Summer University

Speech Recognition

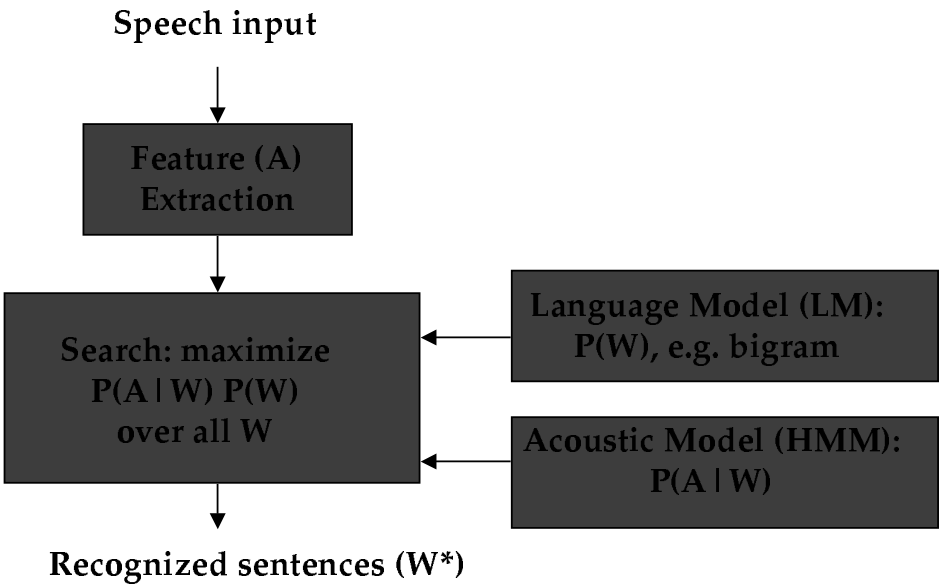
01 July – 30 August 2002

Feature Extraction

Otto-von-Guericke-Universität Magdeburg

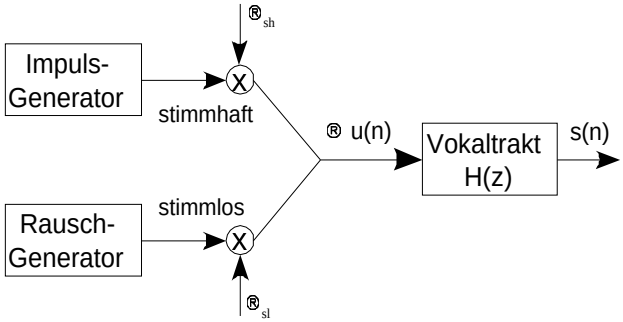
Architecture of a ASR System

W = a word sequence (e.g. word/ sentence/ whole dictation)
A = an acoustic feature vector sequence (the input for the recognizer)



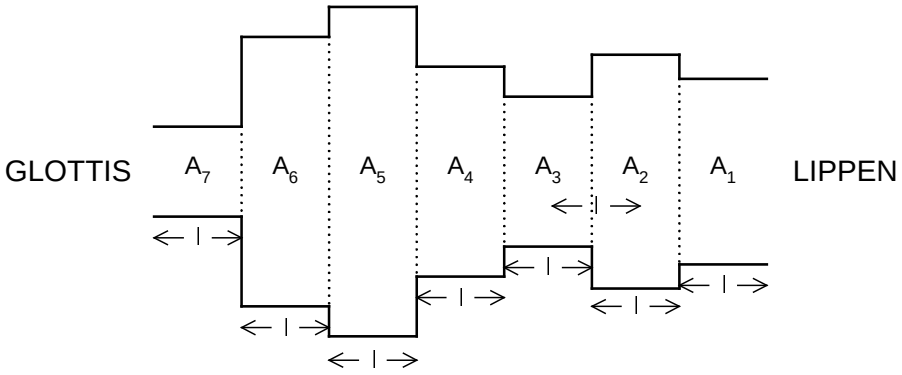
Source Filter Model

- Assumption:
  - Speech is produced by an excitation **u** (impulses or white noise) and filtered by the vocal tract filter with the impulse function **h**.
  - So the resulting speech **s** results from the convolution: **s** = **u** * **h**.



