Introduction:

In this part of the assignment, we are to apply a multi-class perceptron learning rule on the digit classification problem from assignment 3. Given to us are 5000 training data entries and 1000 test data entries for the numbers zero through nine. With these, we are to create a classification rule that is tuned to improve the overall classification rate. The parameters to tune are the learning rate decay function, bias inclusion, weight initialization, training order, and the number of epochs.

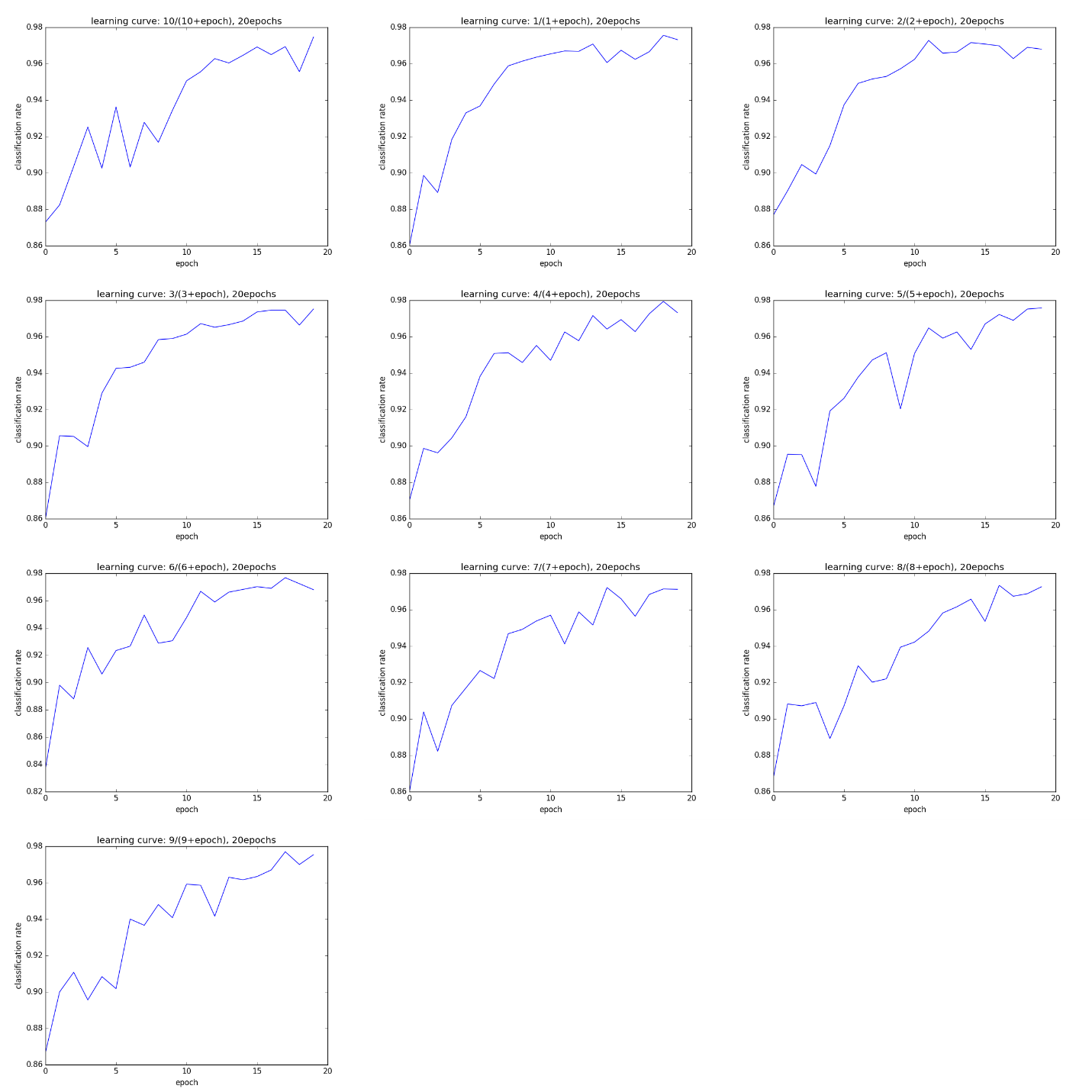
Multi-Class Perceptron­:

