FUNDAMETALS OF SPEECH RECOGNITION: A SHORT COURSE

by

Dr. Joseph Picone
Institute for Signal and Information Processing
Department of Electrical and Computer Engineering
Mississippi State University

ABSTRACT

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 several thousand 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.

Who am I?

Affiliations:

Associate Professor, Electrical and Computer Engineering
Director, Institute for Signal and Information Processing

Location:

Mississippi State University
413 Simrall, Box 9571
Mississippi State, Mississippi 39762
Phone/Fax: (601) 325-3149
Email: picone@isip.msstate.edu (http://isip.msstate.edu)

Education:

Ph.D. in E.E., Illinois Institute of Technology, December 1983
M.S. in E.E., Illinois Institute of Technology, May 1980
B.S. in E.E., Illinois Institute of Technology, May 1979

Areas of Research:

Speech Recognition and Understanding, Signal Processing

Educational Responsibilities:

Signal and Systems (third year UG course)
Introduction to Digital Signal Processing (fourth year B.S./M.S. course)
Fundamentals of Speech Recognition (M.S./Ph.D. course)

Research Experience:

MS State (1994-present):
large vocabulary speech recognition, object-oriented DSP
Texas Instruments (1987-1994):
telephone-based speech recognition, Tsukuba R&D Center
AT&T Bell Laboratories (1985-1987):
isolated word speech recognition, low-rate speech coding
Texas Instruments (1983-1985):
robust recognition, low rate speech coding
AT&T Bell Laboratories (1980-1983):
medium rate speech coding, low rate speech coding

Professional Experience:

Senior Member of the IEEE, Professional Engineer
Associate Editor of the *IEEE Speech and Audio Transactions*
Associate Editor of the *IEEE Signal Processing Magazine*

Course Outline

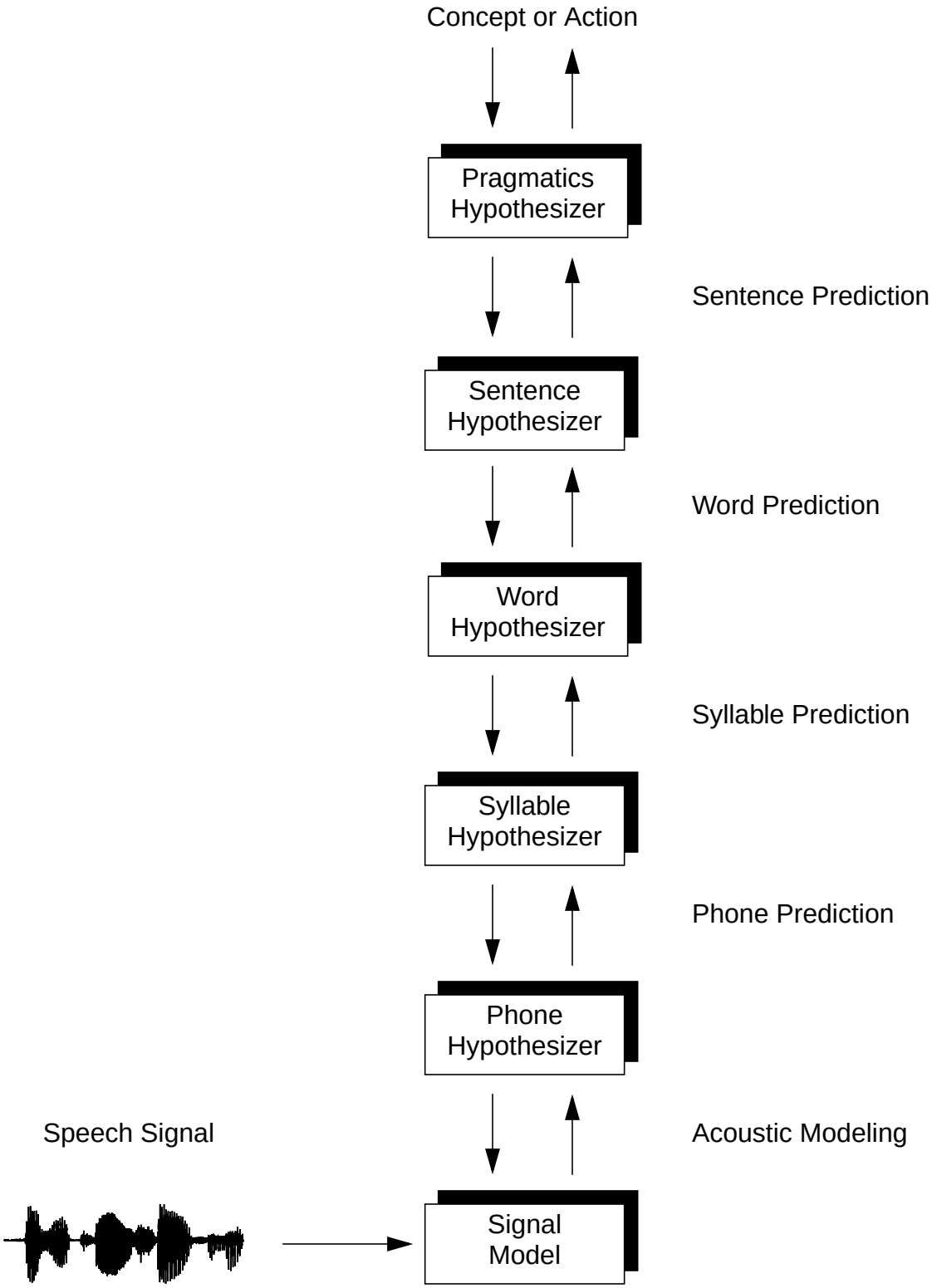
Suggested Textbooks:

1. J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co., ISBN 0-02-328301-7
2. L. Rabiner and B.H. Juang, *Fundamentals of Speech Recognition*, Prentice-Hall, ISBN 0-13-015157-2

Session I:

