CS 4650 Fall 2021: Homework 1

August 30, 2021

Instructions

1. This homework has two parts: questions 1–4 are theory questions, and Q5 is a programming assignment with some parts requiring a written answer.

   We will be using Gradescope to collect your assignments. Please read the following instructions for submitting to Gradescope carefully!

   (a) Each subproblem must be submitted on a separate page. When submitting to Gradescope (under HW1 Writing), make sure to mark which page(s) correspond to each problem or subproblem. For instance, Q2 has 3 subproblems, so the solution to each must start on a new page. Similarly, Q3 and Q4 each have 3 subproblems, so the writeup for each subproblem should start on a new page.

   (b) For the coding problem (Q5), please upload all codes under the HW1 Code assignment on Gradescope. In addition, include the writeup for each of Q5's subproblems in your solution PDF.

   (c) Note: This is a large class and Gradescope's assignment segmentation features are essential. Failure to follow these instructions may result in parts of your assignment not being graded. We will not entertain regrading requests for failure to follow instructions.

2. LaTeX solutions are strongly encouraged (solution template available on the class website), but scanned handwritten copies are also acceptable. Hard copies are not accepted.

3. Question 1 is worth 5 points, questions 2-4 are worth 10 points, and question 5 is worth 50 points (20 points for writing and 30 for coding) plus a bonus.

4. We generally encourage collaboration with other students. You may discuss the questions and potential directions for solving them with another student. However, you need to write your own solutions and code separately, and not as a group activity. Please list the students you collaborated with on the submission site.

Questions

1. Provide answers to the following operations or write "invalid" if not possible (all answers can be placed in a single page): [5 points]

   (a) $\begin{bmatrix} 5 & 7 \\ 3 & 3 \end{bmatrix} \begin{bmatrix} 4 \\ 8 \end{bmatrix}$

   (b) $\begin{bmatrix} 0 & 2 \\ 4 & 1 \\ 6 & 0 \end{bmatrix} \begin{bmatrix} 0 & 2 \\ 2 & 5 \\ 4 & 6 \end{bmatrix}^\intercal$

   (c) $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 5 & 1 \end{bmatrix}$

   (d) $\begin{bmatrix} 2 & 5 \end{bmatrix}^\intercal \begin{bmatrix} 3 & 1 \\ 5 & 0 \end{bmatrix}^\intercal$

   (e) $\begin{bmatrix} 2 & -1 & 0 \end{bmatrix}^\intercal \begin{bmatrix} 2 & 6 \end{bmatrix}$

2. Write regular expressions to help solve the following cases. For parts (a) and (b), the regex expression provided will split on the match in a string (you can look at the Python API for re.split to see example behavior). Part (c) simply uses the regex to test for a match (see re.match).

   (a) Construct a regex to tokenize English text written in the TitleCase (each first letter is capitalized with no spaces between words). Note that the text can also have numbers and punctuation. [3 points]

   Example:
   EngineeringAndComputingColleges → [‘Engineering’, ‘And’, ‘Computing’, ‘Colleges’]

   (b) Construct a regex for a sentence tokenizer that takes an English paragraph as input and divides it into sentences (assume sentences only end in a single period, and have whitespace after the period). [3 points]

   Example:
   I have a grey cat. My cat’s name is Kiwi. → [’I have a grey cat.’, ’My cat’s name is Kiwi.’]

   (c) Construct a regex that matches on strings that contain the individual words ‘NLP’ and ‘BERT’ *without* matching the words that contain them, such as ‘BioBERT.’ [4 points]

   Examples:
   I love NLP, and I use BERT → Match.
   I develop NLP solutions for healthcare, and I use BioBERT → No Match.
   I specialize in NLP for health, and I use BERT and BioBERT→ Match.

3. You've been hired as a consultant in a firm working to distinguish between *fake* and *real* news. Though there are many publishers, the firm has identified that the problem is hardest when news articles are from two publishers: New York Times (NYT) and Buzzfeed (BF). Before building a classifier, researchers at the firm want to run some numbers and they come to you. It's time to show off your knowledge!

   Note: Assume for the following questions that there are only two types of news articles. Write your answer using up to 3 significant figures.

