Notes on MFCCs

\*These notes are an accumulation of multiple articles, as well as the YouTube audio processing series by Velario Velardo

What are MFCCs?

* MFCCs are a feature that can be extracted from audio. MFCC stands for Mel-Frequency Cepstrum Coefficient.
* MFCC coefficients contain information about the rate of change in different spectral bands
* **If the MFCCs have a positive value then most spectral energy is present at low frequencies. [3]**
* **If the MFCCs have a negative value, most spectral energies are concentrated at high frequencies. [3]**
  + Side note (from Wikipedia of Spectral Energy) : Spectral energy is the energy of a signal or a time series distributed with frequency

[What is a power spectrum? - YouTube](https://www.youtube.com/watch?v=Gka11q5VfFI)

Breaking down the meaning of each part of Mel-frequency Cepstral Coefficients:

* Mel-frequency
  + Refers to the mel spectrogram
  + Has to do w/ pitch
  + The mel scale relates to the differences between notes, as perceived by the human ear.
* Coefficient
  + In this context, we are referring to a coefficient about a piece of sound
* Cepstral
  + Cepstrum (noun), related to spectrum (if we take out ‘ceps’ and write the letters backwards, then add it back onto ‘strum’, we get the word ‘spectrum’).
    - We also have the words quefrency, related to ‘frequency’, ‘liftering’, related to ‘filtering’, and rhamonic, related to ‘harmonic’.
  + The cepstrum can be computed with the following equation:
    - C(x(t)) = F-1[log(F[x(t)]]
      * x(t) is the time domain signal (normal waveform)
      * F[(x(t)] is the Discrete Fourier Transform, which is used to come up with a spectrum.
        + This moves from the time domain (the original signal) to the frequency domain).
      * log(F[x(t)]) applies the log to the spectrum to get the **log amplitude spectrum.**
        + (We used the log amplitude spectrum because it allows us to separate E(t), which in the case of speech, is the spectrum from the Glottal Pulse, from H(t), which in the case of speech is the spectrum from the vocal tract frequency response.)
        + Also note that we do not perceive amplitude linearly. We perceive amplitude logarithmically.
        + Log(X(t)) = log(E(t) \* H(t))
        + Log(X(t)) = log(E(t)) + log(H(t))
      * F-1[log(F[x(t)]] applies an Inverse Fourier Transform to the log amplitude spectrum, to get the cepstrum.
        + Getting the cepstrum is calculating a spectrum of a spectrum. We are treating the log amplitude spectrum as if it is the time domain. If we were in the time domain, performing a Fourier Transform would bring us to the frequency domain. However, we are not in the time domain; we have a spectrum. This is why doing the Inverse Fourier Transform calculates a spectrum of a spectrum, called a cepstrum.
        + The cepstrum provides us a natural physical separation of the information related to the spectral envelope (vocal tract frequency response) from the glottal pulse related information. [4]
        + We want to only focus on the features relative to the spectral envelope. We use a low pass lifter (like a low pass filter) which removes the information related to the high quefrencies (glottal pulse, which we don’t want). So now we just have the information related to the spectral envelope!

How do we calculate MFCCs?

* The following steps get the MFCC (found in works cited 3):
  + Take the Fourier Transform of signal
  + Map the power to the mel-scale using triangular overlapping windows or cosine overlapping windows (sort of like autotuning – plotting it on a musical scale; mapping to different notes!) (aka mel scaling)
  + The logs of the powers at each of the mel frequencies
  + Take DCT – Discrete cosine transform of the mel log powers
  + **The MFCCs are the amplitudes of the resulting spectrum** (note: this is in frequency domain)
* The video (works cited 4) shows these steps in a slightly different order\*\*
  + Start with the waveform (audio signal)
  + Perform a Discrete Fourier Transform (DFT) of the signal
  + Convert to the Log-Amplitude Spectrum
  + Mel-Scaling
    - We are going from a linear representation to a mel-based representation
  + Discrete Cosine Transform
    - Similar to applying Inverse Fourier Transform to get to the cepstrum (but there are reasons to use DCT for MFCCs!)
      * DCT is a simplified version of Fourier Transform
      * DCT gives us back real-valued coefficient, whereas a Fourier Transform returns complex numbers, which we don’t need for calculating MFCCs.
    - We get information about the different values that construct the formans, timbre, basic information about the spectrum to understand speech (fornyms, etc.)
    - Try to fit cosines to log spectrum
    - Decorrelates energy in different mel bands
      * **Good b/c w/ ML algos, we want features that have as little correlation as possible**
    - Reduces the number of dimensions we use to represent the spectrum. DCT takes input (log spectrum) and provides us back with a set of features that has less dimensions (also good for ML)
  + Now we have MFCCs!
* Note that the first three steps above (Waveform, DFT, and converting to the log-amplitude spectrum) are the same steps we used in converting to the cepstrum domain.
* The MFCCs for all of the audio samples should have the format below. [2] This format contains one outer array, with a lot of smaller arrays within it. The smaller arrays contain the MFCCs from the smaller segments within the audio signal.

