2 Programming Assignment (80%)

2.1 Naive Bayes (40%)

Much of machine learning with text involves counting events. Naive Bayes fits nicely into this framework. The classifier needs just a few counters. For this assignment we will be performing document classification using Naive Bayes.

First, let y be the labels for the training documents and $\mathcal{W} = \{w_i\}_{i=1}^{L}$ be the sets of words in the document. Here are the counters we need to get for predication:

- $\#(Y = y)$ for each label y the number of training instances of that class
- $\#(Y = *)$ here $*$ means anything, so this is just the total number of training instances
- $\#(Y = y, W = w)$ number of times token w appears in the documents with label y
- $\#(Y = y, W = *)$ total number of tokens for documents with label y.

One way to compute these counters is to convert all the documents into feature vectors, $\mathbf{w} \in \mathbf{Z}^{|\mathcal{V}|}$, i.e. $|\mathcal{V}|$ dimension integer vectors, where $\mathcal{V}$ is the dictionary, which provides indices for all words. Then w_i is the number of appearances of word with index i in this document. Store all of these feature vectors together as a matrix, which is saved in the format of .csv file in your case (see Section 2.3 for details).

Without smoothing, the final prediction is

$$y^* = arg\max_y (\prod_{i=1}^{L} \frac{\#(Y = y, W = w_i)}{\#(Y = y, W = *)}) \frac{\#(Y = y)}{\#(Y = *)} \tag{1}$$

Also, to avoid the numerical problem, you may want to use logarithm of probability

$$y^* = arg\max_y (\sum_{i=1}^{L} \log \frac{\#(Y = y, W = w_i)}{\#(Y = y, W = *)}) + \log \frac{\#(Y = y)}{\#(Y = *)} \tag{2}$$

Note that your implementation will use smoothed probabilities. For example, at classification time, you can use Laplace smoothing with $\alpha = 1$ as described here: https://en.wikipedia.org/wiki/Additive_smoothing.

2.2 Voted Perceptron (40%)

Perceptron algorithm is one of the classic algorithms which has been used in machine learning from early 1960s. It is an online learning algorithm for learning a linear threshold function which works as a classifier. We will also apply the Voted Perceptron algorithm on the same document classification task in this assignment.

Let $\mathcal{T} = \{(y_i, \mathbf{x}_i)\}_{i=1}^{m}$ be the training data, y_i be the labels ($y_i \in \{-1, 1\}$ in binary classification problem) for the ith document and the corresponding feature vector is $\mathbf{x}_i$. The goal of the Perceptron algorithm is to find a vector $\mathbf{w}$ which defines a linear function such that $\forall i, y_i \langle \mathbf{w}, \mathbf{x}_i \rangle > 0$ ($\langle \cdot, \cdot \rangle$ is the inner product operation). For the Voted Perceptron, you need a list of weighted vectors $\{(\mathbf{w_k}, \mathbf{c_k})\}_{k=1}^{K}$, where $\mathbf{w_k}$ is the vector and c_k is its weight. (Please refer https://cseweb.ucsd.edu/~yfreund/papers/LargeMarginsUsingPerceptron.pdf for a clear descrip-

tion of Voted Perceptron)

Initiate $k = 1, c_1 = 0, \mathbf{w}_1 = \mathbf{0}, t = 0$;
while $t \leq T$ *(number of rounds)* **do**
  for *each training example* $(y_i, \mathbf{x}_i)$ **do**
    if $y_i \langle \mathbf{w}_k, \mathbf{x}_i \rangle \leq 0$ **then**
      $\mathbf{w}_{k+1} = \mathbf{w}_k + y_i \mathbf{x}_i$;
      $c_{k+1} = 1$;
      $k = k + 1$;
    else
      $c_k = c_k + 1$;
    end
  end
  $t = t + 1$;
end

Then you can use all these vectors and their weight to do the classification: the predicted label $\hat{y}$ for any unlabeled document with feature vector $\mathbf{x}$ would be

$$\hat{y} = sign(\sum_{k=1}^{K} c_k sign(\langle \mathbf{x}, \mathbf{w}_k \rangle)) \tag{3}$$

2.3 Data Set

For this assignment, you should download the training data from https://www.dropbox.com/s/5nvdcezf85vpetj/hw2_data.zip?dl=0. It contains two csv files: 'Xtrain.csv' and 'Ytrain.csv'.

