| **CIS519: Applied Machine Learning** | **Fall 2019** |
|---|---|
| Homework 4 | |
| *Handed Out: November 18, 2019* | *Due: December 2, 2019 at 11:59pm* |

Version 2

- Feel free to talk to other members of the class in doing the homework. I am more concerned that you learn how to solve the problem than that you demonstrate that you solved it entirely on your own. You should, however, **write down your solution yourself**. Please include at the top of your document the list of people you consulted with in the course of working on the homework.
- While we encourage discussion within and outside the class, cheating and copying code is strictly not allowed. Copied code will result in the entire assignment being discarded at the very least.
- Please use Piazza if you have questions about the homework. Also, please come to the TAs recitations and to the office hours.
- Handwritten solutions are not allowed. All solutions must be typeset in Latex. Consult the class' website if you need guidance on using Latex. If you don't have a lot of experience with Latex (or even if you do), we recommend using Overleaf (`https://www.overleaf.com`) to write your solutions. You will submit your solutions as a single pdf file (in addition to the package with your code; see instructions in the body of the assignment).
- The homework is due at 11:59 PM on the due date. We will be using Gradescope for collecting the homework assignments. You should have been automatically added to Gradescope. If not, please ask a TA for assistance. Please do **not** hand in a hard copy of your write-up. Post on Piazza and contact the TAs if you are having technical difficulties in submitting the assignment.
- Here are some resources you will need for this assignment
  - Latex template, Python template, and data can all be found here: `https://www.seas.upenn.edu/~cis519/fall2019/assets/HW/HW4/hw4-materials.zip`

# 1 Document Classification [60 Points]

In this problem, you will develop Naive Bayes-based models to do text classification. The task of text classification is label a piece of text with one class out of a predefined set of classes.

## 1.1 Dataset

The dataset you will use for the experiments will be a sentiment analysis dataset from IMDb. Each document is a review of a movie and has been labeled as either a positive or negative review. We have provided for you training, validation, and testing splits and the Python code to load the documents into memory. Each document has been pre-processed with tokenization[1] and lemmatization.[2]

[1] We say "token" instead of "word" because it is more general and covers symbols like punctuation which we treat as their own tokens separate from the surrounding words. "Tokenization" is splitting raw text into individual tokens.

[2] `https://en.wikipedia.org/wiki/Lemmatisation`

## 1.2 Document Representation

You will experiment with three different ways to represent the documents. "Representation" means how you convert the raw text of a document to a feature vector. Similar to Homework 2, you will use a sparse representation of the feature vectors which is based on a dictionary that maps from the feature name to the feature value. For each representation, there will be exactly 1 feature per unique token in the vocabulary, but the value of that feature will change.

**Vocabulary Creation** The vocabulary is the set of tokens that you will use to compute features. There is some nuance to how to select which tokens from the dataset should be part of the vocabulary. If you use all of the tokens in the training data, there could be a token that appears for the first time in the test set, and it's not clear what to do with that token. A common solution to this problem is to map infrequent words in the training data to a token called `<unk>`, compute the features with the `<unk>` tokens, then map the novel test words to `<unk>`. You will do exactly that.

For each of the experiments below, you should create the vocabulary by taking all of the tokens in the training data which appear more than once plus a special `<unk>` token. You should still estimate probabilities for the `<unk>` token. At test time, any unknown words should be treated as `<unk>`s.

**Representations** We will use $f_d(v)$ to be the value for feature $v$ and document $d$.

The three ways that you should represent each document (and the corresponding Python functions to implement) are described below.

1. **Binary Bag-of-Words** Each document should be represented with binary features, one for each token in the vocabulary.

$$f_d(v) = \begin{cases} 1 & \text{if } v \in d \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

   Since you are using a sparse representation, you do not need to remember which tokens are not present in each document, only those which *are* present.

2. **Count Bag-of-Words** Instead of having a binary feature for each token, you should keep count of how many times the token appears in the document, a quantity known as *term frequency* and denoted $tf(d, v)$.

$$f_d(v) = tf(d, v) \tag{2}$$

3. **TF-IDF Model** The final representation will use the TF-IDF score of each token. The TF-IDF score combines how frequently the word appears in the document as well as how infrequently that word appears in the document collection as a whole. First, the inverse document frequency (IDF) of a token is defined as

$$idf(v) = \log \frac{|\mathcal{D}|}{|\{d : v \in d, d \in \mathcal{D}\}|} \tag{3}$$

Here, $\mathcal{D}$ is the set of documents[3] and the denominator is the number of documents that the token $v$ appears in. Use the `numpy.log()` function to compute the log. Then, the TF-IDF feature you should use is

$$f_d(v) = tf(d, v) \times idf(v) \tag{4}$$

## 1.3 Naive Bayes Experiment [40 Points]

In the first experiment, you will build three Naive Bayes classifiers, each one using one of the above document representations, and compare their performances on the test dataset.

