Perceptron Report

Wesley Ackerman

Effect of Learning Rate on Perceptron

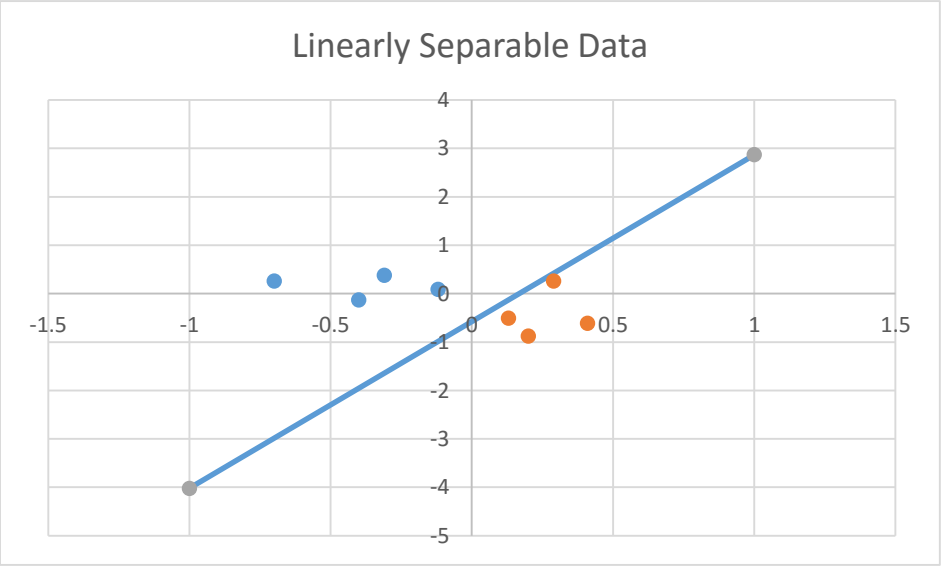
Learning rate had very little effect on the success of the perceptron's learning and number of epochs it took to stabilize. With the linearly separable dataset, it took 13-20 epochs with a learning rate of 0.1, around 15 epochs with a learning rate of 1, around 14 epochs with a learning rate of 5 to 10. There appeared to be more variability in the number of epochs needed for smaller learning rates. With a learning rate of 0.1, it was common to get epoch counts of 8 and 15. With a learning rate of 10, the algorithm almost always took 14 epochs. I think this is because with a smaller learning rate, it may take longer for the random weights to be corrected to their final value, because much smaller incremental changes are being made.

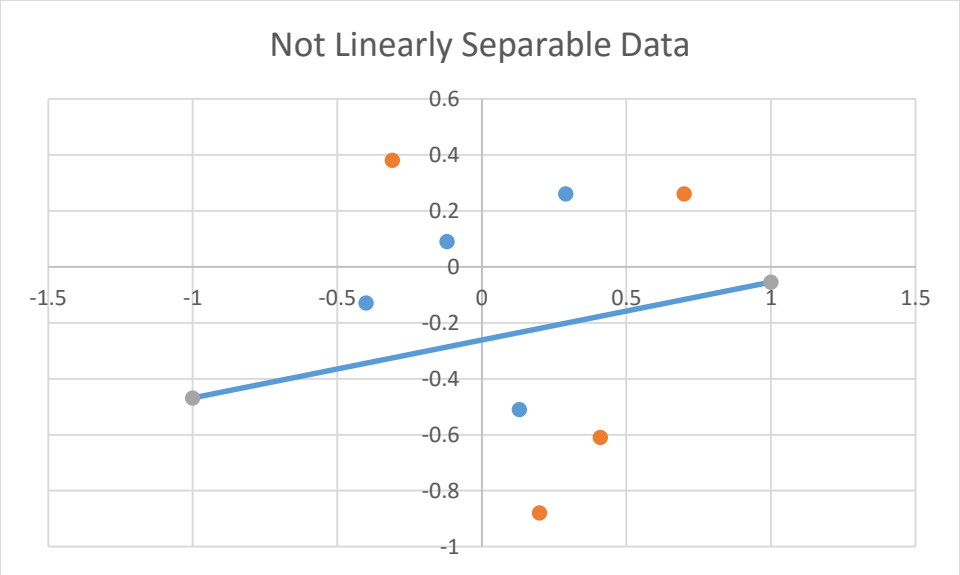
With the non-linearly separable dataset, learning rate had even less effect. Whether the learning rate was 0.1, 10, or in between, the perceptron took around 13 epochs to converge. I think this is because the perceptron isn't complex enough to be able to learn a complex solution surface. It has difficulty obtaining any amount of improvement on each epoch, since it is essentially guessing.

Stopping Criteria

My stopping criteria is as follows: after each epoch, I measured the number of instances correctly guessed divided by the total number of instances. This was then compared to the best percentage of instances guessed correctly that has been obtained so far. If the new percentage correct is better, the "epochs without improvement" variable is set to zero. Otherwise, the variable is incremented. Once the variable reaches ten, the perceptron exits. So, the algorithm runs until there are ten epochs in a row without improvement.

