11-751/18-781 Speech Recognition and Understanding (Fall 2023)
Coding Assignment 1

OUT: September 13th, 4:40 PM ET
DUE: September 29th, 11:59 PM ET

Instructor: Prof. Shinji Watanabe
TAs: Xuankai Chang, Yifan Peng, Brian Yan

Collaboration Policy

Assignments must be completed individually. You are allowed to discuss the homework assignment with other students and collaborate by discussing the problems at a conceptual level. However, your answers to the questions and any code you submit must be entirely your own. If you do collaborate with other students (i.e. discussing how to attack one of the programming problems at a conceptual level), you must report these collaborations. Your grade for Coding Assignment 1 will be reduced if it is determined that any part of your submission is not your individual work.

Collaboration without full disclosure will be handled in compliance with CMU Policy on Cheating and Plagiarism: `https://www.cmu.edu/policies/student-and-student-life/academic-integrity.html`

Late Day Policy

You have a total of **5 late days (i.e., 120 hours)** that you can use over the semester for the assignments. If you do need a one-time extension due to special circumstances, please contact the instructor (Shinji Watanabe) via Piazza.

Programming Notes for Coding Assignment 1

1. The Gradescope testing environment uses Python 3, specifically Python 3.8 with numpy 1.23. Make sure your code is compatible with the same version.
2. Other packages apart from `numpy` and `scipy` are **not allowed** by the autograder, so please do not use any others. Actually, you will not need them.
3. You will not be able to see printed outputs or debugging statements when you submit to Gradescope.

Feature Extraction (5 pts)

Problem Overview

Gradescope assignment: https://www.gradescope.com/courses/564396/assignments/3306887/

Speech recognition systems use acoustic features that can range from raw audio to signal processing motivated features like Log Mel Filterbanks (LMF), spectrograms, and Mel-Frequency Cepstral Coefficients (MFCC). In this problem, you will implement python code to compute LMF features from raw audio files. We also provide code to compute MFCC features based on the LMF features for your understanding. Figure 1 shows the pipeline of speech feature extraction.

Task: Write python code to compute LMF features with the specified configuration. Use the provided template in the handout, and add your code to the template as directed.

Helper code: `feat_extract` directory in the handout.

1. `feat_extract.py`: Main code template. Please fill in the auxiliary functions: `frame_with_overlap`, `compute_powerspec`, and `get_mel_fbank_feat`.
2. `example_data`: Directory with the example test cases you should satisfy. Contains:
   (a) `example_audio.wav`: Example audio file.
   (b) `example_feats.npz`: Expected LMF and MFCC features for the example audio file.

Your Gradescope submission: Submit the completed file `feat_extract.py`.

Grading scheme: 20% `frame_with_overlap`, 30% `compute_powerspec`, 20% `get_mel_fbank_feat`, and 30% the overall LMF feature computation.

Testing locally: You can test your implementation on the audio file `example_audio.wav`, and find the `numpy` array features for LMF and MFCC in `example_feats.npz`. Note that this does not cover all the test cases on Gradescope.

Problem Walkthrough

Consider an audio signal captured by a single-channel microphone, which means the audio is mono. It is converted to a digital signal through sampling (using a sampling rate of 16kHz) and quantization (using Pulse Code Modulation with a 16-bit resolution). This will be the input provided during testing. Now, we will go over how to calculate LMF and MFCC features from this raw audio. MFCCs can be obtained from LMF features using the Discrete Cosine Transform (DCT), so we will first discuss how to obtain LMF features.

Given a digital signal (waveform) $X^{\text{wf}} = (x_n \in \mathbb{R} |\, n = 1, \dots, N)$, where N is the total number of samples, we want to calculate its LMF representation $X^{\text{lmf}} = \big(\mathbf{x}_t \in \mathbb{R}^D | t = 1, \dots, T\big)$ and MFCC representation $X^{\text{mfcc}} = \Big(\mathbf{x}_t \in \mathbb{R}^{D'} | t = 1, \dots, T\Big)$.

1. **Pre-emphasis** Pre-emphasis is used to boost signal components at higher frequencies because higher frequencies are considered important for signal disambiguation.

