CS412 Assignment 4 - Classification MP (Distributed Nov. 1, 2014, Due Dec. 3, 2014)

Before Diving in

- Programming assignments take more time, and it counts more in your final grade, so please start early.
- This is an individual assignment. You can discuss this assignment on Piazza but please do not work together or share code or results.
- Libraries about classification methods can be found on-line, but you are prohibited to use these resources directly. The purpose of this programming assignment is to learn how to build a classification framework step by step.
- You can use C/C++ or Java or Python2 as your programming language. Your programs should be able to compile correctly in EWS Linux environment. This assignment will be graded automatically using an auto-grader so please make sure you follow all the input and output definitions strictly. The auto-grader will try to compile your source code and check the output of your programs with provided datasets. We'll also manually check your code; plagiarism will not be tolerated. Detailed project organization guidance can be found at the end of the assignment.

The Big Picture

In this assignment, you will build a general purpose classification framework from scratch. You need to implement two classification methods: a basic method and an ensemble method. Given certain training dataset following a specific data format, your classification methods should be able to generate a classifier, and use this classifier to assign labels to unseen test data instances.

You will then test your classification framework with several real-world datasets using both the basic method and the ensemble method, and evaluate the quality of the classifiers with different metrics.

In order to build this classification framework, you need to finish four steps as follows.

- **Step 1**: Read in training dataset and test dataset, and store them in memory.
- **Step 2**: Implement Naïve Bayes method, which includes both training process and test process. Given a training dataset, your code should be able to construct a classifier correctly and efficiently. Given a test dataset, your code should be able to predict the labels of the unseen test instances using the learned classifier.
- **Step 3**: Implement AdaBoost using Naïve Bayes implemented in step 2 as the basic classification method. The ensemble classification method should also be able to train a classifier and predict labels for unseen data instances.
- **Step 4**: Test both Naïve Bayes and AdaBoost on provided datasets and write a report.

Step 1: Data I/O and Data Format (20 pts)

The classification framework should be able to parse training and test files, load these datasets into memory. We use the popular LIBSVM format in this assignment.

For training and test data file, each line contains an instance, which takes the following format:

<label> <index1>:<value1> <index2>:<value2> …

<label> is an integer indicating the class label (we only consider two class classification in this assignment). Each pair <index>:<value> provides an attribute-value pair: <index> is an integer starting from 1 indicating the index of the attribute and <value> is an integer (we only consider categorical attributes in this assignment). LIBSVM format uses sparse representation. For a data instance (one line), if the value of an attribute is 0, this attribute

is omitted in this line. It does not mean this attribute is missing. For a particular dataset, you can use the largest index in **both** training and test file as the number of attributes.

You need to be careful about **not** using label information as an attribute when you build classifier, in which case you can get 100% precision easily, but you will get 0 pts for Step 2 and Step 3.

Step 2: Implement Naïve Bayes (20 pts)

In this step, you will implement the **Naïve Bayes** classifier, which should contain two steps, training (build a classifier) and test (assign labels to unseen data instances). Many existing classification or prediction packages have the feature to save the learned classifier as a file, so that users can load the classifier into the framework later and apply the same classifier on different test datasets asynchronously. However, in this assignment, we are going to skip this feature, and assume that the test process happens immediately after the training.

Before you begin with your implementation, please first read the Project Organization and Submission section at the end of the assignment to get a general idea on how to organize your project and how to name your programs so the auto-grading system can compile and execute your programs correctly.

The classification method you implement should be able to take both training file and test file as input parameters from command line, use training dataset to build a classifier and test dataset with labeling information to evaluate the quality of the classifier. For example, if you are implementing Naive Bayes using C/C++, your program should be able to finish the entire classification process when you type this command:

```
./NaiveBayes training_file test_file
```

If you use Java:

```
java classification.NaiveBayes training_file test_file
```

If you use Python:

```
python NaiveBayes.py training_file test_file
```

Your program should output 8 numbers (4 number per line, separated by space) to roughly represent the quality of the classifier with both training and test datasets. These 8 integers are true positive in training, false negative in training, false positive in training and true negative in training and true positive in test, false negative in test, false positive in test and true negative in test. And you will need these 8 values in Step 4.

An example output can be found as follows:

```
23 25 12 36

20 30 16 23
```

In this case, true positive in training dataset is 23, while true positive in test dataset is 20. False positive in training is 12 and false positive in test dataset is 16.