Vocaltract Model

- concatenation of ideal cylindrical tubes
- same width but different surfaces A_n



$$H(z) = \frac{\prod_{j=0}^{M}(1+k_j)}{1-\sum_{j=1}^{M} a_j z^{-j}} \qquad \text{with} \quad k_j = \frac{A_j - A_{j+1}}{A_j + A_{j+1}}$$

Sampling

- Shannon theorem: $f_a \geq 2f_g$
  - signal can be exactly reproduced
- sampling frequencies for ASR:
  - telephone speech: f_a = 8 kHz
  - computer dictation: f_a = 16 kHz

- from the time signal s(t) follows:

$$s[n] = s(t - nT) \quad \text{with } T = 1/f_a$$

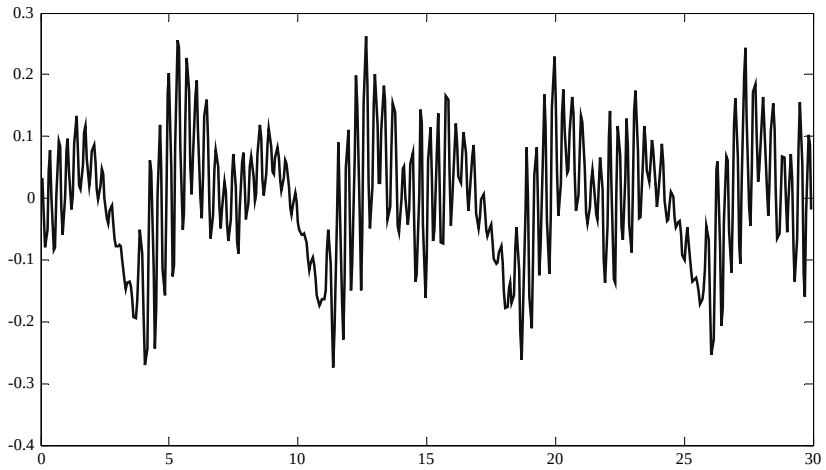
Short-Term Spectral Analysis

Why Short-Term Analysis?

- The frequency distribution over an entire utterance doesn't help much for recognition.
- Most acoustic events (e.g. phonemes) have durations in the range of 10 to 100 ms.
- Many acoustic events are not static and need more detailed analysis.

Short-Term Spectral Analysis

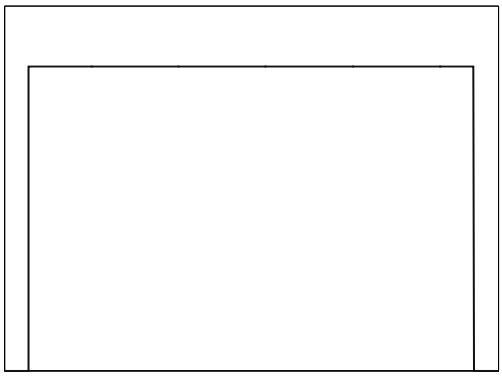
- We assume the speech-signal is short-time stationary!



Windowing

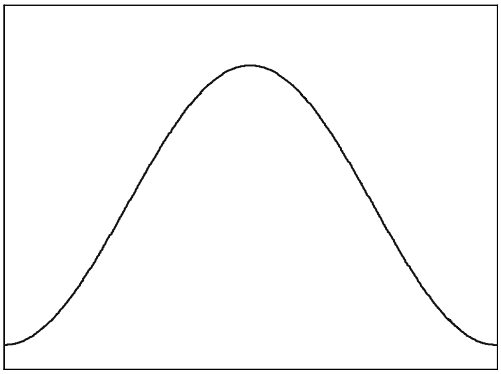
- For Short-term analysis the signal must be zero outside of a defined range
- this is performed by multiplying the signal with a window
- Normally we choose a **window-width** of 20 – 30 ms
- The **window-shift** is usually 10 ms

