CS 478

Scott Leland Crossen

Perceptron Project

Definitions

Part 1

The code that implements the perceptron rule is included with the submission of this project. It is written in Scala and follows the functional paradigm.

Part 2

Below are the true constructed ARFF files. the first is linearly separable and the second is not.

```
% This is linearly seperable
@RELATION custom1
@ATTRIBUTE fact1        Continuous
@ATTRIBUTE fact2        Continuous
@ATTRIBUTE class        {True, False}

@DATA
-0.6, -0.6, True
1.0, 0.0, False
0.5, -0.75, False
-0.5, 0.8, True
0.0, -1.0, True
1.0, 0.5, False
0.75, 1.0, True
0.75, -0.1, False
```

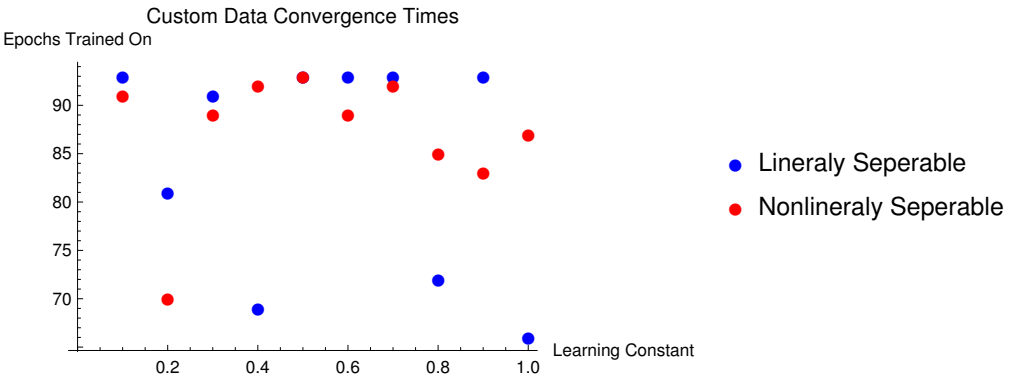
```
% This is not linearly seperable
@RELATION custom2
@ATTRIBUTE fact1        Continuous
@ATTRIBUTE fact2        Continuous
@ATTRIBUTE class        {True, False}

@DATA
-0.15, -0.2, False
0.05, 0.8, True
0.65, 0.2, True
-0.5, -0.75, True
-0.05, 0.2, False
-0.25, 0.05, False
-0.75, 0.1, True
0.05, -0.1, False
```

Part 3

For the purposes of this project, the stopping criteria is considered to be fulfilled when a continuous 5 epochs of the training set improved the weight vector by less than a cumulative 1.5 (half of the amount of features) across the feature vector times the learning constant. This value was chosen so that a