A screenshot of a computer code

Description automatically generated

* Fourier transforms
  + Fourier transforms are an important part of calculating MFCCs. A Fourier transform brings the signal from the time domain into the frequency domain. The product after the Fourier transform is called a spectrum.
  + We use a Fourier Transform to understand where the shape of the peaks come from. The Fourier Transform describes the frequencies. The Fourier Transform takes you from a wave which might be made up of a bunch of noise and decomposes into a weighted sum of sine waves. [2]
  + Magnitude Fourier transform is getting us which frequencies contributed most to the signal. [2]

Number of MFCCs

* The hyperparameter n\_mfcc in Librosa refers to the number of MFCCs **per segment** of the initial audio signal. So, if you specify 20 as the number for n\_mfcc, and you have 100 25 milisecond segments in your audio, you will have 20 x 100 MFCCs. (I am not 100% sure if this line is correct since I know that sometimes there is a gap in between which segments MFCCs are calculated on. I will double check and update)
* **MFCC[0]** is the first element in the vector obtained after the Discrete Cosine Transform captures the **spectral energy across the filterbank**, for each short-time frame. [1]
* The first coefficients have the most relevant information (formants, spectral envelope)
  + Similar to w/ the cepstrum. The quefrency values on the lower end provide info about the spectral envelope. Quefrency values on the higher end provide info about the glottal pulse
  + Note: “Formants are distinctive frequency components of the acoustic signal produced by speech, musical instruments or singing” [7]
  + Higher coefficients provide info about fast-changing spectral details, which we don’t need for speech recognition (would we want this for other sound detection??)
* So, more MFCCs won’t help
* Instead use delta and delta delta MFCCs
  + MFCC values in one frame and subtract from previous frame
  + Like taking the first and second derivatives of the MFCCs

Why do we use MFCCs?

* Goal is to represent the data points with fewer numbers (compact representation) while preserving the essential features of the signal. [1]
* MFCCs are often used in speech processing.
* Help to cut out noise
* #1 choice for speech and music processing

Future Possibilities to Consider:

* Multi-channel MFFCs may improving accuracy:

[[2112.14930] Feature extraction with mel scale separation method on noise audio recordings (arxiv.org)](https://arxiv.org/abs/2112.14930)

[Channels in MFCC, and how to use them - audio - PyTorch Forums](https://discuss.pytorch.org/t/channels-in-mfcc-and-how-to-use-them/59498)

**Notes on Other Audio Features Discussed in Videos by Velario Velardo**

Video “Types of Audio Feature for Machine Learning”

* Signal domain we’re in
  + Certain audio features are in the **time domain** (extracted from a waveform)
    - Amplitude envelope
    - Root-mean square energy
    - Zero crossing rate
  + Waveform
    - X-axis is time
    - Y-axis is amplitude
  + We can look at all of the events in the sound
  + Sound is categorized by frequency (but we don’t have frequency in the time domain)
  + Frequency domain
    - Fourier Transform transforms the signal from the time domain to the frequency domain, creating a spectrum
    - We can then use a spectrogram, to see the frequency components at different points in time. The amount each frequency band contributes is represented by a color (brighter color = more contribution)

**Notes on other audio features (while writing code, what I’m learning along the way)**

Autocorrelation coefficient:

* Helpful visualization: [A Visualization of the Autocorrelation Function (youtube.com)](https://www.youtube.com/watch?v=uf679Qo-bB4)
* How two values in the same series relate/correlate to each other
* Involves using the lags parameter. A lag is how many jumps there are between one segment of the audio to the other. For example, if we are comparing the first sample to itself, the lag will be 0, and the autocorrelation coefficient will be a high number, because there should be no difference between the first sample and itself. Thus, they are highly correlated
* If the correlation coefficient decreases, then increases, the way may be periodic, as this shows that the pattern may start to repeat
* acc = sm.tsa.acf(x, nlags=2000)

Zero crossing rate:

* Librosa.zero\_crossings(x)
  + “Computes a binary mask where True indicates the presence of a zero crossing” [8] (and false indicates there is not a zero crossing)
  + Sum(output\_of\_zero\_crossings) provides the total number of zero crossings (whole number)
* Librosa.feature.zero\_crossing\_rate(x)
  + Returns the fraction of zero crossings in frame I [9]

Works Cited

[1] [What, how, and why of MFCCs | COSWARA (iiscleap.github.io)](https://iiscleap.github.io/coswara-blog/coswara/tutorial/2020/08/20/mfcc.html)

[2] [Lecture8-FourierTransforms.pdf (harvard.edu)](https://scholar.harvard.edu/files/schwartz/files/lecture8-fouriertransforms.pdf)

[3] [How To Generate MFCC From Audio. — ML For Lazy 2021 | by M Shehzen | Analytics Vidhya | Medium](https://medium.com/analytics-vidhya/how-to-generate-mfcc-from-audio-ml-for-lazy-2021-42c2fdfa208)

[4] [Mel-Frequency Cepstral Coefficients Explained Easily (youtube.com)](https://www.youtube.com/watch?v=4_SH2nfbQZ8&t=2s)

[5] [Types of Audio Features for Machine Learning (youtube.com)](https://www.youtube.com/watch?v=ZZ9u1vUtcIA)

[6] [Understanding Time Domain Audio Features (youtube.com)](https://www.youtube.com/watch?v=SRrQ_v-OOSg)

[7] [Formant - Wikipedia](https://en.wikipedia.org/wiki/Formant)

[8] [zcr (musicinformationretrieval.com)](https://musicinformationretrieval.com/zcr.html)

[9] [librosa.feature.zero\_crossing\_rate — librosa 0.10.1 documentation](https://librosa.org/doc/latest/generated/librosa.feature.zero_crossing_rate.html)