Dimitris Koutentakis
6.036 Project I

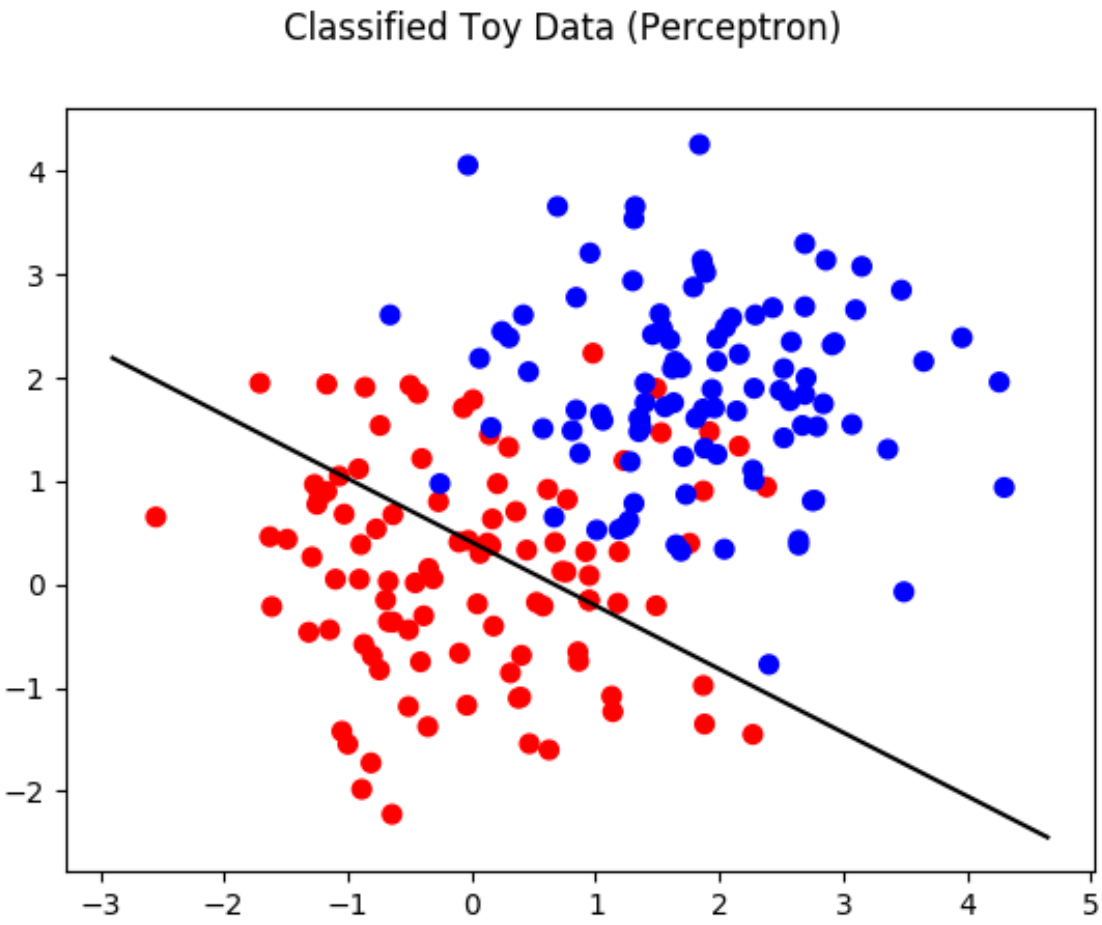
PART I

1)

vii)

