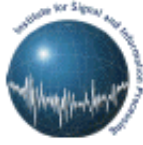


Course:

[Home](#)
[Syllabus](#)
[Lectures](#)

**Introduction:**

01: Organization
[\(html\)](#)

Speech Signals:

02: Production
[\(html\)](#)

03: Digital Models
[\(html\)](#)

04: Perception
[\(html\)](#)

05: Masking
[\(html\)](#)

06: Phonetics and Phonology
[\(html\)](#)

07: Syntax and Semantics
[\(html\)](#)

Signal Processing:

08: Sampling
[\(html\)](#)

09: Resampling
[\(html\)](#)

10: Acoustic Transducers
[\(html\)](#)

11: Temporal Analysis
[\(html\)](#)

12: Frequency Domain Analysis
[\(html\)](#)

13: Cepstral Analysis
[\(html\)](#)

14: **Exam No. 1**
[\(html\)](#) [pdf](#)

15: Linear Prediction
[\(html\)](#)

ECE 8463: FUNDAMENTALS OF SPEECH RECOGNITION

Professor Joseph Picone
 Department of Electrical and Computer Engineering
 Mississippi State University

email: picone@isip.msstate.edu
 phone/fax: 601-325-3149; office: 413 Simrall
 URL: http://www.isip.msstate.edu/resources/courses/ece_8463

Modern speech understanding systems merge interdisciplinary technologies from Signal Processing, Pattern Recognition, Natural Language, and Linguistics into a unified statistical framework. These systems, which have applications in a wide range of signal processing problems, represent a revolution in Digital Signal Processing (DSP). Once a field dominated by vector-oriented processors and linear algebra-based mathematics, the current generation of DSP-based systems rely on sophisticated statistical models implemented using a complex software paradigm. Such systems are now capable of understanding continuous speech input for vocabularies of hundreds of thousands of words in operational environments.

In this course, we will explore the core components of modern statistically-based speech recognition systems. We will view speech recognition problem in terms of three tasks: signal modeling, network searching, and language understanding. We will conclude our discussion with an overview of state-of-the-art systems, and a review of available resources to support further research and technology development.

Tar files containing a compilation of all the notes are available. However, these files are large and will require a substantial amount of time to download. A tar file of the html version of the notes is available [here](#). These were generated using wget:

```
wget -np -k -m
http://www.isip.msstate.edu/publications/courses/ece_8463/lectures/current
```

A pdf file containing the entire set of lecture notes is available [here](#). These were generated using Adobe Acrobat.

Questions or comments about the material presented here can be directed to help@isip.msstate.edu.

16: LP-Based Representations

[\(html\)](#)

17: Spectral Normalization

[\(html\)](#)

Parameterization:

18: Differentiation

[\(html\)](#)

19: Principal Components

[\(html\)](#)

20: Linear Discriminant Analysis

[\(html\)](#)

Acoustic Modeling:

21: Dynamic Programming

[\(html\)](#)

22: Markov Models

[\(html\)](#)

23: Parameter Estimation

[\(html\)](#)

24: HMM Training

[\(html\)](#)

25: Continuous Mixtures

[\(html\)](#)

26: Practical Issues

[\(html\)](#)

27: Decision Trees

[\(html\)](#)

28: Limitations of HMMs

[\(html\)](#)

Language Modeling:

29: Formal Language Theory

[\(html\)](#)

30: Context-Free Grammars

[\(html\)](#)

31: **Exam No. 2**

[\(html\)](#) [pdf](#)

32: N-Gram Models and Complexity

[\(html\)](#)

33: Smoothing

[\(html\)](#)

Search:

34: Basic Search Algorithms

[\(html\)](#)

35: Time Synchronous Search

[\(html\)](#)

36: Stack Decoding

[\(html\)](#)

37: Lexical Trees

[\(html\)](#)

38: Efficient Trees

[\(html\)](#)

Miscellaneous Topics:

39: Adaption

[\(html\)](#)

40: **Exam No. 3**

[\(html\)](#), [pdf](#)

41: Discriminative Training

[\(html\)](#)

42: Neural Networks

[\(html\)](#)

Summary:

43: Scoring and Evaluation

[\(html\)](#)

44: Common Evaluation Tasks

[\(html\)](#)

45: State Of The Art

[\(html\)](#)

46: **Final Exam**

[\(html\)](#), [pdf](#)

[Return to Main](#)**Introduction:**[Overview](#)[Syllabus](#)[Technology](#)**Software Resources:**[Internet Speech](#)[Software](#)[Toolkits](#)**Educational Resources:**[SRSDR'02](#)[Training](#)[Short Course](#)[Misc. Notes](#)

LECTURE 01: COURSE OVERVIEW AND OBJECTIVES

- Objectives:

- Learn about basic technology
- Understand theory at a fundamental level
- Relate to other theory such pattern recognition, signal processing, computational linguistics, etc.
- Develop perspective: Are the approaches we use specific to a speech signal?

- What we won't do:

- Computer programming
 - Computer simulations
 - Matlab exercises
 - Teach you how to tune parameters
 - Train you to be speech technologists...
-
- Why?

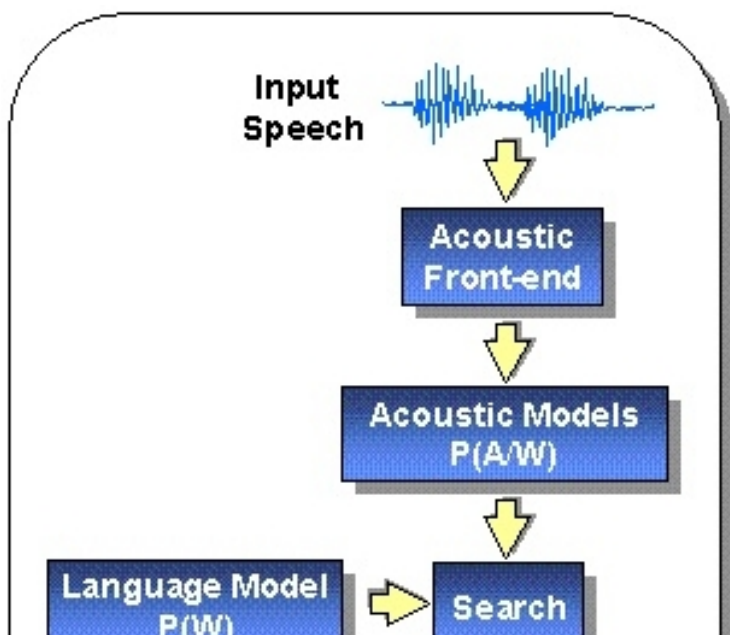
LECTURE 01: COURSE OVERVIEW AND OBJECTIVES

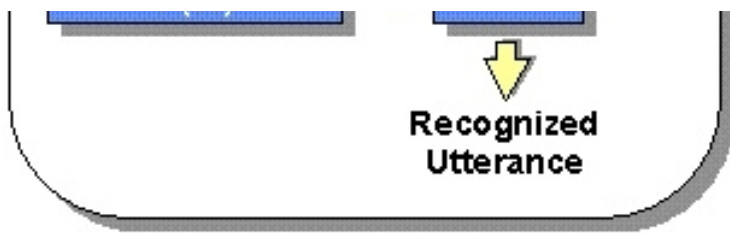
- Objectives:
 - Learn about basic technology
 - Understand theory at a fundamental level
 - Relate to other theory such pattern recognition, signal processing, computational linguistics, etc.
 - Develop perspective: Are the approaches we use specific to a speech signal?
- What we won't do:
 - Computer programming
 - Computer simulations

- Matlab exercises
 - Teach you how to tune parameters
 - Train you to be speech technologists...
-
- Why?

HUMAN LANGUAGE TECHNOLOGY: SPEECH RECOGNITION IS MULTIDISCIPLINARY

- Acoustic Front-End: Signal Processing
- Acoustic Models: Pattern Recognition, Linguistics





- Language Model: Natural Language Processing
- Search: Computational Linguistics, Cognitive Science

[Return to Main](#)

[Objectives](#)

Physiology:

[Saggital View](#)

[Saggital X-ray](#)

[Vocal Cords](#)

[Transduction](#)

[Spectrogram](#)

Acoustics:

[Acoustic Theory](#)

[Wave Propagation](#)

[Helium Speech](#)

On-Line Resources:

[Spectrograms](#)

[Acoustics and Speech](#)

[Sound Waves in Tubes](#)

[Tube Models](#)

[Helium Speech](#)

LECTURE 02: A BRIEF OVERVIEW OF SPEECH PRODUCTION

- Objectives:
 - Basic speech physiology
 - Speech is a sound pressure wave
 - Transduction to an electrical signal introduces distortion
 - Acoustic analysis follows the same principles used in electromagnetic wave propagation

- There are many ways to view a speech signal
- Concatenated tube models (linear acoustics)

This lecture contains material from an excellent textbook on the fundamentals of speech processing:

J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co., ISBN: 0-7803-5386-2, 2000.

as well as information found in the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall,

Upper Saddle River, New Jersey,
USA, ISBN: 0-13-022616-5,
2001.

LECTURE 02: A BRIEF OVERVIEW OF SPEECH PRODUCTION

- Objectives:
 - Basic speech physiology
 - Speech is a sound pressure wave
 - Transduction to an electrical signal introduces distortion
 - Acoustic analysis follows the same principles used in electromagnetic wave propagation
 - There are many ways to view a speech signal

- Concatenated tube models (linear acoustics)

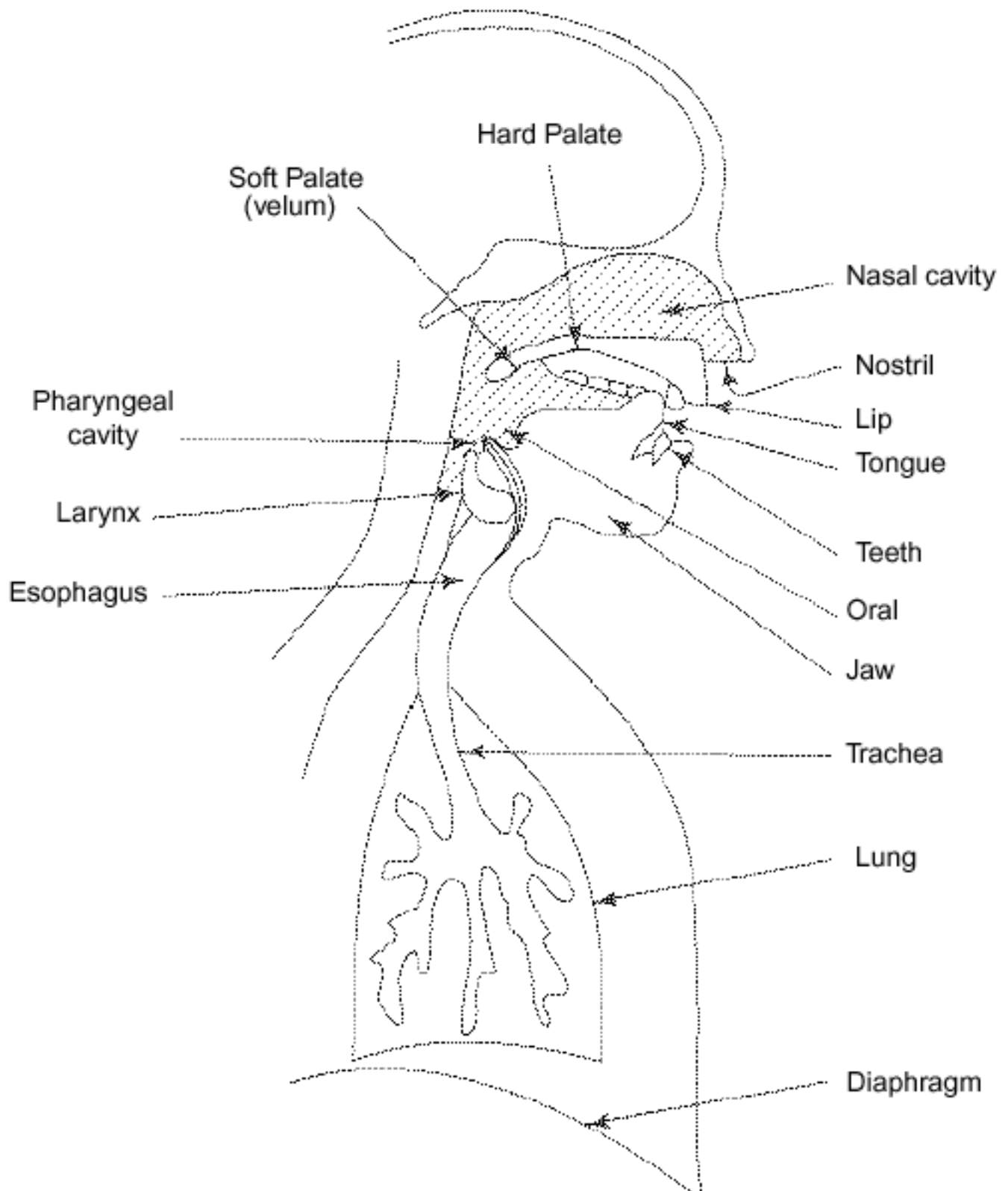
This lecture contains material from an excellent textbook on the fundamentals of speech processing:

J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co., ISBN: 0-7803-5386-2, 2000.

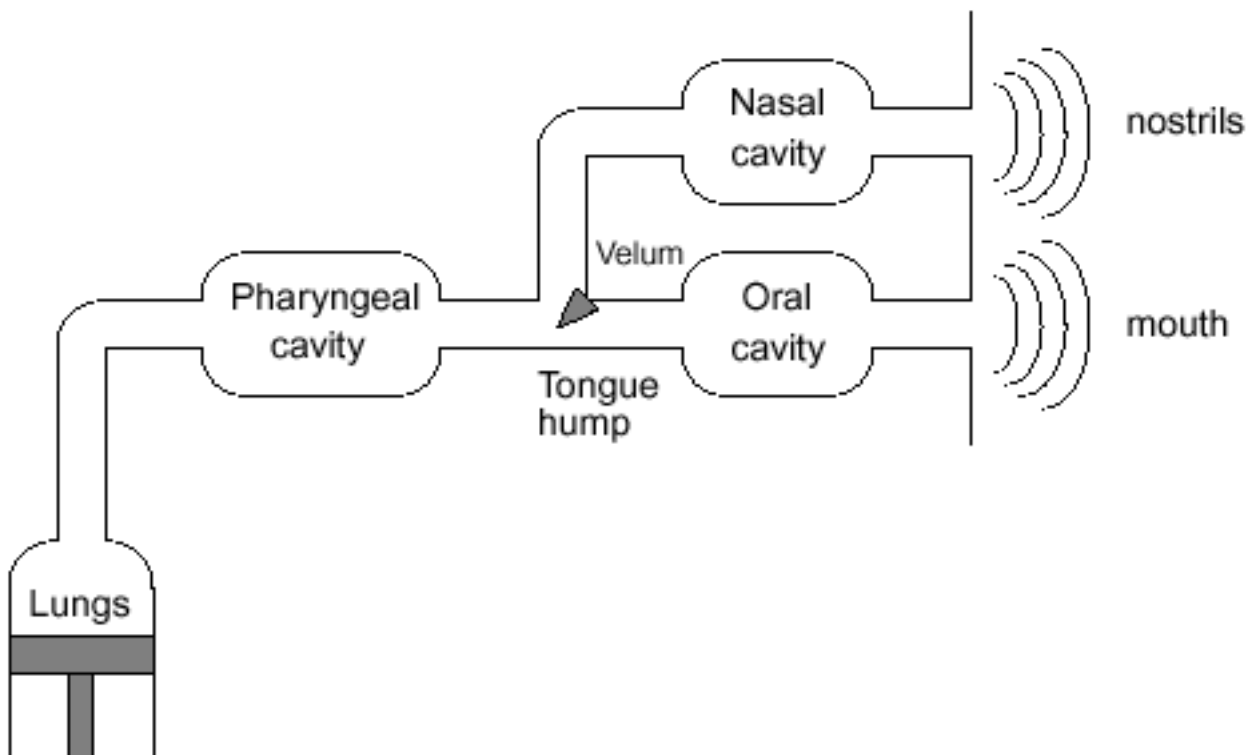
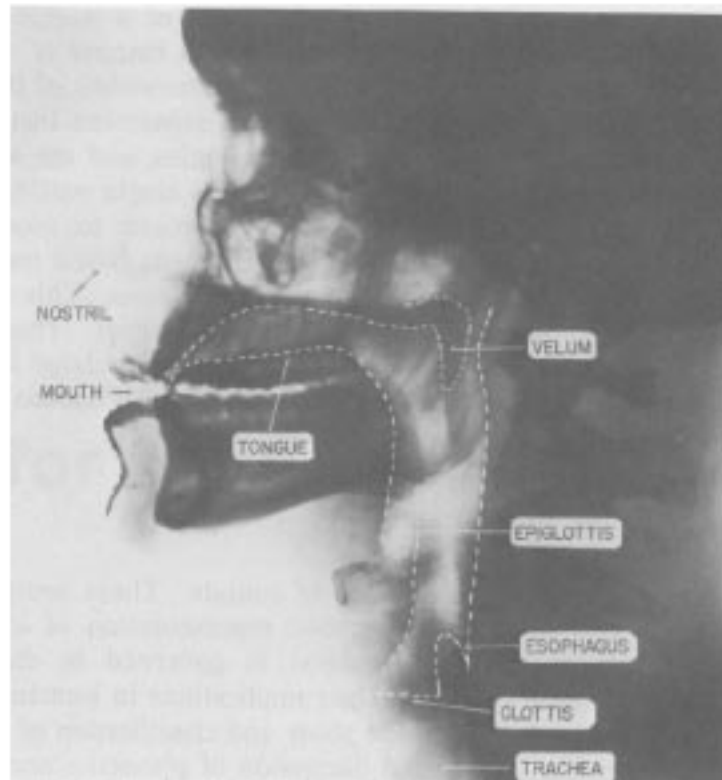
as well as information found in the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

SAGGITAL PLANE VIEW OF THE HUMAN VOCAL APPARATUS

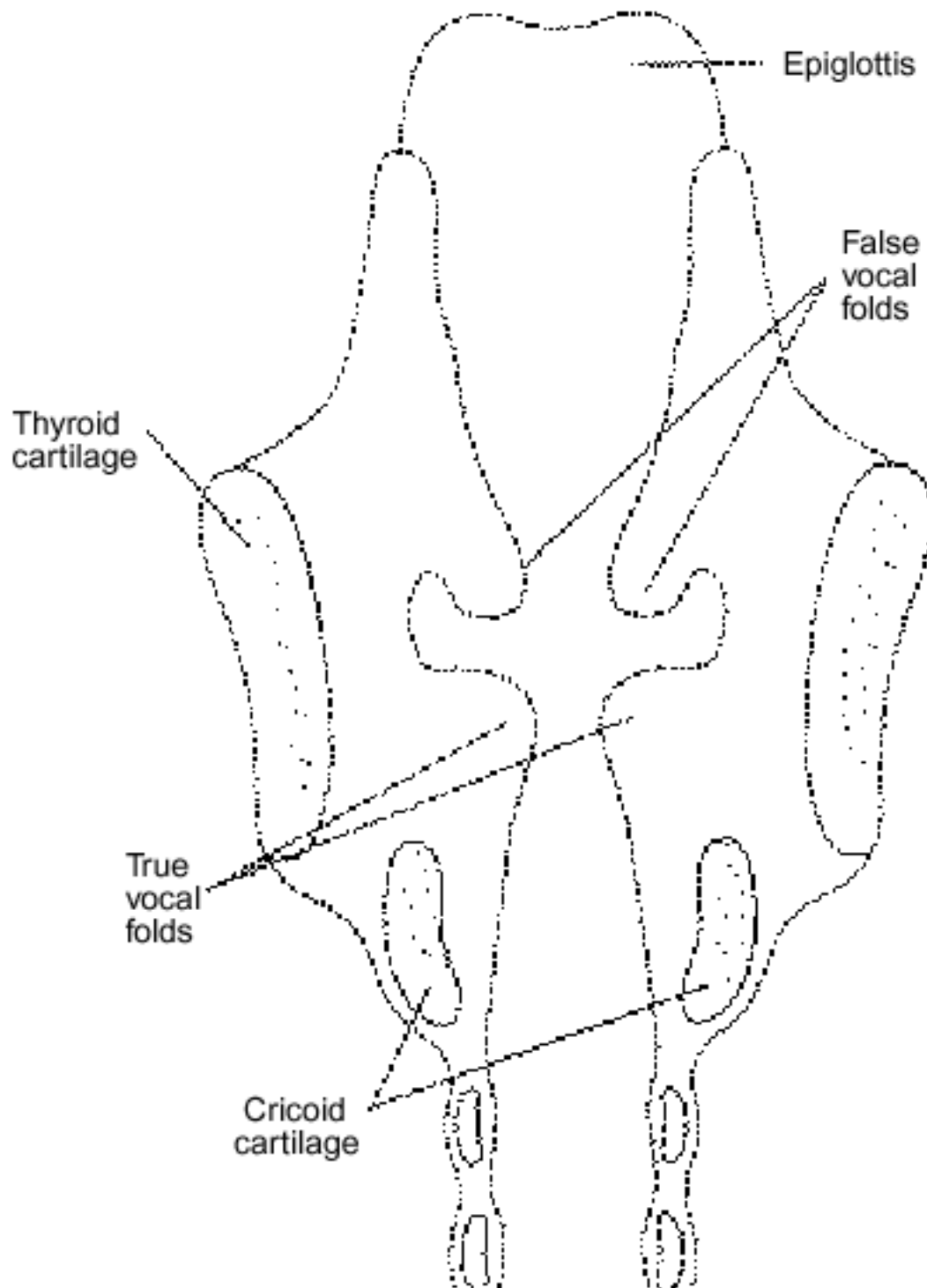


SAGGITAL X-RAY OF THE HUMAN VOCAL APPARATUS



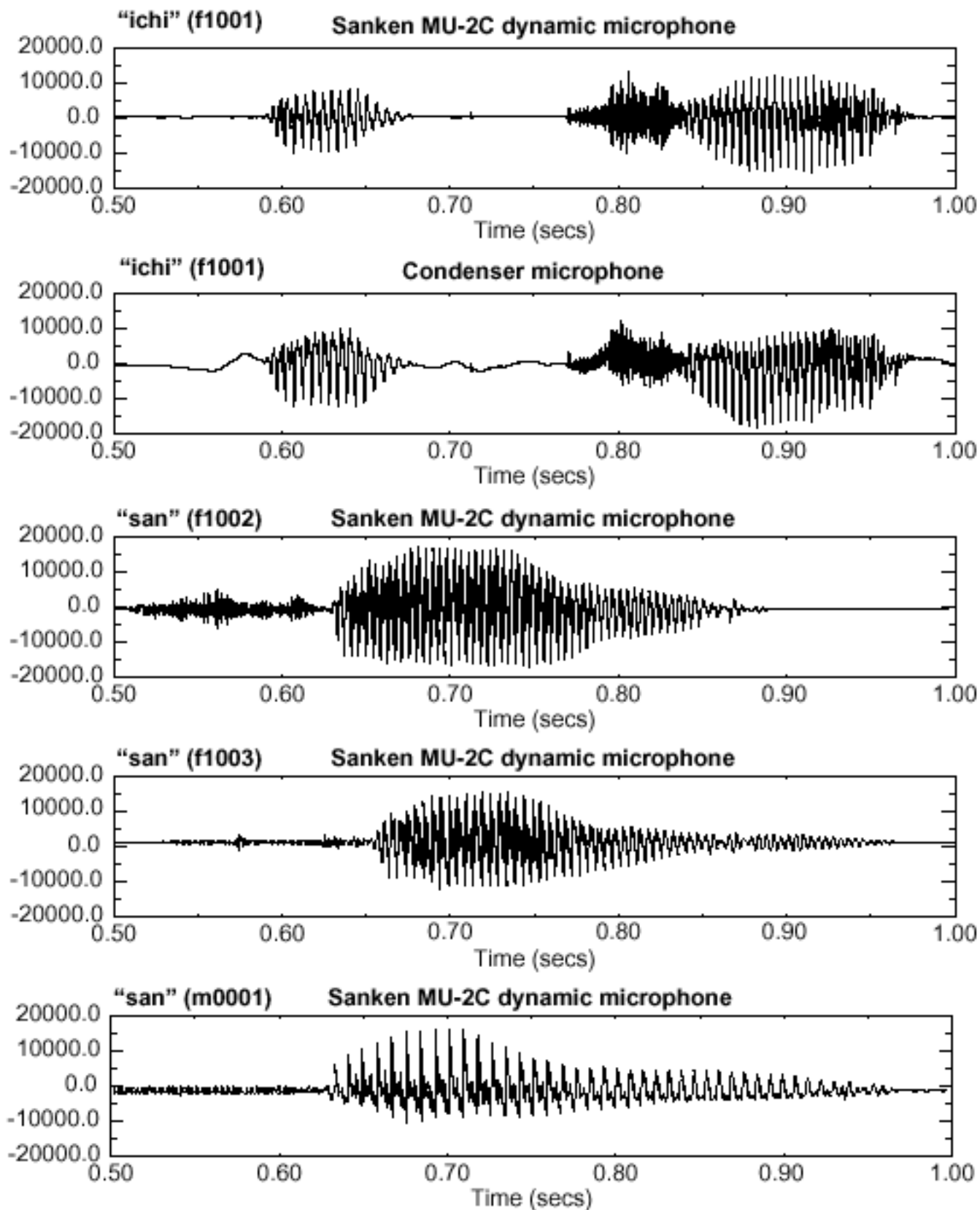


VOCAL CORDS - SOURCE OF EXCITATION



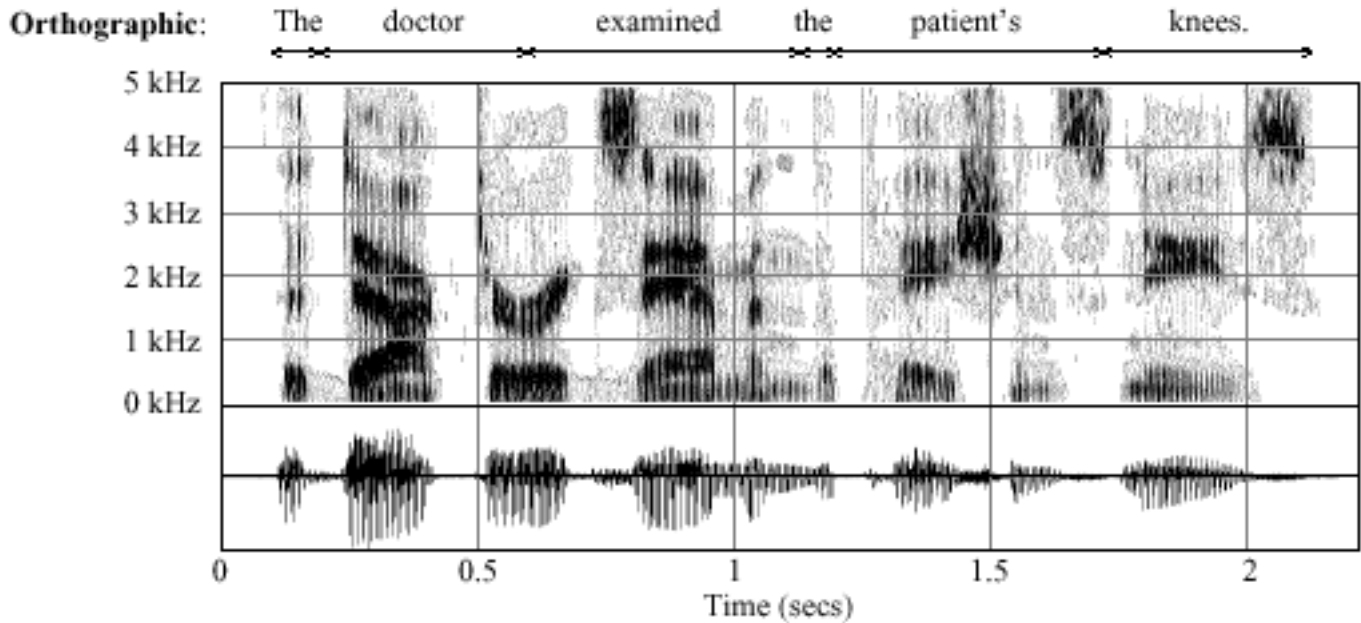
TRANSDUCTION

- Speech is a sound pressure wave that must be converted to an electrical signal, and then a digital signal, to be processed. This conversion process introduces distortion (frequency response, nonlinear dynamics, etc.).

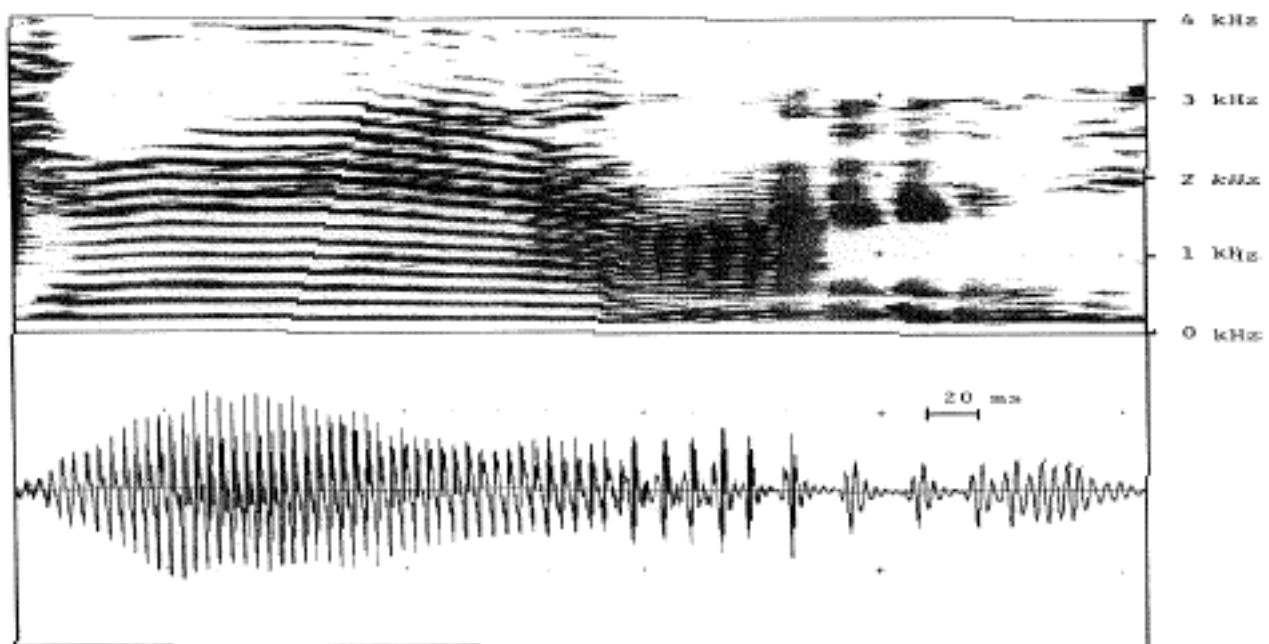


WHAT DOES A SPEECH SIGNAL LOOK LIKE?

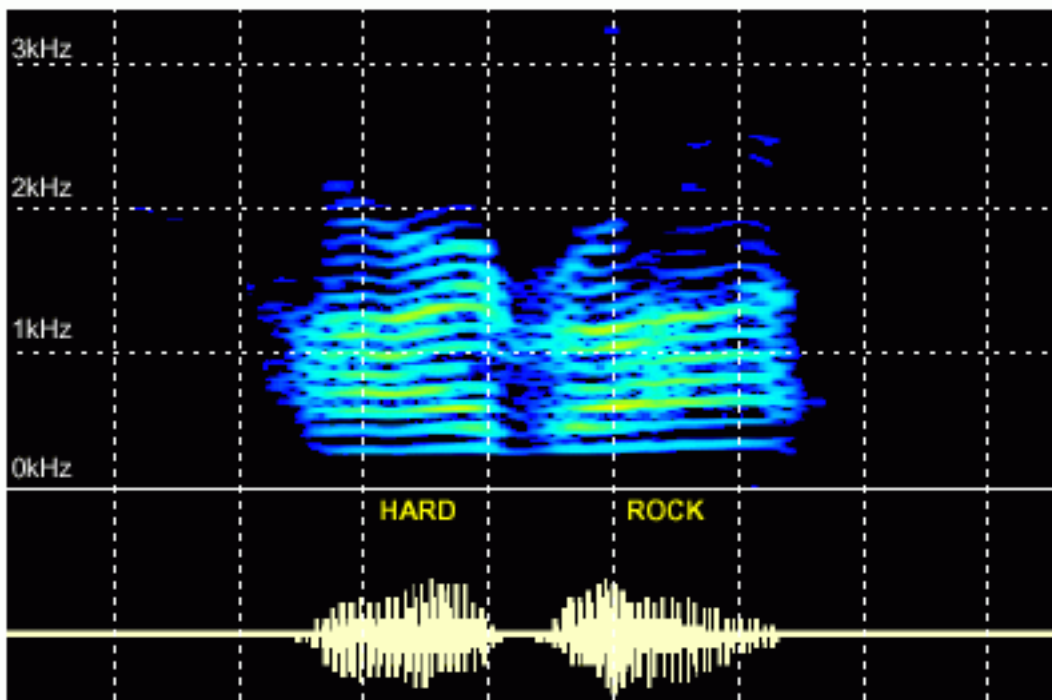
Standard wideband spectrogram ($f_s = 10 \text{ kHz}$, $T_w = 6 \text{ ms}$):



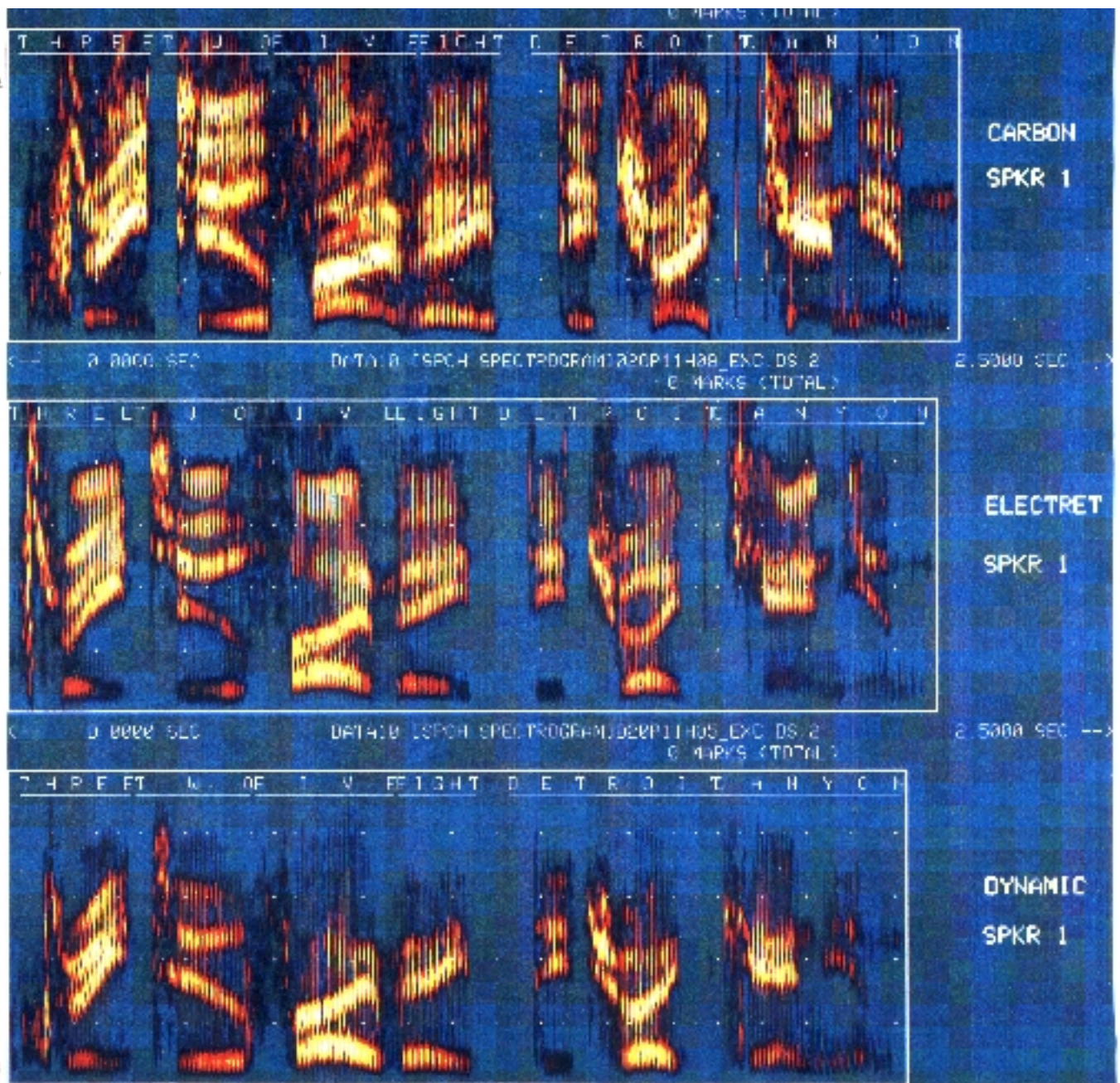
Narrowband Spectrogram ($f_s = 8 \text{ kHz}$, $T_w = 30 \text{ ms}$):

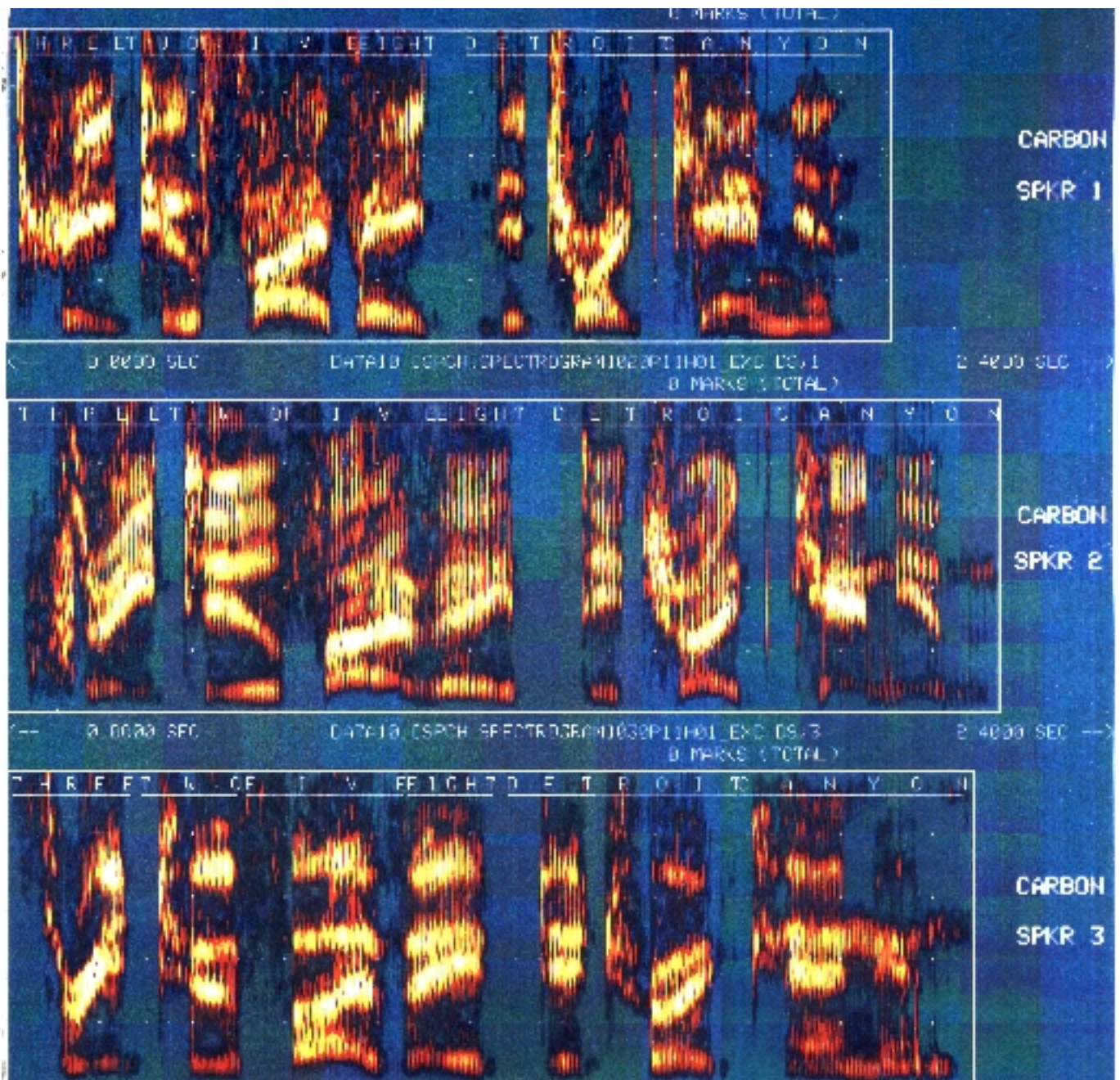


- We often prefer to view a spectrogram using a color visualization in which spectral log magnitude is mapped to "temperature" (the color that emanates from a steel bar when it is heated):



- Here are more examples of color spectrograms using the ever-popular Texas Instruments color map:



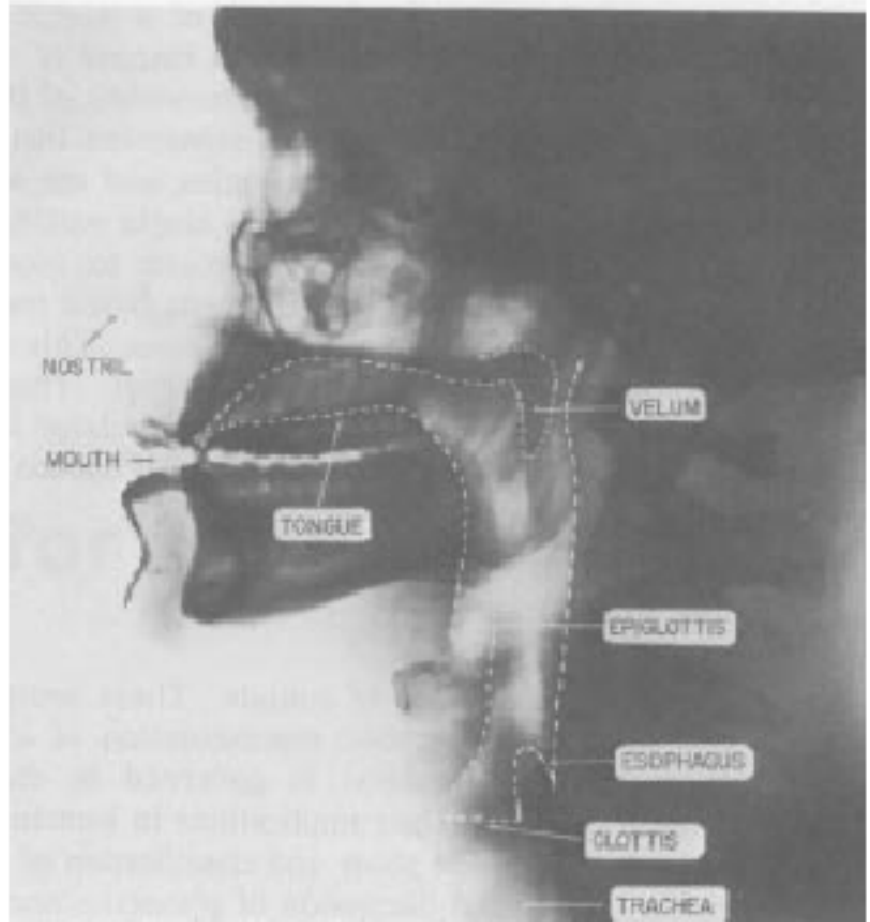


SOUND PROPAGATION - LINEAR ACOUSTICS

A detailed acoustic theory must consider the effects of the following:

- Time variation of the vocal tract shape

- Losses due to heat conduction and viscous friction at the vocal tract walls

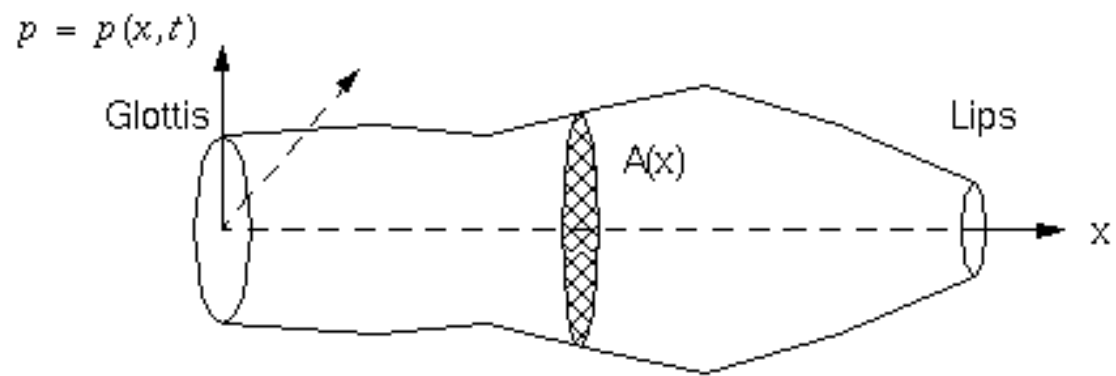


- Softness of the vocal tract walls

- Radiation

of sound at the lips

- Nasal coupling
- Excitation of sound in the vocal tract
- Let us consider a simple case of a lossless tube:



WAVE PROPAGATION

For frequencies that are long compared to the dimensions of the vocal tract (less than about 4000 Hz, which implies a wavelength of 8.5 cm), sound waves satisfy the following pair of equations:

$$\begin{aligned} \rho \frac{\partial(u/A)}{\partial t} + \text{grad } p &= 0 & -\frac{\partial p}{\partial x} &= \rho \frac{\partial(u/A)}{\partial t} \\ \frac{1}{\rho c^2} \frac{\partial p}{\partial t} + \frac{\partial A}{\partial t} + \text{div } u &= 0 & \text{or} & -\frac{\partial u}{\partial x} = \frac{1}{\rho c^2} \frac{\partial(pA)}{\partial t} + \frac{\partial A}{\partial t} \end{aligned}$$

where

$p = p(x, t)$ is the variation of the sound pressure in the tube

$u = u(x, t)$ is the variation in the volume velocity

ρ is the density of air in the tube (1.2 mg/cc)

c is the velocity of sound (35000 cm/s)

$A = A(x, t)$ is the area function (about 17.5 cm long)

HELIUM SPEECH:

RELATIONSHIP BETWEEN FREQUENCY AND DENSITY

- Deep-sea diving to depths exceeding about 140 feet of sea water requires the use of heliox (a mixture of helium and oxygen) as a breathing gas, rather than compressed air.
- Heliox eliminates the danger of nitrogen narcosis and reduces the risk of decompression sickness which would otherwise be present.
- Heliox presents another risk. The diver's speech is rendered unintelligible because the higher velocity of sound in the diver's vocal tract shifts the frequency components of the diver's speech to much higher frequencies - an effect that has been likened to the "Donald Duck" voice.
- Heliox is less dense than air or pure oxygen. Hence, the speed of sound is greater, so the resonances occur at higher frequencies.
- The excitation remains largely unchanged since flesh in your vocal folds still vibrates at the same frequency, so the harmonics occur at the same frequency. (There could be a small change because the less dense Helium loads the vocal folds a bit less than the air, but this effect is slight.)

- Examples of helium speech are always fun to listen to.
- Descramblers are available that will perform real-time spectral shifting.
- Such systems use real-time spectral shifting.

The information on this page comes from two sources:

K. Bryden and J. Hothi
Communications Research Centre
3701 Carling Avenue
P.O. Box 11490, Stn. H
Ottawa, ON K2H 8S2
Tel: (613) 998-2515
Fax: (613) 990-7987
Email: karen.bryden@crc.ca
URL: http://www.crc.ca/en/html/crc/tech_transfer/10085

and,

J. Wolfe
School of Physics
The University of New South Wales
SYDNEY 2052
Australia
Tel: 61 2 9385 4954
Fax: 61 2 9385 6060

Email: J.Wolfe@unsw.edu.au

URL:

<http://www.phys.unsw.edu.au/STAFF/ACADEMIC/wolfe.html>

Work on real-time frequency scaling can be found in several journals including the *IEEE Transactions on Speech and Audio Processing* (formerly *Acoustics, Speech, and Signal Processing*), and the *Journal of the Acoustical Society of America*.

[Return to Main](#)[Objectives](#)**Acoustic Models:**[Acoustic Theory](#)[Lossless Tubes](#)[Resonances](#)[Losses](#)[Lip Radiation](#)[Nasal Cavity](#)**Lossless Tubes:**[Concatenated Tubes](#)[Excitation Models](#)[Two Tube Models](#)[Three Tube Models](#)[Transfer Functions](#)**Digital Models:**[Digital Equivalents](#)[Digital Transfer Functions](#)[Excitation Models](#)[Vocoder Model](#)**On-Line Resources:**[Sound Waves in Tubes](#)[Tube Models](#)[Cool Edit](#)[CD-ROM Tutorial](#)

LECTURE 03: SOUND PROPAGATION

- Objectives:
 - Basic properties of lossless tubes
 - Resonant structure of the vocal tract
 - Articulator positions (basic speech sounds) translate to predictable spectral signatures
 - Digital filter-based models of the vocal tract (linear acoustics)

- Relationship of the parameters of these digital models to speech recognition.

Note that this lecture is based on material in this textbook:

J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co., ISBN: 0-7803-5386-2, 2000.

LECTURE 03: SOUND PROPAGATION

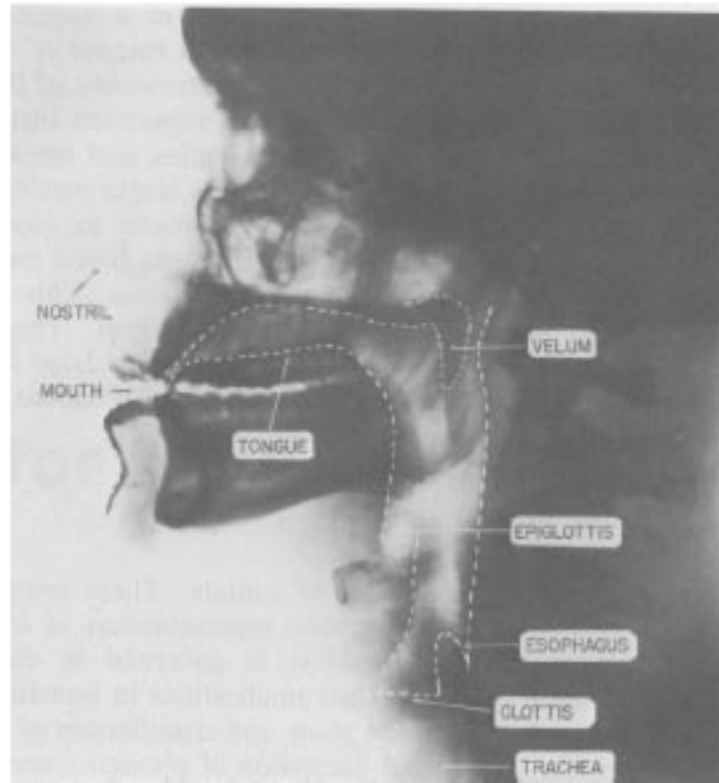
- Objectives:
 - Basic properties of lossless tubes
 - Resonant structure of the vocal tract
 - Articulator positions (basic speech sounds) translate to predictable spectral signatures
 - Digital filter-based models of the vocal tract (linear acoustics)
 - Relationship of the parameters of these digital models to speech recognition.

Note that this lecture is based on material in this

textbook:

J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co., ISBN: 0-7803-5386-2, 2000.

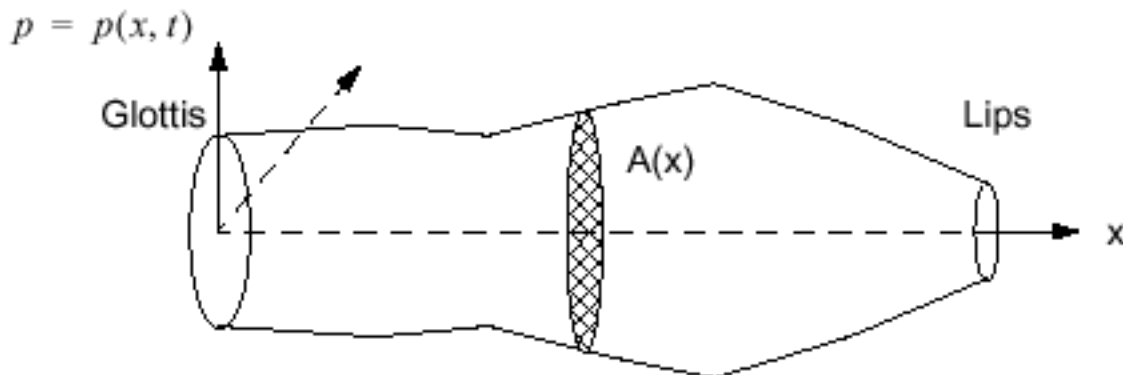
SOUND PROPAGATION



A detailed acoustic theory must consider the effects of the following:

- Time variation of the vocal tract shape
- Losses due to heat conduction and viscous friction at the vocal tract walls
- Softness of the vocal tract walls
- Radiation of sound at the lips
- Nasal coupling
- Excitation of sound in the vocal tract

Let us begin by considering a simple case of a lossless tube:



UNIFORM LOSSLESS TUBE

For frequencies that are long compared to the dimensions of the vocal tract (less than about 4000 Hz, which implies a wavelength of 8.5 cm), sound waves satisfy the following pair of equations:

$$\begin{aligned} \rho \frac{\partial(u/A)}{\partial t} + \text{grad}^m p &= 0 & -\frac{\partial p}{\partial x} &= \rho \frac{\partial(u/A)}{\partial t} \\ \frac{1}{\rho c^2} \frac{\partial p}{\partial t} + \frac{\partial A}{\partial t} + \text{div } u &= 0 & \text{or} & -\frac{\partial u}{\partial x} = \frac{1}{\rho c^2} \frac{\partial(pA)}{\partial t} + \frac{\partial A}{\partial t} \end{aligned}$$

where

$p = p(x, t)$ is the variation of the sound pressure in the tube

$u = u(x, t)$ is the variation in the volume velocity

ρ is the density of air in the tube (1.2 mg/cc)

c is the velocity of sound (35000 cm/s)

$A = A(x, t)$ is the area function (about 17.5 cm long)

Uniform Lossless Tube

If $A(x, t) = A$, then the above equations reduce to:

$$-\frac{\partial p}{\partial x} = \frac{\rho}{A} \frac{\partial u}{\partial t} \quad -\frac{\partial u}{\partial x} = \frac{A}{\rho c^2} \frac{\partial p}{\partial t}$$

The solution is a traveling wave:

$$\begin{aligned} u(x, t) &= u^+(t - x/c) - u^-(t + x/c) \\ p(x, t) &= \frac{\rho c}{A} [u^+(t - x/c) + u^-(t + x/c)] \end{aligned}$$

which is analogous to a transmission line:

$$-\frac{\partial v}{\partial x} = L \frac{\partial i}{\partial t} \quad -\frac{\partial i}{\partial x} = C \frac{\partial v}{\partial t}$$

What are the salient features of the lossless transmission line model?

RESONANT FREQUENCIES OF A LOSSLESS TUBE

where

| <i>Acoustic Quantity</i> | <i>Analogous Electric Quantity</i> |
|---------------------------------------|------------------------------------|
| p - pressure | v - voltage |
| u - volume velocity | i - current |
| ρ/A - acoustic inductance | L - inductance |
| $A/(\rho c^2)$ - acoustic capacitance | C - capacitance |

The sinusoidal steady state solutions are:

$$p(x, t) = jZ_o \frac{\sin[\Omega(l-x)/c]}{\cos[\Omega l/c]} U_G(\Omega) e^{j\Omega t}$$

$$u(x, t) = \frac{\cos[\Omega(l-x)/c]}{\cos[\Omega l/c]} U_G(\Omega) e^{j\Omega t}$$

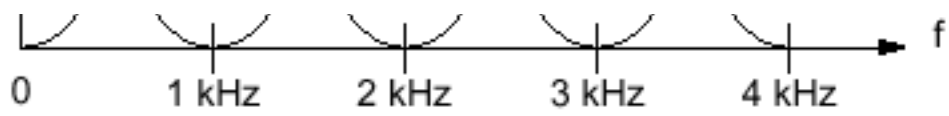
where $Z_o = \frac{\rho c}{A}$ is the characteristic impedance.

The transfer function is given by:

$$\frac{U(l, \Omega)}{U(0, \Omega)} = \frac{1}{\cos(\Omega l/c)}$$

This function has poles located at every $\frac{(2n+1)\pi c}{2l}$. Note that these correspond to the frequencies at which the tube becomes a quarter wavelength: $\left(\frac{\Omega l}{c} = \frac{\pi}{2}\right) \Rightarrow \left(f = \frac{c}{4l}\right)$.

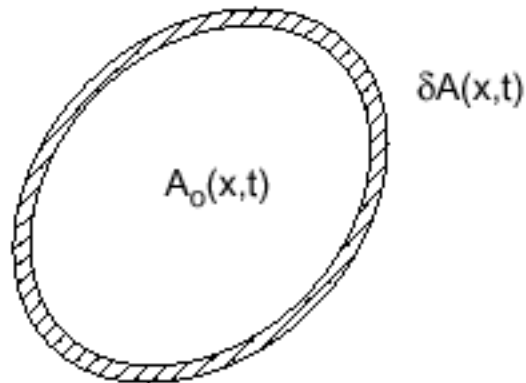




Is this model realistic?

EFFECTS OF LOSSES

What do we predict the effects of yielding walls to be?



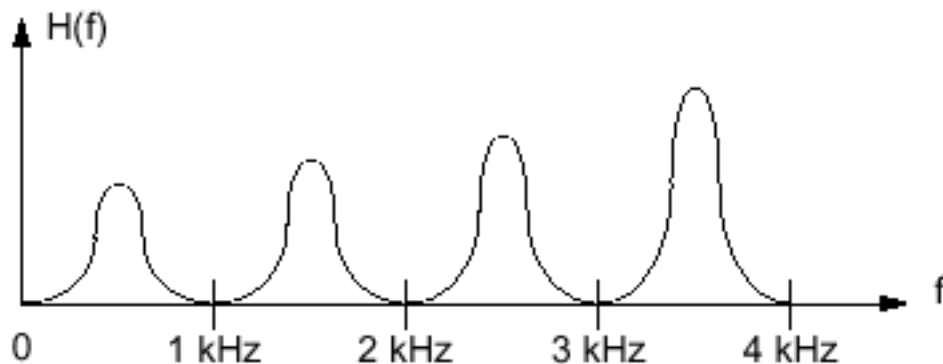
Use perturbation analysis:

$$A(x, t) = A_o(x, t) + \delta A(x, t)$$

We can develop a model that relates $\delta A(x,t)$ to pressure:

$$\frac{m_w d^2(\delta A)}{dt^2} + b_w \frac{d(\delta A)}{dt} + k_w(\delta A) = p(x, t)$$

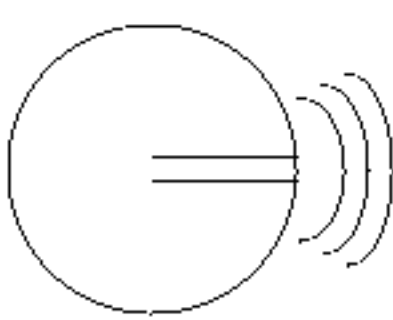
and solve for the new transfer function. But we can easily predict the effect of this:



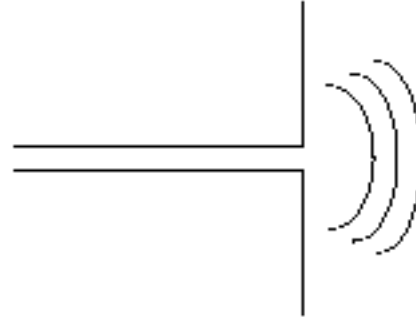
What would you expect to be the effect of friction and thermal losses?

LIP RADIATION

How is the sound pressure wave within the vocal tract coupled into the air?



Radiation from a spherical baffle



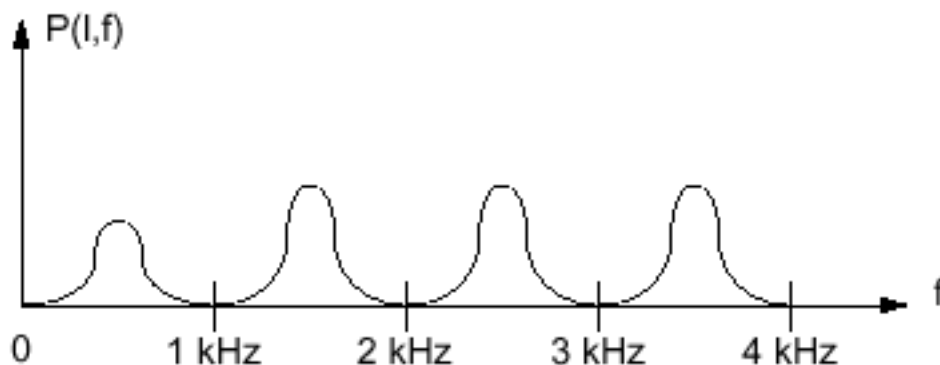
Radiation from an infinite plane baffle

Net effect is to place a complex load on the system:

$$Z_L(\Omega) = \frac{j\Omega L_r R_r}{R_r + j\Omega L_r} \quad \text{and} \quad P(l, \Omega) = Z_L(\Omega) U(l, \Omega)$$

where $R_r = 128/9\pi^2$ and $L_r = 8a/3\pi c$, and a is the radius of the opening.

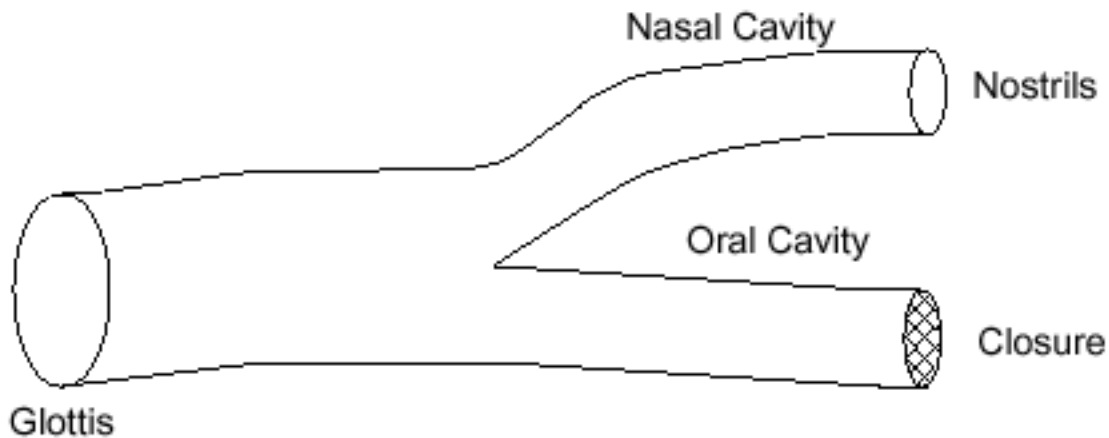
This impedance acts as a short circuit at low frequencies, and an imaginary impedance at high frequencies. The next effect on the volume velocity is to act as a highpass filter and to attenuate low frequencies. Lip radiation introduces a zero in the spectrum at DC and broadens the bandwidths at higher frequencies.



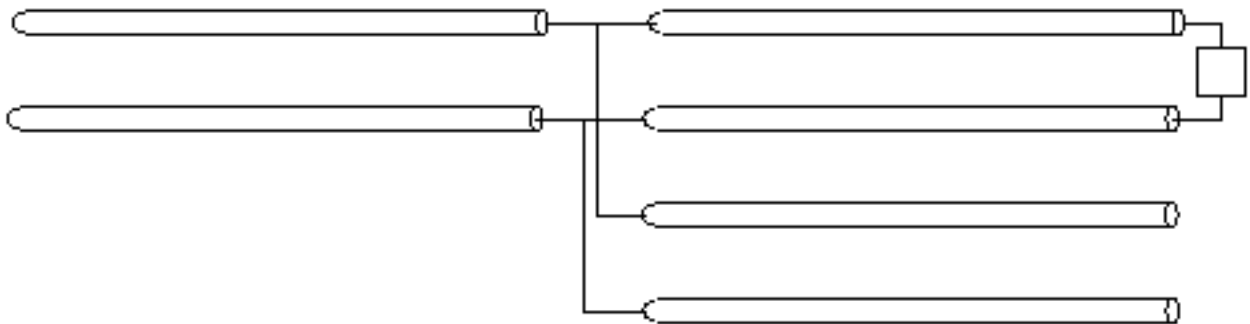
NASAL COUPLING

How is the sound pressure wave within the vocal tract coupled into the air?

We also must worry about the nasal cavity, especially for labial sounds for which the mouth is closed during sound production.



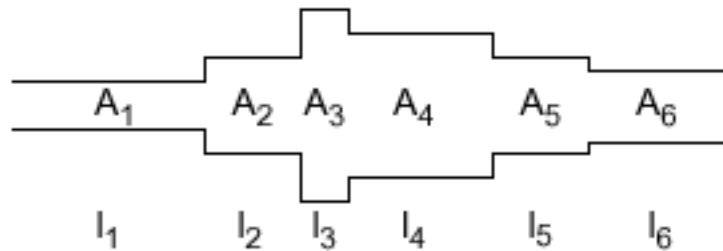
This is the equivalent of placing a transmission line in parallel with the vocal tract (oral cavity). What will the effect be?



The net effect is to produce a zero in the spectrum at about 1 kHz. As a result, nasal sounds (such as "m" and "n" in American English) have very little high frequency energy.

PIECEWISE LINEAR APPROXIMATIONS FOR THE VOCAL TRACT

Consider the following approximation to the vocal tract area function:



Recall,

$$p_k(x, t) = \frac{\rho c}{A_k} [u_k^+(t - x/c) + u_k^-(t + x/c)]$$

$$u(x, t) = u_k^+(t - x/c) - u_k^-(t + x/c)$$

For the k^{th} section, if we apply the boundary conditions:

$$p_k(l_k, t) = p_{k+1}(0, t)$$

$$u_k(l_k, t) = u_{k+1}(0, t)$$

We can combine these two equations to show:

$$u_{k+1}^+(t) = \left[\frac{2A_{k+1}}{A_{k+1} + A_k} \right] u_k^+(t - \tau_k) + \left[\frac{A_{k+1} - A_k}{A_{k+1} + A_k} \right] u_{k+1}^-(t)$$

where $\tau_k = l_k/c$.

We can define a reflection coefficient for the k^{th} junction:

$$r_k = \frac{u_{k+1}^-(t)}{u_{k+1}^+(t)} = \frac{A_{k+1} - A_k}{A_{k+1} + A_k}$$

It is easy to show that the reflection coefficients are bounded: $-1 \leq r_k \leq 1$.

The velocity can be expressed in terms of the reflection coefficients:

$$u_{k+1}^+(t) = (1 + r_k) u_k^+(t - \tau_k) + r_k u_{k+1}^-(t)$$

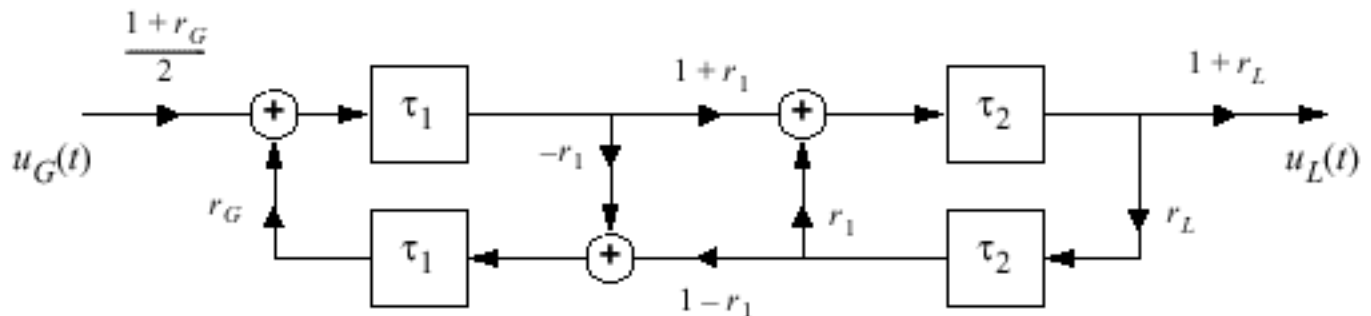
$$u_{k+1}^-(t) = (1 - r_k) u_k^-(t - \tau_k) + (1 + r_k) u_{k+1}^+(t)$$

$$u_k(t + \tau_k) = (1 - r_k)u_k(t - \tau_k) + (1 + r_k)u_{k+1}(t)$$

Ultimately, we will relate $\{r_k\}$ to a discrete model of the velocity profile.

ACOUSTIC EXCITATION MODELS

Consider a two tube approximation to the vocal tract:

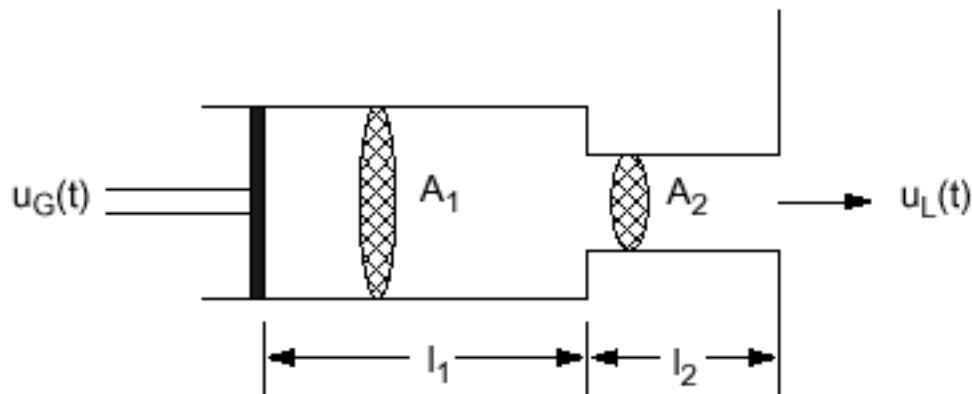


The frequency response of this system is:

$$V_a(\Omega) = \frac{U_L(\Omega)}{U_G(\Omega)} = \frac{0.5(1+r_G)(1+r_L)e^{-j\Omega(\tau_1+\tau_2)}}{1+r_1r_Ge^{-j\Omega 2\tau_1}+r_1r_Le^{-j\Omega 2\tau_2}+r_Lr_Ge^{-j\Omega 2(\tau_1+\tau_2)}}$$

What does this tell us about the frequency response?

If we consider the case $r_G = r_L = 1$:



For this system, the poles are located at values that satisfy the equation:

$$\frac{A_1}{A_2} \tan(\Omega\tau_2) = \cot(\Omega\tau_1)$$

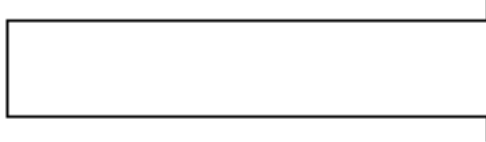
How does this compare to a single lossless tube?

Poles must be found through numerical analysis - nonlinear equation.

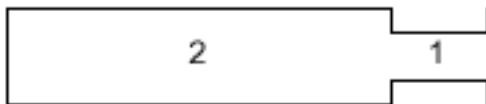
TWO TUBE MODELS

Resonator Geometry

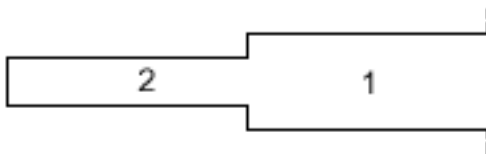
$L = 17.6 \text{ cm}$



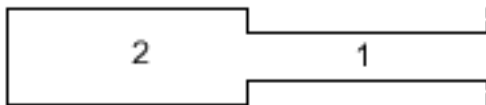
$$L_2/L_1 = 8 \quad A_2/A_1 = 8$$



$$L_2/L_1 = 1.2 \quad A_2/A_1 = 1/8$$



$$L_2/L_1 = 1.0 \quad A_2/A_1 = 8$$



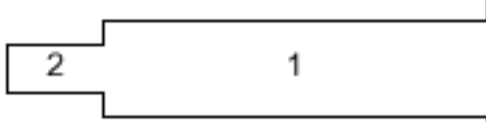
$$L_1 + L_2 = 17.6 \text{ cm}$$

$$L_2/L_1 = 1.5 \quad A_2/A_1 = 8$$



$$L_1 + L_2 = 14.5 \text{ cm}$$

$$L_2/L_1 = 1/3 \quad A_2/A_1 = 1/8$$



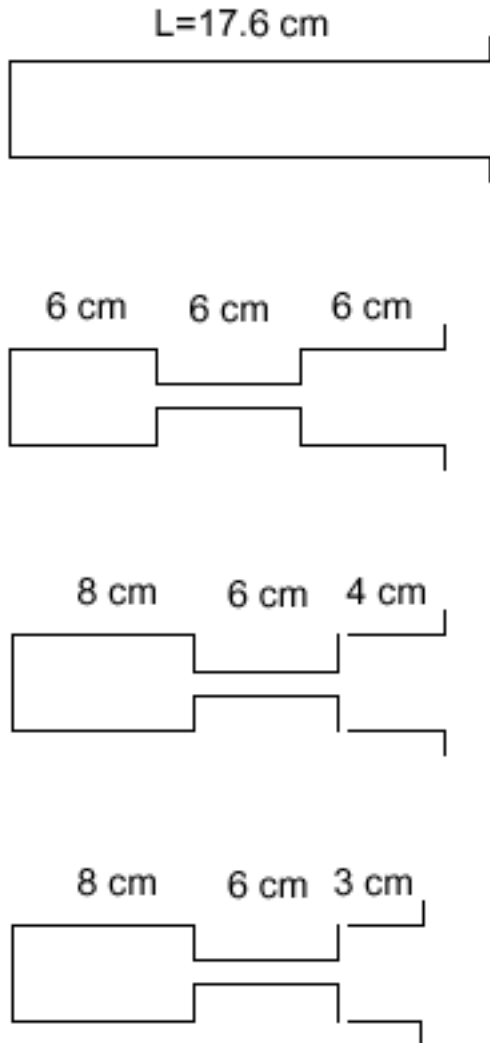
Formant Patterns

| F_1 | F_2 | F_3 | F_4 |
|-------------------|--------------------|--------------------|--------------------|
| x 500 | x 1500 | x 2500 | x 3500 |
| F_1 x 320 | F_2 x 1200 | F_3 x 2300 | F_4 x 3430 |
| F_1 x 780 | F_2 x 1240 | F_3 x 2720 | F_4 x 3350 |
| F_1 x 220 | F_2 x 1800 | F_3 x 2230 | F_4 x 3800 |
| F_1 x 260 | F_2 x 1990 | F_3 x 3050 | F_4 x 4130 |
| F_1 x 630 | F_2 x 1770 | F_3 x 2280 | F_4 x 3440 |

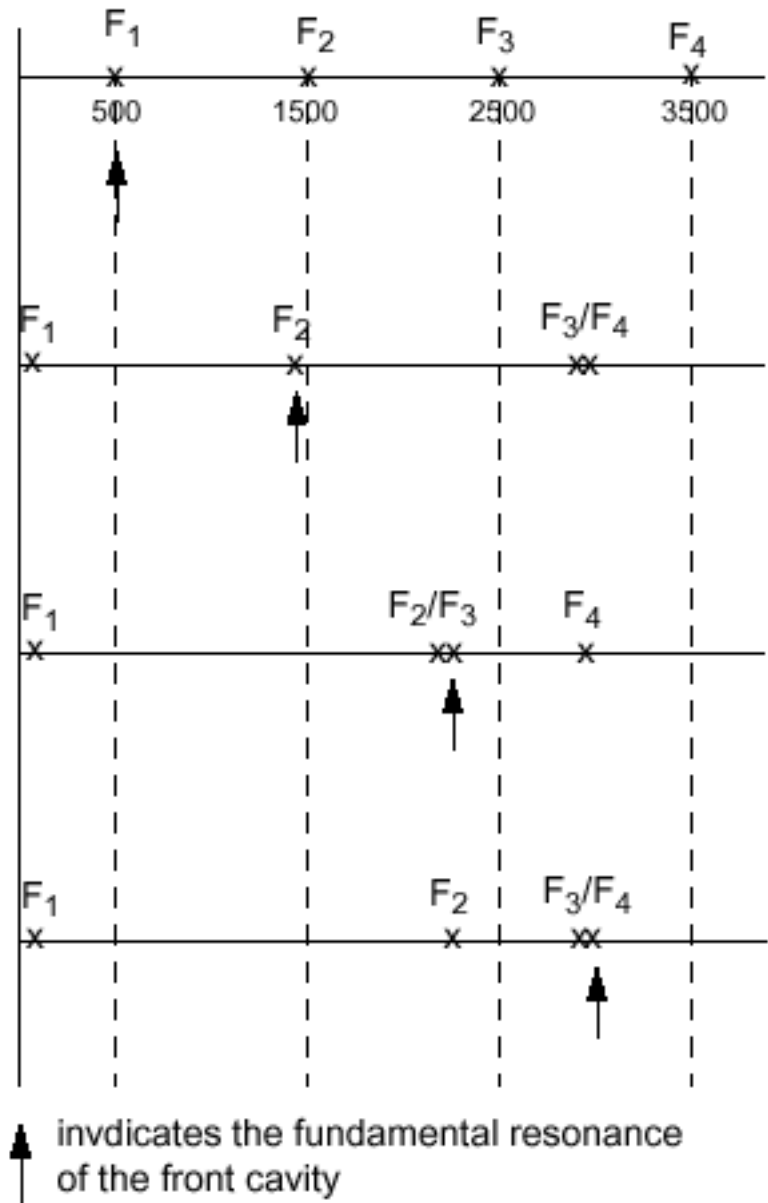
$$\longleftrightarrow L_1 + L_2 = 17.6 \text{ cm} \longrightarrow$$

THREE TUBE MODELS

Resonator Geometry



Formant Patterns



TRANSFER FUNCTION OF THE LOSSLESS TUBE MODEL

Recall, $V(\Omega) = \frac{U_L(\Omega)}{U_G(\Omega)}$. In the discrete domain, we can write: $V(z) = \frac{U_L(z)}{U_G(z)}$.

Following our derivation of the wave equation, we can express the transfer function for a lossless tube as follows:

$$U_k = \mathbf{Q}_k U_{k-1}$$

where

$$U_k = \begin{bmatrix} U_k^+(z) \\ U_k^-(z) \end{bmatrix} \quad \text{and} \quad \mathbf{Q}_k = \begin{bmatrix} \frac{z^{1/2}}{1+r_k} & \frac{-r_k z^{1/2}}{1+r_k} \\ \frac{-r_k z^{1/2}}{1+r_k} & \frac{z^{-1/2}}{1+r_k} \end{bmatrix}$$

The combined transfer function is a product of these matrices. The net result is a transfer function that can be expressed as:

$$V(z) = \frac{0.5(1+r_G) \prod_{k=1}^N (1+r_k) z^{-N/2}}{D(z)}$$

where

$$D(z) = \begin{bmatrix} 1 & -r_G \end{bmatrix} \begin{bmatrix} 1 & -r_1 \\ -r_1 z^{-1} & z^{-1} \end{bmatrix} \cdots \begin{bmatrix} 1 & -r_N \\ -r_N z^{-N} & z^{-1} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

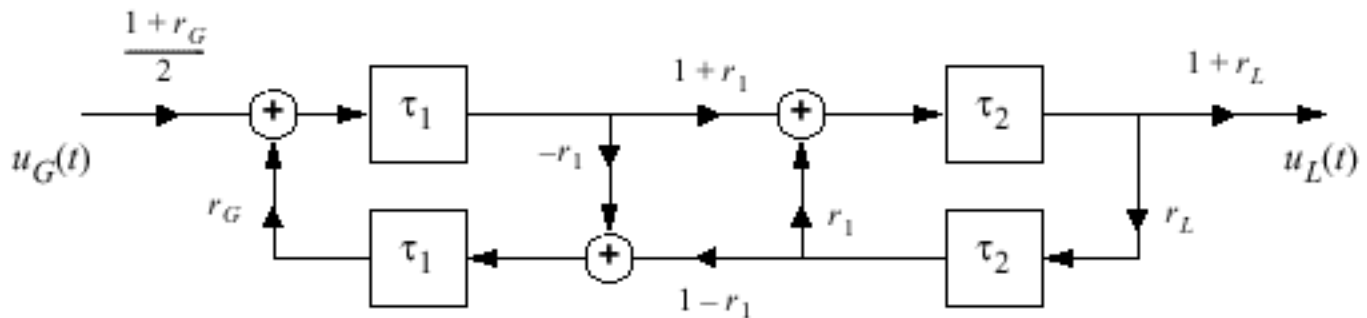
We can write $D(z)$ in a simpler form:

$$D(z) = 1 - \sum_{k=1}^N \alpha_k z^{-k}$$

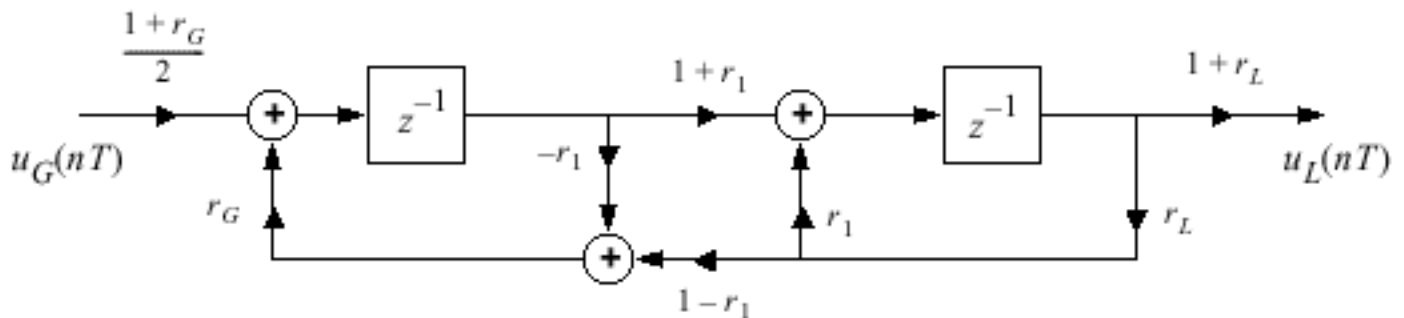
Why is this important?

DIGITAL SPEECH PRODUCTION MODELS

Recall our concatenated lossless tube model:



We can approximate this as a digital filter using the sampling theorem:



The transfer function of an N-tube model is:

$$V(z) = \frac{0.5(1+r_G) \prod_{k=1}^N (1+r_k) z^{-N/2}}{D(z)}$$

where

$$D(z) = \begin{bmatrix} 1 & -r_G \end{bmatrix} \begin{bmatrix} 1 & -r_1 \\ -r_1 z^{-1} & z^{-1} \end{bmatrix} \cdots \begin{bmatrix} 1 & -r_N \\ -r_N z^{-N} & z^{-1} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

We can compute $D(z)$ recursively:

$$D_0(z) = 1$$

$$D_k(z) = D_{k-1}(z) + r_k z^{-k} D_{k-1}(z^{-1}) \quad k = 1, 2, \dots, N$$

$$D(z) = D_N(z)$$

ALTERNATE DIGITAL FILTER IMPLEMENTATIONS USING DIGITAL RESONATORS

Note that for $D(z)$ to have real coefficients, zeros must occur in complex conjugate pairs. We can transform zeros in the Laplace domain:

$$s_k, s_k^* = -\sigma \pm j2\pi F_k$$

The corresponding complex conjugate poles in the discrete-domain are:

$$\begin{aligned} z_k, z_k^* &= e^{-\sigma_k T} e^{\pm j2\pi F_k T} \\ &= e^{-\sigma_k T} \cos(2\pi F_k T) \pm j e^{-\sigma_k T} \sin(2\pi F_k T) \end{aligned}$$

Note that magnitude of the pole in the z -plane is related to the bandwidth.

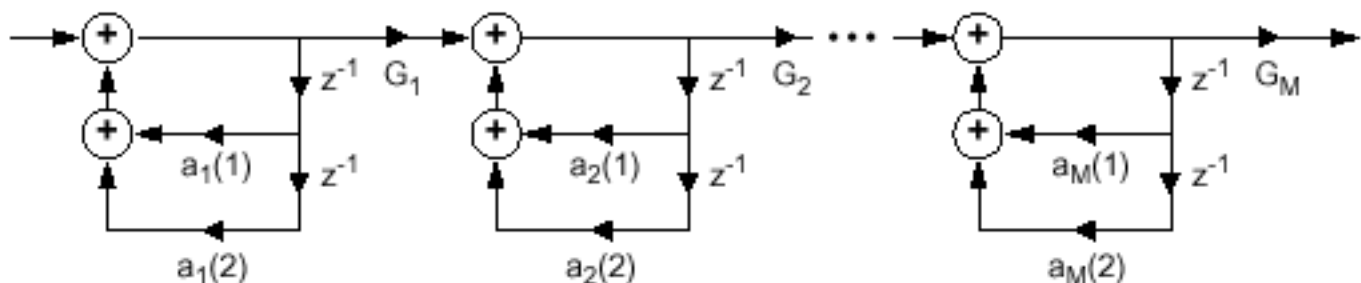
We can write a transfer function as a product of these poles:

$$V(z) = \prod_{k=1}^M V_k(z)$$

where

$$V_k(z) = \frac{(1 - 2|z_k| \cos(2\pi F_k T) + |z_k|^2)}{(1 - 2|z_k| \cos(2\pi F_k T) z^{-1} + |z_k|^2 z^{-2})}$$

This is an all-pole filter. It can be realized using a number of structures:
Under what conditions is this filter stable?



where,

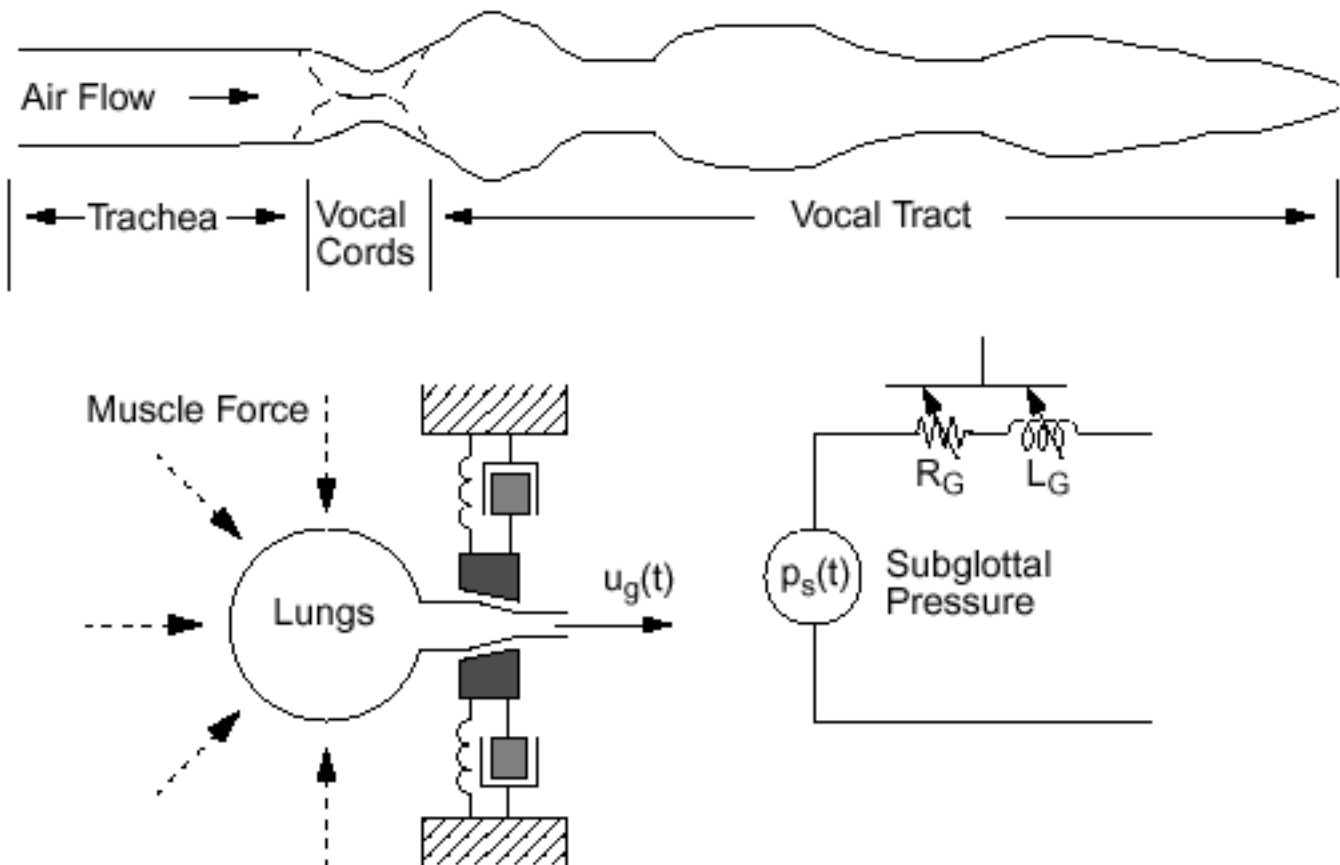
$$V(z) = \prod_{k=1}^M G_k$$

$$r_k(z) = \frac{1}{1 - a_k(1)z^{-1} - a_k(2)z^{-2}}$$

$$a_k(1) = 2|z_k| \cos(2\pi F_k T) \quad a_k(2) = -|z_k|^2 \quad G_k = 1 - 2|z_k| \cos(2\pi F_k T) + |z_k|^2$$

EXCITATION MODELS

How do we couple energy into the vocal tract?



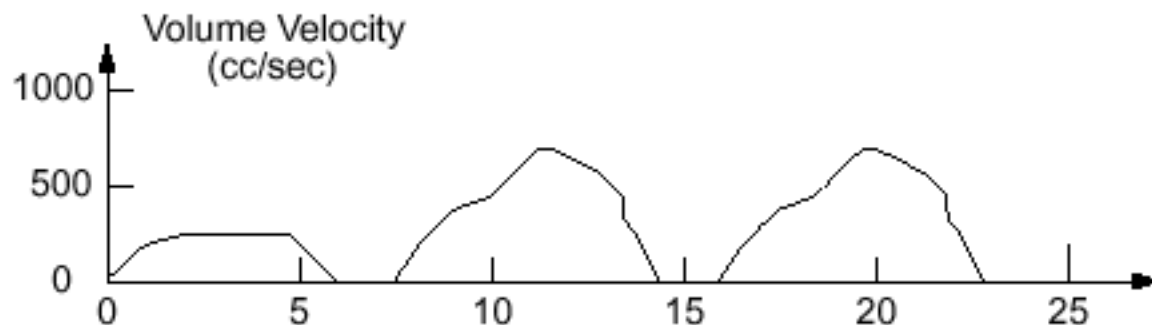
The glottal impedance can be approximated by:

$$Z_G = R_G + j\Omega L_G$$

The boundary condition for the volume velocity is:

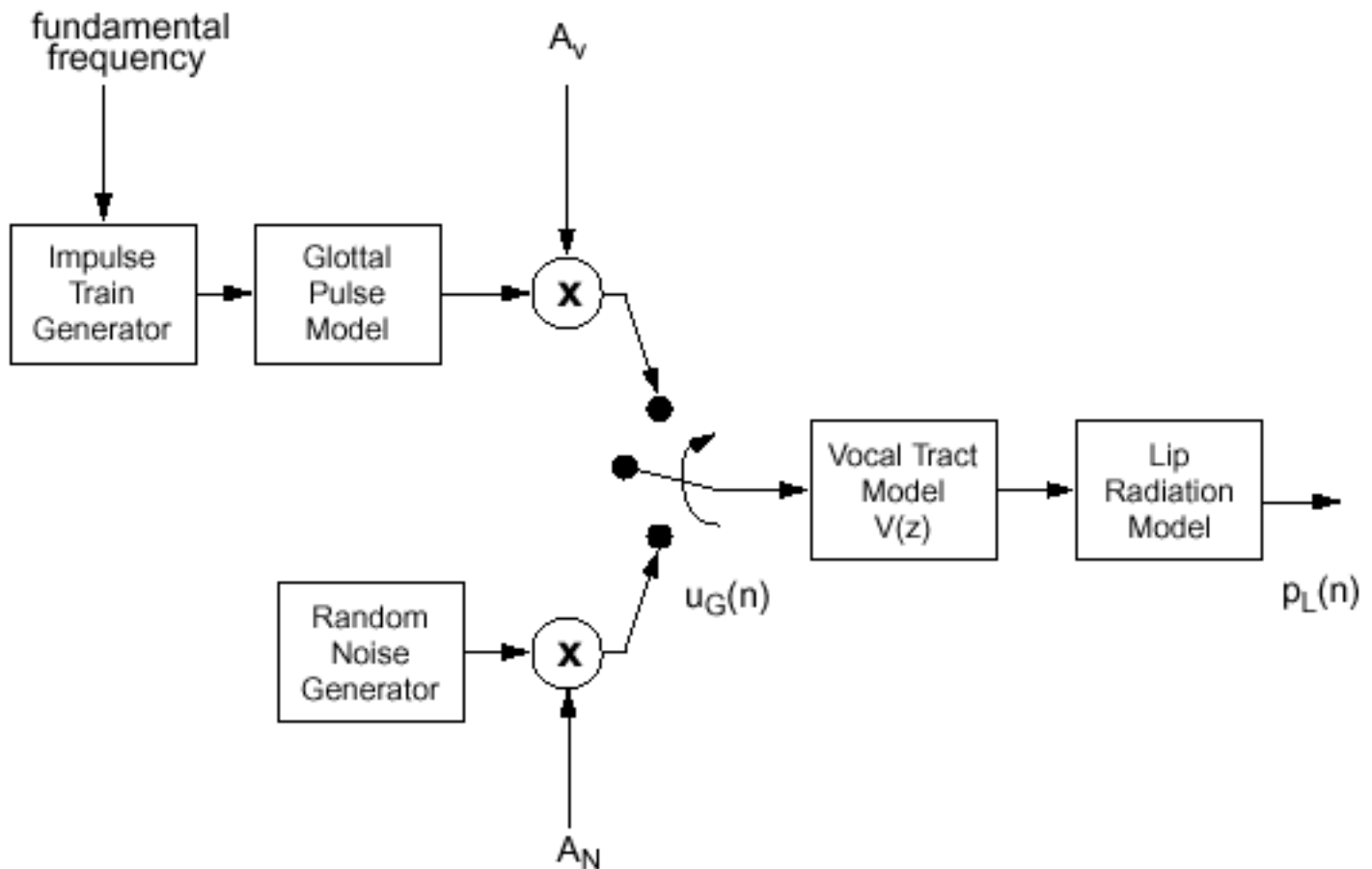
$$U(0, \Omega) = U_G(\Omega) - P(0, \Omega)/Z_G(\Omega)$$

For voiced sounds, the glottal volume velocity looks something like this:



time (ms)

THE VOCODER (COMPLETE) DIGITAL MODEL



Notes:

- Sample frequency is typically 8 kHz to 16 kHz
- Frame duration is typically 10 msec to 20 msec
- Window duration is typically 30 msec
- Fundamental frequency ranges from 50 Hz to 500 Hz
- Three resonant frequencies are usually found within 4 kHz bandwidth
- Some sounds, such as sibilants ("s") have extremely high bandwidths

Questions:

What does the overall spectrum look like?

What happened to the nasal cavity?

What is the form of $V(z)$?

What is the form of $v(x)$?

[Return to Main](#)[Objectives](#)**Transduction:**[Sound Pressure Level](#)[Physiology of the Ear](#)**Perception:**[Psychoacoustics](#)[Equal Loudness](#)[Bark and Mel Scales](#)[Comparison](#)**On-Line Resources:**[Signal Modeling](#)[Bilinear Transform](#)[Auditory Web](#)[Auditory.Org](#)

LECTURE 04: HEARING PHYSIOLOGY

- Objectives:
 - Basic physiology
 - Frequency response implications
 - Nonlinear frequency warping
 - Bark and Mel scales

Note that this lecture is primarily based on material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language*

Processing - A Guide to Theory, Algorithm, and System Development, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

In addition, information from:

D. O'Shaughnessy, *Speech Communications: Human and Machine*, IEEE Press, ISBN: 0-7803-3449-3, 2000.

has been used for the first slide.

LECTURE 04: HEARING PHYSIOLOGY

- Objectives:
 - Basic physiology
 - Frequency response implications
 - Nonlinear frequency warping
 - Bark and Mel scales

Note that this lecture is primarily based on material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA,

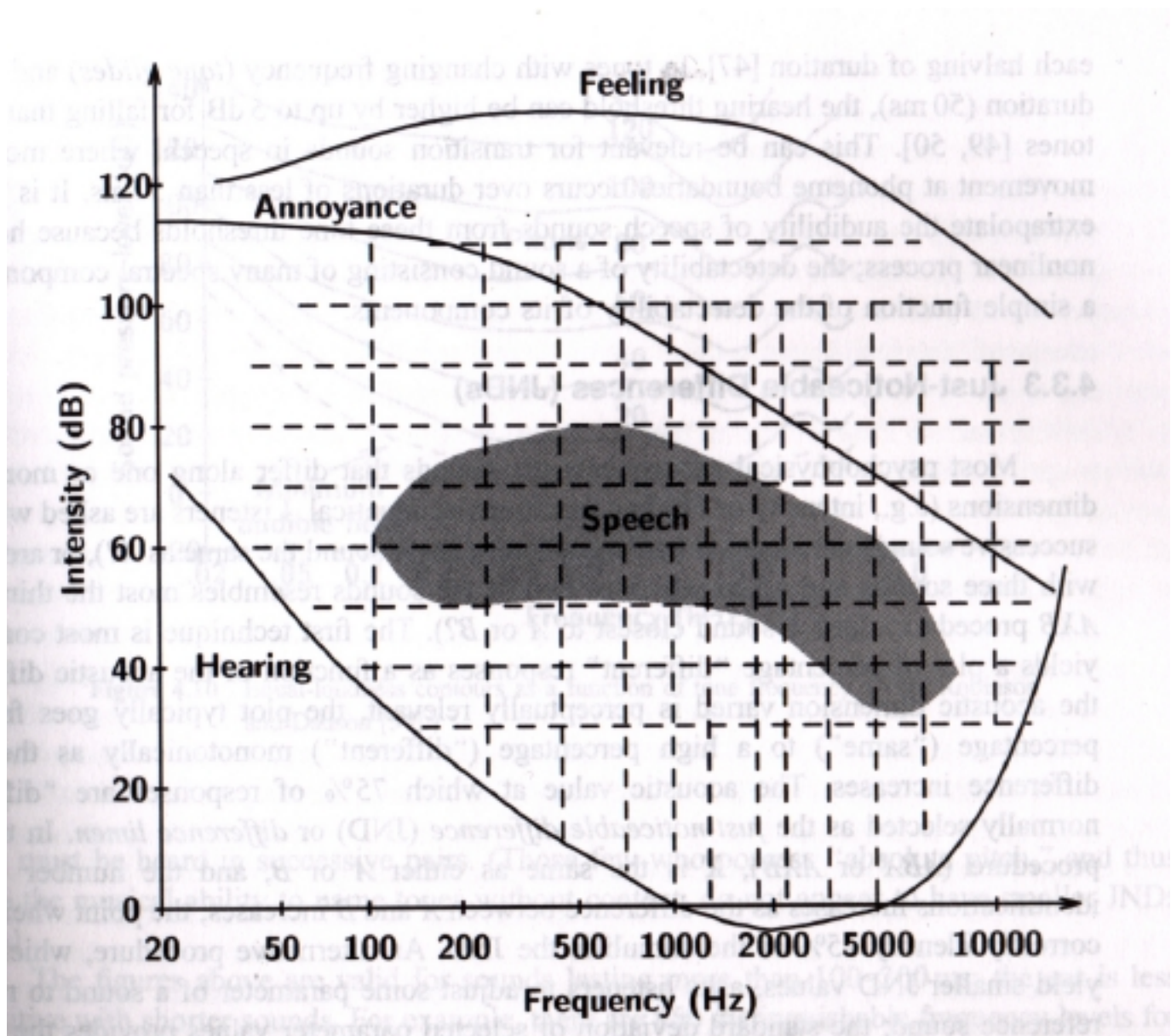
ISBN: 0-13-022616-5, 2001.

In addition, information from:

D. O'Shaughnessy, *Speech Communications: Human and Machine*, IEEE Press, ISBN: 0-7803-3449-3, 2000.

has been used for the first slide.

SOUND PRESSURE LEVEL



Key points:

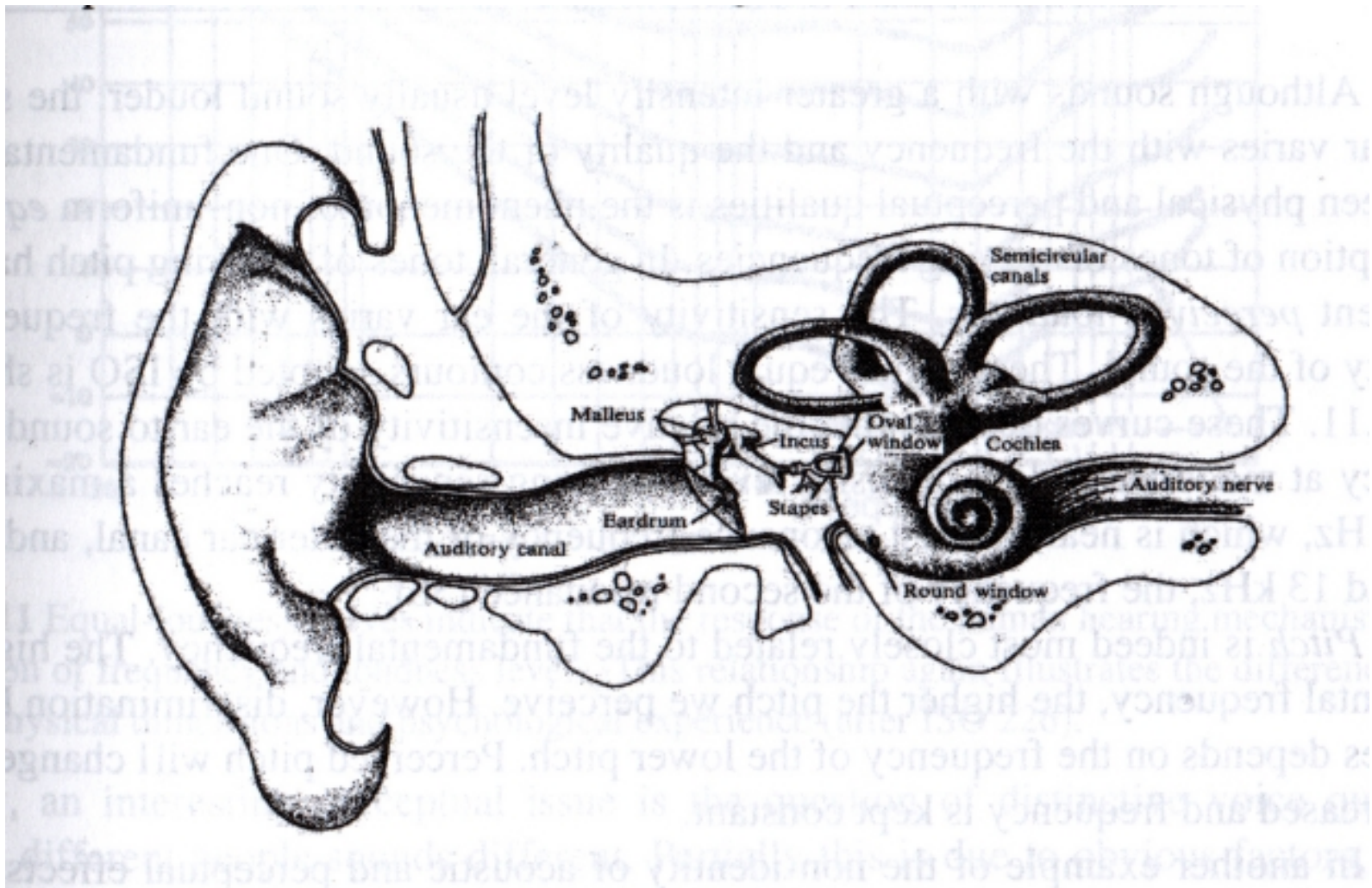
- The ear is the most sensitive human organ. Vibrations on the order of angstroms are used

to transduce sound. It has the largest dynamic range (~ 140 dB) of any organ in the human body.

- The lower portion of the curve is an audiogram - hearing sensitivity. It can vary up to 20 dB across listeners.
- Above 120 dB corresponds to your favorite heavy metal rock and roll band (or standing under a Boeing 747 when it takes off).
- Typical ambient office noise is about 55 dB.
- Three common [weighting scales](#) exist for intensity - A (SPL in the range 20 to 55 dB), B (SPL in the range 55 to 85 dB), and C (85 dB or more). A weighting is used most often in speech research (and by the government

when setting regulations).

PHYSIOLOGY OF THE EAR



Key points:

- Three main sections: outer, middle, and inner. The outer and middle ears reproduce the analog signal (impedance matching); the inner ear transduces the pressure wave into an electrical

signal.

- The outer ear consists of the external visible part and the auditory canal. The tube is about 2.5 cm long.
- The middle ear consists of the eardrum and three bones (malleus, incus, and stapes). It converts the sound pressure wave to displacement of the oval window (entrance to the inner ear).
- The inner ear primarily consists of a fluid-filled tube (cochlea) which contains the basilar membrane. Fluid movement along the basilar membrane displaces hair cells, which generate electrical signals.
- There are a discrete number of hair cells (30,000). Each hair cell is tuned to a different frequency.

- Place vs. Temporal Theory: firings of hair cells are processed by two types of neurons (onset chopper units for temporal features and transient chopper units for spectral features).
- Most mammals have similar hearing systems (cats and chinchillas are popular animals for experimentation).

PHYSICAL VS. PERCEPTUAL ATTRIBUTES

- Psychoacoustics: a branch of science dealing with hearing, the sensations produced by sounds.
- A basic distinction must be made between the perceptual attributes of a sound and measurable physical quantities:

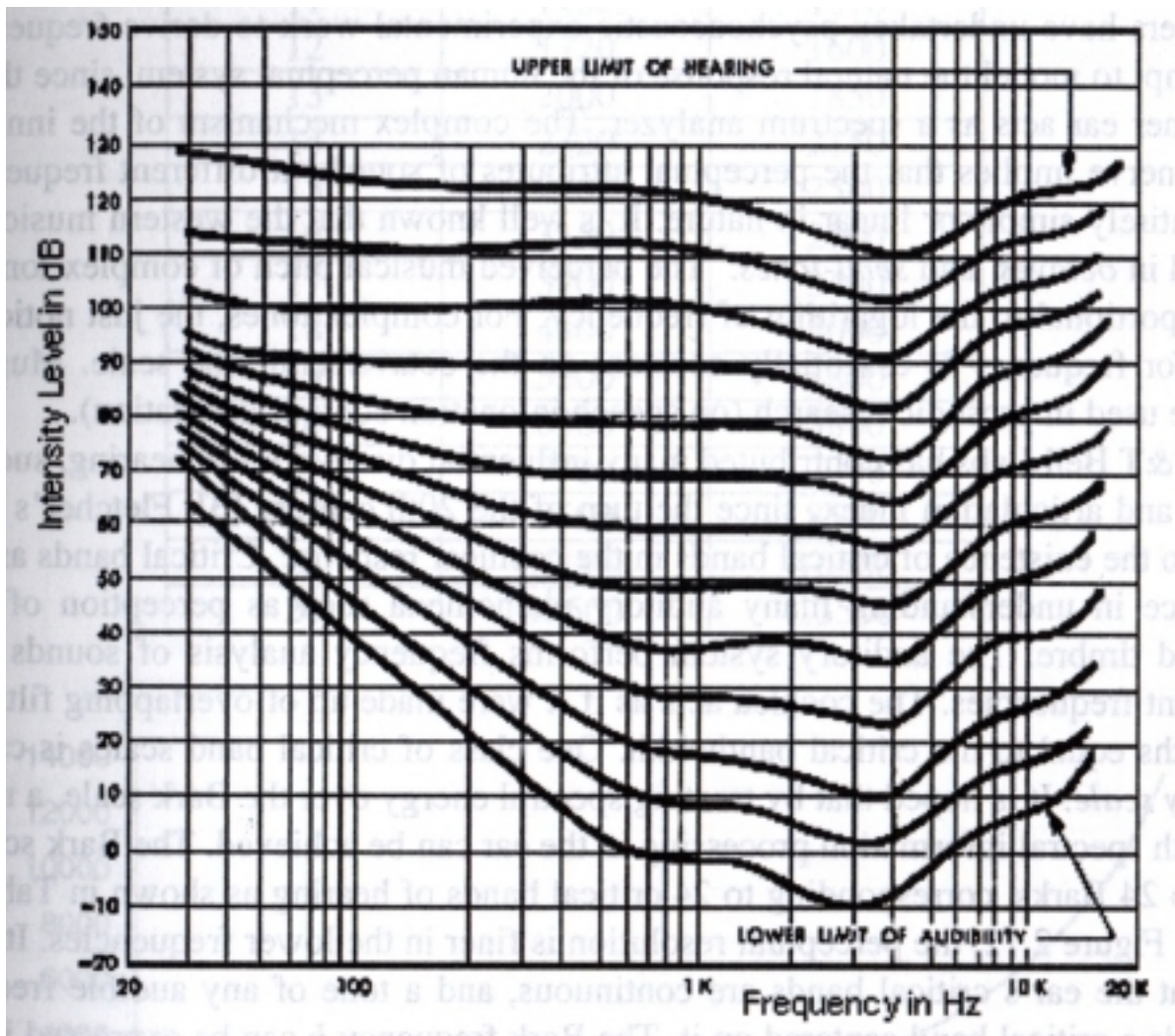
| Physical Quantity | Perceptual Quality |
|----------------------------------|-------------------------------|
| Intensity | Loudness |
| Fundamental Frequency | Pitch |

| | |
|--|----------|
| Spectral Shape | Timbre |
| Onset/Offset Time | Timing |
| Phase Difference (Binaural Hearing) | Location |

- Many physical quantities are perceived on a logarithmic scale (e.g. loudness). Our perception is often a nonlinear function of the absolute value of the physical quantity being measured (e.g. equal loudness).

- Timbre can be used to describe why musical instruments sound different.
- What factors contribute to speaker identity?

EQUAL LOUNDESS CURVES



- **Just Noticeable Difference (JND):** The acoustic value at which 75% of responses judge stimuli to be different (also known as a difference limen).
- The perceptual loudness of a sound is

specified via its relative intensity above the threshold. A sound's loudness is often defined in terms of how intense a reference 1 kHz tone must be heard to sound as loud.

