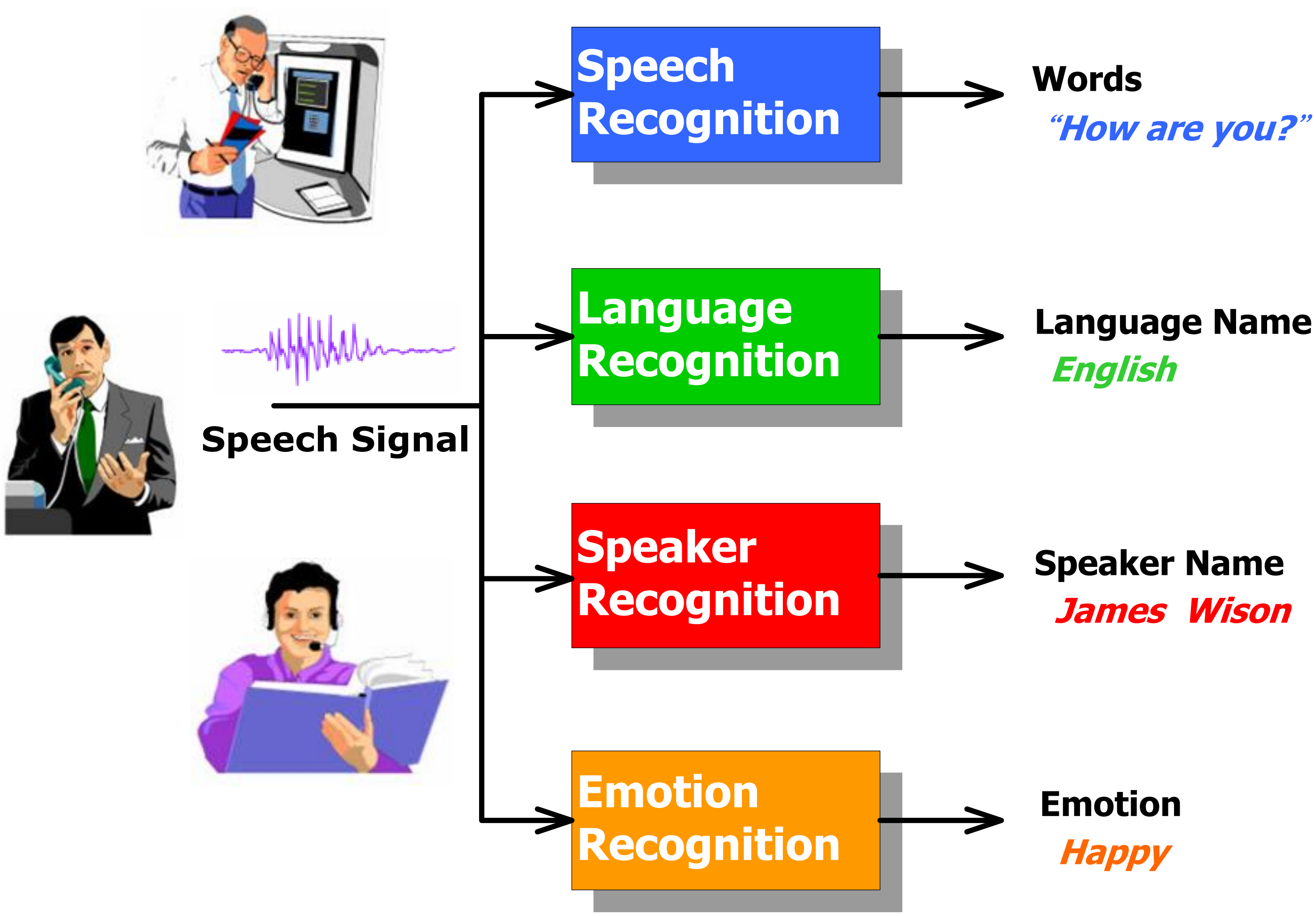
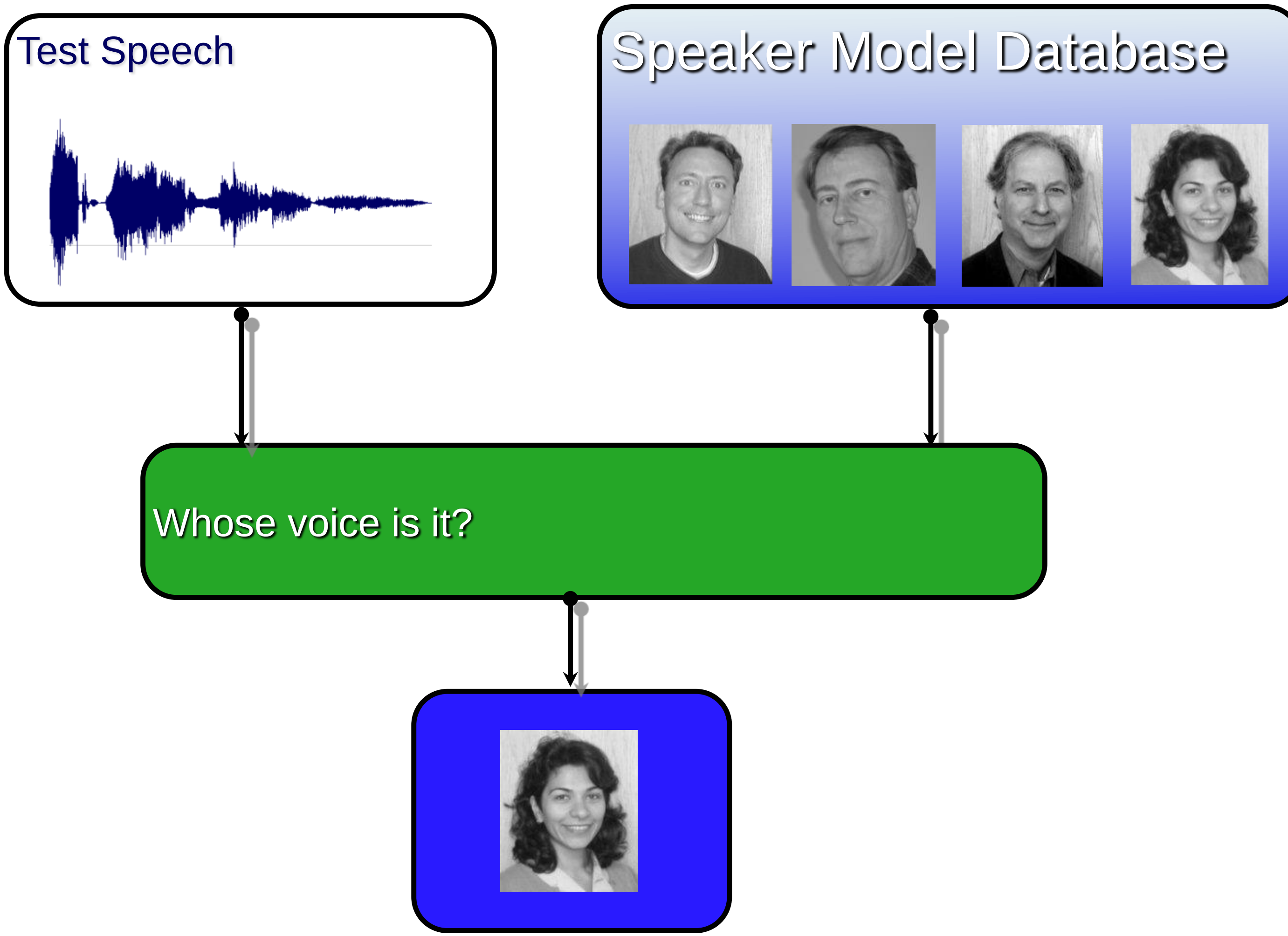
Speaker Segmentation

