HW3

3 Classifying restaurant reviews

> **Problem 3.1 (1 point).** Give a *rough estimate* of the amount of memory (in bytes) that would be used by the training data if all examples were represented using the bag-of-words representation generated by the code shown above. How does it compare to the total memory used by `examples`? Explain how you arrived at your estimates.

Answer: Training data has 1000000 samples, and the total vocabulary size is 207429, therefore each sample should be represented by a 207429-dimension integer list. The amount of memory consumption are different on different framework and programming languages. In python, long integer takes up to 28 bytes, therefore the total consumption is $207429000000 \times 28 = 5808012000000$ bytes; in C++, integer takes 4 bytes, therefore the total consumption is $207429000000 \times 4 = 829716000000$ bytes.

This memory consumption is larger than *examples*, because in *examples* we only record the word that actually appears in the sentence. Actually, the average length of sentence in terms of word is only 115, which is much smaller than vocabulary size.

> **Problem 3.2 (2 points).** Implement the Online Perceptron algorithm and apply it to the training data (in the order that the examples are already given in `reviews_tr.csv`). What is the training error rate of the linear classifier returned by Online Perceptron? And what is the test error rate?

Answer: The training error rate of the linear classifier returned by Online Perceptron is $1 - 0.866841 = 0.133159$; the testing error rate is $1 - 0.86417 = 0.13583$.

> **Problem 3.3 (1 point).** To get a sense of the behavior of the linear classifier that you obtained above, determine the 10 words that have the highest (i.e., most positive) weights in the weight vector, and also determine the 10 words that have the lowest (i.e., most negative) weights in the weight vector. Report the actual words, not the word IDs. You may find it helpful to invert the mapping given in `vocab`.

Answer: The 10 words that have the highest (i.e., most positive) weights in the weight vector are: `'perfection', 'gem', 'incredible', 'heaven', 'superb', 'phenomenal', 'amazing', 'worried', 'perfect', 'heavenly'`

The 10 words that have the smallest (i.e., most negative) weights in the weight vector are: `'mediocre', 'worst', 'meh', 'disappointing', 'lacked', 'underwhelmed', 'flavorless', 'bland', 'poisoning', 'disgusting'`

3.2 Averaged Perceptron

The *Averaged Perceptron* algorithm is a variant of Online Perceptron. It is exactly the same as Online Perceptron, except that the weight vector returned is $\vec{w}_{\text{avg}} := \frac{1}{n}\sum_{i=1}^{n} \vec{w}_i$ (instead of just $\vec{w}_n$). This is more "stable" than Online Perceptron, since the average of the weight vectors is much less sensitive to a single training example than the current weight vector maintained by Online Perceptron.

> **Problem 3.4 (1 point).** Suppose instead of returning $\vec{w}_{\text{avg}}$, your implementation of Averaged Perceptron returns $\vec{w}_{\text{sum}} := \sum_{i=1}^{n} \vec{w}_i$. Why would this be okay?

Answer: This is because a linear classifier with parameter $\vec{w}$ predicts "true" on an input feature vector $\vec{x}$if and only if $\vec{w} \cdot \vec{x} > 0$. Since multiplying $\overrightarrow{w_{avg}}$ by n doesn't effect the signal of $\vec{w} \cdot \vec{x}$, it is okay to use sum instead of average.

> **Problem 3.5 (2 points).** Implement the Averaged Perceptron algorithm and apply it to the training data (in the order that the examples are already given in `reviews_tr.csv`). What are the training error rate and the test error rate of the linear classifier returned by Averaged Perceptron?

Answer: The training error rate of the linear classifier returned by Averaged Perceptron is $1 - 0.8954 = 0.1046$; the testing error is $1 - 0.8931 = 0.1069$.

> **Problem 3.6 (1 point).** Repeat Problem 3.3 for the linear classifier obtained in Problem 3.5.

Answer: The 10 words that have the highest (i.e., most positive) weights in the weight vector are: `'perfection', 'perfect', 'incredible', 'perfectly', 'gem', 'fantastic', 'delicious', 'amazing', 'excellent', 'disappoint'`

The 10 words that have the smallest (i.e., most negative) weights in the weight vector are: `'worst', 'mediocre', 'bland', 'meh', 'disappointing', 'awful', 'horrible', 'terrible', 'lacked', 'flavorless'`

> **Problem 3.7 (1 point).** Implement one of the above suggested improvements, or any other improvement you can think of. (If you come up with your own improvement, give a high-level description of it and explain its motivation. If you one of the suggested improvements from above, please describe it in your own words at a high-level.) Apply the new algorithm to the training data. What is the test error rate of the classifier you obtain?

Answer:

Description: instead of just iterate through the data for one time, we can iterate through the data for multiple times (i.e. train for multiple epochs). The test error rate of

the classifier is $1 - 0.8949 = 0.1051$.