Department of Electrical and Electronics Engineering

EEE 443: Neural Networks

Class Project III Report

Name and Surname: Ali Aral Takak

Student ID: 22001758

Department: EEE

Instructor: Erdem Koyuncu

Problem

The third project of the course aims to implement and analyze the Multicategory Perceptron Algorithm (PTA) for handwritten digit classification using the MNIST dataset. The objective is to train a perceptron model to classify digits (0-9) based on their pixel representations. The dataset consists of grayscale images of size 28×28, which are vectorized into 784-dimensional feature vectors.

The perceptron model is trained by iteratively updating a weight matrix of size 10×784 using a supervised learning approach. The algorithm is evaluated by varying the training sample size n, the learning rate η, and the misclassification tolerance ϵ. The performance of the perceptron is then assessed by analyzing the training convergence and test misclassification rate under different experimental values of sample size, learning rate, and misclassification tolerance. In order to ensure that the system works robustly, a maximum epoch limit is also implemented to handle cases where the algorithm does not converge.

Learning Algorithm

The multicategory perceptron algorithm is a supervised learning method designed for multiclass classification. It is an extension of the perceptron training algorithm that updates a weight matrix W to classify input samples into one of the ten-digit classes (0-9). In this scenario, each training image is represented as a flattened 784-dimensional vector and passed through the perceptron model, which computes an induced local field v for each class. The class with the highest activation is selected as the predicted label. If the prediction is incorrect, the weights are updated using the perceptron learning rule. The steps provided below describes how the algorithm works:

1. Initialize the weight matrix W randomly.
2. For each epoch:
    a. For each training sample:
        i. Compute the induced local field $v = Wx$
        ii. Determine the predicted class y_{pred} by selecting the index with the highest activation $y_{pred} = argmax(v)$.
        iii. Compare y_{pred} with the true label y_{true}.
        iv. If misclassified, update the weight matrix using the Perceptron Learning Rule $W = W + \eta(d(x) - u(Wx))x^T$, where $d(x)$ is the one-hot encoded true label and $u(Wx)$ applies the step function component-wise.
3. Repeat until the misclassification rate falls below a threshold ϵ or maximum epochs are reached.
4. Use the trained W to classify test samples and compute the misclassification rate.

Implementation and Results

We started our implementation by loading the MNIST dataset and converting them into PyTorch tensors. Each grayscale image is normalized and flattened into a 784-dimensional vector. Labels are stored as integer class labels. The figure provided below displays the dimensions, in order to ensure that the dataset samples are loaded properly:

```
Training images shapes and dimensions:  (60000, 28, 28)
Training labels shapes and dimensions:  (60000,)
Testing images shapes and dimensions:  (10000, 28, 28)
Testing labels shapes and dimensions:  (10000,)
2025-02-20 18:49:09.423 python[72561:2669967] +[IMKClient subclass]: chose IMKClient_Modern
2025-02-20 18:49:09.424 python[72561:2669967] +[IMKInputSession subclass]: chose IMKInputSession_Modern
```

Figure 1: Training and testing set shapes and dimensions.

Then, we have displayed ten random samples from both sets, again, in order to ensure that the dataset samples are loaded properly. The figures given below displays example elements from training and testing datasets:

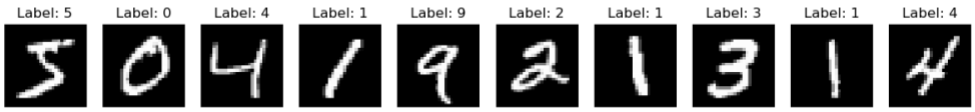

Figure 2: Example elements from the training set.

Figure 3: Example elements from the testing set.

