CS 224S / LINGUIST 285
Spoken Language Processing

Andrew Maas
Stanford University
Spring 2017

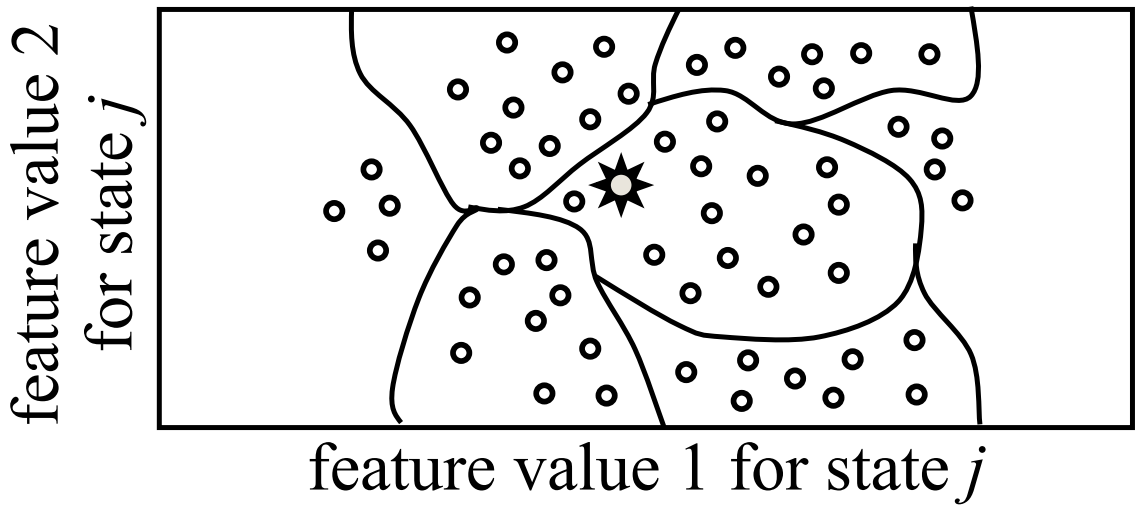
Lecture 5: GMM Acoustic Modeling and Feature Extraction

Original slides by Dan Jurafsky

Outline for Today

- Acoustic Model
  - Acoustic likelihood for each state using Gaussians and Mixtures of Gaussians
  - Where a state is progressively:
    - CI Subphone (3ish per phone)
    - Context Dependent (CD) phone (=triphones)
    - State-tying of CD phone
- MFCC feature extraction
- Handling variation
  - MLLR adaptation
  - MAP adaptation (On your own)

Vector quantization: Discrete observations



Slide from John-Paul Hosum

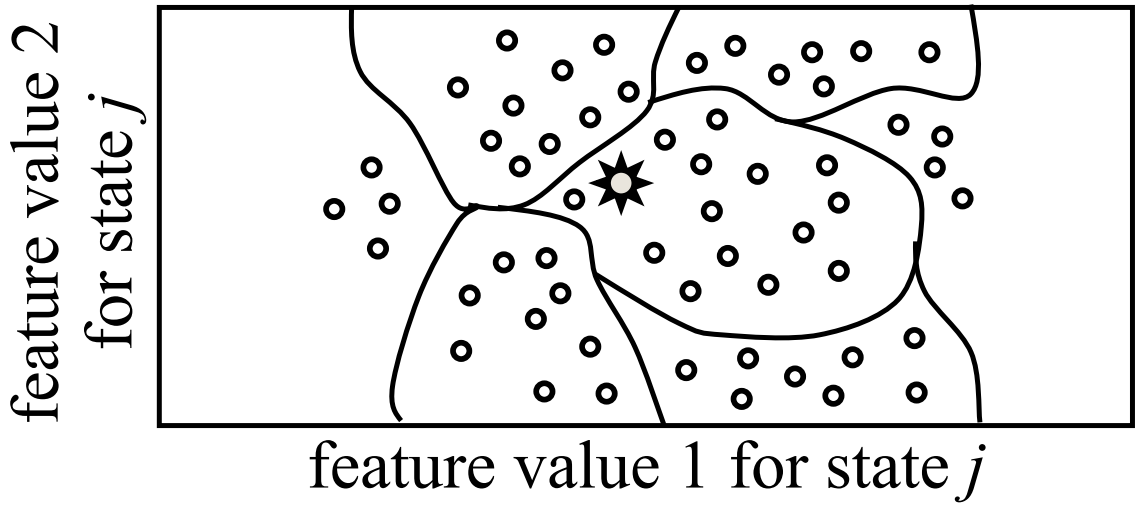
VQ

- 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

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

Vector quantization: Discrete observations



- $b_j(v_k) = \frac{\text{number of vectors with codebook index } k \text{ in state } j}{\text{number of vectors in state } j} = \frac{14}{56} = \frac{1}{4}$

Slide from John-Paul Hosum

Directly Modeling Continuous Observations

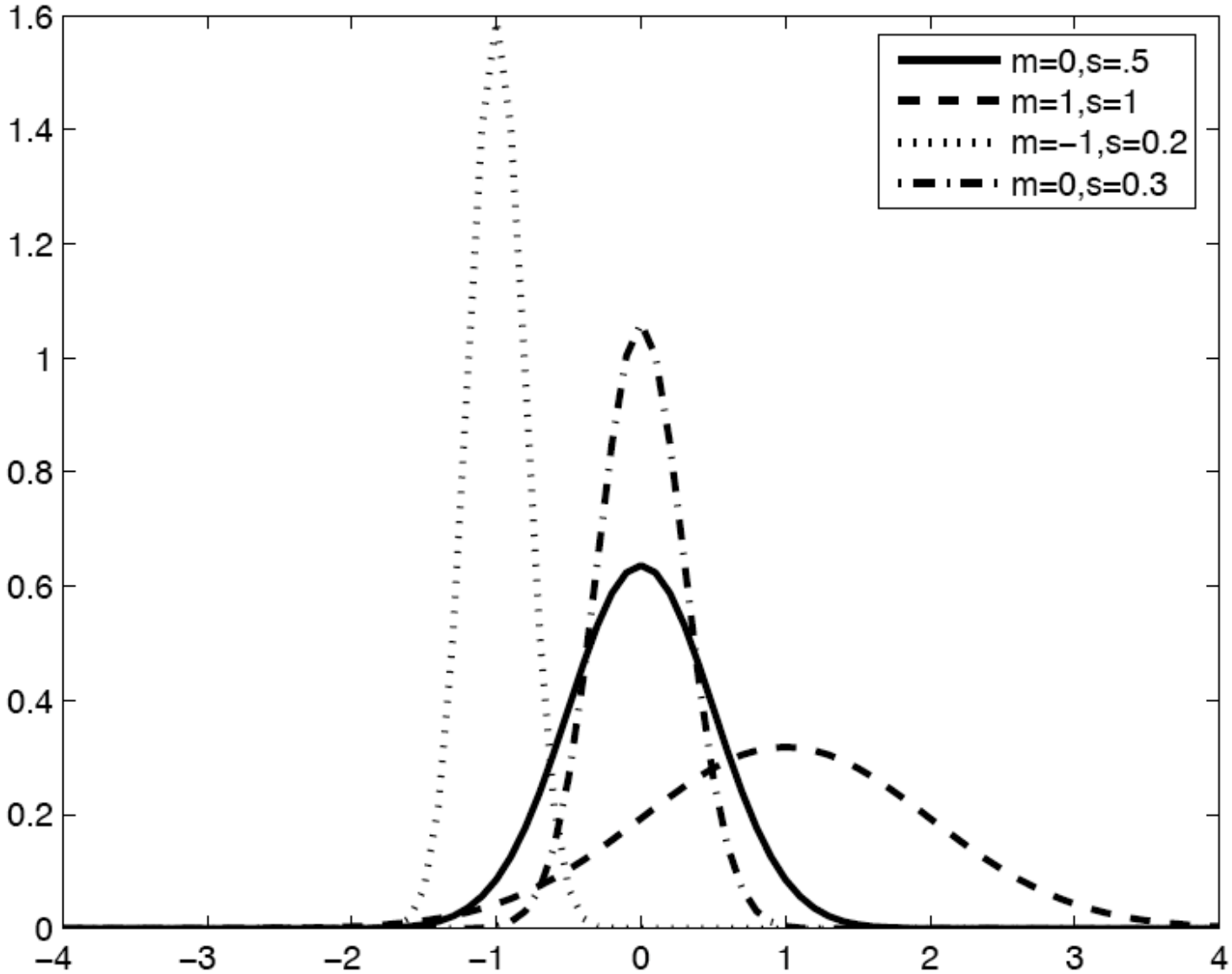
- Gaussians
  - Univariate Gaussians
    - Baum-Welch for univariate Gaussians
  - Multivariate Gaussians
    - Baum-Welch for multivariate Gausians
  - Gaussian Mixture Models (GMMs)
    - Baum-Welch for GMMs

Better than VQ

- VQ is insufficient for real ASR
- Instead: Assume the possible values of the observation feature vector o_t are normally distributed.
- Represent the observation likelihood function $b_j(o_t)$ as a Gaussian with mean μ_j and variance σ_j^2

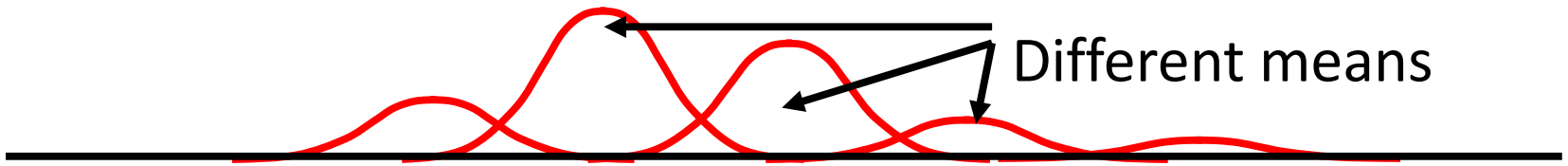
$$f(x \mid \mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

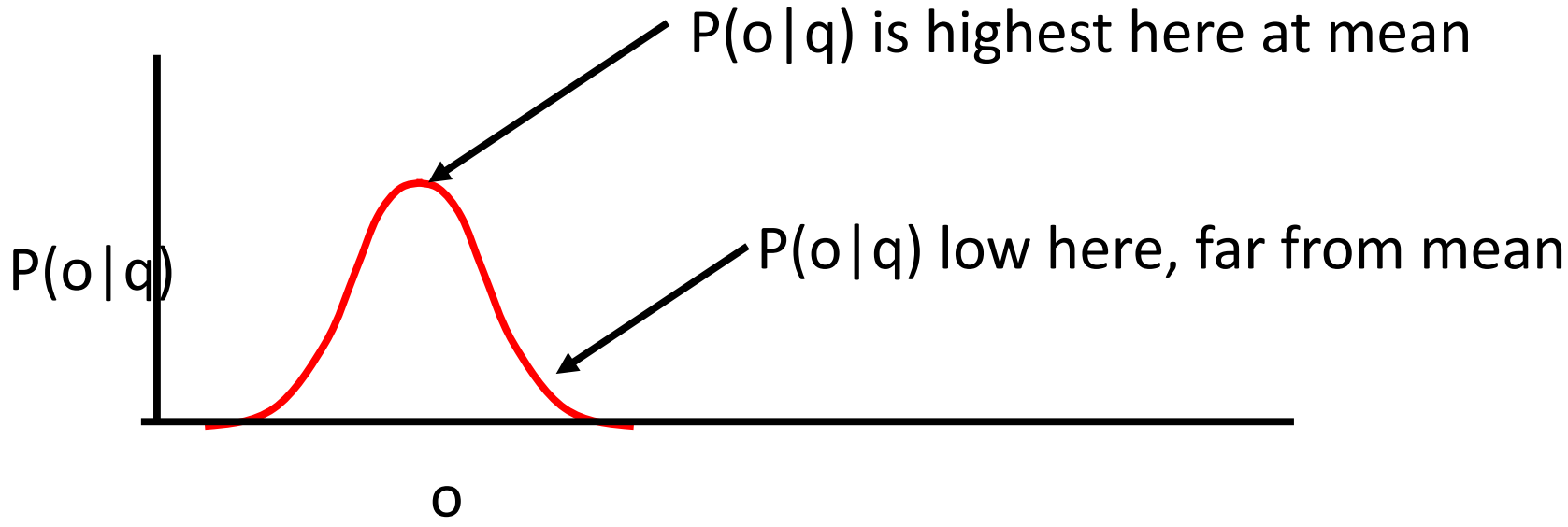
Gaussians are parameterized by mean and variance



Gaussians for Acoustic Modeling

- P(o|q): A Gaussian parameterized by mean and variance:





Using a (univariate) Gaussian as an acoustic likelihood estimator

- Let's suppose our observation was a single real-valued feature (instead of 39D vector)
- Then if we had learned a Gaussian over the distribution of values of this feature
- We could compute the likelihood of any given observation o_t as follows:

$$b_j(o_t) = \frac{1}{\sqrt{2\pi\sigma_j^2}} exp\left(-\frac{(o_t - \mu_j)^2}{2\sigma_j^2}\right)$$

Training a Univariate Gaussian

- A (single) Gaussian is characterized by a mean and a variance
- Imagine that we had some training data in which each state was labeled
- We could just compute the mean and variance from the data:

$$\mu_i = \frac{1}{T}\sum_{t=1}^{T} o_t \quad s.t.\ o_t \ \ is \ \ state \ \ i$$

$$\sigma_i^{\ 2} = \frac{1}{T}\sum_{t=1}^{T} (o_t - \mu_i)^2 \quad s.t.\ q_t \ \ is \ \ state \ \ i$$

Training Univariate Gaussians

- But we don't know which observation was produced by which state!
- What we want: to assign each observation vector o_t to every possible state *i*, prorated by the probability the the HMM was in state *i* at time *t*.
- The probability of being in state *i* at time *t* is $\gamma_t(i)$!!

$$\bar{\mu}_i = \frac{\sum_{t=1}^{T} \gamma_t(i) o_t}{\sum_{t=1}^{T} \gamma_t(i)} \qquad \bar{\sigma}_i^2 = \frac{\sum_{t=1}^{T} \gamma_t(i)(o_t - \mu_i)^2}{\sum_{t=1}^{T} \gamma_t(i)}$$

Multivariate Gaussians

- Instead of a single mean μ and variance σ:

$$f(x \mid \mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}\exp(-\frac{(x-\mu)^2}{2\sigma^2})$$

- Vector of observations x modeled by vector of means μ and covariance matrix Σ

$$f(x \mid \mu,\Sigma) = \frac{1}{(2\pi)^{D/2} \mid \Sigma \mid^{1/2}}\exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$

Multivariate Gaussians

- Defining μ and Σ

$$\mu = E(x)$$

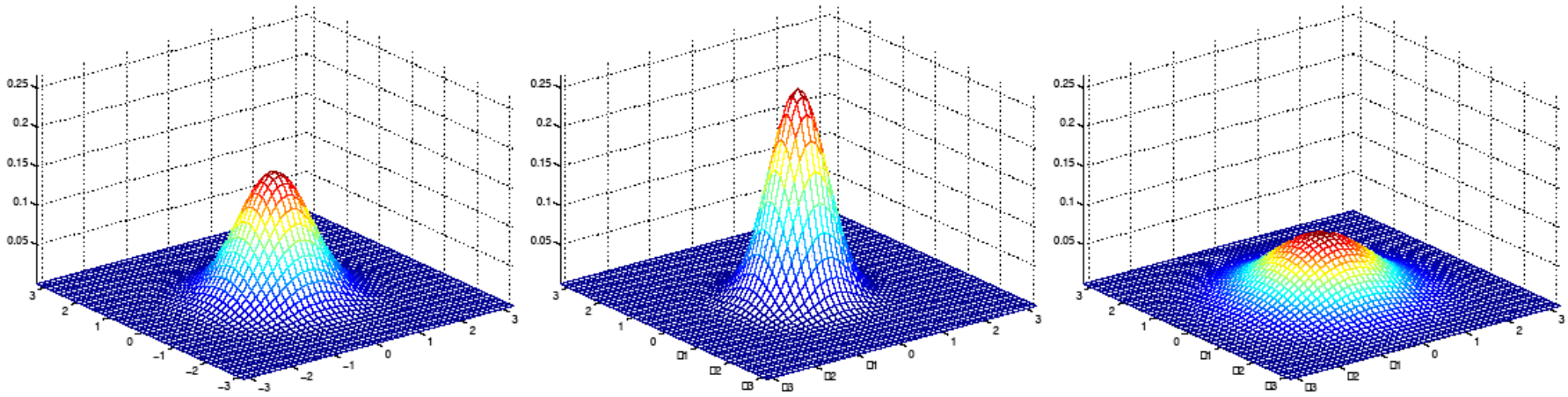
$$\Sigma = E\left[(x-\mu)(x-\mu)^T\right]$$

- So the i-jth element of Σ is:

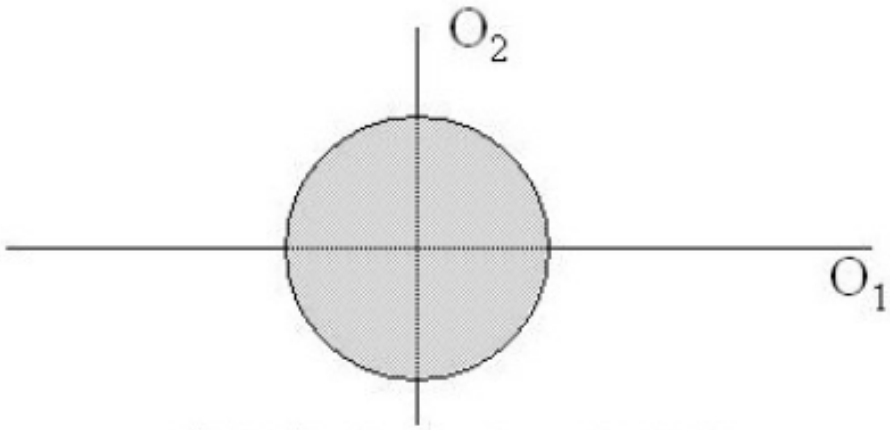
$$\sigma_{ij}^2 = E\left[(x_i-\mu_i)(x_j-\mu_j)\right]$$