Window Shapes



Rectangular Window

$W_n=1$



Hamming Window

$W_n= 0.54-0.46 \cos(2\pi n / (N-1))$

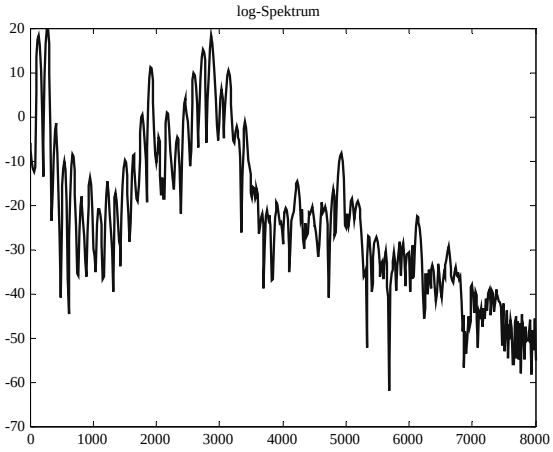
Other common windows: Gauss-, Hann-, Blackmann-Window

Short-Term Spectral Analysis

- Discrete Fourier Transformation (DFT)

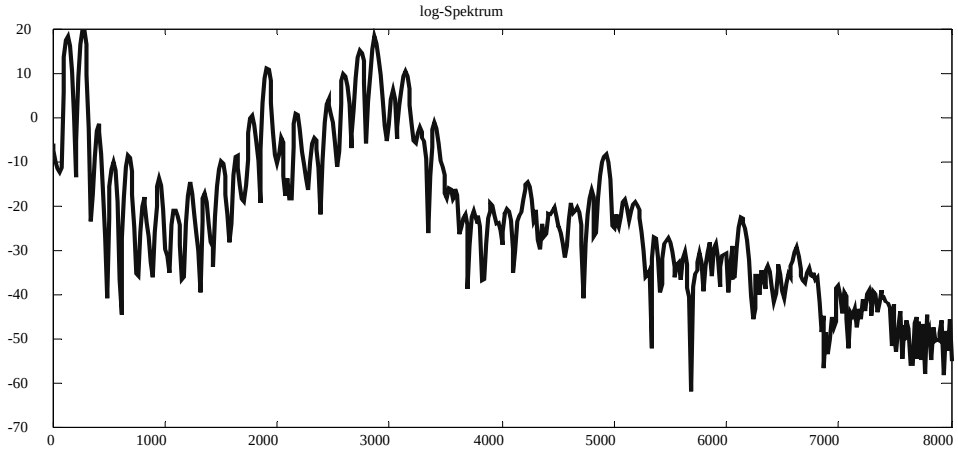
$$S(e^{j\omega}) = \sum_{n=0}^{N-1} s[n] e^{-j\omega n/N}$$

resulting short term spectrum (log-scale)



Deconvolution

- We are interested in the formant-structure
- spectral smoothing
  - Cepstrum
  - Linear Prediction



The Cepstrum

So if $s = u*h$ (source filter model),

then $FT\{s\} = FT\{u\} \cdot FT\{h\}$

and $\log FT\{s\} = \log FT\{u\} + \log FT\{h\}$

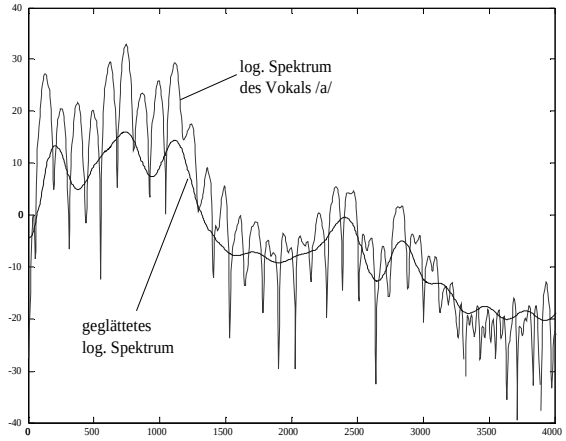
thus $FT^{-1}\{\log FT\{s\}\} = FT^{-1}\{\log FT\{u\}\} + FT^{-1}\{\log FT\{h\}\}$

- The coefficients of this transformation are called cepstral coefficients or simply cepstrum

