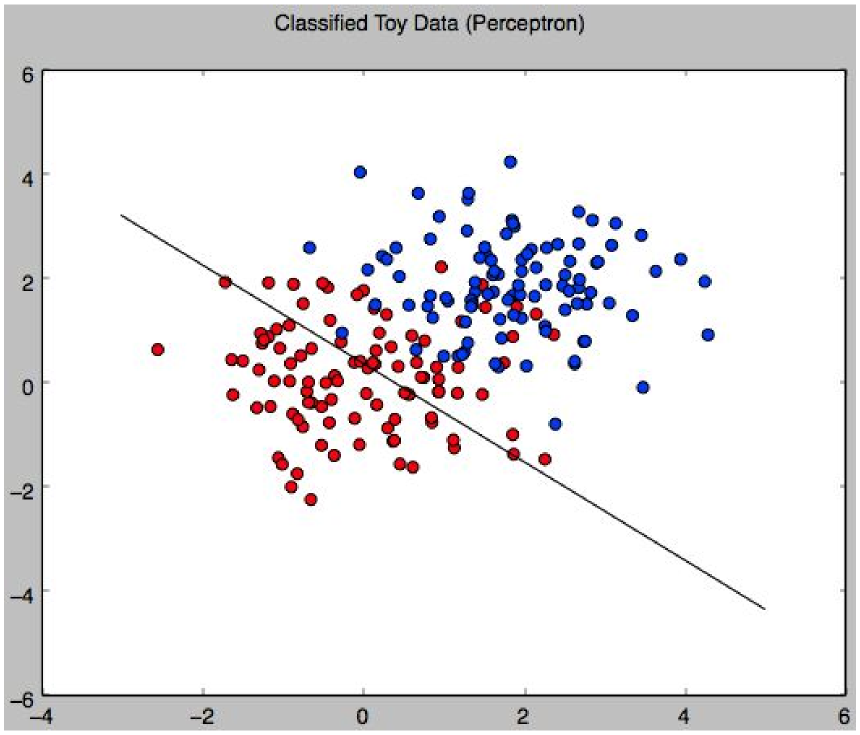
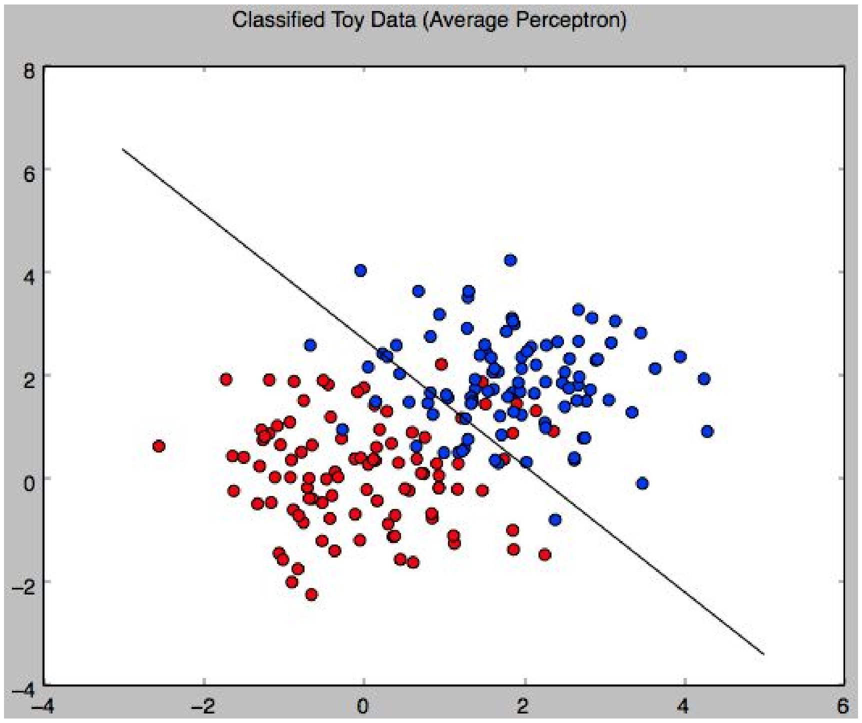
Birkan Uzun
birkanu@mit.edu
No Collaborators

