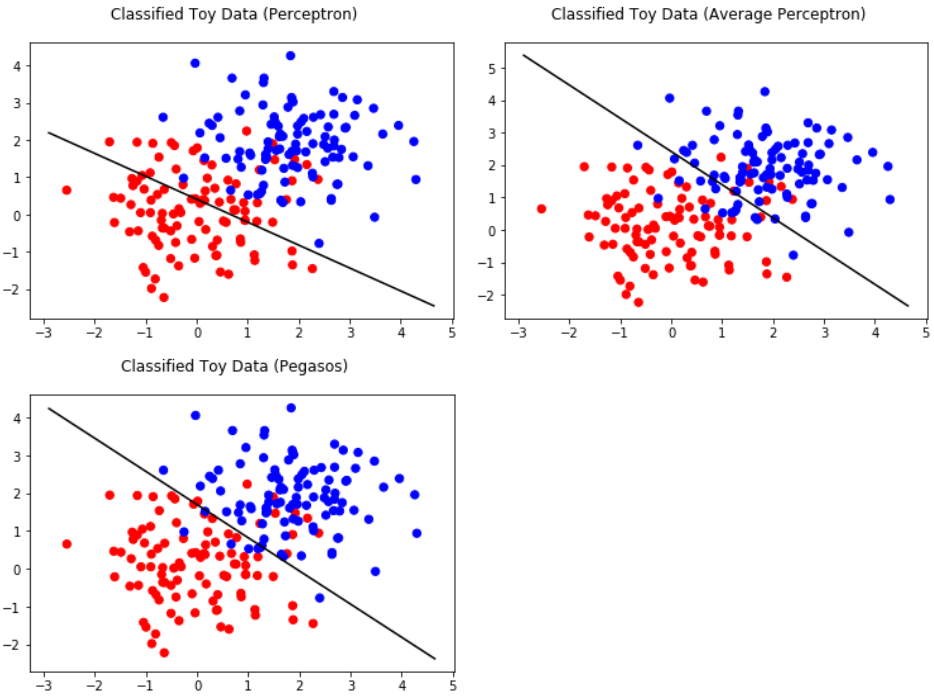
OLADEJI Olamide: MIT 6.862/6.036 Spring 2016
Project One.

Answers
Question 1.7

Plots at T=10 and L=0.2 on toy_data



Why do these algorithms provide different decision boundaries?
The algorithms provide different decision boundaries because they achieve the final value theta differently. For instance, the final value of theta for the averaged perceptron differs from the theta final value of the standard perceptron because in the averaged perceptron, theta_final is an average of different values of Theta after update. The Pegasos value for theta (and consequently the decision boundary) is different because Pegasos updates theta differently and it (theta) is a function of an extra variable (lambda) not included in the perceptron algorithm.

Question 9ab: Accuracies
T=10, Lambda = 0.01
Perceptron:
Training accuracy for perceptron: 0.9593
Validation accuracy for perceptron: 0.8240

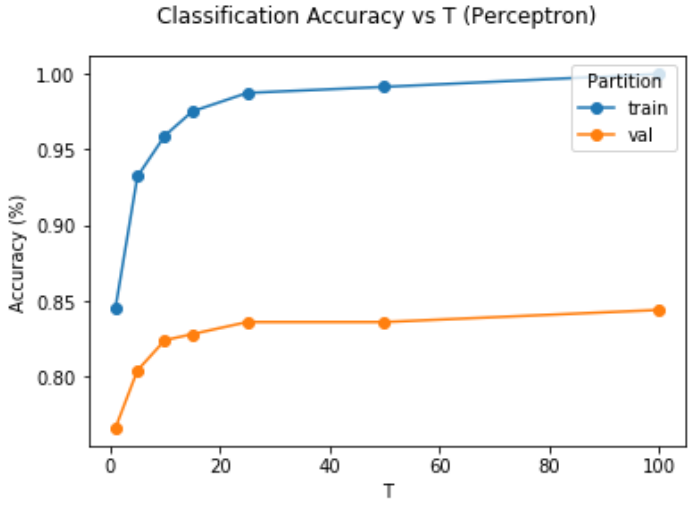
Average Perceptron:
Training accuracy for average perceptron: 0.9760