NONLINEAR FREQUENCY WARPING: BARK AND MEL SCALES

- **Critical Bandwidths:** correspond to approximately 1.5 mm spacings along the basilar membrane, suggesting a set of 24 bandpass filters.
- **Critical Band:** can be related to a bandpass filter whose frequency response corresponds to the tuning curves of an auditory neurons. A frequency range over which two sounds will sound like they are fusing into one.
- **Bark Scale:**

$$Bark = 13 \operatorname{atan}\left(\frac{0.76f}{1000}\right) + 3.5 \operatorname{atan}\left(\frac{f^2}{(7500)^2}\right)$$

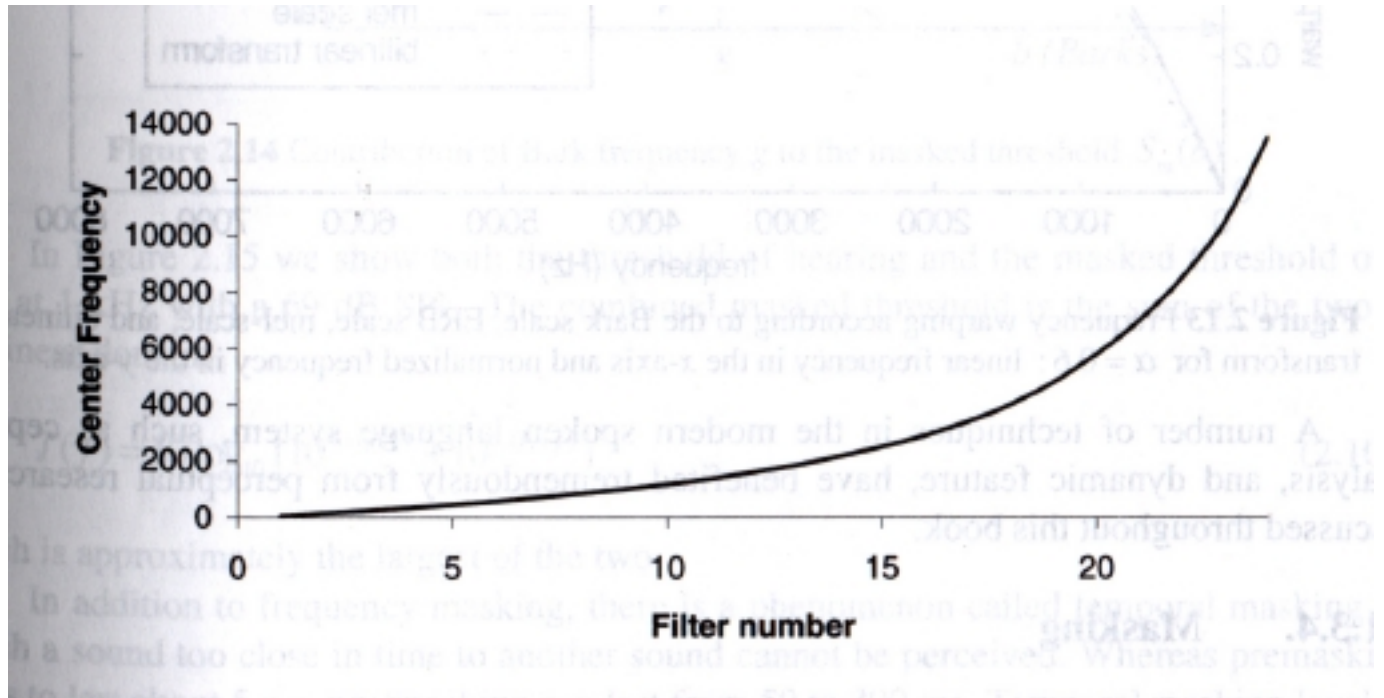
- **Mel Scale:**

$$mel\ frequency = 2595 \log_{10} (1 + f/700.0)$$

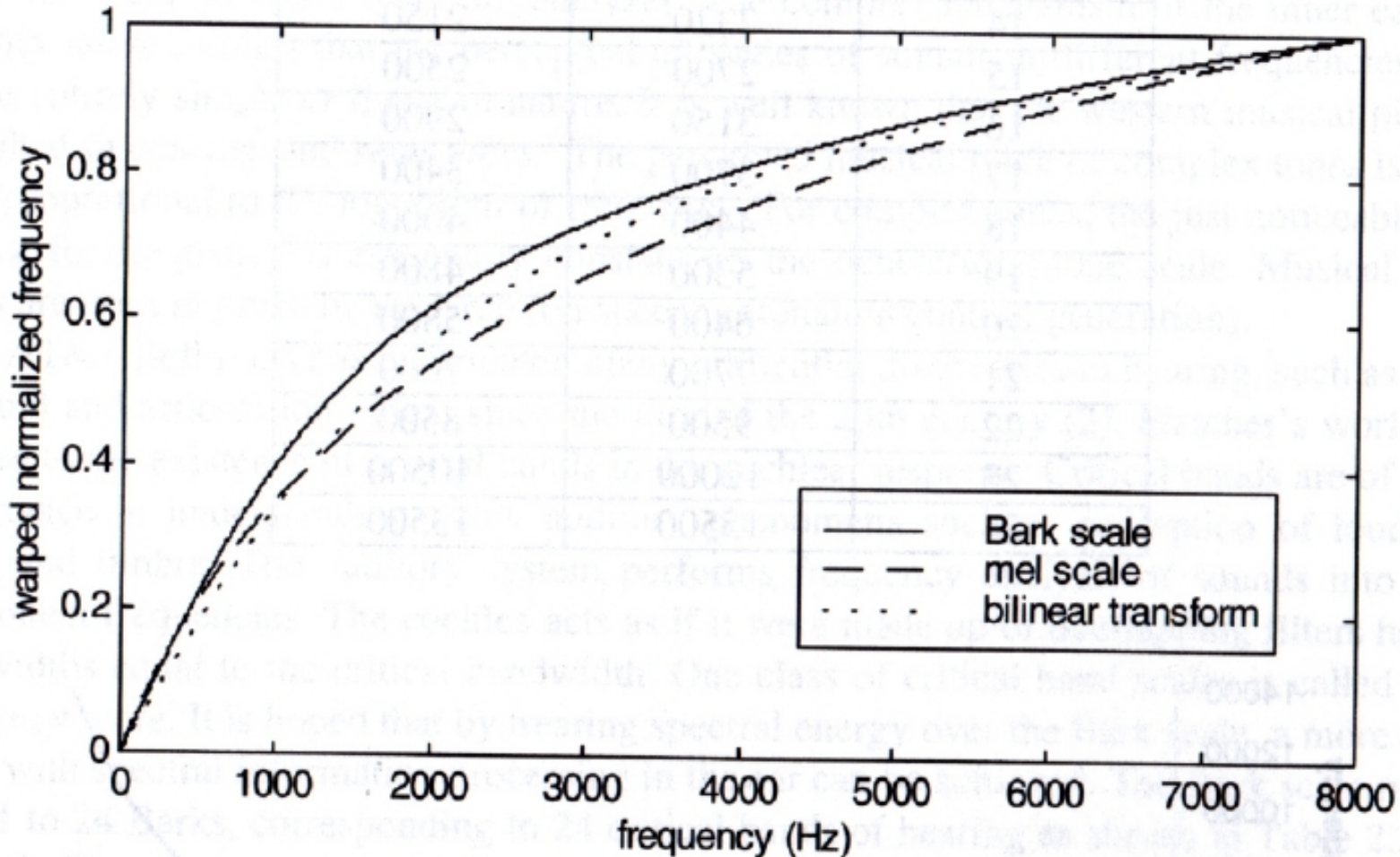
- **Comparison:** filter bank implementations for a typical speech recognizer.

| Index | Bark Scale | | Mel Scale | |
|-------|-------------------|---------|-------------------|---------|
| | Center Freq. (Hz) | BW (Hz) | Center Freq. (Hz) | BW (Hz) |
| 1 | 50 | 100 | 100 | 100 |
| 2 | 150 | 100 | 200 | 100 |
| 3 | 250 | 100 | 300 | 100 |
| 4 | 350 | 100 | 400 | 100 |
| 5 | 450 | 110 | 500 | 100 |
| 6 | 570 | 120 | 600 | 100 |
| 7 | 700 | 140 | 700 | 100 |
| 8 | 840 | 150 | 800 | 100 |
| 9 | 1000 | 160 | 900 | 100 |
| 10 | 1170 | 190 | 1000 | 124 |
| 11 | 1370 | 210 | 1149 | 160 |
| 12 | 1600 | 240 | 1320 | 184 |
| 13 | 1850 | 280 | 1516 | 211 |
| 14 | 2150 | 320 | 1741 | 242 |
| 15 | 2500 | 380 | 2000 | 278 |
| 16 | 2900 | 450 | 2297 | 320 |
| 17 | 3400 | 550 | 2639 | 367 |
| 18 | 4000 | 700 | 3031 | 422 |
| 19 | 4800 | 900 | 3482 | 484 |
| 20 | 5800 | 1100 | 4000 | 556 |
| 21 | 7000 | 1300 | 4595 | 639 |
| 22 | 8500 | 1800 | 5278 | 734 |
| 23 | 10500 | 2500 | 6063 | 843 |
| 24 | 13500 | 3500 | 6964 | 969 |

- **Nonlinear Frequency Warping: The Bark scale** implies a nonlinear frequency mapping of frequency.



A COMPARISON OF FREQUENCY WARPING FUNCTIONS



- Follow this link for more information on the [bilinear transform](#).

[Return to Main](#)

[Objectives](#)

Masking:

[Tone-Masking Noise](#)

[Noise-Masking Tone](#)

[Perceptual Noise-Weighting](#)

Other Phenomena:

[Echo and Delay](#)

[Adaptation](#)

[Timing](#)

Summary:

[Digital Models](#)

On-Line Resources:

[Auditory Masking](#)

[Cochlear Models](#)

[McGurk Effect](#)

LECTURE 05: PERCEPTION AND MASKING

- Objectives:
 - Frequency and temporal masking
 - Introduce other impairments such as echo and appreciate how they impact speech processing systems
 - Appreciate how we can exploit properties of masking in speech analysis
 - Summarize our digital models/approximations

Note that this lecture is primarily

based on material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

In addition, information from:

D. O'Shaughnessy, *Speech Communications: Human and Machine*, IEEE Press, ISBN: 0-7803-3449-3, 2000.

has been used.

LECTURE 05: PERCEPTION AND MASKING

- Objectives:
 - Frequency and temporal masking
 - Introduce other impairments such as echo and appreciate how they impact speech processing systems
 - Appreciate how we can exploit properties of masking in speech analysis
 - Summarize our digital models/approximations

Note that this lecture is primarily based on material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

In addition, information from:

D. O'Shaughnessy, *Speech Communications: Human and Machine*, IEEE Press, ISBN: 0-7803-3449-3, 2000.

has been used.

● TONE-MASKING NOISE

- **Frequency masking:** one sound cannot be perceived if another sound close in frequency has a high enough level. The first sound *masks* the second.
- **Tone-masking noise:** noise with energy E_N (dB) at Bark frequency g masks a tone at Bark frequency b if the tone's energy is below the threshold:

$$T_T(b) = E_N - 6.025 - 0.275g + S_m(b-g)$$

(dB SPL)

where the *spread-of-masking* function $S_m(b)$ is given by:

$$S_m(b) = 15.81 + 7.5(b+0.474) - 17.5 \sqrt{1 + (b+0.474)^2} \quad (\text{dB})$$

- **Temporal Masking:** onsets of sounds are masked in the time domain through a similar masking process.

Key points:

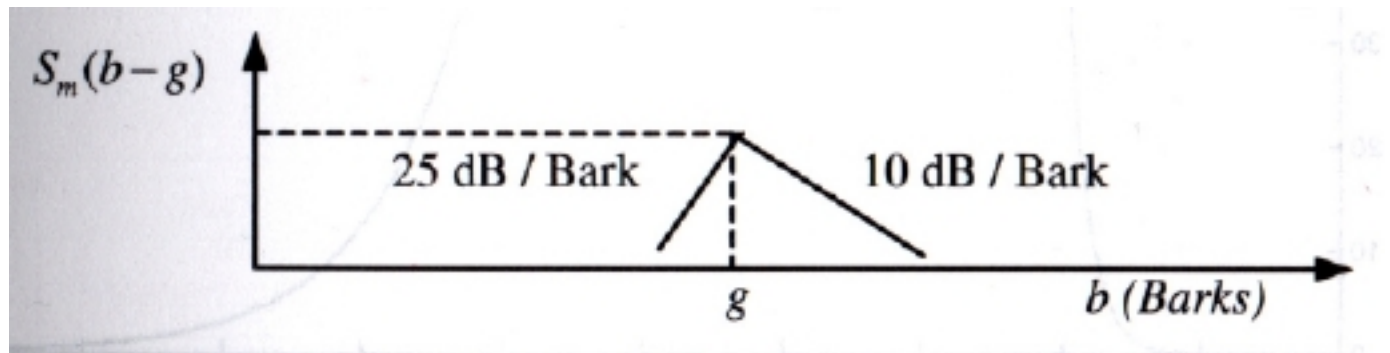
- Thresholds are frequency and energy dependent.
- Thresholds depend on the nature of the sound as well.

NOISE-MASKING TONE

- **Noise-masking tone:** a tone at Bark frequency g energy E_T (dB) masks noise at Bark frequency b if the noise energy is below the threshold:

$$T_N(b) = E_T - 2.025 - 0.17g + S_m(b-g) \quad (\text{dB SPL})$$

- Masking thresholds are commonly referred to as Bark scale functions of *just noticeable differences* (JND).

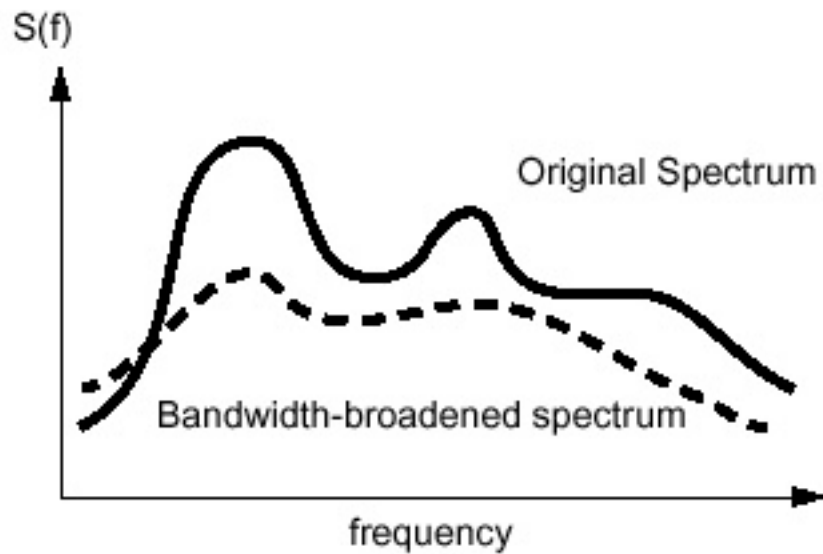


Key points:

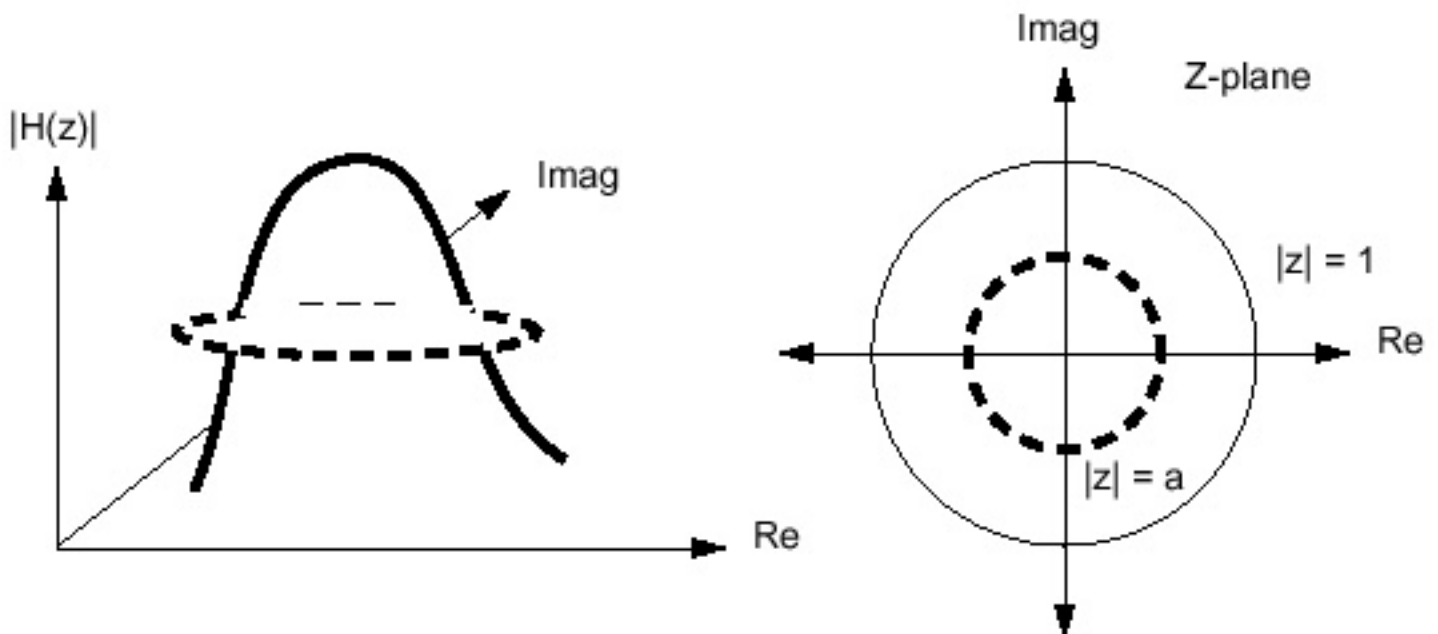
- Thresholds are not symmetric.
- Thresholds depend on the nature of the noise and the sound.

PERCEPTUAL NOISE WEIGHTING

- **Noise-weighting:** shaping the spectrum to hide noise introduced by imperfect analysis and modeling techniques (essential in speech coding).
- Humans are sensitive to noise introduced in low-energy areas of the spectrum.
- Humans tolerate more additive noise when it falls under high energy areas the spectrum. The amount of noise tolerated is greater if it is spectrally shaped to match perception.
- We can simulate this phenomena using "bandwidth-broadening":



- Simple Z-Transform interpretation:



which can be implemented by evaluating the

Z-Transform around a contour closer to the origin in the z -plane: $H_{nw}(z) = H(az)$.

Key points:

- Used in many speech compression systems (Code Excited Linear Prediction).
- Analysis performed on bandwidth-broadened speech; synthesis performed using normal speech. Effectively shapes noise to fall under the formants.

ECHO, THE LOMBARD EFFECT, AND TIME DELAY

- Humans are used to hearing their voice while they speak - real-time feedback (side tone).
- When we place headphones over our ears, which dampens this feedback, we tend to speak louder.
- **Lombard Effect:** Humans speak louder in the presence of ambient noise.
- When this side-tone is delayed, it interrupts our cognitive processes, and degrades our speech.
- This effect begins at delays of approximately

250 ms.

- Modern telephony systems have been designed to maintain delays lower than this value (long distance phone calls routed over satellites).
- Digital speech processing systems can introduce large amounts of delay due to non-real-time processing.

ADAPTATION

- **Adaptation** refers to changing sensitivity in response to a continued stimulus, and is likely a feature of the mechanoelectrical transformation in the cochlea.
- Neurons tuned to a frequency where energy is present do not change their firing rate drastically for the next sound.
- Additive broadband noise does not significantly change the firing rate for a neuron in the region of a formant.
- The [McGurk Effect](#) is an auditory illusion which results from combining a face pronouncing a certain syllable with the sound of a different syllable. The illusion is stronger

for some combinations than for others. For example, an auditory 'ba' combined with a visual 'ga' is perceived by some percentage of people as 'da'. A larger proportion will perceive an auditory 'ma' with a visual 'ka' as 'na'. Some researchers have measured evoked electrical signals matching the "perceived" sound.

TIMING

- Temporal resolution of the ear is crucial.
- Two clicks are perceived monaurally as one unless they are separated by at least 2 ms.
- 17 ms of separation is required before we can reliably determine the order of the clicks.
- Sounds with onsets faster than 20 ms are perceived as "plucks" rather than "bows".
- Short sounds near the threshold of hearing must exceed a certain intensity-time product to be perceived.
- Humans do not perceive individual

"phonemes" in fluent speech - they are simply too short. We somehow integrate the effect over intervals of approximately 100 ms.

- Humans are very sensitive to long-term periodicity (ultra low frequency) - has implications for random noise generation.

DIGITAL MODELS FOR PERCEPTION

- Logarithmic processing of energy.
- Energy normalization.
- Nonlinear warping of the frequency scale.
- Filter bank analysis (wavelets).
- Cochlear models have not been extremely effective.

[Return to Main](#)

[Objectives](#)

Definitions:

[Phonetics and Phonology](#)

[English](#)

[Transcription Standards](#)

[Comparison](#)

Phonetics:

[The Vowel Space](#)

[Formant Frequencies](#)

[Bandwidth](#)

Summary:

[Acoustic Theory](#)

[Consonants](#)

On-Line Resources:

[Ladefoged: Sounds](#)

[Ladefoged's Home Page](#)

[Phonlab](#)

[Peterson-Barney Data](#)

[HLT Central](#)

LECTURE 06: PHONETICS AND PHONOLOGY

- Objectives:
 - Linguistics 101
 - Understand the relationship between acoustic models of speech production physiology and linguistic models of language
 - Introduce potential acoustic units for our speech recognition system
 - Understand how linguistic structure influences our approaches to speech

recognition

Note that this lecture is primarily based on material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

In addition, information from:

J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co., ISBN: 0-7803-5386-2, 2000.

has been used.

LECTURE 06: PHONETICS AND PHONOLOGY

- Objectives:
 - Linguistics 101
 - Understand the relationship between acoustic models of speech production physiology and linguistic models of language
 - Introduce potential acoustic units for our speech recognition system
 - Understand how linguistic structure influences our approaches to speech recognition

Note that this lecture is primarily based on material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

In addition, information from:

J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co., ISBN: 0-7803-5386-2, 2000.

has been used.

PHONEMICS (PHONOLOGY) AND PHONETICS

Some basic definitions:

- **Phoneme:**

- an ideal sound unit with a complete set of articulatory gestures.
- the basic theoretical unit for describing how speech conveys linguistic meaning.
- In English, there are about 42 phonemes.
- Types of phonemes: vowels, semivowels, diphthongs, and consonants.

- **Phonemics:** the study of abstract units and their relationships in a language

- **Phone:** the actual sounds that are produced in speaking (for example, "d" in letter pronounced "l e d er").
- **Phonetics:** the study of the actual sounds of the language
- **Allophones:** the collection of all minor variants of a given sound ("t" in eight versus "t" in "top")
- **Monophones, Biphones, Triphones:** sequences of one, two, and three phones. Most often used to describe acoustic models.

Three branches of phonetics:

- **Articulatory phonetics:** manner in which the speech sounds are produced by the

articulators of the vocal system.

- **Acoustic phonetics:** sounds of speech through the analysis of the speech waveform and spectrum
- **Auditory phonetics:** studies the perceptual response to speech sounds as reflected in listener trials.

Issues:

- Broad phonemic transcriptions vs. narrow phonetic transcriptions

ENGLISH PHONEMES

Vowels and Diphthongs

| Phonemes | Word Examples | Description |
|----------|-------------------|-----------------------------------|
| iy | feel, eve, me | front close unrounded |
| ih | fill, hit, lid | front close unrounded (lax) |
| ae | at, carry, gas | front open unrounded (tense) |
| aa | father, ah, car | back open rounded |
| ah | cut, bud, up | open mid-back rounded |
| ao | dog, lawn, caught | open-mid back round |
| ay | tie, ice, bite | diphthong with quality: aa + ih |
| ax | ago, comply | central close mid (schwa) |
| ey | ate, day, tape | front close-mid unrounded (tense) |
| eh | pet, berry, ten | front open-mid unrounded |
| er | turn, fur, meter | central open-mid unrounded |
| ow | go, own, town | back close-mid rounded |
| aw | foul, how, our | diphthong with quality: aa + uh |
| oy | toy, coin, oil | diphthong with quality: ao + ih |
| uh | book, pull, good | back close-mid unrounded (lax) |
| uw | tool, crew, moo | back close round |

Consonants and Liquids

| Phonemes | Word Examples | Description |
|----------|----------------------|----------------------------------|
| b | big, able, tab | voiced bilabial plosive |
| p | put, open, tap | voiceless bilabial plosive |
| d | dig, idea, wad | voiced alveolar plosive |
| t | talk, sat | voiceless alveolar plosive |
| g | gut, angle, tag | voiced velar plosive |
| ɾ | meter | alveolar flap |
| ŋ | gut, angle, tag | voiced velar plosive |
| k | cut, ken, take | voiceless velar plosive |
| f | fork, after, if | voiceless labiodental fricative |
| v | vat, over, have | voiced labiodental fricative |
| s | sit, cast, toss | voiceless alveolar fricative |
| z | zap, lazy, haze | voiced alveolar fricative |
| θ | thin, nothing, truth | voiceless dental fricative |
| ð | then, father, scythe | voiced bilabial plosive |
| ʃ | she, cushion, wash | voiceless postalveolar fricative |
| ʒ | genre, azure | voice postalveolar fricative |

| | | |
|----|----------------------------|--------------------------------------|
| l | lid | alveolar lateral approximant |
| l | elbow, sail | velar lateral approximant |
| r | red, part, far | retroflex approximant |
| y | yacht, yard | palatal sonorant glide |
| w | with, away | labiovelar sonorant glide |
| hh | help, ahead, hotel | voiceless glottal fricative |
| m | mat, amid, aim | bilabial nasal |
| n | no, end, pan | alveolar nasal |
| ng | sing, anger | velar nasal |
| ch | chin, archer, march | voiceless alveolar affricate: t + sh |
| jh | joy, agile, edge | voiced alveolar affricate: d + zh |

TRANSCRIPTION STANDARDS

Major governing bodies for phonetic alphabets:

- **International Phonetic Alphabet (IPA)**: over 100 years of history
- **ARPAbet**: developed in the late 1970's to support ARPA research
- **TIMIT**: TI/MIT variant of ARPAbet used for the TIMIT corpus
- **Worldbet**: developed by Hieronymous (AT&T) to deal with multiple languages within a single ASCII system
- **Unicode**: character encoding system that includes IPA phonetic symbols.

Here is a chart classifying sounds using the IPA:

THE INTERNATIONAL PHONETIC ALPHABET (revised to 1993)

CONSONANTS (PULMONIC)

| | Bilabial | Labiodental | Dental | Alveolar | Postalveolar | Retroflex | Palatal | Velar | Uvular | Pharyngeal | Glottal |
|---------|----------|-------------|--------|----------|--------------|-----------|---------|-------|--------|------------|---------|
| Plosive | p b | | | t d | | ʈ ɖ | c ɟ | k ɡ | q ɢ | | ʔ |

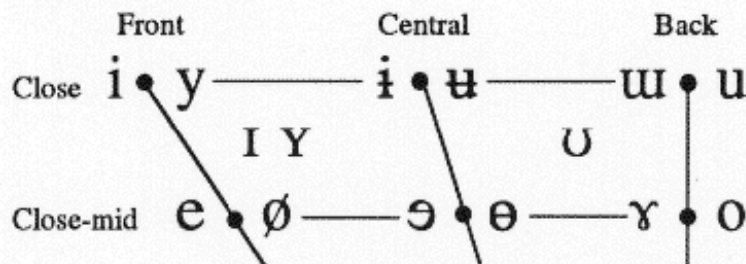
| | | | | | | | | | | | |
|---------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| | p b | t d | c ɟ | | | k ɡ | q ɢ | | | | |
| Nasal | m | ɱ | n | | | ɳ | ɲ | ŋ | N | | |
| Trill | B | | r | | | | | | R | | |
| Tap or Flap | | | ɾ | | | ɽ | | | | | |
| Fricative | ɸ β | f v | θ ð | s z | ʃ ʒ | ʂ ʐ | ç ʝ | x ɣ | χ ʁ | ħ ʕ | h ɦ |
| Lateral fricative | | | ɬ ɮ | | | | | | | | |
| Approximant | | ʋ | ɹ | | | ɻ | j | ɰ | | | |
| Lateral approximant | | | l | | | ɭ | ʎ | L | | | |

Where symbols appear in pairs, the one to the right represents a voiced consonant. Shaded areas denote articulations judged impossible.

CONSONANTS (NON-PULMONIC)

| Clicks | Voiced implosives | Ejectives |
|------------------|-------------------|-----------------------|
| ⦿ Bilabial | ɓ Bilabial | ʼ as in: |
| Dental | ɗ Dental/alveolar | pʼ Bilabial |
| ! (Post)alveolar | ɟ Palatal | tʼ Dental/alveolar |
| ≠ Palatoalveolar | ɡ Velar | kʼ Velar |
| Alveolar lateral | ɠ Uvular | sʼ Alveolar fricative |

VOWELS



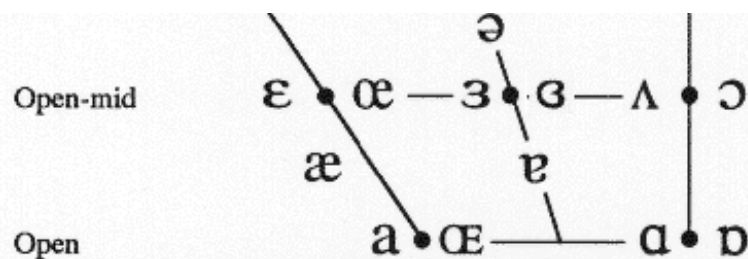
SUPRASEGMENTALS

| | TONES & WORD ACCENTS |
|--------------------------------|-------------------------|
| ' Primary stress | LEVEL |
| ˈ Secondary stress | CONTOUR |
| ː Long | ẽ or ˩ Extra high |
| ˑ Half-long | é ˨ High |
| ˑ Extra-short | ē ˨ Mid |
| ˑ Syllable break | è ˨ Low |
| Minor (foot) group | ẽ ˩ Extra low |
| Major (intonation) group | ↓ Downstep |
| ˌ Linking (absence of a break) | ↑ Upstep |
| | ẽ or ˩ Rising |
| | ê ˩ Falling |
| | ẽ ˩ High rising |
| | ẽ ˩ Low rising |
| | ẽ ˩ Rising-falling etc. |
| | ↗ Global rise |
| | ↘ Global fall |

DIACRITICS

Diacritics may be placed above a symbol with a descender, e.g. ɳ̌

| | | | | | |
|-------------|-------|--------------------|-------|----------|-------|
| ◌ Voiceless | ɳ̌ ɗ̌ | ◌ Breathy voiced | ɳ̌ ɗ̌ | ◌ Dental | ɳ̌ ɗ̌ |
| ◌ Voiced | š ť | ◌ Breathily voiced | ȟ ǎ | ◌ Dental | ť ď |



Where symbols appear in pairs, the one to the right represents a rounded vowel.

OTHER SYMBOLS

ʍ Voiceless labial-velar fricative

W Voiced labial-velar approximant

ɥ Voiced labial-palatal approximant

ħ Voiceless epiglottal fricative

ʕ Voiced epiglottal fricative

ʡ Epiglottal plosive

ç ʝ Alveolo-palatal fricatives

ɺ Alveolar lateral flap

ɧ Simultaneous ʃ and X

Affricates and double articulations can be represented by two symbols joined by a tie bar if necessary.

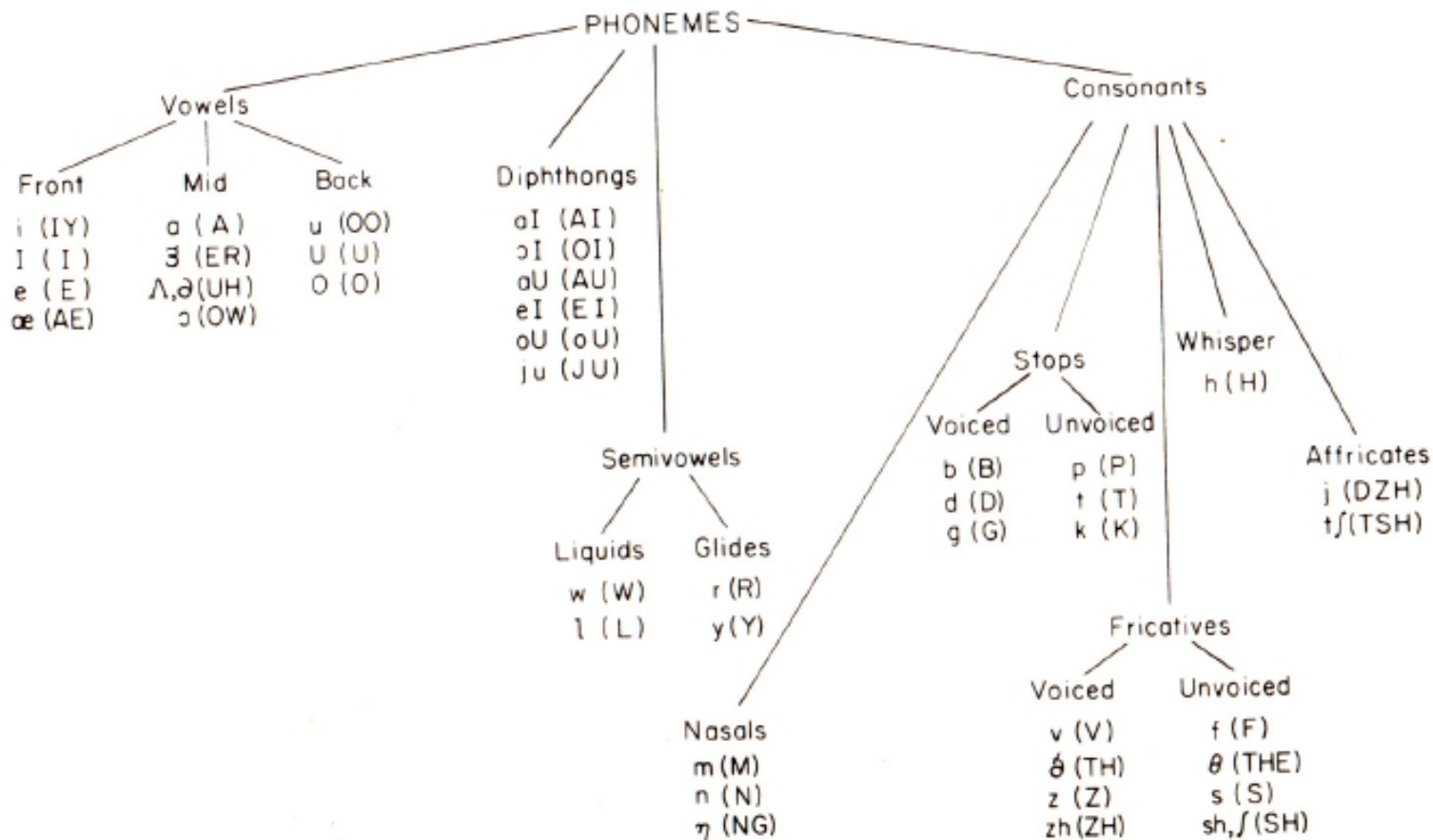
k͡p t͡s

| ✓ Voiced | ✓ | ✓ | ~ Creaky voiced | ~ | ~ | ◌ Apical | ◌ | ◌ |
|-------------------|----------------|----------------|-------------------------------|----------------|------------------------------------|----------------------|----|----------------|
| h Aspirated | t ^h | d ^h | | | | | | |
| | | | ~ Linguolabial | t̠ | d̠ | ◌ Laminal | t̠ | d̠ |
| ◌ More rounded | ɔ̹ | | ◌ Labialized | t ^w | d ^w | ~ Nasalized | | ẽ |
| ◌ Less rounded | ɔ̜ | | j Palatalized | t ^j | d ^j | ◌ Nasal release | | d ⁿ |
| ◌ Advanced | u̟ | | Y Velarized | t ^Y | d ^Y | ◌ Lateral release | | d ^l |
| ◌ Retracted | i̠ | | ◌ Pharyngealized | t ^ʕ | d ^ʕ | ◌ No audible release | | d ^ʔ |
| ◌ Centralized | ẽ | | ~ Velarized or pharyngealized | | ɟ | | | |
| ◌ Mid-centralized | ẽ | | ◌ Raised | e̟ | (ɹ̠ = voiced alveolar fricative) | | | |
| ◌ Syllabic | ɹ̩ | | ◌ Lowered | e̞ | (β̞ = voiced bilabial approximant) | | | |
| ◌ Non-syllabic | e̥ | | ◌ Advanced Tongue Root | | e̟ | | | |
| ◌ Rhoticity | ə̃ | | ◌ Retracted Tongue Root | | e̠ | | | |

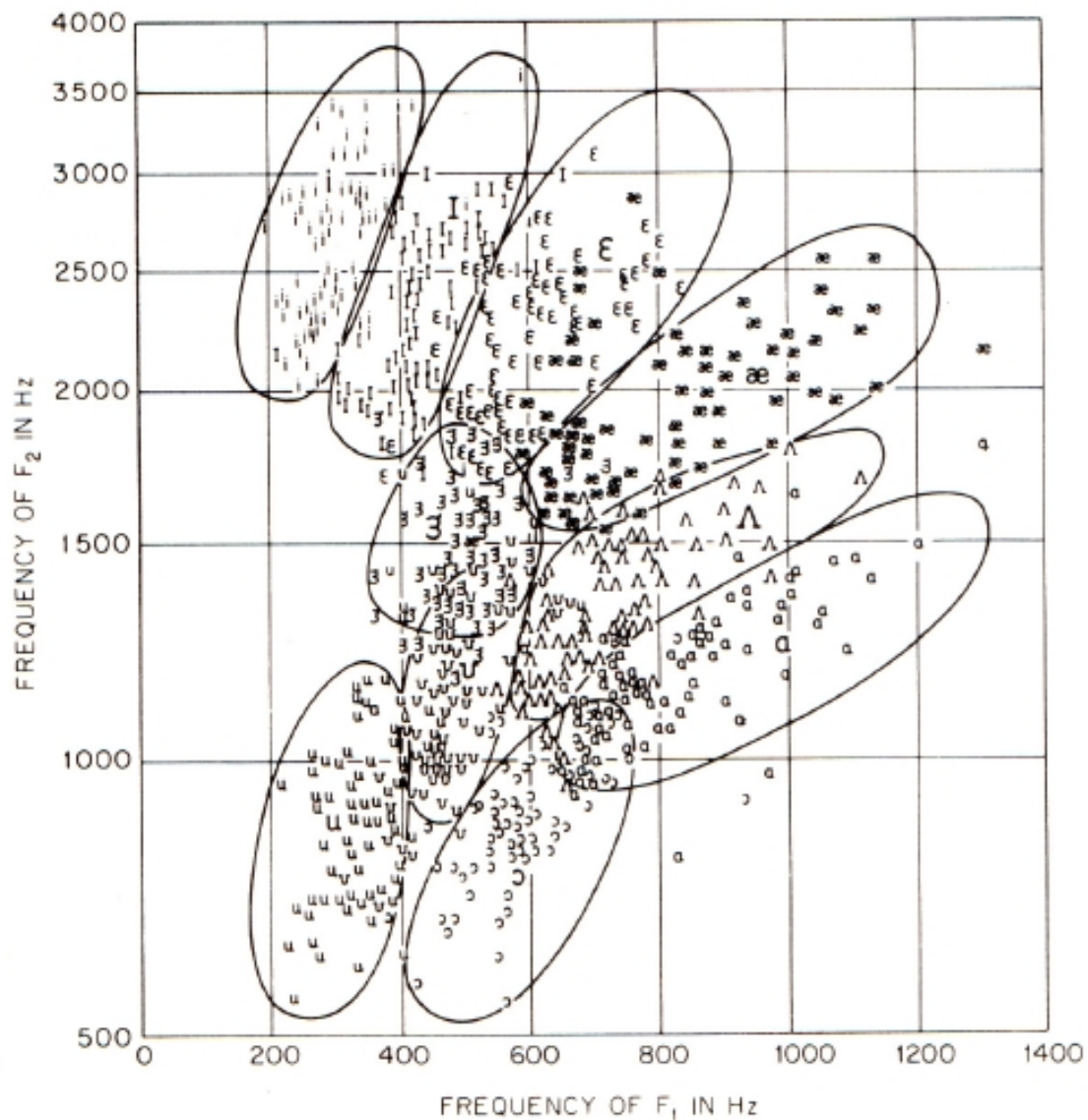
For a more detailed discussion of phone mappings across languages, see [language independent acoustic modeling](http://www.isip.msstate.edu/~gao/net/2002_spring/lecture_06/lecture_06_03.html).

THE VOWEL SPACE

- Each fundamental speech sound can be categorized according to the position of the articulators. This is often known as the study of Acoustic Phonetics.

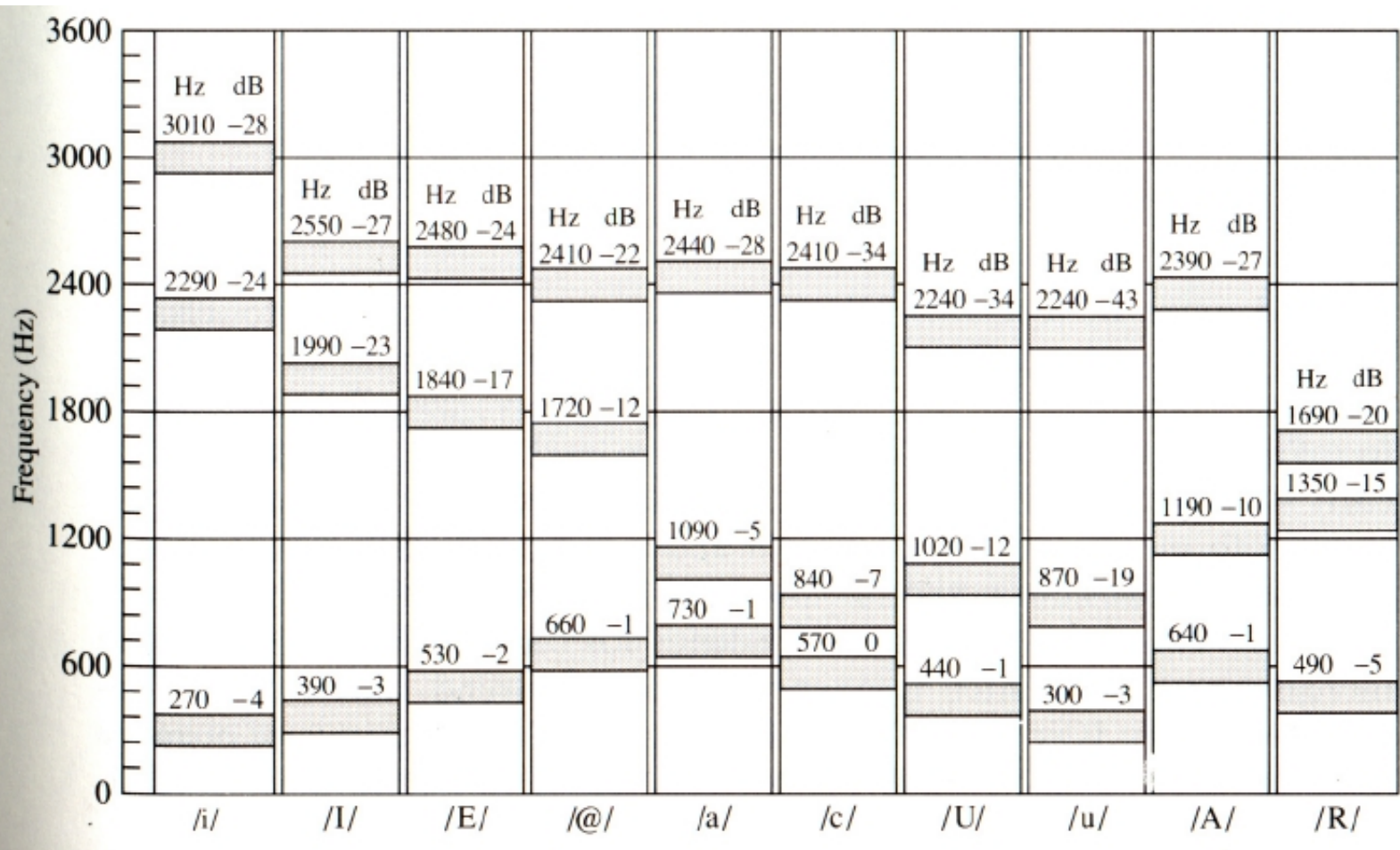


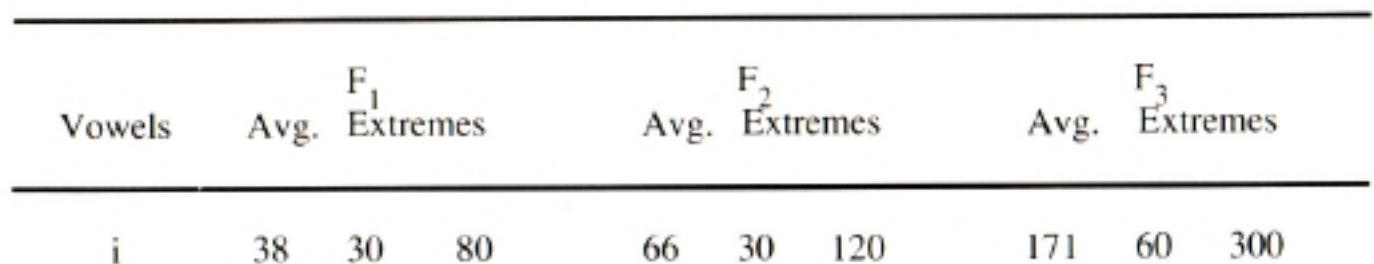
- We can characterize a vowel sound by the locations of the first and second spectral resonances, known as formant frequencies:



- Some voiced sounds, such as diphthongs, are transitional sounds that move from one vowel location to another.

THE RANGE OF FORMANT FREQUENCIES





| | | | | | | | | | |
|------|------|----|-----|------|----|-----|-------|----|-----|
| I | 42 | 30 | 100 | 71 | 40 | 120 | 142 | 60 | 300 |
| E | 42 | 30 | 120 | 72 | 30 | 140 | 126 | 50 | 300 |
| @ | 65 | 30 | 140 | 90 | 40 | 200 | 156 | 50 | 300 |
| a | 60 | 30 | 160 | 50 | 30 | 80 | 102 | 40 | 300 |
| c | 47 | 30 | 120 | 50 | 30 | 200 | 98 | 40 | 240 |
| u | 50 | 30 | 120 | 58 | 30 | 200 | 107 | 50 | 200 |
| U | 51 | 30 | 100 | 61 | 30 | 140 | 90 | 40 | 200 |
| A | 56 | 30 | 140 | 63 | 30 | 140 | 102 | 50 | 300 |
| R | 46 | 30 | 80 | 59 | 30 | 120 | 58 | 40 | 120 |
| Avg. | 49.7 | | | 64.0 | | | 115.2 | | |

AN ACOUSTIC THEORY FOR VOWEL PRODUCTION

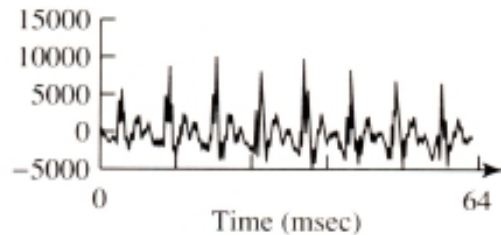
(a)



i (eve)

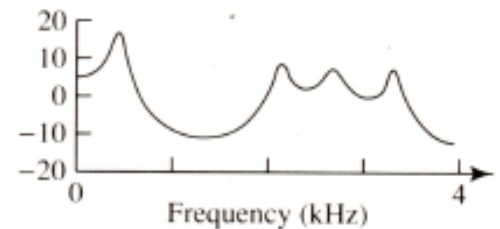
(b)

ALL OCCURENCES



(c)

ALL OCCURENCES



Time (msec)

Frequency (kHz)

Time (msec)

Frequency (kHz)

Time (msec)

Frequency (kHz)

Time (msec)

Frequency (kHz)

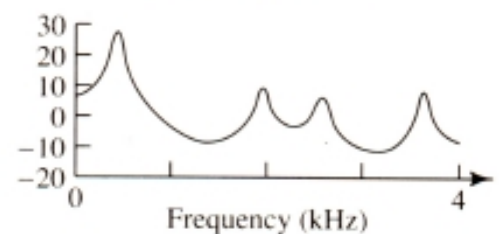
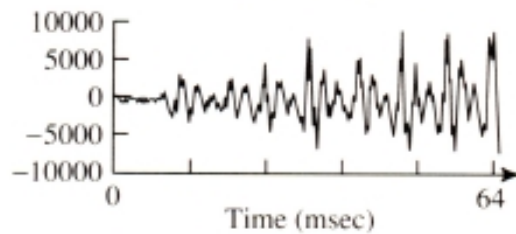
Time (msec)

Frequency (kHz)

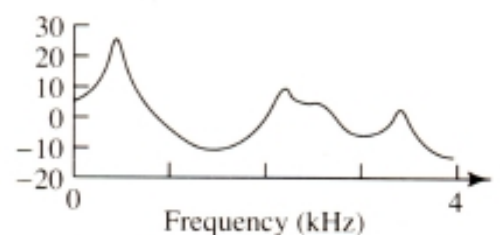
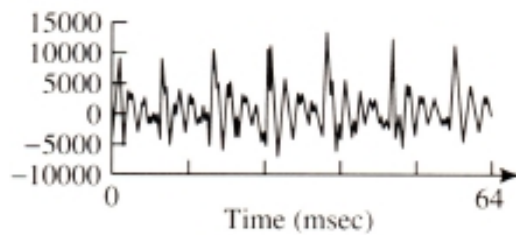
Time (msec)

Frequency (kHz)

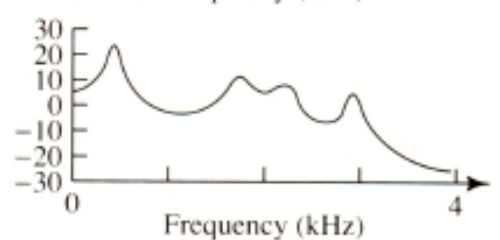
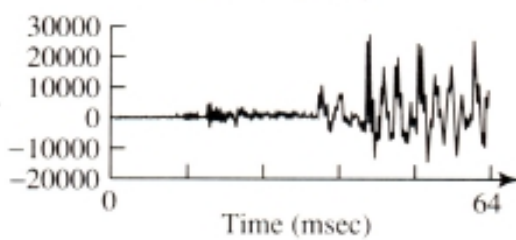
I (it)



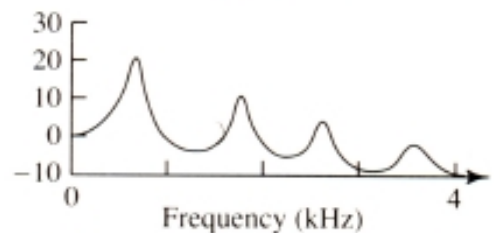
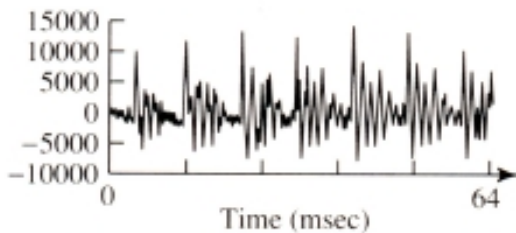
e (hate)



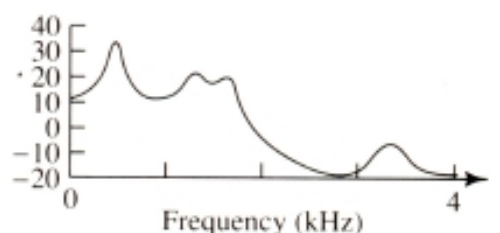
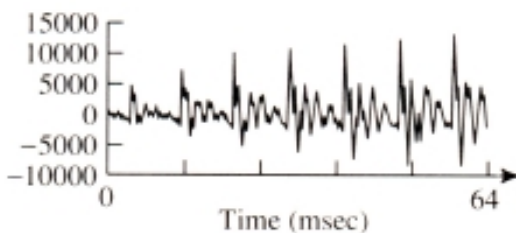
E (met)



@ (at)



R (bird)

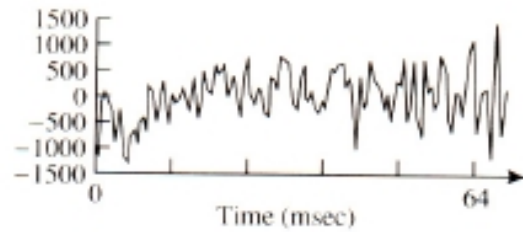


THIS THEORY IS ALSO APPLICABLE TO CONSONANTS

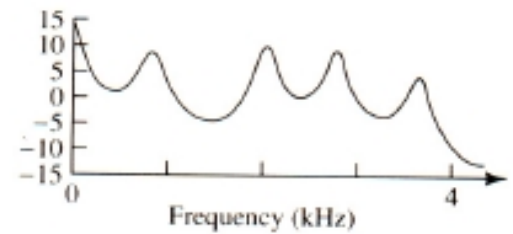
(a)



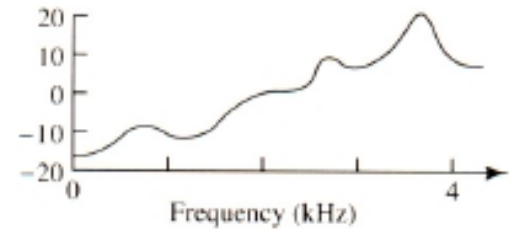
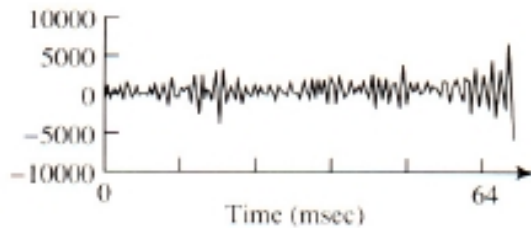
(b)



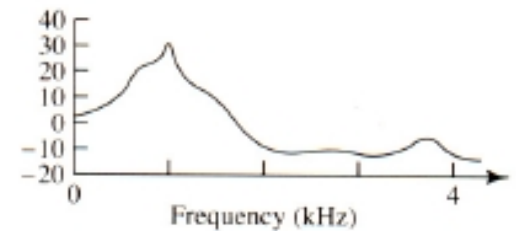
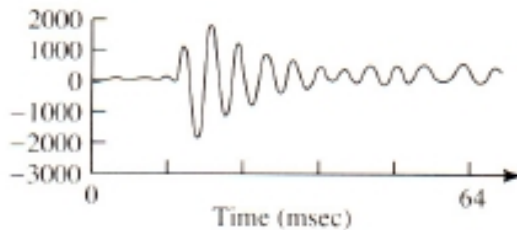
(c)



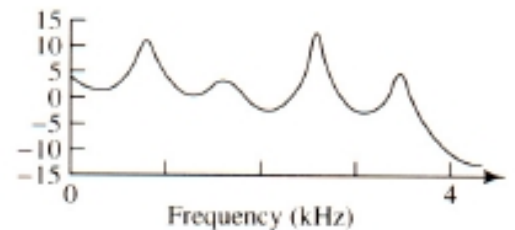
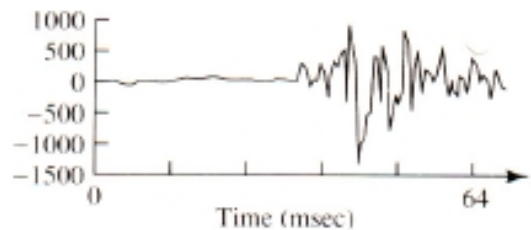
t (to)



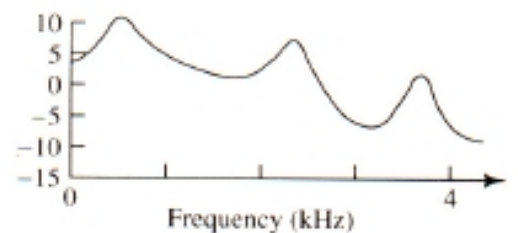
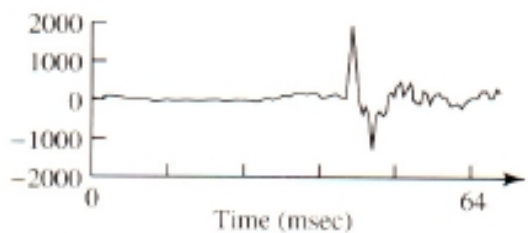
k (key)

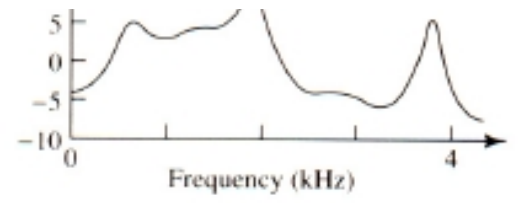
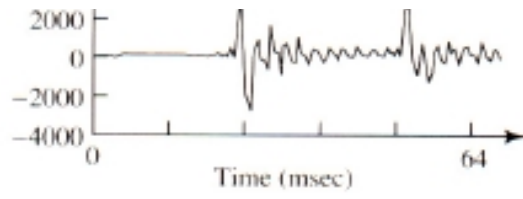
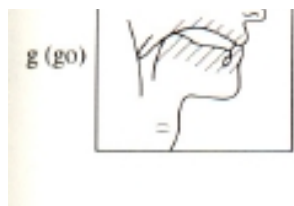


b (be)



d (day)





[Return to Main](#)[Objectives](#)**Words:**[Syllables](#)[Words](#)[Lexical Part of Speech](#)[Morphology](#)[Word Classes](#)**Syntax and Semantics:**[Phrase Schemata](#)[Clauses and Sentences](#)[Parse Trees](#)[Semantic Roles](#)[Lexical Semantics](#)[Logical Form](#)**Summary:**[Integration](#)[Word Prediction](#)**On-Line Resources:**[WordNet](#)[Switchboard](#)[Linguistic Data Consortium](#)

LECTURE 07: SYNTAX AND SEMANTICS

● Objectives:

- Understand the role of [higher level knowledge](#) in speech recognition
- Introduce how we can exploit knowledge about the structure of language to improve speech recognition performance
- Gain an appreciation for the fields of linguistics and natural language processing
- Introduce alternate choices for acoustic units

Note that this lecture is primarily based on material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

A good reference textbook on these topics is:

D. Jurafsky and J.H. Martin, *SPEECH and LANGUAGE PROCESSING: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Prentice-Hall, ISBN: 0-13-095069-6, 2000.

LECTURE 07: SYNTAX AND SEMANTICS

● Objectives:

- Understand the role of higher level knowledge in speech recognition
- Introduce how we can exploit knowledge about the structure of language to improve speech recognition performance
- Gain an appreciation for the fields of linguistics and natural language processing
- Introduce alternate choices for acoustic units

Note that this lecture is primarily based on material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

A good reference textbook on these topics is:

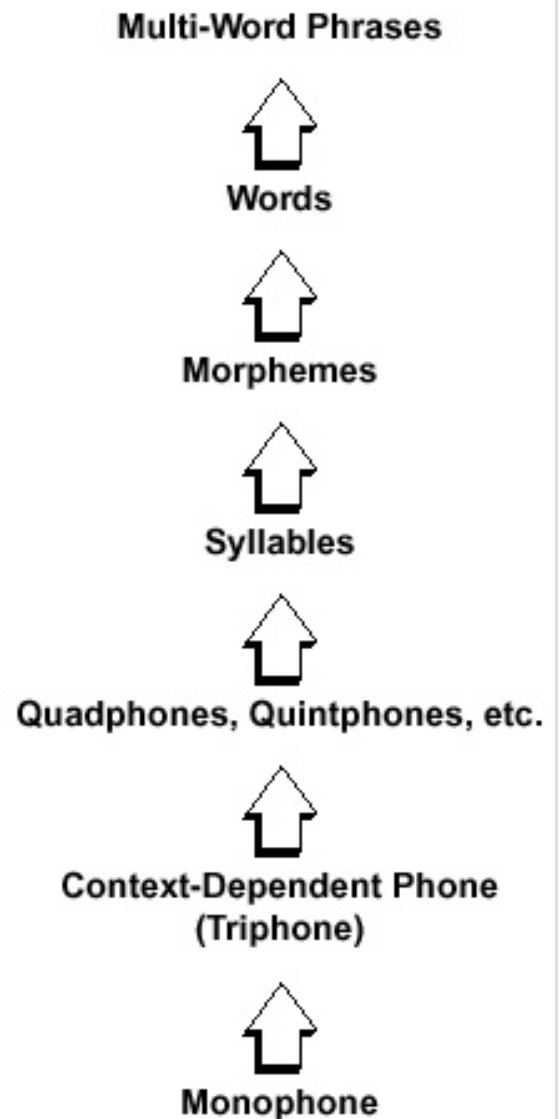
D. Jurafsky and J.H. Martin, *SPEECH and LANGUAGE PROCESSING: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Prentice-Hall, ISBN: 0-13-095069-6, 2000.

SYLLABLES: PRIMARY DOMAIN OF COARTICULATION?

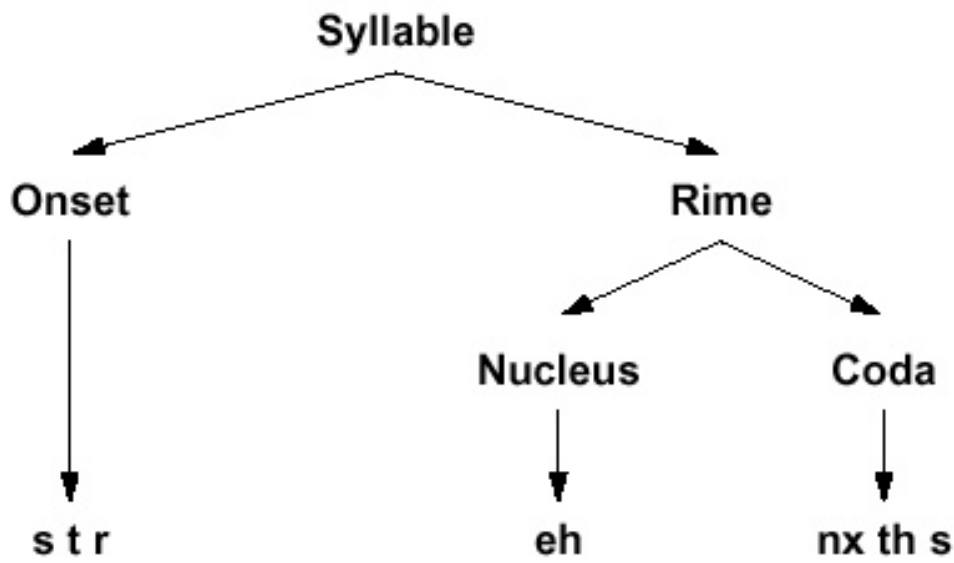
- Acoustically distinct.
- There are over 10,000 syllables in English.
- There is no universal definition of a syllable.
- Can be defined from both a

production and
perception
viewpoint.

- Centered around vowels in English.
- Consonants often span two syllables ("ambisyllabic" - "bottle").
- Three basic parts: onset (initial consonants), nucleus (vowel),



and coda
(consonants
following the
nucleus).



WORDS: OBSERVABLE UNITS OF A LANGUAGE?

- Loosely defined as a lexical unit - there is an agreed upon meaning in a given community.
- In many languages (e.g., Indo-European), easily observed in the orthographic (writing) system since it is separated by white space.
- In spoken language, however, there is a segmentation problem: words run together.
- **Syntax:** certain facts about word structure and combinatorial possibilities are evident to most native speakers.
- **Paradigmatic:** properties related to meaning.

- **Syntagmatic**: properties related to constraints imposed by word combinations (grammar).
- Word-level constraints are the most common form of "domain knowledge" in a speech recognition system.
- **N-gram** models are the most common way to implement word-level constraints.
- [N-gram](#) distributions are very interesting!

LEXICAL PART OF SPEECH:

- **Lexicon:** alphabetic arrangement of words and their definitions. A term often used to describe the list of allowable words for a speech recognition system.
- **Lexical Part of Speech:** A restricted inventory of word-type categories which capture generalizations of word forms and distributions ("dog" and "cat" are nouns and animals).
- **Part of Speech (POS):** noun, verb, adjective, adverb, interjection, conjunction, determiner, preposition, and pronoun.
- **Proper Noun:** names such as "Velcro" or "Spandex". Pose a very challenging problem for speech recognition because of the lack of pronunciation rules (e.g., "Nyugen", "Sorbet").
- **Open POS Categories:**

| Tag | Description | Function | Example |
|-----|-------------|--------------|------------|
| N | Noun | Named entity | <i>cat</i> |

| | | | |
|--------|--------------|--------------------|----------------|
| V | Verb | Event or condition | <i>forget</i> |
| Adj | Adjective | Descriptive | <i>yellow</i> |
| Adv | Adverb | Manner of action | <i>quickly</i> |
| Interj | Interjection | Reaction | <i>Oh</i> |

- **Closed POS Categories:** some level of universal agreement on the categories (e.g, conjunction, determiner, preposition).
- **Penn Treebank:** the LDC's [Penn Treebank](#) is one of the most ambitious projects to date in which large amounts of data have been categorized.
- **Wordnet:** Princeton's [Wordnet](#) is another very important and ambitious project to develop an on-line lexical reference system.

MORPHOLOGY: IMPORTANT IN SPECIALIZED SUB-LANGUAGES

- **Morpheme:** a distinctive collection of phonemes having no smaller meaningful parts (e.g. "pin" or "s" in "pins").
- Morphemes are often words, and in some languages (e.g., Latin), are an important sub-word unit. Some specific speech applications (e.g. medical dictation) are amenable to morpheme level acoustic units.
- **Inflectional Morphology:** variations in word form that reflect the contextual situation of a word, but do not change the fundamental meaning of the word (e.g. "cats" vs. "cat").
- **Derivational Morphology:** a given root word

may serve as the source for new words (e.g., "racial" and "racist" share the morpheme "race", but have different meanings and part of speech possibilities). The baseform of a word is often called the **root**. Roots can be compounded and concatenated with derivational prefixes to form other words.

WORD CLASSES: A STATISTICAL APPROACH

- **Word Classes:** Assign words to similar classes based on their usage in real text (clustering). Can be derived automatically using statistical parsers.
- Typically more refined than POS tags (all words in a class will share the same POS tag). Based on semantics (meaning).
- Word classes are used extensively in language model probability smoothing.
- Examples:
 - {Monday, Tuesday, ..., weekends}

- {great, big, vast, ..., gigantic}
- {down, up, left, right, ..., sideways}

PHRASE SCHEMATA

- **Syntax:** Syntax is the study of the formation of sentences from words and the **rules** for formation of grammatical sentences.
- **Syntactic Constituents:** subdivisions of a sentence into phrase-like units that are common to many sentences. Syntactic constituents explain the word order of a language ("SOV" vs. "SVO" languages).
- **Phrase Schemata:** groups of words that have internal structure and unity (e.g., a "noun phrase" consists of a noun and its immediate modifiers).
- Example: NP → (det) (modifier) **head-noun** (post-modifier)

| NP | Det | Mod | Head Noun | Post-Mod |
|-----------|------------|------------|----------------------|------------------|
| 1 | the | | authority | of government |

| | | | | |
|----|----|--------|---------|-----------------------|
| 7 | an | impure | one | |
| 16 | a | true | respect | for the individual |

CLAUSES AND SENTENCES

- A **clause** is any phrase that has both a subject (NP) and a verb phrase (VP) that has a potentially independent interpretation.
- A **sentence** is a superset of a *clause* and can contain one or more clauses.
- Some typical types of sentences:

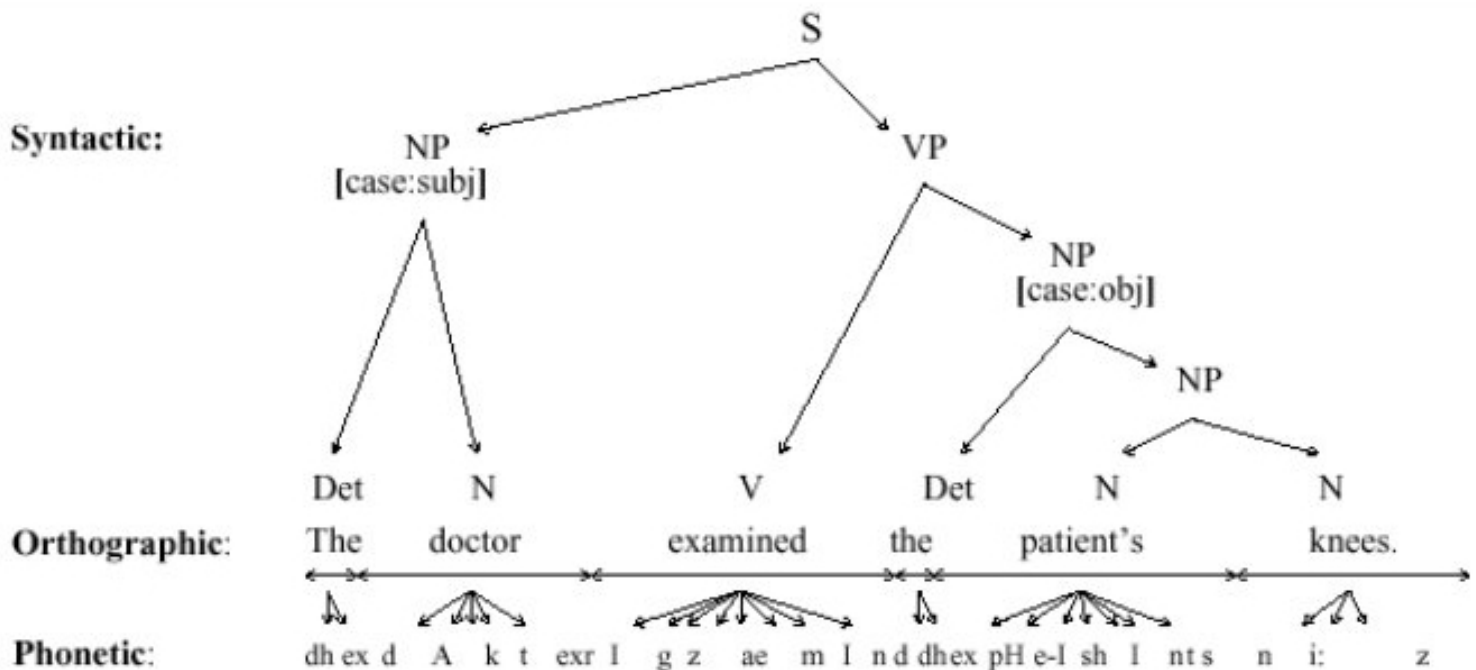
| Type | Example |
|-------------|--------------------|
| Declarative | I gave her a book. |

| | |
|-------------------------|--|
| Yes-No Question | Did you give her a book? |
| Wh-Question | What did you give her? |
| Alternative Question | Did you give her a book or a knife? |
| Tag Question | You gave it to her, didn't you? |

| | |
|-------------|--|
| Passive | She was given a book. |
| Cleft | It must have been a book that she got. |
| Exclamative | Hasn't this been a great birthday! |
| Imperative | Give me the book. |

PARSE TREES

- **Parse Tree:** used to represent the structure of a sentence and the relationship between its constituents.
- Markup languages such as the standard generalized markup language (**SGML**) are often used to represent a parse tree in a textual form.
- Example:



SEMANTIC ROLES

- **Grammatical roles** are often used to describe the direction of action (e.g., subject, object, indirect object).
- **Semantic roles**, also known as **case relations**, are used to make sense of the participants in an event (e.g., "who did what to whom").

| Role | Description |
|---------------|--------------------------------|
| Agent | cause or inhibitor of action |
| Patient/Theme | undergoer of the action |
| Instrument | how the action is accomplished |
| Goal | to whom the action is directed |

| | |
|----------|---------------------------------|
| Result | result or outcome of the action |
| Location | location or place of the action |

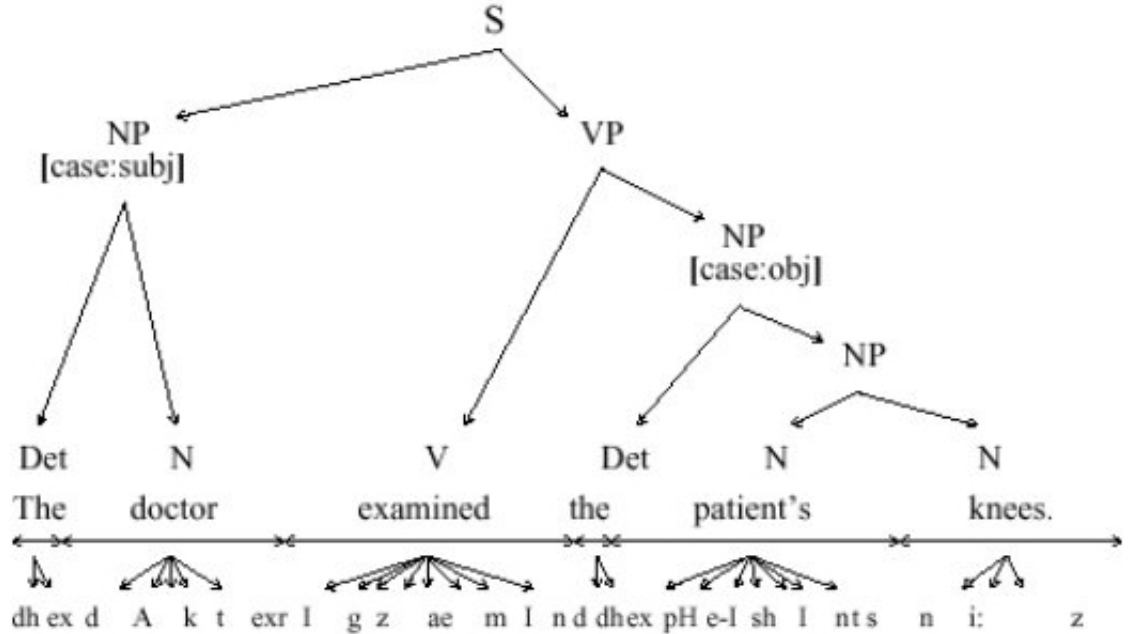
● Example:

- "The doctor examined the patient's knees"

Syntactic:

Orthographic:

Phonetic:



LEXICAL SEMANTICS

- **Lexical Semantics:** the semantic structure associated with a word, as represented in the lexicon.
- **Taxonomy:** orderly classification of words according to their presumed natural relationships.
- Examples:
 - **Is-A Taxonomy:** a *crow* is a bird.
 - **Has-a Taxonomy:** a *car* has a windshield.
 - **Action-Instrument:** a *knife* can cut.

- Words can appear in many relations and have multiple meanings and uses.
- There are no universally-accepted taxonomies:

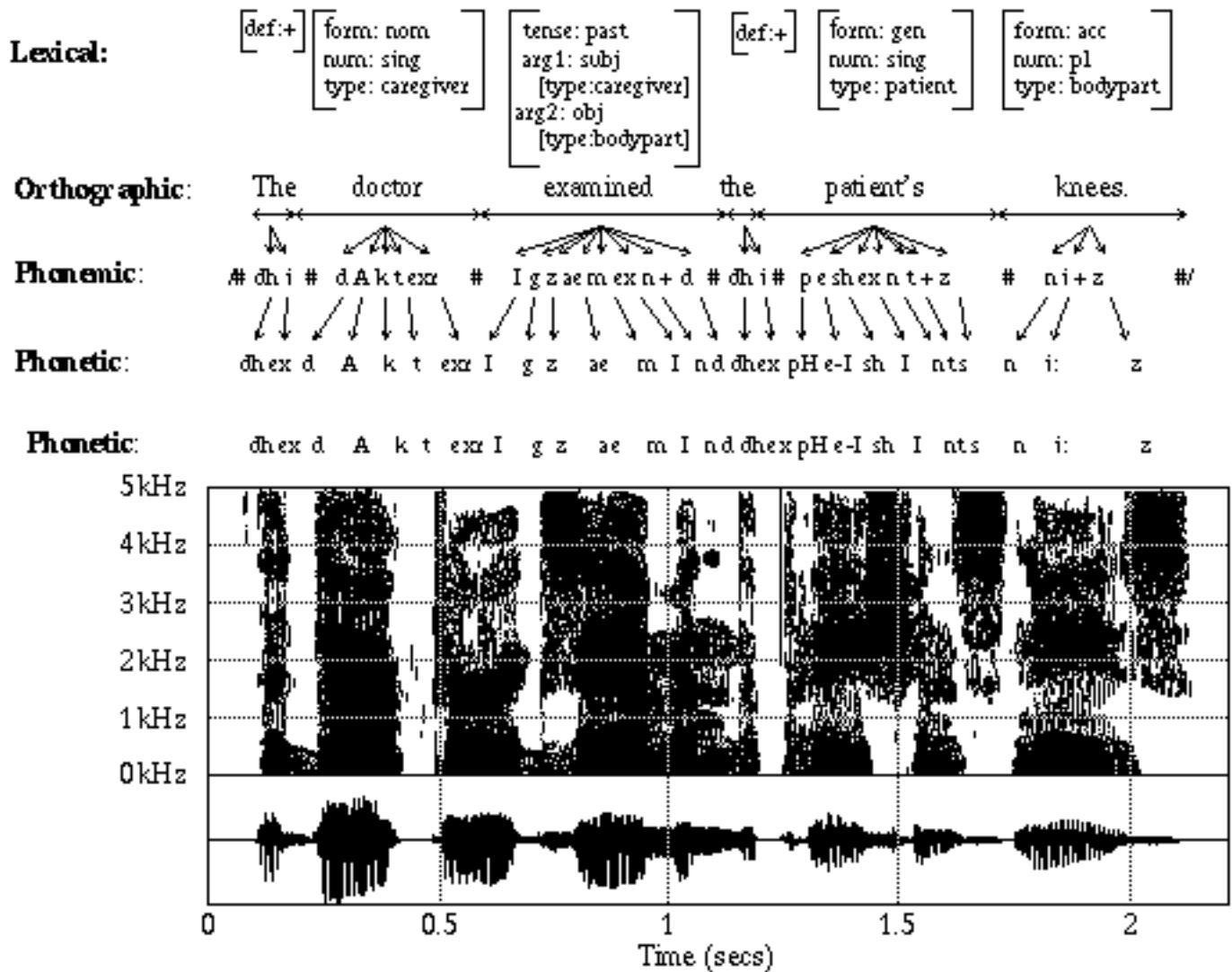
| Family | Subtype | Example |
|-----------|------------------------|--------------|
| Contrasts | Contrary | old-young |
| | Contradictory | alive-dead |
| | Reverse | buy-sell |
| | Directional | front-back |
| | Incompatible | happy-morbid |
| | Asymmetric contrary | hot-cool |

| | | |
|-------------------|-------------------|----------------|
| Case Relations | Attribute similar | rake-fork |
| | Agent-action | artist-paint |
| | Agent-instrument | farmer-tractor |
| | Agent-object | baker-bread |
| | Action-recipient | sit-chair |
| | Action-instrument | cut-knife |

LOGICAL FORMS

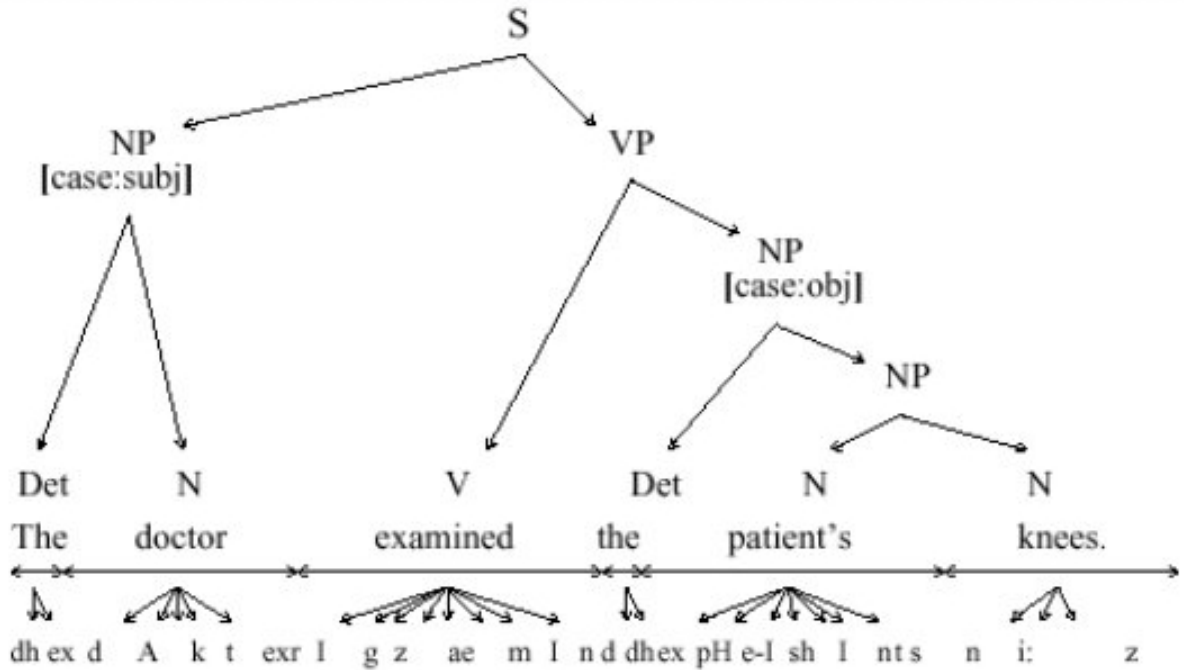
- **Logical form:** a metalanguage in which we can concretely and succinctly express all linguistically possible meanings of an utterance.
- Typically used as a representation to which we can apply discourse and world knowledge to select the single-best (or N-best) alternatives.
- An attempt to bring formal logic to bear on the language understanding problem (**predicate logic**).
- Example:

- If Romeo is happy, Juliet is happy:
Happy(Romeo) -> Happy(Juliet)
- "The doctor examined the patient's knees"



INTEGRATION OF SPEECH AND NATURAL LANGUAGE

Syntactic:

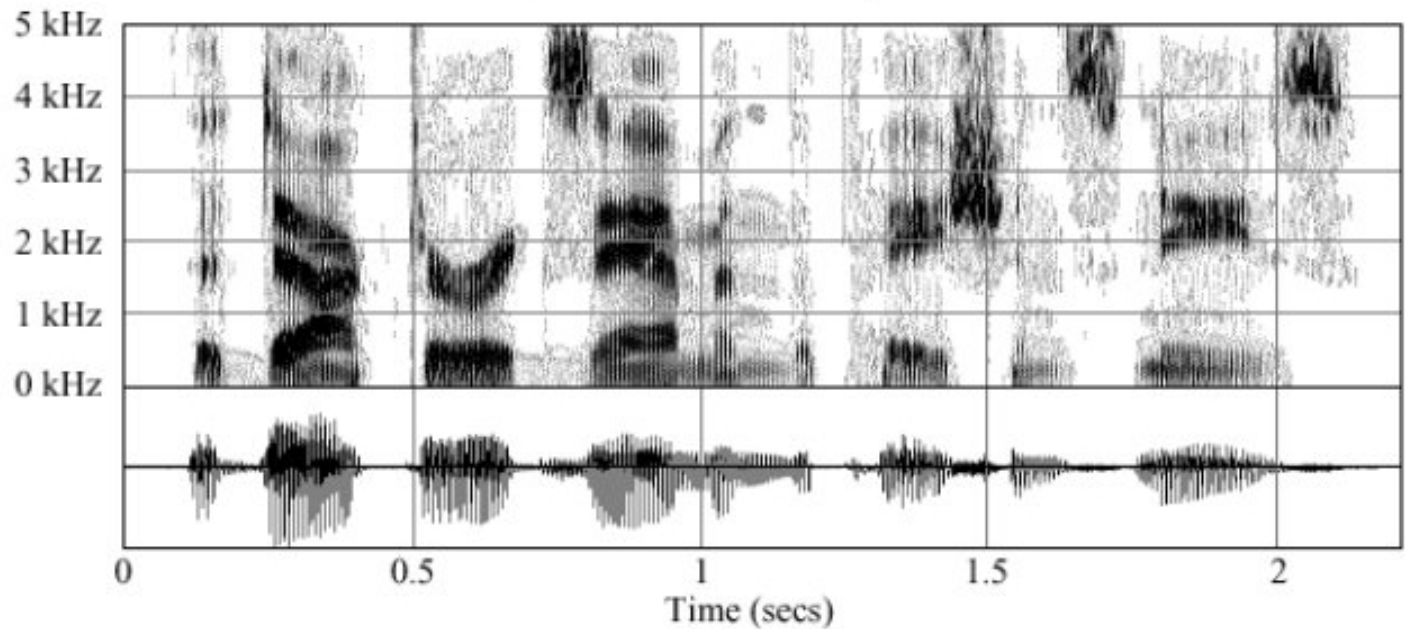


Orthographic:

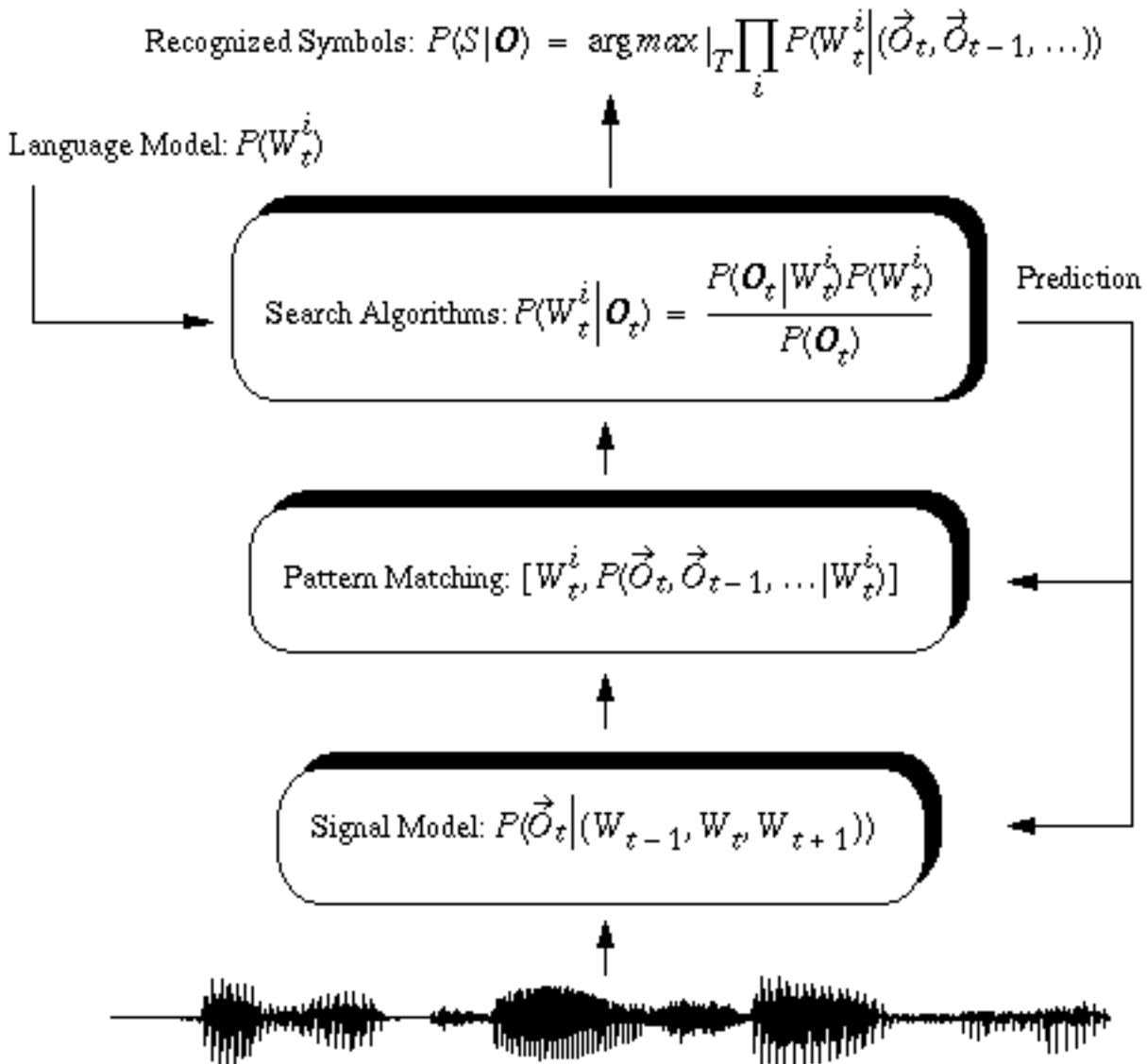
The doctor examined the patient's knees.

Phonetic:

dh ex d A k t ex r l g z ae m l n d dh ex pH e-l sh l nt s n i: z



WORD PREDICTION PLAYS A KEY ROLE



[Return to Main](#)[Objectives](#)**Overview:**[Components](#)[Typical Front End](#)**Sampling:**[Theorem](#)[Derivation](#)[Graphical](#)[Reconstruction](#)[Bandlimited](#)[Aliasing](#)[Overlapping Frames](#)[Conditioning](#)**On-Line Resources:**[Signal Modeling](#)[Applet](#)[Theorem](#)

LECTURE 08: SAMPLING

● Objectives:

- Introduce a typical front end
- Understand sampling issues
- Understand the impact of aliasing
- Appreciate the need for signal preprocessing
- Understand frame-based processing

A good reference textbook on these topics is:

J.G. Proakis and D.G. Manolakis,
*Digital Signal Processing:
Principles, Algorithms, and
Applications*, Prentice Hall,
Upper Saddle River, New Jersey,
USA, ISBN: 0-13-373762-4,
1996 (third edition).

LECTURE 08: SAMPLING

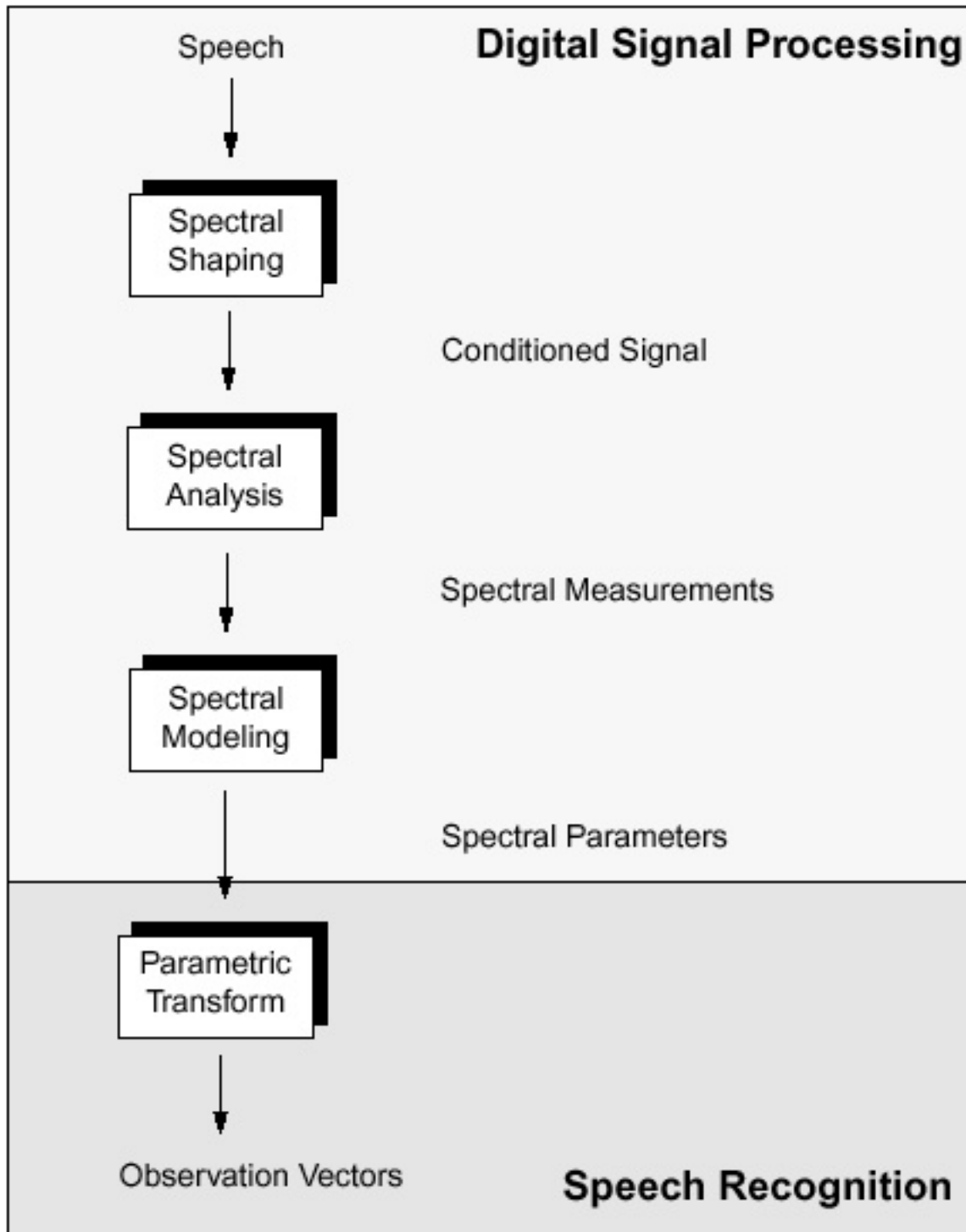
- Objectives:
 - Introduce a typical front end
 - Understand sampling issues
 - Understand the impact of aliasing
 - Appreciate the need for signal preprocessing
 - Understand frame-based processing

A good reference textbook on these topics is:

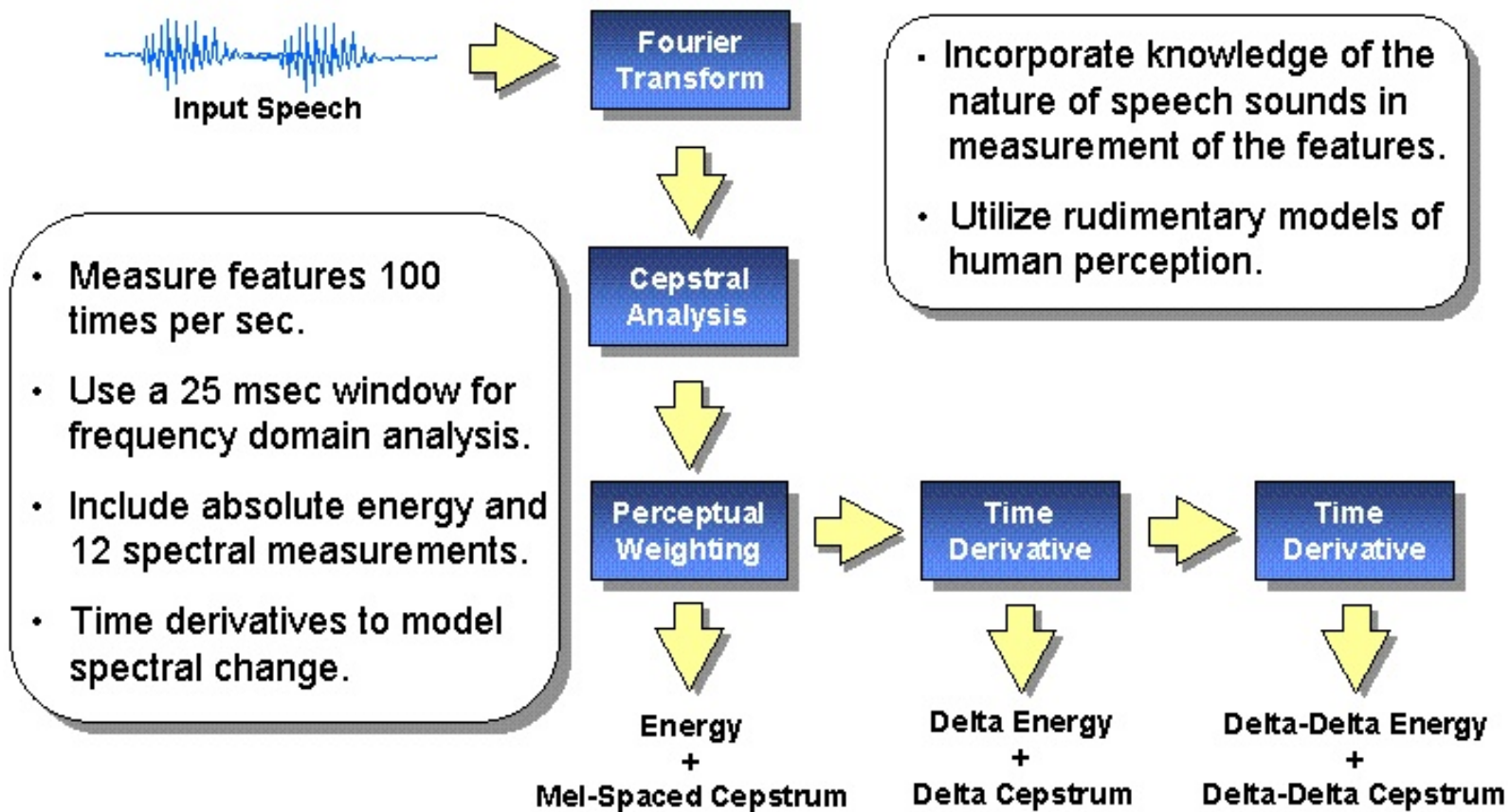
J.G. Proakis and D.G. Manolakis, *Digital Signal Processing: Principles, Algorithms,*

and Applications, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-373762-4, 1996 (third edition).

SIGNAL PROCESSING COMPONENTS IN SPEECH RECOGNITION



A TYPICAL SPEECH RECOGNITION FRONT END



THE SAMPLING THEOREM

Theorem: If the highest frequency contained in an analog signal $x_a(t)$ is $F_{\max} = B$, and the signal is sampled at a frequency $F_s > 2B$, then the analog signal can be *exactly* recovered from its samples using the following reconstruction formula:

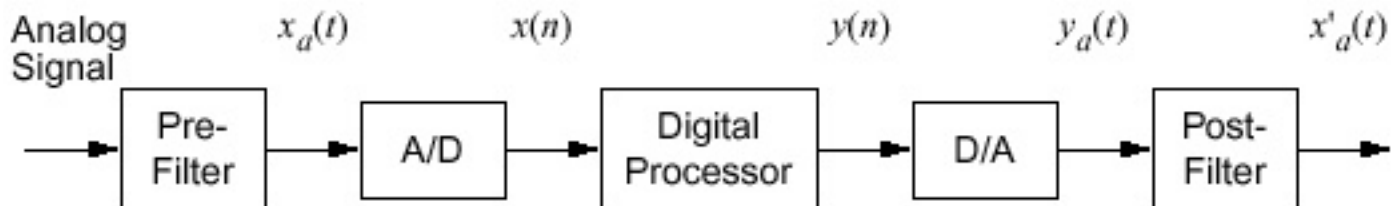
$$x_a(t) = \sum_{n=-\infty}^{\infty} x_a(nT) \frac{\sin((\pi/T)(t-nT))}{(\pi/T)(t-nT)}$$

Note that at the original sample instances ($t = nT$), the reconstructed analog signal is equal to the value of the original analog signal because the sinc functions take on values of zero at multiples of the sample period. At times between the sample instances, the signal is the weighted sum of shifted sinc functions.

DERIVATION OF THE SAMPLING THEOREM

Recall a discrete-time signal is given by:

$$x(n) = x_a(nT), \quad -\infty < n < \infty$$



If $x_a(t)$ is an aperiodic signal with finite energy, its spectrum is given by:

$$X_a(F) = \int_{-\infty}^{\infty} x_a(t) e^{-j2\pi Ft} dt$$

The signal can be recovered from the inverse Fourier transform:

$$x_a(t) = \int_{-\infty}^{\infty} X_a(F) e^{j2\pi Ft} dF$$

The spectrum of the discrete-time signal is given by:

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n) e^{-j\omega n}$$

or, equivalently,

$$X(f) = \sum_{n=-\infty}^{\infty} x(n) e^{-j2\pi fn}$$

The signal can be recovered from its spectrum:

$$\begin{aligned} x(n) &= \frac{1}{2\pi} \int_{-\pi}^{\pi} X(\omega) e^{j\omega n} d\omega \\ &= \int_{-1/2}^{1/2} X(f) e^{j2\pi fn} df \end{aligned}$$

Recall that $t = nT = \frac{n}{F_s}$. This allows us to write the inverse transform as:

$$x(n) \equiv x_a(nT) = \int_{-\infty}^{\infty} X_a(F) e^{j2\pi n(F/F_s)} dF$$

From this, we can conclude that

$$\int_{-1/2}^{1/2} X(f) e^{j2\pi f n} df = \int_{-\infty}^{\infty} X_a(F) e^{j2\pi n(F/F_s)} dF$$

We know that $f = \frac{F}{F_s}$. We can make a change of variables and write:

$$\frac{1}{F_s} \int_{-F_s/2}^{F_s/2} X\left(\frac{F}{F_s}\right) e^{j2\pi n(F/F_s)} dF = \int_{-\infty}^{\infty} X_a(F) e^{j2\pi n(F/F_s)} dF$$

We can express the integral on the right as a sum of integrals:

$$\int_{-\infty}^{\infty} X_a(F) e^{j2\pi F/F_s} dF = \sum_{k=-\infty}^{\infty} \int_{(k-1/2)F_s}^{(k+1/2)F_s} X_a(F) e^{j2\pi n(F/F_s)} dF$$

By interchanging the order of integration and summation, and invoking the periodicity of the complex exponential, we can write:

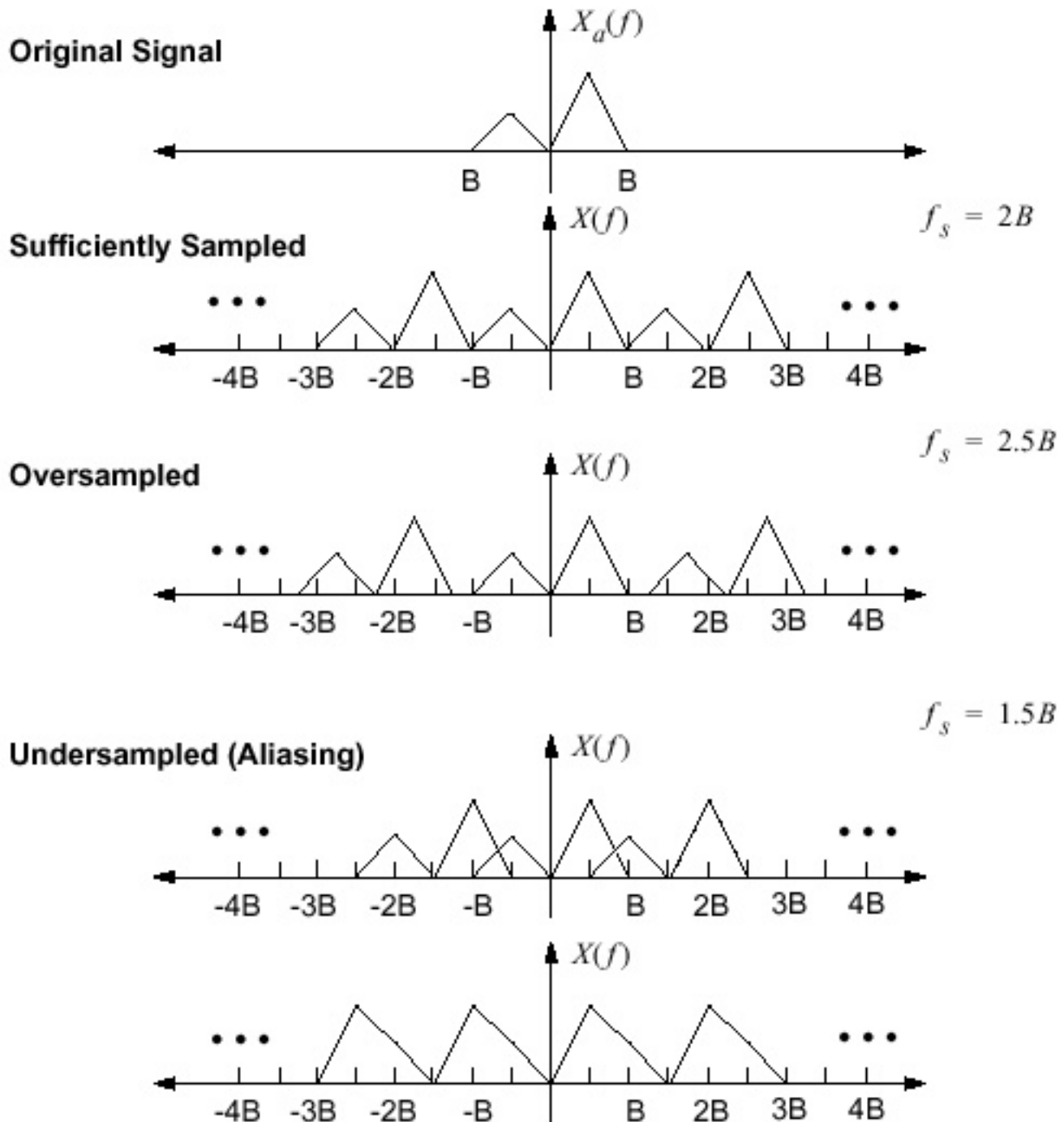
$$\frac{1}{F_s} \int_{-F_s/2}^{F_s/2} X\left(\frac{F}{F_s}\right) e^{j2\pi n(F/F_s)} dF = \int_{-F_s/2}^{F_s/2} \left[\sum_{k=-\infty}^{\infty} X_a(F - kF_s) \right] e^{j2\pi n(F/F_s)} dF$$

By equating terms inside the integral, we have:

$$X\left(\frac{F}{F_s}\right) = F_s \sum_{k=-\infty}^{\infty} X_a(F - kF_s)$$

What does this imply about the spectrum of the sampled signal?

GRAPHICAL INTERPRETATION OF THE SAMPLING THEOREM

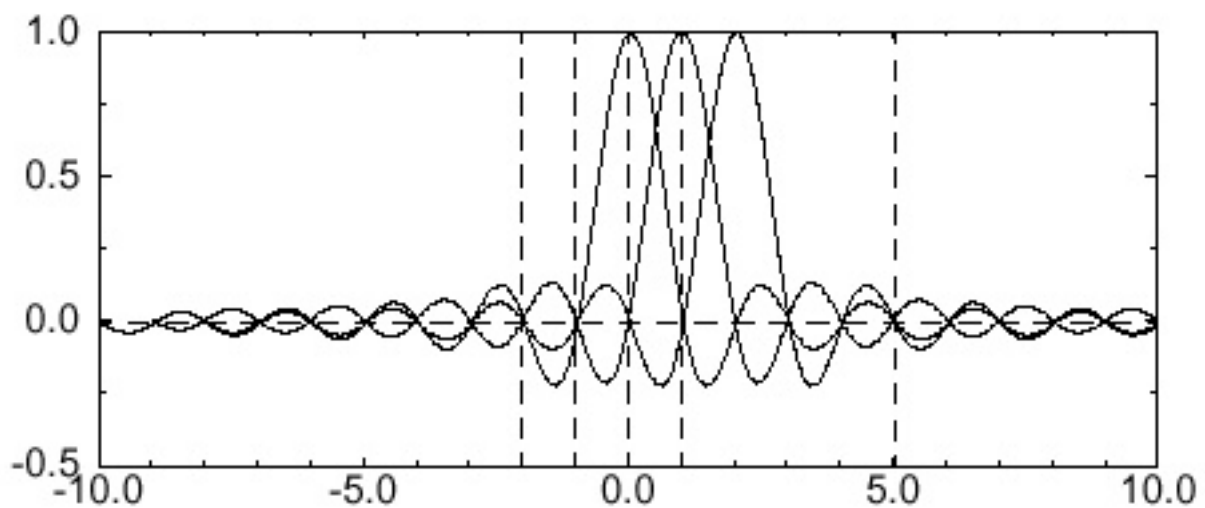
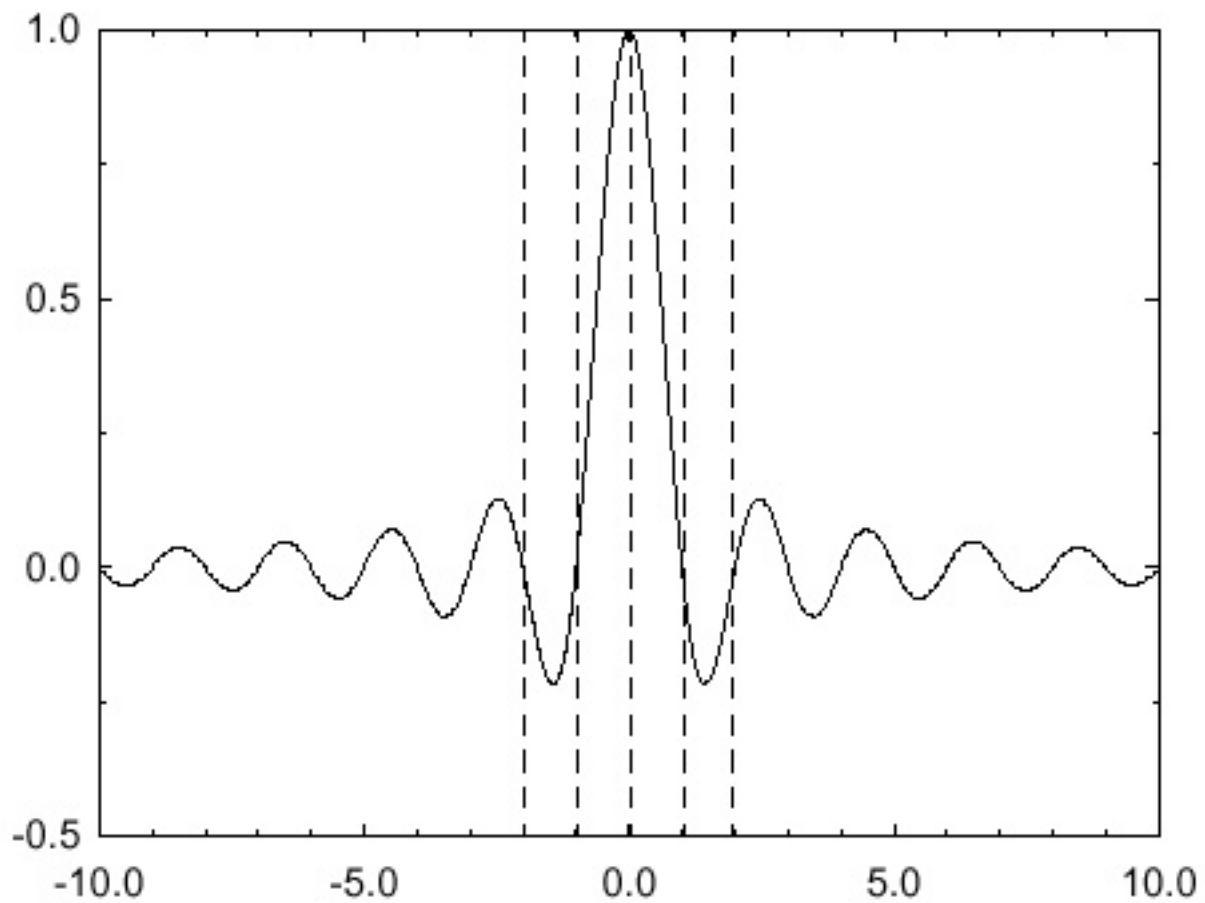


RECONSTRUCTION VIA SINC(X) INTERPOLATION

Recall our equation for reconstruction:

$$x_a(t) = \sum_{n=-\infty}^{\infty} x_a(nT) \frac{\sin((\pi/T)(t-nT))}{(\pi/T)(t-nT)}$$

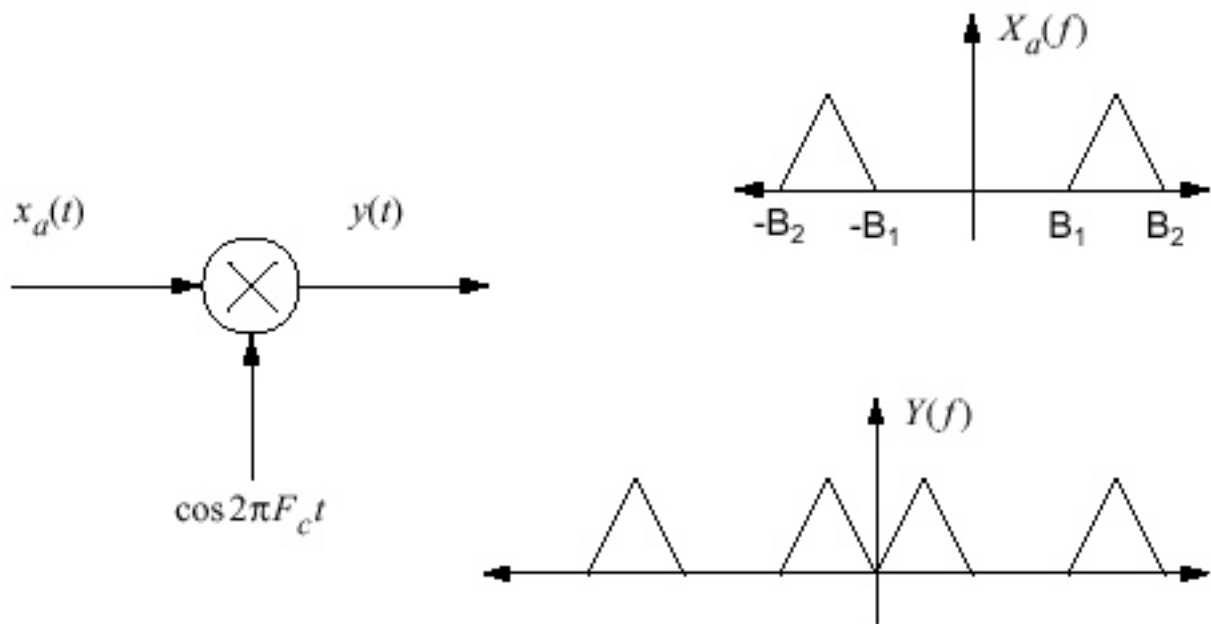
This can be viewed as an interpolation process using shifted and delayed Sinc(x) functions:



Note that these Sinc functions are exactly zero at the original sample instances.

