EECE6036 - Homework 2

Long Dang Vo

October 18th, 2022

1 Problem 1

1.1 Problem Statement

Train a single threshold neuron model using the MNIST dataset for the purpose of distinguishing between two handwritten digits. The training type would be a simple reinforcement paradigm, applying the Hebb Rule. Furthermore, the weights are initialized to random values between 0 and 0.5.

$$s(t) = \sum_{j=1}^{784} w_j x_j(t)$$

where $x_j(t)$ is the *j*th pixel value of the current image.

1.2 System Specification

I trained the weights of the entire training set using 40 epochs and a learning rate η of 0.01 specified. The learning rate is a hyperparameter that regulates how much the weight of the model should change in response to each mode update. Since a lower learning rate can cause values to decay, and if it is too small, it can cause the process to become stuck, a learning rate of 0.01 is what I consider to be optimal. Furthermore, my ROC after training is in the good range, showing that my model has a high level of accuracy.

1.3 Results

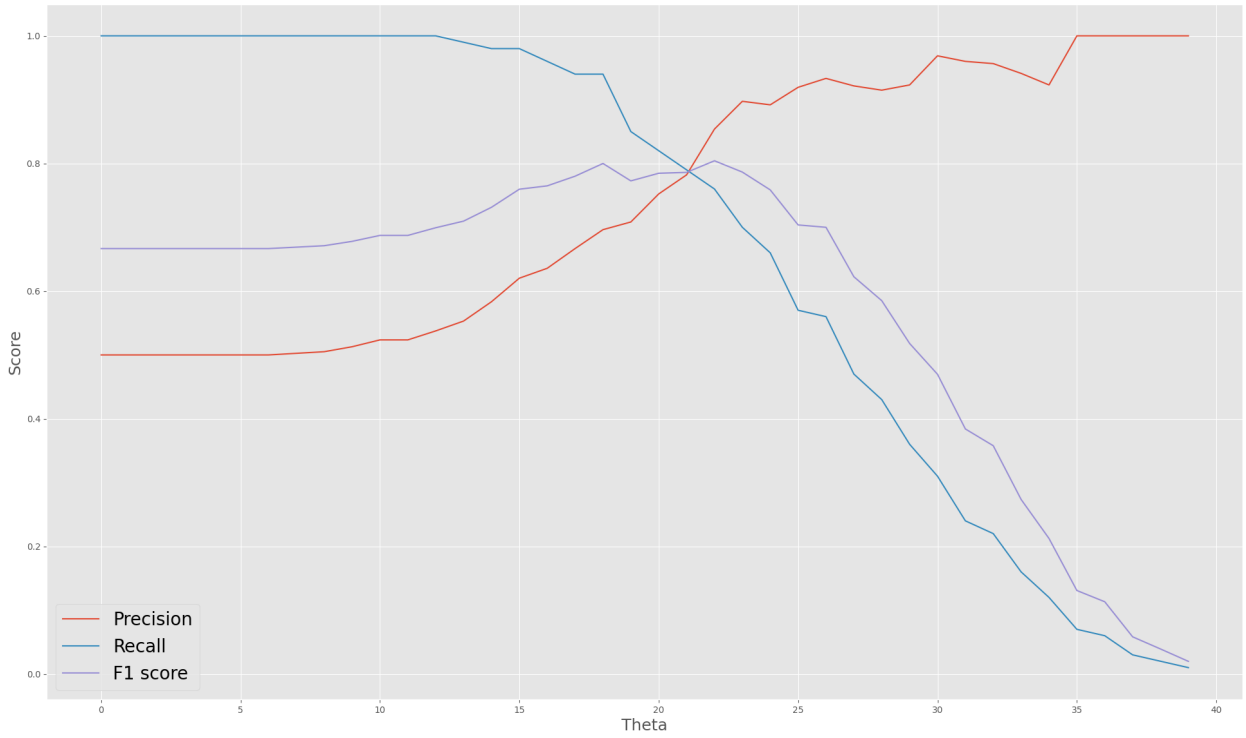

Figure 1.1. Precision, Recal, F1 score post training

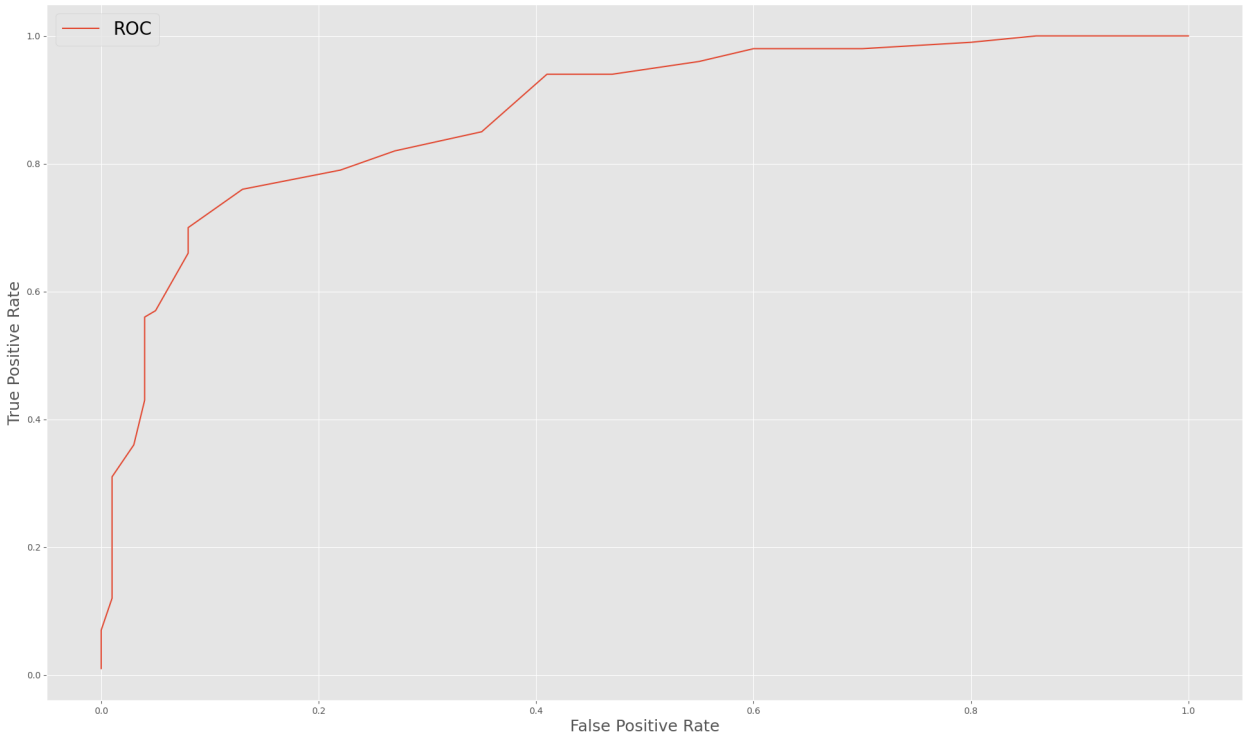

Figure 1.2. ROC curve (true positive vs false positive)

