Cameron Embree
Math 448, S2014

Simple Sound Identification via *Mathematica*

1. INTRODUCION

Sound identification and classification are increasingly integrated into technologies as hands-free user interfaces are developed. Our phones can now more accurately provide us with information from verbal queries. When requesting something of a device, it records our voice, parses our voice into words, and determines our question from these words. I am interested in learning more about how sound identification works including what measurements can be used from audio recordings that characterize recordings in a way that it can be used to classify other similar sounding audio. For example, after measuring a person saying "one" multiple times, by using some measurement of the audio recordings, we seek to identify later utterances of the word "one."

Mathematica offers tools for audio importing and manipulation. Although not as robust as other audio analysis frameworks, *Mathematica* can take Fourier Transforms of an imported audio clip and this alone has promise for making classifications. We will test how effective using a Fourier Transform to classify an audio segment might be. To make a conclusion about the ability for just a Fourier Transform to classify an audio segment, we tested its ability to classify a sample word set of six words. The results of these tests may help determine if using Fourier Transform of previously sampled audio can help make future audio classifications.

2. BACKGROUND

The Fourier Transform is a transformation that takes a signal from the time/spatial domain to the frequency domain. Essentially, a Fourier Transform of an audio sample will provide the frequencies that exist in that sample. Therefore, we can use this transform to see the different frequencies that exists in pronunciations of various words. Different words may have differing amount of various frequencies that could be used to recognize a new recording as being very similar to, as thus classified as, a previously analyzed recording.

3. METHODOLOGY

A word bank of 6 "simple" words with multiple recordings is used for testing. The word bank is the six numbers in English from zero to five.

0	1	2	3	4	5	6
zero	one	two	three	four	five	six

Each of word was recorded 6 times so that testing could be handled in a NxN matrix in *Mathematica*, where N=number of words in the word bank. Multiple recordings of a single word were averaged together by averaging respective Fourier Transform and could then provided an approximate of the Fourier Transform for a given word in the word bank. To only collect information that measures the Fourier Transforms that humans can hear, a high pass filter was applied at 10,000Htz. This

filter was applied under the assumption that the only data that should characterize our audio sample are frequencies that can actually be heard.

After average Fourier Transforms were generated for each of the words in the bank, each of the originally sample of words was compared to the averages based on the differences in Fourier Transforms. Differences are measured by generating functions that best-fit the Fourier data and then integrating to find the difference between that average best fit and the new audio samples' best-fit. The average Fourier best fit that is least different than the new sample, the lowest absolute value integration difference, is what the new sample is classified as. In other words, an audio sample should have a similar best-fit polynomial to it's Fourier Transform as the average best-fit of the Fourier Transforms from which it is generated.

4. RESULTS

Using just the average Fourier Transform of the word bank, accuracy of just over 86% was achieved.

4.1 Classification Results

The second element of each position in the following matrix is what word that a particular piece of audio was classified as being. The first element of each position represents the time in seconds it took to perform the classification operation.