Speech Processing



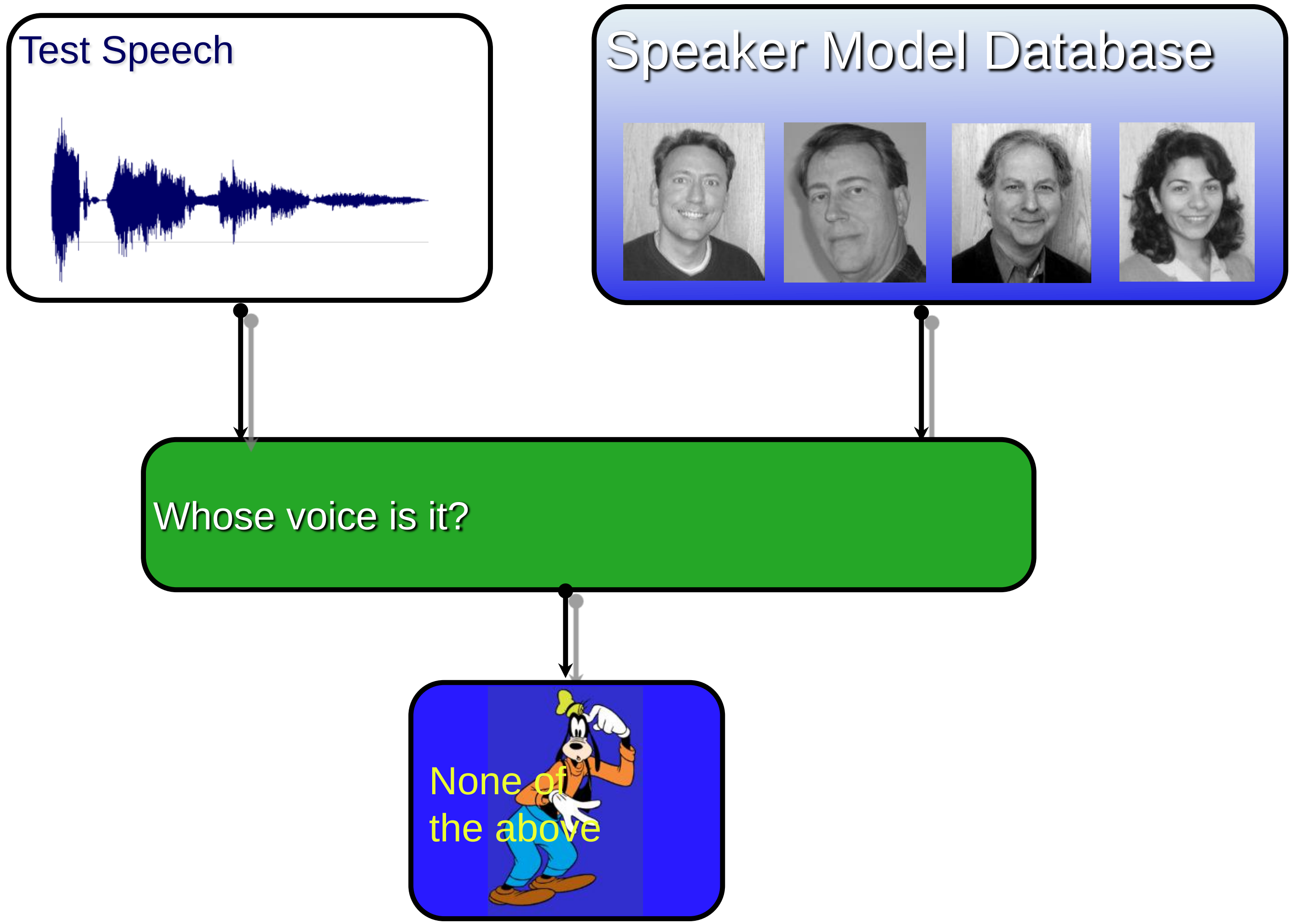
Identification

Closed-set Speaker Identification

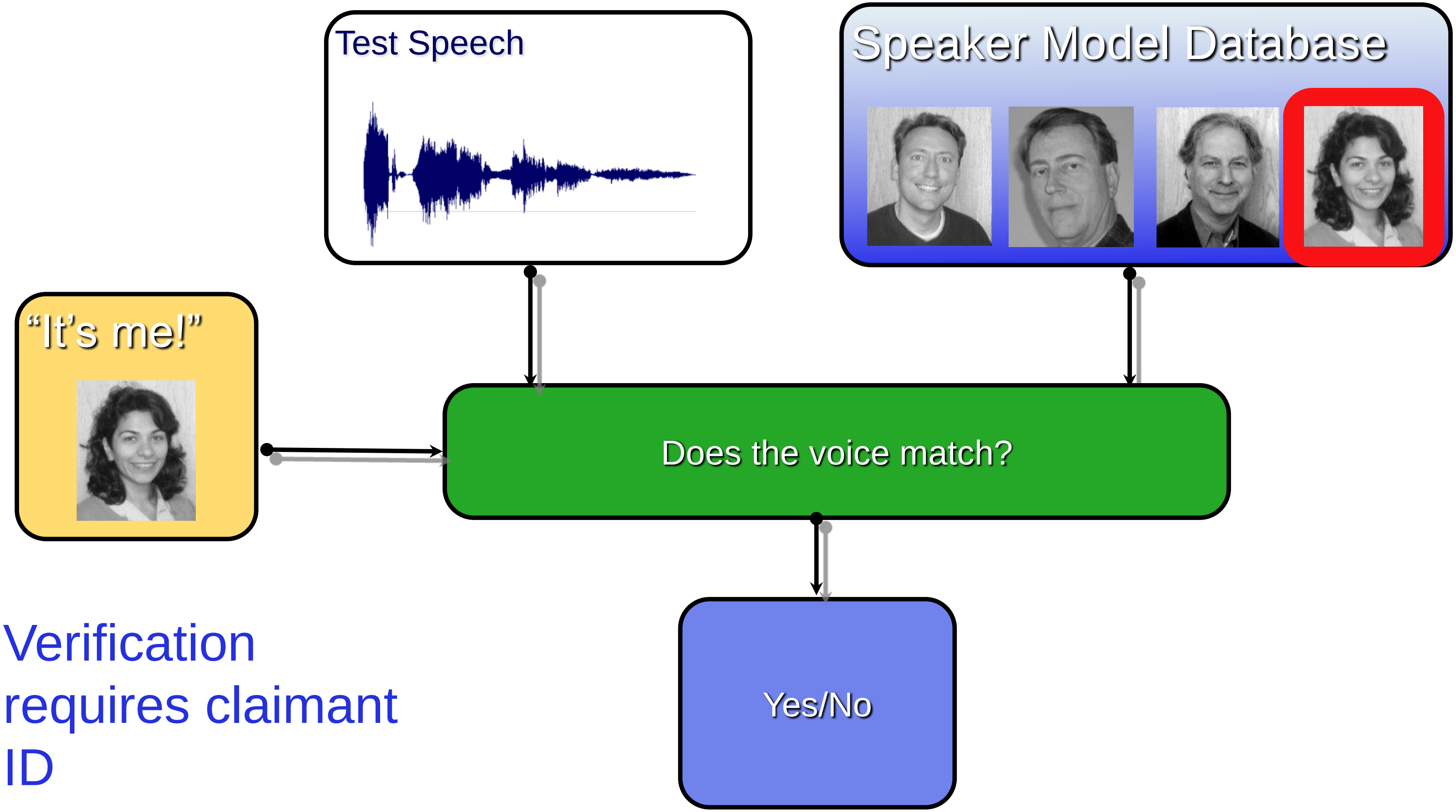


Identification

Open-set
Speaker
Identification

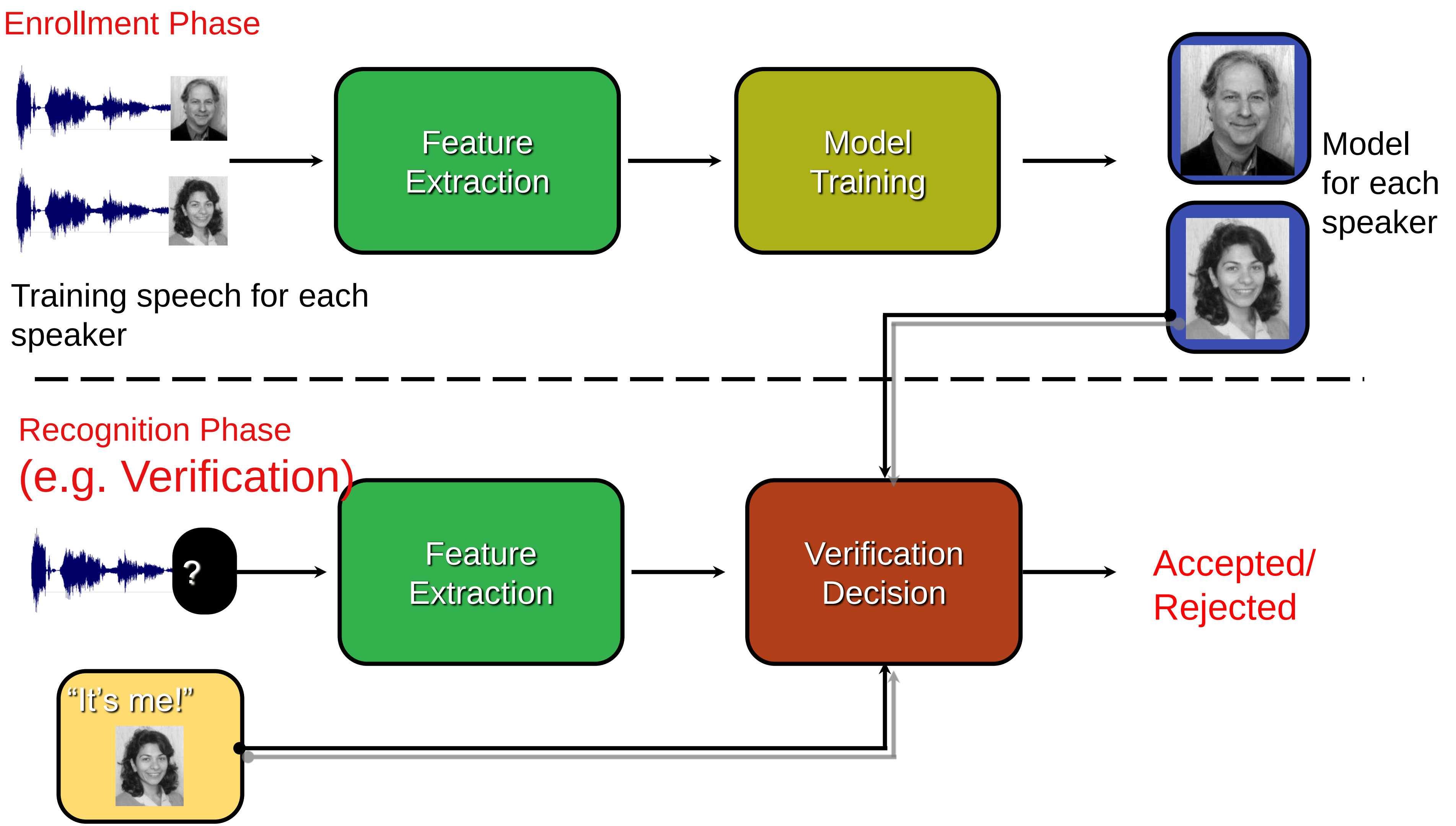


Verification/Authentication/Detection



Verification requires claimant ID

Training & Test Phases

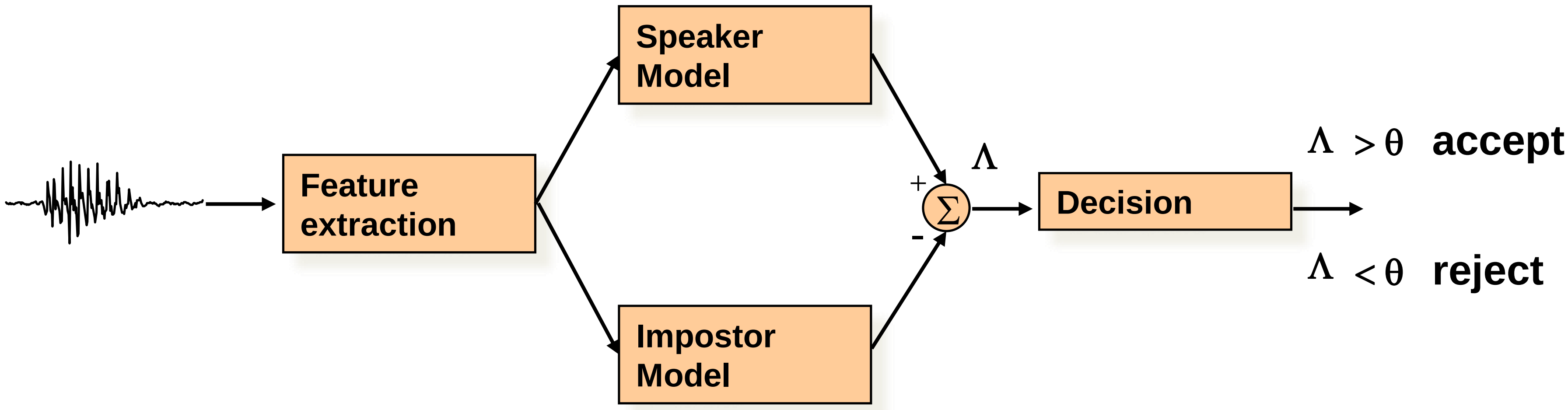


Decision making

- Verification decision approaches have roots in signal detection theory
  - **2-class Hypothesis test:**

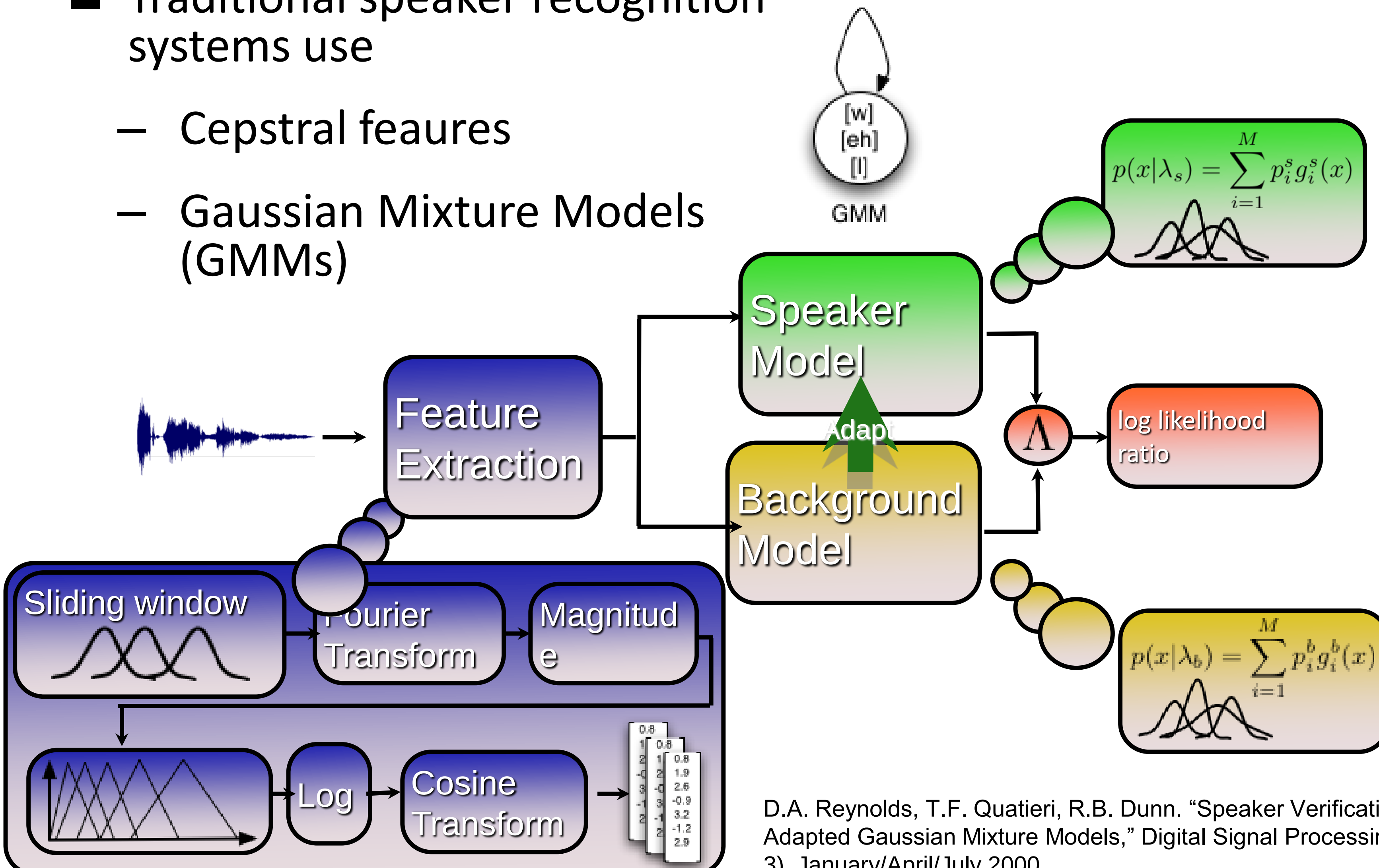
    H0: the speaker is an impostor
    H1: the speaker is indeed the claimed speaker.
  - **Statistic computed on test utterance S as likelihood ratio:**

$$\Lambda = \log \frac{\text{Likelihood } \mathbf{S} \text{ came from speaker model}}{\text{Likelihood } \mathbf{S} \text{ did } \underline{\text{not}} \text{ come from speaker model}}$$



Spectral Based Approach

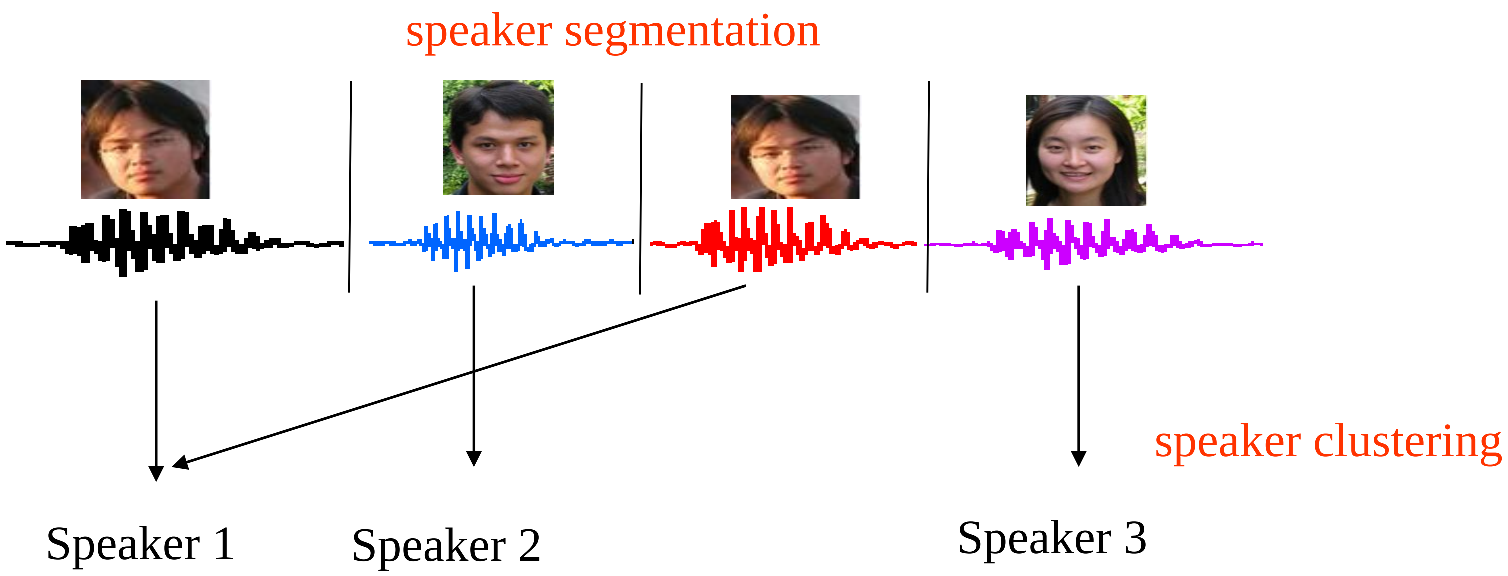
- Traditional speaker recognition systems use
  - Cepstral feaures
  - Gaussian Mixture Models (GMMs)



D.A. Reynolds, T.F. Quatieri, R.B. Dunn. "Speaker Verification using Adapted Gaussian Mixture Models," Digital Signal Processing, 10(1--3), January/April/July 2000

