CSE 353 – Homework I

Instructor: Dr. Ritwik Banerjee

Submission Deadline: Mar 20, 2017, 11:59 pm

This homework document consists of 3 pages. Carefully read the entire document before you start coding. You are required to implement a perceptron and a naïve Bayes classifier for a binary classification task. The dataset you will be working on is a well-known (famous, actually!) movie review dataset.
The homework has four components: (i) data processing, (ii) perceptron classification, (iii) naïve Bayes classification, and (iv) the final report. These are described below:

I: Data Processing **20 points**

The movie review dataset is natural language data, with an equal number of reviews in the "positive" and the "negative" classes. The data processing step is responsible for converting this natural language data into vector data. That is, after this step, you will have a numeric vector for each movie review. Data processing should be *modular* in the sense that the vector form of a text document can be different, depending on your method. For example, on one hand, you may choose to include or exclude punctuation from being features. Further, the data processing stage should allow the user to quickly run your code with one or more such (compatible) options, with one set of options being the default setting. To illustrate this, consider two very short movie reviews:

R_1: was amazed to see how good the action sequence was !

R_2: the action sequence in the end was so good ! ! !

The dataset thus has 13 unique tokens including punctuation, and 12 without punctuation. Clearly, this choice affects the order of your model. Similarly, if "was" corresponds to the first feature in your input vector, $\mathbf{x}(R_1)$ may have $x_1 = 1$ to simply indicate the presence of the word, or $x_1 = 2$ to reflect the frequency of the feature. Thus, the processing also affects the actual input vector representation of the document.

A modular data processing code called, say, `processor.py`, should allow the user to quickly experiment with different processing options by using it as follows:

```
$~ ./processor.py --frequency --nopunct
$~ ./processor.py --binary --punct
```

Of course, you should not be able to specify the `nopunct` and `punct` options at the same time! Also, you are encouraged to think of similar other variations for the data processing stage.

II: Perceptron Classification **30 points**

You must write your own perceptron code, where the perceptron will be used for binary classification of movie reviews into the two classes, *positive* and *negative*. Note that all the data is already labeled.

You cannot train the perceptron classifier on all the labeled data, and then test it on the same data! As a result, you should perform 5-fold cross-validation. We discussed cross-validation in class, and an algorithmic description of cross-validation is given at the end of this document.

You should keep in mind that the classifier may not converge for a given dataset. In that case, you will have to specify some external stopping criteria (such as training data exhausted or some number of training iterations are already complete).

III: Naïve Bayes Classification **30 points**

Just like the perceptron, you will have to write your own naïve Bayes classifier for this part of the assignment. Once again, you should perform 5-fold cross validation to assess how good the classifier is. Here, you will need to use the m-estimation technique to handle situations where the probability becomes zero.

IV: Final Report **20 points**

The final report should not be longer than two pages, and should compare the classifier performance[1] on the movie review dataset across both classifiers, and also across all the different types of data processing. You should report not only the average performance of all folds in each case, but also the maximum and minimum. For example, the report should have the average accuracy of the perceptron classifier with no punctuation features and also mention the worst and the best folds so that the reader is aware of the variations in experimental results. The report should also devote some portion to explain how to run your code (nothing too detailed, but it should allow a user to run your code without having to read and understand it).

The report will be graded based on (a) performance details, (b) replicability of experiments, (c) explanation of how to run your code, and (d) any heuristics you may have adopted (*e.g.*, the stopping criteria for your perceptron when training does not achieve convergence.

Notes

What programming languages are allowed?

Java, Python, Perl, Ruby, C#, C++, C, Scala are allowed. There may be exceptions made for one or two more programming languages if you can argue passionately about how amazing that language is for machine learning algorithms AND that it is not too unpopular among machine learning practitioners. But, specifically, you may **NOT** use MATLAB, R or any other language/library that provides readymade functionality to do what this assignment asks you to do (including data processing).

What should we submit, and how?

Submit a single `.zip` archive containing one folder simply called "code", containing all your code, and a PDF document for your report. Do **NOT** submit a report in any other format.

[1] By 'performance', we mean the three measurements *precision*, *recall*, and *accuracy*. The first two are class-specific, so they should be measured for each class separately, and then averaged. Definitions provided in Appendix B.

Appendix A: Performance Measurements

The items correctly labeled as belonging to the positive class are called *true positives* (TP). The items incorrectly labeled as belonging to the positive class are called *false positives* (FP). Similarly, we have *true negatives* (TN) and *false negatives* (FN).

$$\begin{aligned}
\textbf{Precision}^{+} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \\
\textbf{Recall}^{+} &= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \\
\textbf{Accuracy} &= \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}} \\
\textbf{Precision} &= \frac{\left(\textbf{Precision}^{+}+\textbf{Precision}^{-}\right)}{2} \\
\textbf{Recall} &= \frac{\left(\textbf{Recall}^{+}+\textbf{Recall}^{-}\right)}{2}
\end{aligned}$$

Appendix B: Cross Validation

1: **procedure** n-FOLD CROSS VALIDATION
2: Divide the dataset D into n sub-datasets of equal size, D_1, D_2, ..., D_n
3: **for** $i \leftarrow 0$ **to** n **do**
4: Train classifier on $\bigcup_{j=1\ldots n, j\neq 1} D_j$
5: Test classifier on D_i
6: $R_i \leftarrow$ test results (accuracy, precision and recall)
7: Report overall performance of the classifier as the average accuracy, precision and recall across all folds (*i.e.*, average of $R_i, 1 \leq i \leq n$).