CS373: Homework 3

Due Date: 11:59 PM, April 1, 2019

Instructions for submission: In this programming assignment you will implement the Perceptron and the Naive Bayes algorithms and evaluate them on the Review Polarity dataset. Instructions below detail how to turn in your code and assignment on `data.cs.purdue.edu`

You are given a skeleton code with an existing folder structure. Do not modify the folder structure. You may add any extra files you want to add in the `cs373-hw3/src/` folder. For all other parts write your answers in a single PDF. Name this `report.pdf`. Place your report in the `cs373-hw3/report/` folder. Label all the plots with the question number. Your homework must contain your name and Purdue ID at the start of the file. If you omit your name or the question numbers for the plots, you will be penalized.

To submit your assignment, log into `data.cs.purdue.edu` (physically go to the lab or use ssh remotely) and follow these steps:

1. Place all of your code in the `cs373-hw3/src/` folder and your report in `cs373-hw3/report/` folder.
2. Change directory to outside of `cs373-hw3/` folder (run `cd ..` from inside `cs373-hw3/` folder)
3. Execute the following command to turnin your code: `turnin -c cs373 -p hw3 cs373-hw3`
4. To overwrite an old submission, simply execute this command again.
5. To verify the contents of your submission, execute this command: `turnin -v -c cs373 -p hw3`. Do not forget the `-v` option, else your submission will be overwritten with an empty submission.

1 Specification

1.1 Dataset

In this homework, you will be working with the 'Review Polarity' dataset. You are given movie reviews and the task is to classify them as `positive` or `negative`. The entire dataset has 1000 movie reviews, out of which 500 are labeled `positive` and the other 500 are labeled as `negative`. The dataset is split into training (`train.tsv`) and test (`test.tsv`) sets. Training set consists of 699 instances and the test set has 301 instances. Look into the training set file and try to gauge the nature of the data and the task.

Your submission will be tested on a hidden dataset which is different from given dataset. The hidden set would have similar number of instances and data splits.

Input: Movie review consisting of a few sentences of text.
Label: Positive (+1)/Negative (−1)

The features you will use for the classification task are bag-of-words. Provided skeleton code has 2 feature extractors already implemented. The first feature extractor extracts binary word presence/absence and the second feature extractor extracts word frequency. You are free to implement and add any features that you believe would help you get a better performance on the task, but you can get full credit without using any extra features. Top-10% submissions according to the performance on the hidden test set will be given extra credit. You might need to use additional features to get extra credit. You are allowed to use libraries such as `NLTK` or `Spacy` to extract those features.

1.2 Skeleton Code

You are provided a skeleton code with this homework. The code has the following folder structure:

```
cs373-hw3/
├── handout.pdf
├── data/
│   └── given/
│       ├── train.tsv
│       └── test.tsv
├── src/
│   ├── __init__.py
│   ├── main.py
│   ├── classifier.py
│   ├── utils.py
│   ├── config.py
│   ├── naive_bayes.py
│   └── perceptron.py
└── report/
    └── report.pdf
```

Do not modify the given folder structure. You should not need to, but you may, add any extra files of code in `cs373-hw3/src/` folder. You should place your report (`report.pdf`) in the `cs373-hw3/report/` folder. You **must not** modify `__init__.py`, `main.py` or `classifier.py`. They may be replaced while grading. All your coding work for this homework must be done inside `cs373-hw3/src/` folder. You are not allowed to use any external libraries for classifier implementations such as `scikit-learn` etc. Make sure that you understand the given code fully, before you start coding.

Your code will be tested using the command `python2.7 main.py` from inside the `cs373-hw3/src/` folder. Make sure that you test your code on `data.cs.purdue.edu` before submitting. Output of the TA solution is given below. Your final numbers might differ. The given numbers are just a guideline, meant for sanity checking.

```
Perceptron Results:
Accuracy: 62.46, Precision: 97.27, Recall: 24.32, F1: 38.91

Averaged Perceptron Results:
Accuracy: 66.45, Precision: 91.21, Recall: 35.13, F1: 50.73

Naive Bayes Results:
Accuracy: 77.74, Precision: 78.72, Recall: 74.99, F1: 76.81
```

2 Perceptron (45 points)

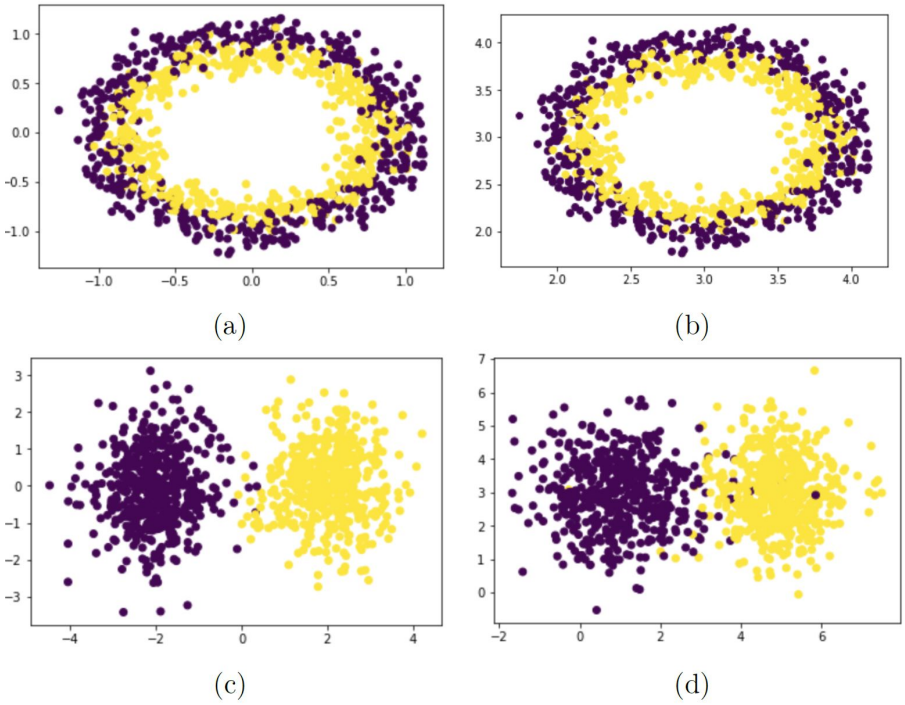
2.1 Theory (15 points)