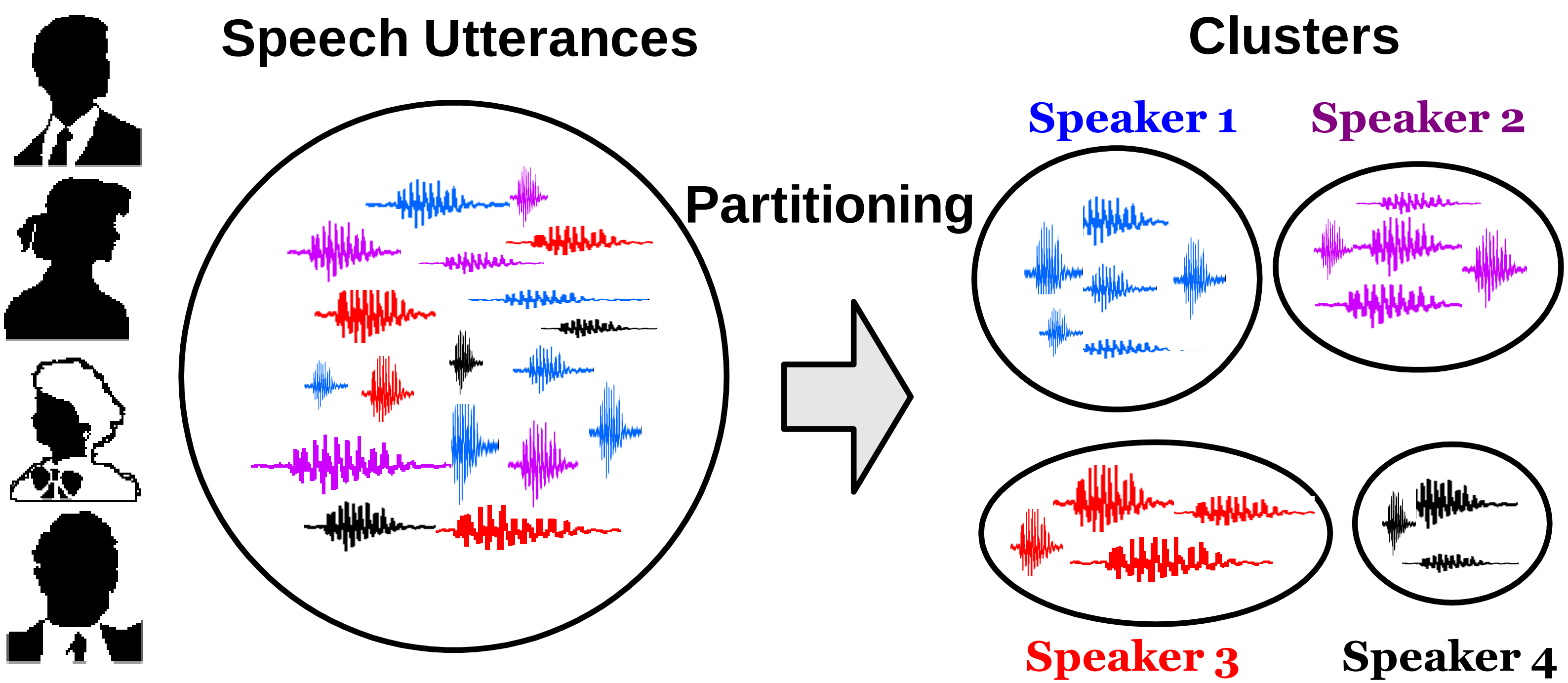
Speaker diarization

■ Problem formulation: the “who spoke when” task on an continuous audio stream



Speaker clustering

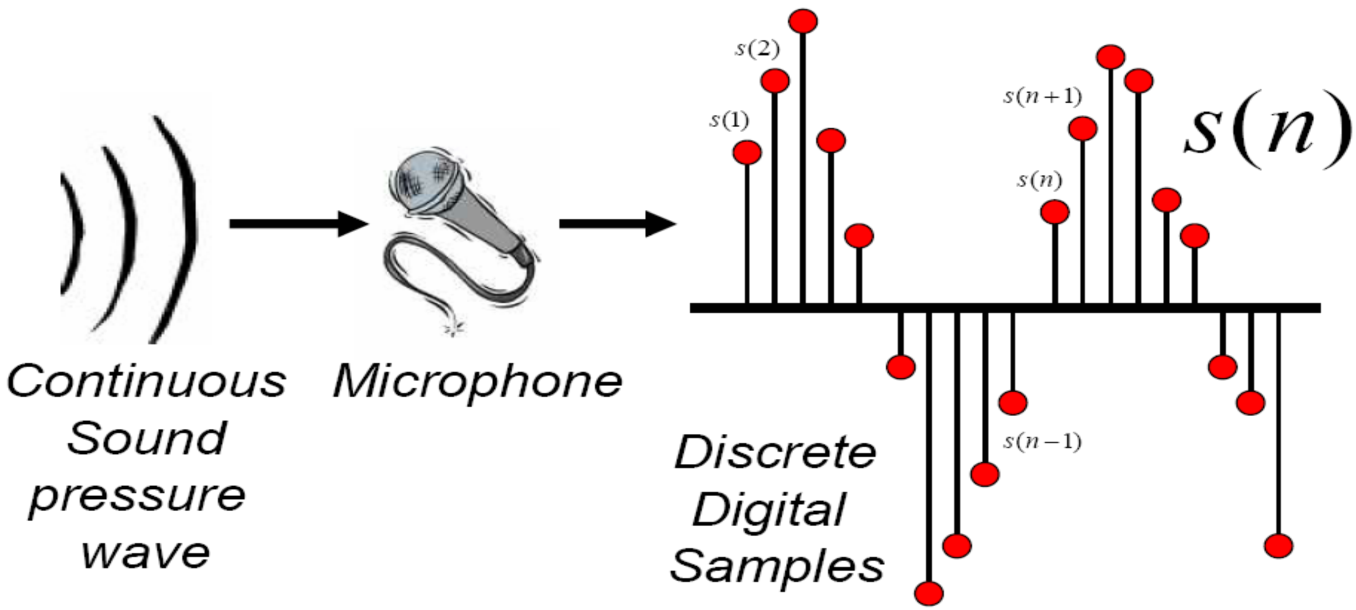
- Problem formulation



  - given N speech utterances from P unknown speakers, partition these utterances into M clusters, such that $M = P$ and each cluster consists exclusively of utterances from only one speaker

Discrete Representation of Signal

- Represent continuous signal into discrete form.



Digitizing the signal (A-D)

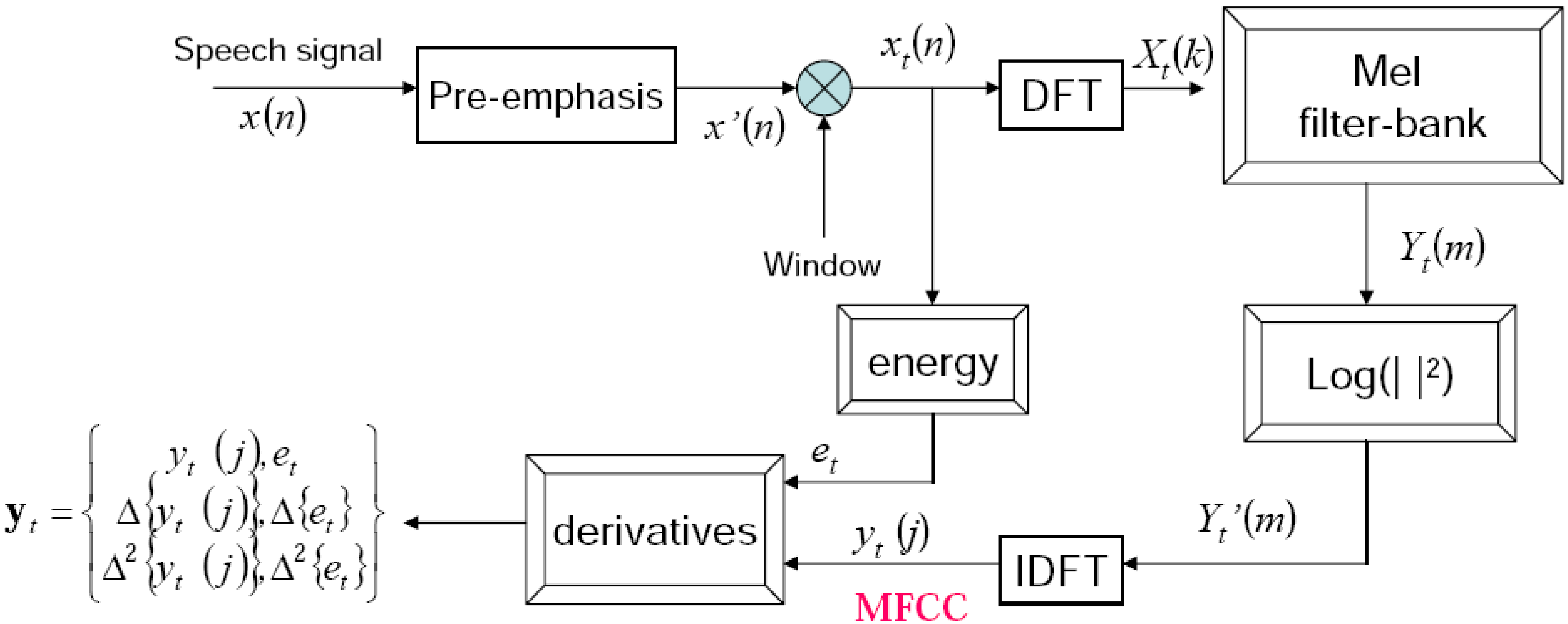
- **Sampling**:
  - measuring amplitude of signal at time t
  - 16,000 Hz (samples/sec) Microphone ("Wideband"):
  - 8,000 Hz (samples/sec) Telephone
  - Why?
    - Need at least 2 samples per cycle
    - max measurable frequency is half sampling rate
    - Human speech < 10,000 Hz, so need max 20K
    - Telephone filtered at 4K, so 8K is enough

Digitizing Speech (II)

- **Quantization**
  - Representing real value of each amplitude as integer
  - 8-bit (-128 to 127) or 16-bit (-32768 to 32767)
- **Formats**:
  - 16 bit PCM
  - 8 bit mu-law; log compression

MFCC

- Mel-Frequency Cepstral Coefficient (MFCC)
  - Most widely used spectral representation in Automatic Speech Recognition (ASR)

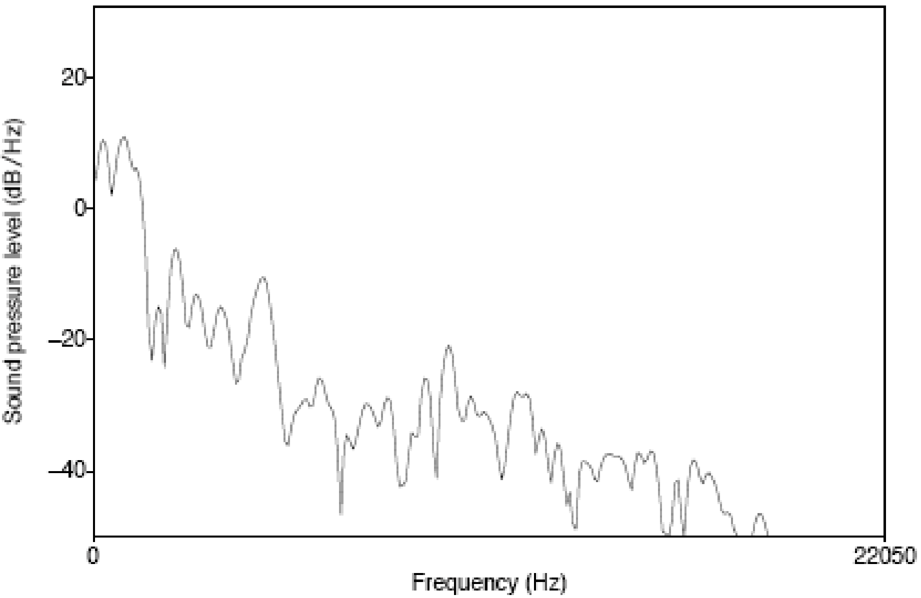


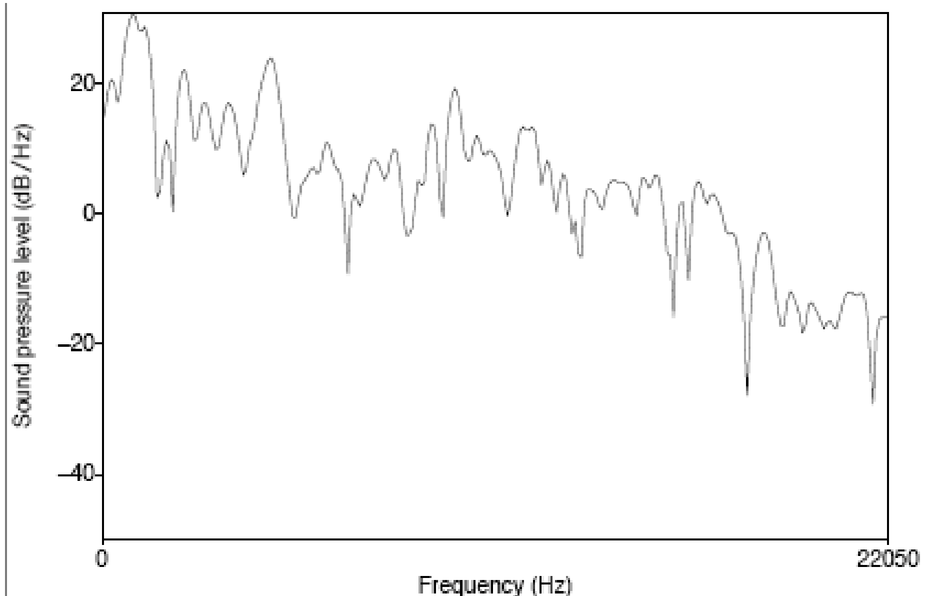
Pre-Emphasis

- Pre-emphasis: boosting the energy in the high frequencies
- Q: Why do this?
- A: The spectrum for voiced segments has more energy at lower frequencies than higher frequencies.
  - This is called **spectral tilt**
  - Spectral tilt is caused by the nature of the glottal pulse
- Boosting high-frequency energy gives more info to Acoustic Model
  - Improves phone recognition performance

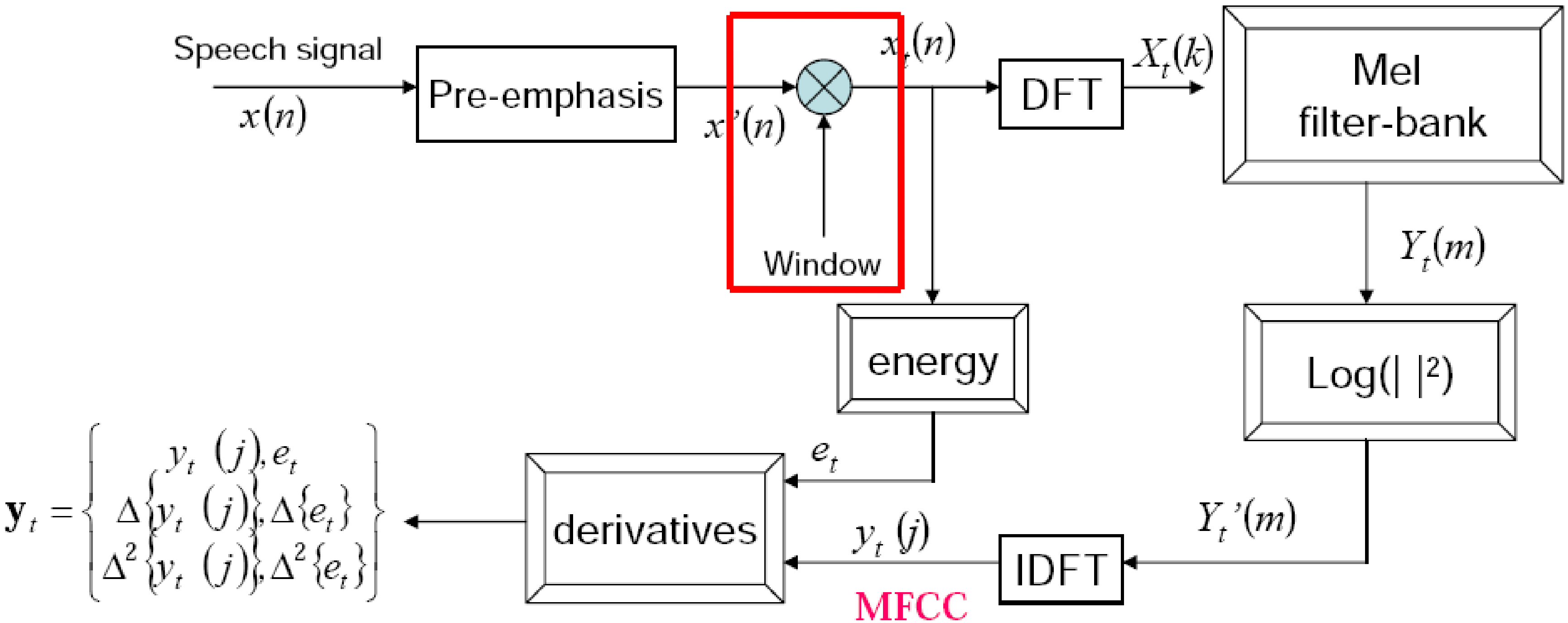
Example of pre-emphasis

- Before and after pre-emphasis
  - Spectral slice from the vowel [aa]



