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 instrucitons, 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 reccommended 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; seperableIsSquare is meant to classify if two lengths would make a square shape (note that length cannot be negative). My “impossible” dataset is recognizing “odd” decimal numbers, but “0” is classified as both.

| @relation seperableIsSquare  @attribute height real  @attribute length real  @attribute class {0,1}  @data  0.5, 0.5, 1  -0.2, 0.5, 0  -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, 0.3, 1 | @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.

[graph]

# 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 | 0.94 | 0.98 | 0.97 | 0.90 | 0.93 |
| --- | --- | --- | --- | --- | --- |
| Epochs | 3 | 3 | 4 | 4 | 3 |

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 11th weight being the next most. Apparently adoption of budget resolution and immigration were the major factors in this set.

[graph]

# 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 think that it would be interesting to see how the other students implemented their stopping criteria as well as their scramble code.