   (a) John, the head of the firm, thinks that only 10% of all articles in all newspapers are fake. But he has seen that 55% of all fake news comes from BF. BF also accounts for 35% of all news articles. What is the probability that a randomly selected article from BF is fake, according to John's estimates? [3 points]

   (b) Christina, another employee, has multiple news provider subscriptions. She has counted that for every 1000 articles that she sees on her feed, she ends up reading the whole article 40 times. If she reads an article it ends up being fake news half of the time. Furthermore, the fake articles she read were from NYT 60% of the time. For a random article that appears on her feed, what is the probability that she ends up reading it, it turns out to be fake, and is from NYT? [3 points]

   (c) Gary thinks the fake news problem is much worse. He estimates that 15% of all articles in all newspapers are fake. He further finds that 20% of all fake news is from NYT and 60% of all real articles comes from NYT. What is the probability that a randomly chosen NYT article is fake news, according to Gary's estimate? [4 points]

4. A collection of movie reviews (data $\mathcal{D}$) contains the following keywords and binary labels for whether each review was positive ($+$) or negative ($-$). The data is shown below: for example, the cell at the intersection of "Review 1" and "epic" indicates that the text of Review 1 contains 2 tokens of the word "epic".
   Answer the following questions, reporting all scores as **log-probabilities**, to three significant figures.

Review	great	amazing	epic	boring	terrible	disappointing	Y
1	2	2	2	1	1	0	+
2	1	4	0	0	0	1	+
3	3	2	3	1	0	0	+
4	0	1	1	2	1	2	-
5	1	0	2	1	2	2	-
6	1	0	1	2	1	2	-

   (a) Assume that you have trained a Naive Bayes model on data $\mathcal{D}$ to detect positive vs. negative movie reviews. Compute the model's predicted scores for both positive and negative classes for the following sentence S (i.e. $P(+|S)$ and $P(-|S)$), and determine which label the model will apply to S. [4 points]

   S: `The film was great, the plot was simply amazing! Makes other superhero movies look terrible, this was not disappointing.`

   (b) The counts in the original data are sparse and may lead to overfitting, e.g. a strong prior on assigning the negative label to reviews that contain "terrible". What would happen if you applied *smoothing*?

   Apply *add-1* smoothing and recompute the Naive Bayes model's predicted scores for S. Did the predicted label change? [4 points]

   (c) What is an additional feature that you could extract from text to improve the classification of sentences like S (not necessarily in NB), and how would it help improve the classification? [2 points]

5. In this problem, you will do text classifications for Hate Speech. You need to both answer the questions and submit your code.

   Hate speech is a

   (a) **deliberate attack,**

   (b) **directed towards a specific group of people,**

   (c) **motivated by aspects of the group's identity.**

   The three premises must be true for a sentence to be categorized as HATE. Here are two examples:

   (a) "Poor white kids being forced to treat apes and parasites as their equals."

   (b) "Islam is a false religion however unlike some other false religions it is crude and appeals to crude people such as arabs."

   In (a), the speaker uses "apes" and "parasites" to refer to children of dark skin and implies they are not equal to "white kids". That is, it is an attack to the group composed of children of dark skin based on an identifying characteristic, namely, their skin colour. Thus, all the premises are true and (a) is a valid example of HATE. Example (b) brands all people of Arab origin as crude. That is, it attacks the group composed of Arab people based on their origin. Thus, all the premises are true and (b) is a valid example of HATE.

   This problem will require programming in **Python 3**. The goal is to build a **Naive Bayes model** and a **logistic regression model** that you learnt from the class on a real-world hate speech classification dataset. Finally, you will explore how to design better features and improve the accuracy of your models for this task.

   The dataset you will be using is collected from Twitter online. Each example is labeled as 1 (hatespeech) or 0 (Non-hatespeech). To get started, you should first download the data and starter code from the following location:

   https://www.cc.gatech.edu/classes/AY2022/cs4650_fall/programming/h1_text_classification.zip.

   Try to run:

   ```
   python main.py --model AlwaysPredictZero
   ```

   This will load the data and run a default classifier `AlwaysPredictZero` which always predicts label 0 (non-hatespeech). You should be able to see the reported train accuracy = 0.4997. That says, always predicting non-hatespeech isn't that good. Let's try to build better classifiers!