Step 3: Implement Adaboost (40 pts)

In this step, you will implement Adaboost using the Naïve Bayes implemented in Step 2 as the basic classification method. Very similarly, your ensemble classification method should take training_file and test_file as input, and output 8 values in two lines, which are true positive, false negative, false positive and true negative in both training dataset and test dataset so that you can use these numbers to evaluate classifier quality in Step 4. For example, you are using C/C++, and the command line of running your program should look like this:

```
./NBAdaBoost train_file test_file
```

If you use Java:

```
java classification.NBAdaBoost train_file test_file
```

If you use Python:

```
python NBAdaBoost.py training_file test_file
```

Step 4: Model Evaluation and Report (20 pts)

In this step, you will apply both methods you implemented in Step 2 and Step 3 on four real world datasets. Notice that, we preprocessed these datasets and partitioned them into training and test for you (you can find the datasets on the course webpage), so please do not use the original datasets directly.

Name	Num of Attributes	Labels
adult	123	+1 is >50K, -1 is <=50K
breast_cancer	9	+1 recurrence-event, -1 no-recurrence-event
led	7	+1 is 3, -1 is other numbers
poker	10	+1 one pair. -1 three of a kind

If you are curious about the related domain application, and the semantic meaning of each attribute or labels, please read the source links associated with each dataset, but you do not need to know these details in terms of finishing this assignment.

You need to apply both Naïve Bayes and Adaboost on each dataset, and calculate the following model evaluation metrics using the output of your classification methods for both training and test datasets. Please do this manually or write a separate program (you do not need to turn this in), by taking the 8 numbers as input. Do not output these metrics from your classifiers directly since the auto-grader won't be able to recognize them.

Accuracy, Error Rate, Sensitivity, Specificity, Precision, F-1 Score, F_β score ($\beta =$ 0.5 and 2)

Now you're ready to write your report. Please include the following sections in your report:

- brief introduction of the classification methods in your classification framework;
- all model evaluation measures you calculated above (8 metrics * 2 methods * 4 datasets * 2 (training and test));

- parameters you chose during implementation and discuss why;
- your conclusion on whether the ensemble method improves the performance of the basic classification method, why or why not;

Project Organization and Submission

The auto-grading process will be performed on a Linux server, so make sure your source code can be compiled in the EWS Linux environment (or in major distro like Ubantu, Mint, Fedora or Arch with a relatively new version of gcc/g++ or JDK).

If you use C/C++

You must attach a Makefile so that the grader can compile your programs correctly. If you are a Windows user and use Visual Studio as your IDE, please add a Makefile after you finish implementation and make sure your code can be compiled properly using gcc/g++. Your project structure should look like this:

```
 yournetid_assign4/

  |----------------report.pdf
              |----------------code/
                                   |----------------Makefile
                                   |----------------*.c/c++
```

After compilation, your Makefile should be able to generate binary targets with corresponding classification method names, i.e., NaiveBayes and NBAdaBoost. Also, as we discussed in Step 2 and 3, the input arguments for your binary should look like this: ./NaiveBayes train_file test_file

If you use Java

Very similarly, please make sure your code can be compiled in Linux environment. Your entrance class should be placed under classification package, and your project structure should look like this:

```
yournetid_assign4/
        |----------------report.pdf
        |----------------code/
                    |----------------other_packages/
                                |----------------*.java
                    |----------------classification/
                                |----------------*.java
```

The grader will simply execute javac classification/*.java to compile your programs, and your program should be able to generate binary files with corresponding classification method names. The input arguments are defined as follows:

```
java classification.NaiveBayes train_file test_file
```

If you use Python2

Very similarly, please make sure your code can be compiled in Linux environment. Your project structure should look like this:

```
yournetid_assign4/
        |----------------report.pdf
        |----------------code/
                   |----------------*.py
```

The grader will run the following command to test your programs:

```
python NaiveBayes.py train_file test_file
```

After your finish your implementation and report, please compress your project into zip format (use the command “zip -r yournetid_assign4.zip yournetid_assign4”), where “yournetid” should be your netid.

Double check before you submit:

Your programs should be able to handle both absolute paths and relative paths to different training and test files. Do not assume that the files are located in the same folder of your programs.

In additional to the four provided datasets, we do have some other datasets, which are roughly the same scale of the 4 datasets that we provided here. Make sure that your programs can finish within 3 minutes for each dataset. The auto-grader will shut-down your program after 3 mins if no results are detected (and you will receive 0 points for the corresponding step).

Whenever we use auto-grader for programming assignment, some submissions won't pass the auto-grader, mostly because these submissions do not follow the project organization rules we stated above strictly. In this case, TAs will manually grade these assignments but we will take 10 to 20 points off your assignment if this happens. Some common mistakes include when using Java, didn't use package structure, named your program a wrong name, incorrect zip file name or folder name after unzip, etc.

Now you can submit your assignment through Compass 2g !!