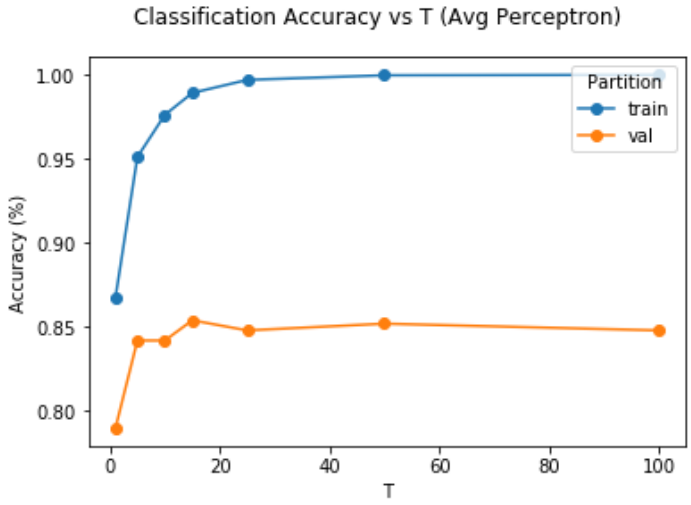
Validation accuracy for average perceptron: 0.8420

Pegasos:

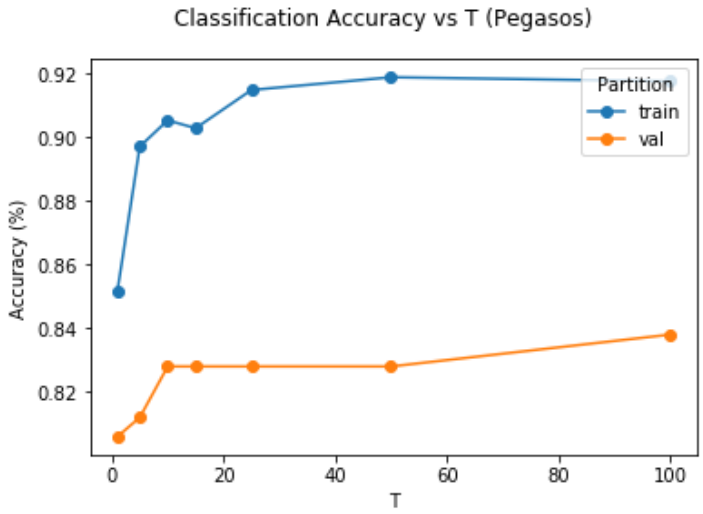
Training accuracy for Pegasos: 0.9050
Validation accuracy for Pegasos: 0.8140

Question 10: Tuning

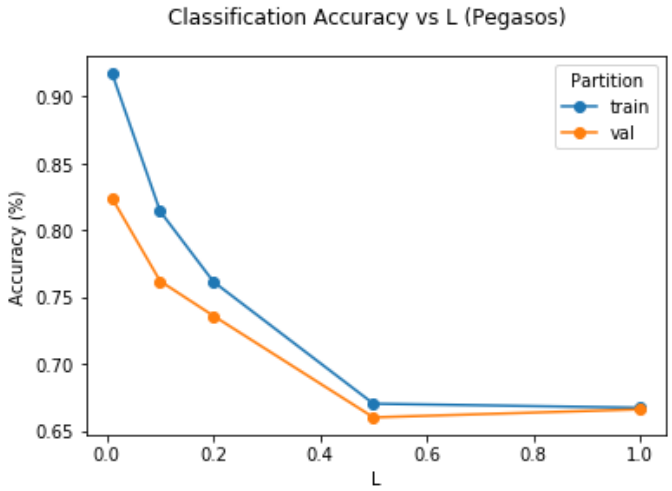




To tune T of Pegasos, lambda was first fixed at λ=0.01



Best T was obtained for Pegasos at T=100 and fixed to obtain Best λ = 0.01 below.



(a) **Do the training and validation accuracies behave similarly as a function of λ and T? Why or why not?**
The reason is that the validation data is somewhat similar to the training data in spread(perhaps because the training data set is a moderately good representation of the overall data) and so if we find a better classifier/theta on the training set (through our selection of T or λ), it also better on the validation set. And the reverse may hold, that is if we chose parameters that lead to worse classifer (higher inaccuracy) then the validation accuracy also reduces.

