MCA HW-2

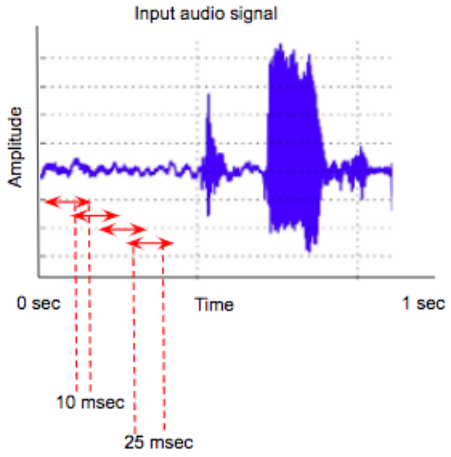
Priysha 2016256

20 March 2020

1 Question 1 - Spectogram

Steps for implementation :

- **Framing** : The input audio signal of length 'n' is divided into 25 msec frames with 10 msec frame overlap. Typically, a signal of length 1 sec with 16000 sampling frequency is divided into frames of 400 data points each with an overlap of 160 data points within two adjacent frames.



  25 msec duration is typically enough to capture information while avoiding drastic changes in frequencies, i.e. assume that frequencies in the signal are stationary over the period of one frame. Hence, Discrete Fourier Transform can be applied to each frame, rather than applying to the whole signal (Short term Fourier transform - STFT).

- **Windowing** : After dividing input audio signal into frames, apply hanning window function to each frame.

$$w(n) = 0.5 - 0.5cos(\frac{2\pi n}{M-1}), 0 \leq n \leq M-1$$

This window (length=M) is used as a tapering function to smoothen the values and removing discontinuties present in the framed signal.

- **Fast Fourier Transform** : FFT (N point, with N = number of sample points in each frame (400)) is applied to each frame of the signal to obtain a vector of length :

$$N/2 + 1$$

This is in accordance to "Nyquist limit" and rule of symmetry of FFT outputs.

- **Power Spectrum** : From the FFT response x of each frame, power of the signal is calculated as :

$$\frac{|x|^2}{NofFFT = 400}$$

This matrix (N/2+1 $\times$ frames) represents the spectogram features of the input signal. This can be then plotted to observe a visual spectogram.

- **Zero padding** : Padding is done wherever necessary to maintain same feature matrix size for each input data point.

1.1 Example plot :

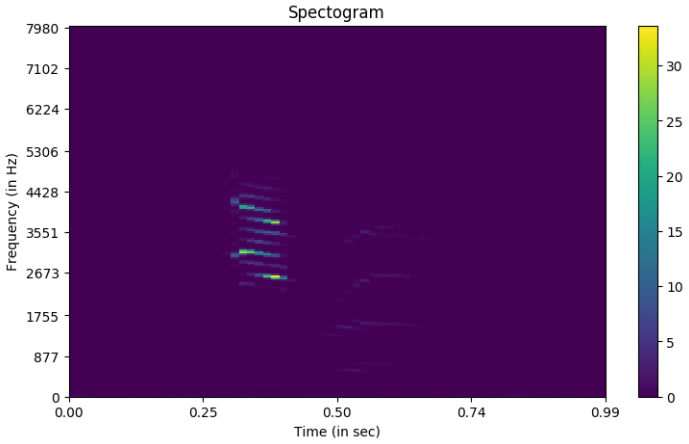


Figure 1: File - training/zero/004ae714_nohash_1.wav

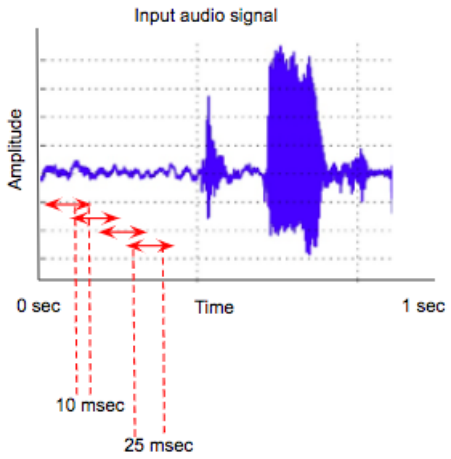
2 Question 2 - MFCC

Steps for implementation :

- **Pre-emphasis** : This filtering step is used to boost the signal quality by increasing the signal to noise ratio. Higher frequencies usually occur in low magnitude as compared to lower frequencies in an audio signal. Pre-emphasis increases the magnitude of the higher frequency signal as compared to the lower frequency signal to boost the amount of energy in the high frequency parts. Now the infomration in higher frequency bands is more available for further processing. If x is the input signal :

$$x[t] = x[t] - 0.95x[t-1]$$

- **Framing** : The input audio signal of length 'n' is divided into 25 msec frames with 10 msec frame overlap. Typically, a signal of length 1 sec with 16000 sampling frequency is divided into frames of 400 data points each with an overlap of 160 data points within two adjacent frames.



