LING 572 Hw2
Due: 11pm on Jan 18, 2022

For all the assignments in ling572:

- If your code is slow, do not run your code on patas directly. Use "condor submit" instead (see Q5 below).
- If patas is down, you can try dryas (e.g., ssh userid@dryas.ling.washington.edu).
- You can assume that all the class labels in the test data have appeared in the training data; that is, the test data does not contain any new class labels.

The example files for hw2 are under /dropbox/21-22/572/hw2/examples/.

Q1 (4 points): Run the Mallet DT learner (i.e., the trainer's name is DecisionTree) with **train.vectors.txt** as the training data and **test.vectors.txt** as the test data. In your note file, write down the following:

(a) The command lines you use for preparing data, training, testing, and getting the training and test accuracy. You can use vectors2classify commands to do training, testing and evaluation in one step.

(b) What are the training accuracy and the test accuracy?

Q2 (6 points): Run the Mallet DT trainer with different depths; that is, when running vectors2classify, replace **–trainer DecisionTree** with

```
--trainer "new DecisionTreeTrainer(nn)"
```

where nn is the depth of the decision tree. Note that you have to use vectors2classify, instead of "mallet train-classifier" and "mallet classify-svmlight" because "mallet train-classifier" does not process "new DecisionTreeTrainer(nn)" properly.

(a) Fill out Table 1

(b) What conclusion can you draw from Table 1?

Q3 (55 points): Write a program, **build_dt.sh**, that builds a DT tree from the training data, classifies the training and test data, and calculates the accuracy.

- This DT learner should treat all features as binary; that is, the feature is considered present if its value is nonzero, and absent if its value is zero.

Table 1: Run Mallet's DT learner with different depths

Depth	Training accuracy	Test accuracy
1		
2		
4		
10		
20		
50		

- Use information gain to select features when building DT.
- The format of the command line would be: build_dt.sh training_data test_data max_depth min_gain model_file sys_output > acc_file
- training_data and test_data are the vector files in the text format (cf. **train.vectors.txt**).
- max_depth is the maximum depth of the DT,[1] and min_gain is the minimal gain. Those parameters are used to determine when to stop building DT; that is, split the current training data set at the node x if and only if **(the depth of x < max_depth) AND (the infoGain of the split ≥ min_gain)**.
- model_file is the DT tree (cf. **model_ex**) produced by the DT trainer. Each line corresponds to a leaf node in the DT and it has the format: path training_instance_num c1 p1 c2 p2 ...
  Where path is the path from the root to the leaf node, training_instance_num is the number of the training examples that "reach" the leaf node, c_i is the class label, and p_i is the probability of c_i (i.e., the percentage of the training examples at the leaf node with the label c_i).
- sys_output is the classification result on the training and test data (cf. **sys_ex**). Each line has the following format:
  instanceName c1 p1 c2 p2 ..., where instanceName is just something like "array:0", "array:1", where "array:0" is the dummy name of the first instance in the training or test data.
- In both model_file and sys_output files, (c_i, p_i) **pairs should be sorted by the spelling of** c_i **alphabetically.** For instance, if the labels are guns, mideast, and misc, the line in sys_output should be instanceName guns p1 mideast p2 misc p3.
- acc_file shows the confusion matrix and the accuracy for the training and the test data (cf. **acc_ex**). In the confusion matrix, a[i][j] is the number of instances where the truth is class i, and the system output is class j.
- As always, model_ex, sys_ex, and acc_ex in the examples/ directory are NOT gold standard. These files were created just to show you the format of the files.
- For your reference, example_output/ shows the first 10 lines of the sys_output file if you build DT using train.vectors.txt and test.vectors.txt under examples/ with the options max_depth=4 and min_gain=0.1 as specified in the readme.txt file.

Run **build_dt.sh** with **train.vectors.txt** as the training data and **test.vectors.txt** as the test data:

[1] The depth of the root is 0, the depth of its children is 1, and so on.

- Fill out Table 2 (where min_gain is set to 0) and Table 3 (where min_gain is set to 0.1).
- submit model_file, sys_output, acc_file produced by running
  build_dt.sh train.vectors.txt test.vectors.txt 4 0.1 model_file sys_output > acc_file

Table 2: Your decision tree results when min_gain=0

Depth	Training accuracy	Test accuracy	CPU time (in minutes)
1			
2			
4			
10			
20			
50			

Table 3: Your decision tree results when min_gain=0.1

Depth	Training accuracy	Test accuracy	CPU time (in minutes)
1			
2			
4			
10			
20			
50			

Q4 (5 points): Slide #12 of class2_DT.pdf shows a DT: f1 and f2 are two features; f1 is in [-20, 30]; f2 is in [-10, 30]. L_i (i=1, ..., 7) represents a leaf node. Each leaf node corresponds to a rectangle in a 2-dimension space, where f1 is the x-axis and f2 is the y-axis. Draw a graph that shows the boundary of the seven rectangles in this 2-dimension space.

Q5 (5 "free" points): If you are not familiar with Patas or Condor submit, please go over the condor information at https://wiki.ling.washington.edu/bin/view.cgi/Main/HowToUseCondor and https://wiki.ling.washington.edu/bin/view.cgi/Main/CondorClusterHomepage. You can run condor submit for the code in Q3. We will use condor submit for many assignments later.

Submission: Submit the following to Canvas:

- Your note file *readme.(txt | pdf)* that includes your answers to Q1-Q4, and any notes that you want the TA to read.

- hw.tar.gz that includes all the files specified in dropbox/21-22/572/hw2/submit-file-list, plus any source code (and binary code) used by the shell scripts.
- Make sure that you run **check_hw2.sh** before submitting your hw.tar.gz.
- No need to submit anything for Q5.