Department of Electrical and Electronics Engineering

EEE 443: Neural Networks

Class Project II Report

Name and Surname: Ali Aral Takak

Student ID: 22001758

Department: EEE

Instructor: Erdem Koyuncu

Problem

The second assignment involves implementing and analyzing the Perceptron Training Algorithm for binary classification. A dataset of 100 randomly generated points in $[-1,1]^2$ is classified using an initial set of weights (w_0, w_1, w_2). The perceptron algorithm is then applied with another set of random weights w_0', w_1', w_2', iteratively updating them until all points are correctly classified or a maximum number of epochs is reached. The impact of different learning rates ($\eta = 0.1, \eta = 1, \eta = 10$) on convergence is examined by tracking misclassification counts per epoch.

Initially, we start by handling a simple linear separation task on a set of random points by generating a decision boundary. Our dataset consists of 100 elements generated in the space $[-1,1]^2$. We start by setting our weights. The rule is as follows:

- A bias term w_0 is selected uniformly at random from the interval $(-0.25, 0.25)$.
- The weights w_1 and w_2 are each drawn independently from a uniform random distribution from the interval $(-1,1)$.

After setting the bias and the weights, we proceed with generating 100 random points in the space $[-1,1]^2$:

$$x_1, x_2 \sim U(-1,1)$$

$$x = (x_1, x_2)\ \epsilon\ S$$

Further, we proceed by classifying the data points x depending on the conditions given below:

- Each point x is classified using the linear decision function $w_0 + w_1 x_1 + w_2 x_2$.
- The point x belongs to the positive class S_1 if it satisfies the condition $w_0 + w_1 x_1 + w_2 x_2 \geq 0$.
- The point x belongs to the negative class S_0 if it satisfies the condition $w_0 + w_1 x_1 + w_2 x_2 < 0$.

The figure given below displays the result of the classification:

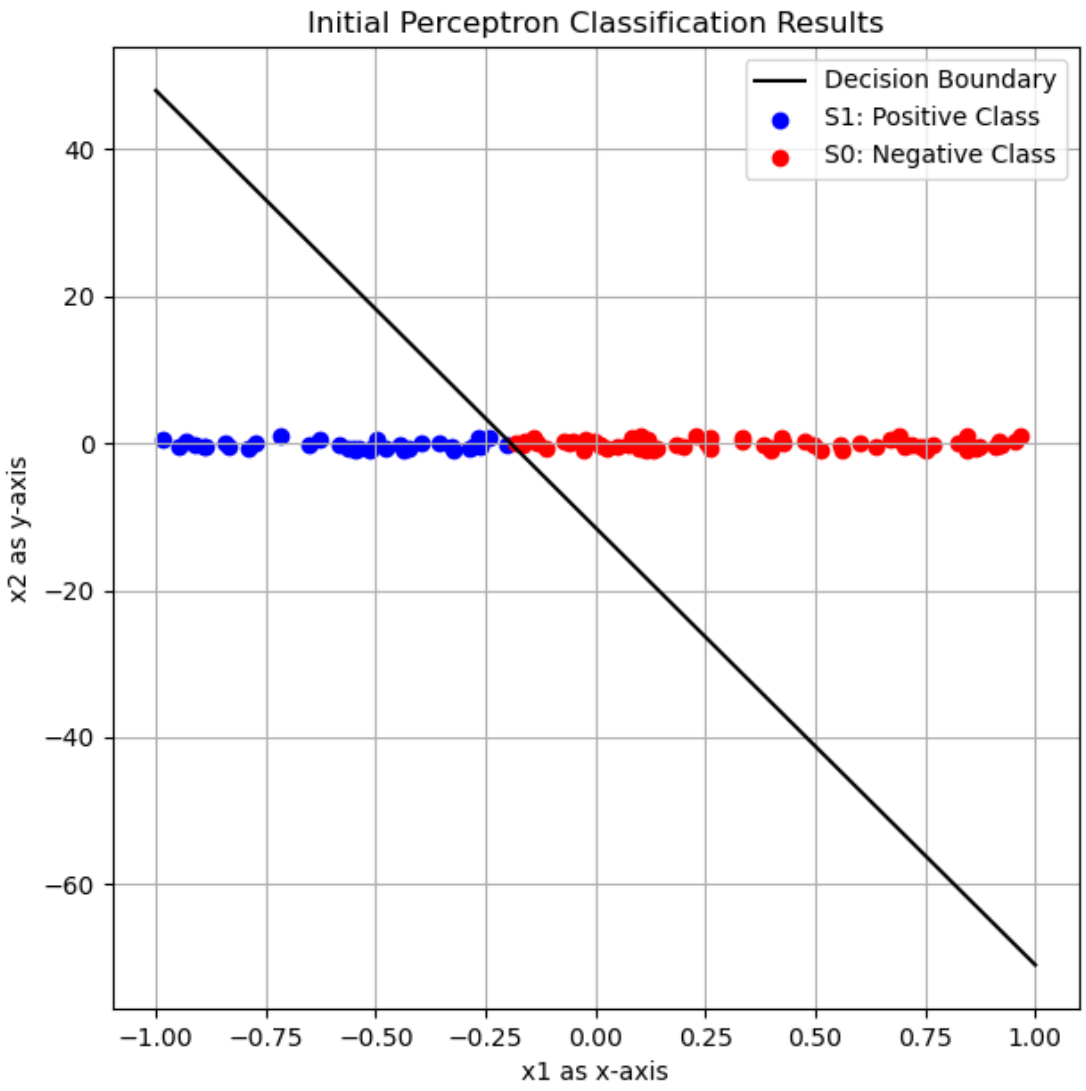

Figure 1: *Plot of the initial perceptron classification results.*

By using a random seed in our Python script, we have generalized our weights and made each run of the script same. Hence, we know that the random weights for the run above are given as:

$$w_0 \cong -0.1734, w_1 \cong -0.8981, w_2 \cong -0.0151$$

Moving further, we must implement and analyze the Perceptron Training Algorithm to find a separating hyperplane that correctly classifies the two sets S_1 and S_0. The perceptron iteratively adjusts its weights based on misclassified points using a fixed learning rate until convergence. The process for this is as follows.

1. A new set of weights w_0', w_1' and w_2' is randomly initialized from a uniform distribution over $[-1, 1]$.

2. Before applying the training process, the perceptron is tested with the initial weights defined in the prior step, and the number of misclassification points is recorded. Recall that a data point $x = (x_1, x_2)$ is classified based on the decision function $f(x) = w_0', +w_1' x_1 + w_2' x_2$; where the predicted label is determined according to the function:

$$\hat{y} = \begin{cases} 1, if\ f(x) \geq 0 \\ 0, otherwise \end{cases}$$

3. The perceptron learning algorithm is then applied iteratively to adjust the weights based on misclassified points. The weight update rule can be stated as follows:

$$w_j' \leftarrow w_j' + \eta(y - \hat{y})x_j, \qquad \forall j \, \epsilon \, \{0, 1, 2\}$$

4. After each epoch, i.e. a full pass over all the training samples, the updated weights are obtained. The number of misclassified points using these new weights is recorded. The training process is repeated until no misclassifications occur or the maximum number of epochs is reached.

We start by setting our learning rate η as $\eta = 1$, and weights as:

$$w_0' = 0.400, w_1' = -0.623, w_2' = -0.12783$$

We then proceed by testing our perceptron training algorithm with varying learning rates. The figure given below displays the results of misclassifications at each epoch for varying learning parameters $\eta = 0.1, 1, 10$:

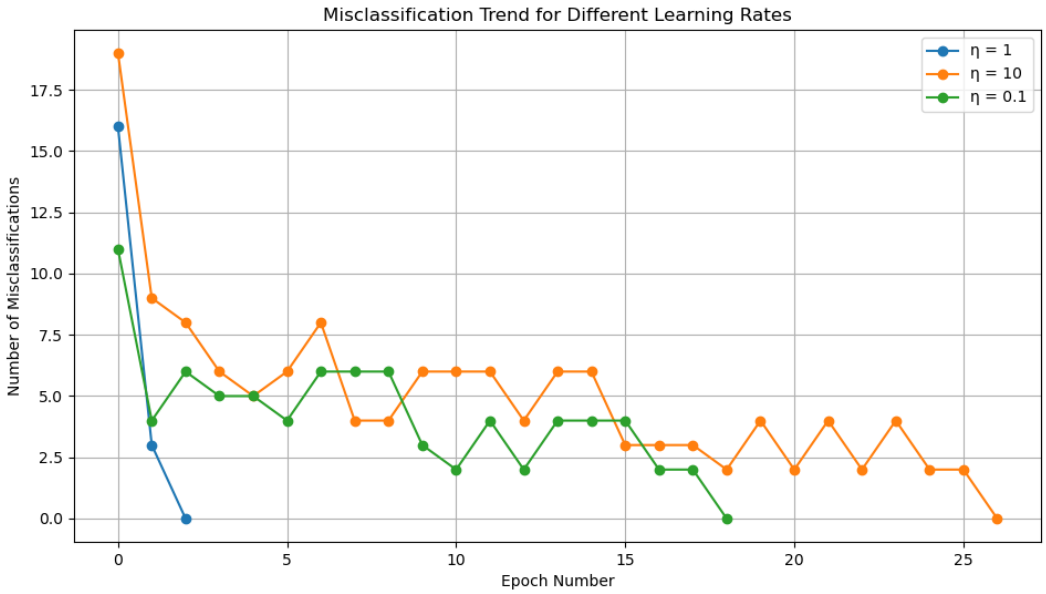

Figure 2: *Misclassification trend for different learning rates.*

Closer observation of the figure yields us with the conclusion that while a learning rates of $\eta = 1$ converged fastest to zero misclassifications, a learning rate of $\eta = 0.1$ converged faster than the learning rate of $\eta = 10$.