25 msec duration is typically enough to capture information while avoiding drastic changes in frequencies, i.e. assume that frequencies in the signal are stationary over the period of one frame. Hence, Discrete Fourier Transform can be applied to each frame, rather than applying to the whole signal (Short term Fourier transform - STFT).

- **Windowing** : After dividing input audio signal into frames, apply hanning window function to each frame.

$$w(n) = 0.5 - 0.5cos(\frac{2\pi n}{M-1}), 0 \leq n \leq M-1$$

This window (length=M) is used as a tapering function to smoothen the values and removing discontinuties present in the framed signal.

- **Fast Fourier Transform** : FFT (N point, with N = number of sample points in each frame (400)) is applied to each frame of the signal to obtain a vector of length :

$$N/2 + 1$$

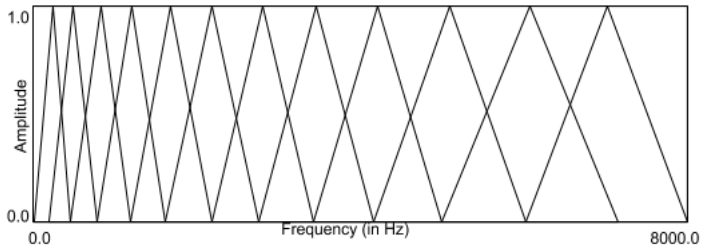
This is in accordance to "Nyquist limit" and rule of symmetry of FFT outputs.

- **Power Spectrum** : From the FFT response x of each frame, power of the signal is calculated as :

$$\frac{|x|^2}{NofFFT = 400}$$

This matrix (N/2+1 $\times$ frames) represents the spectogram features of the input signal.

- **Mel-spaced Filter Bank** : A set of 26 triangular filters is created which is then applied to the power spectrum.



After this, we get 26 energy values per frame. Take log of these values (after ensuring that no value is 0).

- **Discrete Cosine Transform** : We do this orthogonal transformation to decorrelate the filter bank coefficients. We retain only the 1 to 13 coefficients of DCT output. General formula :

$$y = \frac{FFTout[0]}{\sqrt{N}} + 2 * \sum_{m=1}^{M-1} FFTout[m] cos(\frac{\pi(k+0.5)m}{N}), K \in [0, N)$$

- **Energy of input signal** : After recieving 12 coefficients from the previous step, add energy of input signal as the 13th coefficient per frame. Energy of input signal from t1 to t2 time stamps (one frame) :

$$\sum_{t1}^{t2} |x|^2$$

This feature helps identify phonetics in the input signal.

- **Delta and delta-delta features** : These features measure the changes between features from the first frame to the next frame, using first and second derivatives.

$$d(t) = \frac{y(t+1) - y(t-1)}{2}$$

$$dd(t) = \frac{d(t+1) - d(t-1)}{2}$$

This results in 39 mfcc features per frame. I refrain from performing cepstral mean and variance since our audio clips are short and hence these modifications are not reliable. To keep the effect of pre-emphasis preformed earlier, I refrain from sinusodial liftering operations.

- **Zero padding** : Padding is done wherever necessary to maintain same feature matrix size for each input data point.

2.1 Example plot :

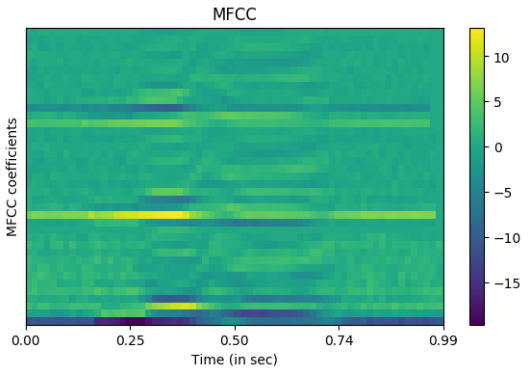


Figure 2: File - training/zero/004ae714_nohash_1.wav

3 Question 3

I try using both SVM and LinearSVM for classification of the validation data points.