The multi-class perceptron implementation we created first initializes a weight array and then tries to classify one of the data entries in the training data. The decision rule to classify the class was argmaxc(**W**c\***X**). If the classification was incorrect, then the weight for the misclassified class is updated as **W**c’🡨**W**c’-α**X**, and the correct class is updated as **W**c🡨**W**c+ α**X**, where α is the learning decay function defined as , such that is any real number, and **W**c and **W**c’ are the weight arrays for class and respectively, and **X** is the data entry such that a class feature is ‘1’ and background is ‘0’. This algorithm was then run over the entire training data number of times, where .

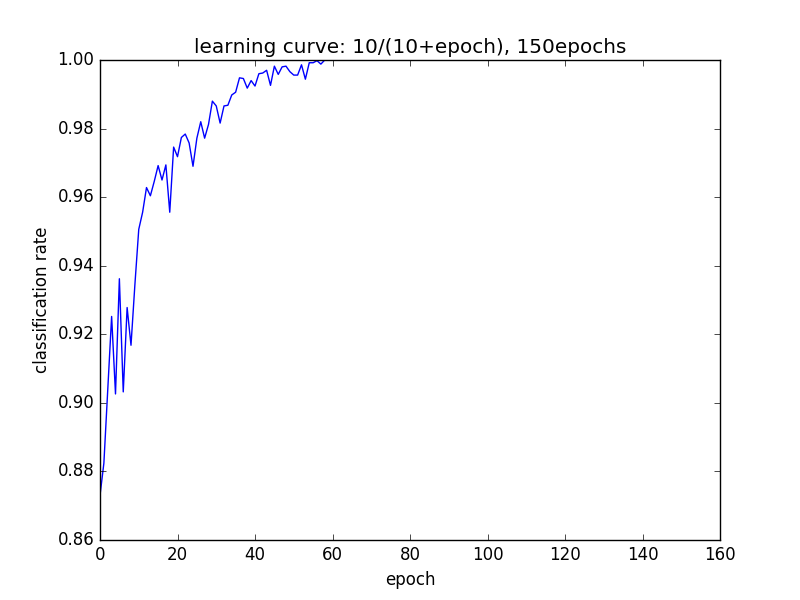
After each epoch, the classification rate on the training data was then returned, thus allowing us to tune the parameters stated in the introduction. Following this method of classification and tuning, we discovered that randomizing the training data and including a bias returns the best classification over a set number of epochs.

Parameter Setting:

We tuned our parameters by first running the training data over ten learning decay functions with 20 epochs each, and found out that they all converged to about 96% accuracy, with a learning function converging the fastest. Thus, the learning decay function we decided to use was . Below are all the classification curves for the given learning curve and 20 epochs.



Classification Curve for 10 Learning Decay Functions



Learning Curve for One Learning Decay Function over 160 Epochs

(The cases below are run with the above learning function and 20 epochs)

We then ran the training data again with and without bias and plotted the classification curve. After this, we then ran a randomized weight following a Gaussian distribution with a mean of zero and a standard deviation of 4.5, all with and without a bias, and then plotted the classification curve. Finally, we ran the data with randomized training data on each epoch both with and without bias, and then plotted this data.

|  |  |  |
| --- | --- | --- |
|  | No Bias | Bias |
| Normal |  | C:\Users\EyeAm\AppData\Local\Microsoft\Windows\INetCache\Content.Word\alpha_1_bias.png |
| Gaussian |  |  |
| Random  Trials |  |  |

Tabulated Trials

From this, we discovered that a randomized training data with a bias returns the best classification rate over 20 epochs, with the classification rate on the 20th epoch to be 99.34%

Discussion:

Once we found the parameters that returned the best classification rate, we ran the classification against the test data, and got a classification rate of 82.1%. The confusion matrix of such is:

% Correct Overall: 99.34

Number Classification Report:

Number of Training Images: 5000

Training Image File: trainingimages

Training Label File: traininglabels

Number of Test Images: 1000

Test Image File: testimages

Test Label File: testlabels

Confusion Matrix:

| |0 |1 |2 |3 |4 |5 |6 |7 |8 |9 |

--------------------------------------------------------------------------| 0|84 |0 |0 |0 |1 |3 |1 |0 |0 |0 |

| 1|0 |105 |2 |0 |0 |0 |1 |2 |5 |0 |

| 2|2 |1 |85 |2 |2 |1 |3 |4 |8 |0 |

| 3|0 |0 |5 |82 |1 |4 |0 |4 |9 |4 |

| 4|0 |1 |1 |0 |90 |1 |3 |3 |1 |9 |

| 5|1 |0 |0 |7 |0 |70 |3 |1 |7 |0 |

| 6|1 |1 |3 |3 |3 |2 |76 |0 |2 |0 |

| 7|0 |0 |2 |4 |2 |2 |2 |82 |2 |3 |

| 8|1 |0 |4 |2 |1 |7 |2 |0 |64 |1 |

| 9|1 |0 |1 |0 |7 |2 |0 |10 |5 |83 |

--------------------------------------------------------------------------% Correct Overall: 82.1

Where the columns are actual classifications and the rows classification guesses.

Compared to the accuracy of the Naïve Bayes method from assignment 3, where we got an accuracy of 77.7%, the classification rate of the perceptron learning rule was better. Below is the confusion matrix for the Naïve Bayes method to discuss any differences to the perceptron confusion matrix.

Confusion Matrix:

| |0 |1 |2 |3 |4 |5 |6 |7 |8 |9 |

--------------------------------------------------------------------------| 0|76 |0 |1 |0 |0 |2 |1 |0 |1 |1 |

