Perceptron Lab

Drake Foltz

# Task 1: Implement Code:

In writing the code, there were several things that I was not sure about, one of which being where to write the code that would split my data into training and testing sets. In the end, that is done out in my \_\_init\_\_ file, and then passed in separately to the appropriate methods. And as for shuffling, rather than move whole rows of data around, I merely made an index to the instances, and then scrambled the index respective to both the instance and the target.

## Task 1.2: Stopping Criteria:

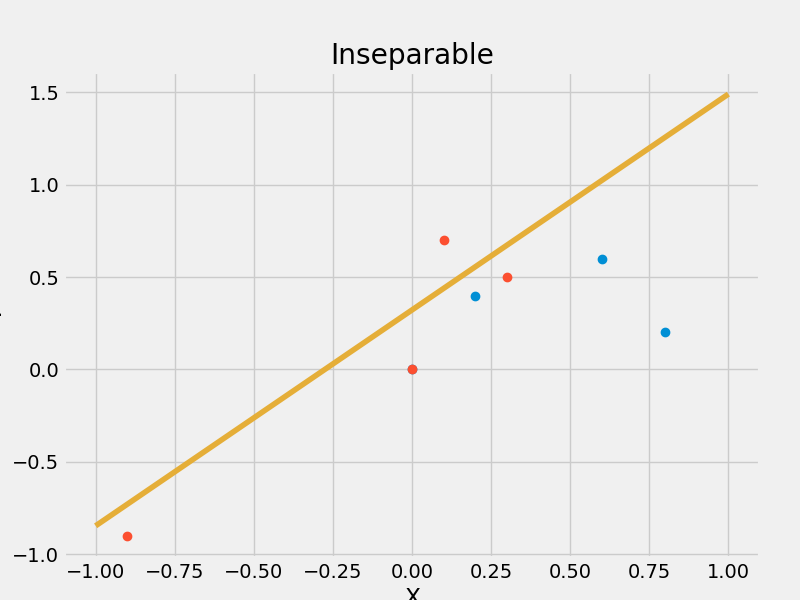
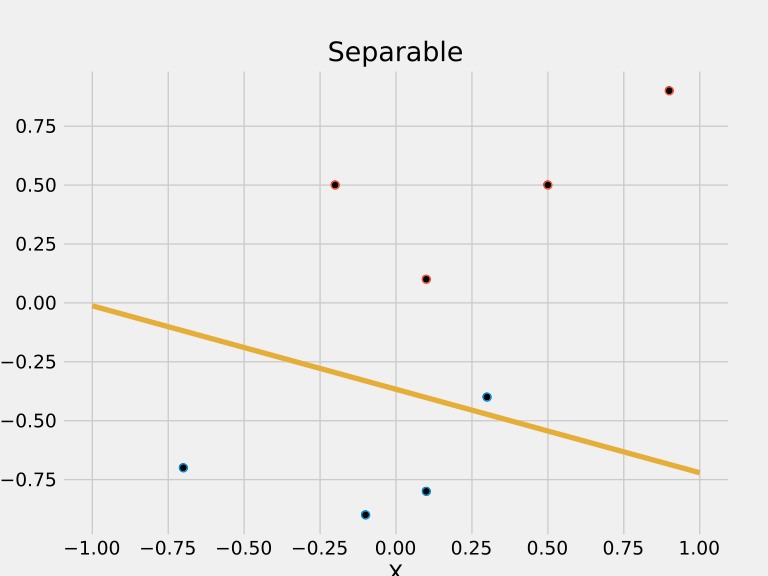
When I was deciding how to implement my stopping criteria, I had not carefully read the instructions, nor the slack channel up to that point. While what I had done was not bad, it was perhaps extra work that I could have avoided. I implemented my stopping criteria as checking for the “L1 Loss” and then minimizing that against the previous loss (initially set to np.inf), give or take some tolerance, which I ended up never using. Meanwhile, the slack channel recommended using the score method to implement this. I only read that comment when I was looking on how to implement the score method itself, and decided not to change my stopping criteria.