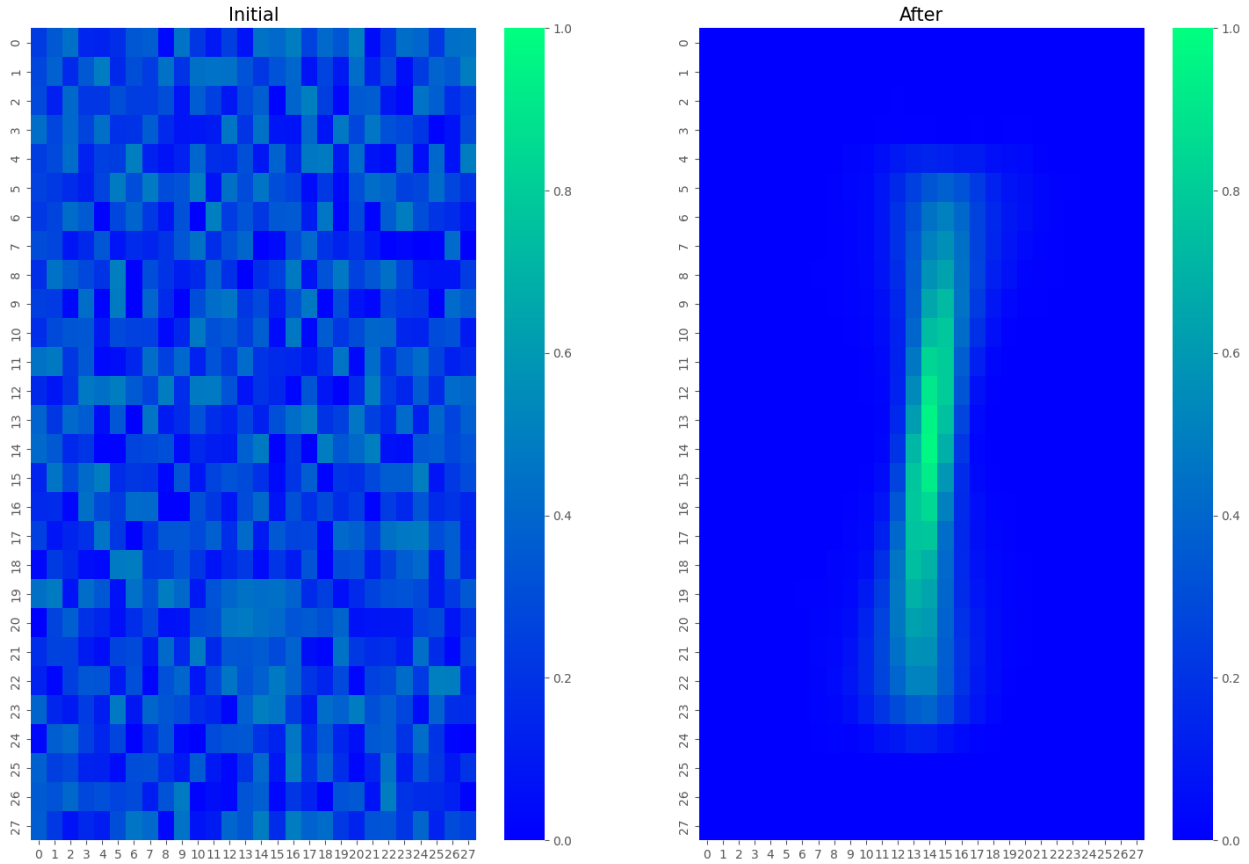

Figure 1.3. Comparison of pre-training data vs post-training data

	2	3	4	5	6	7	8	9
0	51	49	84	78	54	68	17	72
1	49	51	16	22	46	32	83	28

Figure 1.4. Classification of optimal threshold inputs

According to figure 1.2, I estimated the optimal threshold value as 22 using the formula of looping true_positve minus false_positive, which I believe would give the nearest value of the best ROC optimal value of the ROC curve.

1.4 Analysis of Result

Figure 1.1 illustrates how the fluctuation of the true positive value and false positive value causes the three lines of precision, recall, and f1 score to converge at one point. In addition, figure 1.3 shows that all the weights are randomly distributed at various indexes or coordinates prior to training, and that after training, all the weights are capable of forming an ordered shape in which we are able to classify the digit out of them. Figure 1.4 shows that the number of digits that are classified as 0 (standouts are 4, 5, 7, and 9) outweighs those that are classified as 1. In-depth analysis reveals that the generated output is allocated far from the heat map of 1 because those pixels that have a high value are allocated around 1, while those low value pixels would be allocated around 0.

2 Problem 2

2.1 Problem Statement

Repeat the same training with problem 1, but for this particular case, use perceptron learning instead. Your perceptron would then have 784 inputs from the images with 1 more bias input initialized with the value of 1.

