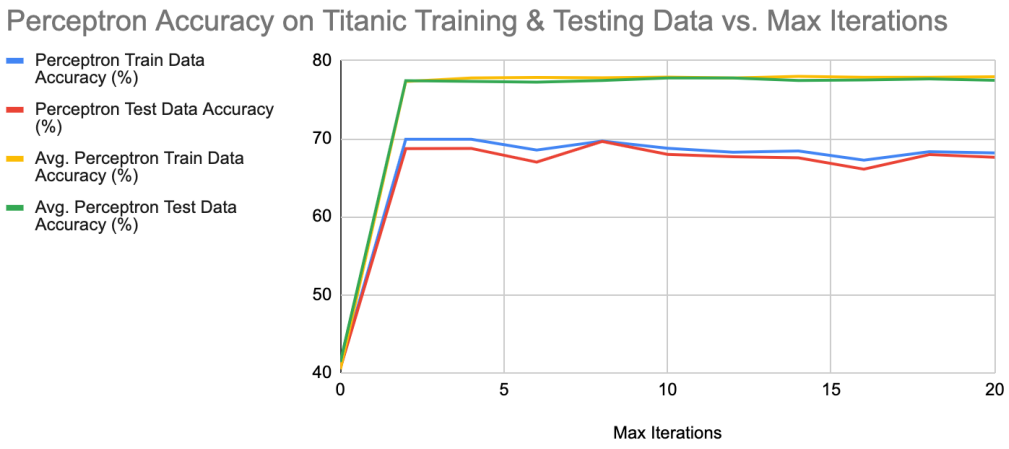
David D'Attile
Prof. Kauchak
CS158
6 February 2022

Assignment 03: Perceptron Experiment Writeup

1. *What is the accuracy of the two perceptron classifiers on the Titanic data set? To calculate this, generate a random 80/20 split, and train the model on the 80% fraction and then evaluate the accuracy on the 20% fraction. Repeat this 100 times and average the results:*

   The perceptron classifier achieved a classification accuracy of ~69.24% (9,902 correct classifications of 14,300 total attempts) whereas the average perceptron classifier achieved a classification accuracy of ~77.87% (11,136 correct classifications of 14,300 total attempts).

2. *Calculate the training and testing accuracies on the dataset as in the previous question (average 100 experiments) for increasing number of training iterations (start: 0, end: 20, increments of 2).*
   a. *Include a table or graph with the number of iterations and the training and testing accuracy:*



   b. *To do proper experimentation, we cannot look at the testing dataset before. Assuming this experimental setup (without a development set), we'd pick the best number of iterations based on the training accuracy.*

Based only on the training accuracy, what iteration limit would you pick? How does this compare to the best number of iterations on the test set? Give a 1-2 sentence explanation.

The best training accuracy in my experiments was achieved at a max iteration value of 2 for the regular perceptron model (~69.94% accuracy) and a value of 14 for the avg. perceptron model (~78% accuracy), and the best testing accuracy was achieved at a max iteration value of 8 for the regular perceptron algorithm (~69.64% accuracy) and a value of 12 for the avg. perceptron algorithm (~77.80% accuracy). In the regular perceptron algorithm, there seemed to be a wider iteration gap between best training/testing accuracies, whereas the avg. perceptron algorithm peaked after nearly the same iteration count. Regardless, after 2 iterations the classification accuracies remained relatively stable (+/- 2%), so based on these results I'd select a maximum training iteration of 10 (between 2 and 14).

c. *Do we see overfitting with this data set? Assuming we don't fix the iteration limit based on the training data, does the model tend to overfit? Include a 2-3 sentence explanation.*

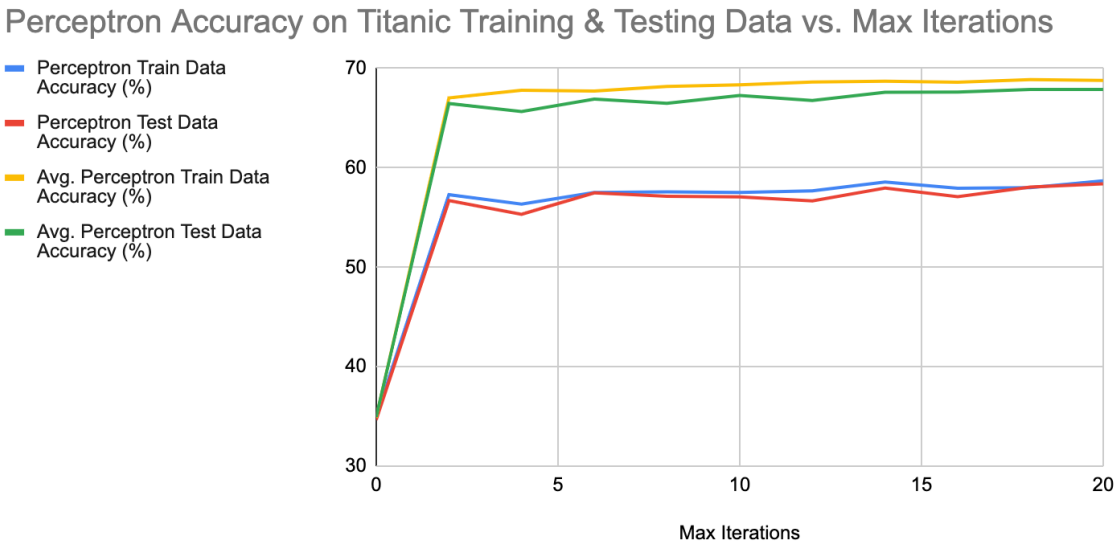
The regular perceptron model overfitted to the training data slightly after the 8th iteration, but the classification accuracy for both testing and training data fell slightly for higher maximum iterations. The avg. perceptron model was more resistant to overfitting, and testing/training accuracy stabilized after the first couple of iterations. This reinforces the in-class notion that voted/avg. Perceptron learning algorithms tend to avoid overfitting!

3. *Use the ClassifierTimer class to compare the runtime performance of the three classifiers we've implemented (decision tree and the two perceptron variants). Do the results make sense? Include a 1-2 sentence explanation.*

   After running the ClassifierTimer tests a few times, I found that the decision tree algorithm was slowest (~0.004s training), the regular perceptron algorithm placed in the middle (~0.0018s training), and the average perceptron algorithm was routinely the fastest (~0.0016s training) - it's worth noting that the regular and avg. perceptron models occasionally swapped places, but the DT model was consistently the slowest. For all models, the classification speed was continuously returned as 0.00s. The DT results make sense given its repeated calculation of the most useful feature to split on and its recursive implementation, but I was a bit confused about why the avg. perceptron model training tended to be faster than the regular perceptron model even though its train() method is slightly more complicated.

4. *Find another dataset with at least 5 features and 500 examples and provide a comparison of your perceptron algorithms on it.*

   I found a binary classification dataset containing Pima Native American diabetes diagnoses (https://machinelearningmastery.com/standard-machine-learning-datasets/) with 768 examples and 8 real-valued input features per example. I quickly tweaked the labels such that "-1" represented no diagnosis rather than "0" and attempted to run the perceptron classifiers against this dataset. The results are graphed below - the avg. perceptron algorithm fared better but failed to achieve a >70% classification accuracy.



(Side note - I think I was able to implement the AveragePerceptronClassifier as an extension of PerceptronClassifier successfully editing only the .train() method :))