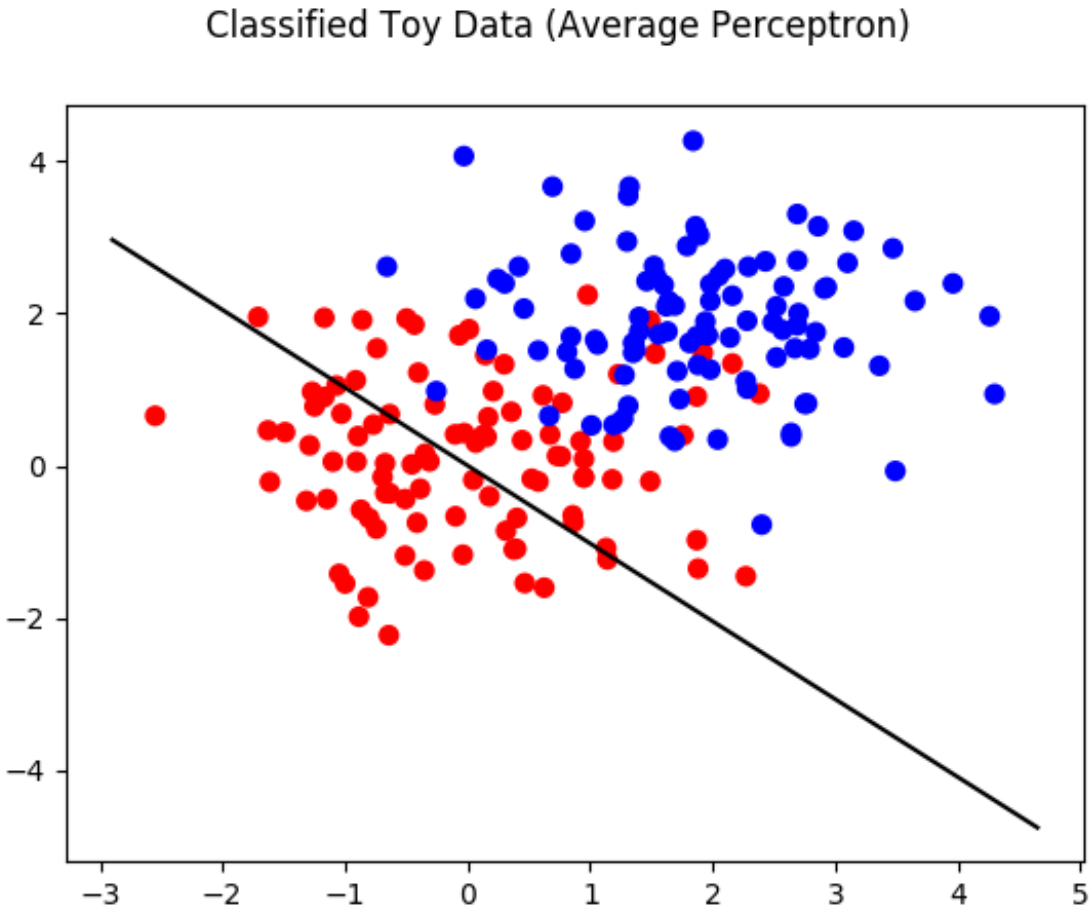
As is evident from the plots, the three algorithms give us slightly different results when classifying the data. This is because each one of them uses a slightly different approach in order to make the classifications.

The first algorithm (Perceptron) produces the following result:



It is obvious from the above graph that the separator does a relatively good job and separating the points, but is a bit skewed away from the blue points.

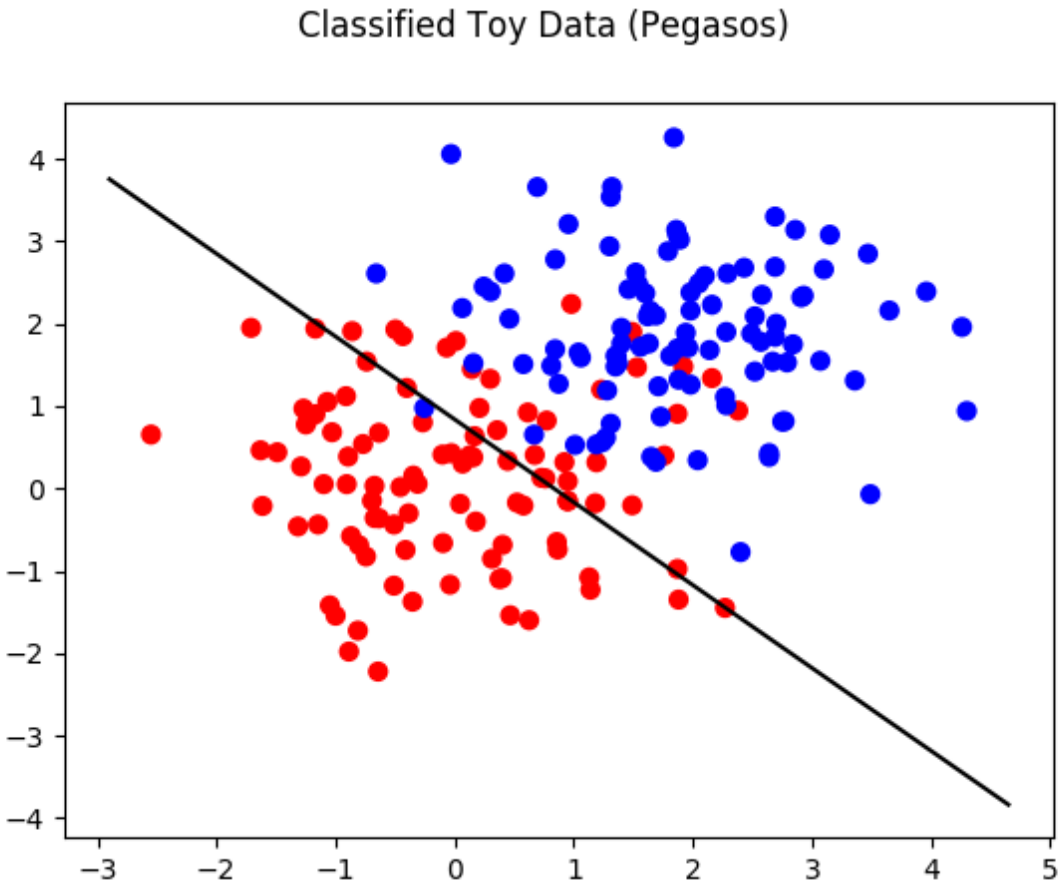
The Average Perceptron algorithm has a similar output, but does a bit of a better job at classifying the points correctly. The graph follows below:



We can see from the graph that the line is a bit better than before at classifying the points, but still skewed away from the blue points.

Finally, we have the Perceptron algorithm, which produces the following output:

Dimitris Koutentakis
6.036 Project I



It is obvious that this algorithm is by far superior at classifying the points. This is largely because of the more sophisticated approach used by the Pegasos algorithm, that is the gradient optimization used.

Dimitris Koutentakis
6.036 Project I

PART II

9)

b)

After implementing the "accuracy" and "classify" functions, we can check how good our algorithms are at classifying the training points or non-training points.

The results of the accuracies for each algorithm can be found in the following table:

Algorithm	**Training Accuracy**	**Validation Accuracy**
Perceptron (Normal)	95.93%	82.40%
Perceptron (Average)	97.42%	84.60%
Pegasos	91.55%	83.00%

From the results of the above table, we can see that on the training points the Average Perceptron has the highest accuracy, followed by the Normal Perceptron and lastly followed by the Pegasos algorithm. However, in the validation points, Average Perceptron scores the highest accuracy, while Pegasos comes second and Normal Perceptron third.

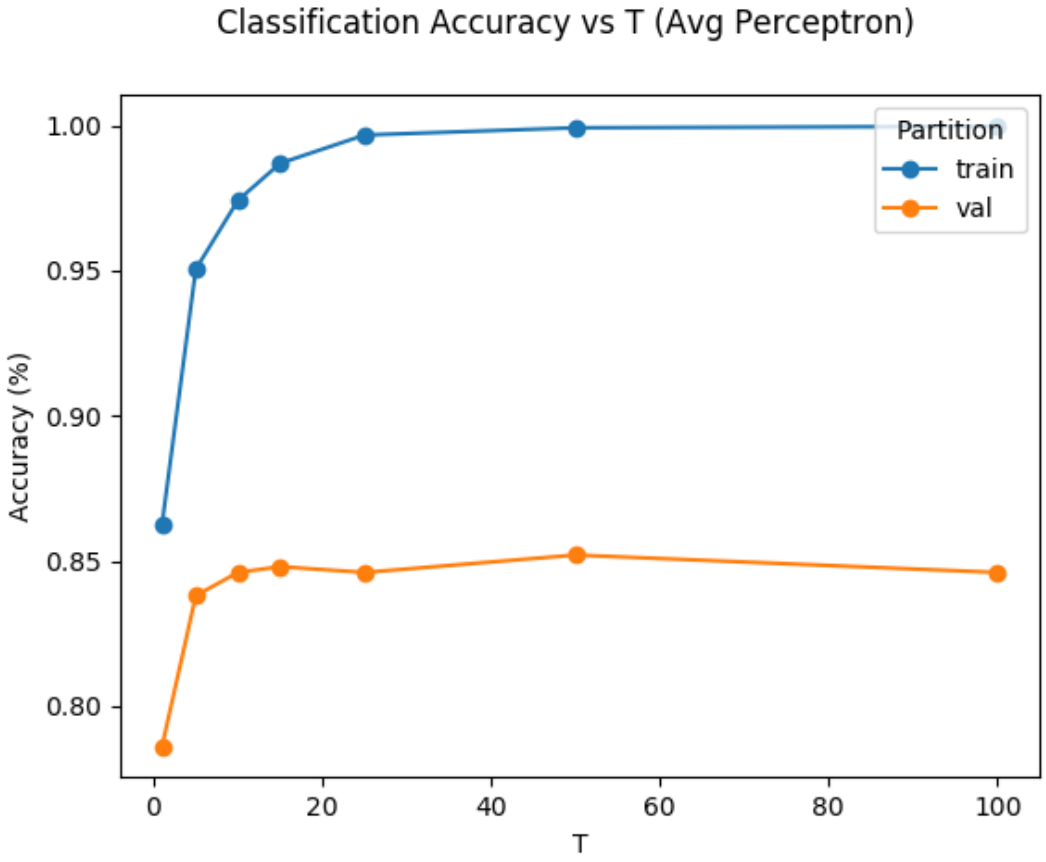
The results viewed in this table are a bit different than those seen visually in the graphs of the previous section. That is mostly due to the change in λ. In the previous section we had a λ of 0.2 and here we used 0.01.