AN INTUITIVE EXPLANATION OF THE SAMPLING THEOREM FOR BANDLIMITED SIGNALS

Consider the following system:



We can sample a bandpass signal at a frequency lower than its “Nyquist rate” by converting it to a lowpass signal.

In general, we suspect we can directly sample the signal, but we to select a sample frequency such that folding does not cause aliasing.

A general guideline is:

$$2B \leq F_s \leq 4B$$

A more rigorous equation is:

$$F_s = 2B \frac{r^s}{r}$$

where

$$r^s = \frac{F_c + B/2}{B}$$

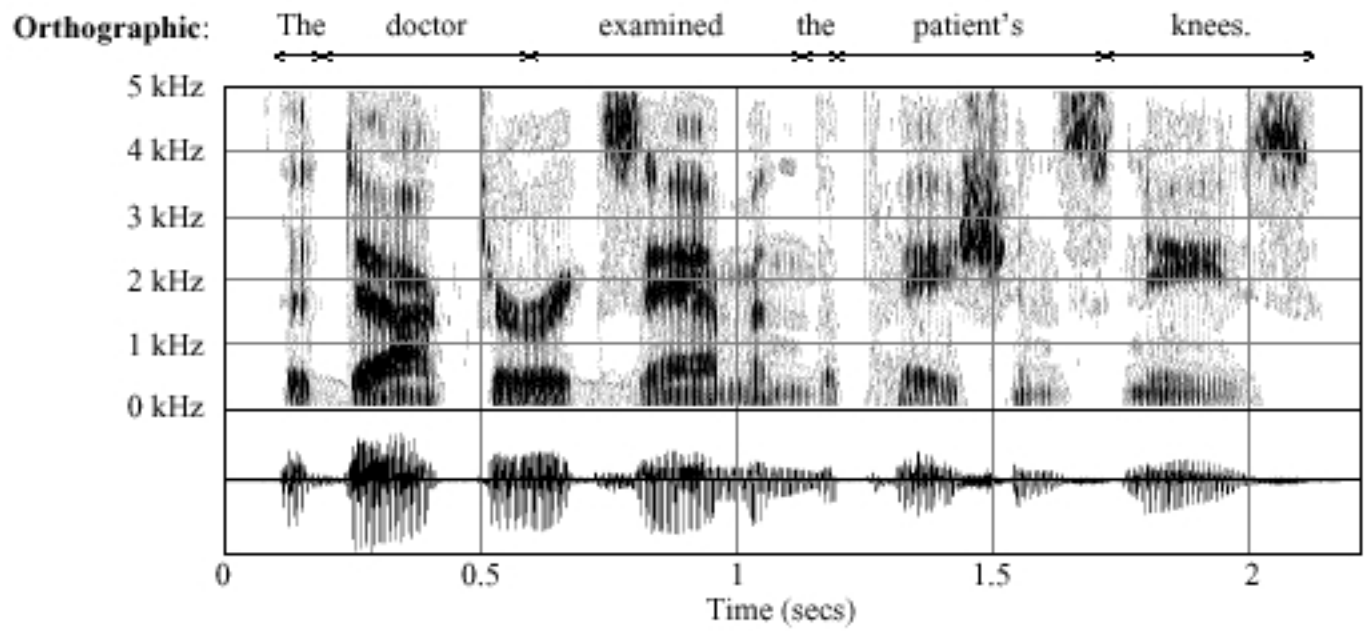
and

$r = \lfloor r' \rfloor$ (greatest integer less than or equal to r)

$$F_c = \frac{B_1 + B_2}{2}$$

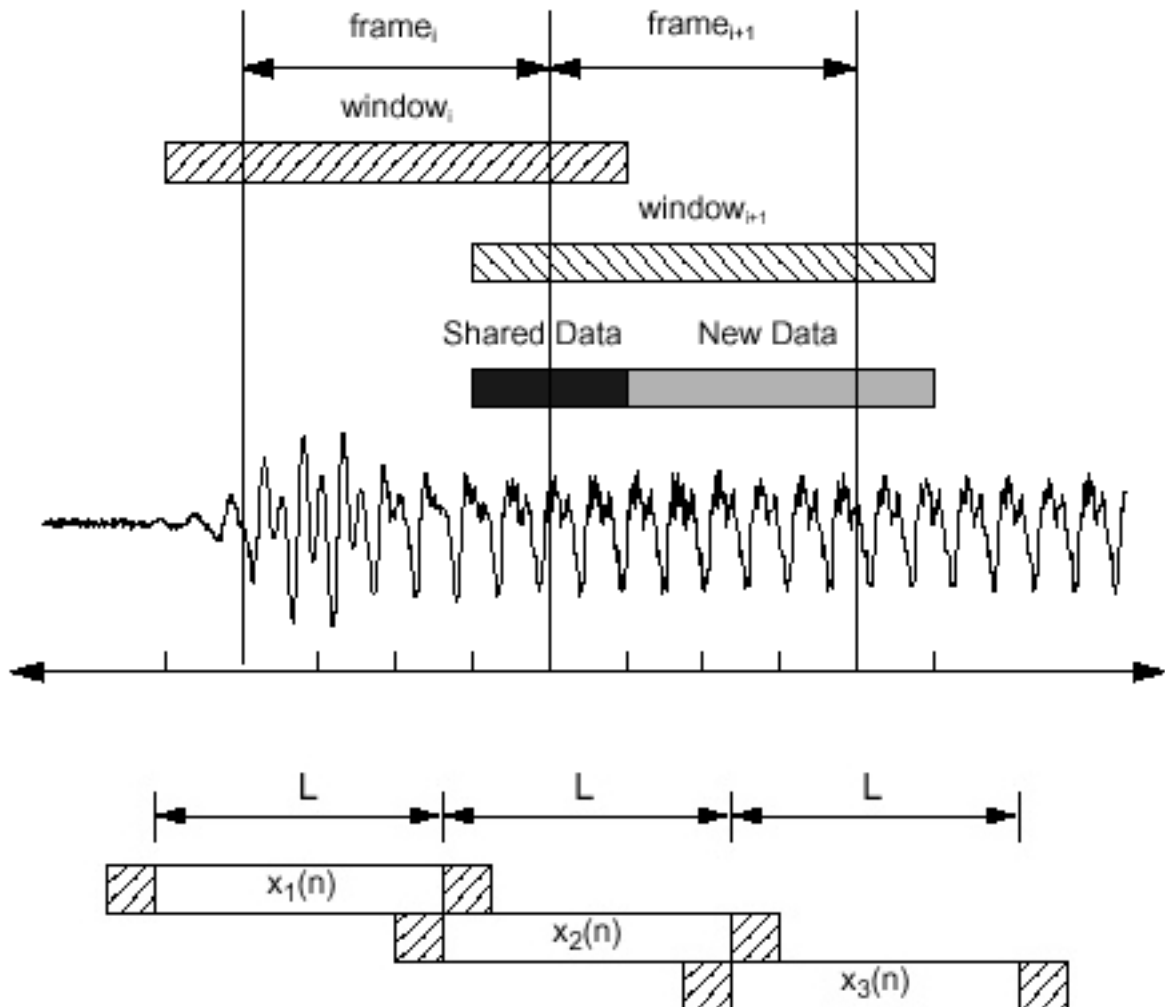
TYPICAL SAMPLING FREQUENCIES IN SPEECH RECOGNITION

- **8 kHz:** Popular in digital telephony. Provides coverage of first three formants for most speakers and most sounds.
- **16 kHz:** Popular in speech research. Why?
- **6.67 kHz:** Why?
- **Sub 8 kHz Sampling:** Can aliasing be useful in speech recognition? Hint: Consumer electronics.



A FRAME-BASED ANALYSIS IS ESSENTIAL

- Consider the problem of performing a piecewise linear analysis of a signal:

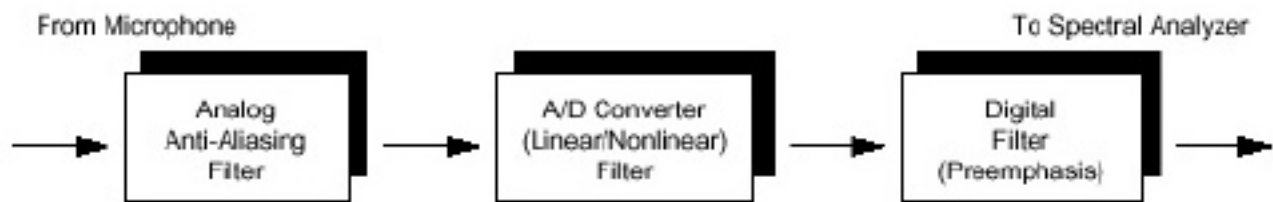


- This is most often implemented in hardware using a circular buffer.
- If we assume the signal is piecewise stationary, we can analyze the signal using a sliding window approach. Two key parameters are:
 - **Frame Duration:** how often we perform the analysis.
 - **Window Duration:** how many samples we use for the analysis.
- Recall we introduced similar parameters for the spectrogram. Typical values are a 10 ms frame duration and 25 ms window duration. Why?

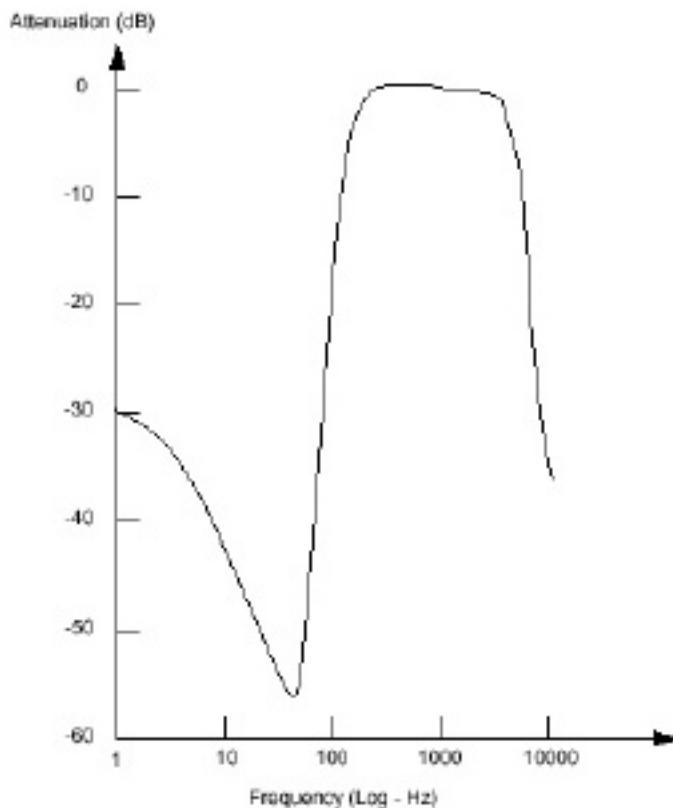
- Important questions:

- How does the window duration impact the spectral resolution?
- Why so much overlap?
- Why do we use a 10 ms frame duration?

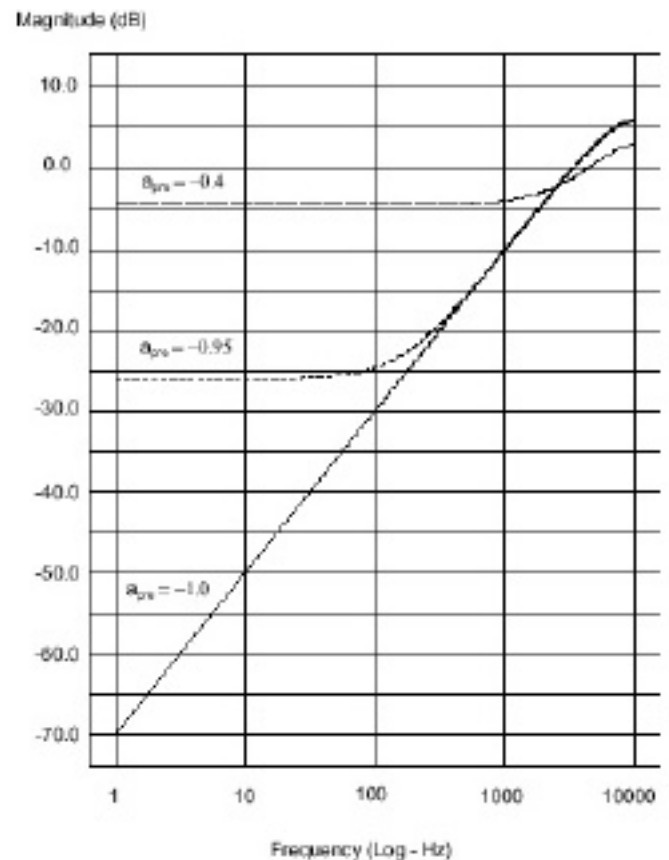
SIGNAL CONDITIONING COMPENSATES FOR MICROPHONE AND CHANNEL CHARACTERISTICS



Frequency Response of a CODEC



Preemphasis Filter



[Return to Main](#)[Objectives](#)**Direct Forms:**[Interpolation](#)[Ratios of Integers](#)**Conjugate Filters:**[Two-Band](#)[Design](#)**Example:**[Speech Waveform](#)[Filterbank](#)**On-Line Resources:**[Signal Modeling](#)[Multirate](#)[Software](#)

LECTURE 09: RESAMPLING

- Objectives:
 - Learn how to change the sample rate of a signal
 - Understand how this can be implemented using time domain interpolation (based on the Sampling Theorem)
 - Understand how this can be implemented efficiently using digital filters
 - Introduction to digital filter banks

A good reference textbook on these topics is:

J.G. Proakis and D.G. Manolakis,
*Digital Signal Processing:
Principles, Algorithms, and
Applications*, Prentice Hall,
Upper Saddle River, New Jersey,
USA, ISBN: 0-13-373762-4,
1996 (third edition).

The course textbook:

X. Huang, A. Acero, and H.W.
Hon, *Spoken Language
Processing - A Guide to Theory,
Algorithm, and System
Development*, Prentice Hall,
Upper Saddle River, New Jersey,
USA, ISBN: 0-13-022616-5,
2001.

has a detailed explanation of filter

banks (sections 5.6 and 5.7).

LECTURE 09: RESAMPLING

- Objectives:
 - Learn how to change the sample rate of a signal
 - Understand how this can be implemented using time domain interpolation (based on the Sampling Theorem)
 - Understand how this can be implemented efficiently using digital filters
 - Introduction to digital filter banks

A good reference textbook on these topics is:

J.G. Proakis and D.G. Manolakis, *Digital*

Signal Processing: Principles, Algorithms, and Applications, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-373762-4, 1996 (third edition).

The course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

has a detailed explanation of filter banks (sections 5.6 and 5.7).

INTERPOLATION USING THE SAMPLING THEOREM

How do we change the sample frequency of a signal:

Method 1: Use the sampling theorem (Lecture No. 3)

Define F_s^1 as the original sample frequency, and F_s^2 as the new sample frequency. Recall our interpolation function, where $B = \frac{F_s^1}{2}$:

$$g(t) = \frac{\sin(2\pi Bt)}{2\pi Bt}.$$

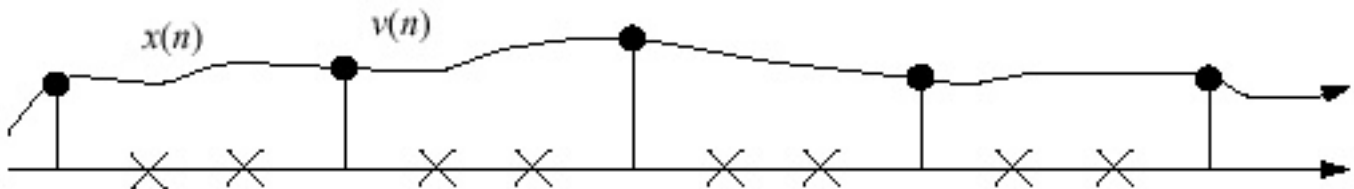
$x(\frac{m}{F_s^2})$ may be expressed as:

$$x(\frac{m}{F_s^2}) = \sum_{n=-\infty}^{\infty} x(\frac{n}{F_s^1}) g(\frac{m}{F_s^2} - \frac{n}{F_s^1}).$$

What are the disadvantages of this method?

Method 2: Downsampling a signal by dropping samples

Consider the signal $x(n)$. What is the spectrum of $v(n) = x(Ln)$?



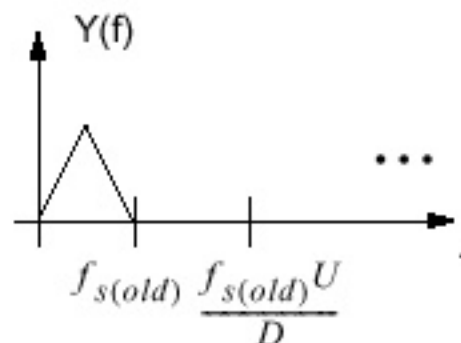
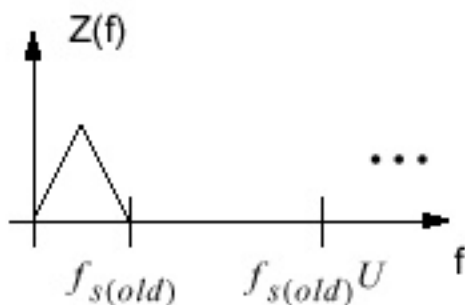
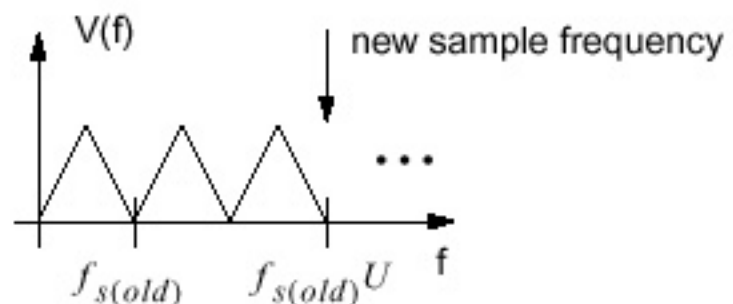
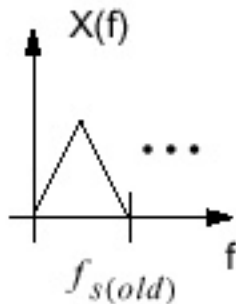
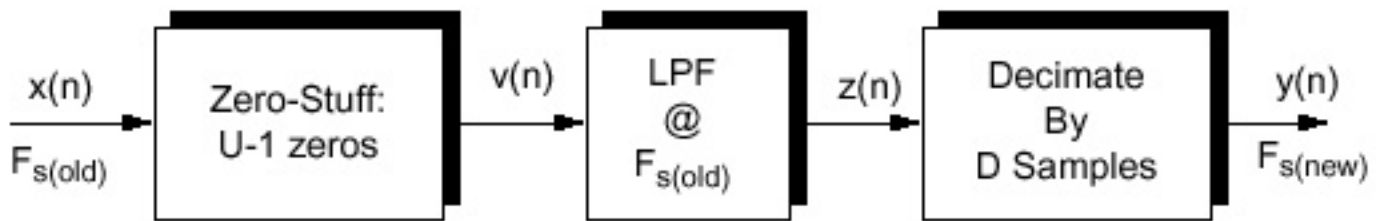
Recall the frequency-scaling property:

$$V(\omega) = \sum_{m=-\infty}^{\infty} v(m) e^{-j\omega m}$$

$$\begin{aligned}
 & \sum_{n=-\infty}^{\infty} x(Ln) e^{-j\omega nL} \\
 &= X(\omega/L)
 \end{aligned}$$

INTERPOLATION AND DECIMATION USING RATIOS OF INTEGERS

$$F_{s(new)} = F_{s(old)} \left(\frac{U}{D} \right)$$



Note that the LPF is run at the decimation rate of D!

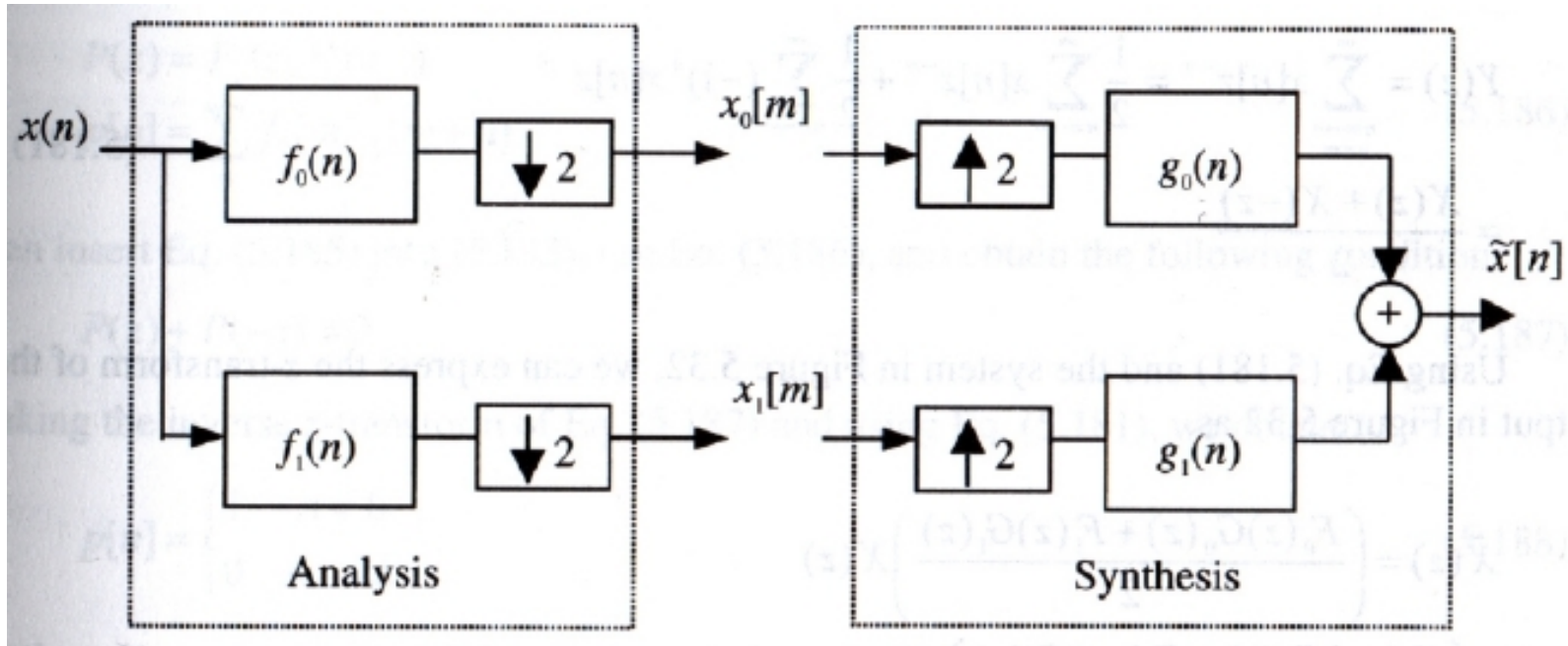
Questions:

- Under what conditions will this introduce no distortion?
- How do we implement this efficiently?
- How should we convert from 8 kHz to 6.4 kHz?

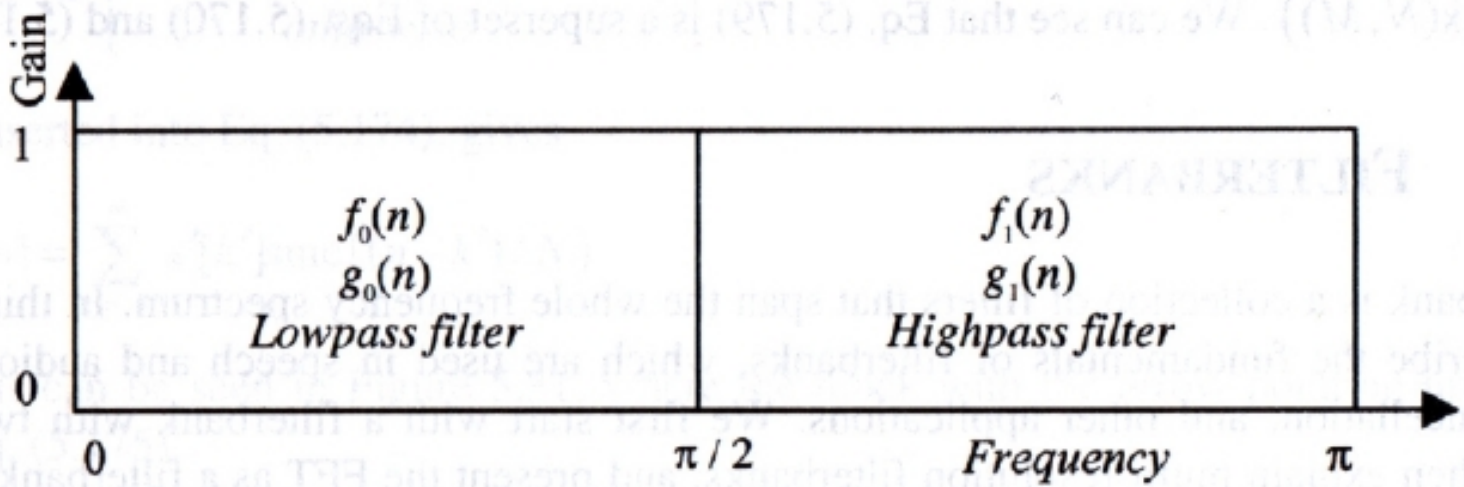
- What about the infamous 44.1 kHz CD sample frequency?

DIGITAL FILTERBANKS BASED ON RESAMPLING

Consider the two-band filterbank shown below:



where $f(n)$ and $g(n)$ are complementary low pass and high pass filters:



- To achieve perfect reconstruction of $x(n)$, we need ideal filters, which are not realizable.
- Is it possible to build a filterbank that has perfect reconstruction?
- Why might such a filterbank be useful for speech recognition?

DESIGN OF CONJUGATE QUADRATURE MIRROR FILTERS

If we specify $f_1(n)$, $g_0(n)$, and $g_1(n)$ as a function of $f_0(n)$, we can derive a compact design procedure:

1. Design a $(2L - 1)$ tap half-band linear phase low-pass filter $p(n)$ (use Parks-McClellan or Kaiser Window approach).
2. Factor $P(z) = F_0(z) F_0(z^{-1})$ by finding roots.
3. Compute the remaining filter impulse responses as follows:

$$f_1(n) = (-1)^n f_0(L-1-n)$$

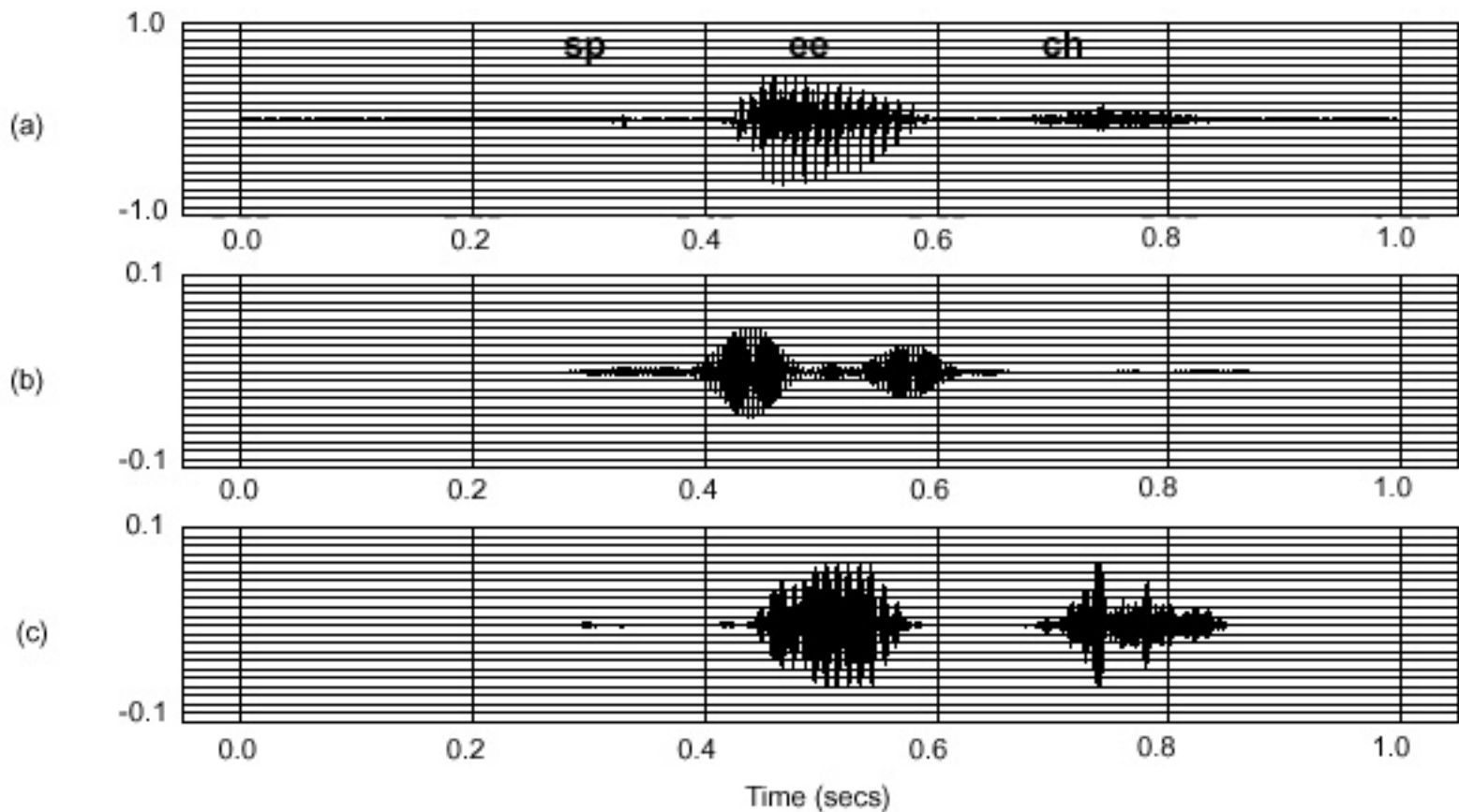
$$g_0(n) = f_0(L-1-n)$$

$$g_1(n) = f_1(L-1-n)$$

What are the advantages of this approach?

There are several popular approaches to implementing such digital filterbanks. This general area of research is known as **multirate signal processing**. What are the merits of a frequency domain approach?

EXAMPLE OUTPUT FROM A DIGITAL FILTER BANK

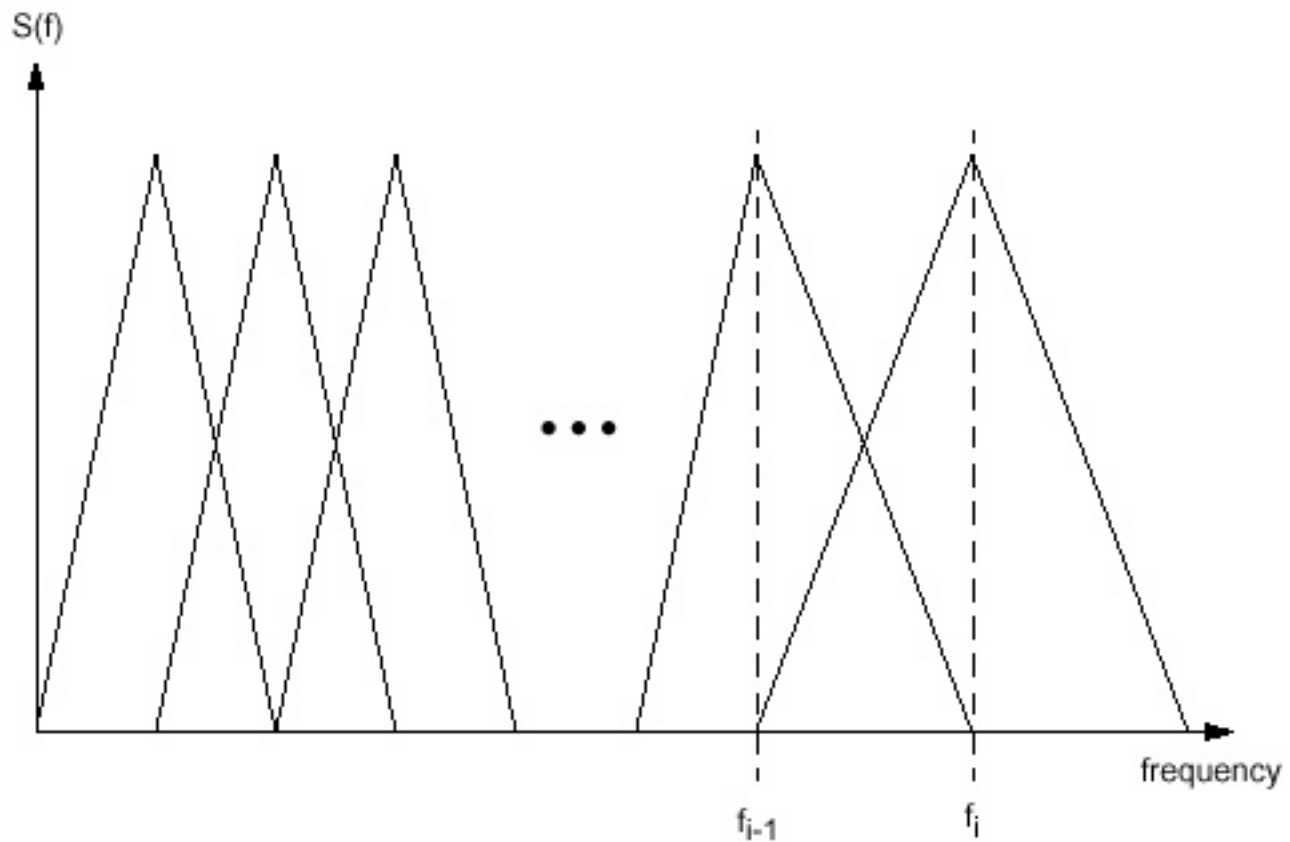


- Digital filter bank outputs for a speech signal shown in (a), consisting of the word *speech*. In (b), the output from a filter with a center frequency of 250 Hz and a bandwidth of 100 Hz is shown. In (c), the output from a filter

centered at 2500 Hz is shown.

- Note that the amplitude of the output for each filter varies depending on the nature of the sound. The final *ch* sound, for example, is mainly composed of high frequency information.

A DIGITAL FILTERBANK



- Note that an FFT yields frequency samples at $(k/N)f_s$.
- Oversampling provides a smoother estimate of the envelope of the spectrum.

- Other efficient techniques exist for different frequency scales (e.g., bilinear transform).

[Return to Main](#)[Objectives](#)**Microphones:**[Condenser](#)[Electret](#)**Directionality:**[Omnidirectional](#)[Bidirectional](#)[Unidirectional](#)[Other](#)**Adaptive Filtering:**[Echo Cancellation](#)[Arrays](#)**On-Line Resources:**[Microphone Basics](#)[Echo Cancellation Software](#)[CAIP Arrays](#)

LECTURE 10: ACOUSTIC TRANSDUCERS

- Objectives:
 - Introduce the basic types of microphones
 - Understand microphone impedance and other physical parameters
 - Learn how these influence the speech signal
 - Introduce the concept of adaptive filtering to improve signal quality
 - Introduce advanced

technology such as
microphone arrays

This material follows the course
textbook closely:

X. Huang, A. Acero, and H.W.
Hon, *Spoken Language
Processing - A Guide to Theory,
Algorithm, and System
Development*, Prentice Hall,
Upper Saddle River, New Jersey,
USA, ISBN: 0-13-022616-5,
2001.

This textbook contains an excellent
discussion of these topics. Another
good reference source for this
material is:

G.S.K. Wong and T.F.W.
Embleton (Eds.), *AIP Handbook
of Condensor Microphones:
Theory, Calibration, and
Measurements*, American

Institute of Physics, New York,
New York, USA, ISBN:
1-56396-284-5, 1995.

This is one of the definitive
publications on condensor
microphones.

LECTURE 10: ACOUSTIC TRANSDUCERS

● Objectives:

- Introduce the basic types of microphones
- Understand microphone impedance and other physical parameters
- Learn how these influence the speech signal
- Introduce the concept of adaptive filtering to improve signal quality
- Introduce advanced technology such as microphone arrays

This material follows the course textbook closely:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

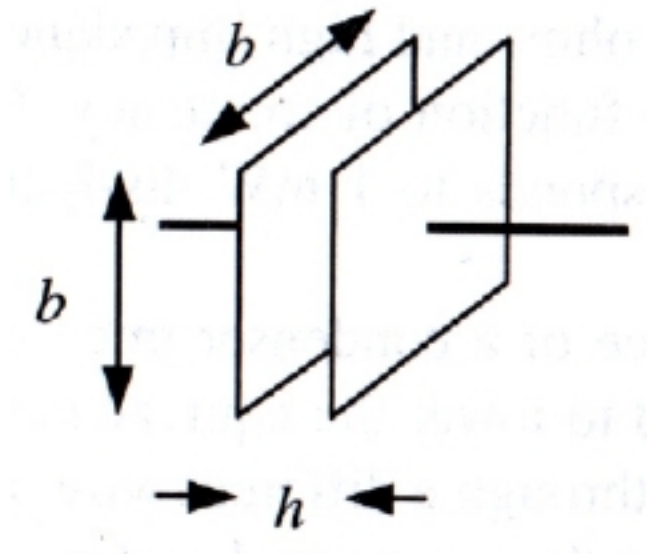
This textbook contains an excellent discussion of these topics. Another good reference source for this material is:

G.S.K. Wong and T.F.W. Embleton (Eds.), *AIP Handbook of Condensor Microphones: Theory, Calibration, and Measurements*, American Institute of Physics, New York, New York, USA, ISBN: 1-56396-284-5, 1995.

This is one of the definitive publications on condensor microphones.

CONDENSER MICROPHONES

A *condensor* microphone has a capacitor consisting of a pair of metal plates separated by an insulating material called a dielectric:



One of its plates is free to move in response to changes in sound pressure. The sensitivity of the microphone is related to its polarizing voltage and distance of separation between these plates.

Key design equations for this type of microphone are:

$$C = \epsilon_0 \pi b^2 / h$$

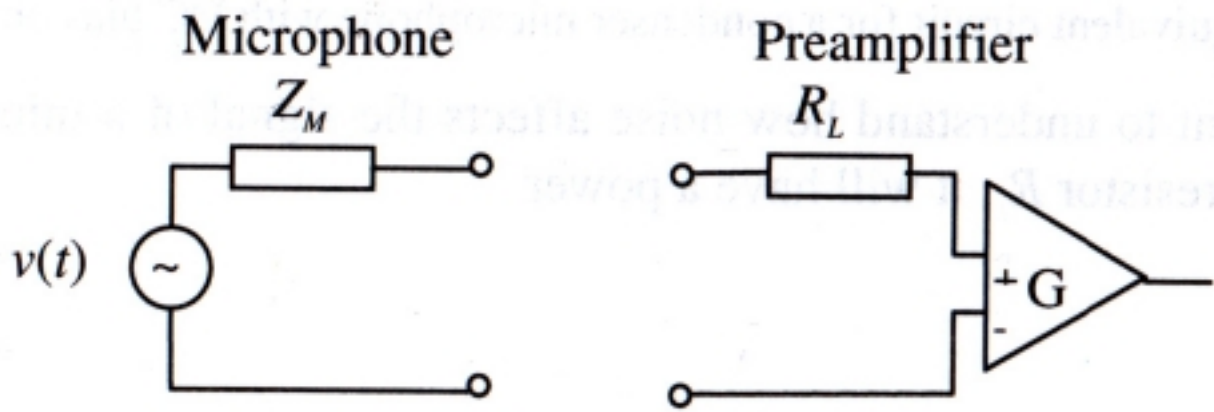
$$Q = CV_{cc}$$

$$\nabla V = \nabla h V_{cc} / h$$

Thus, the sensitivity depends on the polarizing voltage, V_{cc} , which explains why many microphones operate at large voltages (often 100V or more).

ELECTRET MICROPHONES

- An *electret* microphone is a specific type of condenser microphone that does not require a special polarizing voltage because its diaphragm or back plate is permanently charged.
- Electret microphones are small, cheap, durable, and offer good performance at high frequencies. Most modern telephone handsets use electrets.
- The electrical equivalent circuit for a microphone is:



Bridging refers to maximizing the output voltage by increasing the load impedance and/or decreasing the microphone impedance.

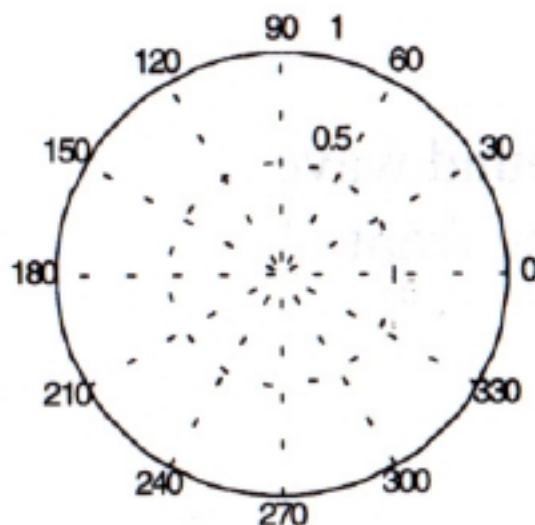
- Low impedance microphones have an impedance of 100 to 300 Ohms (more expensive); high impedance microphones have an impedance of 600 to 1000 Ohms (less expensive).
- Condenser microphones require a DC bias; balanced cables (XLR) are preferred over unbalanced lines (1/8" mini plugs) because the cables are more resistant to thermal and RF noise.

- Digital and wireless microphones are popular alternatives.

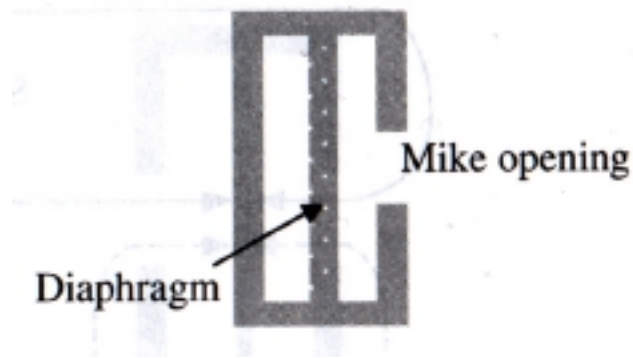
OMNIDIRECTIONAL SENSITIVITY PATTERNS

A microphone's directionality pattern can be described in terms similar to what we use for antennae. Its sensitivity can be measured as a function of direction. Microphones can be classified as *omnidirectional* (nondirectional) and *directional* (e.g., bidirectional and unidirectional).

The polar response of an ideal omnidirectional microphone is:



A cross-sectional view of the microphone:

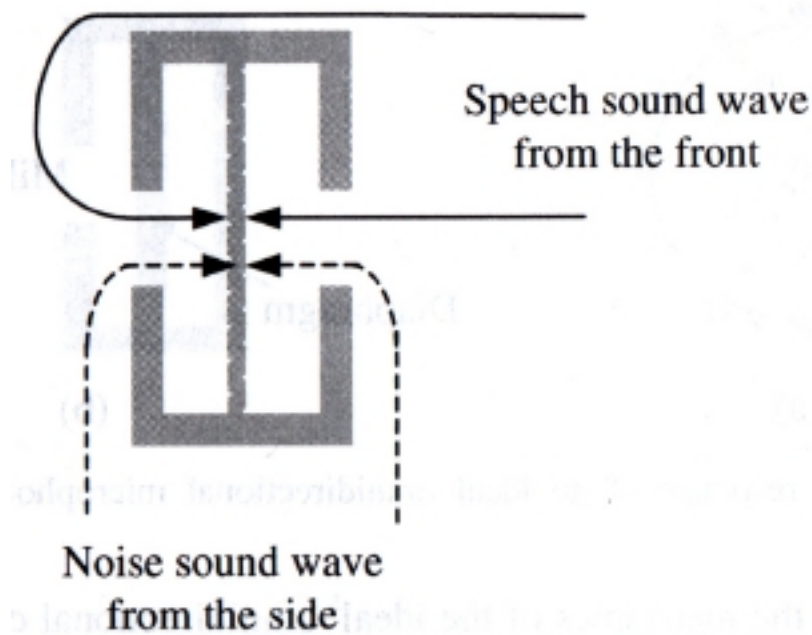


reveals that the diaphragm is designed to be sensitive to signals emanating from any direction.

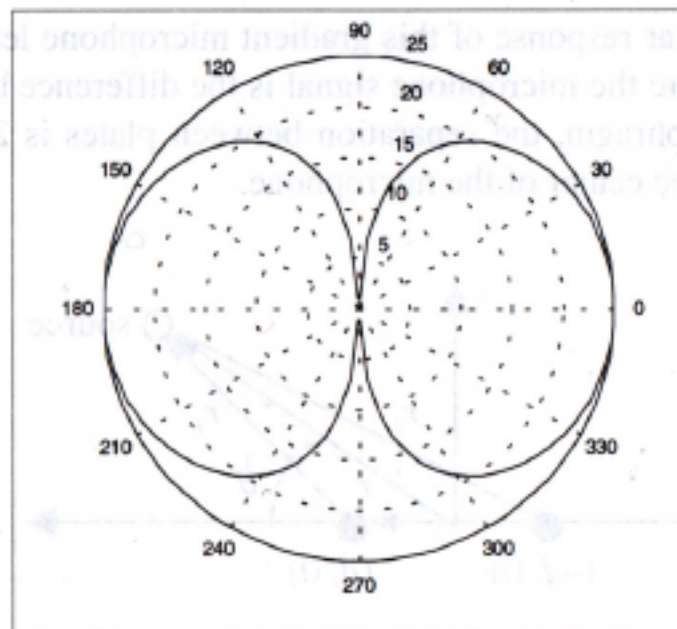
BIDIRECTIONAL SENSITIVITY PATTERNS

A bidirectional microphone is a *noise-cancelling* microphone (such as the Sennheiser HD 414 close-talking microphone that is so popular in speech research).

A bidirectional microphone uses properties of a gradient microphone to achieve noise cancellation. Sound pressure never arrives at the front and the back of the microphone at the same time. However, noise, which arrives from the side, does, and hence is cancelled.



These microphones have directional sensitivity patterns:

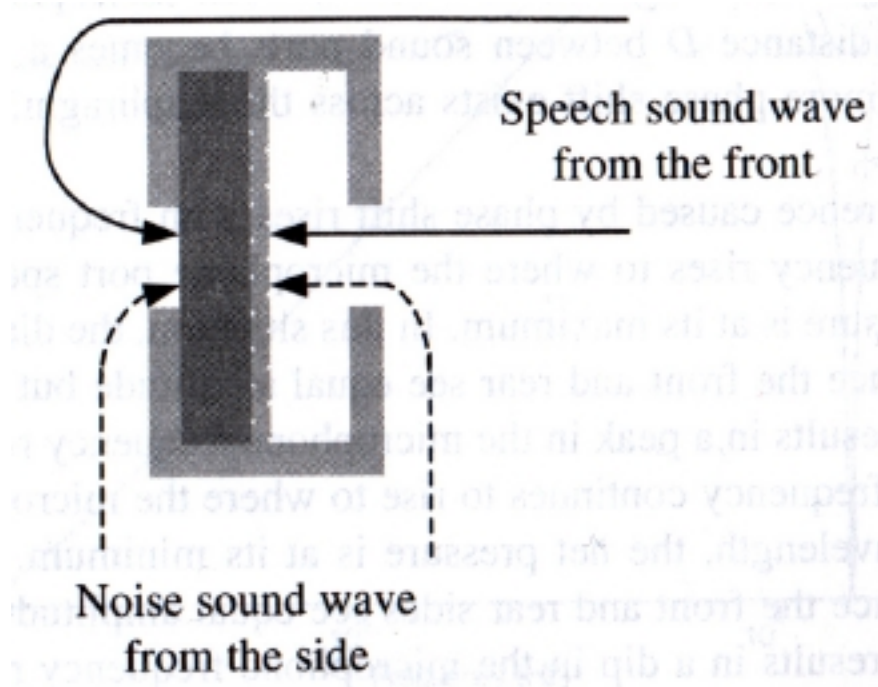


Such microphones are very sensitive to placement, and cannot be used interchangeably with recognition systems (microphone

independence?).

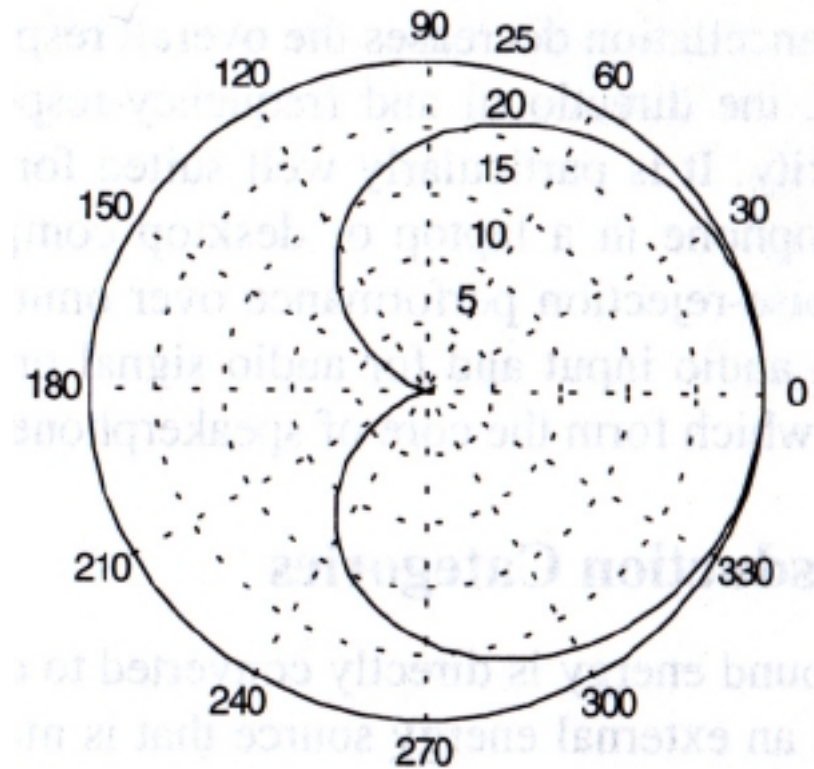
UNIDIRECTIONAL SENSITIVITY PATTERNS

Unidirectional microphones, which are popular in computer applications involving desktop microphones, are similar to close-talking microphones:



These are often referred to as *cardiod* microphones. They have the following

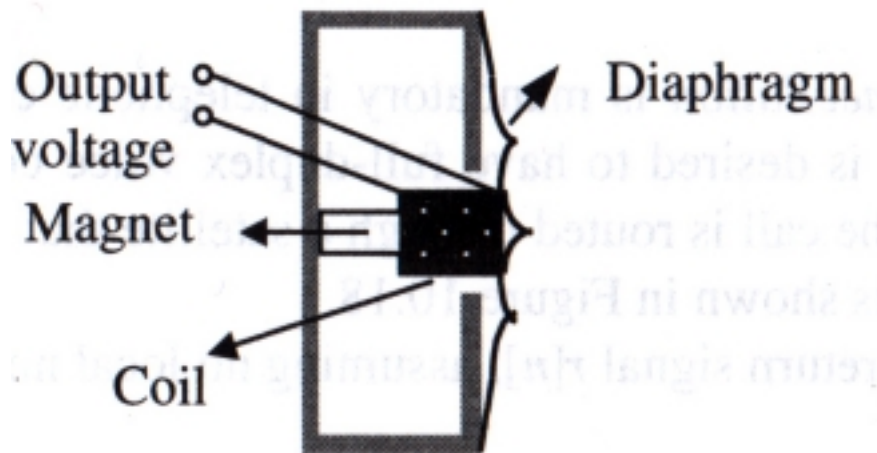
sensitivity pattern:



PIEZOELECTRIC AND OTHER TYPES OF MICROPHONES

Microphones can be classified in terms of how they create an electrical signal:

- **Electromagnetic:** Ribbons (thin metal ribbon suspended between magnets), dynamic (moving coil)

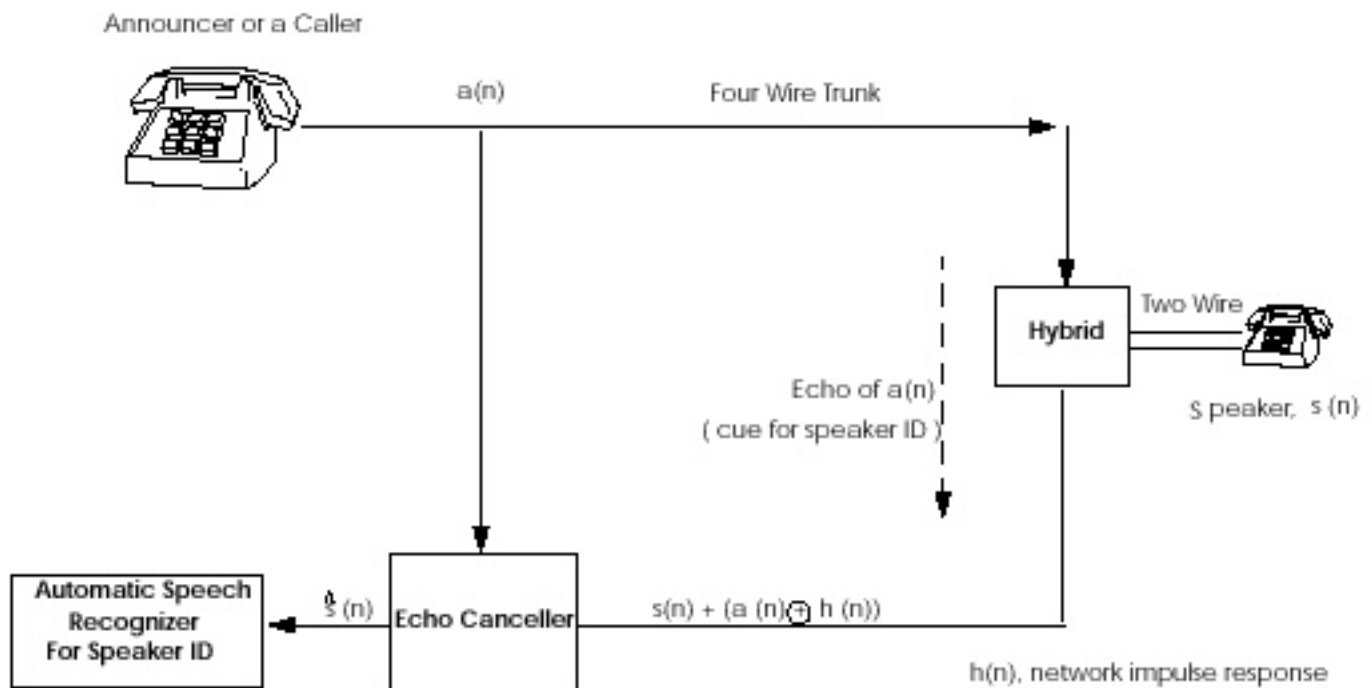


- **Electrostatic:** condensers, electrets, etc.
- **Piezoelectric:** based on variations of electric

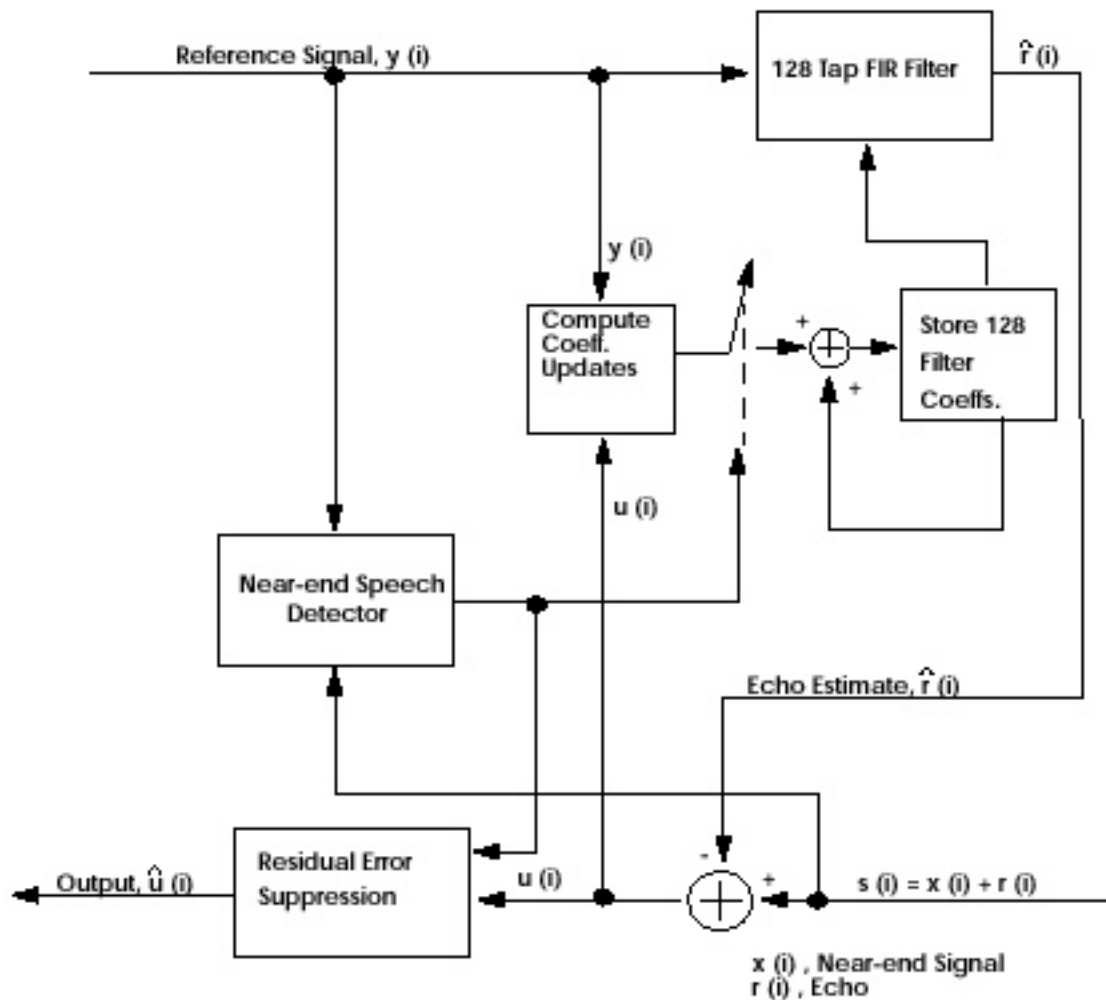
resistance of their sensor induced by changes in sound pressure. Carbon button microphones (old AT&T handsets) and desktop "far-field" microphones are popular examples of these. Lower sensitivity, more distortion, and non-flat frequency responses are characteristics of these microphones.

ECHO CANCELLATION

Echo cancellers are often required in speech recognition systems due to analog impairments present in the telephone system:



These are implemented using simple FIR adaptive filtering techniques:



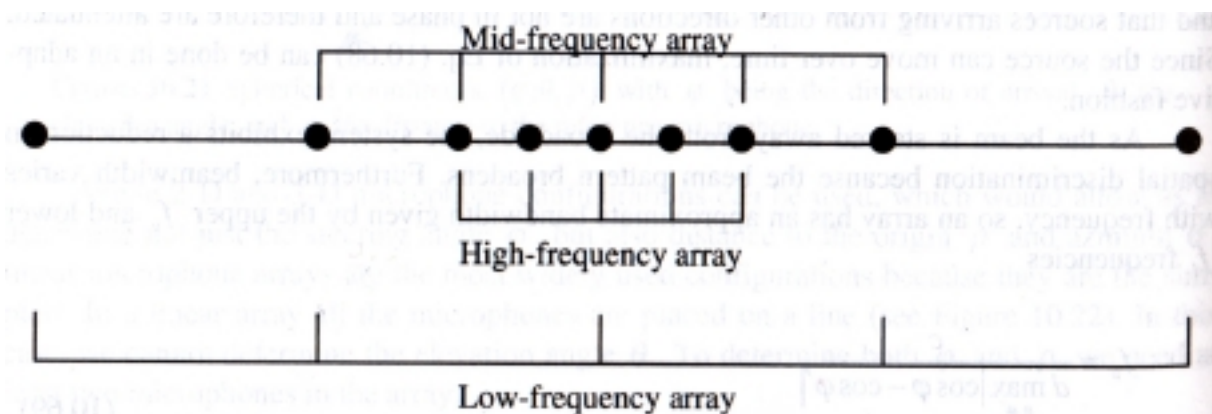
The coefficients of these filters are computed using a least mean-squared error approach (LMS). Such systems are used to allow users to speak during a prompt (barge-in), which is a very important feature of a practical recognition system. A reference implementation of a standard [FIR echo canceller](#) is available on-line, along with many [educational resources](#) and conference papers on [applications to speech](#)

recognition.

MICROPHONE ARRAYS AND BEAM STEERING

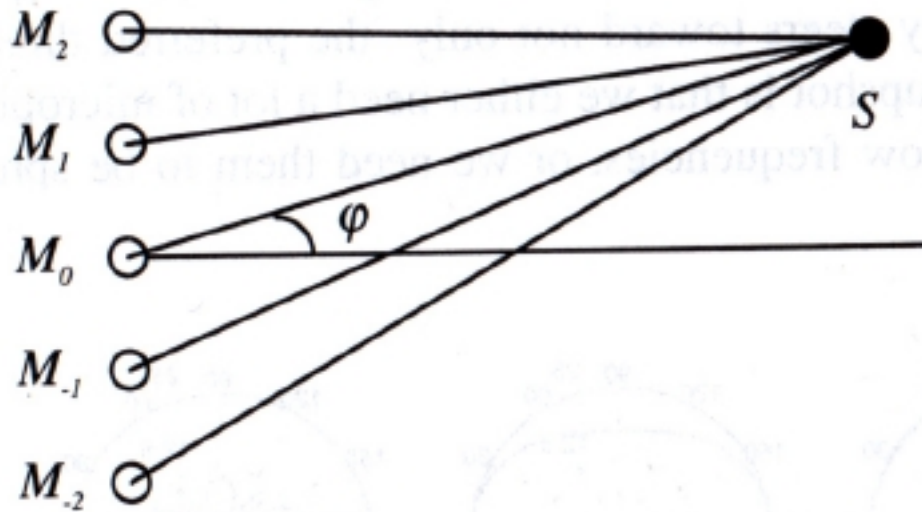
Question: Can we improve performance using multiple microphones?

The goal of a *microphone array* is to localize a sound source by directing a group of microphones to be most sensitive in a specific direction. The procedure is completely analogous to analog antenna theory.



Steering of the array has several uses: increase SNR, direct video in teleconferencing, enhance the human interface (hands-free).

Steering the array amounts to adjusting the delays in each microphone. The most common implementation is the *delay and sum beamformer*:



An example of a sensitivity pattern for this type of array is:

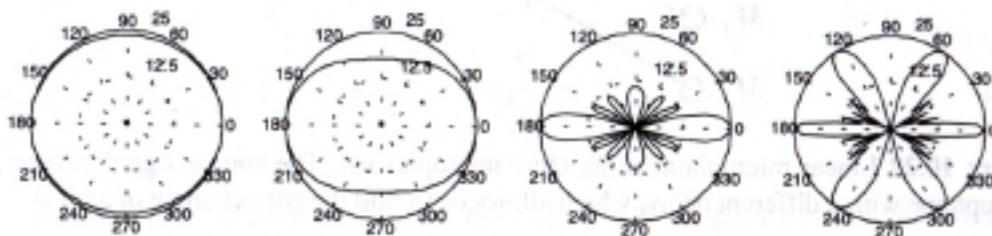


Figure 10.23 Polar pattern of a microphone array with steering angle of $\phi' = 0$, five microphones spaced 5 cm apart for 400, 880, 4400, and 8000 Hz from left to right, respectively, for a source located at 5 m.

A 3D array can be used to localize sound to a

single point in a room (direction and distance). 2D arrays are most commonly used to enhance SNR. Unfortunately, performance increases slowly as a function of the number of microphones (1 dB rule). Hence, this technology is impractical for many consumer applications.

Two microphone versions of this idea based on adaptive filtering are popular in automotive applications for noise suppression.

[Return to Main](#)[Objectives](#)**Short-Time Measurements:**[Energy](#)[Sums and Filters](#)[Examples](#)**Windows:**[Spectrograms](#)[Rectangular Windows](#)[Frequency Response](#)[Popular Windows](#)[Recursive](#)[Control Systems](#)**On-Line Resources:**[Signal Modeling](#)[The HTK Book](#)[Windows](#)

LECTURE 11: TEMPORAL ANALYSIS

- Objectives:
 - Understand the relationship between sums and filters
 - Understand the relationship between temporal resolution and frequency resolution
 - Introduce common window functions
 - Explain their use in speech processing
 - Understand how we compute our first recognition feature:

energy

One of the best explanations of this material can be found in:

L.R. Rabiner and B.W. Juang,
*Fundamentals of Speech
Recognition*, Prentice-Hall, Upper
Saddle River, New Jersey, USA,
ISBN: 0-13-015157-2, 1993.

This textbook is unfortunately out of print. Another excellent reference is:

J. Deller, et. al., *Discrete-Time
Processing of Speech Signals*,
MacMillan Publishing Co., ISBN:
0-7803-5386-2, 2000.

LECTURE 11: TEMPORAL ANALYSIS

- Objectives:
 - Understand the relationship between sums and filters
 - Understand the relationship between temporal resolution and frequency resolution
 - Introduce common window functions
 - Explain their use in speech processing
 - Understand how we compute our first recognition feature: energy

One of the best explanations of this material can be found in:

L.R. Rabiner and B.W. Juang, *Fundamentals of Speech Recognition*, Prentice-Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-015157-2, 1993.

This textbook is unfortunately out of print. Another excellent reference is:

J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co., ISBN: 0-7803-5386-2, 2000.

ENERGY AND POWER

Energy:

$$E \equiv \sum_{n=-\infty}^{\infty} |x(n)|^2$$

Average Power:

$$P = \lim_{N \rightarrow \infty} \frac{1}{2N+1} \sum_{n=-N}^N |x(n)|^2$$

Finite Energy:

$$E_N = \sum_{n=-N}^N |x(n)|^2$$

$$E = \lim_{N \rightarrow \infty} E_N$$

$$P = \lim_{N \rightarrow \infty} \frac{1}{2N+1} E_N$$

Comments:

- (1) If a signal's energy is finite, $P = 0$.
- (2) If a signal's energy is infinite, its power may or may not be zero.
- (3) RMS value is the square root of the power.

Examples:

- (1) The average power of a sinewave is $\frac{A^2}{2}$.

- (2) What does the following compute?

$$E(n) = E(n-1) + \alpha x^2(n)$$

FINITE SUMS AND FILTERS

- Consider a simplified equation for energy:

$$E = \sum_{n=0}^{N-1} |x(n)|^2 \quad n = 0, 1, \dots$$

- We can write this as a digital filter:

$$\begin{aligned} x(n) &= s^2(n) \\ E(n) &= x(n - (N - 1)) + x(n - (N - 2)) + \dots + x(n) \\ &= \sum_{k=0}^{N-1} |x(n - k)|^2 \\ H(z) &= 1 + z^{-1} + z^{-2} + \dots + z^{-(N-1)} \end{aligned}$$

- What is the frequency response of this filter? (Hint: FIR)
- Are there other ways we can implement such a filter? (Hint: IIR)

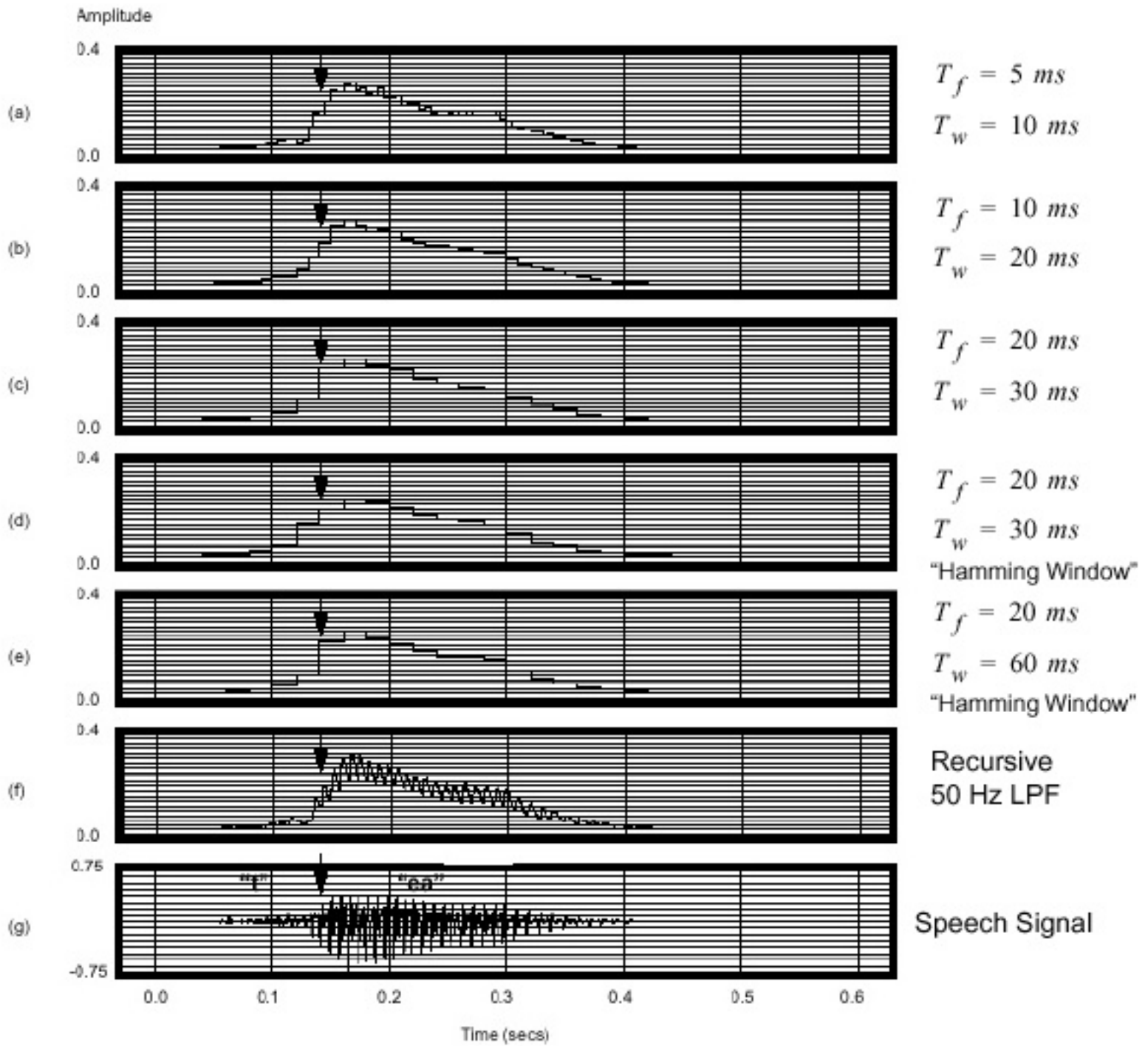
Consider these three approaches:

$$E(n) = E(n-1) - x(n - (N-1)) + x(n)$$

$$E(n) = \alpha E(n-1) + x(n)$$

$$E(n) = \alpha E(n-1) + \beta E(n-2) + x(n)$$

EXAMPLES OF ENERGY COMPUTATIONS



RECTANGULAR WINDOWS

Let $\{x(n)\}$ denote a sequence to be analyzed. Let's limit the duration of $\{x(n)\}$ to L samples:

$$\hat{x}(n) = x(n)w(n)$$

where $w(n)$ is a rectangular window and is defined as

$$w(n) = \begin{cases} 1, & 0 \leq n \leq L-1 \\ 0, & \text{otherwise} \end{cases}$$

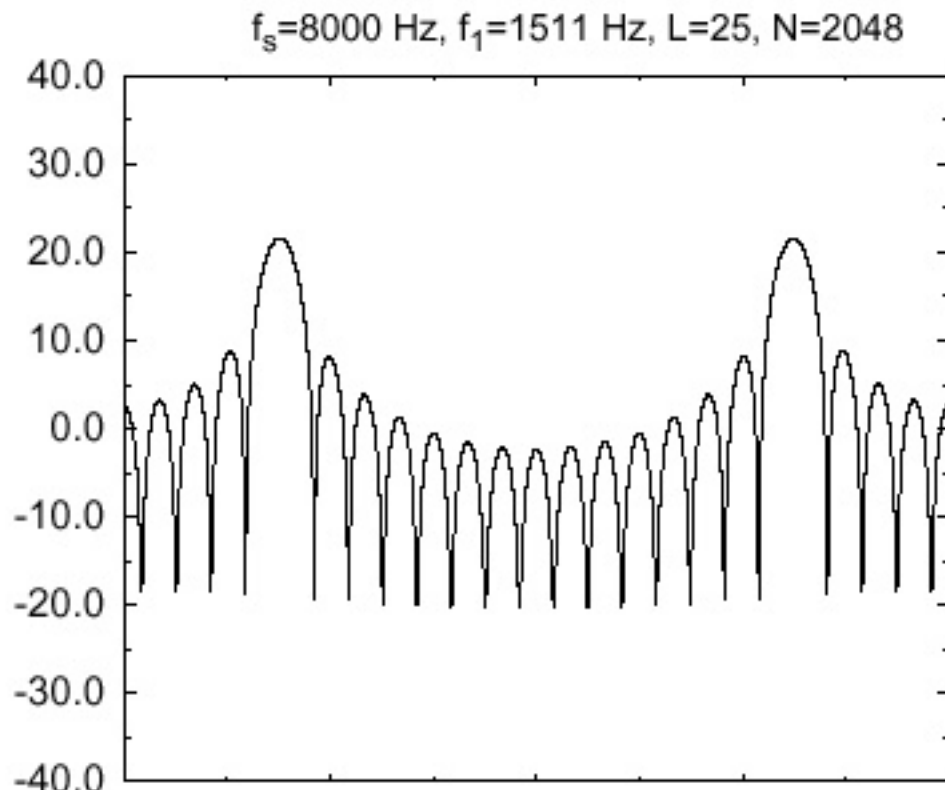
The Fourier transform of $w(n)$ is given by:

$$W(\omega) = \frac{\sin(\omega(L/2))}{\sin(\omega/2)} e^{-j\omega((L-1)/2)}$$

The transform of $\hat{x}(n)$ is given by:

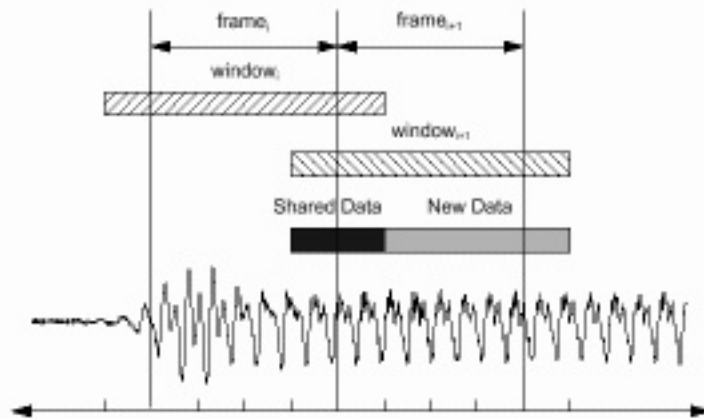
$$\hat{X}(\omega) = \frac{1}{2} [W(\omega - \omega_o) + W(\omega + \omega_o)].$$

This introduces frequency domain aliasing (the so-called picket fence effect):



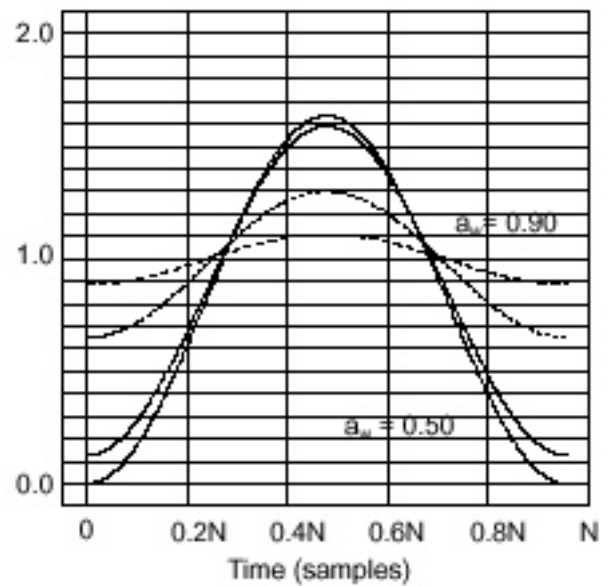
0.0 2000.0 4000.0 6000.0 8000.0

TEMPORAL/FREQUENCY RESPONSE



(a) Temporal Response

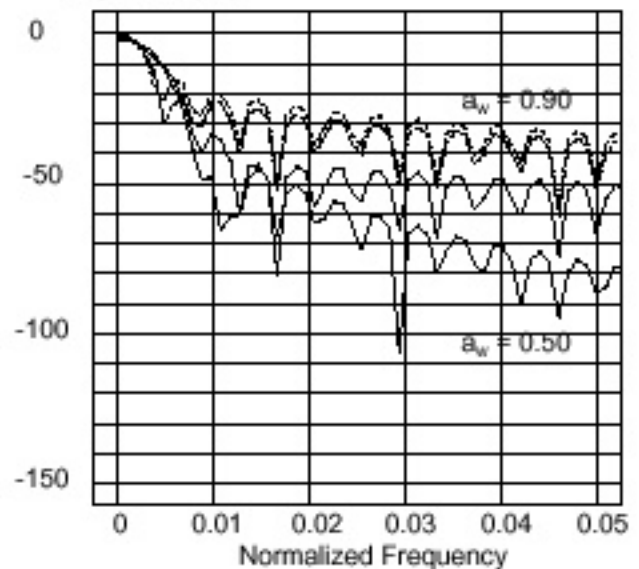
Amplitude



$$\%Overlap = \frac{(T_w - T_f)}{T_w} \times 100\%$$

(b) Frequency Response

Magnitude (dB)



POPULAR WINDOW FUNCTIONS

1. Rectangular:
$$w(k) = \begin{cases} 1, & |k| \leq N \\ 0, & \text{otherwise} \end{cases}$$

2. Generalized Hanning:
$$w_H(k) = w(k) \left[\alpha + (1 - \alpha) \cos\left(\frac{2\pi}{N}k\right) \right] \quad 0 < \alpha < 1$$

$$\alpha = 0.54, \quad \text{Hamming window}$$

$$\alpha = 0.50, \quad \text{Hanning window}$$

3. Bartlett
$$w_B(k) = w(k) \left[1 - \frac{|k|}{N+1} \right]$$

4. Kaiser
$$w_K(k) = w(k) I_0\left(\alpha \sqrt{1 - \frac{K^2}{N}}\right) / I_0(\alpha)$$

5. Chebyshev:
$$w_N(k) = 2(x_0^2 - 1)w_{N-1}(k) + x_0^2[w_{N-1}(k-1) + w_{N-1}(k+1)] - w_{N-2}(k)$$

6. Gaussian
$$w_G(k) = \begin{cases} \exp\left[-\frac{1}{2}k^2 \tan^2\left(\frac{\theta_0}{2}\right)\right] & |k| < N \\ w_G(N-1) / \left[2N \sin^2\left(\frac{\theta_0}{2}\right)\right] & |k| < N \\ 0 & |k| > N \end{cases}$$

There are many others. The most important characteristics are the width of the main lobe and the attenuation in the stop-band (height of highest sidelobe). The Hamming window is used quite extensively.

RECURSIVE-IN-TIME APPROACHES

Define the short-term estimate of the power as:

$$P(n) = \frac{1}{N_s} \sum_{m=0}^{N_s-1} \left(w(m) s\left(n - \frac{N_s}{2} + m\right) \right)^2$$

We can view the above operation as a moving-average filter applied to the sequence $s^2(n)$.

This can be computed recursively using a linear constant-coefficient difference equation:

$$P(n) = - \sum_{i=1}^{N_a} a_{pw}(i) P(n-i) + \sum_{j=1}^{N_b} b_{pw}(j) s^2(n-j)$$

Common forms of this general equation are:

$$P(n) = \alpha P(n-1) + s^2(n) \quad (\text{Leaky Integrator})$$

$$P(n) = \alpha P(n-1) + (1-\alpha) s^2(n) \quad (\text{First-order weighted average})$$

$$P(n) = \alpha P(n-1) + \beta P(n-2) + s^2(n) + \gamma s^2(n-1) \quad (2^{\text{nd}}\text{-order Integrator})$$

Of course, these are nothing more than various types of low-pass filters, or adaptive controllers. How do we compute the constants for these equations?

In what other applications have we seen such filters?

RELATIONSHIP TO CONTROL SYSTEMS

The first-order systems can be related to physical quantities by observing that the system consists of one real pole:

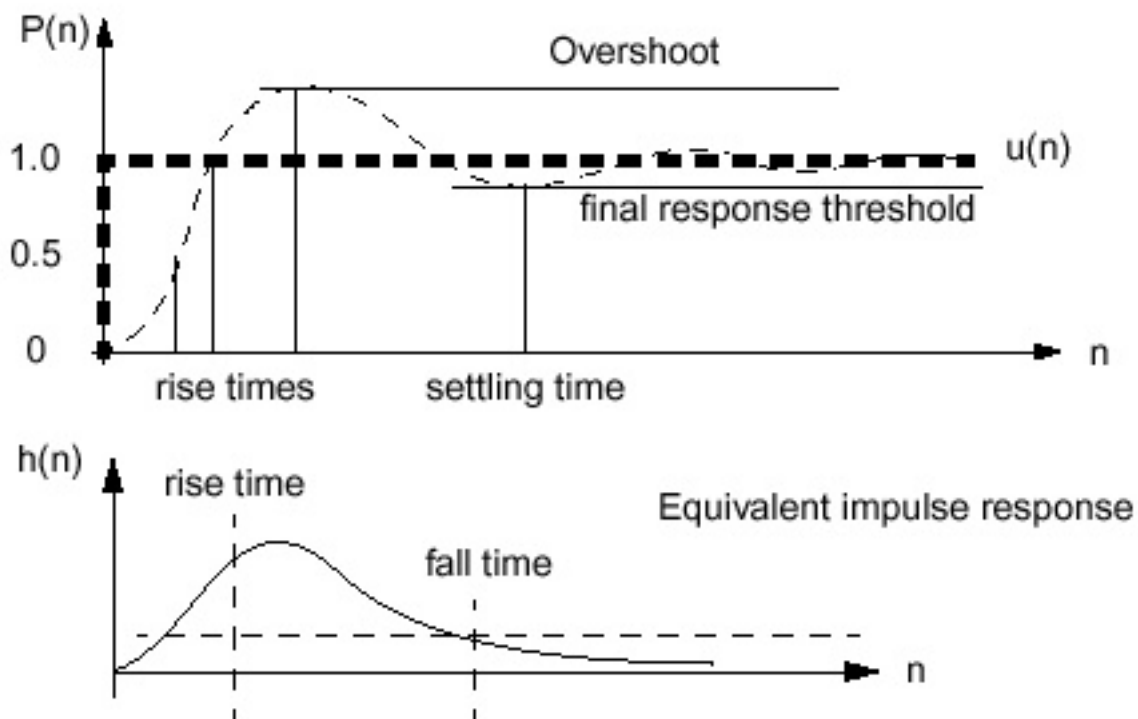
$$H(z) = \frac{1}{1 - \alpha z^{-1}}$$

α can be defined in terms of the bandwidth of the pole.

For second-order systems, we have a number of alternatives. Recall that a second-order system can consist of at most one zero and one pole and their complex conjugates. Classical filter design algorithms can be used to design the filter in terms of a bandwidth and an attenuation.

An alternate approach is to design the system in terms of its unit-step response:

There are many forms of such controllers (often known as



servo-controllers). One very interesting family of such systems are those that correct to the velocity and acceleration of the input. All such systems can be implemented as a digital filter.

[Return to Main](#)[Objectives](#)**Fourier Transform:**[Z-Transform](#)[Discrete Fourier Transform](#)[Fast Fourier Transform](#)**Discrete Cosine Transform:**[Definition](#)[Types](#)**Filterbanks:**[Non-linear Frequency Warping](#)[Overlapping Filters](#)[Oversampling](#)**Summary:**[Signal Modeling](#)[Phase](#)**On-Line Resources:**[Spectrum Analysis](#)[Software](#)[FFTW](#)[DCT](#)

LECTURE 12: FREQUENCY DOMAIN ANALYSIS

- Objectives:
 - Understand the Fourier Transform
 - Introduce the Discrete Cosine Transform
 - Understand frequency domain filterbanks
 - Justify the use of oversampling

This lecture combines material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory,*

Algorithm, and System Development, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

and information found in most standard DSP textbooks, including:

J.G. Proakis and D.G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-373762-4, 1996 (third edition).

LECTURE 12: FREQUENCY DOMAIN ANALYSIS

- Objectives:
 - Understand the Fourier Transform
 - Introduce the Discrete Cosine Transform
 - Understand frequency domain filterbanks
 - Justify the use of oversampling

This lecture combines material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice

Hall, Upper Saddle River, New Jersey, USA,
ISBN: 0-13-022616-5, 2001.

and information found in most standard DSP
textbooks, including:

J.G. Proakis and D.G. Manolakis, *Digital
Signal Processing: Principles, Algorithms,
and Applications*, Prentice Hall, Upper Saddle
River, New Jersey, USA, ISBN:
0-13-373762-4, 1996 (third edition).

Z-TRANSFORM

The z-transform of a discrete-time signal is defined as:

$$X(z) \equiv \sum_{n=-\infty}^{\infty} x(n)z^{-n}$$

$$x(n) = \frac{1}{2\pi j} \oint_C X(z)z^{n-1} dz$$

Its properties include:

| Property | Time-Domain | z-Domain |
|-----------------------------|-------------------------------------|-------------------------------------|
| Notation | $x(n)$ | $X(z)$ |
| | $x_1(n)$ | $X_1(z)$ |
| | $x_2(n)$ | $X_2(z)$ |
| Linearity and Superposition | $\alpha_1 x_1(n) + \alpha_2 x_2(n)$ | $\alpha_1 X_1(z) + \alpha_2 X_2(z)$ |
| Time-Shifting | $x(n - k)$ | $z^{-k} X(z)$ |
| Scaling in the z-domain | $\alpha^n x(n)$ | $X(\alpha^{-1} z)$ |
| Time reversal | $x(-n)$ | $X(z^{-1})$ |
| Conjugation | $x^*(n)$ | $X^*(z^*)$ |
| Real part | $\text{Re}[x(n)]$ | $\frac{1}{2} [X(z) + X^*(z^*)]$ |
| Imag part | $\text{Imag}[x(n)]$ | $\frac{1}{2j} [X(z) - X^*(z^*)]$ |

We typically assume the signal is time-limited, and compute the z-transform using a finite sum:

$$X(z) \equiv \sum_{n=0}^{N-1} x(n) z^{-n}$$

Note that the process of truncating a signal using a finite sum is essentially a windowing process,

and hence, frequency domain aliasing is introduced.

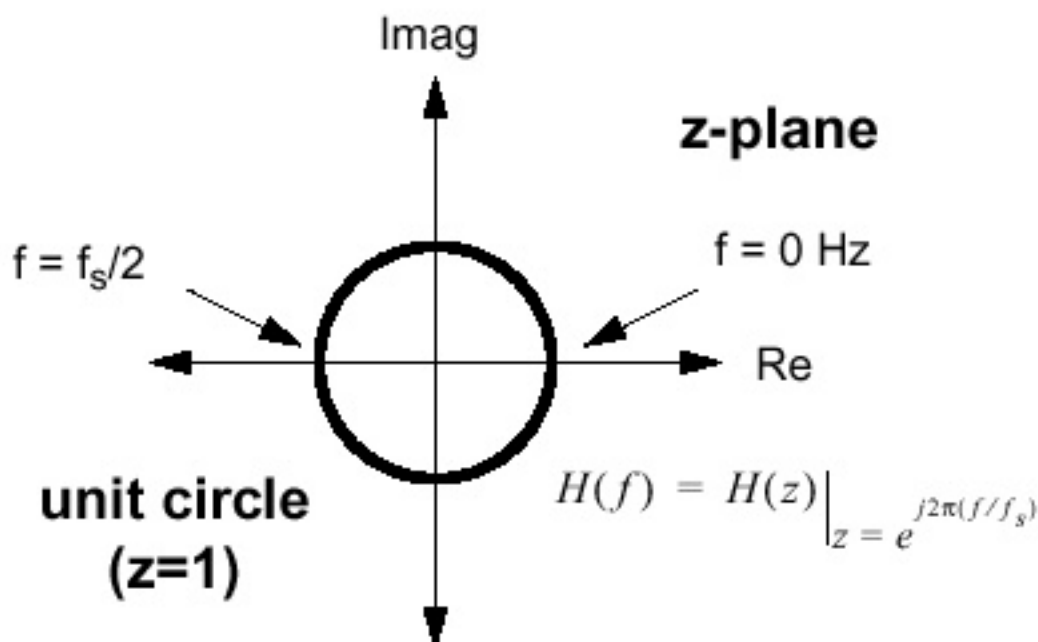
For a more detailed discussion of the z-transform, see [DSP notes](#).

DISCRETE FOURIER TRANSFORM

The Fourier transform of $x(n)$ can be computed from the z-transform as:

$$X(\omega) = X(z) \Big|_{z = e^{j\omega}} = \sum_{n=0}^{N-1} x(n) e^{-j\omega n}$$

The Fourier transform may be viewed as the time-limited (finit) z-transform evaluated around the unit circle:



The Discrete Fourier Transform (DFT) is defined as a sampled version of the (continuous)

Fourier transform shown above:

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N}, \quad k = 0, 1, 2, \dots, N-1$$

The inverse Discrete Fourier Transform (IDFT) is given by:

$$x(n) = \sum_{k=0}^{N-1} X(k)e^{j2\pi kn/N}, \quad n = 0, 1, 2, \dots, N-1$$

The DFT obeys the same properties one would expect for any **linear** transform (linearity, superposition, duality, etc.).

Note that these are not the only transforms used in speech processing (wavelets, Wigner distributions, fractals, etc.).

FAST FOURIER TRANSFORMS

The Fast Fourier Transform (FFT) is nothing more than a computationally efficient version of the Discrete Fourier Transform (DFT):

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N}, \quad k = 0, 1, 2, \dots, N-1$$

or,

$$X(k) = \sum_{n=0}^{N-1} x(n)W_N^{kn}, \quad k = 0, 1, 2, \dots, N-1$$

where $W_N = e^{-j2\pi/N}$ and $W_N^{kn} = e^{-j2\pi kn/N}$.

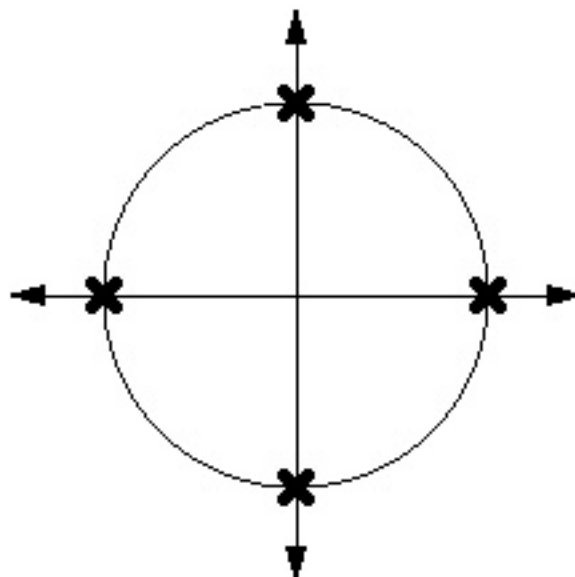
Note that W_N^{kn} are just samples on the unit circle:

For example, $W_4^{(3)(2)} = e^{(-j2\pi/4)(3)(2)} = e^{-j3\pi} = e^{-j\pi} = -1$.

$N = 4$

$k = 0, 1, 2, 3$

$n = 0, 1, 2, 3$



Only four unique values!

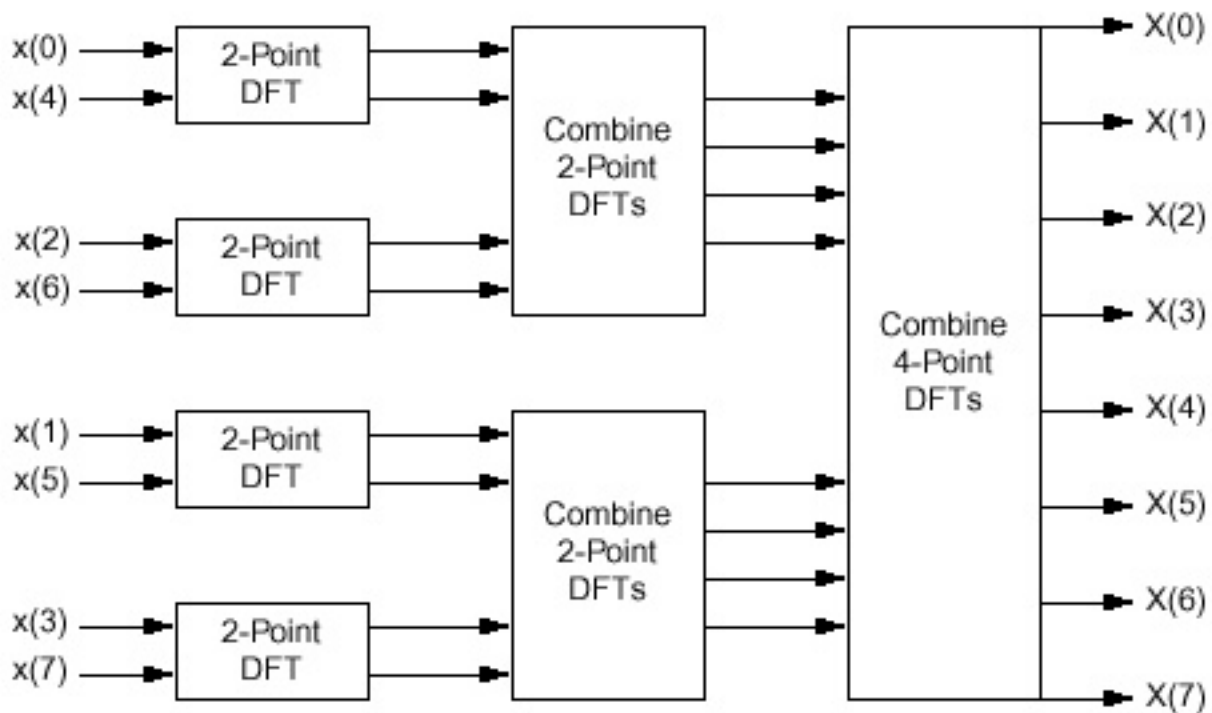
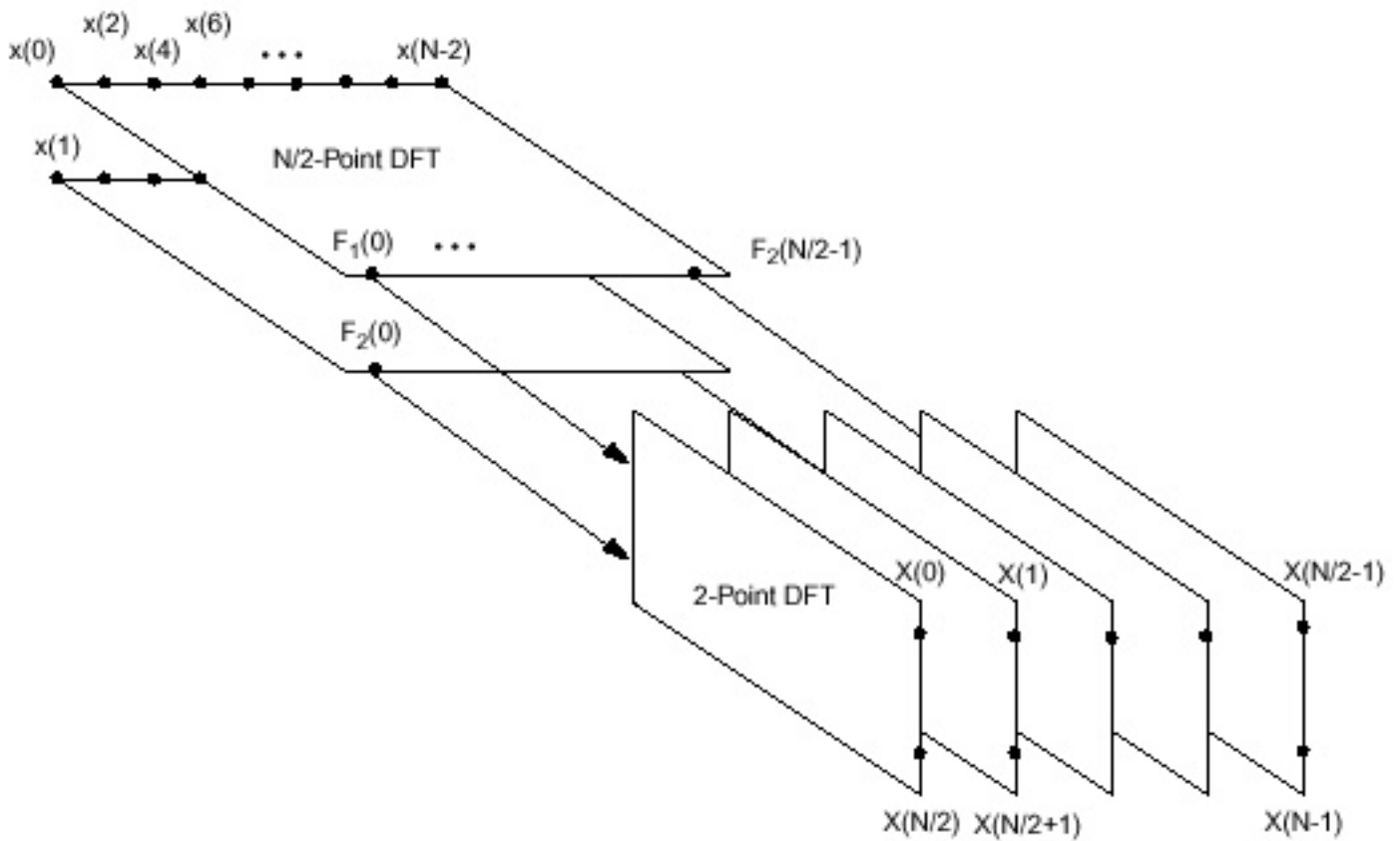
We note two important symmetry properties of W_N^{kn} :

$$W_N^{k+N/2} = -W_N^k \quad (\text{symmetry about the imaginary axis})$$

$$W_N^{k+N} = W_N^k \quad (\text{periodicity})$$

This symmetry allows the number of computations for a DFT to be reduced significantly.

The most common approach to achieving this efficiency is to use a decimation-in-time strategy that benefits from the non-linear computational complexity of the transform:



The Radix-2 and Radix-4 algorithms are extremely popular due to their computational efficiency and relatively simple implementations:



A definitive work on the computational

complexity of FFT algorithms, including benchmarks and software, can be found at [parallel FFTs](#).

DISCRETE COSINE TRANSFORMS

The Discrete Cosine Transform (DCT) is simply a computationally efficient version of the DFT for signals that are real and even ($x(n) = x(N-n)$): reduces to:

$$X(k) = \sum_{n=0}^{N-1} x(n) \cos(2\pi kn/N) \quad 0 \leq k < N$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) \cos(2\pi kn/N) \quad 0 \leq n < N$$

The DCT-II, which is one of two common implementations of the DCT used in speech processing, is defined as:

$$C(k) = \sum_{n=0}^{N-1} x(n) \cos(\pi k(n + 1/2)/N) \quad 0 \leq k < N$$

$$x(n) = \frac{1}{N} \left\{ C(0) + 2 \sum_{k=1}^{N-1} C(k) \cos(\pi k(n + 1/2)/N) \right\} \quad 0 \leq n < N$$

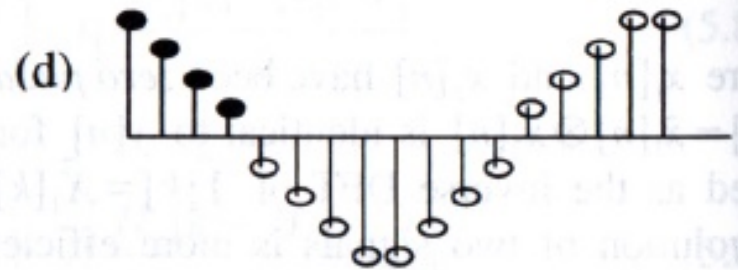
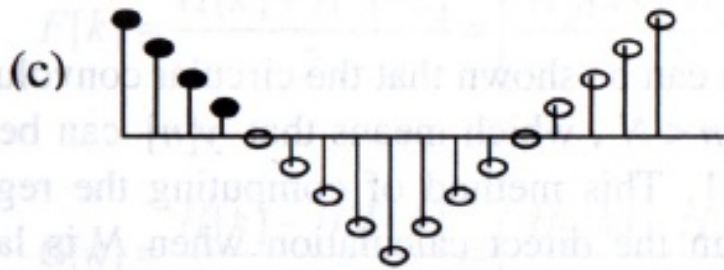
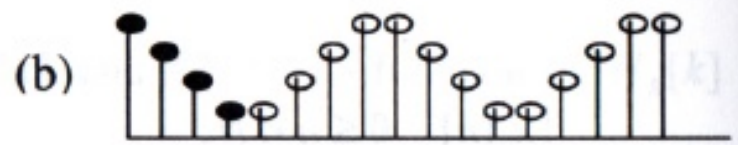
The DFT and DCT are related by the following equations:

$$\begin{aligned}X(k) &= 2e^{j\pi k/2N}C(k) & 0 \leq k < N \\X(2N-k) &= 2e^{-j\pi k/2N}C(k) & 0 \leq k < N\end{aligned}$$

The forward DCT is used in a speech recognition front end to convert samples of the log magnitude spectrum to cepstral coefficients.

PERIODIC EXTENSION AND THE DCT

There are four common ways to extend a real signal to make it both periodic and have even symmetry:

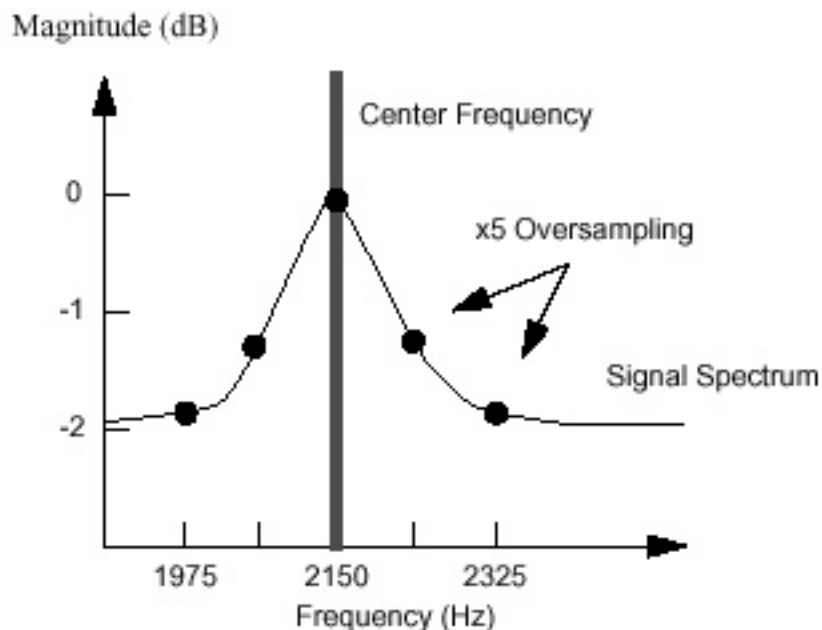


which correspond to DCT types I, II, III, and IV respectively.

Types II and III are most commonly used in speech processing because they tend to offer the most *energy compaction* resulting in representing the signal in the fewest number of coefficients.

OVERSAMPLING IMPROVES PERFORMANCE

The spectrum is oversampled to avoid biased estimates and to reduce variation in the measurements due to quantization of the frequency scale (for example, formants with narrow bandwidths):



For example, consider the parameters of a typical front end:

sample
frequency = 8 kHz

frame
duration = 10 msec

window = 25 msec (200
duration = points)

FFT length = 256 points

max sample
frequency = frequency / 2
= 4 kHz

$$\text{max mel frequency} = \frac{\text{max frequency in mel}}{\text{mel}} = 2146.1$$

$$\text{number of mel frequency scale bins} = 24 \text{ bins}$$

$$\text{mel frequency resolution} = \frac{\text{max mel frequency}}{(24 + 1)} = 85.84 \text{ mel}$$

$$\text{center frequency} = i * 85.84 \text{ mel}$$

This approach generates the table shown below:

| Bin # | Continuous Frequency | | | Discrete Frequency |
|-------|----------------------|--------------------|------------------|--------------------|
| | Start (Hz/Mel) | Center (Hz/Mel) | Stop (Hz/Mel) | Range (Index) |
| 1 | 0.0 0.0 | 55.4 85.8 | 115.2 171.7 | 0 - 3 |
| 2 | 55.4 85.8 | 115.2 171.7 | 179.7 257.5 | 2 - 5 |
| 3 | 115.2 171.7 | 179.7 257.5 | 249.3 343.4 | 4 - 7 |
| 4 | 179.7 257.5 | 249.3 343.4 | 324.5 429.2 | 6 - 10 |
| 5 | 249.3 343.4 | 324.5 429.2 | 405.5 515.1 | 8 - 12 |
| 6 | 324.5 429.2 | 405.5 515.1 | 493.0 600.9 | 11 - 15 |
| 7 | 405.5 515.1 | 493.0 600.9 | 587.5 686.7 | 13 - 18 |
| 8 | 493.0 600.9 | 587.5 686.7 | 689.4 772.6 | 16 - 22 |
| 9 | 587.5 686.7 | 689.4 772.6 | 799.3 858.4 | 19 - 25 |
| 10 | 689.4 772.6 | 799.3 858.4 | 918.0 944.3 | 23 - 29 |
| 11 | 799.3 858.4 | 918.0 944.3 | 1046.1 1030.1 | 26 - 33 |
| 12 | 918.0 944.3 | 1046.1 1030.1 | 1184.2 1116.0 | 30 - 37 |
| 13 | 1046.1 1030.1 | 1184.2 1116.0 | 1333.4 1201.8 | 34 - 42 |
| 14 | 1184.2 1116.0 | 1333.4 1201.8 | 1494.3 1287.6 | 38 - 47 |
| 15 | 1333.4 1201.8 | 1494.3 1287.6 | 1668.0 1373.5 | 43 - 53 |

| | | | | |
|----|------------------|------------------|------------------|-----------|
| 16 | 1494.3 1287.6 | 1668.0 1373.5 | 1855.4 1459.3 | 48 - 59 |
| 17 | 1668.0 1373.5 | 1855.4 1459.3 | 2057.6 1545.2 | 54 - 65 |
| 18 | 1855.4 1459.3 | 2057.6 1545.2 | 2275.9 1631.0 | 60 - 72 |
| 19 | 2057.6 1545.2 | 2275.9 1631.0 | 2511.4 1716.9 | 66 - 80 |
| 20 | 2275.9 1631.0 | 2511.4 1716.9 | 2765.6 1802.7 | 73 - 88 |
| 21 | 2511.4 1716.9 | 2765.6 1802.7 | 3039.9 1888.5 | 81 - 97 |
| 22 | 2765.6 1802.7 | 3039.9 1888.5 | 3335.9 1974.4 | 89 - 106 |
| 23 | 3039.9 1888.5 | 3335.9 1974.4 | 3655.3 2060.2 | 98 - 116 |
| 24 | 3335.9 1974.4 | 3655.3 2060.2 | 4000.0 2146.1 | 107 - 127 |

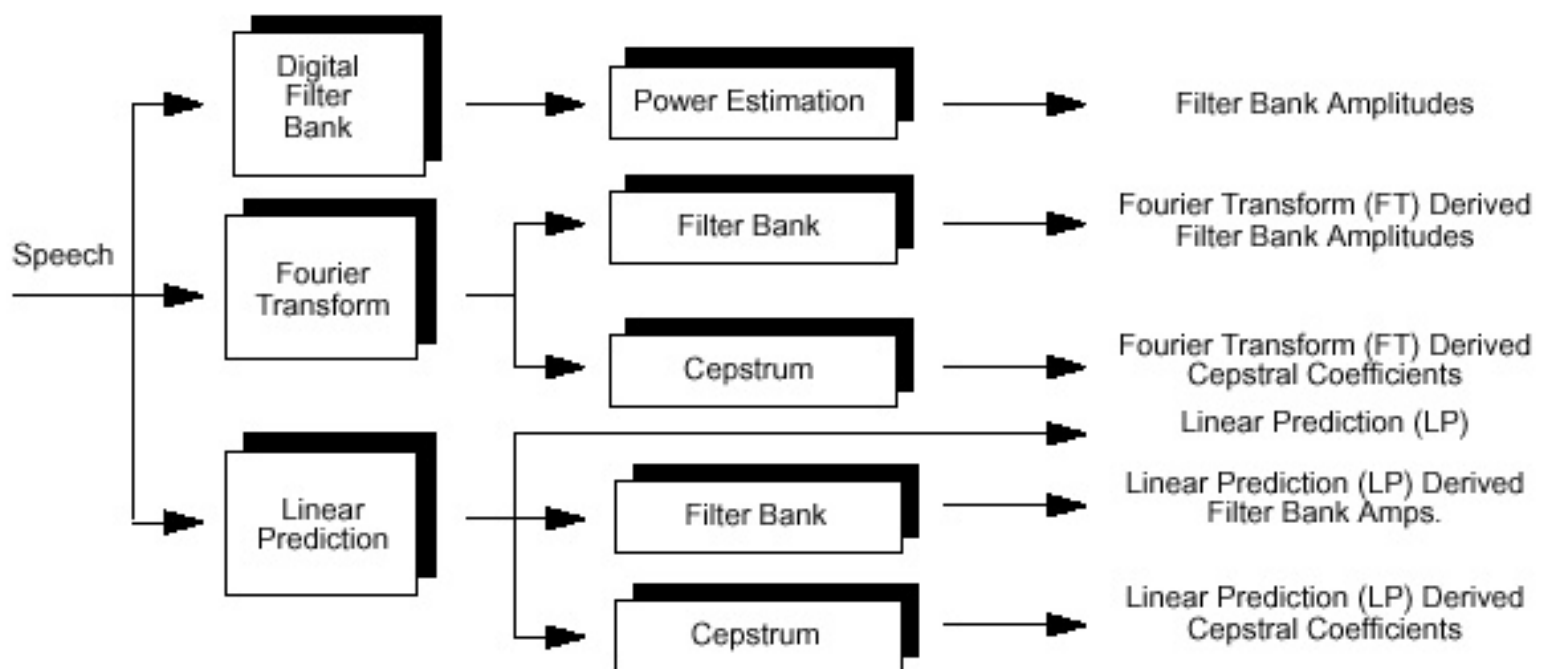
Finally, these 24 points are used to compute a forward DCT (extended to be a 48-point periodic and even sequence). The first 12 coefficients are retained.