Gaussian Intuitions: Size of Σ

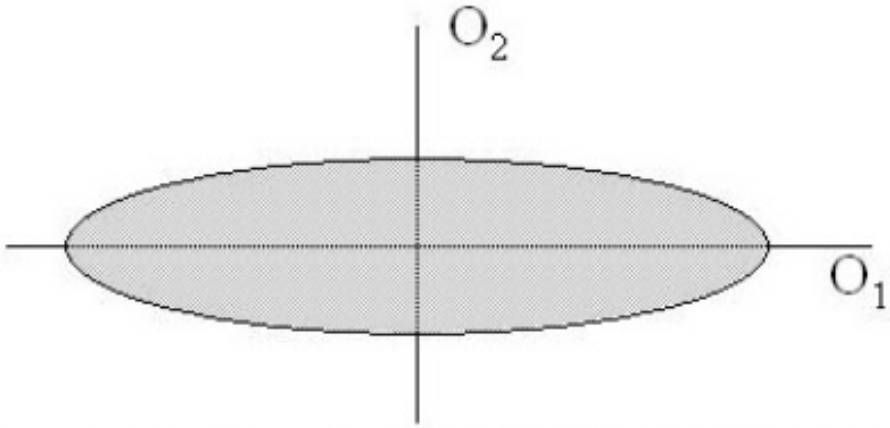
- $\mu = [0\ 0]$ $\mu = [0\ 0]$ $\mu = [0\ 0]$
- $\Sigma = I$ $\Sigma = 0.6I$ $\Sigma = 2I$
- As Σ becomes larger, Gaussian becomes more spread out; as Σ becomes smaller, Gaussian more compressed



From Andrew Ng's lecture notes for CS229



O_1 and O_2 are uncorrelated – knowing O_1 tells you nothing about O_2



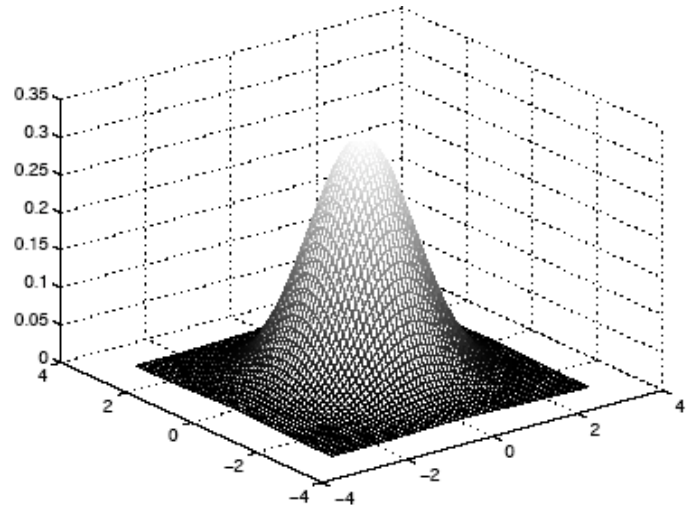
O_1 and O_2 can be uncorrelated without having equal variances

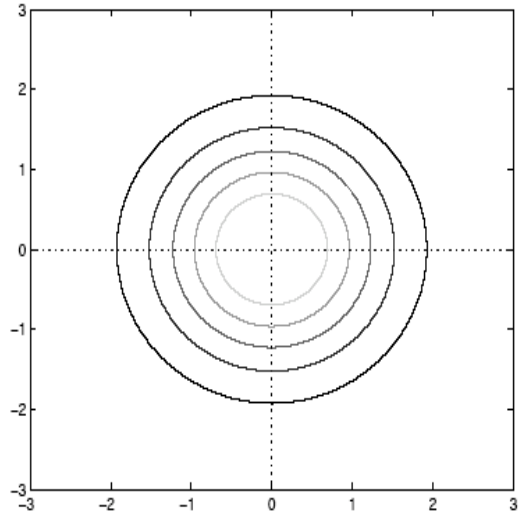
From Chen, Picheny et al lecture slides

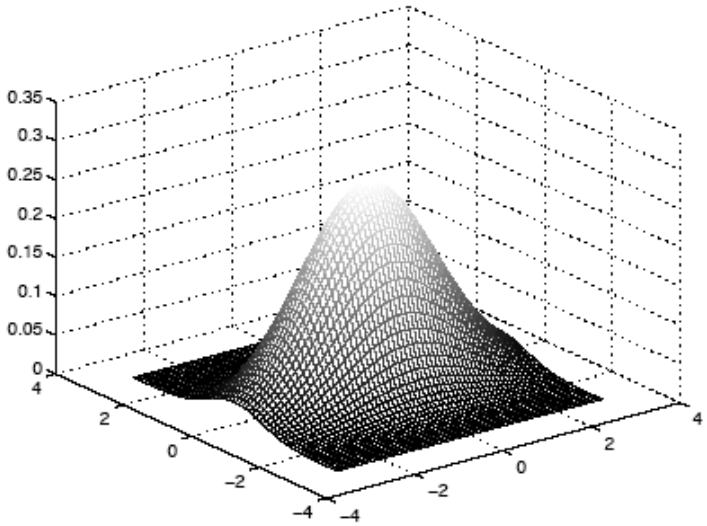
[1 0]
[0 1]

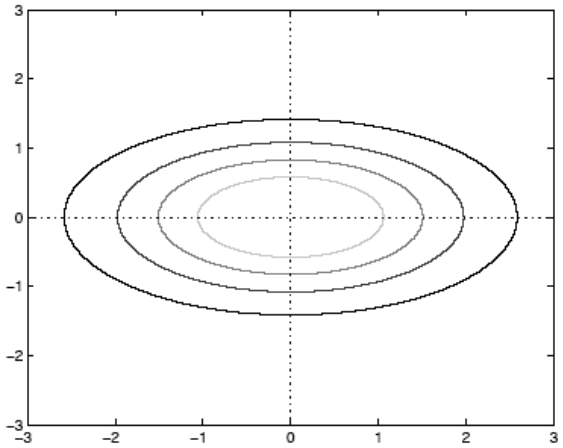
[.6 0]
[0 2]

- Different variances in different dimensions

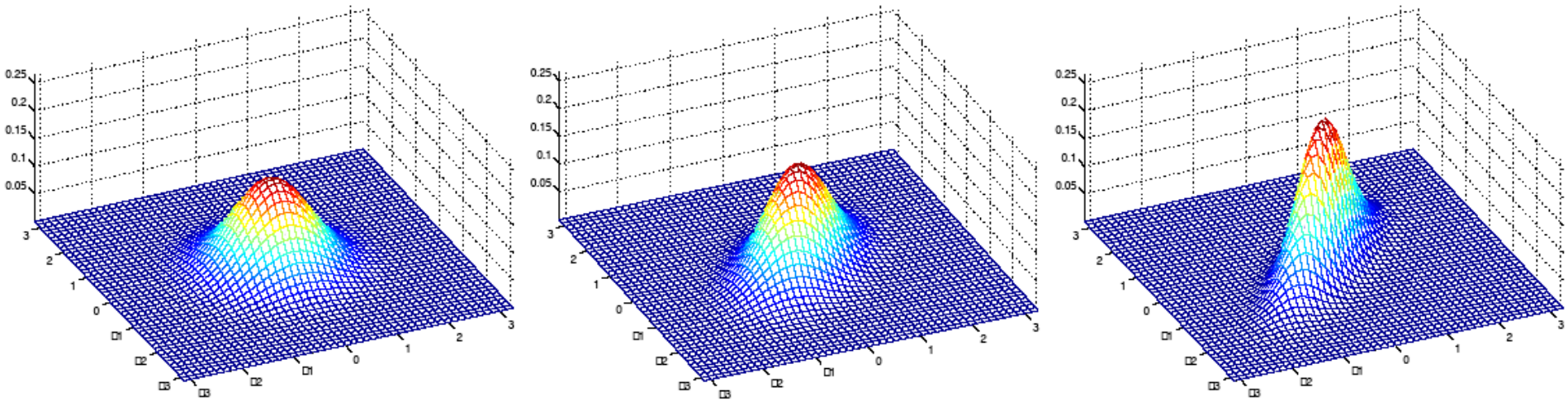








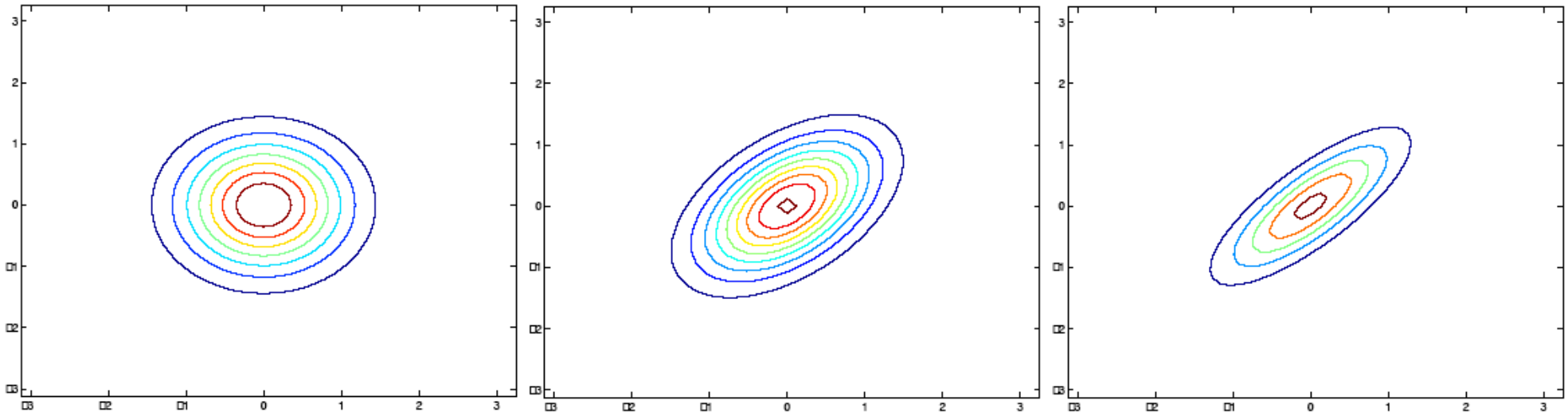
Gaussian Intuitions: Off-diagonal



$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}; \quad .\Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

Text and figures from Andrew Ng's lecture notes for CS229

Gaussian Intuitions: off-diagonal

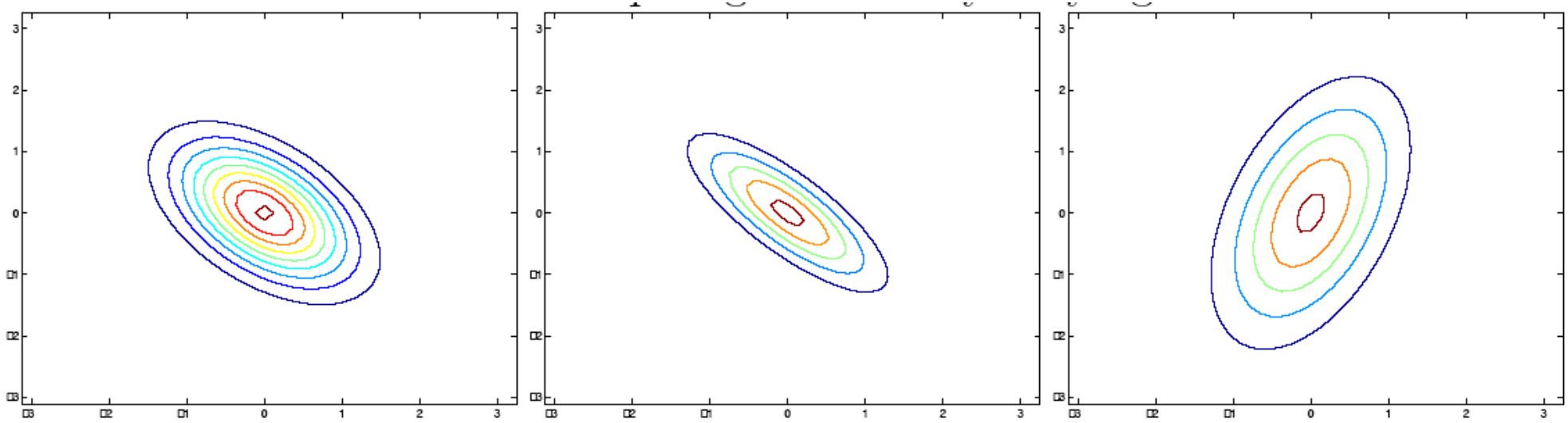


$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}; \quad .\Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

Text and figures from Andrew Ng's lecture notes for CS229

Gaussian Intuitions: off-diagonal and diagonal

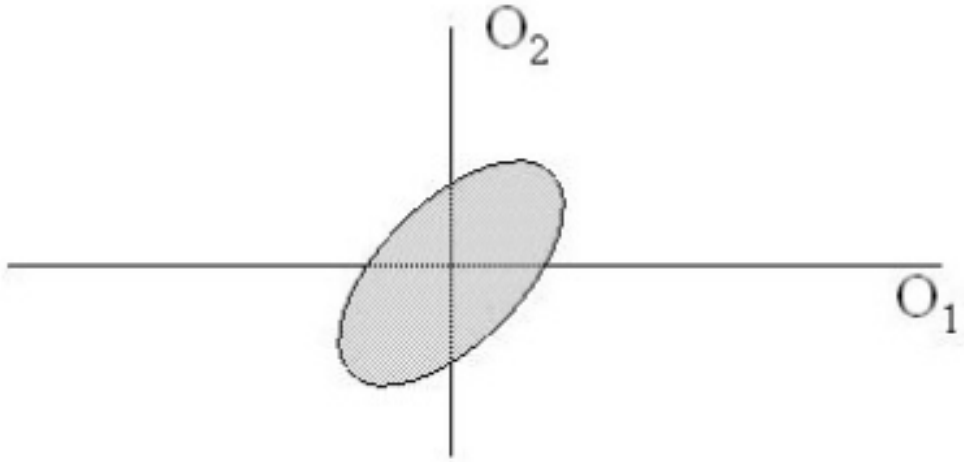
- Decreasing non-diagonal entries (#1-2)
- Increasing variance of one dimension in diagonal (#3)



$$\Sigma = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}; \quad \Sigma = \begin{bmatrix} 1 & -0.8 \\ -0.8 & 1 \end{bmatrix}; \quad .\Sigma = \begin{bmatrix} 3 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

Text and figures from Andrew Ng's lecture notes for CS229

In two dimensions



O_1 and O_2 are correlated – knowing O_1 tells you something about O_2

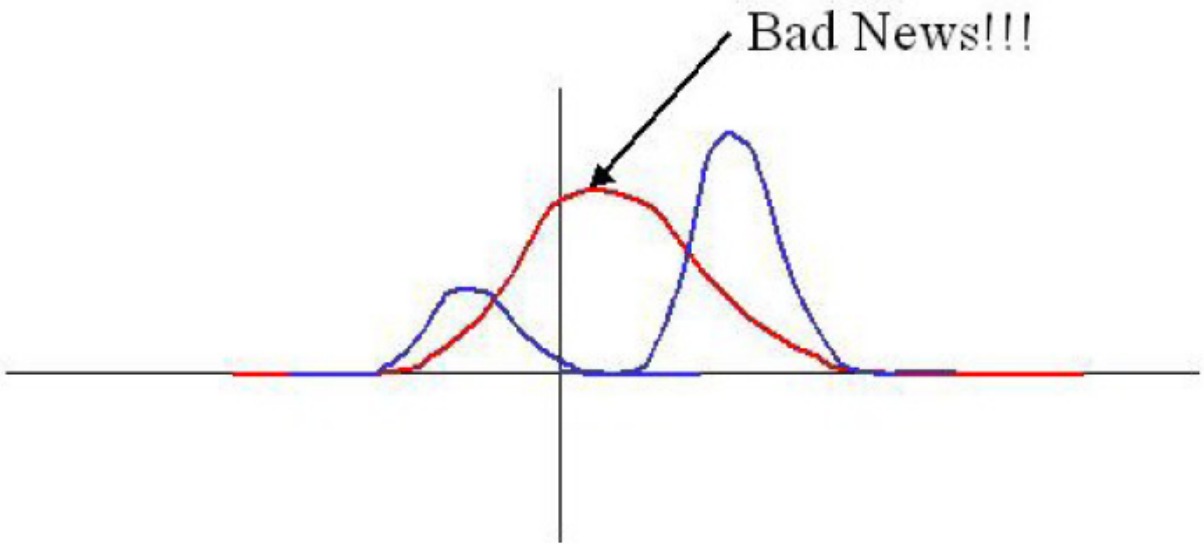
From Chen, Picheny et al lecture slides

But: assume diagonal covariance

- I.e., assume that the features in the feature vector are uncorrelated
- This isn't true for FFT features, but is true for MFCC features, as we will see later.
- Computation and storage much cheaper if diagonal covariance.
  - I.e. only diagonal entries are non-zero
- Diagonal contains the variance of each dimension σ_{ii}^2
- So this means we consider the variance of each acoustic feature (dimension) separately