2.2 System Specification

I trained all of the training set's weights using 100 epoch iterations and a specified learning rate η of 0.01. The learning rate, as previously mentioned, is a hyperparameter that controls how much the model's weight should change in response to each mode update. The lower the learning rate, the more steps would be involved, which would require more epochs. Furthermore, training error indicates the error in the trained data model, while test error shows the disjointness of the dataset, in which the lower the test error, the higher the accuracy of our model can produce. Also, after 100 epochs, the test error of mine was estimated at around 0.005, which is very low to indicate the highest possible value my model could then produce.

2.3 Results

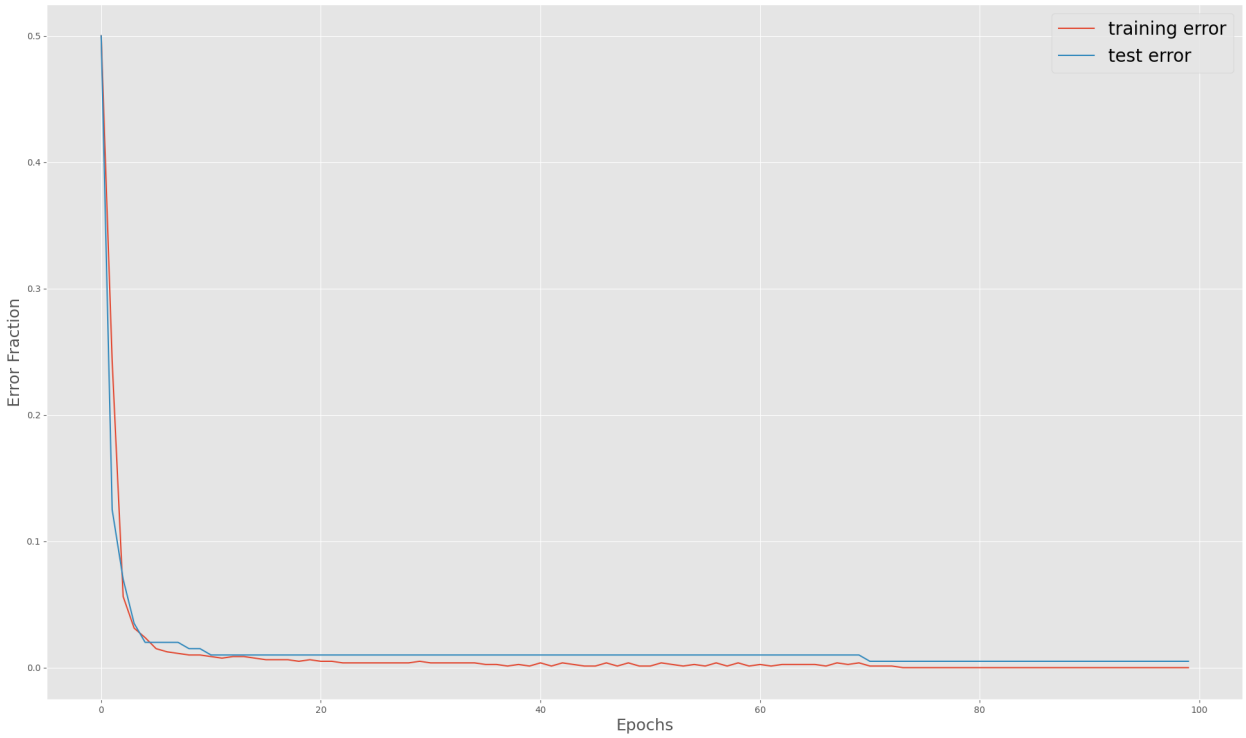

Figure 2.1. Training error and Test training error of perceptron after 100 epochs

