Master's degree ICT Internet Multimedia Engineering

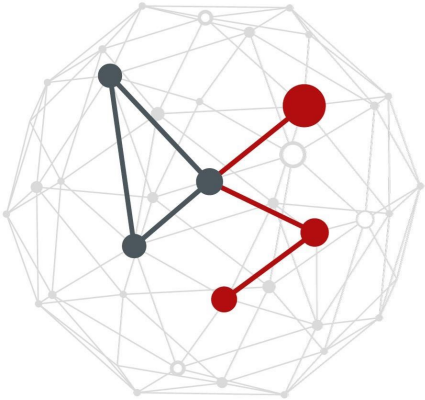
SPEECH MODELS AND AUTOMATIC SPEECH RECOGNITION (ASR) SYSTEMS

Michele Rossi

michele.rossi@dei.unipd.it

Dept. of Information Engineering

University of Padova, IT



DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DIPARTIMENTO MATEMATICA

UNIVERSITÀ DEGLI STUDI DI PADOVA

Outline (1/2)

- Quick intro to speech recognition systems
- Mel Frequency Cepstral Coefficients (MFCC)
  - What are they for?
- Computing MFCC
  - The 6 basic steps for their computation
  - The **Mel scale** for human perceived frequency
  - The 6 steps in greater detail
- Additional features
  - Energy
  - Delta coefficients
  - Delta-Delta coefficients

Outline (2/2)

- Automatic Speech Recognition (ASR) sytems
  - The noisy channel model
  - Introduction to ASR systems

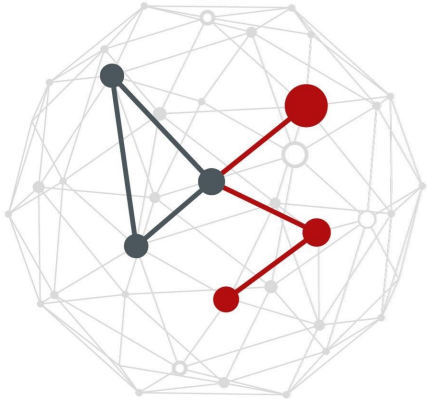
Speech recognition: a quick intro

- Raw audio samples are not suitable for statistical modeling
  - We represent them using feature vectors (MFCCs)
  - **From raw speech signals to MFCCs**
    - waveforms are modeled as a sequence of piecewise stationary signals
- Speech recognition
  - Input: sampled (at regular intervals) & digitized speech data
  - Desired output: sequence of words that were spoken
  - It is a pattern matching problem
    - Combines *acoustic* and *language* models
  - Machine learning tools
    - Historically Hidden Markov Models (HMM) were the tool of choice
      - It has been so for nearly two decades, until around 2009
    - Most modern systems use neural networks
      - Convolutional (CNN), Graph (GNN) and Recurrent (RNN) Neural Networks

MIME

Master's degree ICT Internet Multimedia Engineering

SPEECH FEATURES



DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DIPARTIMENTO MATEMATICA

UNIVERSITÀ DEGLI STUDI DI PADOVA

MFCC – what are they for?

- First step in any speech recognition system
  - **MFCC** are audio features: they are good to identify the *linguistic content* (neglecting background noise, emotion, etc.)
- Sounds generated by humans
  - Are filtered by the shape of the vocal tract, including tongue, teeth, etc.
  - This shape determines which sound comes out
  - If we determine this shape accurately, this should give a good representation of the phoneme that is being produced
- Shape of the vocal tract
  - Manifests itself in the envelope of the short-term power spectrum
  - The job of MFCCs is to accurately represent this envelope [Davis1980]

[Davis1980] S. Davis, P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sequences," *IEEE Transactions on Acoustic, Speech and Signal Processing*, Vol. 28, No. 4, 1980.

Computing MFCC – what is it?

- A highly engineered preprocessing of the audio waveform
- Designed
  - to discard the large amount of information in waveforms that has been found to be irrelevant for discrimination
  - to express the remaining information in a form that facilitates discrimination, e.g., with GMM-HMMs or Neural Networks (NN)
- GMM advantages
  - given enough components, (i) they can model PDFs to any degree level of accuracy & (ii) they are fairly easy to train through EM
- Neural Network advantages
  - Capable of representing complex dynamics without relying on a priori built mathematical models of the phenomenon
  - They have now replaced Hidden Markov models

MFCC computation involves 6 steps

1. Segment the signal into short frames
2. For each frame: calculate its power spectrum
3. Apply a Mel filterbank to the power spectrum
   - Sum the energy within each filter
4. Take the logarithm of *all* filterbank energies
5. Take the DCT of the log filterbank energies
6. Retain DCT coefficients 2-13, discard the others

Step 1 - framing

- An audio signal is constantly changing (*non-stationary*)
- Dealing with non-stationary data is difficult
- To simplify things
  - We assume that on short time scales the signal does not change much
  - "Does not change" means it is statistically stationary (almost)
- This is why we frame the signal into 20-40ms time windows
  - Standard value is 25ms
  - **Shorter:** not enough samples to get a reliable spectral estimate
  - **Longer:** the signal (statistics) changes too much within each frame

Step 2 – power spectrum

- Compute the power spectrum of a frame
  - This is motivated by the human *cochlea* (organ in the ear)
  - It vibrates at different spots depending on the frequency of the incoming sound waves
  - Depending on the location of the cochlea that vibrates, different nerves fire informing the brain that certain frequencies are present
  - This is then translated by the brain into words, sentences, etc.

- The periodogram estimate of the power spectrum
  - Does a similar job
  - Informing us on which frequencies are present in the current frame

Step 3 – filterbank

- The periodogram estimate of the power spectrum
  - Still contains a lot of information that is not required by Autonomous Speech Recognition (ASR) systems
  - In fact, the cochlea cannot discern the difference between two closely spaced frequencies
  - This effect becomes more apparent as frequencies increase
- Hence
  - We take clumps of periodogram **bins** and sum them up to get an *idea of how much energy there exists in each region*
  - This is performed using a Mel filterbank
    - **First Mel filter is very narrow:** to get an idea of how much energy there is around 0 Hertz
    - **Filters get wider as frequencies get higher:** we become less concerned about variations (Mel scale tells us how to space filterbanks)