But we're not there yet

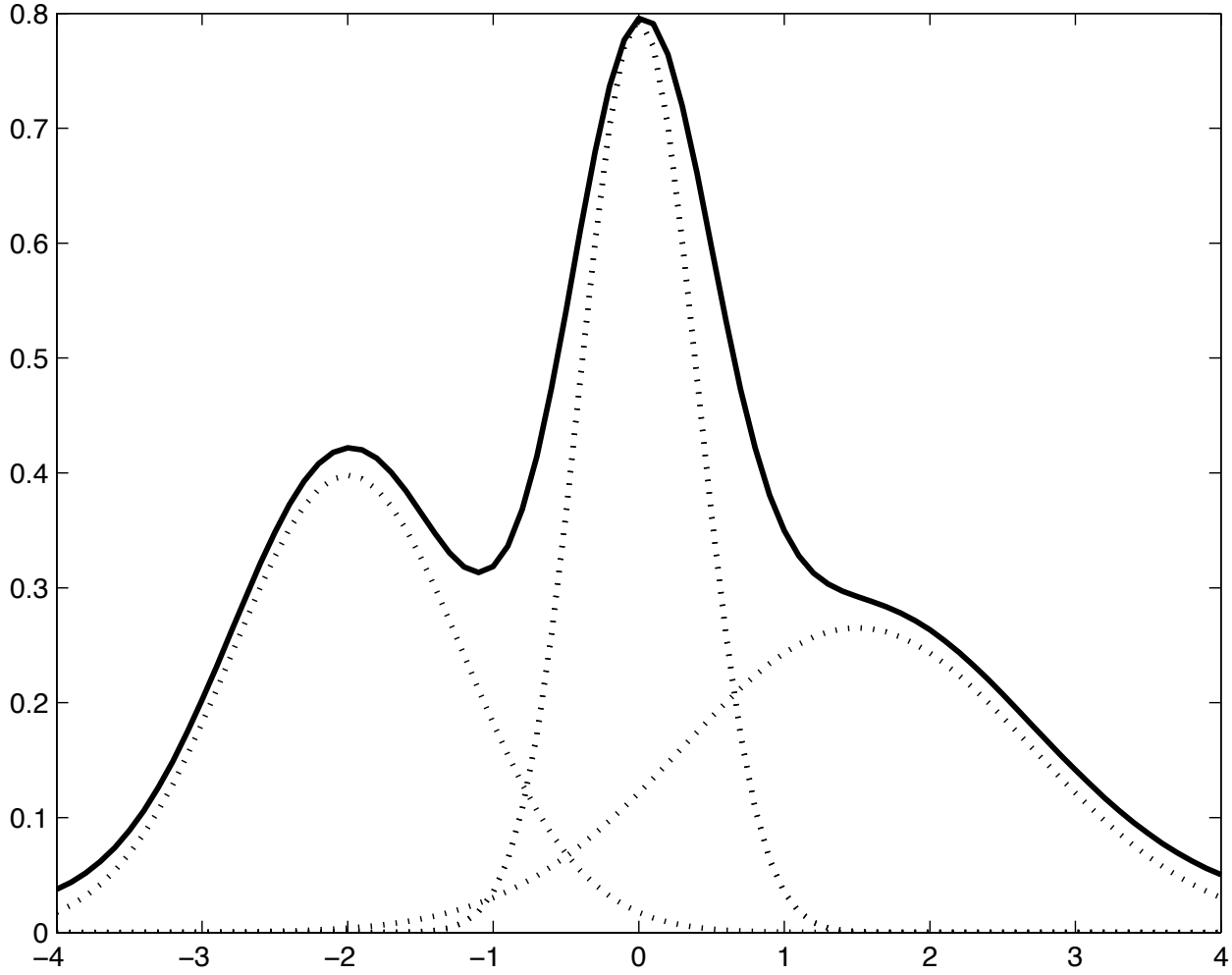
- Single Gaussian may do a bad job of modeling distribution in any dimension:



- Solution: Mixtures of Gaussians

Figure from Chen, Picheney et al slides

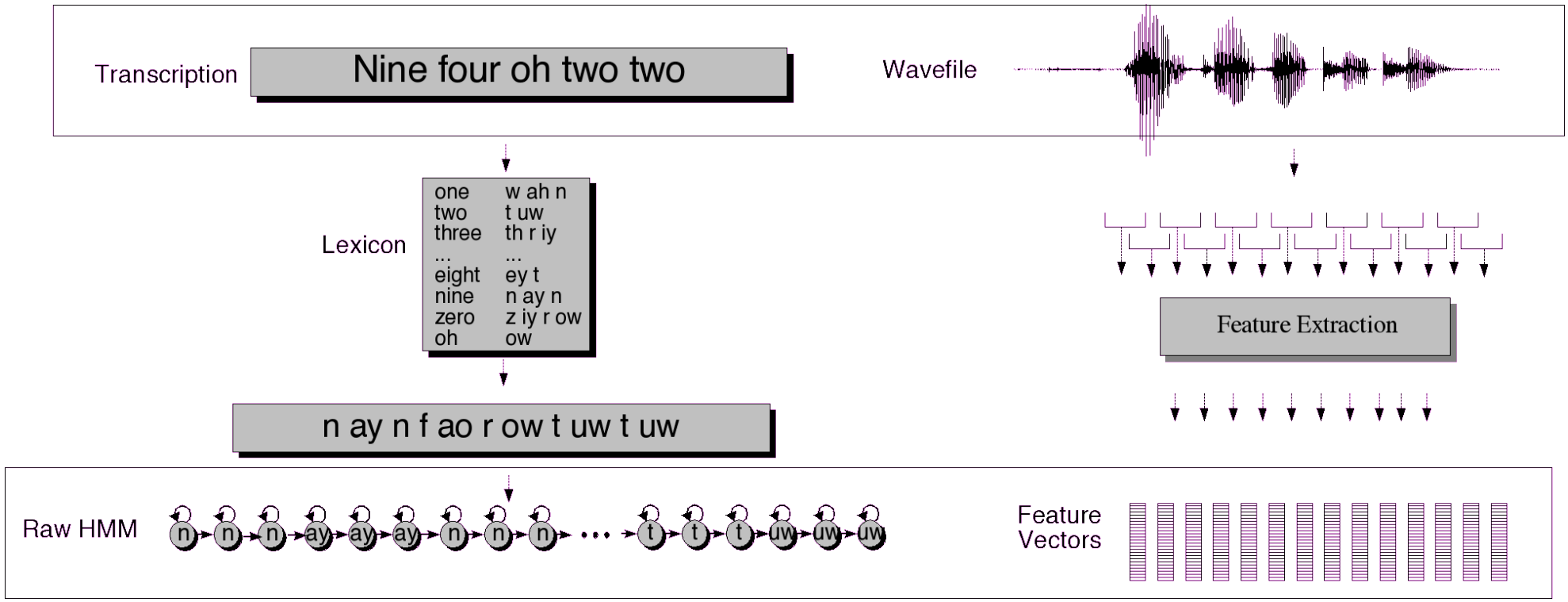
Mixture of Gaussians to model a function



GMMs

- Summary: each state has a likelihood function parameterized by:
  - M Mixture weights
  - M Mean Vectors of dimensionality D
  - Either
    - M Covariance Matrices of DxD
  - Or more likely
    - M Diagonal Covariance Matrices of DxD
    - which is equivalent to
    - M Variance Vectors of dimensionality D

Embedded Training



Initialization: “Flat start”

- Transition probabilities:
  - set to zero any that you want to be “structurally zero”
  - Set the rest to identical values
- Likelihoods:
  - initialize μ and σ of each state to global mean and variance of all training data

Embedded Training

- Given: phoneset, lexicon, transcribed wavefiles
  - Build a whole sentence HMM for each sentence
  - Initialize A probs to 0.5, or to zero
  - Initialize B probs to global mean and variance
  - Run multiple iterations of Baum Welch
    - During each iteration, we compute forward and backward probabilities
  - Use them to re-estimate A and B
  - Run Baum-Welch until convergence.

Viterbi training

- Baum-Welch training says:
  - We need to know what state we were in, to accumulate counts of a given output symbol o_t
  - We'll compute $\gamma_i(t)$, the probability of being in state i at time t, by using forward-backward to sum over all possible paths that might have been in state i and output o_t.
- Viterbi training says:
  - Instead of summing over all possible paths, just take the single most likely path
  - Use the Viterbi algorithm to compute this "Viterbi" path
  - Via "forced alignment"

Forced Alignment

- Computing the “Viterbi path” over the training data is called “forced alignment”
- Because we know which word string to assign to each observation sequence.
- We just don’t know the state sequence.
- So we use a_{ij} to constrain the path to go through the correct words
- And otherwise do normal Viterbi
- Result: state sequence!

Viterbi training equations

Viterbi

$$\hat{a}_{ij} = \frac{n_{ij}}{n_i}$$

For all pairs of emitting states,
1 <= i, j <= N

$$\hat{b}_j(v_k) = \frac{n_j(s.t.o_t = v_k)}{n_j}$$

Baum-Welch

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \gamma_t(i,j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} \gamma_t(i,j)}$$

$$\gamma_t(i,$$

$$\hat{b}_j(v_k) = \frac{\sum_{t=1 s.t. O_t = v_k}^{T} \xi_j(t)\gamma}{\sum_{t=1}^{T} \xi_j(t)\gamma}$$

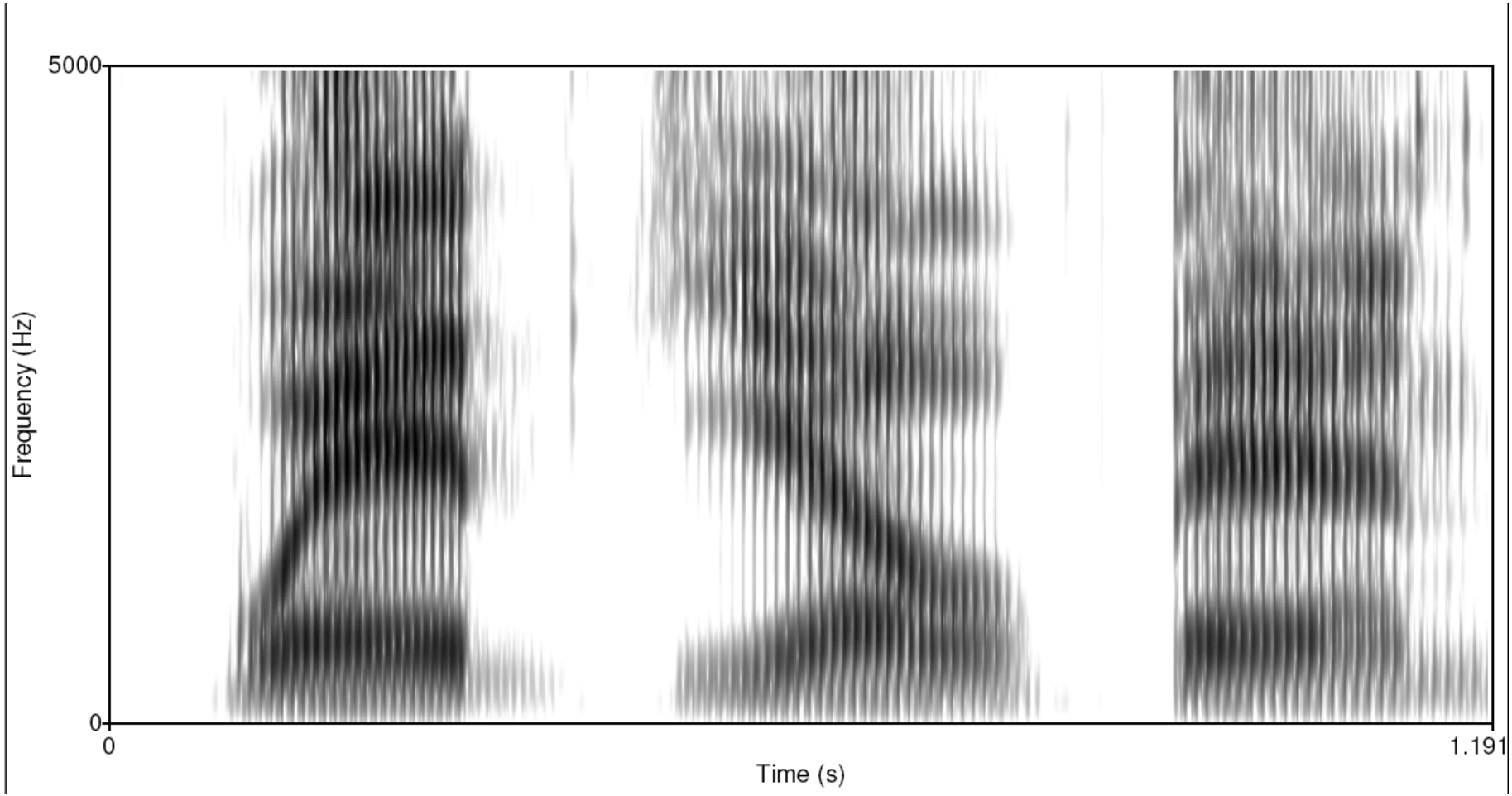
Where n_{ij} is number of frames with transition from i to j in best path
And n_j is number of frames where state j is occupied

Outline for Today

- Acoustic Model
  - Acoustic likelihood for each state using Gaussians and Mixtures of Gaussians
  - Where a state is progressively:
    - CI Subphone (3ish per phone)
    - **Context Dependent (CD) phone (=triphones)**
    - **State-tying of CD phone**
- MFCC feature extraction
- Handling variation
  - MLLR adaptation
  - MAP adaptation (On your own)

Phonetic context: different “eh”s

w eh d y eh l b eh n



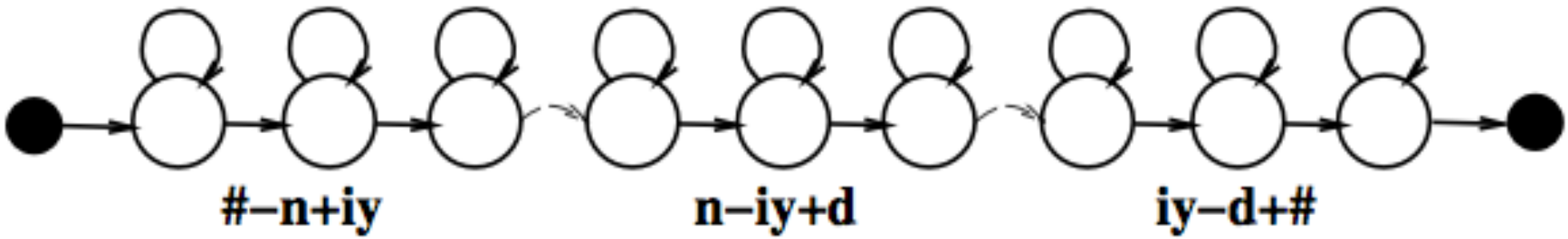
Modeling phonetic context

- The strongest factor affecting phonetic variability is the neighboring phone
- How to model that in HMMs?
- Idea: have phone models which are specific to context.
- Instead of Context-Independent (CI) phones
- We'll have Context-Dependent (CD) phones

CD phones: triphones

- Triphones
- Each triphone captures facts about preceding and following phone
- Monophone:
  - p, t, k
- Triphone:
  - iy-p+aa
  - a-b+c means “phone b, preceding by phone a, followed by phone c”

“Need” with triphone models



Word-Boundary Modeling

- Word-Internal Context-Dependent Models

  ‘OUR LIST’:

  SIL AA+R AA-R L+IH L-IH+S IH-S+T S-T

- Cross-Word Context-Dependent Models

  ‘OUR LIST’:

  SIL-AA+R AA-R+L R-L+IH L-IH+S IH-S+T S-T+SIL

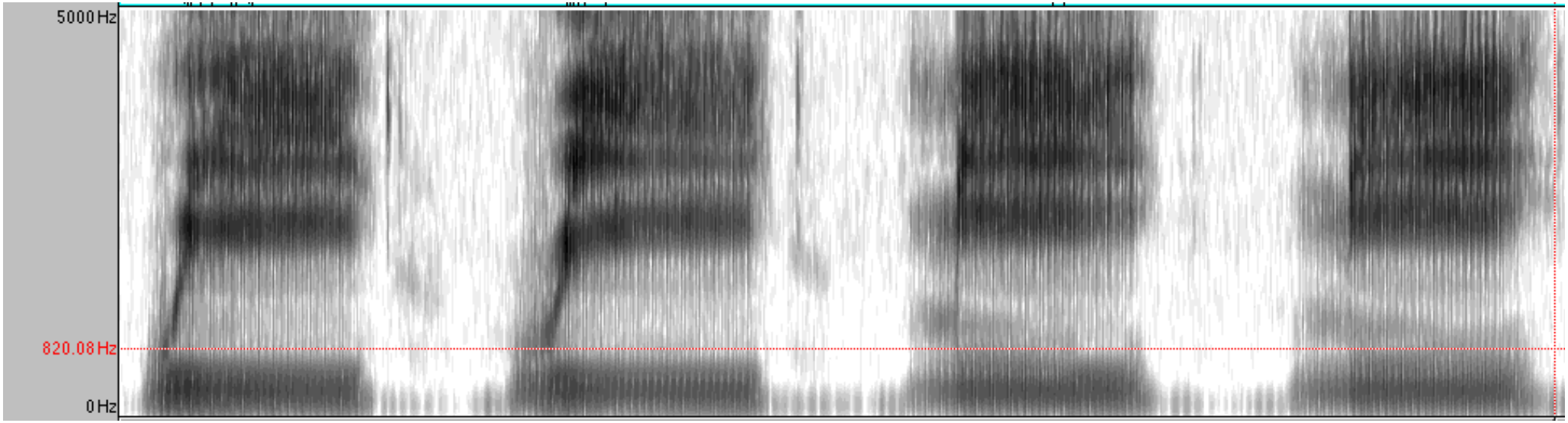
- Dealing with cross-words makes decoding harder!

Implications of Cross-Word Triphones

- Possible triphones: 50x50x50=125,000
- How many triphone types actually occur?
- 20K word WSJ Task, numbers from Young et al
- Cross-word models: need 55,000 triphones
- But in training data only 18,500 triphones occur!
- Need to generalize models

Modeling phonetic context: some contexts look similar

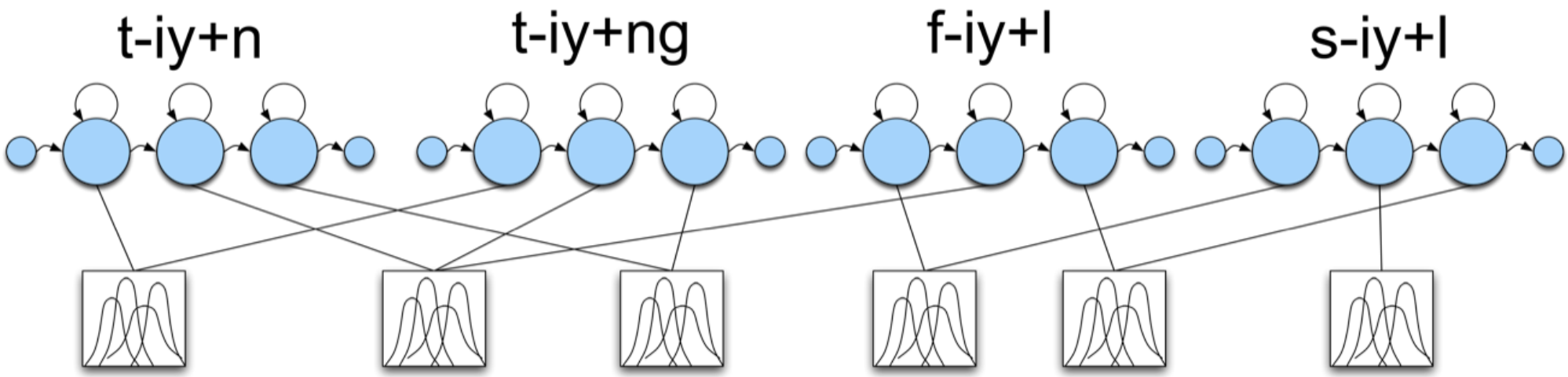
w iy r iy m iy n iy



Solution: State Tying

- Young, Odell, Woodland 1994
- Decision-Tree based clustering of triphone states
- States which are clustered together will share their Gaussians
- We call this “state tying”, since these states are “tied together” to the same Gaussian.