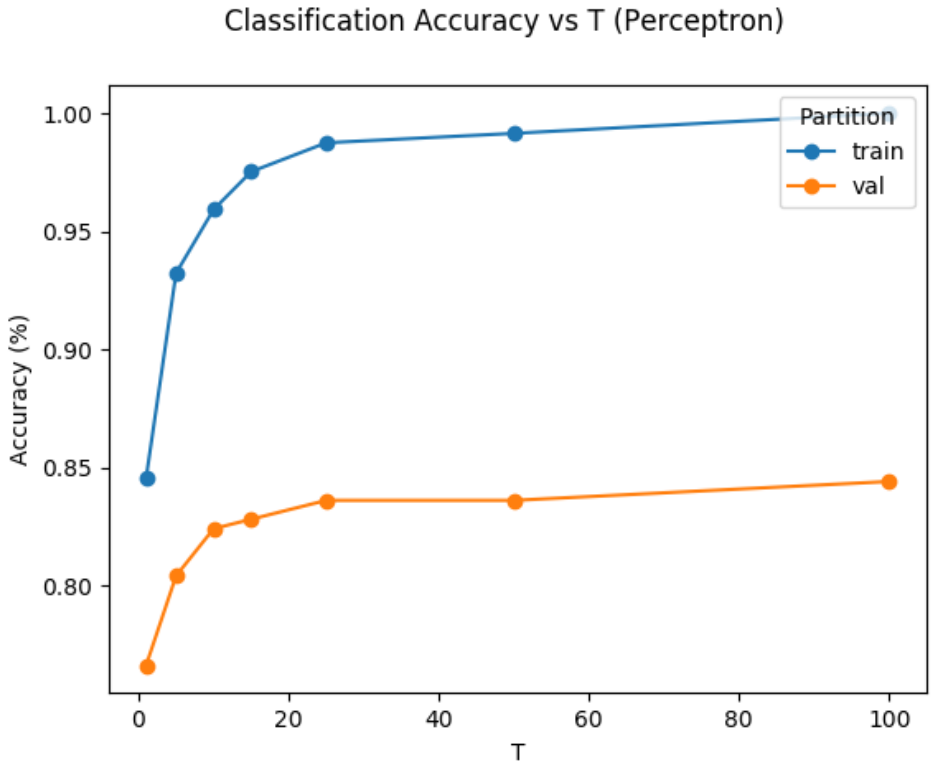
10)

In order to find the best performing T and λ, I started with λ=0.01 and then plotted the accuracies vs T. It was then easy to find from the plots the best T for each algorithm. After finding the best T, I made new plots and saw that the best λ was in fact 0.01.

