10601 Machine Learning Assignment 2: Naive Bayes and Voted Perceptron

Due: Sept. 18th, 23:59 EST, via AutoLab
Late submission due: Sept. 20th, 23:59 EST with 50% discount of credits

TAs-in-charge: Guanyu Wang, William Wang

Policy on Collaboration among Students

These policies are the same as were used in Dr. Rosenfeld's previous version of 2013. The purpose of student collaboration is to facilitate learning, not to circumvent it. Studying the material in groups is strongly encouraged. It is also allowed to seek help from other students in understanding the material needed to solve a particular homework problem, provided no written notes are shared, or are taken at that time, and provided learning is facilitated, not circumvented. The actual solution must be done by each student alone, and the student should be ready to reproduce their solution upon request. The presence or absence of any form of help or collaboration, whether given or received, must be explicitly stated and disclosed in full by all involved, on the first page of their assignment. Specifically, each assignment solution must start by answering the following questions in the report:

- Did you receive any help whatsoever from anyone in solving this assignment? Yes / No. If you answered 'yes', give full details: _____ (e.g. "Jane explained to me what is asked in Question 3.4")
- Did you give any help whatsoever to anyone in solving this assignment? Yes / No. If you answered 'yes', give full details: _____ (e.g. "I pointed Joe to section 2.3 to help him with Question 2".

Collaboration without full disclosure will be handled severely, in compliance with CMU's Policy on Cheating and Plagiarism. As a related point, some of the homework assignments used in this class may have been used in prior versions of this class, or in classes at other institutions. Avoiding the use of heavily tested assignments will detract from the main purpose of these assignments, which is to reinforce the material and stimulate thinking. Because some of these assignments may have been used before, solutions to them may be

(or may have been) available online, or from other people. It is explicitly forbidden to use any such sources, or to consult people who have solved these problems before. You must solve the homework assignments completely on your own. I will mostly rely on your wisdom and honor to follow this rule, but if a violation is detected it will be dealt with harshly. Collaboration with other students who are currently taking the class is allowed, but only under the conditions stated below.

1 HelloWorld in Autolab

Before you start the homework 2, you are encouraged to log on the Autolab website (`https://autolab.cs.cmu.edu/10601-f13/`) using your Andrew ID, and get familiar with Matlab, Octave, and Autolab through this HelloWorld program. To do this, you just need to print "Hello World" in Matlab or Octave, and submit the corresponding HelloWorld.m file. Note that this program will not be a part of your grade.

2 Naive Bayes

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.

2.1 Algorithm Description

First, let y be the labels for the training documents and $\mathcal{W} = \{w_i\}_{i=1}^{L}$ be the set 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 (see Section 4 for details). Then it is easy to get all the counters with matrix operations in Octave.

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, for avoiding 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.

2.2 Implementation Notes

At classification time, please use Laplace smoothing with $\alpha = 1$ as described here:`http://en.wikipedia.org/wiki/Additive_smoothing`.

You should implement your own Naive Bayes algorithm using Octave (very similar as Matlab, actually most of the time they are compatible with each other). Your implementation should have three functions:

```
model = nb_train(Xtrain, Ytrain)
Pred_nb = nb_test(model, Xtest)
nb_run(Xtrain, Ytrain, Xtest, Ytest)
```

Here X, Y are sets of document feature vectors and labels respectively, *model* is a container for all the model parameters, you can use it to store the counters you generate. $Pred_nb$ is the predicated label for the input test data $Xtest$ based on your model (its true labels contained in $Ytest$). The last nb_run function should basically call nb_train first and then nb_test, and store the final predication *Pred_nb* in **Pred_nb.mat**. Finally, it should print out your testing precision, recall and accuracy on screen.
You can use

```
save Pred_nb.mat Pred_nb;
fprintf('P:%.3f, R:%.3f, A:%.3f', precision, recall, accuracy);
```

to save the **.mat** file and output the three values on screen.

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

$$precision = \frac{d}{b+d}, \quad recall = \frac{d}{c+d}, \quad accuracy = \frac{a+d}{a+b+c+d} \tag{3}$$

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

Figure 1: Precision, Recall and Accuracy

Note: Matlab or Octave's programs (scripts/functions) are saved as the .m files. However, files with a .mat extension contain Matlab formatted data, which can be loaded from or written to these files using the functions "load" and "save".

3 Voted Perceptron

Perceptron algorithm is one of the classic algorithms which has been used in machine learning from early 1960s. It is an online style 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.