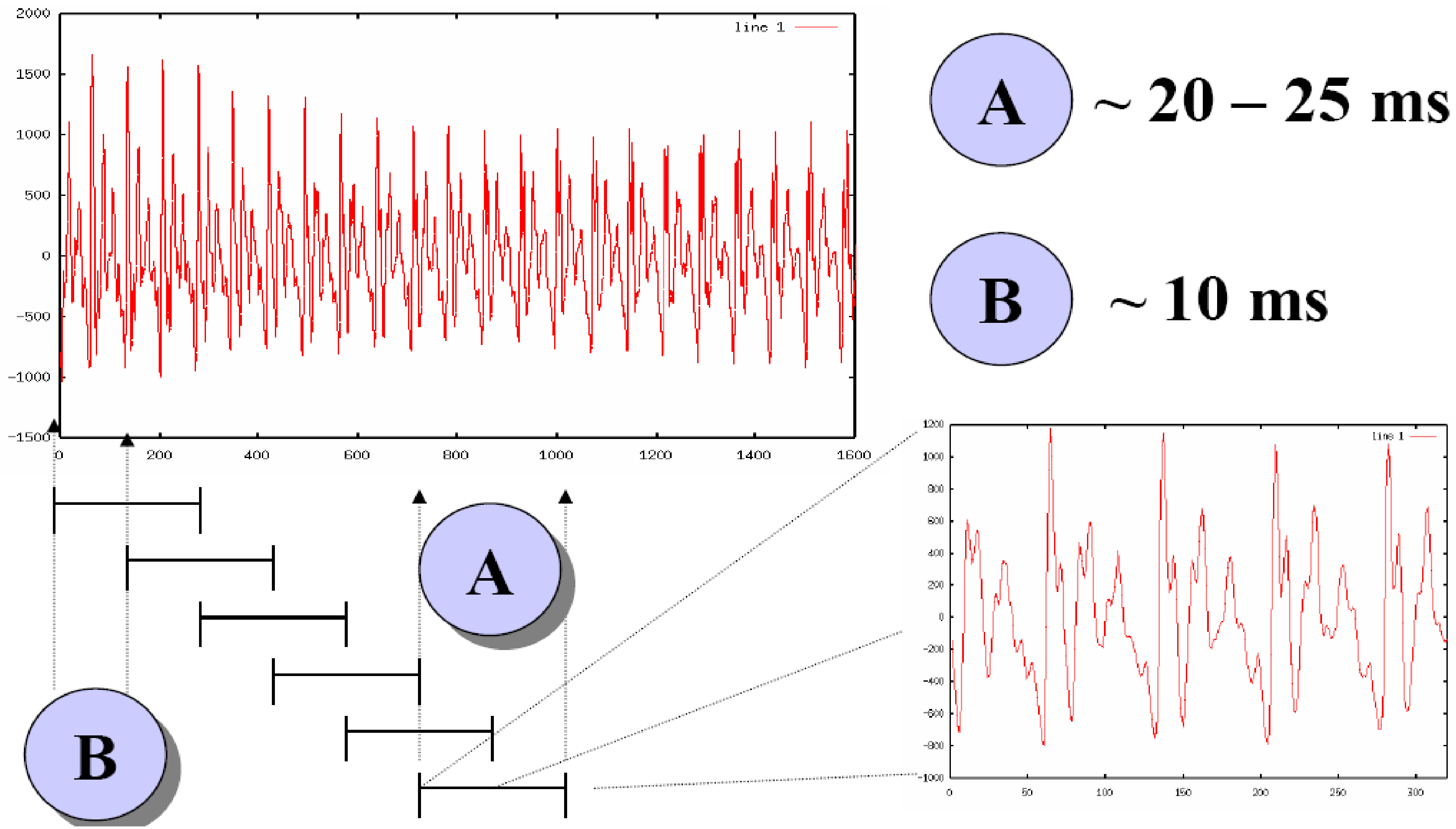


MFCC



Speech signal
$x(n)$
Pre-emphasis
$x'(n)$
Window
$x_t(n)$
DFT
$X_t(k)$
Mel filter-bank
$Y_t(m)$
Log(| |2)
$Y_t'(m)$
IDFT
$y_t(j)$
MFCC
energy
e_t
derivatives
$\mathbf{y}_t = \left\{ \begin{array}{c} y_t(j), e_t \\ \Delta\{y_t(j)\}, \Delta\{e_t\} \\ \Delta^2\{y_t(j)\}, \Delta^2\{e_t\} \end{array} \right\}$

Windowing



Windowing

- Why divide speech signal into successive overlapping frames?
  - Speech is not a stationary signal; we want information about a small enough region that the spectral information is a useful cue.
- Frames
  - Frame size: typically, 10-25ms
  - Frame shift: the length of time between successive frames, typically, 5-10ms

Common window shapes

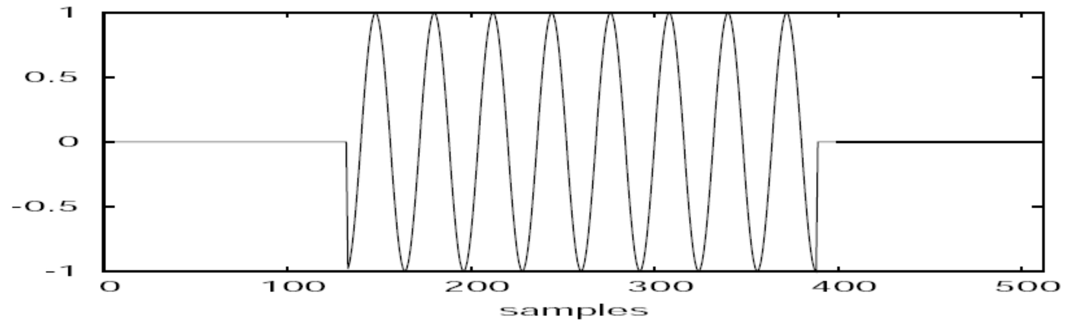
- Rectangular window:

$$w[n] = \begin{cases} 1 & 0 \leq n \leq L-1 \\ 0 & \text{otherwise} \end{cases}$$

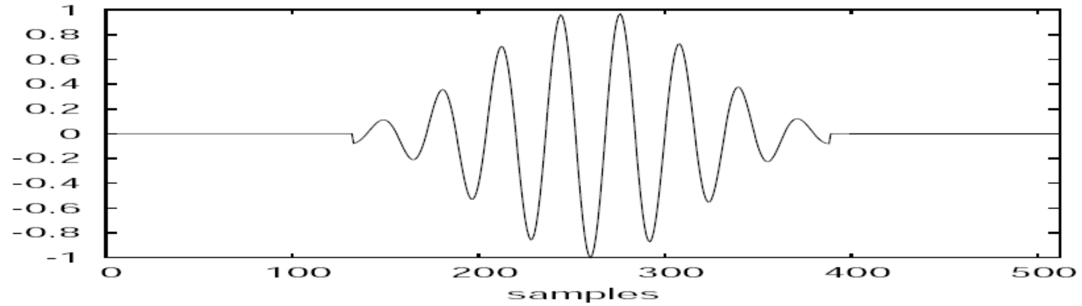
- Hamming window

$$w[n] = \begin{cases} 0.54 - 0.46\cos\left(\frac{2\pi n}{L-1}\right) & 0 \leq n \leq L-1 \\ 0 & \text{otherwise} \end{cases}$$

Window in time domain

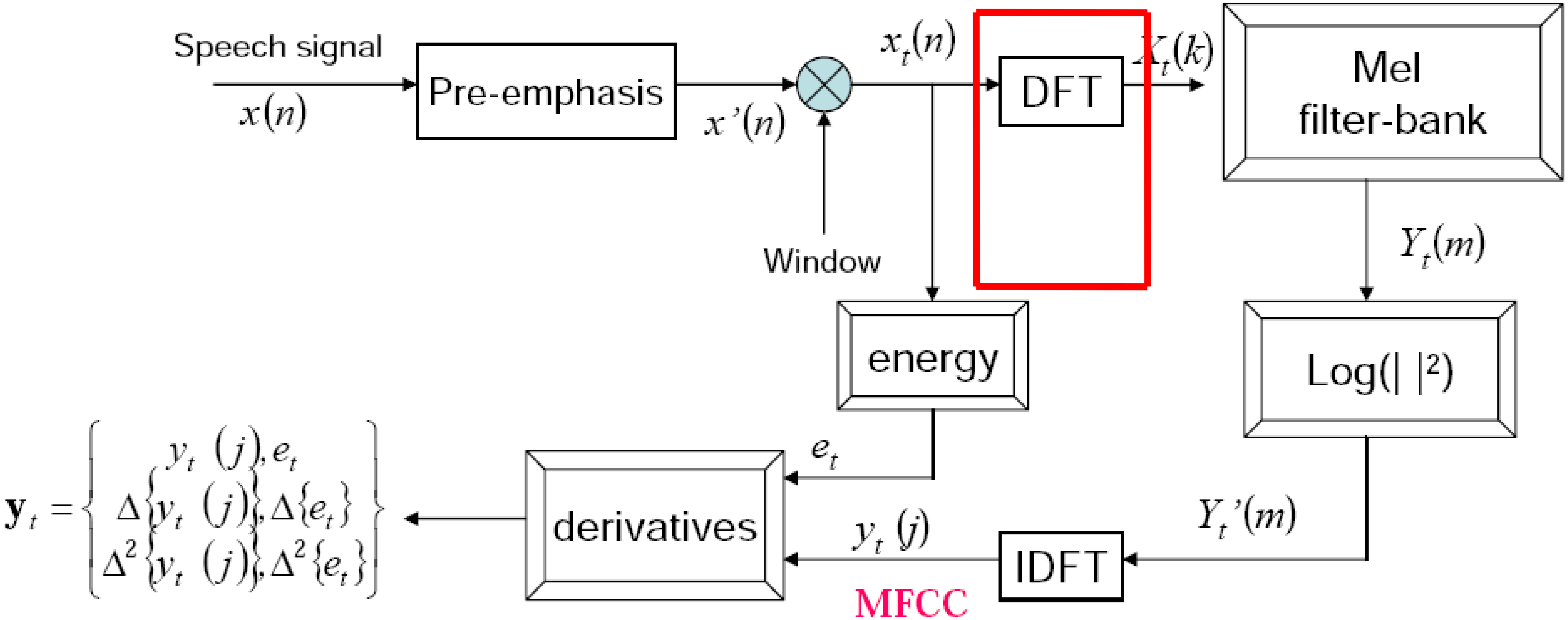


(a) Rectangular window



(c) Hamming window

MFCC



Speech signal
$x(n)$
Pre-emphasis
$x'(n)$
Window
$x_t(n)$
DFT
$X_t(k)$
Mel filter-bank
$Y_t(m)$
energy
Log(| |2)
e_t
$Y_t'(m)$
derivatives
$y_t(j)$
IDFT
MFCC
$\mathbf{y}_t = \left\{ \begin{array}{c} y_t(j), e_t \\ \Delta\{y_t(j)\}, \Delta\{e_t\} \\ \Delta^2\{y_t(j)\}, \Delta^2\{e_t\} \end{array} \right\}$

Discrete Fourier Transform

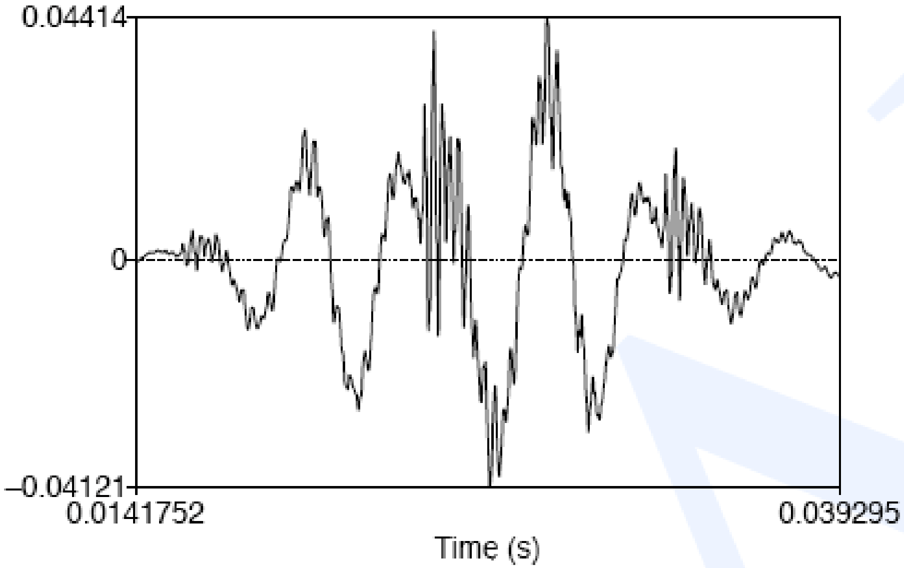
- Input:
  - Windowed signal x[n]…x[m]
- Output:
  - For each of N discrete frequency bands
  - A complex number X[k] representing magnidue and phase of that frequency component in the original signal
- Discrete Fourier Transform (DFT)

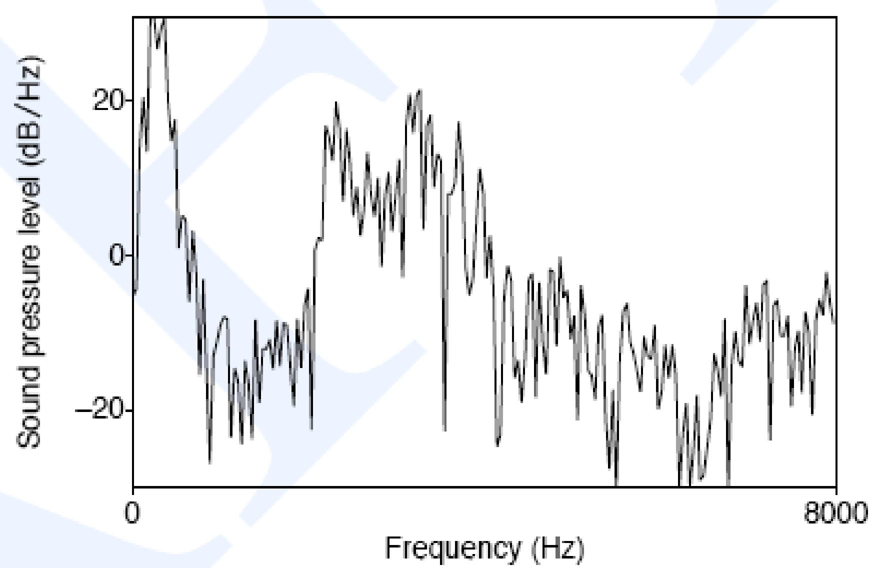
$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\frac{\pi}{N}kn}$$

- Standard algorithm for computing DFT:
  - Fast Fourier Transform (FFT) with complexity N*log(N)
  - In general, choose N=512 or 1024