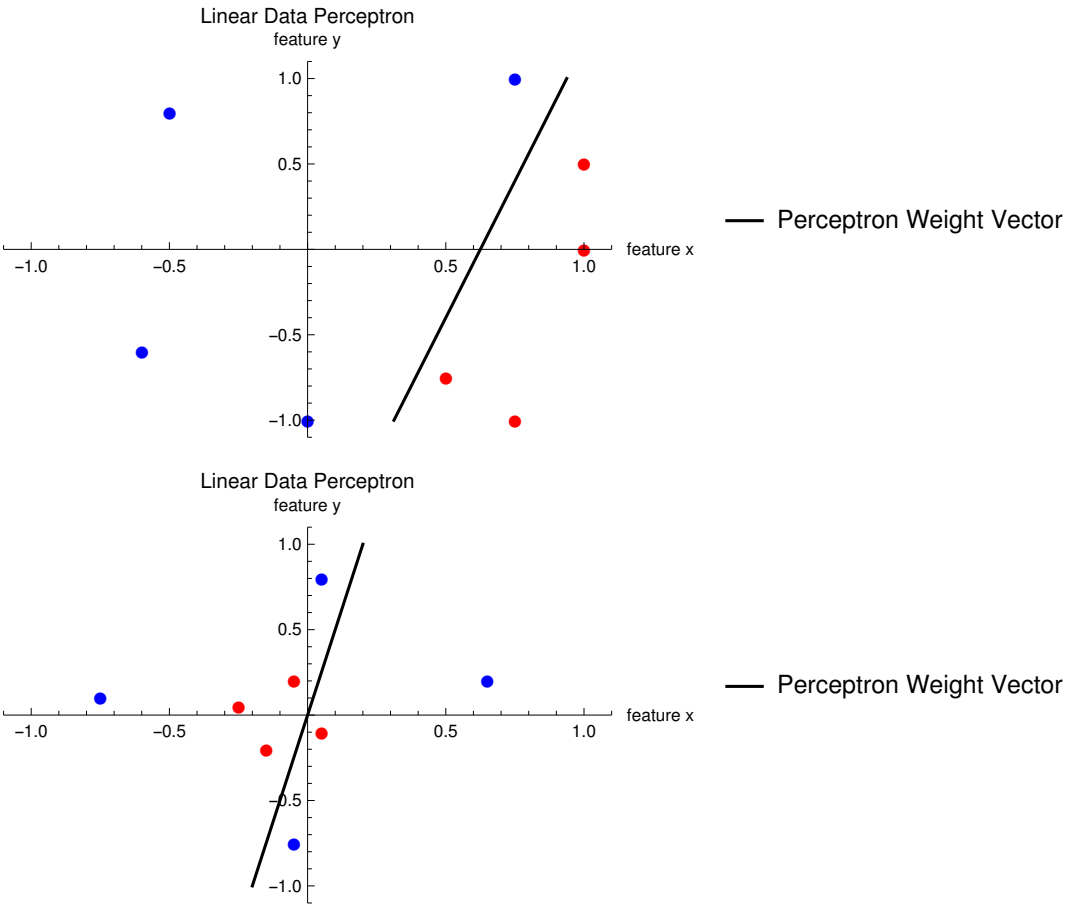
smaller learning constant would be taken into account and allow the algorithm to arrive at a more precise result. Moreover, under this value a larger feature set allows more ability for the algorithm to change before stopping. This is why the overall equation is written this way.

The results show that the learning constant doesn't really affect the amount of time it takes to train the perceptron. Data is given below showing the amount of epochs it takes to converge vs the learning constant. For a given single learning constant it was observed that a larger learning constant resulted in more sporadic changes in the weight vector whereas a smaller learning constant was more gradual in achieving the optimal solution. This of course is probably a result of the more-volatile nature of a larger training constant.



Part 4

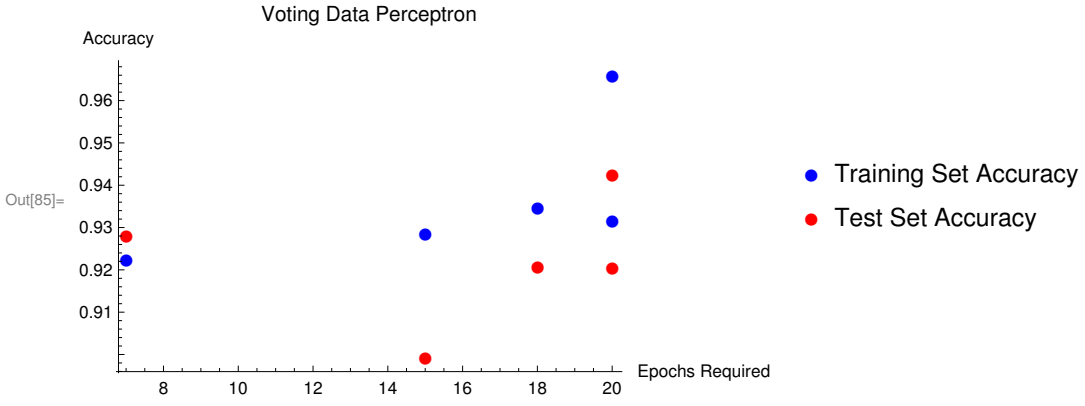
The below plots show the two different solutions to the above ARFF files. The first shows a correct separation of the data and the second shows a poor separation of the data (because it is not linearly separable)



Part 5

For a perceptron that is forced to train on all the data we found an accuracy in the 90s. The plots below show the results for 5 trials.