Step 4 – logarithm

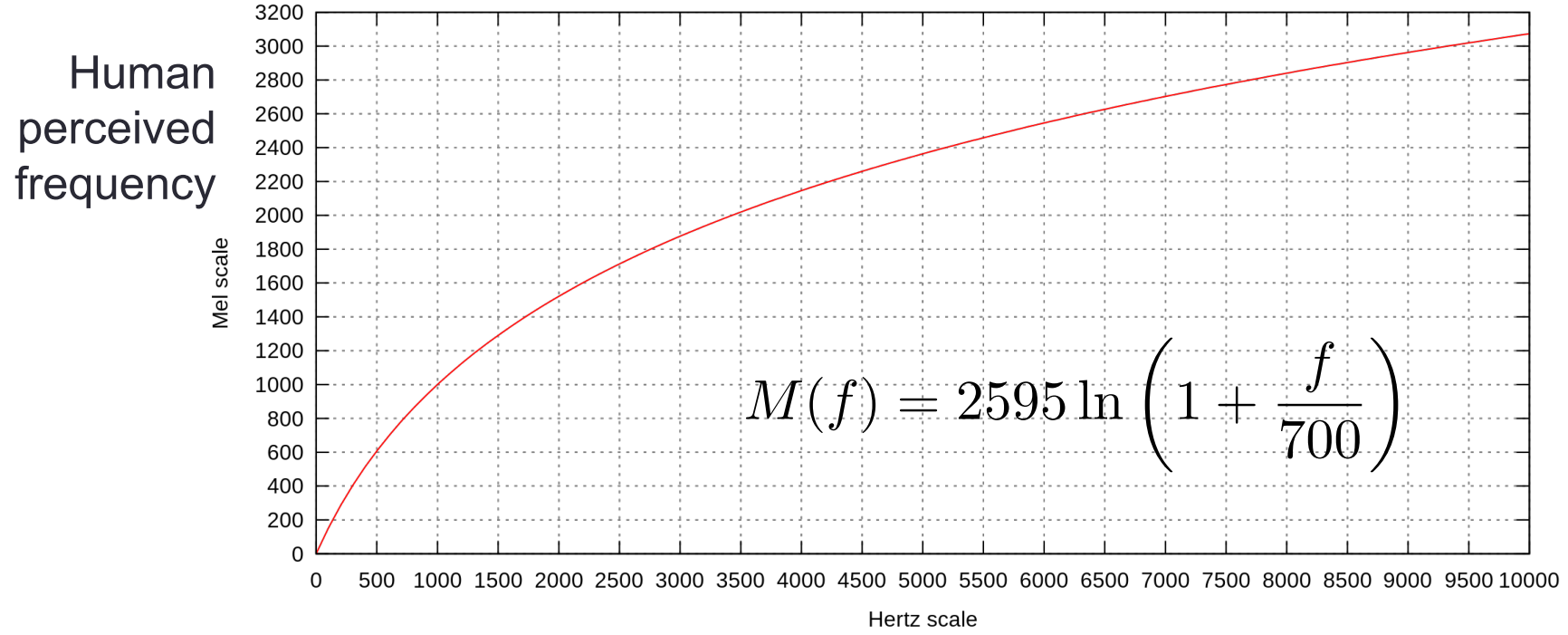
- Once we have the filterbank energies
  - We take their logarithm
  - This is also motivated by human hearing: we do not hear loudness on a linear scale
  - Generally, to double the *perceived* volume of a sound we need to put 8 times as much energy into it
  - This means that large variations in energy *may not sound that different* if the sound is loud
- Hence
  - This compression operation (logarithm) makes our features match more closely what humans actually hear

Steps 5 and 6 – DCT

- Discrete Cosine Transform (DCT)
  - There are two main reasons for applying it
  - First: filterbanks are quite overlapping (meaning that filterbank energies are correlated with each other)
    - DCT decorrelates filterbank energies
    - This means that diagonal covariance matrices can be used to model the features with GMM and/or GMM/HMM systems (very desirable)
    - Also (energy compaction property of DCT) we obtain a compact representation in the first DCT coefficients
  - Second: discard some information for improved performance
    - Only 12 of the 26 DCT coefficients are kept (energy compaction of DCT)
    - Higher DCT coefficients represent fast changes in the filterbank energies
    - These fast changes degrade ASR performance
    - A (small) improvement is achieved by dropping them

The Mel scale - M(f)

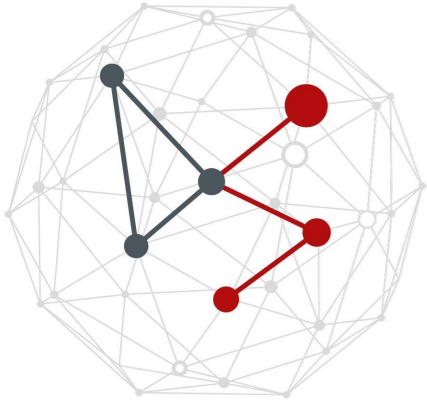
- Relates the *human perceived frequency* (or pitch) of a pure tone to its *actual measured (perceived) frequency*
- Humans are much better at discerning small changes in pitch at low frequencies that they are at high frequencies
- Incorporating it makes ASR match more closely what humans hear



MIME

Master's degree ICT Internet Multimedia Engineering

STEPS 1-6 IN MORE DETAIL

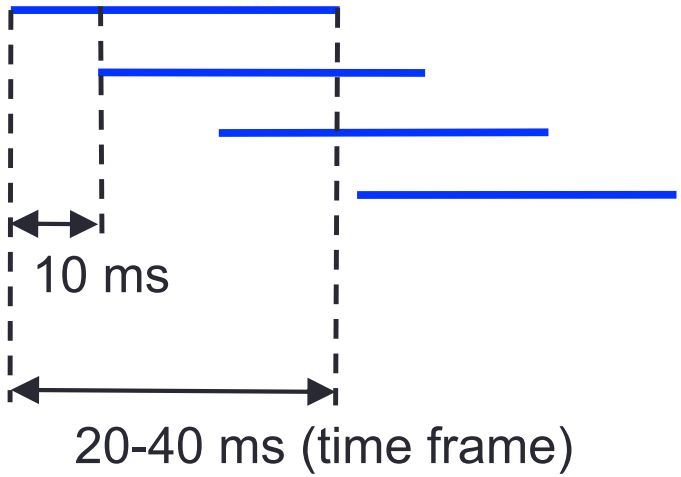


DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DIPARTIMENTO MATEMATICA

UNIVERSITÀ DEGLI STUDI DI PADOVA

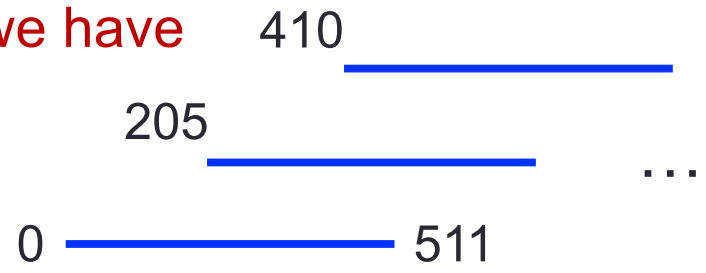
Step 1 – framing (1/2)



- the speech signal is segmented into a sequence of partially overlapping frames
- signal is assumed to be quasi stationary within each frame
- signal processing pipeline applied to each frame → per frame MFCCs

Step 1 – framing (2/2)

- Frame the audio signal into [20,40] ms frames (standard value is 25 ms)
- Example: 20.480 kHz signal, with 25 ms we have
  - sampling rate: 20,480 samples/second
  - 0.025 x 20,480 = 512 samples/frame



- Frame step is usually 10 ms (205 samples)
  - This allows for some overlap between subsequent frames
  - First frame (of 512 samples) starts at sample 0
  - Second frame (still of 512 samples) start at sample 205, etc.
  - Keep framing until the end of the speech signal
  - If the signal does not divide into an integer number of frames, pad it with zeros
- Notation
  - s(n) is the “raw” audio signal (i.e., time domain signal)
  - $s_i(n)$ is audio frame i with n ranging from 0 to 511 in our example

Step 2 – power spectrum (1/4)

- For each frame $s_i(n)$, $n = 0, 1, \ldots, N-1$ $(N = 512)$
  - Calculate the complex DFT (Discrete Fourier Transform)

