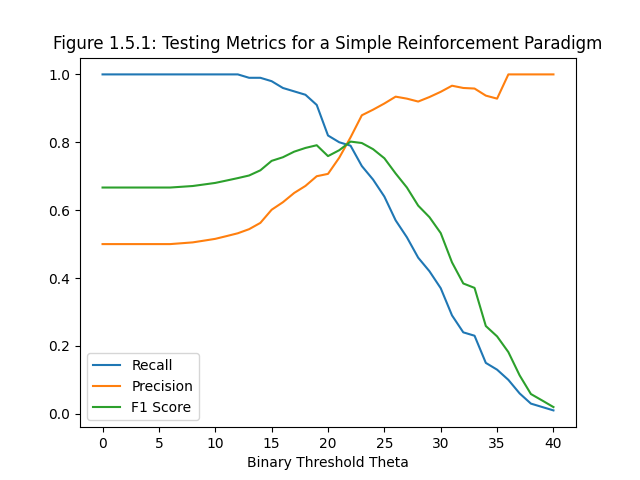
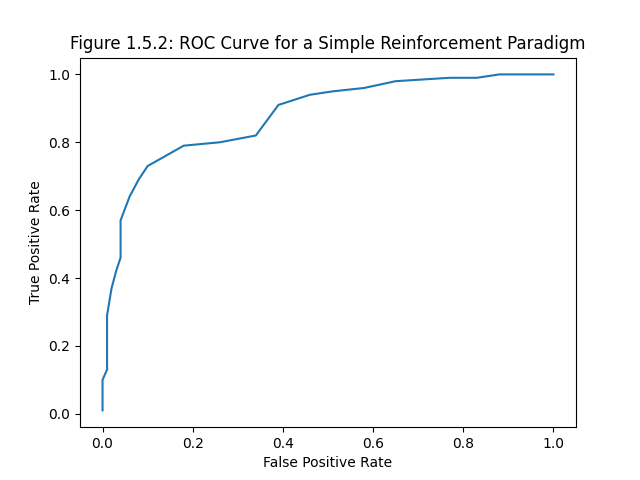
**Ryan Connolly**

**Problem 1**

**System Specification**

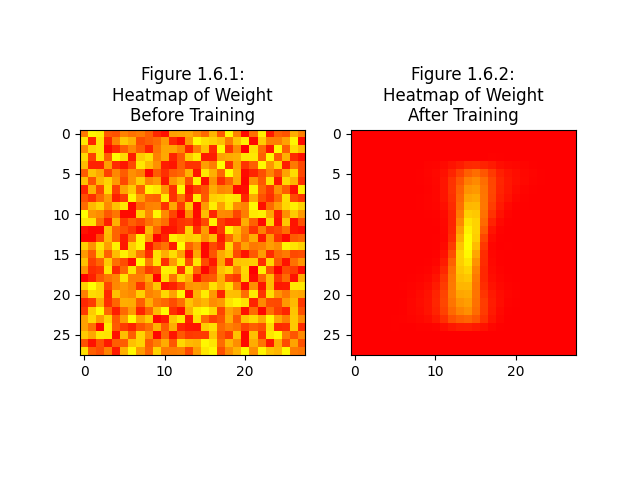
For my training epochs, I used a total of 101 epochs, meaning θ values 0 to 101 (non-inclusive), with an eta of 0.0075. I began with the original 41 epochs and eta of 0.01, but when I increased them, I noticed a concurrent increase in the metrics of the best θ case, especially the F1 score. However, the weight’s heat map did not change visibly, so I settled on these values because they produced metric values above .8 without being too radically high themselves.

**Results**

|  |  |  |
| --- | --- | --- |
| **Figure 1.5.1:** three lines plotted over each other, representing the testing set’s recall, precision, and F1 score as a function of binary threshold θ. |  | **Figure 1.5.2:** plot of the receiver operating characteristic curve (ROC), which is the TPR (recall) of the testing set over its FPR (1 – specificity). |

My estimate for best θ value is 20. Optimal θ is where the slope becomes less than one, and as θ progresses so does TPR and FPR, so I believe the point on the line near (.01, .73) is the best point for θ. This is almost a quarter of the way through the graph, leading to my estimate of θ = 20.



**Figure 1.6:** side-by-side heatmap plots of the weights (wj). Original, randomly generated   
in 0.0-0.5 range pre-training weights are on the left, and adjusted, “learned”   
weights are on the right. Yellow represents higher values, red lower values.

Table

Description automatically generated

**Figure 1.7:** a table showing how many of each additional digit 2-9 were classified   
as 0 and how many as 1. The top row and leftmost column are labels, showing   
which column is for which digit and which row is for which classification.