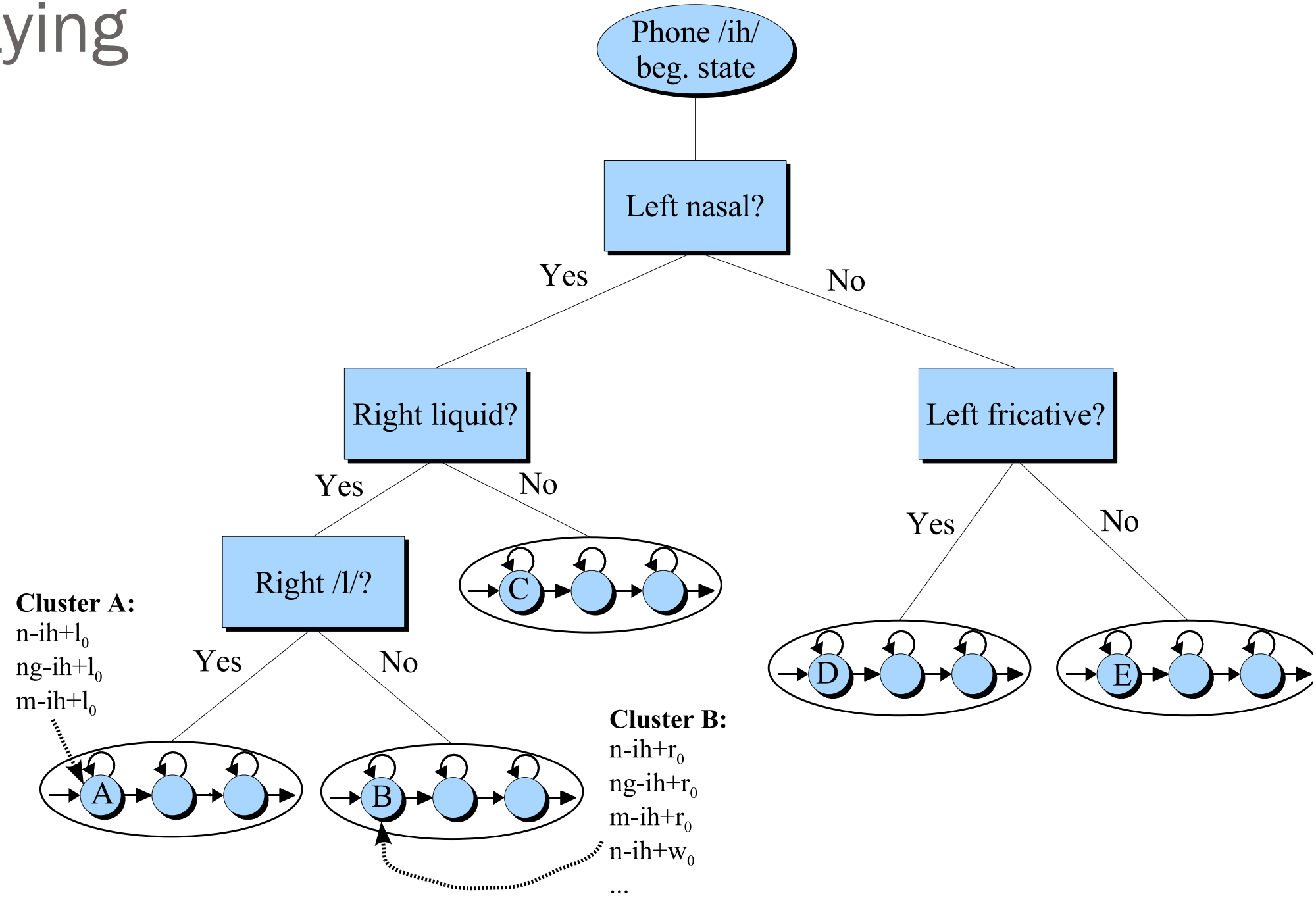
Young et al state tying



State tying/clustering

- How do we decide which triphones to cluster together?
- Use phonetic features (or 'broad phonetic classes')
  - Stop
  - Nasal
  - Fricative
  - Sibilant
  - Vowel
  - Lateral

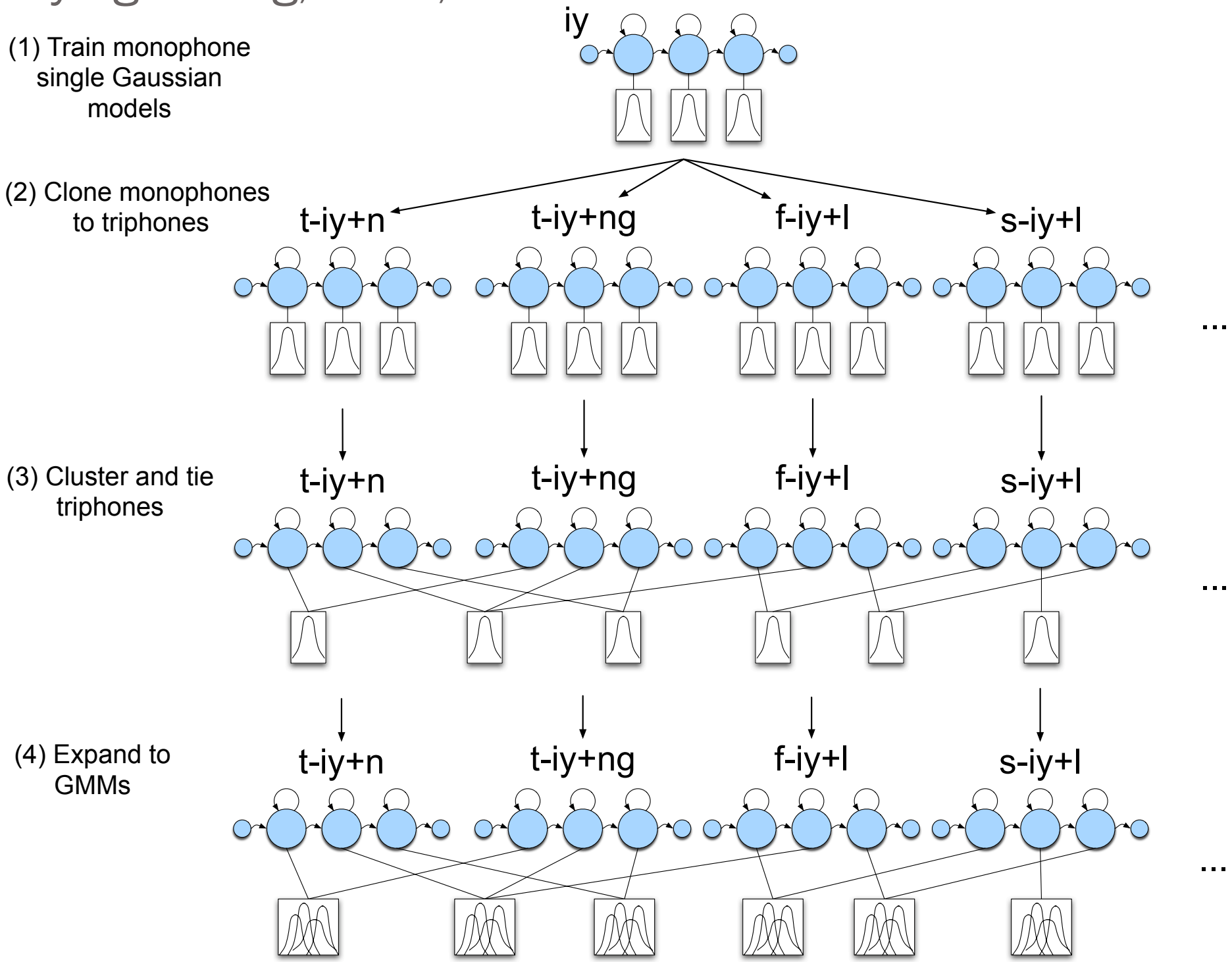
Decision tree for clustering triphones for tying



Decision tree for clustering triphones for tying

Feature	Phones
Stop	b d g k p t
Nasal	m n ng
Fricative	ch dh f jh s sh th v z zh
Liquid	l r w y
Vowel	aa ae ah ao aw ax axr ay eh er ey ih ix iy ow oy uh uw
Front Vowel	ae eh ih ix iy
Central Vowel	aa ah ao axr er
Back Vowel	ax ow uh uw
High Vowel	ih ix iy uh uw
Rounded	ao ow oy uh uw w
Reduced	ax axr ix
Unvoiced	ch f hh k p s sh t th
Coronal	ch d dh jh l n r s sh t th z zh

State Tying: Young, Odell, Woodland 1994



Summary: Acoustic Modeling for LVCSR

- Increasingly sophisticated models
- For each state:
  - Gaussians
  - Multivariate Gaussians
  - Mixtures of Multivariate Gaussians
- Where a state is progressively:
  - CI Phone
  - CI Subphone (3ish per phone)
  - CD phone (=triphones)
  - State-tying of CD phone
- Neural network acoustic models after all of the above
- Viterbi training

Summary: ASR Architecture

- Five easy pieces: ASR Noisy Channel architecture
  - Feature Extraction:
    - 39 "MFCC" features
  - Acoustic Model:
    - Gaussians for computing p(o|q)
  - Lexicon/Pronunciation Model
    - HMM: what phones can follow each other
  - Language Model
    - N-grams for computing p(wi|wi-1)
  - Decoder
    - Viterbi algorithm: dynamic programming for combining all these to get word sequence from speech!

Outline for Today

- Acoustic Model
  - Acoustic likelihood for each state using Gaussians and Mixtures of Gaussians
  - Where a state is progressively:
    - CI Subphone (3ish per phone)
    - Context Dependent (CD) phone (=triphones)
    - State-tying of CD phone
- **MFCC feature extraction**
- Handling variation
  - MLLR adaptation
  - MAP adaptation (On your own)

Discrete Representation of Signal

- Represent continuous signal into discrete form.

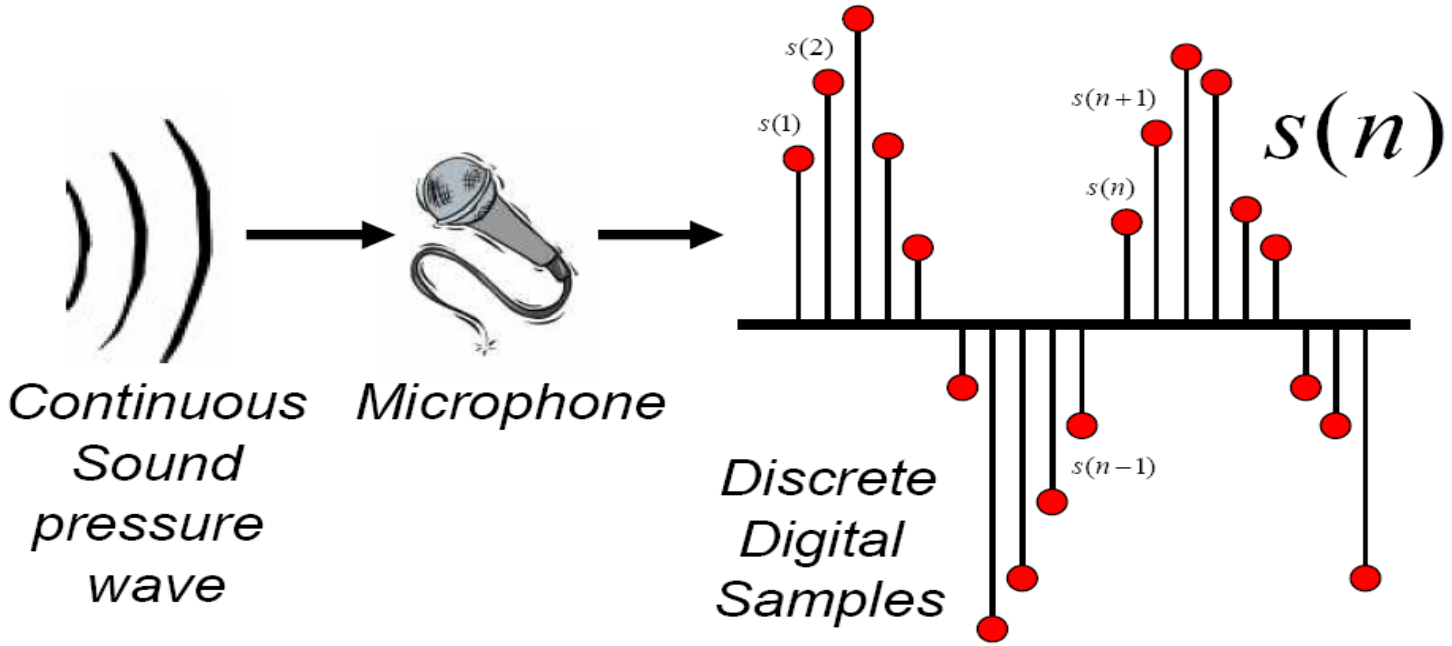
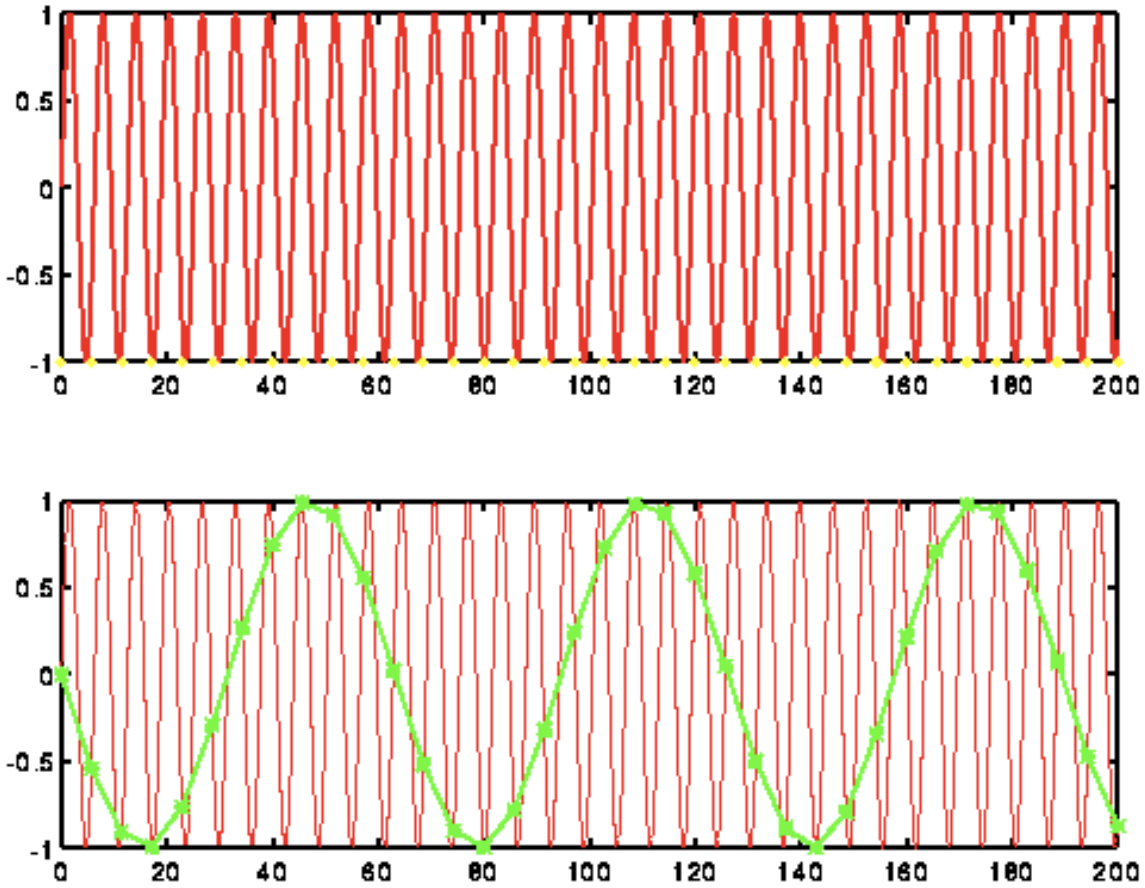


Image from Bryan Pellom

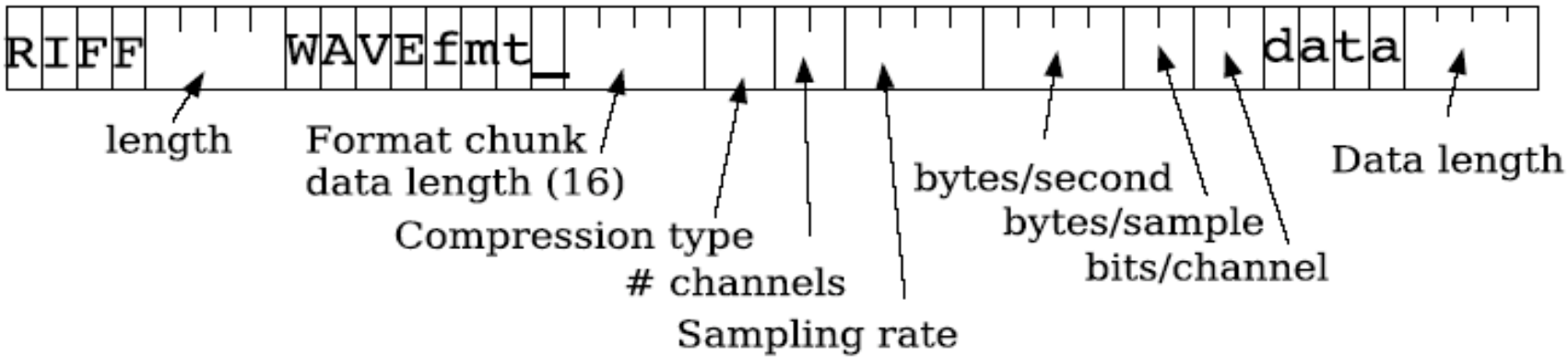
Sampling

If measure at green dots, will see a lower frequency wave and miss the correct higher frequency one!