3.1 Algorithm Description

Let $\mathcal{T} = \{(y_i, \mathbf{x_i})\}_{\mathbf{i=1}}^{\mathbf{m}}$ 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 $\forall i, y_i \langle w, x_i \rangle > 0$ ($\langle \cdot, \cdot \rangle$ is the inner product operation). For the Voted Perceptron, you need a list of weighted vectors $\{(\mathbf{w_k}, c_k)\}_{k=1}^{K}$, where $\mathbf{w_k}$ is the vector and c_k is its weight. (Please refer `http://curtis.ml.cmu.edu/w/courses/index.php/Voted_Perceptron` for a clear description 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 if
end for
$t = t + 1$
end while

Then you can use the 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

$$\hat{y} = sign(\sum_{k=1}^{K} c_k sign(\langle \mathbf{x}, \mathbf{w_k} \rangle)) \tag{4}$$

3.2 Implementation Notes

You should implement the Voted Perceptron algorithm using Octave. Your implementation should have three similar functions as your implementation of Naive Bayes:

```
model = perceptron_train(Xtrain, Ytrain)
Pred_per = perceptron_test(model, Xtest)
perceptron_run(Xtrain, Ytrain, Xtest, Ytest)
```

All the parameters are the same as in the Naive Baye implementation notes 2.2. At this time, you might want to save your perceptron vectors and their weights $\{(\mathbf{w_k}, c_k)\}_{k=1}^{K}$ in the model container.
Still, you should save the *Pred_per* and output the precision, recall as well as accuracy using

```
save Pred_per.mat Pred_per;
fprintf('P:%.3f, R:%.3f, A:%.3f', precision, recall, accuracy);
```

4 Data Set

For this assignment, you should download the handout data from `http://www.cs.cmu.edu/~yww/class/hw2.zip`. It contains two items:

- the **handout.mat** file
- the resource folder (this is not necessary for completing your assignment)

If you load the **handout.mat** into Octave (or Matlab) with

```
load 'handout.mat'
```

there are six matrices have been included: "Xtrain", "Ytrain", "Xtest", "Ytest", "DocIdtrain", "DocIdtest".

In the X (training set and testing set) matrices, 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. The word Id is defined in the dictionary (which can be found in the resource folder). The Y matrices provide the binary labels for corresponding documents in the X matrices. For example, assume the first row in the Xtrain matrix is:

```
0  1  2  10  0  2  ...
```

this 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.

The corresponding label for this document has been saved at the first row of the Ytrain matrix.

There are two different kind of documents (original documents are also included in the resource folder):

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

The DocId matrices are the corresponding DocIds defined in docId.txt file which is also included in the source folder. The docIds are not necessary for the algorithms, but combing docId, wordId as well as the dictionary, you can create the training and testing data by yourself without using the given .mat file (**This is not required in this assignment**).

The way we convert all the articles into words (features) is to use the following function (wrote in Java). Then all the words have been assigned with a id in the dictionary file.

```
static Vector<String> tokenizeDoc(String cur_doc) {
        String[] words = cur_doc.split("\\s+");
        Vector<String> tokens = new Vector<String>();
        for (int i = 0; i < words.length; i++) {
                words[i] = words[i].replaceAll("\\W", "");
                if (words[i].length() > 0) {
                        tokens.add(words[i]);
                }
        }
        return tokens;
 }
```

Note: you do not need to use anything included in the resource folder in this assignment, which is just for someone who might want to create your own training and testing data, or someone who might want to check the original documents.

5 Suggestions

- You might want to develop/test your code in Matlab, but you **must verify** that your programs work in octave as the last step.
- You do not need to install Matlab or Octave, since they are already installed on the clusters and andrew machines. (You can use the following command to connect to andrew linux machines:

  `ssh yourandrewid@linux.andrew.cmu.edu`

6 Deliverables

Submit your codes of two algorithm implementations via AutoLab. You should implement the Naive Bayes and Voted Perceptron algorithm by yourself instead of using any built-in functions.

You should upload your codes (including all 6 function files) along with a report, which should solve the following question:

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

You should tar gzip the following items into **hw2.tgz** and submit to the homework 2 assignment under Autolab:

- nb_train.m
- nb_test.m
- nb_run.m
- perceptron_train.m
- perceptron_test.m
- perceptron_run.m

- report.pdf

Tar gzip the files directly using "tar -cvf hw2.tgz *.m report.pdf". Do **NOT** put the above files in a folder and then tar gzip the folder. You do not need to upload the saved predicted labels (i.e. the .mat files). In addition to the provided data, we will test your codes with a held-out testing set as well. Please make sure your code is working fine under Octave before you submit. Again, note that you are not allowed to use the NaiveBayes or perceptron functions from Matlab for this homework. In fact, those functions will not work in Octave. You have a total of 5 possible submissions.