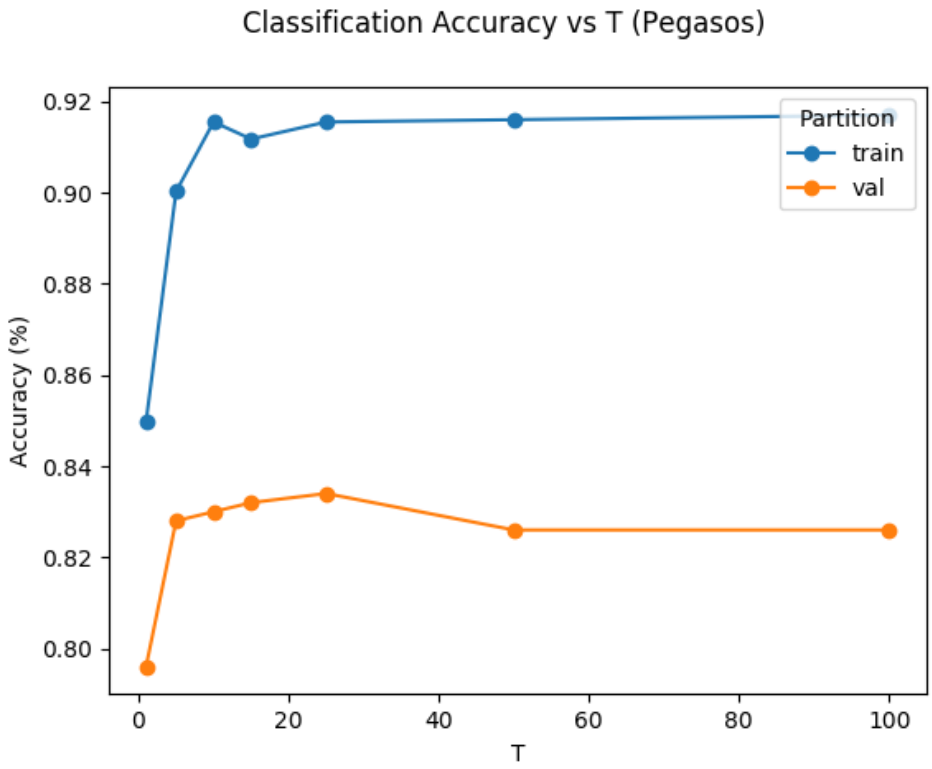
The plots that resulted initially:

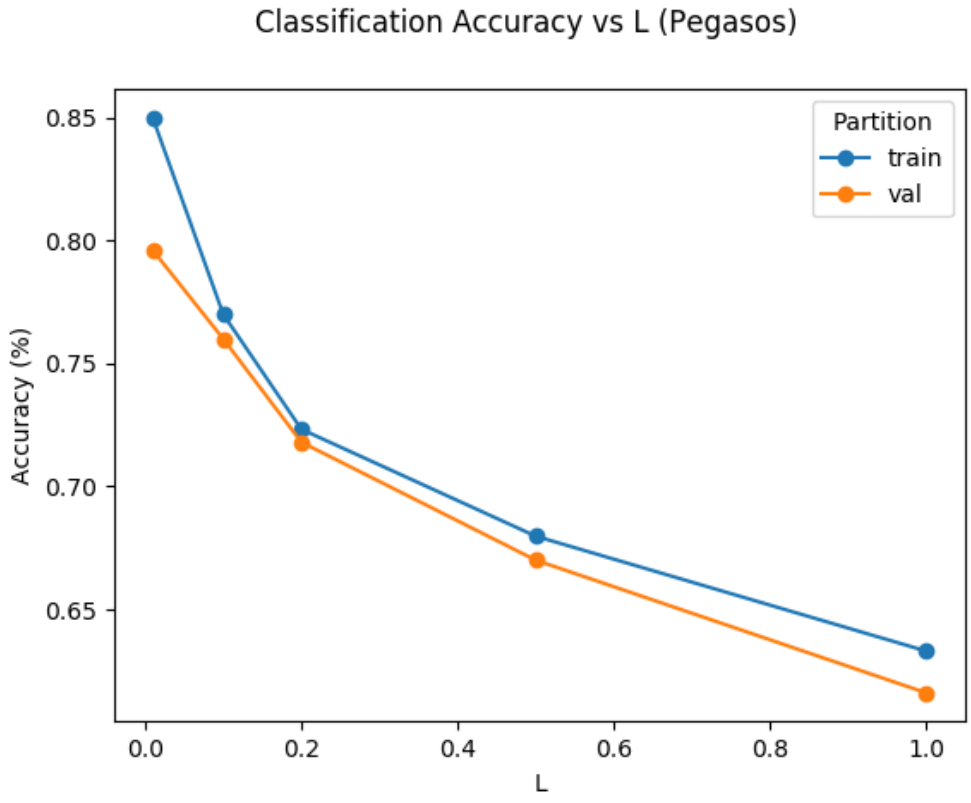
Dimitris Koutentakis
6.036 Project I





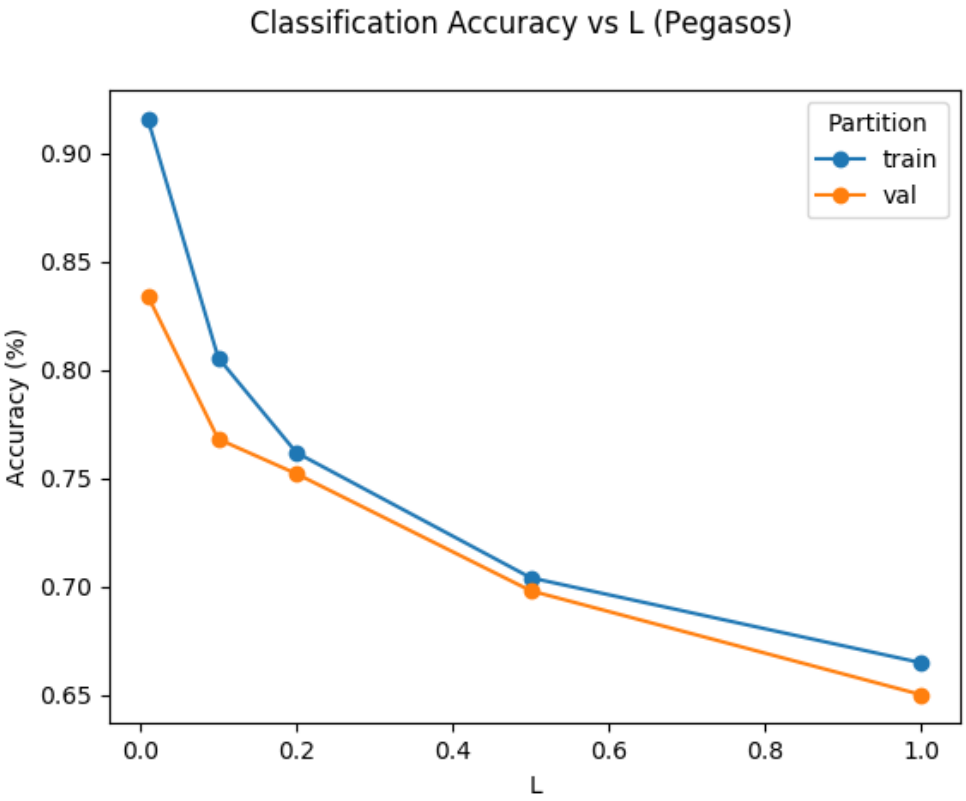
Dimitris Koutentakis
6.036 Project I





Dimitris Koutentakis
6.036 Project I

With T fixed at 25, the last plot becomes:



a)

The Training and validation accuracies do not have the same values. However, as demonstrated in the graphs above, one closely follows the other as function of λ and T. This is reasonable when we think that one set of points is the subset of the other. Since the training and the validation points are of the same type, an algorithm optimized for the training points should be good at classifying the according validation points (as long as it does not overfit).

b)

From the plots above, it is obvious that the best accuracy is obtained by the Average Perceptron Algorithm at right around 85% on the validation data.

c)

The optimal values for the three algorithms can be summarized in the following table:

Dimitris Koutentakis
6.036 Project I

Algorithm	**Best_T**	**Best_λ**
Average Perceptron	50	N/A
Normal Perceptron	100	N/A
Pegasos	25	0.01

11)

a)

From running the best performing algorithm at the optimal parameters, we get that the Average Perceptron achieves an accuracy of 99.675% on the training set and 84.59% on the validation set.

b)

According to the best_theta of the average perceptron algorithm, the top ten most explanatory words are:

['delicious', 'amazing', 'wonderful', 'glad', 'plan', 'helped', 'method', 'pleased', 'canned', 'dessert'].

Dimitris Koutentakis
6.036 Project I

PART III

The changes I performed were in the functions “bag_of_words”, “extract_words”, “extract_bow_feature_vectors”, and “extract_additional_features”. More specifically the changes I made were:

- Include bigrams and trigrams in the dictionary (union, equally weighted)
- Not include any of the stop words in the dictionary
- Add a threshold and a limit of the number of times a word is used (not less than three times and not more than five times the number of reviews
- Normalize the feature vectors
- Add additional feature of whether the review length is above or below average.

With these changes, I saw a very small change in performance, but I expect it to be much more generalizable, and perform better on other sets of data.

The new results I got were: **99.75**% accuracy on the training data and **84.19**% accuracy on the validation data.

My ten new most explanatory words are:

['delicious', 'great', 'best', '!', 'love', 'always', 'wonderful', 'amazing', 'excellent', 'tasty'].