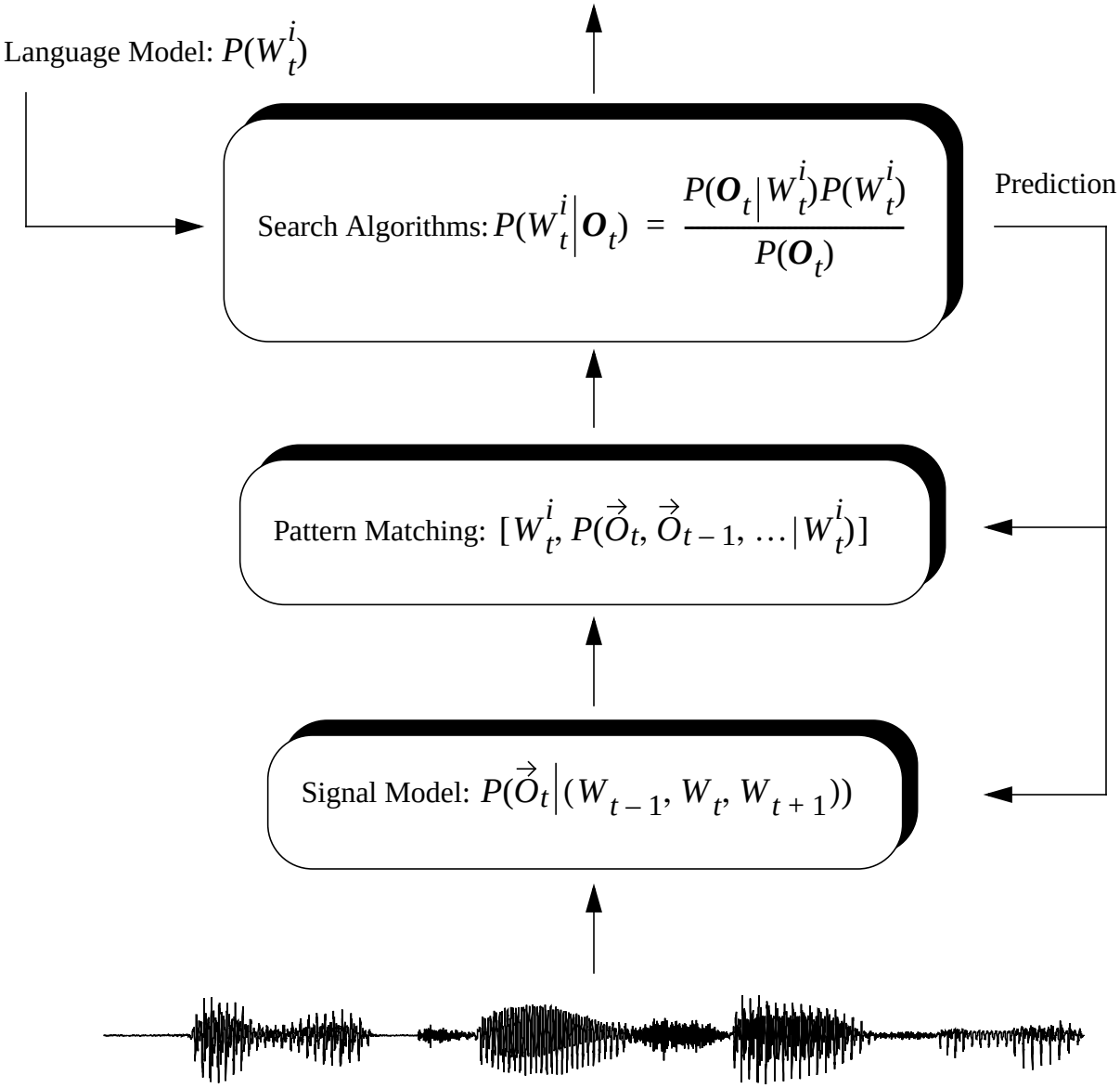
Fundamentals of Speech

A GENERIC SOLUTION



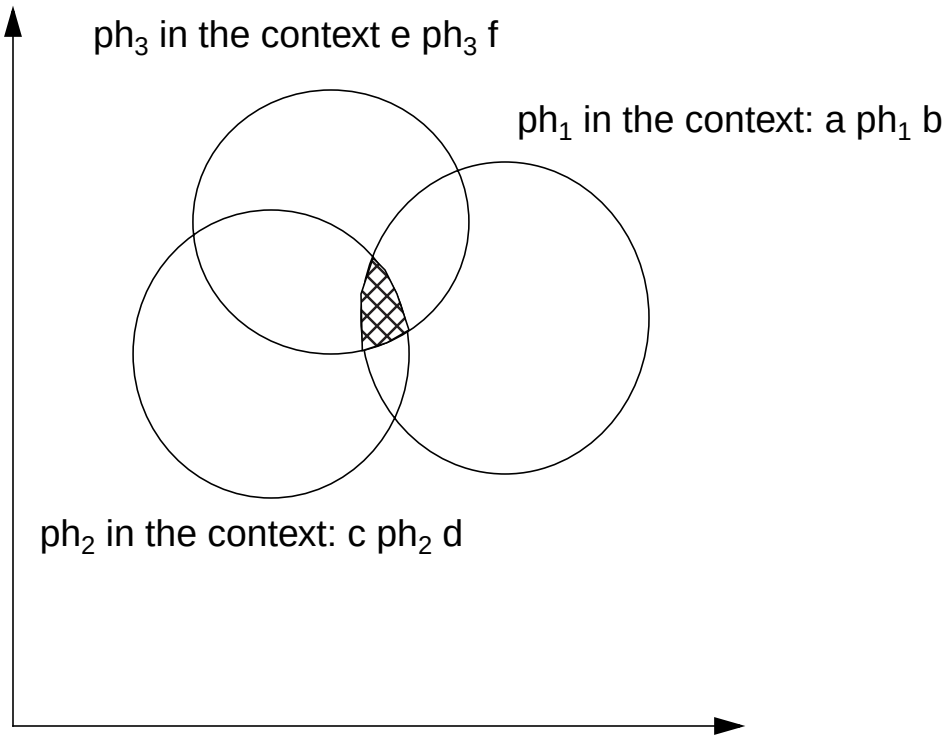
BASIC TECHNOLOGY: A PATTERN RECOGNITION PARADIGM BASED ON HIDDEN MARKOV MODELS

Recognized Symbols: $P(S|\boldsymbol{O}) = \arg max|_T \prod_i P(W_t^i | (\vec{O}_t, \vec{O}_{t-1}, \ldots))$



WHAT IS THE BASIC PROBLEM?

Feature No. 2



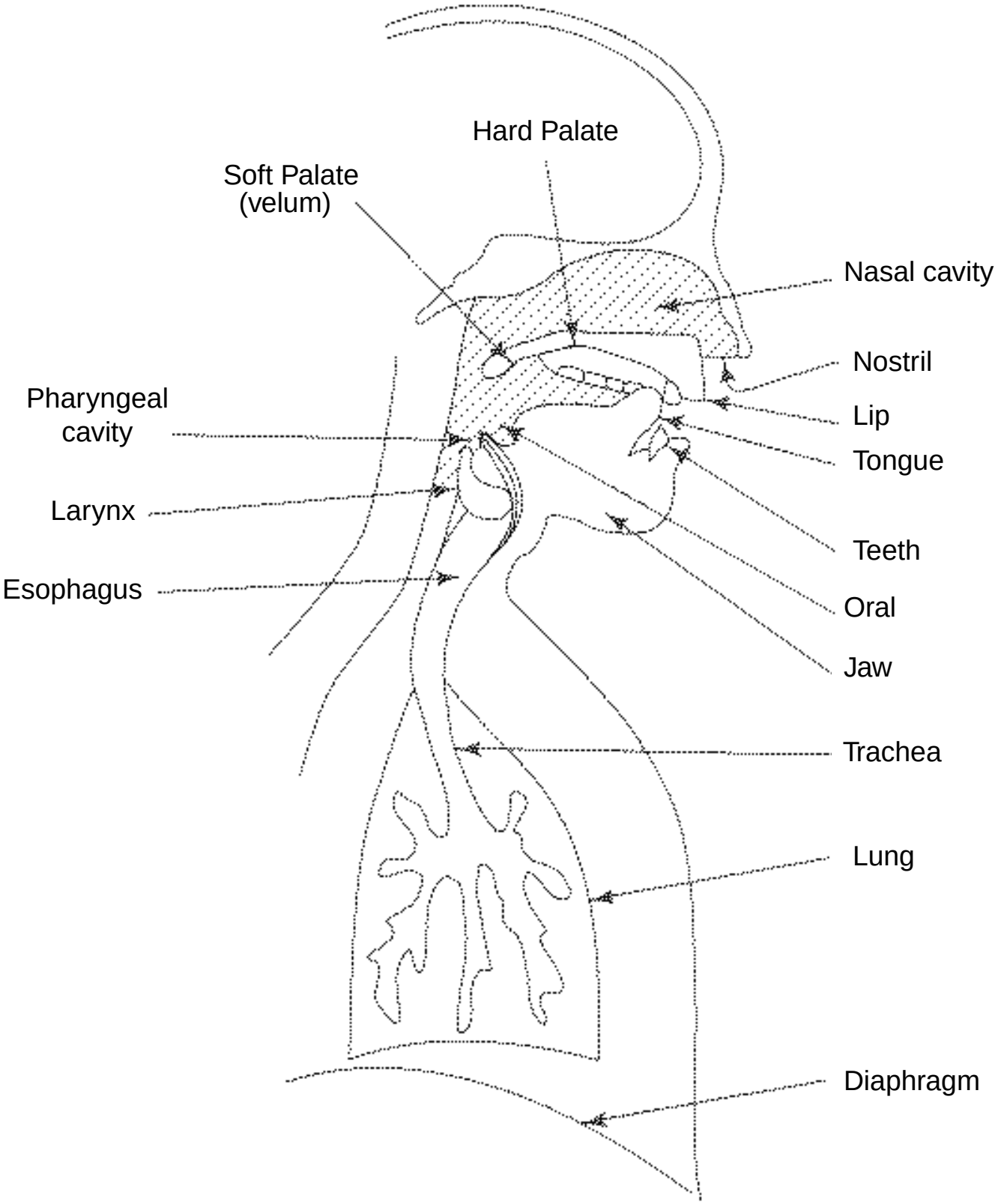
Feature No. 1

What do we do about the regions of overlap?

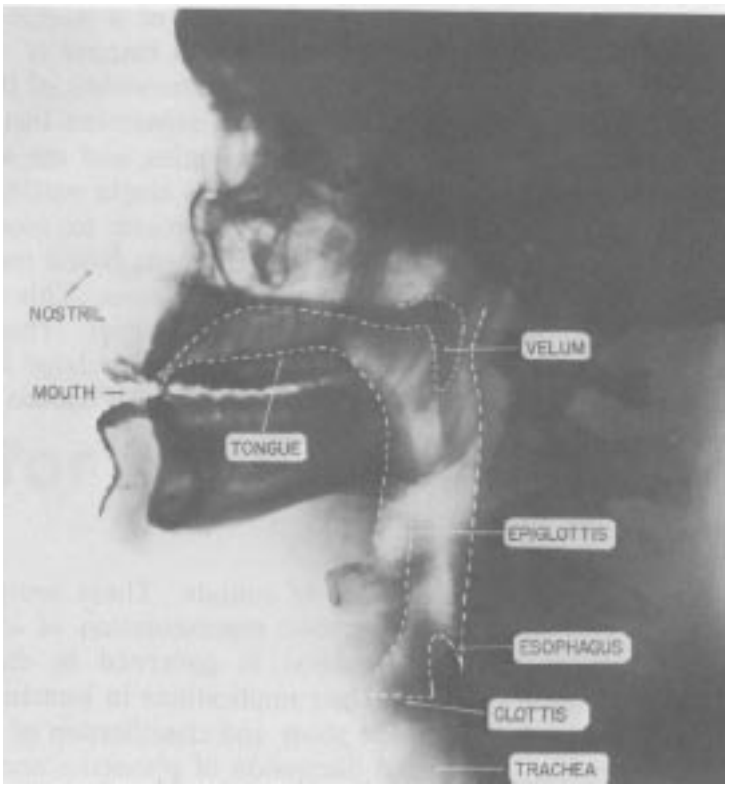
Context is required to disambiguate them.

