Nate Boldt
Professor Dligach
COMP479
October 3, 2021

Homework 2 Report

For this report, each question will be answered in its respective numbering order, and there will a conclusion at the end. In almost every case, the performance is measured by how many correct labels are seen in the program execution. Most of these cases are only tested once or twice – in a real world setting, we would want to repeat test these models and use different data to get a much more realistic performance estimate!

1. I created a test set of the following 2D data to analyze my Perceptron implementation:
    a. [(1, 3, 0), (2, 2, 0), (3, 3, 0), (2, 4, 0), (1, 7, 0), (5, 3, 1), (6, 2, 1), (7, 1, 1), (8, 10, 1), (9, 0, 1)] in which the first two are the cartesian coordinates and the last digit is the label.
    b. My code finds the 'threshold'/decision boundary based on the training data and randomized weights that will be used for predictions. Since the weights are random and the data set is pretty small, the program is not deterministic. For one run, the threshold was 0.378 and the weights were 0.3578 and -0.2350.

```
The weights for this training experiment are: [0.35785199048593264, -0.23503717881644481]
The threshold for this training experiment is: -0.3780000000000003
```

    c. The perceptron implementation is 100% accurate on the training data, which makes sense as this data has been seen before and trained on.

```
The tuple: (1, 3, 0) is predictive of label 0
The tuple: (2, 2, 0) is predictive of label 0
The tuple: (3, 3, 0) is predictive of label 0
The tuple: (2, 4, 0) is predictive of label 0
The tuple: (1, 7, 0) is predictive of label 0
The tuple: (5, 3, 1) is predictive of label 1
The tuple: (6, 2, 1) is predictive of label 1
The tuple: (7, 1, 1) is predictive of label 1
The tuple: (8, 10, 1) is predictive of label 1
The tuple: (9, 0, 1) is predictive of label 1
```

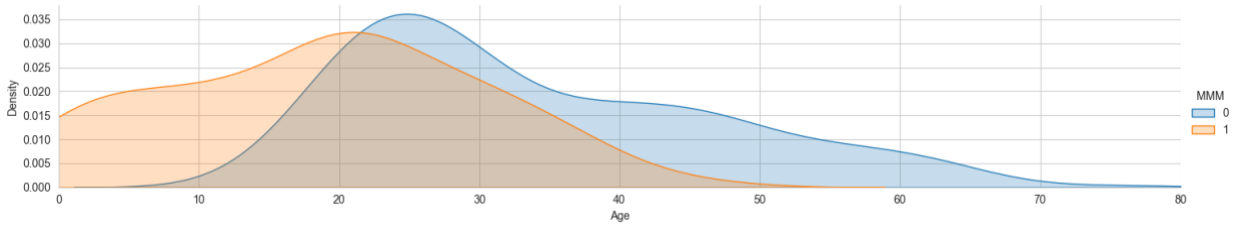
2. The Perceptron model did not converge when handed non-linearly-separable data as expected. It is impossible for the Perceptron to converge with non-linearly-separable

data because there will always be an error in the training set and the weights will update forever, unless we put in logic to allow a certain amount of errors.

   a. The data used is the following: [(1, 3, 1), (2, 2, 0), (3, 3, 1), (2, 4, 0), (1, 7, 1), (5, 3, 0), (6, 2, 1), (7, 1, 0), (8, 10, 1), (9, 0, 0)]
   b. The accuracy of the Perceptron model on this non-linearly-separable data is 30%. This is not surprising because of the explanation above – since we did not converge, the model is basically guessing in the dark. We also see variance in labels on the training data across different runs – which we did not see in the linearly-separable data(where the accuracy is always 100%).

```
The weights for this non-linearly-separable training experiment are: [-0.0043119741964652535, 0.03854968430294801]
The threshold for this non-linearly-separable training experiment is: -0.10799999999999998
The tuple: (1, 3, 0) is predictive of label 1
The tuple: (2, 2, 0) is predictive of label 0
The tuple: (3, 3, 0) is predictive of label 0
The tuple: (2, 4, 0) is predictive of label 1
The tuple: (1, 7, 0) is predictive of label 1
The tuple: (5, 3, 1) is predictive of label 0
The tuple: (6, 2, 1) is predictive of label 0
The tuple: (7, 1, 1) is predictive of label 0
The tuple: (8, 10, 1) is predictive of label 1
The tuple: (9, 0, 1) is predictive of label 0
```