After implementing the provided multicategory perceptron training algorithm, we have started to experiment with varying values of test samples, learning rate, and misclassification tolerance. The figure given below displays the results for $n = 50, \eta = 1$, and $\epsilon = 0$:

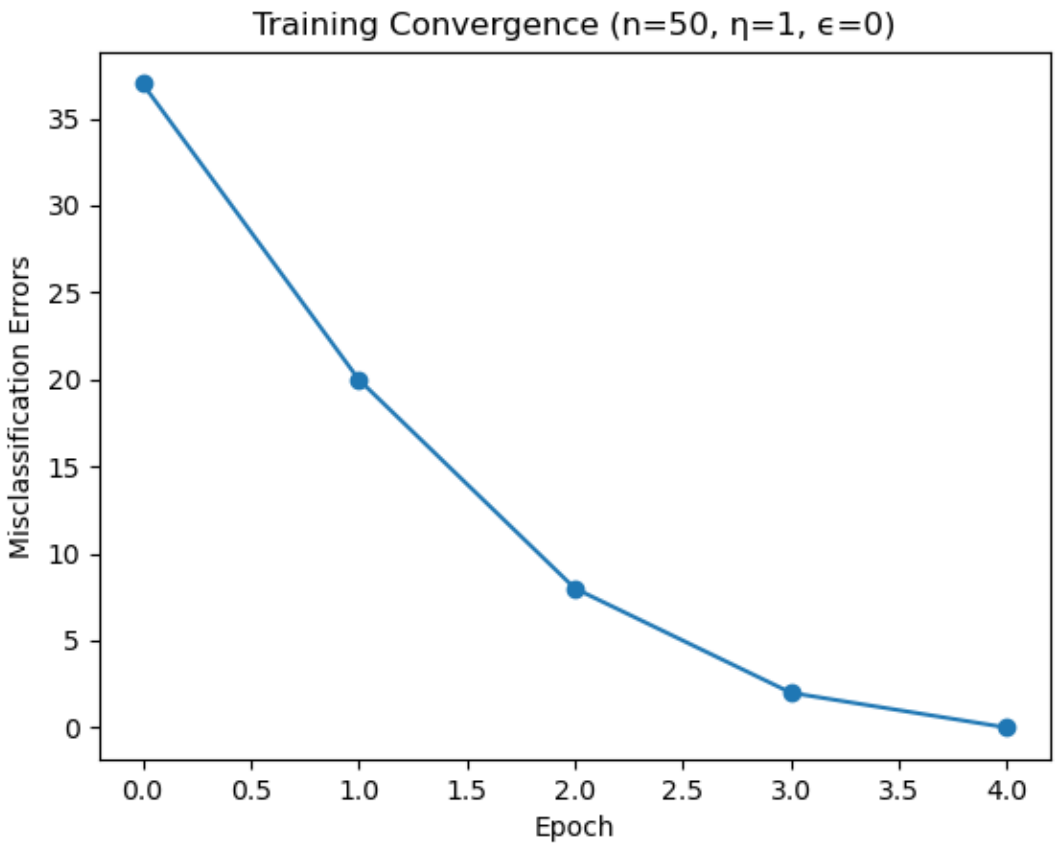

Figure 4: Training results for given parameters.

The test misclassification rate can be observed in the figure provided below, with number of misclassifications in each epoch during training:

```
==================================================
Training for n=50, η=1, ϵ=0
==================================================
Epoch 0: Errors = 41
Epoch 1: Errors = 17
Epoch 2: Errors = 7
Epoch 3: Errors = 2
Epoch 4: Errors = 2
Epoch 5: Errors = 4
Epoch 6: Errors = 2
Epoch 7: Errors = 0
Test Misclassification Rate: 44.78%
```

Figure 5: Errors during training and test misclassification results.

The figure given below displays the results for $n = 1000, \eta = 1$, and $\epsilon = 0$:

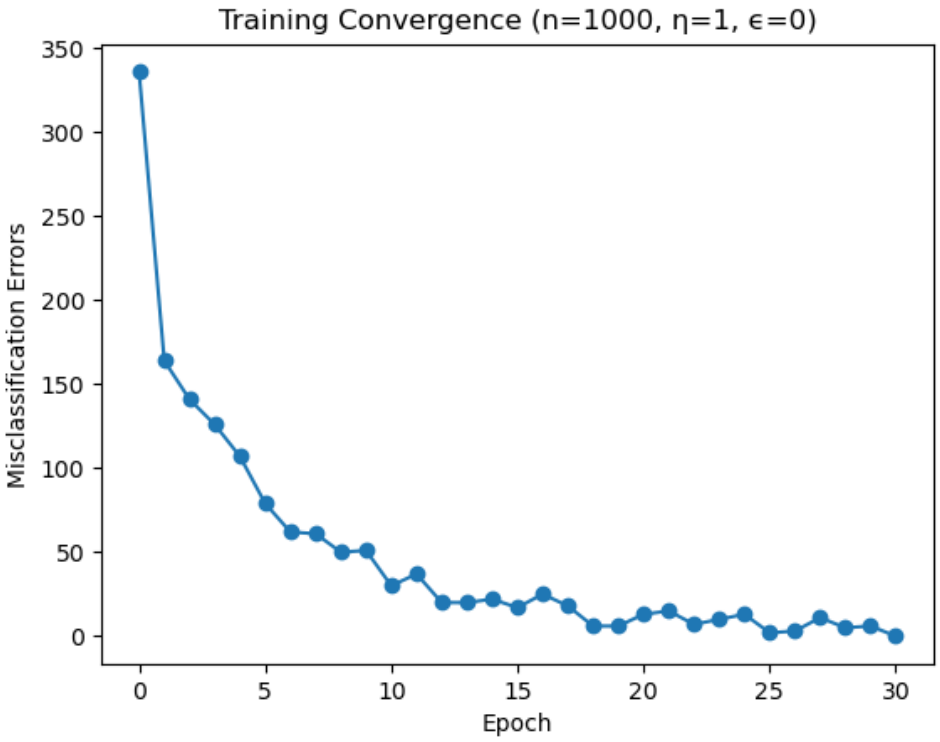

Figure 6: Training results for given parameters.

The test misclassification rate can be observed in the figure provided below, with number of misclassifications in each epoch during training:

```
==================================================
Training for n=1000, η=1, ϵ=0
==================================================
Epoch 0: Errors = 338
Epoch 1: Errors = 184
Epoch 2: Errors = 141
Epoch 3: Errors = 131
Epoch 4: Errors = 87
Epoch 5: Errors = 84
Epoch 6: Errors = 68
Epoch 7: Errors = 64
Epoch 8: Errors = 51
Epoch 9: Errors = 42
Epoch 10: Errors = 42
Epoch 11: Errors = 36
Epoch 12: Errors = 16
Epoch 13: Errors = 18
Epoch 14: Errors = 17
Epoch 15: Errors = 21
Epoch 16: Errors = 13
Epoch 17: Errors = 14
Epoch 18: Errors = 7
Epoch 19: Errors = 15
Epoch 20: Errors = 6
Epoch 21: Errors = 15
Epoch 22: Errors = 18
Epoch 23: Errors = 9
Epoch 24: Errors = 7
Epoch 25: Errors = 7
Epoch 26: Errors = 6
Epoch 27: Errors = 7
Epoch 28: Errors = 2
Epoch 29: Errors = 0
Test Misclassification Rate: 15.95%
```

Figure 7: Errors during training and test misclassification results.

During training, the perceptron achieves 0% training error, which means that all the training samples are classified correctly. However, when evaluated on the testing dataset, a nonzero misclassification rate occurs. These results may arise due to several factors, which can be stated as:

- **Overfitting:** The simple perceptron model learns to classify the exact training sample it was fed. However, the model lacks generalization, which means that it may struggle with unseen test samples.
- **Lack of Bias and Nonlinearity:** The implemented perceptron does not use a bias term, which results with a limited flexibility of decision boundaries. Additionally, the lack of nonlinearity limits the learning capabilities of the model.
- **Training Sample Size:** Small training sets may lead to overfitting since the perceptron learns a small number of samples rather than more generalized features. Larger datasets may improve generalization; however, the linear nature of perceptron inhibits the complex learning capabilities, which results with nonzero test errors.