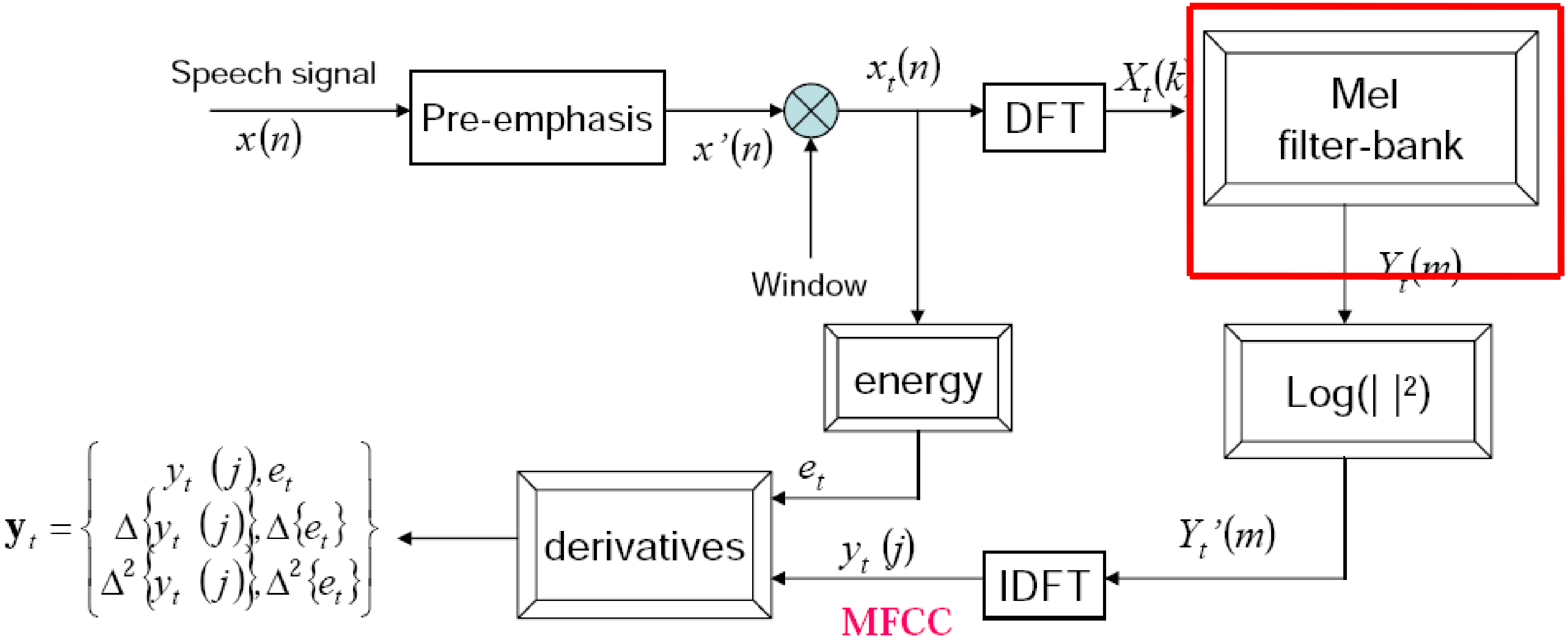
Discrete Fourier Transform computing a spectrum

- A 24 ms Hamming-windowed signal
  - And its spectrum as computed by DFT (plus other smoothing)



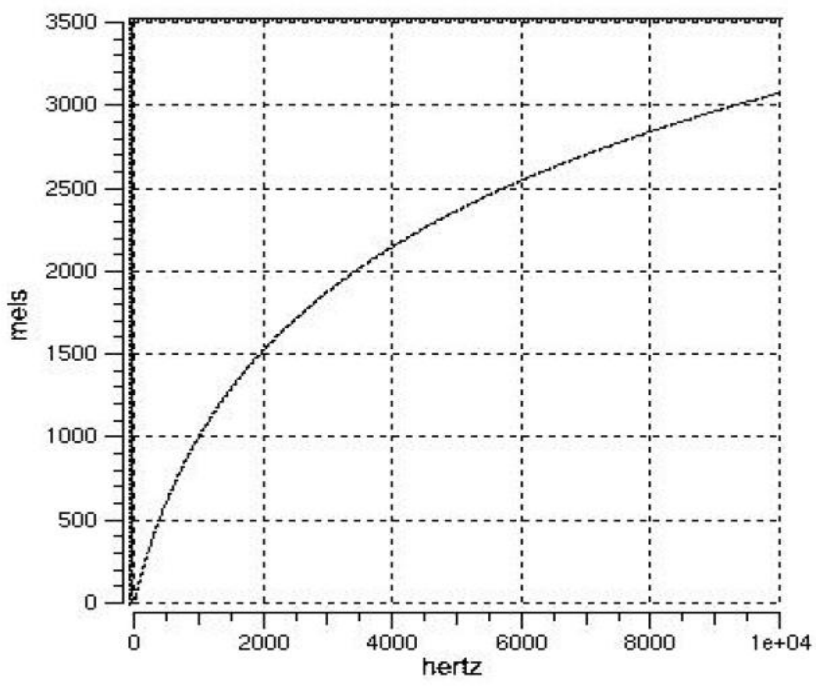


MFCC


Speech signal
$x(n)$
Pre-emphasis
$x'(n)$
Window
$x_t(n)$
DFT
$X_t(k)$
Mel
filter-bank
$Y_t(m)$
energy
Log(| |2)
e_t
$Y_t'(m)$
derivatives
$y_t(j)$
IDFT
MFCC
$\mathbf{y}_t = \left\{ \begin{array}{c} y_t(j), e_t \\ \Delta\{y_t(j)\}, \Delta\{e_t\} \\ \Delta^2\{y_t(j)\}, \Delta^2\{e_t\} \end{array} \right\}$

Mel-scale

- Human hearing is not equally sensitive to all frequency bands
- Less sensitive at higher frequencies, roughly > 1000 Hz
- I.e. human perception of frequency is non-linear:



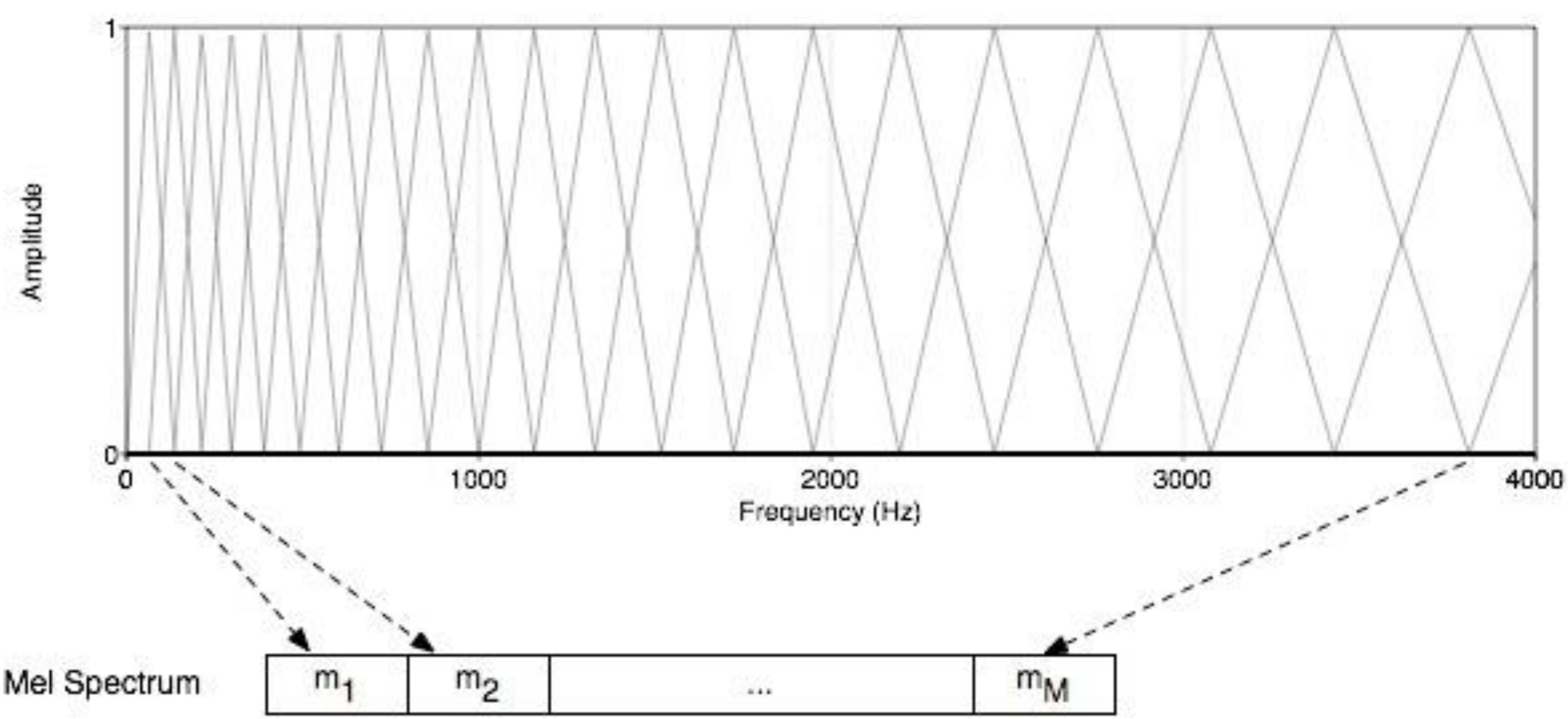
Mel-scale

- A **mel** is a unit of pitch
  - Definition:
    - Pairs of sounds perceptually equidistant in pitch
      - Are separated by an equal number of mels:
- Mel-scale is approximately linear below 1 kHz and logarithmic above 1 kHz
- Definition:

$$Mel(f) = 2595 \log_{10}(1 + \frac{f}{700})$$

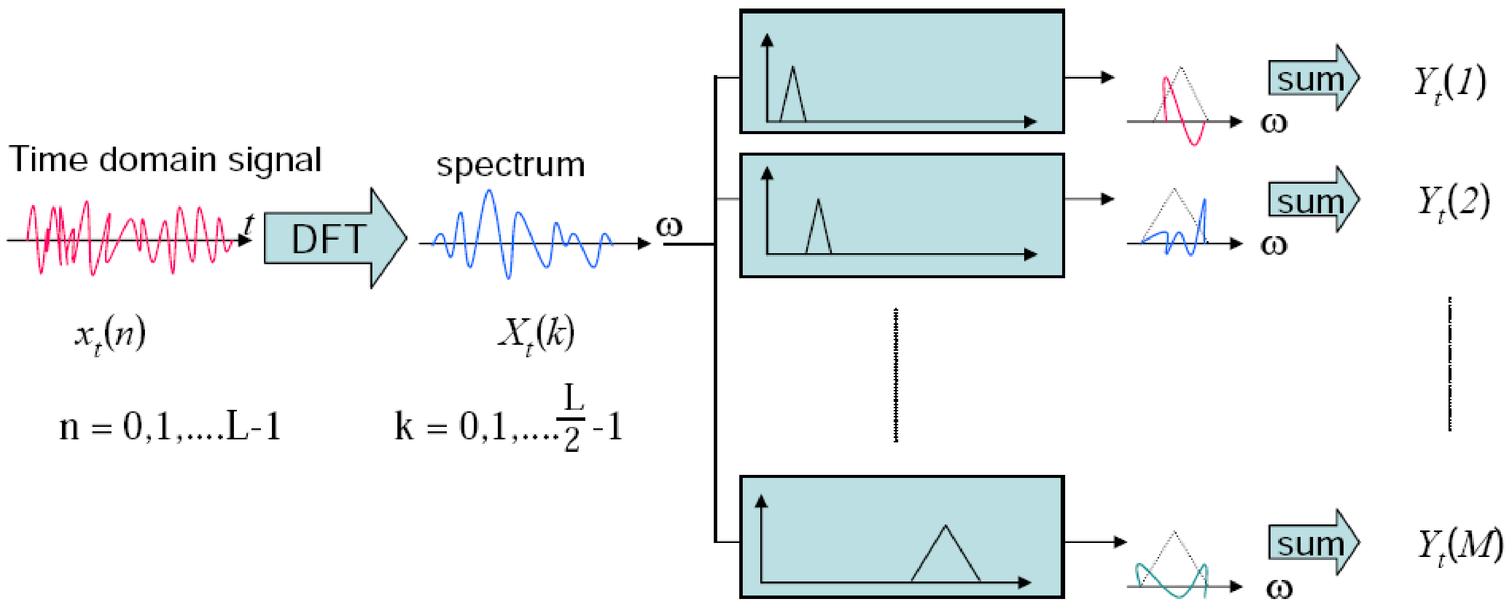
Mel Filter Bank Processing

- Mel Filter bank
  - Uniformly spaced before 1 kHz
  - logarithmic scale after 1 kHz

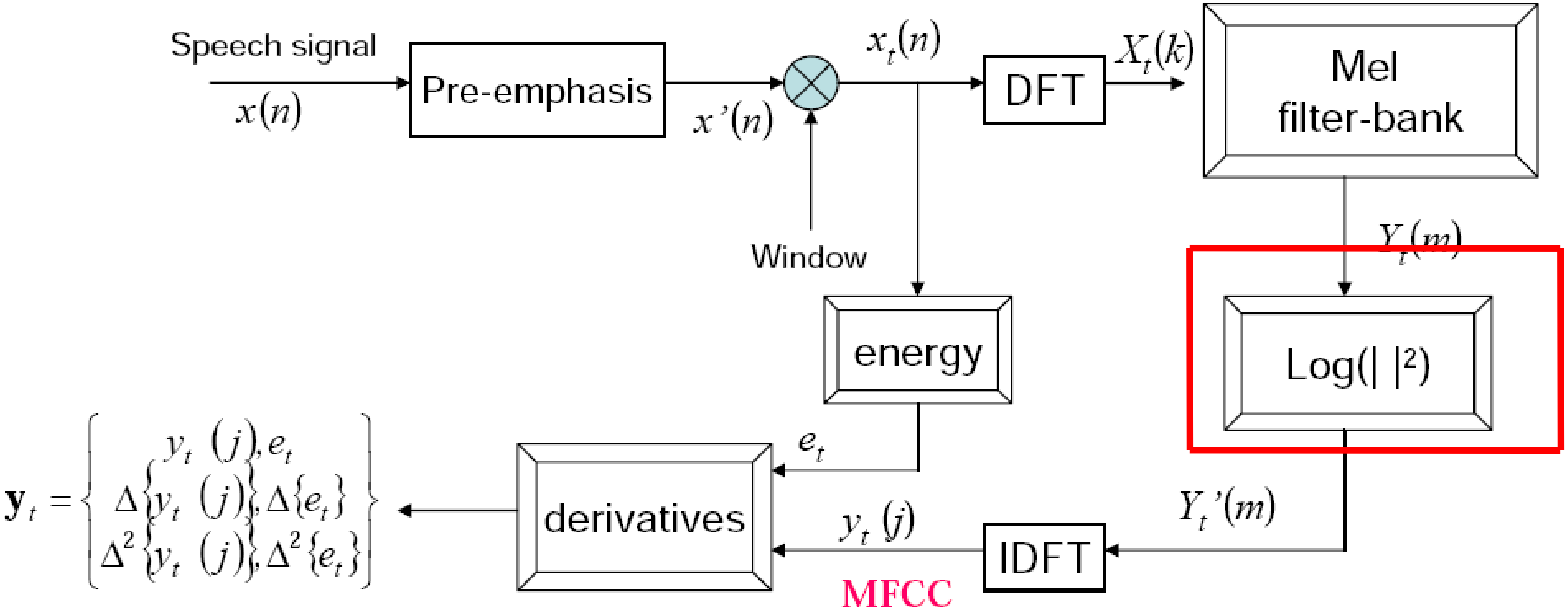


Mel-filter Bank Processing

- Apply the bank of filters according Mel scale to the spectrum
- Each filter output is the sum of its filtered spectral components