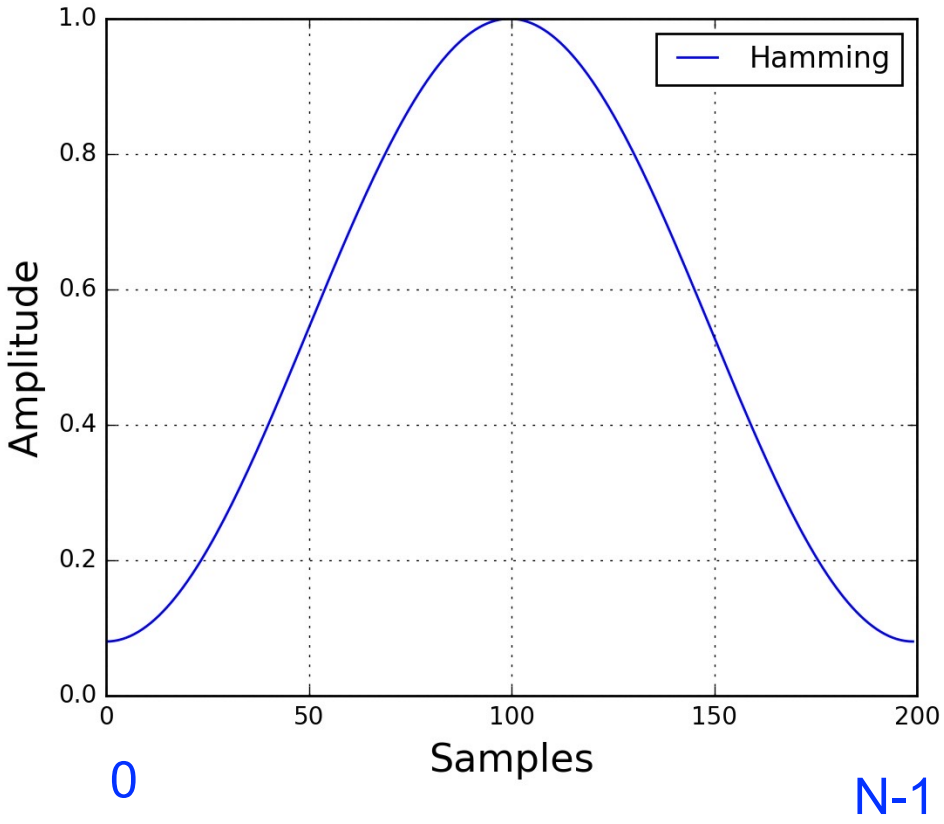
$$S_i(k) = \sum_{n=0}^{N-1} s_i(n) w(n) e^{-j2\pi kn/N}, \; k = 0, 1, \ldots, N-1$$

  - DFT
    - Converts a time sequence of N equally spaced samples into an N-long complex sequence, which is a complex-valued function of frequency
    - The DFT (with w(n)=1) completely describes the discrete Fourier transform of an N-periodic time sequence
    - When applying DFT, we are implicitly applying it to an infinitely repeating (periodic) signal. However, if this is not the case, e.g., first and last samples of $s_i(n)$ do not match, this is *interpreted as a discontinuity* in the signal and *generates a lot of high frequency response (artifacts) in the transformed signal*, that we do not want → use of "windowing" w(n)

Step 2 – power spectrum (2/4)

- Often used: Hamming window - w(n)

$$w(n) = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right), \; n = 0, \ldots, N-1$$



Frequency domain

- Frequency index $k = 0, 1, \ldots, \dfrac{N}{2}$
- DFT step size (frequency resolution) $\Delta f = F_s/N$
- Frequency components $f_k = k\Delta f$

Example – frequency domain grid after DFT

$$F_s = 20,480 \text{ Hz}, N = 512 \rightarrow \Delta f = 40 \text{ Hz}$$

Nyquist freq.

Step 2 – power spectrum (4/4)

- Note
  - Among the N=512 (in our example) complex DFT values
  - Only the first 1+N/2 values are significant, the remaining ones are complex conjugates of the first 1+N/2 values (this is how DFT works)
  - Hence, the first 1+N/2=257 elements *are kept*, while the remaining ones (N/2-1 values) are discarded
- Compute the periodogram estimate of the power spectrum:

$$P_i(k) = \frac{|S_i(k)|^2}{N}, \; k = 0, \ldots, N/2$$

Remember that: k=0 is the DC component (frequency f=0). Then, with DFT the step size is F_s/N (F_s is the *sampling frequency*). Hence, k=1 corresponds to frequency $f_1=F_s/N$ and element k=N/2 corresponds to frequency $f_{N/2} = F_s/N \times N/2 = F_s/2$ (the Nyquist critical frequency, i.e., the highest signal frequency that can be represented by sampling at F_s)

Step 3 – filterbanks (1/3)

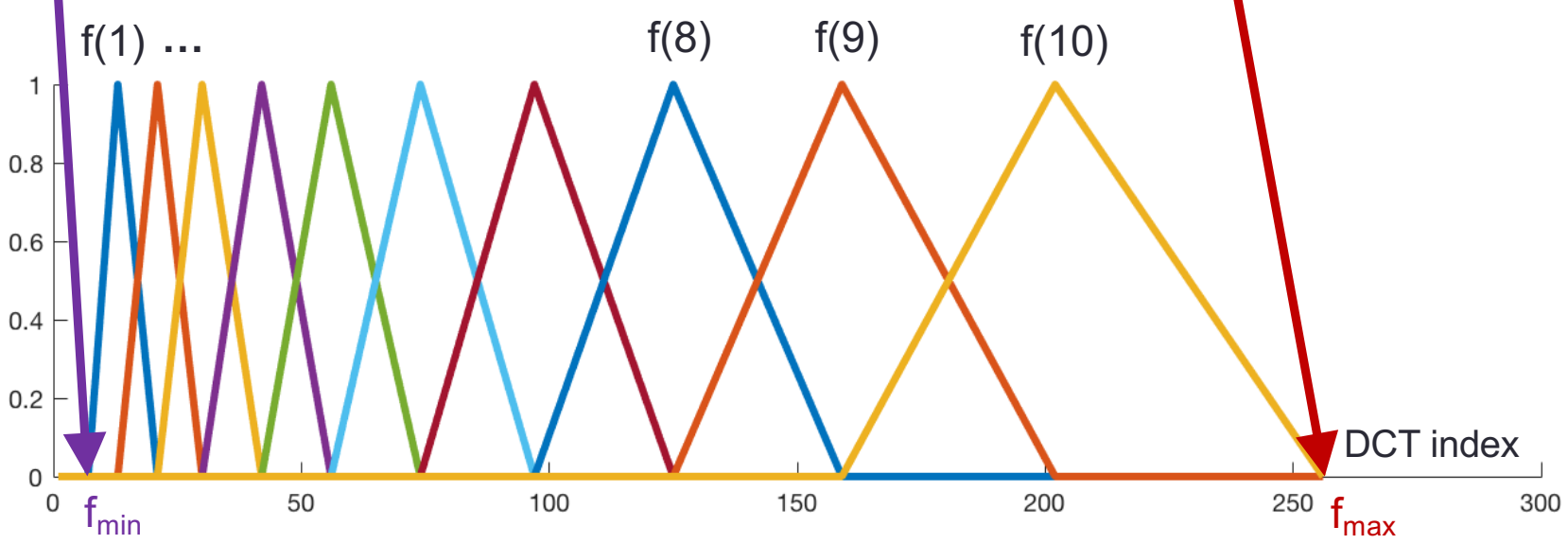
- Obtain N_{fb} (user defined) Mel-spaced triangular filters
  - Pick the number of filters (usually N_{fb}=26)
  - Set the maximum frequency as $f_{max}=F_s/2$
  - Set the minimum frequency as, e.g., f_{min}=300 Hz (user defined)
  - Compute N_{fb} center frequencies f(1), f(2), …, f(N_{fb})
    - Linearly spaced in Mel's domain
    - Non-linearly spaced in frequency domain [Hz]
- Each filter m
  - Is centered at frequency f(m)
  - Is equal to 1 for f(m) and decreases linearly otherwise
  - Is zero at f(m-1), f(m+1) and outside [f(m-1), f(m+1)]
  - N_{fb} filters require N_{fb}+2 thresholds

Step 3 – filterbanks (2/3)