6.036 Project 1: Automatic Review Analyzer

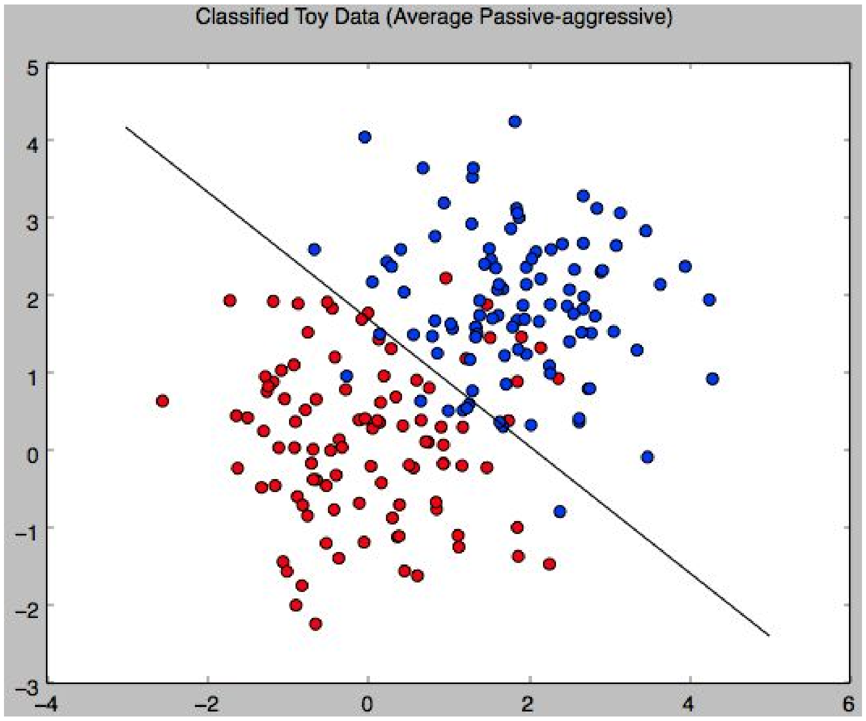
Section 1.7

Below are the plots obtained for each algorithm:





Birkan Uzun
birkanu@mit.edu
No Collaborators



As it can be seen from the plots, the algorithms each provide different decision boundaries. With the linearly inseparable data, the Average Perceptron emphasizes parameters that seem to work longer than others by averaging them. That is why it is different than the regular Perceptron. On the other hand, the Average Passive-aggressive is different because it tries to minimize the Hinge loss while keeping the parameters close to where they were prior to seeing another data point.

Section 2.9

b) Below are the training and validation accuracies of each algorithm with $T = 5$ and $\lambda = 1$:

- Training accuracy for perceptron: 0.9760
  Validation accuracy for perceptron: 0.8229

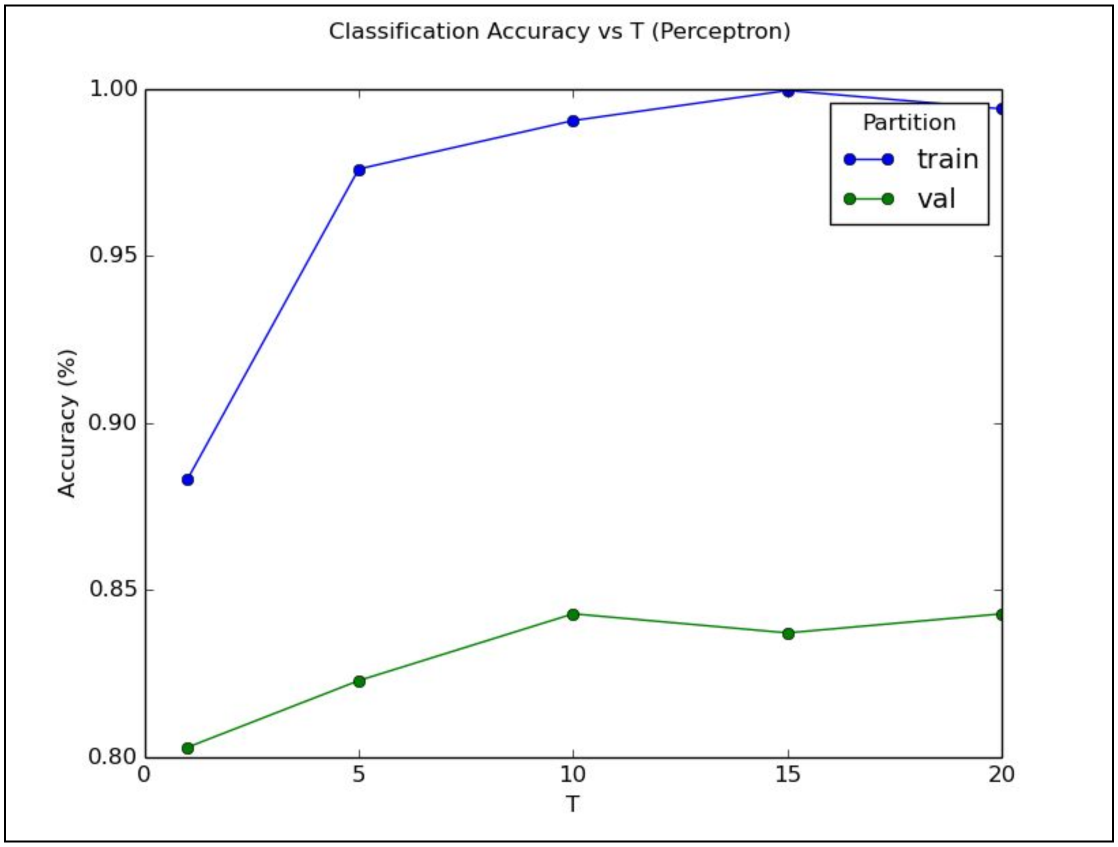
- Training accuracy for average perceptron: 0.9810
  Validation accuracy for average perceptron: 0.8286

