UNIVERSITÀ
DEGLI STUDI
FIRENZE

Department of Information Engineering

Author: Awad Elia ***Student ID:*** 7140815

Multiclass Perceptron: OVA vs AVA

Contents

1 Introduction

This project aims to implement and make an analysis on the Perceptron algorithm for multiclass classification on the MNIST dataset. Two main decomposition strategies were explored: **One-Vs-All (OVA)** and **All-Vs-All (AVA)**. The analysis is framed within the context of the thesis proposed by Rifkin and Klautau (2004), which defends the efficacy of the simpler OVA scheme. In the paper it is clear they want to refute that the use of complex strategies for multiclass classification necessarily leads to much better results than classical and intuitive strategies such as the OVA. In fact it is shown that if the underlying binary classifier is strong (such as a well tuned SVM), the OVA scheme is much less expensive and at the same time achieves results that are practically identical to those of more complex strategies. In fact, the goal of this experiment is to show how using a weak binary classifier such as the Rosenblatt's Perceptron (1957) without any hidden layers or other sophisticated optimization techniques, shows better results with the more complex AVA strategy, compared to the OVA one, precisely because there is a very weak binary classifier underneath.

2 Methodology

2.1 Binary Perceptron

The core component is the Rosenblatt's Perceptron which was implemented following the explanation described in Cristianini & Shawe-Taylor 1999. The perceptron receives a training set:

$$S = \{(x_i, y_i)\}_{i=1}^{l} \subseteq (XY)^l, y_i \in \{-1, 1\}$$

The decision function is:

$$f(x) = w^T \cdot x + b$$

Since the MNIST dataset is not linearly separable, the algorithm's convergence is not assured, which is why, as implemented in `binary_perceptron.py`, we follow the standard rule to ensure convergence, which consists of setting a limit of iterations. The weights and biases are updated as described in Cristianini & Shawe-Taylor 1999:

$$w_{k+1} = w_k + y_i x_i$$

$$b_{k+1} = b_k + y_i R^2$$

where R is the radius of the smallest sphere that contains all data, which is calculated as the maximum norm of the input vectors.

2.2 Multiclass Strategies

In this section we are going to explore how the two strategies work.

- **One-Vs-All (OVA)**: This strategy consists of training K binary classifiers, where K is the number of classes in the training set. In the case of handwritten digits (MNIST dataset), we train $K = 10$ models, each one on recognizing a specific digit against the collection of all other remaining digits (e.g., Digits $\{0\}$ vs. Digits $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$). The prediction is performed by running all N classifiers and choosing the class with the largest output value.

- **All-Vs-All (AVA)**: This other strategy, also known as "all-pairs" classification, consists of training a separate binary classifier for every possible pair of classes. The number of required classifiers is given by the formula

$$\binom{K}{2} = \frac{K(K-1)}{2}$$

  The final prediction is made by following a voting system, which involves selecting the class that receives the most votes across all pairwise challenges.

3 Implementation and Code Documentation

The project is structured into Python classes to ensure a clear separation between the underlying binary classifier and the multiclass classification strategies.

3.1 BinaryPerceptron

The `BinaryPerceptron` class implements the classic Rosenblatt algorithm explored previously in the report. The only thing that may need further explanation is the training loop. The `train` method implements an epoch-based loop. It terminates either when the data is perfectly separated (`n_errors` $= 0$) or when the `max_epochs` limit (set to 100 on default) is reached to prevent infinite loops on non-linearly separable data.

3.2 One-Vs-All Implementation (OVA)

The `MulticlassOVAPerceptron` class manages the N binary models required for the OVA strategy.

- **Label Binarization**: During training, the class utilizes `np.where` to create a temporary "mask," transforming the multiclass problem into a binary one where the digit of interest is $+1$ and all other digits labels are replaced with -1.
- **Confidence-Based Prediction**: Instead of using binary predictions, the `predict` method calls the `decision_function` of each perceptron. This returns the raw score $w^T x + b$, and the class with the highest score is chosen, picking in this way the "most confident" classifier.

3.3 All-Vs-All Implementation (AVA)

The `MulticlassAVAPerceptron` class implements the $\frac{K(K-1)}{2}$ pairs challenges strategy.