   Math Description:

$$x_n^{\text{emp}} = x_n - k x_{n-1}, \tag{1}$$

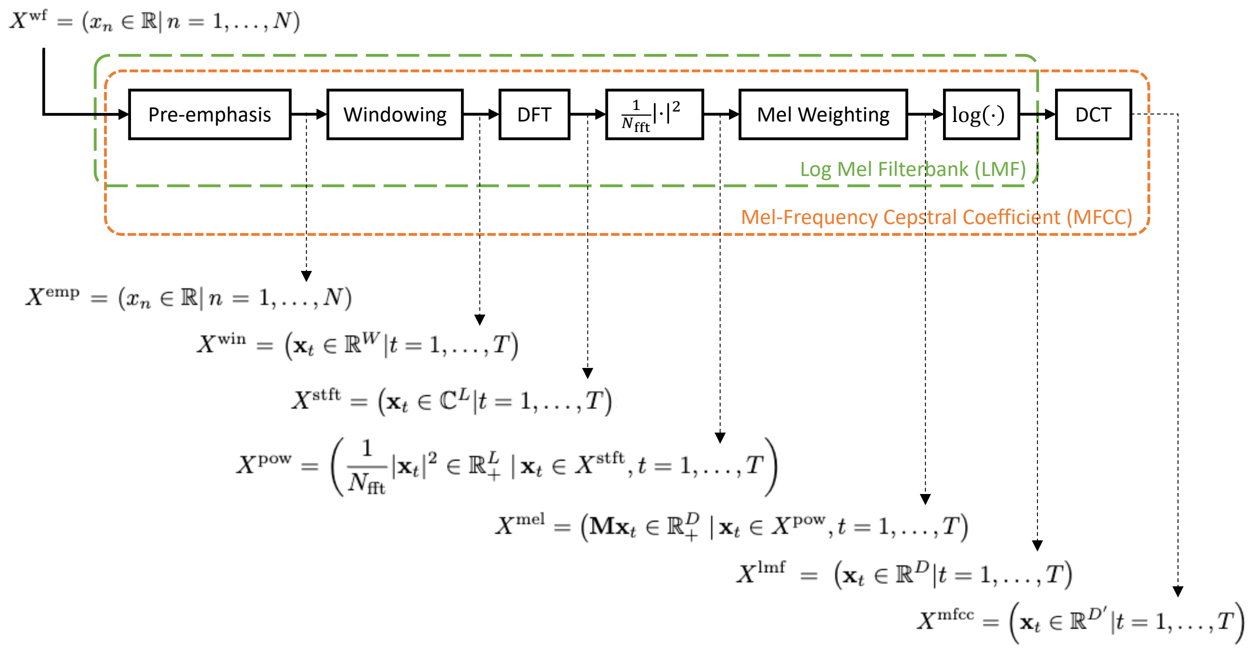
$X^{\text{wf}} = (x_n \in \mathbb{R} \,|\, n = 1, \ldots, N)$
Pre-emphasis
Windowing
DFT
$\frac{1}{N_{\text{fft}}}|\cdot|^2$
Mel Weighting
$\log(\cdot)$
DCT
Log Mel Filterbank (LMF)
Mel-Frequency Cepstral Coefficient (MFCC)
$X^{\text{emp}} = (x_n \in \mathbb{R} \,|\, n = 1, \ldots, N)$
$X^{\text{win}} = \left(\mathbf{x}_t \in \mathbb{R}^W | t = 1, \ldots, T\right)$
$X^{\text{stft}} = \left(\mathbf{x}_t \in \mathbb{C}^L | t = 1, \ldots, T\right)$
$X^{\text{pow}} = \left(\frac{1}{N_{\text{fft}}}|\mathbf{x}_t|^2 \in \mathbb{R}_+^L \,|\, \mathbf{x}_t \in X^{\text{stft}}, t = 1, \ldots, T\right)$
$X^{\text{mel}} = \left(\mathbf{M}\mathbf{x}_t \in \mathbb{R}_+^D \,|\, \mathbf{x}_t \in X^{\text{pow}}, t = 1, \ldots, T\right)$
$X^{\text{lmf}} = \left(\mathbf{x}_t \in \mathbb{R}^D | t = 1, \ldots, T\right)$
$X^{\text{mfcc}} = \left(\mathbf{x}_t \in \mathbb{R}^{D'} | t = 1, \ldots, T\right)$

Figure 1: The pipeline of feature extraction.

where x_n^{emp} is the pre-emphasised audio signal, x_n is the raw audio, and k is the pre-emphasis factor, which is set to 0.95.

Code: Pre-emphasis has been applied on the input time-domain audio signal in the template by calling the provided function `preemphasis`. Nothing needs to be done by you in this step.

2. **Windowing/Framing** Typically, for speech recognition applications, we perform feature extraction over short windows of audio input, because the audio signal frequencies vary over time. The first task is to compute the window length and overlap duration in terms of samples rather than seconds, and this can be done by multiplying the sampling rate in Hertz with the duration in second. Thus, we obtain the frame length W in samples, and overlap length O in samples. Then we extract speech frames with frame length being W.

   Math Description: The pre-emphasised signal $X^{\text{emp}} = (x_n \in \mathbb{R} \,|\, n = 1, \ldots, N)$ is converted into a two-dimensional signal $X^{\text{win}} = \left(\mathbf{x}_t \in \mathbb{R}^W | t = 1, \ldots, T\right)$ which is stored as a 2-D array of size $[T, W]$, where each row represents a window/frame of audio samples.