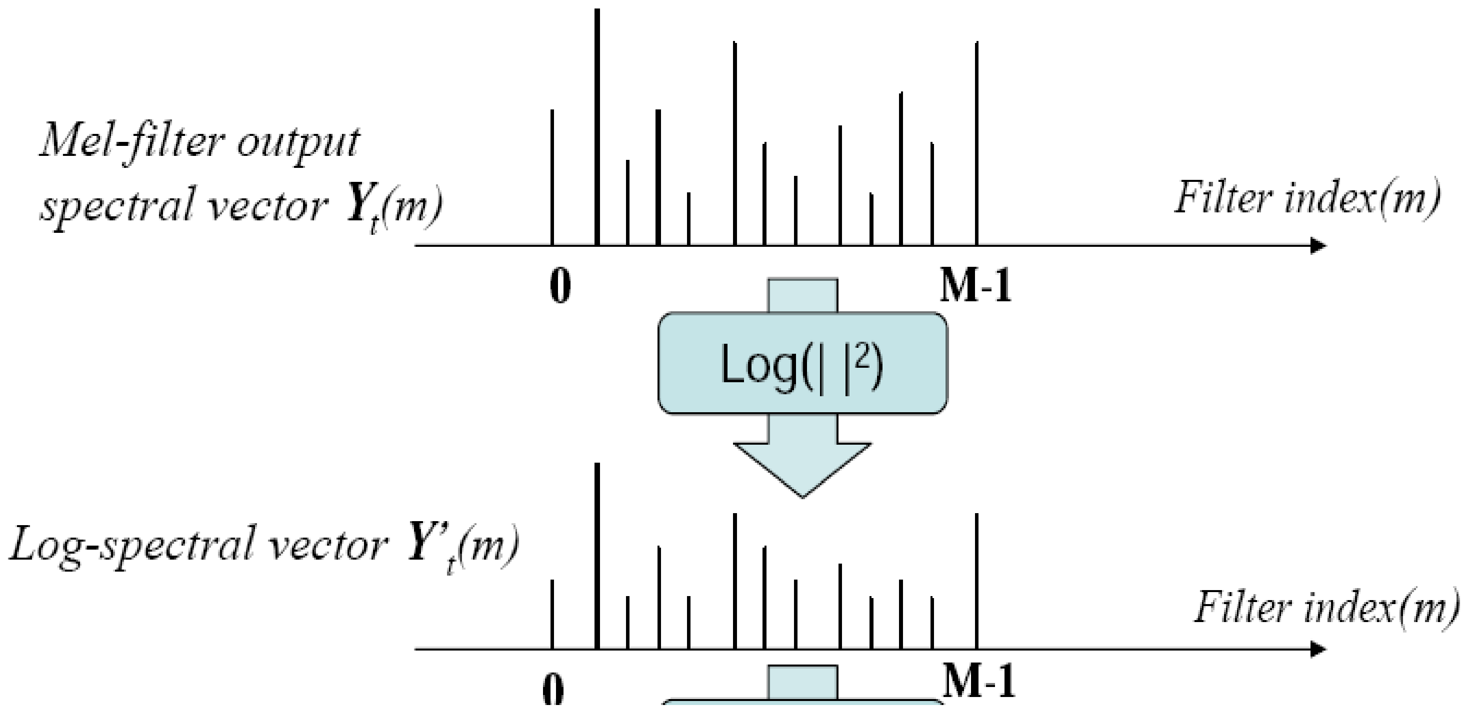
MFCC



Speech signal
$x(n)$
Pre-emphasis
$x'(n)$
Window
$x_t(n)$
DFT
$X_t(k)$
Mel filter-bank
$Y_t(m)$
Log(| |2)
energy
e_t
$Y_t'(m)$
IDFT
$y_t(j)$
MFCC
derivatives
$\mathbf{y}_t = \begin{Bmatrix} y_t(j), e_t \\ \Delta\{y_t(j)\}, \Delta\{e_t\} \\ \Delta^2\{y_t(j)\}, \Delta^2\{e_t\} \end{Bmatrix}$

Log energy computation

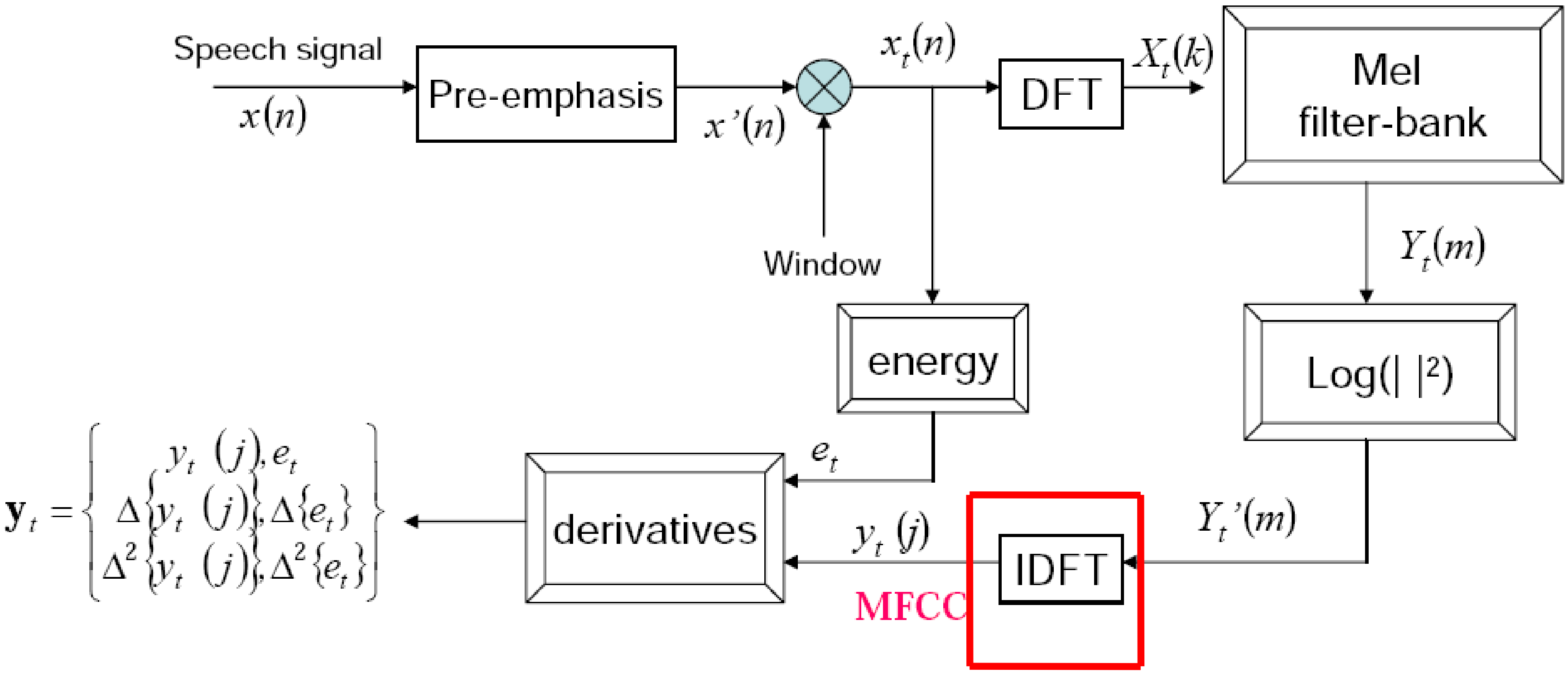
- Compute the logarithm of the square magnitude of the output of Mel-filter bank



Log energy computation

- Why log energy?
  - Logarithm compresses dynamic range of values
    - Human response to signal level is logarithmic
    - humans less sensitive to slight differences in amplitude at high amplitudes than low amplitudes
  - Makes frequency estimates less sensitive to slight variations in input (power variation due to speaker's mouth moving closer to mike)
  - Phase information not helpful in speech

MFCC

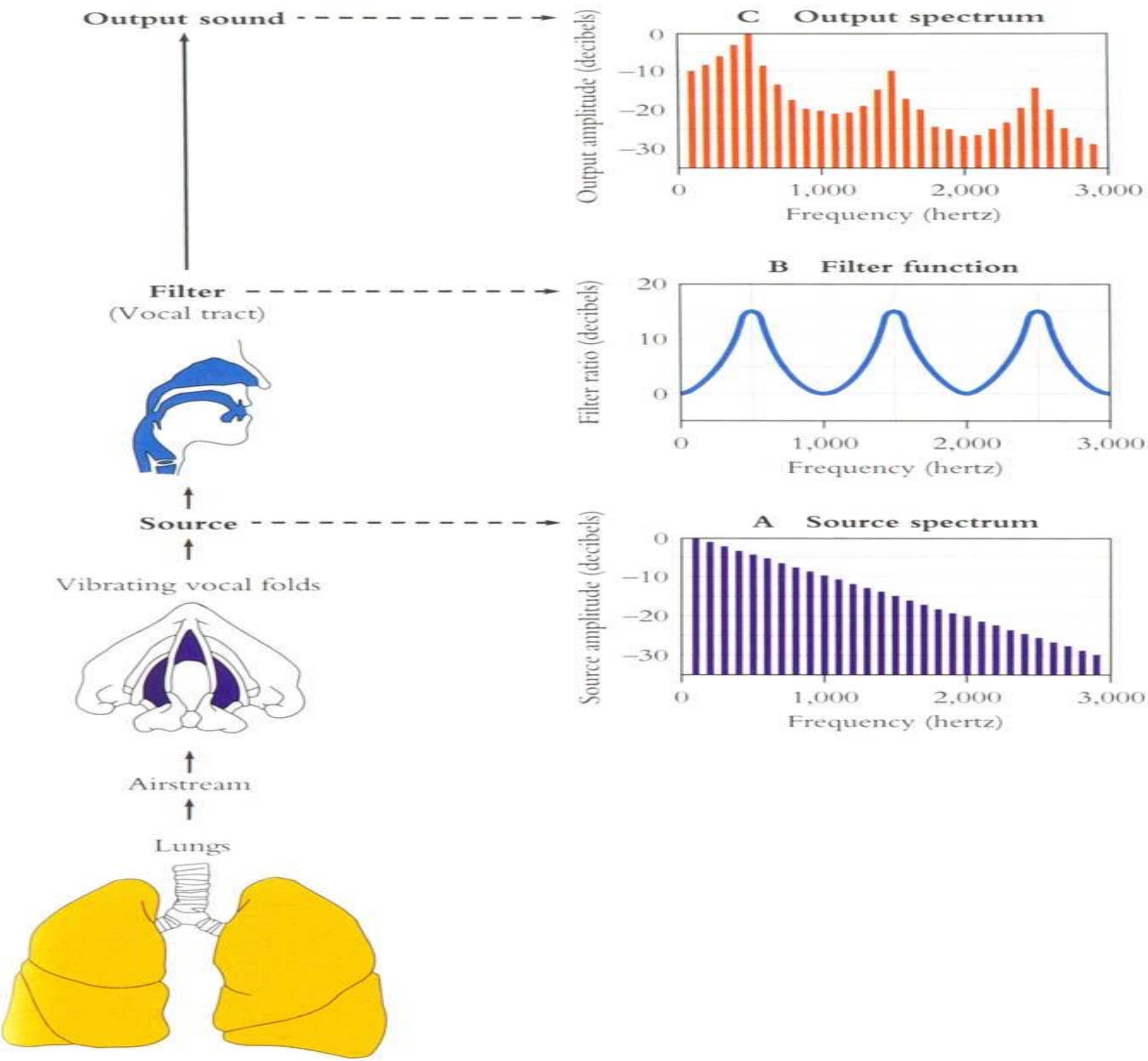


Speech signal
$x(n)$
Pre-emphasis
$x'(n)$
Window
$x_t(n)$
DFT
$X_t(k)$
Mel filter-bank
$Y_t(m)$
energy
Log(| |2)
e_t
derivatives
$y_t\ (j)$
IDFT
$Y_t'(m)$
MFCC
$\mathbf{y}_t = \left\{ \begin{matrix} y_t\ (j), e_t \\ \Delta\{y_t\ (j)\}, \Delta\{e_t\} \\ \Delta^2\{y_t\ (j)\}, \Delta^2\{e_t\} \end{matrix} \right\}$

The Cepstrum

- One way to think about this
  - Separating the **source** and **filter**
  - Speech waveform is created by
    - A glottal source waveform
    - Passes through a vocal tract which because of its shape has a particular filtering characteristic
- Articulatory facts:
  - The vocal cord vibrations create harmonics
  - The mouth is an amplifier
  - Depending on shape of oral cavity, some harmonics are amplified more than others

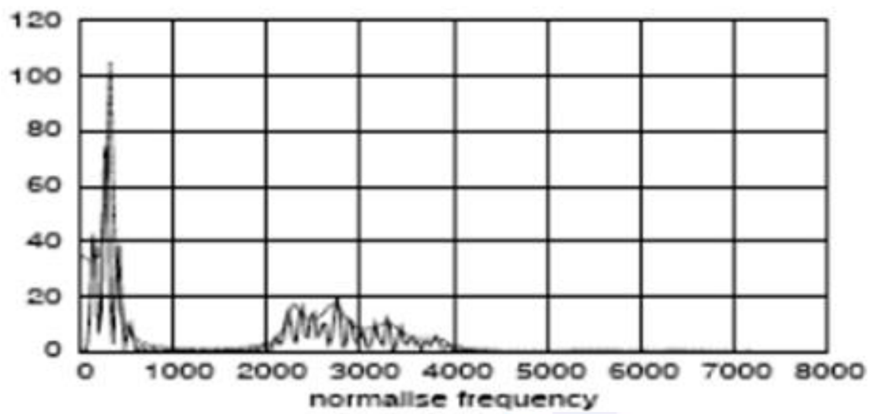
Vocal Fold Vibration


Output sound
Filter
(Vocal tract)
Source
Vibrating vocal folds
Airstream
Lungs
C Output spectrum
Output amplitude (decibels)
0
−10
−20
−30
0
1,000
2,000
3,000
Frequency (hertz)
B Filter function
Filter ratio (decibels)
20
10
0
0
1,000
2,000
3,000
Frequency (hertz)
A Source spectrum
Source amplitude (decibels)
0
−10
−20
−30
0
1,000
2,000
3,000
Frequency (hertz)