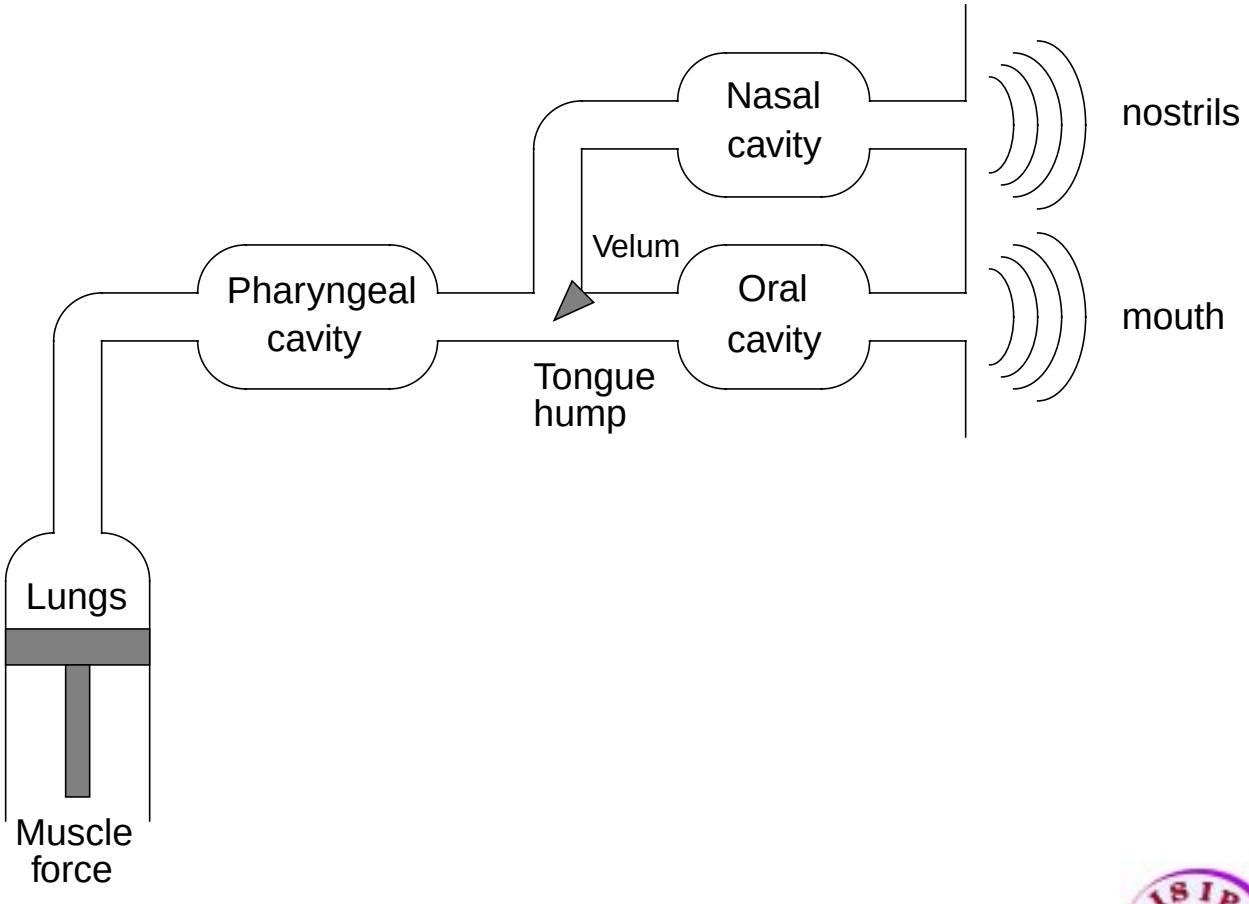
The problem of where words begin and end is similar.

Speech Production Physiology

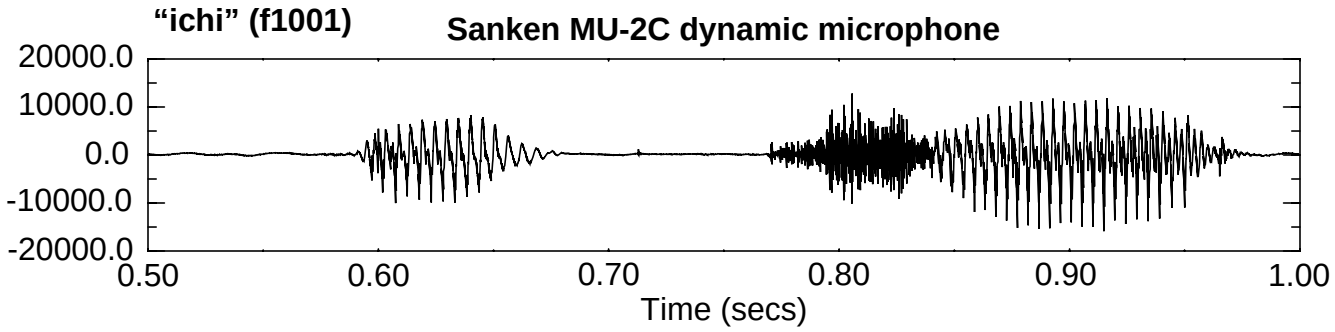


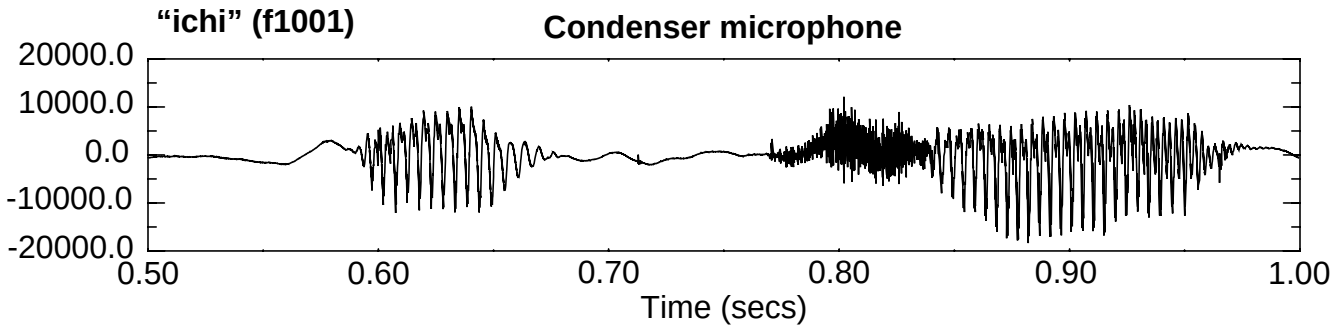
A Block Diagram of Human Speech Production

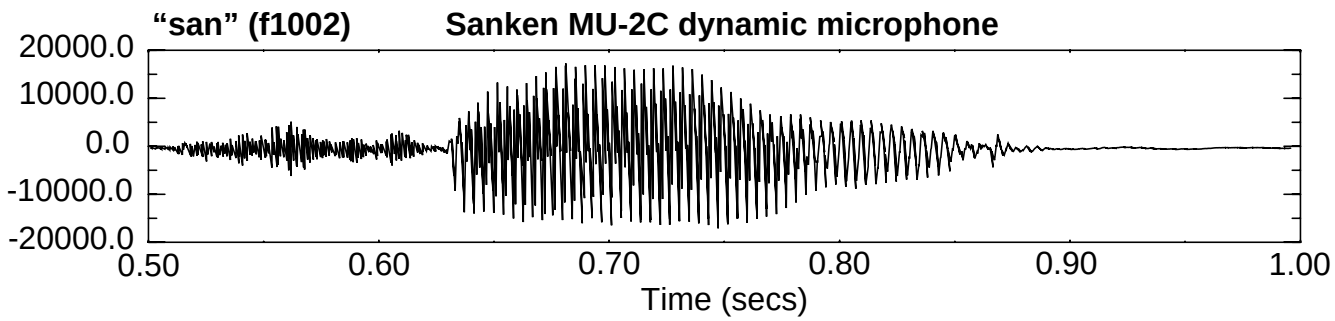


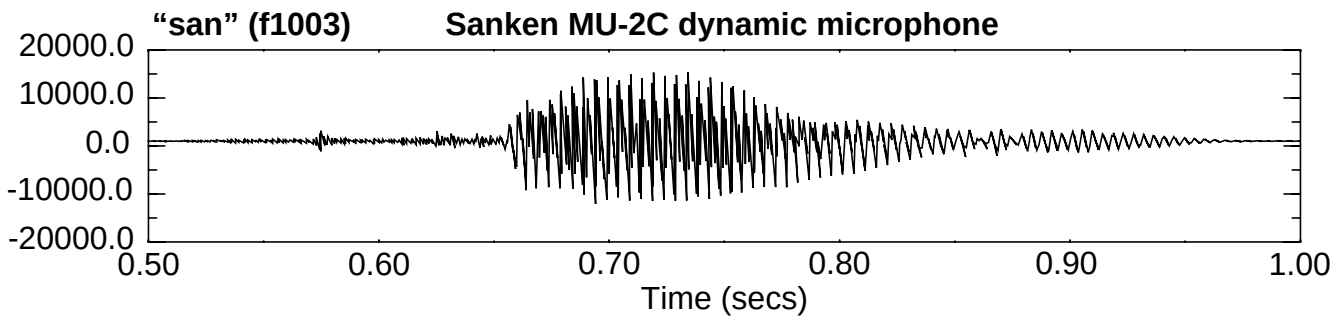


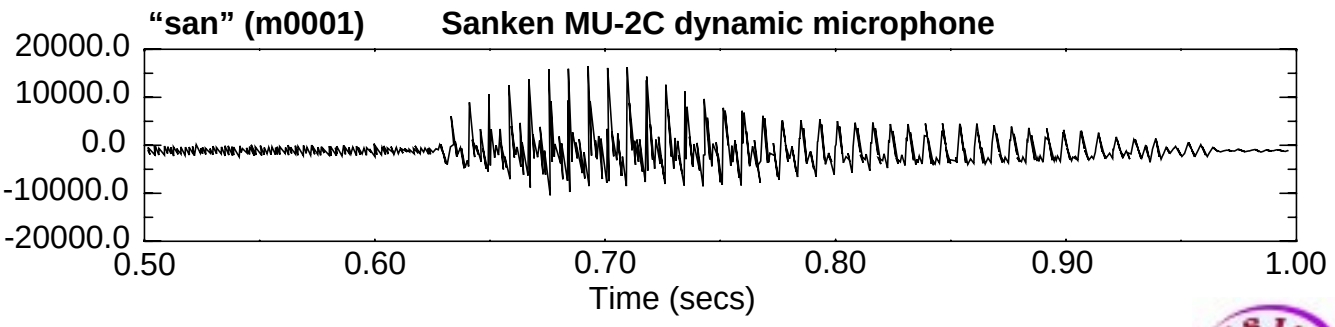
What Does A Speech Signal Look Like?