Recall that the prediction rule for Naive Bayes is

$$\arg\max_y P(d \mid y) \cdot P(y) \tag{5}$$

and so you need to compute those two quantities. The simpler of the two quantities $P(y)$ can be computed by counting the number of times each label appears divided by the total number of instances.

$$P(y) = \frac{|\{d : y_d = y\}|}{|\mathcal{D}|} \tag{6}$$

The other quantity can be computed as follows. Under the Naive Bayes assumption, $P(d \mid y)$ is defined as

$$P(d \mid y) = \prod_{v \in d} P(v \mid y) \tag{7}$$

For this question, you will use the features to compute $P(v \mid y)$ as follows:

$$P(v \mid y) = \frac{\sum_{d \in \mathcal{D}: y_d = y} f_d(v)}{\sum_{w \in \mathcal{V}} \sum_{d \in \mathcal{D}: y_d = y} f_d(w)} \tag{8}$$

The summands in the numerator and inner summand of the denominator are over all of the documents in the training data that have label $y$. The outer summand of the denominator is over all of the words in the vocabulary.

Finally, as you saw in class, you will implement Laplace smoothing on $P(v \mid y)$ as follows:

$$P(v \mid y) = \frac{k + \sum_{d \in \mathcal{D}: y_d = y} f_d(v)}{k \cdot |\mathcal{V}| + \sum_{w \in \mathcal{V}} \sum_{d \in \mathcal{D}: y_d = y} f_d(w)} \tag{9}$$

where $k$ is a hyperparameter which controls the strength of the smoothing. This is the final equation which you should implement to build your Naive Bayes classifier.

[3] You should use only the training documents to compute the IDF score. (A previous version of this assignment said you should use the training, validation, and testing. It is OK if you do either of these two options.)

**Implementation Details** When you are computing the values for the prediction rule (Equation 5), you need to multiply many probabilities together, which may result in underflow. Instead, you should implement the classifier in log-space. That is, instead of computing with probabilities $P(v \mid y)$, you should compute with $\log P(v \mid y)$ and use the following equation as the prediction rule:

$$\arg\max_y P(d \mid y) \cdot P(y) = \prod_{v \in d} P(v \mid y) \cdot P(y) \tag{10}$$

$$= \sum_{v \in d} \log P(v \mid y) + \log P(y) \tag{11}$$

Note that there is a term in the summand for each token in the document, so if a token appears twice, you will add the corresponding term twice.

**What to Report** For each of the three document representations, run a hyperparameter sweep over the smoothing parameter $k \in \{0.001, 0.01, 0.1, 1.0, 10.0\}$ using the training and validation data. Then, using the best value of $k$ for each representation, report and compare the test performance. Use accuracy to evaluate the models. Note that the most complicated document representation may not get the best results.

Answer the following question: As the value of $k$ goes to infinity, what will happen to $P(y \mid d)$?

## 1.4 Semi-Supervised Learning Experiment [20 Points]

In this experiment, you will implement a semi-supervised learning algorithm to learn from both labeled and unlabeled data. The high-level idea is as follows. You will start with two sets of data, a labeled set $(S_X, S_Y)$ and an unlabeled set $U_X$. Then you will repeatedly learn a model using the labeled data, predict labels on the unlabeled data, assume that the unlabeled instances the model labeled with high confidence were correctly labeled, then add those high-confidence unlabeled instances with the model predictions to the supervised set, then repeat. The exact implementation is provided in Algorithm 1.

In order to select the instances to add to the supervised set from the unsupervised set, you need to compute a confidence score. You should not use $P(d \mid y)$ because longer documents will be unfairly penalized (think about why). Instead, you should compute the confidence as follows:

$$\text{confidence} = \max_y P(y \mid d) \tag{12}$$

Then, you will experiment with two different ways of selecting instances based on their confidence scores as follows (these should be used to implement the "filter" function in the pseudocode):

1. **Threshold** Add all instances from $U$ to $S$ that have a confidence score above a predefined threshold.
2. **Top-$K$** Add the top-$K$ most-confidence instances in $U$ to $S$.

**Algorithm 1** Semi-Supervised Classifier

**procedure** SEMI-SUPERVISED CLASSIFIER
  **Input:** $S_X, S_Y, U_X$
  **while** not terminated **do**
    model $\leftarrow$ train_naive_bayes_classifier($S_X, S_Y$)
    $U_Y, P_Y \leftarrow$ predict(model, $U_X$)
    $S_X^{\text{new}}, S_Y^{\text{new}} \leftarrow$ filter($U_X, U_Y, P_Y$)
    **if** $|S_X^{\text{new}}| = 0$ **then**
      **return** model
    **end if**
    $S_X \leftarrow S_X \cup S_X^{new}$
    $S_Y \leftarrow S_Y \cup S_Y^{new}$
    $U_X \leftarrow U_X \setminus S_X^{new}$
  **end while**
**end procedure**

**What to Report** For each of the two filtering techniques, run the semi-supervised algorithm with 50, 500, and 5000 random training instances in $S$ and the rest in $U$. Report the initial accuracy (using just the starting $S$) and the final accuracy (when the algorithm terminates) on the validation dataset. Additionally report the accuracy after the algorithm terminates on the test dataset.