- Example
  - F_s= 20.480 kHz, N=512, N_{fb}=10, $f_{max}=F_s/2$=10,240 Hz, f_{min}=300 Hz
- Frequencies [Hz]

  300=f_{min} 543=f(1) 845=f(2) 1,220=f(3) 1,687=f(4) 2,267=f(5) 2,988=f(6)
  3,883=f(7) 4,997=f(8) 6,381=f(9) 8,102=f(10) 10,240=f_{max} (F_s/2)
- Corresponding DFT indices

  [7 13 21 30 42 56 74 97 125 159 202 256]



S

```
FminMel=1125*log(1+(Fmin/700)); % min Mel's frequency
FmaxMel=1125*log(1+(Fmax/700)); % Max Mel's frequency

Nthresh=Nfb+2;      % number of required threshold values
step=(FmaxMel-FminMel)/(Nthresh-1); % step size (linearly partition Mel's space)
```

- Example
  - F_s= 20.480 kHz, N=512, N
- Frequencies [Hz]

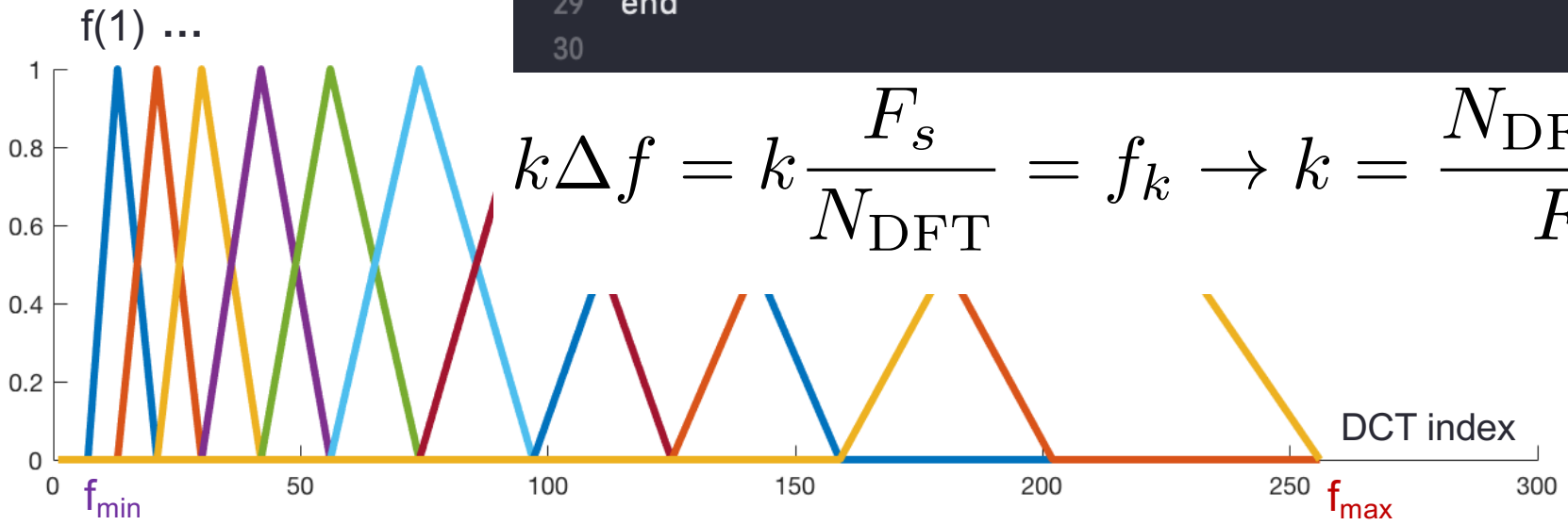
  300=f_{min} 543=f(1) 845=f
  3,883=f(7) 4,997=f(8) 6,
- Corresponding DFT indice

  [7 13 21 30 42 56

```
% for each threshold do
for i = 0:(Nthresh-1)
    % Mel frequencies, evenly distributed
    melvec(i+1) = FminMel+i*step;

    % Hz frequencies, obtained inverting Mel relation
    freqvec(i+1) = 700*(exp(melvec(i+1)/1125)-1);

    % DFT indices, obtained from Hz frequencies
    f(i+1) = floor((nDFT+1)*freqvec(i+1)/Fs);
end
```

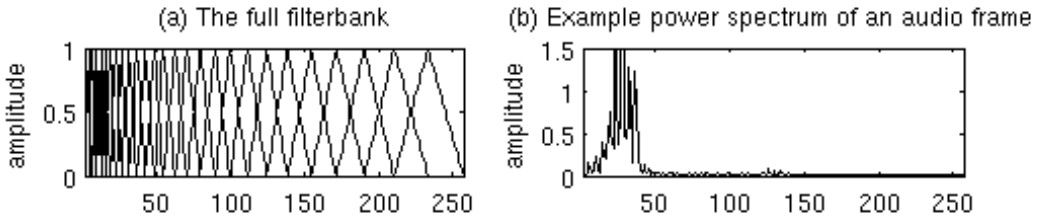


$$k\Delta f = k\frac{F_s}{N_{\mathrm{DFT}}} = f_k \rightarrow k = \frac{N_{\mathrm{DFT}} f_k}{F_s}$$

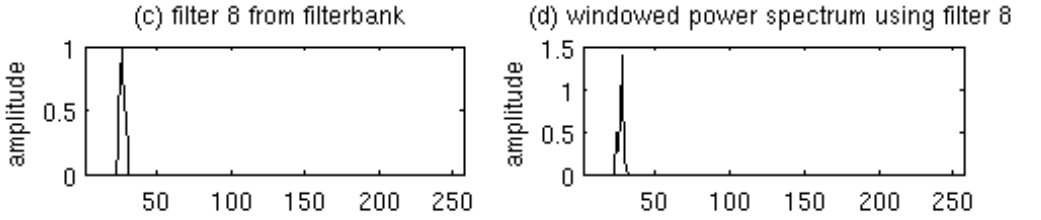
Step 3 – filterbanks (3/3)

- Compute Mel-spaced filterbanks (through Mel's equation)
  - They amounts to 20-40 triangular filters
  - Each filter (in our case) has 257 values (1+N/2, to match the output of step 2) and it is multiplied by the entire power spectrum from step 2

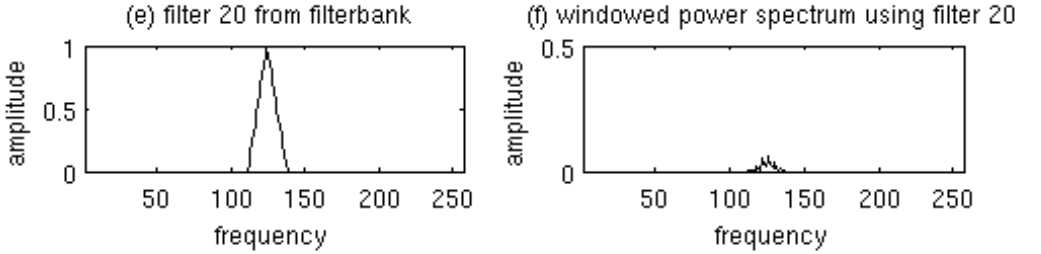
full filterbank (left)
full power spectrum (right)



filter 8 from filterbank (left)
output when this filter is applied (right)



filter 20 from filterbank (left)
output when this filter is applied (right)



Step 4 – logarithms

- We now have N_{fb} Mel-spaced triangular filters
- For each filter m=1,2,…, N_{fb}
  1. Multiply filter m by the full power spectrum from step 2 (the resulting energy trace will be mostly 0 (besides around f(m), the filter acts as a mask, centered at f(m))
  2. Add up the non-zero coefficients in this trace → obtain E_m
  3. Take the logarithm: $\log(E_m)$