The forward DCT is used because of its energy compaction property (a property shared by many orthogonal transforms). This transform allows us to approximate the data with fewer coefficients, since the coefficients are more concentrated at

lower indices. Hence, we truncate the representation to 12 coefficients and retain most of the important information, as well as ensure that the coefficients are orthogonal to one another.

ALTERNATIVE METHODS FOR FREQUENCY DOMAIN ANALYSIS

We have now established two different ways to perform a filterbank analysis of the speech signal (temporal and spectral):



The most popular front ends are those that use cepstral coefficients derived from the Fourier transform. Why?

IS PHASE IMPORTANT IN SPEECH RECOGNITION?

An FIR filter composed of all zeros that are inside the unit circle is minimum phase. There are many realizations of a system with a given magnitude response; one is a minimum phase realization, one is a maximum-phase realization, others are in-between. Any non-minimum phase pole-zero system can be decomposed into:

$$H(z) = H_{min}(z)H_{ap}(z)$$

It can be shown that of all the possible realizations of $|H(f)|$, the minimum-phase version is the most compact in time. Define:

$$E(n) = \sum_{k=0}^n |h(k)|^2$$

Then, $E_{min}(n) \geq E(n)$ for all n and all possible realizations of $|H(f)|$.

Why is minimum phase such an important concept in speech processing?

We prefer systems that are invertible:

$$H(z) H^{-1}(z) = 1$$

We would like both systems to be stable. The inverse of a non-minimum phase system is not stable.

We end with a very simple question: is phase important in speech recognition?

[Return to Main](#)[Objectives](#)**The Real Cepstrum:**[Homomorphic](#)[Definitions](#)[Pole-Zero](#)[Linear Prediction](#)**Applications:**[Vowel Cepstrum](#)[Source-Filter](#)[Frequency Warping](#)[Mel-Frequency](#)[Liftering](#)**Summary:**[Signal Modeling](#)[Typical Front End](#)**On-Line Resources:**[Cepstrum](#)[Hunt, ASRU'99 \(pdf\)](#)[Hunt, ASRU'99 \(ppt\)](#)[Ben Gold Oral History](#)

LECTURE 13: CEPSTRAL ANALYSIS

- Objectives:
 - Introduce homomorphic transformations
 - Understand the real cepstrum
 - Introduce alternate ways to compute the cepstrum
 - Explain how we compute mel-frequency "cepstrum" coefficients

This lecture combines material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

and information found in most standard DSP or speech textbooks:

J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co., ISBN: 0-7803-5386-2, 2000.

LECTURE 13: CEPSTRAL ANALYSIS

- Objectives:
 - Introduce homomorphic transformations
 - Understand the real cepstrum
 - Introduce alternate ways to compute the cepstrum
 - Explain how we compute mel-frequency "cepstrum" coefficients

This lecture combines material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory,*

Algorithm, and System Development, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

and information found in most standard DSP or speech textbooks:

J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co., ISBN: 0-7803-5386-2, 2000.

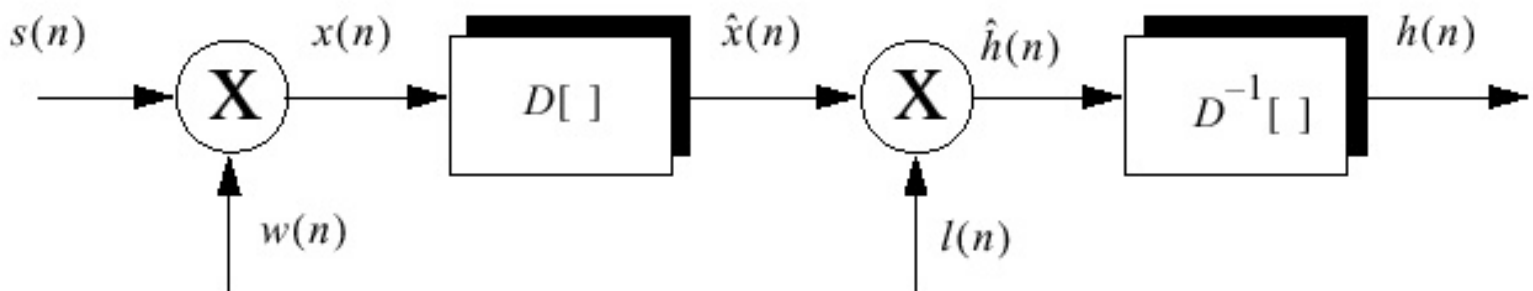
HOMOMORPHIC TRANSFORMATIONS

A *homomorphic* transformation converts a convolution into a sum:

$$x(n) = e(n) \otimes h(n)$$

$$\hat{x}(n) = \hat{e}(n) + \hat{h}(n)$$

Consider the problem of recovering a filter's response from a periodic signal (such as a voiced excitation):



The filter response can be recovered if we can separate the output of the homomorphic transformation using a simple filter:

$$l(n) = \begin{cases} 1 & |n| < N \\ 0 & |n| \geq N \end{cases}$$

Note that the process of separating the signals is

essentially a windowing processing. Is this useful for speech processing?

THE REAL AND COMPLEX CEPSTRUM

The *real* cepstrum of a digital signal $x(n)$ is defined as:

$$c(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \ln |X(\omega)| e^{j\omega n} d\omega$$

and the *complex cepstrum* is defined as:

$$\hat{x}(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \ln X(\omega) e^{j\omega n} d\omega$$

where the complex logarithm is used:

$$\begin{aligned}\hat{X}(\omega) &= \ln X(\omega) = \ln |X(\omega)| + j\theta(\omega) \\ \theta(\omega) &= \arg(X(\omega))\end{aligned}$$

Note that the real cepstrum, $c(n)$, is the even part of complex cepstrum:

$$c(n) = \frac{\hat{x}(n) + \hat{x}(-n)}{2}$$

The word *cepstrum* was coined by reversing the first syllable in the word *spectrum*. The cepstrum exists in a domain referred to as *quefrency* (reversal of the first syllable in *frequency*) which has units of time.

THE CEPSTRUM OF POLE-ZERO FILTERS

Consider a minimum phase system with a rational transfer function:

$$H(z) = \frac{\prod_{k=1}^Q (1 - b_k z^{-1})}{\prod_{k=1}^P (1 - a_k z^{-1})}$$

Taking the complex logarithm:

$$\hat{H}(z) = \sum_{k=1}^Q \log(1 - b_k z^{-1}) - \sum_{k=1}^P \log(1 - a_k z^{-1})$$

Taking an inverse z-transform:

$$\hat{h}(n) = \begin{cases} 0 & n \leq 0 \\ \sum_{k=1}^P a_k^n / n - \sum_{k=1}^Q b_k^n / n & n > 0 \end{cases}$$

It is easy to see that the cepstrum is a decaying function of time (compact). Why is this desirable?

Recalling that the real cepstrum can be computed from the even part of the complex cepstrum, the complex cepstrum can also be easily determined from the real cepstrum, $c(n)$, as follows:

$$\hat{h}(n) = \begin{cases} 0 & n < 0 \\ c(n) & n = 0 \\ 2c(n) & n > 0 \end{cases}$$

LINEAR PREDICTION AND THE CEPSTRUM

Consider an all-pole filter:

$$H(z) = \frac{G}{\prod_{k=1}^p (1 - a_k z^{-1})}$$

The cepstrum can be determined by the following recursion:

$$\hat{h}(n) = \begin{cases} 0 & n < 0 \\ \ln G & n = 0 \\ a_n + \sum_{k=1}^{n-1} \left(\frac{k}{n}\right) \hat{h}(k) a_{n-k} & 0 < n \leq p \\ \sum_{k=n-p}^{n-1} \left(\frac{k}{n}\right) \hat{h}(k) a_{n-k} & n > p \end{cases}$$

Note that if there are a finite number of filter

coefficients, there are still an infinite number of cepstral coefficients. However, the series decays to zero and can be truncated.

The proof of this result is shown below for completeness:

Given an all-pole filter:

$$H(z) = \frac{G}{\prod_{l=1}^P (1 - a_l z^{-1})}$$

We can take the complex logarithm:

$$\hat{H}(z) = \ln G - \ln \left(\prod_{l=1}^P (1 - a_l z^{-1}) \right) \equiv \sum_{k=-\infty}^{\infty} \hat{h}(k) z^{-k}$$

Taking the derivative of both sides with respect to z :

$$\frac{-\sum_{n=1}^P n a_n z^{-n-1}}{1 - \sum_{l=1}^P a_l z^{-l}} = - \sum_{k=-\infty}^{\infty} k \hat{h}(k) z^{-k-1}$$

Multiplying both sides by $-z \left(\prod_{l=1}^P (1 - a_l z^{-1}) \right)$, we obtain:

$$\sum_{n=1}^P n a_n z^{-n} = \sum_{n=-\infty}^{\infty} n \hat{h}(k) z^{-n} - \sum_{l=1}^P \sum_{k=-\infty}^{\infty} k \hat{h}(k) a_l z^{-k-l}$$

After replacing $l = n - k$, and equating terms in z^{-n} , we obtain:

$$na_n = n\hat{h}(n) - \sum_{k=1}^{n-1} k\hat{h}(k)a_{n-k} \quad 0 < n \leq p$$

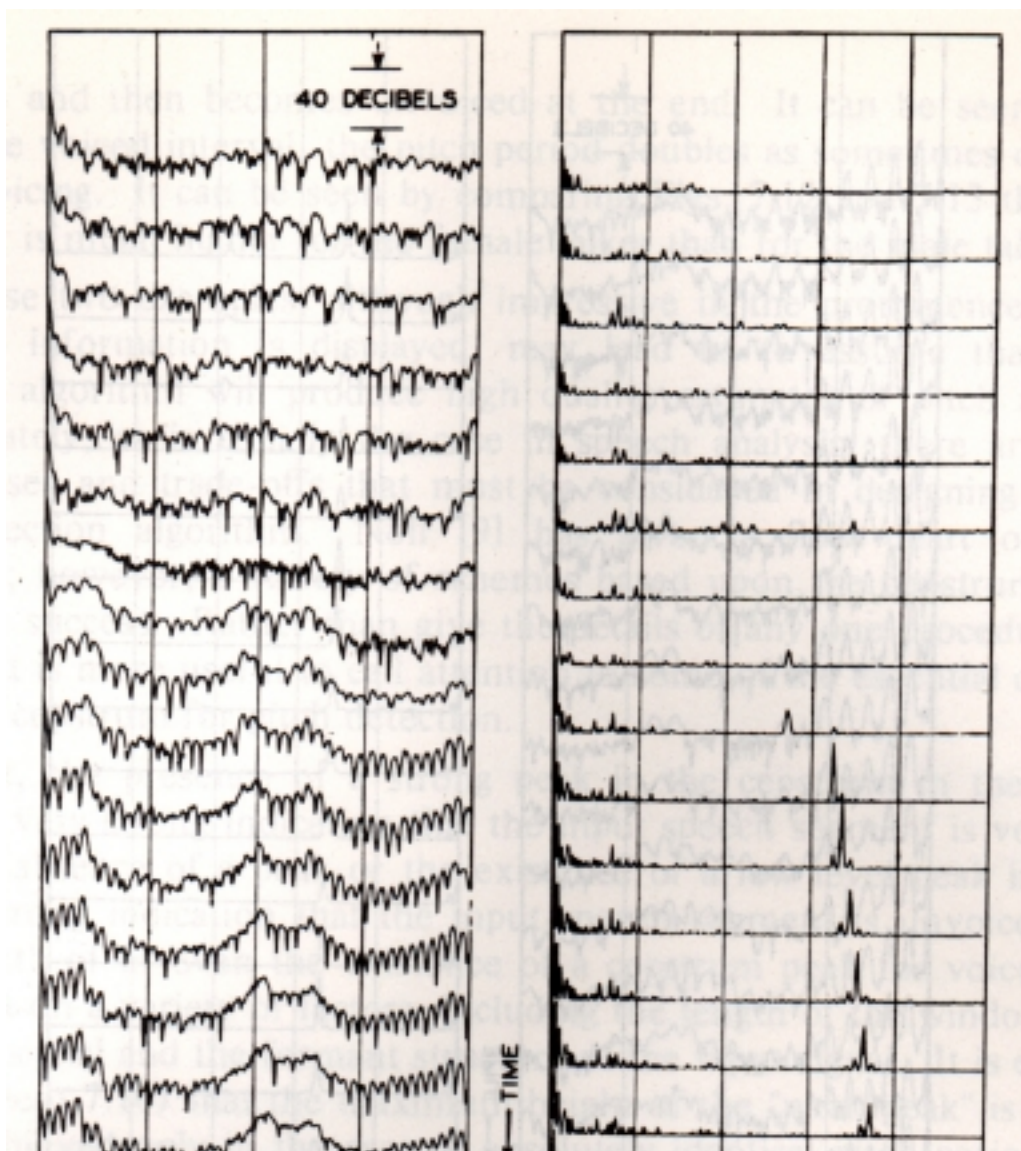
$$0 = n\hat{h}(n) - \sum_{k=n-p}^{n-1} k\hat{h}(k)a_{n-k} \quad n > p$$

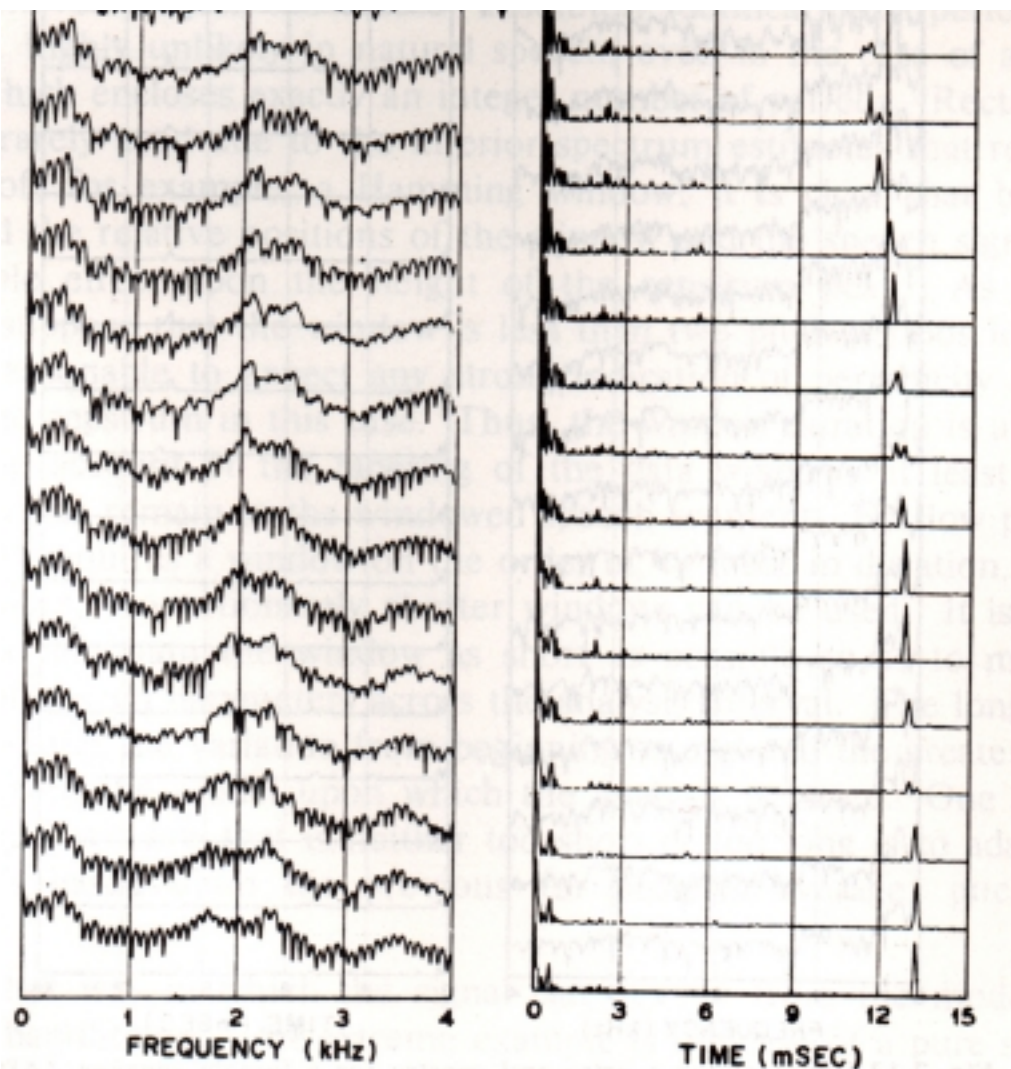
Hence, the complex cepstral cepstrum can be obtained directly from the all-pole filter coefficients:

$$\hat{h}(n) = \begin{cases} 0 & n < 0 \\ \ln G & n = 0 \\ a_n + \sum_{k=1}^{n-1} \left(\frac{k}{n}\right) \hat{h}(k) a_{n-k} & 0 < n \leq p \\ \sum_{k=n-p}^{n-1} \left(\frac{k}{n}\right) \hat{h}(k) a_{n-k} & n > p \end{cases}$$

AN EXAMPLE OF THE CEPSTRUM FOR A VOWEL

Below is an example (from Noll) that demonstrates a typical cepstrum sequence for a vowel. The cepstrum is computed every 10 msec.





From this example, we can see two important things:

- At the onset of the vowel, where the signal is not quite periodic, the peak in the cepstrum at the fundamental frequency is not well-formed. The amplitude of this peak grows as the signal becomes more regular

(periodic). The same phenomena is true for the autocorrelation function.

- It is clear that the low order coefficients of the cepstrum contain information about the vocal tract, while the higher order coefficients contain primarily information about the excitation. (Actually, the higher order coefficients contain both types of information, but the frequency of periodicity dominates.)

Hence, for speech signals, it seems the vocal tract response and the excitation signal can be separated using simple windowing in the frequency domain.

SOURCE-FILTER SEPARATION VIA THE CEPSTRUM

An example of source-filter separation using voiced speech:

(a)

Windowed
Signal

(b) Log

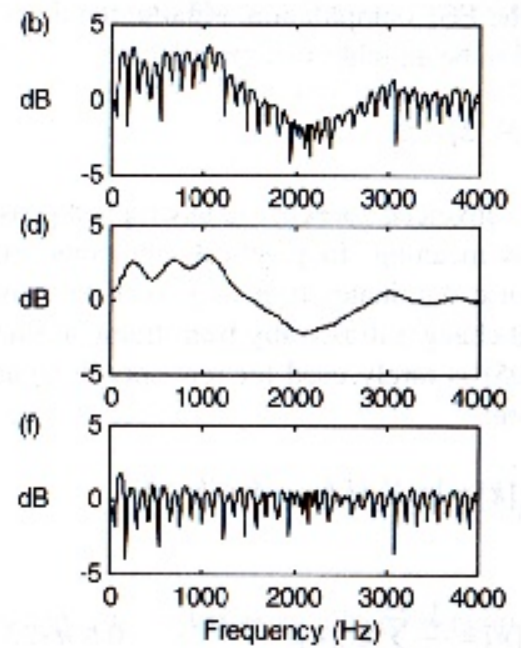
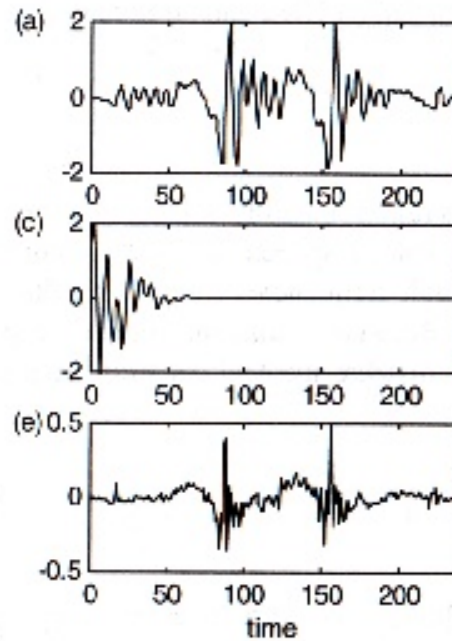
Spectrum

(c)

Filtered
Cepstrum

$(n < N)$

(d) Smoothed Log Spectrum



(e) Excitation Signal (f) Log Spectrum (high freq.)

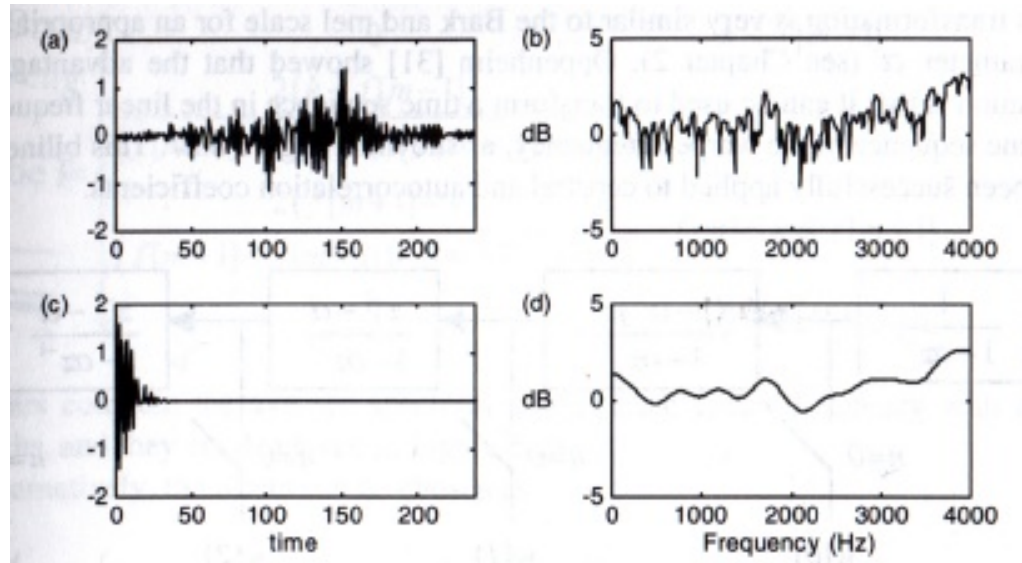
An example of source-filter separation using unvoiced speech:

(a)
Windowed
Signal

(b) Log
Spectrum

(c) Filtered
Cepstrum
($n < N$)

(d)
Smoothed
Log



Spectrum

The reason this works is simple: the fundamental frequency for the speaker produces a peak in the cepstrum sequence that is far removed ($n > N$) from the influence of the vocal tract ($n < N$). You can also demonstrate this using an autocorrelation function. What happens for an extremely high-pitched female or child?

FREQUENCY WARPING USING ALL-PASS TRANSFORMATIONS

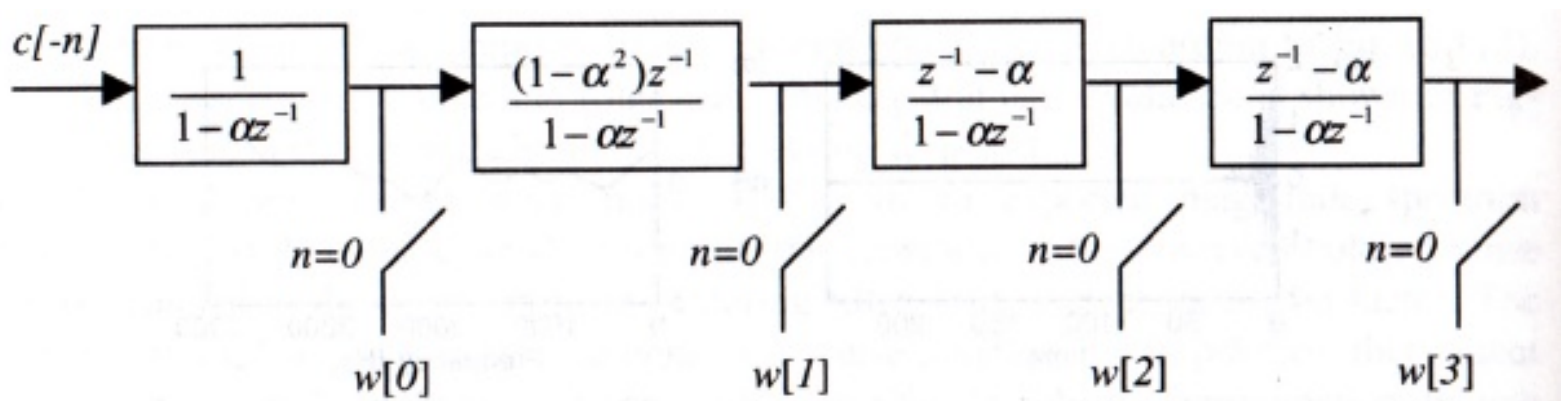
Recall the bilinear transform:

$$s = \frac{z^{-1} - \alpha}{1 - \alpha z^{-1}}$$

which implements a nonlinear warping of the frequency axis:

$$\Omega = \omega + 2 \arctan\left(\frac{\alpha \sin(\omega)}{1 - \alpha \cos(\omega)}\right)$$

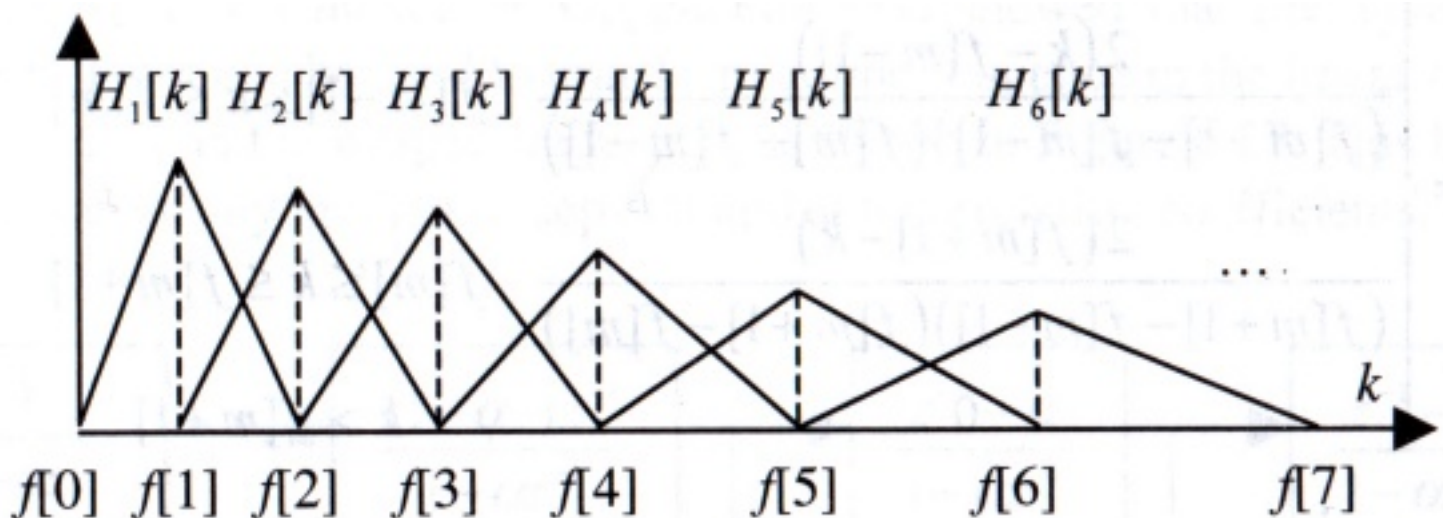
This can be implemented as a series of all-pass transformations:



The cepstral coefficients are input and the output are frequency-warped cepstral coefficients. This is an interesting way to implement speaker-specific warpings (e.g., male vs. female speakers).

MEL-FREQUENCY CEPSTRUM

Recall our filterbank, which we construct in mel-frequency domain using a triangularly-shaped weighting function applied to mel-transformed log-magnitude spectral samples:



After computing the DFT, and the log magnitude spectrum (to obtain the real cepstrum), we compute the filterbank outputs, and then use a discrete cosine transform:

$$c(n) = \sum_{m=0}^{M-1} S(m) \cos\left(\pi n \left(m - \frac{1}{2}\right) / M\right)$$

where

$$S(m) = \ln \left(\sum_{k=0}^{N-1} |X_a(k)|^2 H_m(k) \right), \quad 0 < m \leq M$$

to compute the *mel-frequency cepstrum coefficients*. Note that the triangular weighting functions are applied directly to the magnitude spectrum, and then the logarithm is taken after the spectral samples are averaged. The resulting coefficients are an approximation to the cepstrum, and in reality simply represent an orthogonal and compact representation of the log magnitude spectrum.

We typically use [24 filterbank samples](#) at an 8 kHz sampling frequency, and truncate the DCT to 12 MFCC coefficients. Adding energy gives

us a total of 13 coefficients for our base feature vector.

LIFTERING: WINDOWING CEPSTRAL COEFFICIENTS

- Low order cepstral coefficients are sensitive to spectral slope, glottal pulse shape, etc.
- High order cepstral coefficients are sensitive to the analysis window position and other temporal artifacts.
- For speaker independent recognition, it is best to minimize such speaker-dependent variations in the features prior to recognition.
- We can reduce the variations in these coefficients by using a raised sine window that emphasizes coefficients at the center of the window:

$$w(n) = G \left\{ \begin{array}{ll} 1 + h \sin((n\pi)/L) & 1 \leq n \leq L \\ 0 & \text{elsewhere} \end{array} \right\}$$

- L is the number of cepstral coefficients (typically 24), and G is a constant normally designed to make the energy of the window equal to 1.

[Return to Main](#)

[Home](#)

[Exam Database](#)

[First Exam](#)

LECTURE 14: EXAM NO. 1

The first exam can be found [here](#).

LECTURE 14: EXAM NO. 1

The first exam can be found [here](#).

[Return to Main](#)

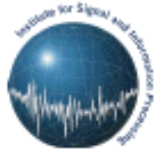
LECTURE 14: EXAM NO. 1

[Home](#)

The first exam can be found [here](#).

[Exam Database](#)

[First Exam](#)



Introduction:

- 01: Organization
([html](#), [pdf](#))

Speech Signals:

- 02: Production
([html](#), [pdf](#))
- 03: Digital Models
([html](#), [pdf](#))
- 04: Perception
([html](#), [pdf](#))
- 05: Masking
([html](#), [pdf](#))
- 06: Phonetics and Phonology
([html](#), [pdf](#))
- 07: Syntax and Semantics
([html](#), [pdf](#))

Signal Processing:

- 08: Sampling
([html](#), [pdf](#))
- 09: Resampling
([html](#), [pdf](#))
- 10: Acoustic Transducers
([html](#), [pdf](#))
- 11: Temporal Analysis
([html](#), [pdf](#))
- 12: Frequency Domain Analysis
([html](#), [pdf](#))
- 13: Cepstral Analysis
([html](#), [pdf](#))
- 14: **Exam No. 1**
([html](#), [pdf](#))
- 15: Linear Prediction
([html](#), [pdf](#))
- 16: LP-Based Representations
([html](#), [pdf](#))

Parameterization:

- 17: Differentiation
([html](#), [pdf](#))
- 18: Principal Components
([html](#), [pdf](#))

ECE 8463: FUNDAMENTALS OF SPEECH RECOGNITION

Professor Joseph Picone
Department of Electrical and Computer Engineering
Mississippi State University

email: picone@isip.msstate.edu
phone/fax: 601-325-3149; office: 413 Simrall
URL: http://www.isip.msstate.edu/resources/courses/ece_8463

Modern speech understanding systems merge interdisciplinary technologies from Signal Processing, Pattern Recognition, Natural Language, and Linguistics into a unified statistical framework. These systems, which have applications in a wide range of signal processing problems, represent a revolution in Digital Signal Processing (DSP). Once a field dominated by vector-oriented processors and linear algebra-based mathematics, the current generation of DSP-based systems rely on sophisticated statistical models implemented using a complex software paradigm. Such systems are now capable of understanding continuous speech input for vocabularies of hundreds of thousands of words in operational environments.

In this course, we will explore the core components of modern statistically-based speech recognition systems. We will view speech recognition problem in terms of three tasks: signal modeling, network searching, and language understanding. We will conclude our discussion with an overview of state-of-the-art systems, and a review of available resources to support further research and technology development.

Tar files containing a compilation of all the notes are available. However, these files are large and will require a substantial amount of time to download. A tar file of the html version of the notes is available [here](#). These were generated using wget:

```
wget -np -k -m http://www.isip.msstate.edu/publications/courses/ece_8463/lectures/current
```




A pdf file containing the entire set of lecture notes is available [here](#). These were generated using Adobe Acrobat.

Questions or comments about the material presented here can be directed to help@isip.msstate.edu.

LECTURE 14: EXAM NO. 1

The first exam can be found [here](#).




Index of /publications/courses/ece_8463/exams

| Name | Last modified | Size | Description |
|--|-------------------|------|-------------|
|  Parent Directory | 09-Feb-2002 08:26 | - | |
|  2002_spring/ | 08-Feb-2002 15:53 | - | |
|  current/ | 08-Feb-2002 15:53 | - | |

Apache/1.3.9 Server at www.isip.msstate.edu Port 80

Index of




/publications/courses/ece_8463/exams/current

| | Name | Last modified | Size | Description |
|---|----------------------------------|-------------------------------|----------------------|-----------------------------|
|  | Parent Directory | 08-Feb-2002 16:03 | - | |
|  | sp02_01.fm | 08-Feb-2002 15:53 | 23k | |
|  | sp02_01.pdf | 08-Feb-2002 15:53 | 26k | |

Apache/1.3.9 Server at www.isip.msstate.edu Port 80

Index of

/publications/courses/ece_8463/lectures/current/lecture_14/exams

| Name | | Last modified | Size | Description |
|---|----------------------------------|-------------------------------|----------------------|-----------------------------|
| <hr/> | | | | |
|  | Parent Directory | 08-Feb-2002 16:03 | - | |
|  | sp02_01.fm | 08-Feb-2002 15:53 | 23k | |
|  | sp02_01.pdf | 08-Feb-2002 15:53 | 26k | |

Number:

| Problem | Points | Score |
|---------|--------|-------|
| 1 (a) | 10 | |
| 1 (b) | 10 | |
| 1 (c) | 10 | |
| 1 (d) | 10 | |
| 2 (a) | 10 | |
| 2 (b) | 10 | |
| 2 (c) | 10 | |
| 3 (a) | 10 | |
| 3 (b) | 10 | |
| 3 (c) | 10 | |
| Total | 100 | |

Notes:

1. The exam is closed books and notes. You are allowed one 8 1/2" x 11" double-sided sheet of notes.
2. Please indicate clearly your answer to the problem by some form of highlighting (underlining).
3. Your solutions must be legible and easy to follow. If I can't read it or understand it, it is wrong. Random scribbling will not receive credit.
4. Please show ALL work. Answers with no supporting explanations or work will be given no credit.
5. Several problems on this exam are fairly open-ended. Since the evaluation of your answers is obviously a subjective process, we will use a market place strategy in determining the grade. Papers will be rank-ordered in terms of the quality of the solutions, and grades distributed accordingly.

1. Deep sea divers breath a mixture of air and helium called Heliox to avoid several problems associated with breathing compressed air under water. Heliox is lighter than air — its density is 75% lower than air. Speech produced while breathing Heliox sounds distorted (a classroom demonstration is provided).
 - (a) Predict the effect breathing Heliox has on the formant frequencies. Justify this answer using our linear acoustics model.

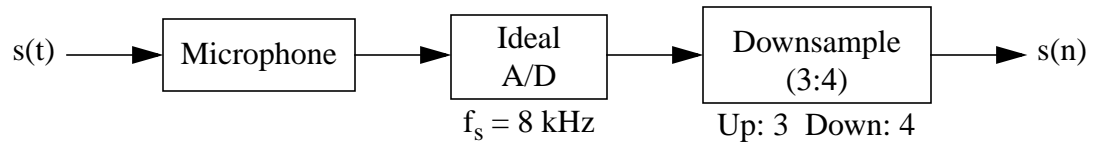
(b) Does the excitation signal change? Explain.

(c) For what value of the density (relative to air) would the speech become unintelligible to a human listener?

(d) Design a system that would descramble the diver's speech and produce a normal sounding speech signal. Is such a system physically realizable?

2. Consider a language which has a phoneme set that only contains four English consonants: *b*, *p*, *d*, *t*.
- (a) Describe the similarities and differences between these sounds in as much linguistic detail as possible.
- (b) Do the assumptions we make to justify frame-based processing in speech recognition of spoken English hold for this type of signal? Explain.
- (c) Consider a voiced sound in this language produced with a fundamental frequency of 100 Hz. Would a listener perceive the 5th and 6th harmonics to be closer in frequency than the 20th and 21st harmonics? Explain.

3. Consider the system shown below:



(a) Suppose the microphone can be modeled by this equation: $y(t) = \alpha x(t) + \beta x^2(t)$. Typically, $\alpha > \beta$. Sketch the spectrum of the output signal, $s(n)$, over the frequency range $[0, 8 \text{ kHz}]$ for an input that consists of a sinewave at 1 kHz.

(b) Sketch the spectrum of the output signal for a white noise input (flat spectrum). Assume both the A/D and the downsampler use ideal low pass filters.

- (c) Suppose the input signal is the sum of a 1 kHz and 1.5 kHz sinewave. Sketch the spectrum of the output signal. Explain the influence of the microphone on this result. What aspect of the microphone would you improve? Why?

[Return to Main](#)[Objectives](#)**Basic Theory:**[Simple Case](#)[General Case](#)[Covariance](#)[Autocorrelation](#)[Error](#)**Spectral Matching:**[Spectrum](#)[Noise-Weighting](#)[Preemphasis](#)[Signal Modeling](#)[Typical Front End](#)**On-Line Resources:**[Signal Modeling](#)[Linear Prediction](#)[MAD Speech](#)

LECTURE 15: LINEAR PREDICTION

- Objectives:
 - Introduce the theory of linear prediction
 - Develop autocorrelation and covariance techniques for solution
 - Understand similarities with regression
 - Explain the relationship to windowing and maximum entropy

- Add a new technique to our signal modeling block diagram

There is a classic textbook on this subject:

J.D. Markel and A.H. Gray,
Linear Prediction of Speech,
Springer-Verlag, New York, New
York, USA, ISBN:
0-13-007444-6, 1976.

This lecture also includes material from two other textbooks:

J. Deller, et. al., *Discrete-Time Processing of Speech Signals*,
MacMillan Publishing Co., ISBN:
0-7803-5386-2, 2000.

and,

L.R. Rabiner and B.W. Juang,

*Fundamentals of Speech
Recognition*, Prentice-Hall, Upper
Saddle River, New Jersey, USA,
ISBN: 0-13-015157-2, 1993.

LECTURE 15: LINEAR PREDICTION

- Objectives:
 - Introduce the theory of linear prediction
 - Develop autocorrelation and covariance techniques for solution
 - Understand similarities with regression
 - Explain the relationship to windowing and maximum entropy
 - Add a new technique to our signal modeling block diagram

There is a classic textbook on this subject:

J.D. Markel and A.H. Gray, *Linear Prediction of Speech*, Springer-Verlag, New York, New York, USA, ISBN: 0-13-007444-6, 1976.

This lecture also includes material from two other textbooks:

J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co., ISBN: 0-7803-5386-2, 2000.

and,

L.R. Rabiner and B.W. Juang, *Fundamentals of Speech Recognition*, Prentice-Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-015157-2, 1993.

LINEAR PREDICTION - A SIMPLE DERIVATION

Let us define a speech signal, $s(n)$. Consider the problem of predicting the current value from the previous value, $\tilde{s}(n) = \alpha s(n-1)$. This prediction will be in error by some amount:

$$e(n) = s(n) - \tilde{s}(n) = s(n) - \alpha s(n-1)$$

We would like to minimize the error by finding the best, or optimal, value of α . Let us define the short-time average prediction error:

$$\begin{aligned} E &= \sum_n e^2(n) \\ &= \sum_n \{s(n) - \alpha s(n-1)\}^2 \end{aligned}$$

We can minimize the error w.r.t α by differentiating E and setting the result equal to zero:

$$E = \sum_n (s^2(n) - 2\alpha s(n)s(n-1) + \alpha^2 s^2(n-1))$$

Differentiating w.r.t α ,

$$\frac{\partial E}{\partial \alpha} = 0 = \sum_n -2s(n)s(n-1) + 2\alpha s^2(n-1)$$

or,

$$\sum_n s(n)s(n-1) = \alpha \sum_n s^2(n-1)$$

which implies:

$$\alpha = \frac{\sum_n s(n)s(n-1)}{\sum_n s^2(n-1)} = \frac{c(1, 0)}{c(1, 1)} \approx \frac{r(1)}{r(0)}$$

Notes:

- related to the correlation structure of the signal ($\alpha < 1$)

- independent of the energy level of the signal

What short-term technique do we use to compute correlation/covariance?

LINEAR PREDICTION - GENERAL CASE

Let us define a speech signal, $s(n)$, and a predicted value: $\tilde{s}(n) = \sum_{k=1}^p \alpha_k s(n-k)$.

Why the added terms? This prediction error is given by:

$$e(n) = s(n) - \tilde{s}(n) = s(n) - \sum_{k=1}^p \alpha_k s(n-k)$$

We would like to minimize the error by finding the best, or optimal, value of $\{\alpha_k\}$.

Let us define the short-time average prediction error:

$$\begin{aligned} E &= \sum_n e^2(n) \\ &= \sum_n \left\{ s(n) - \sum_{k=1}^p \alpha_k s(n-k) \right\}^2 \\ &= \sum_n s^2(n) - \sum_n \left\{ 2s(n) \sum_{k=1}^p \alpha_k s(n-k) \right\} + \sum_n \left\{ \sum_{k=1}^p \alpha_k s(n-k) \right\}^2 \\ &= \sum_n s^2(n) - 2 \sum_{k=1}^p \alpha_k \sum_n s(n)s(n-k) + \sum_n \left\{ \sum_{k=1}^p \alpha_k s(n-k) \right\}^2 \end{aligned}$$

We can minimize the error w.r.t α_l for each $1 \leq l \leq p$ by differentiating E and setting the result equal to zero:

$$\frac{\partial E}{\partial \alpha_l} = 0 = -2 \sum_n s(n)s(n-l) + 2 \sum_n \left\{ \sum_{k=1}^p \alpha_k s(n-k) \right\} s(n-l)$$

THE COVARIANCE METHOD

Rearranging terms:

$$\sum_n s(n)s(n-l) = \sum_{k=1}^p \alpha_k \left(\sum_n s(n-k)s(n-l) \right)$$

or,

$$c(l, 0) = \sum_{k=1}^p \alpha_k c(k, l)$$

This equation is known as the linear prediction (Yule-Walker) equation. $\{\alpha_k\}$ are known as linear prediction coefficients, or *predictor coefficients*.

By enumerating the equations for each value of l , we can express this in matrix form:

$$\bar{c} = \underline{C} \bar{\alpha}$$

where,

$$\bar{\alpha} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \dots \\ \alpha_p \end{bmatrix} \quad \underline{C} = \begin{bmatrix} c(1, 1) & c(1, 2) & \dots & c(1, p) \\ c(2, 1) & c(2, 2) & \dots & c(2, p) \\ \dots & \dots & \dots & \dots \\ c(p, 1) & c(p, 2) & \dots & c(p, p) \end{bmatrix} \quad \bar{c} = \begin{bmatrix} c(1, 0) \\ c(2, 0) \\ \dots \\ c(p, 0) \end{bmatrix}$$

The solution to this equation involves a matrix inversion:

$$\bar{\alpha} = \underline{C}^{-1} \bar{c}$$

and is known as the *covariance method*. Under what conditions does \underline{C}^{-1} exist?

Note that the covariance matrix is symmetric. A fast algorithm to find the solution to this equation is known as the Cholesky decomposition (a $\underline{V} \underline{D} \underline{V}^T$ approach in which the covariance matrix is factored into lower and upper triangular matrices).

THE AUTOCORRELATION METHOD

Using a different interpretation of the limits on the error minimization — forcing data only within the frame to be used — we can compute the solution to the linear prediction equation using the autocorrelation method:

$$\bar{\alpha} = \underline{R}^{-1} \bar{r}$$

where,

$$\bar{\alpha} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \dots \\ \alpha_p \end{bmatrix} \quad \underline{R} = \begin{bmatrix} r(0) & r(1) & \dots & r(p-1) \\ r(1) & r(0) & \dots & r(p-2) \\ \dots & \dots & \dots & \dots \\ r(p-1) & r(p-2) & \dots & r(0) \end{bmatrix} \quad \bar{r} = \begin{bmatrix} r(1) \\ r(2) \\ \dots \\ r(p) \end{bmatrix}$$

Note that \underline{R} is symmetric, and all of the elements along the diagonal are equal, which means (1) an inverse always exists; (2) the roots are in the left-half plane.

The linear prediction process can be viewed as a filter by noting:

$$e(n) = s(n) - \sum_{k=1}^p \alpha_k s(n-k)$$

and

$$E(z) = S(z)A(z)$$

where

$$A(z) = 1 - \sum_{k=1}^p \alpha_k z^{-k}$$

$A(z)$ is called the analyzer; what type of filter is it? (pole/zero? phase?)

$\frac{1}{A(z)}$ is called the synthesizer; under what conditions is it stable?

LINEAR PREDICTION ERROR

We can return to our expression for the error:

$$E = \sum_n e^2(n) = \sum_n \left\{ s(n) - \sum_{k=1}^p \alpha_k s(n-k) \right\}^2$$

and substitute our expression for $\{\alpha_k\}$ and show that:

Autocorrelation Method:

$$E = r(0) - \sum_{k=1}^p \alpha_k r(k)$$

Covariance Method:

$$E = c(0, 0) - \sum_{k=1}^p \alpha_k c(0, k)$$

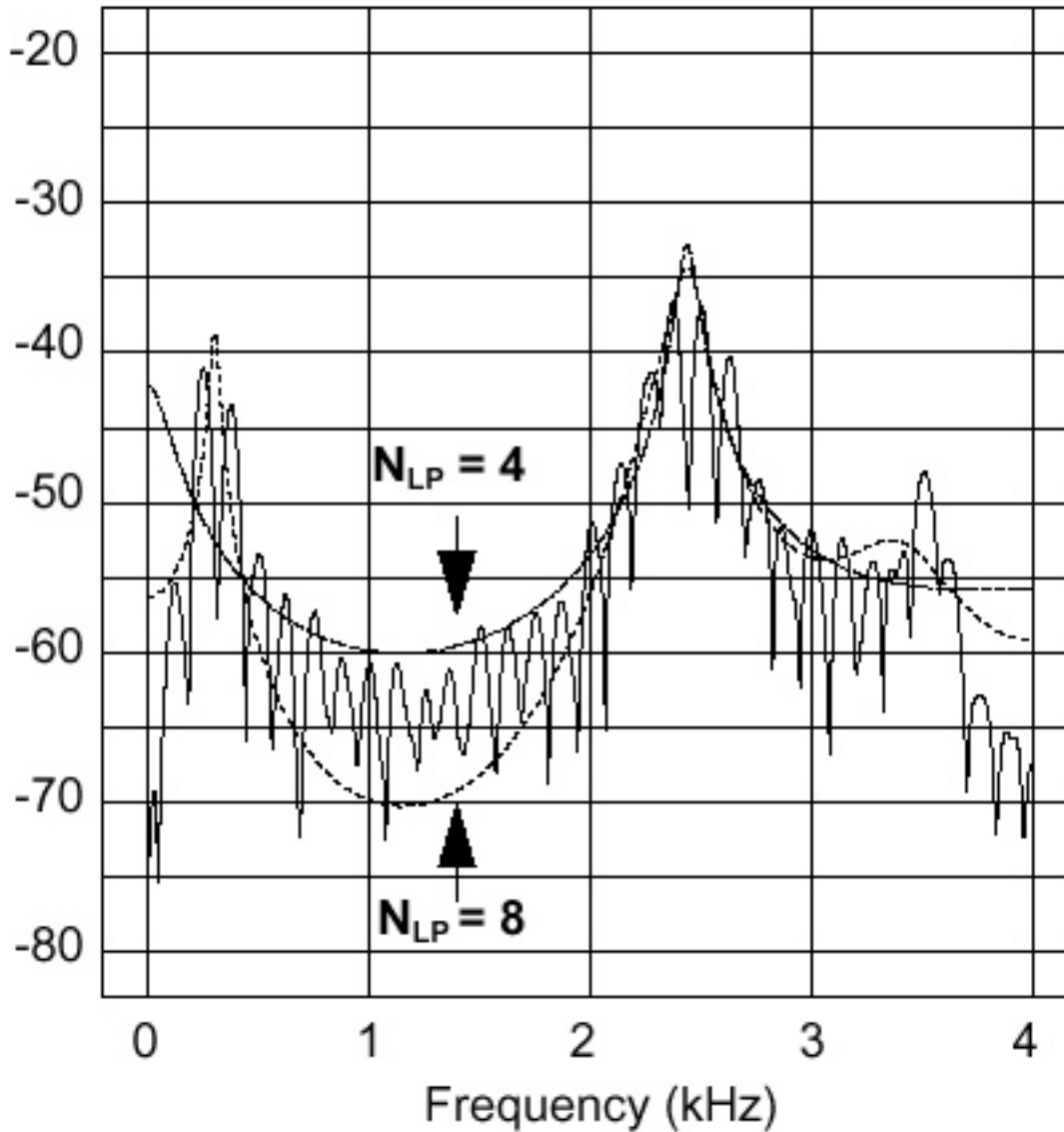
Later, we will discuss the properties of these equations as they relate to the magnitude of the error. For now, note that the same linear prediction equation that applied to the signal applies to the autocorrelation function, except that samples are replaced by the autocorrelation lag (and hence the delay term is replaced by a lag index).

Since the same coefficients satisfy both equations, this confirms our hypothesis that this is a model of the minimum-phase version of the input signal.

Linear prediction has numerous formulations including the covariance method, autocorrelation formulation, lattice method, inverse filter formulation, spectral estimation formulation, maximum likelihood formulation, and inner product formulation. Discussions are found in disciplines ranging from system identification, econometrics, signal processing, probability, statistical mechanics, and operations research.

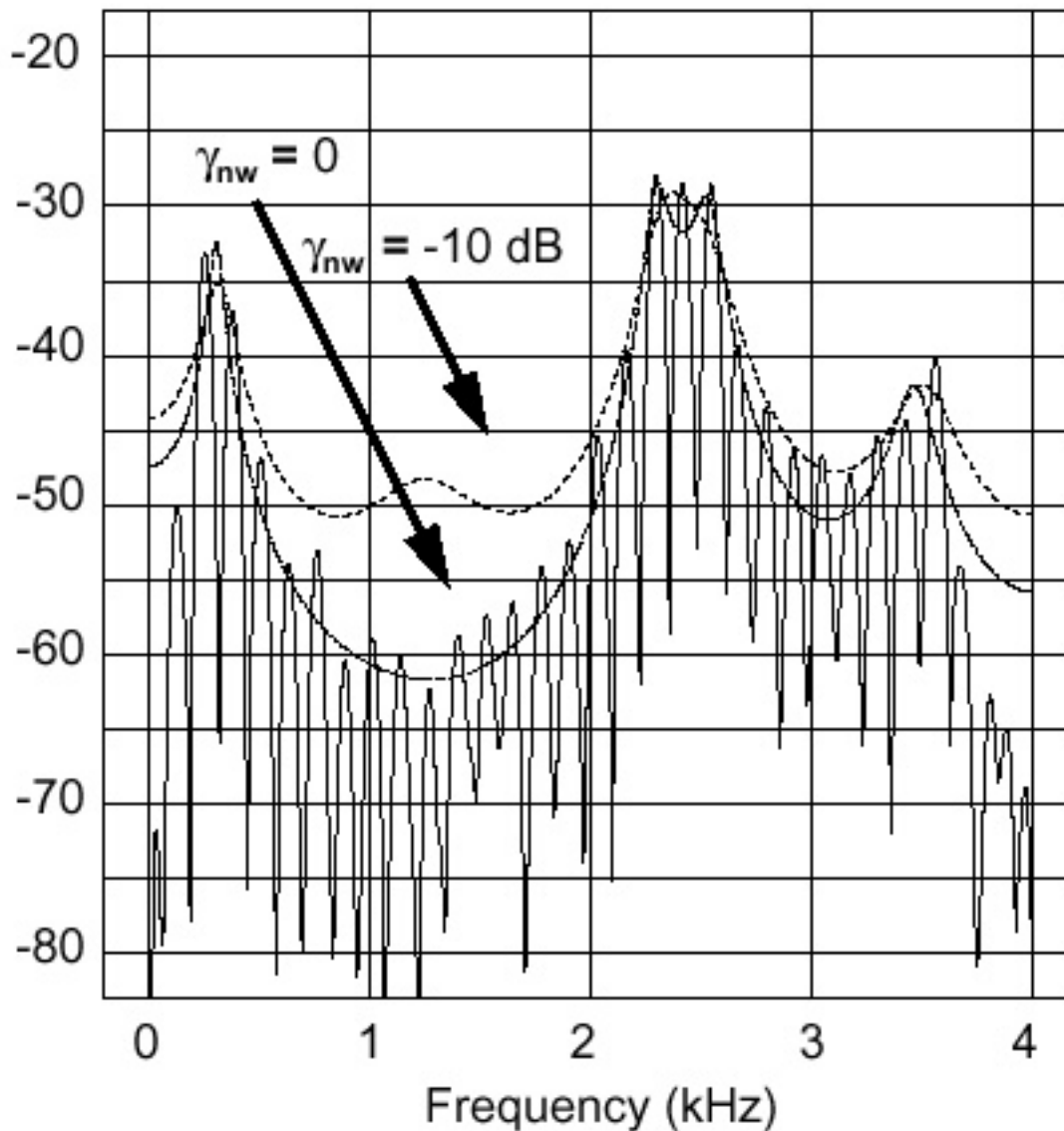
SPECTRAL MATCHING INTERPRETATION

Magnitude(dB)



NOISE FLOORING VIA THE AUTOCORRELATION FUNCTION

Magnitude(dB)



[Return to Main](#)[Objectives](#)**System View:**[Analysis/Synthesis](#)[The Error Signal](#)[Gain Matching](#)**Transformations:**[Levinson-Durbin](#)[Reflection Coefficients](#)[Transformations](#)[The Burg Method](#)**Summary:**[Signal Modeling](#)[Typical Front End](#)**On-Line Resources:**[AJR: Linear Prediction](#)[LPC10E](#)[MAD LPC](#)[Filter Design](#)

LECTURE 16: LINEAR PREDICTION-BASED REPRESENTATIONS

- Objectives:
 - Introduce analysis/synthesis systems
 - Discuss error analysis and gain-matching
 - Relate linear prediction coefficients to other spectral representations
 - Introduce reflection and prediction coefficient recursions

- Lattice/ladder filter implementations

There is a classic textbook on this subject:

J.D. Markel and A.H. Gray,
Linear Prediction of Speech,
Springer-Verlag, New York, New
York, USA, ISBN:
0-13-007444-6, 1976.

This lecture also includes material
from two other textbooks:

J. Deller, et. al., *Discrete-Time
Processing of Speech Signals*,
MacMillan Publishing Co., ISBN:
0-7803-5386-2, 2000.

and,

L.R. Rabiner and B.W. Juang,
Fundamentals of Speech

Recognition, Prentice-Hall, Upper
Saddle River, New Jersey, USA,
ISBN: 0-13-015157-2, 1993.

LECTURE 16: LINEAR PREDICTION-BASED REPRESENTATIONS

- Objectives:
 - Introduce analysis/synthesis systems
 - Discuss error analysis and gain-matching
 - Relate linear prediction coefficients to other spectral representations
 - Introduce reflection and prediction coefficient recursions
 - Lattice/ladder filter implementations

There is a classic textbook on this subject:

J.D. Markel and A.H. Gray, *Linear Prediction of Speech*, Springer-Verlag, New York, New York, USA, ISBN: 0-13-007444-6, 1976.

This lecture also includes material from two other textbooks:

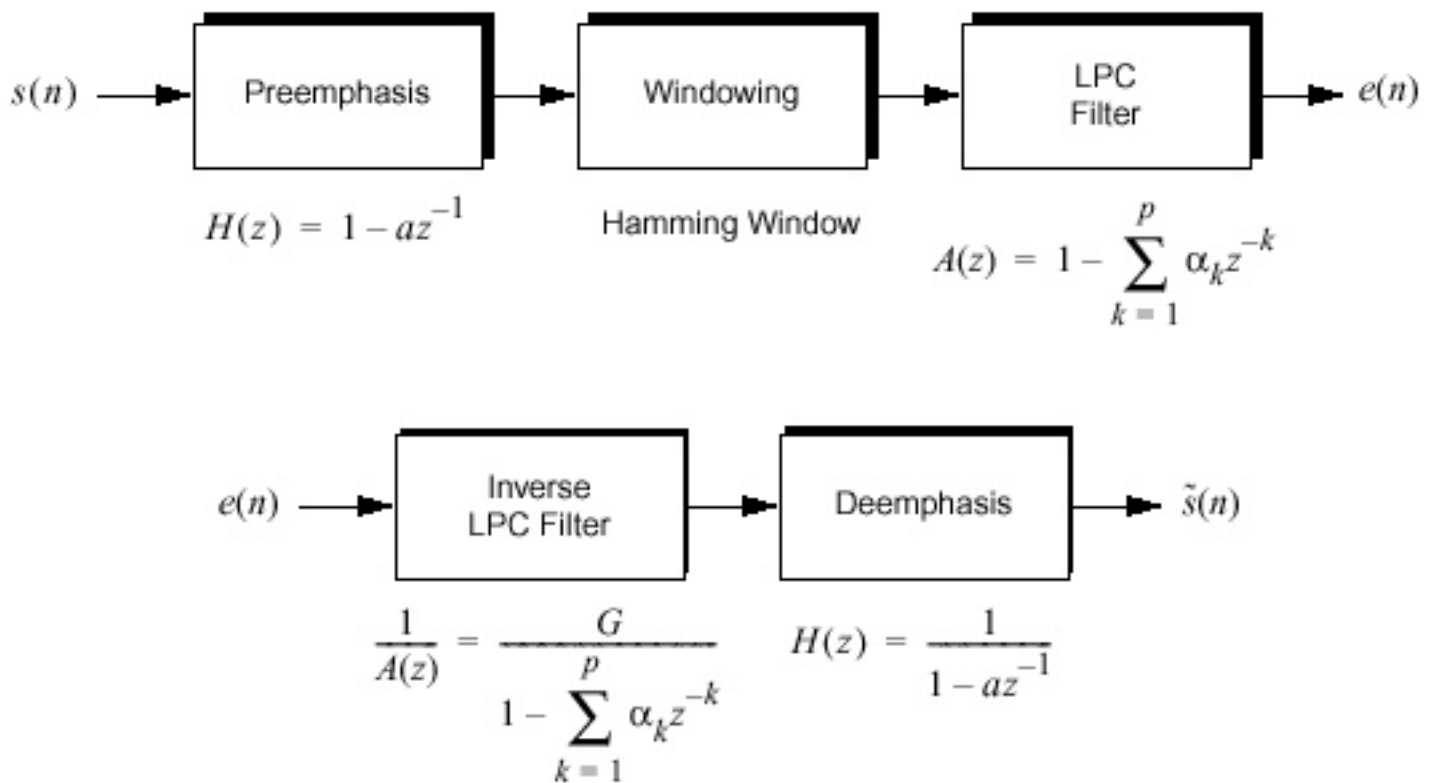
J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co., ISBN: 0-7803-5386-2, 2000.

and,

L.R. Rabiner and B.W. Juang, *Fundamentals of Speech Recognition*, Prentice-Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-015157-2, 1993.

AN LPC-BASED ANALYSIS SYNTHESIS SYSTEM

We can use the linear prediction model to create an analysis/synthesis system:

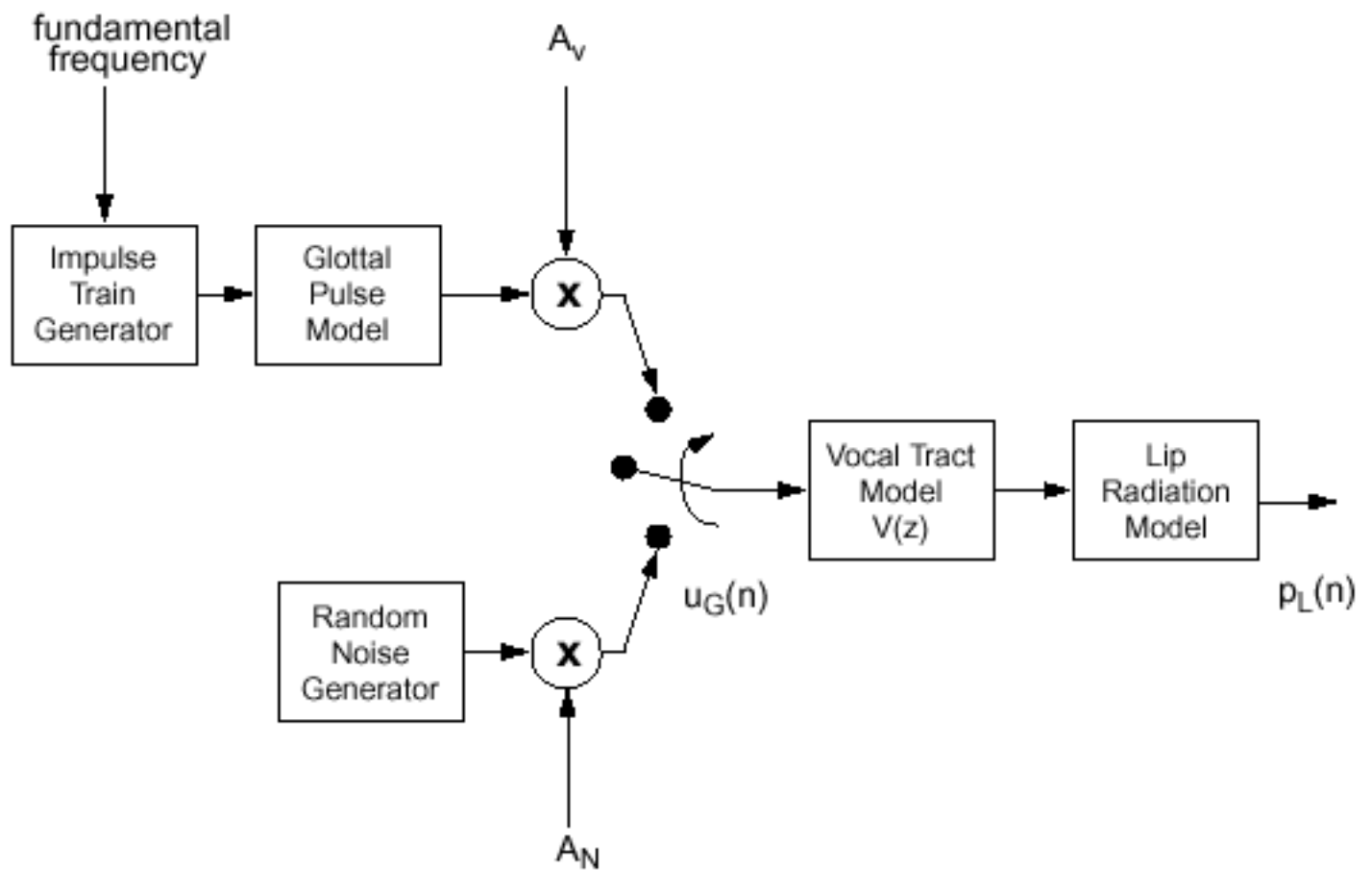


Recall that the linear prediction model, $A(z)$, takes on the form:

$$A(z) = 1 - \sum_{k=1}^p \alpha_k z^{-k}$$

This is an all-zero filter which is minimum phase. The inverse of this filter, which is used to synthesize the signal, is an all-pole filter. If the [autocorrelation method](#) is used to compute the LP coefficients, the filter is guaranteed to be stable.

Note the similarities of this system to our digital model of speech production:

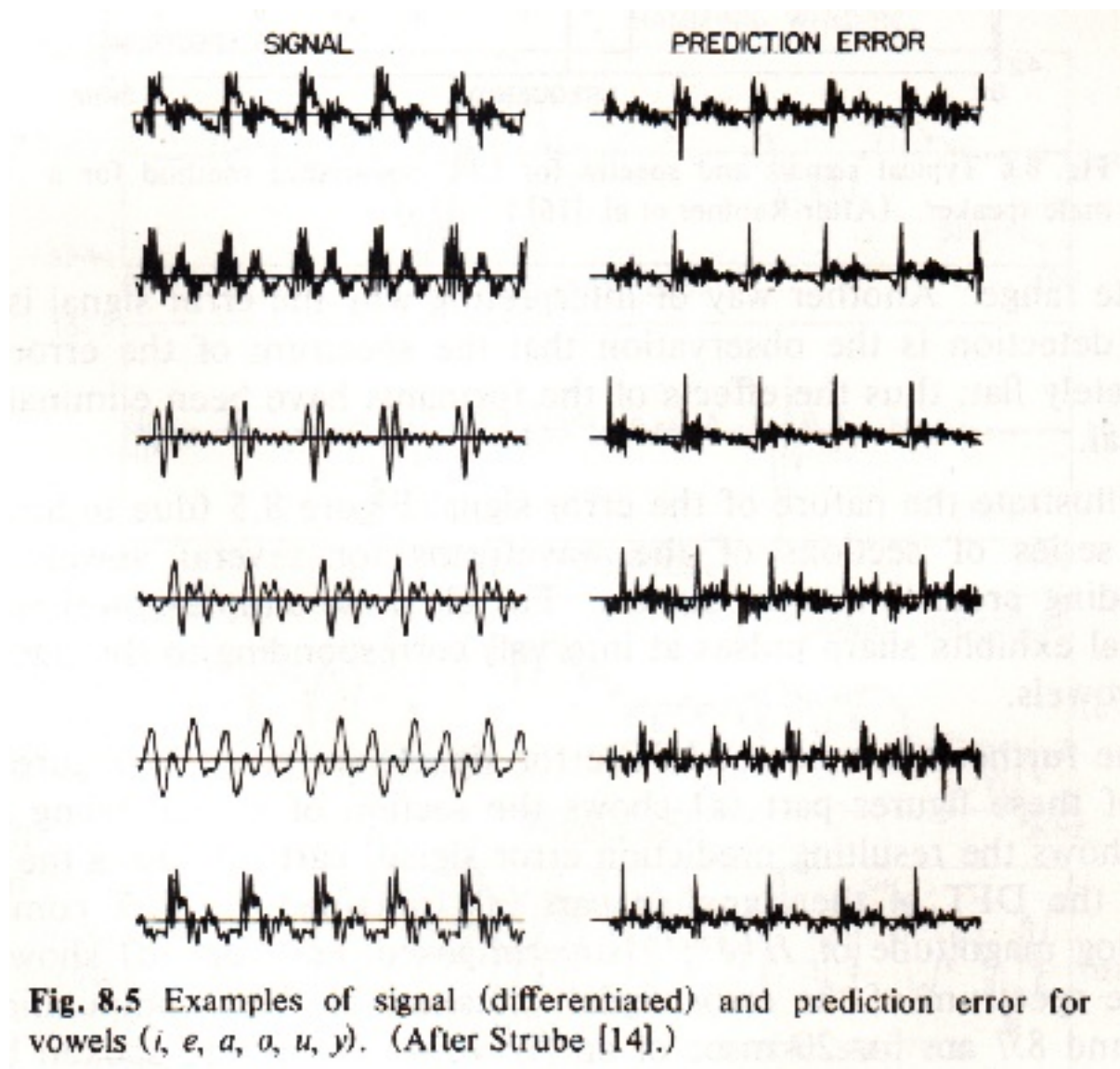


THE LPC ERROR SIGNAL

The LPC error signal can be computed using:

$$E(z) = S(z)A(z) \qquad A(z) = 1 - \sum_{k=1}^p \alpha_k z^{-k}$$

Below are some examples of the LPC error signal:



The error energy (related to accuracy) can be computed from the LPC reflection coefficients:

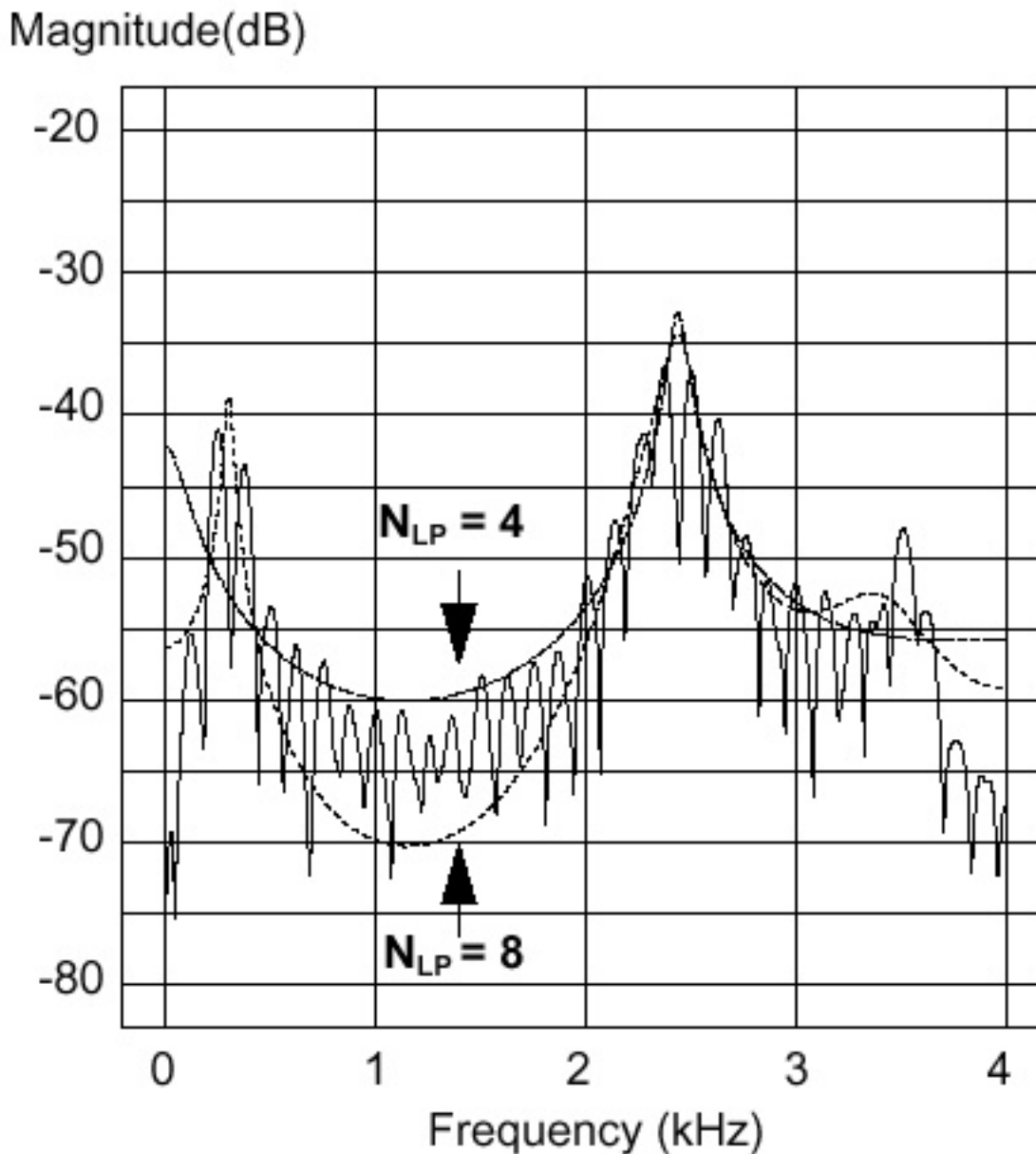
$$E = R_0 \prod_{j=1}^p (1 - k_j^2)$$

Note that if the reflection coefficients have a

magnitude less than one, the error energy is monotonically decreasing with the linear prediction order.

GAIN MATCHING THE LPC MODEL

Recall the spectral matching interpretation of the LP model:



LP analysis is invariant to energy (the same

signal at different energy levels produces the same LP coefficients). How do we match the spectrum implied by the LP model to the signal:

- Gain matching:

$$S(z) = \frac{1}{A(z)} = \frac{G}{1 - \sum_{k=1}^p \alpha_k z^{-k}} \quad G^2 = R(0) - \sum_{k=1}^p \alpha_k R(k)$$

- Error energy:

$$G = R_0 \prod_{j=1}^p (1 - k_j^2)$$

Either techniques produces the same result. The latter technique is popular in speech compression systems.

LEVINSON-DURBIN RECURSION

The prediction coefficients can be efficiently computed for the autocorrelation method using the Levinson-Durbin recursion:

$$\begin{aligned}
 & \text{for } i = 1, 2, \dots, p \\
 & \quad E_0 = r(0) \\
 & \quad k_i = \left(r(i) - \sum_{j=1}^{i-1} a_{i-1}(j)r(i-j) \right) / E_{i-1} \\
 & \quad \text{for } j = 1, 2, \dots, i-1 \\
 & \quad \quad a_i(j) = k_i \\
 & \quad \quad a_i(j) = a_{i-1}(j) - k_i a_{i-1}(i-j) \\
 & \quad E_i = (1 - k_i^2) E_{i-1}
 \end{aligned}$$

This recursion gives us great insight into the linear prediction process. Let us examine a simple example in which we compute a second order model:

$$E_0 = r(0)$$

$$k_1 = r(1)/r(0)$$

$$a_1(1) = k_1 = r(1)/r(0)$$

$$\begin{aligned} E_1 &= (1 - k_1^2)E_0 \\ &= \frac{r^2(0) - r^2(1)}{r(0)} \end{aligned}$$

$$k_2 = \frac{r(2)r(0) - r^2(1)}{r^2(0) - r^2(1)}$$

$$a_2(2) = k_2 = \frac{r(2)r(0) - r^2(1)}{r^2(0) - r^2(1)}$$

$$\begin{aligned} a_2(1) &= a_1(1) - k_2 a_1(1) \\ &= \frac{r(1)r(0) - r(1)r(2)}{r^2(0) - r^2(1)} \end{aligned}$$

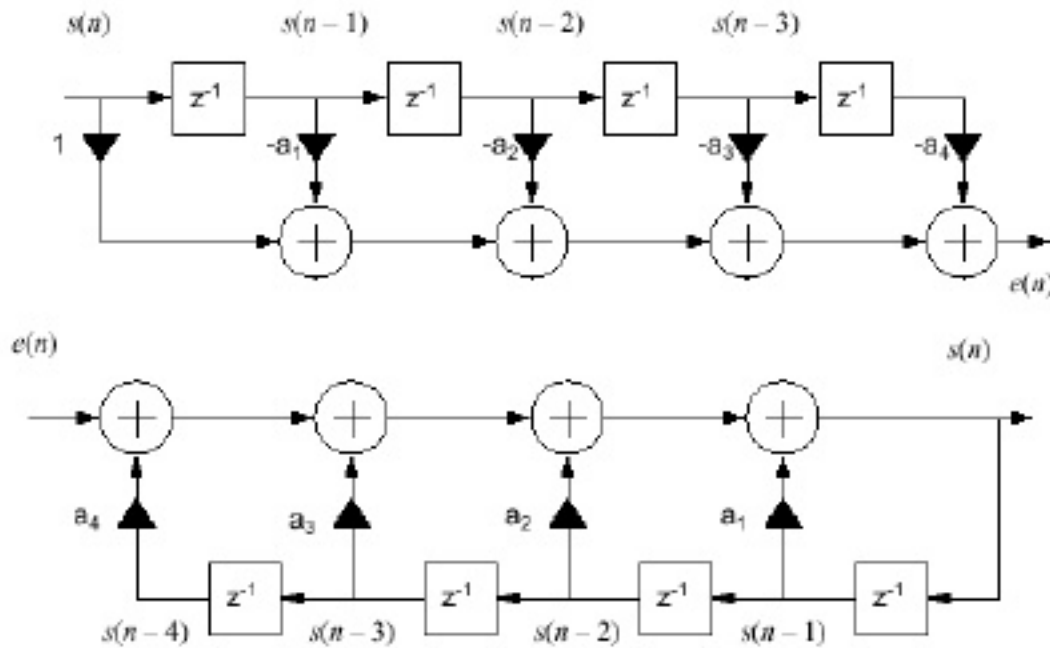
$$\alpha_1 = a_2(1)$$

$$\alpha_2 = a_2(2)$$

This reduces the LP problem to $O(p)$ complexity and saves an order of magnitude in computational complexity. This calculation also makes the process amenable to fixed-point digital signal processors and microprocessors.

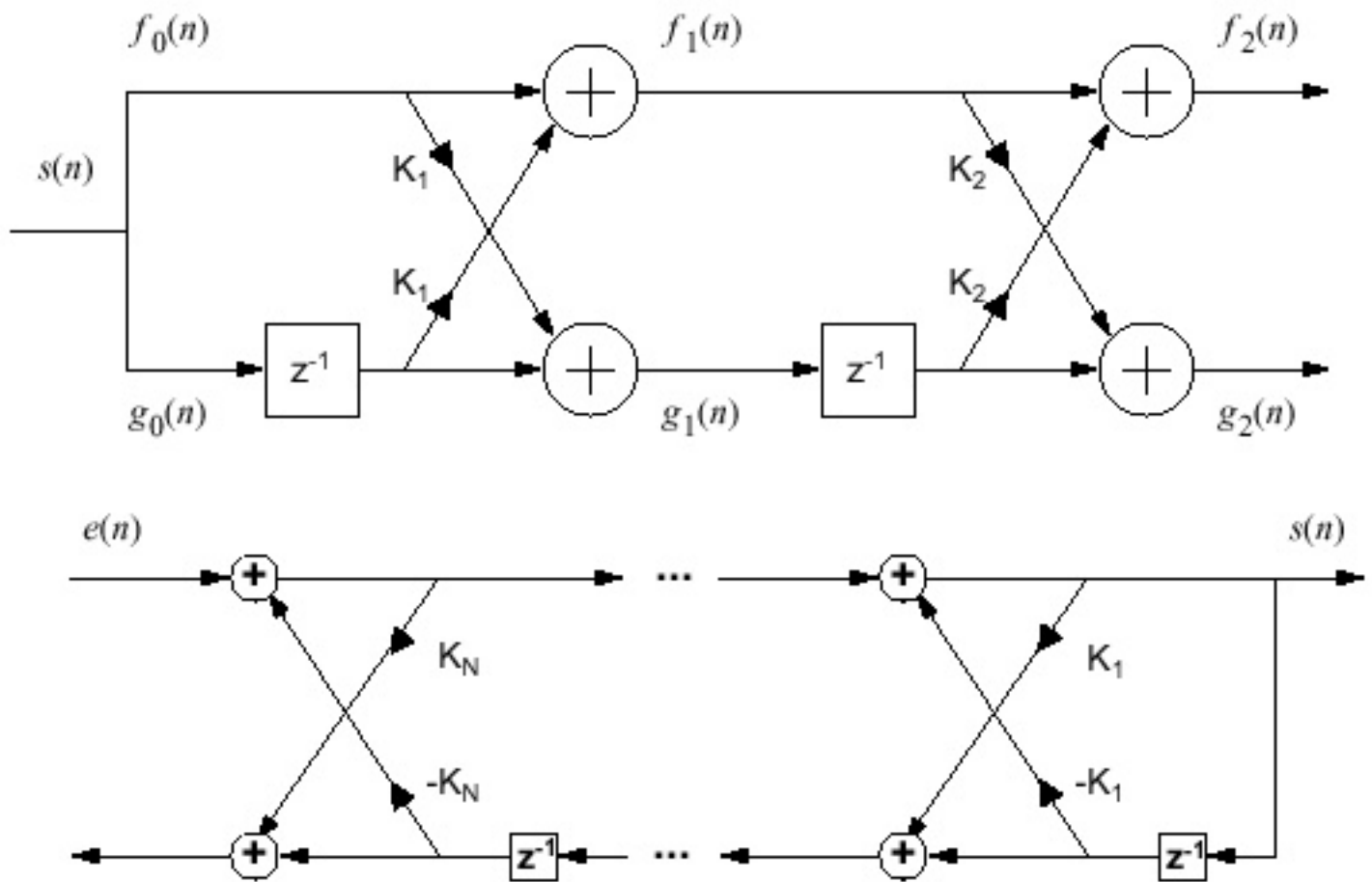
PREDICTION COEFFICIENTS

The LP model can be implemented as a direct form filter:



The coefficients of this model are referred to as *prediction coefficients*.

This same filter can be implemented using a lattice filter:



The coefficients of this filter are referred to as *reflection coefficients*. They can be related directly to the lossless concatenated tube model for the vocal tract (log area ratios).

LINEAR PREDICTION COEFFICIENT TRANSFORMATIONS

The prediction and reflection coefficients can be transformed back and forth with no loss of information:

Prediction to reflection coefficient transformation:

for $i = p, p-1, \dots, 1$

$$k_i = \alpha_i(i)$$

$$\alpha_{i-1}(j) = \frac{\alpha_i(j) + k_i \alpha_i(i-j)}{1 - k_i^2} \quad 1 \leq j \leq i-1$$

Reflection to prediction coefficient transformation:

for $i = 1, 2, \dots, p$

$$\alpha_i(i) = k_i$$

$$\alpha_i(j) = \alpha_{i-1}(j) - k_i \alpha_{i-1}(i-j) \quad 1 \leq j \leq i-1$$

Also, note that these recursions require intermediate storage for $\{\alpha_i\}$.

From the above recursions, it is clear that $|k_i| \neq 1$. In fact, there are several important results related to k_i :

(1) $|k_i| < 1$

(2) $|k_i| = 1$, implies a harmonic process (poles on the unit circle).

(3) $|k_i| > 1$ implies an unstable synthesis filter (poles outside the unit circle).

(4) $E_{-1} > E_{-2} > \dots > E$

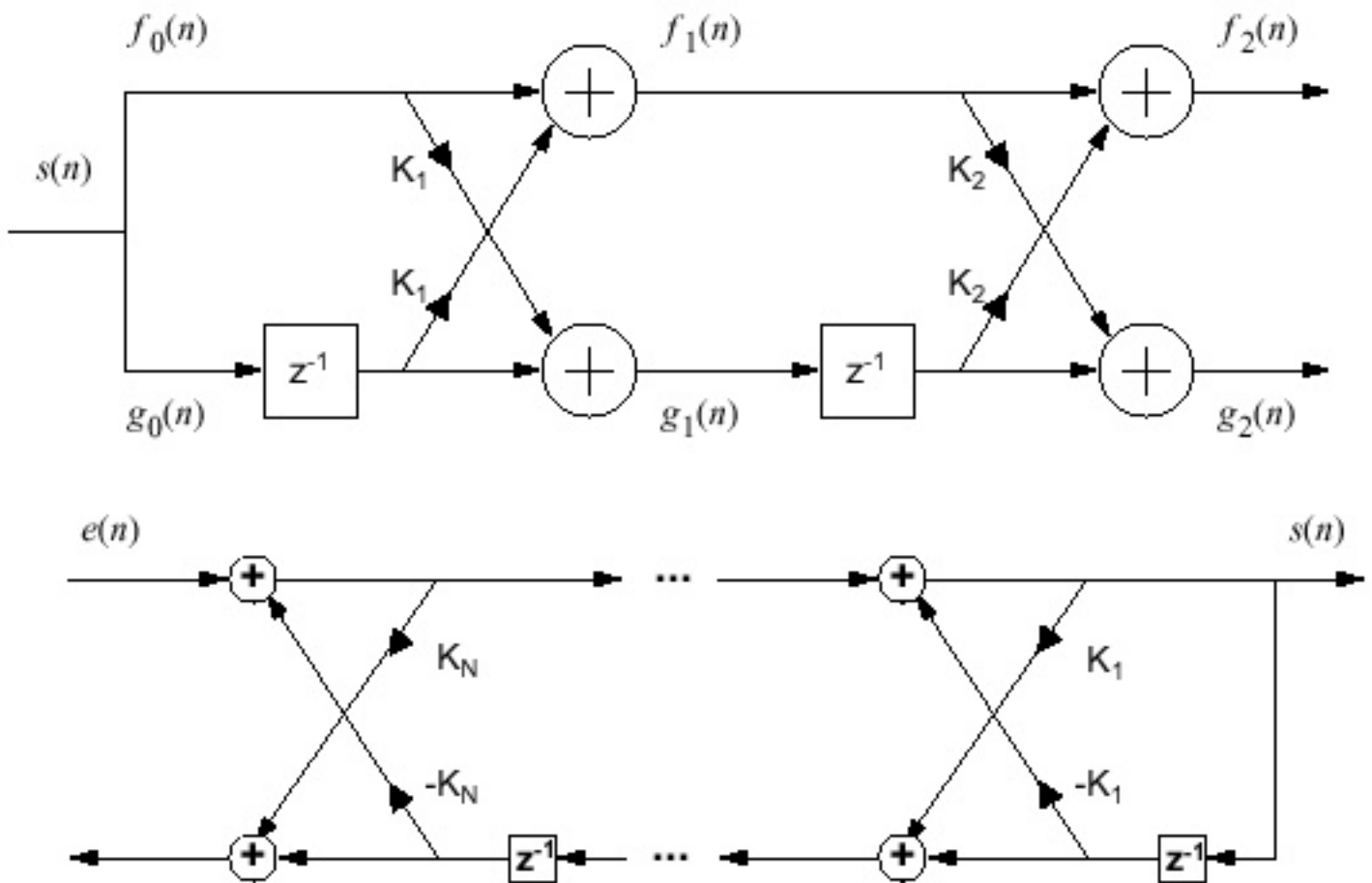
$$x[n] = 0.5x[n-1] + 0.4x[n-2] + p$$

This gives us insight into how to determine the LP order during the calculations. We also see that reflection coefficients are orthogonal in the sense that the best order " p " model is also the first " p " coefficients in the order " $p+1$ " LP model (very important!).

We can also convert LP coefficients to/from autocorrelation coefficients, cepstral coefficients, and log area ratios. See [Signal Modeling Techniques](http://www.isip.msstate.edu/~gao/net/2002_spring/lecture_16/lecture_16_06.html) for further details.

THE BURG METHOD

The standard direct-form FIR filter can be implemented in a lattice structure:



The reflection coefficients can be computed directly using this equation:

$$K_i = \frac{\sum_{m=0}^{N-1} f_{i-1}(m)g_{i-1}(m-1)}{\left\{ \left(\sum_{m=0}^{N-1} (f_{i-1}(m))^2 \right) \left(\sum_{m=0}^{N-1} (f_{i-1}(m-1))^2 \right) \right\}^2}$$

For this filter to be stable, the reflection coefficients must be bounded by 1.

This equation represents one solution from a family of solutions based on lattice methods.

One of the most famous lattice formulations in this family is the *Burg algorithm*, originally introduced in the mid-60's prior to the introduction of LP in speech. There are actually a family of lattice solutions of a similar form. The Burg algorithm requires the coefficients to be computed using:

$$K_i = \frac{\sum_{m=0}^{N-1} f_{i-1}(m)g_{i-1}(m-1)}{\frac{1}{2} \left\{ \sum_{m=0}^{N-1} f_{i-1}(m)^2 + \sum_{m=0}^{N-1} f_{i-1}(m-1)^2 \right\}}$$

We see this is a weighted average of the forward and backward error terms, and that the reflection coefficients are **FORCED** to be bounded.

[Return to Main](#)[Objectives](#)**Perceptual Linear Prediction:**[Spectral Matching](#)[Equal Loudness](#)[Overview](#)[Block Diagram](#)[Equations](#)**Vocal Tract Length Normalization:**[Motivation](#)[Bilinear](#)[Direct](#)**Summary:**[Signal Modeling](#)[Typical Front End](#)**On-Line Resources:**[AJR: PLP](#)[Steffen: PLP](#)[JMCD: VTLN](#)

LECTURE 17: SPECTRAL TRANSFORMATIONS

- Objectives:
 - Introduce perceptual linear prediction
 - Discuss speaker-dependent frequency scaling
 - Introduce vocal tract length normalization
 - Review

The original reference for perceptual linear prediction is:

H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *J. Acoust. Soc. Amer.*, vol. 87, no. 4, pp. 1738--1752,

1990.

Similarly, the original reference for vocal tract length normalization is reprinted here:

A. Andreou, T. Kamm, and J. Cohen, "Experiments in vocal tract normalization," *Proceedings CAIP Workshop: Frontiers in Speech Recognition II*, 1994.

The course textbook and resource links also contain good explanations of this material.

LECTURE 17: SPECTRAL TRANSFORMATIONS

- Objectives:
 - Introduce perceptual linear prediction
 - Discuss speaker-dependent frequency scaling
 - Introduce vocal tract length normalization
 - Review

The original reference for perceptual linear prediction is:

H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *J. Acoust. Soc.*

Amer., vol. 87, no. 4, pp. 1738--1752, 1990.

Similarly, the original reference for vocal tract length normalization is reprinted here:

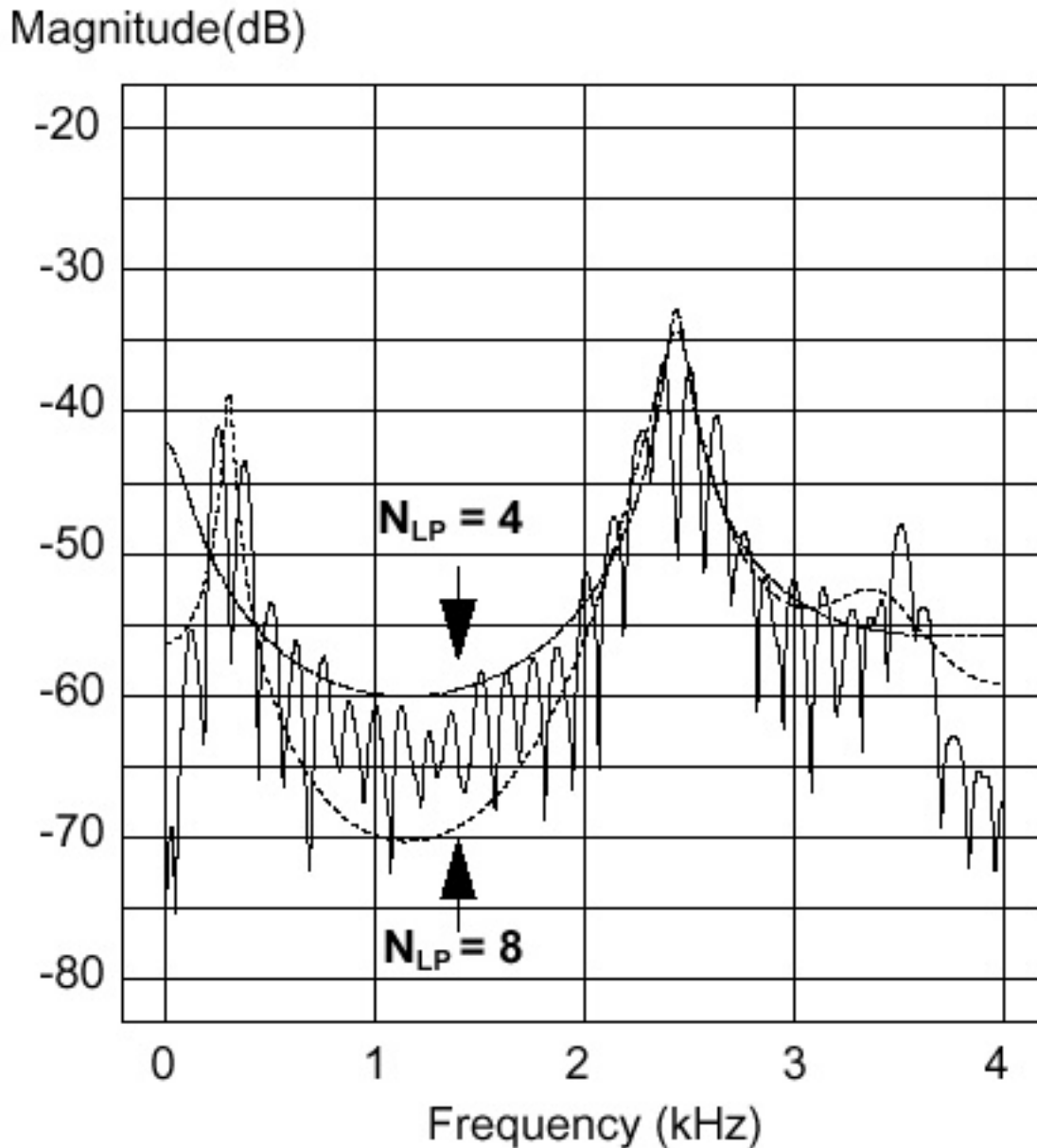
A. Andreou, T. Kamm, and J. Cohen,
"Experiments in vocal tract normalization,"
*Proceedings CAIP Workshop: Frontiers in
Speech Recognition II*, 1994.

The course textbook and resource links also contain good explanations of this material.

SPECTRAL MATCHING INTERPRETATION

Recall that the LP model uses mean-square error approach to optimize its coefficients. This implies:

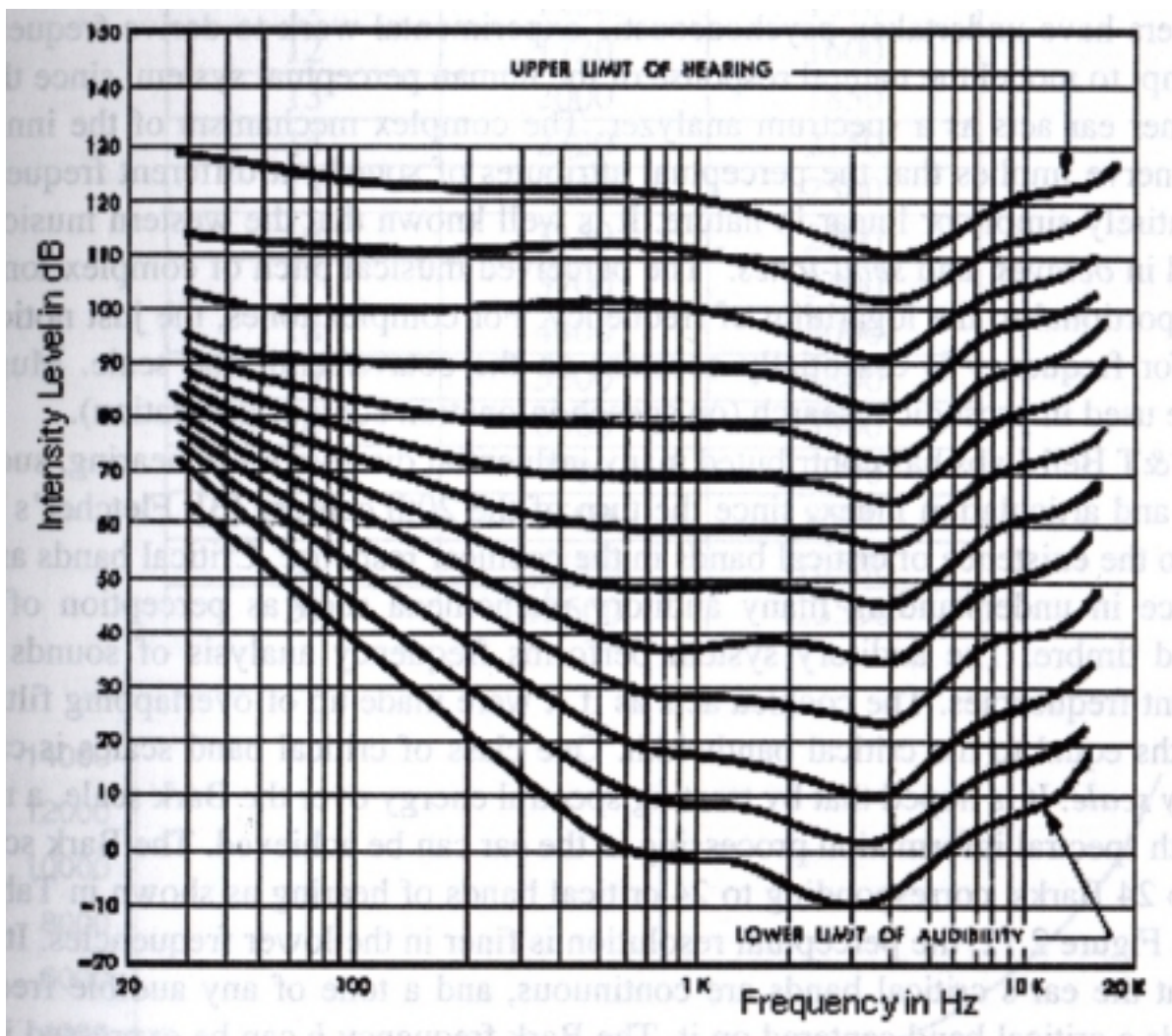
- The LP model attempts spectral flatten the error signal.
- The LP model focuses on the extremely high or low energy areas of the spectrum - whatever it takes to makes the error signal spectrum as flat as possible. Example:



- Note that the eighth-order analysis models the floor of the spectrum more precisely than the third formant.

EQUAL LOUDNESS CURVES

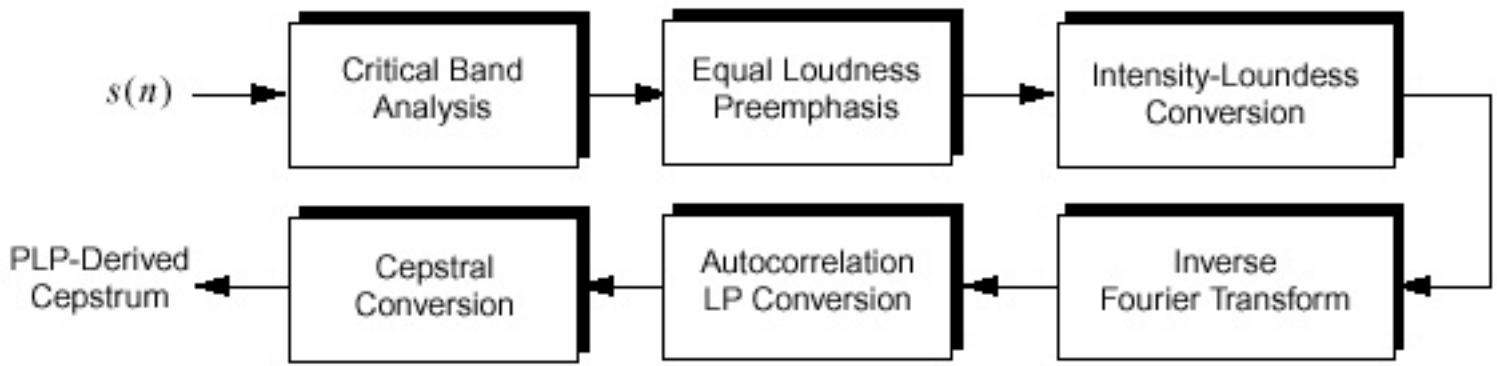
Recall our observation that perceptual loudness of a sound is a function of its absolute intensity:



- The sensitivity of the ear varies with the frequency content and the quality of a sound.

- The graph above represents equal loudness contours adopted by the ISO (ISO 226).
- Hearing sensitivity peaks at 4K Hz, and has a secondary peak at 13K Hz.

PERCEPTUAL LINEAR PREDICTION BLOCK DIAGRAM



- Goals:
 - Apply greater weight to perceptually-important portions of the spectrum
 - Avoid uniform weighting across the frequency band
- Algorithm:

- Compute the spectrum via a DFT
- Warp the spectrum along the Bark frequency scale
- Convolve the warped spectrum with the power spectrum of the simulated critical band masking curve and downsample (to typically 18 spectral samples)
- Preemphasize by the simulated equal-loudness curve:
 - Simulate the nonlinear relationship between intensity and perceived loudness by performing a cubic-root amplitude compression
- Compute an LP model

- Compute an LP-derived cepstrum
- Claims:
 - Improved speaker independent recognition performance
 - Increased robustness to noise, variations in the channel, and microphones

EQUAL LOUDNESS PREEMPHASIS AND PERCEIVED INTENSITY

- Equal loudness preemphasis is implemented using:

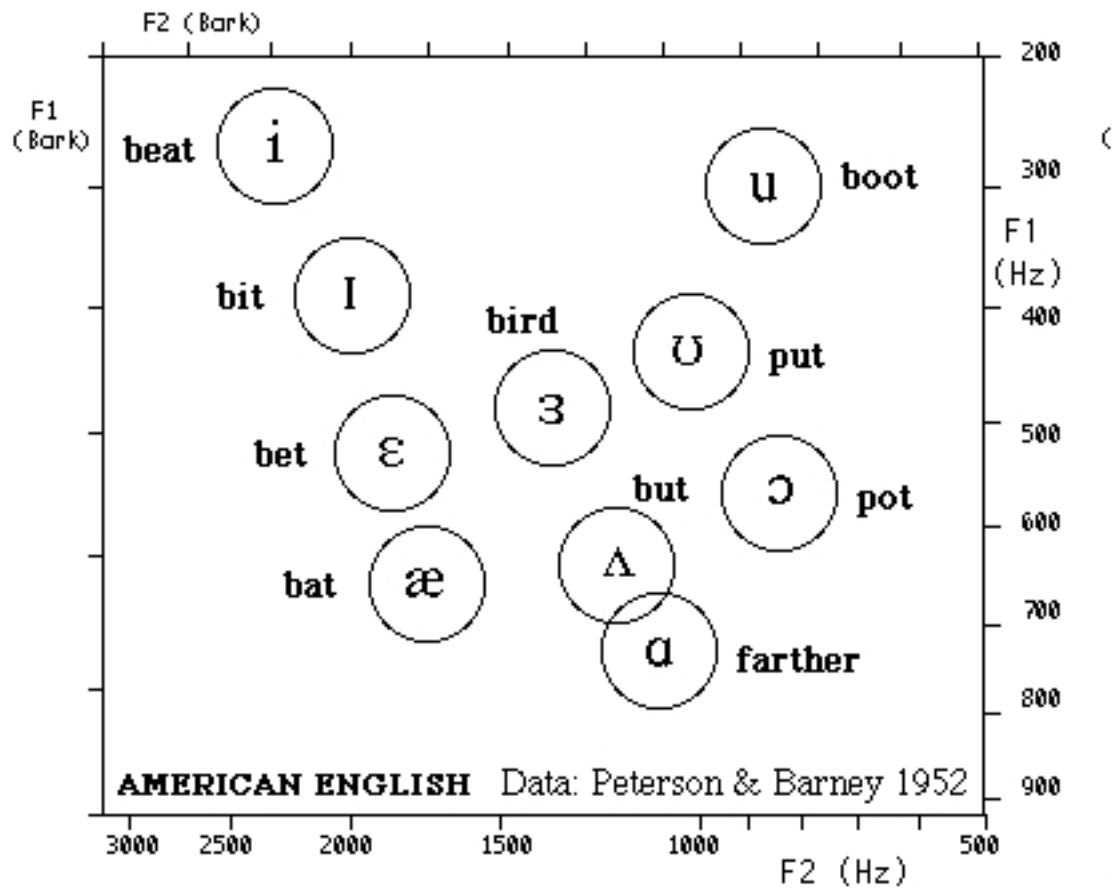
$$E(\omega) = \frac{(\omega^2 + 56.8 \times 10^6) \omega^4}{(\omega^2 + 6.3 \times 10^6)(\omega^2 + 0.38 \times 10^9)(\omega^6 + 9.58 \times 10^{26})}$$

- Perceived loudness can be approximated using:

$$L(\omega) = I(\omega)^{1/3}$$

FORMANT FREQUENCY DISTRIBUTIONS FOR VOWELS

- Recall the Peterson-Barney vowel data:



Mean formant values of 33 male speakers for ten American English vowels. (Data from: G. Peterson and H.L. Barney, "Control Methods Used In A Study Of The Vowels," *Journal of the Acoustical Society of America*, vol. 24, pp. 175-184, 1952. Figure from: [Projektit: Vowel Charts](http://projektit.org/VowelCharts/).)

- What causes this natural variation?

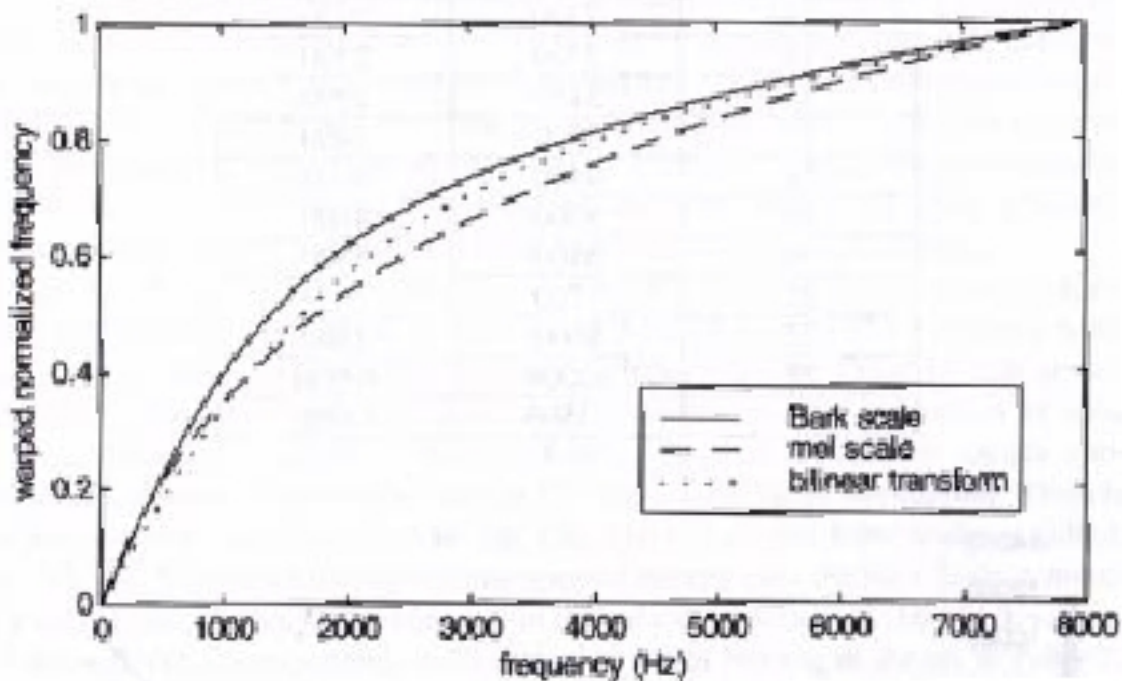
- Is this type of variation desirable?
- How can we offset this variation?

SPEAKER-DEPENDENT FREQUENCY WARPING

- Recall the bilinear transform:

$$s = \frac{z^{-1} - \alpha}{1 - \alpha z^{-1}} \quad \Omega = \omega + 2 \arctan\left(\frac{\alpha \sin(\omega)}{1 - \alpha \cos(\omega)}\right)$$

- This was a good approximation to the Bark and mel scales:



- How can we compute the optimal warping factor?

DIRECT MEL-SCALE FREQUENCY WARPING

- We can warp the linear frequency axis directly. Let $k\Delta f_{mel}$, $k = 1, \dots, K$ denote the center frequencies on the mel scale. We can warp these frequencies using a simple linear transformation:

$$f_{Hz}^{\alpha}(k\Delta f_{mel}) = 700(10^{(k\Delta f_{mel})/2595} - 1)/\alpha$$

- We can also warp the discrete Fourier transform samples directly using a similar linear compression.
- A typical range for the warping factors is $[0.8, 1.2]$.
- What are the relative merits of this approach?

[Return to Main](#)[Objectives](#)**Introduction:**[Components](#)[Temporal Information](#)[Z-Transform](#)[Curve-Fitting](#)[Alternatives](#)**Finite Differences:**[First-Order Difference](#)[Frequency Response](#)**Regression:**[Mean-Square Error](#)[Central Difference](#)**On-Line Resources:**[SRSTW: Derivatives](#)[Polynomial Fitting](#)[Finite Differences](#)[STRUT: Post-Processing](#)[Regression Theory](#)[Regression Tutorial](#)[Regression Applet](#)[Autofit](#)

LECTURE 18: DIFFERENTIATION OF FEATURES

- Objectives:
 - Introduce the concept of a derivative
 - Appreciate the computational issues
 - Derivatives based on finite differences
 - Derivatives based on linear regression

Three important references for this material are:

- F.K. Soong and A.E. Rosenberg, "On the Use of Instantaneous and Transitional Spectral Information in Speaker Recognition," *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, Tokyo, Japan, pp. 877-880, April 1986.
- J.G. Proakis and D.G. Manolakis, *Digital Signal Processing (Third Edition)*, Prentice-Hall, Upper Saddle River, New Jersey, USA, 1996.
- A.J. Hayter, *Probability and Statistics For Engineers and Scientists*, International Thomson Publishing, Cincinnati, Ohio, USA, 1996.