Use $k = 0.1$ for the Naive Bayes smoothing parameter. Use 0.98 as the confidence score filtering threshold. Use $K = 10000$ as the number of instances to take in the top-$K$ selection method. The accuracy may not always go up, and that is OK, but for some settings it should increase. The entire experiment should take around 5 minutes to run.

# 2 Theory [40 points]

## 2.1 Multivariate Exponential naïve Bayes [30 points]

In this question, we consider the problem of classifying piazza posts ($Y$) into two categories: student posts ($A$), and instructor posts ($B$). For every post, we have two attributes: number of words ($X_1$), and number of mathematical symbols ($X_2$). We assume that each attribute ($X_i$, $i = 1, 2$) is related to a post category ($A/B$) via an Exponential distribution[4] with a particular mean ($\lambda_{A;i}^{-1}/\lambda_{B;i}^{-1}$). That is

$$P(X_i = x \mid Y = A) = e^{-x\lambda_{A;i}}\lambda_{A;i} \quad \text{and} \quad P(X_i = x \mid Y = B) = e^{-x\lambda_{B;i}}\lambda_{B;i} \text{ for } i = 1, 2$$

[4] http://en.wikipedia.org/wiki/Exponential_distribution

| $X_1$ | $X_2$ | $Y$ |
|---|---|---|
| 0 | 3 | $A$ |
| 4 | 8 | $A$ |
| 2 | 4 | $A$ |
| 6 | 2 | $B$ |
| 3 | 5 | $B$ |
| 2 | 1 | $B$ |
| 5 | 4 | $B$ |

Table 1: Dataset for Exponential naïve Bayes

Assume that the given data in Table 1 is generated by a Exponential naïve Bayes model. You will use this data to develop a naïve Bayes predictor over the Exponential distribution.

| $P(Y{=}A) =$ | $P(Y{=}B) =$ |
|---|---|
| $\lambda_{A;1} =$ | $\lambda_{B;1} =$ |
| $\lambda_{A;2} =$ | $\lambda_{B;2} =$ |

Table 2: Parameters for Exponential naïve Bayes

1. **[10 points]** Compute the prior probabilities ($P(Y = A)$ and $P(Y = B)$). Use maximum likelihood estimation to find the corresponding $\lambda$ parameter values. Fill in the results in Table 2. Please show all the intermediate steps and results.
2. **[10 points]** Based on the parameter values from Table 2, compute

$$\frac{P(X_1{=}2, X_2{=}3|Y{=}A)}{P(X_1{=}2, X_2{=}3|Y{=}B)}$$

   Write the full expression. You do not need to simplify.
3. **[5 points]** Write the decision rule for the Exponential Naive Bayes predictor based on the previous equation. The answer we are looking for uses an if statement based on the value of the previous ratio and an additional probability term.
4. **[5 points]** Using the parameter values you estimated in the previous steps and the decision rule, what will the classifier predict as the value of $Y$, given the data point: $X_1{=}2, X_2{=}3$?

## 2.2 Coin Toss [10 points]

Consider the following way to generate a series of Heads and Tails. For each element in the series, first a coin is tossed. If it comes up as a $T$, it is shown to the user. On the other hand, if the coin toss comes up as an $H$, then the coin is tossed the second time, and the outcome of this toss is shown to the user.
Assume that the probability of a coin toss coming up as an H is p (and hence, the probability

of it coming up as a $T$ is $1 - p$). Suppose you see a sequence $TTHTHHTHTT$ generated based on the scheme given above. What is the most likely value of $p$?
(Assume a Bernoulli model to compute probability of the coin toss sequence.)

# Submission Instructions

We will be using Gradescope to turn in both the Python code and writeup pdfs. You should have been automatically added to Gradescope. If you do not have access, please ask the TA staff on Piazza.

For this homework assignment, there are two Gradescope assignments:

- "Homework 4 - Code": This is the assignment where you should upload your a python version of your Jupyter Notebook and your implementations of `BBoWFeaturizer`, `CBoWFeaturizer`, `TFIDFFeaturizer`, `compute_idf`, `get_vocabulary`, `train_naive_bayes`, and `predict_naive_bayes` for unit testing. Please name your file `hw4.py`
- "Homework 4 - PDF": This is the assignment where you should upload your writeup as a PDF.

# Changelog

## Version 2

1. Added the "log" to the IDF equation
2. Changed the footnote about which documents should be used to compute the IDF score. The original note said to use training, validation, and testing when it should really only be the training documents. It is OK if you do not update your code to only use the training documents.