The figure given below displays the results for $n = 60000, \eta = 1$, and $\epsilon = 0$:

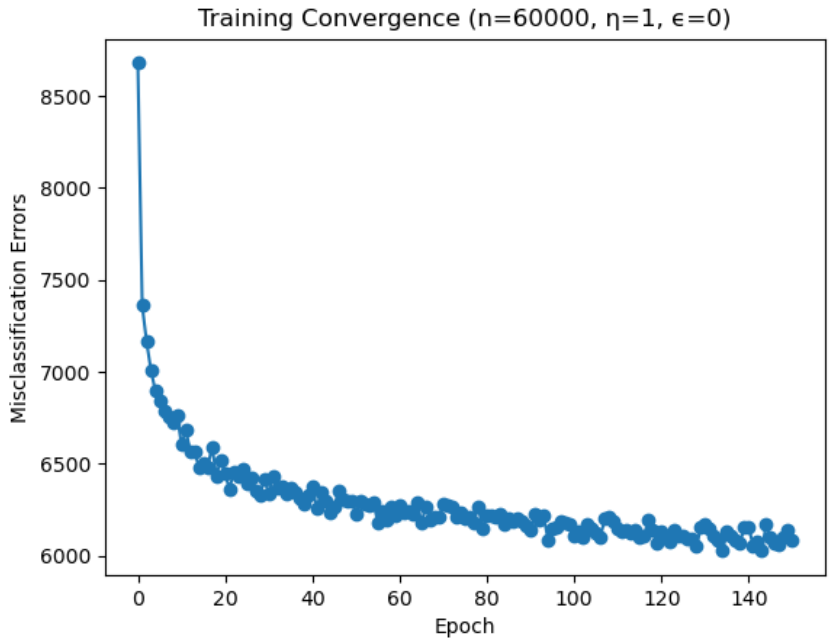

Figure 8: Training results for given parameters.

The test misclassification rate can be observed in the figure provided below, with number of misclassifications in each epoch during training:

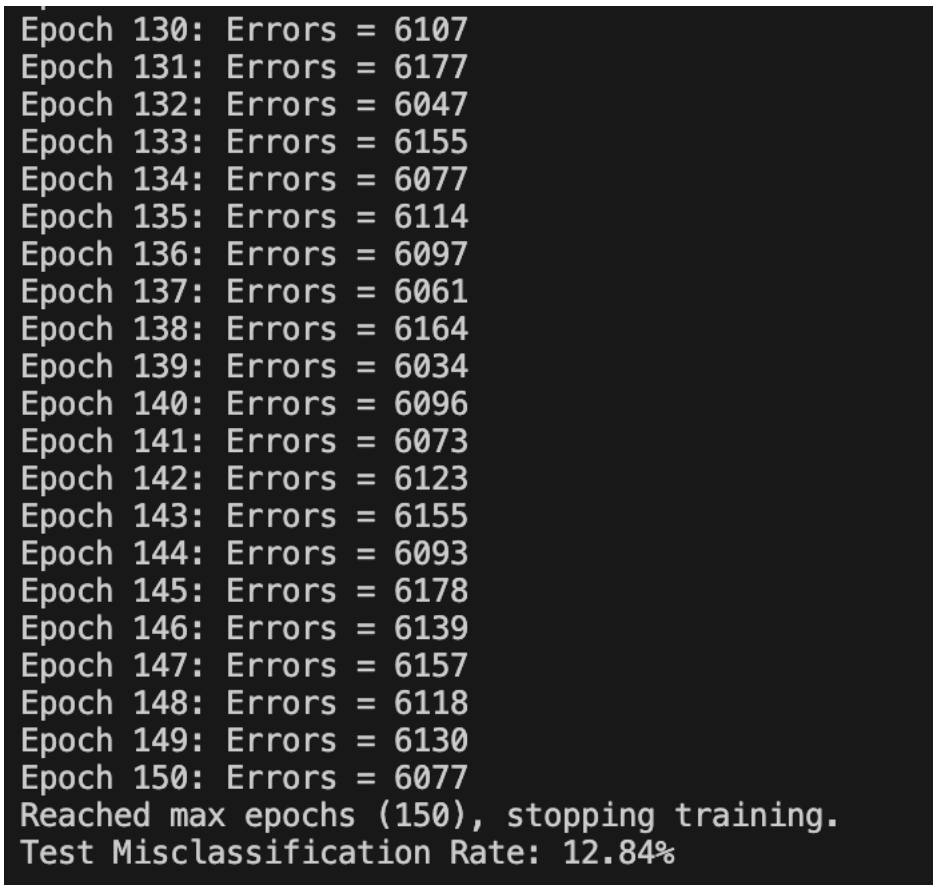

Figure 9: Errors during training and test misclassification results.

