Michael Fryer
C.S 478
May 14, 2019

Perceptron

The Algorithm

Overall, the Perceptron algorithm was fairly straightforward to implement. However, there were a few challenges. One difficulty was trying to keep the design scalable for different implementations such as the 3 class classification. Another difficulty arose from making bad assumptions about the structure of the data that could be fed into the machine.

The Two ARFF Files

Creating the ARFF files gave a good opportunity to learn a bit more about the ARFF format structure. I learned that ARFF is a standard for some research groups.

Learning Rates

Linearly Seperable Learning Rate Preformance

Learning Rate	Accuracy	Epochs
1e-07	1.0	4.0
0.001	1.0	4.0
0.1	1.0	4.0
0.2	1.0	4.0
0.5	1.0	4.0
0.8	1.0	4.0
0.99	1.0	4.0
0.99999	1.0	4.0

Linearly Inseperable Learning Rate Preformance

Learning Rate	Accuracy	Epochs
1e-07	0.625	4.0
0.001	0.625	4.0
0.1	0.625	4.0
0.2	0.625	4.0
0.5	0.625	4.0
0.8	0.625	4.0
0.99	0.625	4.0
0.99999	0.625	4.0

Mixed lin. Sep/Insep data

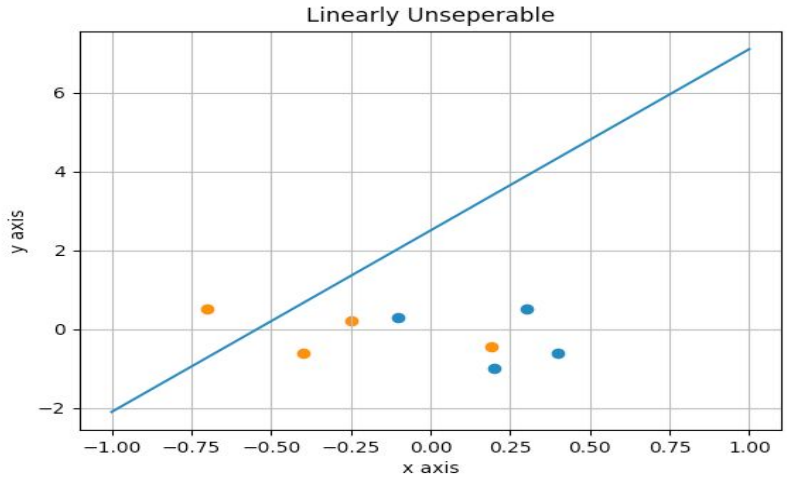
Learning Rate	Accuracy	Epochs
1e-07	0.875	3.0
0.001	0.75	4.0
0.1	0.875	3.0
0.2	0.875	3.0
0.5	0.875	3.0
0.8	0.875	3.0
0.99	0.75	4.0
0.99999	0.75	4.0

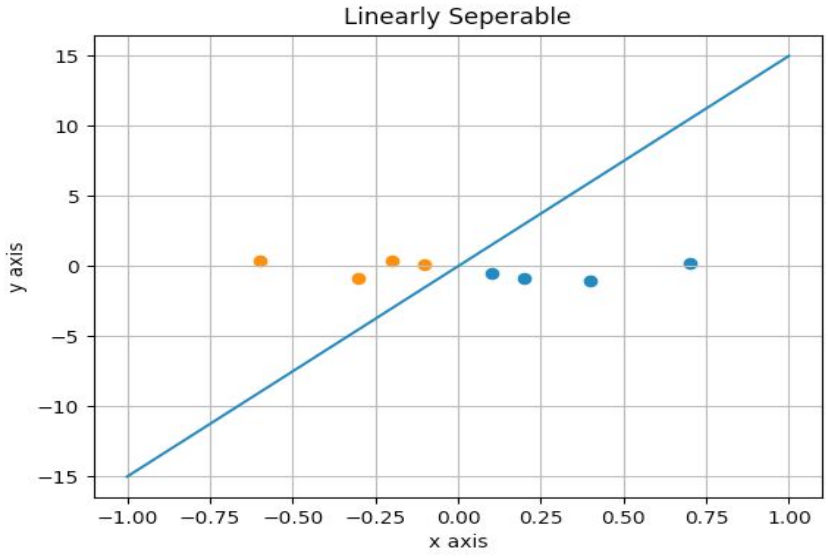
While training, the frequency of successful guesses was recorded for each epoch. At each successive epoch, variance of the corresponding frequency was measured in terms of all the other measured frequencies. Over time as the level of error of the Perceptron converges over the test set, the frequency variance diminishes. The Perceptron Rule was allowed to run so long as the variance of the success frequency stayed above a threshold and the quantity of epochs stayed below a given threshold.