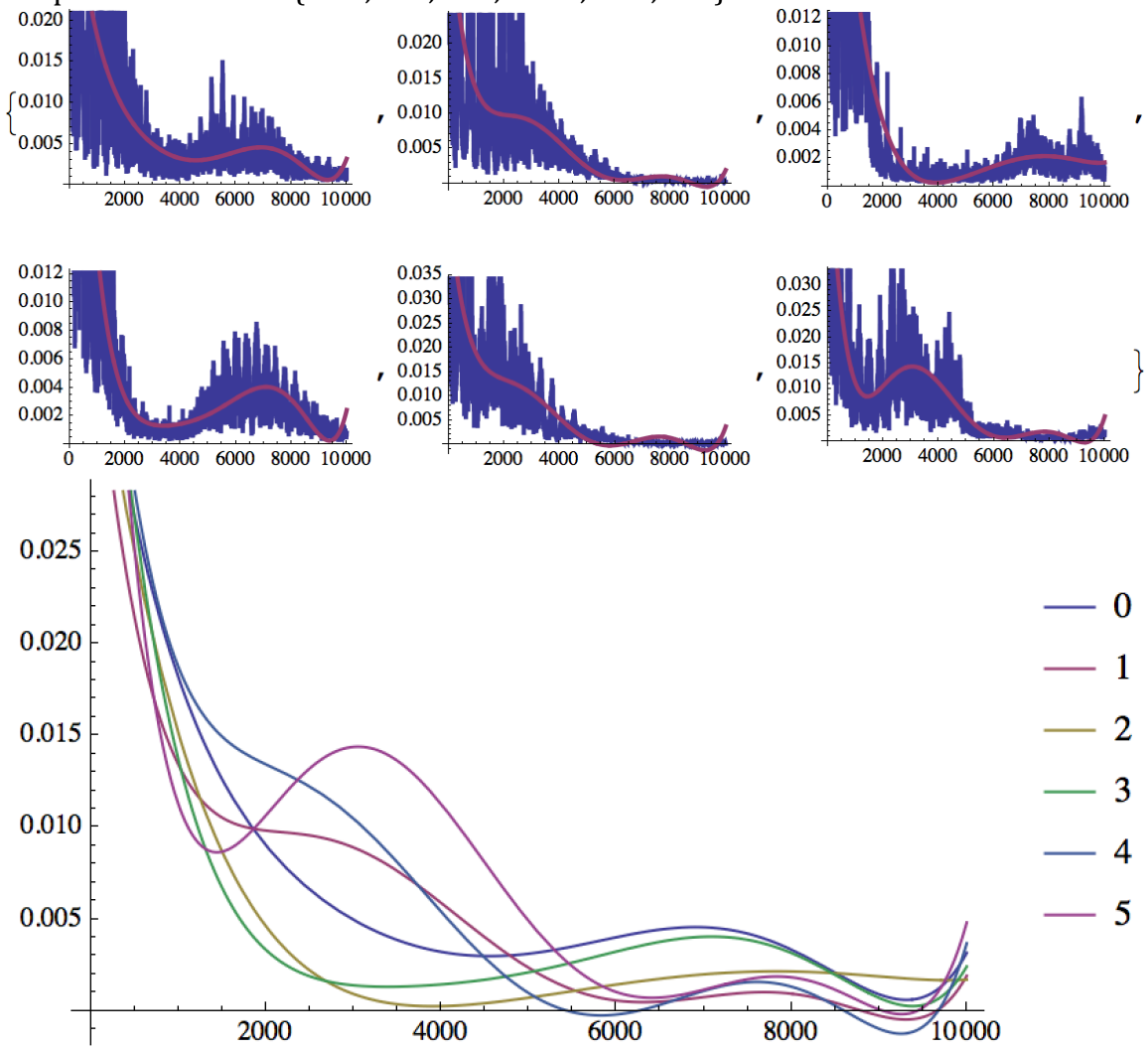
$$\begin{pmatrix} \begin{pmatrix} 0.000055 \\ 0 \end{pmatrix} & \begin{pmatrix} 0.000032 \\ 0 \end{pmatrix} & \begin{pmatrix} 0.000030 \\ 0 \end{pmatrix} & \begin{pmatrix} 0.000028 \\ 0 \end{pmatrix} & \begin{pmatrix} 0.000029 \\ 2 \end{pmatrix} & \begin{pmatrix} 0.000028 \\ 1 \end{pmatrix} \\ \begin{pmatrix} 0.000029 \\ 1 \end{pmatrix} & \begin{pmatrix} 0.000028 \\ 1 \end{pmatrix} & \begin{pmatrix} 0.000028 \\ 1 \end{pmatrix} & \begin{pmatrix} 0.000027 \\ 1 \end{pmatrix} & \begin{pmatrix} 0.000028 \\ 1 \end{pmatrix} & \begin{pmatrix} 0.000027 \\ 1 \end{pmatrix} \\ \begin{pmatrix} 0.000026 \\ 2 \end{pmatrix} & \begin{pmatrix} 0.000027 \\ 3 \end{pmatrix} & \begin{pmatrix} 0.000028 \\ 2 \end{pmatrix} & \begin{pmatrix} 0.000028 \\ 2 \end{pmatrix} & \begin{pmatrix} 0.000028 \\ 2 \end{pmatrix} & \begin{pmatrix} 0.000028 \\ 2 \end{pmatrix} \\ \begin{pmatrix} 0.000028 \\ 3 \end{pmatrix} & \begin{pmatrix} 0.000027 \\ 3 \end{pmatrix} & \begin{pmatrix} 0.000028 \\ 3 \end{pmatrix} & \begin{pmatrix} 0.000029 \\ 3 \end{pmatrix} & \begin{pmatrix} 0.000028 \\ 3 \end{pmatrix} & \begin{pmatrix} 0.000027 \\ 3 \end{pmatrix} \\ \begin{pmatrix} 0.000028 \\ 4 \end{pmatrix} & \begin{pmatrix} 0.000028 \\ 4 \end{pmatrix} & \begin{pmatrix} 0.000028 \\ 4 \end{pmatrix} & \begin{pmatrix} 0.000028 \\ 1 \end{pmatrix} & \begin{pmatrix} 0.000032 \\ 4 \end{pmatrix} & \begin{pmatrix} 0.000032 \\ 4 \end{pmatrix} \\ \begin{pmatrix} 0.000029 \\ 1 \end{pmatrix} & \begin{pmatrix} 0.000028 \\ 5 \end{pmatrix} & \begin{pmatrix} 0.000027 \\ 5 \end{pmatrix} & \begin{pmatrix} 0.000027 \\ 5 \end{pmatrix} & \begin{pmatrix} 0.000028 \\ 5 \end{pmatrix} & \begin{pmatrix} 0.000028 \\ 5 \end{pmatrix} \end{pmatrix}$$