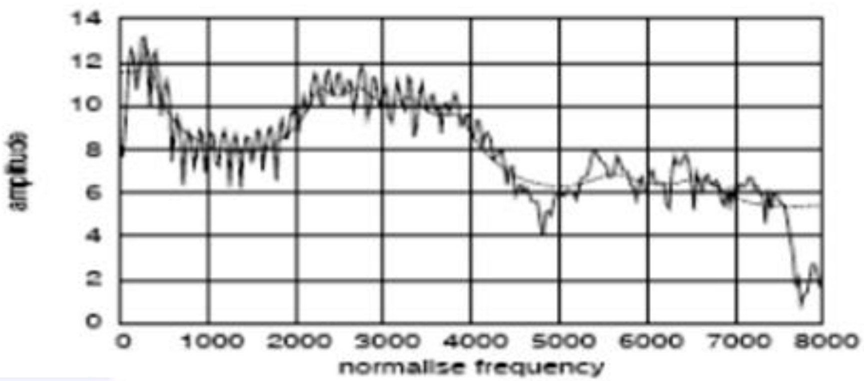
The Cepstrum

- The spectrum of the log of the spectrum

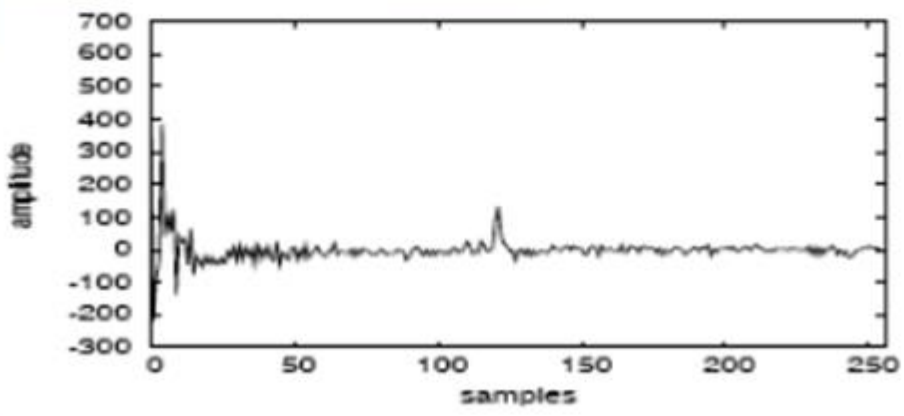
Spectrum



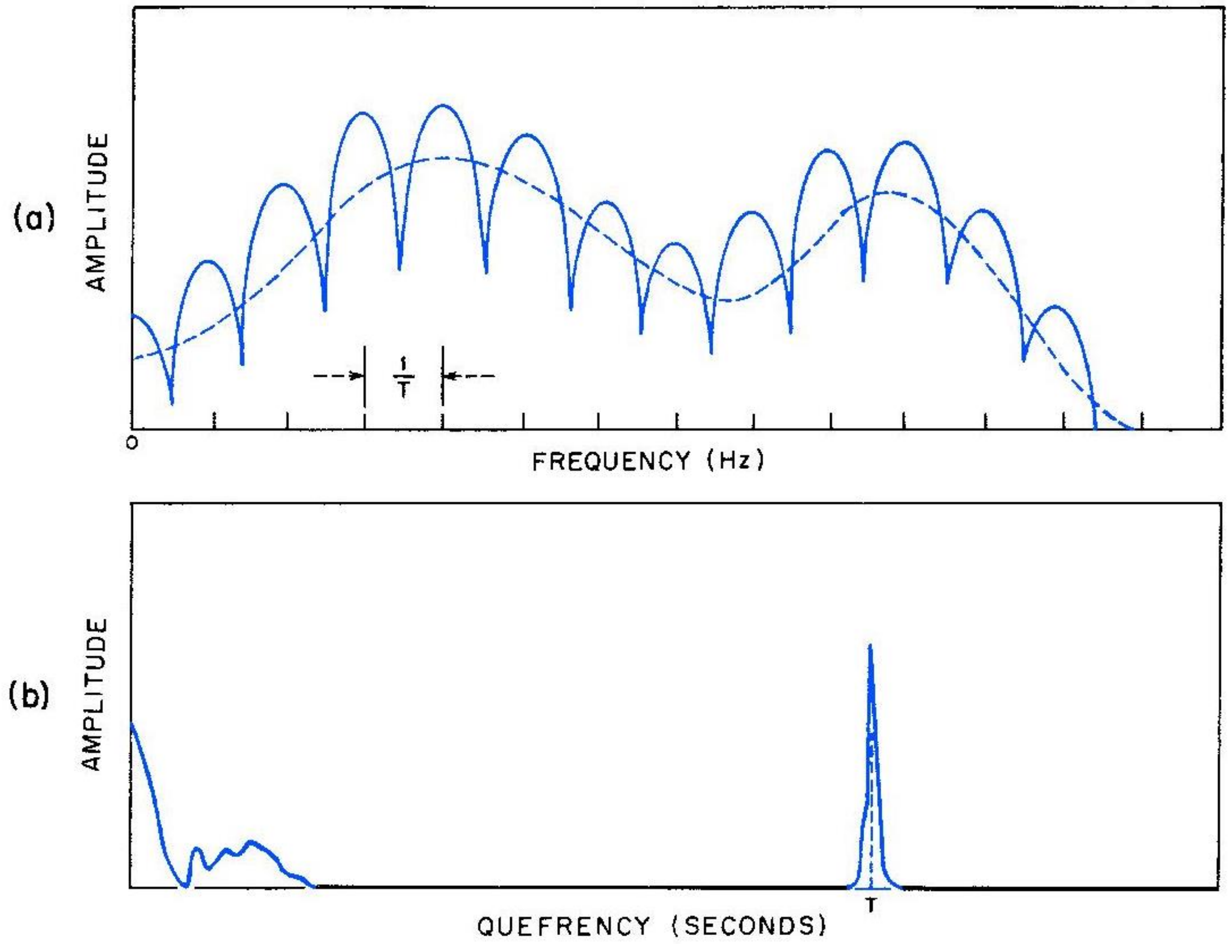
Log spectrum



Spectrum of log spectrum



Thinking about the Cepstrum



Mel Frequency cepstrum

- The cepstrum requires Fourier analysis
- But we’re going from frequency space back to time
- So we actually apply inverse DFT

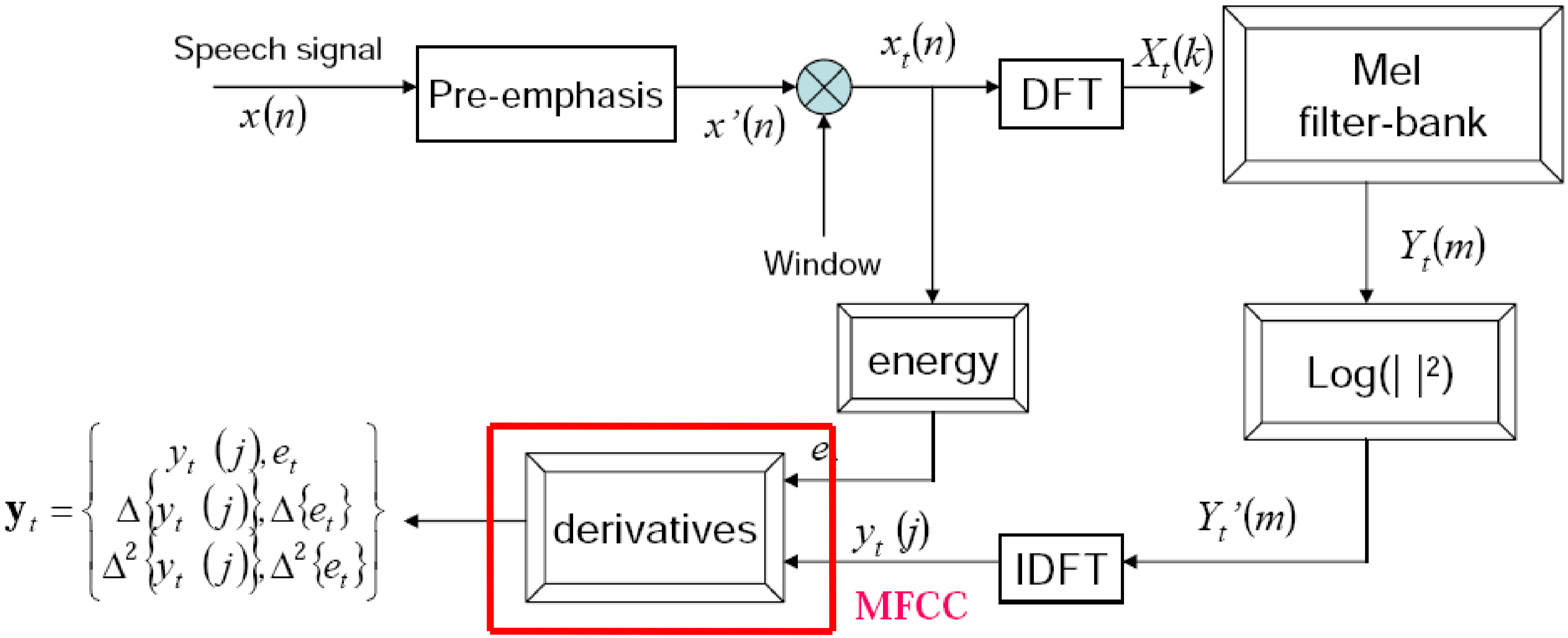
$$y_t[k] = \sum_{m=1}^{M} \log(|Y_t(m)|) \cos(k(m-0.5)\frac{\pi}{M}), \text{ k=0,...,J}$$

- Details for signal processing gurus: Since the log power spectrum is real and symmetric, inverse DFT reduces to a Discrete Cosine Transform (DCT)

Another advantage of the Cepstrum

- DCT produces highly **uncorrelated** features
- We'll see when we get to acoustic modeling that these will be much easier to model than the spectrum
  - Simply modelled by linear combinations of Gaussian density functions with diagonal covariance matrices
- In general we'll just use the first 12 cepstral coefficients (we don't want the later ones which have e.g. the F0 spike)

MFCC



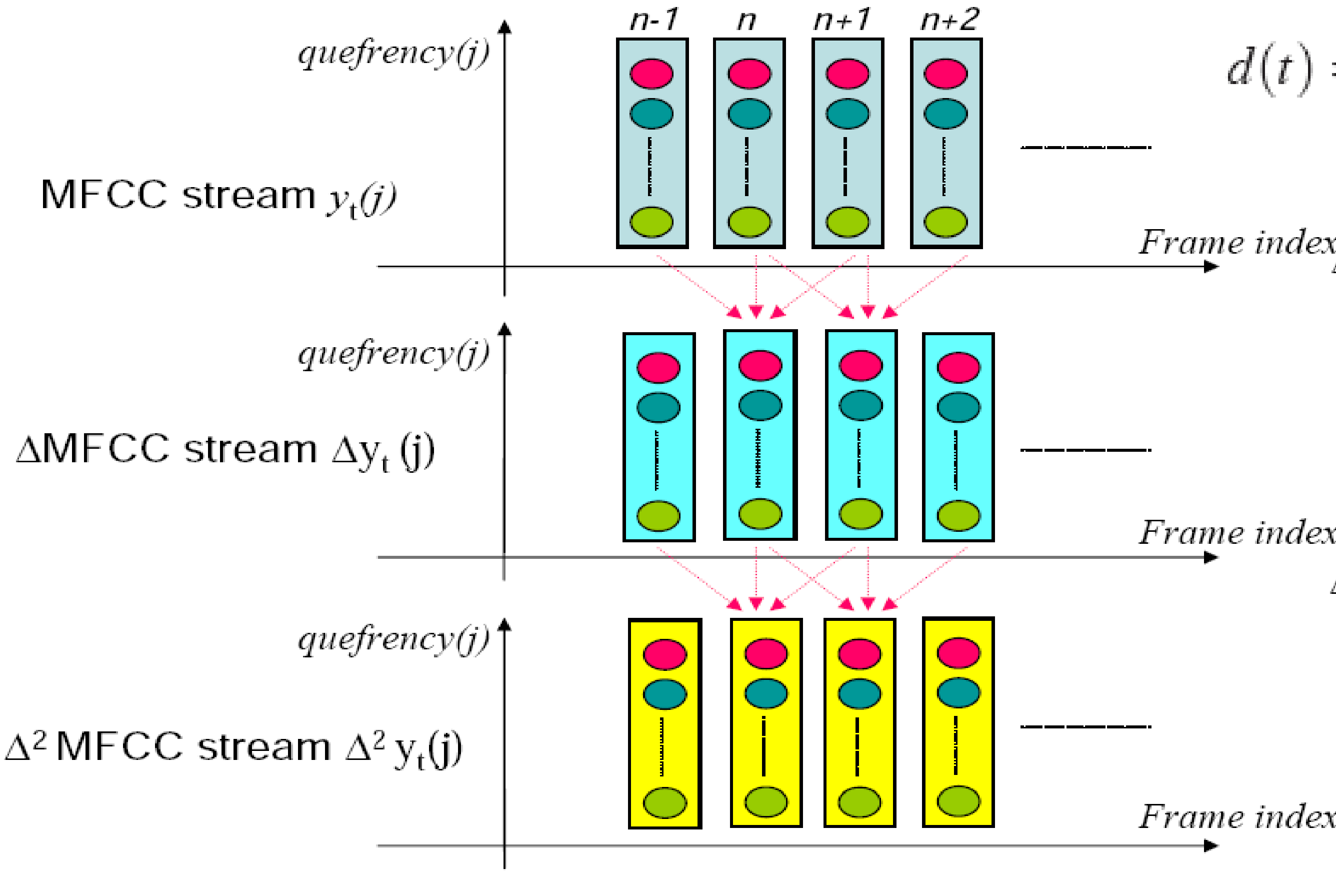
Speech signal
$x(n)$
Pre-emphasis
$x'(n)$
Window
$x_t(n)$
DFT
$X_t(k)$
Mel filter-bank
$Y_t(m)$
energy
Log(| |²)
e_t
derivatives
$y_t(j)$
IDFT
$Y_t'(m)$
MFCC
$$\mathbf{y}_t = \begin{Bmatrix} y_t(j), e_t \\ \Delta\{y_t(j)\}, \Delta\{e_t\} \\ \Delta^2\{y_t(j)\}, \Delta^2\{e_t\} \end{Bmatrix}$$

Dynamic Cepstral Coefficient

- The cepstral coefficients do not capture energy
- So we add an energy featur $Energy = \sum_{t=t_1}^{t_2} x^2[t]$
- Also, we know that speech signal is not constant (slope of formants, change from stop burst to release).
- So we want to add the changes in features (the slopes).
- We call these **delta** features
- We also add **double-delta** acceleration features

Delta and double-delta

- Derivative: in order to obtain temporal information



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

$$\Delta y_t(j) = \frac{\sum_{m=-p}^{p} m \bullet y_{t-m}(j)}{\sum_{m=-p}^{p} m^2}$$

$$\Delta^2 y_t(j) = \frac{\sum_{m=-p}^{p} m \bullet \Delta y_{t-m}(j)}{\sum_{m=-p}^{p} m^2}$$

Typical MFCC features

- Window size: 25ms
- Window shift: 10ms
- Pre-emphasis coefficient: 0.97
- MFCC:
  - 12 MFCC (mel frequency cepstral coefficients)
  - 1 energy feature
  - 12 delta MFCC features
  - 12 double-delta MFCC features
  - 1 delta energy feature
  - 1 double-delta energy feature
- Total 39-dimensional features

