CS 4786 Spring 2015 - Competition 2

Prefatory note Competition 2, "you said what"?!

Due date The due date is May 11th, 4:30 PM on CMS. Submit what you have at least once by an hour before that deadline, even if you haven't quite added all the finishing touches — CMS allows resubmissions up to, but not after, the deadline. If there is an emergency such that you need an extension, contact the professors.

How to form groups for this competition You may work in groups of one up to four[1], where your group was formed by you (ASAP, and certainly by May 1st) . Please ensure that each member of the group can individually defend or explain your group's submission equally well.

> 1. Footnote: *The choice of the number "four" is intended to reflect the idea of allowing collaboration, but requiring that all group members be able to fit "all together at the whiteboard", and thus all be participating equally at all times. (Admittedly, it will be a tight squeeze around a laptop, but please try.)*

Collaboration and academic integrity policy

Students may discuss and exchange ideas with students not in their group, but only at the conceptual level.

We distinguish between "merely" violating the rules for a given assignment and violating academic integrity. To violate the latter is to commit fraud by claiming credit for someone else's work. For this assignment, an example of the former would be getting detailed feedback on your approach from person X who is not in your group but stating in your homework that X was the source of that particular answer. You would cross the line into fraud if you did not mention X. The worst-case outcome for the former is a grade penalty; the worst-case scenario in the latter is academic-integrity hearing procedures.

The way to avoid violating academic integrity is to always document any portions of work you submit that are due to or influenced by other sources, even if those sources weren't permitted by the rules.[2]

> *2. Footnote: We make an exception for sources that can be taken for granted in the instructional setting, namely, the course materials. To minimize documentation effort, we also do not expect you to credit the course staff for ideas you get from them, although it's nice to do so anyway.*

Data

Here is a link to a page where you can view "diffs" between any two versions: use the "compare selected versions" feature to highlight precisely what text was added or deleted.

Version	Published	Changed By	Comment
CURRENT (v. 9)	**May 19, 2015 08:31**	**Lillian Lee**	added reference t
v. 8	May 10, 2015 18:30	**Lillian Lee**	references to othe
v. 7	Apr 30, 2015 15:39	**Lillian Lee**	add requirement o
v. 6	Apr 30, 2015 15:33	**Lillian Lee**	minor: kaggle lea
v. 5	Apr 29, 2015 11:03	**Lillian Lee**	remove mention o
v. 4	Apr 28, 2015 22:10	**Lillian Lee**	update due date,
v. 3	Apr 28, 2015 21:55	**Karthik Sridharan**	
v. 2	Apr 28, 2015 15:25	**Karthik Sridharan**	
v. 1	Apr 28, 2015 13:30	**Lillian Lee**	

First, download the data zipfile and unzip it. The data provided to you this time is speech data (spoken words). Specifically, the data consists of people (both male and female) speaking one of a set of seven words. Your goal is, given the speech data for a particular word utterance identify the word being spoken. Mel-frequency Cepstral coefficients (MFCC) is a standard feature extraction technique for speech data. This has been performed for you and you are provided with the feature-extracted data-set. Specifically, each word occurrence is broken into windows of time unit and for every window of speech, there are 13 numbers representing the MFC coefficients of that window. Each utterance of a word is represented by a sequence of 83 windows (so overall 83 times 13 = 1079 numbers for each data point or word utterance). If the utterance was shorter than 83 windows, we simply padded the remainder with 0's so that every word utterance produces a vector of same size. We have provided a training set consisting of word utterances in file "train.data" in csv format where each line represents a particular word utterance by a speaker. The labels of which word was spoken is provided in file "train.labels". This file is in Kaggle format. That is, the first line of this file is "Id,Prediction" and every subsequent line has two comma separated values, the first indicating which line in the "train.data" file is being referred to and the second value indicates which of the seven words (from "0" to "6") was the actual spoken word. You can use this data to train a machine learning algorithm to automatically classify word utterances into one of the 7 words being said based on the feature extracted speech data.

Q1 (Challenge: identify what was said).

As for the task, you are also provided with another file "test.data" consisting of 1540 test utterances that you need to classify as being one of the seven words based on the machine learning model you trained based on the training sample. Your goal is to produce a file consisting of 1541 comma-separated lines, where, to conform to what Kaggle is expecting:

1. The first line should be "Id,Prediction".
2. Each subsequent line consists of two comma-separated numbers: the first number is which word utterance is being referred to, *starting with line **0***, and the second number is one of "0" to "6". So, the second line in your csv file (the one after "Id,Prediction" will be "0,0" ,...,or "0,6". You will name this file "test.csv". We will evaluate your grouping against the ground truth (which is of course hidden from you). Your goal is to make as few mistakes as possible.