- **Pairs Initialization**: The class uses a dictionary to store the $\frac{K(K-1)}{2} = 45$ distinct perceptrons, indexed by tuples (i, j) where $i < j$. This avoids redundant pairs and self-comparison.
- **Data Filtering**: For each pair (i, j), a boolean mask is applied to the training set to extract only the examples belonging to those two classes.
- **Voting Scheme**: The `predict` method implements a voting array. Each perceptron provides a binary vote ($+1$ or -1), and the final prediction will be the index in the voting array with the most accumulated votes.

3.4 Preprocessing and Execution (main.py)

The `main.py` file handles data processing from the MNIST dataset provided in the `.arff`, performance measurements and results visualization.

- **Normalization**: Input pixels are scaled from $[0, 255]$ to $[0, 1]$ to improve the convergence of the perceptron algorithm by avoiding hyperplane overshooting during the update of weights and bias.
- **Data Shuffling**: To ensure that after the dataset is divided into training set and test set, there are examples in the training set for all classes, the data is first shuffled by obtaining a random permutation of the indexes using the `np.random.permutation` function, which is used with a fixed seed (42) for reproducibility.
- **Performance Metrics**: The measurements regard
  - **Time Performances**: obtained using the `time.perf_counter` function
  - **Accuracy**: calculated at the end of the `train` method of both AVA and OVA classes.
  - **Confusion Matrices**: also calculated inside each `train` method of AVA and OVA classes, and then plotted in `main.py` using `Matplotlib` and `Seaborn` libraries.

4 Experimental Results

The results below summarize the performance metrics:

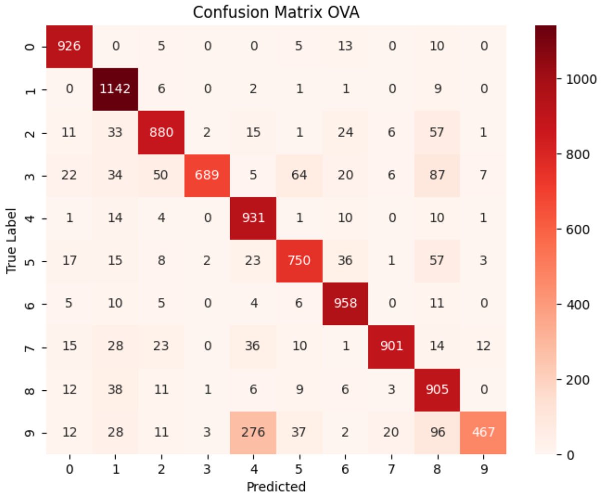
4.1 Time Performances

Table 1: Time and accuracy performances

Model	Training Time (s)	Test Time (s)	Average Prediction Time (s)	Accuracy (%)
OVA	47.75	0.08	0.000008	85.49
AVA	40.54	0.32	0.000032	91.59

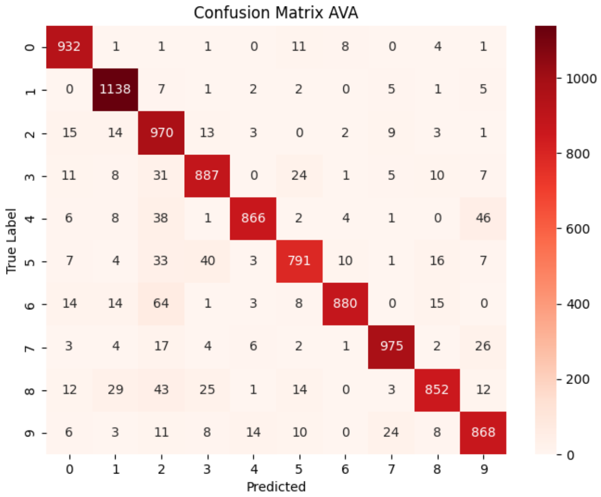
4.2 Confusion Matrices

The confusion matrices allow us to visualize the errors of the two models across all digits classifications. As shown in Figure 1, the OVA strategy highlights specific patterns of missclassification.



	0	1	2	3	4	5	6	7	8	9
0	926	0	5	0	0	5	13	0	10	0
1	0	1142	6	0	2	1	1	0	9	0
2	11	33	880	2	15	1	24	6	57	1
3	22	34	50	689	5	64	20	6	87	7
4	1	14	4	0	931	1	10	0	10	1
5	17	15	8	2	23	750	36	1	57	3
6	5	10	5	0	4	6	958	0	11	0
7	15	28	23	0	36	10	1	901	14	12
8	12	38	11	1	6	9	6	3	905	0
9	12	28	11	3	276	37	2	20	96	467

Figure 1: Confusion Matrix for the One-Vs-All (OVA) strategy.