Original signal in red:



WAV format



Many formats, trade-offs in compression, quality

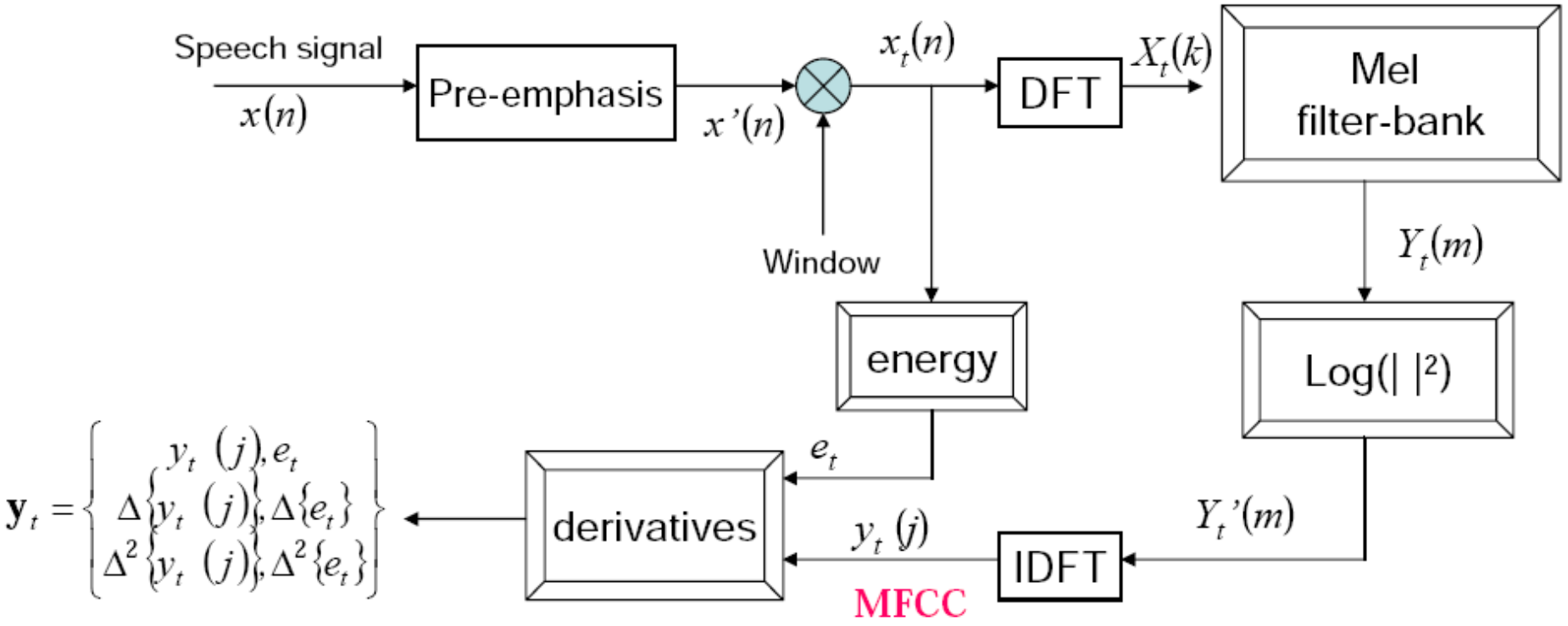
Nice sound manipulation tool: Sox

http://sox.sourceforge.net/

convert speech formats

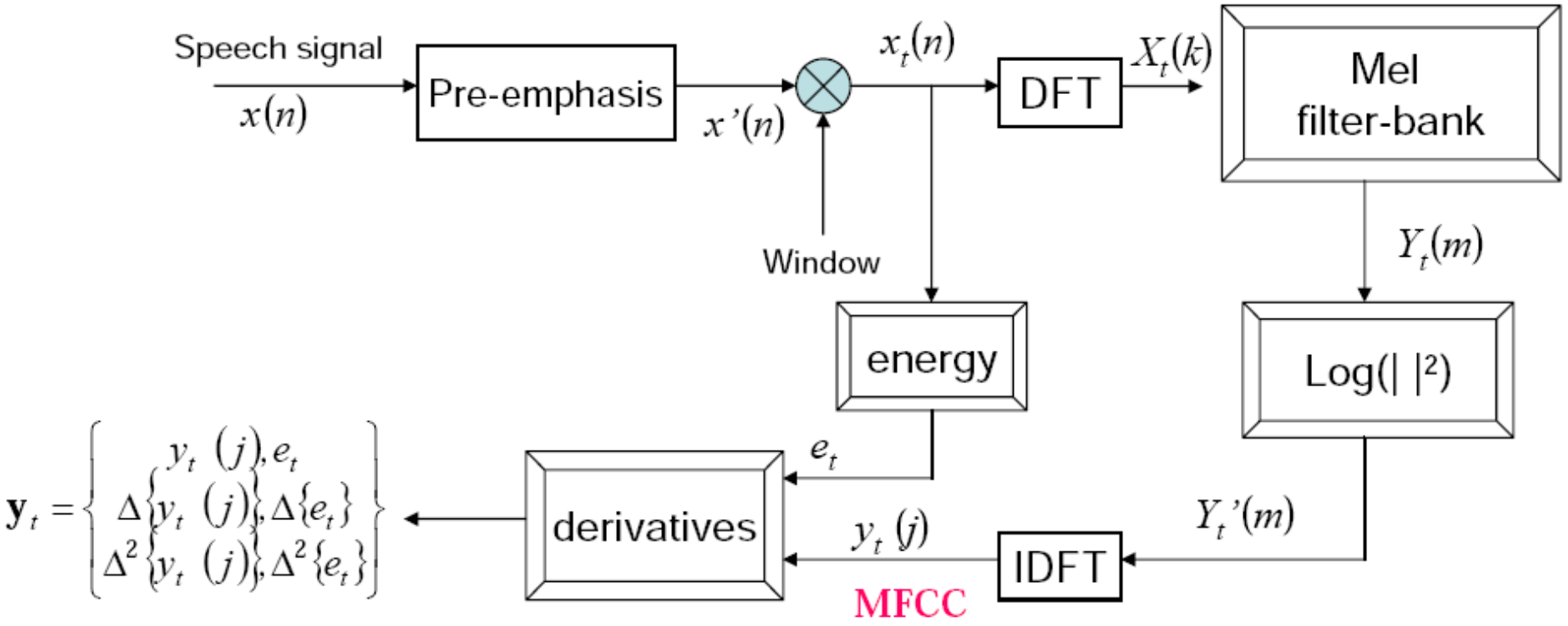
MFCC

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

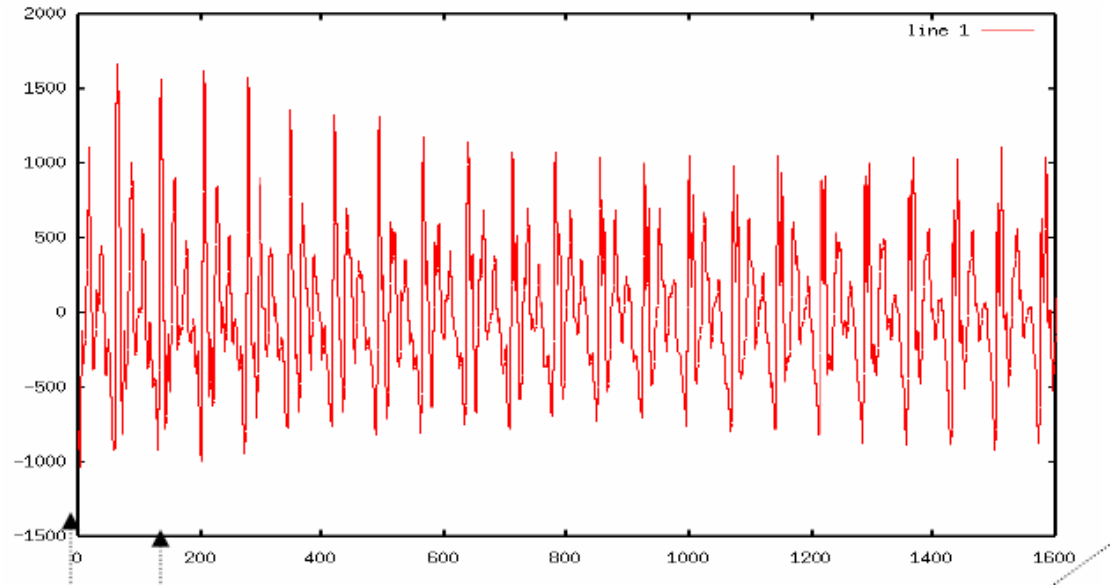


MFCC

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

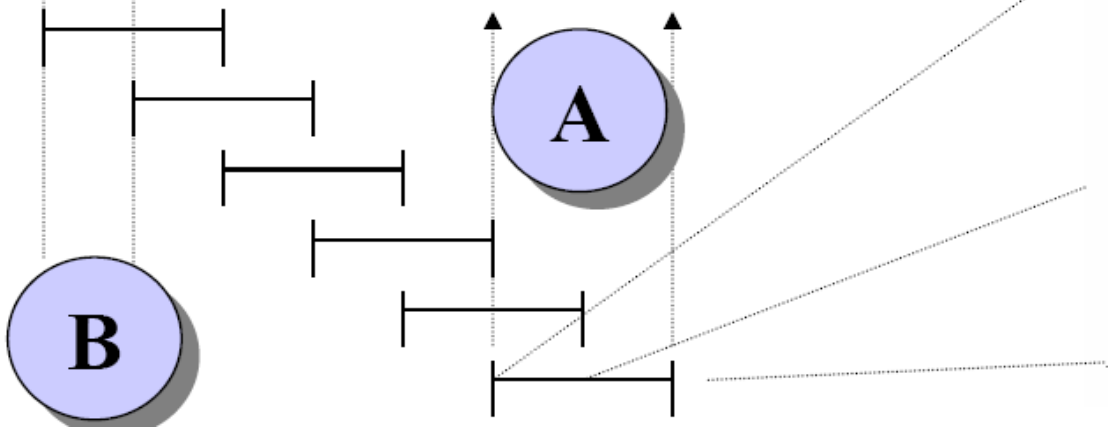


Windowing



A ~ 20 – 25 ms

B ~ 10 ms



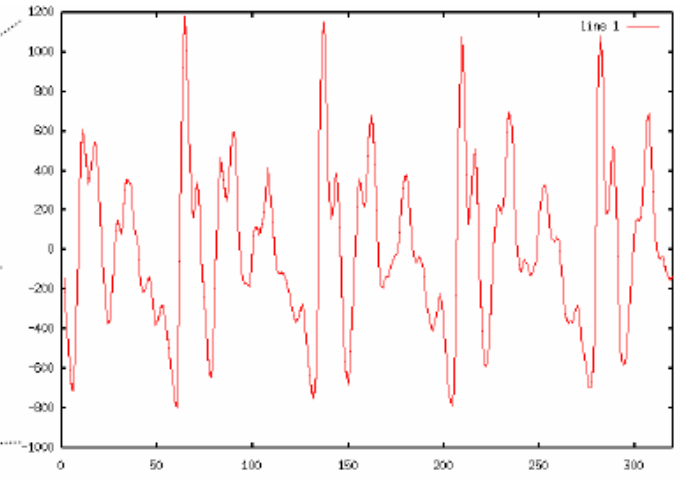
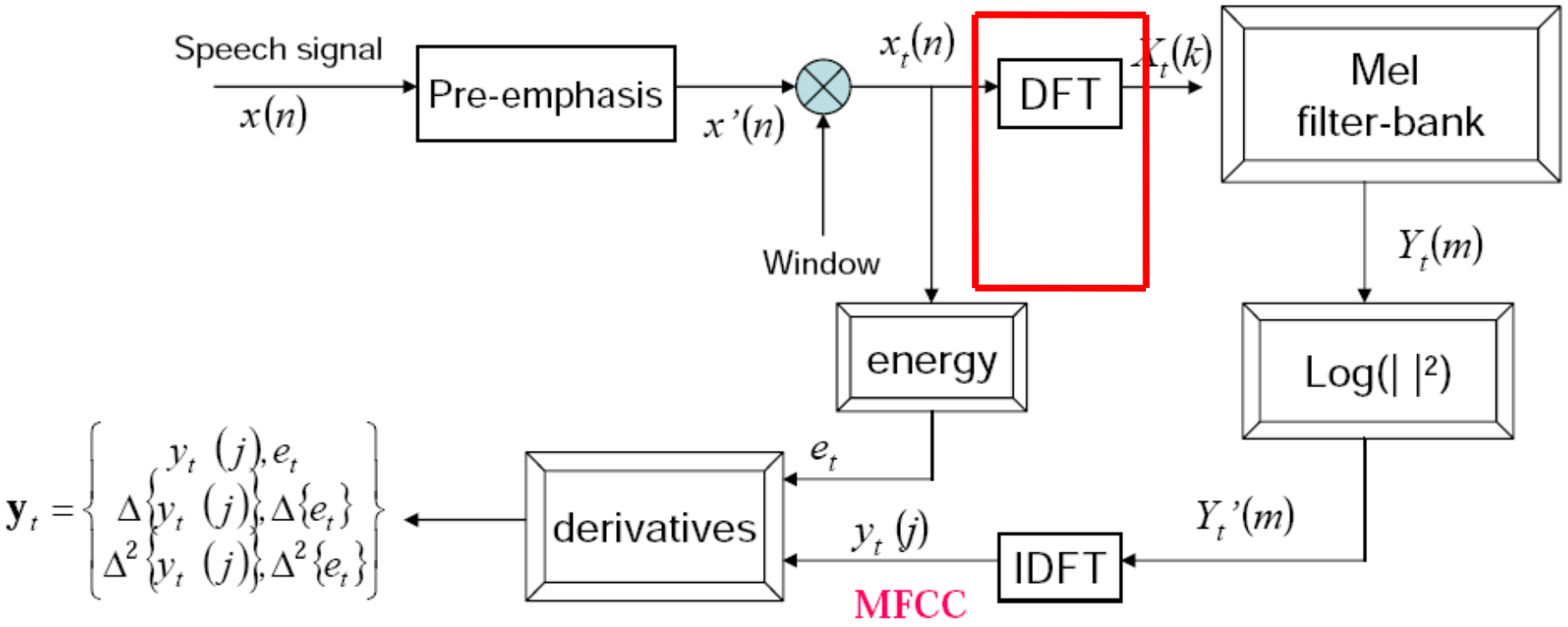


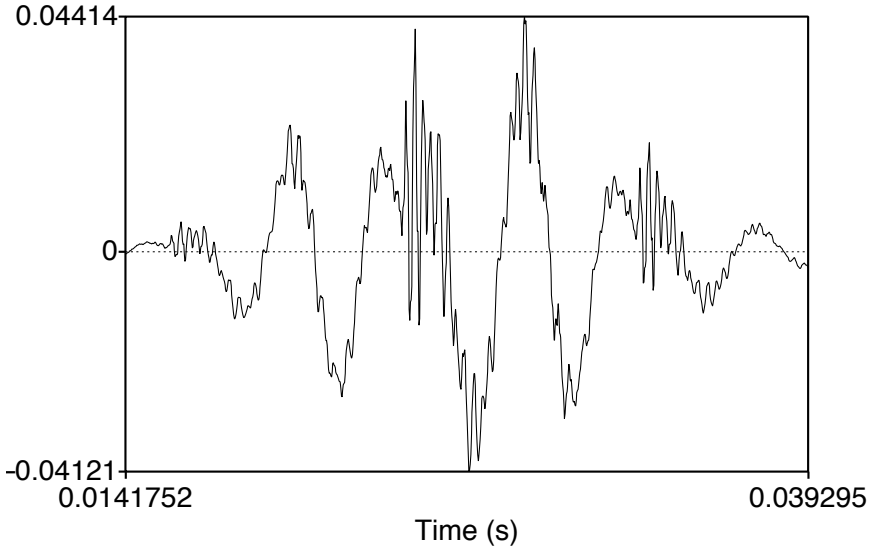
Image from Bryan Pellom

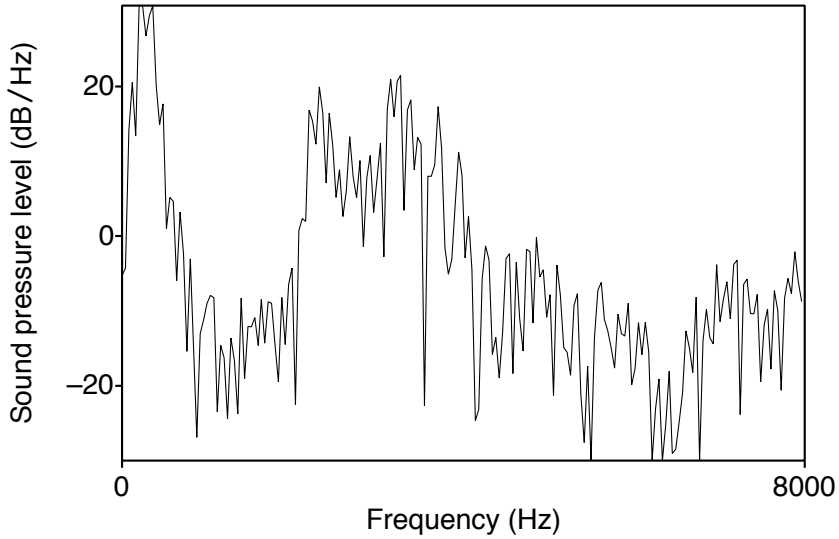
MFCC


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
$y_t(j)$
energy
e_t
derivatives
$\mathbf{y}_t = \begin{Bmatrix} y_t(j), e_t \\ \Delta\{y_t(j)\}, \Delta\{e_t\} \\ \Delta^2\{y_t(j)\}, \Delta^2\{e_t\} \end{Bmatrix}$
MFCC