- This gives us N_{fb} real values (one for each filterbank):
- $\log(E_1)$, $\log(E_2)$, …, $\log(E_M)$, with $M= N_{fb}$
  - One for each Mel's frequency band
  - They tell how much energy there is within each band

Intermission - DCT

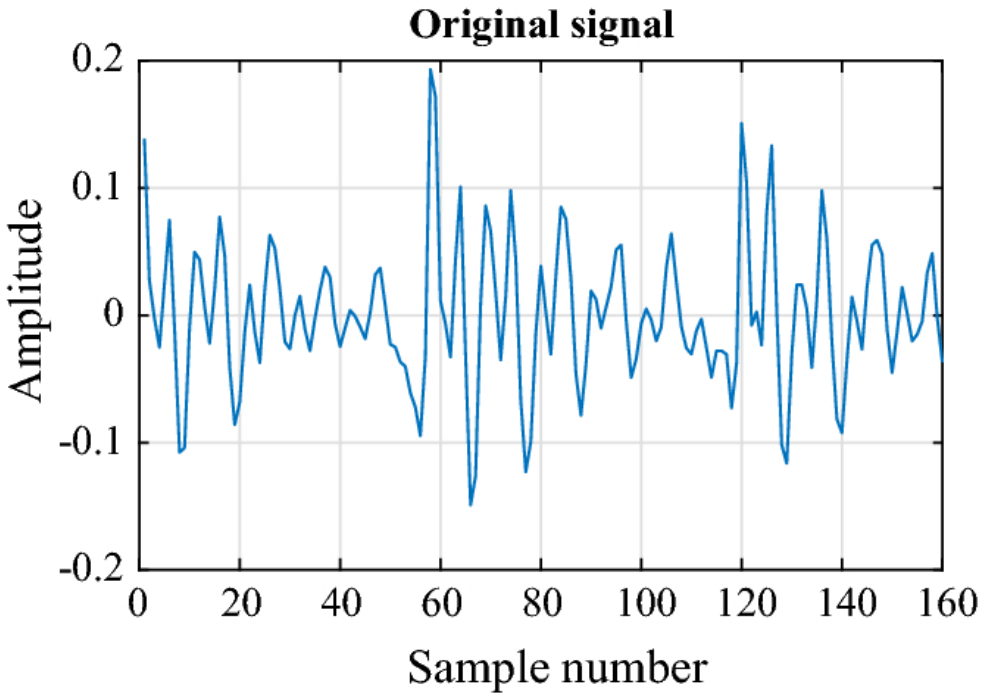
- Discrete Cosine Transform
  - Very popular (and successful) for *audio* and *video* signals
  - Has nice properties: energy compaction, decorrelation
  - Represents a discrete signal (1D vector or 2D matrix)
    - As the sum of cosines (the basis functions)
    - Those cosine functions are orthogonal wrt one another
  - For 1D signals
    - Let x[n] with n=0, …, N-1, be an input vector (time series) of length N
    - The DCT (DCT-2) of x[n] is defined as:

$$C[u] = \alpha(u) \sum_{n=0}^{N-1} x[n] \cos\left(\frac{\pi(2n+1)u}{2N}\right) \quad \alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & u = 0 \\ \sqrt{\frac{2}{N}} & u \neq 0 \end{cases}$$

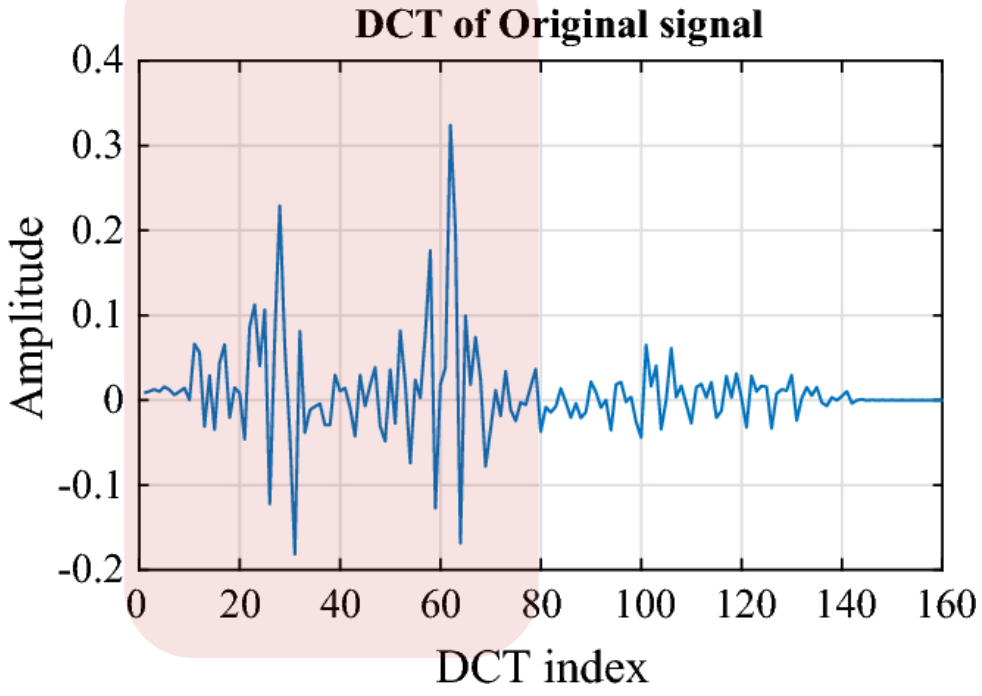
$$u = 0, \ldots, N-1$$

Intermission - DCT

- Energy compaction
  - Audio signal (ampled at 8 kHz)



most significant terms



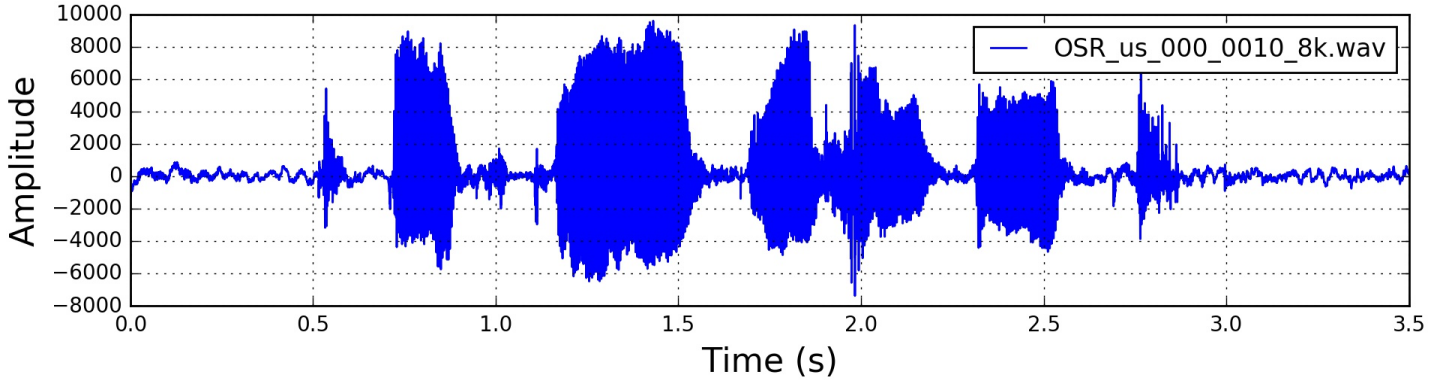
Steps 5 and 6

- Step 5 – for each frame
  - Take the DCT (Discrete Cosine Transform) of the vector containing the logarithms of the energy within each band
  - The result of the DCT is a vector of N_{fb} cepstral coefficients (e.g., N_{fb}=26)
- Step 6 – for each frame
  - For ASR, the cepstral coefficients no. 2,3, …,13 are retained
  - The remaining ones are discarded
  - Note that: the first DCT coefficient is proportional to the sum of all the log-energies computed at the previous step (by the very def. of DCT) – thus, it is an overall measure of signal loudness and is not very informative - it is often discarded for *speech recognition* or *speaker id applications* where the system has to be robust to loudness variations
- Final result of this processing is
  - 12 cepstral coefficients for each frame. For frame i:

$$\boldsymbol{c}(i) = [c_1(i), c_2(i), \ldots, c_{12}(i)]^T$$

Some visual insight (1/2)