Xtrain.csv Each row is a feature vector of one document. The values in the ith columns are number of occurrences of the word with Id i in the corresponding documents. For example, assume the first row in 'Xtrain.csv' is:

0, 1, 2, 10, 0, 2, ...

which means the first word in the dictionary does not appear in this document, the second word in the dictionary appears once in this document, the third word in the dictionary appears twice in the document, and so on.

Ytrain.csv The csv file 'Ytrain.csv' provides the binary labels for corresponding documents in the file 'Xtrain.csv'. The corresponding label for the document in the above example has been saved at the first row of the file 'Ytrain.csv'.

There are two different kind of news documents, your task is to detect onion documents (fake news) from them:

- **onion** (fake news, labeled with 1), containing articles from the not-so-serious American magazine TheOnion.com
- **economist** (normal news, labeled with 0), containing articles from the serious European magazine Economist.com

2.4 Implementation & Submission

For this assignment, we publish a CodaLab competition (https://competitions.codalab.org/competitions/20280?secret_key=a9ed97f1-66b4-4d71-8151-0ec85ccd7856), where you can submit your code and get evaluation scores that will be listed on a leader-board. To submit during four late days, please go to https://competitions.codalab.org/competitions/20279?secret_key=d99d8c12-2269-488d-a592-d48ab91991d1. If you are not familiar with CodaLab Competitions, check out here: participating in a competition.

Remember to register a CodaLab Competitions account using your umail account, so that the username will be your UCSBNetID. After that, log into CodaLab Compectitions and set up your team name (whatever nickname you like). To protect your privacy, only the team names will be shown on the leader-board and your usernames will be anonymous. After your submission finishes running, please choose to submit it to the leader-board. Note that here team

name is equivalent to your nickname, and it is still an independent homework assignment. You must implement the two algorithms according to the instructions above. You must avoid calling APIs from existing machine learning toolkits such as scikit-learn. Using numpy is allowed.

There are two phases, *Naive Bayers* and *Voted Perceptron*, representing the two parts of the assignment. You must submit one zip file per phase via the corresponding submission page. Your code must be written in Python.

Submission format The final submission format should be:
naive_bayers.zip

- run.py
- other python scripts you wrote

voted_perceptron.zip

- run.py
- report.pdf (see Sec. 3 for more details)
- other python scripts you wrote

Note that to create a valid submission, zip all the file with 'zip -r zipfilename *' starting from this directory. **DO NOT zip the directory itself, just its contents. (THIS IS VERY IMPORTANT!)**

Here is a sample 'run.py' file: https://www.dropbox.com/s/a2ahv2qh7solhhn/run.py?dl=0. You must submit the 'run.py' file with the exactly same format. For each phase, you have a total of 30 possible submissions.

Computational restrictions For both phases, the prescribed "time budget" is 300 seconds, including both the running time of your code and the evaluation time of the predicted results. In other words, the running time of your program must be less than 300 seconds.

Count		predicted class	
		0	1
true class	0	a	b
	1	c	d

Figure 1: Precision, Recall and Accuracy

2.5 Evaluation Criteria

As illustrated in Section 2.4, the prediction file generated by your code should be in the exactly same format of the file 'Ytrain.csv'. We will then compare your predictions with the ground truth labels and use a weighted combination of accuracy and F-measure to evaluate your results. So your final score for each submission

$$final\_score = 50 \times accuracy + 50 \times F\_measure.$$

For someone who is not familiar with the definitions of precision and F-measure, here is a simple explanation (see Figure. 1, a, b, c, d are the number of documents with different true and predicted labels):

$$precision = \frac{d}{b+d}, \qquad recall = \frac{d}{c+d},$$
$$F\_measure = 2 \times \frac{precision \times recall}{precision + recall}, \quad accuracy = \frac{a+d}{a+b+c+d}.$$

Even though your code is automatically evaluated on CodaLab Competitions, we will download your latest submissions and run a Plagiarism detection program on them. So please be honest.

3 Report (20%)

In addition to the code submission on CodaLab Competitions, you are also supposed to write a report and include it in the submission zip file for the second phase. The report should solve the following question:

First, use the last 10% of the training data as your test data. Compare Naive Bayes and Voted Perceptron on several fractions of your remaining training data. For this purpose, pick $1\%, 2\%, 5\%, 10\%, 20\%$ and 100% of the first 90% training data to train and compare the performance of Naive Bayes and Voted Perceptron on the test data respectively. Plot the accuracy as a function of the size of the fraction you picked (x-axis should be "percent of the remaining training data" and y-axis should be "accuracy").