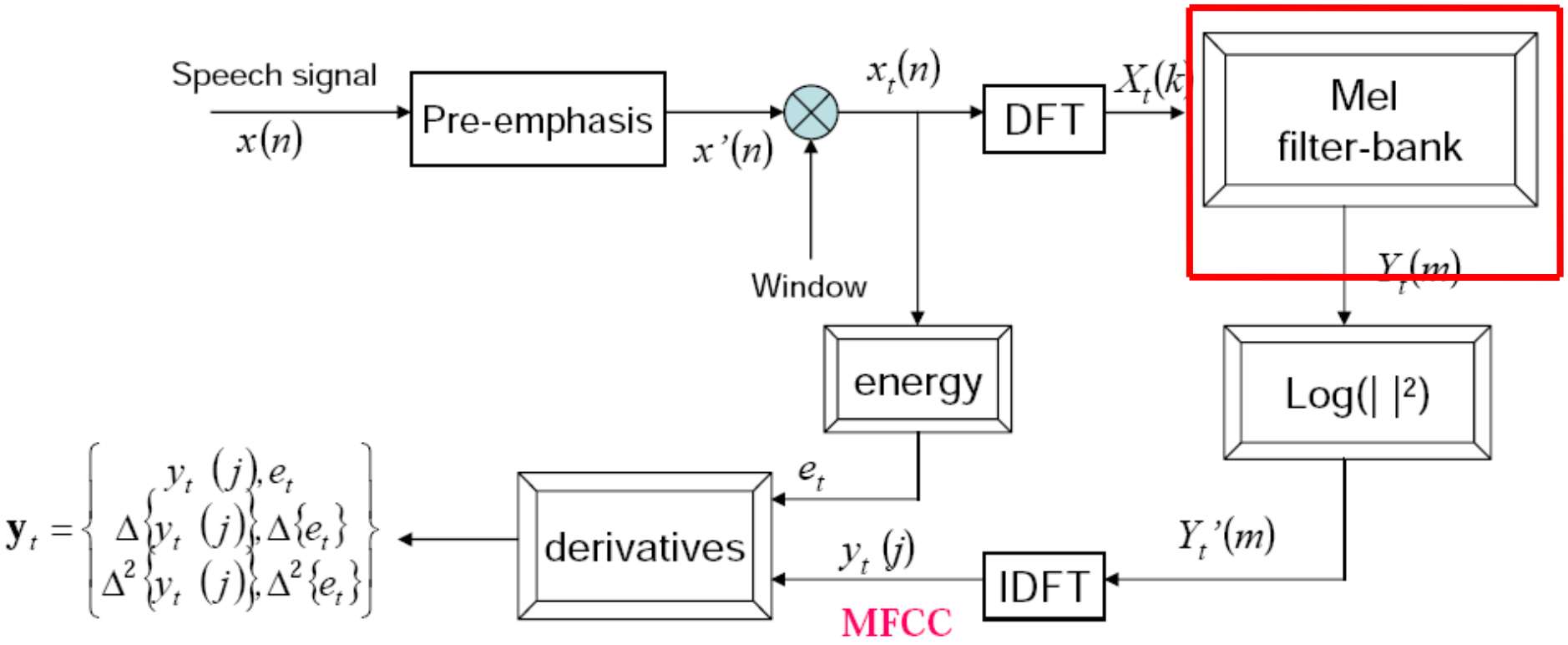
Discrete Fourier Transform computing a spectrum

- A 25 ms Hamming-windowed signal from [iy]
  - And its spectrum as computed by DFT (plus other smoothing)



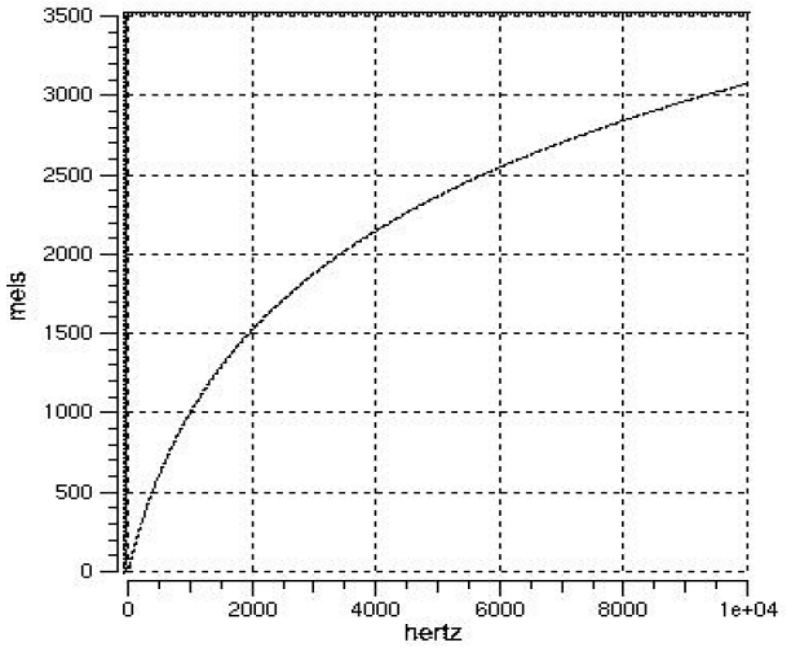


MFCC


Speech signal
x(n)
Pre-emphasis
x'(n)
Window
x_t(n)
DFT
X_t(k)
Mel
filter-bank
Y_t(m)
Log(| |²)
Y_t'(m)
IDFT
y_t (j)
MFCC
energy
e_t
derivatives
y_t (j), e_t
Δ{y_t (j)}, Δ{e_t}
Δ²{y_t (j)}, Δ²{e_t}
y_t =

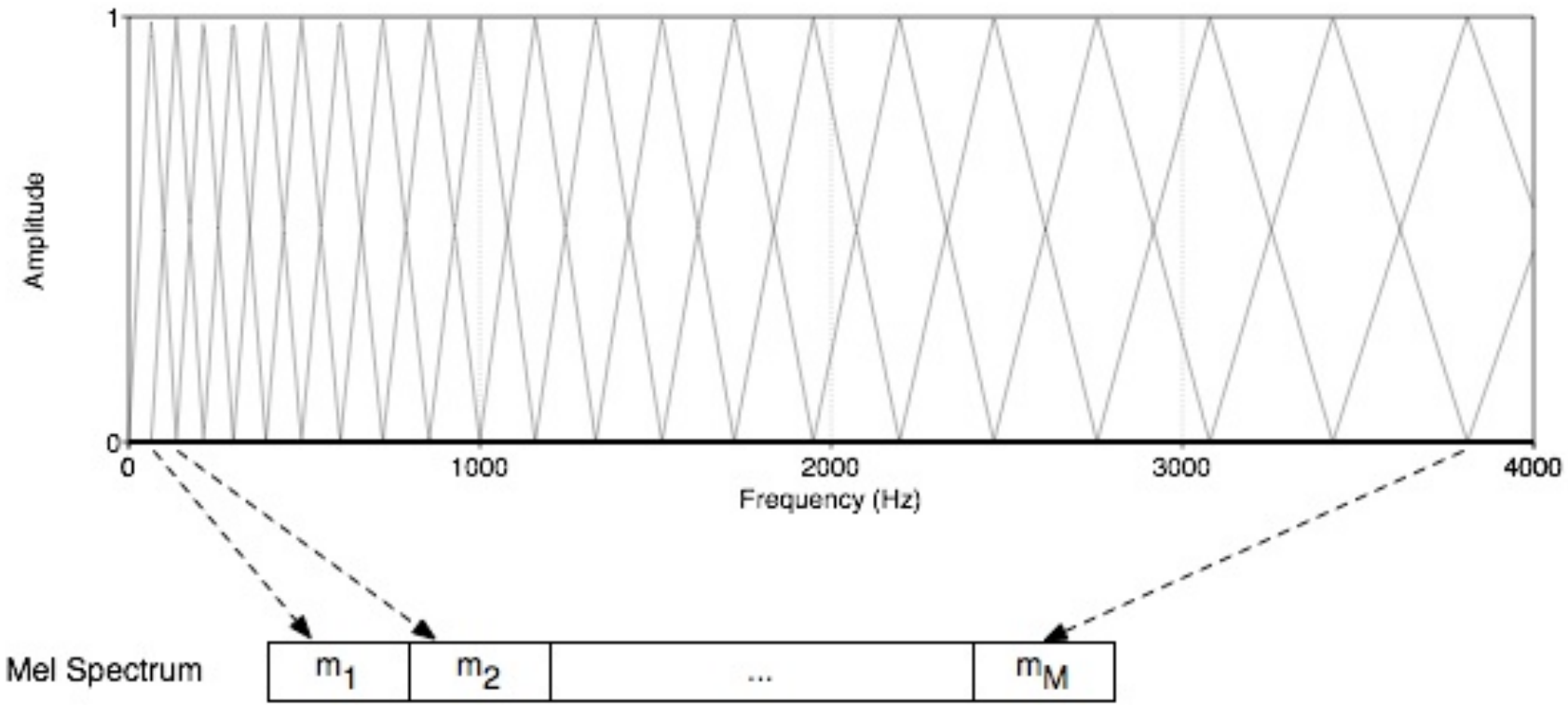
Mel-scale

- 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:



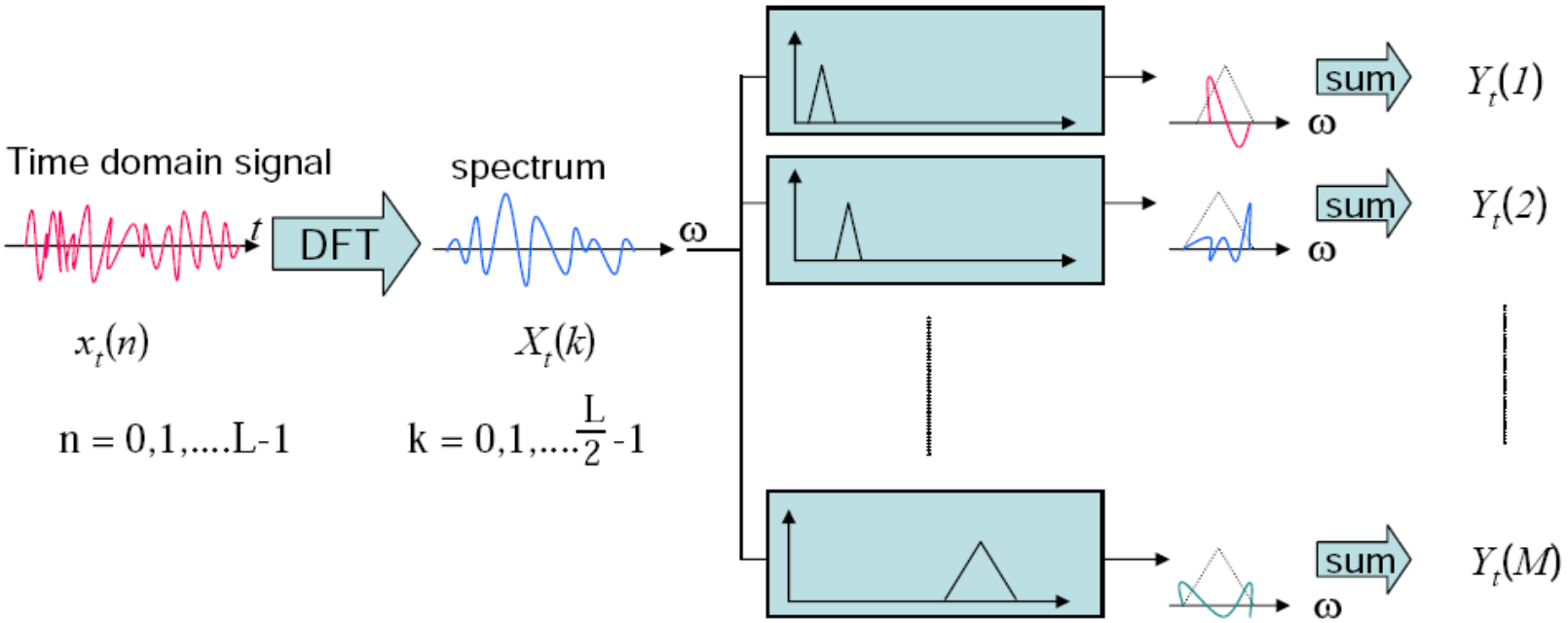
Mel Filter Bank Processing

- Mel Filter bank
  - Roughly uniformly spaced before 1 kHz
  - logarithmic scale after 1 kHz

