Signal Processing
Lab 6: Cepstral analysis

Ahmed Besbes *Panagiotis Koutsourakis* *Loic Simon*
Chaohui Wang

December 12, 2008

Introduction

Cepstral analysis is widely applied in signal processing. It is particularly useful in speech synthesis, speech recognition and other audio applications. During this lab, you will be asked to compute the cepstrum of a speech signal, "lifter" it (i.e. "filter" in cepstral analysis) and resynthetize it. In order to achieve the latter, you will need to estimate the impulse response of the vocal track for the targeted phonem.

Open file *run_lab.m* to familiarize yourself with the commands for loading and playing sounds (Octave users: you might not be able to play the sounds directly from Octave. If so, save the sound by using command `wavwrite(sound, fs, 16, 'sound.wav')` and listen to it with any media player).

Compute the cepstrum

Definition of $\hat{x}$, the complex cepstrum of x, with $X = \mathrm{DTFT}(x)$:

$$\hat{x}[n] = \mathrm{DTFT}^{-1}\left(\hat{X}(f)\right) \text{ with } \hat{X}(f) = \log|X(f)| + j \arg X(f)$$

$\hat{X}(f)$ is the complex log spectrum. Note: the real cepstrum $\check{x}[n] = \mathrm{DTFT}^{-1}(\log|X(f)|)$.

Let $s = h \star t$, with h the impulse response of a filter and t an excitation signal.

$$\begin{aligned}
\hat{S}(f) &= \log(H(f).T(f)) \\
&= \log H(f) + \log T(f) \\
&= \hat{H}(f) + \hat{T}(f)) \\
\hat{s}[n] &= \hat{h}[n] + \hat{t}[n]
\end{aligned}$$

The cepstrum provides a way to identify h and t from s.

Q. Write a Matlab function to compute the cepstrum of a signal

- Fill the blanks in the file `compute_cepstrum.m`
  - Compute `S` the FFT of `s`
  - Compute `C` as the log of the absolute value of `S`
  - Compute `P` as the phase of `S` (Matlab function: `angle`). Use the `rcunwrap` function (included in the file) to unwrap the phase of `S`.
  - Compute the complex log spectrum from `C` and `P`.
  - Compute `c` and `rc`, respectively the complex and real cepstra. Both should be real signals. You can enforce this by taking the real part of the results.
- Load the sound *a.wav*. Play it with the command `soundsc`. Is it a voiced or unvoiced phonem ? What should you expect from the shape of its spectrum ?
- Plot `C` and `P`. On the figure, give the pitch of the sound.
- Plot `c`. On the figure, tell the excitation apart from the vocal track component.

Q. Use the function `cepstrum_window` to decompose the cepstrum into two parts

- In general, the pitch of a voice is comprised between 80 Hz and 250 Hz. Knowing this, what should be the duration (or "quefrency") of the square window applied to the cepstrum to keep only the part that corresponds to the vocal track effect.
- On the same figure, plot the vocal track part and the excitation part of the cepstrum for the sound *a.wav*.

Voiced/unvoiced detection and pitch estimation

We have seen previously that it is relatively easy to tell the periodic excitation from the vocal track effect from a cepstrum. However, before providing a pitch, we have to know if the excitation is periodic (voiced phonem) or random (unvoiced phonem). We have to determine whether the excitation part of the cepstrum contains peaks (voiced) or not (unvoiced). This can be achieved by thresholding the excitation (after normalizing it to make it more robust).

Write such a Matlab function

- Use the prototype `[v, p] = is_voiced( ex, fs )` to write a function which determines if the speech sound is voiced or not and, if it is, estimates the pitch.
- `ex` is the excitation part of the cepstrum obtained from `cepstrum_window`. `fs` is the sampling frequency.
- `v` is binray, equals to 1 when the sound is voiced, equals to 0 otherwise. `p` is the pitch, in the case of a voiced signal (it is set to 0 for unvoiced sounds).
- Test your function with the provided sounds (e.g. *a.wav*, *o.wav* and *s.wav*).

Speech synthesis

We now have enough information to generate an excitation signal. We also know that the "low quefrency" part of the cepstrum (in vector `vt`) mostly represent the vocal track filtering effect.

Q. Compute an estimation of the impulse response of the vocal track from the data in vector `vt`.

Q. Synthetize a phonem

- Use Matlab command `filter2` to filter an excitation signal.
- Synthetize a vowel and listen to it. Do the same with sound `s.wav`.

Q. Synthetize a whole sentence

- Fill the blanks in *synthetize.m* and run it (c.f. *run_lab.m* for parameter values) for different speech examples.
- You can check the quality of your pitch estimation by ploting vector `pitch`.