Maximum Entropy Model and Perceptron Implementation

Carson B. Hanel

10/5/2017

Natural Language Processing

- How to compile and run the code:
    - Perceptron:
        - Set your working directory to the file folder
        - Open a shell session from Python
        - Input:
            - Python Perceptron.py ../data/imdb1/ 10
    - MaxEnt:
        - Set your working directory to the file folder
        - Open a shell session from Python
        - Input:
            - python Maxent.py ../data/imdb1/ 0.01 .001 0.1
- Results and analysis:
    - Perceptron:
        - The maximum probability I've gotten was 81% accuracy with perceptron for a singular case fold, with an average of around 71% being the highest average across 10 folds.
        - The features are bag of words, and I exclude stopwords when scoring. The reason I do this is so that I'm only filtering the final 200 documents to score and then applying the scores the those values that hold sentiment. I realize this is a shortcoming and a bit of a workaround. My plan was initially to filter all documents of their stopwords before doing the test/train splits so that the only words that occur are ones that hold sentiment. I didn't have the time to implement this, but my hypothesis is that this would greatly increase the scoring. This is due to the fact that in both Perceptron and SGD, all weights are co-dependent, so when I'm training I still have the co-dependencies of the stopwords included in the weights overall even though they're not scored at the end.
        - Hyperparameters:
            - As I increased eta, the dispersion of weights increased.
            - As I increased lambda, the dispersion of weights decreased.
            - As I increased epsilon, the rigor of the weights increased to a certain degree, and the program took much longer to evaluate.
                - NOTE!: I did not add checking for convergence because all test cases converged to the epsilon value that I have put into the running instructions. The program may not converge if the test cases are wildly inaccurate.
    - MaxEnt:

            - The maximum probability I've gotten was 84.5% accuracy, with 76.4% accuracy being the highest average across 10 folds.
            - Essentially similar implementation of Perceptron.
            - For both, I made sure I didn't parse every word of a document, but simply all words in the set of words (sets confirm words are non repeating), so the weights are based on whether or not that word occurs in a positive/negative document rather than how many times it occurs.
                - The reason for this is, because I'm not excluding stopwords in calculation. To make it more accurate, I'd exclude stopwords before the test train split, and only train solely on those words that contain sentiment. At that point, I would reinstate word frequency analysis back into the code to get a much higher rate of classification.
- Bugs, problems, limitations:
    - I believe most have been discussed in the analysis portion.