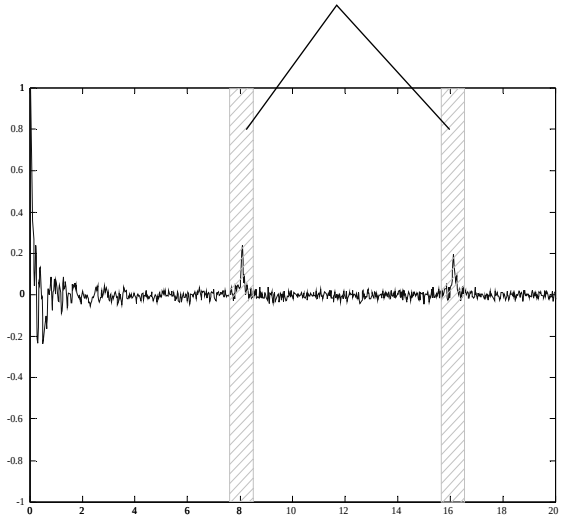
The Cepstrum

Example of the vowel „a“

Original and smoothed spectrum



Cepstral Peaks



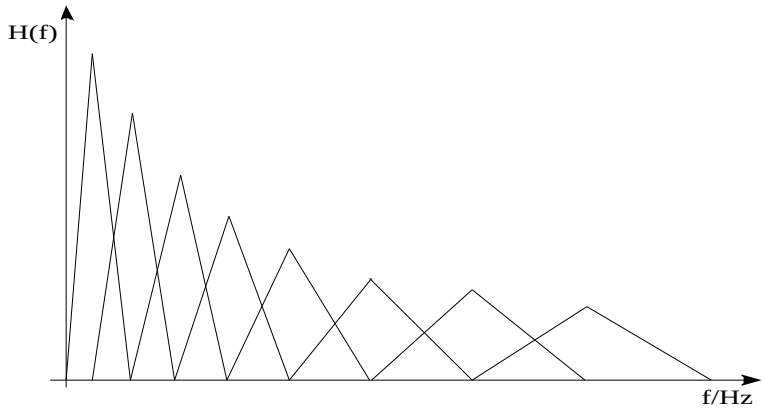
The Cepstrum

- For speech recognition, only the lower cepstral coefficiens are used.
- When we set some of the coefficients to 0.0 then this process is called **liftering**.
- The lower coefficients reflect the macrostructure and the higher coefficients the microstructure of the spectrum.
- The 0th coefficients reflects the signal energy

Non-linear frequency scales

- Mel-scale:

$$f_{mel} = 1125 \log(0.0016\ f + 1)$$



Linear Predictive Coding (LPC)

Idea:

- samples can be approximated from past samples:

$$s[n] \approx a_0 + a_1 s[n-1] + a_2 s[n-2] + \cdots + a_p s[n-p]$$

- The actual signal differs from the estimated signal:

$$s[n] = -\sum_{j=1}^{p} a_j s[n-j] + e[n] \Rightarrow e[n] = s[n] - \hat{s}[n] = \sum_{j=0}^{p} a_j s[n-j]$$

Linear Predictive Coding (LPC)

which after a z-transformation becomes:

$$A(z) = \frac{E(z)}{S(z)}$$

using the Z-Transformation:

$$A(z) = \frac{S(z) - \sum_{j=1}^{p} \alpha_j \, S(z) z^{-j}}{S(z)}$$

$$= 1 - \sum_{j=1}^{p} \alpha_j \, z^{-j}$$

if we assume a=α, we can model the vocaltract parameters!

Linear Predictive Coding (LPC)

- Because we want to find the LPC coefficients α_j, we have to minimize the squared error:

$$\sum_{n=0}^{N} e_n^2 = \sum_{n=0}^{N} \left(\sum_{j=0}^{p} a_j f_{n-j} \right)^2$$

- using the autocorrelation method and the Levinson-Durbin recursion, we‘ll receive the LPC coefficients α_j.

Linear Predictive Coding (LPC)

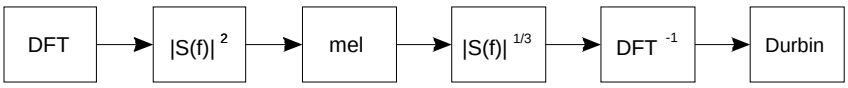
- the frequency response is computed by

$$H(z) = \frac{1}{DFT\{[1, \alpha_1, \alpha_2, ..., \alpha_p, 0, ..., 0]\}}$$

N-p-1 Zeros

Perceptual Linear Prediction (PLP)

- The combination of Cepstrum and LPC is called Perceptual Linear Prediction
- We can show that the autorrelation of a discrete signal (but without the log scale) is equal to the real cepstrum.