3. The Adaline implementation I chose is actually one I found online from someone specifically targeting this model type at the titanic data. Within it, they examined different combinations of traits and the percentages of survival likelihood of them, which I thought was interesting!
   a. The performance of this model is 76.5%, which was found by generating an output that the Kaggle titanic website could ingest and analyze. This is about 10% better than the make-shift model I implemented in homework 1, so a significant improvement!
   b. This model uses the gradient descent, not SGD.
   c. This model attempts to convert non-integer features into workable data, and it normalizes some of the features as well.
   d. This code provides a cost graph as we aim to minimize the cost – we can see that as the weights update, the cost trends down logarithmically.
   e. The output of the test data is stored in **titanic.csv**
4. The most predictive features of this model are:
   a. Age – This graph shows the people who survived (orange) versus people who did not (blue). Clearly, passengers 10 and under were able to survive at a much higher likelihood than passengers 60+.



b. The other highly impactful feature is sex – women had a much higher chance of survival than men across many different analyses. A perfect example is this probability distribution comparing sex and number of siblings – while number of siblings has little to no effect across the sexes(see women from Pclass 1 with 0 and 1 siblings having a little over 2% difference), sex has a much larger impact than number of siblings (see women having 40+ percentage points on men across most categories).

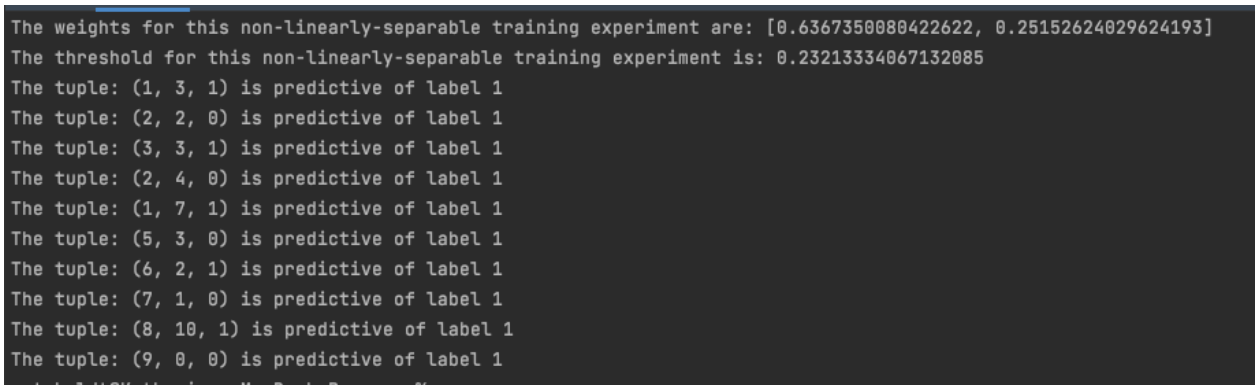
```
Percent of survival under each condition:
Pclass                 1         2         3
SibSp female
1     1         0.955556  0.906250  0.380952
      0         0.470588  0.250000  0.155844
0     1         0.979592  0.931818  0.592593
      0         0.329545  0.118421  0.129630
```

5. I set up a baseline model using randomized weights and a random threshold to obtain the following performance metrics:
    a. On linearly separable data, we have an accuracy of 30%

```
The weights for this linearly-separable training experiment are: [0.021887563389403453, 0.7357644459606648]
The threshold for this linearly-separable training experiment is: -0.9890554628768503
The tuple: (1, 3, 0) is predictive of label 1
The tuple: (2, 2, 0) is predictive of label 1
The tuple: (3, 3, 0) is predictive of label 1
The tuple: (2, 4, 0) is predictive of label 1
The tuple: (1, 7, 0) is predictive of label 1
The tuple: (5, 3, 1) is predictive of label 1
The tuple: (6, 2, 1) is predictive of label 1
The tuple: (7, 1, 1) is predictive of label 0
The tuple: (8, 10, 1) is predictive of label 1
```

    b. With a new set of randomized weights, on non-linearly separable data, we have an accuracy of 50%. In this case, it looks like the model is exclusively picking the '1' label.
    c. In both cases here, the performance of the base line is worse than the Perceptron model when applied to linearly separable data.
    d. When it comes to non-linearly-separable data, the performance is the same between the base line and Perceptron, which makes sense.
    e. The Adaline model performs significantly better on the titanic data than the base line on either data set.

f. Base line on non-linearly separable:



Extra Notes / Conclusion:

The Perceptron model clearly performs well within the constraints of linearly separable data – and shows that it can converge in these cases. However, when the data is not perfectly aligned (like in most real world examples), it falters to even recognize its own training data. The Adaline model certainly performs well on the larger Titanic data set, and the implementation I used provides some insight into how each feature affects the 'final tally'.

I included some Adaline implementations that I attempted before choosing this version, but they are tucked away in the **unused_titanic_adaline_implementation_attempts** folder.