The course textbook contains

references to the seminal papers in this area as well.

LECTURE 18: DIFFERENTIATION OF FEATURES

- Objectives:
 - Introduce the concept of a derivative
 - Appreciate the computational issues
 - Derivatives based on finite differences
 - Derivatives based on linear regression

Three important references for this material are:

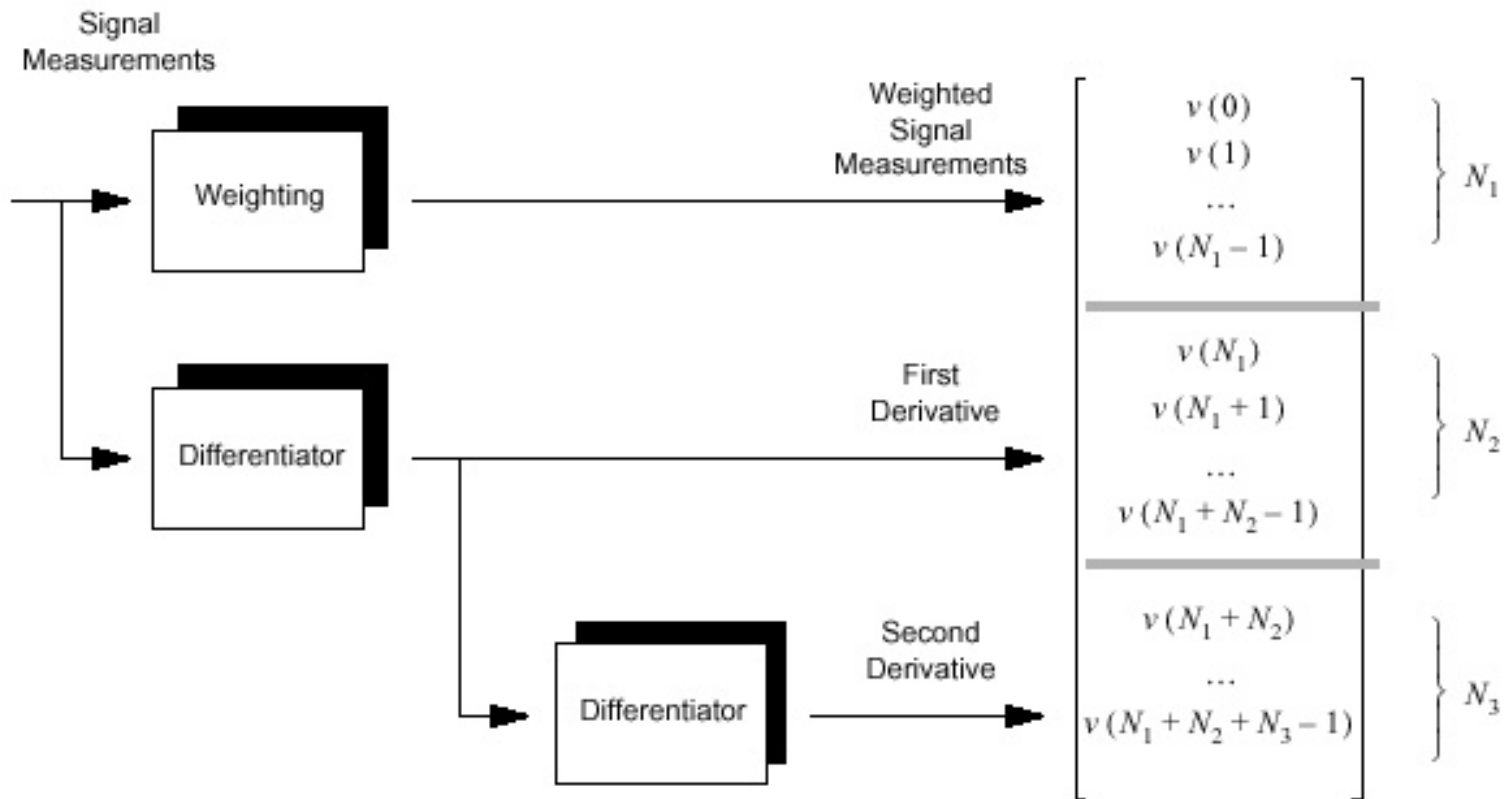
- F.K. Soong and A.E. Rosenberg, "On the Use of Instantaneous and Transitional Spectral Information in Speaker Recognition,"
Proceedings of the International Conference

on Acoustics, Speech, and Signal Processing, Tokyo, Japan, pp. 877-880, April 1986.

- J.G. Proakis and D.G. Manolakis, *Digital Signal Processing (Third Edition)*, Prentice-Hall, Upper Saddle River, New Jersey, USA, 1996.
- A.J. Hayter, *Probability and Statistics For Engineers and Scientists*, International Thomson Publishing, Cincinnati, Ohio, USA, 1996.

The course textbook contains references to the seminal papers in this area as well.

ADDING TEMPORAL INFORMATION: DERIVATIVES



- Temporal derivatives of the spectrum are commonly approximated by differentiating cepstral features using a linear regression.

ADDING TEMPORAL INFORMATION: DERIVATIVES

- We would like to add information about the change in the spectrum to our feature vector to improve our ability to distinguish between stationary sounds (vowels) and nonstationary sounds (consonants).
- Recall the definition of differentiation in the time domain:

$$\frac{d}{dt}x(t) \Leftrightarrow j\omega X(\omega)$$

- Differentiation is an inherently noisy process since it amplifies high frequencies. Hence, we must be careful how we compute this. In practice, we use low-pass filtered derivatives (the derivative of a low-pass filtered version

of the signal).

- What we really want to measure is the time derivative of the spectrum:

$$\frac{\partial}{\partial t} X(\omega, t) = ???$$

But derivatives of continuous time signals are difficult to compute for discrete-time signals.

- Recall the definition of a derivative:

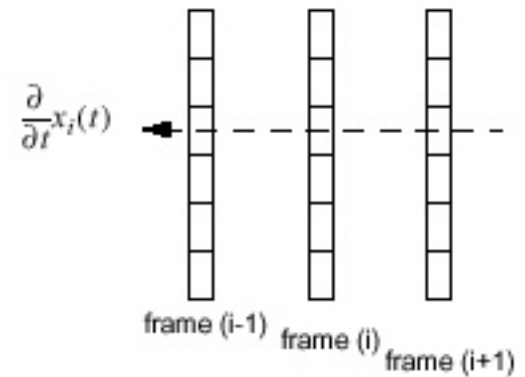
$$\frac{d}{dt} x(t) = \lim_{T \rightarrow 0} \frac{x(t) - x(t - T)}{T}$$

This can be viewed as a digital filter:

$$y(n) = \frac{x(n) - x(n-1)}{T} \Leftrightarrow H(z) = \left(\frac{1}{T}\right) 1 - z^{-1}$$

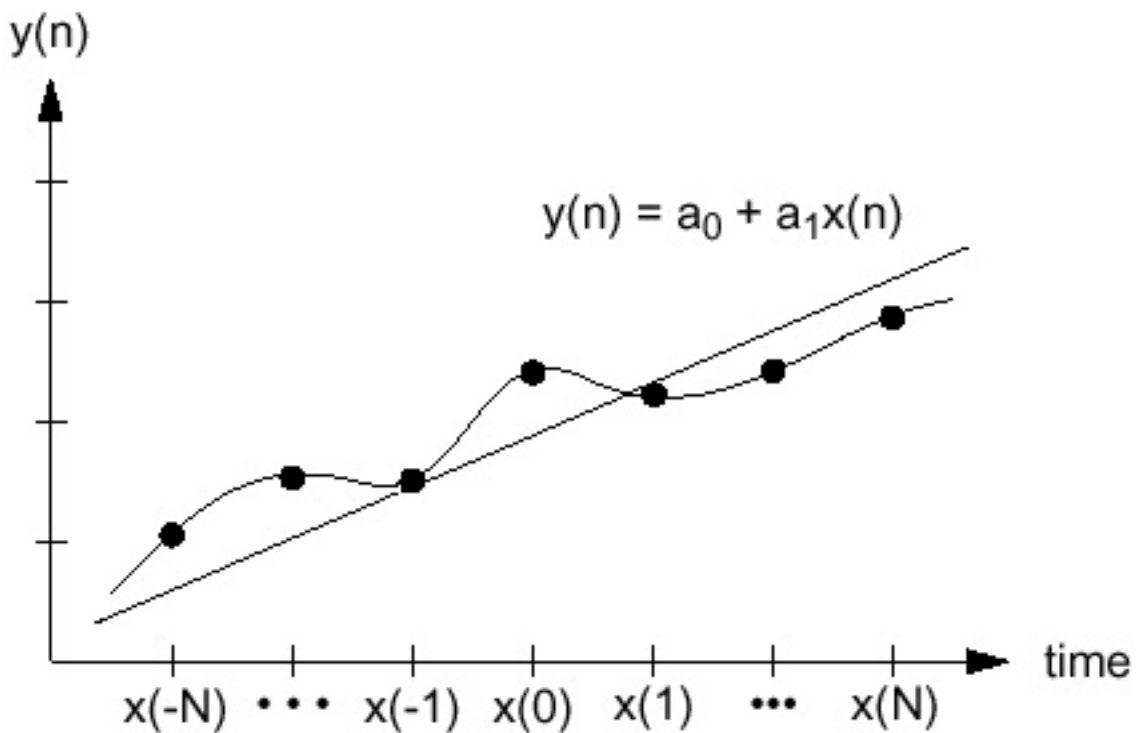
Later we will explore the frequency response of this filter.

- In practice, we compute temporal derivatives of feature vectors by differentiating each element as a function of time. Since feature vectors measure the spectrum, this gives us a realistic measure of spectral change. These derivatives, called delta parameters, are concatenated with the absolute measurements to form an extended feature vector that contains absolute and rate of change information.



GRAPHICAL INTERPRETATION: CURVE FITTING

- What we seek is the value of the slope, not the differentiated signal. This can be directly estimated using the principle of linear regression.
- We can cast this estimation problem as a curve-fitting problem with some special constraints that result from the signal processing nature of the problem.
- Consider the estimation problem shown below:



- The slope of a signal can be estimated directly using a linear regression approach. More precisely, we are using a least mean square error parameter estimation approach to finding the equation of a line that best approximates the signal.
- Note that the slope of the line is represented by the parameter a_1 in the equation shown in the figure above. **This is the parameter of**

interest in this analysis.

ALTERNATE WAYS TO COMPUTE A DERIVATIVE

In conventional digital signal processing (DSP) textbooks, derivatives, $\partial^k y(t)/\partial t$, are estimated several ways:

- simple backward difference (first-order — $p=1$):

$$y(nT) = \frac{x(nT) - x((n-1)T)}{T} \quad (2)$$

- central difference (first-order — $p=1$):

$$y(nT) = \frac{x((n+1)T) - x((n-1)T)}{2T} \quad (3)$$

- digital filters:

$$y(n) = a_1 y(n-1) + a_2 y(n-2) + \dots + b_0 x(n) + b_1 x(n-1) + \dots \quad (4)$$

- higher-order approximations (Taylor Series, Splines):

$$y(nT) = a_0 + a_1 \frac{\partial}{\partial t} y(nT) + \frac{\partial^2}{\partial t^2} y(nT) + \dots \quad (5)$$

- What we must keep clear here is the difference between the order of the derivative (**k**), the order of the approximation (**p**), and the length of the filter or difference equation used to

compute the approximation (**N**).

PROPERTIES OF A FIRST-ORDER DIFFERENCE

We can compute the frequency response of a first-order difference:

$$y(n) = \frac{1}{2}[x(n) - x(n-1)]$$

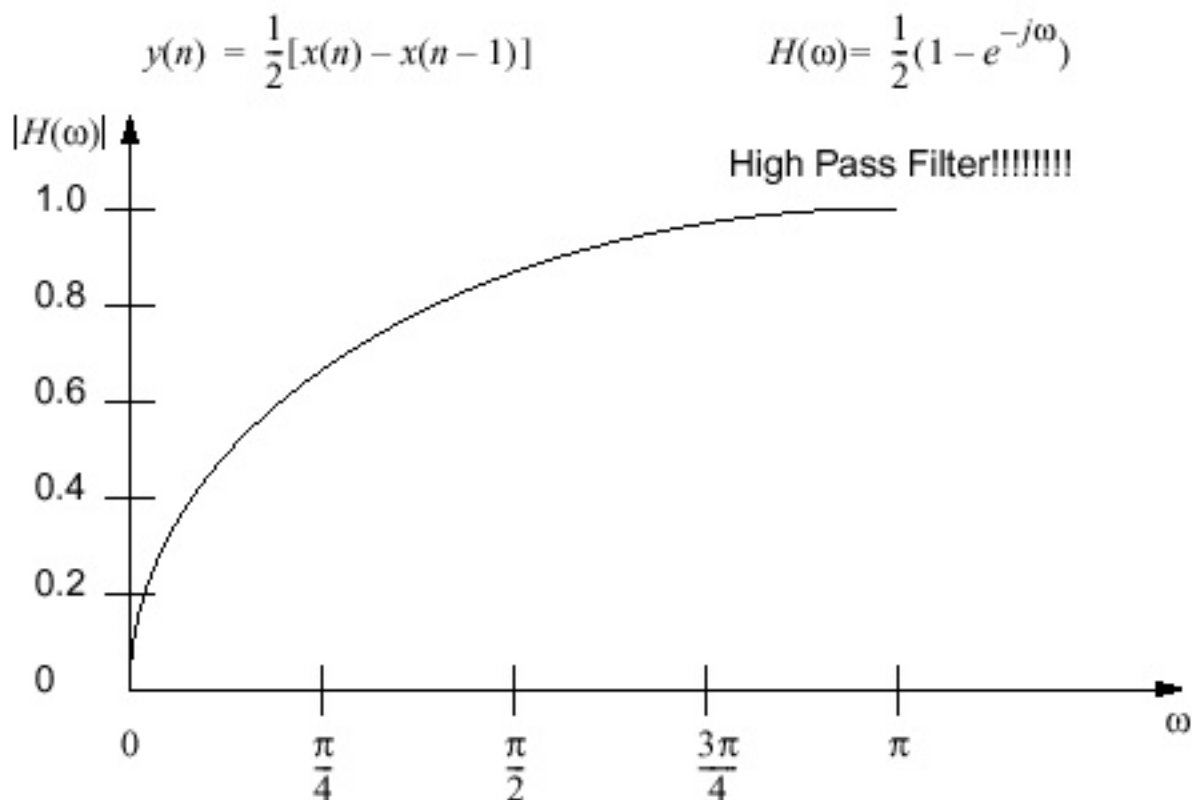
$$h(n) = \left\{ -\frac{1}{2}, \frac{1}{2} \right\}$$

$$H(\omega) = \frac{1}{2}(1 - e^{-j\omega})$$

$$\begin{aligned} |H(\omega)| &= \frac{1}{2} |1 - \cos \omega + j \sin \omega| \\ &= \frac{1}{2} \sqrt{(1 - \cos \omega)^2 + (\sin \omega)^2} \\ &= \frac{1}{2} \sqrt{(1 - 2 \cos \omega + (\cos \omega)^2) + (\sin \omega)^2} \\ &= \frac{1}{2} \sqrt{2 - 2 \cos \omega} \\ &= \frac{1}{\sqrt{2}} \sqrt{1 - \cos \omega} \end{aligned}$$

FREQUENCY RESPONSE OF A FIRST-ORDER DIFFERENCE

A plot of the frequency response for this filter is shown below:



- Because this filter acts as a high-pass filter, it has a tendency to amplify noise.

MEAN SQUARE ERROR DERIVATION

- In speech recognition, we prefer to use a statistical approach to estimating the derivative. Why?
- This technique uses a statistical method known as linear regression. In this approach, we choose the regression coefficients to minimize the mean squared error:

$$E = \sum_{n=-\infty}^{\infty} [y(n) - (a_0 + a_1 x(n))]^2$$

- The solution to this equation is well-known (in DSP literature, this is known as linear prediction), and is found by differentiating the error equation with respect to the regression coefficients, setting the derivative to zero, and solving for the regression coefficients. This

results in the following equations:

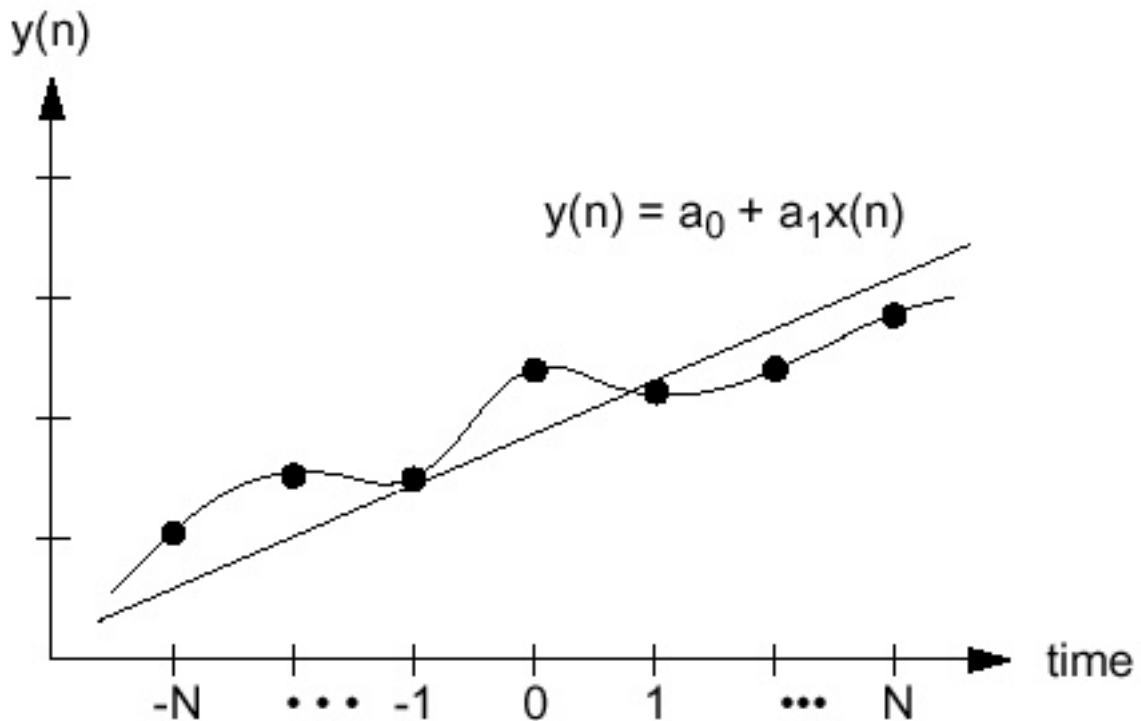
$$a_1 = \frac{n \sum x(n)y(n) - (\sum x(n))(\sum y(n))}{n \sum x^2(n) - (\sum x(n))^2}$$

$$a_0 = \left(\frac{1}{n}\right) \sum y(n) - a_1 \left(\frac{1}{n}\right) \sum x(n)$$

- This equation is fairly general. Note that if the input data have an average value of zero, the resulting equations are even simpler.

LINEAR REGRESSION

- We can simplify the previous equation by imposing a central difference type formulation of the problem, as shown below:



The x-axis is relabeled in terms of equispaced sample indices, and centered about zero.

- This simplifies the calculation to:

$$\begin{aligned}
 a_1 &= \frac{n \sum x(n)y(n) - (\sum x(n))(\sum y(n))}{n \sum x^2(n) - (\sum x(n))^2} \\
 &= \frac{n \sum x(n)y(n)}{n \sum x^2(n)} \\
 &= \frac{\sum_{n=-N}^N ny(n)}{\sum_{n=-N}^N n^2}
 \end{aligned}$$

- This equation is the form we desire, and is extremely efficient to compute. The denominator can be precomputed, and the integer multiplications are easily implemented even in fixed-point DSPs.
- Obviously, this approach can be extended to higher order derivatives. However, historically, second derivatives in speech recognition have been computed by applying two first-order derivatives in succession.

- Further, the order of regression used, N , is most commonly set to 2, which means a five-frame sequence of features is required to compute the first-order derivative.

[Return to Main](#)[Objectives](#)**Distance Measures:**[Distance](#)[Weighted Distance](#)[Prewhitening](#)[Relationship](#)[Noise Reduction](#)[Computation](#)**Maximum Likelihood:**[Classification](#)[Maximization](#)[Mahalanobis](#)**On-Line Resources:**[PCA](#)[ICA](#)[Factor Analysis](#)[Statistical Normalization](#)[Pattern Recognition Applet](#)

LECTURE 19: PRINCIPLE COMPONENTS ANALYSIS

- Objectives:
 - Introduce the concept of a distance measure
 - Introduce statistically-weighted distance measures
 - Review maximum likelihood classification
 - Explore the relationship between weighted distance measures and maximum likelihood classification

- Introduce the Mahalanobis distance measure

This material can be found in most pattern recognition textbooks. This is the book we recommend:

R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification* (Second Edition), Wiley Interscience, New York, New York, USA, ISBN: 0-471-05669-3, 2000.

and use in our pattern recognition course. The material in this lecture follows this textbook closely:

J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co., ISBN: 0-7803-5386-2, 2000.

Each of these sources contain references to the seminal

publications in this area, including our all-time favorite:

K. Fukunga, *Introduction to Statistical Pattern Recognition*, MacMillan Publishing Company, San Diego, California, USA, ISBN: 0-1226-9851-7, 1990.

LECTURE 19: PRINCIPLE COMPONENTS ANALYSIS

- Objectives:

- Introduce the concept of a distance measure
- Introduce statistically-weighted distance measures
- Review maximum likelihood classification
- Explore the relationship between weighted distance measures and maximum likelihood classification
- Introduce the Mahalanobis distance

measure

This material can be found in most pattern recognition textbooks. This is the book we recommend:

R.O. Duda, P.E. Hart, and D.G. Stork,
Pattern Classification (Second Edition),
Wiley Interscience, New York, New York,
USA, ISBN: 0-471-05669-3, 2000.

and use in our pattern recognition course. The material in this lecture follows this textbook closely:

J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co.,
ISBN: 0-7803-5386-2, 2000.

Each of these sources contain references to the seminal publications in this area, including our all-time favorite:

K. Fukunga, *Introduction to Statistical*

Pattern Recognition, MacMillan Publishing
Company, San Diego, California, USA,
ISBN: 0-1226-9851-7, 1990.

DISTANCE MEASURES

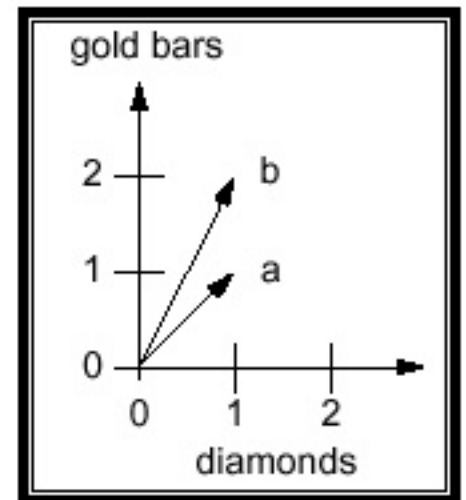
What is the distance between pt. a and pt. b?

The N-dimensional real Cartesian space,

denoted \mathfrak{R}^N is the collection of all N-dimensional vectors with real elements. A metric, or distance measure, is a real-valued function with three properties:

$$\forall \bar{x}, \bar{y}, \bar{z} \in \mathfrak{R}^N:$$

1. $d(\bar{x}, \bar{y}) \geq 0$.
2. $d(\bar{x}, \bar{y}) = 0$ if and only if $\bar{x} = \bar{y}$
3. $d(\bar{x}, \bar{y}) \leq d(\bar{x}, \bar{z}) + d(\bar{z}, \bar{y})$



The Minkowski metric of order s , or the l_s metric, between \bar{x} and \bar{y} is:

$$d_s(\bar{x}, \bar{y}) \equiv \sqrt[s]{\sum_{k=1}^N |x_k - y_k|^s} = \|\bar{x} - \bar{y}\|_s$$

(the norm of the difference vector).

Important cases are:

1. l_1 or city block metric (sum of absolute values),

$$d_1(\bar{x}, \bar{y}) = \sum_{k=1}^N |x_k - y_k|$$

2. l_2 , or Euclidean metric (mean-squared error),

$$d_2(\bar{x}, \bar{y}) = \sqrt{\sum_{k=1}^N |x_k - y_k|^2}$$

3. l_∞ or Chebyshev metric,

$$d_\infty(\bar{x}, \bar{y}) = \max_k |x_k - y_k|$$

WEIGHTED EUCLIDEAN DISTANCES

We can similarly define a weighted Euclidean distance metric:

$$d_{2w}(\bar{x}, \bar{y}) = \sqrt{|\bar{x} - \bar{y}|^T \underline{W} |\bar{x} - \bar{y}|}$$

where:

$$\bar{x} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_k \end{bmatrix}, \bar{y} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_k \end{bmatrix}, \text{ and } \underline{W} = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1k} \\ w_{21} & w_{22} & \dots & w_{2k} \\ \dots & \dots & \dots & \dots \\ w_{k1} & w_{k2} & \dots & w_{kk} \end{bmatrix}.$$

Why are Euclidean distances so popular?

One reason is efficient computation. Suppose we are given a set of M reference vectors, \bar{x}_m , a measurement, \bar{y} , and we want to find the nearest neighbor:

$$NN = \min_m d_2(\bar{x}_m, \bar{y})$$

This can be simplified as follows:

We note the minimum of a square root is the same as the minimum of a square (both are monotonically increasing functions):

$$\begin{aligned} d_2(\bar{x}_m, \bar{y})^2 &= \sum_{j=1}^k (x_{m_j} - y_j)^2 = \sum_{j=1}^k x_{m_j}^2 - 2x_{m_j}y_j + y_j^2 \\ &= \|\bar{x}_m\|^2 - 2\bar{x}_m \bullet \bar{y} + \|\bar{y}\|^2 \\ &= C_m + C_y - 2\bar{x}_m \bullet \bar{y} \end{aligned}$$

Therefore,

$$NN = \min_m d_2(\bar{x}_m, \bar{y}) = C_m - 2\bar{x}_m \bullet \bar{y}$$

Thus, a Euclidean distance is virtually equivalent to a dot product (which can be computed very quickly on a vector processor). In fact, if all reference vectors have the same magnitude, C_m can be ignored (normalized

vectors have the same magnitude, c_m can be ignored (normalized codebook).

PREWHITENING OF FEATURES

Consider the problem of comparing features of different scales:

Suppose we represent these points in space in two coordinate systems using the transformation:

$$\bar{z} = \underline{V}\bar{x}$$

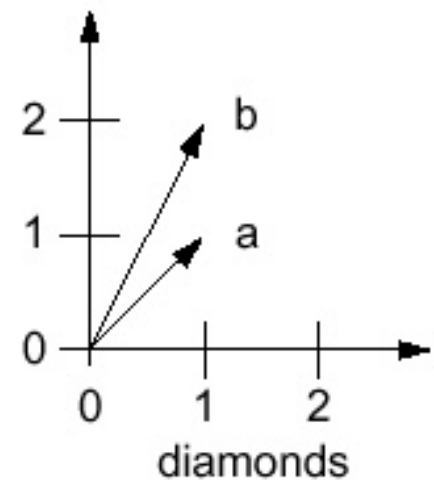
System 1:

$$\beta_1 = 1\hat{i} + 0\hat{j} \text{ and } \beta_2 = 0\hat{i} + 1\hat{j}$$

$$\bar{a} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \bar{b} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$d_2(\bar{a}, \bar{b}) = \sqrt{0^2 + 1^2} = 1$$

gold bars

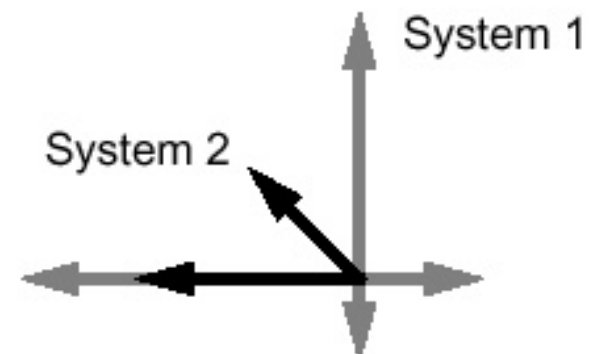


System 2:

$$\gamma_1 = -2\hat{i} + 0\hat{j} \text{ and } \gamma_2 = -1\hat{i} + 1\hat{j}$$

$$\bar{a} = \begin{bmatrix} -2 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix} \quad \bar{b} = \begin{bmatrix} -2 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} -\frac{3}{2} \\ 2 \end{bmatrix}$$

$$d_2(\bar{a}, \bar{b}) = \sqrt{\left(-1 - \left(-\frac{3}{2}\right)\right)^2 + (1 - 2)^2} = \sqrt{\frac{5}{4}}$$



The magnitude of the distance has changed. Though the rank-ordering of distances under such linear transformations won't change, the cumulative effects of such changes in distances can be damaging in pattern recognition. Why?

WEIGHTED EUCLIDEAN DISTANCES REVISITED

We can simplify the distance calculation in the transformed space:

$$\begin{aligned} d_2(\underline{V}\bar{x}, \underline{V}\bar{y}) &= \sqrt{[\underline{V}\bar{x} - \underline{V}\bar{y}]^T [\underline{V}\bar{x} - \underline{V}\bar{y}]} \\ &= \sqrt{[\bar{x} - \bar{y}]^T \underline{V}^T \underline{V} [\bar{x} - \bar{y}]} \\ &= d_{2W}(\bar{x}, \bar{y}) \end{aligned}$$

This is just a weighted Euclidean distance.

Suppose all dimensions of the vector are not equal in importance. For example, suppose one dimension has virtually no variation, while another is very reliable. Suppose two dimensions are statistically correlated. What is a statistically optimal transformation?

Consider a decomposition of the covariance matrix (which is symmetric):

$$\underline{C} = \underline{\Phi} \underline{\Delta} \underline{\Phi}^T$$

where $\underline{\Phi}$ denotes a matrix of eigenvectors of \underline{C} and $\underline{\Delta}$ denotes a diagonal matrix whose elements are the eigenvalues of \underline{C} . Consider:

$$\bar{z} = \underline{\Delta}^{-1/2} \underline{\Phi} \bar{x}$$

The covariance of \bar{z} , $\underline{C}_{\bar{z}}$ is easily shown to be an identity matrix (prove this!)

We can also show that:

$$d_2(\bar{z}_1, \bar{z}_2) = \sqrt{[\bar{x}_1 - \bar{x}_2]^T \underline{C}_{\bar{x}}^{-1} [\bar{x}_1 - \bar{x}_2]}$$

Again, just a weighted Euclidean distance.

- If the covariance matrix of the transformed vector is a diagonal matrix, the transformation is said to be an orthogonal transform.
- If the covariance matrix is an identity matrix, the transform is said to be an orthonormal transform.

- A common approximation to this procedure is to assume the dimensions of \bar{x} are uncorrelated but of unequal variances, and to approximate \underline{C} by a diagonal matrix, $\underline{\Delta}$. Why? This is known as variance-weighting.

NOISE REDUCTION

The prewhitening transform, $z = \underline{\Lambda}^{-1/2} \underline{\Phi} \bar{x}$, is normally created as a $k \times k$ matrix in which the eigenvalues are ordered from largest to smallest:

$$\begin{bmatrix} z_1 \\ z_2 \\ \dots \\ z_k \end{bmatrix} = \begin{bmatrix} \lambda_1^{-1/2} & ? & ? & ? \\ ? & \lambda_2^{-1/2} & ? & ? \\ ? & ? & \dots & ? \\ ? & ? & ? & \lambda_k^{-1/2} \end{bmatrix} \begin{bmatrix} v_{11} & v_{12} & \dots & v_{1k} \\ v_{21} & v_{22} & \dots & v_{2k} \\ \dots & \dots & \dots & \dots \\ v_{k1} & v_{k2} & \dots & v_{kk} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_k \end{bmatrix}$$

where

$$\lambda_1 > \lambda_2 > \dots > \lambda_k.$$

In this case, a new feature vector can be formed by truncating the transformation matrix to $l < k$ rows. This is essentially discarding the least important features.

A measure of the amount of discriminatory power contained in a feature, or a set of features, can be defined as follows:

$$\% \text{ var} = \frac{\sum_{j=1}^l \lambda_j}{\sum_{j=1}^k \lambda_j}$$

This is the percent of the variance accounted for by the first l features.

Similarly, the coefficients of the eigenvectors tell us which dimensions of the input feature vector contribute most heavily to a dimension of the output feature vector. This is useful in determining the “meaning” of a particular feature (for example, the first decorrelated feature often is correlated with the overall spectral slope in a speech recognition system — this is sometimes an indication of the type of microphone).

COMPUTATIONAL ISSUES

Computing a “noise-free” covariance matrix is often difficult. One might attempt to do something simple, such as:

$$c_{ij} = \sum_{n=0}^{N-1} (x_i - \mu_i)(x_j - \mu_j) \text{ and } \mu_i = \sum_{n=0}^{N-1} x_i$$

On paper, this appears reasonable. However, often, the complete set of feature vectors contains valid data (speech signals) and noise (nonspeech signals). Hence, we will often compute the covariance matrix across a subset of the data, such as the particular acoustic event (a phoneme or word) we are interested in.

Second, the covariance matrix is often ill-conditioned. Stabilization procedures are used in which the elements of the covariance matrix are limited by some minimum value (a noise-floor or minimum SNR) so that the covariance matrix is better conditioned.

But how do we compute eigenvalues and eigenvectors on a computer?
One of the hardest things to do numerically! Why?

Suggestion: use a canned routine (see Numerical Recipes in C).

The definitive source is EISPACK (originally implemented in Fortran, now available in C). A simple method for symmetric matrices is known as the Jacobi transformation. In this method, a sequence of transformations are applied that set one off-diagonal element to zero at a time. The product of the subsequent transformations is the eigenvector matrix.

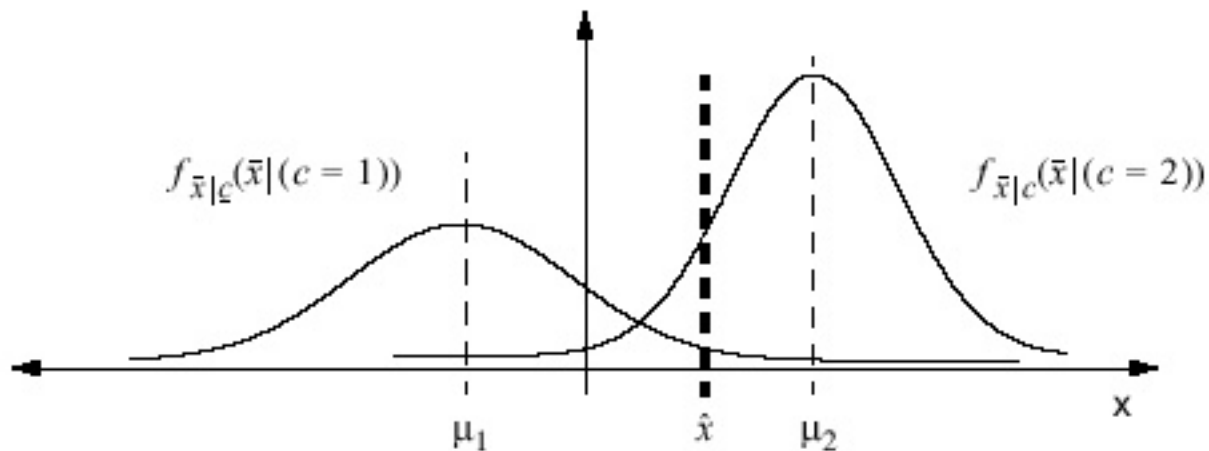
Another method, known as the QR decomposition, factors the covariance matrix into a series of transformations:

$$\underline{C} = \underline{Q}\underline{R}$$

where \underline{Q} is orthogonal and \underline{R} is upper diagonal. This is based on a transformation known as the Householder transform that reduces columns of a matrix below the diagonal to zero.

MAXIMUM LIKELIHOOD CLASSIFICATION

Consider the problem of assigning a measurement to one of two sets:



What is the best criterion for making a decision?

Ideally, we would select the class for which the conditional probability is highest:

$$c^* = \operatorname{argmax}_c P((c = \hat{c}) | (\bar{x} = \hat{\bar{x}}))$$

However, we can't estimate this probability directly from the training data. Hence, we consider:

$$c^* = \operatorname{argmax}_c P((\bar{x} = \hat{\bar{x}}) | (c = \hat{c}))$$

By definition

$$P((c = \hat{c}) | (\bar{x} = \hat{\bar{x}})) = \frac{P((c = \hat{c}), (\bar{x} = \hat{\bar{x}}))}{P(\bar{x} = \hat{\bar{x}})}$$

and

$$P((\bar{x} = \hat{\bar{x}}) | (c = \hat{c})) = \frac{P((c = \hat{c}), (\bar{x} = \hat{\bar{x}}))}{P(c = \hat{c})}$$

from which we have

$$P((c = \hat{c}) | (\bar{x} = \hat{\bar{x}})) = \frac{P((\bar{x} = \hat{\bar{x}}) | (c = \hat{c})) P(c = \hat{c})}{\sum_{c=1}^C P((\bar{x} = \hat{\bar{x}}) | (c = \hat{c})) P(c = \hat{c})}$$

$$P(\bar{x} = \bar{x})$$

SPECIAL CASE: GAUSSIAN DISTRIBUTIONS

Clearly, the choice of c that maximizes the right side also maximizes the left side. Therefore,

$$\begin{aligned} c^* &= \operatorname{argmax}_c [P((\bar{x} = \hat{\bar{x}}) | (c = \hat{c}))] \\ &= \operatorname{argmx}_c [P((\bar{x} = \hat{\bar{x}}) | (c = \hat{c})) P(c = \hat{c})] \end{aligned}$$

if the class probabilities are equal,

$$c^* = \operatorname{argmx}_c [P((\bar{x} = \hat{\bar{x}}) | (c = \hat{c}))]$$

A quantity *related* to the probability of an event which is used to make a decision about the occurrence of that event is often called a *likelihood measure*.

A decision rule that maximizes a likelihood is called a maximum likelihood decision.

In a case where the number of outcomes is not finite, we can use an analogous continuous distribution. It is common to assume a multivariate Gaussian distribution:

$$\begin{aligned} f_{\bar{x}|c}(x_1, \dots, x_N | c) &= f_{\bar{x}|c}(\hat{\bar{x}} | \hat{c}) \\ &= \frac{1}{\sqrt{2\pi} |C_{\bar{x}|c}|} \exp \left\{ -\frac{1}{2} (\hat{\bar{x}} - \bar{\mu}_{\bar{x}|c})^T C_{\bar{x}|c}^{-1} (\hat{\bar{x}} - \bar{\mu}_{\bar{x}|c}) \right\} \end{aligned}$$

We can elect to maximize the log, $\ln[f_{\bar{x}|c}(\bar{x}|c)]$ rather than the likelihood (we refer to this as the log likelihood). This gives the decision rule:

$$c^* = \operatorname{argmin}_c \left[(\hat{\bar{x}} - \bar{\mu}_{\bar{x}|c})^T C_{\bar{x}|c}^{-1} (\hat{\bar{x}} - \bar{\mu}_{\bar{x}|c}) + \ln \left\{ |C_{\bar{x}|c}^{-1}| \right\} \right]$$

(Note that the maximization became a minimization.)

We can define a distance measure based on this as:

we can define a distance measure based on this as:

$$d_{ml}(\bar{x}, \bar{\mu}_{\bar{x}|c}) = (\hat{x} - \bar{\mu}_{\bar{x}|c})^T \underline{C}_{\bar{x}|c}^{-1} (\hat{x} - \bar{\mu}_{\bar{x}|c}) + \ln \left\{ \left| \underline{C}_{\bar{x}|c}^{-1} \right| \right\}$$

THE MAHALANOBIS DISTANCE

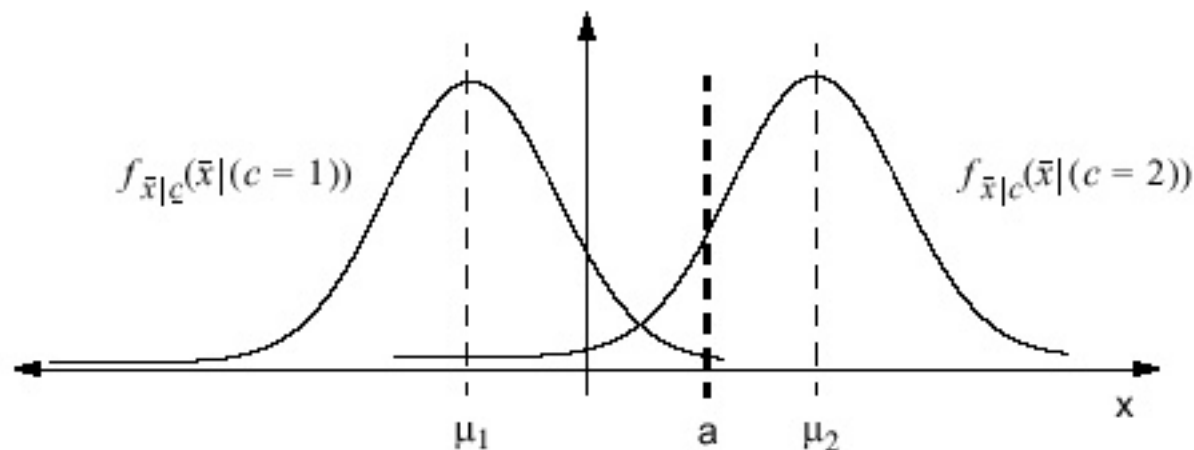
Note that the distance is conditioned on each class mean and covariance. This is why “generic” distance comparisons are a joke.

If the mean and covariance are the same across all classes, this expression simplifies to:

$$d_M(\bar{x}, \bar{\mu}_{\bar{x}|c}) = (\hat{x} - \bar{\mu}_{\bar{x}|c})^T \underline{C}_{\bar{x}|c}^{-1} (\hat{x} - \bar{\mu}_{\bar{x}|c})$$

This is frequently called the *Mahalanobis distance*. But this is nothing more than a weighted Euclidean distance.

This result has a relatively simple geometric interpretation for the case of a single random variable with classes of equal variances:



The decision rule involves setting a threshold:

$$a = \left(\frac{\mu_1 + \mu_2}{2} \right) + \frac{\sigma^2}{\mu_1 - \mu_2} \ln \left(\frac{P(c=2)}{P(c=1)} \right)$$

and,

$$\begin{array}{ll} \text{if} & x < a & x \in (c=1) \\ \text{else} & & x \in (c=2) \end{array}$$

If the variances are not equal, the threshold shifts towards the distribution with the smaller variance.

What is an example of an application where the classes are not

What is an example of an application where the classes are not equiprobable?

[Return to Main](#)[Objectives](#)**Maximum Likelihood:**[Classification](#)[Maximization](#)[Mahalanobis](#)**Discrimination:**[2D Gaussian](#)[Support Region](#)[Decision Regions](#)[Approach](#)[Scatter](#)[Example](#)**On-Line Resources:**[LDA Tutorial](#)[ICA](#)[Statistical Normalization](#)[Pattern Recognition Applet](#)[LNKNET Software](#)

LECTURE 20: LINEAR DISCRIMINANT ANALYSIS

- Objectives:
 - Review maximum likelihood classification
 - Appreciate the importance of weighted distance measures
 - Introduce the concept of discrimination
 - Understand under what conditions linear discriminant analysis is useful

This material can be found in most pattern recognition textbooks. This

is the book we recommend:

R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification* (Second Edition), Wiley Interscience, New York, New York, USA, ISBN: 0-471-05669-3, 2000.

and use in our pattern recognition course. The material in this lecture follows this textbook closely:

J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co., ISBN: 0-7803-5386-2, 2000.

Each of these sources contain references to the seminal publications in this area, including our all-time favorite:

K. Fukunga, *Introduction to Statistical Pattern Recognition*,

MacMillan Publishing Company,
San Diego, California, USA,
ISBN: 0-1226-9851-7, 1990.

LECTURE 20: LINEAR DISCRIMINANT ANALYSIS

- Objectives:
 - Review maximum likelihood classification
 - Appreciate the importance of weighted distance measures
 - Introduce the concept of discrimination
 - Understand under what conditions linear discriminant analysis is useful

This material can be found in most pattern recognition textbooks. This is the book we recommend:

R.O. Duda, P.E. Hart, and D.G. Stork,
Pattern Classification (Second Edition),
Wiley Interscience, New York, New York,
USA, ISBN: 0-471-05669-3, 2000.

and use in our pattern recognition course. The material in this lecture follows this textbook closely:

J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co., ISBN: 0-7803-5386-2, 2000.

Each of these sources contain references to the seminal publications in this area, including our all-time favorite:

K. Fukunga, *Introduction to Statistical Pattern Recognition*, MacMillan Publishing Company, San Diego, California, USA, ISBN: 0-1226-9851-7, 1990.

TWO-DIMENSIONAL GAUSSIAN DISTRIBUTIONS

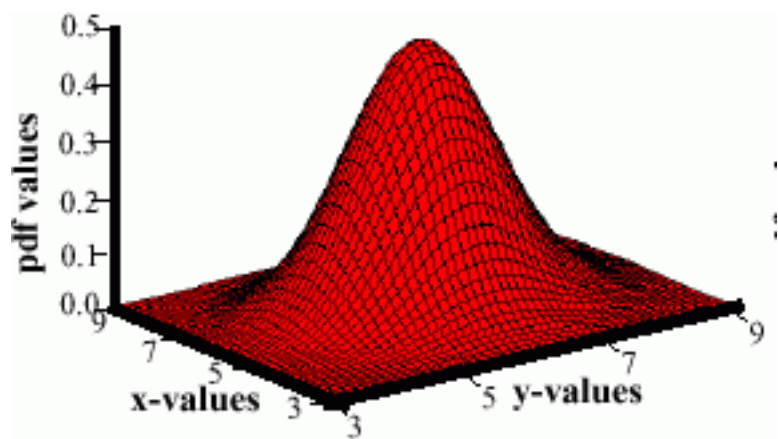
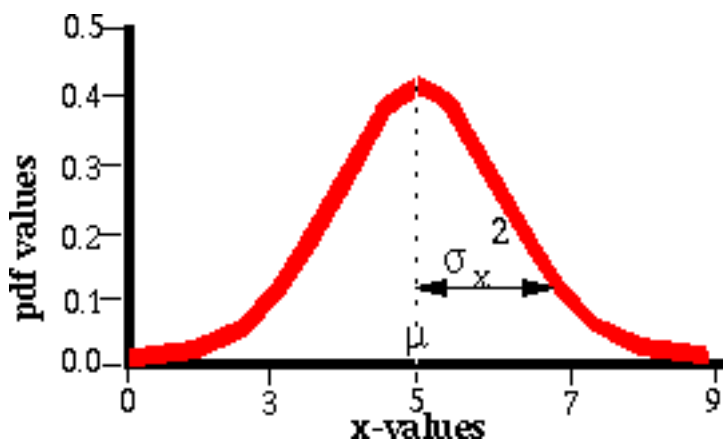
The equation of multivariate Gaussian distribution is as follows:

$$z = p(x,y) = \frac{1}{(2\pi)^{d/2} \sqrt{|C|}} e^{-\frac{1}{2}((x,y) - \mu) C^{-1} ((x,y) - \mu)^T}$$

$$C = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{yx} & \sigma_y^2 \end{bmatrix}$$

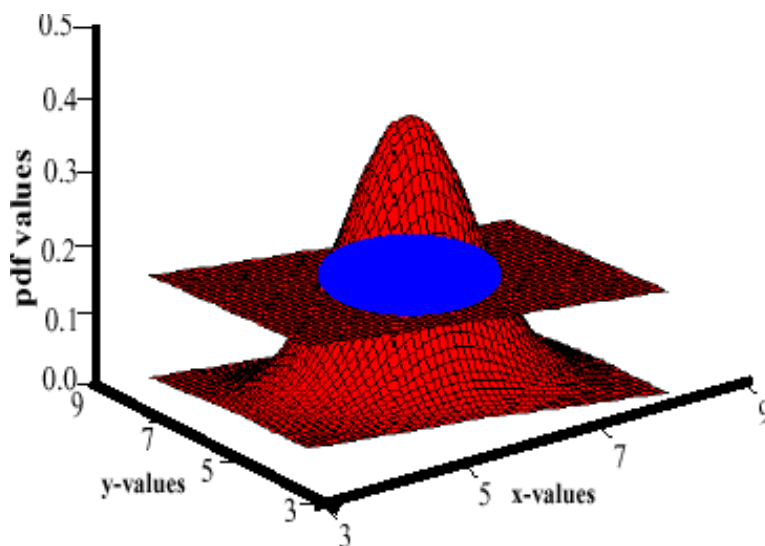
where C is the covariance matrix and d is the dimension of the distribution.

1D Gaussian distribution 2D Gaussian distribution



SUPPORT REGION: A CONVENIENT VISUALIZATION TOOL

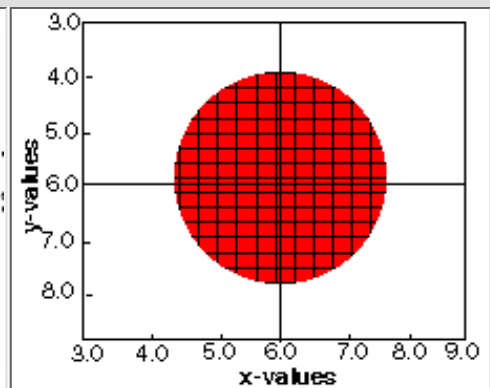
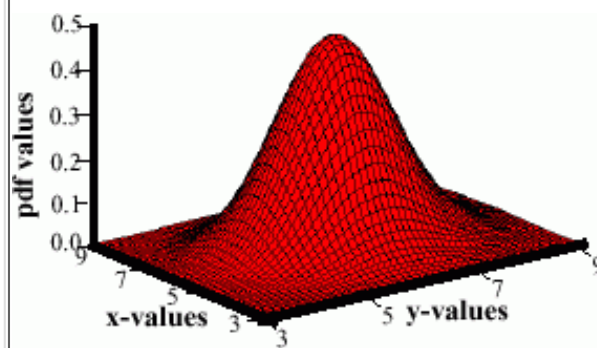
- A support region is the surface defined by the intersection of a Gaussian distribution with a plane.
- The shape of a support region is elliptical and depends on the covariance matrix of the original data.
- A convenient visualization tool in pattern recognition.



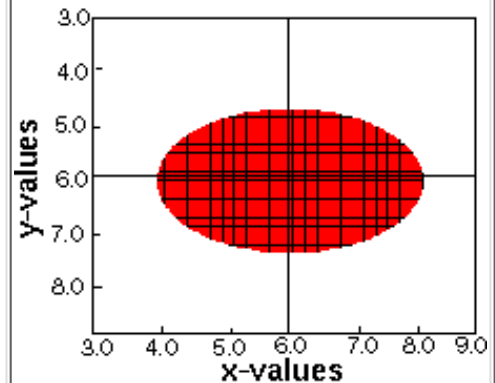
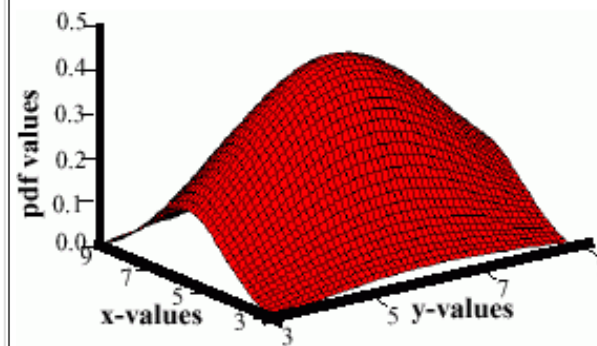
Some examples demonstrating the relationship between the covariance matrix and the 2D Gaussian distribution are shown below:

Identity:

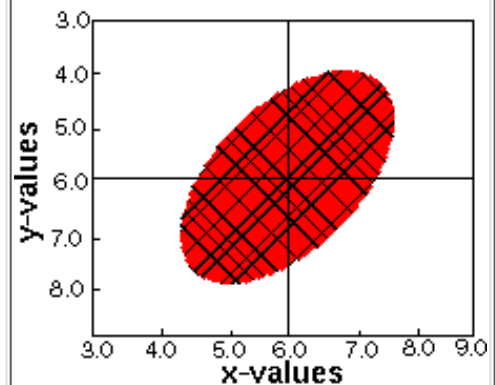
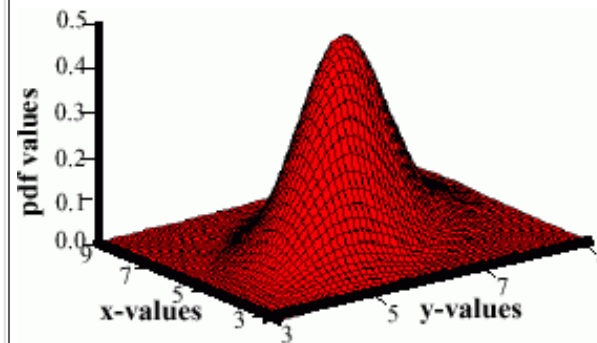
$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

**Unequal Variances:**

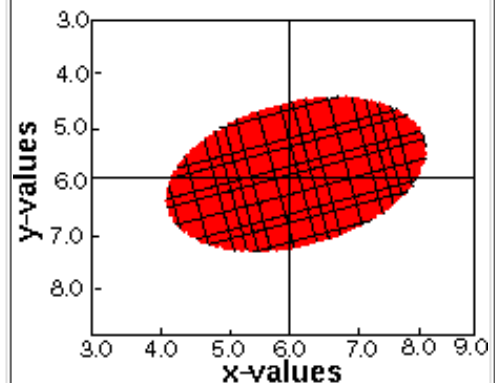
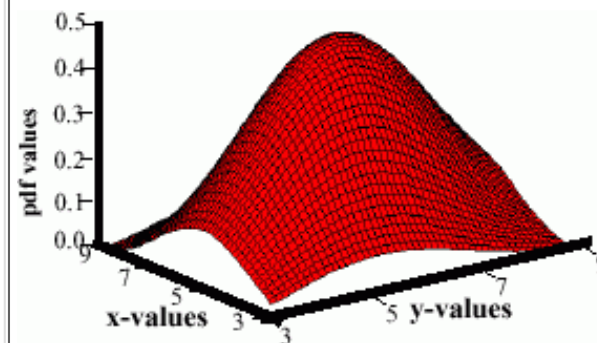
$$C = \begin{bmatrix} 5 & 0 \\ 0 & 2 \end{bmatrix}$$

**Nonzero off-diagonal:**

$$C = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$$

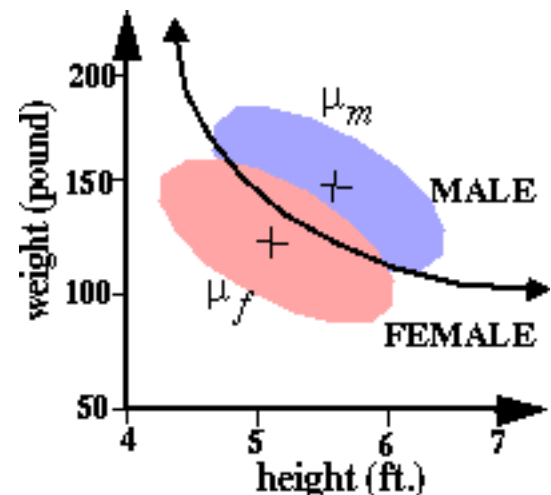
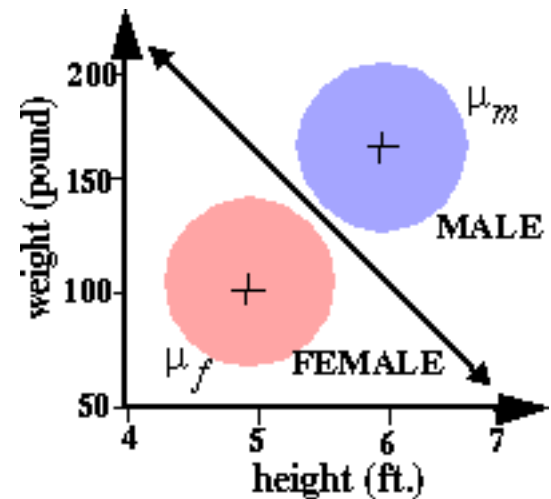
**Unconstrained:**

$$C = \begin{bmatrix} 5 & 0.5 \\ 0.5 & 2 \end{bmatrix}$$



DISTANCE MEASURES AND DECISION REGIONS

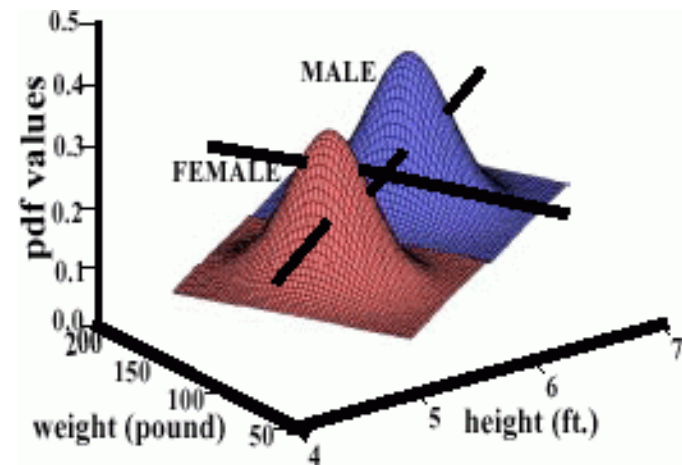
- **Case I: Distributions with equal variances**
- **Decision surface boundary is a line separating the two distributions (general case is a hyperplane).**
- **Case II: Distributions with unequal variances**
- **Direction of greatest variance is not the direction of discrimination.**



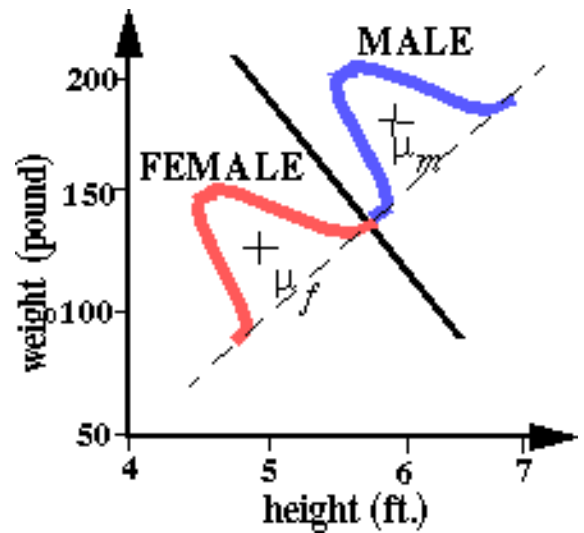
GOAL: MAXIMIZE SEPARABILITY

Principal Component Analysis:
Transform features to a new space in which the features are uncorrelated.

Linear Discriminant Analysis:
Projection of



d -dimensional data onto a line; Dimensionality reduction by mapping L distributions to $(L-1)$ -dimensional subspace; maximize class separability.



OPTIMIZATION CRITERIA BASED ON SCATTER

- Within-class scatter defines the scatter of samples around their respective means.

$$S_w = \sum_{i=1}^L P_i E[(X - \mu_i)(X - \mu_i)^T]$$

- Between-class scatter defines scatter of expected vectors around the global mean.

$$S_b = \sum_{i=1}^L P_i [(\mu_i - \mu_0)(\mu_i - \mu_0)^T]$$

- Mixture-class scatter is the overall scatter obtained from the covariance of all samples:

$$S_m = E[(X - \mu_0)(X - \mu_0)^T] = S_w + S_b$$

where μ_0 is the overall mean.

- Optimizing criteria used to obtain the

transforms is a combination of within-class scatter, between-class scatter and overall scatter.

$$\begin{aligned} \text{criterion} &= \frac{\text{tr}(S_b)}{\text{tr}(S_w)} & \text{criterion} &= S_b - \mu(S_w - c) \\ \text{criterion} &= \text{inv}(S_w) \times S_b & \text{criterion} &= \text{inv}(S_m) \times S_b \end{aligned}$$

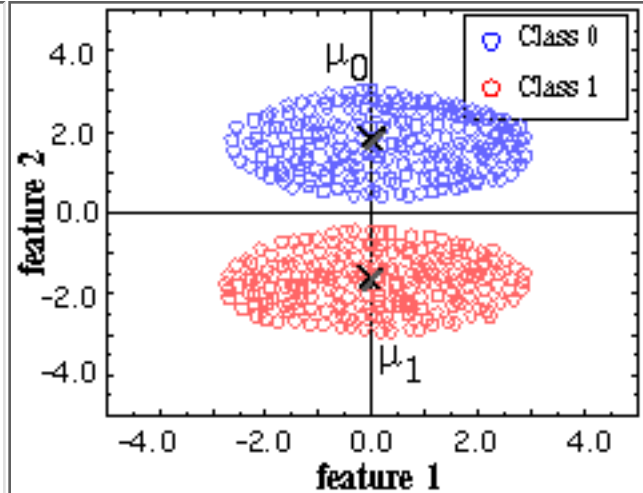
- Transformation matrix is given by:

$$T = \Phi$$

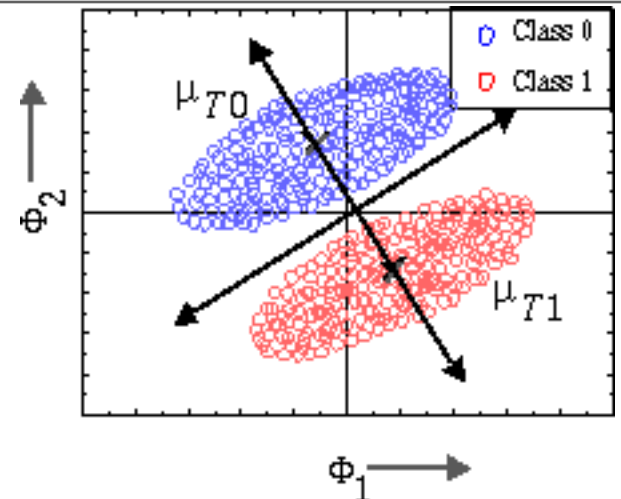
where Φ are the eigenvectors corresponding to non-zero eigenvalues.

A COMPARISON OF PCA AND LDA

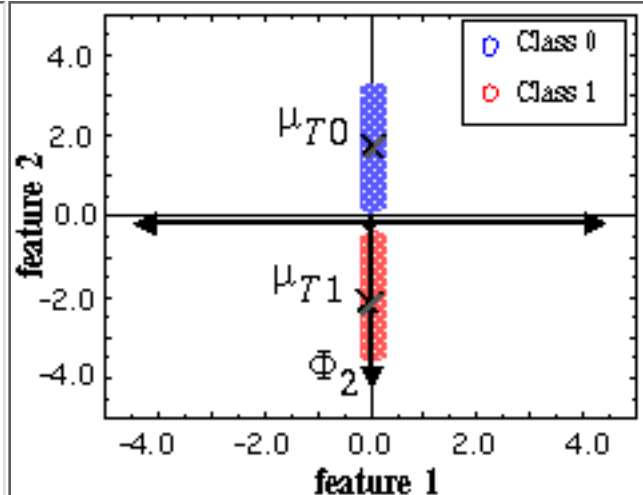
Original Data



Class-Independent PCA



Class-Independent LDA



[Return to Main](#)[Objectives](#)**Overview:**[Technology](#)[Optimal Search](#)**Description:**[Optimality](#)[Example](#)[Variants](#)[Discussion](#)**On-Line Resources:**[Trick: Tutorial](#)[Cassidy: Dynamic Time-Warping](#)[DTW Applet](#)

LECTURE 21: DYNAMIC PROGRAMMING

- Objectives:
 - Historical significance of dynamic programming
 - Introduce a fixed-endpoint solution
 - Understand optimality constraints
 - Explain use in speech recognition

This material can be found in most older speech recognition textbooks. This is the book we recommend:

J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co., ISBN: 0-7803-5386-2, 2000.

For a more detailed description of the use of this technology in speech recognition, see:

H.F. Silverman, D.P. Morgan, "The Application of Dynamic Programming to Connected Speech Recognition," *IEEE Acoustics, Speech, and Signal Processing Magazine*, vol. 7, pp. 6-25, July 1990.

LECTURE 21: DYNAMIC PROGRAMMING

- Objectives:
 - Historical significance of dynamic programming
 - Introduce a fixed-endpoint solution
 - Understand optimality constraints
 - Explain use in speech recognition

This material can be found in most older speech recognition textbooks. This is the book we recommend:

J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co., ISBN: 0-7803-5386-2, 2000.

For a more detailed description of the use of this technology in speech recognition, see:

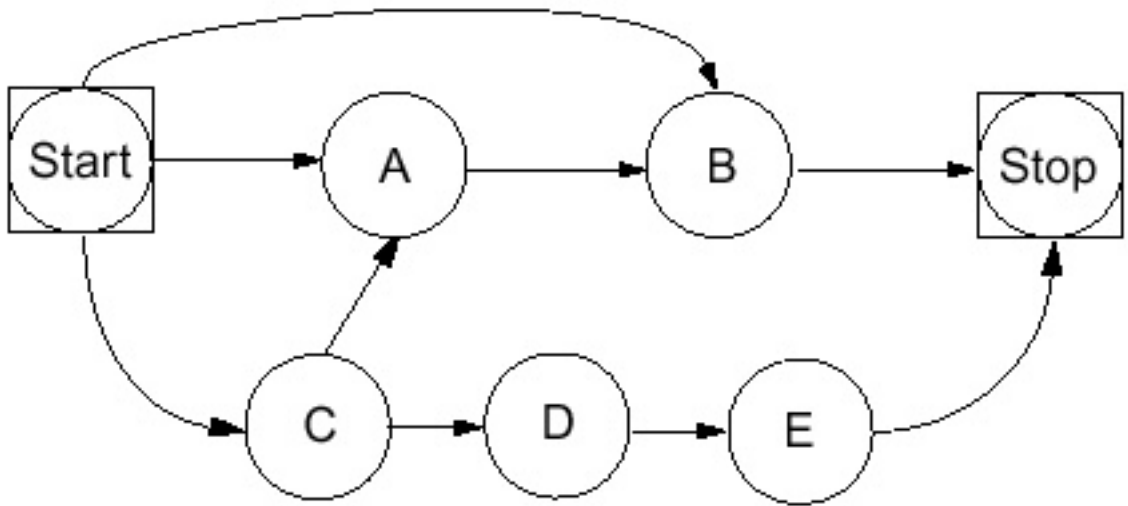
H.F. Silverman, D.P. Morgan, "The Application of Dynamic Programming to Connected Speech Recognition," *IEEE Acoustics, Speech, and Signal Processing Magazine*, vol. 7, pp. 6-25, July 1990.

DYNAMIC PROGRAMMING: FAST BUT OPTIMAL SEARCH

- The search space for vocabularies of hundreds of words can become unmanageable if we allow any word to follow any other word (often called the no-grammar case).
- Our rudimentary knowledge of language tells us that, in reality, only a small subset of the vocabulary can follow a given word hypothesis, but that this subset is sensitive to the given word (we often refer to this as "context-sensitive").
- In real applications, user-interface design is crucial (much like the problem of designing GUI's), and normally results in a specification of a language or collection of sentence

patterns that are permissible.

- A simple way to express and manipulate this information in a dynamic programming framework is via a state machine:



- For example, when you enter state C, you output one of the following words: {daddy, mommy}.

If:

state A: give

state B: me

state C: {daddy, mommy}

state D: come

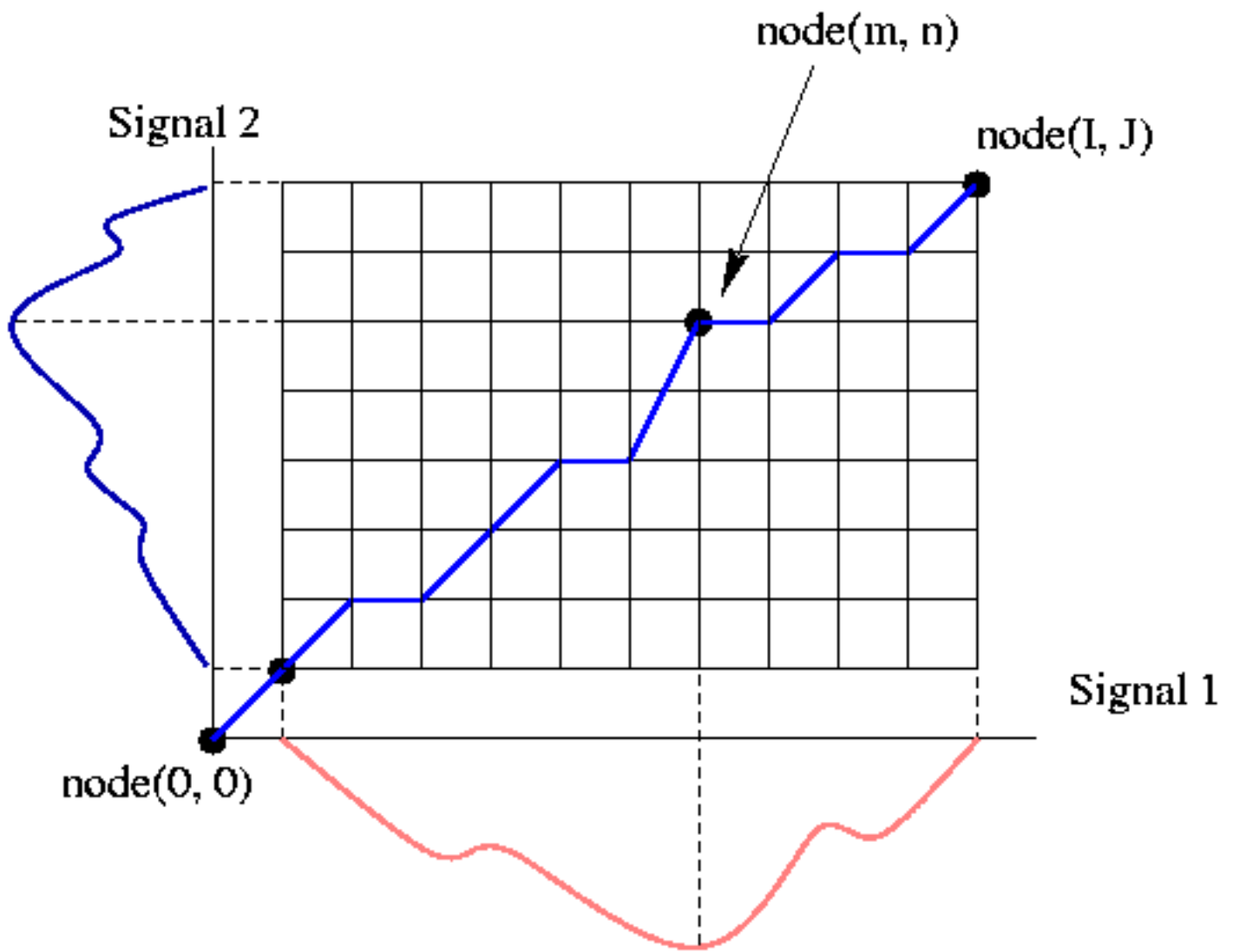
state E: here

With such a state machine, we can generate phrases such as:

Daddy give me

transitions: Start \rightarrow C \rightarrow A \rightarrow B \rightarrow Stop

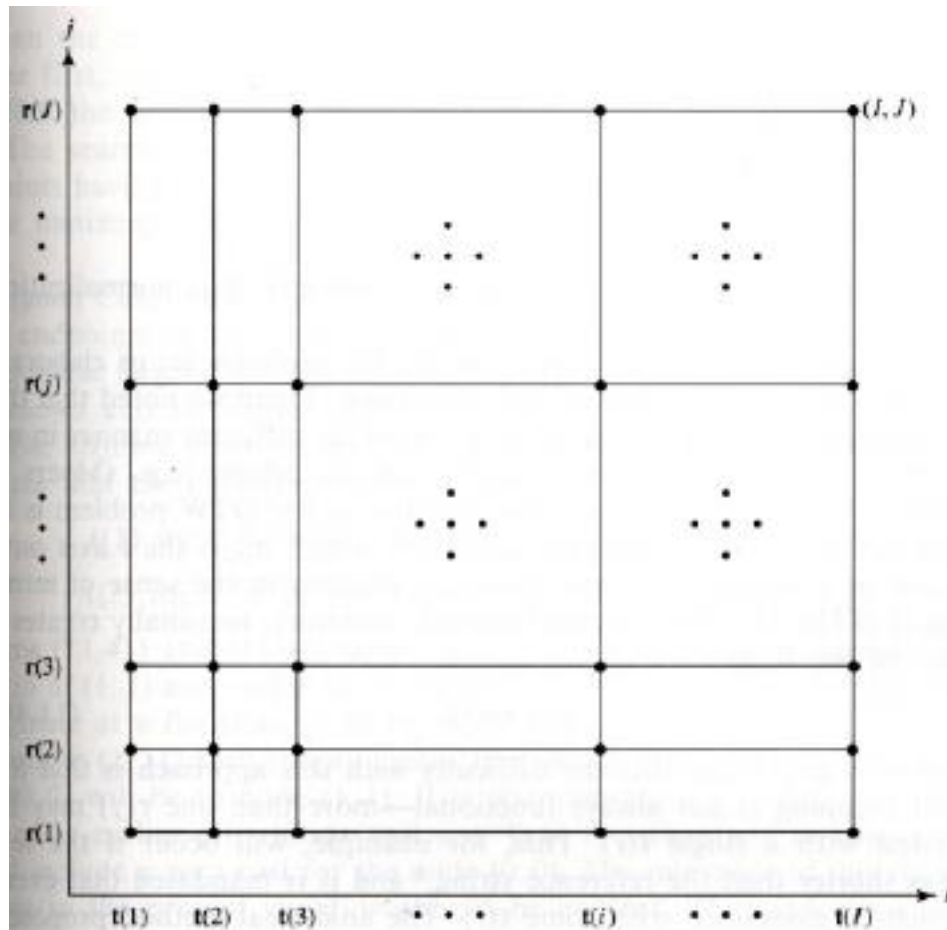
- We can also use this technology to compare feature streams from two signals:



- The latter example is similar to an early approach to speech recognition denoted **dynamic time-warping**. Why might this be useful for speech recognition?

BELLMAN'S PRINCIPLE OF OPTIMALITY

- Consider the discrete space (grid) shown below:



- Define a path from node (s, t) to (u, v) as an n -tuple:

$$(s,t), (i_1,j_1), (i_2,j_2), \dots, (u,v)$$

- Define a distance in moving from node i_{k-1}, j_{k-1} to i_k, j_k as:

$$d[(i_k, j_k) | (i_{k-1}, j_{k-1})] = d_T[(i_k, j_k) | (i_{k-1}, j_{k-1})] + d_N((i_k, j_k))$$

- Define an overall path cost as:

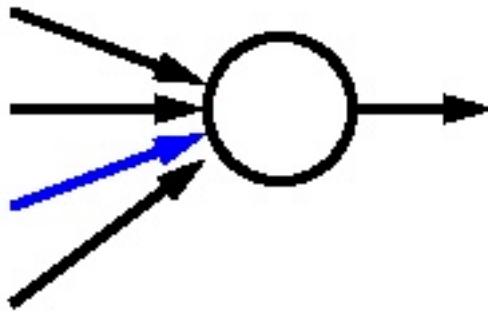
$$D(i, j) = \sum_{k=1}^K d[(i_k, j_k) | (i_{k-1}, j_{k-1})]$$

- **Bellman's Principle of Optimality:**

$$(s, t) \rightarrow (u, v) = (s, t) \rightarrow (w, x) \oplus (w, x) \rightarrow (u, v)$$

for any s, t, u, v, w , and x such that $0 \leq s, w, u \leq I$ and $0 \leq t, x, v \leq J$, where \oplus denotes concatenation of path segments.

- This theorem has remarkable consequences: We do not need to exhaustively search for the best path. Instead, we can build the best path by considering a sequence of partial paths, and retaining the best local path:

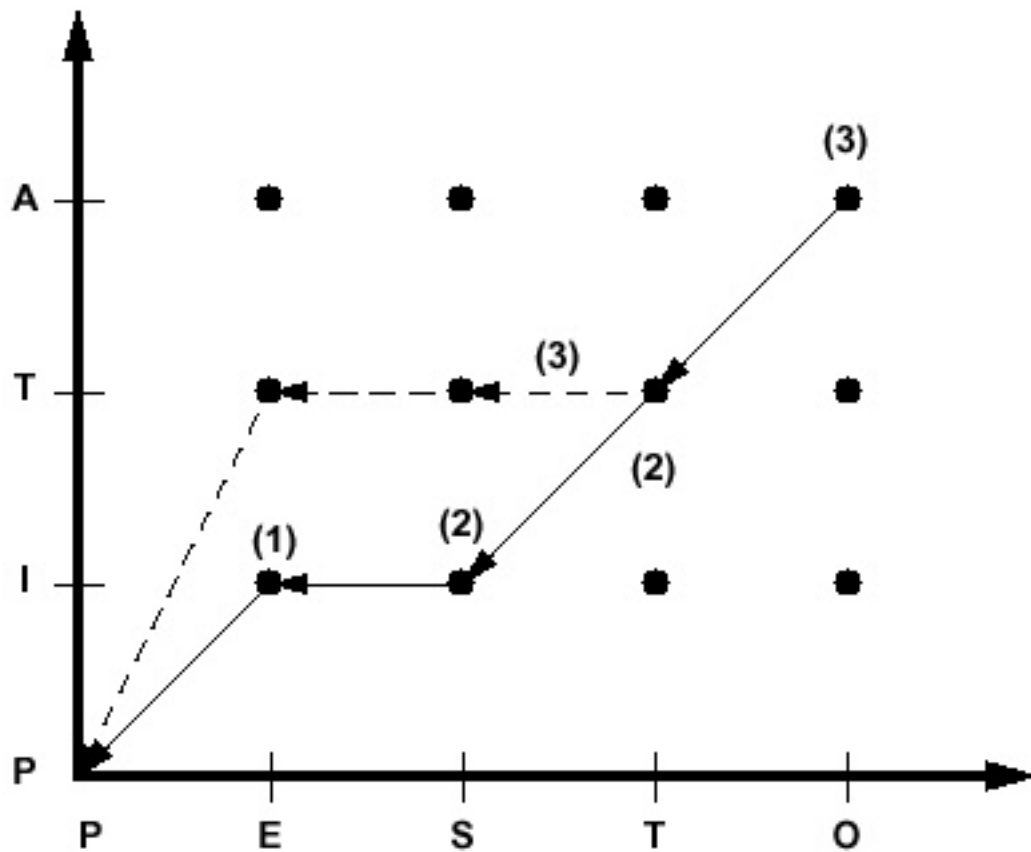


- The savings in computations are enormous: $O(KVL)$ vs. $O(KV^L)$.
- For this reason, dynamic programming is one of the most widely used algorithms in computing. It has been applied to many areas of speech recognition including language

modeling (search) and scoring (string edits).

EXAMPLE: STRING SIMILARITY

- Consider the problem of measuring the distance between two strings below:

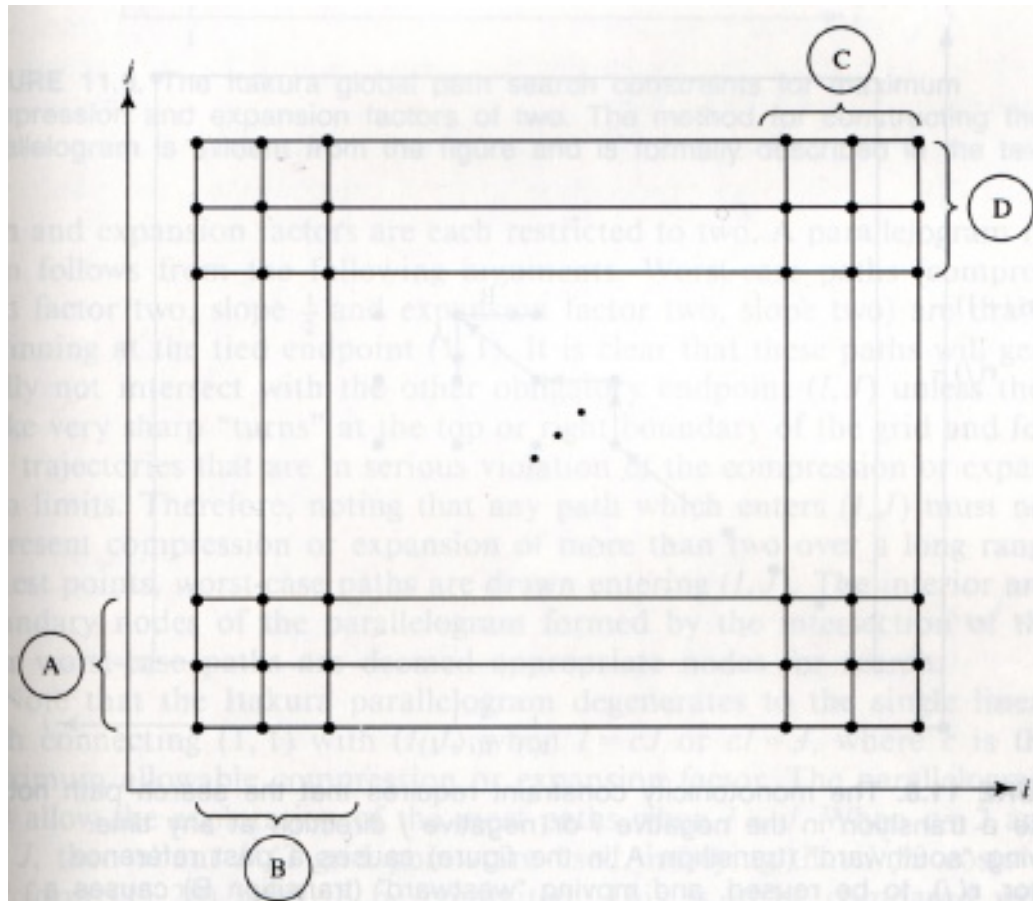


- Transition Penalty:** Any non-diagonal transition has a penalty of 1 unit.

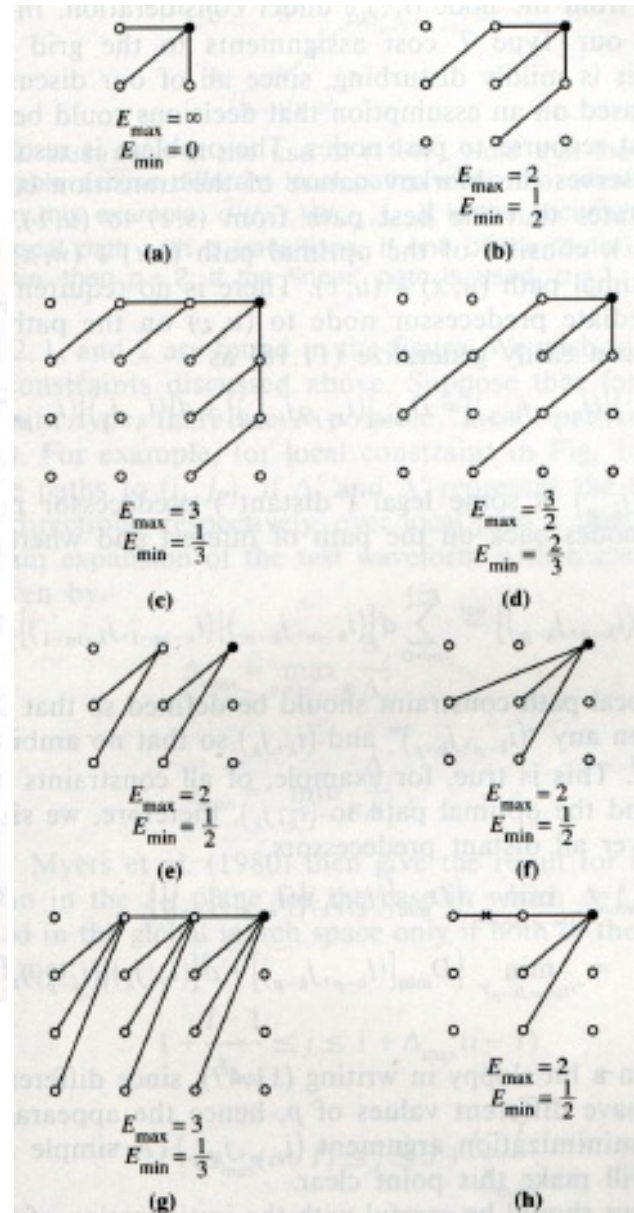
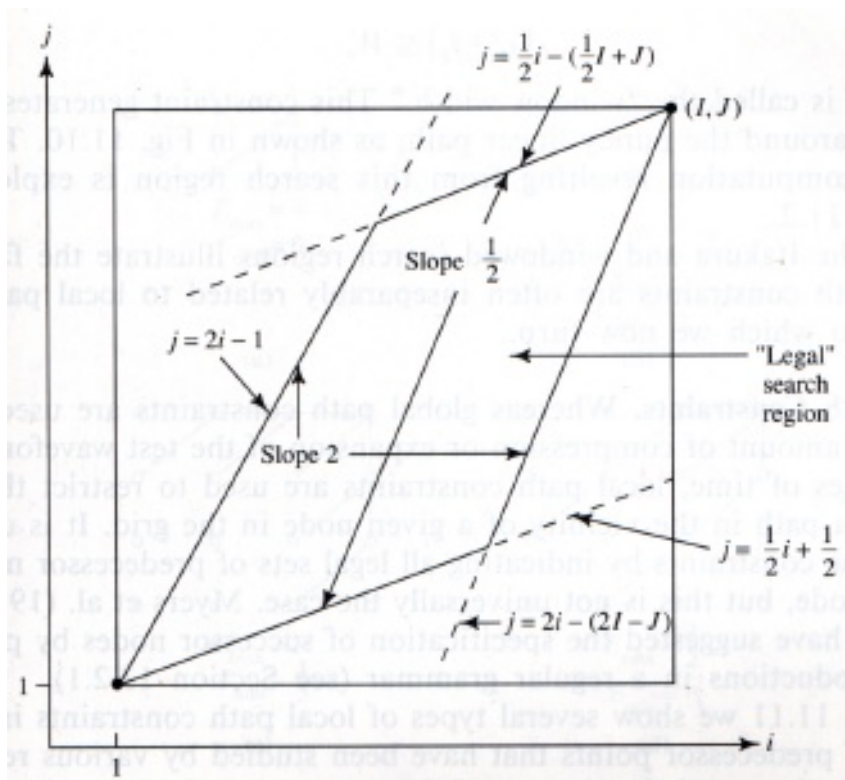
- **Node Penalty:** Any two dissimilar letters that are matched at a node incur a penalty of 1.
- How much memory do we need if we just want to keep the best score?
- When would we want to retain the backpointers at each node?
- How can we constrain the search space to make this algorithm more efficient for speech processing?

COMMON VARIANTS OF THE DYNAMIC PROGRAMMING FRAMEWORK

- Previously, we demonstrated **fixed-endpoint** DP.
- Relaxed endpoint and free endpoint solutions are also possible:



- Since the speech signal (or text) flows left to right, we can impose "slope constraints" to improve performance and decrease computation:



- The slope constraints are implemented by limiting the search space and imposing transition penalties.
- What aspects of the speech signal influence the design of the slope constraints?

DISCUSSION: HOW DO WE APPLY DYNAMIC PROGRAMMING TO SPEECH RECOGNITION?

- The optimal solution is only as good as the cost function. What is an appropriate cost function for speech recognition? What features should we use?
- How should we train our reference models ("templates")?
- How can we attempt continuous speech recognition using this approach?
(Hints: [Word Spotting](#), [Level Building](#), [Bridle Algorithm](#))

[Return to Main](#)[Objectives](#)**Introduction:**[Example](#)[Calculations](#)**Definitions:**[Hidden States](#)[Doubly Stochastic](#)[Basic Model Elements](#)[Matrix Solutions](#)[Likelihood](#)**On-Line Resources:**[OGI: HLT Survey](#)[Cambridge: HTK](#)[MS State: ASR](#)[Berkeley: Matlab HMM](#) [UIUC: Multimodal](#)

LECTURE 22: FUNDAMENTALS OF MARKOV MODELS

- Objectives:
 - Introduce a Markov model
 - Understand the difference between an observable and a hidden Markov model
 - Appreciate the reason we use Markov models: to model temporal evolution of the spectrum (important in speech recognition!)
 - Demonstrate basic calculations
 - Demonstrate the infeasibility of these basic calculations for real problems

This material can be found in most speech recognition and pattern recognition textbooks. These notes follow material presented in:

J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co., ISBN: 0-7803-5386-2, 2000.

Another useful reference is:

L.R. Rabiner and B.W. Juang, *Fundamentals of Speech Recognition*, Prentice-Hall, ISBN: 0-13-015157-2, 1993.

The course textbook also follows these traditional references closely.

LECTURE 22: FUNDAMENTALS OF MARKOV MODELS

- Objectives:
 - Introduce a Markov model
 - Understand the difference between an observable and a hidden Markov model
 - Appreciate the reason we use Markov models: to model temporal evolution of the spectrum (important in speech recognition!)
 - Demonstrate basic calculations
 - Demonstrate the infeasibility of these basic

calculations for real problems

This material can be found in most speech recognition and pattern recognition textbooks. These notes follow material presented in:

J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co., ISBN: 0-7803-5386-2, 2000.

Another useful reference is:

L.R. Rabiner and B.W. Juang, *Fundamentals of Speech Recognition*, Prentice-Hall, ISBN: 0-13-015157-2, 1993.

The course textbook also follows these traditional references closely.

A SIMPLE MARKOV MODEL FOR WEATHER PREDICTION

What is a first-order Markov chain?

$$P[q_t = j | (q_{t-1} = i, q_{t-2} = k, \dots)] = P[q_t = j | q_{t-1} = i]$$

We consider only those processes for which the right-hand side is independent of time:

$$a_{ij} = P[q_t = j | q_{t-1} = i] \quad 1 \leq i, j \leq N$$

with the following properties:

$$a_{ij} \geq 0 \quad \forall j, i$$

$$\sum_{j=1}^N a_{ij} = 1 \quad \forall i$$

The above process can be considered observable because the output process is a set of states at each instant of time, where each state corresponds to an observable event.

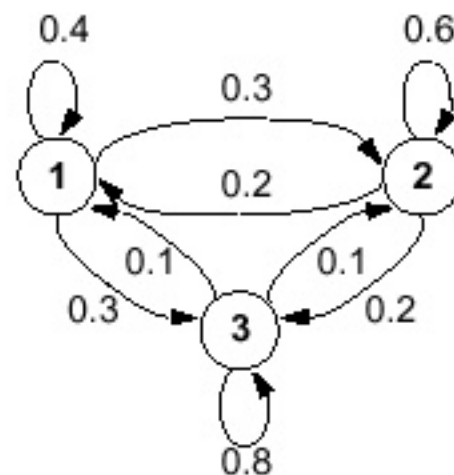
Later, we will relax this constraint, and make the output related to the states by a second random process.

Example: A three-state model of the weather

State 1: precipitation (rain, snow, hail, etc.)

State 2: cloudy

State 3: sunny



BASIC CALCULATIONS

Example: What is the probability that the weather for eight consecutive days is "sun-sun-sun-rain-rain-sun-cloudy-sun"?

Solution:

O = sun sun sun rain rain sun cloudy sun
 3 3 3 1 1 3 2 3

$$\begin{aligned} P(\bar{O} | Model) &= P[3]P[3|3]P[3|3]P[1|3]P[1|1]P[3|1]P[2|3]P[3|2] \\ &= \pi_3 a_{33} a_{31} a_{11} a_{13} a_{32} a_{23} \\ &= 1.536 \times 10^{-4} \end{aligned}$$

Example: Given that the system is in a known state, what is the probability that it stays in that state for d days?

O = i i i ... i j

$$\begin{aligned} P(\bar{O} | Model, q_1 = i) &= P(\bar{O}, q_1 = i | Model) / P(q_1 = i) \\ &= \pi_i a_{ii}^{d-1} (1 - a_{ii}) / \pi_i \\ &= a_{ii}^{d-1} (1 - a_{ii}) \\ &= p_i(d) \end{aligned}$$

Note the exponential character of this distribution.

We can compute the expected number of observations in a state given that we started in that state:

$$\bar{d}_i = \sum_{d=1}^{\infty} d p_i(d) = \sum_{d=1}^{\infty} d a_{ii}^{d-1} (1 - a_{ii}) = \frac{1}{1 - a_{ii}}$$

Thus, the expected number of consecutive sunny days is $(1/(1-0.8)) = 5$; the expected number of cloudy days is 2.5, etc.

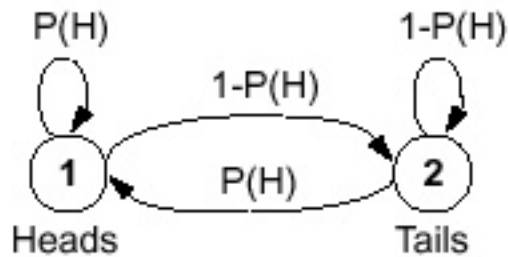
What have we learned from this example?

"HIDDEN" MARKOV MODELS

Consider the problem of predicting the outcome of a coin toss experiment.
You observe the following sequence:

$$\bar{O} = (HHTTTHTTH...H)$$

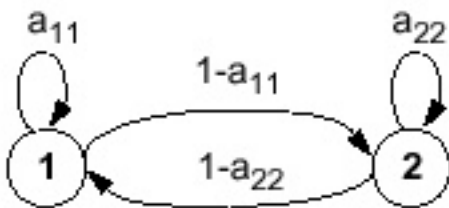
What is a reasonable model of the system?



1-Coin Model
(Observable Markov Model)

O = H H T T H T H H T T H ...

S = 1 1 2 2 1 2 1 1 2 2 1 ...



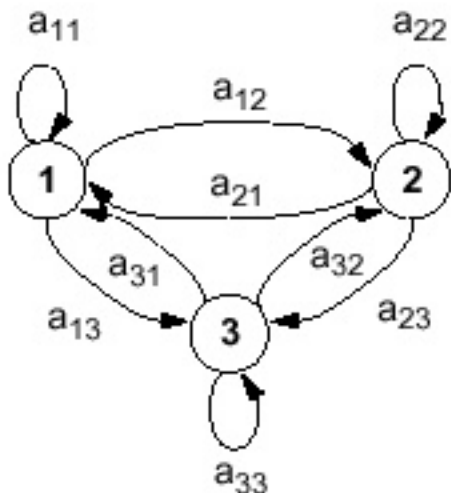
2-Coins Model
(Hidden Markov Model)

O = H H T T H T H H T T H ...

S = 1 1 2 2 1 2 1 1 2 2 1 ...

$P(H) = P_1$
 $P(T) = 1 - P_1$

$P(H) = P_2$
 $P(T) = 1 - P_2$



3-Coins Model
(Hidden Markov Model)

O = H H T T H T H H T T H ...

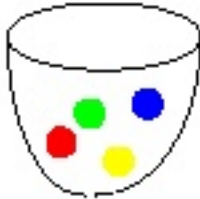
S = 3 1 2 3 3 1 1 2 3 1 3 ...

P(H): P_1 P_2 P_3

$$P(T): \quad 1-P_1 \quad 1-P_2 \quad 1-P_3$$

DOUBLY STOCHASTIC SYSTEMS

The Urn-and-Ball Model



$$P(\text{red}) = b_1(1)$$

$$P(\text{green}) = b_1(2)$$

$$P(\text{blue}) = b_1(3)$$

$$P(\text{yellow}) = b_1(4)$$

...



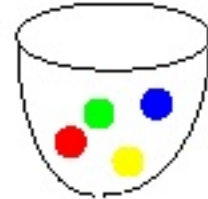
$$P(\text{red}) = b_2(1)$$

$$P(\text{green}) = b_2(2)$$

$$P(\text{blue}) = b_2(3)$$

$$P(\text{yellow}) = b_2(4)$$

...



$$P(\text{red}) = b_3(1)$$

$$P(\text{green}) = b_3(2)$$

$$P(\text{blue}) = b_3(3)$$

$$P(\text{yellow}) = b_3(4)$$

...

$$\vec{O} = \{\text{green, blue, green, yellow, red, ..., blue}\}$$

How can we determine the appropriate model for the observation sequence given the system above?

BASIC ELEMENTS OF A HIDDEN MARKOV MODEL

- N — the number of states
- M — the number of distinct observations per state
- The state-transition probability distribution $\underline{A} = \{a_{ij}\}$
- The output probability distribution $\underline{B} = \{b_j(k)\}$
- The initial state distribution $\pi = \{\pi_i\}$

We can write this succinctly as: $\lambda = (\underline{A}, \underline{B}, \pi)$

Note that the probability of being in any state at any time is completely determined by knowing the initial state and the transition probabilities:

$$\pi(t) = \underline{A}^{t-1} \pi$$

Two basic problems:

- (1) how do we train the system?
- (2) how do we estimate the probability of a given sequence (recognition)?

This gives rise to a third problem:

If the states are hidden, how do we know what states were used to generate a given output?

How do we represent continuous distributions (such as feature vectors)?

MATRIX CALCULATIONS FOR DISCRETE HMMS

The *discrete observation* HMM is restricted to the production of a finite set of discrete observations (or sequences). The output distribution at any state is given by:

$$b(k, i) \equiv P(\underline{y}(t) = k | \underline{x}(t) = i)$$

The observation probabilities are assumed to be independent of time. We can write the probability of observing a particular observation, $\underline{y}(t)$, as:

$$b(\underline{y}(t) | i) \equiv P(\underline{y}(t) = \underline{y}(t) | \underline{x}(t) = i)$$

The observation probability distribution can be represented as a matrix whose dimension is K rows x S states.

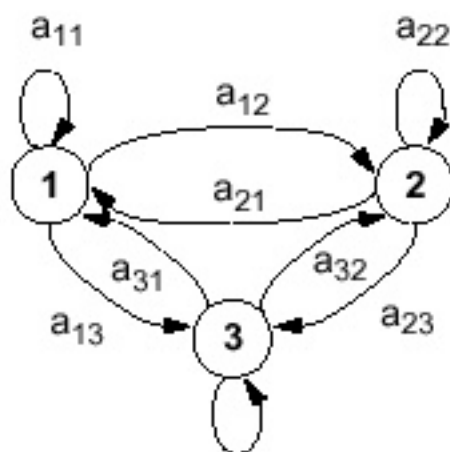
We can define the observation probability vector as:

$$p(t) = \begin{bmatrix} P(\underline{y}(t) = 1) \\ P(\underline{y}(t) = 2) \\ \dots \\ P(\underline{y}(t) = K) \end{bmatrix}, \quad \text{or,} \quad p(t) = \mathbf{B}\pi(t) = \mathbf{B}\mathbf{A}^{t-1}\pi(1)$$

The mathematical specification of an HMM can be summarized as:

$$\mathbf{M} = \{S, \pi(1), \mathbf{A}, \mathbf{B}, \{y_k, 1 \leq k \leq K\}\}$$

For example, reviewing our coin-toss model:



$$S = 3$$

$$\pi(1) = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

a_{33}

| | | | |
|-------|---------|---------|---------|
| P(H): | P_1 | P_2 | P_3 |
| P(T): | $1-P_1$ | $1-P_2$ | $1-P_3$ |

$$\mathbf{B} = \begin{bmatrix} P_1 & P_2 & P_3 \\ 1-P_1 & 1-P_2 & 1-P_3 \end{bmatrix}$$

RECOGNITION USING DISCRETE HMMS

Denote any partial sequence of observations in time by:

$$y_{t_1}^{t_2} \equiv \{y(t_1), y(t_1 + 1), y(t_1 + 2), \dots, y(t_2)\}$$

The forward partial sequence of observations at time t is

$$y_1^t \equiv \{y(1), y(2), \dots, y(t)\}$$

The backward partial sequence of observations at time t is

$$y_{t+1}^T \equiv \{y(t+1), y(t+2), \dots, y(T)\}$$

A complete set of observations of length T is denoted as $y \equiv y_1^T$.

What is the likelihood of an HMM?

We would like to calculate $P(M|\underline{y} = y)$ — however, we can't. We can (see the introductory notes) calculate $P(\underline{y} = y|M)$. Consider the brute force method of computing this. Let $\vartheta = \{i_1, i_2, \dots, i_T\}$ denote a specific state sequence. The probability of a given observation sequence being produced by this state sequence is:

$$P(y|\vartheta, M) = b(y(1)|i_1)b(y(2)|i_2)\dots b(y(T)|i_T)$$

The probability of the state sequence is

$$P(\vartheta|M) = P(x(1) = i_1)a(i_2|i_1)a(i_3|i_2)\dots a(i_T|i_{T-1})$$

Therefore,

$$P(y, (\vartheta|M)) = P(x(1) = i_1)a(i_2|i_1)a(i_3|i_2)\dots a(i_T|i_{T-1}) \\ \times b(y(1)|i_1)b(y(2)|i_2)\dots b(y(T)|i_T)$$

To find $P(y|M)$, we must sum over all possible paths:

$$P(y|M) = \sum_{\forall \vartheta} P(y, (\vartheta|M))$$

This requires $O(2TS^T)$ flops. For $S = 5$ and $T = 100$, this gives about 1.6×10^{72} computations per HMM.

1.6×10^8 computations per MVM!

[Return to Main](#)[Objectives](#)**Review:**[Elements](#)**Calculations:**[HMM Evaluation](#)[The Forward Algorithm](#)[The Viterbi Algorithm](#)[The EM Algorithm](#)**On-Line Resources:**[Parameter Estimation](#)[Baum Welch](#)[Expectation Maximization \(EM\)](#)

LECTURE 23: PARAMETER ESTIMATION

- Objectives:
 - Review the components of a hidden Markov model
 - Forward probability calculation
 - The Viterbi algorithm
 - The Expectation Maximization (EM) algorithm
 - Understand the use of these techniques in HMM training and decoding

This lecture combines material from

the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

and information found in most standard speech textbooks:

J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co., ISBN: 0-7803-5386-2, 2000.

LECTURE 23: PARAMETER ESTIMATION

- Objectives:
 - Review the components of a hidden Markov model
 - Forward probability calculation
 - The Viterbi algorithm
 - The Expectation Maximization (EM) algorithm
 - Understand the use of these techniques in HMM training and decoding

This lecture combines material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

and information found in most standard speech textbooks:

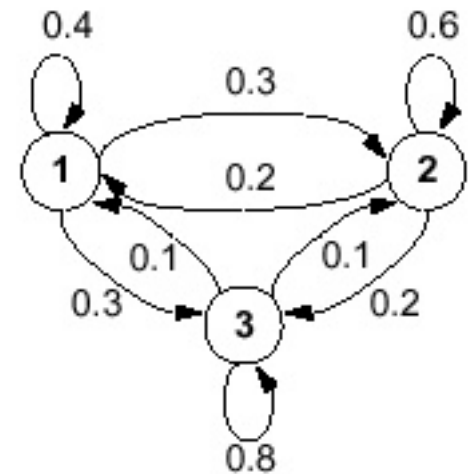
J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co., ISBN: 0-7803-5386-2, 2000.

ELEMENTS OF AN HIDDEN MARKOV MODEL

Recall the elements of a hidden Markov Model:

- N — the number of states
- M — the number of distinct observations per state
- The state-transition probability distribution $\underline{A} = \{a_{ij}\}$
- The output probability distribution $\underline{B} = \{b_j(k)\}$
- The initial state distribution $\pi = \{\pi_i\}$

We can write this succinctly as: $\lambda = (\underline{A}, \underline{B}, \pi)$



There are three problems we must solve:

- The Evaluation Problem: Given a model and a set of observations, what was the probability that the model produced these observations?
- The Decoding Problem: What was the most likely state sequence in the model that produced a given set of observations?

- The Learning Problem: Given a model and a set of observations, how can we adjust the model parameters to increase the probability of the observations (data) given the model?

HMM EVALUATION (EXHAUSTIVE SEARCH)

Denote any partial sequence of observations in time by:

$$y_{t_1}^{t_2} \equiv \{y(t_1), y(t_1 + 1), y(t_1 + 2), \dots, y(t_2)\}$$

The forward partial sequence of observations at time t is

$$y_1^t \equiv \{y(1), y(2), \dots, y(t)\}$$

The backward partial sequence of observations at time t is

$$y_{t+1}^T \equiv \{y(t+1), y(t+2), \dots, y(T)\}$$

A complete set of observations of length T is denoted as $y \equiv y_1^T$.

What is the likelihood of an HMM?

We would like to calculate $P(\lambda | \underline{y} = y)$ — however, as we will explain later, we can't. Instead, we can calculate $P(\underline{y} = y | \lambda)$. Consider the brute force method of computing this. Let $\vartheta = \{i_1, i_2, \dots, i_T\}$ denote a specific state sequence. The probability of a given observation sequence being produced by this state sequence is:

$$P(y | \vartheta, \lambda) = b(y(1) | i_1) b(y(2) | i_2) \dots b(y(T) | i_T)$$

The probability of the state sequence is

$$P(\vartheta | \lambda) = P(x(1) = i_1) a(i_2 | i_1) a(i_3 | i_2) \dots a(i_T | i_{T-1})$$

Therefore,

$$P(y, (\vartheta | \lambda)) = P(x(1) = i_1) a(i_2 | i_1) a(i_3 | i_2) \dots a(i_T | i_{T-1}) \\ \times b(y(1) | i_1) b(y(2) | i_2) \dots b(y(T) | i_T)$$

To find $P(y | \lambda)$, we must sum over all possible paths:

$$P(y | \lambda) = \sum_{\forall \vartheta} P(y, (\vartheta | \lambda))$$

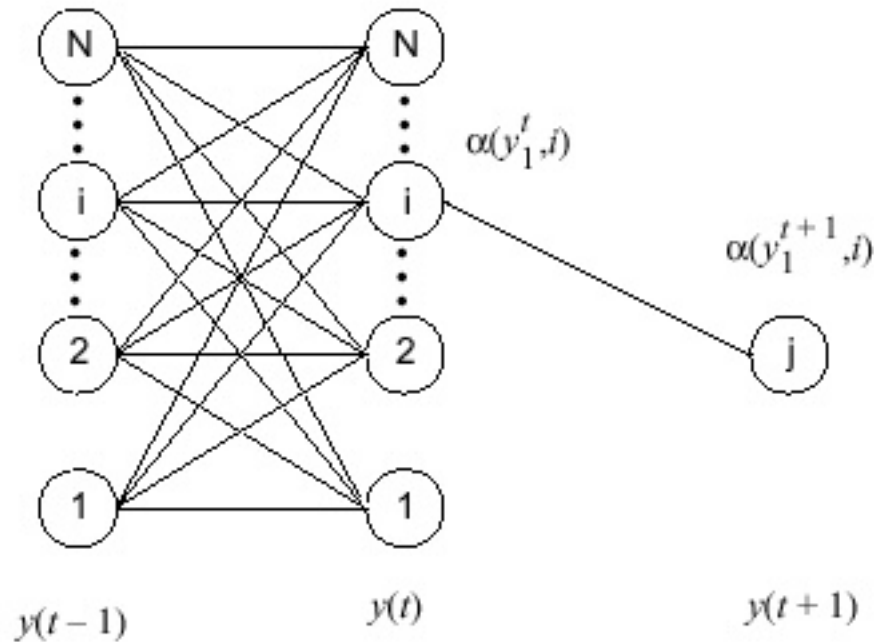
This requires $O(2TN^I)$ flops. For $N = 5$ and $T = 100$, this gives about 1.6×10^{72} computations per HMM!

THE FORWARD ALGORITHM

The *forward algorithm* begins by defining a “forward-going” partial probability sequence:

$$\alpha(y_1^t) \equiv P(\underline{y}_1^t = y_1^t, \underline{x}(t) = i | \lambda)$$

Let us next consider the contribution to the overall sequence probability made by a single transition:



$$\begin{aligned} \alpha(y_1^{t+1}, j) &= \alpha(y_1^t, i) P(\underline{x}(t+1) = j | \underline{x}(t) = i) \times \\ &\quad P(y(t+1) = y(t+1) | \underline{x}(t+1) = j) \\ &= \alpha(y_1^t, i) a(j|i) b(y(t+1)|j) \end{aligned}$$

Summing over all possibilities for reaching state “j”:

$$\alpha(y_1^{t+1}, j) = \sum_{i=1}^N \alpha(y_1^t, i) a(j|i) b(y(t+1)|j)$$

The recursion is initiated by setting:

$$\alpha(y_1^t, j) = P(\underline{x}(1) = j) b(y(1)|j)$$

We still need to find $P(y|\lambda)$:

$$P(y, \underline{x}(t) = i | \lambda) = \alpha(y_1^t, i) \beta(y_{t+1}^T | i)$$

for any state i . Therefore,

$$P(y|\lambda) = \sum_{i=1}^N \alpha(y_1^T, i)$$

These equations suggest a recursion in which, for each value of t we iterate over ALL states and update $\alpha(y_1^t, j)$. When $t = T$, $P(y|\lambda)$ is computed by summing over ALL states:

The Forward Algorithm

Step 1: Initialization

$$\alpha(y_1^1, j) = \pi_j b(y(1)|j) \quad 1 \leq j \leq N$$

Step 2: Induction

$$\alpha(y_1^{t+1}, j) = \left[\sum_{i=1}^N \alpha(y_1^t, i) a(j|i) \right] b(y(t+1)|j)$$

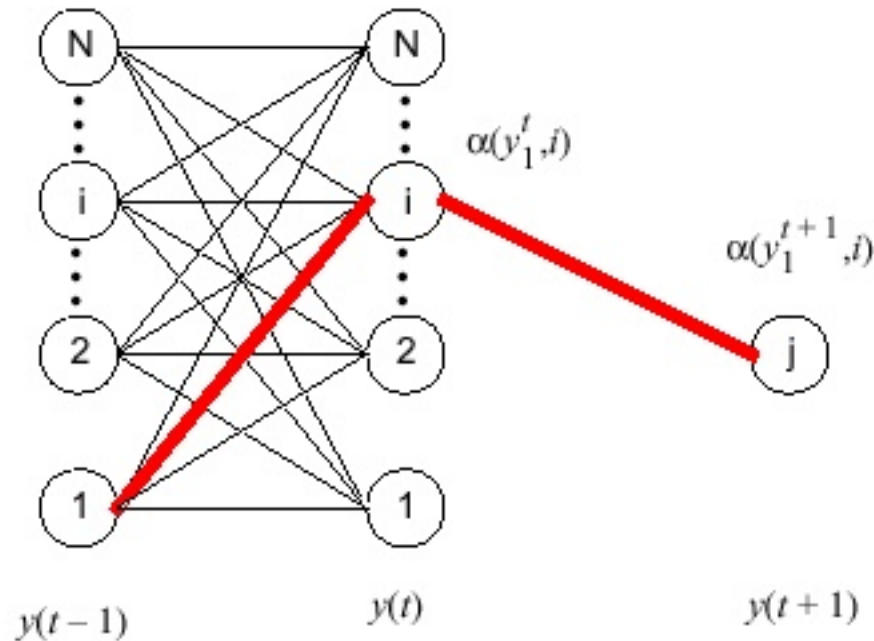
Step 3: Termination

$$P(y|\lambda) = \sum_{i=1}^N \alpha(y_1^T, i)$$

The complexity of this algorithm is $O(N^2T)$, or for $N = 5$ and $T = 100$, approximately 2500 flops are required (compared to 10^{72} flops for the exhaustive search method).