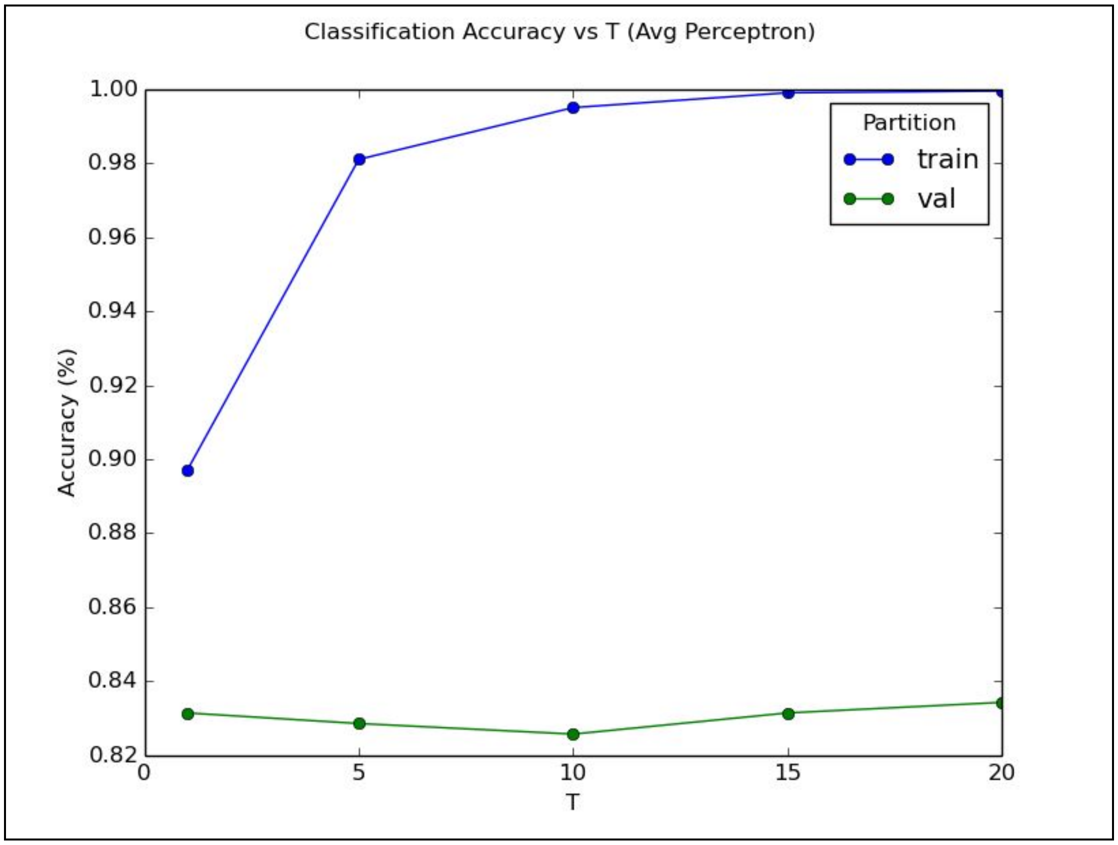
- Training accuracy for average passive-aggressive: 0.9805
  Validation accuracy for average passive-aggressive: 0.8429