	0	1	2	3	4	5	6	7	8	9
0	932	1	1	1	0	11	8	0	4	1
1	0	1138	7	1	2	2	0	5	1	5
2	15	14	970	13	3	0	2	9	3	1
3	11	8	31	887	0	24	1	5	10	7
4	6	8	38	1	866	2	4	1	0	46
5	7	4	33	40	3	791	10	1	16	7
6	14	14	64	1	3	8	880	0	15	0
7	3	4	17	4	6	2	1	975	2	26
8	12	29	43	25	1	14	0	3	852	12
9	6	3	11	8	14	10	0	24	8	868

Figure 2: Confusion Matrix for the All-Vs-All (AVA) strategy.

5 Analysis and Discussion

The experimental results provide a clear perspective on the multiclass classification problem when handled by a "weak" binary learner like a basic Rosenblatt's Perceptron. The data obtained from the time measurements (Table: 1) and the visualized errors in the confusion matrices (Figures: 1 and 2) allow us to make a series of observations in favor of the thesis proposed by Rifkin and Klautau (2004).

5.1 Accuracy analysis

We can observe a significant performance gap between the two strategies: the **AVA strategy achieved 91.59% accuracy**, overcoming the **OVA strategy at 85.49%**. In the paper, Rifkin and Klautau argue that OVA is as accurate as any other approach provided that the underlying binary classifiers are optimized, well-tuned and therefore strong learners such as SVMs. However, in our experiment, we use a simple perceptron without any hidden layers or other type of optimization, thus representing a weak learner. As the paper says, when weak learners are used, more sophisticated methods like AVA can better solve the independence of errors improving the overall accuracy.

The confusion matrices underline this discrepancy. In the OVA confusion matrix (Figure 1), we observe a critical failure in classifying the digit 9, which is correctly identified only 467 times and is misclassified as 4 (276 times) or 8 (96 times). This information suggests that the OVA Perceptron struggles to find a single hyperplane capable of separating the digit 9 from the "mixture" of everything that is not 9. On the other side, the AVA matrix (Figure 2) shows a much more balanced performance; the digit 9 is correctly identified 868 times. The AVA Perceptron solves this problem because it does not exploit a single binary classifier but exploits all the 45 hyperplanes decomposing the problem into smaller ones (e.g. 9 vs 1, 9 vs 6), thus identifying a more defined boundary between the digit 9 and the others.

5.2 Computational analysis

The time performance metric reveals an interesting trade-off between training and testing efficiency.

- **Training Time**: The AVA strategy trained faster (40.54 s) than OVA (47.75 s). This happens because, while AVA trains 45 classifiers versus OVA's 10, each AVA perceptron only processes a small subset of the data belonging to two specific classes corresponding to the challenge of the two specific digits of interest. Since the perceptron is superlinear relatively to the number of samples, therefore training many models on small datasets is more efficient than training fewer models on the whole dataset.

- **Testing Time**: The OVA strategy is significantly faster during the inference phase, with an average prediction time of 0.000008 s compared to AVA's 0.000032 s. This is because in both strategies, we have to evaluate $w^T x + b$ for each perceptron, therefore OVA strategy is faster because it has to calculate it for 10 perceptron, compared to the 45 required by the AVA strategy.

6 Conclusion

This project successfully demonstrated the implementation of Multiclass Perceptron strategies on the MNIST dataset. The results confirm that the choice of multiclass decomposition is heavily dependent on the strength of the underlying binary learner.

While Rifkin and Klautau effectively defend the One-Vs-All scheme as a simple and powerful tool when paired with strong classifiers like SVMs, our experiment highlights its limitations when using a basic Rosenblatt's Perceptron. The 91.59% accuracy of AVA compared to the 85.49% of OVA proves that for weak learners, the All-Vs-All strategy works better giving off better results by decomposing the geometric complexity of classes boundaries.

In conclusion, OVA remains the superior choice if a strong classifier can be used. However, if computational constraints or model simplicity limit the developer to use weak learners, the AVA strategy offers a necessary boost in accuracy and training efficiency, even if prediction takes more time.

References

[1] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*, 1999.

[2] R. Rifkin and A. Klautau. *In Defense of One-Vs-All Classification*, 2004.