(b) **Which algorithm performed the best of the three?**
The Average Perceptron performed best of the three.

(c) **What are the optimal values of T for your perceptron? For average perceptron? What are the optimal values of T and λ for your Pegasos algorithm?**
For the Perceptron, optimal T = 100 from plot.
For the Average Perceptron, Optimal T is at 15.

For the Pegasos Algorithm, optimal T = 100. Optimal $\lambda = 0.01$

Question: 11a and b

(a) Accuracy

Training set accuracy for average perceptron: 0.9890
Testing accuracy for average perceptron: 0.7920

(b) Word Features

Most Explanatory Word Features
['perfect', 'love', '!', 'best', 'great', 'delicious', 'good', 'more', 'chips', 'works']

12: Improvements

The following improvements were considered for the features:

1) Bag of Words Modification: Removal of Stopwords.
2) Bag of Words Modification: Consideration of Bigrams in addition to Unigrams.
3) Bag of Words Modifcation: Removal of words with length of 1 from dictionary.
4) Combinaton of the above viz
    a. Unigrams + Bigrams+ Removal of Stopwords
    b. Unigrams + Bigrams + Removal of words with length of 1 from dictionary
    c. Unigrams + Bigrams + Removal of Stopwords + Removal of words with length of 1 from dictionary.

Motivation behind Improvements.

Stopwords: The motivation behind the removal of stopwords is the description provided in the project instruction, i.e. removal of words which may not truly represent reviews but just serve as sentence fillers.

Words with Length of 1: The motivation behind removing these is the fact that in the course of running the codes to determine how to improve features, words such as “s” and “-“ came up as most explanatory words. These words, alongside other one-length words were deemed not to be useful representations of the reviews of customers so the bag of words was modified to reflect this.

Bigrams: The motivation behind bigrams is that provided in the project text instruction.

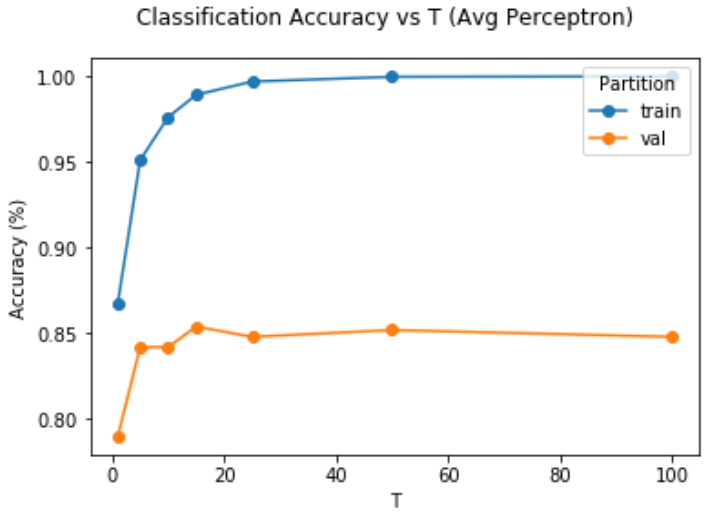
Experiment to determine best features.

The seven possible combinations of the above improvements were tested in comparison to the default bag_of_words function in order to see improvement.

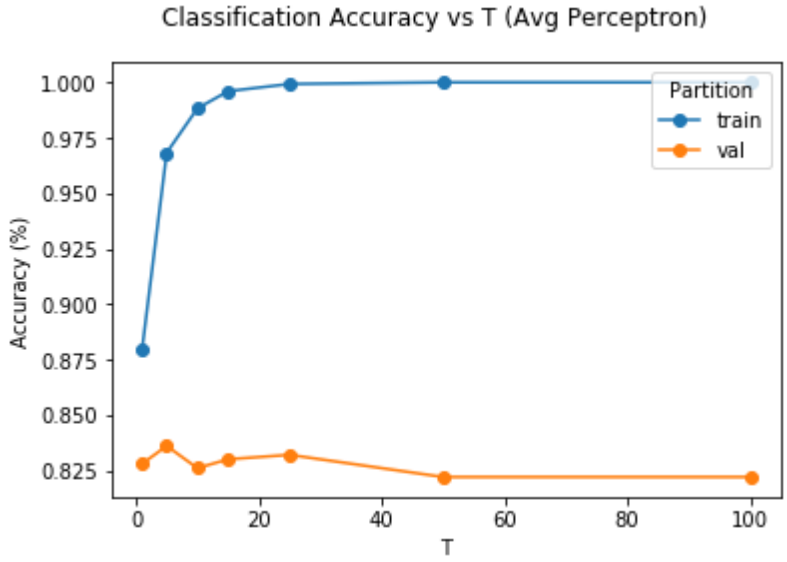
A plot tuning approach, similar to that done for question 10 was taking to plot the accuracy of the average perceptron algorithm on the training vs validation data, using different dictionaries that result from different bag of words. The dictionary variable in Section 3.13 of the main.py was modified repeatedly to consider different bagofwords functions and the results plotted.