# Task 2: Own dataset:

Here are my two datasets that I made; additionPositive is meant to classify if two numbers would add to a positive. My “impossible” dataset is recognizing “odd” decimal numbers, but “0” is classified as both.

| @relation additionPositive  @attribute height real  @attribute length real  @attribute class {0,1}  @data  0.5, 0.5, 1  -0.2, 0.5, 1  -0.1, -0.9, 0  0.9, 0.9, 1  -0.7, -0.7, 0  0.1, -0.8, 0  0.1, 0.1, 1  0.3, -0.4, 0 | @relation impossible  @attribute x1 real  @attribute y1 real  @attribute class {0,1}  @data  -0.9, -0.9, 1  0.3, 0.5, 1  0.1, 0.7, 1  0.0, 0.0, 1  0.0, 0.0, 0  0.2, 0.4, 0  0.6, 0.6, 0  0.8, 0.2, 0 |
| --- | --- |

Looking at the printed data, there seems to be no change in accuracy respective to the change in learning rate (with learning rate changing by 0.1 every iteration), resulting in .88 accuracy across the board. This might be because the instructions did not specify a standard initial weight value. On running it with no standard initial weight, the accuracy does change, and as this is more interesting.



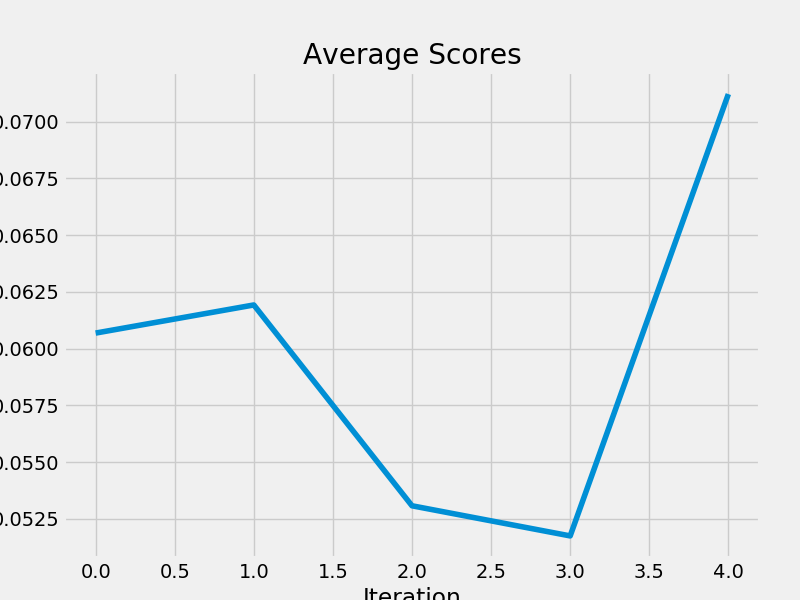
I must admit that I do not know why the axis labels are being cut off, even though they are being provided as per default. Perhaps it is which file format that I have chosen. But it is quite clear that the Separable dataset does not have the learning rate required for a good result, only 88% accuracy at a .1 rate. It could be just a little better. Meanwhile the Inseparable dataset had a 62% accuracy, which is not good at all.

# Task 3: Voting:

As stated above, I needed to figure out where I was doing the split of data into test and training sets. In the end, I used code similar to my shuffle code to randomly pick instances for one set or the other.

| Accuracy | 0.91 | 0.96 | 0.95 | 0.97 | 0.96 |
| --- | --- | --- | --- | --- | --- |
| Epochs | 3 | 4 | 5 | 5 | 3 |
| Test Accuracy | 0.91 | 0.91 | 0.93 | 0.95 | 0.93 |

Looking at the weights, it appears that the fourth feature (or the weights[3]) is the most significant, being usually greater than 1, with the 2nd, 8th, and 10th vying for second place. The later teens of features, along with the 1st feature are usually in competition for least significant.



# Conclusion:

While this was an interesting exercise, I did not allow myself the time necessary for either figuring out graphing nor for playing with sci-kit. I also spent too much time trying to implement methods/functions/features that would simply be extraneous. I think that it would be interesting to see how the other students implemented their stopping criteria as well as their scramble code. But as of now my entry is already late, and I do not wish to use up more of my late days.