
    - The low-dimensional feature set (symmetry + intensity) enabled easier visualization of decision boundaries, highlighting limitations in linear separability.
    - The high-dimensional HOG features, though not directly visualizable, significantly improved classifier performance for both strategies.

- Performance Metrics:
  High accuracy, precision, recall, and Kappa scores were achieved using HOG features, especially with the One-vs-One approach.

- Modular Design:
  The pipeline was designed in a modular fashion, making it easy to plug in new feature extractors, classifiers, or evaluation metrics.