100% accuracy would have been a matrix of the approximate form

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 & 5 \end{pmatrix}$$

4.2 Average Best-fit Functions for Fourier Transforms

The following are the overlay of each best-fit to average Fourier Transform for each word as well as the average Fourier transform with each best fit function overlaid. Output is in the form {zero, one, two, three, four, five}.



5. REFLECTION

The best fit polynomial performed most optimally when using the function f(x)=a*x^1+b*x^2+c*x^3+d*x^4+e*x^5+f*x^6, where a,…,f are real numbers. Various scalar combinations of periodic functions were tested to attempt to provide a more accurate fit to the original Fourier data. Unfortunately, attempts to make an improved fit were not successful and results of one such best-fit attempt can be seen in the **References** section. The basic issue was that these more accurate functions may have provided quality representations of the averaged Fourier Data but they were sensitive to small perturbations in the FTs. Therefore, single sound samples analyzed did not match correctly because they were best fitted poorly because of the sensitivity. Essentially, f(x) defined above was used because it was less sensitive to perturbations in single recordings experimentally matched better.

A better performance than 80% may be achieved if more than just the FT of audio data was being compared. Using multiple measurements and finding similarities across all measurements may perform better because no single measurement performing poorly would overrule the entire classification. Additionally, the current use of only the Fourier Transform is sensitive to background noise and poor quality of microphones introducing noise. A dataset much larger than the 36 samples currently used to define 6 words would likely have much poorer performance. Introduction of noise in a single recording might begin to become more significant than the small differences defining each of the many different word's Fourier Transform best fits. Even if the system performed less accurately, however, this process might still be a quick way to prune out candidates for classification quickly.

6. REFERENCES

The following are the results of best-fit operations using the following periodic model with scalar multiples {aa,bb,…,tt}.

```
model = aa * Cos[x / 500] + bb * Cos[x / 1000] + cc * Cos[x / 1500] + dd * Cos[x / 2000] +
  ee * Cos[x / 2500] + ff * Cos[x / 3000] + gg * Cos[x / 3500] + hh * Cos[x / 4000] +
  ii * Cos[x / 4500] + jj * Cos[x / 5000] + kk * Cos[x / 5500] + ll * Cos[x / 6000] +
  mm * Cos[x / 6500] + nn * Cos[x / 7000] + oo * Cos[x / 7500] + pp * Cos[x / 8000] +
  qq * Cos[x / 8500] + rr * Cos[x / 9000] + ss * Cos[x / 9500] + tt * Cos[x / 10 000];
```