3.1 Accuracies without adding noise :

Features	SVM	LinearSVM
Spectogram features	27.43%	37.93%
MFCC features	10.00%	41.41%

Clearly, LinearSVM performs better than SVM(). So i proceed to use linearSVM on noisy data.

3.2 Precision and Recall

Classwise precision and recall values :

Spectrogram

	0	1	2	3	4	5	6	7	8	9
Precision	0.32	0.26	0.30	0.31	0.50	0.43	0.49	0.39	0.44	0.35
Recall	0.33	0.26	0.29	0.34	0.51	0.43	0.41	0.34	0.38	0.47

MFCC

	0	1	2	3	4	5	6	7	8	9
Precision	0.43	0.36	0.19	0.33	0.67	0.35	0.57	0.47	0.51	0.36
Recall	0.53	0.37	0.27	0.45	0.44	0.46	0.61	0.29	0.37	0.24

Average precision and recall :

	Spectrogram	MFCC
Precision	0.38	0.43
Recall	0.38	0.41

3.3 Accuracies after random noise augmentation in the data :

Features	LinearSVM
Spectogram features	37.00%
MFCC features	39.45%

Random noise augmentation is done by replacing features of random number of frames in random input signals with random noise features.

3.4 Precision and Recall

Classwise precision and recall values :

Spectrogram

	0	1	2	3	4	5	6	7	8	9
Precision	0.30	0.27	0.30	0.31	0.49	0.44	0.48	0.38	0.44	0.34
Recall	0.33	0.29	0.28	0.36	0.42	0.43	0.39	0.35	0.38	0.47

MFCC

	0	1	2	3	4	5	6	7	8	9
Precision	0.55	0.29	0.28	0.34	0.39	0.42	0.54	0.34	0.52	0.32
Recall	0.44	0.35	0.28	0.31	0.56	0.38	0.73	0.37	0.21	0.27

Average precision and recall :

	Spectrogram	MFCC
Precision	0.38	0.40
Recall	0.37	0.39

3.5 Observations and Inferences

- Clearly, prediction accuracies, precision and recall all decrease for both spectrogram features and MFCC features after adding random noise. This simply aligns with the intuition that random noise addition to clean signals will hamper the prediction of digits.

- MFCC features outperform spectrogram features in terms of all prediction accuracies, precision and recall. Moreover, MFCC features are less in size than spectrogram features for all input audio data files. This makes it computationally easier to predict using MFCC features.

- Precision: Finds how precise the model is, i.e. how many of those images classified as positive class actually belong to the positive class.

$$Precion = \frac{Truepositive}{Truepositive + Falsepositive}$$

  Recall: Finds what portion of the positive class is predicted by the model correctly.

$$Recall = \frac{Truepositive}{Truepositive + Falsenegative}$$

  So, LinearSVM() trained on MFCC features makes a more precise model without compromising proportion of correctly predicted samples per class.

- MinMax Scaling of the feature vectors can be done to introduce normalization, may be increasing the accuracy.

Features	LinearSVM
Spectogram features	37.01%
MFCC features	39.35%

  However, here the accuracies did not change considerably. Normalization is done to compress the data to a (0,1) range and supress the effect of outliers. Minimal change in accuracies shows that the input data may not have many outlier data points.

- For all the results produced above, feature matrices of each data point (both spectrogram feature matrix and MFCC feature matrix) have been ravelled to get a single feature vector per data point. This approach lines up all features in the order in which they appear in the signal temporaly (time-wise / frame-wise).
  Another approach to converting the matrix into a vector can be to take an average of all feature vectors per frame. This puts equal weightage

on signal information from all time instances and eliminates the temporal information, hence gives less accuracy. Accuracy using MFC features without noise came out to be 9.08% in this case.

References

[1] Precision-Recall-F1

[2] Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs)

[3] Speech Recognition — Feature Extraction MFCC & PLP

[4] Music Feature Extraction in Python

[5] Mel Frequency Cepstral Coefficient (MFCC) tutorial

[6] Yumi's blog - Implement the Spectrogram from scratch in python

[7] Plotting A Spectrogram Using Python And Matplotlib

[8] Create audio spectrograms with Python

[9] Discrete Fourier Transform - Simple Step by Step

[10] Yumi's Blog - Review on Discrete Fourier Transform

[11] Towards Data Science - Fast Fourier Transform

[12] Understanding the FFT Algorithm

[13] DCT