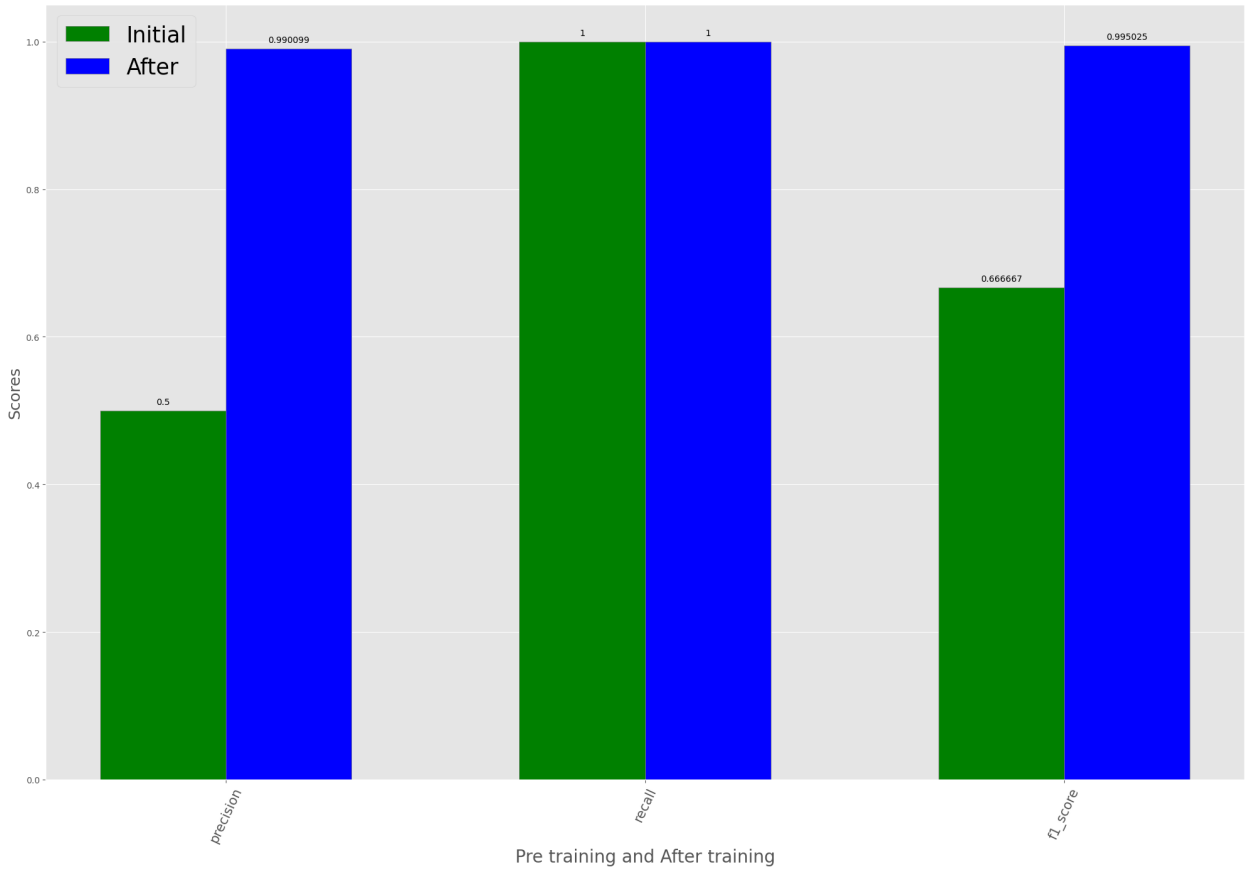

Figure 2.2. Paired bar plot for pre-train vs post-train in precision, recall, and F1 score