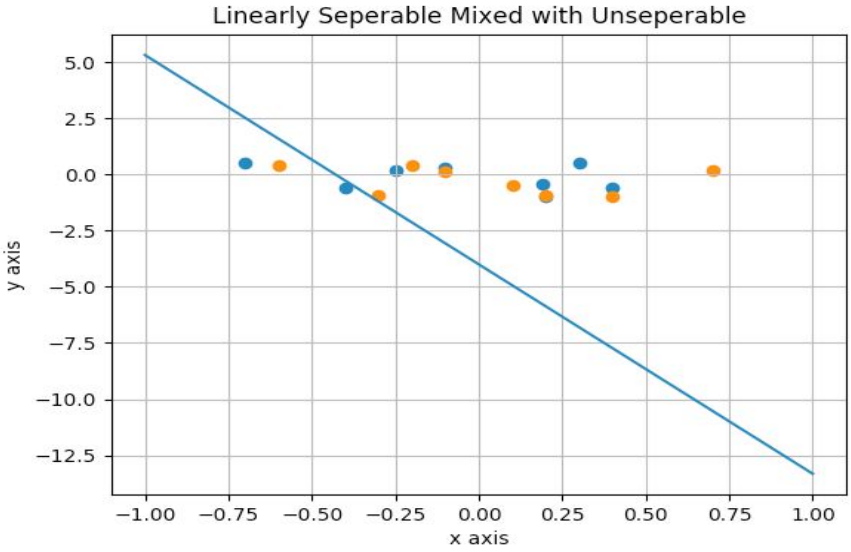
As seen from the tables above, the learning rates did not have a significant impact upon the Perceptron. Nonetheless, when both the linearly separable and inseparable datasets were combined, the behavior slightly differed. For example, when learning rate was very close to 1, the Perceptron Rule overstepped the optimum value and soon after, the step sizes began to converge faster than the accuracy.

The number of epochs stayed at 4 for both the linearly separable and inseparable sets. When the two sets were combined, the number of epochs dropped to 3 on average unless the learning rate was very high or very low.