The final weights for each of the learning rates can be observed in the figure provided below:

```
Final weights for η=1: [-0.6000000000000001, -3.3583834635259286, -0.19747448922280586]
Final weights for η=10: [-19.6, -97.5189849729469, -1.8947718237769222]
Final weights for η=0.1: [-0.19999999999999998, -1.011468111985229, -0.013869317194738114]
```

Figure 3: *Final weights for each learning rate.*

The figure given below displays the results of misclassifications at each epoch for varying learning parameters $\eta = 0.1, 1, 10$; however, in this case, we are using a total of 1000 random samples:

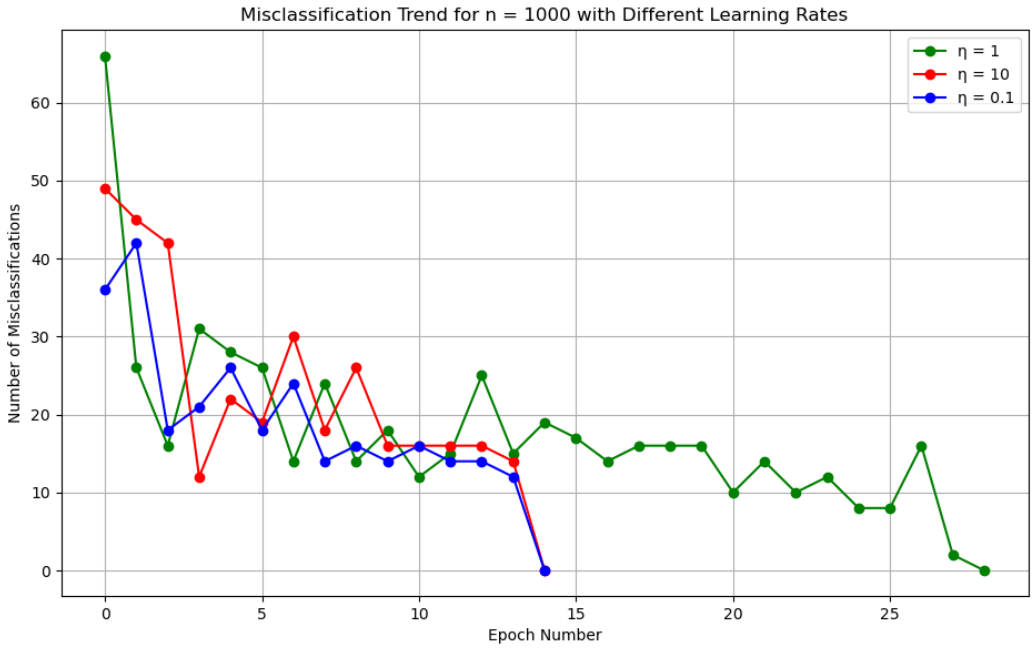

Figure 4: *Misclassification trend for 1000 random samples with different learning rates.*

When we compare the results of 100 samples and 1000 samples, we observe that the learning rates of $\eta = 10$ and $\eta = 0.1$ nearly performed the same, whereas the learning rate of $\eta = 1$ performed poorly than its counterparts. Also, it is observable that while the learning rate of $\eta = 1$ was the best performer for 100 random samples, it has been passed by $\eta = 0.1$ and $\eta = 10$ in a larger sample set.

The figure given below displays the final weights for the 1000 sample case:

```
Final weights for n=1000, η=1: [-3.5999999999999996, -18.72662931901931, -0.3467914590435113]
Final weights for n=1000, η=10: [-29.6, -156.6966126965461, -2.147547576941947]
Final weights for n=1000, η=0.1: [-0.30000000000000004, -1.5669169695506342, -0.02904389804570283]
```

Figure 5: Final weights for each learning rate for n =1000.

Lastly, let us analyze the effects of starting with different $w_0, w_1, w_2, S, w_0', w_1'$ and w_2' values:

- **Case A:** Consider starting with different initial weights. The final (or true) weights define the decision boundary for classification. If initial weights that are closer to the true weights would be chosen, the converge rate of the perceptron would be higher, which result with better performance. On the other hand, if we were to choose initial weight that are further from the true value of the weights, the perceptron would be performing poorer.
- **Case B:** The misclassification rate depends on the properties of the dataset too. Since the dataset is generated randomly, if some points occur to be closer to the decision boundary, the perceptron learning algorithm may need more updates to converge. The number of dataset elements would also affect the performance, since the computational costs of the dataset would increase on a larger dataset.
- **Case C:** Similar to Case A, starting with initial weights that are closer to the true weights will help the perceptron to converge faster. On the other hand, if the initial weights define a boundary completely misaligned with the data, the perceptron learning algorithm will take more time and more epochs to converge.