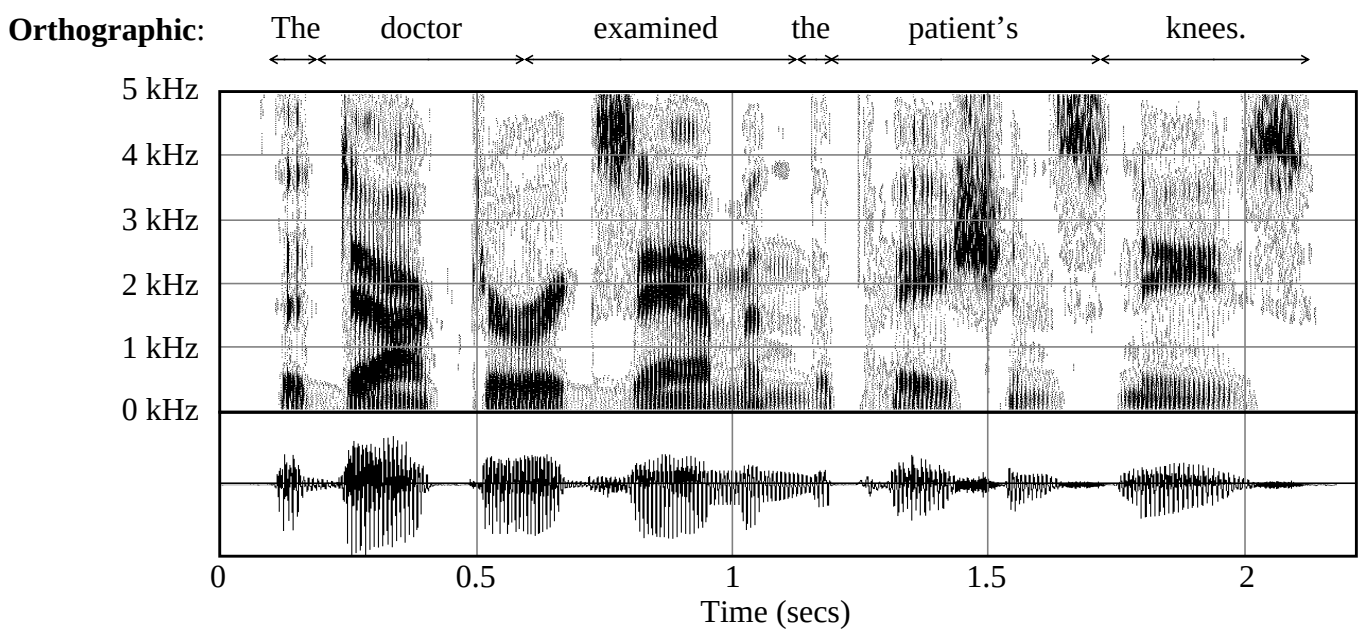




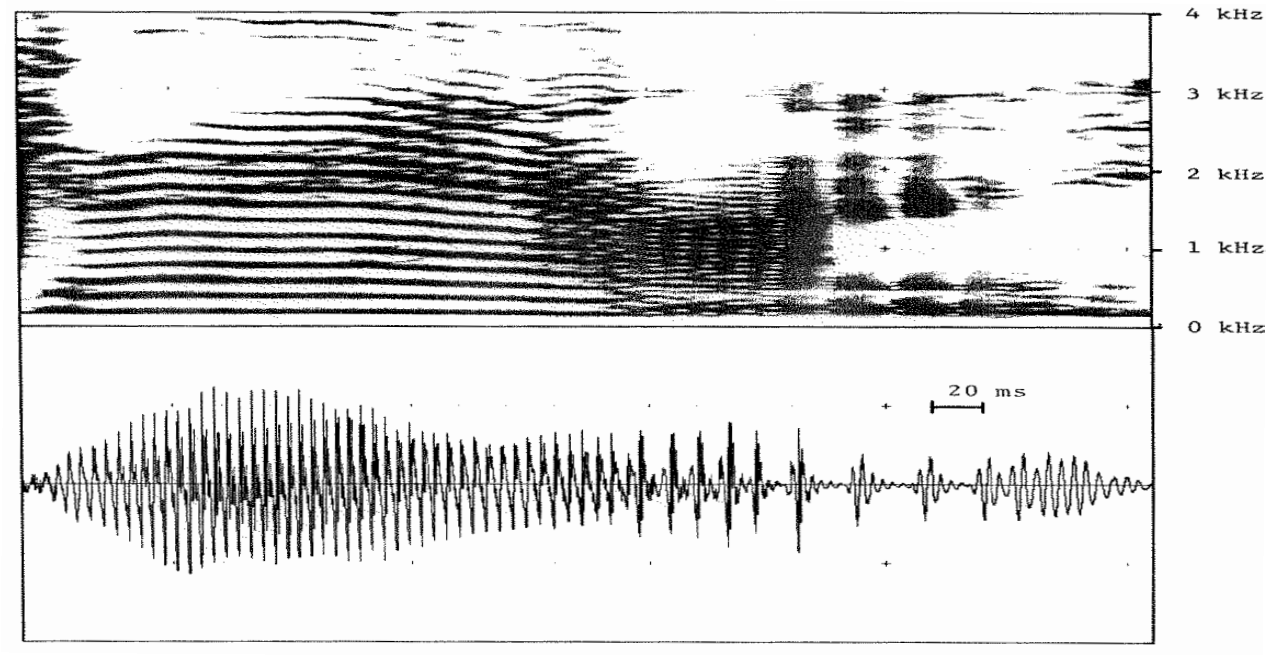


What Does A Speech Signal Spectrogram Look Like?

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



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



"Drown" (female)

Phonemics and Phonetics

Some simple definitions:

Phoneme:
- an ideal sound unit with a complete set of articulatory gestures.
- the basic theoretical unit for describing how speech conveys linguistic meaning.
- (For English, there are about 42 phonemes.)
- Types of phonemes: vowels, semivowels, dipthongs, 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

Phonemic and Phonetic 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 recently by Jim Hieronymous (AT&T) to deal with multiple languages within a single ASCII system

Example:

CONSONANTS

The Worldbet representation of each IPA symbol is written below it. IPA symbols in parentheses are rare phonemes, for which no machine-readable coding has yet been proposed. (In these cases a coding employing diacritics is proposed.)

	Bi-labial	Labio-dental	Dental	Alveolar	Post-alveolar	Retro-flex	Palatal	Velar	Uvular	Pharyn-geal	Glottal
Plosive	p b p b			t d t d		ʈ ɖ tr dr	c ɟ c J	k g k g	q ɢ q Gj		ʔ ?
Nasal	m m	ɱ M		n n		ɳ nr	ɲ n~	ŋ N	(ɴ) Nq		
Trill	ʙ B			r r					ʀ R		
Tap or Flap				ɾ ɽ t(d(		ɽr ɭ r(rr					
Fricative	ɸ β F V	f v	θ ð T D	s z s z	ʃ ʒ S Z	ʂ ʐ sr zr	ç (j) C jA	x ɣ x G	χ ʁ X K	ħ ʕ H !	h ɦ h hv
Lateral fricative				(ɬ) (ɮ) hl Zl							
Approxi-mant		(ʋ) V[		ɹ 9		ɻ 9r	j j	(ɰ) 4)			
Lateral approx.				l l		ɭ lr	ʎ L	(ʟ) Lg			
Ejective stop	p' p>			t' t>		ʈ' t> r	c' c>	k' k>	q' q>		
Implosive	ɓ p<b<			ɗ t< d<			ɟ' c< J<	ɠ ɡ' k< g<	q<G<		

VOWELS

	Front		Central		Back	
Close	i y i y		ɨ ʉ ix ux		ɯ u 4 u	
		ɪ ʏ I Y	ɪ̈ Ix		ʊ U	
Close-mid	e ø e 7				ɤ o 2 o	
			ɔ ɔ̈ Ax ox			
Open-mid	ɛ œ E 8				ʌ ɔ A >	
	æ @		ɐ ax			
Open	a ɶ a 6				ɑ ɒ A 5	

†= iis not an approved IPA symbol, but it is in such common use that we have propose Ix as the most natural ASCII representation for a "centralized r".

TABLE 1: Worldbet Consonant and Vowel Symbols

When We Put This All Together:

We Have An Acoustic Theory of Speech Production

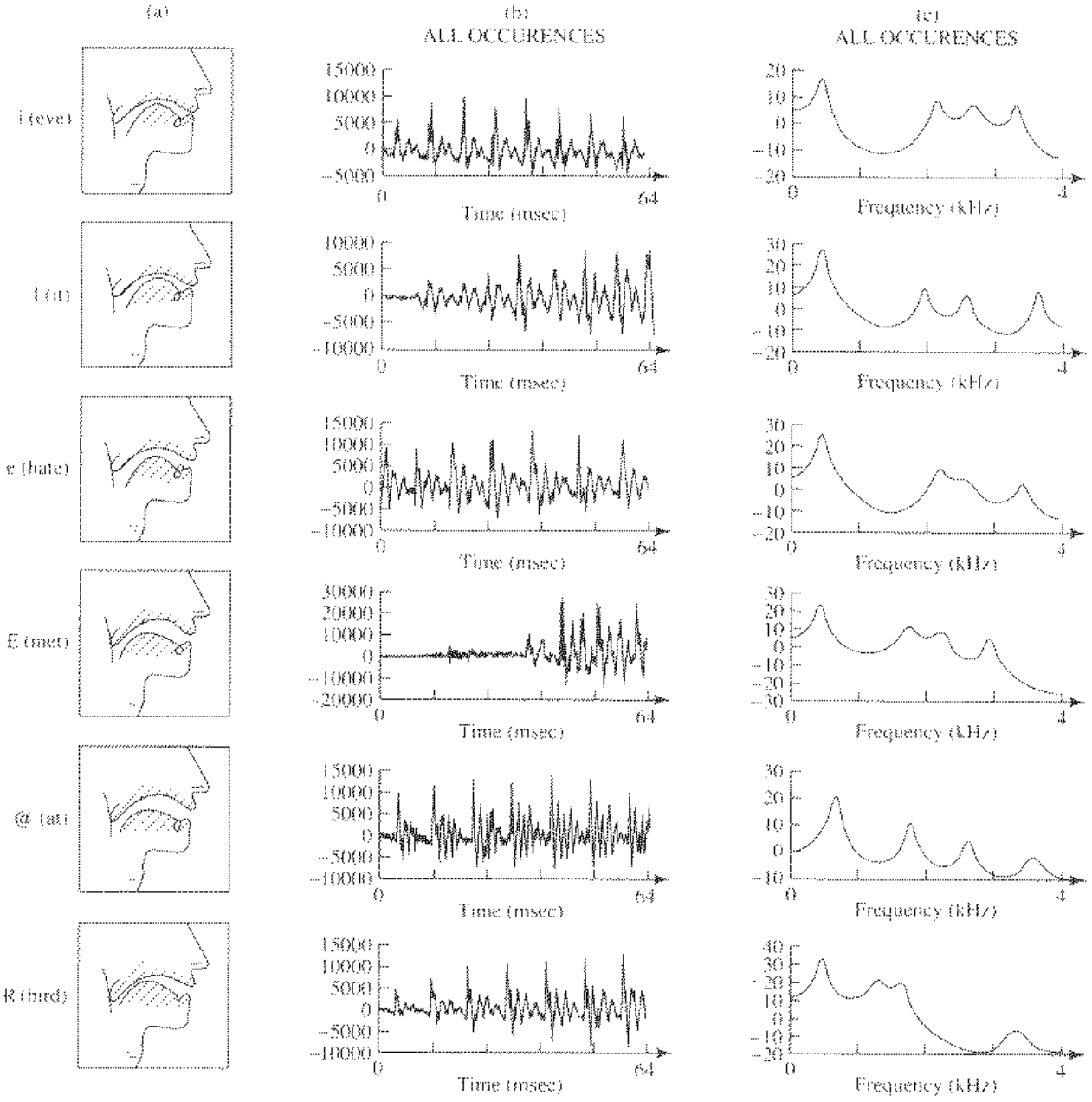


FIGURE 2.10. A collection of features for vowels in American English. Column (a) represents schematic vocal-tract profiles, (b) typical acoustic waveforms, and (c) the corresponding vocal-tract magnitude spectrum for each vowel.

Consonants Can Be Similarly Classified

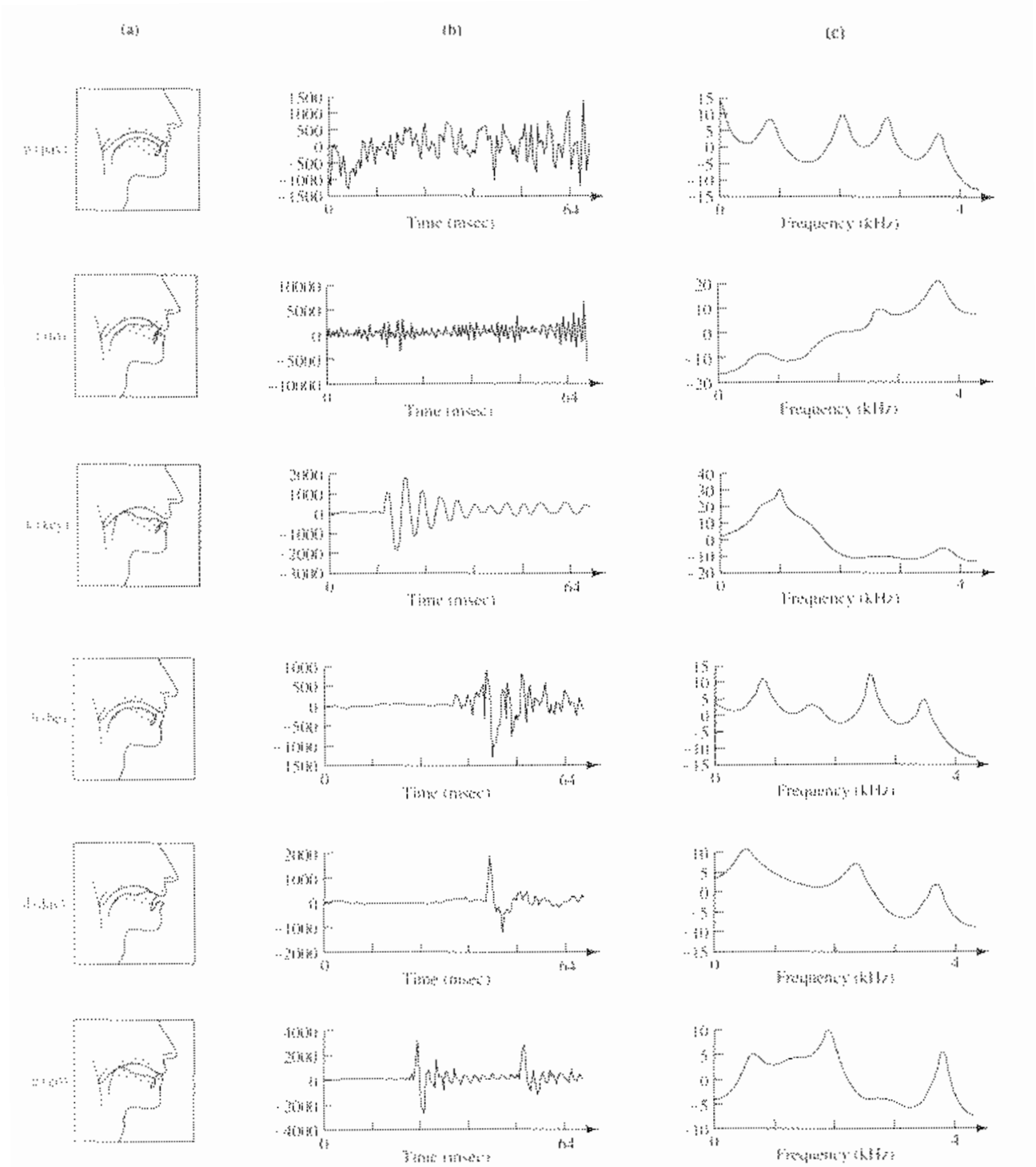
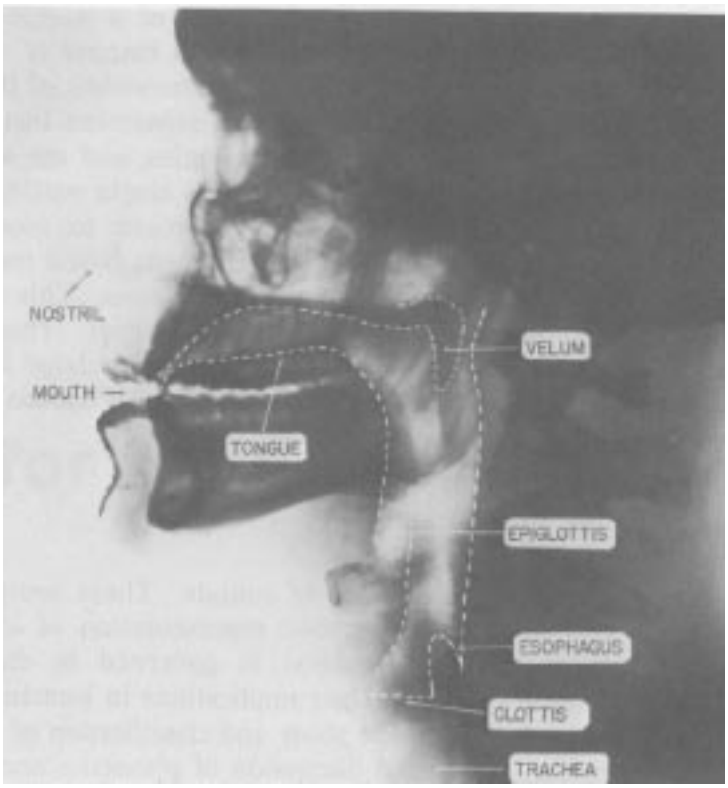


FIGURE 2.18. A collection of features for voiced and unvoiced stops in American English. Column (a) represents schematic vocal-tract profiles just prior to release, (b) typical acoustic waveforms, and (c) the corresponding vocal-tract magnitude spectra.

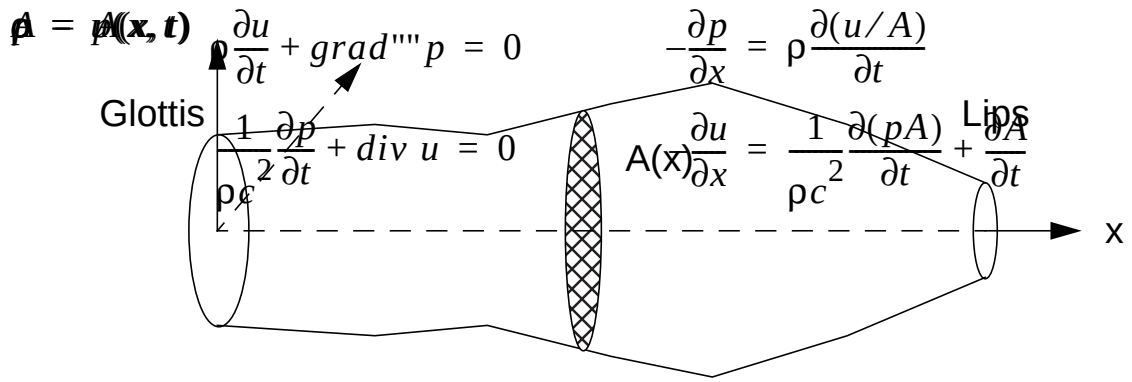
Sound Propagation



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:



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:

$$\rho\frac{\partial(u/A)}{\partial t} + grad""p = 0 \qquad \frac{1}{\rho c^2}\frac{\partial p}{\partial t} + \frac{\partial A}{\partial t} + div\ u = 0$$

or

$$-\frac{\partial p}{\partial x} = \rho\frac{\partial(u/A)}{\partial t} \qquad -\frac{\partial u}{\partial x} = \frac{1}{\rho c^2}\frac{\partial(pA)}{\partial t} + \frac{\partial A}{\partial t}$$

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:

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

The solution is a traveling wave:

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

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

which is analogous to a transmission line:

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

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

where

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

The sinusoisdal steady state solutions are:

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

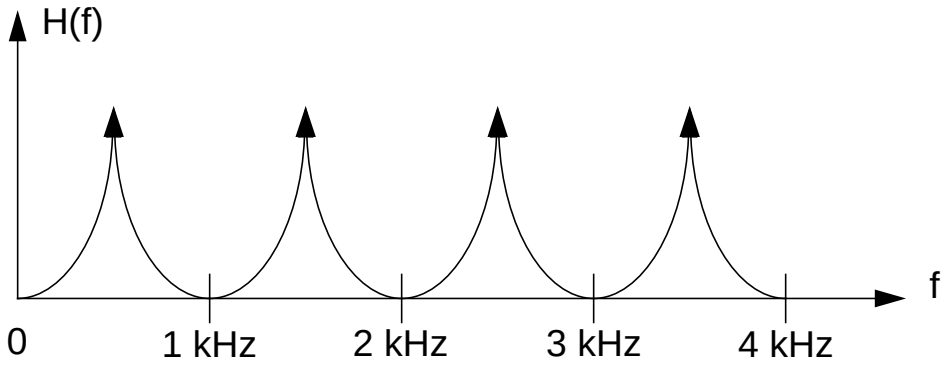
$$u(x,t) = \frac{\cos[\Omega(l-x)/c]}{\cos[\Omega l/c]} U_G(\Omega) e^{j\Omega t}$$

where $Z_0 = \frac{\rho c}{A}$ is the characteristic impedance.

The transfer function is given by:

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

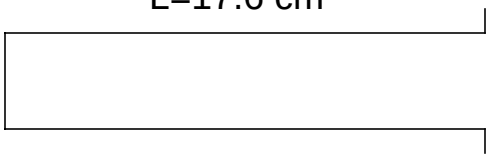
This function has poles located at every $\frac{(2n+1)\pi c}{2l}$. Note that these correspond to the frequencies at which the tube becomes a quarter wavelength: $\left(\frac{\Omega l}{c} = \frac{\pi}{2}\right) \Rightarrow \left(\Omega = \frac{c}{4l}\right)$.



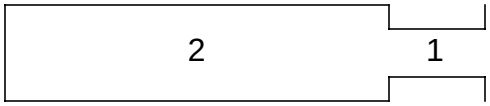
Is this model realistic?

Resonator Geometry

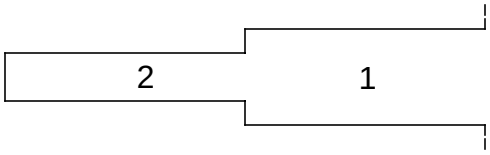
L=17.6 cm



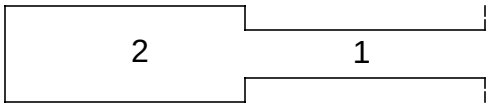
$L_2/L_1 = 8 \qquad A_2/A_1 = 8$



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

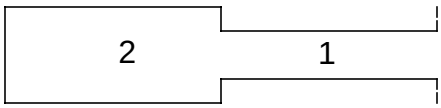


$L_2/L_1 = 1.0 \qquad A_2/A_1 = 8$



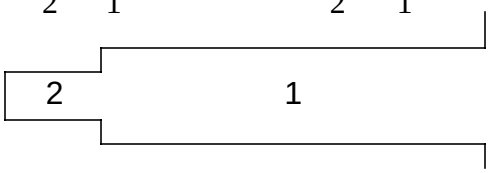
$\leftarrow L_1 + L_2 = 17.6\ cm \rightarrow$

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



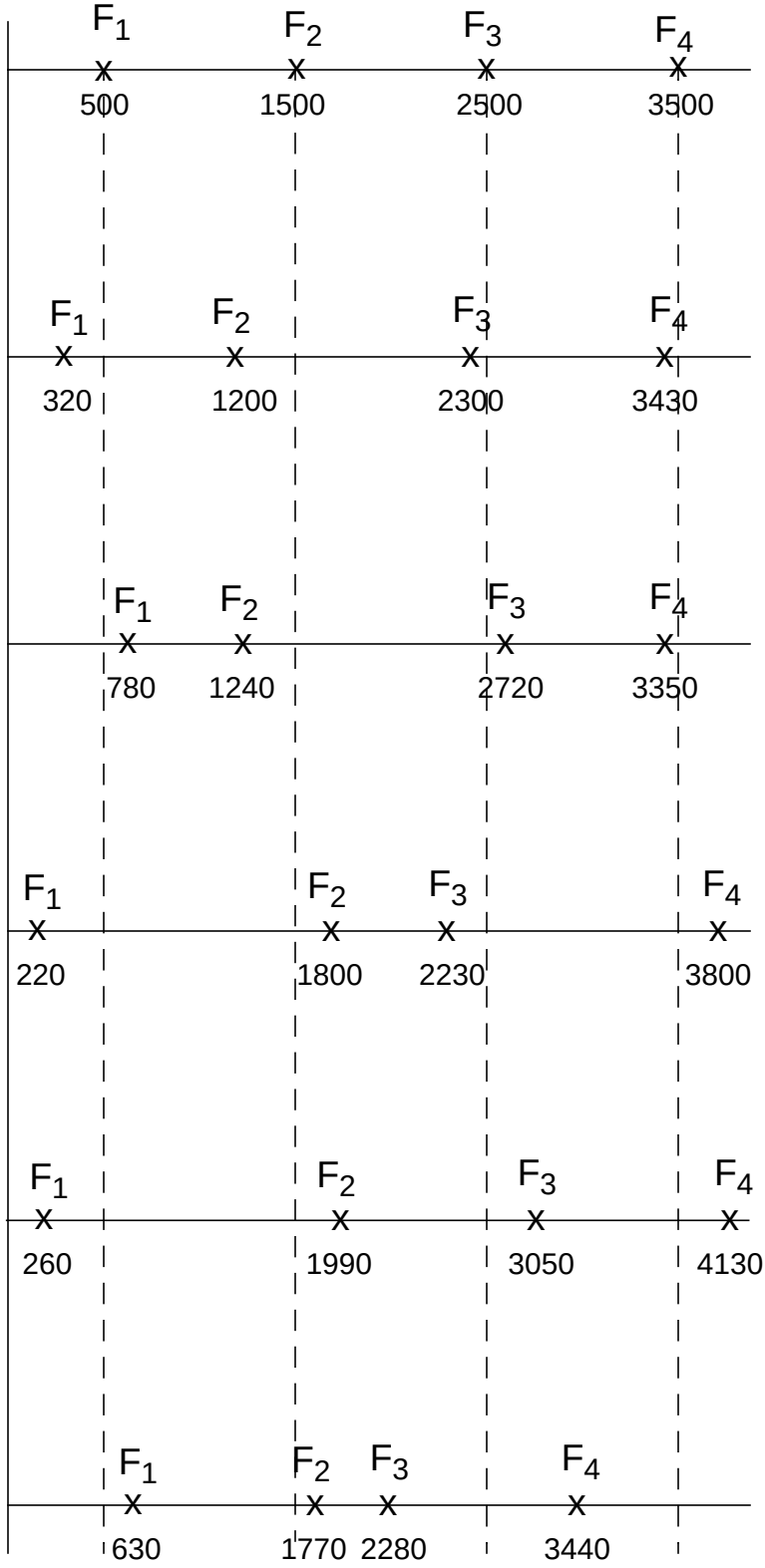
$\leftarrow L_1 + L_2 = 14.5\ cm \rightarrow$

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



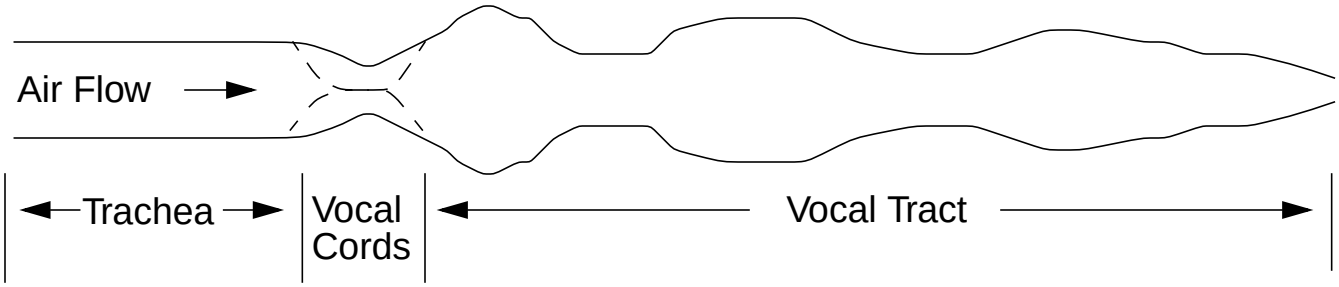
$\leftarrow L_1 + L_2 = 17.6\ cm \rightarrow$

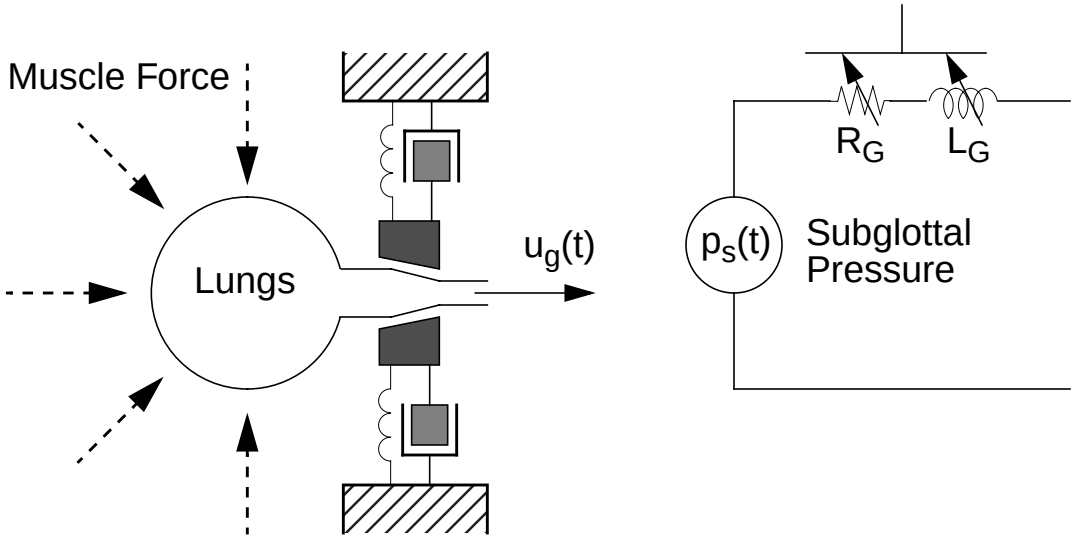
Formant Patterns



Excitation Models

How do we couple energy into the vocal tract?