Deltas and Acceleration Coefficients

- after these signal processing methods we want to put more context into the feature vectors.
  - Deltas:

$$\Delta x_\mu = \frac{1}{2}(x_{\mu+1} - x_{\mu-1})$$

  - Acceleration:

$$\Delta\Delta x_\mu = \frac{1}{2}(\Delta x_{\mu+1} - \Delta x_{\mu-1})$$

Concatenating feature vectors

- concatenating:

$$y(m) = \begin{pmatrix} x(m-1) \\ x(m) \\ x(m+1) \end{pmatrix}$$

- resulting dimensions:
  - HTK (typical):
    - using 13 Mel-Cepstrum Comp. + Deltas + Acc.
  - other systems:
    - using 16 Mel-Cepstrum Comp. + Deltas + Signal Energy => 33 components
    - 165 dimensions after concatenating 5 feature vectors!!!

Linear Discriminant Analysis (LDA)

- Lots of the used features contain no relevant informations.
- So we can reduce the dimensionality of the feature-vectors „without“ loss.
- LDA is a statistical method to transform features with respect to
  - the main components of the feature-class
  - the main components of the whole distribution

Linear Discriminant Analysis

- Original feature vectors are multiplied by:

$$y = \Theta^T x$$

- the LDA-Matrix Θ is defined as the solution of the eingenvalue problem:

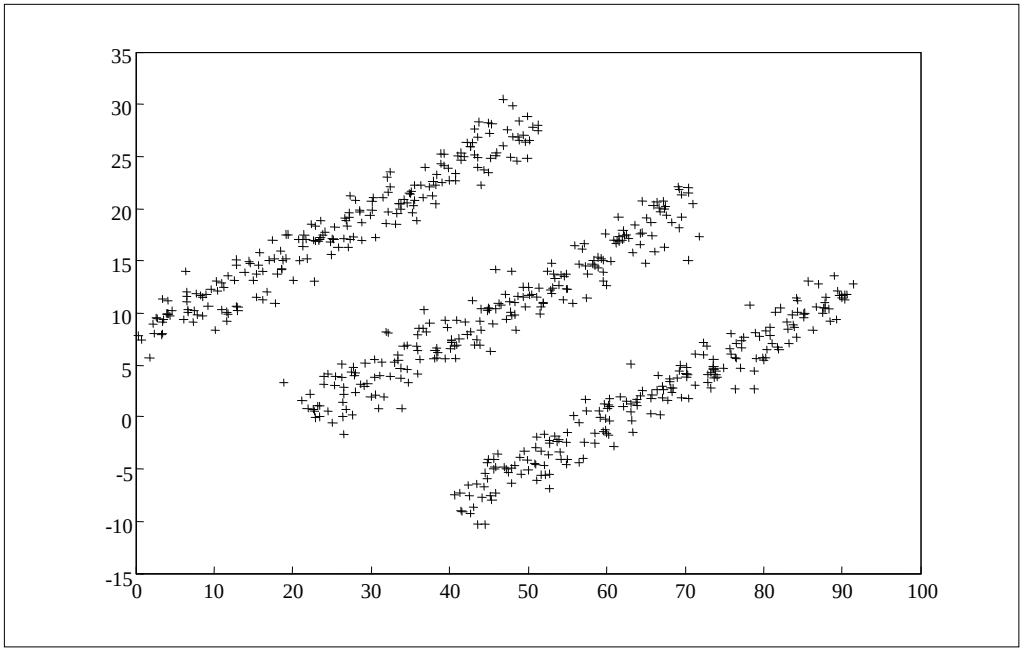
$$S_W^{-1} S_T \Theta = \Theta \Lambda$$

- with the Within-Scatter matrix S_W and the Total-Scatter matrix S_T

$$S_W = \frac{1}{N} \sum_{k=1}^{K} \sum_{\substack{i=1 \\ g(i)=k}}^{N_k} (x - m_k)(x - m_k)^T \qquad S_T = \frac{1}{N} \sum_{i=1}^{N} (x - m_0)(x - m_0)^T$$

Linear Discriminant Analysis

- Example: we want to discriminate the three classes in the 1 dimensional space



From Speech to a resulting feature-vector