Section 2.10

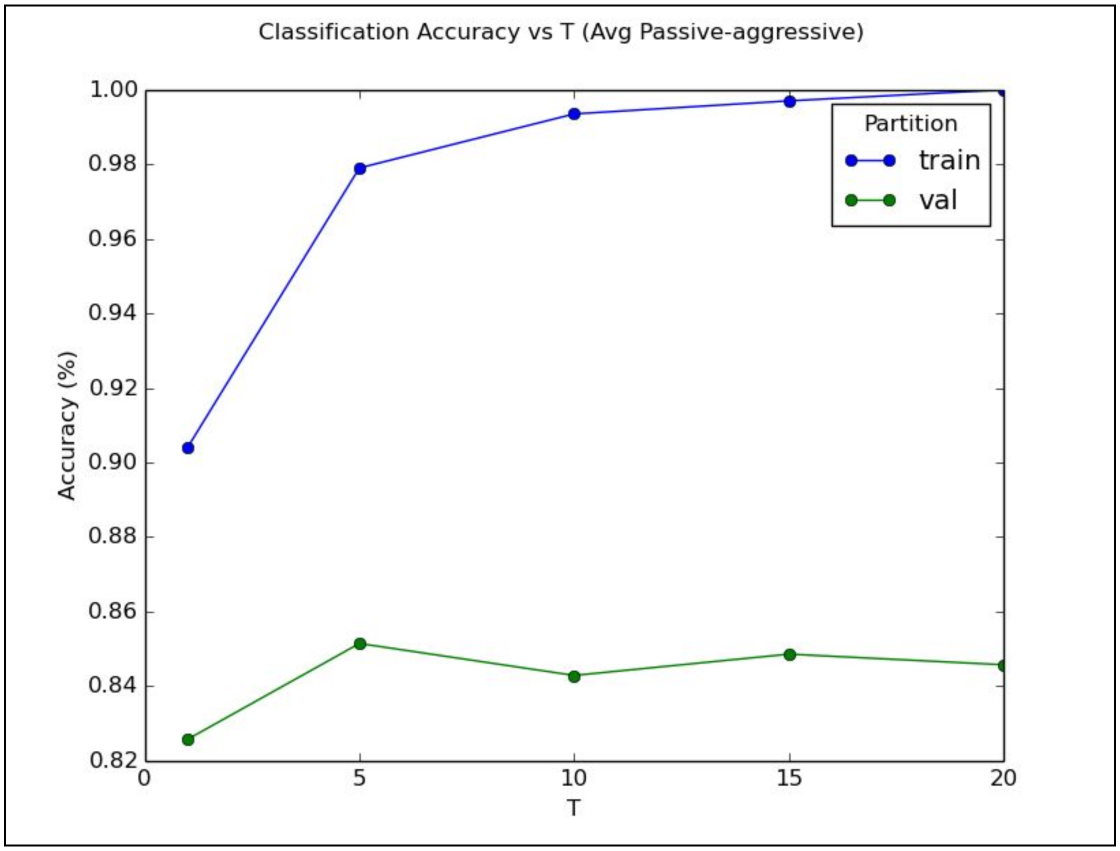
Below are the plots generated by `main.py`:

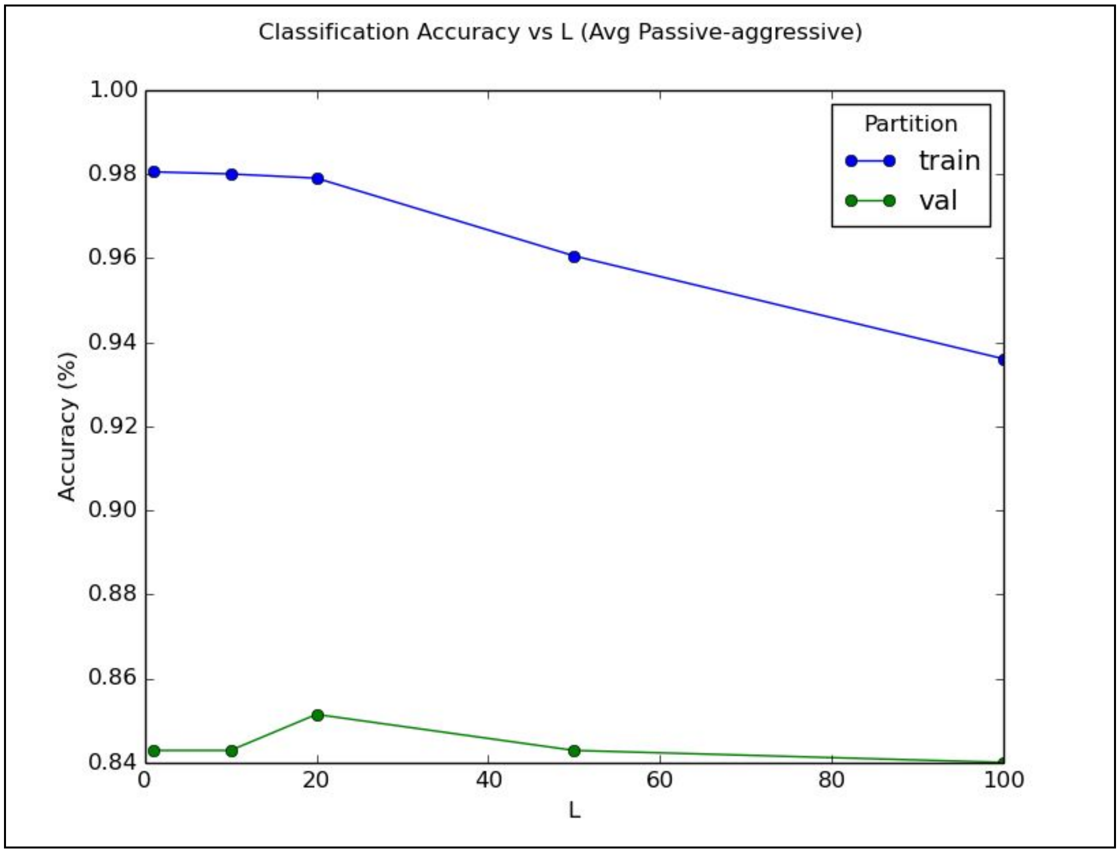
Birkan Uzun
birkanu@mit.edu
No Collaborators