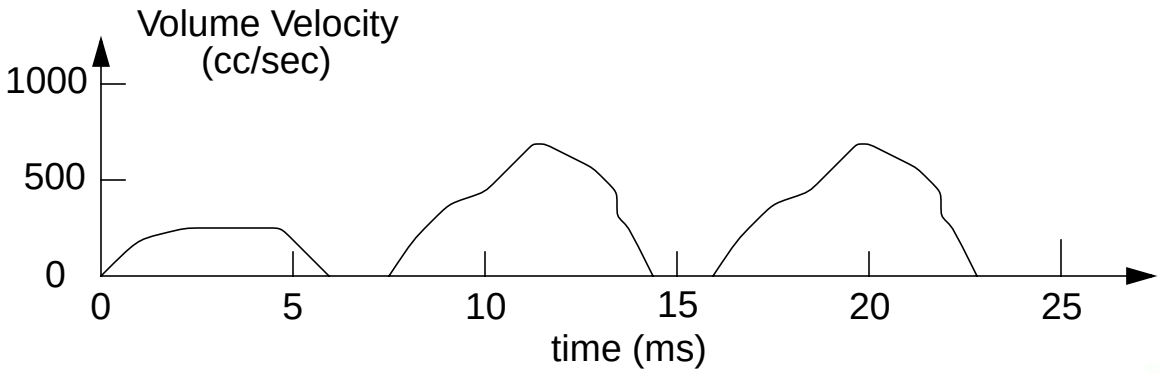
The glottal impedance can be approximated by:

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

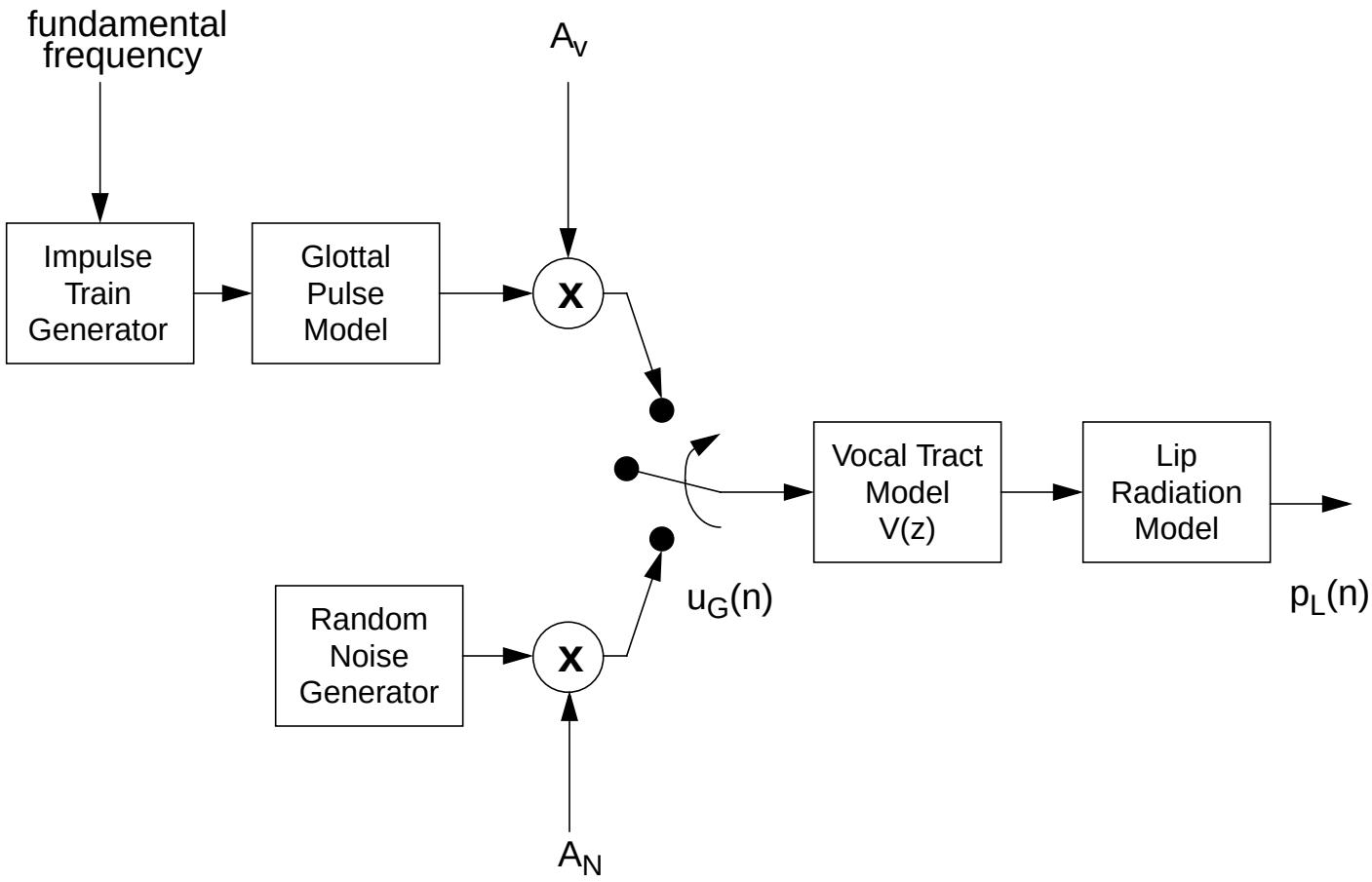
The boundary condition for the volume velocity is:

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

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



The Complete Digital Model (Vocoder)



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:

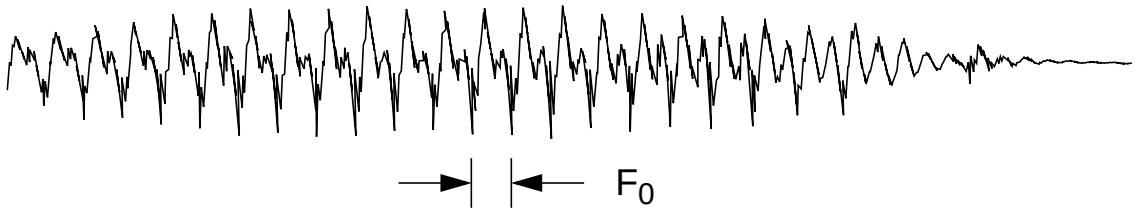
What does the overall spectrum look like?

What happened to the nasal cavity?

What is the form of V(z)?

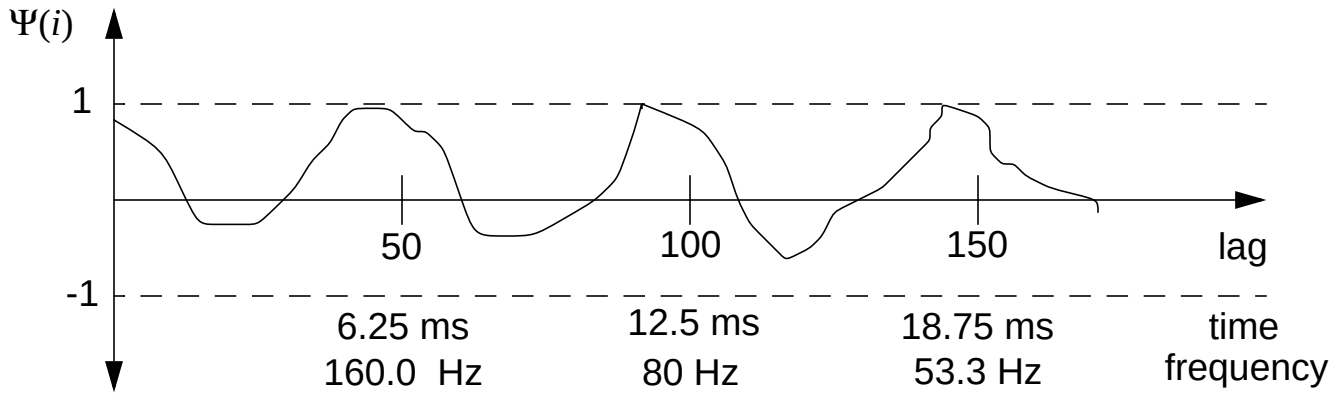
Fundamental Frequency Analysis

How do we determine the fundamental frequency?