   Code: Complete the function `frame_with_overlap` which takes as input an audio signal in the time domain, and then creates overlapping windows of audio with duration `window_length` and overlap `overlap_length`. Remember to pad zeros to the last window if it has less than W samples. In this implementation, only the last window can be padded. In other words, you should stop appending new frames when you find a frame that is already shorter than W.

3. **Power Spectrum** To compute the power spectrum, we first calculate the Short-Time Fourier Trans-

form (STFT) of the raw audio signal. Then, we consider the magnitude of the complex STFT, square it, and finally scale it by the inverse of the Discrete Fourier Transform (DFT) length N_{fft}.

Note that the STFT has multiple realizations. Here, we just use the windowing method here, which means we take the raw audio, extract audio frames using a rectangular window function (already done in the previous step windowing/framing), and finally compute the DFT for each window (this step). The DFT can be implemented by using the Fast Fourier Transform (FFT) algorithm, which requires the framed signal X^{win}, and the DFT size parameter N_{fft}. Remember that this N_{fft} parameter should be a power of 2 to efficiently use the FFT algorithm, and that the DFT size should be greater than or equal to the signal length, which means that in this case, N_{fft} should be a power of 2 which is greater than or equal to W. Furthermore, the DFT of a real signal is conjugate symmetric about the center frequency, and therefore, we only need to consider one side of the frequency representation.

Math Description: We first compute the value of N_{fft} as follows:

$$N_{\text{fft}} = 2^{\lceil \log_2 W \rceil}, \tag{2}$$

where W is the length of the window in samples and $\lceil \cdot \rceil$ is the ceiling function.

Then we obtain the power spectrum X^{pow} from the STFT $X^{\text{stft}} = \big(\mathbf{x}_t \in \mathbb{C}^L | t = 1, \dots, T\big)$:

$$X^{\text{pow}} = \left(\frac{1}{N_{\text{fft}}} |\mathbf{x}_t|^2 \in \mathbb{R}_+^L \mid \mathbf{x}_t \in X^{\text{stft}}, t = 1, \dots, T\right), \tag{3}$$

where L is half of the DFT size parameter $L = N_{\text{fft}}/2 + 1$.

Code: Given the framed signal, we need to obtain the power spectrum in the `compute_powerspec` function. To do this, first obtain the DFT of the windowed signal using the `numpy.fft.rfft` function with the appropriate N_{fft}. Then compute the power spectrogram as instructed above.

4. **Mel Scaled Weighting Filterbanks** We need to generate the Mel triangular weighting filterbanks $\mathbf{M} \in \mathbb{R}_+^{D \times L}$ for the given specifications. These filterbanks are based on human perception of sound, and the triangular filters are closer at lower frequencies, while they are farther apart at higher frequencies.

   Code: This function `compute_mel_bank` has been completed for you. Nothing needs to be done by you in this step.

5. **Mel Scaled Filterbank Features** Given the Mel triangular weighting filterbanks $\mathbf{M} \in \mathbb{R}_+^{D \times L}$, and the power spectrum X^{pow} of the framed signal, we obtain the Mel scaled filterbank features X^{mel} by taking the dot product between the two.

   Math Description:

$$X^{\text{mel}} = \big(\mathbf{M}\mathbf{x}_t \in \mathbb{R}_+^D \mid \mathbf{x}_t \in X^{\text{pow}}, t = 1, \dots, T\big). \tag{4}$$

   Code: Complete the `get_mel_fbank_feat` function to obtain the Mel filterbank features as described above. The next step involves taking the logarithm of this. To ensure that is possible, we add an `eps` value of `1e-08` wherever the resulting filterbank feature has a value of zero in the `get_mel_fbank_feat` function.

6. **LMF Features** The LMF features X^{lmf} are then obtained by taking the logarithm over the obtained Mel filterbank features X^{mel}.

   Math Description:
   $$X^{\text{lmf}} = \left(\log(\mathbf{x}_t) \in \mathbb{R}^D \mid \mathbf{x}_t \in X^{\text{mel}}, t = 1, \ldots, T\right). \tag{5}$$

   Code: The provided function `compute_log_feats` is called over the obtained Mel filterbank features. Nothing needs to be done by you in this step.

7. **(Extra) MFCC Features from LMF** First, take the Discrete Cosine Transform (DCT) of the LMF features obtained in the previous step. Then, slice the second dimension of the DCT coefficients so that it contains only the first `num_ceps` elements, where `num_ceps` is the number of MFCC coefficients desired. An optional filtering operation (called liftering or filtering in the cepstral domain) is then applied to boost MFCC amplitudes at higher frequencies.

   Code: Code has been provided in the function `compute_mfcc_feats` for your understanding. Nothing needs to be done by you in this step.