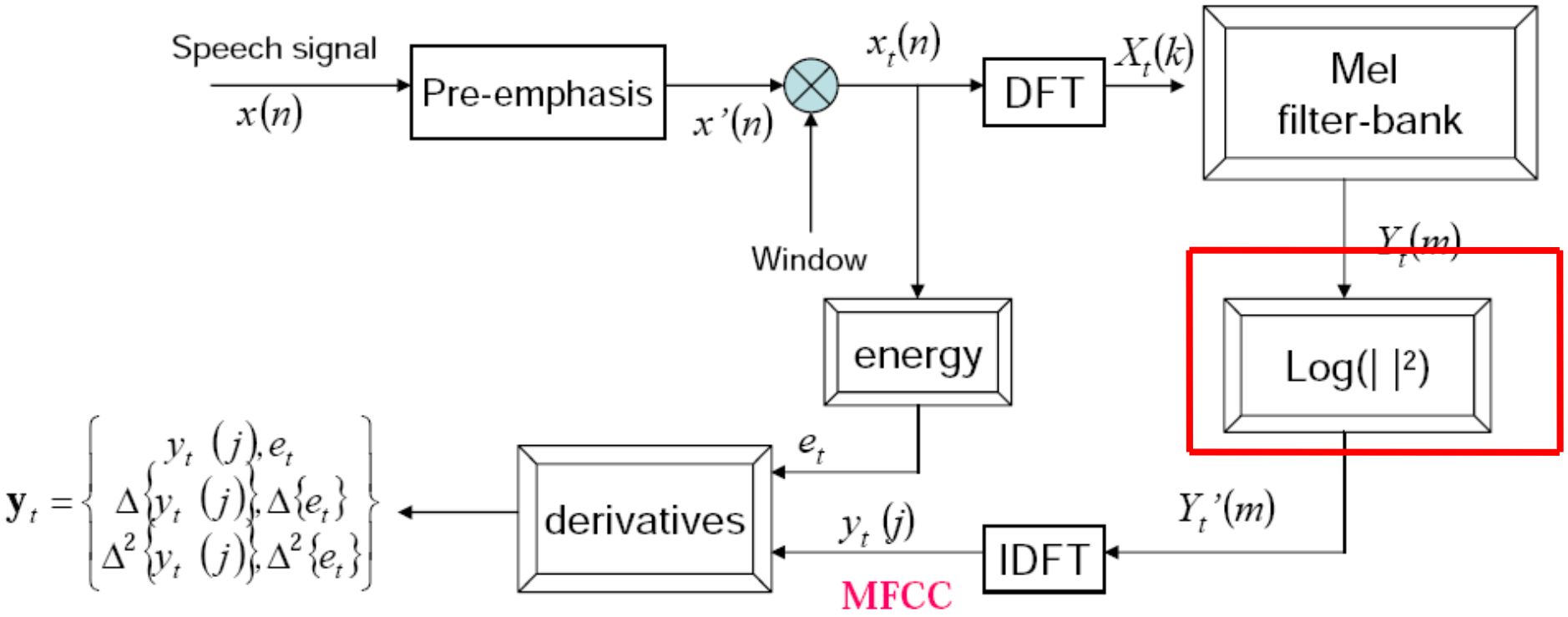


Mel-filter Bank Processing

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

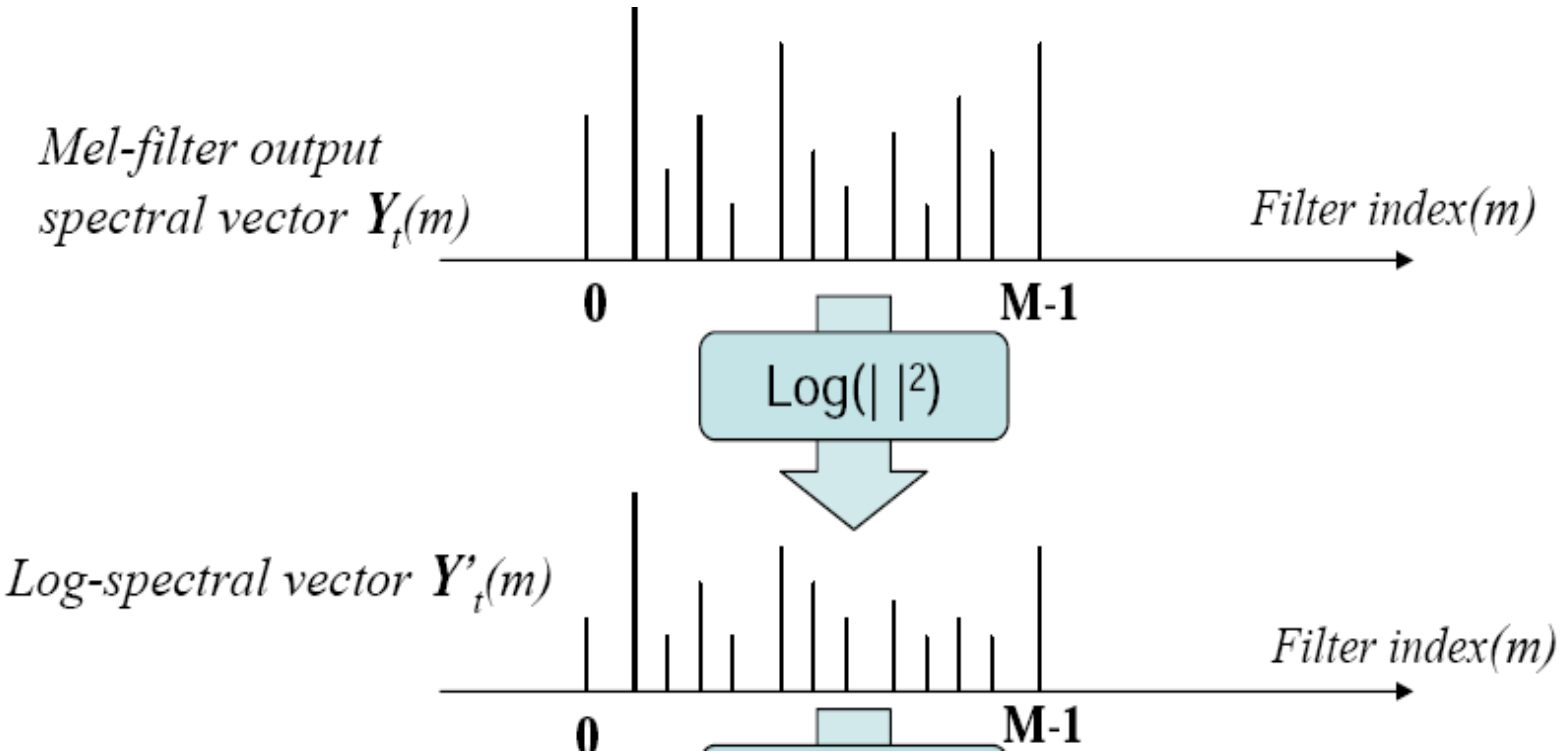


MFCC

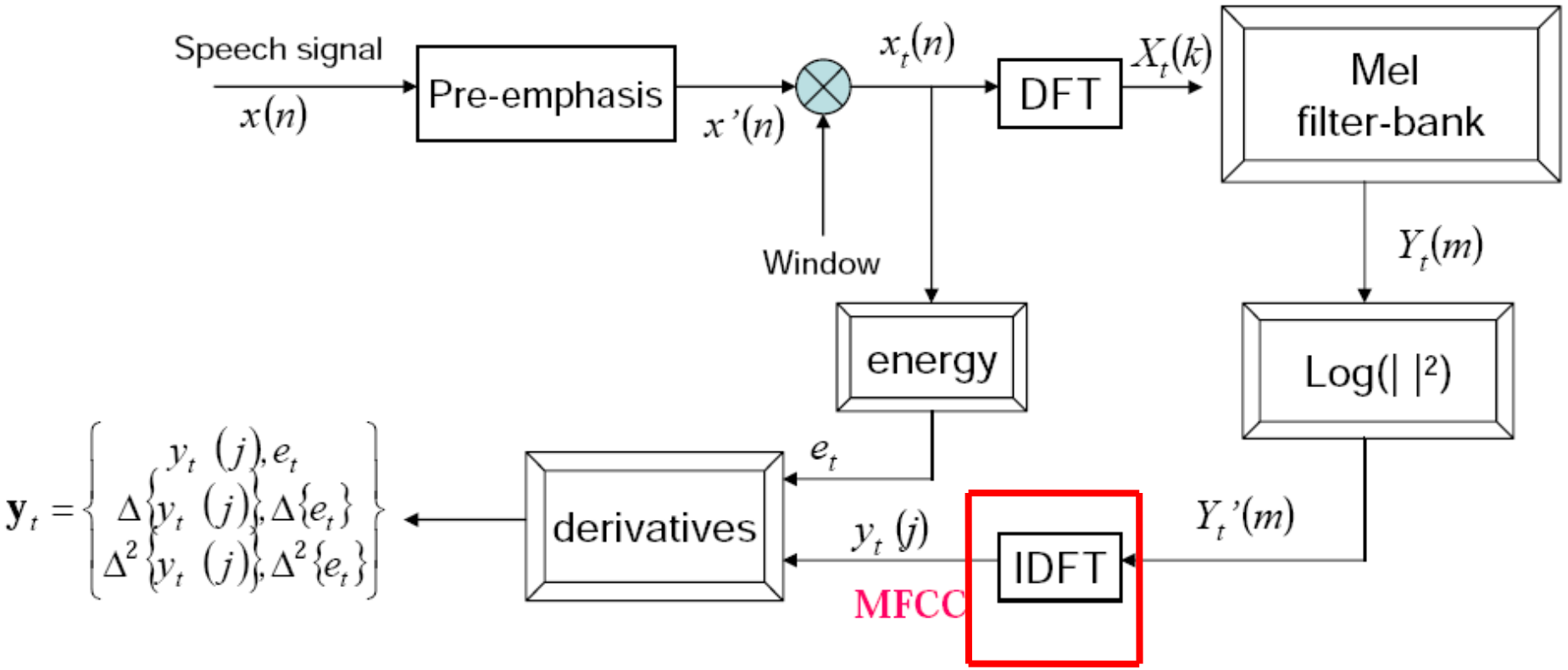

Speech signal
$x(n)$
Pre-emphasis
$x'(n)$
Window
$x_t(n)$
DFT
$X_t(k)$
Mel
filter-bank
$Y_t(m)$
Log(| |²)
$Y_t'(m)$
IDFT
$y_t(j)$
MFCC
energy
e_t
derivatives
$\mathbf{y}_t = \begin{Bmatrix} y_t(j), e_t \\ \Delta\{y_t(j)\}, \Delta\{e_t\} \\ \Delta^2\{y_t(j)\}, \Delta^2\{e_t\} \end{Bmatrix}$

Log energy computation

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



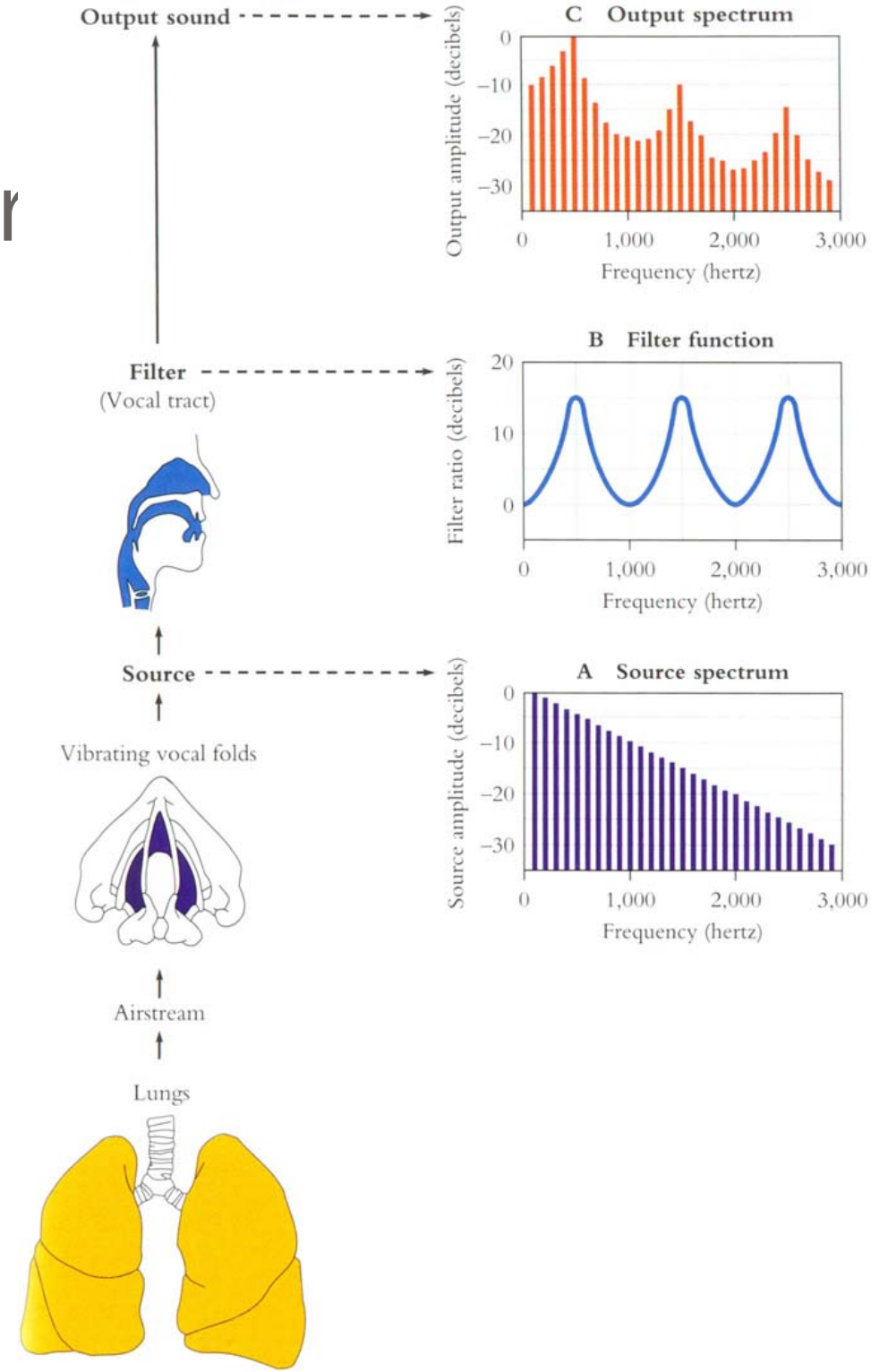
MFCC


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
$\mathbf{y}_t = \left\{ \begin{matrix} y_t(j), e_t \\ \Delta\{y_t(j)\}, \Delta\{e_t\} \\ \Delta^2\{y_t(j)\}, \Delta^2\{e_t\} \end{matrix} \right\}$

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
- Remember articulatory facts from lecture 2:
  - The vocal cord vibrations create harmonics
  - The mouth is an amplifier
  - Depending on shape of oral cavity, some harmonics are amplified more than others

George Miller figur

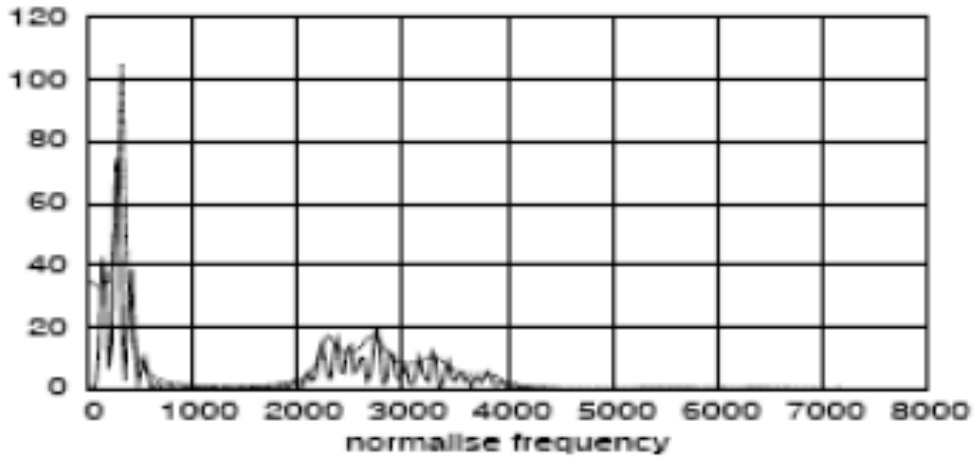


We care about the filter not the source

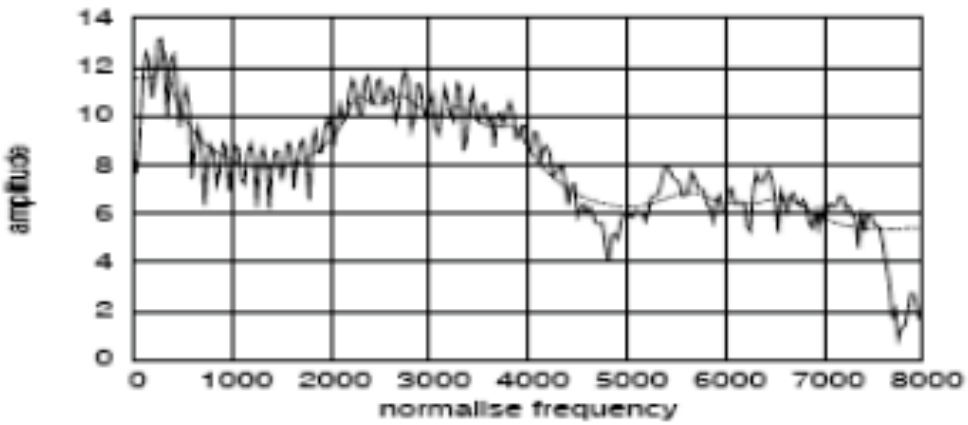
- Most characteristics of the source
  - F0
  - Details of glottal pulse
- Don’t matter for phone detection
- What we care about is the filter
  - The exact position of the articulators in the oral tract
- So we want a way to separate these
  - And use only the filter function

The Cepstrum

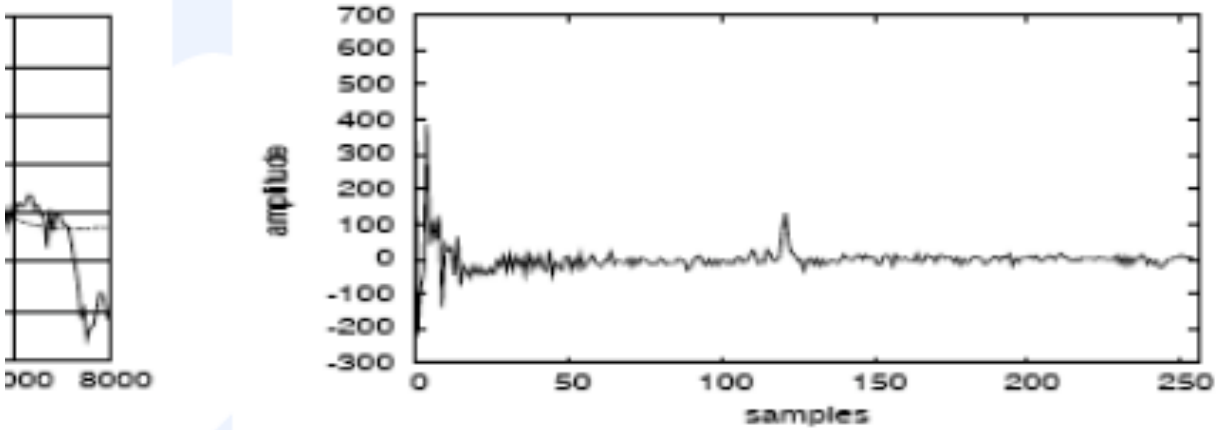
- The spectrum of the log of the spectrum



Spectrum



Log spectrum

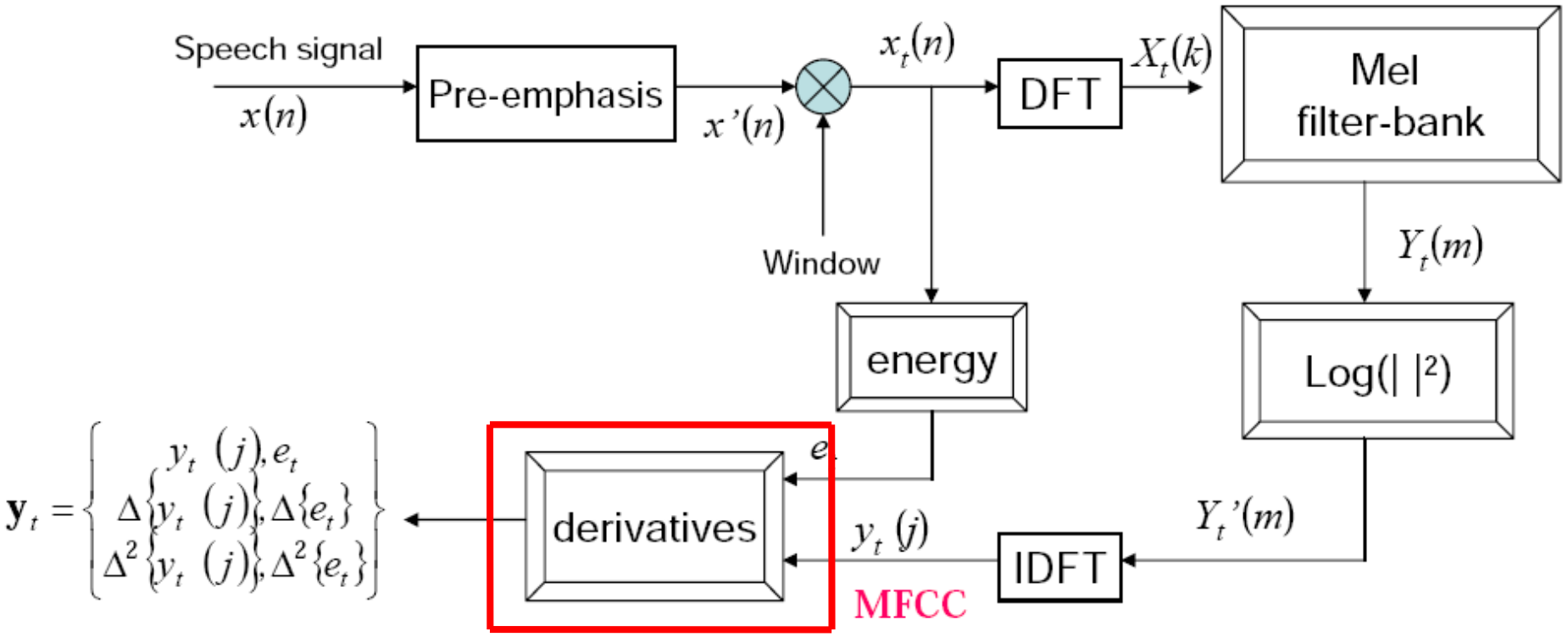


Spectrum of log spectrum

Another advantage of the Cepstrum

- DCT produces highly uncorrelated features
- If we use only the diagonal covariance matrix for our Gaussian mixture models, we can only handle uncorrelated features.
- 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
$Y_t(m)$
Log(| |²)
$Y_t'(m)$
IDFT
$y_t(j)$
MFCC
energy
e_t
derivatives
$\mathbf{y}_t = \begin{Bmatrix} y_t(j), e_t \\ \Delta\{y_t(j)\}, \Delta\{e_t\} \\ \Delta^2\{y_t(j)\}, \Delta^2\{e_t\} \end{Bmatrix}$