| 1|0 |104 |3 |1 |0 |1 |3 |5 |1 |1 |

| 2|1 |1 |83 |0 |1 |1 |5 |4 |3 |0 |

| 3|0 |0 |4 |80 |0 |12 |0 |0 |14 |3 |

| 4|1 |0 |2 |0 |80 |3 |3 |3 |3 |9 |

| 5|5 |2 |0 |3 |1 |63 |6 |0 |8 |2 |

| 6|3 |1 |5 |2 |4 |1 |71 |0 |0 |0 |

| 7|0 |0 |1 |7 |1 |1 |0 |77 |1 |2 |

| 8|4 |0 |4 |1 |2 |2 |2 |3 |62 |1 |

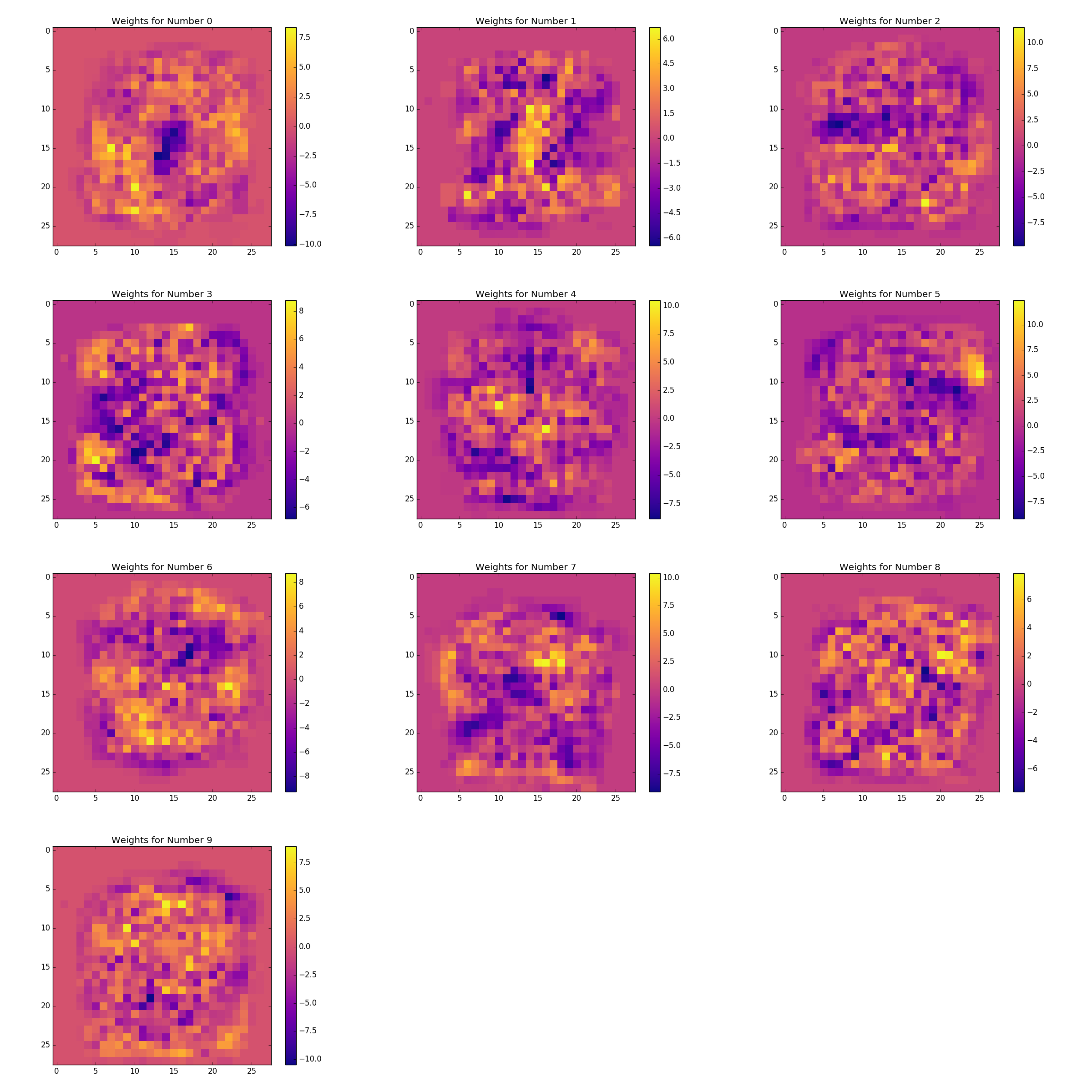
| 9|0 |0 |0 |6 |18 |6 |0 |14 |10 |81 |

--------------------------------------------------------------------------% Correct Overall: 77.7

Overall, the confusion matrix for perceptron is much better compared to the Naïve Bayes confusion matrix. However, one difference that stood out is that the perceptron model classified the number 8 as 1 and 2 many more times than the Naïve Bayes classification.

Extra Credit:

*Weight Images*



Weight Plotting for Numbers 0 – 9

The above images are the perceptron weights for each class number, with the lighter portions being the more important parts of the classification. The significance of these lighter portions tell us that these spots are the discerning factors that differentiate one number from another. One example of such is when looking at the weight map of the number 5, the top right start of the number seems to be important in discerning it apart from the other numbers. By this train of thought, the other numbers can be seen as such, with the most important part of identifying the number being the lighter of the map colors.

*Ternary*

To implement the ternary perceptron rule, we followed the same basic idea as the binary perceptron rule, except that when updating the rule, the **X** matrix for the number is now ‘1’ for a ‘#’, ‘0.5’ for ‘+’, and ‘0’ for a blank space. The method for classification and training is still the same as the binary perceptron however.

|  |  |  |
| --- | --- | --- |
|  | No Bias | Bias |
| Normal |  |  |
| Gaussian |  |  |
| Random  Trials |  |  |

Tabulated Ternary Trials

From these above trials, we found out that the best classification rate after 20 epochs, with the same training curve, , as the binary perceptron rule, was the randomized trials including bias.

With the best parameter settings, we found that the classification rate on the training data to be 96.82%, while the classification rate for the binary perceptron rule to be 99.34%. Thus, from this we saw that even though implementing a feature that improved the Naïve Bayes model classification rate, instead lowered the classification rate for the perceptron rule. The classification rate for the test data was also worse, with a classification rate or 80.8%, compared to the binary perceptron rate of 82.1%.