Speech recognition for isolated words

Master 2 – Voice recognition and automated language processing

This practical aims to develop in Python, a small recognition system for isolated words using a DTW. The procedure for computing the MFCC acoustic features is detailed, and a database of single-digit read in English will allow the system's performance evaluation.

1 Training and test sets

English single-digit recordings are available in `.wav` format (16-bit linear mono sampled at 8 kHz). The filenames are of the form: `xy_3a.wav`, where the 1st 2 characters are a speaker's identifier, the 4th character is the number spoken (zero has two pronunciations: 'o' for 'oh' and 'z' for 'zero'). The `digits` directory contains two subdirectories `train` with an average of 25 pronunciations per digit and `test` with about 10 per digit. There is no overlap between the learning and testing speakers, and each speaker pronounces an average of 5 different digits.

2 Acoustic features extraction

The computation process of acoustic parameters MFCC (Mel Frequency Cepstral Coefficients), common in speech processing, is presented in this section.

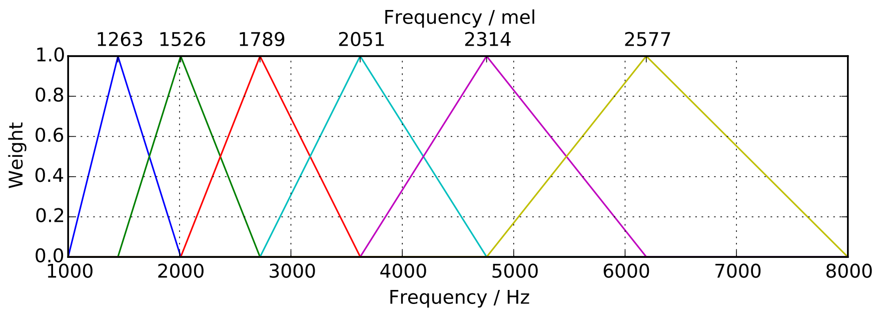
2.1 Windowing

- The speech signal x_n is recorded at a sampling frequency f_s of typically 8 kHz for voice in telephone quality or 16 kHz for better speech quality
- Pre-emphasis step: to decrease high frequency energy, with a pre-emphasis factor of $\alpha = 0.95$ ou 0.97:
$$x'_n = x_n - \alpha\, x_{n-1}$$
- Extraction of a 20 to 40 ms frame of the signal to perform short term sliding window analysis. You will use a 32 ms window corresponding to $win_n = 256$ samples for $f_s = 8$ kHz. With a hop size of 10 ms, the frame t starts at the sample $n_0 = 0.010 \times F_e \times t$
- Weighting to reduce the discontinuity at the frame border, typically using a Hamming window (available through the `numpy.hamming(win_n)` function):
$$s_n = w_n \times x'_{n_0+n} \quad \text{with} \;\; w_n = 0.54 - 0.46 \cos\left(2\pi \frac{n}{N-1}\right), \quad 0 \le n \le N-1$$

2.2 Cepstral coefficients

- Computation of frame spectrum amplitude $|S|$ by the real part of the Fourier Transform (`numpy.fft.rfft`, and `abs`), which leads to $N/2+1$ values corresponding to the frequencies from 0 to $f_s/2$.
- Energy measurement on a triangular filter bank uniformly distributed on the Mel frequency scale (quasi-linear below 1 kHz, logarithmic above):
$$\text{Mel}(f) = 2595 \log(1 + f/700)$$

We typically use about twenty filters B_n^i, with $i = 0 \dots n_f - 1$ and $n = 0 \dots N/2$. For convenience, the function that computes the filters is provided (`melbank`):



The computation of the energy at the output of each filter is then a simple cumulative product:

$$E_i = \sum_{n=0}^{N/2} |S_n| B_n^i, \quad \text{for} \quad i = 0 \dots n_f - 1$$

- The computation of the logarithm of the energies E_i and the subsequent transformation into discrete cosine (DCT) is equivalent to (and can thus be replaced by) an inverse discrete Fourier transform: $\mathbf{c} = \text{FFT}^{-1}(\log(\mathbf{E}))$ (with `numpy.fft.irfft`)
- Extraction of the first 13 cepstral coefficients c_0 to c_{12}.
  The first coefficient c_0 is generally replaced by the logarithm of the energy calculated directly from the signal (while paying attention to zero values) and normalized with respect to a reference level:
  $$e = \log \sum_{n=0}^{N-1} s_n^2 - e_0$$

2.3 Feature *delta*

For each frame t, we obtained a vector of cepstral coefficients MFCC $\mathbf{c}(t)$ of dimension 13. The addition of differential features makes it possible to take better account of the speech signal dynamics, thanks to the preceding and following frames (context limited to ± 1 frames here):

$$\Delta \mathbf{c}(t) = [\mathbf{c}(t+1) - \mathbf{c}(t-1)] / 2$$

The vector of parameters used for each frame results from the concatenation of the vectors $\mathbf{c}(t)$ and $\Delta \mathbf{c}(t)$, for a final vector of dimension 26. It is possible to use a broader context by, e.g., computing additional 2nd order differential coefficients.

3 Dynamic Time Warping

Implement a program for computing the distance between words using the DTW algorithm seen in class. You will use a simple Euclidean distance between the MFCC vectors of these words.

4 Experiments

You can measure the recognition performance of the test set recordings, by comparing them using the DTW to all or part of the recordings in the training set and by checking whether the resulting label matches that in the filename.