   Note that you need to implement models without use of any machine learning packages such as `sklearn`. Your solution should use simple Python lists or `numpy` arrays. (`numpy` arrays are recommended since they are much faster than Python lists. Your program should finish running within an acceptable time limit in Gradescope.) Note the Gradescope machine will not have any external packages installed so make sure

to also delete/comment any statements importing external packages. We will only provide a training set, and evaluate your code based on our test set.

To have a quick check with your implementations, you can randomly split the dataset we give you into train and test set at a ratio of 8:2, compare the accuracy between the models you have implemented and related models in `sklearn` packages. You are encouraged to try `sklearn` in non-submitted code, to get an idea of what the expected outputs should be and how much training time your models should require. None of the models you implement for this question should take longer than a few minutes to train.

Once you are done, please **only submit** your `utils.py` and `classifiers.py` to Gradescope.

(a) **(Naive Bayes)** In this part, you should implement a Naive Bayes model with add-1 smoothing, as taught in the class. You are required to implement the `NaiveBayesClassifier` class in `classifiers.py`. You would probably want to take a look at the `UnigramFeature` class in `utils.py` that we have implemented for you already. After you finish coding, run `python main.py --model NaiveBayes` to check the performance.
We will test your implementation on our test set. [15 points if `accuracy` $\geq$ 0.65, 9 pts if `accuracy` $\geq$ 0.60]
In your writeup, **list** the 10 words that, under your model, have the highest ratio of $\frac{P(w|1)}{P(w|0)}$ (the most distinctly hatespeech words) and **list** the 10 words with the lowest ratio. What trends do you see? [10 points]

(b) **(Logistic Regression)** In this part, you will implement a Logistic Regression model. You are required to implement the `LogisticRegressionClassifier` class in `classifiers.py`. First, implement a logistic regression model without regularization and run `python main.py --model LogisticRegression`, compare the performance with your Naive Bayes approach. Describe how both of these models differ from the perceptron classifier we talked about in class. Next, we would like to experiment with L_2 regularization. Add L_2 regularization with different weights, such as $\lambda = \{0.0001, 0.001, 0.01, 0.1, 1, 10\}$.
In your writeup, describe what you observed. (Again, you may want to split the training set we give you into your own training and test sets to observe the performance.) [10 points]
For the **code** submission, hard-code the best λ you observed. Your model accuracy on our test set must be greater than 0.68 to receive full credit. Similar to part (a), partial credit will be awarded for accuracy above 0.60. Please make sure your model doesn't train for too long (more than a few minutes) locally, otherwise gradescope will time out while grading your assignment and you will receive no credit. [15 points]

(c) **(Bonus)** In the last part, you'll explore and implement a more sophisicated set of features. You need to implement the class `BigramFeature` or modify the class `CustomFeature` in `utils.py`. Here are some common strategies (you are welcome to implement some of them but try to come up with more!):

i. Remove stopwords (e.g. a, the, in),
ii. Use a mixture of unigrams, bigrams or trigrams, (It may take exponentially more time to run bigram features than unigram features. Make sure total running time is within the acceptable time limit in Gradescope. For example, you can choose to implement bigrams based on Naive Bayes model, since it takes much less time than Logistic Regression.)
iii. Use TF-IDF (refer to `http://www.tfidf.com/`) features.

Use your creativity for this problem and try to obtain an accuracy as high as possible on your test set! We will call the `BonusClassifier`, which should simply specify which classifier to use, and the `CustomFeature` class which should contain any custom features you'd like to be used for this part. **Note**: if these two classes are unimplemented, Gradescope will not be able to run your code and you will not receive any bonus.

In your write, describe the features you implemented, the model you decided to use for this part and your reasoning for this. [Bonus: 10 points]

You will receive up to 10 bonus points: up to 3 points based on novel features you try, 2 points based on whether you implemented perceptron, and the rest based on how well your model performs compared to other submissions:

$$\text{Bonus} = 5 + 5 \cdot \frac{1}{\text{rank}}$$

e.g. if you ranked first in the class, implemented novel features and perceptron, you will receive the full bonus points! If you implemented new features but did not implement perceptron and ranked first in class, you will receive 8 points.