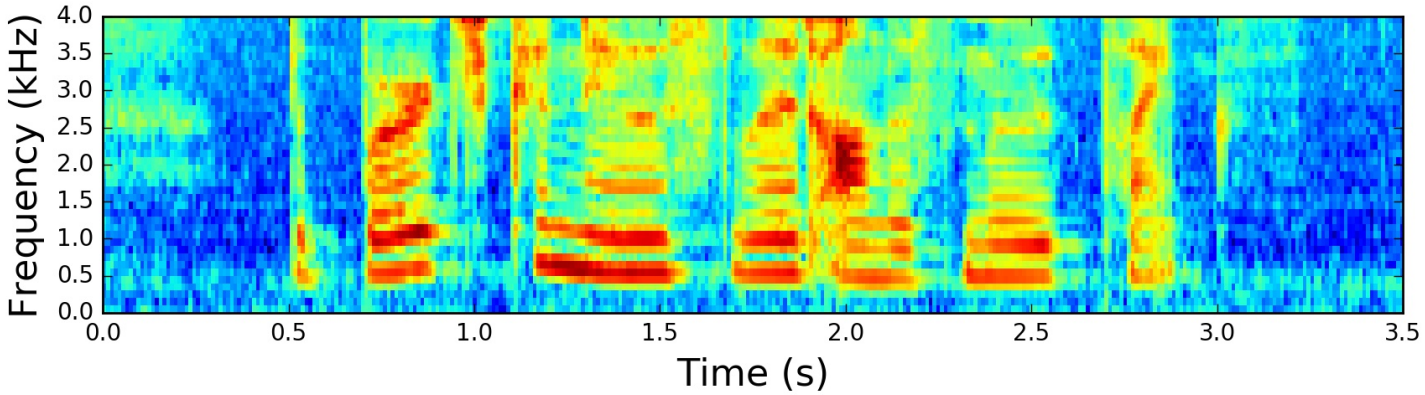
F_s = 8 kHz time signal



Signal in the Time Domain

After applying the filterbank to the power spectrum (periodogram):

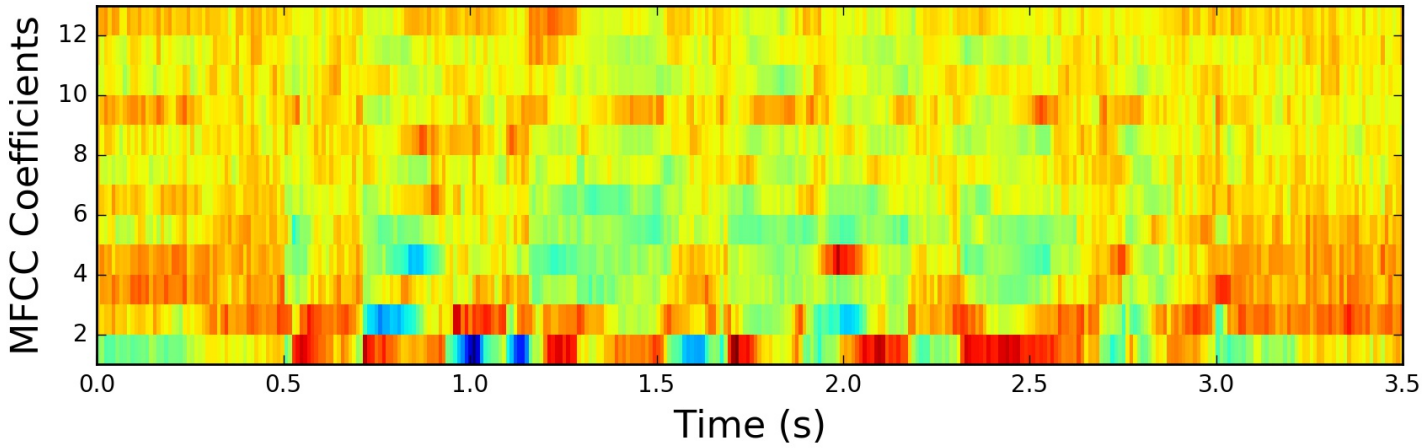
- colors represent log-energies



Spectrogram of the Signal

Some visual insight (2/2)

Mel's Frequency Cepstral Coefficients (MFCCs):

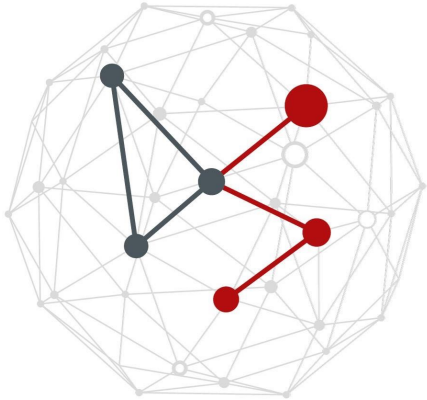


MFCCs

MIME

Master's degree ICT Internet Multimedia Engineering

ADDITIONAL FEATURES



DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DIPARTIMENTO MATEMATICA

UNIVERSITÀ DEGLI STUDI DI PADOVA

Additional features

- Up to now, we got 12 MFCC coefficients, for each frame i:

$$\boldsymbol{c}(i) = [c_1(i), c_2(i), \ldots, c_{12}(i)]^T$$

- Additional features are usually tracked
  1. Energy (1 additional coefficient)
  2. Δ coefficients (12 additional coefficients)
  3. Δ–Δ coefficients (12 additional coefficients)

- Adding these leads to much better ASR performance

Energy

- Let the raw signal (time domain) be represented by $s(n)$
- Where n is the discrete sampling time
- The energy of the generic frame i is defined as:

$$E(s_i) = \sum_{n=n_1(i)}^{n_2(i)} s(n)^2$$

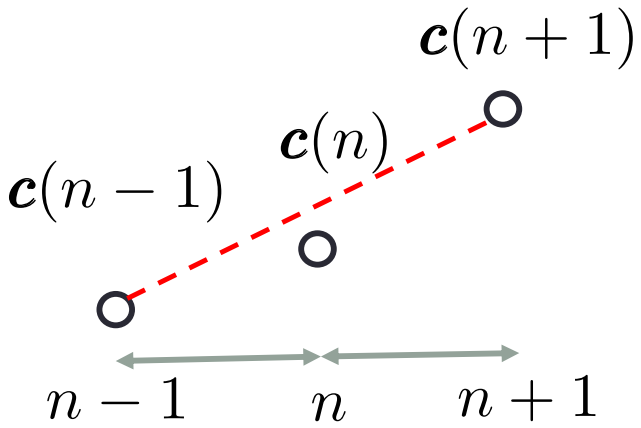
- $n_1(i)$ and $n_2(i)$ are first and last audio samples in frame i
- Sometimes, logarithms are used to limit dynamics

$$\log_{10} E(s_i)$$

Delta coefficients (1/2)