Birkan Uzun
birkanu@mit.edu
No Collaborators





Birkan Uzun
birkanu@mit.edu
No Collaborators

a) Looking at the plots, it can be said that they behave somewhat similarly as a function of λ and T. For the regular Perceptron and the Average Perceptron, we can see that the accuracy improves as T is increasing. For the Average Passive-aggressive Algorithm, it can be seen that lower values of λ and T produce higher accuracies.
b) The Average Passive-aggressive Algorithm performed the best.
c) The optimal values are $T = 5$ and $\lambda = 20$, and they produced an accuracy of 85.14% on the Average Passive-aggressive Algorithm. For the regular Perceptron and the Average Perceptron, $T = 20$ produced the highest accuracy.

Section 2.11

a) The accuracy obtained by using the Average Passive-aggressive algorithm with $T = 5$ and $\lambda = 20$ was 84.57%.
b) Top ten most explanatory word features:

- For the test data: `['!', 'you', 'great', 'delicious', 'can', 'favorite', 'and', 'have', 'excellent', 'that']`
- For the training data: `['best', 'great', 'delicious', 'perfect', 'wonderful', 'excellent', 'love', 'highly', 'use', '!']`

Section 3.12

1) First improvement that I tried was to remove common, or stop, words from the unigram-bigram dictionary. Removing the stopwords from the dictionary is good because it removes irrelevant words that could act as noise, and thus, reduce overfitting. Below is the code that shows the changes:

```
# Create a set of stopwords
stopwords = open("stopwords.txt").read().splitlines();
stopwords = Set(stopwords)
dictionary = {} # maps word to unique index
for text in texts:
    word_list = extract_words(text)
    for word in word_list:
        # Only add into dictionary if the word is not in stopwords
        if word not in dictionary and word not in stopwords:
            dictionary[word] = len(dictionary)
return dictionary
```

I first ran the experiments in Section 2.10 so that I could find the best λ and T that result in the highest accuracy. The Average Passive-aggressive algorithm with $T = 5$ and $\lambda = 20$ gave the best accuracy. Therefore, I conducted the same experiment as described in Section 2.11 (a) where I ran the Average Passive-aggressive algorithm with $T = 5$ and $\lambda = 20$ to compare the accuracy results of the training data and the test data. The accuracy this time improved to 85.14%. The code can be found under Section 3.12 of `main.py`.

Birkan Uzun
birkanu@mit.edu
No Collaborators

2) As a second improvement trial, instead of just adding 1s and 0s to the feature vector values, I added the number of occurrences of each word to put more weight on the words that appeared a lot. For the same example where we have “Mary loves apples and apples and apples”, our dictionary would be {Mary, loves, apples, and} and the feature vector would be (1, 0, 3, 2). I ran the same set of experiments that I explain in Section 3.12 (1), however, this time the accuracy did not improve at all, i.e. remained 85.14%. I believe this is because same words do not occur that many times in a sentence to actually affect the accuracy. In other words, most reviews must have had unique words in them once the stopwords were removed.