CS460G Programming Assignment 2

Tyler Filbert

Binary Classification Using Multilayer Perceptrons:

My code is straightforward and was derived from the lecture notes. I chose to implement only 1 hidden layer that holds 256 weights as just a starting point, though it seemed to work well with that, so I kept it. I also only have a single output node. I make the distinction between an output of 0 or 1 by seeing if the value is less than 0.5. If it is, I record it as a 0 and vis versa. Sigmoid was the activation function I used as this was clearly modeled and discussed frequently in class, so it made implementation straightforward. Finally, as a learning rate (alpha) I chose a value of 0.3. I found any value near this was relatively similar, so I kept it at this in the final implementation.

To read in the data I just made a simple helper script to save each image to a dictionary (based on where it occurred in the csv file) that held its class label and image values. I divided the image values by 255 to avoid overflow errors when creating the neural net.

Test results:

```
Total images tested: 2115
Total correct: 2114
Total accuracy: 99.95%
```

To run the code is simple, with numpy installed. Simply open neural_net.py and run and it should report the accuracy. The runtime should be a minute or two for this one. The data file is included in the zip file, so no need to change anything with reading in the data.

Bonus: Multiclass Classification using Multilayer Perceptrons:

I built of the above section by adding 5 output nodes instead of 1. For the output, I declared the correct value as being a one-hot encoding of the value at the specified index. I used the same architecture as the first section and that worked well. To compare the guessed value, I took the index of the largest value from the output and declared that as the outputted guess. I also increased the learning rate slightly, to 0.4, which I found to help my accuracy. Other than this, the implementation was like the first section. As for the accuracy, it can be across the board, but frequently reports accuracies near 97%.

Test results:

```
Run 1:
Total images tested: 5139
Total correct: 4999
Total accuracy: 97.28%
Run 2:
Total images tested: 5139
Total correct: 4134
Total accuracy: 80.44%
```

To run this code it is the exact same as the section above, but this time just open and run neural_net_multi.py. You will need numpy installed. Again, the data file is included in the zip file, so it is just a matter of unzipping my submission and running it.