Deliverables and instructions: Part of the competition is on Kaggle so do sign up and compete. ~~This time however we are planning to make the leader board private. So you will get to know your accuracy but not your rank. We will release statistics from leaderboard from time to time.~~ (We were not able to configure the leaderboard to be private-only.) At the end of the competition you will be required to submit one data file described in the question above *and* (as a zip file) the code for your best-performing submission, such that we could reproduce your results if necessary for grading. (In order to do so, we need to know the values of any parameters you set; please include these in your README and writeup.) But more importantly you need to also submit a writeup/report ("writeup.pdf") of the things you tried and why you tried them (irrespective of whether they worked or failed). The writeup will count for at least as much of the grade as the empirical results of the final labels you submit.

Here are a few other remarks:

0. We are declining to pre-specify too many regulations on your writeup because we want this competition to be somewhat open-ended, so with respect to format, our intent is just that you submit "something we find reasonable". To that end, we do require that

1. this report is at least 5 pages long and not more than 15 pages, but with font sizes, spacing, etc., we just expect you to do something reasonable. (See item 6 below.)

2. you include all of your names, netids, and the name of your Kaggle team at the beginning of your document

1. Include visualizations of both successful and unsuccessful trials.

2. Make a note of all successful and unsuccessful methods you tried. Explain why you made the choices you made and why you expected them to work both for successful and not-so-successful choices and take a shot at explaining why the less successful ones were in fact not so successful.

3. Organize your writeup into sections where each section (and its title) corresponds to a particular method.

4. You are certainly encouraged to try methods you might have picked up outside those covered in class and maybe even extensions you develop on your own for the problem!. If you use methods other than ones covered in class, **do compare the performance both empirically and conceptually with (a reasonable choice of) methods covered in class.**

5. We will definitely award karma points for work that goes above and beyond, and have made this competition somewhat open-ended to that end. Our definition of "work" emphasizes novelty and well-foundedness of methods tried, but also includes presentation aspects of your writeup, and we are hoping to see some striking visualizations (for instance, that demonstrate a clever analysis of an algorithm or its failings/successes.)

6. Your ranking on Kaggle will form a small part of your grade for this assignment. Much more importantly, you can (and, really, ought to) get feedback on how good your approaches are by getting them judged on Kaggle. The earlier you start, the more tries you get, since your group is limited to two Kaggle submissions a day! However unlike competition 1, you have a training sample provided to you. You can set aside some percentage of this data for validation of your algorithm or model without having to submit to kaggle everytime.

8. **Please check for new/edited drafts of this instruction document, as it will keep getting updated. (a)** We recommend you set things up so that you get emailed an update when this page pages: to do this, log in to confluence using your netid and netid password and then set a "watch" on this page by clicking the "Watch" option near the top right (it has an "eye" icon next to it). **(b)** Regardless of whether you set a watch or not, the right-hand side of this page shows when the page has been changed and, hopefully, a comment documentation what the changes were. Even better, you can see a diff of what changes have been made between different versions ---- a handy feature that is the main reason we switched to a wiki format for these instructions. To do so, click on the link indicated at the top of the right-hand side of this page. This will bring you to the "Page History" page. There, you can select checkboxes to select any two current or past versions of this page; then, click on "Compare selected versions".

9. The zipfile of code you submit needs to include a README.txt file that explains how we can run your code. Include in the README the exact values of any parameters you set to achieve your best result. The code should be "standalone" in the sense you should include any extra modules, libraries, etc. that you wouldn't expect our standard installs of R, Matlab, python, numpy, etc. to necessarily have.

10. Include in your writeup the values of any parameters you set for your best results. These values should also be in your README, as stated in point 9.

11. So that this competition exercises some of the graphical-models material we have covered, we **require** that you experiment with HMMs and describe your experiments in your writeup.

Matlab provides an HMM Toolbox. I've heard some recommendations of Kevin Murphy's Matlab/Octave code: general toolbox, older HMM toolbox (page also list some other HMM packages). Here is some discussion of Python libraries for HMMs. TA Jack Hessel's basic python HMM implementation uses hmmlearn and is under 100 lines long (with about twenty lines being devoted to a progress bar).

Since a progress bar might be of general interest, we show Jack's implementation here.

```
print "Running Classification"
 lb = LoadBar(30); lb.setup()
 with open(args.output, 'w') as f:
     f.write("Id,Prediction\n")
     for ind, t in enumerate(test):
         if lb.test(ind, len(test)): lb += 1
         curMaxInd = -1
         curMax = np.finfo('f').min
         for i in range(7):
             curProb = hmms[i].score(t)
             if curProb > curMax:
                  curMax = curProb
                  curMaxInd = i
         f.write(str(ind) + "," + str(curMaxInd) +
"\n")
 lb.clear()



#Fancy things for the loading bar
class LoadBar():
 def __init__(self, width):
 self.width = width

def setup(self):
 sys.stdout.write("[%s]" % (" " * self.width))
 sys.stdout.flush()
 sys.stdout.write("\b" * (self.width+1))

def __iadd__(self, amnt):
 sys.stdout.write("*"*amnt)
 sys.stdout.flush()
 return self

def test(self, i, length):
 try:
 return i!=0 and i%(length/self.width)==0
 except ZeroDivisionError:
 return True
 def clear(self):
 sys.stdout.write("\n")
```