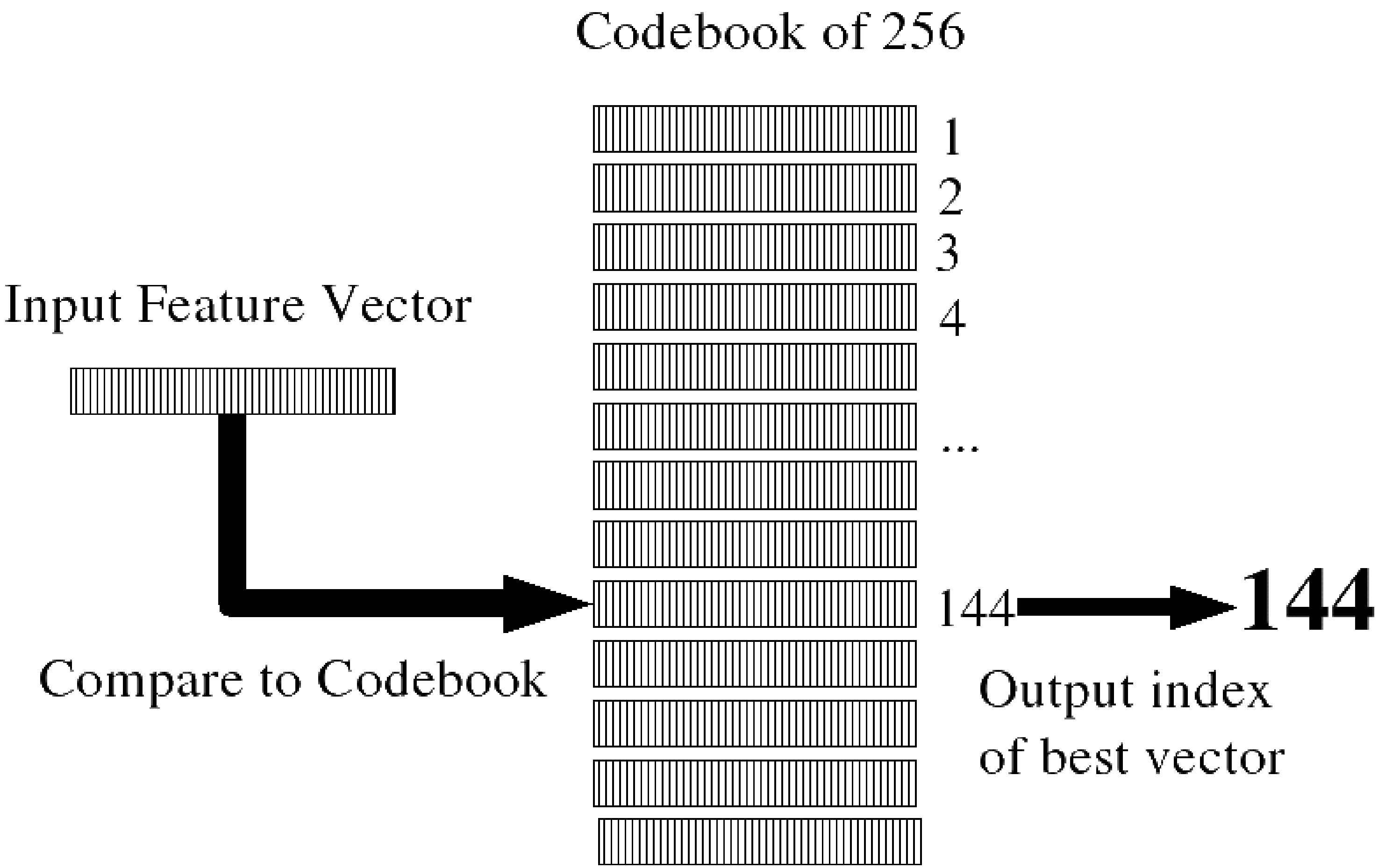
Vector Quantization

- Create a training set of feature vectors
- Cluster them into a small number of classes
- Represent each class by a discrete symbol
- For each class v_k, we can compute the probability that it is generated by a given HMM state using Baum-Welch as above

VQ

- We'll define a
  - Codebook, which lists for each symbol
  - A prototype vector, or codeword
- If we had 256 classes ('8-bit VQ'),
  - A codebook with 256 prototype vectors
  - Given an incoming feature vector, we compare it to each of the 256 prototype vectors
  - We pick whichever one is closest (by some 'distance metric')
  - And replace the input vector by the index of this prototype vector

VQ


Codebook of 256
1
2
3
4
...
144
144
Input Feature Vector
Compare to Codebook
Output index
of best vector

VQ requirements

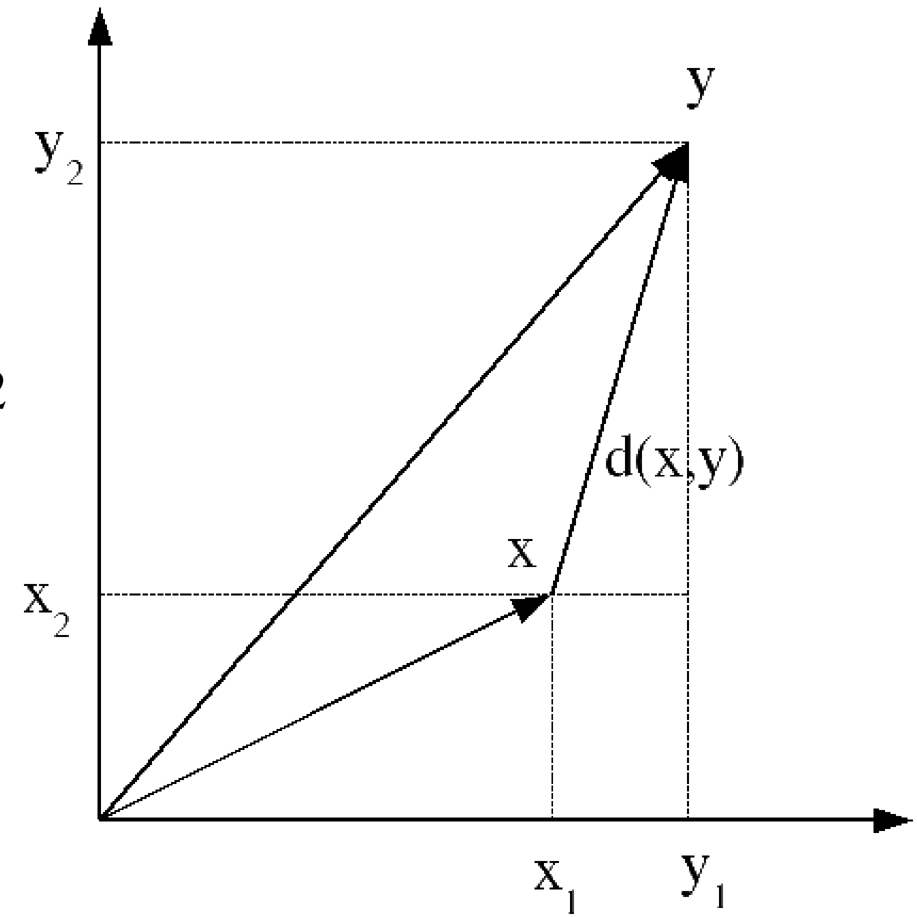
- A <u>distance metric</u> or <u>distortion metric</u>
  - Specifies how similar two vectors are
  - Used:
    - to build clusters
    - To find prototype vector for cluster
    - And to compare incoming vector to prototypes
- A clustering algorithm
  - K-means, etc.

Distance metrics

- Simplest:
  - (square of) Euclidean distance

$$d^2(x,y) = \sum_{i=1}^{D} (x_i - y_i)^2$$

  - Also called 'sum-squared error'



Distance metrics

- More sophisticated:
  - (square of) Mahalanobis distance
  - Assume that each dimension of feature vector has variance σ^2

$$d^2(x,y) = \sum_{i=1}^{D} \frac{(x_i - y_i)^2}{\sigma_i^2}$$

  - Equation above assumes diagonal covariance matrix; more on this later

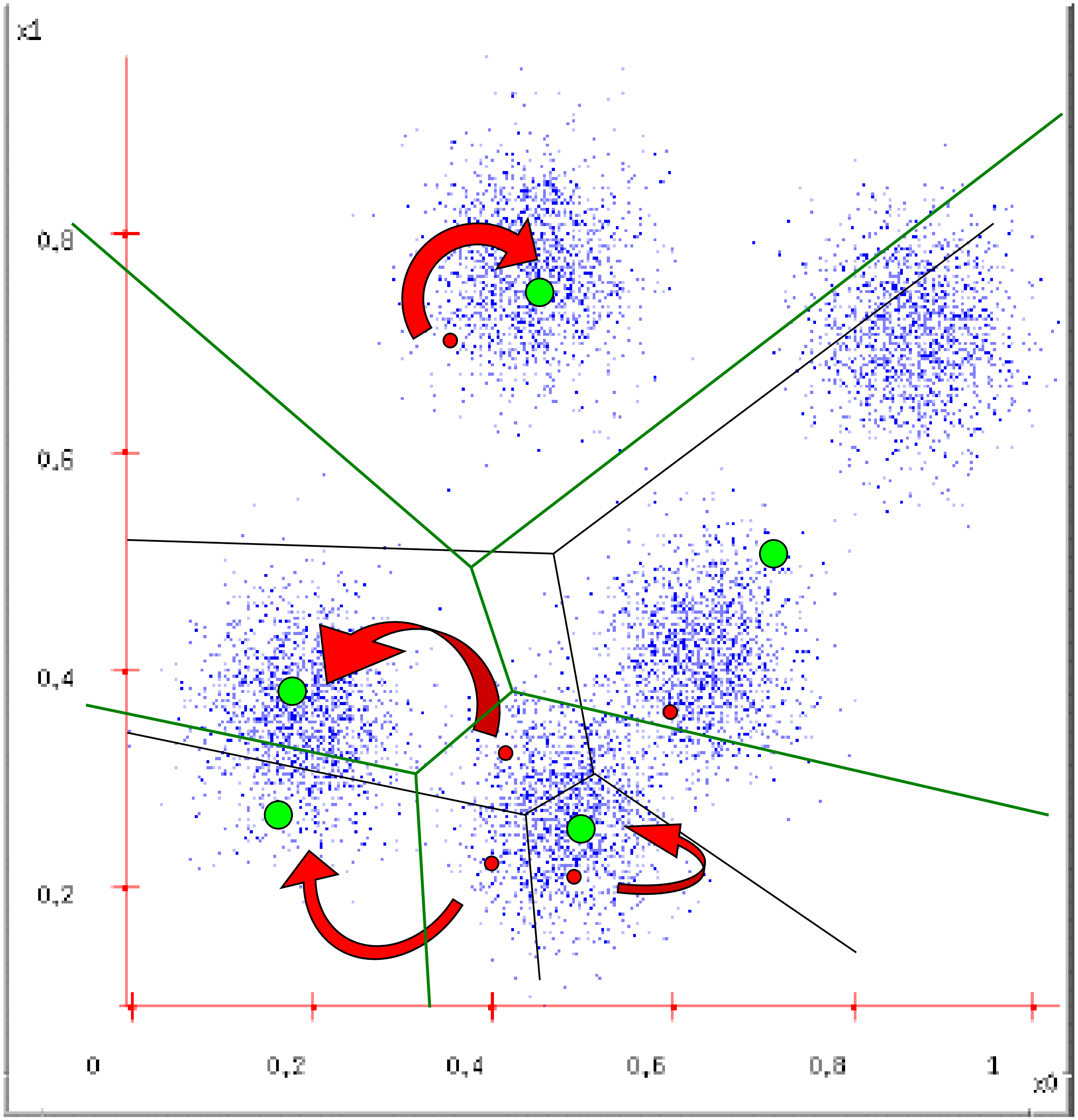
Training a VQ system (generating codebook): **K-means clustering**

1. Initialization
   choose M vectors from L training vectors (typically $M=2^B$)
   as initial code words… random or max. distance.
2. Search:
   for each training vector, find the closest code word, assign this training vector to that cell
3. Centroid Update:
   for each cell, compute centroid of that cell. The new code word is the centroid.
4. Repeat (2)-(3) until average distance falls below threshold (or no change)

1. **Ask user how many clusters they'd like.(e.g. k=5).**
2. **Randomly select k cluster centers locations**
3. **Each data point finds out which center it's closest to. (Thus each center "owns" a set of data points)**
4. **Each cluster finds the centroid of the points it owns.**

$$M_k = \frac{1}{N_k} \cdot \sum_{j=1}^{N_k} X_{jk}$$

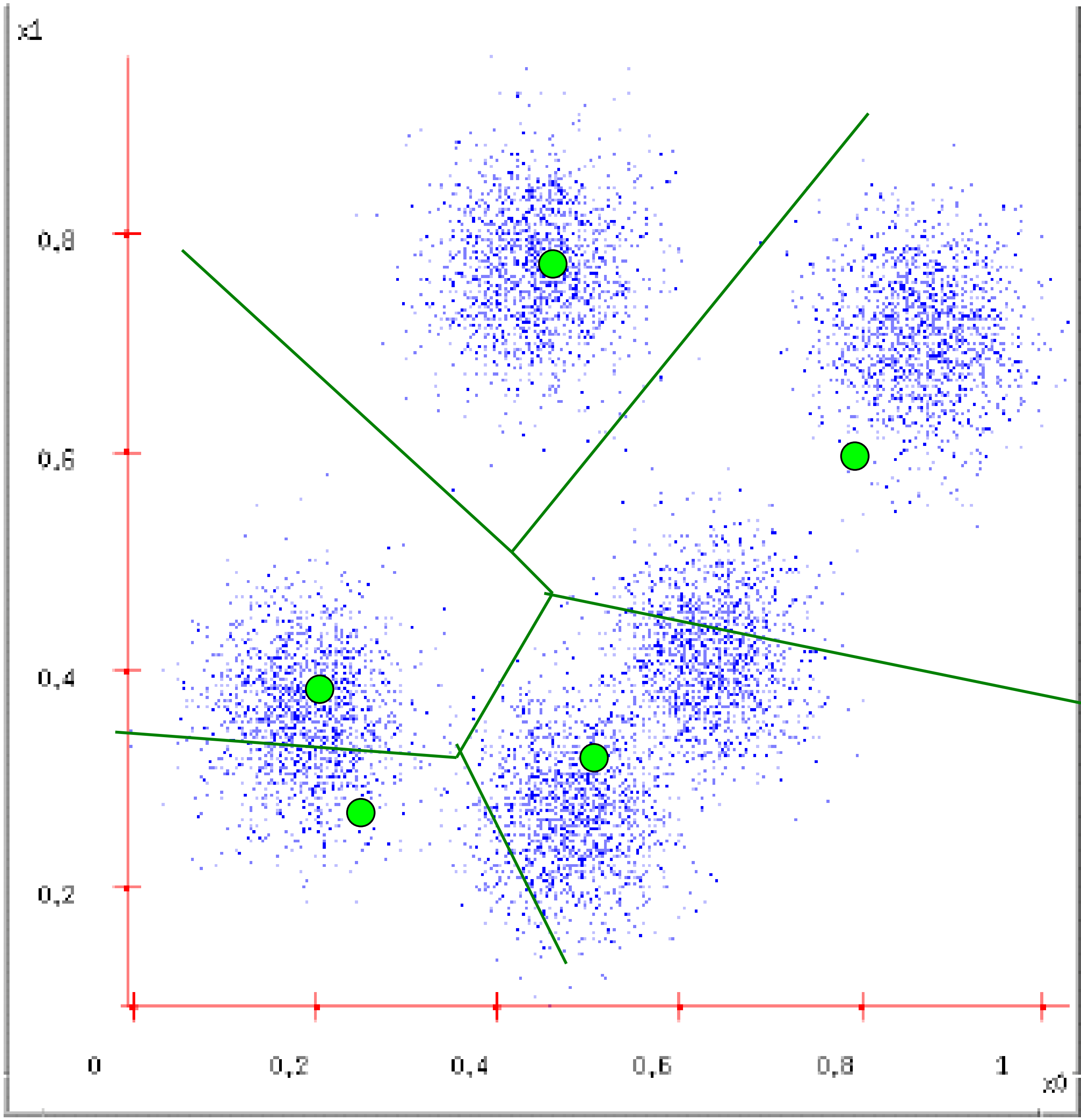
5. **…and jumps to there**
6. **…repeat step 3 to 5 until terminated**



1. Ask user how many clusters they'd like.(e.g. k=5).
2. Randomly select k cluster centers locations
3. Each data point finds out which center it's closest to. (Thus each center "owns" a set of data points)
4. Each cluster finds the centroid of the points it owns.

$$M_k = \frac{1}{N_k} \cdot \sum_{j=1}^{N_k} X_{jk}$$

5. …and jumps to there
6. …repeat step 3 to 5 until terminated



1. **Ask user how many clusters they'd like.(e.g. k=5).**
2. **Randomly select k cluster centers locations**
3. **Each data point finds out which center it's closest to. (Thus each center "owns" a set of data points)**
4. **Each cluster finds the centroid of the points it owns.**

$$M_k = \frac{1}{N_k} \cdot \sum_{j=1}^{N_k} X_{jk}$$

5. **…and jumps to there**
6. **…repeat step 3 to 5 until terminated**