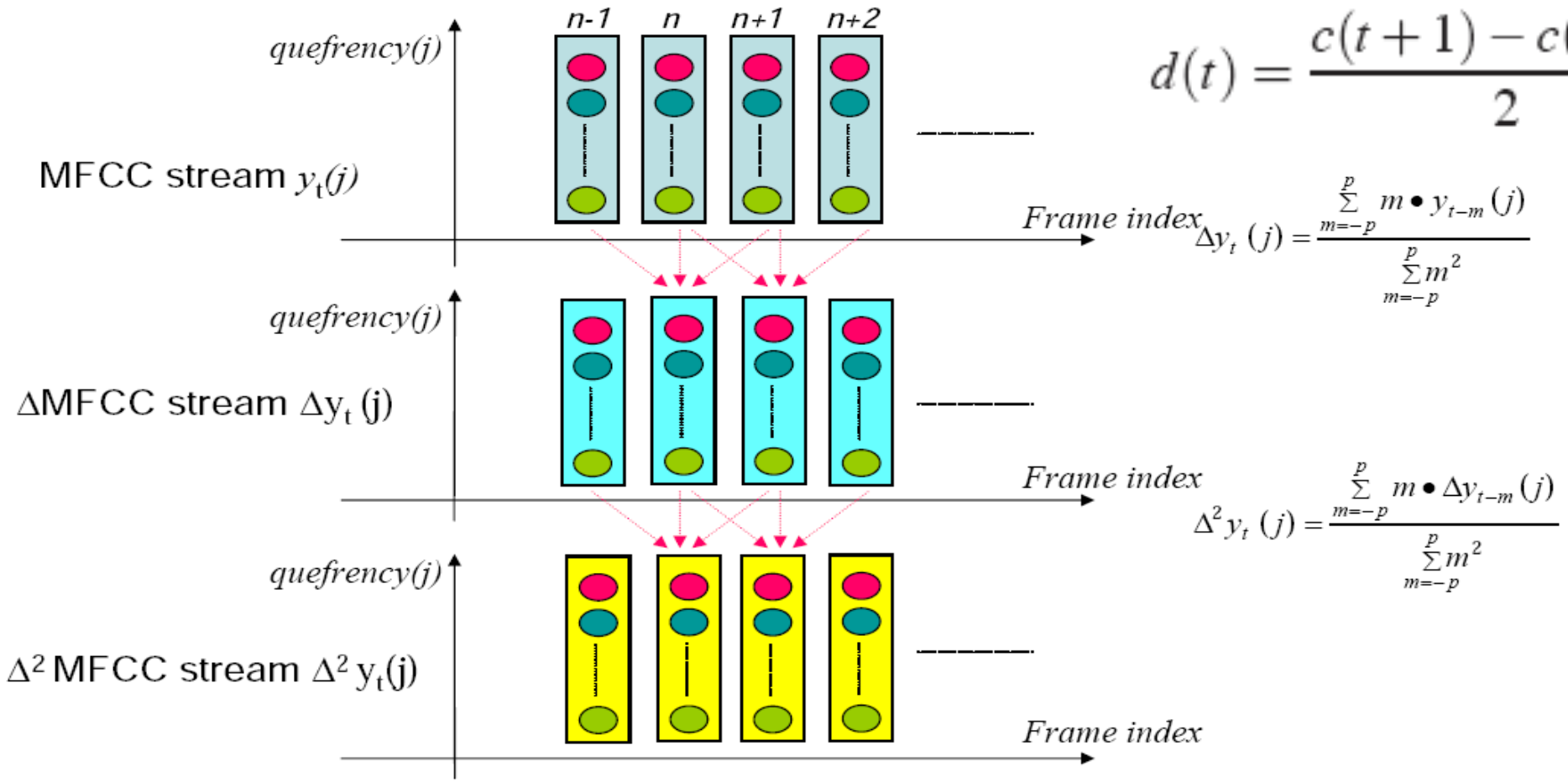
“Delta” features

- Speech signal is not constant
  - slope of formants,
  - change from stop burst to release

- So in addition to the cepstral features
- Need to model changes in the cepstral features over time.
  - “delta features”
  - “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

Why is MFCC so popular?

- Efficient to compute
- Incorporates a perceptual Mel frequency scale
- Separates the source and filter
- IDFT(DCT) decorrelates the features
  - Necessary for diagonal assumption in HMM modeling
- There are alternatives like PLP
- Choice matters less for neural network acoustic models (in future lectures)

Outline for Today

- Acoustic Model
  - Acoustic likelihood for each state using Gaussians and Mixtures of Gaussians
  - Where a state is progressively:
    - CI Subphone (3ish per phone)
    - CD phone (=triphones)
    - State-tying of CD phone
- MFCC feature extraction
- **Handling variation**
  - MLLR adaptation
  - MAP adaptation (On your own)

Acoustic Model Adaptation

- Shift the means and variances of Gaussians to better match the input feature distribution
  - Maximum Likelihood Linear Regression (MLLR)
  - Maximum A Posteriori (MAP) Adaptation
- For both speaker adaptation and environment adaptation
- Widely used!

Maximum Likelihood Linear Regression (MLLR)

- Leggetter, C.J. and P. Woodland. 1995. Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. Computer Speech and Language 9:2, 171-185.
- Given:
  - a trained AM
  - a small "adaptation" dataset from a new speaker
- Learn new values for the Gaussian mean vectors
  - Not by just training on the new data (too small)
  - But by learning a linear transform which moves the means.

Maximum Likelihood Linear Regression (MLLR)

- Estimates a linear transform matrix (W) and bias vector (ω) to transform HMM model means:

$$\mu_{new} = W_r \mu_{old} + \omega_r$$

- Transform estimated to maximize the likelihood of the adaptation data

Slide from Bryan Pellom

MLLR

- New equation for output likelihood

$$b_j(o_t) = \frac{1}{(2\pi)^{n/2} |\Sigma_j|^{1/2}} \exp\left(-\frac{1}{2} (o_t - (W\mu_j + \omega)) |\Sigma_j|^{-1} (o_t - (W\mu_j + \omega))^T \right)$$

MLLR

- Q: Why is estimating a linear transform from adaptation data different than just training on the data?
- A: Even from a very small amount of data we can learn 1 single transform for all triphones! So small number of parameters.
- A2: If we have enough data, we could learn more transforms (but still less than the number of triphones). One per phone (~50) is often done.

MLLR: Learning

- Given
  - a small labeled adaptation set (a couple sentences)
  - a trained AM
- Do forward-backward alignment on adaptation set to compute state occupation probabilities $\gamma_j(t)$.
- W can now be computed by solving a system of simultaneous equations involving $\gamma_j(t)$

MLLR performance on baby task (RM)

(Leggetter and Woodland 1995)

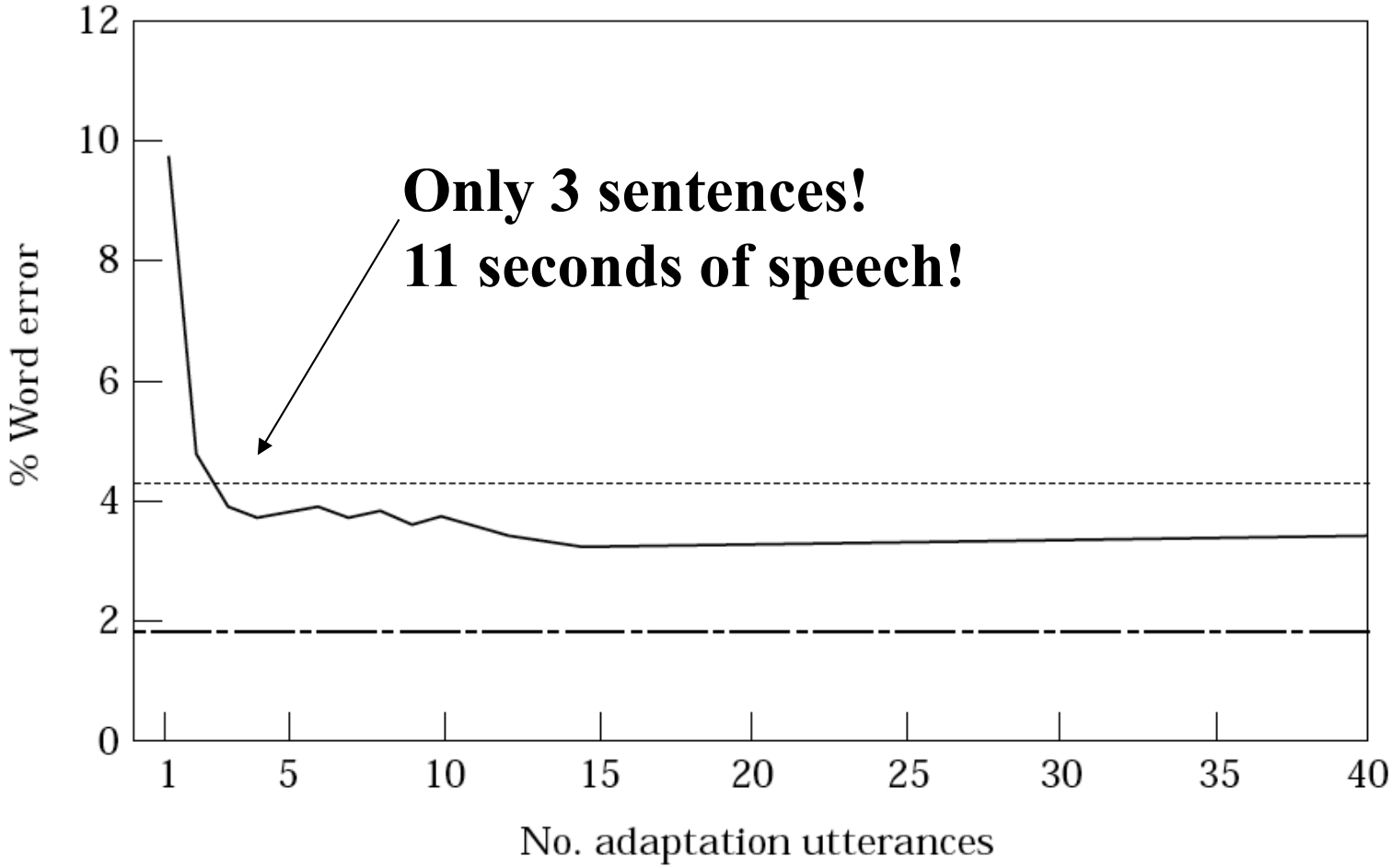


Figure 2. Full matrix maximum likelihood linear regression using global regression class. (·····), Speaker independent; (—-—-—-), speaker dependent; (——), speaker adapted.

MLLR doesn't need supervised adaptation set!

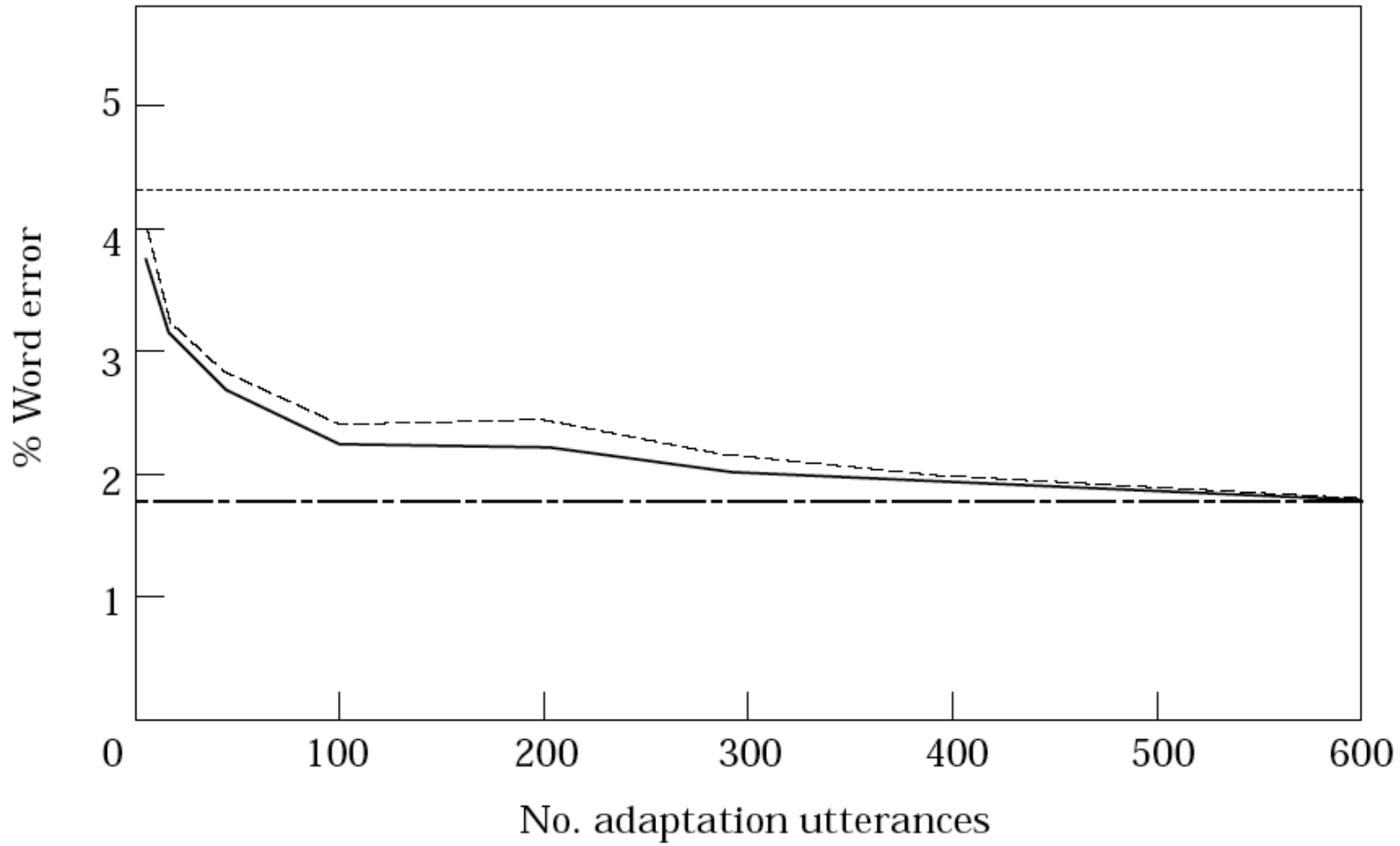


Figure 3. Supervised vs. unsupervised adaptation using maximum likelihood linear regression. (·····), Speaker independent; (—-—-—-), speaker dependent; (——), supervised adapted; (– – –), unsupervised adapted.

Maximum A Posteriori Adaptation (MAP)

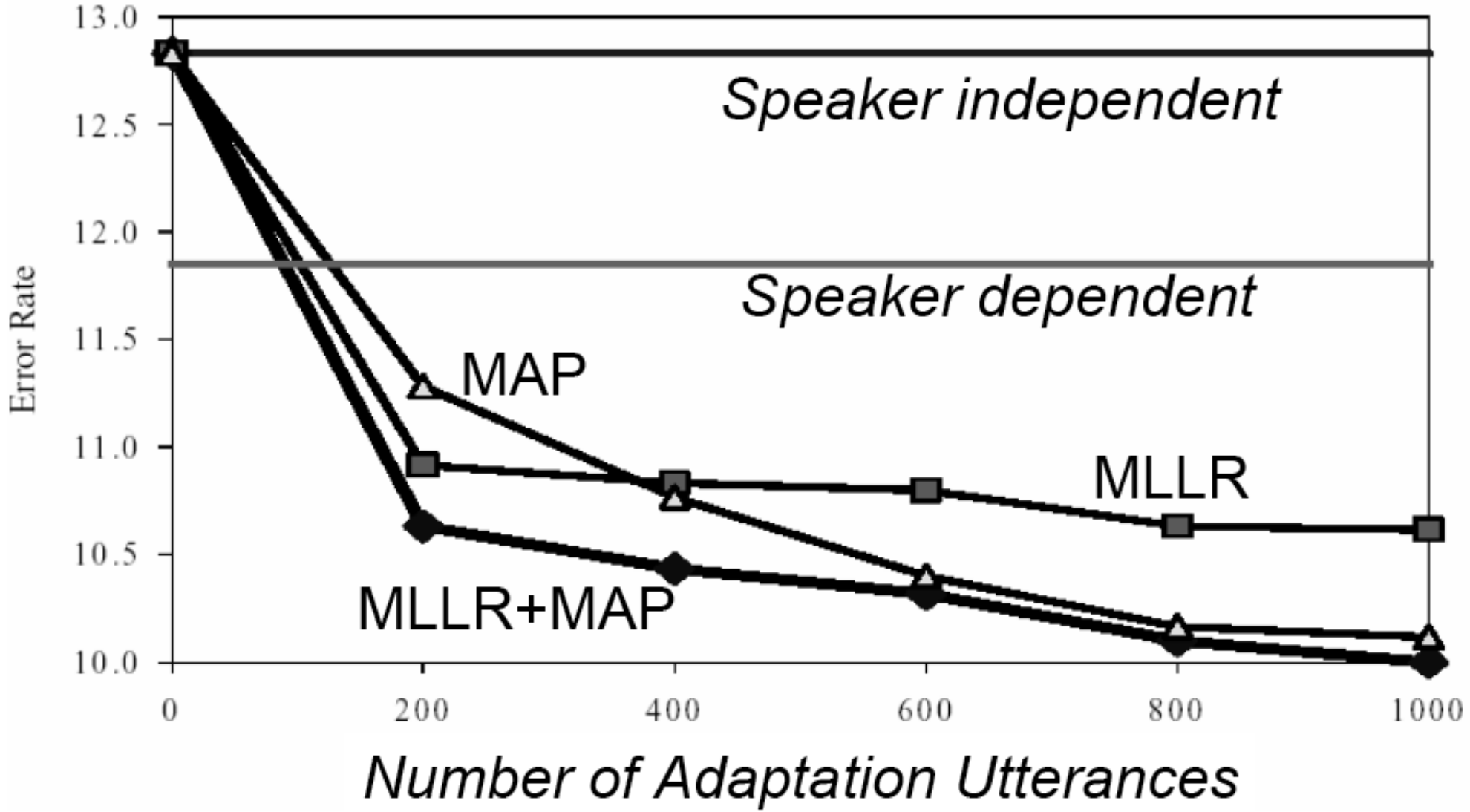
- **MAP Adaptation can only be applied Gaussians that are "seen" in the test data,**

$$\mu_{new} = \frac{\hat{N}}{\hat{N}+\alpha}\hat{m}_{obs} + \frac{\alpha}{\hat{N}+\alpha}\mu_{old}$$

$\hat{N}$	Number of frames of adaptation data
α	Weight for prior estimate of old mean
$\hat{m}_{obs}$	Mean vector of adaptation data assigned to Gauss.

Slide from Bryan Pellom

Performance of MLLR and MAP



Slide from Bryan Pellom after Huang et al

Summary

- MLLR: works on small amounts of adaptation data
- MAP: Maximum A Posterior Adaptation
  - Works well on large adaptation sets
- Acoustic adaptation techniques are quite successful at dealing with speaker variability
- If we can get 10 seconds with the speaker.