THE VITERBI (BEST PATH) ALGORITHM

The *Viterbi algorithm* can also be used to find the best state sequence. Note that the principal difference is that we model the overall sequence probability by the probability of the single best path:



The Viterbi Algorithm

Step 1: Initialization

$$V_1(j) = \pi_j b(y(1)|j) \quad 1 \leq j \leq N$$

$$B_1(j) = 0$$

Step 2: Induction

$$V_t(j) = \underset{1 \leq i \leq N}{\text{Max}} [V_{t-1}(i) a_{ij}] b(y(t)|j) \quad 1 \leq j \leq N \quad 2 \leq t \leq T$$

$$B_t(j) = \underset{1 \leq i \leq N}{\text{Argmax}}_i [V_{t-1}(i) a_{ij}]$$

Step 3: Termination

$$P(y|\lambda) = \underset{1 \leq i \leq N}{\text{Max}} [V_T(i)] \quad s_t^* = \underset{1 \leq i \leq N}{\text{Argmax}}_i [B_T(i)]$$

Step 4: Backtracking

$$s_t^* = B_{t+1}(s_{t+1}^*) \quad S^* = (s_T^*, s_{T-1}^*, \dots, s_1^*)$$

Note that the complexity for the Viterbi algorithm is $O(N^2T)$. Often it is much less than this because the topology being scored is extremely restrictive (left-to-right models).

THE EXPECTATION MAXIMIZATION (EM) ALGORITHM

- The Expectation Maximization (EM) algorithm can be viewed as a generalization of maximum likelihood parameter estimation (MLE) when the data observed is incomplete. We observe some data y , and seek to maximize $P(Y = y|\lambda)$. However, in order to do this, we need to know some hidden data x .
- We assume a parameter vector λ and estimate the probability that each x occurred in the generation of y . In this way, we can assume we observed the pair (x, y) with probability $P(X = x, Y = y|\lambda)$.
- To compute the new $\bar{\lambda}$, we use the maximum likelihood estimate of λ .
- Does this process converge?

According to Bayes' rule:

$$P(X = x, Y = y|\bar{\lambda}) = P(X = x|Y = y, \bar{\lambda})P(Y = y|\bar{\lambda})$$

The log likelihood can be expressed as:

$$\log P(Y = y|\bar{\lambda}) = \log P(X = x, Y = y|\bar{\lambda}) - \log P(X = x|Y = y, \bar{\lambda})$$

We take the conditional expectation of $\log P(Y = y|\bar{\lambda})$ over X :

$$\begin{aligned} E_{\lambda}[\log P(Y = y|\bar{\lambda})]_{X|Y=y} &= \sum_x (P(X = x|Y = y, \lambda)) \log P(Y = y|\bar{\lambda}) \\ &= \log P(Y = y|\bar{\lambda}) \end{aligned}$$

Combining the previous two expressions:

$$\begin{aligned} \log P(Y = y|\bar{\lambda}) &= E_{\lambda}[\log P(X, Y = y|\bar{\lambda})]_{X|Y=y} - E_{\lambda}[\log P(X|Y = y, \bar{\lambda})]_{X|Y=y} \\ &= Q(\lambda, \bar{\lambda}) - H(\lambda, \bar{\lambda}) \end{aligned}$$

The convergence of the EM algorithm lies in the fact that if we choose $\bar{\lambda}$ such that $Q(\lambda, \bar{\lambda}) \geq Q(\lambda, \lambda)$, then $\log P(Y = y|\bar{\lambda}) \geq \log P(Y = y|\lambda)$.

This follows because we can show that $H(\lambda, \bar{\lambda}) \leq H(\lambda, \lambda)$ using a special case

of Jensen's inequality ($\sum_x p(x) \log p(x) \geq \sum_x p(x) \log q(x)$).

A summary of the procedure is:

The EM Algorithm

Step 1: Choose an initial estimate λ .

Step 2: **E-step**: Compute auxiliary Q -function $Q(\lambda, \bar{\lambda})$ (which is also the expectation of the log likelihood of the data) based on λ .

Step 3: **M-step**: Compute $\hat{\lambda} = \arg \max Q(\lambda, \bar{\lambda})$ to maximize the auxiliary Q -function.

Step 4: Iteration: Set $\lambda = \bar{\lambda}$, and repeat from Step 2 until convergence.

[Return to Main](#)[Objectives](#)**Review:**[Three Fundamental Problems](#) [The Forward Algorithm](#)**Forward Backward:**[Probability Calculations](#)[Transitions](#)[Application of EM](#)[Reestimation Equations](#)[The Forward-Backward Algorithm](#)**On-Line Resources:**[HLTSurvey: HMM](#)[Cassidy: SR](#)[Software: Discrete HMMs](#)

LECTURE 24: HMM TRAINING

● Objectives:

- Pose parameter reestimation as an unsupervised learning problem
- Introduce the Baum-Welch (Forward-Backward) Algorithm
- Apply EM algorithm to reestimate parameters
- Describe Viterbi training

This lecture combines material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5,

2001.

and information found in most standard speech textbooks:

J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co., ISBN: 0-7803-5386-2, 2000.

LECTURE 24: HMM TRAINING

- Objectives:
 - Pose parameter reestimation as an unsupervised learning problem
 - Introduce the Baum-Welch (Forward-Backward) Algorithm
 - Apply EM algorithm to reestimate parameters
 - Describe Viterbi training

This lecture combines material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken*

Language Processing - A Guide to Theory, Algorithm, and System Development, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

and information found in most standard speech textbooks:

J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co., ISBN: 0-7803-5386-2, 2000.

THREE FUNDAMENTAL PROBLEMS

Recall there are three fundamental problems we must solve:

- The Evaluation Problem: Given a model and a set of observations, what was the probability that the model produced these observations?
- The Decoding Problem: What was the most likely state sequence in the model that produced a given set of observations?
- The Learning Problem: Given a model and a set of observations, how can we adjust the model parameters to increase the probability of the observations (data) given the model?

In this lecture, we will focus on the third problem.

PROBABILITY CALCULATIONS

Just as the *forward algorithm* computed probabilities recursively:

$$\alpha(y_1^{t+1}, j) = \left[\sum_{i=1}^N \alpha(y_1^t, i) a(j|i) \right] b(y(t+1)|j)$$

we can similarly define a “backward-going” probability calculation:

$$\beta(y_{t+1}^T | i) \equiv P(y_{t+1}^T = y_{t+1}^T | \mathbf{x}(t) = i, \lambda)$$

This probability can be computed recursively:

$$\beta(y_{t+1}^T | i) = \sum_{j=1}^N \beta(y_{t+2}^T | j) a(j|i) b(y(t+1)|j)$$

This recursion is initialized by:

$$\beta(y_T^T | i) \equiv \begin{cases} 1/N, & \text{if } i \text{ is a legal final state} \\ 0, & \text{otherwise} \end{cases}$$

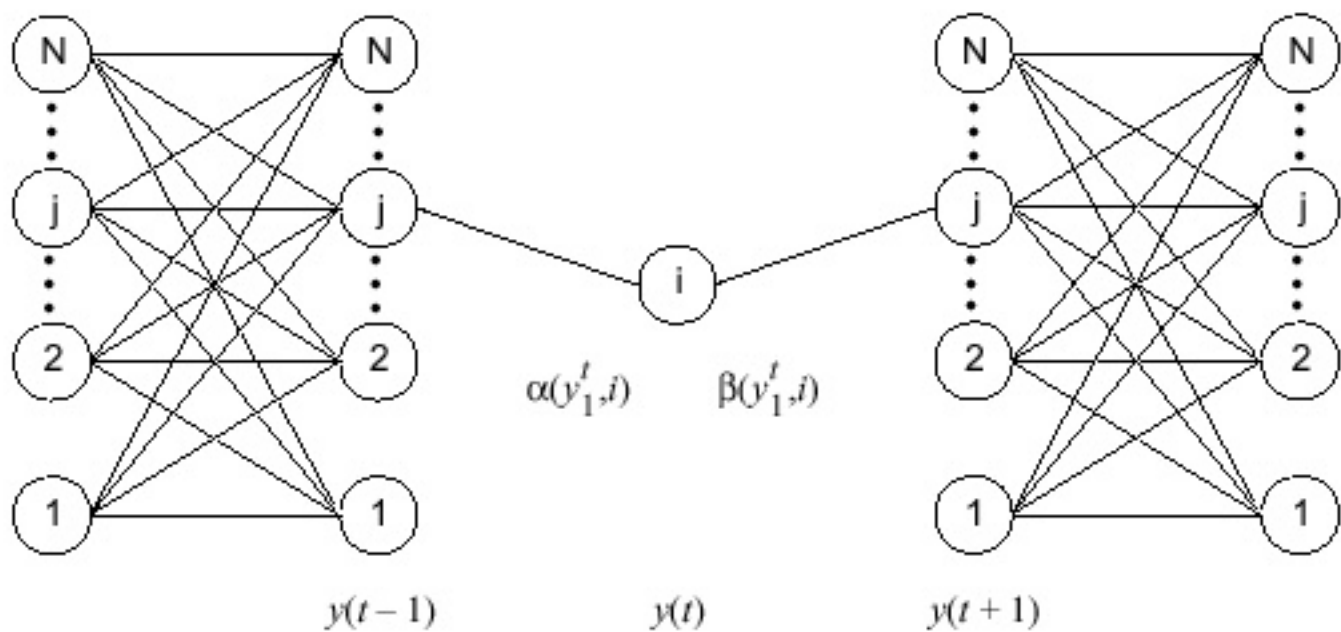
We can now compute $P(y|\lambda)$ in terms of α and β :

$$P(y, \mathbf{x}(t) = i | \lambda) = \alpha(y_1^t, i) \beta(y_{t+1}^T | i)$$

for any state i . Therefore,

$$P(y|\lambda) = \sum_{i=1}^N \alpha(y_1^t, i) \beta(y_{t+1}^T | i)$$

The calculation can be visualized below:



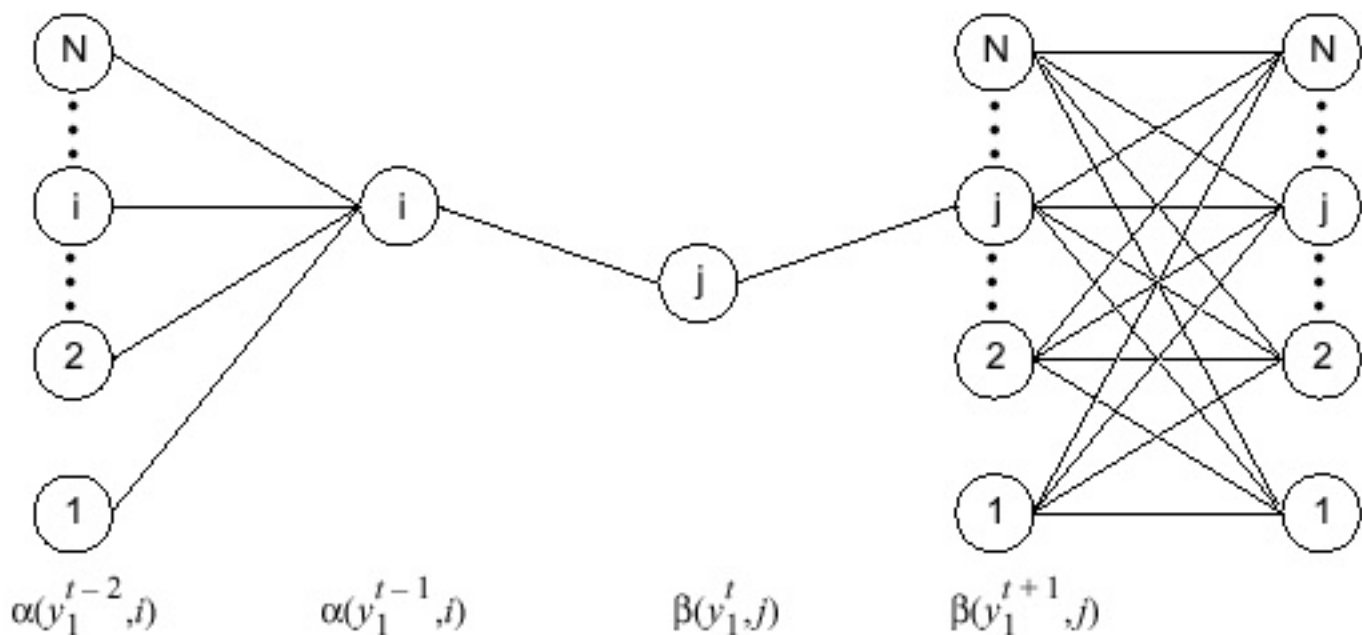
Note that these computations consider all possible state transitions that could have generated the output sequence. This is why this calculation is more expensive than the Viterbi algorithm.

TRANSITION PROBABILITY CALCULATIONS

Define $\gamma_t(j|i)$ as the probability of taking the transition from state i to state j at time t given the model and observation sequence:

$$\begin{aligned}\gamma_t(j|i) &= P(s_{t-1} = i, s_t = j | y_1^T, \lambda) \\ &= \frac{P(s_{t-1} = i, s_t = j, y_1^T | \lambda)}{P(y_1^T | \lambda)} \\ &= \frac{\alpha(y_1^{t-1}, i) a(j|i) b_t(y_t) \beta(y_t^T, j)}{\sum_{k=1}^N \alpha(y_1^T, k)}\end{aligned}$$

This computation is illustrated below:



APPLICATION OF THE EM ALGORITHM

According to the EM algorithm, the maximization process is equivalent to maximizing:

$$Q(\lambda, \bar{\lambda}) = \sum_{i=1}^S \frac{P(y_1^T, S | \lambda)}{P(y_1^T | \lambda)} \log P(y_1^T, S | \bar{\lambda})$$

where:

$$P(y_1^T, S | \lambda) = \prod_{t=1}^T a(s_t | s_{t-1}) b_t(y_t)$$

and

$$\log P(y_1^T, S | \lambda) = \sum_{t=1}^T \log(a(s_t | s_{t-1})) + \sum_{t=1}^T \log(b_t(y_t))$$

This can be rewritten as:

$$Q(\lambda, \bar{\lambda}) = Q_a(\lambda, \hat{a}) + Q_b(\hat{b})$$

where:

$$Q_a(\lambda, \hat{a}) = \sum_{i=1}^N \sum_{j=1}^N \sum_{t=1}^T \frac{P(y_1^T, s_{t-1}=i, s_t=j | \lambda)}{P(y_1^T | \lambda)} \log \hat{a}(j|i)$$

and:

$$Q_b(\lambda, \hat{b}) = \sum_{j=1}^N \sum_{k=1}^M \sum_{t \in (y^t = o^t)} \frac{P(y_1^T, s_t=j | \lambda)}{P(y_1^T | \lambda)} \log \hat{b}_j(k)$$

These terms can be maximized separately under the constraints:

$$\sum_{j=1}^N a(j|i) = 1 \quad \text{and} \quad \sum_{k=1}^M b_j(k) = 1 \quad \forall \text{all } i, j$$

ESTIMATION EQUATIONS

The functions in our auxiliary functions Q are of the form $F(x) = \sum_i y_i \log x_i$,

where $\sum_i x_i = 1$. Using Lagrange multipliers, this function can be shown to

have a maximum value at $x_i = y_i / \sum_i y_i$. The model reestimation equations that result from this optimization are:

$$\hat{a}(j|i) = \frac{\frac{1}{P(y_1^T|\lambda)} \sum_{t=1}^T P(y_1^T, s_{t-1}=i, s_t=j|\lambda)}{\frac{1}{P(y_1^T|\lambda)} \sum_{t=1}^T P(y_1^T, s_{t-1}=i|\lambda)} = \frac{\sum_{t=1}^T \gamma_t(j|i)}{\sum_{t=1}^T \sum_{k=1}^N \gamma_t(k|i)}$$

This is just the ratio of the expected number of transitions from state i to state j and the expected number of transitions from state i .

Similarly,

$$\hat{b}_j(k) = \frac{\frac{1}{P(y_1^T|\lambda)} \sum_{t=1}^T P(y_1^T, s_t=j|\lambda) \delta(y^t, o^t)}{\frac{1}{P(y_1^T|\lambda)} \sum_{t=1}^T P(y_1^T, s_t=j|\lambda)} = \frac{\sum_{t \in (y^t = o^t)} \sum_{i=1}^M \gamma_t(j|i)}{\sum_{t=1}^T \sum_{i=1}^M \gamma_t(j|i)}$$

This is the ratio of the number of times the k^{th} observation vector was emitted from state j and the number of times any observation vector was emitted from state j .

THE FORWARD-BACKWARD ALGORITHM

The Forward Backward Algorithm

Step 1: Initialization: Choose an initial estimate λ .

Step 2: E-step: Compute auxiliary function $Q(\lambda, \hat{\lambda})$ based on λ .

Step 3: M-step: Compute $\bar{\lambda}$ according to the equations for \hat{a} and \hat{b} to maximize the auxiliary Q -function.

Step 4: Iteration: Set $\lambda = \bar{\lambda}$ and repeat steps 2 and 3 until convergence.

The previous derivation assumed one data sequence. To train an HMM from M data sequences, we simply maximize the joint likelihood:

$$\prod_{i=1}^M P(Y_i | \lambda)$$

We can define a partial update for the r^{th} data sequence:

$$\hat{a}(j|i) = \frac{\sum_{r=1}^R \sum_{t=1}^{T^r} \gamma_t^r(j|i)}{\sum_{r=1}^R \sum_{t=1}^{T^r} \sum_{k=1}^N \gamma_t^r(k|i)}$$

The latter equation is important since we can use it to implement HMM training in parallel. We simply run (R/N) files on N processors, and accumulate the sums inside the first summation. After all N jobs complete, we can combine the outputs for the final estimates of the new parameters.

[Return to Main](#)[Objectives](#)**Review:**[The EM Algorithm](#)[Reestimation Equations](#)**Mixtures:**[Continuous Densities](#)[Mixture Densities](#)[Probability Calculations](#)[Application of EM](#)[Reestimation Equations](#)**On-Line Resources:**[Mixture Modeling](#)[Semicontinuous HMMs](#)[Ten Years of HMMs](#)

LECTURE 25: CONTINUOUS MIXTURE DENSITIES

- Objectives:
 - Review the EM algorithm
 - Introduce continuous mixture densities
 - Understand why they are useful in speech recognition
 - Develop reestimation equations for the parameters of a mixture density

This lecture combines material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

and information found in most standard speech textbooks:

J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co., ISBN: 0-7803-5386-2, 2000.

LECTURE 25: CONTINUOUS MIXTURE DENSITIES

- Objectives:
 - Review the EM algorithm
 - Introduce continuous mixture densities
 - Understand why they are useful in speech recognition
 - Develop reestimation equations for the parameters of a mixture density

This lecture combines material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken*

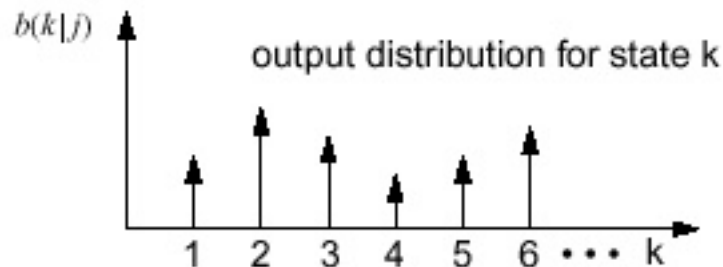
Language Processing - A Guide to Theory, Algorithm, and System Development, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

and information found in most standard speech textbooks:

J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co., ISBN: 0-7803-5386-2, 2000.

CONTINUOUS PROBABILITY DENSITY FUNCTIONS

The discrete HMM incorporates a discrete probability density function, captured in the matrix B , to describe the probability of outputting a symbol:



Signal measurements, or feature vectors, are continuous-valued N-dimensional vectors. In order to use our discrete HMM technology, we must vector quantize (VQ) this data — reduce the continuous-valued vectors to discrete values chosen from a set of M codebook vectors. Initially, most HMMs were based on VQ front-ends. However, for the past 15 years or so, the continuous density model has become widely accepted.

The likelihood of generating observation $y(t)$ in state j is defined as:

$$b(y(t)|j) \equiv f_{y|x}(y(t)|j)$$

Note that taking the negative logarithm of $b(\cdot)$ will produce a log-likelihood, or a Mahalanobis-like distance. But what form should we choose for $f(\cdot)$?

Let's assume a Gaussian model, of course:

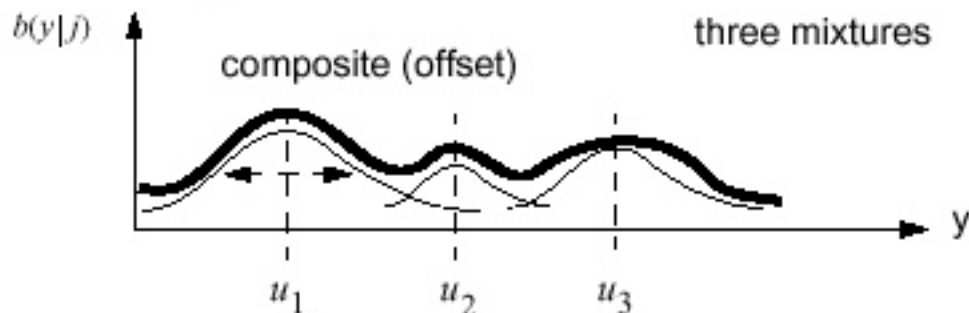
$$f_{y|x}(y|i) = \frac{1}{\sqrt{2\pi|\Sigma_i|}} \exp\left\{-\frac{1}{2}(y - \mu_i)^T \Sigma_i^{-1} (y - \mu_i)\right\}$$

Note that this amounts to assigning a mean and covariance matrix to each state — a significant increase in complexity. However, shortcuts such as variance-weighting can help reduce complexity.

Also, note that the log of the output probability at each state becomes precisely the Mahalanobis distance (principal components) we studied at the beginning of the course.

MIXTURE DENSITIES: A POWERFUL MODELING TOOL

Of course, the output distribution need not be Gaussian, or can be multimodal to reflect the fact that several contexts are being encoded into a single state (male/female, allophonic variations of a phoneme, etc.). Much like a VQ approach can model any discrete distribution, we can use a weighted linear combination of Gaussians, or a mixture distribution, to achieve a more complex statistical model.



Mathematically, this is expressed as:

$$f_{\underline{y}|\underline{x}}(\underline{y}|i) = \sum_{m=1}^M c_{im} \mathfrak{N}(\underline{y}; \underline{\mu}_{im}, \underline{\Sigma}_{im})$$

In order for this to be a valid pdf, the mixture coefficients must be nonnegative and satisfy the constraint:

$$\sum_{m=1}^M c_{im} = 1, \quad 1 \leq i \leq S$$

Note that mixture distributions add significant complexity to the system: m means and covariances at each state.

PROBABILITY CALCULATIONS

Consider a Gaussian mixture density function

$$b_j(\mathbf{y}) = \sum_{k=1}^M c_{jk} b_{jk}(\mathbf{y}) = \sum_{k=1}^M c_{jk} \mathfrak{N}(\mathbf{y}, \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk})$$

Recall that the mixture weights are constrained to sum to 1: $\sum_{k=1}^M c_{jk} = 1$.

We can write the joint probability of the data and states given the model as:

$$p(\mathbf{Y}, \mathbf{S} | \lambda) = \prod_{t=1}^T a_{s_{t-1}, s_t} b_{s_t}(\mathbf{y}_t) = \sum_{k_1=1}^M \sum_{k_2=1}^M \cdots \sum_{k_T=1}^M \left\{ \prod_{t=1}^T a_{s_{t-1}, s_t} b_{s_t k_t}(\mathbf{y}_{s_t}) c_{s_t k_t} \right\}$$

This can be considered the sum of the product of all densities with all possible state sequences, \mathbf{S} , and all possible mixture components, \mathbf{K} :

$$p(\mathbf{Y}, \mathbf{S}, \mathbf{K} | \lambda) = a_{s_{t-1}, s_t} b_{s_t k_t}(\mathbf{y}_{s_t}) c_{s_t k_t}$$

and

$$p(\mathbf{X} | \lambda) = \sum_{\mathbf{S}} \sum_{\mathbf{K} \in \Omega^T} p(\mathbf{Y}, \mathbf{S}, \mathbf{K} | \lambda)$$

APPLICATION OF EM: OPTIMIZATION

We can write the auxiliary function:

$$Q(\lambda|\bar{\lambda}) = \sum_S \sum_{K \in \Omega^T} \frac{p(Y, S, K|\lambda)}{p(Y, \lambda)} \log(p(Y, S, K|\lambda))$$

which has the expected decomposition:

$$\log(p(Y, S, K|\lambda)) = \sum_{t=1}^T \log \hat{a}_{s_{t-1}s_t} + \sum_{t=1}^T \log \hat{b}_{s_t k_t}(\mathbf{y}_t) + \sum_{t=1}^T \log \hat{c}_{s_t k_t}$$

The auxiliary function has three components:

$$Q(\lambda|\bar{\lambda}) = \sum_{i=1}^N Q_{a_i}(\lambda, \hat{a}_i) + \sum_{j=1}^N \sum_{k=1}^M Q_{b_{jk}}(\lambda, \hat{b}_{jk}) + \sum_{j=1}^N \sum_{k=1}^M Q_{c_{jk}}(\lambda, \hat{c}_{jk})$$

The first resembles the term we had for discrete distributions. The remaining two terms are new:

$$Q_{b_{jk}}(\lambda, \hat{b}_{jk}) = \sum_{t=1}^T p(s_t = j, k_t = k | Y, \lambda) \log \hat{b}_{jk}(\mathbf{y}_t)$$

and

$$Q_{c_{jk}}(\lambda, \hat{c}_{jk}) = \sum_{t=1}^T p(s_t = j, k_t = k | Y, \lambda) \log \hat{c}_{jk}$$

REESTIMATION EQUATIONS

Maximization of $Q_{b_{jk}}(\lambda, \hat{b}_{jk})$ requires differentiation with respect to the parameters of the Gaussian: $[\mu_{jk}, \Sigma_{jk}]$. This results in the following equations:

$$\begin{aligned}\hat{\mu}_{jk} &= \frac{\frac{1}{p(\mathbf{Y}|\lambda)} \sum_{t=1}^T p(\mathbf{Y}, s_t = j, k_t = k | \lambda) \mathbf{y}_t}{\frac{1}{p(\mathbf{Y}|\lambda)} \sum_{t=1}^T p(\mathbf{Y}, s_t = j, k_t = k | \lambda)} = \frac{\sum_{t=1}^T \zeta_t(j, k) \mathbf{y}_t}{\sum_{t=1}^T \zeta_t(j, k)} \\ \hat{\Sigma}_{jk} &= \frac{\frac{1}{p(\mathbf{Y}|\lambda)} \sum_{t=1}^T p(\mathbf{Y}, s_t = j, k_t = k | \lambda) (\mathbf{y}_t - \hat{\mu}_{jk})(\mathbf{y}_t - \hat{\mu}_{jk})^\dagger}{\frac{1}{p(\mathbf{Y}|\lambda)} \sum_{t=1}^T p(\mathbf{Y}, s_t = j, k_t = k | \lambda)} \\ &= \frac{\sum_{t=1}^T \zeta_t(j, k) (\mathbf{y}_t - \hat{\mu}_{jk})(\mathbf{y}_t - \hat{\mu}_{jk})^\dagger}{\sum_{t=1}^T \zeta_t(j, k)}\end{aligned}$$

where $\zeta_t(j, k)$ is computed as:

$$\zeta_t(j, k) = \frac{p(\mathbf{Y}, s_t = j, k_t = k | \lambda)}{p(\mathbf{Y}|\lambda)} = \frac{\sum_{i=1}^N \alpha_{t-1}(i) a(j|i) c_{jk} b_{jk}(\mathbf{y}_t) \beta_t(j)}{\sum_{i=1}^N \alpha_T(i)}$$

Similarly, we can reestimate the mixture coefficients using a similar equation:

T

$$\hat{c}_{jk} = \frac{\sum_{t=1}^T \zeta_t(j, k)}{T \sum_{k=1}^M \zeta_t(j, k)}$$

[Return to Main](#)[Objectives](#)**Topology:**[Transition Matrix](#)[DTW Analogy](#)[Alternatives](#)**Duration Modeling:**[State Duration Probabilities](#)[Number of States](#)**Training Schedules:**[Simple Schedule](#)[More Complex Schedules](#)[Complex Models](#)**On-Line Resources:**[Training Workshop](#)[Using HMMs](#)[The HTK Book](#)

LECTURE 26: PRACTICAL ISSUES

- Objectives:
 - Discuss common model topologies
 - Provide model design guidelines
 - Introduce typical training schedules

This lecture combines material from this paper:

J. Picone, "Continuous Speech Recognition Using Hidden Markov Models", *IEEE ASSP Magazine*, vol. 7, no. 3, pp. 26-41, July 1990.

and information found in this workshop:

Speech Recognition System Training Workshop, Institute for Signal and Information Processing, Mississippi State University, Mississippi State, Mississippi, 39762, USA, January 2002.

LECTURE 26: PRACTICAL ISSUES

- Objectives:
 - Discuss common model topologies
 - Provide model design guidelines
 - Introduce typical training schedules

This lecture combines material from this paper:

J. Picone, "Continuous Speech Recognition Using Hidden Markov Models", *IEEE ASSP Magazine*, vol. 7, no. 3, pp. 26-41, July 1990.

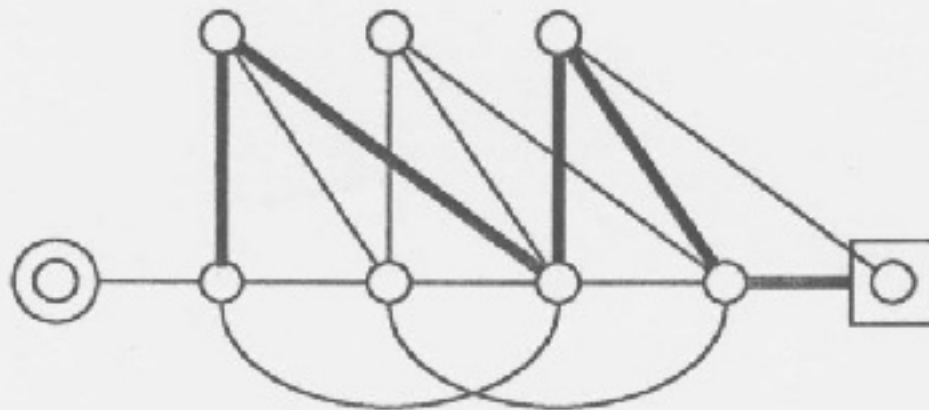
and information found in this workshop:

[Speech Recognition System Training Workshop](http://www.isip.msstate.edu/~gao/net/2002_spring/lecture_26/lecture_26_00.html), Institute for Signal and Information Processing, Mississippi State

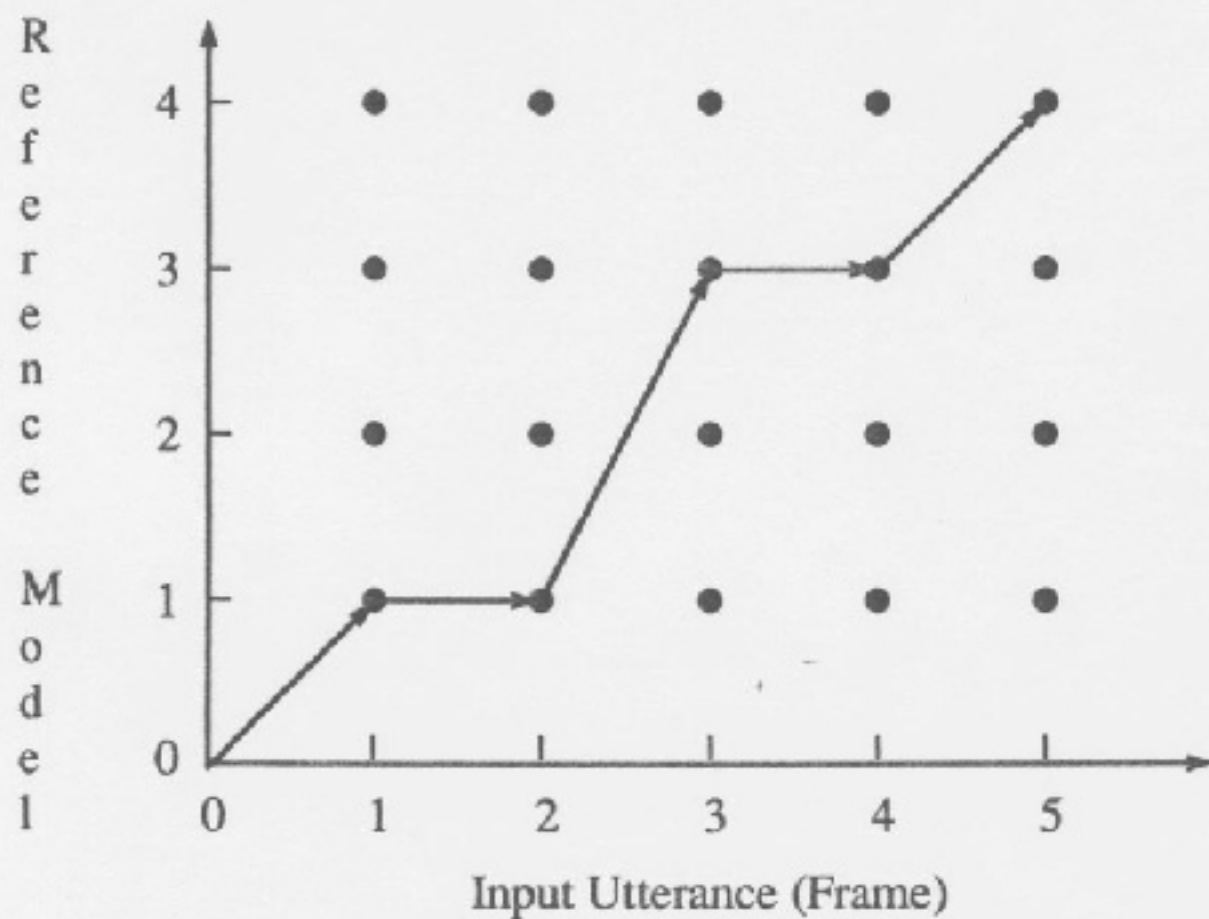
University, Mississippi State, Mississippi,
39762, USA, January 2002.

THE DTW ANALOGY

HMM Recognition Using The Viterbi Algorithm



Dynamic Time Warping Using The Viterbi Algorithm



- Note the similarity to DTW with slope constraints

ALTERNATIVE MODEL TOPOLOGIES



Figure 8(a). A simple progressive HMM topology. In general, the duration probability density function at a state has an exponential behavior.

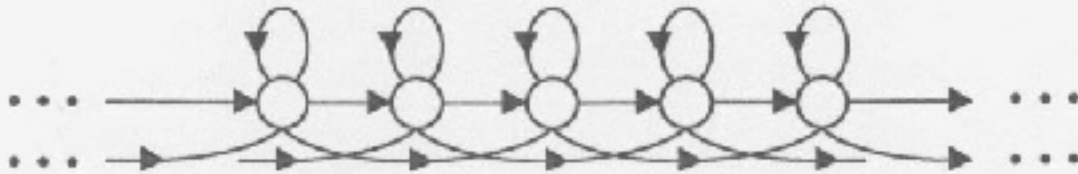


Figure 8(b). The Bakis topology (a progressive model with skip states).

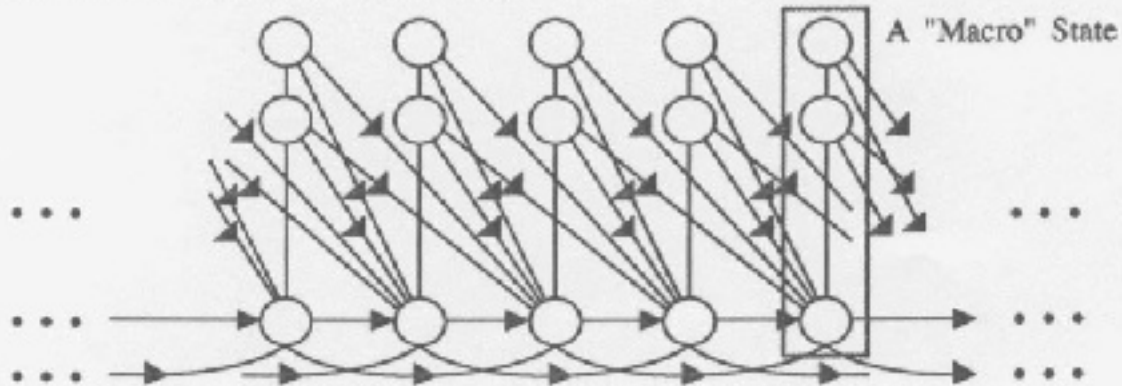


Figure 8(c). A finite duration topology. This topology is most analogous to DTW.

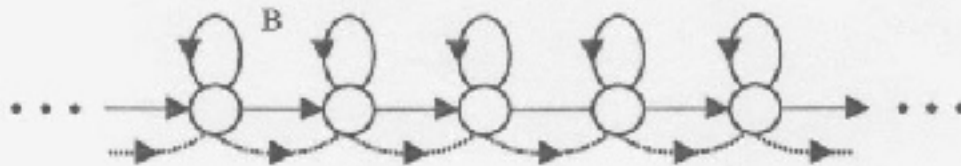


Figure 8(d). A fenonic baseform topology. The dashed line indicates a transition that produces no output.

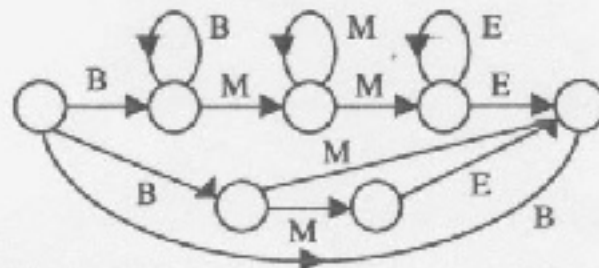


Figure 8(e). A modified fenonic baseform with tied transitions. Transitions in the same group share output probabilities.

- Note the similarity to DTW with slope constraints

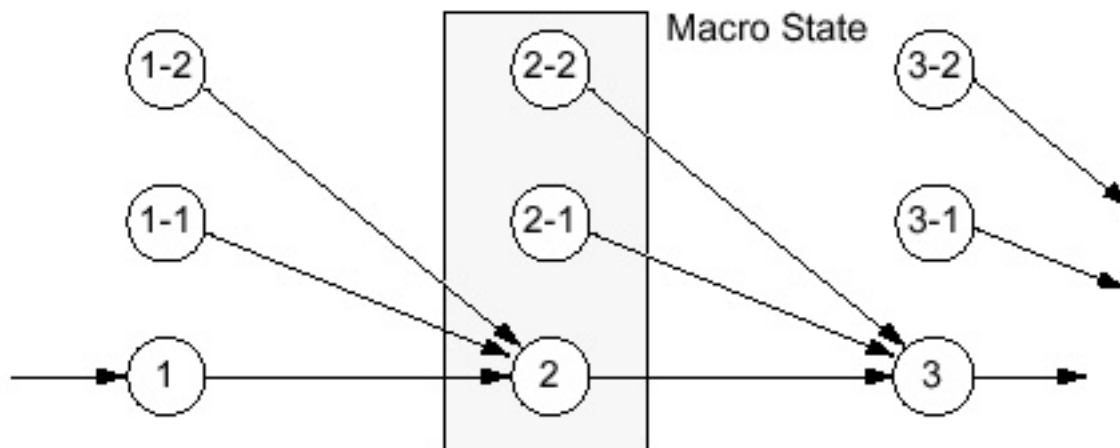
STATE DURATION PROBABILITIES

Recall that the probability of staying in a state was given by an exponentially-decaying distribution:

$$P(\bar{O} | \text{Model}, q_1 = i) = P(\bar{O}, q_1 = i | \text{Model}) / P(q_1 = i) = a_{ii}^{d-1} (1 - a_{ii})$$

This model is not necessarily appropriate for speech. There are three approaches in use today:

- Finite-State Models (encoded in acoustic model topology)



(Note that this model doesn't have skip states; with skip states, it becomes much more complex.)

- Discrete State Duration Models (D parameters per state)

$$P(d_i = d) = \tau_d \quad 1 \leq d \leq D$$

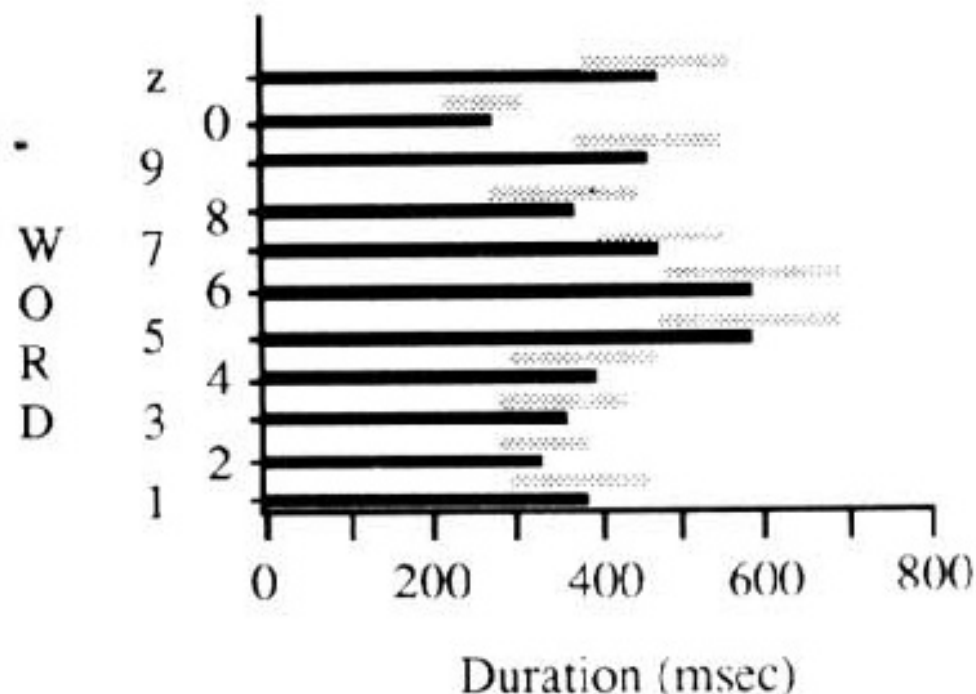
- Parametric State Duration Models (one to two parameters)

$$f(d_i) = \frac{1}{\sqrt{2\sigma_i^2}} \exp\left\{-\frac{\sqrt{2}|d|}{\sigma_i}\right\}$$

Reestimation equations exist for all three cases. Duration models are often important for larger models, such as words, where duration variations can be significant, but not as important for smaller units, such as context-dependent phones, where duration variations are much better understood and predicted.

DURATION CONSIDERATIONS AND CLUSTERING

- For word model-based systems, we often will consider the duration of the word when assigning the initial number of states in the model:



- Clustering approaches can be used to learn pronunciation variants:

Duration (msec)

**First Cluster:**

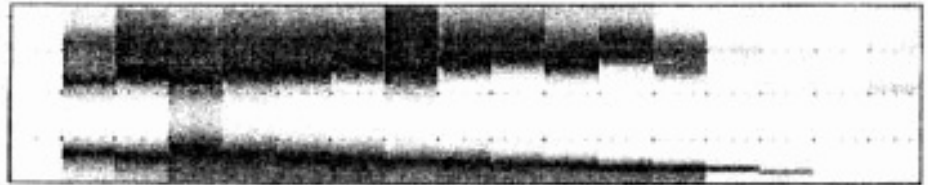
34% of Tokens

95% Female

5% Male

Slight "T" Release

340 msec

**Second Cluster:**

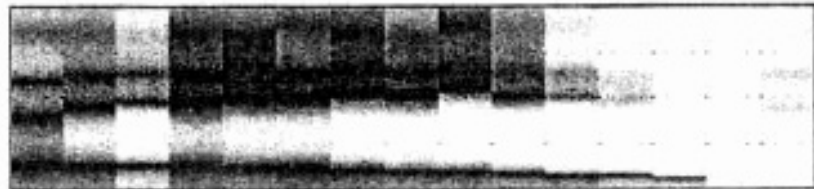
30% of Tokens

97% Male

3% Female

Slight "T" Release

300 msec

**Third Cluster:**

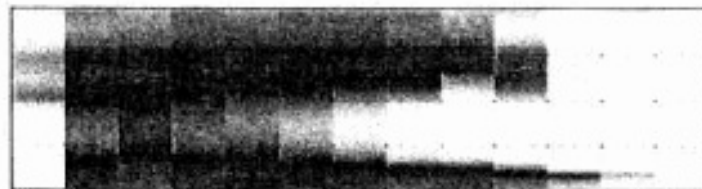
20% of Tokens

0% Male

100% Female

Unreleased "T"

260 msec

**Fourth Cluster:**

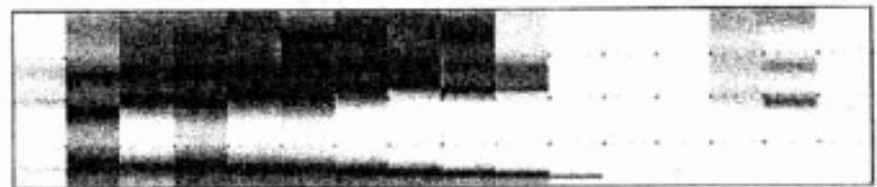
16% of Tokens

87% Male

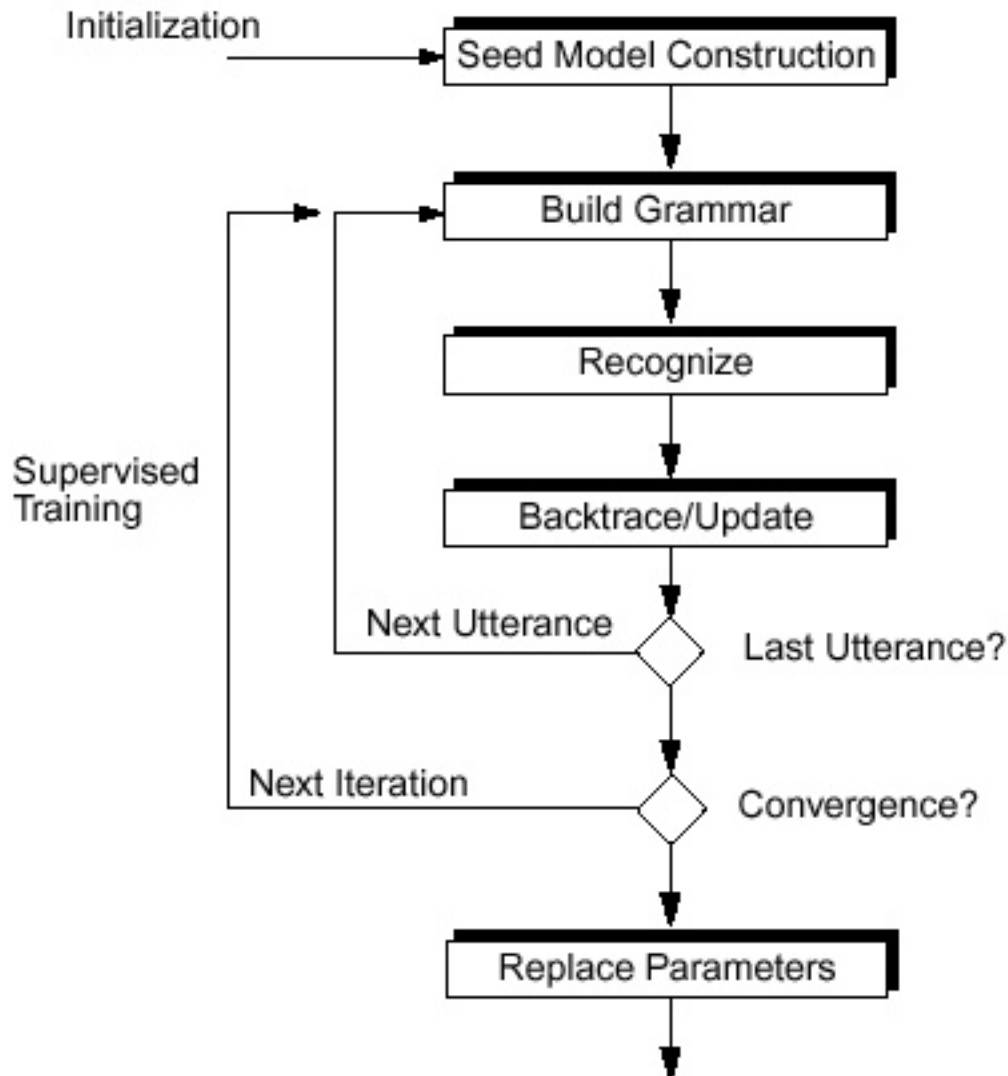
13% Female

Strong "T" Release

320 msec



TYPICAL TRAINING SCHEDULES



Note that *a priori* segmentation of the utterance is not required, and that the recognizer is forced to recognize the utterance during training (via the build grammar operation). This forces the recognizer to learn contextual variations, provided the seed model construction is done “properly.”

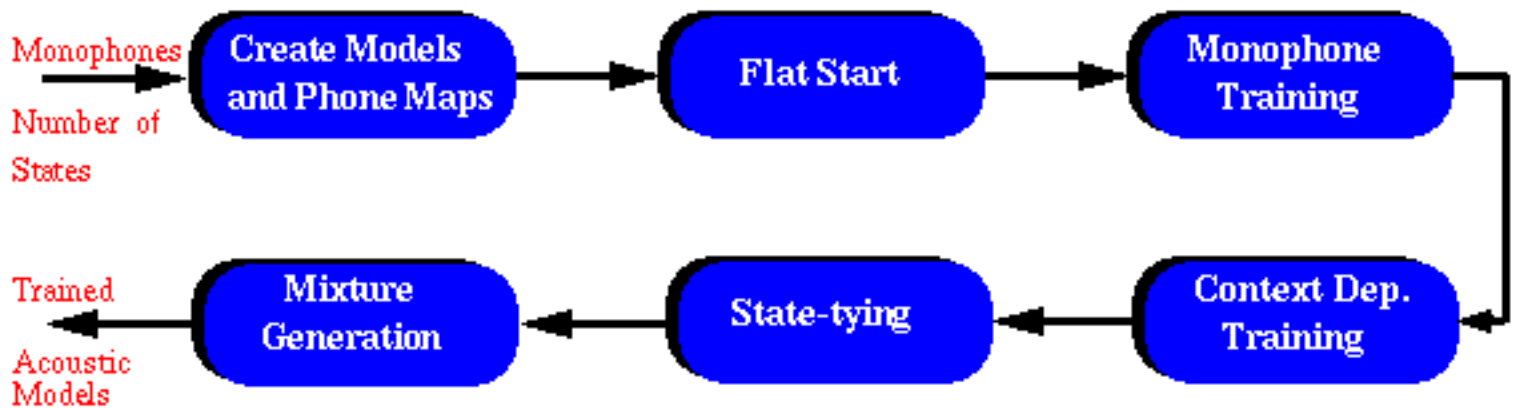
What about speaker independence?

Speaker dependence?

Speaker adaptation?

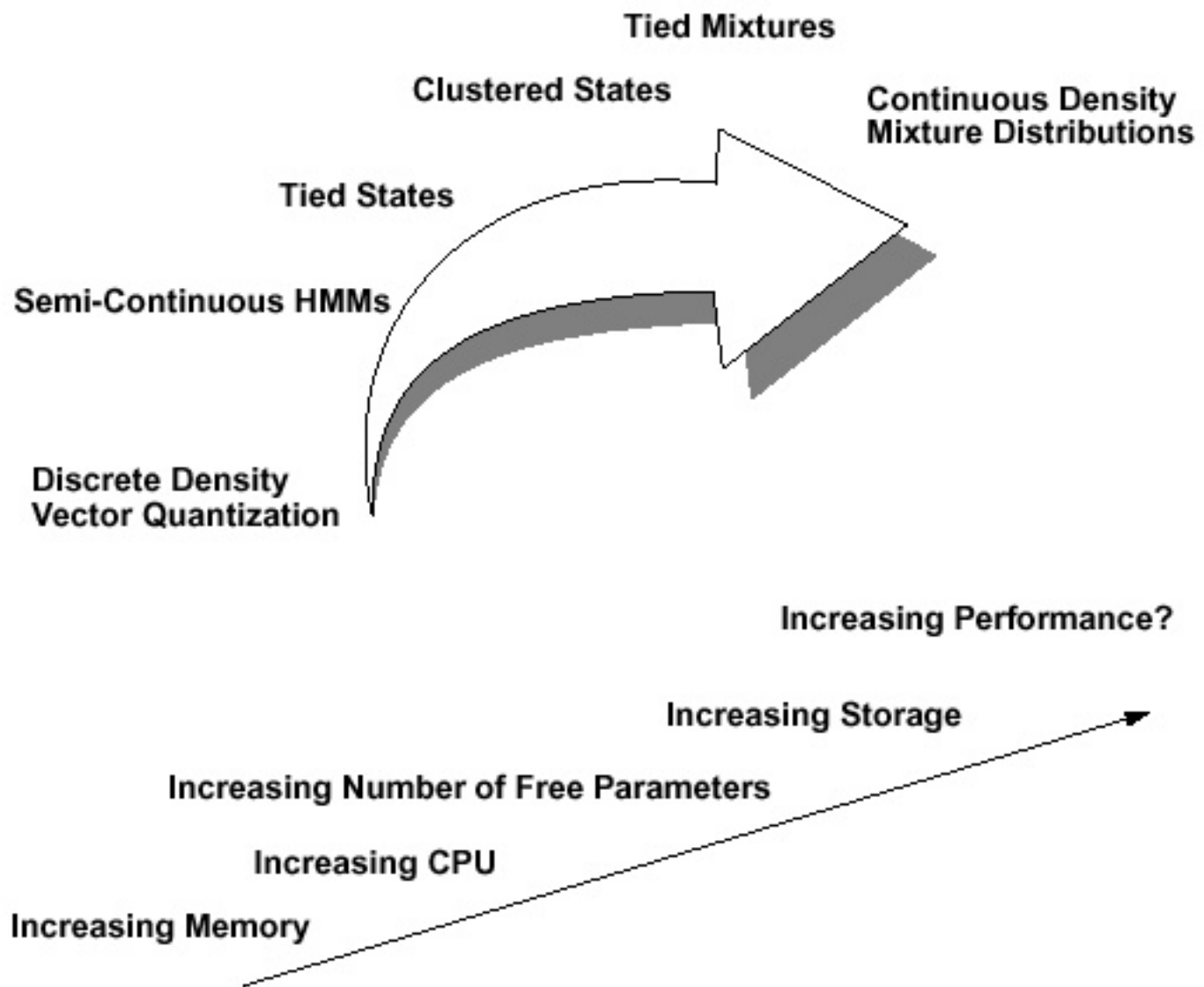
Channel adaptation?

MORE EXTENSIVE TRAINING SCHEDULES



- Phone-based HMM systems require a more extensive training process.
- Mixture generation is usually done last, and is performed using a cluster-splitting approach.
- Retraining after mixture generation is important.

MODEL COMPLEXITY



- Numerous techniques to robustly estimate model parameters; among the most popular is deleted interpolation:

$$A = \epsilon_t A_t + (1 - \epsilon_t) A_u$$

$$B = \epsilon_t B_t + (1 - \epsilon_t) B_u$$

[Return to Main](#)[Objectives](#)**Motivation:**[Classification](#)[Parameter Count](#)**Basic Concepts:**[Terminology](#)[Operation](#)[Splitting](#)[Growing](#)[Pruning](#)[CART](#)**Applications:**[Acoustic Modeling](#)[Pronunciation Modeling](#)**On-Line Resources:**[AAAI: Decision Trees](#)[Zhao: Tutorial](#)[Ngan: Applications](#)[Le: Applications](#)[Software](#)

LECTURE 27: DECISION TREES

- Objectives:
 - Why do we need a smart algorithm to reduce the number of parameters? On what type of information should this smart algorithm operate?
 - Basic concepts of *classification and regression trees* (CART)
 - How do we apply them to acoustic modeling? What are the benefits?

This lecture combines material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

and these MS project presentations:

- J. Ngan, "Information Theory Based Decision Trees for Data Classification," *Master of Science Special Project Presentation*, December 10, 1998 (available at [Ngan: MS project presentation](#))
- A. Le, "Bayesian Decision Tree for Classification of Nonlinear

Signal Processing Problems,"
*Master of Science Special Project
Presentation*, November 12, 1998
(available at [Le: MS project
presentation](#))

LECTURE 27: DECISION TREES

- Objectives:
 - Why do we need a smart algorithm to reduce the number of parameters? On what type of information should this smart algorithm operate?
 - Basic concepts of *classification and regression trees* (CART)
 - How do we apply them to acoustic modeling? What are the benefits?

This lecture combines material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory,*

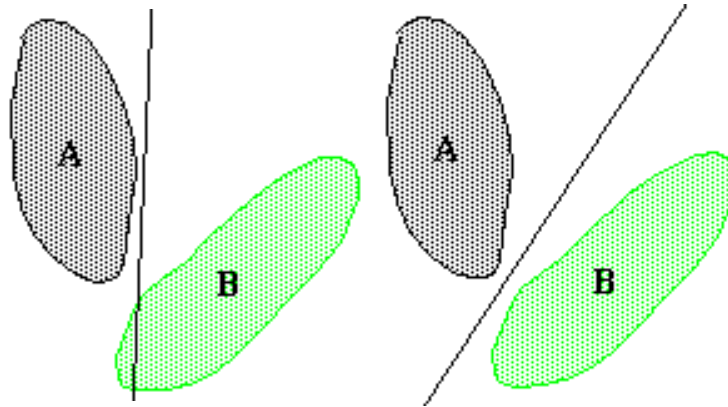
Algorithm, and System Development, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

and these MS project presentations:

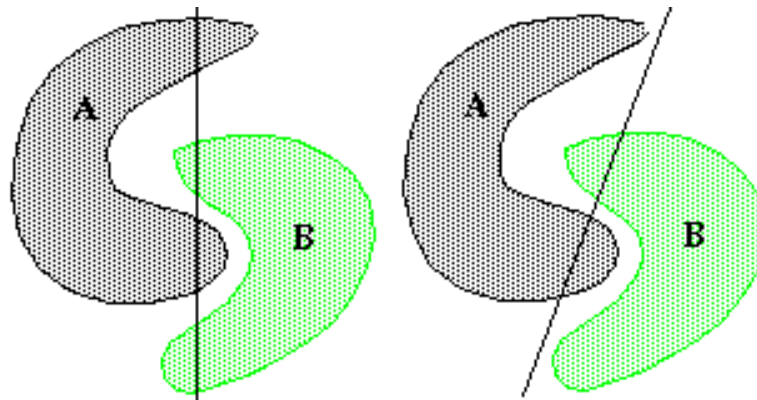
- J. Ngan, "Information Theory Based Decision Trees for Data Classification," *Master of Science Special Project Presentation*, December 10, 1998 (available at [Ngan: MS project presentation](#))
- A. Le, "Bayesian Decision Tree for Classification of Nonlinear Signal Processing Problems," *Master of Science Special Project Presentation*, November 12, 1998 (available at [Le: MS project presentation](#))

DECISION TREES: A POWERFUL DATA-DRIVEN CLASSIFICATION ALGORITHM

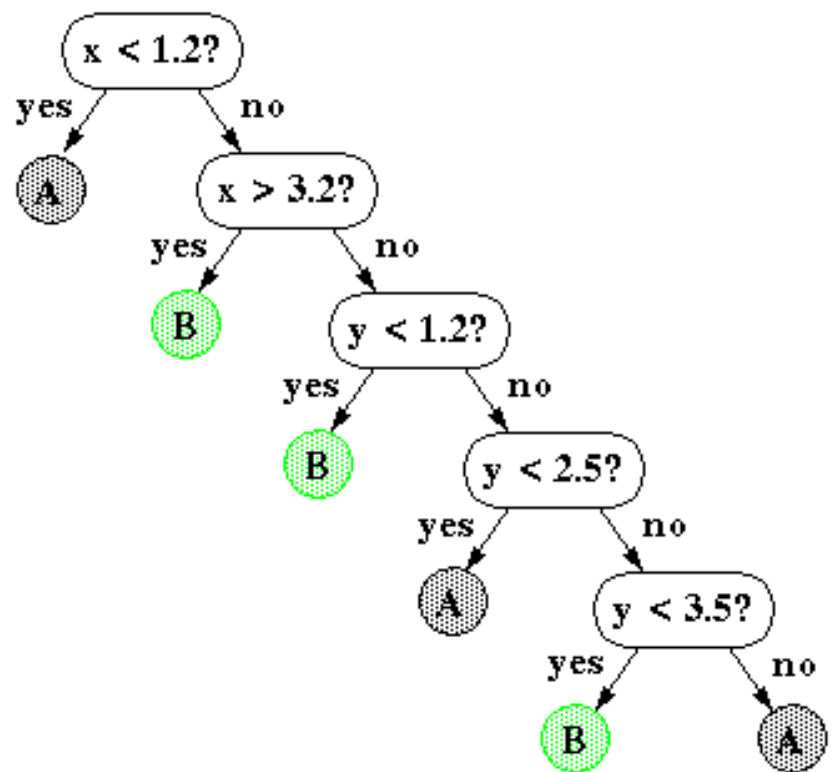
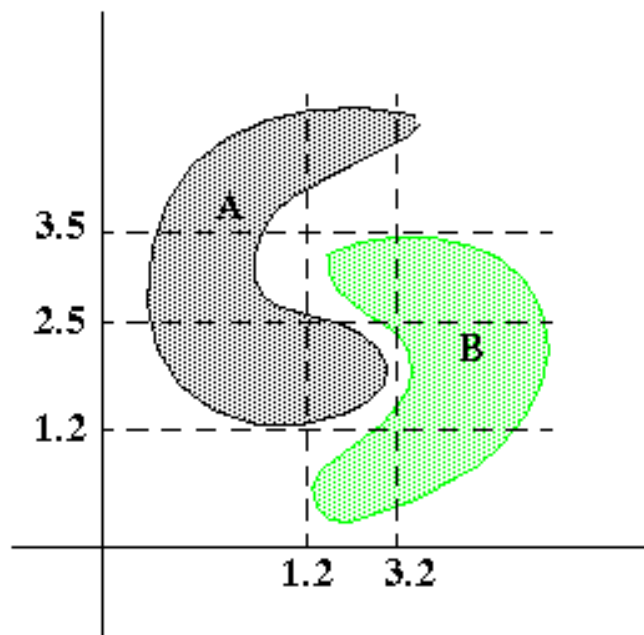
- When PCA fails:



- and LDA fails:

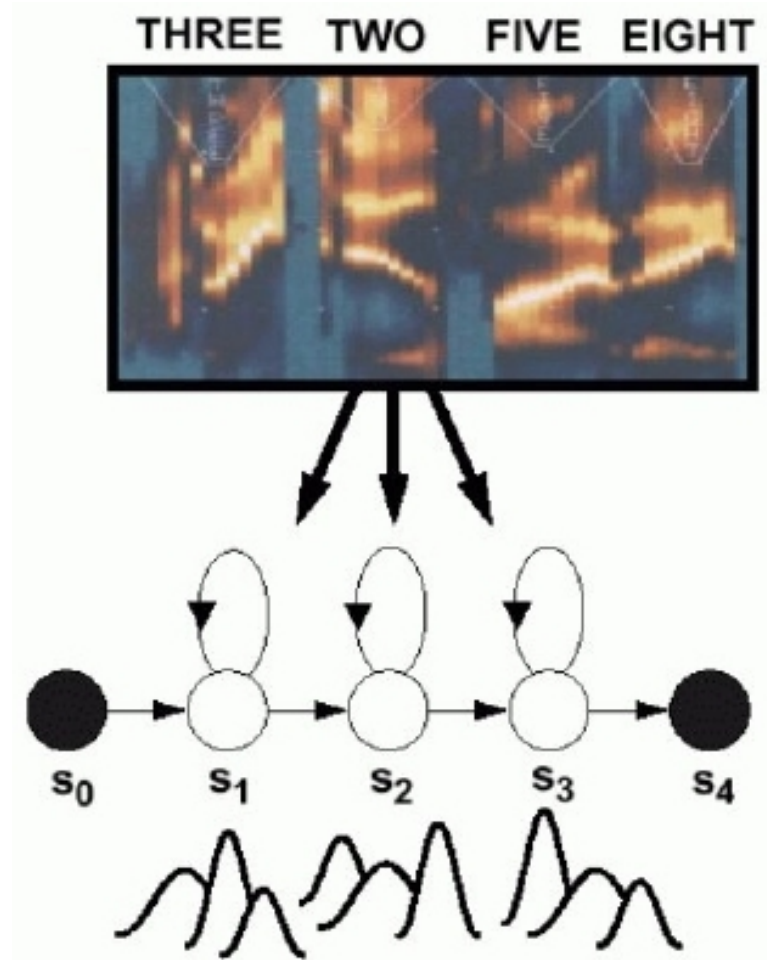


- we can imagine a more powerful data driven approach:



CONTROLLING PARAMETER COUNT IS AN IMPORTANT REALITY

- Acoustic models encode the temporal evolution of the features (spectrum).
- Gaussian mixture distributions are used to account for



variations in
speaker,
accent, and
pronunciation.

- Phonetic
model
topologies are
simple
left-to-right
structures.

- Sharing
model

parameters is a
common
strategy to
reduce
complexity
and avoid
undertraining:

(39 features +
39 covariance
values +
1 mixture
weight) x
16 Gaussian
per state x

3 states/phone

X

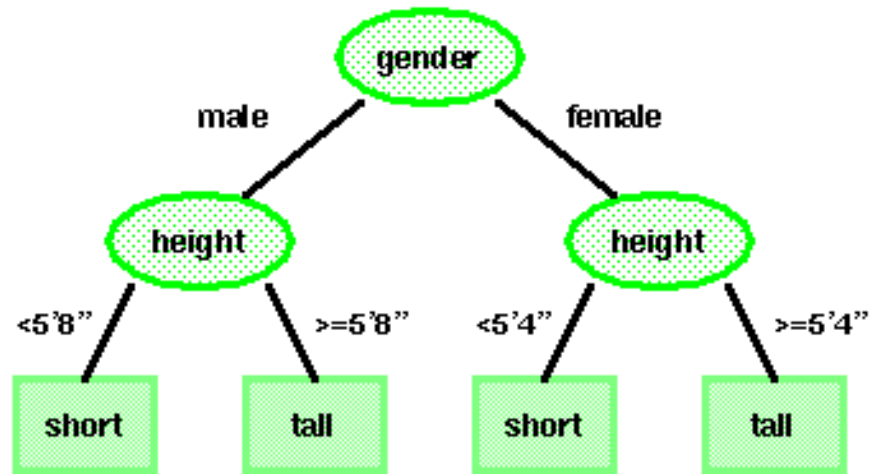
80,000 CD

phones =

$\sim 300 \times 10^6$
parameters!

BASIC TERMINOLOGY

- A decision tree consists of nodes and leaves, with each leaf denoting a class.



- Classes

(tall or short) are the outputs of the tree.

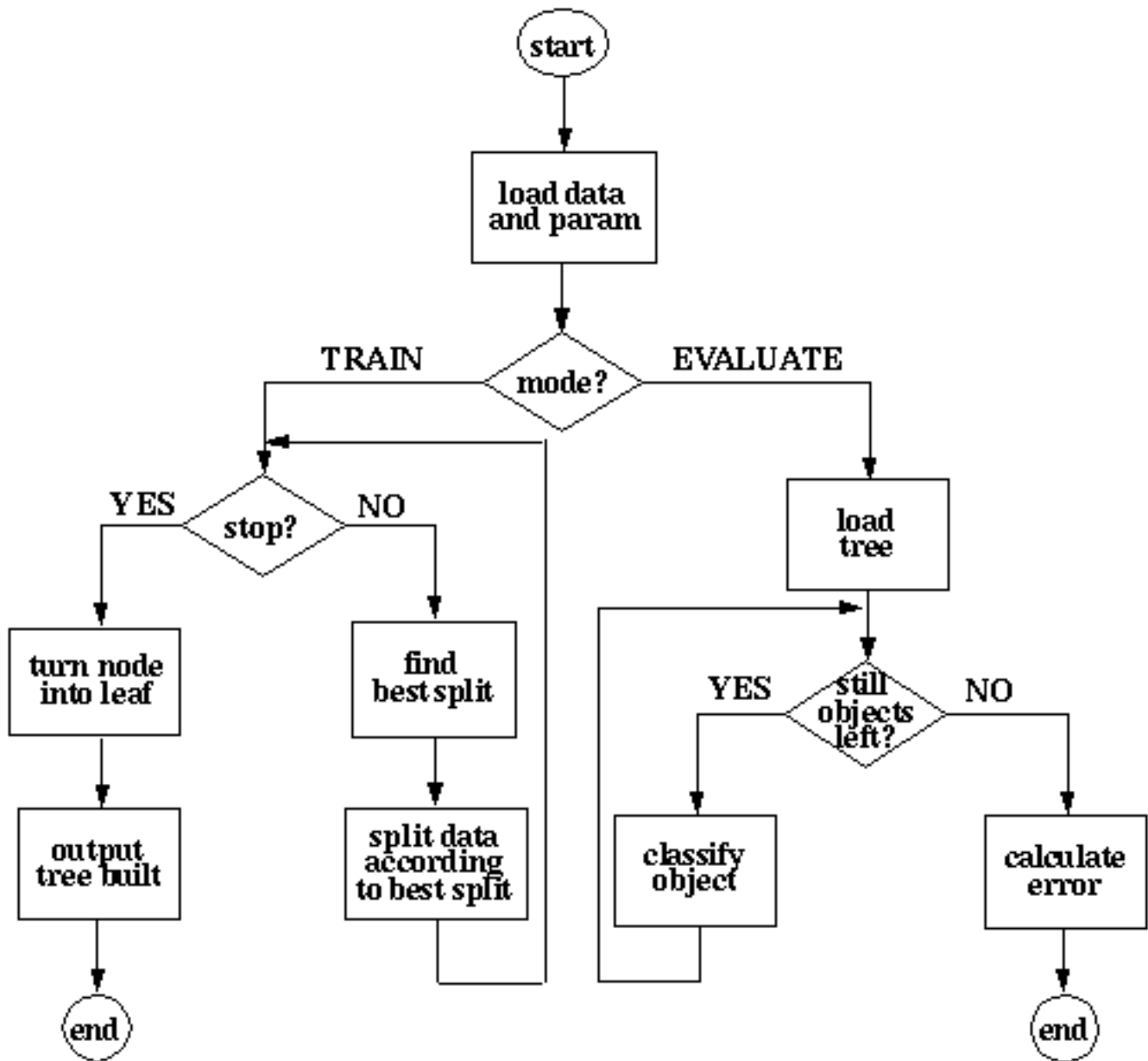
- Attributes (gender and height) are a set of features that describe

the data.

- The input data consists of values of the different attributes. Using these attribute values, the decision

tree
generates
a class as
the output
for each
input data.

DATA-DRIVEN OPERATION



There are four important operations in constructing a decision tree:

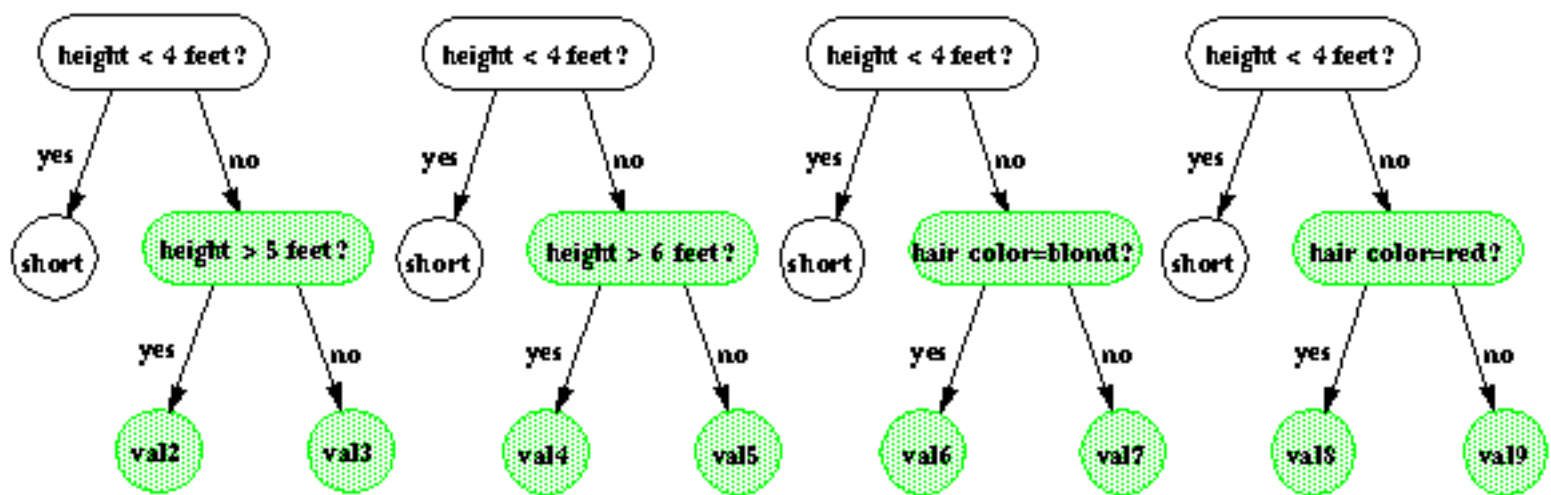
- Question selection: choosing a set of questions to categorize your data (some algorithms can derive questions automatically).
- Splitting: partitioning data assigned to a node into N groups ($N=2$ for binary trees).
- Growing: expanding the tree to better represent the training data.
- Pruning: removing nodes to improve generalization.

In speech recognition, we operate on continuous-valued feature vectors, and use likelihood computations derived directly from HMM training. This is a major reason why decision trees are so popular in speech

recognition systems - the implementation is very elegant.

SPLITTING CRITERIA

To split data at a node, we need to find the question that results in the greatest entropy reduction (removes uncertainty in the data):



In speech recognition, we can show this amounts to maximizing the increase in likelihood:

$$dL = L(\text{parent}) - L(\text{left child}) - L(\text{right child})$$

These likelihoods can be computed from the state occupancies computed during training (see

[decision tree-based state tying](#) for a detailed derivation and the important references).

GROWING THE TREE

We typically grow the tree by successively splitting each node until nodes can no longer be split. Though this is locally optimal, it is not globally optimal. Nevertheless this produces useful trees with minimum computational complexity.

We can continue splitting nodes until:

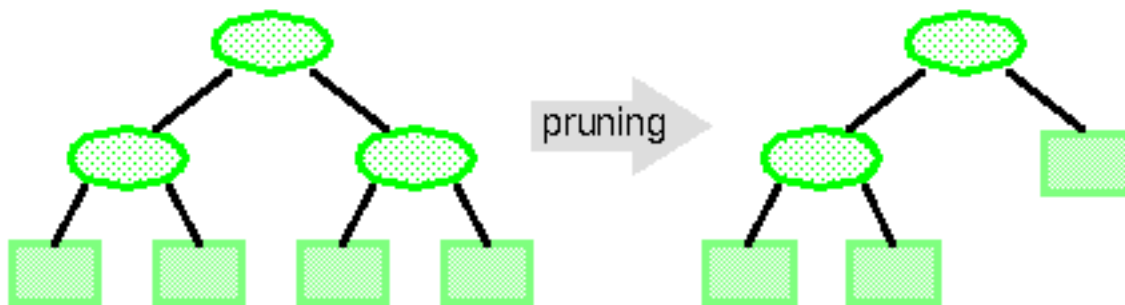
- No more splits are possible (all samples at a node belong in the same class).
- The greatest likelihood increase (entropy reduction) falls below our pre-set threshold.
- The number of data samples falling in a leaf node falls below some threshold.

Nodes which can no longer be split are declared

terminal nodes. When all active nodes are terminal nodes, tree growing terminates.

PRUNING A TREE IMPROVES GENERALIZATION

The most fundamental problem with decision trees is that they "overfit" the data and hence do not provide good generalization. A solution to this problem is to prune the tree:



Cost-complexity pruning is a popular technique for pruning. Cost-complexity can be defined as:

$$R_{\alpha}(t) = R(T) + \alpha|\tilde{T}|$$

where $|\tilde{T}|$ represents the number of terminal

nodes in the subtree.

Each node in the tree can be classified in terms of its impact on the cost-complexity if it were pruned. Nodes are successively pruned until certain thresholds (heuristics) are satisfied.

By pruning the nodes that are far too specific to the training set, it is hoped the tree will have better generalization. In practice, we use techniques such as cross-validation and held-out training data to better calibrate the generalization properties.

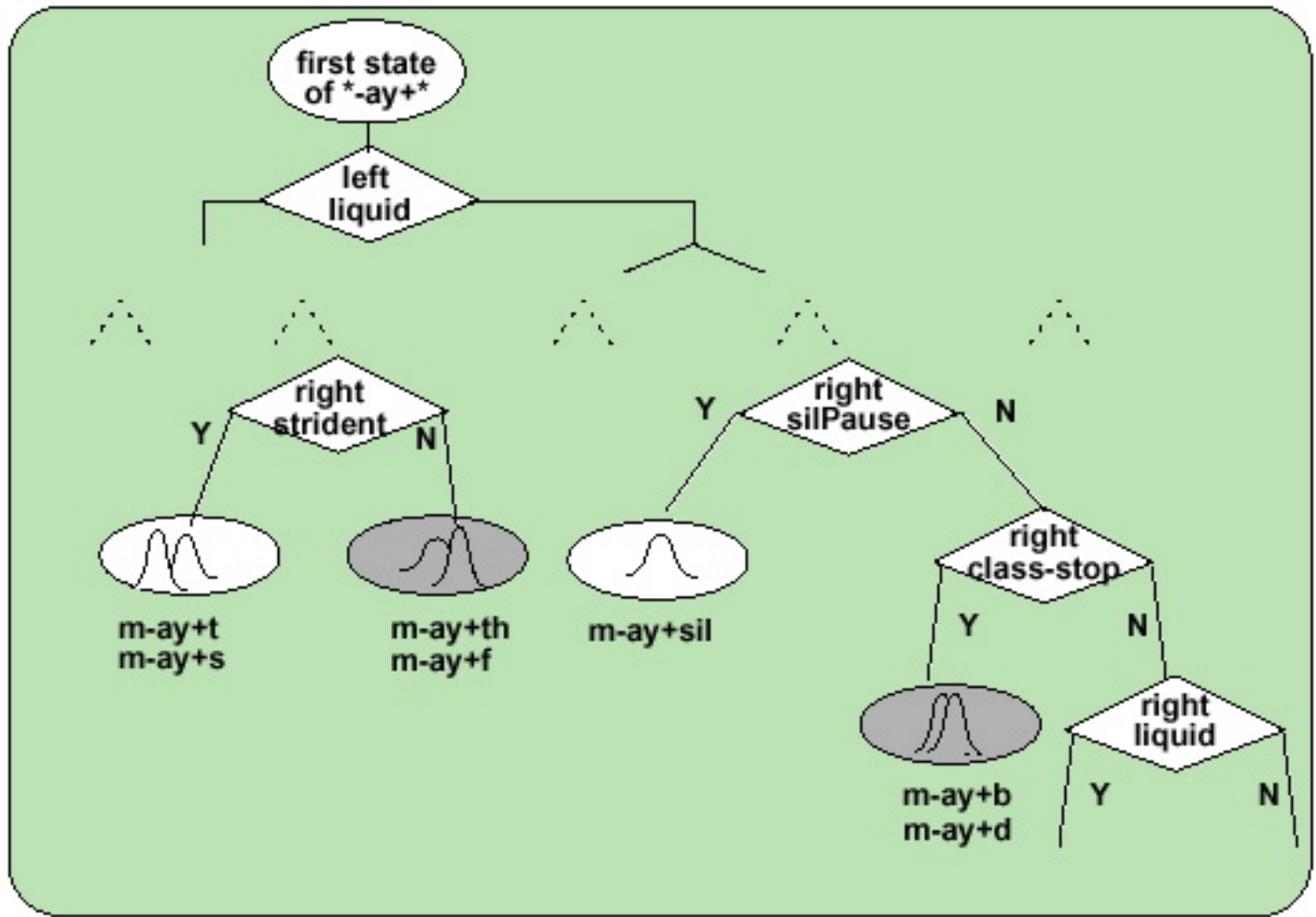
THE CART ALGORITHM

The classification and regression tree (CART) algorithm can be summarized as follows:

1. Create a set of questions that consists of all possible questions about the measured variables (phonetic context).
2. Select a splitting criterion (likelihood).
3. Initialization: create a tree with one node containing all the training data.
4. Splitting: find the best question for splitting each terminal node. Split the one terminal node that results in the greatest increase in the likelihood.

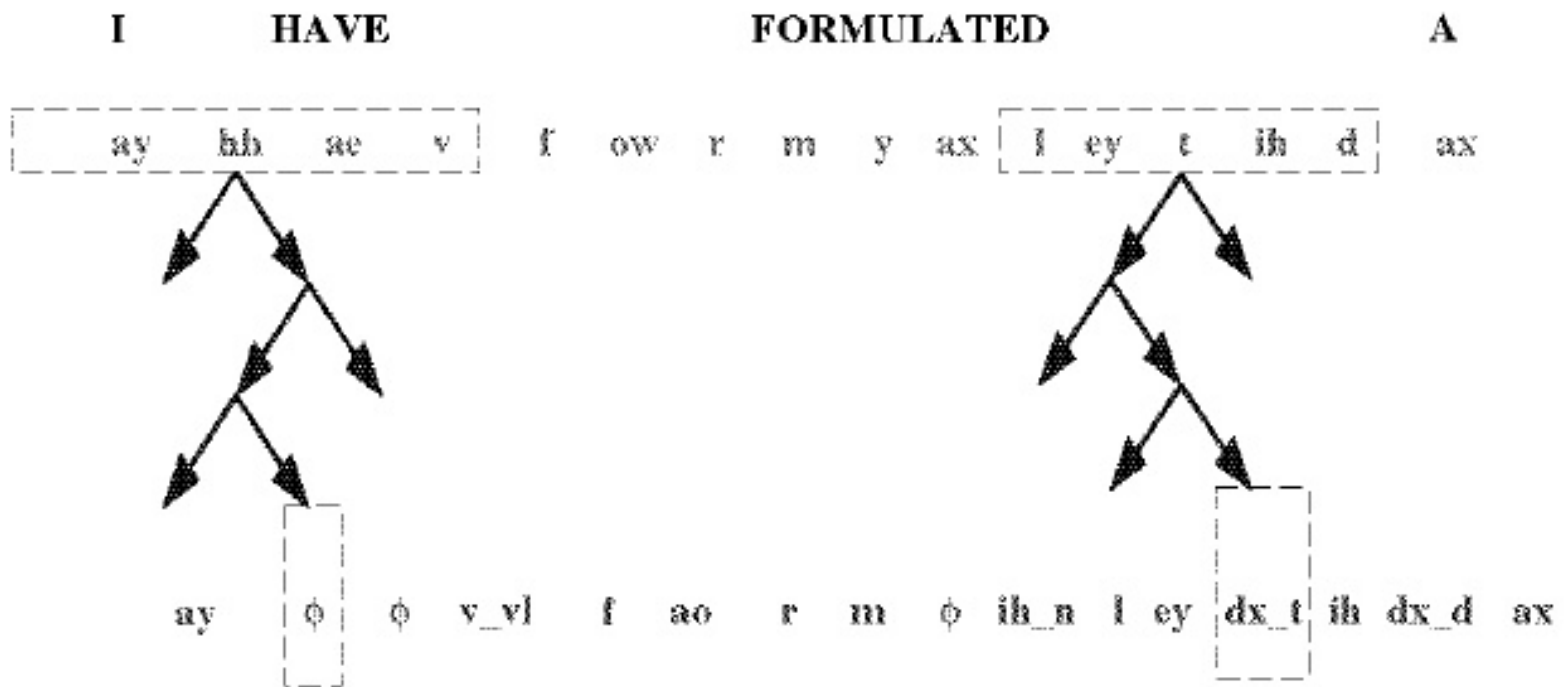
5. Stopping: if each leaf node contains data samples from the same class, or some pre-set threshold is not satisfied, stop. Otherwise, continue splitting.
6. Pruning: use an independent test set or cross-validation to prune the tree.

APPLICATION: ACOUSTIC MODELING



APPLICATION: PRONUNCIATION MODELING

Goal: Condition mappings from baseforms to pronunciations using as much linguistic information as possible (e.g., syllable boundaries). Train using hand-labeled data.



[Return to Main](#)[Objectives](#)**Mixture Generation:**[EM Estimation](#)[Clustering](#)[Variance-Splitting](#)**Temporal Modeling:**[Independence](#)[Duration](#)[First-Order](#)**Review:**[Syllabus](#)**On-Line Resources:**[Clustering](#)[Conditional Independence](#)[Ten Years of HMMs](#)

LECTURE 28: PRACTICAL ISSUES

- Objectives:
 - Mixture splitting
 - Clustering
 - Conditional independence
 - Duration modeling
 - Higher order processes

This lecture combines material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language*

*Processing - A Guide to Theory,
Algorithm, and System
Development*, Prentice Hall,
Upper Saddle River, New Jersey,
USA, ISBN: 0-13-022616-5,
2001.

and from this source:

S. Young, *et al*, *The HTK Book*
(v3.0), Cambridge University
Engineering Department,
September 2000.

LECTURE 28: PRACTICAL ISSUES

- Objectives:
 - Mixture splitting
 - Clustering
 - Conditional independence
 - Duration modeling
 - Higher order processes

This lecture combines material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory,*

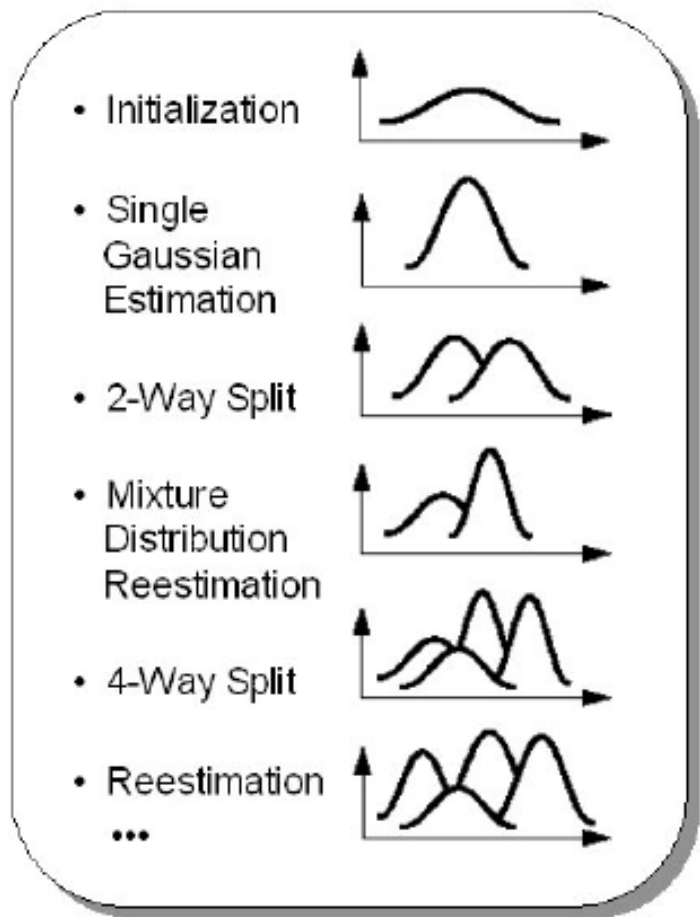
Algorithm, and System Development, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

and from this source:

S.Young, *et al*, *The HTK Book (v3.0)*, Cambridge University Engineering Department, September 2000.

EM ESTIMATION OF MIXTURES

- Closed-loop data-driven modeling supervised only from a word-level transcription
- The expectation/maximization (EM) algorithm is used to improve our parameter estimates.



- Computationally efficient training algorithms (Forward-Backward) have been crucial.
- Batch mode parameter updates are typically preferred.
- Decision trees are used to optimize parameter-sharing, system

complexity, and the use of
additional linguistic
knowledge.

K-MEANS CLUSTERING

Algorithm Overview:

- **Initialization:** Choose K centroids
- **Recursion:**
 - Assign all vectors to their nearest neighbor.
 - Recompute the centroids as the average of all vectors assigned to the same centroid.
- **Termination:** Check overall distortion.

For a typical implementation of K-MEANS, see our [pattern recognition applet](http://www.isip.msstate.edu/~gao/net/2002_spring/lecture_28/lecture_28_02.html).

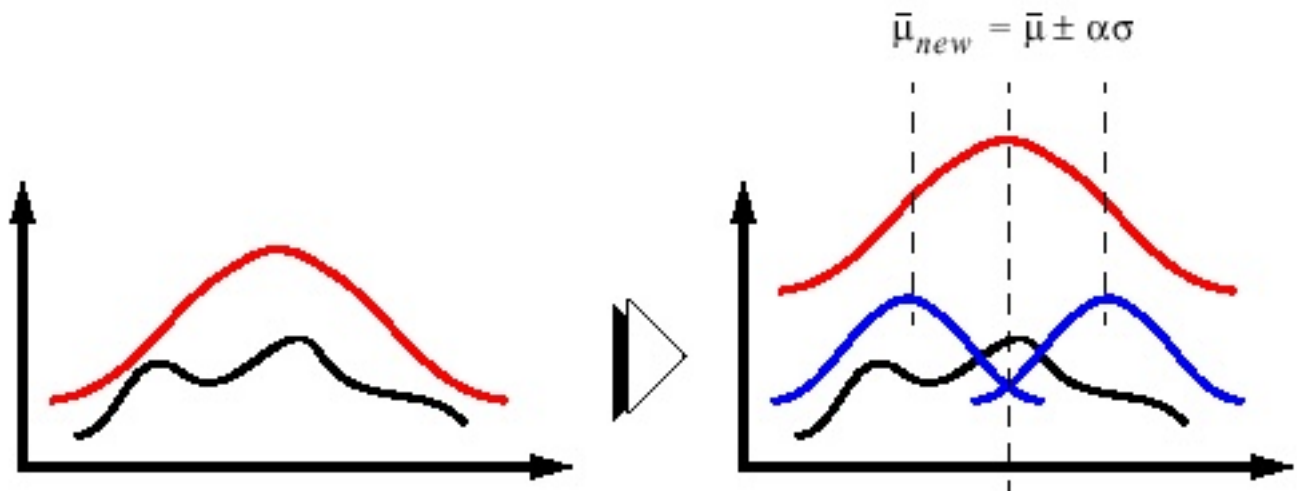
Issues:

- **Distance measure:** Euclidean? Mahalanobis?
- **Centroid computation:** Average? Median? Min-Max?
- **Splitting/Merging:** Sparsity? Separability?
- **Number of clusters:** When do we stop?

TREE-BASED CLUSTERING: VARIANCE-SPLITTING

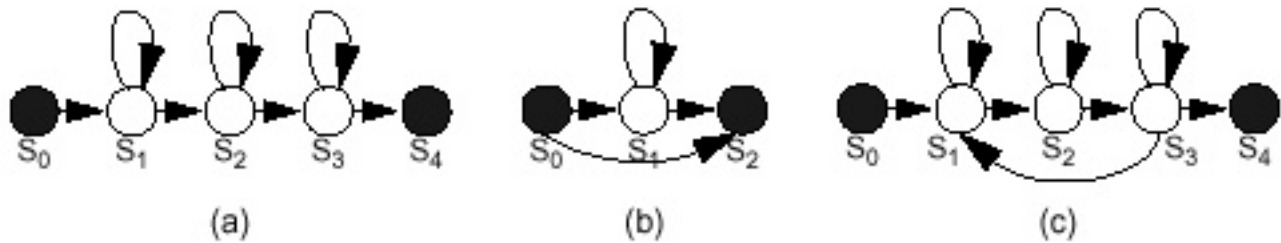
Algorithm Overview:

- Iteratively split the Gaussian with the highest mixture weight.
- Perturb the mean by a fraction of the variance:



HMM LIMITATIONS: CONDITIONAL INDEPENDENCE

Recall our basic acoustic model topology:



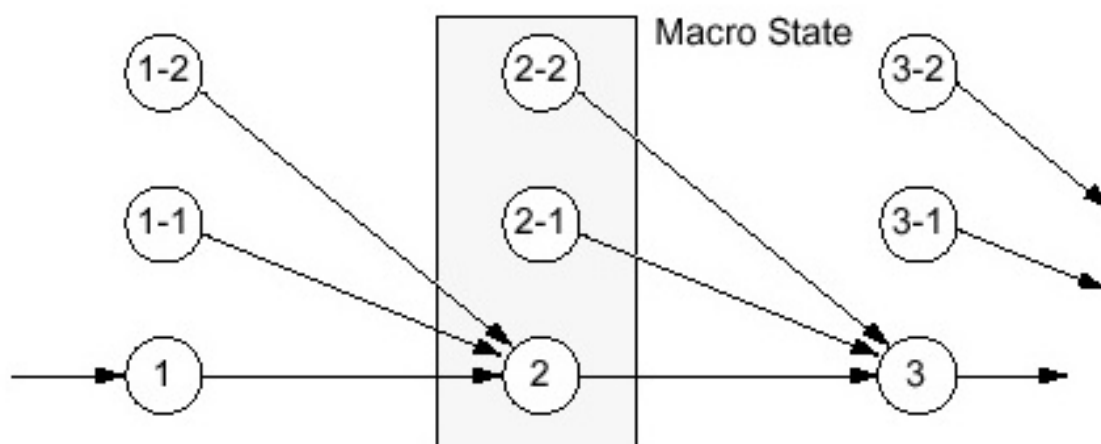
It can be argued that HMMs do not provide a realistic model for the temporal structure of speech:

- Observation probabilities for each frame (or state) are independent of previous or future frames (conditional independence). Is this a realistic model?
- The probability of staying in a state decays exponentially.

What can we do to overcome these deficiencies?

DURATION MODELING

We can explicitly model duration using an alternate acoustic model topology:



We can derive suitable reestimation equations for a probability density function at each state:

Let $d_i(\tau)$ be the probability of staying in a state i for τ frames. The transition probability from state i at time t to state j at time $t + \tau$, denoted by $\gamma_{t, \tau}$, can be written as:

$$\gamma_{t, \tau}(j|i) = \frac{\alpha_t(i)a(j|i)d_i(\tau)\left(\prod_{l=1}^{\tau} b_j(y(t+l))\right)\beta_{t+\tau}(j)}{\sum_{k=1}^N \alpha_T(k)}$$

The probability of being in state j at time t with duration τ can be computed as:

$$\gamma_{t, \tau}(j) = \sum_{i=1}^N \gamma_{t, \tau}(j|i)$$

In practice, such refinements have given minimal improvements in performance.

HIGHER-ORDER MARKOV PROCESSES

Recall our first-order Markov process:

$$P[q_t = j | (q_{t-1} = i, q_{t-2} = k, \dots)] = P[q_t = j | q_{t-1} = i]$$

We considered only those processes for which the right-hand side is independent of time:

$$a_{ij} = P[q_t = j | q_{t-1} = i] \quad 1 \leq i, j \leq N$$

We can extend this model to account for previous transitions:

$$a_{ijk} = P[q_t = k | (q_{t-1} = i, q_{t-2} = j)] \quad 1 \leq i, j, k \leq N$$

We now have a second-order Markov process. We can derive suitable maximum likelihood reestimation equations:

$$\gamma_t(k | (i, j)) = \frac{\alpha_t(i, j) a(k | (i, j)) b_k(y_t) \beta_{t+1}(i, j)}{\sum_{l=1}^N \alpha(y_1^T, l)}$$

However, in practice, the benefits of this model have not offset the significant increase in computational costs.

[Return to Main](#)[Objectives](#)**Statistical Models:**[Noisy Channel Model](#)[Wheel of Fortune](#)[Word Prediction](#)**Syntactic Constraints:**[State Machine](#)[Ad Hoc Approaches](#)[Networks](#)**Formal Language:**[Rewrite Rules](#)[Chomsky Hierarchy](#)**On-Line Resources:**[HLTSurvey](#)[Statistical Methods in NLP](#)[Software: SRILM](#)[Software: CMUSLM](#)

LECTURE 29: FORMAL LANGUAGE THEORY

- Objectives:
 - Communication Theoretic Approach
 - Chomsky Hierarchy
 - Network Grammars
 - Production Rules

This lecture combines material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System*

Development, Prentice Hall,
Upper Saddle River, New Jersey,
USA, ISBN: 0-13-022616-5,
2001.

and from this source:

F. Jelinek, *Statistical Methods for
Speech Recognition*, MIT Press,
Boston, Massachusetts, USA,
ISBN: 0-262-10066-5, 1998.

LECTURE 29: FORMAL LANGUAGE THEORY

- Objectives:
 - Communication Theoretic Approach
 - Chomsky Hierarchy
 - Network Grammars
 - Production Rules

This lecture combines material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice

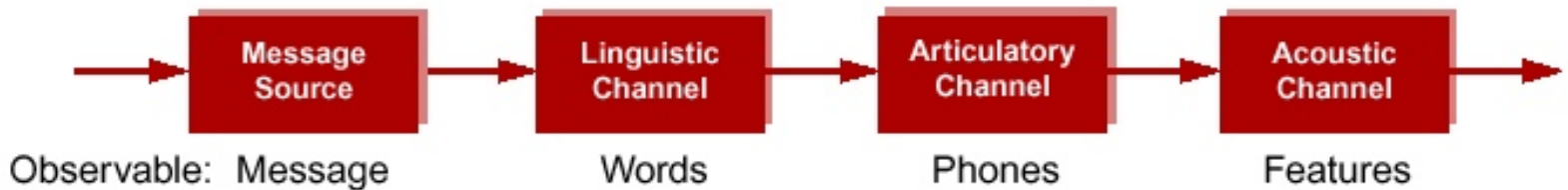
Hall, Upper Saddle River, New Jersey, USA,
ISBN: 0-13-022616-5, 2001.

and from this source:

F. Jelinek, *Statistical Methods for Speech
Recognition*, MIT Press, Boston,
Massachusetts, USA, ISBN: 0-262-10066-5,
1998.

A NOISY COMMUNICATION CHANNEL MODEL OF SPEECH RECOGNITION

A noisy communication theory model for speech production and perception:



Bayesian formulation for speech recognition:

$$P(W|A) = P(A|W)P(W)/P(A)$$

Objective: minimize the word error rate by maximizing $P(W|A)$

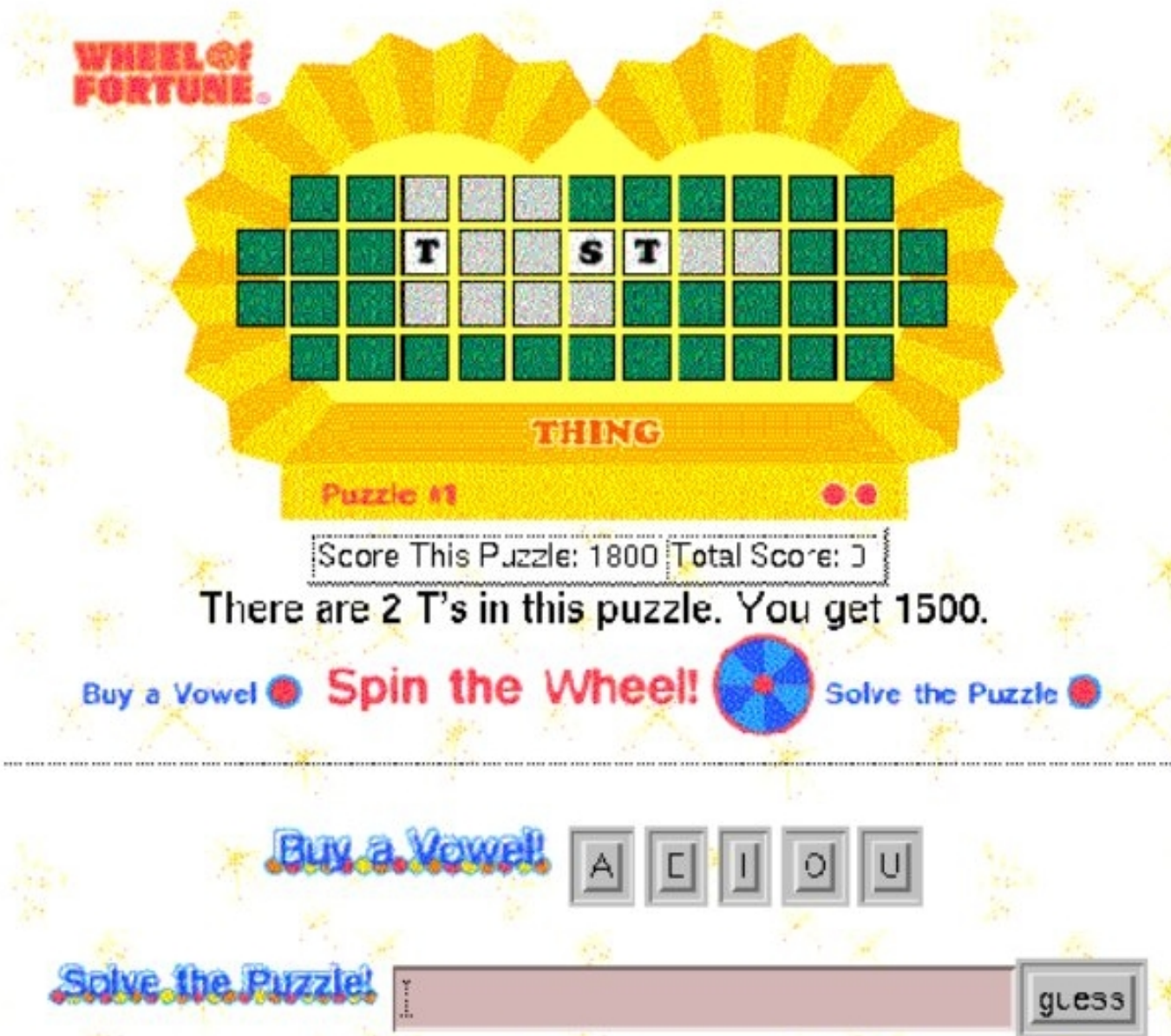
Approach: maximize $P(A|W)$ (training)

Components:

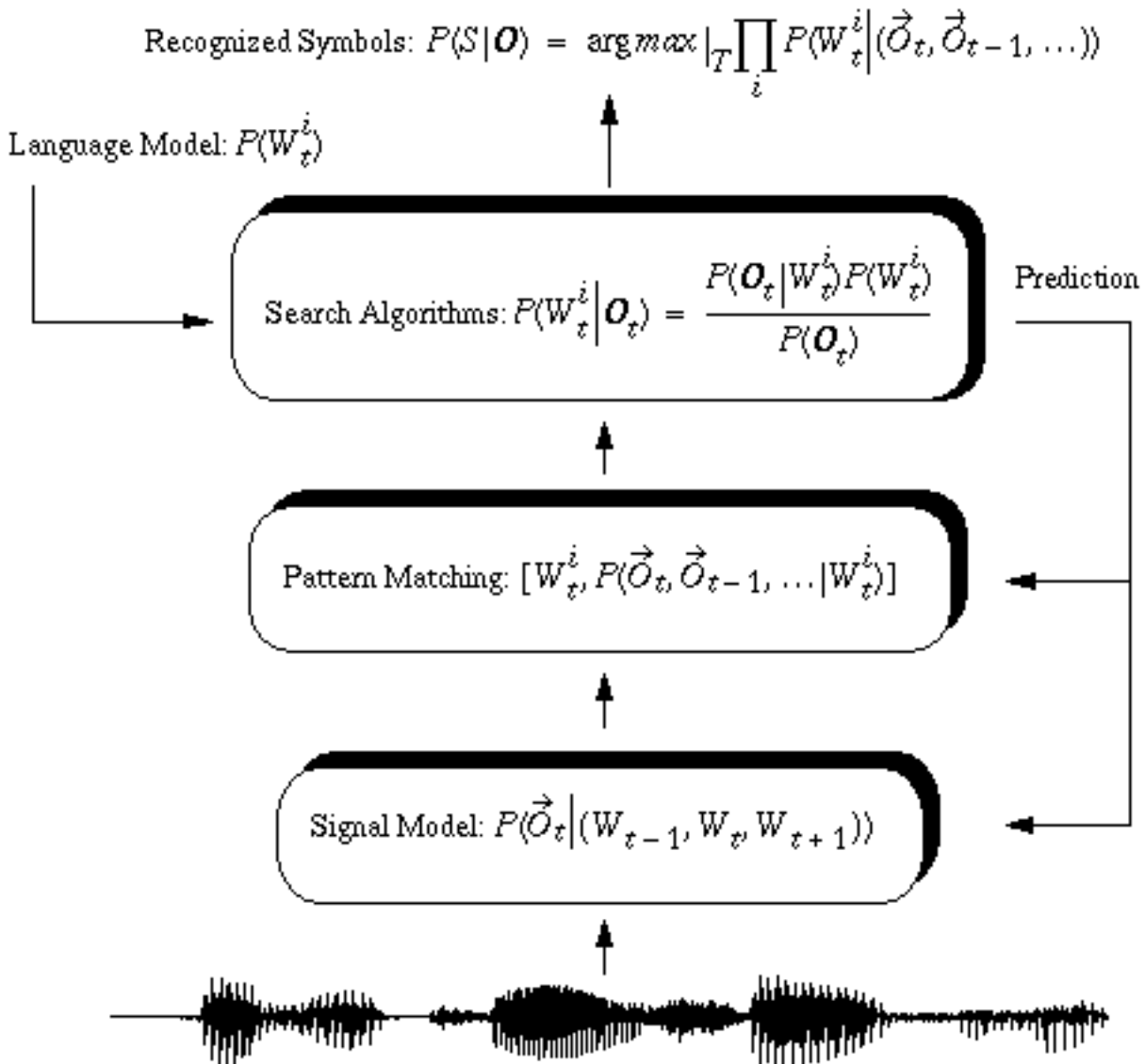
- $P(A|W)$: acoustic model (hidden Markov models, mixture of Gaussians)
- $P(W)$: language model (statistical, N-grams, finite state networks)
- $P(A)$: acoustics (ignore during maximization)

The language model typically predicts a small set of next words based on knowledge of a finite number of previous words (N-grams) — leads to search space reduction.

LANGUAGE MODELING IS SIMILAR TO PLAYING ... WHEEL ... OF ... FORTUNE

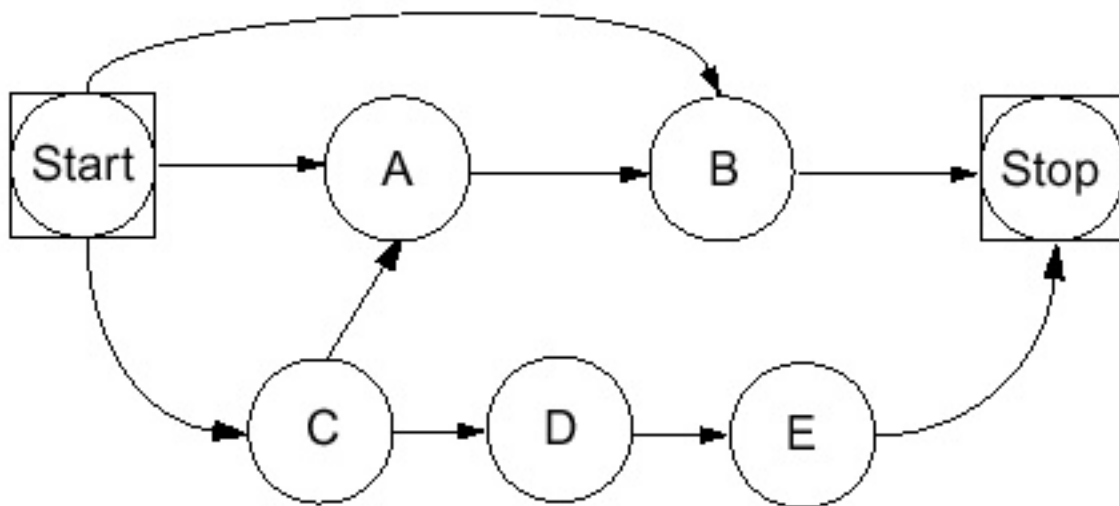


WORD PREDICTION IS CRITICAL



SYNTACTIC CONSTRAINTS CAN IMPROVE PERFORMANCE

- The search space for vocabularies of hundreds of words can become unmanageable if we allow any word to follow any other word (often called the no-grammar case)
- Our rudimentary knowledge of language tells us that, in reality, only a small subset of the vocabulary can follow a given word hypothesis, but that this subset is sensitive to the given word (we often refer to this as "context-sensitive")
- In real applications, user-interface design is crucial (much like the problem of designing GUI's), and normally results in a specification of a language or collection of sentence patterns that are permissible
- A simple way to express and manipulate this information in a dynamic programming framework is via a state machine:



For example, when you enter state C, you output one of the following words: {daddy, mommy}.

