CSE 5525 Text Classification (part 2)

Alan Ritter

In this assignment you will implement the perceptron algorithm (with parameter averaging) using the same sentiment analysis dataset you experimented with in assignment 1. This assignment will ask you to inspect the model's parameters and compare the performance of the two algorithms.

As before, the experiments required to complete the assignment may take some time to run - so the efficiency of your code will be an important consideration, and it is a good idea to start early.

Gradient of Logistic Regression (written question - 2 points)

Derive the gradient of the conditional log likelihood objective for multi-class logistic regression:

$$\frac{\partial}{\partial w_i} \sum_j \log P(y_j|x_j) \tag{1}$$

Perceptron (3 points)

Implement the perceptron classification algorithm (analagous starter code is provided in `Perceptron.py`). The only hyperparameter is the number of iterations. Run the classifier and report results on the test set with various numbers of iterations (for exaple: 1, 10, 50, 100, 1000).

Parameter Averaging (2 points)

Modify the perceptron code to implement parameter averaging. Instead of using parameters from the final iteration, w_n, to classify test examples, use the average of the parameters from every iteration, $\sum_{i=1}^{N} w_i$. A nice trick for doing this efficiently is described in section 2.1.1 of Hal Daume's thesis (`http://www.umiacs.umd.edu/~hal/docs/daume06thesis.pdf`).

Compare the performance of your implementation with and without parameter averaging. Also compare to the results of your Naive Bayes classifier.

Features (2 points)

Print out the 20 most positive and 20 most negative words in the volcabulary sorted by their weight according to your model This will require a bit of thought how to do because the words in each document have been converted to IDs (see `Vocab.py`). The output should look something like so:

```
word1_pos weight1
word2_pos weight2
word3_pos weight3
...

word1_neg weightk
word2_neg weightk+1
word3_neg weightk+2
...
```

Where `wordn_pos` and `wordn_neg` are the top 20 positive and negative words. (Hint: you might find the `numpy.argsort` method useful - `http://docs.scipy.org/doc/numpy/reference/generated/numpy.argsort.html`). Please include this output in your report.

What to Turn In

Please turn in the following to the dropbox on Carmen:

1. Your code.
2. Your answer to the written question.
3. A brief writeup that includes the numbers / evaluation requested above (text file format is fine).