The following graphs are the results of the plots of the experiment.

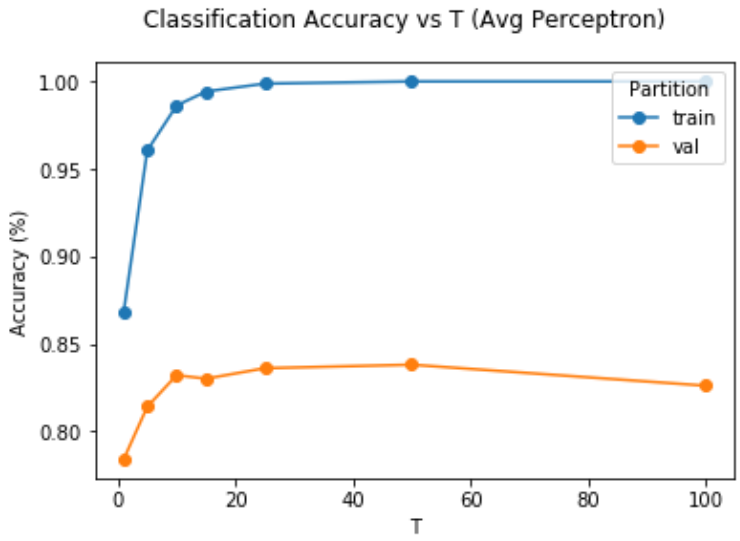
1. Bag Of Words Only



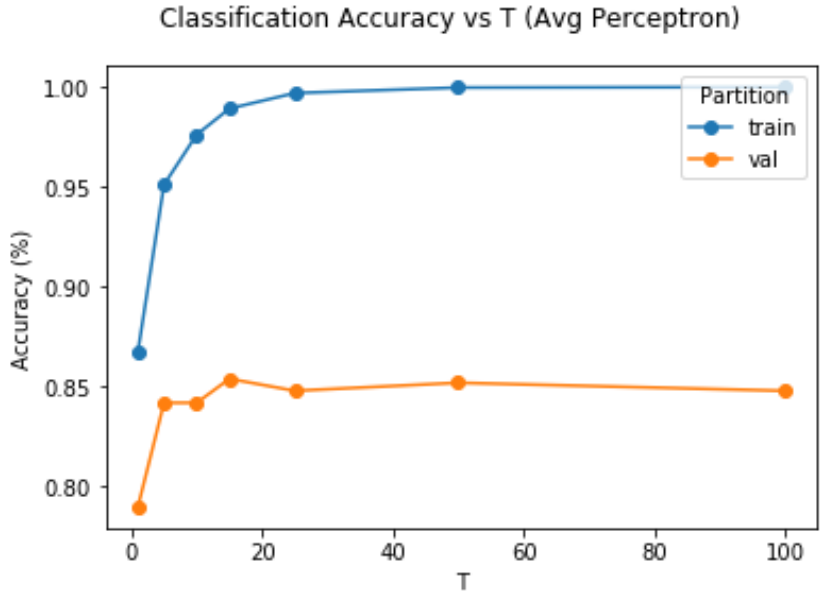
2. Bag of Words + StopWords Removal + Length Constraint