**Analysis of Results**

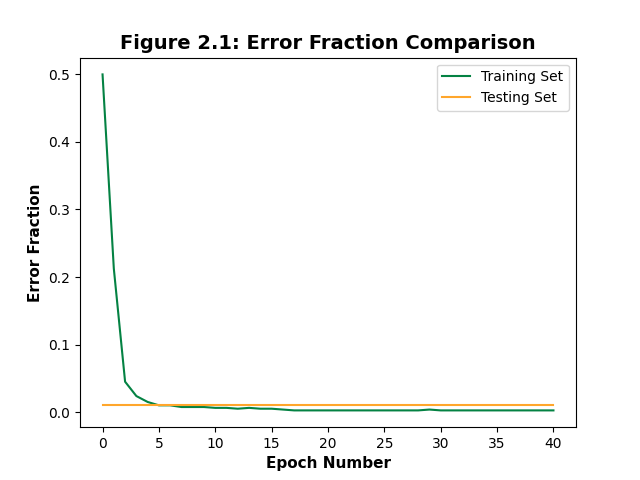
Figure 1.5.1 shows precision rising and recall drop, implying that the positive accuracy rate rose because it drastically cut down on all positive classification. The Figure 1.5.2 ROC curve has a large area beneath it, signifying high classifier quality. The challenge set results in Figure 1.7 have a weak pattern of classifying letters with loops or circular segments as 0s and the others as 1s, which 6, 7, and 8 all break. Given that the neuron was taught to identify 1s, so its identifying of 0s is analogous to identifying a number as *not 1*, this is sure to be a coincidence. More likely, it classifies numbers that have the most of their shape on the center vertical line of their space, like the neuron knows a 1 to have, shown in its trained weights in heatmap Figure 1.6.2.

**Problem 2**

**System Specification**

I used 100 epochs in order to tune my perceptron to optimal weight. I initially did the same 41 as with Problem 1, but this proved to be too low. Going too far into the hundreds seemed to be possibly counterproductive, based on the heatmap comparisons. It was as though continuing on for too long muddled the weight, which has been clearer earlier on, seemingly because it passed the peak epoch previously. With this observation, I decided on 100. Adjusting along the way based on outcomes, I decided on the same 0.0075 for the learning rate like in Problem 1. Too small of a learning rate would result in too much noise in the field of the weight heatmap, and too large would make it harder to detect variant cases of 1s.

**Results**

****Chart, bar chart

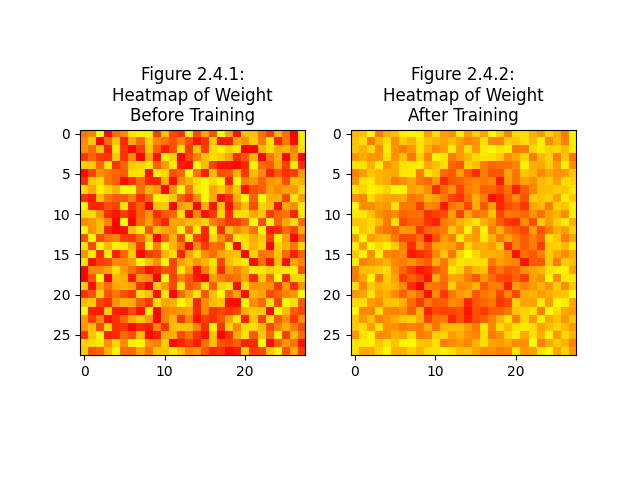
Description automatically generated

|  |  |  |
| --- | --- | --- |
| **Figure 2.1:** the training set error fraction and the testing set error fraction plotted as functions  of the epoch number. |  | **Figure 2.2:** the perceptron’s metrics of precision, recall, and F1 score before being trained (red)  and after being trained (blue). |

Table

Description automatically generated

**Figure 2.3:** a table showing the results of the perceptron’s 0-vs-1 classification model   
applied to the numbers 2-9. The top row and leftmost column are labels, showing   
which column is for which digit and which row is for which classification.



**Figure 2.4:** side-by-side heatmap plots of the perceptron’s weights. Original, randomly generated   
in 0.0-0.5 range pre-training weights are on the left, and adjusted, “learned” weights   
are on the right. Yellow represents higher values, red lower values.

**Analysis of Results**

There is a key, fundamental difference between the neuron and the perceptron in terms of how they learn, and that is this: the neuron learns something by seeing it for what it is, and the perceptron learns something by seeing it for what it is *not*. This is illustrated in heatmaps Figure 1.6.2 and Figure 2.4.2. The former (the neuron’s) shows a yellow 1-shape, yellow being higher weight and signifying that pixel’s importance to 1. The latter (the perceptron’s) shows a red 0-shape, showing that these pixels are where things that are *not* 1 are and using that to, through a binary process of elimination, declare what is 1. This is the inverse of my looped numbers observation. For the neuron, recall, precision, and F1 score all converged on around .79 for the optimal θ. Compared to the perceptron where they converge toward 1.0, it is clear which algorithm is more efficient.