Out[85]=




```
The average amount of epochs trained on was 16.

The average training accuracy was 0.936646

The average testing accuracy was 0.92223
```

For this data set a good stopping criteria was determined to be when the perceptron reached 5 successive iterations of it's data set with no significant increase in the weight vector. A significant increase is

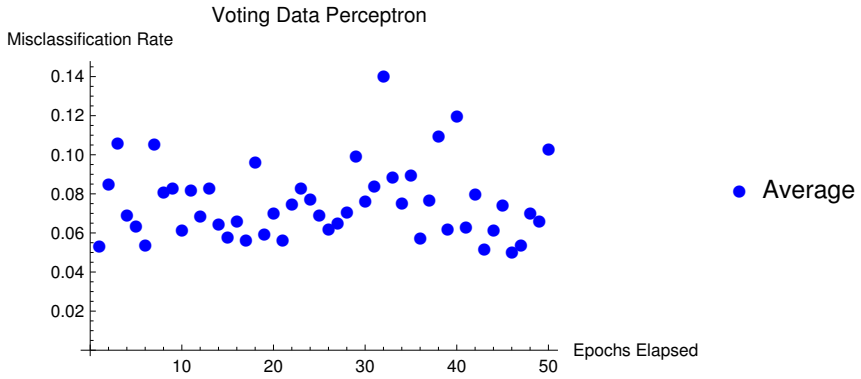
considered to be when the summed changed in the weight vector is more than 1/2 times the number of features times the training constant. The logic behind this algorithm is explained more fully in part 3 of this project.

The final weight vector for a perceptron with a test accuracy of 96 % turns out to be {0.0, 0.47336482815648373, 1.1834120703912094, -1.6567768985476934, 0.0, 0.23668241407824187, -0.23668241407824187, -0.23668241407824187, 0.9467296563129675, -0.47336482815648373, 0.7100472422347256, -0.47336482815648373, 0.47336482815648373, -0.23668241407824187, 0.9467296563129675, -0.23668241407824187, 0.47336482815648373}.
According to this data, there is a slight offset (bias) in the data. Moreover, the 3rd and 4th features seem to be the most influential with the 1st and 5th features being least influential. It's important to note that not all returned weight vectors had a bias. The most-accurate ones all did though. Note that the longer the perceptron was allowed to train the more accurate it tended to be. This is best shown by the upward-sloping trend in the above plot.

Most influential features: 3, 4
Least Influential features: 1, 5

Below is a plot of average misclassification rate vs epochs elapsed. It would appear that not much improvement happens after the first epoch. Though the graph below would have stopped much earlier than 50 epochs, I suspended the halting criteria to demonstrate the misclassification rate over a longer period. Again, I suspect that the misclassification rate would not be so noisy if I allowed for a smaller time frame. More than that, but it is likely that my perceptron was able to accurately train itself after one epoch and continuing the training was just overkill.