Graphs of Linearly Separable/Inseparable



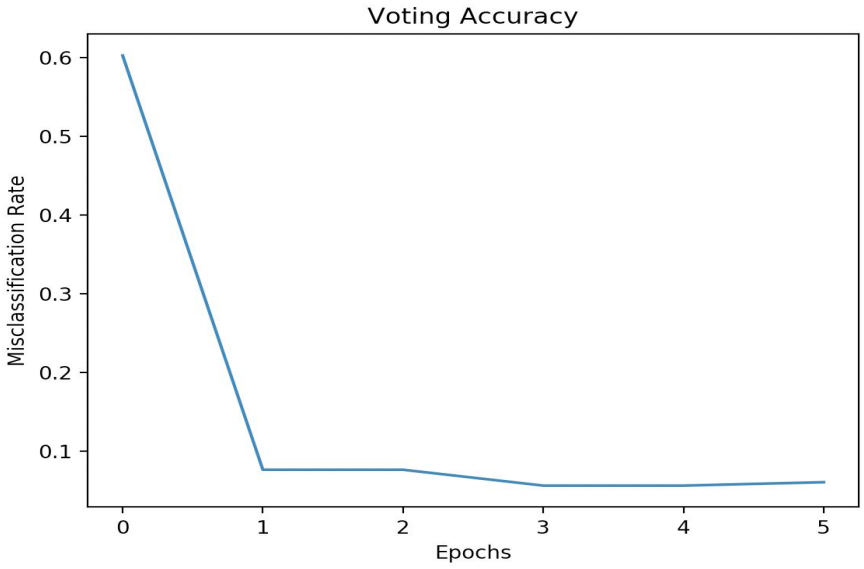

Linearly Seperable
15
10
5
0
−5
−10
−15
y axis
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
x axis


Linearly Seperable Mixed with Unseperable
5.0
2.5
0.0
−2.5
−5.0
−7.5
−10.0
−12.5
y axis
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
x axis

Voting Task

Voting Accuracy

	Test Set Accuracy	Training Set Accuracy	Average Accuracy	Epochs
split:1	0.95683	0.96273	0.95978	4.0
split:2	0.92806	0.9441	0.93608	3.0
split:3	0.94245	0.9472	0.94483	3.0
split:4	0.95683	0.95652	0.95668	3.0
split:5	0.94964	0.95342	0.95153	3.0
Avg	0.94676	0.9528	0.94978	3.2



It was surprising how fast the average misclassification rate of the Perceptorn converged with the given voting task. It seems reasonable to assume that this occurred because of the limited output space of the task and the consistent policy polarization of political parties with regards to policy. It appears as though Perceptrons could be effective in general for voting tasks as long as the data holds consistent patterns.

Important voting themes corresponding to the weights with the most magnitude include: superfund right to sue, export administration act south africa, handicapped infants and education

spending. The least important voting themes included: physician free freeze, synfuel corporation cutback, and mx-missles.

Experimenting

For the experimentation section, I built a three class image classifier with with the Perceptron. The classes consisted of facial images of my wife, my brother, and myself. Each image was converted to grayscale, resized, and then normalized. Columns of the image pixel values were then stacked into a single column vector. The these image vectors were used to train three different perceptrons that corresponded to each pair of classes. For classification, an image was sent through all three perceptrons and then the guess from the perceptron with the least uncertainty was used.

The overall accuracy of the three class perceptron averaged at about 65%. The heavily weighted weights for the predicting corresponded to pixels in the top mid region of the images. I believe this was largely due to my wife having much darker hair than me and my brother. Thus, the pixel magnitudes corresponding to her hair tended to be much larger. On the contrary, my brother and I look very much alike and so there is no isolated radius of pixels in our images that would be very different. It appears that as a result, the accuracy between predicting me or my brother was around 50%. Training the algorithm on images from just my brother and I appears to give little structure or visible patterns in the weights being used.

In conclusion, it seems as though Perceptrons could have a minimum amount of efficacy in image classification if the images being classified have consistent large radii of differing average pixel values such as the radius surrounding my wife's hair and mine.

ARFF FILES

@relation linearlySeparable
@attribute a1 real
@attribute a2 real
@attribute class {0,1}
@data
-0.6, 0.4, 1
-0.3, -0.9, 1
-0.2, 0.4, 1
-0.1, 0.1, 1
0.1, -0.5, 0
0.2, -0.9, 0
0.7, 0.2, 0
0.4, -1, 0

@relation linearlyInseparable
@attribute a1 real
@attribute a2 real
@attribute class {0,1}
@data
-0.7, 0.5, 1
-0.4, -0.6, 1
-0.25, 0.2, 1
0.19, -0.45, 1
-0.1, 0.3, 0
0.4, -0.6, 0
0.3, 0.5, 0
0.2, -1, 0