1. (5 points) Perceptron model discussed in class is shown as:

$$f(x) = \begin{cases} 1 & \text{if } \Sigma w_j x_j > 0 \\ 0 & \text{if } \Sigma w_j x_j \leq 0 \end{cases}$$

   The *bias* term is missing in the shown equation. Write the equation with the *bias* term included. Is perceptron with a *bias* term more expressive (can represent more classification scenarios) compared to the one without bias? Why or why not?

2. (5 points) Given below are four figures that show the distribution of data points with binary classes. The two colors denote the different classes. For each of these, reason whether the following would give a high(> 0.95) classification accuracy.
   (i) a perceptron without bias
   (ii) a perceptron with bias



3. (5 points) What is the update rule for the *bias* term b of a vanilla perceptron, given the learning rate γ and gold label y, when the classifier label doesn't match the gold label during training? What is the update rule when the classifier label matches the gold label?

2.2 Implementation (30 points)

You need to implement a vanilla perceptron and an averaged perceptron model for this part. Both the models should be implemented with a bias term. This part could be completed by editing only `perceptron.py`, unless you plan to extract any new features for the task. You need to initialize the parameters (`__init__()` method), learn them from training data (`fit()` method) and use the learned parameters to classify new instances (`predict()` method) for each of the models. Take note that `__init__.py`, `main.py` and `classifier.py` may be replaced while grading. Do not make any modifications to these files. You need to follow the description of the models discussed in the lecture slides (link). Report the results obtained on the given test set for both the models in your report. You should submit your code with the hyperparameter settings (`config.py`) that produce the best performance.

3 Naive Bayes (40 points)

3.1 Theory (20 points)

1. (5 points) Given a text document d which is a sequence of words $w_1, w_2, \ldots, w_n$, we want to compute $P(c^+|d)$ and $P(c^-|d)$. We use Bayes theorem to estimate the probabilities. Compute the equation for $P(c^+|d)$ in terms of $P(d|c^+)$ using Bayes theorem.

2. (5 points) To estimate $P(d|c^+)$ using the training data without making any assumptions, we need impractically large amounts of data. Let us say that the size of the vocabulary is V and length of all documents is exactly l (we can ensure this by padding shorter texts with dummy tokens). In a binary classification task, how many parameters do we need to learn, in order to correctly estimate $P(d|c^+)$ for any given document without making independence assumptions?

3. (5 points) If we make the unigram assumption, that is, if we assume that occurrence of each word in the document is independent of the other words, then how many parameters do we need to learn to be able to estimate $P(d|c^+)$?

4. (5 points) In a binary text classification task, mention the equations you would use to estimate $P(c^+)$ and $P(c^-)$ from the training data (c^+ and c^- are the two classes for the classification problem).

3.2 Implementation (20 points)

You need to implement a Naive Bayes classifier for the given task. This part could be completed by editing only `naive_bayes.py`, unless you plan to extract any new features for the task. You need to initialize the parameters (`__init__()` method), learn them from training data (`fit()` method) and use the learned parameters to classify new instances (`predict()` method). Take note that `__init__.py`, `main.py` and `classifier.py` may be replaced while grading. Do not make any modifications to these files. You need to follow the description of the model discussed in the lecture slides (link). Report the results obtained on the given test set in your report. You should submit your code with the hyperparameter settings (`config.py`) that produce the best performance.

4 Analysis (15 points)

You need to perform additional experiments to answer the following questions. You don't need to submit your code for this part. You only need to include your plots and discussions in your report. Make sure that the code you submit doesn't include any changes you don't want to be included, as that might affect your chances of getting extra credit.

1. (5 points) Remove the bias term from the Perceptron and Averaged Perceptron and report the performance of the models. Is the performance better or worse than the respective models with a bias term? Discuss in one sentence.

2. (5 points) Plot a graph of test accuracy vs number of iterations for Averaged Perceptron for number of iterations = { 1, 2, 5, 10, 20, 50 }. Does Perceptron converge i.e. does the training accuracy become 100? If yes, after how many iterations? If not, why?

3. (5 points) Plot graphs of Naive Bayes test accuracy vs vocabulary size, Averaged Perceptron test accuracy vs vocabulary size for vocabulary sizes = {100, 500, 1000, 5000, 10000, 20000}. How does the performance change? Discuss in two sentences.

5 Time Limit

Your code must terminate with in 2 minutes for all 3 models combined together. If it doesn't terminate with in 2 minutes, you will be graded for the output your code generates at the end of the 2 minute time limit.

6 Extra Credit (10 points)

Your submission will be evaluated on a hidden dataset of similar nature. The hidden dataset is of similar size and splits. Top-10% (17 in number) of the students in the class would be awarded 10 extra credit points. You are not allowed to use any optimization libraries but you are allowed to use feature extraction software. If in doubt, ask a question on piazza and the instructors would respond. Remember that the extra credit only depends on the results on the hidden dataset. Overfitting the given dataset might prove counter-productive.