Observing the training plot and epoch error values yields that our multicategory perceptron model converges to an error value of around 6000, and does not converge to 0 error while training. In test misclassification, we observe that we obtain a lower misclassification rate than our previous tests.

The figure given below displays the results for $n = 60000, \eta = 0.5$, and $\epsilon = 0.01$:

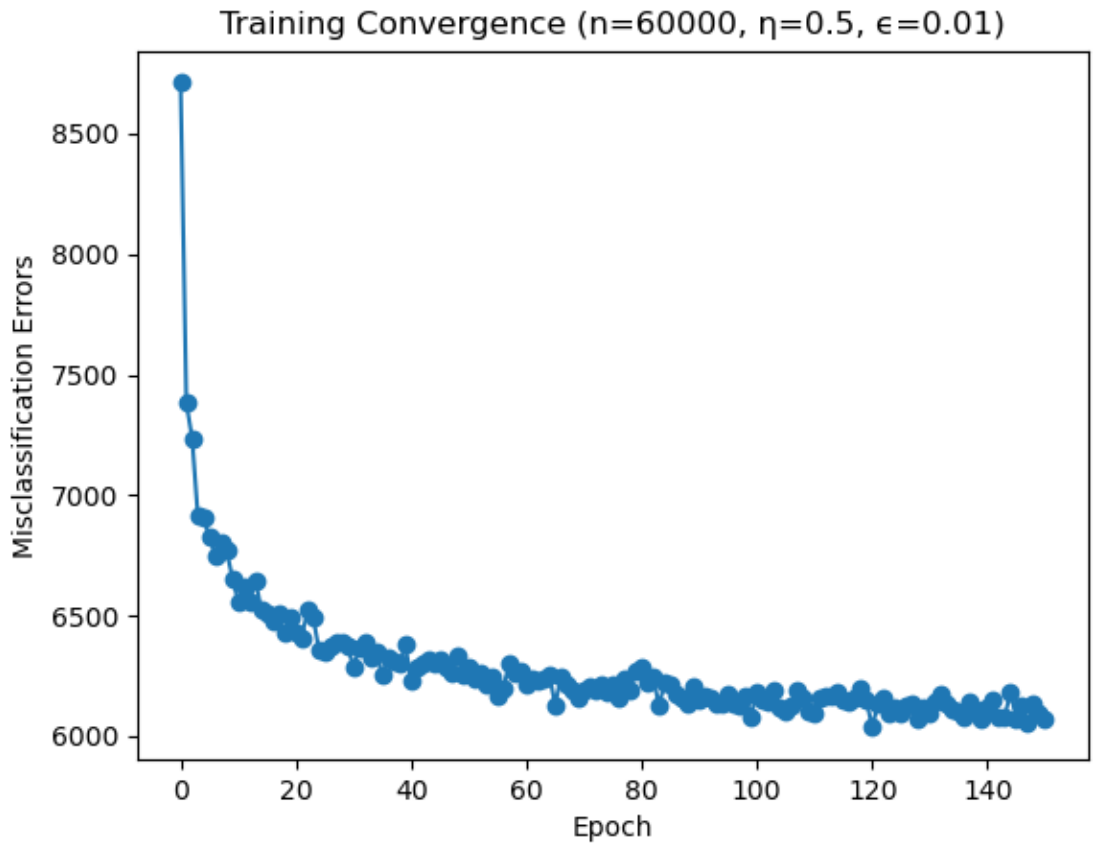

Figure 10: Training results for given parameters.