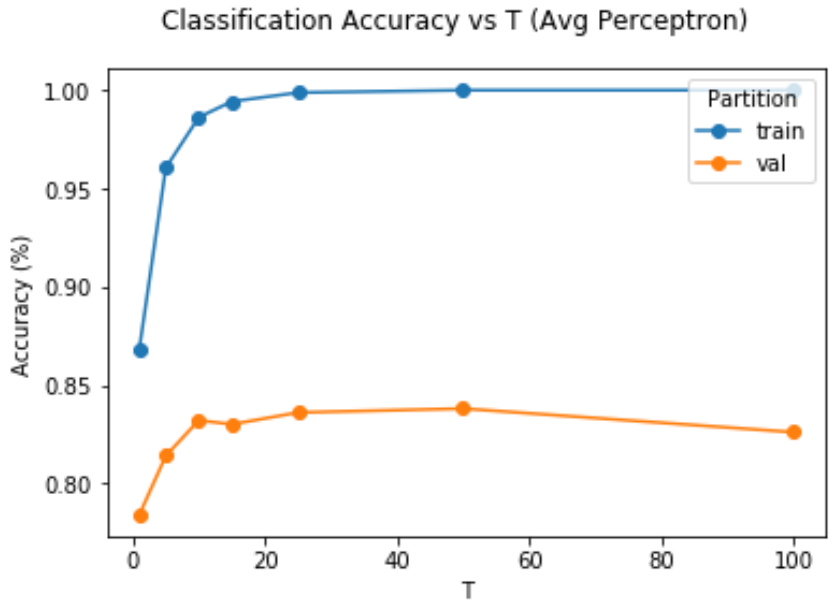
3. Bag of Words + StopWords Removed



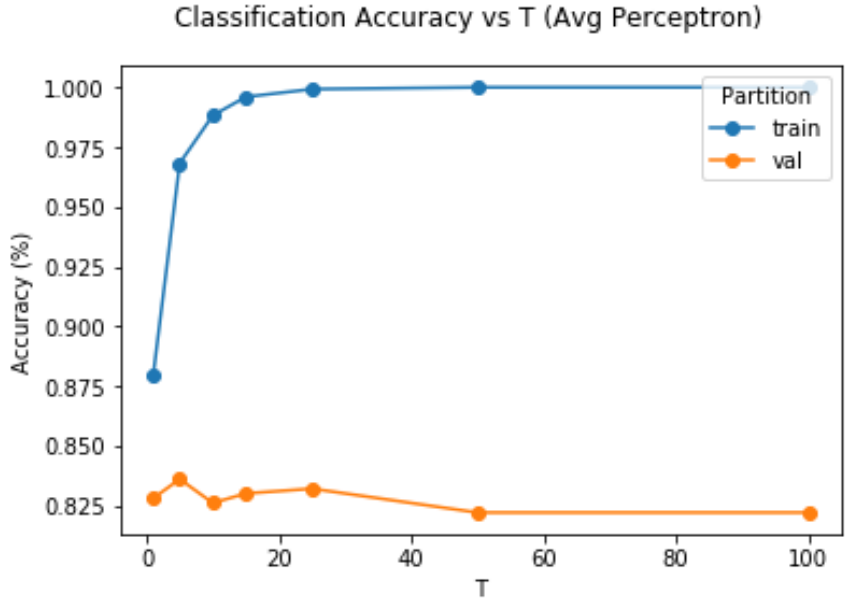
4. Bag Of Words with + Bigrams



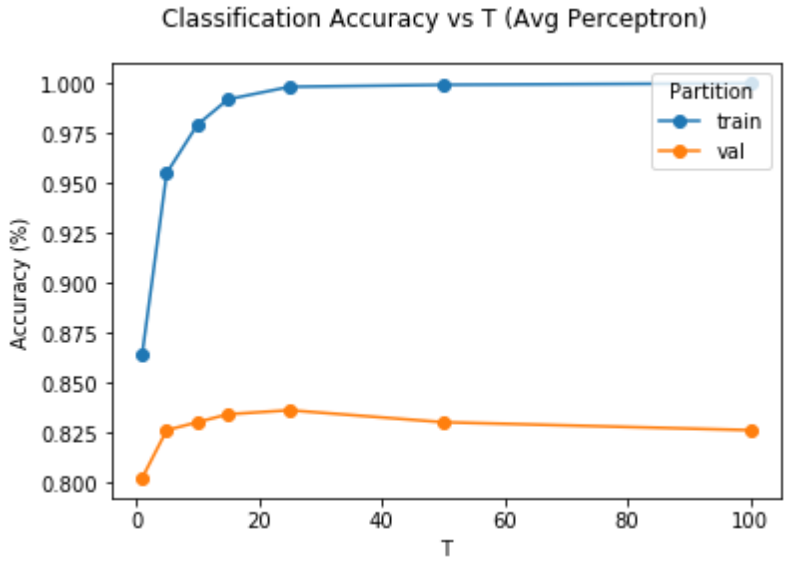
5. Bag of Words + Stopwords Removal + Bigrams