Out[83]=



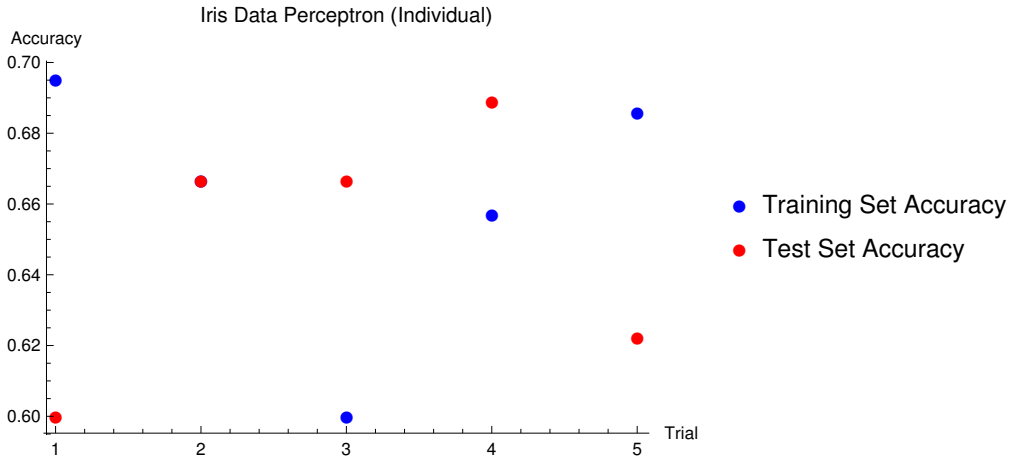
Part 6

For this part I created two different perceptrons to solve the iris data set. The two perceptrons are the same as described in the problem statement. The perceptron described in part (a) is called a individual-test perceptron. The perceptron described in part (b) is called the matched-pairs perceptron. Each implementation can be found in the code included in the submission of this project.

Each perceptron was allowed to train on 70% of the data and then test the remaining 30%. Only one epoch was used in order to demonstrate baseline viability. Stopping criteria was found not to affect the

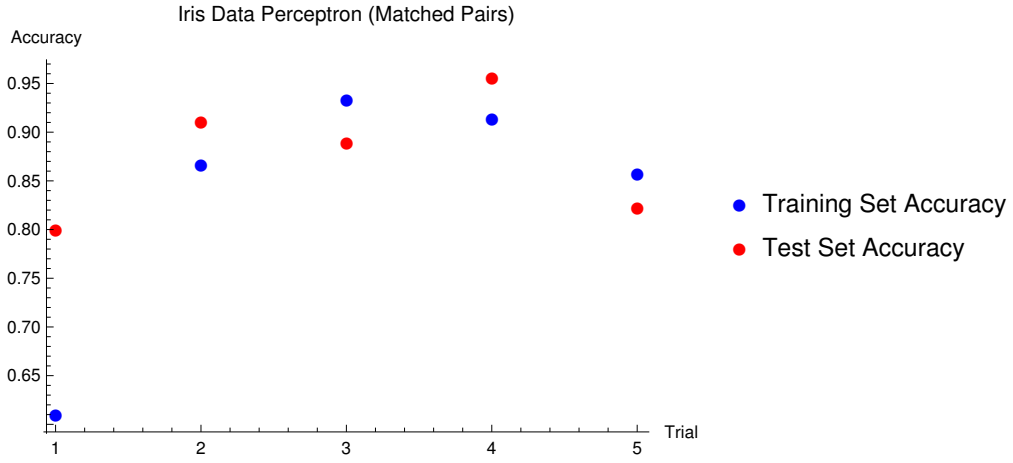
overall results and so was not included for consistency.

As shown by the data, the matched pairs perceptron achieved a higher and more stable accuracy compared to the individual perceptron model. In fact, it would appear as if in almost every case the matched pairs model performed better than the individual selection model. Average accuracy is included for reference. I propose that the results are because the matched pairs experiment is a more defined experiment - The machine only needs to separate between two sets of data rather than 3 at a time. For example, it is conceivable to have three sets of data with one set in the middle. A individually-selecting perceptron would not be able to separate this middle portion from the two outside portions whereas the matched-pairs perceptron could. This is why the matched-pairs perceptron is likely better.




```
The average training accuracy was 0.660952
```

```
The average testing accuracy was 0.648889
```




```
The average training accuracy was 0.83619
```

```
The average testing accuracy was 0.875556
```