The test misclassification rate can be observed in the figure provided below, with number of misclassifications in each epoch during training:

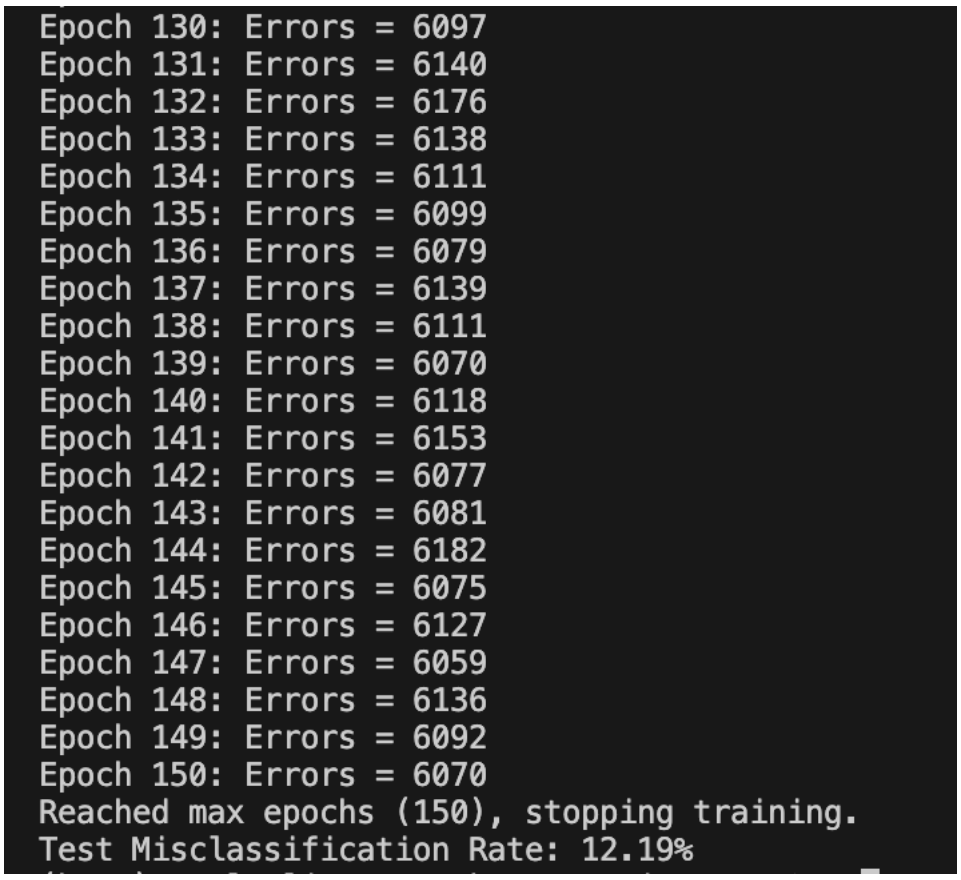

Figure 11: Errors during training and test misclassification results.

Again, the training plot and epoch error values yields that our multicategory perceptron model converges to an error value of around 6000, and does not converge to 0 error while training. In test misclassification, we observe that we obtain a lower misclassification rate than our previous tests.

The non-converge phenomenon may be occurring to several factors, which can be stated as:

- **Lack of Bias and Nonlinearity:** The implemented perceptron does not use a bias term, which results with a limited flexibility of decision boundaries. Additionally, the lack of nonlinearity limits the learning capabilities of the model.
- **Variability of Handwritten Digits:** Handwritten digits may vary in style. Lacking complex representations and feature extraction, the perceptron model cannot handle these variations.

Conclusion

In this class project, the Multicategory Perceptron Training Algorithm (PTA) was implemented to classify handwritten digits from the MNIST dataset. The algorithm was evaluated under different training conditions, varying the number of training samples, learning rate, and stopping threshold. The results demonstrated that while the perceptron achieves zero error on training data, its test error remains nonzero, highlighting the limited generalization ability of the perceptron model. The primary reasons for this may include overfitting, the linearity of the perceptron model, and the non-linearity of the MNIST dataset. Additionally, training on the complete dataset yielded that the algorithm failed to converge, as the perceptron learning rule is only guaranteed to converge for linearly separable data. Hence, we can state that while the perceptron model is effective for simple classification tasks, more advanced architectures, such as multi-layer perceptrons, are more suitable for complex and nonlinear datasets.