If:

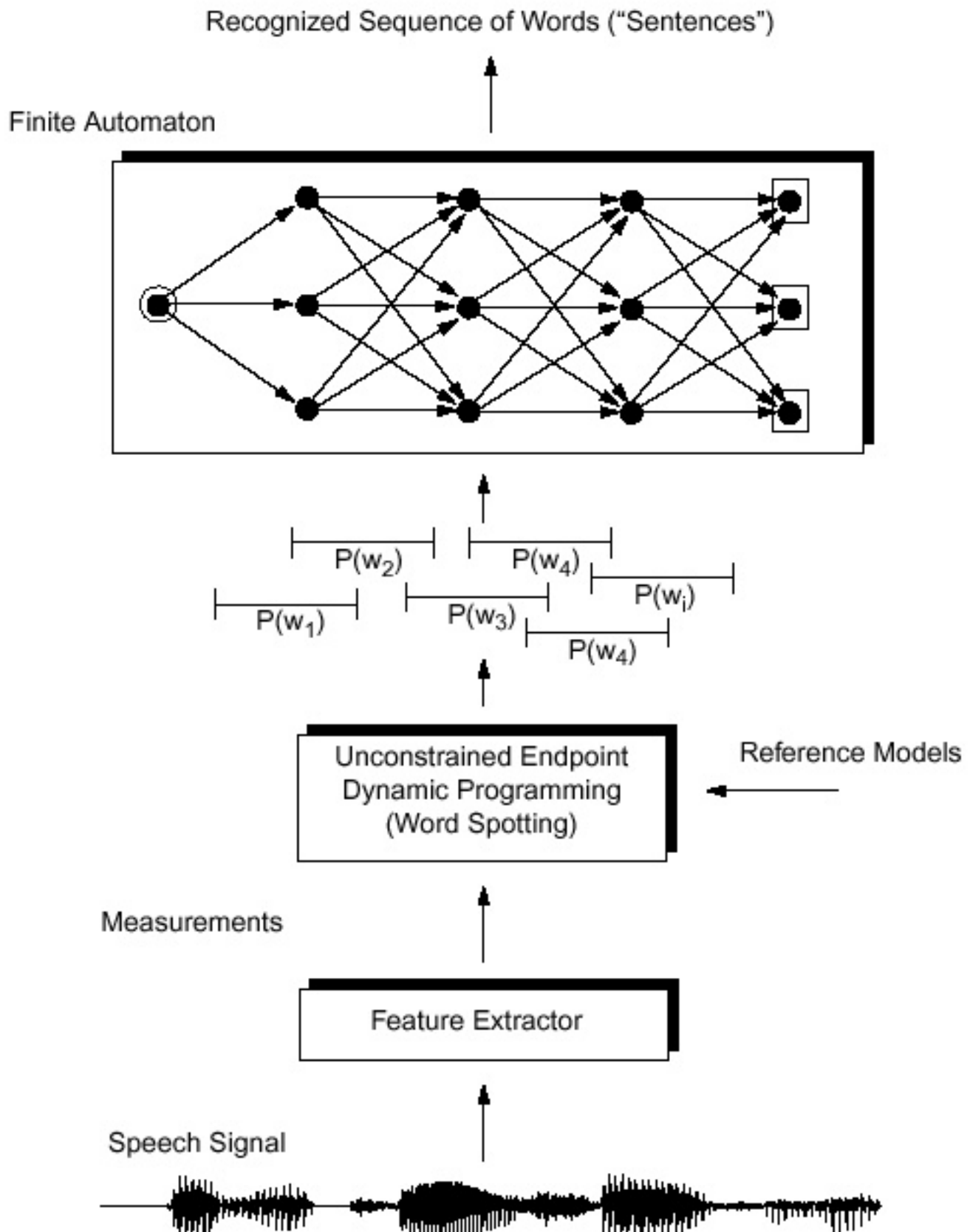
| | |
|----------|----------------|
| state A: | give |
| state B: | me |
| state C: | {daddy, mommy} |
| state D: | come |
| state E: | here |

We can generate phrases such as:

Daddy give me

- We can represent such information numerous ways (as we shall see)

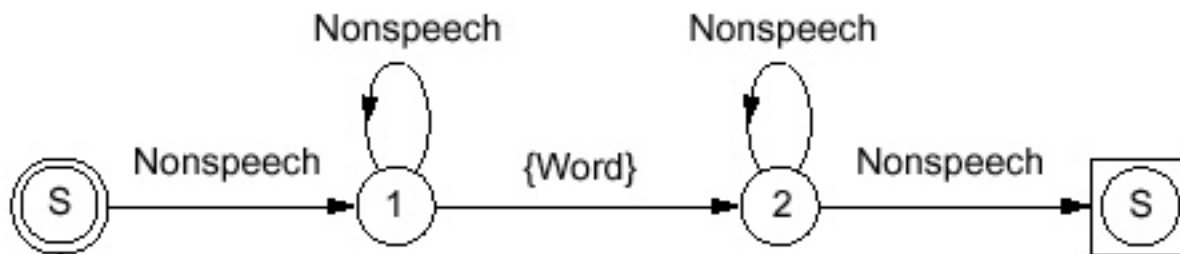
EARLY ATTEMPTS WERE AD HOC





NETWORK DECODING IS POPULAR FOR COMMAND AND CONTROL APPLICATIONS

Isolated Word Recognition:

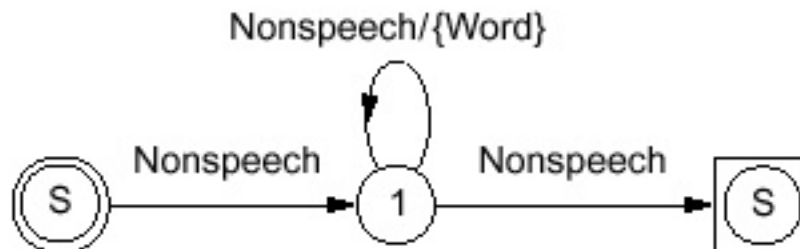


Nonspeech: typically an acoustic model of one frame in duration that models the background noise.

{Word}: any word from the set of possible words that can be spoken

- The key point here is that, with such a system, the recognizer finds the optimal start/stop times of the utterance with respect to the acoustic model inventory (a hypothesis-directed search)

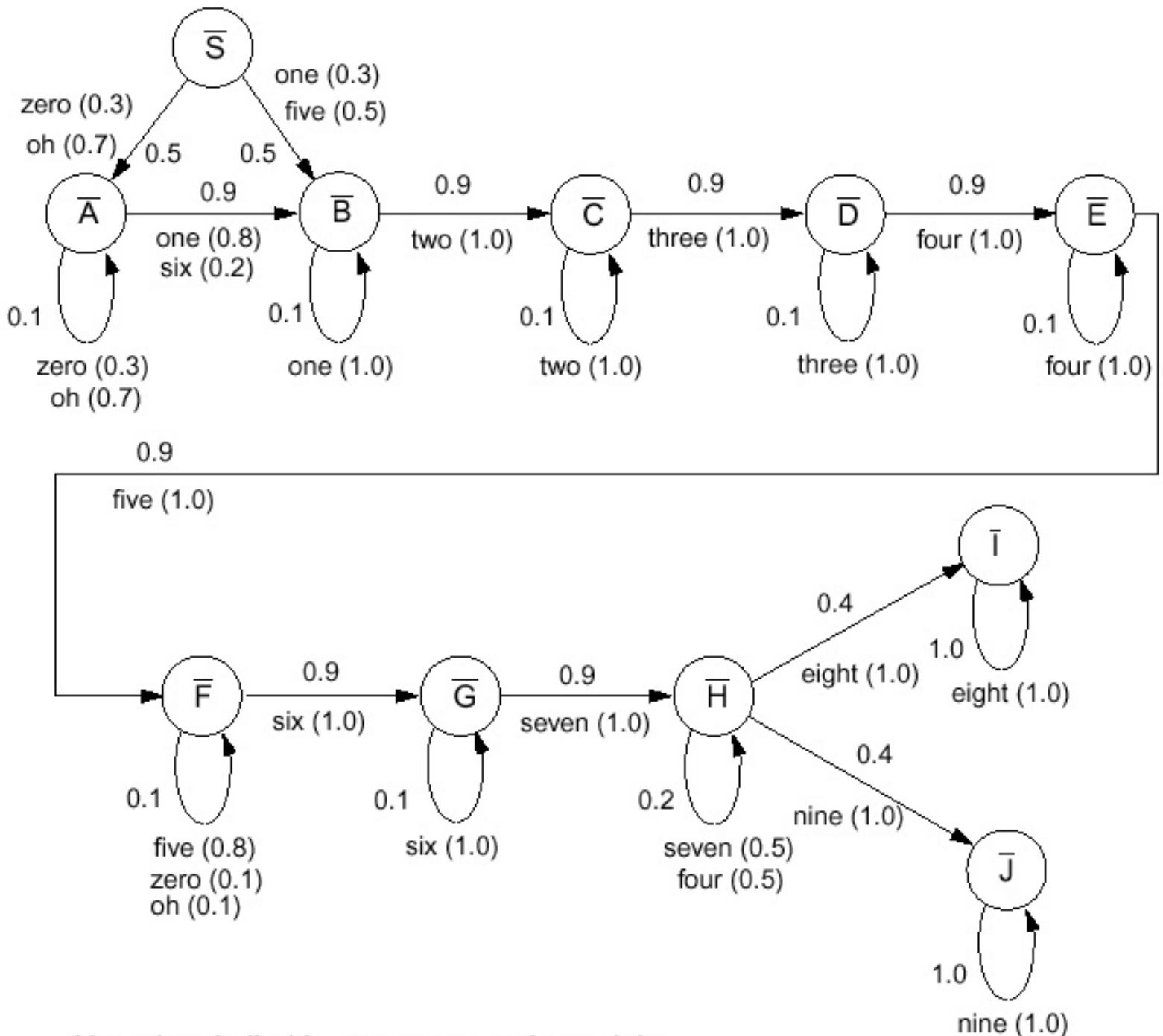
Simple Continuous Speech Recognition ("No Grammar"):



- system recognizes arbitrarily long sequences of words or nonspeech events

ALTERNATE REPRESENTATIONS OF FINITE STATE AUTOMATON

Consider the following state diagram showing a simple language model involving constrained digit sequences:



Note the similarities to our acoustic models.

What is the probability of the sequence "zero one two three four five zero six

what is the probability of the sequence "zero one two three four five zero six six seven seven eight eight" ?

How would you find the average length of a digit sequence generated from this language model?

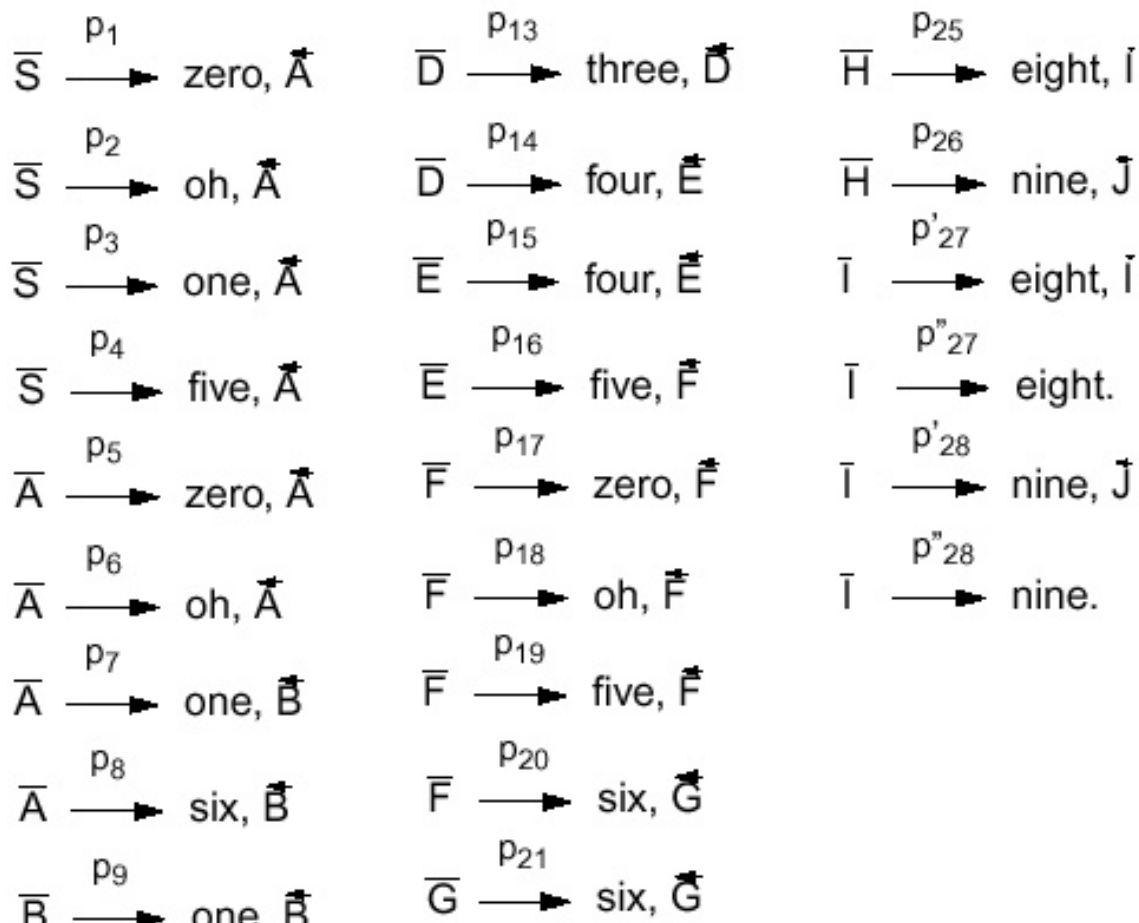
In the terminology associated with formal language theory, this HMM is known as a *finite state automaton*.

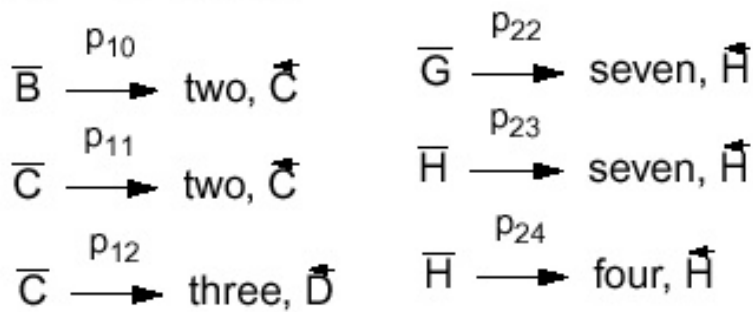
The word *stochastic* can also be applied because the transitions and output symbols are governed by probability distributions.

Further, since there are multiple transitions and observations generated at any point in time (hence, ambiguous output), this particular graph is classified as a *nondeterministic* automaton.

In the future, we will refer to this system as a stochastic finite state automaton (FSA or SFSA) when it is used to model *linguistic* information.

We can also express this system as a *regular grammar*.





Note that rule probabilities are not quite the same as transition probabilities, since they need to combine transition probabilities and output probabilities. For example, consider p_7 :

$$p_7 = (0.9)(0.8)$$

In general,

$$P(\underline{y} = y_k | \underline{x} = x_i) = \sum_j a_{ij} b(k)$$

Note that we must adjust probabilities at the terminal systems when the grammar is nondeterministic:

$$p_k = p'_k + p''_k$$

to allow generation of a final terminal.

Hence, our transition from HMMs to stochastic formal languages is clear and well-understood.

What types of language models are used?

- No Grammar (Digits)
- Sentence pattern grammars (Resource Management)
- Word Pair/Bigram (RM, Wall Street Journal)
- Word Class (WSJ, etc.)
- Trigram (WSJ, etc.)
- Back-Off Models (Merging Bigrams and Trigrams)
- Long Range N-Grams and Co-Occurrences (SWITCHBOARD)
- Triggers and Cache Models (WSJ)
- Link Grammars (SWITCHBOARD)

How do we deal with OOV and dysfluencies?

THE CHOMSKY HIERARCHY

We can categorize language models by their generative capacity:

| Type of Grammar | Constraints | Automata |
|-------------------|---|---|
| Phrase Structure | $A \rightarrow B$ | Turing Machine (Unrestricted) |
| Context Sensitive | $aAb \rightarrow aBb$ | Linear Bounded Automata (N-grams, Unification) |
| | $A \rightarrow B$ Constraint: A is a non-terminal. | |

| | | |
|--------------|--|--|
| Context Free | <p>Equivalent to:</p> $A \rightarrow w$ $A \rightarrow BC$ <p>where "w" is a terminal; B,C are non-terminals (Chomsky normal form)</p> | <p>Push down automata (JSGF, RTN, Chart Parsing)</p> |
| Regular | $A \rightarrow w$ $A \rightarrow wB$ <p>(Subset of CFG)</p> | <p>Finite-state automata (Network decoding)</p> |

- CFGs offer a good compromise between parsing efficiency and representational power.
- CFGs provide a natural bridge between speech recognition

and natural language processing.

[Return to Main](#)[Objectives](#)**Training:**[Independence](#)[Inside Probability](#)[Outside Probability](#)[Accumulation](#)[Reestimation Equation](#)**Recognition:**[Chart Parsing](#)[Example](#)**On-Line Resources:**[Manning: Probabilistic Models of Language Structure](#)[Manning: Statistical NLP](#)[Gazdar: NLP](#)[Stolcke: CFG Parsing](#)

LECTURE 30: PROBABILISTIC CONTEXT FREE GRAMMARS

● Objectives:

- Training probabilistic context free grammars
- Inside outside algorithm
- Recognition using chart parsing

This lecture combines material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

LECTURE 30: PROBABILISTIC CONTEXT FREE GRAMMARS

- Objectives:
 - Training probabilistic context free grammars
 - Inside outside algorithm
 - Recognition using chart parsing

This lecture combines material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

TRAINING PROBABILISTIC CONTEXT FREE GRAMMARS (PCFG)

The *training problem* involves estimating the probability of each rule in a probabilistic context free grammar (PCFG). Let define the probability of a rule, $A \rightarrow \alpha$ by $P(A \rightarrow \alpha | G)$.

The simplest approach to learning these probabilities would be to use a Viterbi-style training algorithm and approximate these probabilities by the number of times a rule was used divided by the total number of times a rule could have been used. This requires a hard decision. We seek a method analogous to Baum-Welch training in which we can make soft decisions:

$$P(A \rightarrow \alpha_j | G) = \frac{C(A \rightarrow \alpha_j)}{\sum_{i=1}^m C(A \rightarrow \alpha_i)}$$

where $C(\cdot)$ denotes the number of times each rule is used.

If you have hand-annotated data (parse trees), you can estimate these probabilities directly. However, such data is expensive to develop, and hence is in short supply (especially since language model training requires enormous amounts of text for LVCSR applications).

To estimate these probabilities from data, we can use EM techniques to derive the *inside-outside* algorithm. To do this, we must make one important independence assumption:

The probability of a constituent being derived by a rule is independent of how the constituent is being used as a subconstituent.

For example, we assume the probability of a noun phrase rule (NP) is the

same no matter where this NP rule is used (e.g, whether it is used as a subject of object of a verb).

INSIDE PROBABILITY CALCULATION

Let the word sequence $W = w_1 w_2 \dots w_T$ be generated by the PCFG, G with rules on the Chomsky normal form:

$$A_i \rightarrow A_m A_n$$

$$A_i \rightarrow w_i$$

where w_i is a terminal symbol (it cannot be further expanded) and A_m, A_n are non-terminal symbols.

The probability for these rules must satisfy the following constraint:

$$\sum_{m, n} P(A_i \rightarrow A_m A_n | G) + \sum_l P(A_i \rightarrow w_l | G) = 1 \quad \forall i$$

A non-terminal symbol, A_k , can generate a sequence of words $w_j w_{j+1} \dots w_k$. We define the constituent probability:

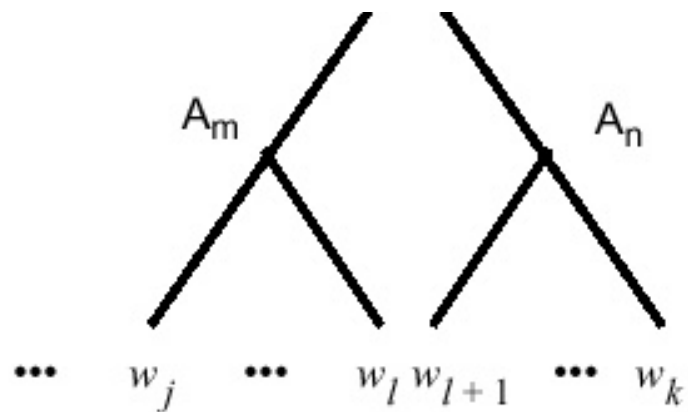
$$Inside(j, A_i, k) = P(A_i \Rightarrow w_j w_{j+1} \dots w_k | G)$$

This can be computed recursively as:

$$\begin{aligned} Inside(j, A_i, k) &= \sum_{m, n} \sum_{l=j}^{k-1} P(A_i \rightarrow A_m A_n) P(A_m \Rightarrow w_j \dots w_l) P(A_n \Rightarrow w_{l+1} \dots w_k) \\ &= \sum_{m, n} \sum_{l=j}^{k-1} P(A_i \rightarrow A_m A_n) Inside(j, A_m, l) Inside(l+1, A_n, k) \end{aligned}$$

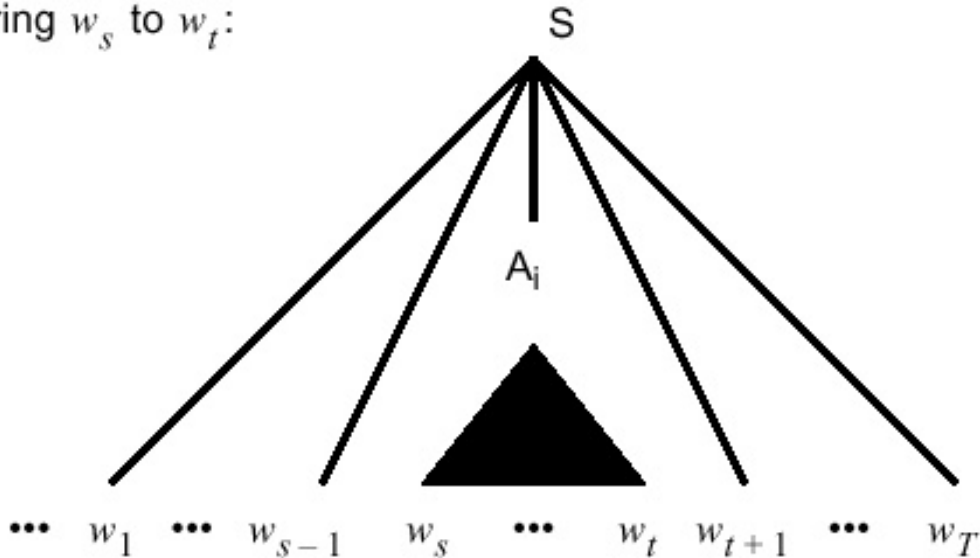
The inside probability is the sum of the probabilities of all derivations for the section over the span from j to k :

$$\bigwedge_{A_i}$$



OUTSIDE PROBABILITY CALCULATION

We can also define an outside probability for a non-terminal node, A_i , covering w_s to w_t :



$$Outside(s, A_i, t) = P(S \Rightarrow w_1 \dots w_{s-1} A_i w_t \dots w_T)$$

After the inside probabilities are computed bottom-up, we can compute the outside probabilities top-down. For each non-terminal symbol, A_i , there are one of two possible configurations $A_m \rightarrow A_n A_i$ or $A_m \rightarrow A_i A_n$:



We must consider both possibilities:

$$\text{Outside}(s, A_t, t) = P(S \Rightarrow w_1 \dots w_{s-1} A_i w_t \dots w_T)$$

$$= \sum_{m,n} \left\{ \begin{aligned} & \sum_{l=1}^{s-1} P(A_m \rightarrow A_n A_i) P(A_n \Rightarrow w_1 \dots w_{s-1}) P(S \Rightarrow w_1 \dots w_{s-1} A_i w_t \dots w_T) \\ & + \sum_{l=t+1}^T P(A_m \rightarrow A_i A_n) P(A_n \Rightarrow w_{t+1} \dots w_l) P(S \Rightarrow w_1 \dots w_{s-1} A_m w_{l+1} \dots w_T) \end{aligned} \right\}$$

$$= \sum_{m,n} \left\{ \begin{aligned} & \sum_{l=1}^{s-1} P(A_m \rightarrow A_n A_i) \text{Inside}(l, A_n, s-1) \text{Outside}(l, A_m, t) \\ & + \sum_{l=t+1}^T P(A_m \rightarrow A_i A_n) \text{Inside}(t+1, A_n, l) \text{Outside}(s, A_m, l) \end{aligned} \right\}$$

ACCUMULATING COUNTS FOR RULE PROBABILITIES

The inside and outside probabilities are used to compute the sentence probability as:

$$P(S \Rightarrow w_1 \dots w_T) = \sum_i Inside(s, A_i, t) Outside(s, A_i, t) \quad \forall s \leq t$$

The probability for the entire sentence can be computed using the forward probability:

$$P(S \Rightarrow W | G) = Inside(1, S, T)$$

The probability that a particular rule, $A_i \rightarrow A_m A_n$, is used to cover the span $w_s \dots w_t$ given the sentence and grammar is:

$$\begin{aligned} \xi(i, m, n, s, t) &= P(A_i \rightarrow w_s \dots w_t, A_i \rightarrow A_m A_n | (S \Rightarrow W, G)) \\ &= \frac{1}{P(S \Rightarrow W, G)} \sum_{k=s}^{t-1} P(A_i \rightarrow A_m A_n) Inside(s, A_m, k) Inside(k+1, A_n, t) Outside(s, A_i, t) \end{aligned}$$

REESTIMATION EQUATION

Summing these counts across all sentences gives us an estimate of the number of times a rule has been used. Dividing by the total counts of productions used for each non-terminal gives:

$$P(A_i \rightarrow A_m A_n | G) = \frac{\sum_{s=1}^{T-1} \sum_{t=s+1}^T \xi(i, m, n, s, t)}{\sum_{m, n} \sum_{s=1}^{T-1} \sum_{t=s+1}^T \xi(i, m, n, s, t)}$$

In a similar manner, we can estimate $P(A_i \rightarrow w_m | G)$, the probability a terminal symbol rule was used.

This algorithm can also be used to select rules (learn rules), dynamical prune rules, etc.

RECOGNITION: DATA-DIRECTED SEARCH VIA CHART PARSING

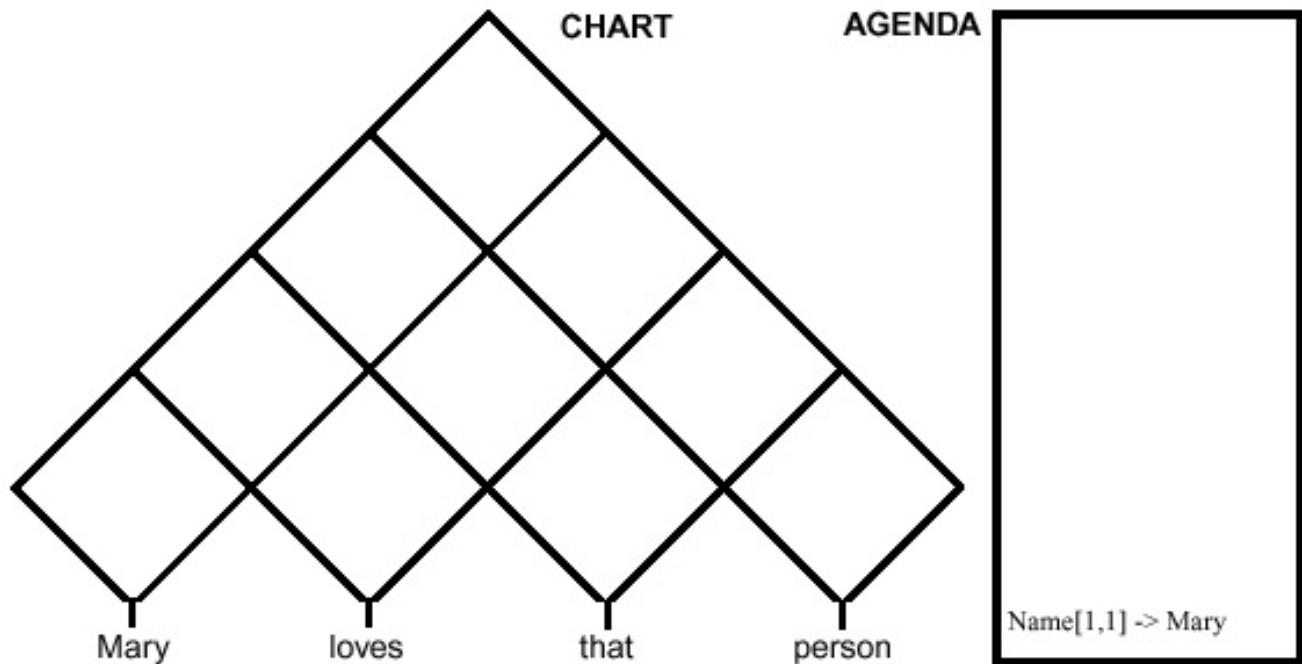
The recognition problem requires computation of $P(S \Rightarrow W|G)$. This can be performed using a probabilistic chart parser operating in breadth-first mode (and using some notion of beam pruning):

| Step | Description |
|------|---|
| 1 | Initialization: Define a list called <i>chart</i> to store active arcs, and a list called an <i>agenda</i> to store active constituents until they are added to the chart. |
| 2 | Repeat: Repeat steps 2 to 7 until there is no input left. |
| 3 | Push and pop the agenda: If the agenda is empty, look up the interpretations of the next word in the input and push them to the agenda. Pop a constituent C from the agenda. If C corresponds to position from w_i to w_j of the input sentence, we denote it $C[i,j]$. |
| 4 | Add C to the chart: Insert $C[i,j]$ into the chart. |
| 5 | Add active arcs: For each rule, add to the chart an active arc of the form $X[i, j] \rightarrow {}^\circ CY$ where ${}^\circ$ denotes the point (key) after which things are not matched. |
| 6 | Move ${}^\circ$ forward: For any active arc, add a new active arc of the form $X[1, j] \rightarrow Y...C{}^\circ...Z$. |
| 7 | Add new constituents: For any active arc of the form $X[1, l] \rightarrow Y...{}^\circ C$, add a new constituent of type $X[1,j]$ to the agenda.. |
| 8 | Exit: If $S[1,n]$ is in the chart, where n is the length of the input, |

| | |
|--|--|
| | terminate. (We can also recover all possible interpretations.) |
|--|--|

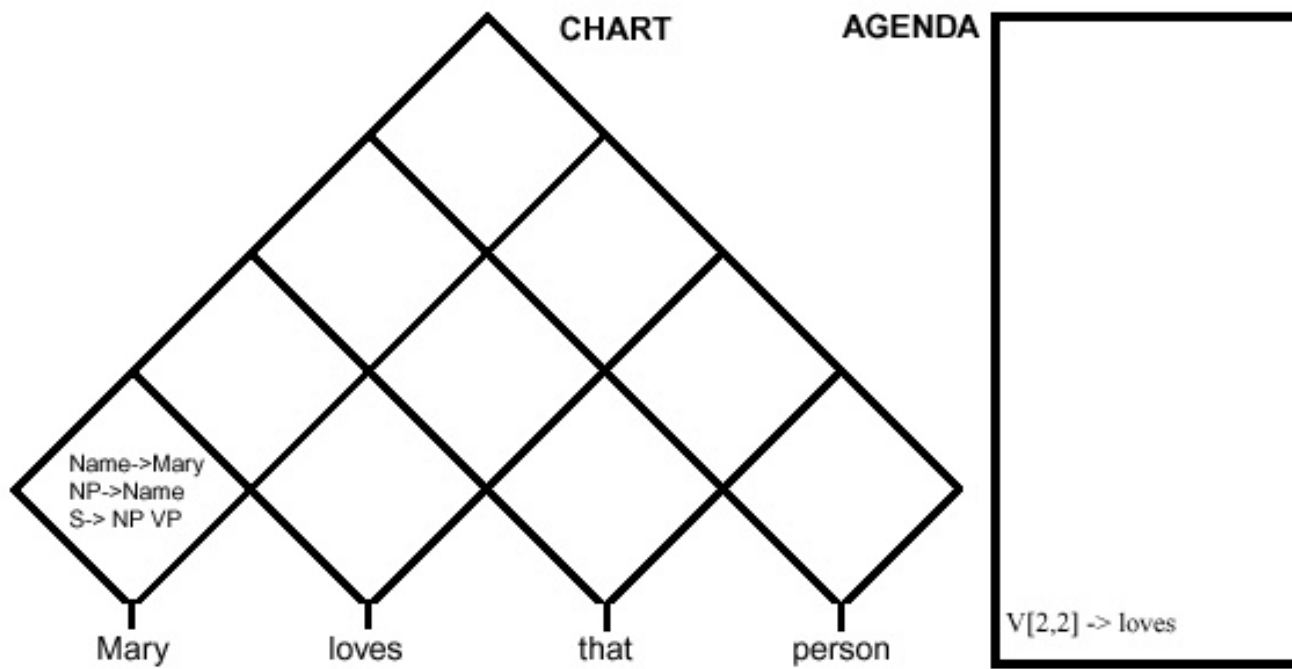
A CHART PARSING EXAMPLE

Initialization:

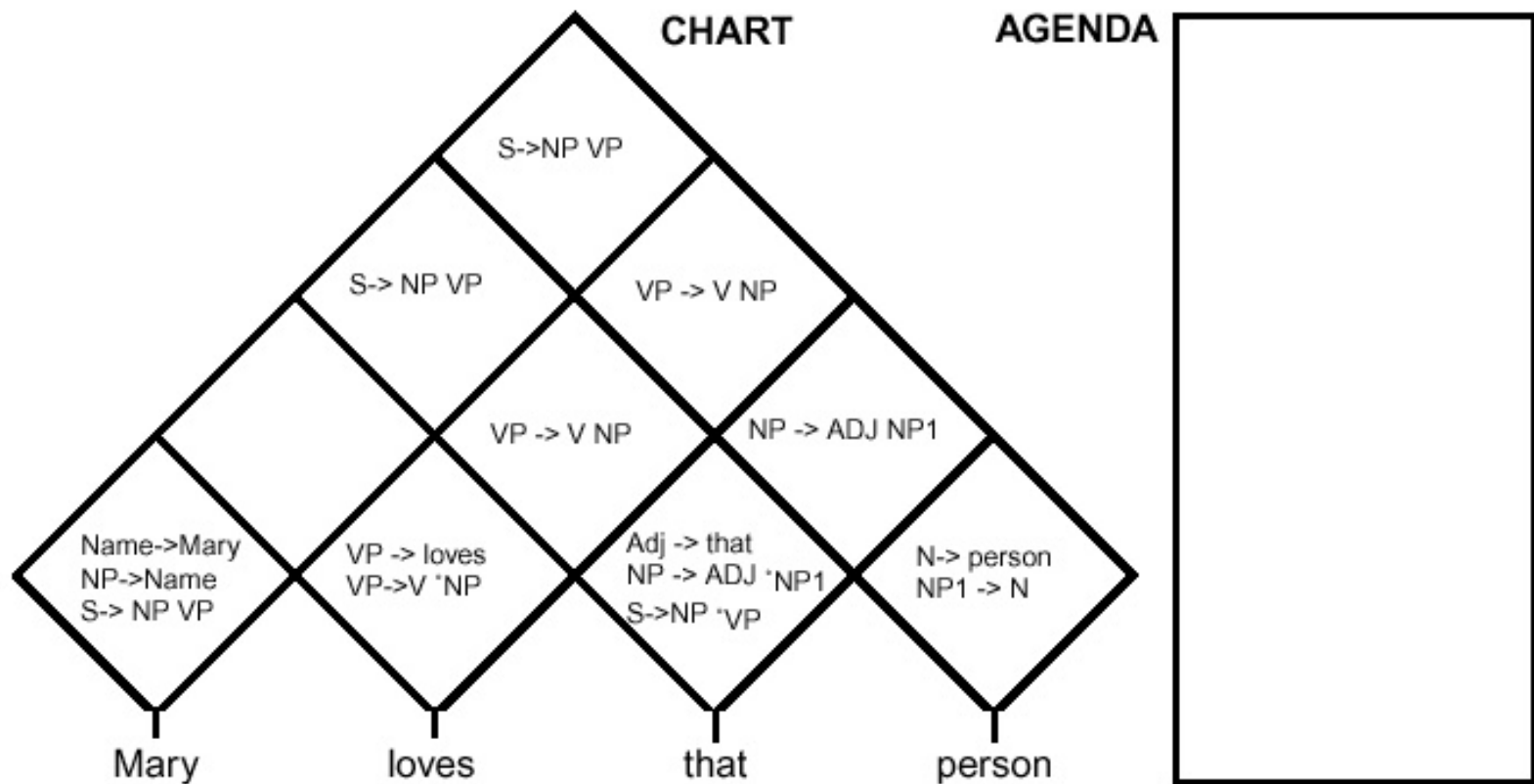


After "Mary", the chart now has rules:

- "Name -> Mary"
- "NP -> Name"
- $S \rightarrow NP \circ VP$



The chart after the entire sentence is parsed:



[Return to Main](#)

[Home](#)

[Exam Database](#)

[First Exam](#)

[Second Exam](#)

LECTURE 31: EXAM NO. 2

The second exam can be found [here](#).

LECTURE 31: EXAM NO. 2

The second exam can be found [here](#).

[Return to Main](#)[Objectives](#)**Review:**[Noisy Channel Model](#)[Chomsky Hierarchy](#)**N-grams:**[Derivation](#)[Examples](#)**Complexity:**[Perplexity](#)[Examples](#)**On-Line Resources:**[XML](#)[W3C](#)[Software: SRILM](#)

LECTURE 32: N-GRAM LANGUAGE MODELS

- Objectives:
 - Communication theory model of speech recognition
 - Statistical language models
 - N-gram language models
 - Perplexity

This lecture combines material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory,*

Algorithm, and System Development, Prentice Hall,
Upper Saddle River, New Jersey,
USA, ISBN: 0-13-022616-5,
2001.

and from this source:

F. Jelinek, *Statistical Methods for Speech Recognition*, MIT Press,
Boston, Massachusetts, USA,
ISBN: 0-262-10066-5, 1998.

LECTURE 32: N-GRAM LANGUAGE MODELS

- Objectives:
 - Communication theory model of speech recognition
 - Statistical language models
 - N-gram language models
 - Perplexity

This lecture combines material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory,*

Algorithm, and System Development, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

and from this source:

F. Jelinek, *Statistical Methods for Speech Recognition*, MIT Press, Boston, Massachusetts, USA, ISBN: 0-262-10066-5, 1998.

N-GRAM LANGUAGE MODELS

Consider a word sequence $\mathbf{W} = w_1 w_2 w_3 \dots w_n$. The probability of this word sequence can be decomposed as follows:

$$\begin{aligned} P(\mathbf{W}) &= P(w_1 w_2 w_3 \dots w_n) \\ &= P(w_1) P(w_2 | w_1) P(w_3 | w_1, w_2) \dots P(w_n | w_1, w_2, \dots, w_{n-1}) \\ &= \prod_{i=1}^n P(w_i | w_1, w_2, \dots, w_{i-1}) \end{aligned}$$

The choice of w_i thus depends on the history, which we define as the preceding $i-1$ words.

Clearly, estimating $P(w_i | w_1, w_2, \dots, w_{i-1})$ for every unique history is prohibitive. Why?

A practical approach is to assume this probability depends only on an equivalence class:

$$\begin{aligned} P(\mathbf{W}) &= \prod_{i=1}^n P(w_i | w_1, w_2, \dots, w_{i-1}) \\ &= \prod_{i=1}^n P(w_i | \Phi(w_1, w_2, \dots, w_{i-1})) \end{aligned}$$

There are three obvious simplifications we can make:

- Unigram: $\Phi(w_1, w_2, \dots, w_{i-1}) = \phi$.

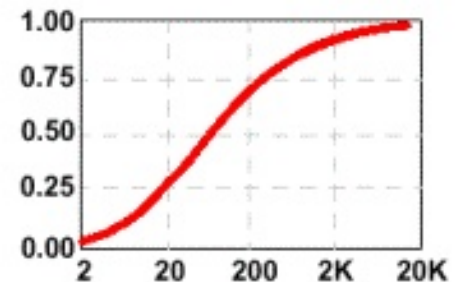
- Bigram: $\Phi(w_1, w_2, \dots, w_{i-1}) = w_{i-1}$
- Trigram: $\Phi(w_1, w_2, \dots, w_{i-1}) = w_{i-1}, w_{i-2}$

Of course, we can also merge histories based on linguistic considerations (e.g., grouping all nouns that describe animals, grouping all articles). What might be the advantages of doing this?

N-GRAM DISTRIBUTIONS FOR A CONVERSATIONAL SPEECH (SWITCHBOARD) CORPUS

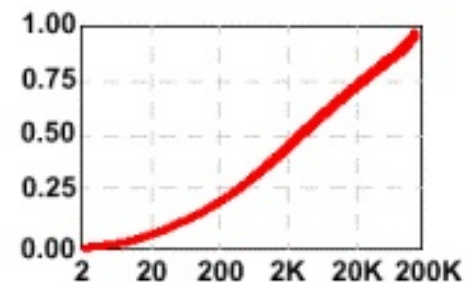
Unigrams (SWB):

- Most Common: I, and, the , you, a
- Rank-100: she, an, going
- Least Common: Abraham, Alastair, Acura



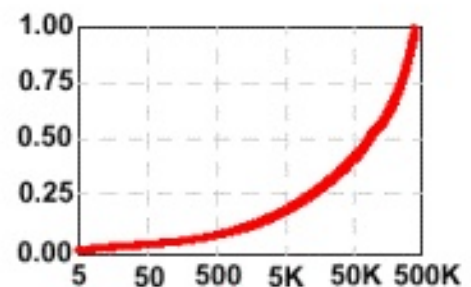
Bigrams (SWB):

- Most Common: “you know”, “yeah S!”, “!S um-hum”, “I think”
- Rank -100: “do it”, “that we”, “don’t think”
- Least Common: “raw fish”, “moisture content”, “Reagan Bush”



Trigrams (SWB):

- Most Common: “!S um-hum S!”, “a lot of”, “I don’t know”
- Rank-100: “it was a”, “you know that”
- Least Common: “you have parents”, “you seen Brooklyn”



PERPLEXITY AS A MEASURE OF COMPLEXITY

How what can measure the complexity of a language model?
What is wrong with using the average branching factor?

Consider a word sequence $\mathbf{W} = w_1 w_2 w_3 \dots w_n = w_1^n$ as a random process. The entropy of this process is:

$$\begin{aligned} H(\mathbf{W}) &= - \lim_{n \rightarrow \infty} \frac{1}{n} E[\log(P(w_1^n))] \\ &= - \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{w_1^n} P(w_1^n) \log(P(w_1^n)) \end{aligned}$$

For an ergodic source, we can use a temporal average:

$$H(\mathbf{W}) = - \lim_{n \rightarrow \infty} \frac{1}{n} \log(P(w_1^n))$$

Of course, we must estimate these probabilities from the training data:

$$\hat{H}(\mathbf{W}) = - \lim_{n \rightarrow \infty} \frac{1}{n} \log(\hat{P}(w_1^n))$$

Jelinek showed that $\hat{H}(\mathbf{W}) \geq H(\mathbf{W})$ if \mathbf{W} is ergodic.

We can define **perplexity** as:

we can define **perplexity** as:

$$PP(W) = 2^{\hat{H}(W)} \approx \frac{1}{\sqrt[n]{\hat{P}(w_1^n)}}$$

Note that if all words are equally likely, and there are L words in the vocabulary:

$$PP(W) = 2^{\log_2 L} = L$$

We can define the **training-set perplexity** as a measure of how the training set fits the language model. Similarly, we can define a **test-set perplexity** as the perplexity computed over the test set. It can be interpreted as the inverse of the (geometric) average probability assigned to each word in the test set.

PERFORMANCE VS. PERPLEXITY

- Though perplexity is not the best measure for task complexity, it provides some useful insights:

| Corpus | Vocabulary Size | Perplexity | Word Error Rate |
|--------------------------------|------------------------|-------------------|------------------------|
| TI Digits | 11 | 11 | ~0.0% |
| OGI Alphadigits | 36 | 36 | 8% |
| Resource Management (RM) | 1,000 | 60 | 4% |

| | | | |
|--|----------|-----------|-----|
| Air Travel Information Service (ATIS) | 1,800 | 12 | 4% |
| Wall Street Journal | 20,000 | 200 - 250 | 15% |
| Broadcast News | > 80,000 | 200 - 250 | 20% |
| Conversational Speech | > 50,000 | 100 - 150 | 30% |

- Acoustic confusibility of highly probable and interchangeable words most often dominates performance.

- $WER \approx -12.37 + 6.48 \cdot \log_2(\text{Perplexity})$ [William Fisher, NIST, May 2000]

[Return to Main](#)[Objectives](#)**Introduction:**[Noisy Channel Model](#)[Chomsky Hierarchy](#)[Motivation](#)**Techniques:**[Simple](#)[Generalized Interpolation](#)[Deleted Interpolation](#)[Good-Turing Estimates](#)[Katz Smoothing](#)[Knesser-Ney Bigram Smoothing](#)[Class N-grams](#)**On-Line Resources:**[LM Overview](#)[Ngram Smoothing](#)[Turing Intro](#)[Good Turing Smoothing](#)

LECTURE 33: SMOOTHING N-GRAM LANGUAGE MODELS

- Objectives:
 - Why do we need N-gram smoothing?
 - Deleted interpolation
 - Backoff language models
 - Discounting

This lecture combines material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall,

Upper Saddle River, New Jersey,
USA, ISBN: 0-13-022616-5,
2001.

and from this source:

F. Jelinek, *Statistical Methods for
Speech Recognition*, MIT Press,
Boston, Massachusetts, USA,
ISBN: 0-262-10066-5, 1998.

LECTURE 33: SMOOTHING N-GRAM LANGUAGE MODELS

- Objectives:
 - Why do we need N-gram smoothing?
 - Deleted interpolation
 - Backoff language models
 - Discounting

This lecture combines material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice

Hall, Upper Saddle River, New Jersey, USA,
ISBN: 0-13-022616-5, 2001.

and from this source:

F. Jelinek, *Statistical Methods for Speech Recognition*, MIT Press, Boston, Massachusetts, USA, ISBN: 0-262-10066-5, 1998.

WHY IS SMOOTHING SO IMPORTANT?

- A key problem in N-gram modeling is the inherent data sparseness.
- For example, in several million words of English text, more than 50% of the trigrams occur only once; 80% of the trigrams occur less than five times (see [SWB data](#) also).
- Higher order N-gram models tend to be domain or application specific. Smoothing provides a way of generating generalized language models.
- If an N-gram is never observed in the training data, can it occur in the evaluation data set?

- Solution: **Smoothing** is the process of flattening a probability distribution implied by a language model so that all reasonable word sequences can occur with some probability. This often involves broadening the distribution by redistributing weight from high probability regions to zero probability regions.

SMOOTHING IS AN INTUITIVELY SIMPLE CONCEPT

- **Simple smoothing:** pretend each bigram occurs once more than it actually does in the training data set

$$P(w_i | w_{i-1}) = \frac{1 + C(w_{i-1}, w_i)}{\sum_{w_i} (1 + C(w_{i-1}, w_i))} = \frac{1 + C(w_{i-1}, w_i)}{V + \sum_{w_i} C(w_{i-1}, w_i)}$$

- Note that the probability density function must be balanced so that it still sums to one.

THE BACKOFF MODEL: A FLEXIBLE TRADE-OFF BETWEEN ACCURACY AND COMPLEXITY

- **Backoff smoothing:** Approximate the probability of an unobserved N-gram using more frequently occurring lower order N-grams

$$P_{smooth}(w_i | w_{i-n+1} \dots w_{i-1}) = \begin{cases} \alpha(w_i | w_{i-n+1} \dots w_{i-1}) & C(w_{i-n+1} \dots w_i) > 0 \\ \gamma(w_{i-n+1} \dots w_{i-1}) P_{smooth}(w_i | w_{i-n+2} \dots w_{i-1}) & C(w_{i-n+1} \dots w_i) = 0 \end{cases}$$

- If an N-gram count is zero, we approximate its probability using a lower order N-gram.
- The scaling factor is chosen to make the conditional distribution sum to one.
- Extremely popular for N-gram modeling in speech recognition because you can control complexity as

well as generalization.

DELETED INTERPOLATION SMOOTHING

- We can linearly interpolate a bigram and a unigram model as follows:

$$P_I(w_i | w_{i-1}) = \lambda P(w_i | w_{i-1}) + (1 - \lambda) P(w_i)$$

- We can generalize this to interpolating an N-gram model using and (N-1)-gram model:

$$P_I(w_i | w_{i-n+1} \dots w_{i-1}) = \lambda_{w_i | w_{i-n+1} \dots w_{i-1}} P(w_i | w_{i-n+1} \dots w_{i-1}) + (1 - \lambda_{w_i | w_{i-n+1} \dots w_{i-1}}) P(w_i | w_{i-n+2} \dots w_{i-1})$$

Note that this leads to a recursive procedure if the lower order N-gram probability also doesn't exist. If necessary, everything can be estimated in terms of a unigram model.

- A scaling factor is used to make sure that the

conditional distribution will sum to one.

- An N-gram specific weight is used. In practice, this would lead to far too many parameters to estimate. Hence, we need to cluster such weights (by word class perhaps), or in the extreme, use a single weight.
- The optimal value of the interpolation weight can be found using Baum's reestimation algorithm. However, Bahl *et al* suggest a simpler procedure that produces a comparable result. We demonstrate the procedure here for the case of a bigram language model:
 1. Divide the total training data into kept and held-out data sets.
 2. Compute the relative frequency for the bigram and the unigram from the kept data.
 3. Compute the count for the bigram in the held-out data set.

4. Find a weight by maximizing the likelihood:

$$\sum_{N(v) \in R} \sum_{w_2} C(v, w_2) \log(\lambda \times f(w_2|w_1) + (1 - \lambda)f(w_2|w_1))$$

This is equivalent to solving this equation:

$$\sum_{N(v) \in R} \sum_{w_2} N(v, w_2) \left[\lambda + \frac{f(w_2|v)}{f(w_3) - f(w_2|v)} \right]^{-1} = 0$$

GOOD-TURING ESTIMATES

The Good-Turing estimate states that for any N-gram, α , that occurs r times, we should reestimate this frequency of occurrence as:

$$r^* = (r + 1) \frac{n_r + 1}{n_r}$$

where n_r is the number of N-grams that occur exactly r times. This count can be converted to a probability by dividing by the total number of N-gram tokens:

$$P(\alpha) = \frac{r^*}{N}$$

where

$$N = \sum_{r=0}^{\infty} n_r r = \sum_{r=0}^{\infty} n_r r^* = \sum_{r=0}^{\infty} (r + 1) n_{r+1}.$$

The justification for this equation is as follows:

Suppose we have training data for N-grams $\alpha_1 \dots \alpha_s$. Let $c(\alpha_i)$ denote the number of times the N-gram α_i occurs in the training data, and p_i be the true probability of α_i .

Estimating p_i by using its frequency of occurrence can be expanded as:

$$E(p_i | c(\alpha_i) = r) = \sum_{k=1}^s p(i = k | c(\alpha_i) = r) p_k$$

We are using the chance that a randomly selected N-gram, α_i , with count r , is in fact α_k . This can be rewritten into:

$$\begin{aligned}
 p(i = k | c(\alpha_i) = r) &= \frac{p(c(\alpha_k) = r)}{\sum_{l=1}^s p(c(\alpha_l) = r)} \\
 &= \frac{\binom{N}{r} p_k^r (1 - p_k)^{N-r}}{\sum_{l=1}^s \binom{N}{r} p_l^r (1 - p_l)^{N-r}} \\
 &= \frac{p_k^r (1 - p_k)^{N-r}}{\sum_{l=1}^s p_l^r (1 - p_l)^{N-r}}
 \end{aligned}$$

Substituting this into our expression for $E(p_i | c(\alpha_i) = r)$:

$$E(p_i | c(\alpha_i) = r) = \frac{\sum_{k=1}^s (p_k^r (1 - p_k)^{N-r}) p_k}{\sum_{l=1}^s p_l^r (1 - p_l)^{N-r}}$$

Noting that every N-gram token counts as 1, we can express the expected value of n_r as:

$$E(n_r) = \sum_{i=1}^s p(c(\alpha_i) = r) = \sum_{i=1}^s \binom{N}{r} p_i^r (1 - p_i)^{N-r}$$

We can show that:

$$\frac{r+1}{N+1} \frac{E_{N+1}(n_r + 1)}{E_N(n_r)} = E(p_i | c(\alpha_i) = r)$$

We can make the approximation that $n_r \approx E_N(n_r)$ and

$$n_{r+1} \approx \frac{N}{N+1} E_{N+1}(n_r + 1)$$

Combining these results:

$$\begin{aligned} P(\alpha_i) &= \frac{r^*}{N} \\ r^* &= NP(\alpha_i) \\ &= NE(p_i | c(\alpha_i) = r) \\ &= N \frac{r+1}{N+1} \frac{E_{N+1}(n_r + 1)}{E_N(n_r)} \\ &\approx (r+1) \frac{n_{r+1}}{n_r} \end{aligned}$$

Note that we must pre-smooth the distribution so $n_r > 0$.

KATZ SMOOTHING BASED ON GOOD-TURING ESTIMATES

- **Katz smoothing** applies Good-Turing estimates to the problem of backoff language models.
- Katz smoothing uses a form of *discounting* in which the amount of discounting is proportional to that predicted by the Good-Turing estimate.
- The total number of counts discounted in the global distribution is equal to the total number of counts that should be assigned to N-grams with zero counts according to the Good-Turing estimate (preserving the unit area constraint for the pdf).

● Katz Smoothing:

$$P_{Katz}(w_i | w_{i-1}) = \begin{cases} C(w_{i-1}w_i)/C(w_{i-1}) & r > k \\ d_r C(w_{i-1}w_i)/C(w_{i-1}) & k \geq r > 0 \\ \alpha(w_{i-1})P(w_i) & r = 0 \end{cases}$$

$$\text{where } d_r = \frac{\frac{r^*}{r} - \frac{(k+1)n_{k+1}}{n_1}}{1 - \frac{(k+1)n_{k+1}}{n_1}} \text{ and } \alpha(w_{i-1}) = \frac{1 - \sum_{w_i: r > 0} P_{Katz}(w_i | w_{i-1})}{1 - \sum_{w_i: r > 0} P_{Katz}(w_i)}.$$

KNESER-NEY BIGRAM SMOOTHING

- **Absolute discounting** involves subtracting a fixed discount, D , from each nonzero count, and redistributing this probability mass to N-grams with zero counts.
- We implement absolute discounting using an interpolated model:

$$P_{abs}(w_i | w_{i-n+1} \dots w_{i-1}) = \frac{\max\{C(w_{i-n+1} \dots w_i) - D, 0\}}{\sum_{w_i} C(w_{i-n+1} \dots w_i)} + (1 - \lambda_{w_{i-n+1} \dots w_{i-1}}) P_{abs}(w_i | w_{i-n+2} \dots w_{i-1})$$

- **Kneser-Ney smoothing** combines notions of discounting with a backoff model. Here is an

algorithm for bigram smoothing:

$$P_{KN}(w_i | w_{i-1}) = \begin{cases} \frac{\max\{C(w_{i-1}w_i) - D, 0\}}{C(w_{i-1})} & C(w_{i-1}w_i) > 0 \\ \alpha(w_{i-1})P_{KN}(w_i) & \text{otherwise} \end{cases}$$

where

$$P_{KN}(w_i) = \frac{C(\bullet w_i)}{\sum_{w_i} C(\bullet w_i)}$$

and $C(\bullet w_i)$ is the number of unique words preceding w_i .

$\alpha(w_{i-1})$ is chosen to make the distribution sum to 1:

$$\alpha(w_{i-1}) = \frac{1 - \sum_{w_i: C(w_{i-1}w_i) > 0} \frac{\max\{C(w_{i-1}w_i) - D, 0\}}{C(w_{i-1})}}{1 - \sum_{w_i: C(w_{i-1}w_i) > 0} P_{KN}(w_i)}$$

- Knesser-Ney smoothing constructs a lower order distribution that is consistent with the smoothed higher order distribution.

CLASS N-GRAMS

- Recall we previously discussed defining equivalence classes for words that exhibit similar semantic and grammatical behavior.
- Class based language models have been shown to be effective for reducing memory requirements for real-time speech applications, and supporting rapid adaption of language models.
- A word probability can be conditioned on the previous N-1 word classes:

$$P(w_i | c_{i-n+1} \dots c_i) = P(w_i | c_i) P(c_i | c_{i-n+1} \dots c_i)$$

- We can express the probability of a word sequence in terms of class N-grams:

$$P(W) = \sum_{c_1 \dots c_n} \prod_i P(w_i | c_i) P(c_i | c_{i-n+1} \dots c_i)$$

- If the classes are non-overlapping:

$$P(W) = \prod_i P(w_i | c_i) P(c_i | c_{i-n+1} \dots c_i)$$

- If we consider the case of a bigram language model, we can derive a simple estimate for a bigram probability in terms of word and class counts:

$$\begin{aligned} P(w_i | w_{i-1}) &\approx P(w_i | c_{i-1}) = P(w_i | c_i) P(c_i | c_{i-1}) \\ &= \frac{C(w_i c_i)}{C(c_i)} \frac{C(c_{i-1} c_i)}{C(c_{i-1})} \\ &= \frac{C(w_i)}{C(c_i)} \frac{C(c_{i-1} c_i)}{C(c_{i-1})} \end{aligned}$$

- Class N-grams have not provided significant improvements in performance, but have provided a simple means of integrating linguistic knowledge and data-driven statistical knowledge.

[Return to Main](#)[Objectives](#)**Introduction:**[Technology](#)[Motivation](#)**Techniques:**[General Search](#)[Depth-First](#)[Breadth-First](#)[Best-First](#)[Beam Search](#)[Hierarchical Search](#)**On-Line Resources:**[Tutorial](#)[Hierarchical](#)[AI Search](#)[Code and Complexity](#)

LECTURE 34: BASIC SEARCH ALGORITHMS

- Objectives:
 - The importance of search in speech recognition
 - General search algorithms
 - Breadth-First vs. Depth-First
 - Beam Search

This lecture follows the course textbook closely:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory,*

Algorithm, and System Development, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

This material can also be found in most computer science textbooks on algorithms:

T. Corment, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, McGraw-Hill, New York, New York, USA, ISBN: 0-07-013151-1.

LECTURE 34: BASIC SEARCH ALGORITHMS

- Objectives:
 - The importance of search in speech recognition
 - General search algorithms
 - Breadth-First vs. Depth-First
 - Beam Search

This lecture follows the course textbook closely:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice

Hall, Upper Saddle River, New Jersey, USA,
ISBN: 0-13-022616-5, 2001.

This material can also be found in most
computer science textbooks on algorithms:

T. Corment, C. Leiserson, R. Rivest, and C.
Stein, *Introduction to Algorithms*,
McGraw-Hill, New York, New York, USA,
ISBN: 0-07-013151-1.

SPEECH RECOGNITION REQUIRES GOOD PATTERN RECOGNITION AND SEARCH

- Continuous speech recognition is both a pattern recognition and search problem.
Why?
- The decoding process of a speech recognizer finds the most probable sequence of words given the acoustic and language models.
Recall our basic equation for speech recognition:

$$P(W|A) = \frac{P(W)P(A|W)}{P(A)}$$

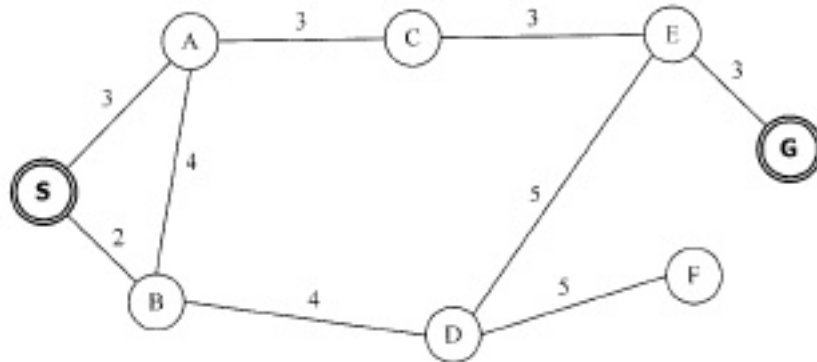
Search is the process of finding the most probable word sequence:

$$\begin{aligned}\hat{W} &= \operatorname{argmax}_W \left[\frac{P(W)P(A|W)}{P(A)} \right] \\ &= \operatorname{argmax}_W [P(W)P(A|W)]\end{aligned}$$

- The complexity of the search algorithm depends heavily on the nature of the search space, which in turn, depends heavily on the language model constraints (e.g., networks vs. N-grams).
- Speech recognition typically uses a hierarchical Viterbi beam search for decoding/recognition, and A* stack decoding for N-best and word graph generation.

GENERAL GRAPH SEARCH

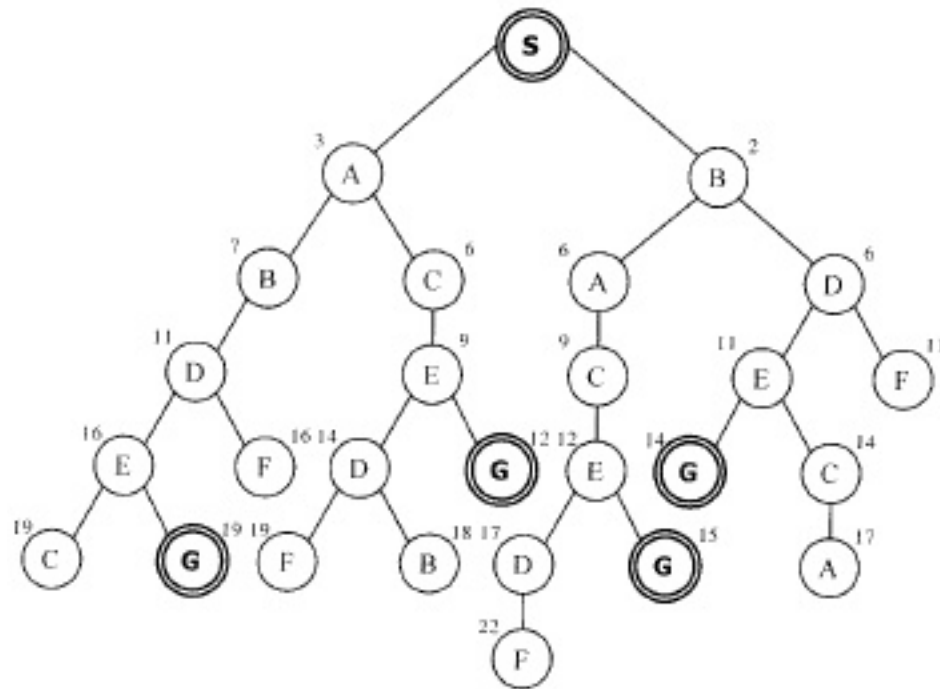
- Many interesting and useful problems cannot be handled solely by dynamic programming. For example, consider the traveling salesman problem - finding the shortest distance tour covering N cities and only visiting each city once:



The complexity of an exhaustive search solution to such problems can be $O(N^T)$ - which is prohibitive for speech recognition.

- A search tree solution to the traveling

salesman problem is show below:



- The search space is defined by a triplet (S, O, G) , where S is the set of initial states, O is a set of operators or rules, and G is a set of goal states.
- A general algorithm for searching such spaces can be defined as follows:

ALGORITHM 12.1: THE GRAPH-SEARCH ALGORITHM

Step 1: Initialization: Put S in the *OPEN* list and create an initially empty *CLOSE* list.

Step 2: If the *OPEN* list is empty, exit and declare failure.

Step 3: Pop up the first node N in the *OPEN* list, remove it from the *OPEN* list and put it into the *CLOSE* list.

Step 4: If node N is a goal node, exit successfully with the solution obtained by tracing back the path along the pointers from N to S .

Step 5: Expand node N by applying the successor operator to generate the successor set $SS(N)$ of node N . Be sure to eliminate the ancestors of N from $SS(N)$.

Step 6: $\forall v \in SS(N)$ do

6a. (optional) If $v \in OPEN$ and the accumulated distance of the new path is smaller than that for the one in the *OPEN* list, do

 (i) change the traceback (parent) pointer of v to N and adjust the accumulated distance for v .

 (ii) go to Step 7.

6b. (optional) If $v \in CLOSE$ and the accumulated distance of the new path is smaller than the partial path ending at v in the *CLOSE* list, do

 (i) change the traceback (parent) pointer of v to N and adjust the accumulated distance for all paths that contain v .

 (ii) go to Step 7.

6c. Create a pointer pointing to N and push it into the *OPEN* list.

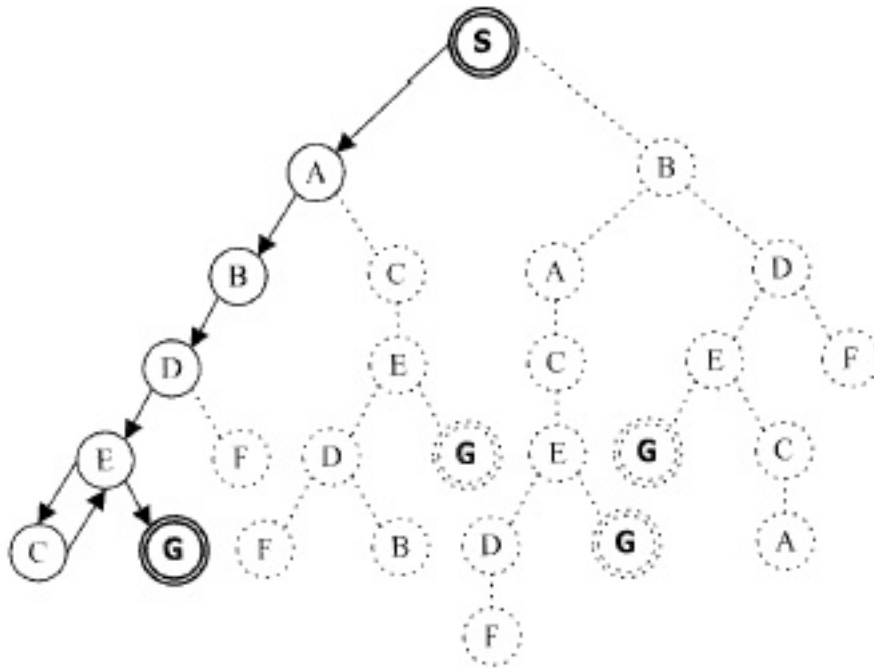
Step 7: Reorder the *OPEN* list according to search strategy or some heuristic measurement.

Step 8: Go to Step 2.

- An important part of any search algorithm is the *successor operator* which generates the list of all possible nodes that can follow a given node, and computes the distance associated with each of these arcs.

DEPTH-FIRST SEARCH

- Depth-first search explores a single path until its conclusion. If this path does not terminate on a goal state, we backtrack and arbitrarily continue with another path:



Such a strategy is common for solving problems such as mazes where the first solution that reaches a goal state is acceptable (though this might not be the fastest solution).

- A general algorithm for searching such spaces can be defined as follows:

ALGORITHM 12.2: THE DEPTH-FIRST SEARCH ALGORITHM

Step 1: Initialization: Put S in the *OPEN* list and create an initially empty the *CLOSE* list.

Step 2: If the *OPEN* list is empty, exit and declare failure.

Step 3: Pop up the first node N in the *OPEN* list, remove it from the *OPEN* list and put it into the *CLOSE* list.

Step 4: If node N is a goal node, exit successfully with the solution obtained by tracing back the path along the pointers from N to S .

4a. If the depth of node N is equal to the depth bound, go to Step 2.

Step 5: Expand node N by applying the successor operator to generate the successor set $SS(N)$ of node N . Be sure to eliminate the ancestors of N from $SS(N)$.

Step 6: $\forall v \in SS(N)$ do

6c. Create a pointer pointing to N and push it into the *OPEN* list.

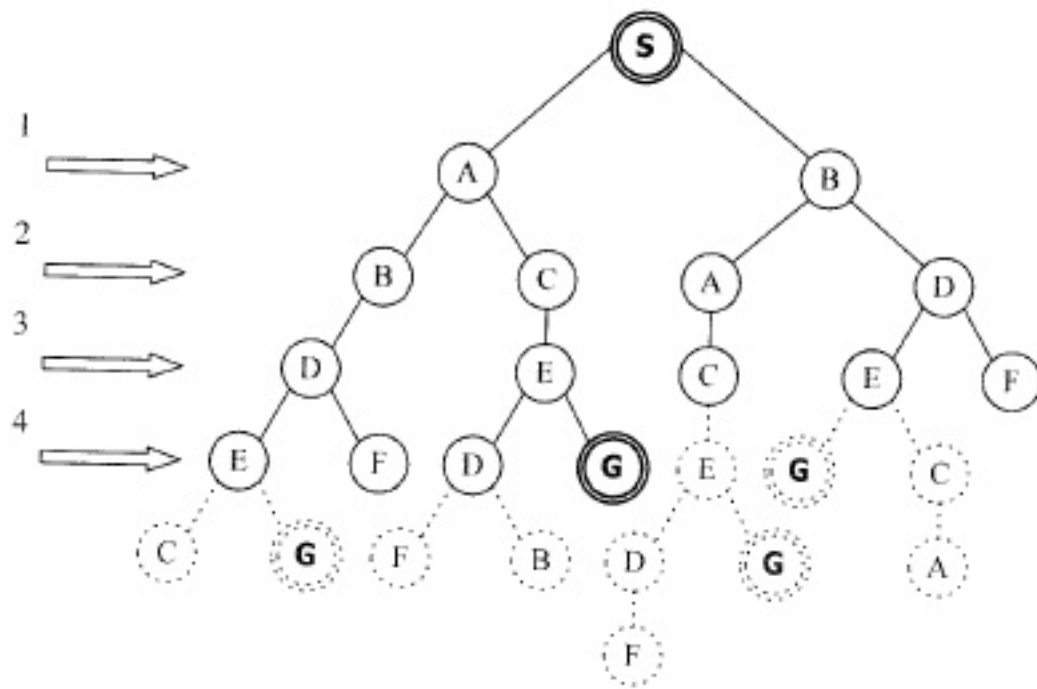
Step 7: Reorder the the *OPEN* list in descending order of the depth of the nodes.

Step 8: Go to Step 2.

- Depth-first search, as we will see, is useful when dealing with beam search and fast-matching algorithms.

BREADTH-FIRST SEARCH

- Breadth-first search explores all alternatives simultaneously level-by-level:



- A general algorithm for searching such spaces can be defined as follows:

ALGORITHM 12.3: THE BREADTH-FIRST SEARCH ALGORITHM

Step 1: Initialization: Put S in the *OPEN* list and create an initially empty the *CLOSE* list.

Step 2: If the *OPEN* list is empty, exit and declare failure.

Step 3: Pop up the first node N in the *OPEN* list, remove it from the *OPEN* list and put it into the *CLOSE* list.

Step 4: If node N is a goal node, exit successfully with the solution obtained by tracing back the path along the pointers from N to S .

Step 5: Expand node N by applying the successor operator to generate the successor set $SS(N)$ of node N . Be sure to eliminate the ancestors of N , from $SS(N)$.

Step 6: $\forall v \in SS(N)$ do

 6c. Create a pointer pointing to N and push it into the *OPEN* list.

Step 7: Reorder the *OPEN* list in increasing order of the depth of the nodes.

Step 8. Go to Step 2.

- Breadth-first search is a critical part of a speech recognition system. Why?

HEURISTIC SEARCH: BEST-FIRST (A* SEARCH)

- Best-first search uses an evaluation function, $h(N)$, which indicates the relative goodness of pursuing that node. If we combine this with the partial path score, we can define a general evaluation function:

$$f(N) = g(N) + h(N)$$

which can be used to evaluate hypotheses as they evolve. If we always pursue the best path according to this evaluation function, what are the merits of this approach? What constraints must be placed on this function to guarantee an optimal solution? How would that solution compare to other search algorithms?

- A general algorithm for searching such spaces can be defined as follows:

ALGORITHM 12.4: THE BEST-FIRST SEARCH ALGORITHM

Step 1: Initialization: Put S in the *OPEN* list and create an initially empty the *CLOSE* list.

Step 2: If the *OPEN* list is empty, exit and declare failure.

Step 3: Pop up the first node N in the *OPEN* list, remove it from the *OPEN* list and put it into the *CLOSE* list.

Step 4: If node N is a goal node, exit successfully with the solution obtained by tracing back the path along the pointers from N to S .

Step 5: Expand node N by applying the successor operator to generate the successor set $SS(N)$ of node N . Be sure to eliminate the ancestors of N , from $SS(N)$.

Step 6: $\forall v \in SS(N)$ do

6a. (optional) If $v \in OPEN$ and the accumulated distance of the new path is smaller than that for the one in the the *OPEN* list, do

 (i) Change the traceback (parent) pointer of v to N and adjust the accumulated distance for v .

 (ii) Evaluate heuristic function $f(v)$ for v and go to Step 7.

6b. (optional) If $v \in CLOSE$ and the accumulated distance of the new path is small than the partial path ending at v in the the *CLOSE* list,

 (i) Change the traceback (parent) pointer of v to N and adjust the accumulated distance and heuristic function f for all the paths containing v .

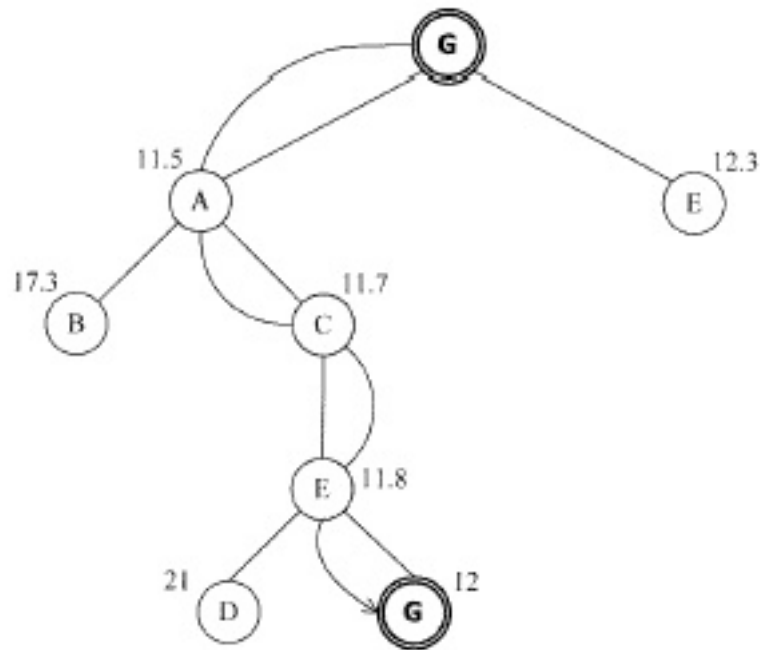
 (ii) go to Step 7.

6c. Create a pointer pointing to N and push it into the *OPEN* list.

Step 7: Reorder the the *OPEN* list in the increasing order of the heuristic function $f(N)$.

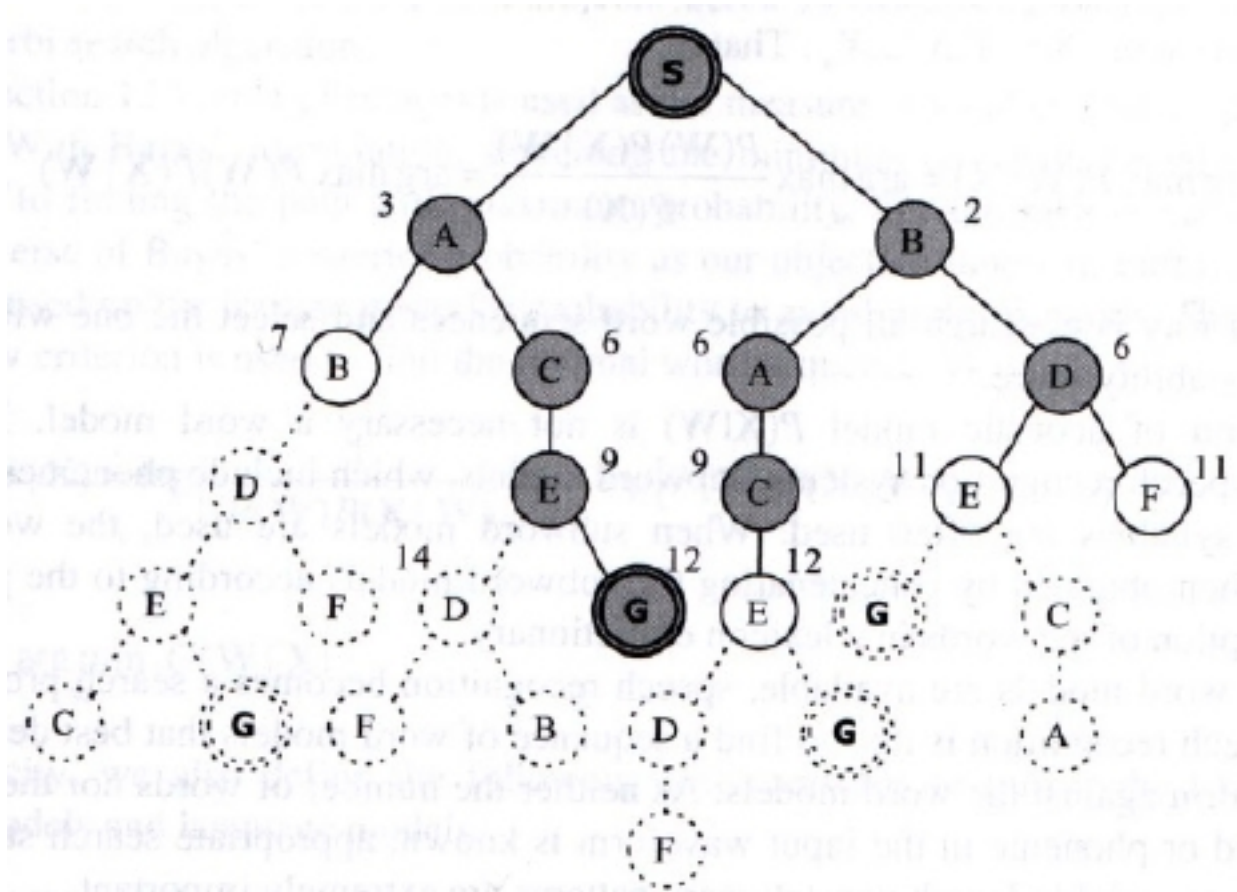
Step 8: Go to Step 2.

- A search algorithm is said to be *admissible* if it can guarantee an optimal solution.
- One possible solution to the traveling salesman problem using best-first search might look like this:



HEURISTIC SEARCH: BEAM SEARCH

- Beam search is another form of heuristic search in which we terminate hypotheses that do not appear to be promising:



- A general algorithm for searching such spaces can be defined as follows:

ALGORITHM 12.5: THE BEAM SEARCH ALGORITHM

Step 1: Initialization: Put S in the *OPEN* list and create an initially empty *CLOSE* list.

Step 2: If the *OPEN* list is empty, exit and declare failure.

Step 3: $\forall N \in OPEN$ do

3a. Pop up node N in the *OPEN* list, remove it from the *OPEN* list and put it into the *CLOSE* list.

3b. If node N is a goal node, exit successfully with the solution obtained by tracing back the path along the pointers from N to S .

3c. Expand node N by applying a successor operator to generate the successor set $SS(N)$ of node N . Be sure to eliminate the successors, which are ancestors of N , from $SS(N)$.

3d. $\forall v \in SS(N)$ Create a pointer pointing to N and push it into *Beam-Candidate* list.

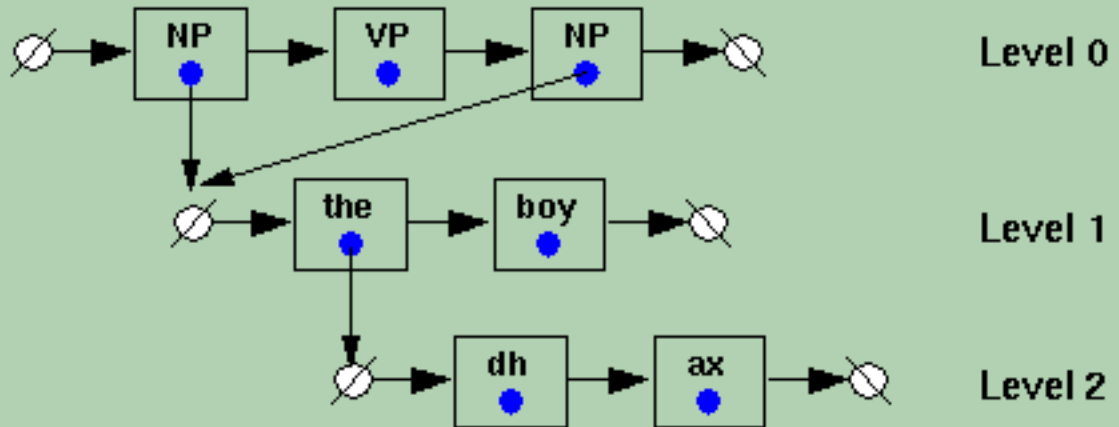
Step 4: Sort the *Beam-Candidate* list according to the heuristic function $f(N)$ so that the best w nodes can be pushed into the *OPEN* list. Prune the rest of nodes in the *Beam-Candidate* list.

Step 5: Go to Step 2.

- Why is beam search very appropriate for speech recognition?

HIERARCHICAL SEARCH: OPTIMAL SEARCH IS SUBTLE

Search Space Specification: DiGraph<SearchNode>



Search Level Specifications: Vector<SearchLevel>

Level 0: Symbols = "NP", "VP"

beam prune = true

beam width = 1000

Level 1: Symbols = "the", "boy", "ran", ...

use Nsymbol probabilities

Nsymbol length = 3

Level 2: Symbols = "dh", "ax", "b", "oy", ...

use context dependency

context = 1 on left, 1 on right

Current Search Paths: Vector<DoubleLinkedList<Trace> >

| | |
|----------------|------------------------------|
| Level 0 Traces | Trace (NP, VP) |
| Level 1 Traces | Trace (NP, the, dh, ax, boy) |
| Level 2 Traces | Trace (NP, the, dh) |

- To maintain optimality in the search, we must maintain a history of predecessor words *and* states, since the same word sequence can be produced by multiple paths in the network.
- Dynamic expansion of context is generally preferred over precompilation. Why?

[Return to Main](#)[Objectives](#)**Review:**[Motivation](#)**Decoder Basics:**[Combining Scores](#)[Continuous Speech](#)**N-grams:**[Unigram](#)[Bigram](#)[Trigram](#)**Time Synchronous:**[Level Building](#)[Viterbi Beam](#)**On-Line Resources:**[Tutorial](#)[Hierarchical](#)[AJR: Search](#)[SR: Search](#)

LECTURE 35: TIME SYNCHRONOUS SEARCH

- Objectives:
 - No endpointing!
 - N-gram-specific search
 - Time synchronous search
 - Time synchronous Viterbi beam search

This lecture follows the course textbook closely:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory,*

Algorithm, and System Development, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

Another good source for some of this information is:

F. Jelinek, *Statistical Methods for Speech Recognition*, MIT Press, Boston, Massachusetts, USA, ISBN: 0-262-10066-5, 1998.

LECTURE 35: TIME SYNCHRONOUS SEARCH

- Objectives:
 - No endpointing!
 - N-gram-specific search
 - Time synchronous search
 - Time synchronous Viterbi beam search

This lecture follows the course textbook closely:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA,

ISBN: 0-13-022616-5, 2001.

Another good source for some of this information is:

F. Jelinek, *Statistical Methods for Speech Recognition*, MIT Press, Boston, Massachusetts, USA, ISBN: 0-262-10066-5, 1998.

COMBINING SCORES IN THE LOG PROBABILITY SPACE

- Recall our basic equation defining the search problem:

$$\hat{W} = \underset{W}{\operatorname{argmax}} [P(W)P(A|W)]$$

- It is convenient to process probabilities in the log domain:

$$\begin{aligned} C(W|A) &\approx -\log[P(W)P(A|W)] \\ &= -(\log[P(W)] + \log[P(A|W)]) \\ \hat{W} &= \underset{W}{\operatorname{argmin}} [C(W|A)] \end{aligned}$$

Why?

- It is also convenient to combine the language model and acoustic scores using a weighting

factor:

$$P(W) \approx P(W)^{LW} \text{IP}^{N(W)}$$

$$\log(P(W)) = LW \log(P(W)) + N(W) \log(\text{IP})$$

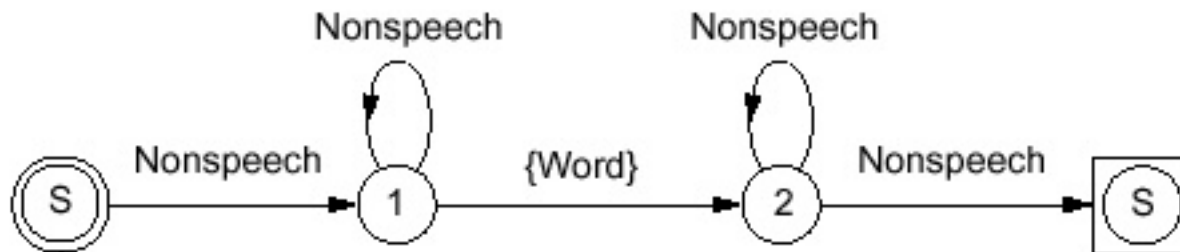
where LW is a language model weight,

$N(W)$ is the number of words in the hypothesis,

and IP is a word insertion penalty ($\text{IP} \in [0, 1]$).

ISOLATED WORD RECOGNITION USING NETWORK DECODING

Isolated Word Recognition:

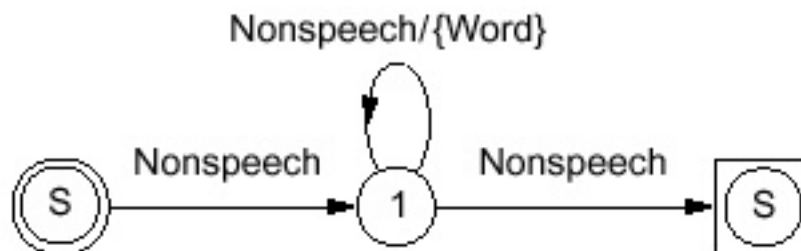


Nonspeech: typically an acoustic model of one frame in duration that models the background noise.

{Word}: any word from the set of possible words that can be spoken

- The key point here is that, with such a system, the recognizer finds the optimal start/stop times of the utterance with respect to the acoustic model inventory (a hypothesis-directed search)

Simple Continuous Speech Recognition ("No Grammar"):



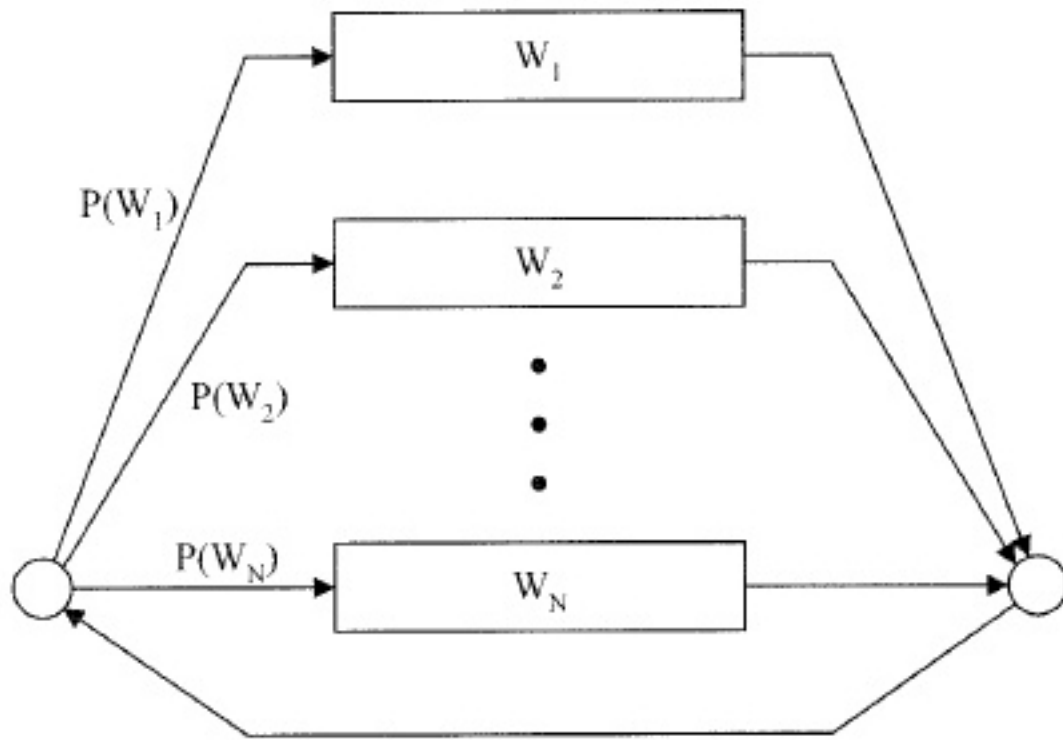
- system recognizes arbitrarily long sequences of words or nonspeech events

UNIGRAM SEARCH: SIMPLE BECAUSE IT IS MEMORYLESS

- The simplest N-gram search is the unigram search, since it is memoryless. The language model probability depends only on the current word:

$$P(W) = \prod_{i=1}^N P(w_i)$$

- The grammar network can be viewed as follows:



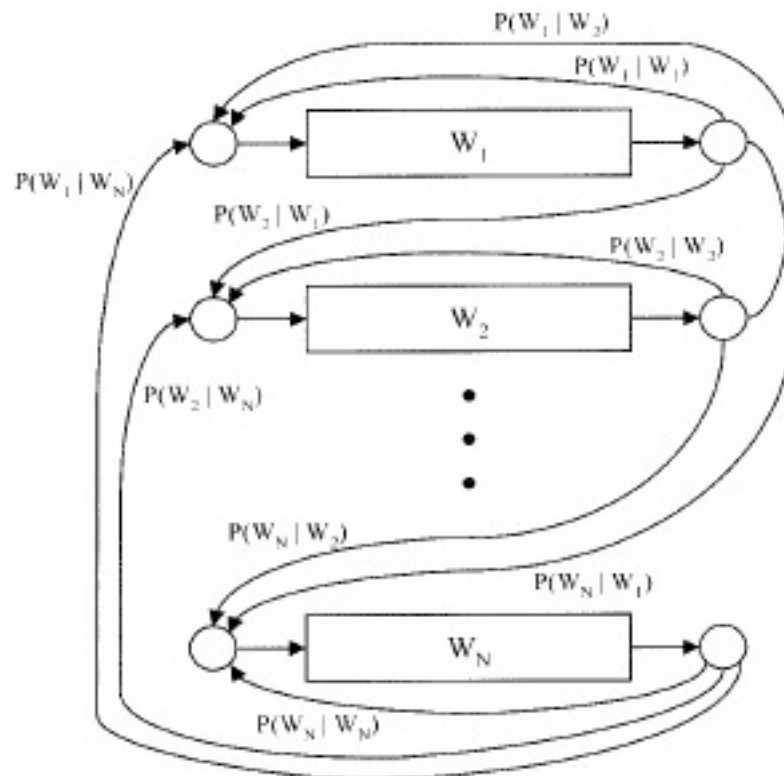
The final state of each word is connect to the collector state by a null transition. The collector state is connected to the start state with another null transition. Word expansion is trivial.

BIGRAM SEARCH: GOOD COMPROMISE BETWEEN PERFORMANCE AND COMPLEXITY

- A bigram search is still relatively simple:

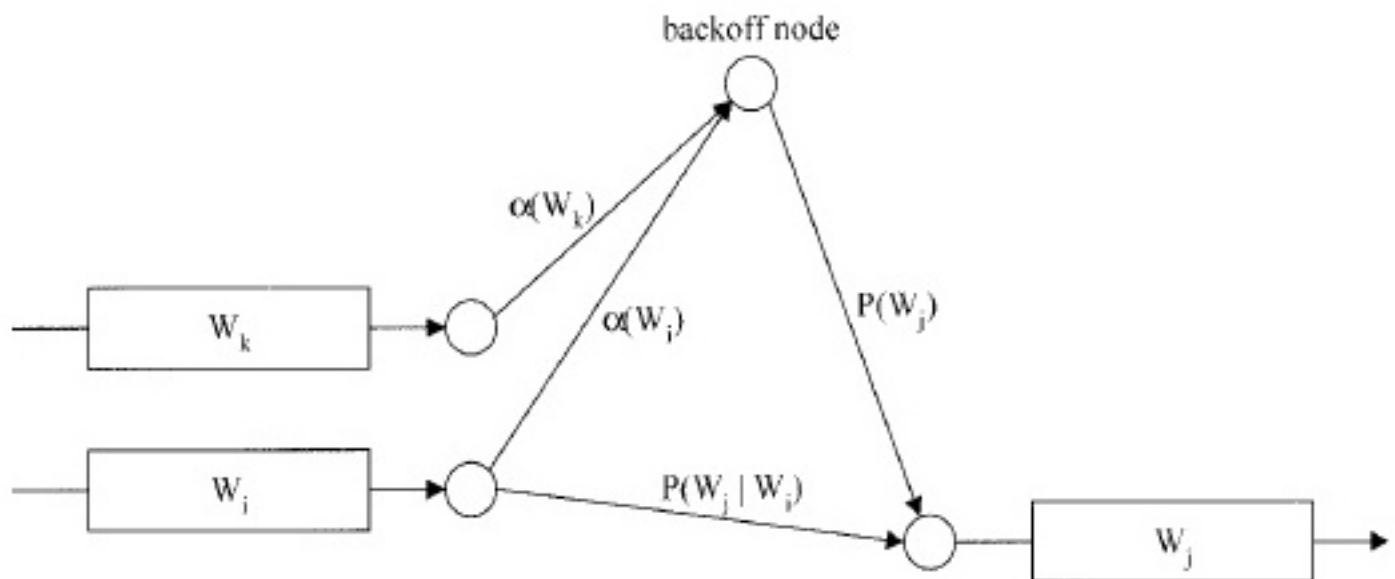
$$P(W) = P(w_1 | \langle s \rangle) \prod_{i=2}^N P(w_i | w_{i-1})$$

- A bigram search requires expand and merge:



The search complexity can be N^2 with a backoff model (if any word can follow an other word).

- We can reduce the complexity of a bigram search with backoffs by using a dynamic expansion:

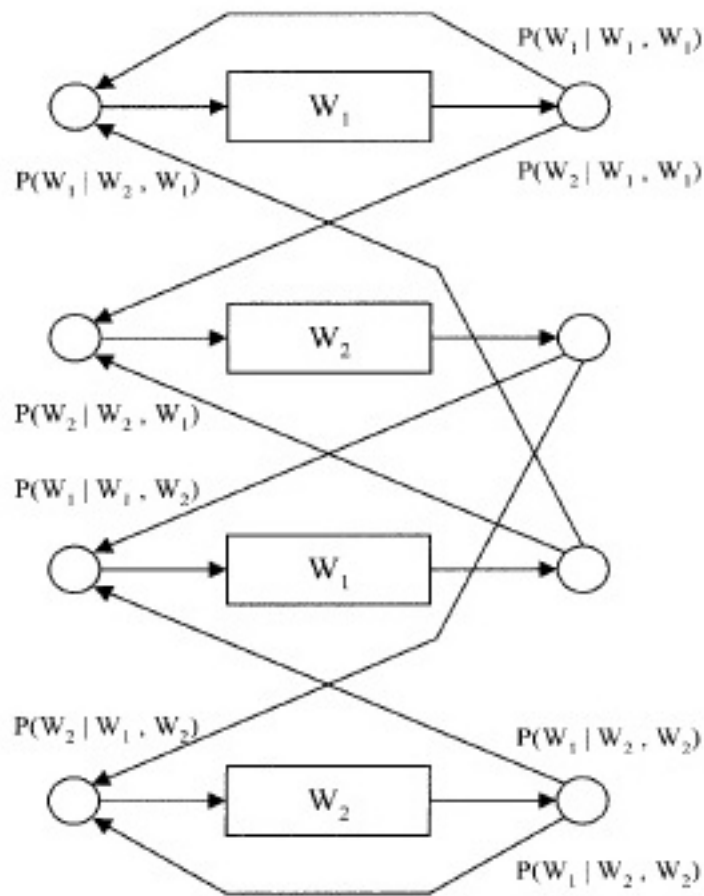


TRIGRAM SEARCH: OFTEN TOO COMPLEX FOR A FORWARD SEARCH

- A trigram search is fairly computationally intensive:

$$P(W) = (P(w_1|\langle s \rangle)P(w_2|\langle s \rangle, w_1)) \prod_{i=3}^N P(w_i|w_{i-2}, w_{i-1})$$

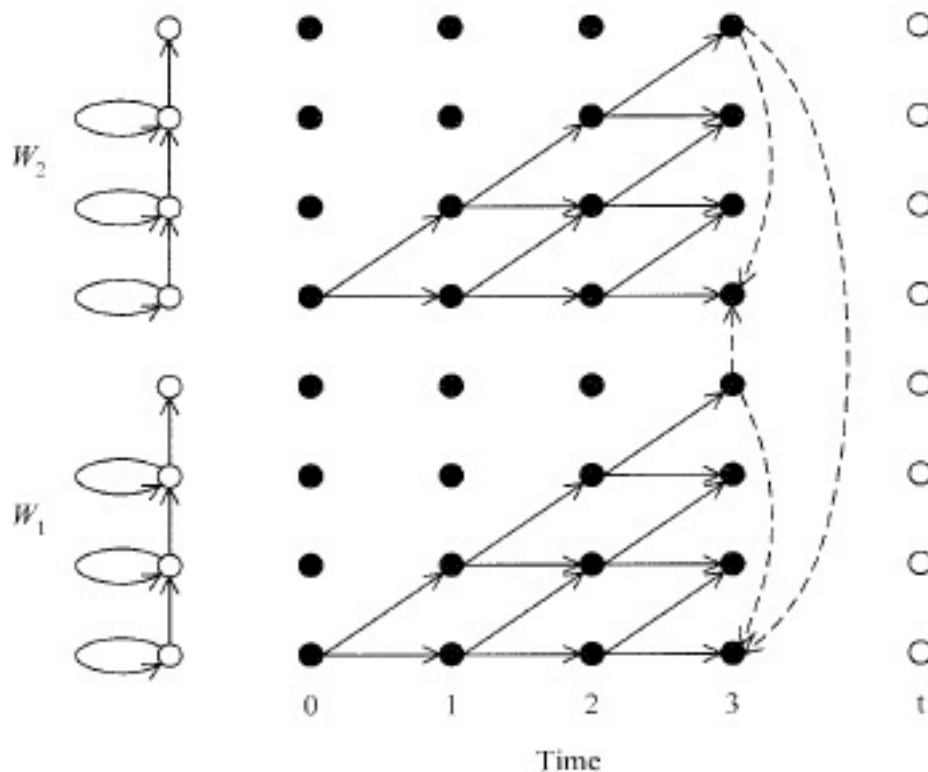
- A trigram search is shown below:



- A trigram search is often too complex for a single-pass forward search, and is instead typically implemented as a postprocessing step (rescoring) after a word graph has been generated.

TIME SYNCHRONOUS DECODING: TRELLIS EXPANSION

- Time synchronous decoding of a network can be viewed as a trellis expansion operation:

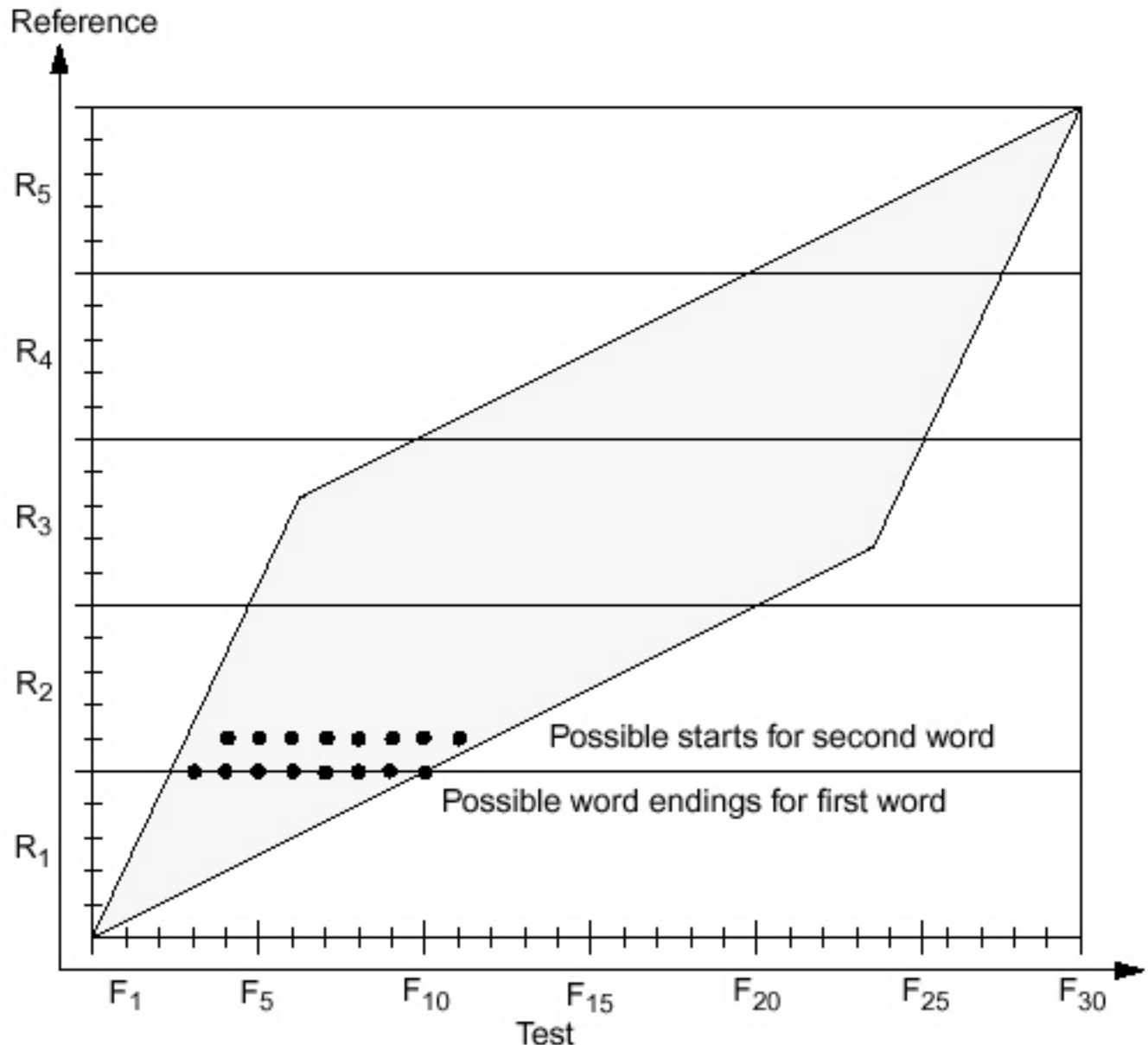


- This algorithm is based on the simple principle of dynamic programming (Sakoe and Chiba):

DTW, Syntactic Constraints, and Beam Search

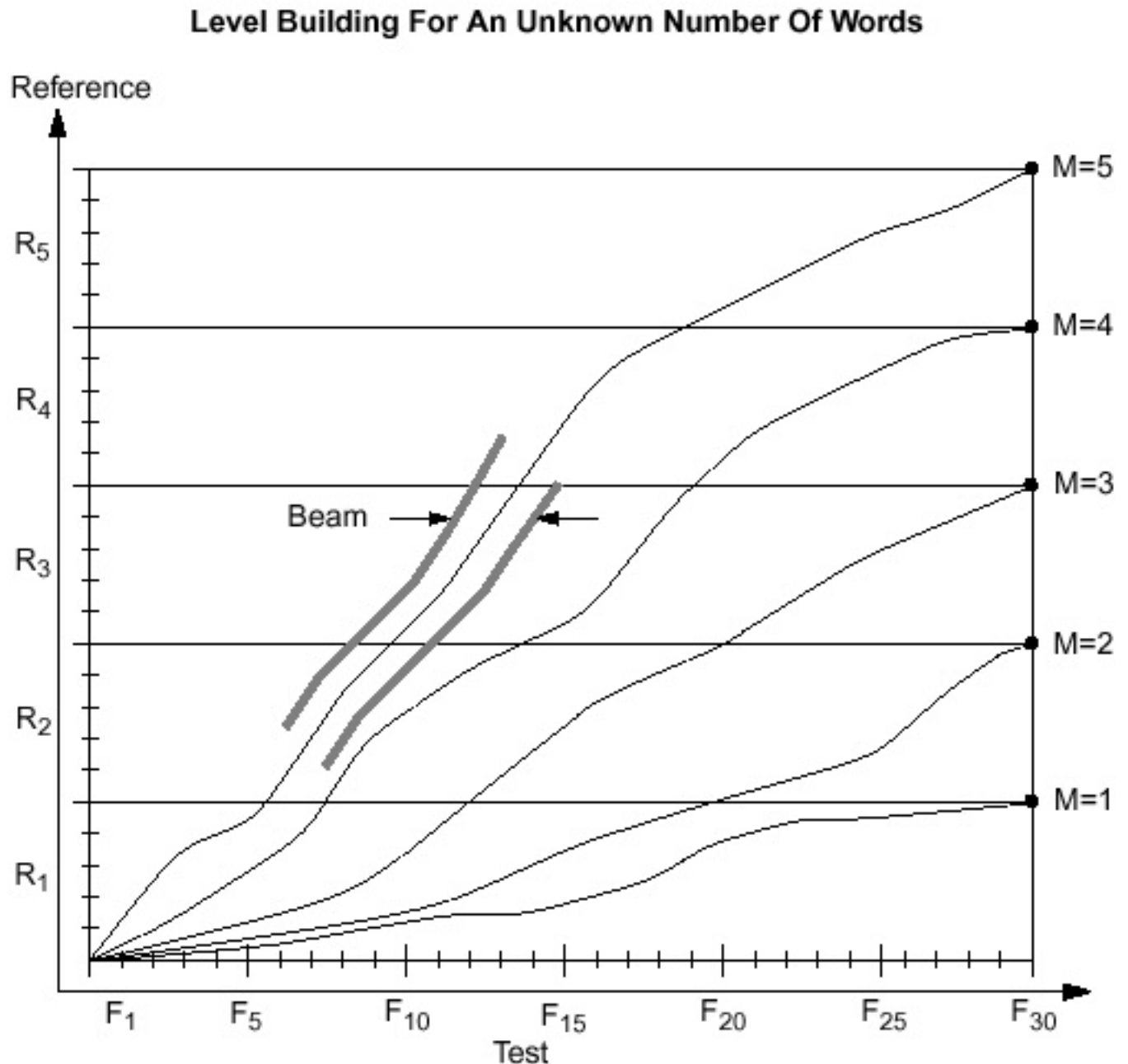
Consider the problem of connected digit recognition: "325 1739". In the simplest case, any digit can follow any other digit, but we might know the exact number of digits spoken.

An elegant solution to the problem of finding the best overall sentence hypothesis is known as level building (typically assumes models are same length).



- Though this algorithm is no longer widely used, it gives us a glimpse into the complexity of the syntactic pattern recognition problem.

- This algorithm goes by many names including level building:



- Paths can terminate on any level boundary indicating a different number of words was recognized (note the significant increase in complexity)
- A search band around the optimal path can be maintained to reduce the search space
- Next-best hypothesis can be generated (N-best)
- Heuristics can be applied to deal with free endpoints. insertion of silence

between words, etc.

- Major weakness is the assumption that all models are the same length!

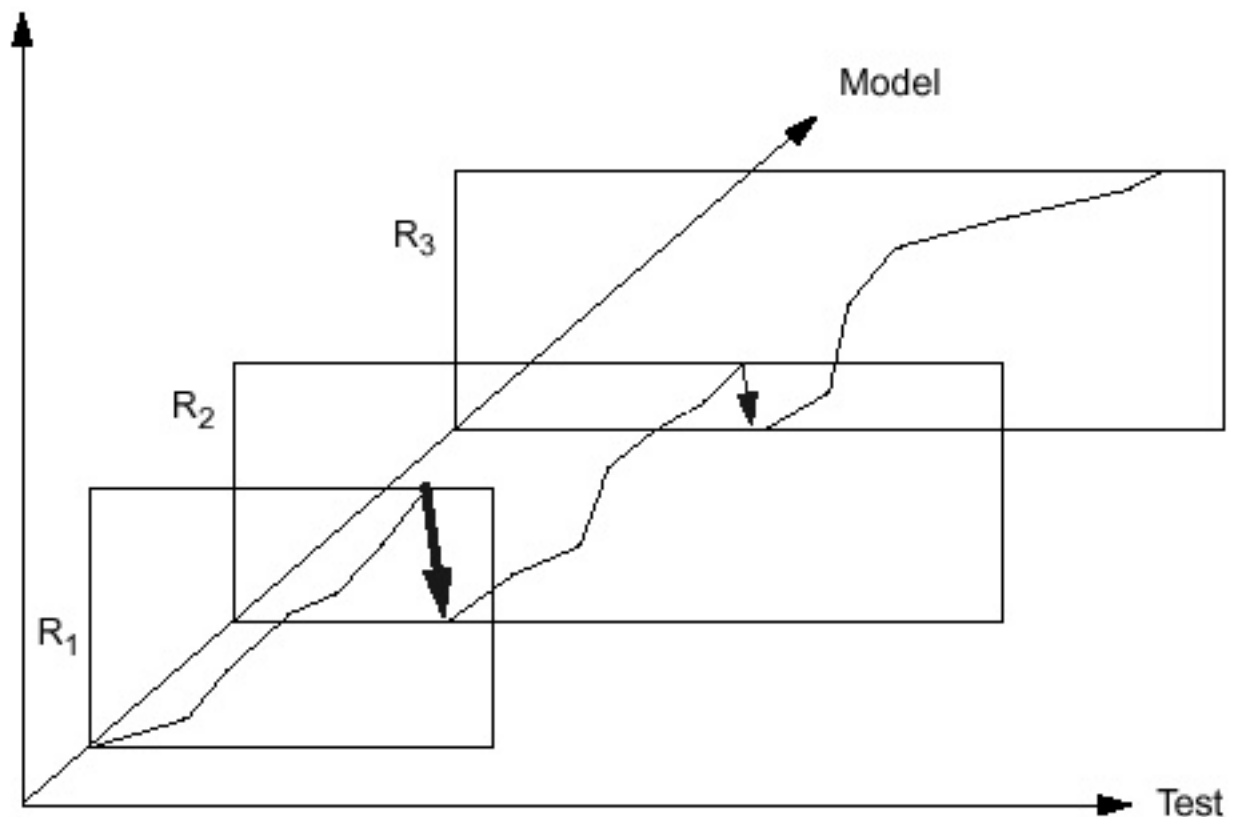
● and the Bridle algorithm (one-stage DP):

The One-Stage Algorithm ("Bridle Algorithm")

The level building approach is not conducive to models of different lengths, and does not make it easy to include syntactic constraints (which words can follow previous hypothesized words).

An elegant algorithm to perform this search in one pass is demonstrated below:

Reference



- Very close to current state-of-the-art doubly-stochastic algorithms (HMM)
- Conceptually simple, but difficult to implement because we must remember information about the interconnections of hypotheses
- Amenable to beam-search concepts and fast-match concepts

- Supports syntactic constraints by limited the choices for extending a hypothesis
- Becomes complex when extended to allow arbitrary amounts of silence between words
- How do we train?

It was first introduced for dynamic time-warping (DTW) systems.