- MFCC feature vector
  - Only describes the power spectrum of a single frame
  - However, speech also bears information in the dynamics, i.e.,
  - trajectories of MFCC coefficients over time
- Delta MFCC coefficients
  - Represent local derivatives of MFCC coefficients ("velocity" of s(n))
  - Append those coefficients to the MFCC feature vector
  - A delta is computed for each element of the MFCC feature vector
    - 12 MFCC elements → 12 Δ coefficients
    - At sampling time n we have:

$$\boldsymbol{\Delta}(n) = \frac{\boldsymbol{c}(n+1) - \boldsymbol{c}(n-1)}{2}$$



Delta coefficients (2/2)

- Common regression formula (e.g., used by HTK)

$$\boldsymbol{\Delta}(n) = \frac{\sum_{m=1}^{M} m(\boldsymbol{c}(n+m) - \boldsymbol{c}(n-m))}{2\sum_{m=1}^{M} m^2}, \text{ usually } M = 2$$

Delta-Delta coefficients

- Trajectory in the Delta coefficients ("acceleration" of s(n))
- So the new estimation formula is:

$$\boldsymbol{\Delta}^2(n) = \frac{\boldsymbol{\Delta}(n+1) - \boldsymbol{\Delta}(n-1)}{2}$$

- In practice, the following is implemented:

$$\boldsymbol{\Delta}^2(n) = \frac{\sum_{m=1}^{M} m(\boldsymbol{\Delta}(n+m) - \boldsymbol{\Delta}(n-m))}{2\sum_{m=1}^{M} m^2}, \text{ usually } M = 2$$

Full feature vector

- For each frame:
  - 12 MFCC coefficients
  - 12 Delta coefficients
  - 12 Delta-Delta coefficients
  - 1 signal energy coefficient
  - 1 Delta energy coefficient
  - 1 Delta-Delta energy coefficient

- Total: 39 coefficients
  - Used in most speech recognition applications

$$\begin{bmatrix} c_1 \\ \vdots \\ c_{12} \\ \hline \Delta_1 \\ \vdots \\ \Delta_{12} \\ \hline \Delta_1^2 \\ \vdots \\ \Delta_{12}^2 \\ \hline E(s) \\ E(\boldsymbol{\Delta}) \\ E(\boldsymbol{\Delta}^2) \end{bmatrix}$$

Implementations

- HTK (Hidden Markov Model Toolkit) for speech recognition toolkit

http://htk.eng.cam.ac.uk/

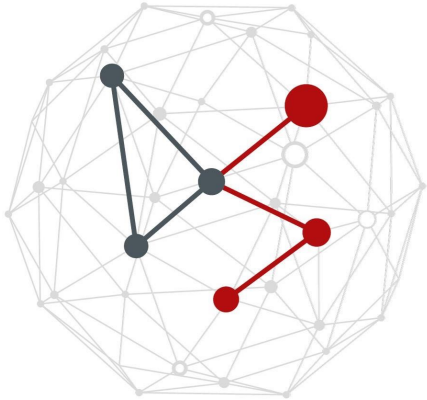
- Phyton library for music and audio analysis
- Extract MFCCs (and other features too)

https://librosa.org/

MIME

Master's degree ICT Internet Multimedia Engineering

AUTOMATIC SPEECH RECOGNITION SYSTEMS



DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DIPARTIMENTO MATEMATICA

UNIVERSITÀ DEGLI STUDI DI PADOVA

Automatic Speech Recognition Systems (ASRS): some history

- Initial work at Carnergie Mellon & IBM in the 1970's
  - Discrete density HMM (to model observations)
- Subsequent work @ Bell Labs
  - Continuous density HMM (using Gaussian Mixture Models, GMM)
- Initially
  - Speaker *dependent* large vocabularies
  - Speaker *independent* small vocabularies
- In the mid-1990's
  - Large speaker independent vocabularies
  - DARPA funded several research programs (since 1971), e.g., the "1,000 words vocabulary challenge"
  - Worthy of note: the HTK Software Toolkit (Cambridge University)

The statistical HMM approach

- Main tool was the HMM
  - Learning is based on forward-backward algo
  - Also called the "Baum-Welch algorithm"
  - Named after Leonard Esau-Baum, a mathematician that developed HMM algorithms at the Institute for Defense Analyses (IDA), Priceton University, in the late 1960's

Leonard E. Baum
(August 23, 1931 – August 14, 2017)

1980 – 1990: HMM become ubiquitous

HMMs become extensively used

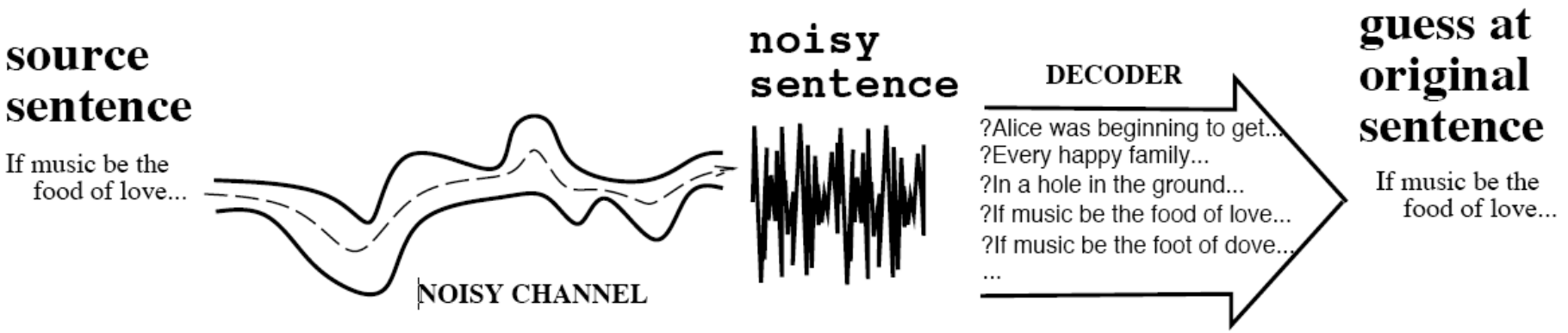
- Lawrence R. Rabiner
  - wrote a wonderful tutorial
  - pushes on ASR research using HMMs

R. V. Cox, C. A. Kamm, L. R. Rabiner, J. Schroeter, and J. G. Wilpon, Speech and Language Processing for Next-Millennium Communications Services, Proceedings of the IEEE, Vol. 88, No. 8, August 2000.

L. R. Rabiner, A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, Proceeding of the IEEE, Vol. 77, No. 2, February 1989.

