Programming Assignment

Submission guidelines:

- Download the supplied files from Moodle (2 python files and 1 `tar.gz` file). Details on every file will be given in the exercises. You need to update the code only in the skeleton files, i.e., the files that have a prefix "skeleton". Written solutions, plots and any other non-code parts should be included in the written solution submission.
- Your code should be written in Python 3.
- Make sure to comment out or remove any code which halts code execution, such as matplotlib popup windows.
- Your code submission should include these files: `adaboost.py`,`process_data.py`

1. **(30 points) AdaBoost.** In this exercise, we will implement AdaBoost and see how boosting can be applied to real-world problems. We will focus on binary sentiment analysis, the task of classifying the polarity of a given text into two classes - positive or negative. We will use movie reviews from IMDB as our data.

   Download the provided files from Moodle and put them in the same directory:

   - `review_polarity.tar.gz` - a sentiment analysis dataset of movie reviews from IMBD.[1] Extract its content in the same directory (with any of zip, 7z, winrar, etc.), so you will have a folder called `review_polarity`.
   - `process_data.py` - code for loading and preprocessing the data.
   - `skeleton_adaboost.py` - this is the file you will work on, change its name to `adaboost.py` before submitting.

   The main function in `adaboost.py` calls the `parse_data` method, that processes the data and represents every review as a 5000 vector x. The values of x are counts of the most common words in the dataset (excluding stopwords like "a" and "and"), in the review that x represents. Concretely, let $w_1, \dots, w_{5000}$ be the most common words in the data, given a review r_i we represent it as a vector $x_i \in N^{5000}$ where $x_{i,j}$ is the number of times the word w_j appears in r_i. The method `parse_data` returns a training data, test data and a vocabulary. The vocabulary is a dictionary that maps each index in the data to the word it represents (i.e. it maps $j \to w_j$).

   (a) **(10 points)** Implement the AdaBoost algorithm in the `run_adaboost` function. The class of weak learners we will use is the class of hypothesis of the form:

   $$h(x_i) = \begin{cases} 1 & x_{i,j} \leq \theta \\ -1 & x_{i,j} > \theta \end{cases}, \quad h(x_i) = \begin{cases} 1 & x_{i,j} \geq \theta \\ -1 & x_{i,j} < \theta \end{cases},$$

   That is, comparing a single word count to a threshold. At each iteration, AdaBoost will select the best weak learner. Note that the labels are $\{-1, 1\}$. Run AdaBoost for $T = 80$ iterations. Show plots for the training error and the test error of the classifier implied at each iteration t, $sign(\sum_{j=1}^{t} \alpha_j h_j(x))$.

[1] `http://www.cs.cornell.edu/people/pabo/movie-review-data/`

(b) **(10 points)** Run AdaBoost for $T = 10$ iterations. Which weak classifiers the algorithm chose? Pick 3 that you would expect to help to classify reviews and 3 that you did not expect to help, and explain possible reasons for the algorithm to choose them.

(c) **(10 points)** In next recitation you will see that AdaBoost minimizes the average exponential loss:

$$\ell = \frac{1}{m} \sum_{i=1}^{m} e^{-y_i \sum_{j=1}^{T} \alpha_j h_j(x_i)}.$$

Run AdaBoost for $T = 80$ iterations. Show plots of ℓ as a function of T, for the training and the test sets. Explain the behavior of the loss.