A1

Numerical problems must show work to receive credit. Programs must be well structured and commented.

Part I (20 points each)

1. Read (Williams and Kessler, 2000) and answer the following short questions:
   a. Which of the following statements **is not supported** by the text:
      i. Many programmers are skeptical of pair programming.
      ii. Pair programming teams in some studies complete tasks 40% faster than solo programmers.
      iii. About 5% of pair programming teams fail without support.
      iv. Pair programmers tend to help each other stay focused and productive
   b. Which of the following statements **does not reflect** the ideas listed in the article about good practices for pair programming?
      i. Excess ego can prevent a programmer from considering other ideas.
      ii. Excess ego can translate to defensiveness when ideas are being critiqued.
      iii. Pair programmers should engage in healthy debate.
      iv. Getting used to pair programming takes time.
      v. Pair programming is most effective when both programmers have similar strengths
2. An audio file is sampled at 44.1 kHz. How many samples are in a 20 ms frame?
3. When using Bayes decision rule to decide a class, we frequently omit the probability of the observation. Does this shortcut affect the classification decision? Why or why not?
4. Intensity in decibels is often computed as $20 \log_{10}(p_{RMS})$ where p_{RMS} is the root-mean-square pressure. Why do multiply by 20?
5. Explain in your own words the difference between posterior and class-conditional probability.
6. We indicated in class that a signal that has an increase of 3 dB represents roughly a doubling of intensity. Prove this. Hint: Think about the definition of a decibel and what it means to add 3 dB to it: $10log_{10}(I) + 3$ and how we obtained the 3: $3 = 10 \log_{10}(I_D)$.

Part II (100 points)

Before starting this project, you should install Python 3 and several libraries. Once you have read Williams and Kessler (2000), you may elect to work in a pair programming team of two people for part II. Part I must be done individually.

Useful libraries for this assignment are:

- numpy – numerical library
- matplotlib – plotting library
- sklearn – scikit learn machine learning library
- pyaudio – Bindings to portaudio sound library, only needed if you want to do the optional interactive portion of the assignment

If you are using Anaconda (recommended, install miniconda if you want a smaller disk footprint), you can use the conda_env.yml file that can be downloaded from the Canvas assignments page to install these and Python easily. See comments in the file for how to do this from an Anaconda command line prompt. It will create an environment called cs682. Afterwards, you will need to activate the environment. See instructions if needed.

If you are not using Anaconda, feel free to install the packages manually using pip.

For this assignment, you will need two audio files. One is provided for you and is in the assets directory of the skeleton code that is provided to you. The second, me.wav, is one that you will create. Record me.wav in a relatively quiet setting, it should be no more than 5 to 10 s and should contain some speech and some silence/noise. If you need recording software, I recommend the lightweight and freely available cross-platform package wavesurfer developed at the Royal Inst. of Technology in Stockholm. For more capable audio clients, the freely available Audacity is fine as is Adobe Audition which is available through the campus Adobe site license. We have occasionally seen compatibility problems with audio recorded in Adobe.

Set your recording so that it only records on one channel; some devices have multiple microphones and record in stereo. Be sure not to speak too loudly into the microphone. When you do this, the pressure (and voltage) is greater than the extrema values that the are permitted by the quantization strategy. This is called clipping and creates serious problems for processing audio. Here's an example of a recording that was clipped about halfway through. Notice how there appears to be a "flat" top and bottom:

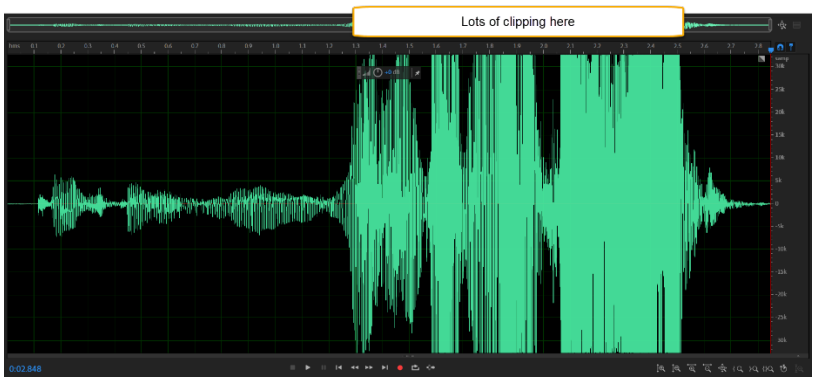

Figure 1 – Clipped speech from a signal that overdrove a microphone.

You will need to modify the given code to create a working program. You must conform to the interfaces. Skeleton code is available on Canvas on the assignments page and provides details on the arguments and functionality.

The program entry point is in driver.py. It should read in the audio (use the provided read_wav in audio_io.py[1]), create a Framer object (see framing.py) responsible for framing the signal. The Framer instance should be passed to the RMS class whose get_energy function returns the root-mean-square energy in dB.

A 2 mixture GMM should be trained from the RMS energy. Determine which class is speech and which is noise. Compute the mixture label for each frame of speech's RMS energy.

At the end, it should create plots with a time axis in seconds and an RMS axis (dB rel. 1) showing the intensity. Add to the plot an indication of where your classifier decided that speech vs. noise occurred, similar to what was done in the plots shown in class.

Try this on your own recording and on the audio file in the assets directory.

You can use any type of plotting software you wish. The matplotlib package is relatively simple to use and has the advantage that I am likely to be able to help you with questions about plots. Other alternatives include seaborn and plotly.

classifer.py – This is your machine learning module. It will contain a single class Endpointer:

- The constructor takes an intensity signal and trains a Gaussian mixture model with two mixtures (use scikit learn's GMM implementation). The GMM will not decide which mixture represents noise and which represents speech, but you can

[1] The lib.audio_io module also contains Recorder, a class for recording directly from the microphone. You do not need to use Recorder for this assignment.

examine the mixture means (attribute means_ of your model) to determine which mixture represents noise (think about how we did this in class).

- Endpointer's predict method takes an intensity signal and returns a Boolean vector that is set to True wherever the model predicts speech. While you could compute a static threshold, it is easier to use the GMM's predict_proba which gives the likelihood of observation for each mixture component, allowing you to determine which mixture has the higher class-conditional likelihood for an observation.

If you are using PyCharm, you may wish to undock your plots as the plots in sciview tend to be small. When using Matplotlib, I like to make my plots interactive, the driver skeleton code has a line that turns this on.

What to turn in:

- Parts I and II should be submitted separately to Canvas.
- Part I documents will be accepted as Word or PDF.
- Part II
    - Be sure to comment your programs, comments account for 10% of the program points.
    - If you are pair programming, make sure to add both programmers to the submission (the submission program has facilities for doing this)
    - Program files are to be uploaded individually or as a zip file. Be sure to preserve directory structure. Top-level files should be at the top level of your zip file, not within a subdirectory.
    - All submission must have an affidavit. See submitting work for details and guidelines on commenting.
    - Figures can be submitted as standard graphic files (e.g. png, jpg) separately or as a pdf. When submitting zip archives, just include them.

References

Williams, L. A. and Kessler, R. R. (2000). All I really need to know about pair programming I learned in kindergarten. *Comm. ACM* **43**, 108-114.