Lawrence R. Rabiner
(born 28 September 1943)

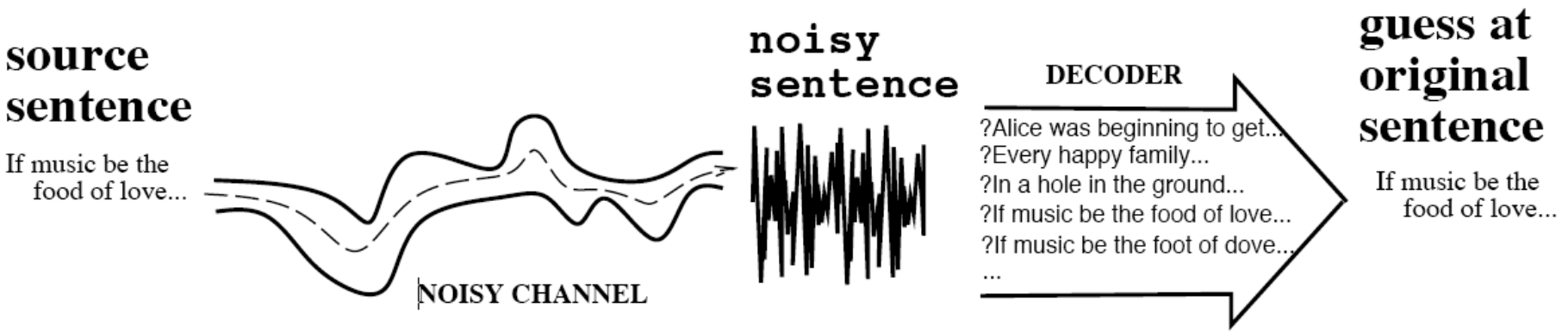
The noisy channel model



- Idea: treat the acoustic waveform as a "noisy" version of the string of words that was spoken, i.e., a string that has been passed through a *noisy communication channel*
- Goal: is to build a model of the noisy channel so that we can figure out how it modifies the true sentence and, in turn, retrieve it
- Insight: if we know how the channel distorts the source, we could find the true source sentence by:

(i) taking every possible sentence in the language

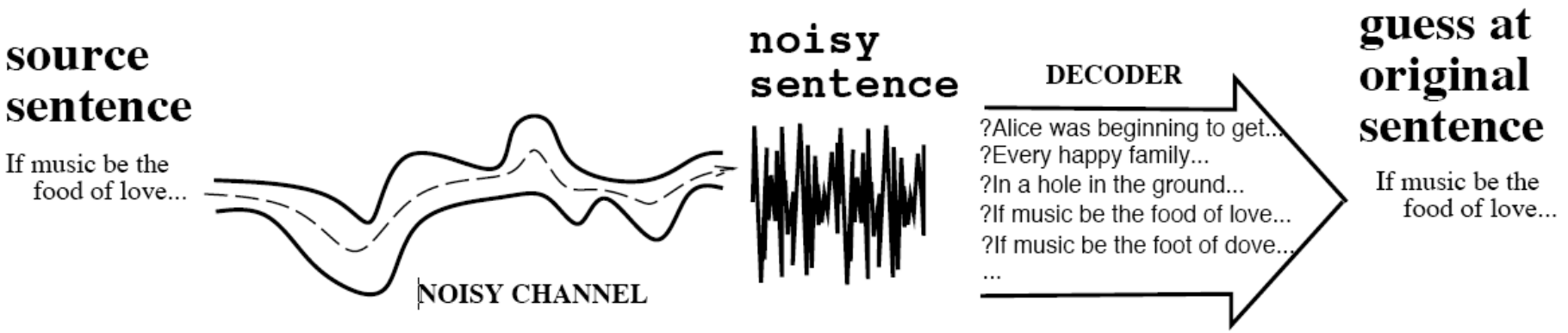
The noisy channel model



- Idea: treat the acoustic waveform as a "noisy" version of the string of words that was spoken, i.e., a string that has been passed through a *noisy communication channel*
- Goal: is to build a model of the noisy channel so that we can figure out how it modifies the true sentence and, in turn, retrieve it
- Insight: if we know how the channel distorts the source, we could find the true source sentence by:

(ii) running each sentence through our noisy channel model

The noisy channel model

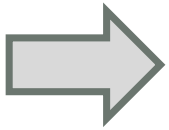


- Idea: treat the acoustic waveform as a "noisy" version of the string of words that was spoken, i.e., a string that has been passed through a *noisy communication channel*
- Goal: is to build a model of the noisy channel so that we can figure out how it modifies the true sentence and, in turn, retrieve it
- Insight: if we know how the channel distorts the source, we could find the true source sentence by:

(iii) seeing whether it matches the output: we then select the best matching source sentence as our desired source sentence

Decoding

- We need a complete metric for a “best match”
- Problem: speech is very variable
  - No acoustic model will provide an exact match for a given sentence
- Solution: use probabilistic models
  - Speech recognition as a special case of Bayesian inference
- Goal: to combine various probabilistic models to get a complete estimate for the probability of a noisy acoustic observation-sequence *given a candidate source sentence*. We can then search through the space of all sentences, and choose the one with the highest probability



DECONDING and SEARCH

ARS: the challenge

1) Record a speech waveform representing a sentence

2) Feature extraction: *input speech waveform* is converted into a number of *feature vectors* (e.g., MFCCs, obtained every 25ms, spaced 10 ms apart)

acoustic input $\boldsymbol{Y}_{1:T} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_T\}$

feature vectors

What is the most likely sentence out of all sentences in the language $\mathcal{L}$ given some acoustic input $\boldsymbol{Y}_{1:T}$?

ARS (1/3)

3) Decoder: attempts to find the sequence of words

$$\boldsymbol{w}_{1:L} = \{w_1, w_2, \ldots, w_L\}$$

which is most likely to have generated $\boldsymbol{Y}_{1:T}$
(subscripts dropped for the sake of notation compactness)

$$\tilde{\boldsymbol{w}} = \underset{\boldsymbol{w} \in \mathcal{L}}{\operatorname{argmax}}\, P(\boldsymbol{w}|\boldsymbol{Y})$$

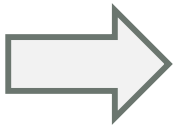
ARS (2/3)

$$\tilde{\boldsymbol{w}} = \underset{\boldsymbol{w} \in \mathcal{L}}{\operatorname{argmax}} P(\boldsymbol{w}|\boldsymbol{Y})$$

- This equation returns the optimal sentence estimate
- But it is very difficult to use, we need to make it operational:

$$\tilde{\boldsymbol{w}} = \underset{\boldsymbol{w} \in \mathcal{L}}{\operatorname{argmax}} P(\boldsymbol{w}|\boldsymbol{Y}) = \underset{\boldsymbol{w} \in \mathcal{L}}{\operatorname{argmax}} \frac{P(\boldsymbol{Y}|\boldsymbol{w})P(\boldsymbol{w})}{P(\boldsymbol{Y})}$$

- P(**w**): prob. of the word string, estimated by an N-gram language model
- P(**Y**|**w**): can be easily estimated as well (HMM or Neural Networks)
- P(**Y**): very difficult to estimate, but we do not need it. As it is a constant term that appears for each **w**.



The search for a maximizer does not depend on it!

ARS (3/3)

$$\tilde{\boldsymbol{w}} = \underset{\boldsymbol{w} \in \mathcal{L}}{\operatorname{argmax}} \overbrace{P(\boldsymbol{Y}|\boldsymbol{w})}^{\text{likelihood}} \overbrace{P(\boldsymbol{w})}^{\text{prior}}$$

- The prior P(w) is computed through the language model
- The likelihood P(**Y**|**w**) is computed by the acoustic model (HMM or Neural Networks)

Master's degree ICT Internet Multimedia Engineering

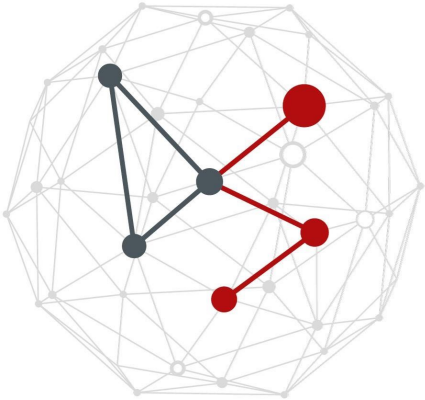
SPEECH MODELS AND AUTOMATIC SPEECH RECOGNITION (ASR) SYSTEMS

Michele Rossi

michele.rossi@dei.unipd.it

Dept. of Information Engineering

University of Padova, IT



DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

DIPARTIMENTO MATEMATICA

UNIVERSITÀ DEGLI STUDI DI PADOVA