CS 434 — Implementation Assignment 2 — Due Apr 25th 11:59PM, 2020

General instructions.

1. The assignment should be implemented in Python3. You should make sure that your code runs on our class server (vm-cs434-1). For this you can do the following steps:

- a) ssh username@flip.engr.oregonstate.edu. (ssh to Flip)
- b) ssh vm-cs434-1 (ssh to class server)
- c) test code there

2. You can work in team of up to 3 people. Submit the source code and report on Canvas.
3. You need to submit your source code (self contained, well documented and with clear instruction for how to run) and a report on canvas. In your submission, please clearly indicate your team members' information and also how your team divided the work.
4. Be sure to answer all the questions in your report. Your report should be typed, submitted in the pdf format. You will be graded based on both your code as well as the report. In particular, the clarity and quality of the report will be worth 10 points. So please write your report in clear and concise manner. Clearly label your figures, legends, and tables.
5. In your report, the results should always be accompanied by discussions of the results. Do the results follow your expectation? Any surprises? What kind of explanation can you provide?

Naive Bayes (Multinomial) for sentiment analysis

In this assignment you will implement the Naive Bayes (Multinomial) classifier for Sentiment Analysis of an IMDB movie review dataset (a highly polarized dataset with 50000 movie reviews). The primary task is to classify the reviews into negative and positive.

More about the dataset: http://ai.stanford.edu/ãmaas/data/sentiment/

For the Multinomial model, each document is represented by a vector of integer-valued variables, i.e., $\mathbf{x} = (x_1, x_2, ..., x_{|V|})^T$ and each variable x_i corresponds to the i-th word in a vocabulary V and represents the number of times it appears in the document. The probability of observing a document $\mathbf{x}$ given its class label y is defined as (for example, for $y = 1$):

$$p(\mathbf{x}|y=1) = \prod_{i=1}^{|V|} P(w_i|y=1)^{x_i}$$

Here we assume that given the class label y, each word in the document follows a multinomial distribution of $|V|$ outcomes and $P(w_i|y = 1)$ is the probability that a randomly selected word is word i for a document of the positive class. Note that $\sum_{i=1} P(w_i|y) = 1$ for $y = 0$ and $y = 1$. Your implementation need to estimate $p(y)$, and $P(w_i|y)$ for $i = 1, \cdots, |V|$, and $y = 1, 0$ for the model. For $p(y)$, you can use the MLE estimation. For $P(w_i|y)$, you MUST use Laplace smoothing for the model. One useful thing to note is that when calculating the probability of observing a document given its class label, i.e., $p(\mathbf{x}|y)$, it can and will become overly small because it is the product of many probabilities. As a result, you will run into underflow issues. To avoid this problem, your implementation should operate with log of the probabilities.

1 Description of the dataset

The data set provided are in two parts:

- IMDB.csv: This contains a single column called Reviews where each row contains a movies review. There are total of 50K rows. The first 30K rows should be used as your Training set (to train your model). The next 10K should be used as the validation set (use this for parameter tuning). And the last 10K rows should be used as the test set (predict the labels).
- IMDB_labels.csv: This contains 40K labels. Please use the first 30K labels for the training data and the last 10K labels for validation data. The labels for test data is not provided, we will use that to evaluate your predicted labels.

2 Data cleaning and generating BOW representation

Data Cleaning. Pre-processing is need to makes the texts cleaner and easy to process. The reviews columns are comments provided by users about the movie. These are known as "dirty text" that required further cleaning. Typical cleaning steps include a) Removing html tags; b) Removing special characters; c) Converting text to lower case d) replacing punctuation characters with spaces; and d) removing stopwords i .e . articles, pronouns from consideration. You will not need to implement these functionalities and we will provide some starter code containing these functions for you to use.

Generating BOW representation. To transform from variable length reviews to fixed-length vectors, we use the Bag Of Words technique. It uses a list of words called "vocabulary", so that given an input text we can output a vector of word counts for each word in the vocabulary. You can use the CountVectorizer functionality from sklearnstarter to go over the full 50K reviews to generate the vocabulary and create the feature vectors representing each review. Not that the CountVectorizer function has several tunable parameters that can directly impact the result feature representation. This includes $max\_features$:, which specifies the maximum number of features (by considering terms with high frequency); *max_df* and *min_df*, which filter the words from the dictionary if its document frequency is too high ($>$ *max_df*) or too low ($<$ *min_df*) respectively.

3 What you need to do

1. (10 pts) Apply the above described data cleaning and feature generation steps to generate the BOW representation for all 50k reviews. For this step, we will use the default value for *max_df* and *min_df* and set $max\_features = 2000$.
2. (20 pts) Train a multi-nomial Naive Bayes classifier with Laplace smooth with $\alpha = 1$ on the training set. This involves learning $P(y = 1), P(y = 0)$, $P(w_i|y = 1)$ for $i = 1, ..., |V|$ and $P(w_i|y = 0)$ for $i = 1, ..., |V|$ from the training data (the first 30k reviews and their associated labels).
3. (20 pts) Apply the learned Naive Bayes model to the validation set (the next 10k reviews) and report the validation accuracy of the your model. Apply the same model to the testing data and output the predictions in a file, which should contain a single column of 10k labels (0 (negative) or 1 (positive)). Please name the file test-prediction1.csv.
4. (20 pts) Tuning smoothing parameter *alpha*. Train the Naive Bayes classifier with different values of α between 0 to 2 (incrementing by 0.2). For each *alpah* value, apply the resulting model to the validation data to make predictions and measure the prediction accuracy. Report the results by creating a plot with value of α on the x-axis and the validation accuracy on the y-axis. Comment on how the validation accuracy change as α changes and provide a short explanation for your observation. Identify the best α value based on your experiments and apply the corresponding model to the test data and output the predictions in a file, named test-prediction2.csv.
5. (20 pts) **Tune your heart out.** For the last part, you are required to tune the parameters for the CountVectorizer ($max\_feature$, *max_df*, and *min_df*). You can freely choose the range of values

you wish[1] to test for these parameters and use the validation set to select the best model. Please describe your strategy for choosing the value ranges and report the best parameters (as measured by the prediction accuracy on the validation set) and the resulting model's validation accuracy. You are also required to apply the chosen best model to make predictions for the testing data, and output the predictions in a file, named test-prediction3.csv.

[1]You are encouraged to try your best to tune these parameters. Higher validation accuracy and testing accuracy will be rewarded with possible bonus points.