Natural Language Processing
Week 2: Text Classification

Andreas Vlachos

The goal of this lab session (5% of the course) is to learn how to implement one of the oldest but also most successful classification learning algorithms in NLP, the perceptron. Here is a brief reminder from the lecture slides:

Task: learn weights w^y for each label $y \in \mathcal{Y}$ so that $argmax_y(w^y \cdot x)$ returns the correct answer

Input: training examples $\mathcal{D} = (x^1, y^1) \dots (x^M, y^M)$
Initialize weights $w_0^{(y \in \mathcal{Y})} = (0, ..., 0)$
for $(x, y) \in \mathcal{D}$ **do**
  Predict label $\hat{y} = argmax_y(w_t^y \cdot \phi(x))$
  if $\hat{y} \neq y$ **then**
    Update $w_{t+1}^y = w_t + \phi(x)$
    Update $w_{t+1}^{\hat{y}} = w_t - \phi(x)$
  else
    $w_{t+1}^{(k)} = w_t^{(k)}$
  end if
end for

- Implement the algorithm with two functions: `test` and `train`.
- Download the movie review dataset from this page: `http://www.cs.cornell.edu/people/pabo/movie-review-data/review_polarity.tar.gz` and take the first (alphabetic file order) 800 instances of each class as training data and the remaining 200 as testing. Convert each review into a bag-of-words representation and evaluate the performance. What is your accuracy on the test data?
- Implement the following improvements:

  - Multiple passes over the training instances (show the learning progress in a graph)
  - Randomizing the order of the training instances (fix the random seed so that your results are reproducible!)
  - Instead of using the last weight vector in the `test` function, taking the average of all the weight vectors calculated for each class

  Does any of them help improve accuracy?

- What are the most positively-weighted features for each class? Give the top 10 for each class and comment on whether they make sense. (If they don't you might have a bug.) If we were to apply the classifier we learnt to a different domain such laptop reviews or restaurant reviews, do you think these features would generalize well? Can you propose better features for the task?

You should submit a python file (lab2.py) which executes from the directory with the neg/pos subdirectories trains and tests your best performing model and has comments explaining it. And you should accompany it with a lab2.pdf (no more than two A4 pages) answering the questions about it. Make sure your code is Python3 compatible. There is no correct answer on what your accuracy should be, but correct implementations usually achieve around 80%. The quality of the analysis in your report is more important than the accuracy itself.