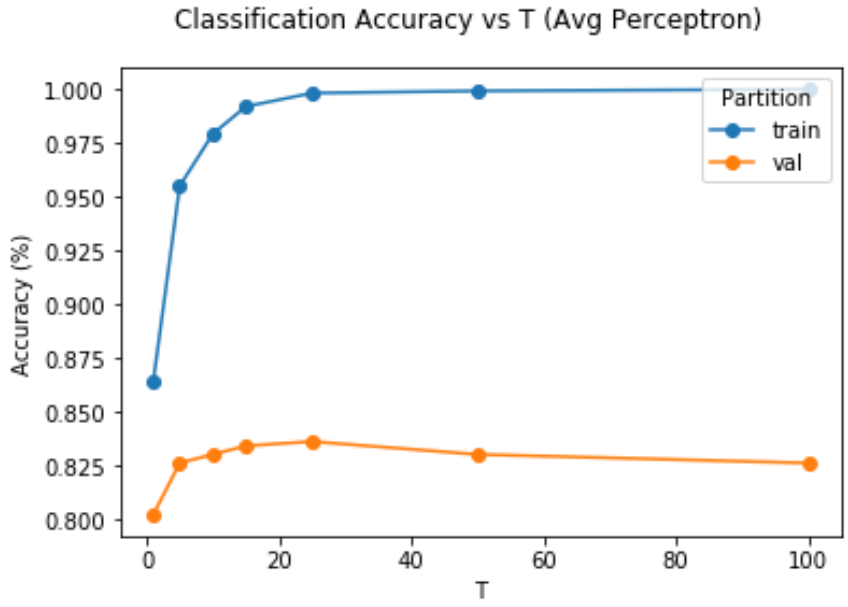
6.Bag of Words + Stopwords Removal + Bigrams + Length Constraint



7. Bag of Words + Bigrams + Length Constraint



8. Bag of Words (Unigram only) + Length Constraint



Accuracy values at T=15 for the above are presented below:
Accuracy values at T=15

1. Bag of Words only. Unmodified (Unigram only)

Training accuracy for average perceptron for final features: 0.9890
Validation accuracy for average perceptron for final features: 0.8540

2. Bag of words –Stopwords removed and Unigram Only.

Training accuracy for average perceptron for final features: 0.9942
Validation accuracy for average perceptron for final features: 0.8300

3. BagOfWords – Stopwords Removed, Length of Words>=1 and Unigram only
Training accuracy for average perceptron for final features: 0.9960
Validation accuracy for average perceptron for final features: 0.8300

4. BagOfWords – StopWords Removed, Unigrams + Bigram
Training accuracy for average perceptron for final features: 0.9942
Validation accuracy for average perceptron for final features: 0.8300

5. BagofWords – Stopwords removed, Unigrams + Bigram, Length of Words>=2
Training accuracy for average perceptron for final features: 0.9960
Validation accuracy for average perceptron for final features: 0.8300

6. BagOfWords – Unigrams+Bigrams, Length of Words
Training accuracy for average perceptron for final features: 0.9920
Validation accuracy for average perceptron for final features: 0.8340

7. Bag of Words – Unigram + Length of words.
Training accuracy for average perceptron for final features: 0.9920
Validation accuracy for average perceptron for final features: 0.8340

8. Bag of Words – Unigram + Bigrams only
Training accuracy for average perceptron for final features: 0.9890
Validation accuracy for average perceptron for final features: 0.8540

Observations

1. Introduction of the bigrams increased the running time of the algorithm without increasing the accuracy as seen in the graph. It only maintained accuracy as similar to unigram case only.
2. None of the improvement approaches led to an increase in accuracy. The implemented functions can be found in the project1.py file for perusal and confirmation.