TIME SYNCHRONOUS VITERBI BEAM SEARCH

- We can define many of these search concepts into a single algorithm: expansion operation:

ALGORITHM 12.6: TIME-SYNCHRONOUS VITERBI BEAM SEARCH

Step 1: Initialization: For all the grammar word states w which can start a sentence,
 $D(0; I(w); w) = 0$
 $h(0; I(w); w) = null$

Step 2: Induction: For time $t = 1$ to T do
 For all active states do
 Intra-word transitions according to Eq. (12.17) and (12.18)

$$D(t; s_t; w) = \min_{s_{t-1}} \{d(\mathbf{x}_t, s_t | s_{t-1}; w) + D(t-1; s_{t-1}; w)\}$$

$$h(t; s_t; w) = h(t-1, b_{\min}(t; s_t; w); w)$$

 For all active word-final states do
 Inter-word transitions according to Eq. (12.21), (12.22) and (12.23)

$$D(t; \eta; w) = \min_v \{\log P(w | v) + D(t; F(v); v)\}$$

$$h(t; \eta; w) = \langle v_{\min}, t \rangle :: h(t, F(v_{\min}); v_{\min})$$

 if $D(t; \eta; w) < D(t; I(w); w)$

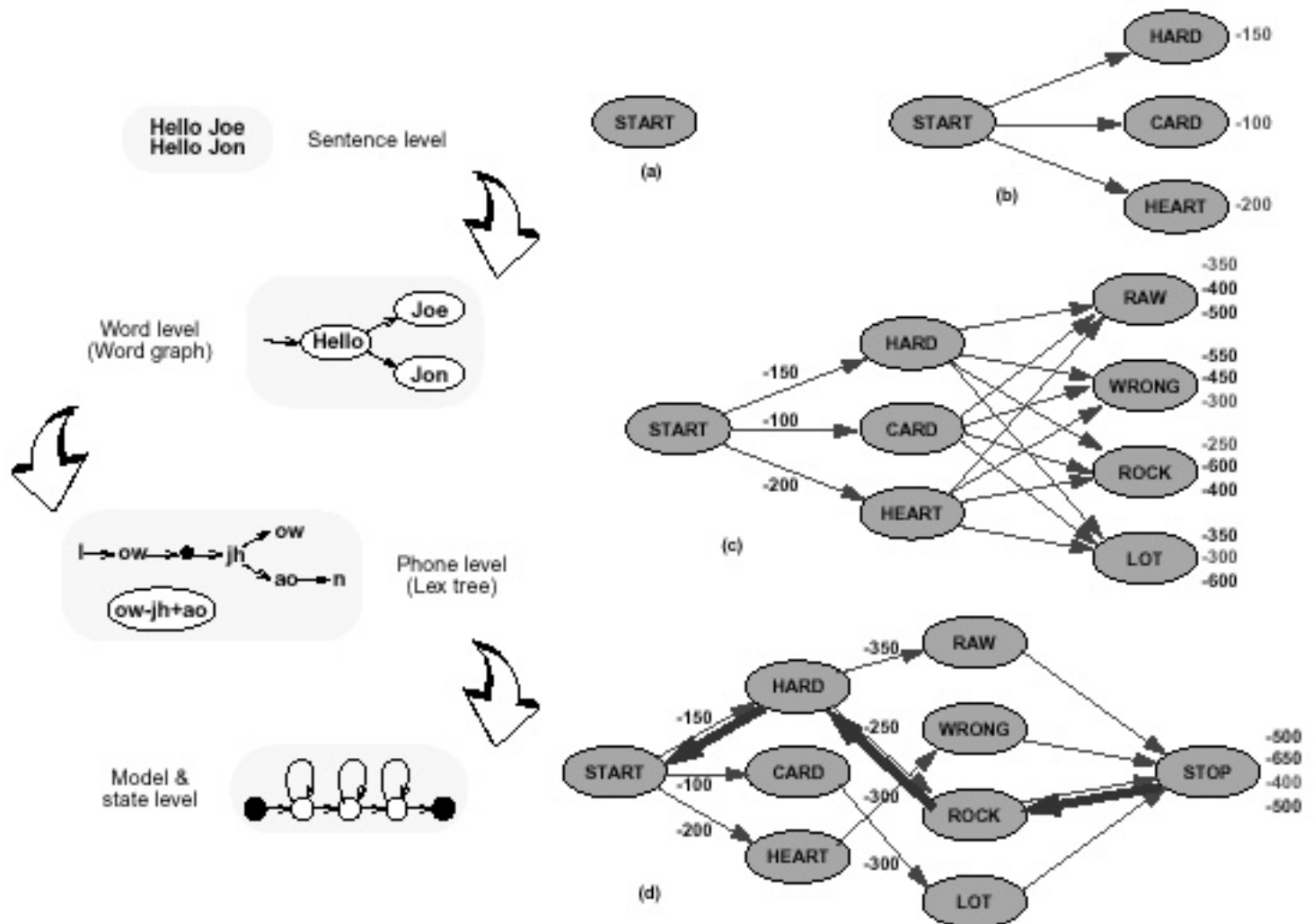
$$D(t; I(w); w) = D(t; \eta; w) \text{ and } h(t; I(w); w) = h(t; \eta; w)$$

 Pruning: Find the cost for the best path and decide the beam threshold
 Prune unpromising hypotheses

Step 3: Termination: Pick the best path among all the possible final states of grammar at time T . Obtain the optimal word sequence according to the backtracking pointer $h(t; \eta; w)$

- This algorithm uses dynamic expansion of the

network to minimize memory requirements:



We will have more to say about this algorithm later.

[Return to Main](#)[Objectives](#)**Stack Decoding:**[Best-First](#)[Admissible Heuristics](#)[Fast Match](#)**Multi-pass Search:**[N-Best Generation](#)[Lattice Generation](#)**Other:**[Course Evaluations](#)**On-Line Resources:**[JA: Stack](#)[AJR: Search](#)[SR: Search](#)

LECTURE 36: STACK DECODING

- Objectives:
 - Best-first search with admissible heuristics
 - Fast matching
 - Cross-word decoding and lexical trees
 - N-best and word graph generation

This lecture follows the course textbook closely:

X. Huang, A. Acero, and H.W.

Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

Another good source for some of this information is:

F. Jelinek, *Statistical Methods for Speech Recognition*, MIT Press, Boston, Massachusetts, USA, ISBN: 0-262-10066-5, 1998.

LECTURE 36: STACK DECODING

- Objectives:
 - Best-first search with admissible heuristics
 - Fast matching
 - Cross-word decoding and lexical trees
 - N-best and word graph generation

This lecture follows the course textbook closely:

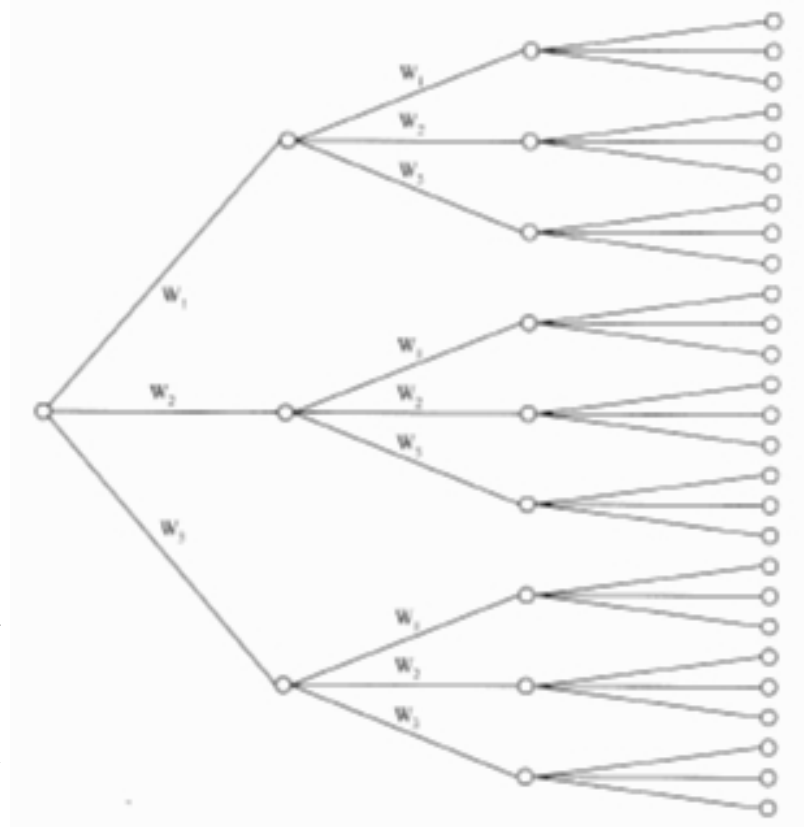
X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

Another good source for some of this information is:

F. Jelinek, *Statistical Methods for Speech Recognition*, MIT Press, Boston, Massachusetts, USA, ISBN: 0-262-10066-5, 1998.

STACK DECODING (A^* SEARCH)

- If some heuristics are available to guide the decoding, the search can be done in a depth-first fashion around the best path.



- We can avoid wasting computation on unpromising paths via time synchronous decoding.
- Such a heuristic function is

very difficult
to attain in
speech
recognition
since it must
combine
elements of
acoustic and
language
model
scoring.

● **Stack
decoding is**

a variant of tree search.

- Note that the Viterbi search finds the optimal state sequence while stack decoding focuses on the optimal word sequence.
- The search process can be summarized as follows:
 - Add all possible one-word sequences to the OPEN list.
 - Remove the best path from the OPEN list; all paths from it are extended, evaluated, and placed back in the OPEN list (sorted).
 - Continue until a complete path that is guaranteed to be better than any path on the

OPEN list has been found.

- Two key operations:
 - Finding an effective heuristic function for estimating the probability of the "future" part of the path.
 - Determining when to extend the search to the next word/phone.

ADMISSIBLE HEURISTICS

- Recall the general form of our evaluation function:

$$f(H) = g(H) + h(H)$$

Where $g()$ represents the evaluation function for the partial path up to time t , and $h()$ represents the estimate of the remaining path.

- An admissible heuristic function is one that always underestimates the true cost of the remaining path (e.g., a zero function).
- The evaluation function can simply be the forward probability.

- The expected cost of the remaining part of the path can be estimated by gathering statistics from the training data:

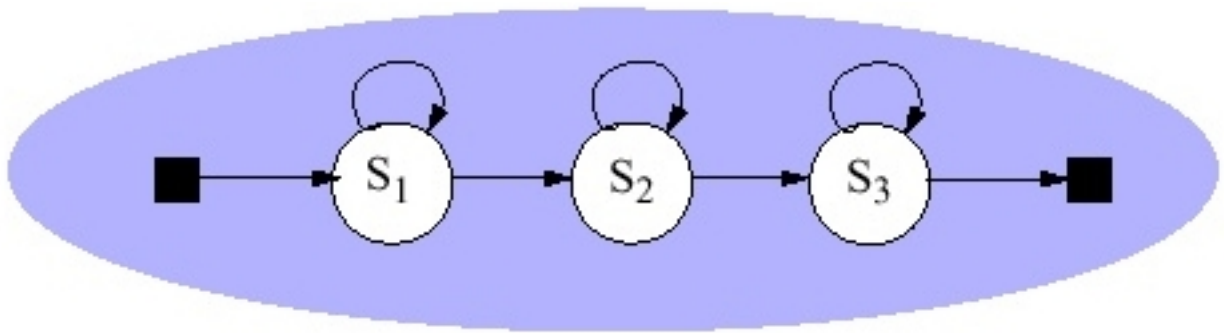
$$h(H) = (T-t)P_{\min}$$

- It can be shown that this same heuristic can be applied to the problem of extending the path into the next word.

FAST MATCH IS CRITICAL IN STACK DECODING

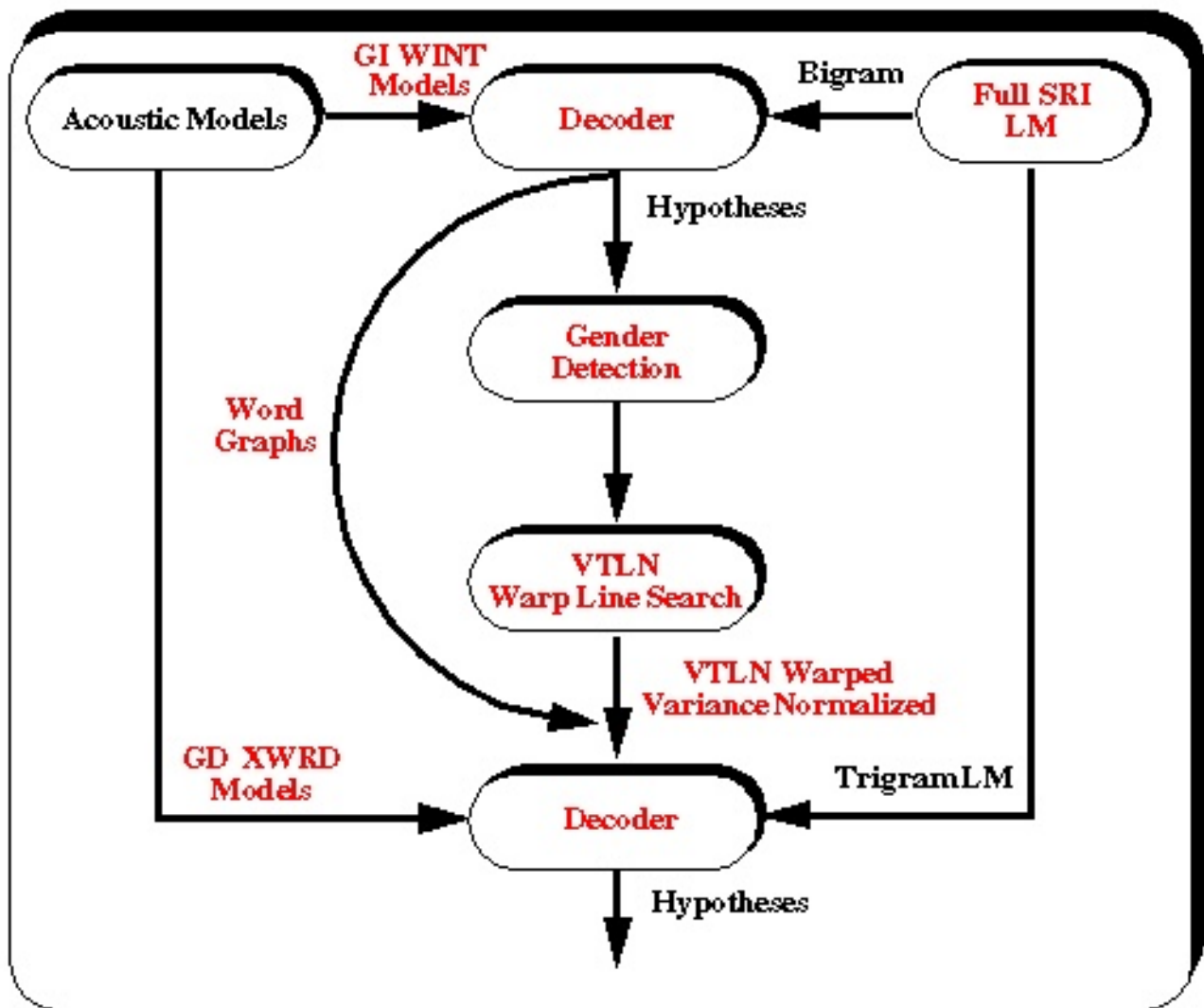
- An effective underestimated heuristic function for the remaining portion of speech is very difficult to derive.
- In asynchronous stack decoding, the most expensive step is to extend the best subpath.
- **Fast match** is a method for the rapid computation of a list of candidates that constrain successive search phases.
- Fast match can be regarded as an additional pruning threshold to meet before a new word or phone can be started.

- A fast match method is called admissible if it never prunes away the optimal path.
- One popular fast match approach is to estimate the probability a phone model by using the "straight-thru" path:



N-BEST DECODING IS USEFUL FOR INTEGRATING KNOWLEDGE SOURCES

- Often the search space becomes unmanageable for real problems due to complex language model constraints (e.g., trigrams) and acoustic models (e.g., cross-word context-dependent phones), and our interest in integrating multiple knowledge sources (e.g., phrase structured grammars).
- A pragmatic alternative is to use a multipass search approach:



- It is common to perform a first pass of decoding with a bigram language model (or a word-internal triphone/trigram system), and postprocess (or rescore) the output with a more sophisticated system. We refer to this process as **multipass** decoding.

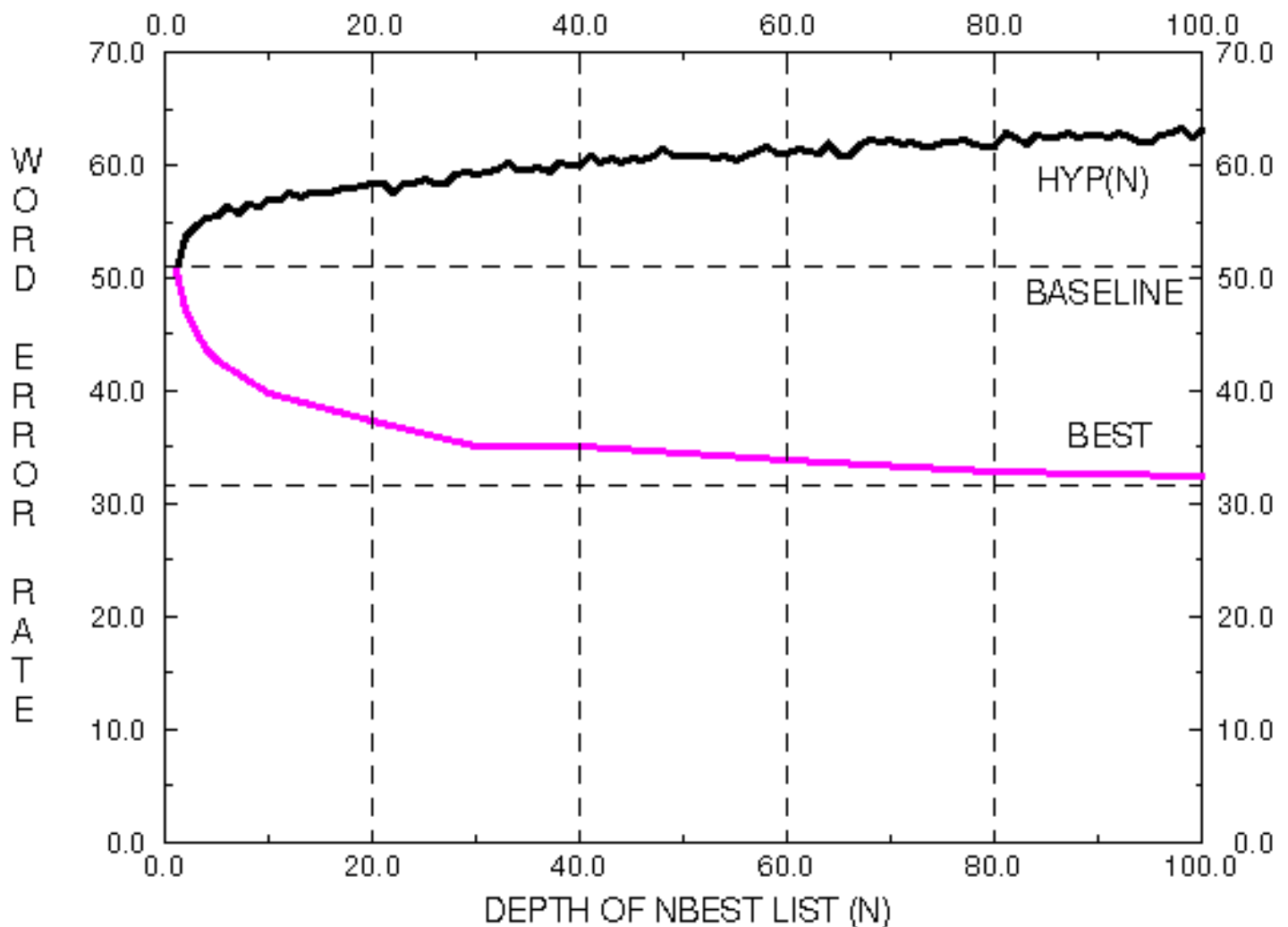
- Stack decoding is naturally suited to generating N-best lists. Consider this example from the North American Business (NAB) Corpus:

| | |
|----|---|
| 1. | I will tell you would I think in my office |
| 2. | I will tell you what I think in my office |
| 3. | I will tell you when I think in my office |
| 4. | I would sell you would I think in my office |
| 5. | I would sell you what I think in my office |

| | |
|-----|--|
| 6. | I would sell you when I think in my office |
| 7. | I will tell you that I think in my office |
| 8. | I will tell you why I think in my office |
| 9. | I will tell you would I think on my office |
| 10. | I Wilson you think on my office |

- N-best lists are very compact representations of the search space since timing information is discarded.

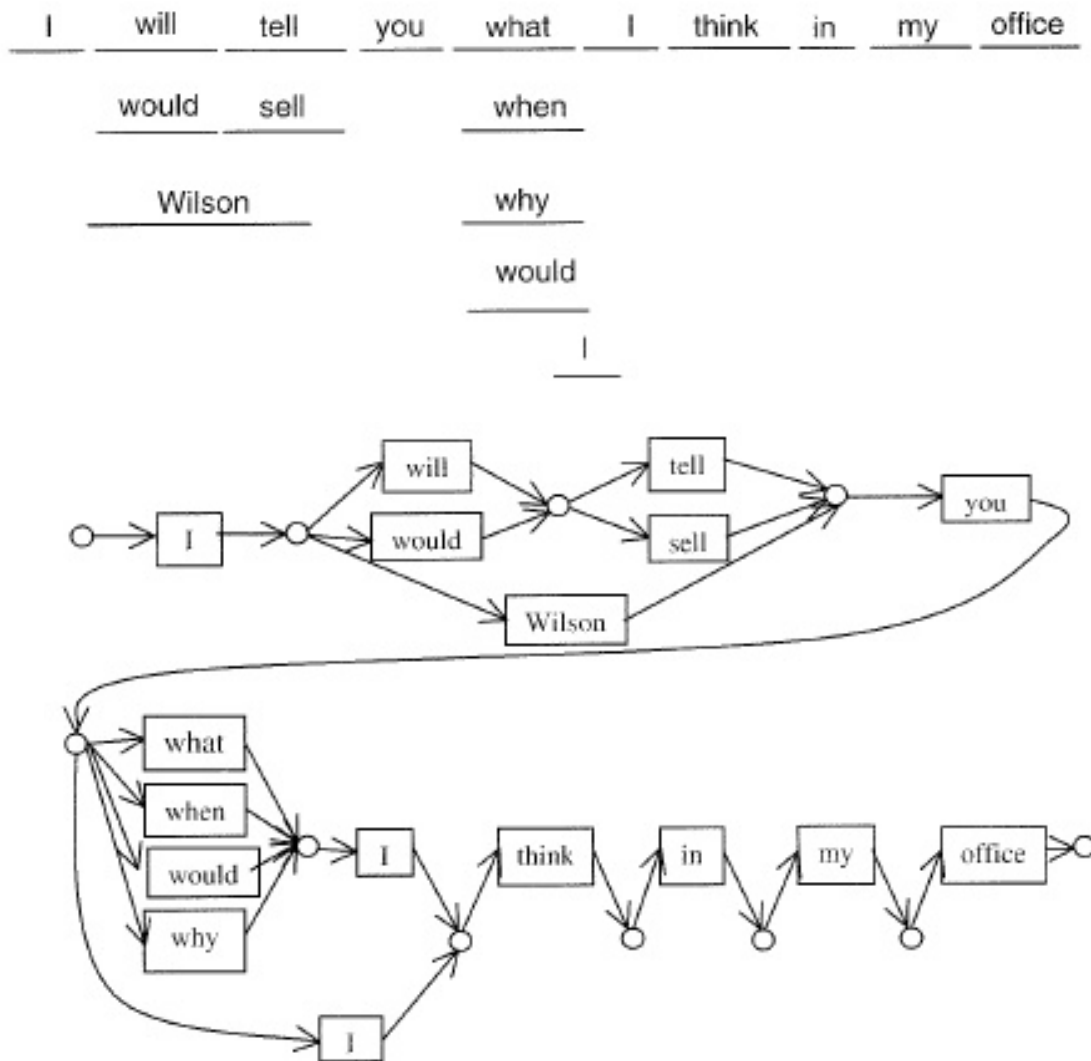
- One popular rescoring experiment that can be performed with N-best lists is referred to as an *oracle* experiment: How often does the correct hypothesis appear, and at what depth in the list does it appear?



- Oracle experiments are one form of a *cheating* experiment. Cheating experiments are very important diagnostic tools.

WORD GRAPH GENERATION AND RESCORING

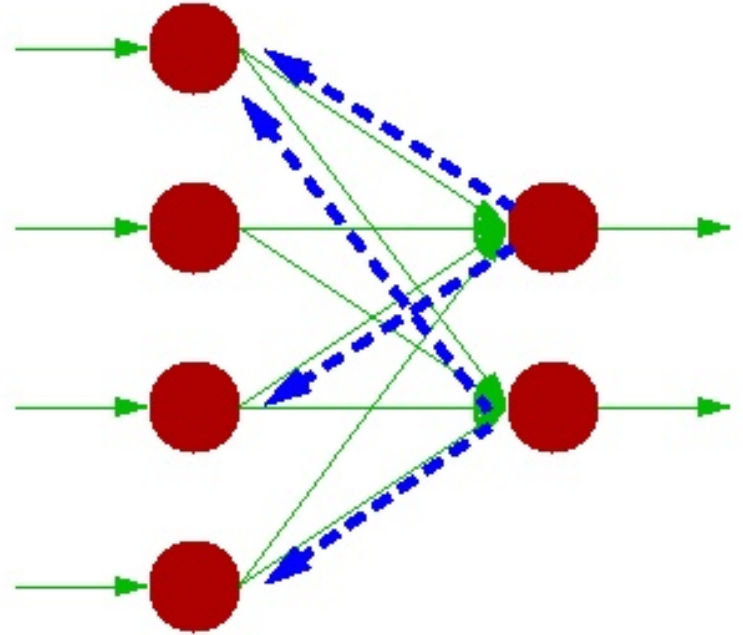
- An N-best list can be viewed as a graph:



This representation is often referred to as a **lattice**.

- How might we generate such a graph using time synchronous Viterbi decoding?

- Solution:
keep multiple
choices at
each node in
the graph
during the
dynamic
programming
step:



- It is hard to

underestimate
the impact
word graph
rescoring has
had on speech
recognition
research.

However.
these graphs
are very large
and take at
least an order
of magnitude
more time to

generate (than the one-best choice). Why are these worth the trouble?

- Word graphs can be very large: 10 to 50 MBytes per file; 1 Gbyte or more per corpus.

- What figure of merit can we use to describe such graphs? (Hint: lattice word error rate) Explain the significance of this measure.

[Return to Main](#)[Objectives](#)**Review:**[Typical System](#)**Efficient Search:**[Complexity](#)[Lexical Trees](#)[Multiple Trees](#)[Factored Probabilities](#)**Examples:**[Search Demonstration](#)**On-Line Resources:**[Ney: LM Lookahead](#)[OGI: Lexical Trees](#)[SRSDR'02: Decoding](#)

LECTURE 37: LEXICAL TREES

- Objectives:
 - A typical speech recognition architecture
 - Complexity of N-gram decoding
 - Definition of a lexical tree
 - Basic computational issues

This lecture follows the course textbook closely:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory,*

Algorithm, and System Development, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

Another good source for some of this information is:

N. Deshmukh, A. Ganapathiraju and J. Picone, "Hierarchical Search for Large Vocabulary Conversational Speech Recognition," *IEEE Signal Processing Magazine*, vol. 16, no. 5, pp. 84-107, September 1999.

LECTURE 37: LEXICAL TREES

- Objectives:
 - A typical speech recognition architecture
 - Complexity of N-gram decoding
 - Definition of a lexical tree
 - Basic computational issues

This lecture follows the course textbook closely:

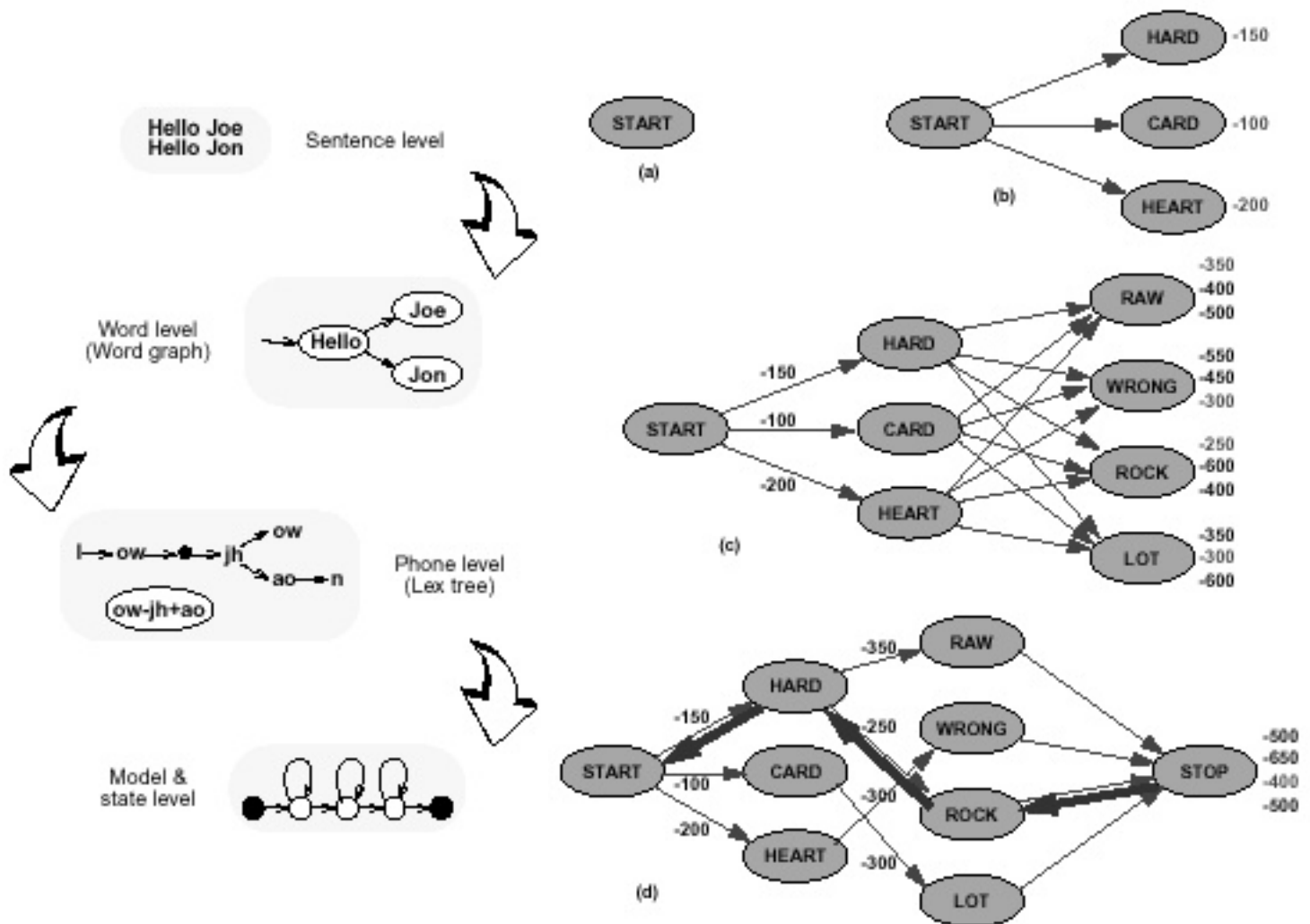
X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

Another good source for some of this information is:

N. Deshmukh, A. Ganapathiraju and J. Picone, "Hierarchical Search for Large Vocabulary Conversational Speech Recognition," *IEEE Signal Processing Magazine*, vol. 16, no. 5, pp. 84-107, September 1999.

A TYPICAL SPEECH RECOGNITION SYSTEM

- A typical system decomposes sentences into words, words into phone sequences, and phones into HMM models:



COMPLEXITY OF N-GRAM SEARCH

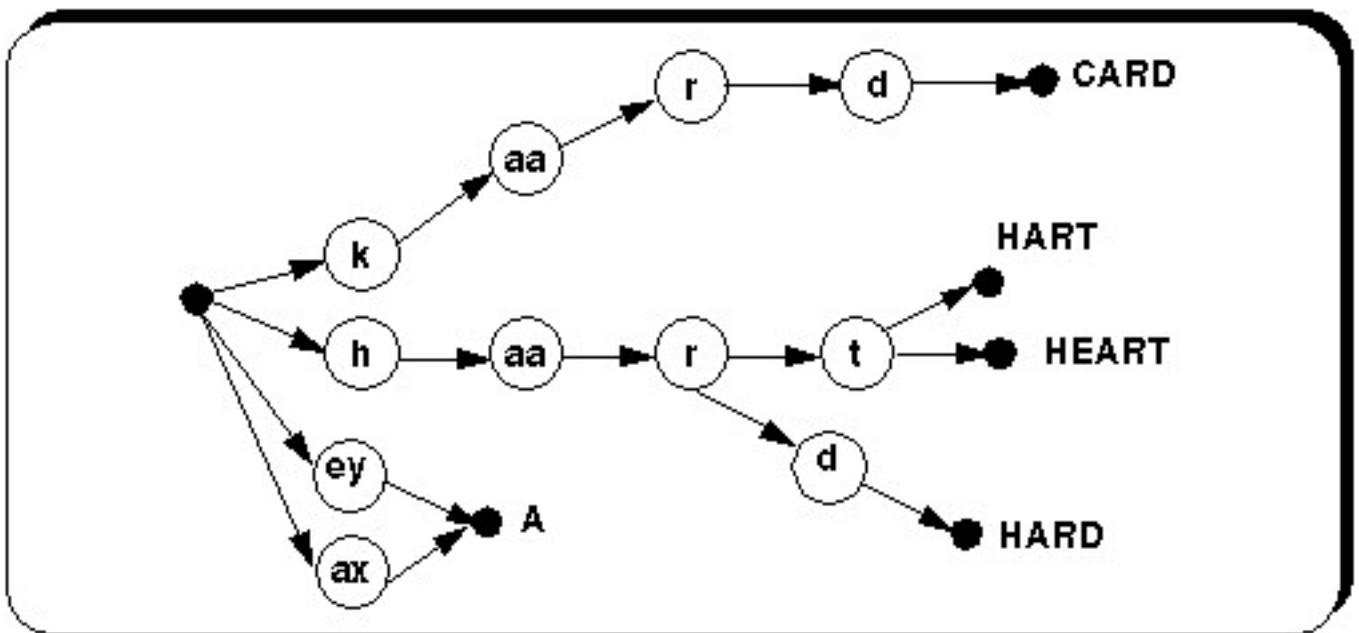
- Although it is always desirable to use as many knowledge sources as possible, there are practical problems integrating such information into a time-synchronous search.
- One alternate strategy is to use a mult-pass search. However, the more accurate the first-pass, the better the performance on subsequent passes.
- One of the most critical parts of search is the **tree lexicon**.
- In a **linear lexicon**, each word is represented as a linear sequence of phonemes independent of other words. For example, though *task* and *tasks* share the same root, we do not share any

of their history during the search process.

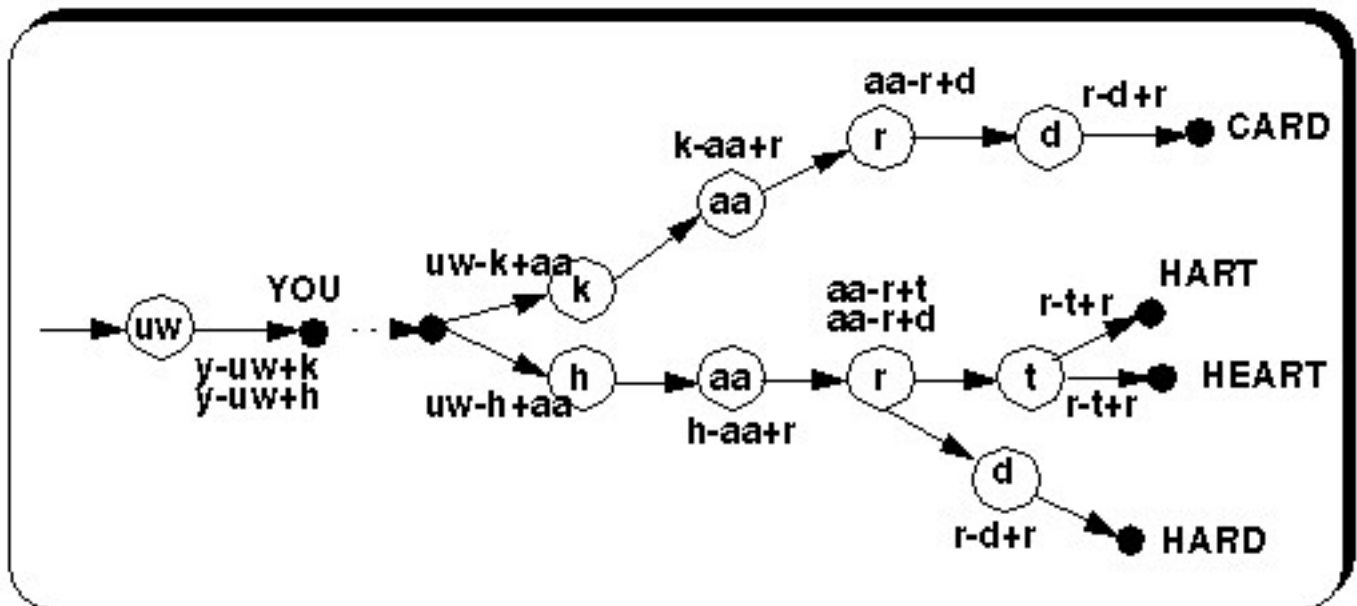
- Large lexicons introduce enormous complexity for a backoff N-gram language model because we must "start" all words in the lexicon for every unique history in our current search space.
- Can we share some of the underlying phonetic structure of words in the lexicon?

LEXICAL TREES

- A **lexical tree** is a data structure that allows us to share histories between words in the lexicon:



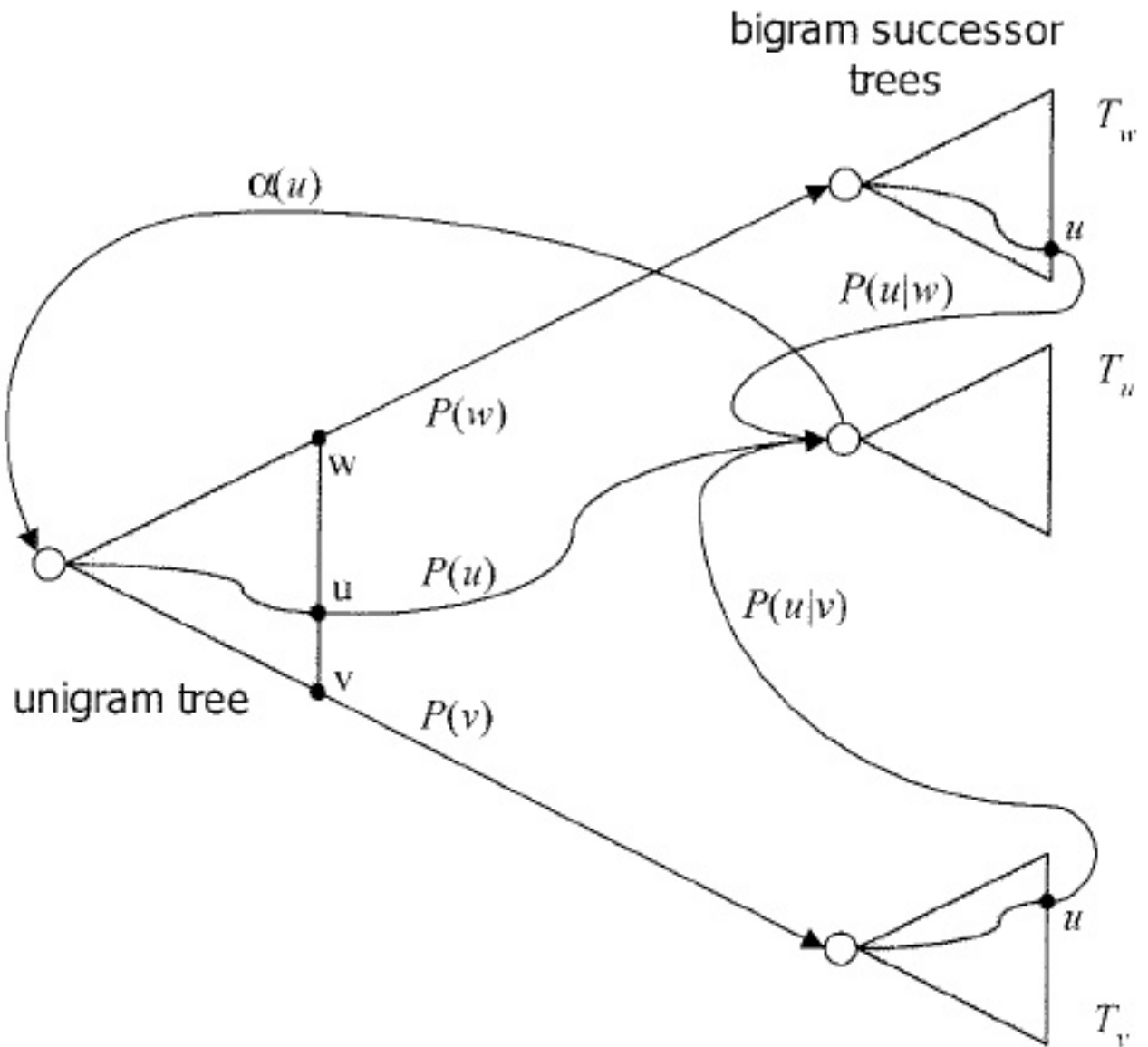
- Most systems use some form of dynamic context expansion:



MULTIPLE COPIES OF PRONUNCIATION TREES

- A simple lexical tree is sufficient if no language model is used.
- For higher-order N-gram models, the linguistic state cannot be determined locally. History plays an important role. For example, for bigrams, a tree copy is required for each predecessor word.
- Efficient pruning can eliminate most of the unnecessary tree copies.
- To deal with tree copies, you can create redundant trees for each word context. However, this is expensive.

- Many of the active state hypotheses correspond to the same redundant unigram state because the language model probability cannot be applied until the next word has been observed.
- A successor tree approach can be used to eliminate this redundancy:



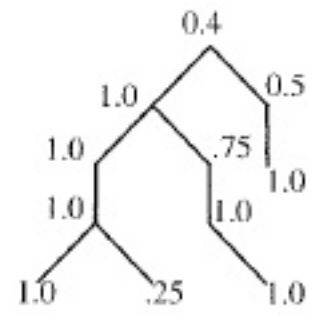
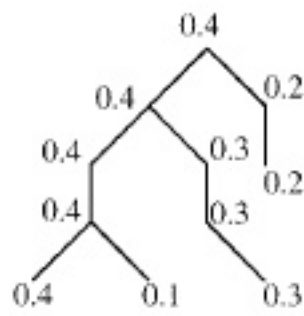
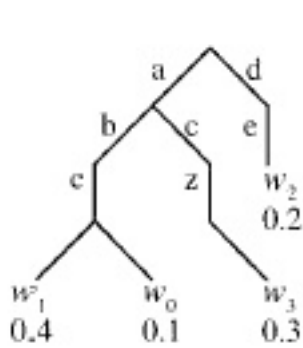
- However, for backoff language models, this isn't as efficient as it might seem (unless aggressive pruning is used to allow only a small subset of words to follow a given word).

FACTORED LANGUAGE MODEL PROBABILITIES

- Premise: We can improve accuracy and minimize resources if we can apply the language model score as soon as possible.
- Factoring LM probabilities across a tree is one such idea:

$$P^*(n) = \max_{x \in \text{child}(n)} P(x)$$
$$F^*(n) = \frac{P^*(n)}{P^*(\text{parent}(n))}$$

- We can embed these probabilities in the pronunciation tree:



- In practice, such ideas require strong emphasis on the language model to be effective (such as WSJ or NAB). Applications such as conversational speech, in which acoustic ambiguity is extreme, and acoustic scores tend to dominate decoding, do not benefit as much from such approaches.

[Return to Main](#)[Objectives](#)**Review:**[Lexical Trees](#)[Factorization](#)**Efficient Search:**[Memory Organization](#)[Subtree Isomorphism](#)[Sharing Tails](#)[Exploiting Polymorphism](#)**Examples:**[Search Demonstration](#)**On-Line Resources:**[Odell: Context](#)[Software Tools for NLP](#)[Search Tool](#)

LECTURE 38: OPTIMIZATION OF LEXICAL TREES

● Objectives:

- Subtree Isomorphism
- Sharing Trees
- Polymorphism
- Search Demo

This lecture follows the course textbook closely:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall,

Upper Saddle River, New Jersey,
USA, ISBN: 0-13-022616-5,
2001.

Another good source for some of
this information is:

N. Deshmukh, A. Ganapathiraju
and J. Picone, "Hierarchical
Search for Large Vocabulary
Conversational Speech
Recognition," *IEEE Signal
Processing Magazine*, vol. 16, no.
5, pp. 84-107, September 1999.

LECTURE 38: OPTIMIZATION OF LEXICAL TREES

- Objectives:
 - Subtree Isomorphism
 - Sharing Trees
 - Polymorphism
 - Search Demo

This lecture follows the course textbook closely:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA,

ISBN: 0-13-022616-5, 2001.

Another good source for some of this information is:

N. Deshmukh, A. Ganapathiraju and J. Picone, "Hierarchical Search for Large Vocabulary Conversational Speech Recognition," *IEEE Signal Processing Magazine*, vol. 16, no. 5, pp. 84-107, September 1999.

MEMORY ORGANIZATION AND EFFICIENCY

- A major drawback to the use of successor trees is the large memory overhead required.
- For example, the 1994 NAB LM contains 5M bigrams and over 70M bytes to store predecessor-dependent lexical trees.
- Need efficient ways to handle the multiple copies of lexical trees.
- Factorization of the LM scores pushes the application of probabilities earlier in the tree (before we leave the leaves and transition to the next word). Hence, we open the possibility to merge duplicated trees.

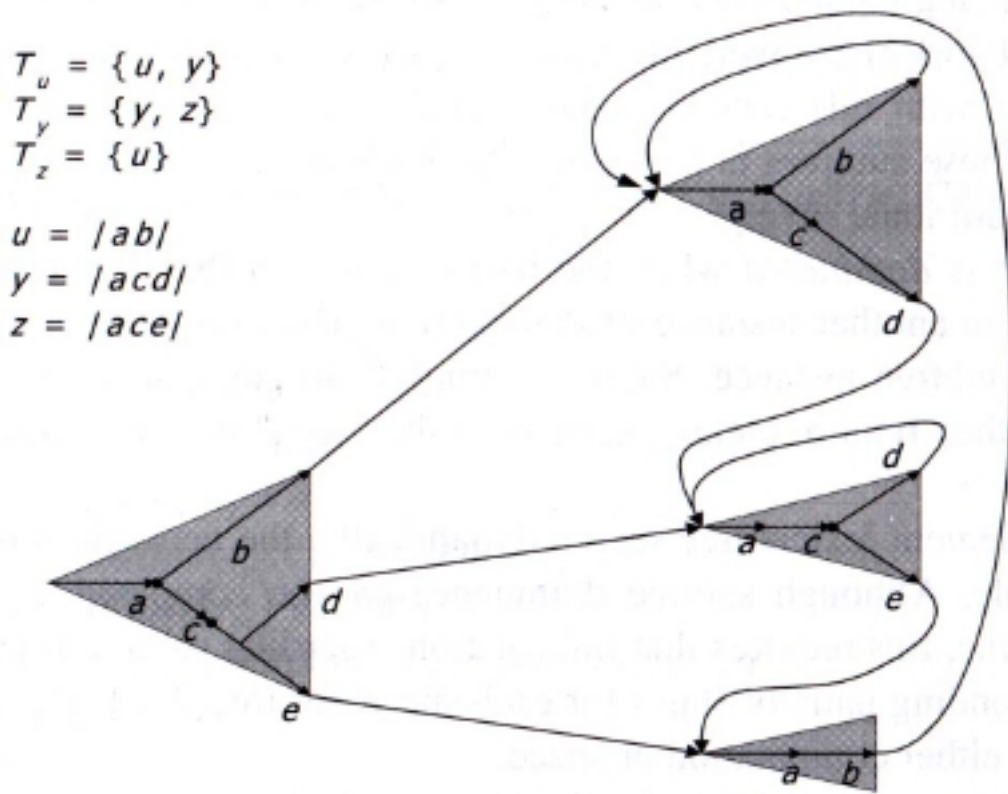
- These trees can be merged to avoid redundant state evaluation, thereby saving space and computation, with no loss of optimality.

SUBTREE ISOMORPHISM

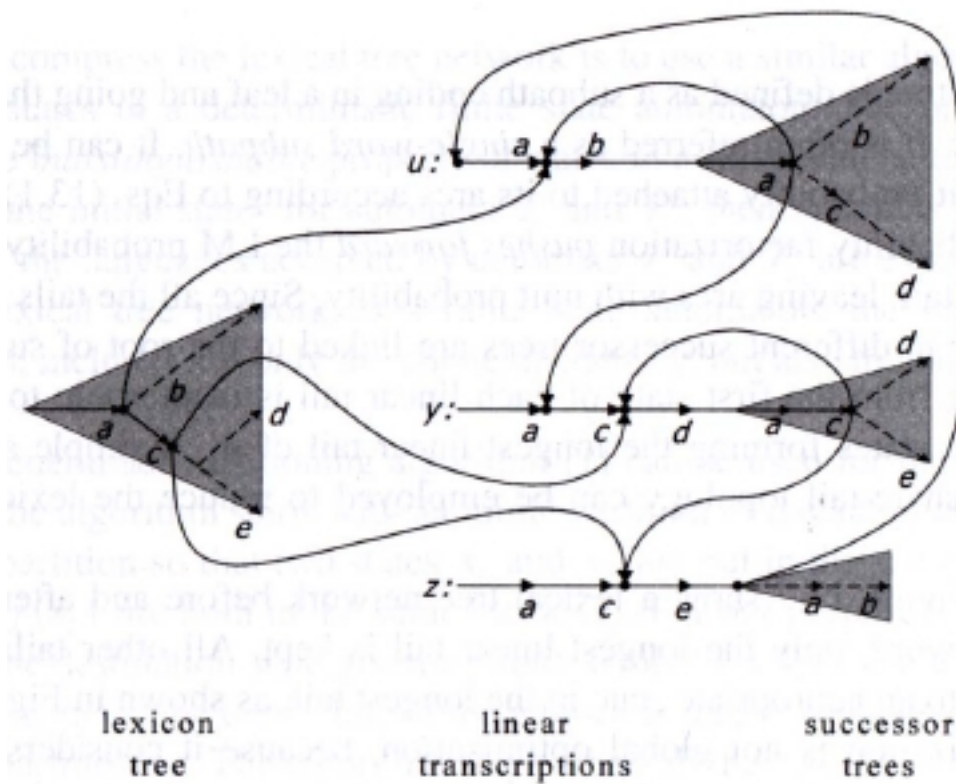
- One way we could reduce complexity is to optimize the number of states in a deterministic finite state automaton. If we exploit the tree structure of the graph, we can do more aggressive optimization.
- Two subtrees are said to be *isomorphic* to each other if they can be made equivalent by permuting the successors.
- Similarly, two states are indistinguishable if and only if their subtrees are isomorphic.
- We can merge subtrees that are isomorphic within a lexical tree. There are automated algorithms to do this.

SHARING TAILS OF TREES

- Assume a bigram language model.
- A *linear tail* in a lexical tree is defined as a subpath ending in a leaf node and going through states with a unique successor (also called a *single-word subpath*).
- LM factorization pushes forward the LM probability to the last arc of the linear tail.
- We can optimize a tree to take advantage of shared-tail optimization. Consider this tree before optimization:



and this tree after shared-tail optimization:



- What are the advantages of this approach?

EXPLOITING SUBTREE POLYMORPHISM

- The previous techniques we have discussed only eliminate identical subtrees.
- There are many subtrees that share the same nodes and topology, but have different LM scores. Can we avoid redundant acoustic model state evaluations for such trees?
- A subtree is *dominated* when the best outcome in that subtree is not better than the worst outcome in another instance of that tree. State evaluation is redundant and unnecessary for the dominated tree.
- A *polymorphic* linguistic context assignment to reduce redundancy is employed.

- Polymorphic context assignment involves keeping a single copy of the tree and allowing each state to assume the linguistic context of the most promising history (in essence, a pruning technique).
- Each node in the tree is evaluated once. However, this approach can introduce search errors.
- One approach (WHISPER) to mitigate the effects of this pruning is to keep a heap of the most promising contexts, and to delay the decision regarding which context is most promising.
- For trigram decoding, many of the approaches we have just described are still not practical.

[Return to Main](#)[Objectives](#)**Introduction:**[Motivation](#)**Approaches:**[MAP](#)[MLLR](#)[Comparison](#)**On-Line Resources:**[MAP](#)[MLLR](#)[Comparison](#)

LECTURE 39: ADAPTATION

- Objectives:

- Maximum a posteriori estimation
- Maximum likelihood linear regression
- Comparison in performance

This lecture draws on material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

and this presentation/paper:

J. Hamaker, "A Speaker Adaptation Techniques for LVCSR",
http://www.isip.msstate.edu/publications/courses/ece_7000_speech/lectures/current/lecture_10/,
ECE 7000: Special Topics in Speech Recognition, Department of Electrical and Computer Engineering, Mississippi State University, Mississippi, USA, November 1999.

LECTURE 39: ADAPTATION

- Objectives:

- Maximum a posteriori estimation
- Maximum likelihood linear regression
- Comparison in performance

This lecture draws on material from the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

and this presentation/paper:

J. Hamaker, "A Speaker Adaptation Techniques for LVCSR",
http://www.isip.msstate.edu/publications/courses/ece_7000_speech/lectures/current/lecture_10/,
ECE 7000: Special Topics in Speech Recognition, Department of Electrical and Computer Engineering, Mississippi State University, Mississippi, USA, November 1999.

ADAPTIVE TECHNIQUES - MINIMIZING MISMATCH

- We can improve recognition performance by training on a single speaker. This is known as *speaker dependent* speech recognition.
- However, there are numerous training problems (long enrollment). An alternate approach is to *adapt* speaker independent models.
- Such adaptation techniques are generally used to reduce mismatch between the acoustic models and the decoding environment (e.g., microphone, acoustic channel and speaker mismatch).
- There are two basic approaches:

- **Maximum A Posteriori (MAP):** choosing an estimate that maximizes the posterior probability (consistent with the observed data and prior information).
- **Maximum Likelihood Linear Regression (MLLR):** ML estimate of a linear transformation.

MAXIMUM A POSTERIORI ADAPTATION

Given observation data X , the MAP estimate is:

$$\hat{\Phi} = \arg \max_{\Phi} [p(X|\Phi)p(\Phi)]$$

If we have no prior information, $p(\Phi)$ is the uniform distribution, and the MAP estimate is identical to the MLE estimate. However, if we have prior information, we can use EM to estimate the new parameters:

$$Q_{MAP}(\Phi, \hat{\Phi}) = \log p(\Phi) + Q(\Phi, \hat{\Phi})$$

Under a significant number of assumptions, we can derive a rather simple and intuitive expression for updating Gaussian means:

$$\hat{\mu}_{ik} = \frac{c_{ik}}{\tau_{ik} + c_{ik}} \mu'_{ik} + \frac{c_{ik}}{\tau_k + c_{ik}} \mu_{ik}$$

where:

μ'_{ik} : ML estimate of the mean

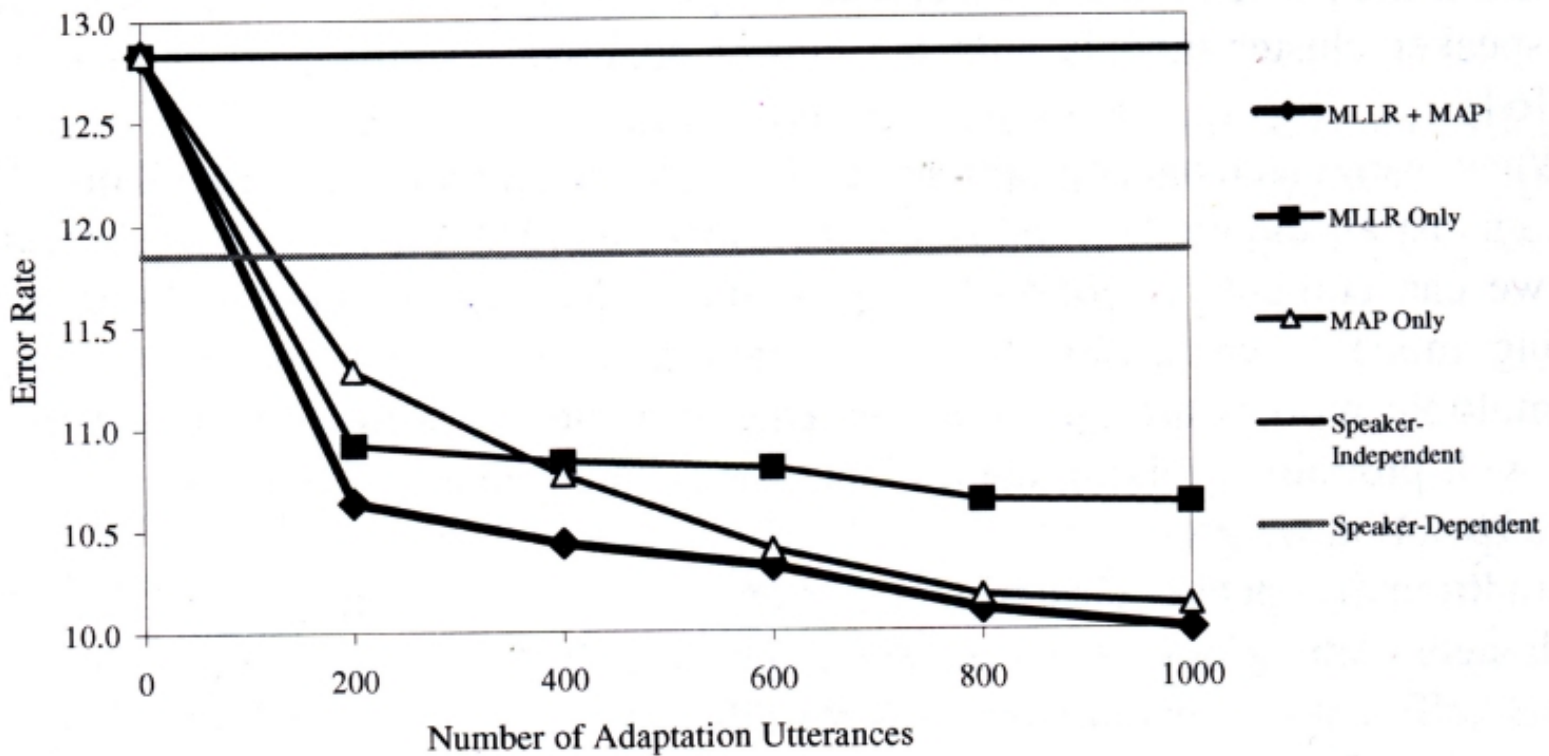
μ_{ik} : existing estimate of the mean (prior information)

c_{ik} : $\sum_{t=1}^T \zeta_{ik}(i, k)$

τ_{ik} : parameter controlling relative weight of prior information

A COMPARISON OF MLLR AND MAP

Below is a comparison of MLLR and MAP on a 60,000 word dictation task. The speaker dependent system was trained on 1,000 sentences.



Though MAP appears to be fairly powerful in this example, MLLR is much more popular. MLLR+MAP combines the best of both approaches, but also leads to a more complicated system.

[Return to Main](#)

[Home](#)

[Exam Database](#)

[First Exam](#)

[Second Exam](#)

[Third Exam](#)

LECTURE 40: EXAM NO. 3

The third exam can be found [here](#).

LECTURE 40: EXAM NO. 3

The third exam can be found [here](#).

[Return to Main](#)[Objectives](#)**Introduction:**[Pattern Recognition](#)**Mutual Information:**[Conditional Likelihood](#)[MMIE](#)[Discussion](#)**Minimum Error Classification:**[Loss Functions](#)[Gradient Descent](#)[Comparison](#)**On-Line Resources:**[Ganapath: Discriminative Techniques](#)[Woodland: MLLR](#)[Juang: MCE](#)

LECTURE 41: DISCRIMINATIVE TRAINING

- Objectives:
 - Mutual Information
 - Maximum Mutual Information Estimation
 - Minimum Error Rate Estimation

This lecture follows the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

Another good source is:

A. Ganapathiraju, *Support Vector Machines for Speech Recognition*, Ph.D. Dissertation, Department of Electrical and Computer Engineering, Mississippi State University, January 2002.

LECTURE 41: DISCRIMINATIVE TRAINING

- Objectives:
 - Mutual Information
 - Maximum Mutual Information Estimation
 - Minimum Error Rate Estimation

This lecture follows the course textbook:

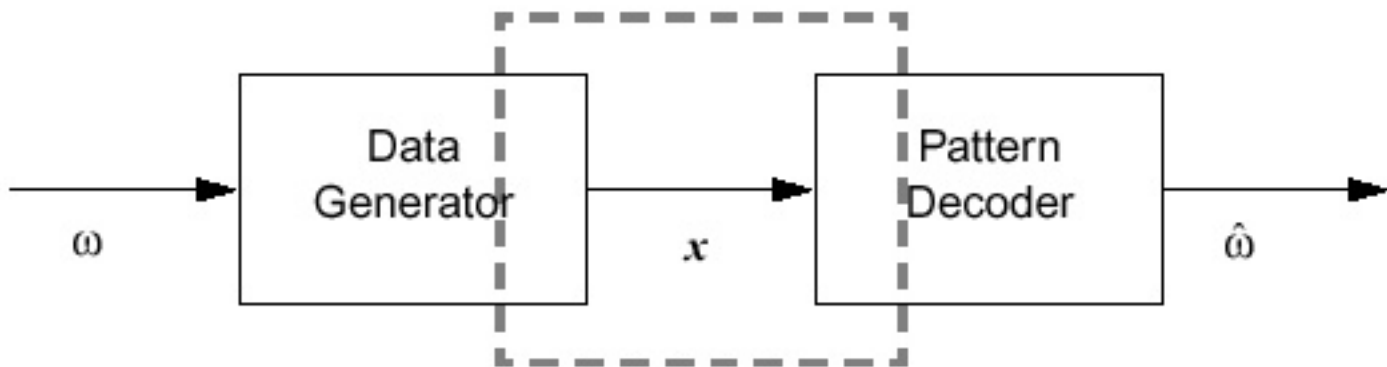
X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

Another good source is:

A. Ganapathiraju, *Support Vector Machines for Speech Recognition*, Ph.D. Dissertation, Department of Electrical and Computer Engineering, Mississippi State University, January 2002.

THE PATTERN RECOGNITION PROBLEM

Recall our communication theory model for speech recognition (simplified):



The rule for minimum error rate classification is to select the class ω_i with the maximum posterior probability, $P(\omega_i|\mathbf{x})$. Recalling Bayes' rule:

$$P(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{p(\mathbf{x})}$$

The probability of the observation, $p(\mathbf{x})$, can be expressed as:

$$p(\mathbf{x}) = \sum_k p(\mathbf{x}|\omega_k)p(\omega_k)$$

Recall that in the classification stage, $p(\mathbf{x})$ can be considered a constant.

A conditional maximum likelihood estimator (CMLE), denoted θ , is defined as follows:

$$\theta_{CMLE}(\mathbf{x}) = \Phi_{MAP} = \underset{\Phi}{\operatorname{argmax}} p_{\Phi}(\omega_i|\mathbf{x})$$

Note that the summation in our equation for $p(\mathbf{x})$ extends over all possible classes (correct and incorrect!) and sums partial probabilities. How will we

estimate these? Similarly, the parameter vector Φ includes not only Φ_i , the parameters for the correct class ω_i , but also those for all other classes.

CONDITIONAL LIKELIHOOD AND MUTUAL INFORMATION

The mutual information between the random variable X (observed data) and the class assignment, Ω , is defined as:

$$I(X, \Omega) = E\left(\log\left(\frac{p(X, \Omega)}{p(X)P(\Omega)}\right)\right) = E\left(\log\frac{p(X|\Omega)P(\Omega)}{p(X)P(\Omega)}\right)$$

Since we don't know the probability distribution for $p(X, \Omega)$, we can assume our sample is representative and define the *instantaneous mutual information*:

$$I(\mathbf{x}, \omega_i) = \log\left(\frac{p(\mathbf{x}, \omega_i)}{p(\mathbf{x})P(\omega_i)}\right)$$

If equal prior information, $P(\omega_i)$, is assumed for all classes, maximizing the conditional likelihood is equivalent to maximizing mutual information. In this case, CMLE becomes *maximum mutual information estimation* (MMIE).

A DISCRIMINANT MODEL

In contrast to MLE, MMIE is concerned with distributions over all possible classes. We can rewrite our equation for $p(\mathbf{x})$ in terms of the correct class assignment and the competing models:

$$\begin{aligned} p(\mathbf{x}) &= \sum_k p(\mathbf{x}|\omega_k)p(\omega_k) \\ &= p(\mathbf{x}|\omega_i)P(\omega_i) + \sum_{k \neq i} p(\mathbf{x}|\omega_k)p(\omega_k) \end{aligned}$$

The posterior probability can be rewritten as:

$$\begin{aligned} P(\omega_i|\mathbf{x}) &= \frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{p(\mathbf{x})} \\ &= \frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{p(\mathbf{x}|\omega_i)P(\omega_i) + \sum_{k \neq i} p(\mathbf{x}|\omega_k)p(\omega_k)} \\ &= \frac{1}{1 + \frac{\sum_{k \neq i} p(\mathbf{x}|\omega_k)p(\omega_k)}{p(\mathbf{x}|\omega_i)P(\omega_i)}} \end{aligned}$$

Maximization of $P(\omega_i|\mathbf{x})$ with respect to all models leads to a discriminative model. It implies the contribution of $p(\mathbf{x}|\omega_i)P(\omega_i)$ from the correct model needs to be reinforced, while the contribution from the competing models, $\sum_{k \neq i} p(\mathbf{x}|\omega_k)p(\omega_k)$, needs to be reduced.

MMIE AND MLE ARE SIMILAR AND YET DIFFERENT

- In MLE, only the correct model is updated during training. In MMIE, all models are updated during training, even with one training sample.
- The greater the prior information on the class assignment, the more effect it has on the MMIE estimator.
- If the assumption of the underlying distribution is correct, MMIE and MLE should converge to the same result. However, in practice, MMIE must produce a lower likelihood for the true class assignment (underlying distribution).
- MMIE and MLE are consistent estimators, but

MMIE has greater variance. MMIE tries not only to increase the likelihood of the correct class, but decrease the likelihood of the incorrect class.

- MMIE is computationally expensive. Why?
- How do we estimate the probability of the class assignment for the incorrect classes?
- Experimental results: CU/HTK word error rates on eval97sub and eval98 using h5train00sub training:

| MMIE | % WER | |
|------------|-----------|--------|
| Iteration | eval97sub | eval98 |
| 0 (MLE) | 46.0 | 46.5 |

| | | |
|---|------|------|
| 1 | 43.8 | 45.0 |
| 2 | 43.7 | 44.6 |
| 3 | 44.1 | 44.7 |

The results in Table [3](#) show that again the peak improvement comes after two iterations, but there is an even larger reduction in WER: 2.3% absolute on eval97sub and 1.9% absolute on eval98. The word error rate for the 1-best hypothesis from the original bigram word lattices measured on 10% of the training data was 27.4%. The MMIE models obtained after two iterations on the same portion of training data gave an error rate of 21.2%, so again MMIE provided a very sizeable reduction in training set error.

MISCLASSIFICATION ERROR RATE AND LOSS FUNCTIONS

Parameter estimation techniques discussed thus far aim to maximize the likelihood (MLE and MAP) or the posterior probability (MMIE). We can also minimize the error rate directly:

$$e_i(\mathbf{x}) = -d_i(\mathbf{x}, \Phi) + \left[\frac{1}{s-1} \sum_{j \neq i} d_j(\mathbf{x}, \Phi)^\eta \right]^{1/\eta}$$

where d_i represent a family of s discriminant functions. $e_i(\mathbf{x}) \geq 0$ implies a recognition error; $e_i(\mathbf{x}) \leq 0$ implies correct recognition. η is a positive constant that controls how we weight the competing classes ($\eta \rightarrow \infty$ favors the top score; $\eta = 1$ implies the average of scores for all competing classes is used).

To transform $e_i(\mathbf{x})$ into a smooth function that can be differentiated, we use a sigmoid function (as is used in neural networks):

$$l_i(\mathbf{x}) = \frac{1}{1 + e^{-e_i(\mathbf{x})}}$$

The recognizer's loss function can be defined as:

$$l(\mathbf{x}, \Phi) = \sum_{i=1}^s l_i(\mathbf{x}) \delta(\omega = \omega_i)$$

We can further define the expected loss as:

$$L(\Phi) = E_{\mathbf{x}}(l(\mathbf{x}, \Phi)) = \sum_{i=1}^s \int_{\omega = \omega_i} l(\mathbf{x}, \Phi) p(\mathbf{x}) d\mathbf{x}$$

GRADIENT DESCENT SOLUTIONS

The expected loss function:

$$L(\Phi) = \sum_{i=1}^s \int_{\omega = \omega_i} l(\mathbf{x}, \Phi) p(\mathbf{x}) d\mathbf{x}$$

can rarely be solved analytically. Instead, we must use an iterative solution (such as a neural network). We can find the optimal parameters by choosing an initial estimate, Φ_0 and following this gradient descent equation:

$$\Phi^{t+1} = \Phi^t - \varepsilon_t \nabla l(\mathbf{x}, \Phi) \big|_{\Phi = \Phi^t}$$

where ε_t is a positive constant controlling the speed of convergence, and $\nabla l(\mathbf{x}, \Phi)$ is the gradient of the recognizer's loss function. We refer to this technique as **minimum classification error rate (MCE)**. The gradient descent is often referred to as **generalized probabilistic descent (GPD)**.

COMPARISON OF PERFORMANCE

- MMIE and MCE are very expensive and often application specific. A similar, more pragmatic approach, is **corrective training**.
- In corrective training, we keep a "near-miss" list and reinforce correct choices, and penalize near misses. This is an ad-hoc procedure that works well in practice.
- MCE and MMIE produce very similar results.

[Return to Main](#)[Objectives](#)**Introduction:**[Neurons](#)[Thresholds](#)[Radial Basis Functions](#)[Perceptrons](#)**Applications:**[Classification](#)[Training](#)[Recurrent Networks](#)**On-Line Resources:**[Ganapath: Overview](#)[OGI: Training](#)[AJR: Speech Applications](#)[ANN Software Links](#)

LECTURE 42: NEURAL NETWORKS

- Objectives:
 - Comparison to HMM States (Neurons)
 - Nonlinearities
 - Multi-layer Perceptron
 - Recurrent Networks

This lecture uses material from:

J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co., ISBN: 0-7803-5386-2, 2000.

and the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

LECTURE 42: NEURAL NETWORKS

- Objectives:
 - Comparison to HMM States (Neurons)
 - Nonlinearities
 - Multi-layer Perceptron
 - Recurrent Networks

This lecture uses material from:

J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co., ISBN: 0-7803-5386-2, 2000.

and the course textbook:

X. Huang, A. Acero, and H.W. Hon, *Spoken*

Language Processing - A Guide to Theory, Algorithm, and System Development, Prentice Hall, Upper Saddle River, New Jersey, USA, ISBN: 0-13-022616-5, 2001.

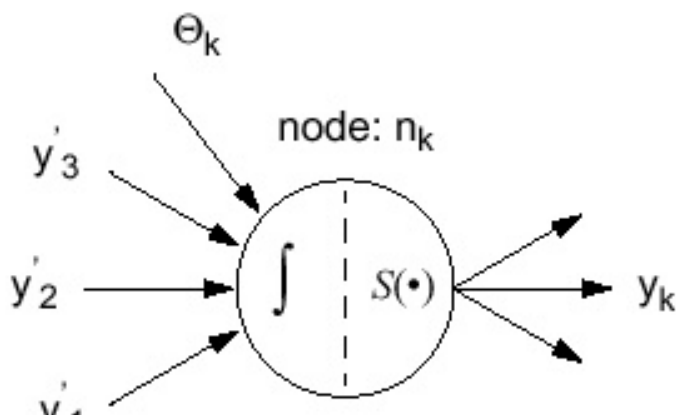
THE ARTIFICIAL NEURAL NETWORK (ANN)

- ❑ Premise: complex computational operations can be implemented by massive integration of individual components
- ❑ Topology and interconnections are key: in many ANN systems, spatial relationships between nodes have some physical relevance
- ❑ Properties of large-scale systems: ANNs also reflect a growing body of theory stating that large-scale systems built from a small unit need not simply mirror properties of a smaller system (contrast fractals and chaotic systems with digital filters)

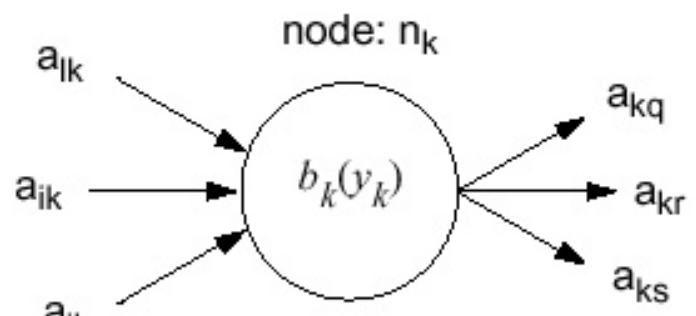
Why Artificial Neural Networks?

- ❑ Important physical observations:
 - The human central nervous system contains $10^{11} - 10^{14}$ nerve cells, each of which interacts with $10^3 - 10^4$ other neurons
 - Inputs may be excitatory (promote firing) or inhibitory

The Artificial Neuron — Nonlinear



The HMM State — Linear



y_i w_{ik} **vector input****scalar output**

$$y_k \equiv S\left(\sum_{n=1}^N w_{ki} y'_i - \theta_k\right)$$

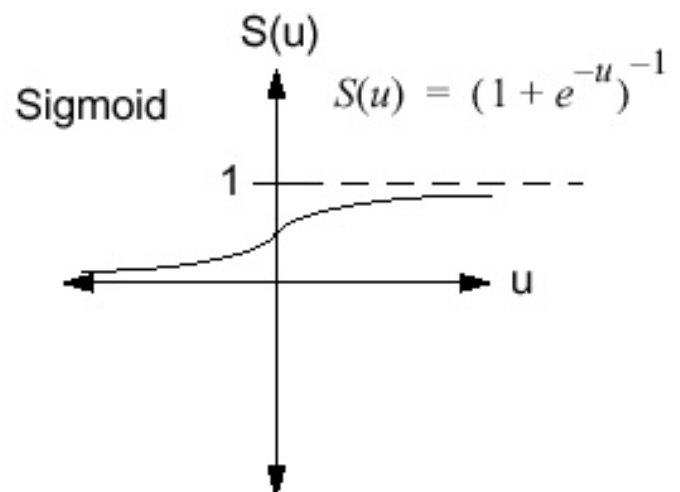
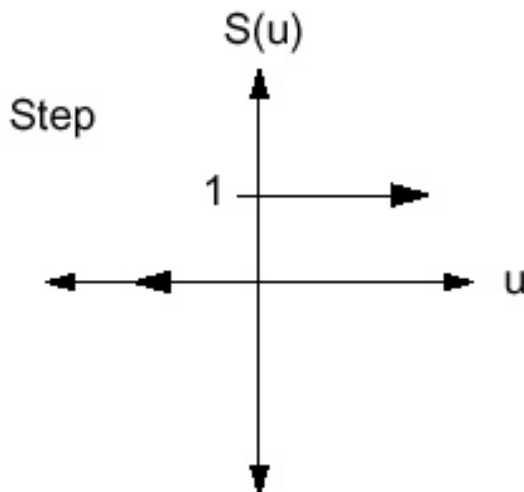
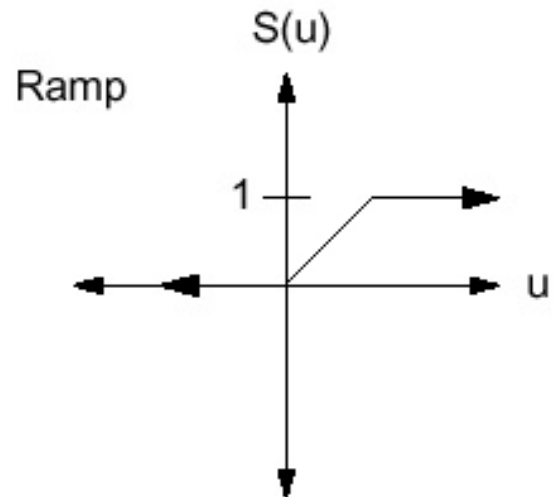
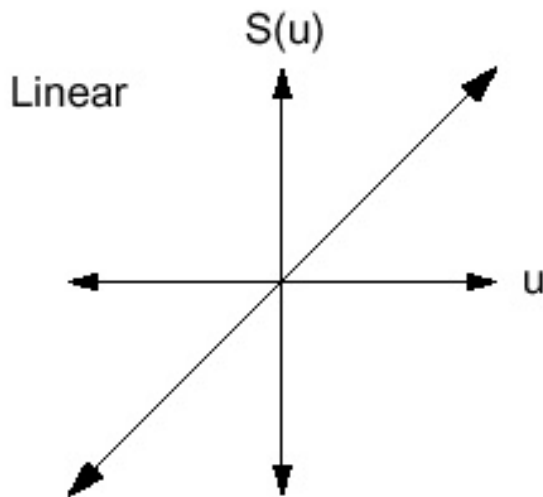
$$\alpha(y_1^{t+1}, j) = \alpha(y_1^t, i) a(j|i) b(y(t+1)|j)$$

TYPICAL THRESHOLDING FUNCTIONS - A KEY DIFFERENCE

The input to the thresholding function is a weighted sum of the inputs:

$$u_k \equiv \mathbf{w}_k^T \mathbf{y}'$$

The output is typically defined by a nonlinear function:



Sometimes a bias is introduced into the threshold function:

$$y_k \equiv S(\mathbf{w}_k^T \mathbf{y}' - \theta_k) = S(u_k - \theta_k)$$

This can be represented as an extra input whose value is always -1:

$$y'_{N+1} = -1 \qquad w_{k, N+1} = \theta_k$$

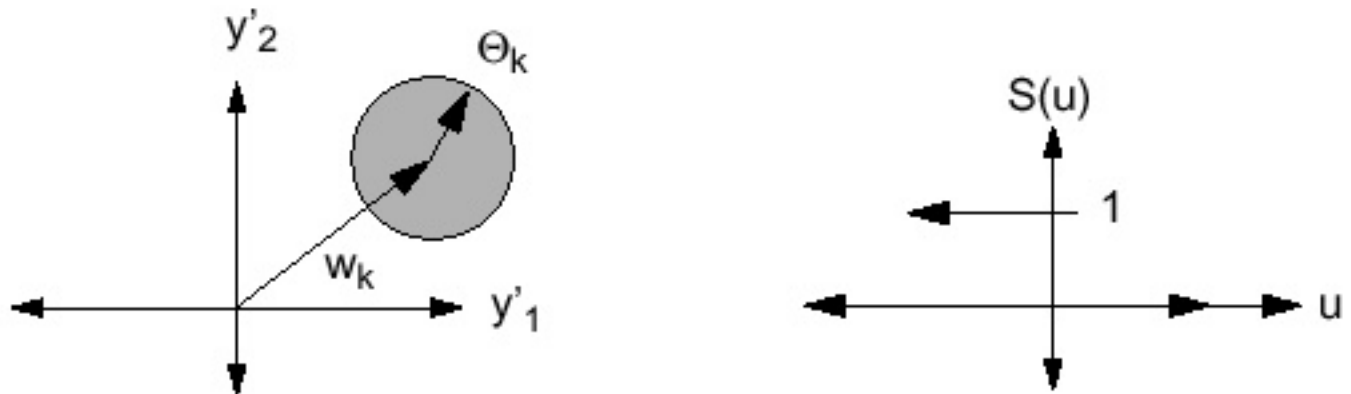
RADIAL BASIS FUNCTIONS

Another popular formulation involves the use of a Euclidean distance:

$$y_k = S\left(\sqrt{\sum_{i=1}^N (w_{ik} - y'_i)^2} - \theta_k\right) = S(\|\mathbf{w}_k - \mathbf{y}'\|_2 - \theta_k)$$

Note the parallel to a continuous distribution HMM.

This approach has a simple geometric interpretation:



Another popular variant of this design is to use a Gaussian nonlinearity:

$$S(u) = e^{-u^2}$$

What types of problems are such networks useful for?

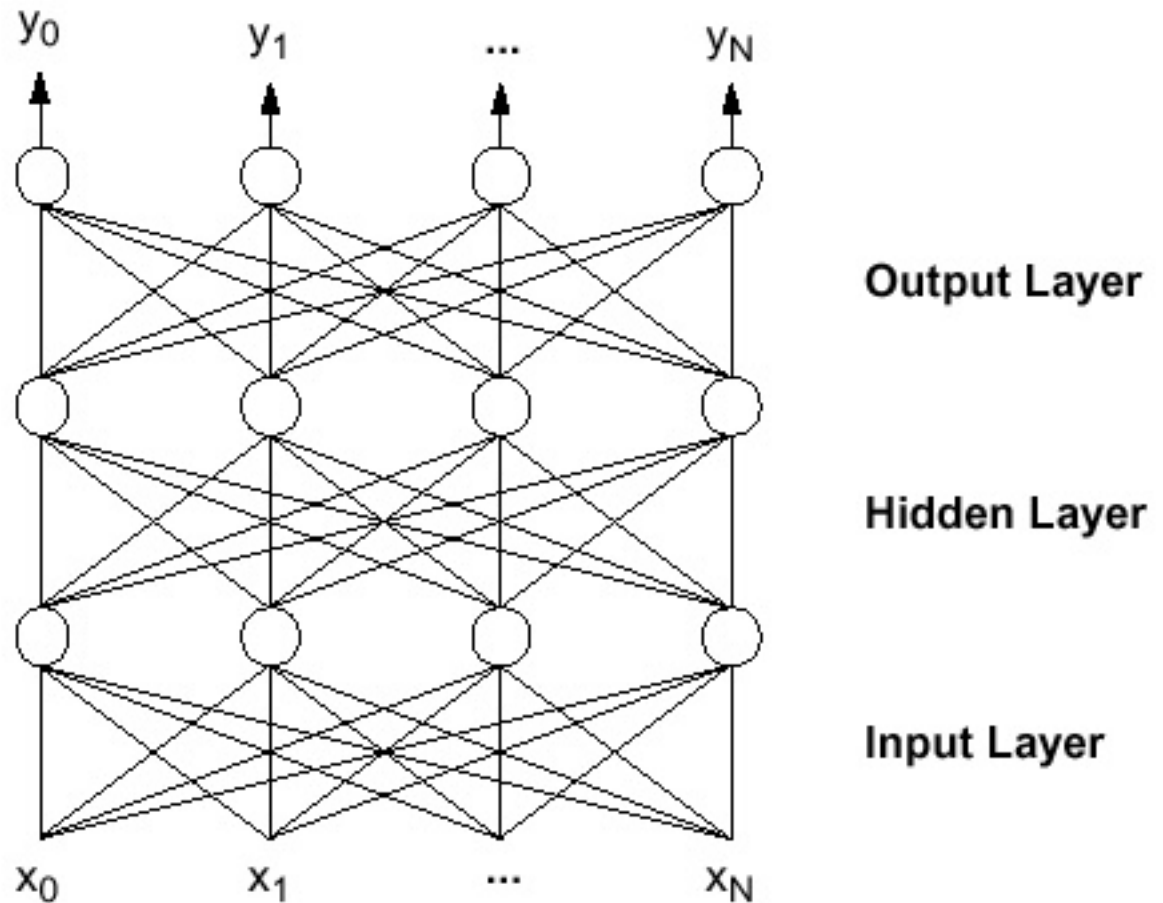
- pattern classification (N-way choice; vector quantization)
- associative memory (generate an output from a noisy input; character recognition)
- feature extraction (similarity transformations; dimensionality reduction)

We will focus on multilayer perceptrons in our studies. These have been shown to be quite useful for a wide range of problems.

MULTI-LAYER PERCEPTRONS

This architecture has the following characteristics:

- Network segregated into layers: N_i cells per layer, L layers
- feedforward, or nonrecurrent, network (no feedback from the output of a node to the input of a node)



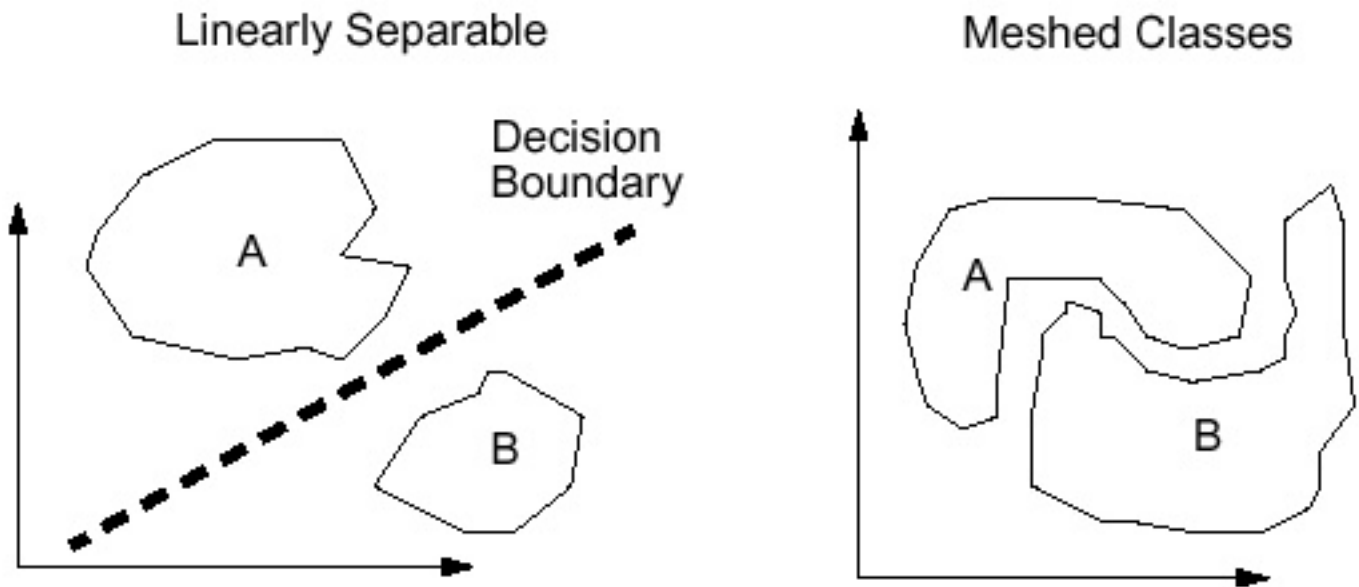
An alternate formulation of such a net is known as the learning vector quantizer (LVQ) — to be discussed later.

The MLP network, not surprisingly, uses a supervised learning algorithm. The network is presented the input and the corresponding output, and must learn the optimal weights of the coefficients to minimize the difference between these two.

The LVQ network uses unsupervised learning — the network adjusts itself automatically to the input data, thereby clustering the data (learning the boundaries representing a segregation of the data). LVQ is popular because it supports discriminative training.

WHY ARTIFICIAL NEURAL NETWORKS FOR SPEECH?

- An ability to separate classes that are not linearly separable:



A three-layer perceptron is required to determine arbitrarily-shaped decision regions.

- Nonlinear statistical models

The ANN is capable of modeling arbitrarily complex probability distributions, much like the difference between VQ and continuous distributions in HMM.

- Context-sensitive statistics

Again, the ANN can learn complex statistical dependencies provided there are enough degrees of freedom in the system.

Why not Artificial Neural Networks? (The Price We Pay...)

- Difficult to deal with patterns of unequal length

- Difficult to deal with patterns of unequal length
- Temporal relationships not explicitly modeled

And, of course, both of these are extremely important to the speech recognition problem.

MLP TRAINING: BACK PROPAGATION

By incorporating a nonlinear transfer function that is differentiable, we can derive an iterative gradient descent training algorithm for a multi-layer perceptron (MLP). This algorithm is known as **back propagation**:

ALGORITHM 4.1: THE BACK PROPAGATION ALGORITHM

Step 1: Initialization: Set $t = 0$ and choose initial weight matrices \mathbf{W} for each layer. Let's denote $w_{ij}^k(t)$ as the weighting coefficients connecting i^{th} input node in layer $k-1$ and j^{th} output node in layer k at time t .

Step 2: Forward Propagation: Compute the values in each node from input layer to output layer in a propagating fashion, for $k = 1$ to K

$$v_j^k = \text{sigmoid}(w_{0j}^k(t) + \sum_{i=1}^N w_{ij}^k(t)v_i^{k-1}) \quad \forall j \quad (4.72)$$

where $\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$ and v_j^k is denoted as the j^{th} node in the k^{th} layer

Step 3: Back Propagation: Update the weights matrix for each layer from output layer to input layer according to:

$$\bar{w}_{ij}^k(t+1) = w_{ij}^k(t) - \alpha \frac{\partial E}{\partial w_{ij}^k(t)} \quad (4.73)$$

where $E = \sum_{i=1}^s \|y_i - o_i\|^2$ and (y_1, y_2, \dots, y_s) is the computed output vector in Step 2.

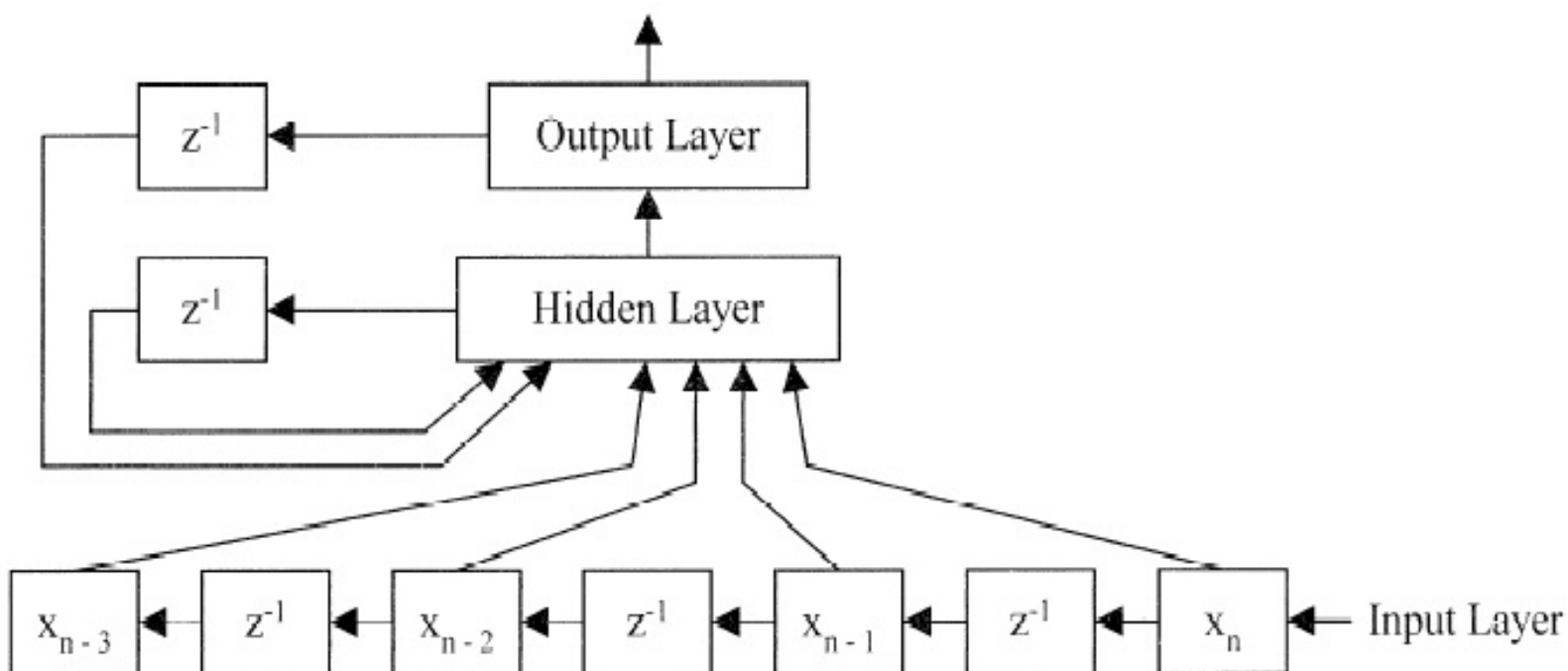
α is referred to as the learning rate and has to be small enough to guarantee convergence. One popular choice is $1/(t+1)$.

Step 4: Iteration: Let $t = t + 1$. Repeat Steps 2 and 3 until some convergence condition is met.

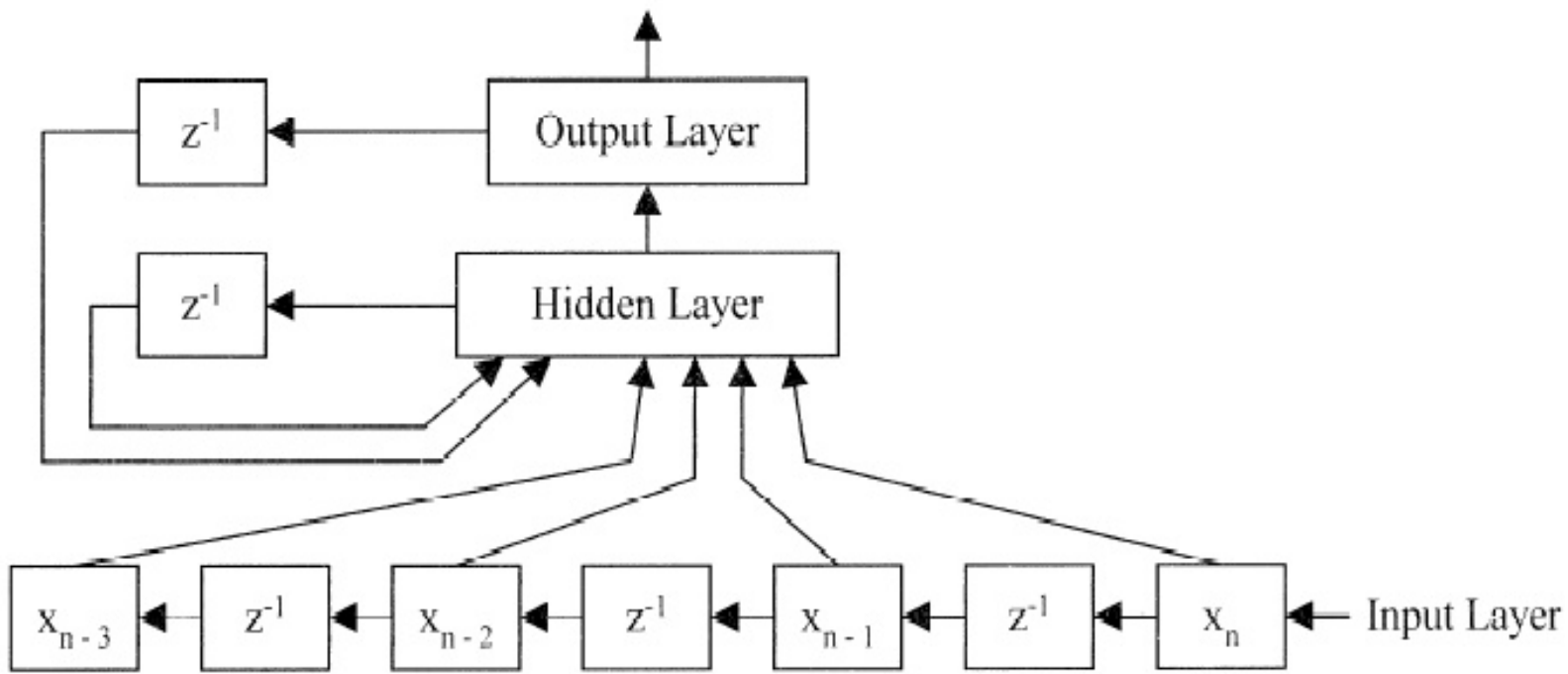
The MLP network has been the most popular architecture for speech processing applications due to the existence of robust training algorithms and its powerful classification properties.

RECURRENT NETWORKS: TOWARDS TIME SYNCHRONOUS DECODING

To incorporate time synchronous behavior into a neural network, we need some sort of feedback loop. The architecture below is known as a recurrent network:



A more popular version of this is the time delay neural network (TDNN):



These recurrent networks have been extremely important to allowing the integration of neural networks into the Markov model statistical framework we use in speech recognition. Such systems are known as **hybrid systems**.

[Return to Main](#)[Objectives](#)**Introduction:**[Evolution](#)[Human Performance](#)**Machine Performance:**[Evaluations](#)[Evolution of Task](#)[String Alignment](#)[NIST Scoring](#)**Other Metrics:**[Information Retrieval](#)[Named Entity](#)[Correlation with WER](#)[Statistical Significance](#)**On-Line Resources:**[AAAS: Recognition](#)[NIST: Tools](#)[Precision and Recall](#)

LECTURE 43: EVALUATION METRICS

- Objectives:
 - Human Performance
 - Machine Performance
 - Automated Scoring: String Alignment
 - Precision and Recall

This lecture uses material from the instructor's notes. Most NLP books contain information about scoring. A good resource is:

D. Jurafsky and J.H. Martin,

*SPEECH and LANGUAGE
PROCESSING: An Introduction
to Natural Language Processing,
Computational Linguistics, and
Speech Recognition,*
Prentice-Hall, ISBN:
0-13-095069-6, 2000.

LECTURE 43: EVALUATION METRICS

- Objectives:
 - Human Performance
 - Machine Performance
 - Automated Scoring: String Alignment
 - Precision and Recall

This lecture uses material from the instructor's notes. Most NLP books contain information about scoring. A good resource is:

D. Jurafsky and J.H. Martin, *SPEECH and LANGUAGE PROCESSING: An Introduction to Natural Language Processing*,

Computational Linguistics, and Speech Recognition, Prentice-Hall, ISBN: 0-13-095069-6, 2000.

AUTOMATED SCORING USING STRING EDITS AND DYNAMIC PROGRAMMING

To automatically score a hypothesis, we must first align it with the reference text, and then count word errors (substitutions, deletions, and insertions).

The desired output is shown below:

Input REF: CUT TALL SPRUCE TREES

Input HYP: HAUL MOOSE FOR FREE

Align REF: CUT TALL SPRUCE *** TREES

Align HYP: *** HAUL MOOSE FOR FREE

| | | | | | | | | | | | | | |
|-----|-------|-----|--|---|-----|--|---|-----|--|---|-----|--|---|
| < | 3 | Sub | | 1 | Ins | | 1 | Del | | 0 | Cor | | 4 |
| Ref | Words | > | | | | | | | | | | | |

The solution to this problem can be achieved using dynamic programming with a Levenshtein distance metric (each non-matching pair adds one to the accumulated distance). We can demonstrate this using a DP grid:


```

R   *** ne5:  . . . . . . . . . . . . . .
E   *** nb5:  . . . . . . . . . . . . . .
F   TRE we4:  . . . . . . . . . . . . . .
E   TRE wb4:  . . . . . . . . . . . . . .
R   SPR we3:  . . ne3: . . . . . . . . . .
E   SPR wb3:  . . wb3: . . . . . . . . . .
N   TAL we2:  . . we2: . . . . . . . . . .
C   TAL wb2:  . . wb2: . . . . . . . . . .
E   CUT we1:  . . we1: . . . . . . . . . .
   CUT wb1:  . . wb1: . . . . . . . . . .
S   ** ne0:   . . ne0: . . . . . . . . . .
T   ** ne0:   . . ne0: . . . . . . . . . .
R
I
N
G

```

nb0 ne0 wb1 we1 wb2 we2 wb3 we3 wb4 we4 nb5 ne5
 *** *** HAU HAU MOO MOO FOR FOR FRE FRE *** ***
 Hypothesis String

THE NIST SCORING REPORT

A typical scoring report from the NIST standard scoring software is shown below:

DETAILED OVERALL REPORT FOR THE SYSTEM:

hypotheses_808080_total.out

SENTENCE RECOGNITION PERFORMANCE

| | | |
|--------------------|------|---|
| sentences | | |
| 12547 | | |
| with errors | 1.9% | (|
| 241) | | |
| with substitutions | 1.1% | (|
| 134) | | |
| with deletions | 0.2% | (|
| 20) | | |
| with insertions | 0.8% | (|
| 102) | | |

WORD RECOGNITION PERFORMANCE

Percent Total Error = 0.6% (263)

| | | | |
|-----------------------|---|-------|-----------|
| Percent Correct | = | 99.6% | (41061) |
| Percent Substitution | = | 0.3% | (138) |
| Percent Deletions | = | 0.1% | (21) |
| Percent Insertions | = | 0.3% | (104) |
| Percent Word Accuracy | = | 99.4% | |

| | | |
|---------------|---|-----------|
| Ref. words | = | (41220) |
| Hyp. words | = | (41303) |
| Aligned words | = | (41324) |

CONFUSION PAIRS
(38)

Total

With >= 1

occurrences (38)

| | | | | |
|----|----|----|-----------|-------|
| 1: | 13 | -> | five ==> | oh |
| 2: | 12 | -> | oh ==> | nine |
| 3: | 9 | -> | nine ==> | oh |
| 4: | 8 | -> | two ==> | three |
| 5: | 7 | -> | oh ==> | eight |
| 6: | 7 | -> | oh ==> | four |
| 7: | 6 | -> | four ==> | five |
| 8: | 5 | -> | eight ==> | three |

| | | | |
|-----|---|----|-----------------|
| 9: | 5 | -> | five ==> nine |
| 10: | 5 | -> | four ==> oh |
| 11: | 5 | -> | three ==> eight |
| 12: | 5 | -> | zero ==> oh |
| 13: | 4 | -> | oh ==> seven |
| 14: | 4 | -> | seven ==> oh |
| 15: | 4 | -> | three ==> two |
| 16: | 3 | -> | eight ==> six |
| 17: | 3 | -> | eight ==> two |
| 18: | 3 | -> | nine ==> one |
| 19: | 3 | -> | oh ==> two |
| 20: | 3 | -> | two ==> oh |
| 21: | 2 | -> | eight ==> one |
| 22: | 2 | -> | five ==> eight |
| 23: | 2 | -> | nine ==> five |
| 24: | 2 | -> | oh ==> zero |
| 25: | 2 | -> | seven ==> one |
| 26: | 2 | -> | six ==> eight |
| 27: | 1 | -> | eight ==> five |
| 28: | 1 | -> | eight ==> nine |
| 29: | 1 | -> | eight ==> seven |
| 30: | 1 | -> | four ==> one |
| 31: | 1 | -> | one ==> five |
| 32: | 1 | -> | one ==> four |
| 33: | 1 | -> | seven ==> nine |
| 34: | 1 | -> | seven ==> six |

```

35:      1  ->  seven ==> zero
36:      1  ->  six ==> three
37:      1  ->  three ==> one
38:      1  ->  zero ==> two

```

138

INSERTIONS

(11)

occurrences (11)

Total

With >= 1

```

1:      43  ->  oh
2:      17  ->  eight
3:      13  ->  six
4:       9  ->  one
5:       8  ->  nine
6:       6  ->  two
7:       3  ->  three
8:       2  ->  four
9:       1  ->  five
10:      1  ->  seven
11:      1  ->  zero

```

DELETIONS
(3)

occurrences (3)

| | | | |
|----|----|----|-------|
| 1: | 11 | -> | oh |
| 2: | 6 | -> | eight |
| 3: | 4 | -> | two |

21

Total

With >= 1

SUBSTITUTIONS
(11)

occurrences (11)

| | | | |
|----|----|----|-------|
| 1: | 35 | -> | oh |
| 2: | 20 | -> | five |
| 3: | 16 | -> | eight |
| 4: | 14 | -> | nine |

Total

With >= 1

| | | | |
|-----|----|----|-------|
| 5: | 12 | -> | four |
| 6: | 11 | -> | two |
| 7: | 10 | -> | three |
| 8: | 9 | -> | seven |
| 9: | 6 | -> | zero |
| 10: | 3 | -> | six |
| 11: | 2 | -> | one |

138

* NOTE: The 'Substitution' words are those reference words
for which the recognizer supplied an incorrect word.

FALSELY RECOGNIZED
(11)

Total

With >= 1

occurrences (11)

| | | | |
|----|----|----|-------|
| 1: | 39 | -> | oh |
| 2: | 19 | -> | nine |
| 3: | 16 | -> | eight |
| 4: | 14 | -> | three |

| | | | |
|-----|----|----|-------|
| 5: | 11 | -> | two |
| 6: | 10 | -> | five |
| 7: | 9 | -> | one |
| 8: | 8 | -> | four |
| 9: | 5 | -> | seven |
| 10: | 4 | -> | six |
| 11: | 3 | -> | zero |

138

* NOTE: The 'Falsely Recognized' words are those hypothesis words which the recognizer incorrectly substituted for a reference word.

DUMP OF SYSTEM ALIGNMENT STRUCTURE

System name: hypotheses_808080_total.out

Speakers:

0: bg

1: bk

...

161: sn

162: tb

Speaker sentences 0: bg #utts: 77

id: (bg_119oo39a)

Scores: (#C #S #D #I) 7 0 0 0

REF: one one nine oh oh three nine

HYP: one one nine oh oh three nine

Eval:

id: (bt_41722a)

Scores: (#C #S #D #I) 5 0 0 1

REF: four ** one seven two two

HYP: four OH one seven two two

Eval: I

id: (gf_886374oa)

Scores: (#C #S #D #I) 6 1 0 0

REF: eight eight six three seven FOUR oh

HYP: eight eight six three seven OH oh

Eval: S

...

id: (gf_886a)

Scores: (#C #S #D #I) 3 0 0 0

REF: eight eight six

HYP: eight eight six

Eval:

id: (gf_892a)

Scores: (#C #S #D #I) 3 0 0 0

REF: eight nine two

HYP: eight nine two

Eval:

id: (gf_8a)

Scores: (#C #S #D #I) 1 0 0 0

REF: eight

HYP: eight

Eval:

id: (gf_8b)

Scores: (#C #S #D #I) 1 0 0 0

REF: eight

HYP: eight

Eval:

id: (gf_8o156a)

Scores: (#C #S #D #I) 5 0 0 0

REF: eight oh one five six

HYP: eight oh one five six

Eval:

id: (gf_914a)
Scores: (#C #S #D #I) 3 0 0 0

SYSTEM SUMMARY PERCENTAGES by
SPEAKER

| | | | | | | | | | |
|-----------------------------|------|-------|-------|-------|-------|-----|-----|--|--|
| -----. | | | | | | | | | |
| | | | | | | | | | |
| hypotheses_808080_total.out | | | | | | | | | |
| ----- | | | | | | | | | |
| | SPKR | | # Snt | # Wrđ | Corr | Sub | Del | | |
| Ins | Err | S.Err | | | | | | | |
| -----+-----+----- | | | | | | | | | |
| | bg | | 77 | 253 | 100.0 | 0.0 | 0.0 | | |
| 0.0 | 0.0 | 0.0 | | | | | | | |
| -----+-----+----- | | | | | | | | | |
| | bk | | 77 | 253 | 99.6 | 0.4 | 0.0 | | |
| 0.0 | 0.4 | 1.3 | | | | | | | |
| -----+-----+----- | | | | | | | | | |

. . .

| | | | | | | | | | | | |
|---------|-----|---------|--|-------|-------|-------|--|-----|-----|--|--|
| ===== | | | | | | | | | | | |
| | | Sum/Avg | | 12547 | 41220 | 99.6 | | 0.3 | 0.1 | | |
| 0.3 | 0.6 | 1.9 | | | | | | | | | |
| ===== | | | | | | | | | | | |
| | | Mean | | 77.0 | 252.9 | 99.6 | | 0.3 | 0.1 | | |
| 0.3 | 0.6 | 1.9 | | | | | | | | | |
| | | S.D. | | 0.2 | 1.1 | 1.1 | | 1.0 | 0.2 | | |
| 0.7 | 1.4 | 3.9 | | | | | | | | | |
| | | Median | | 77.0 | 253.0 | 100.0 | | 0.0 | 0.0 | | |
| 0.0 | 0.4 | 1.3 | | | | | | | | | |
| `-----' | | | | | | | | | | | |

SYSTEM SUMMARY PERCENTAGES by
SPEAKER

| | | | | | | | | | |
|-----------------------------|-----|-------|--|-------|-------|------|--|-----|-----|
| ,-----. | | | | | | | | | |
| | | | | | | | | | |
| hypotheses_808080_total.out | | | | | | | | | |
| ----- | | | | | | | | | |
| | | SPKR | | # Snt | # Wrđ | Corr | | Sub | Del |
| Ins | Err | S.Err | | | | | | | |

A number line diagram illustrating the addition of 77 and 253. The number line is marked from 0 to 1000. The first jump is from 0 to 77, labeled '77'. The second jump is from 77 to 330, labeled '253'. The final position is 330, labeled '330'.

A number line from 0 to 1000. The line is marked with tick marks every 100 units. The number 1000 is at the far right. The number 0 is at the far left. The number 100 is labeled 'bk'. The number 77 is labeled '77'. The number 253 is labeled '253'. The number 252 is labeled '252'. The number 1 is labeled '1'. The number 0 is labeled '0'.

• • •

| | | | | | | | | | |
|-------|--|-----|--|-------|-------|--|-------|-----|----|
| ===== | | | | | | | | | |
| | | Sum | | 12547 | 41220 | | 41061 | 138 | 21 |
| 104 | | 263 | | 241 | | | | | |

| | | | | | | | |
|-----|-----|--------|-------|------|-------|--|-------------------------|
| | | | ===== | | | | |
| | | Mean | | 77.0 | 252.9 | | 251.9 0.8 0.1 |
| 0.6 | 1.6 | 1.5 | | | | | |
| | | S.D. | | 0.2 | 1.1 | | 2.9 2.5 0.4 |
| 1.7 | 3.5 | 3.0 | | | | | |
| | | Median | | 77.0 | 253.0 | | 253.0 0.0 0.0 |
| 0.0 | 1.0 | 1.0 | | | | | |

EXPERIMENTAL DESIGN: STATISTICAL SIGNIFICANCE

Why is this important?

[Click here](#) if you want to learn more about how to measure statistical significance.

[Return to Main](#)

[Objectives](#)

Review:

[Technology](#)

State of the Art:

[LVCSR Systems](#)

On-Line Resources:

[AAAS: Recognition](#)

[SRSTW'02](#)

[2001 Hub 5E](#)

[2001 SPINE](#)

LECTURE 44: STATE OF THE ART SYSTEMS

● Objectives:

- Review of the basic components of a speech recognition system
- Compare a generic trigram/triphone system to state of the art
- Introduce more exotic features of real systems

This lecture uses material from the instructor's notes and ISIP's annual speech recognition workshop:

J. Picone, et al, "Speech Recognition System Training Workshop,"

<http://www.isip.msstate.edu/conferences/srstw/>,

Institute for Signal and Information Processing,
Mississippi State University, Mississippi,
USA, May 2002.

LECTURE 44: STATE OF THE ART SYSTEMS

- Objectives:

- Review of the basic components of a speech recognition system
- Compare a generic trigram/triphone system to state of the art
- Introduce more exotic features of real systems

This lecture uses material from the instructor's notes and ISIP's annual speech recognition workshop:

J. Picone, et al, "Speech Recognition System Training Workshop,"

<http://www.isip.msstate.edu/conferences/srstw/>,
Institute for Signal and Information Processing,
Mississippi State University, Mississippi,
USA, May 2002.

[Return to Main](#)

[Home](#)

[Exam Database](#)

[First Exam](#)

[Second Exam](#)

[Third Exam](#)

[Final Exam](#)

LECTURE 46: FINAL EXAM

The final exam can be found [here](#).