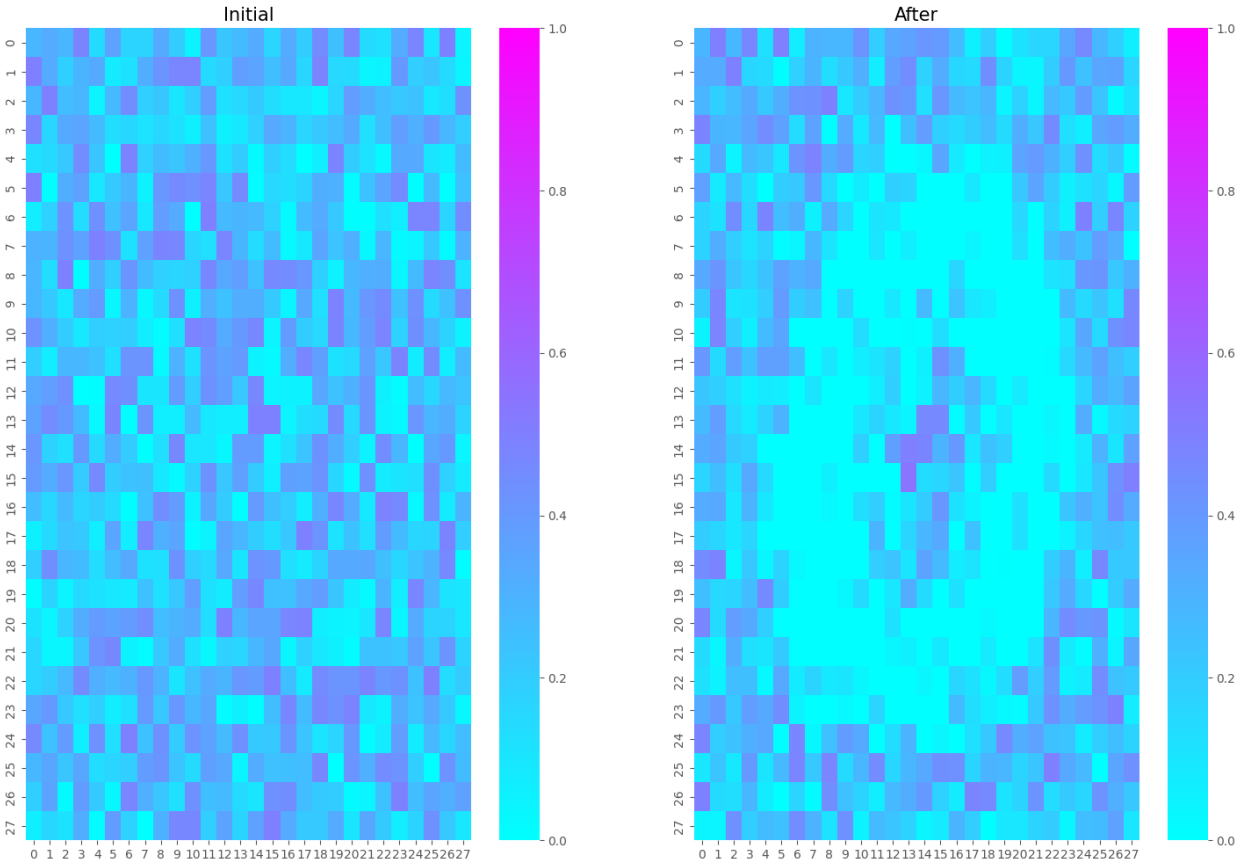

Figure 2.3. Comparison of pre-training data vs post-training data

	2	3	4	5	6	7	8	9
0	31	39	30	67	55	11	31	30
1	69	61	70	33	45	89	69	70

Figure 2.4. Classification of perceptron optimal threshold inputs

2.4 Analysis of Result

According to figure 1.2, the best theta evaluated would be where three lines of precision, recall, and f1 score converge. We can infer that they received scores that were roughly 0.8 from that point. In the meantime, the estimated f1 score after perceptron, recall, and precision varied roughly between 0.9 and 1. Due to the perceptron's higher precision and recall, the model's f1 score value also suggests that it is a better one. Figure 2.4 demonstrates that as 1 outweighs the classification of 0, more classifications of the digits from 2 to 9 are emerging. It makes sense that the weight vectors were distributed around the number 1 and close to the heat map of 1. The single-threshold neuron model employs the Hebb rule for updating weights, and Peceptron learning employs a similar strategy but with bias to correct for weight error. Additionally, both learning algorithms generated high precision, recall, and f1 scores. The perceptron learning algorithm ultimately performs better when performance metrics are evaluated.