Simple Dataset Graphs

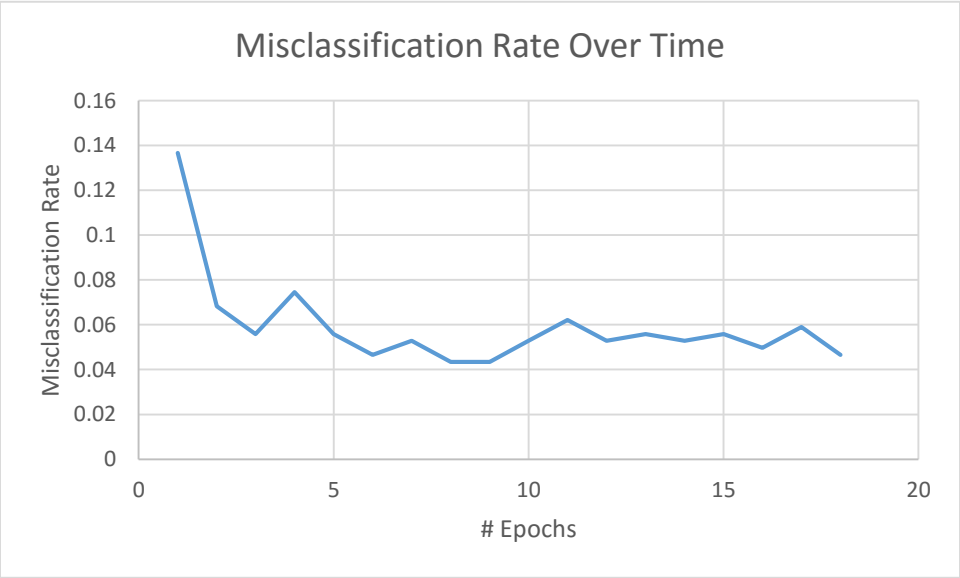




The perceptron was unable to linearly separate data that is intermingled between the two types.

Voting Task Performance and Analysis

	Number of Epochs	Avg Training Set Accuracy	Avg Test Set Accuracy
	19	0.97205	0.94964
	26	0.97205	0.96403
	16	0.98137	0.94964
	24	0.95963	0.94245
	19	0.91615	0.92086
Average	20.8	0.96025	0.945324



This graph shows the oscillation of misclassification rate. It starts as a relatively large value, and decreases over time. Once the smallest misclassification rate is reached, around epoch 7, ten more epochs are performed. Since no improvement was obtained, the algorithm ended.

The model learned the largest weights for features 3 (+1.82870) and 8 (-1.0086). Since feature 3's weight has the greatest value, it has the most bearing on whether the congressman is a democrat or a republican. Since it is positive, it has a strong positive correlation with the output. This means that voting "yes" for the "Physician's fee freeze" is a strong indicator that the congressman was a republican. Since feature 8 has a negative learned weight, it has a strong negative correlation with the output. This means that voting "yes" for "MX-Missile" is a fairly strong indicator that the congressman was a democrat. Features 1 (-.01652) and 15 (.04908) have the smallest weights, which means they have the least correlation with a congressman's party. This means that whether or not a congressman voted "yes" for "Handicapped Infants" and "Export Administration Act South Africa" has little bearing or correlation with their party affiliation.

Iris Classification

I chose to use the perceptron to learn the iris task. I took the route of creating one perceptron for each pair of iris types. Learning rate was kept at 0.1, and the algorithm ran until no improvement was gained for 10 epochs. I learned it is definitely harder to use a perceptron when there are more than 2 output classes. I had to edit the ARFF files by hand to make them usable.

Classifying Setosa vs. Virginica was easy for the perceptron to learn. Those two types must have features that are fairly distinct from each other. On average, it took about two epochs for the perceptron to be able to get 100% accuracy predicting between them. The petal width feature had the largest learned weight, which means that petal width holds the most bearing in deciding whether the iris is a Setosa or a Virginica type.

Classifying Setosa vs. Versicolor was similarly easy to learn. It took 2-3 epochs on average for the perceptron to reach 100% prediction accuracy. The petal length feature had the largest learned weight, which means that petal width holds the most bearing in deciding whether the iris is a Setosa or Versicolor type.

Classifying Setosa vs. Virginica and Setosa vs. Versicolor are both linearly separable problems. The perceptron is able to converge on a correct solution line that completely divides the types of irises.

Classifying Versicolor vs. Virginica was harder for the perceptron. The differences between these two types must be more nuanced. On average, accuracy on the training set was 0.9238, and accuracy on the test set was 0.8889. Deciding whether a specimen is a Versicolor or a Virginica appears to be a harder task, and they overlap enough that they aren't completely linearly separable.