We use the (statistical) autocorrelation function:

$$\Psi(i) = \frac{\sum_{n=0}^{N-1} x(n)x(n-i)}{\sqrt{\left(\sum_{n=0}^{N-1} x(n)^2\right)\left(\sum_{n=0}^{N-1} x(n-i)^2\right)}}$$



Other common representations:

Average Magnitude Difference Function (AMDF):

$$\gamma(i) = \sum_{n=0}^{N-1} |x(n) - x(n-i)|$$

Zero Crossing Rate:

$$F_0 = \frac{ZF_s}{2} \qquad Z = \sum_{n=0}^{N-1} |\mathrm{sgn}[x(n)] - \mathrm{sgn}[x(n-1)]|$$

Session II:

Basic Signal Processing Concepts

The Sampling Theorem and Normalized Time/Frequency

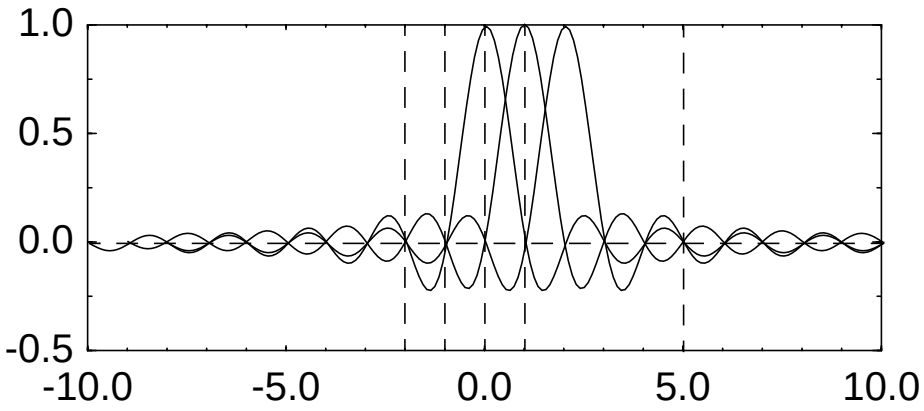
If the highest frequency contained in an analog signal, $x_a(t)$, is $F_{max} = B$ and the signal is sampled at a rate $F_s > 2F_{max} = 2B$, then $x_a(t)$ can be EXACTLY recovered from its sample values using the interpolation function:

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

$x_a(t)$ may be expressed as:

$$x_a(t) = \sum_{n=-\infty}^{\infty} x_a(\frac{n}{F_s}) g(t - \frac{n}{F_s})$$

where $x_a(\frac{n}{F_s}) = x_a(nT) = x(n)$.



Given a continuous signal:

$$x(t) = A\cos(2\pi ft + \theta) ,$$

A discrete-time sinusoid may be expressed as:

$$x(n) = A\cos\left(2\pi f\left(\frac{n}{f_s}\right) + \theta\right),$$

which, after regrouping, gives:

$$x(n) = A\cos(\omega n + \theta) ,$$

where $\omega = 2\pi\left(\frac{f}{f_s}\right)$, and is called normalized radian frequency and n represents normalized time.

Transforms

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

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

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

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

The Fourier transform may be viewed as the z-transform evaluated around the unit circle.

The Discrete Fourier Transform (DFT) is defined as a sampled version of the Fourier shown above:

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

The inverse DFT is given by:

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

The Fast Fourier Transform (FFT) is simply an efficient computation of the DFT.

The Discrete Cosine Transform (DCT) is simply a DFT of a signal that is assumed to be real and even (real and even signals have real spectra!):

$$X(k) = x(0) + 2 \sum_{n=1}^{N-1} x(n) \cos 2\pi kn/N, \qquad k = 0, 1, 2, \ldots, N-1$$

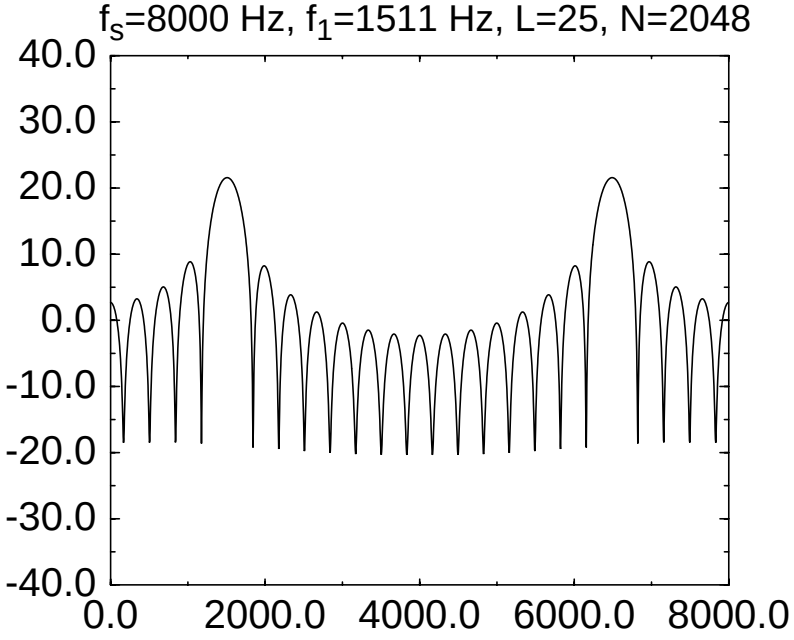
Note that these are not the only transforms used in speech processing (wavelets, fractals, Wigner distributions, etc.).

Time-Domain Windowing

Let $\{x(n)\}$ denote a finite duration segment of a signal:

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

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



Popular Windows

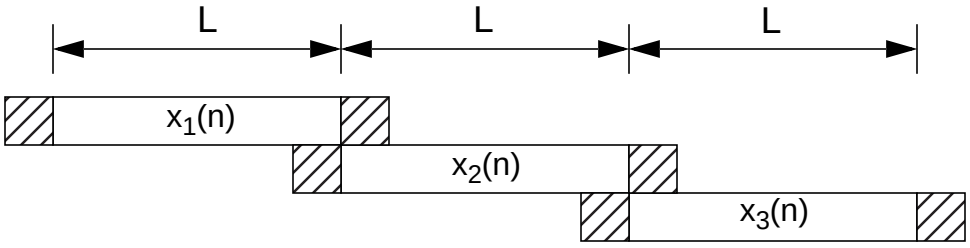
Generalized Hanning: $$w_H(k) = w(k)\left[\alpha + (1-\alpha)\cos\left(\frac{2\pi}{N}k\right)\right] \qquad 0 < \alpha < 1$$

$\alpha = 0.54$, *Hamming window*

$\alpha = 0.50$, *Hanning window*

Frame-Based Analysis With Overlap

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



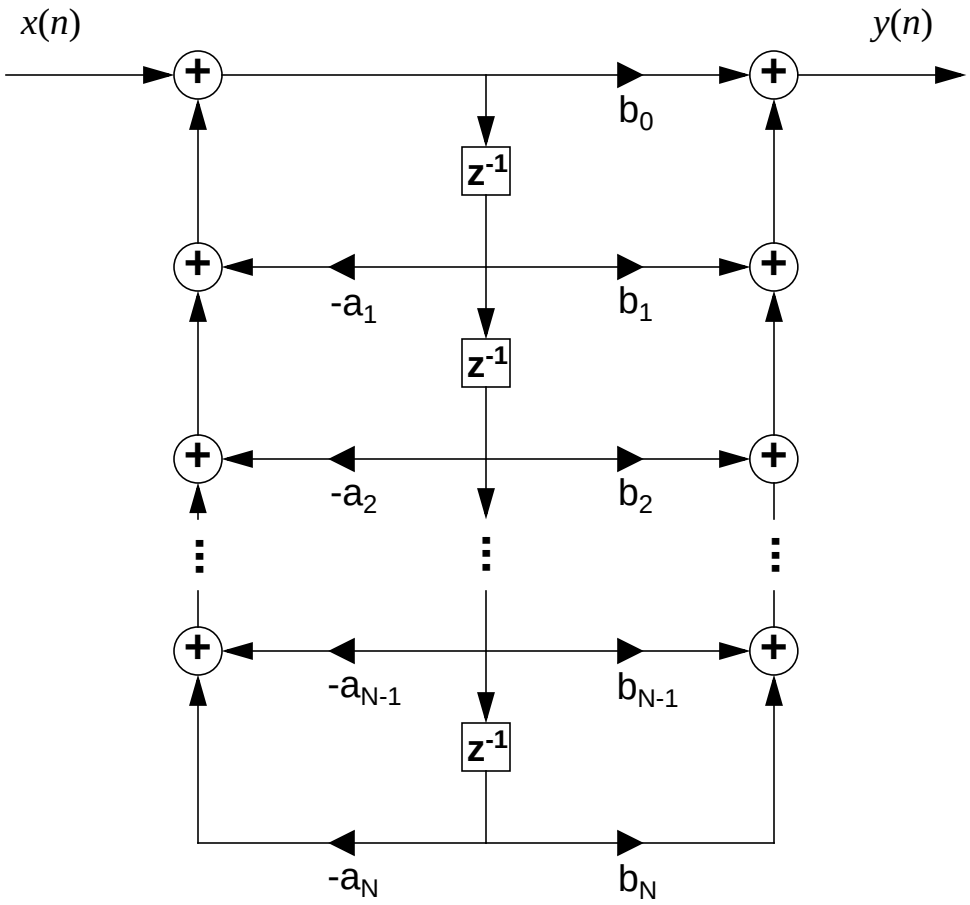
Difference Equations, Filters, and Signal Flow Graphs

A linear time-invariant system can be characterized by a constant-coefficient difference equations:

$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k)$$

Is this system linear if the coefficients are time-varying?

Such systems can be implemented as signal flow graphs:



Minimum Phase, Maximum Phase, MIxed Phase, and Speech Perception

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 $|H(\omega)|$, 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 $|H(\omega)|$.

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 processing?

Probability Spaces

A formal definition of probability involves the specification of:

- a sample space

  The sample space, S, is the set of all possible outcomes, plus the null outcome. Each element in S is called a sample point.

- a field (or algebra)

  The field, or algebra, is a set of subsets of S closed under complementation and union (recall Venn diagrams).

- and a probability measure.

  A probability measure obeys these axioms:

  1. $P(S) = 1$ (implies probabilities less than one)
  2. $P(A) \geq 0$
  3. For two mutually exclusive events:

$$P(A \cup B) = P(A, B) = P(A) + P(B)$$

Two events are said to be statistically independent if:

$$P(A \cap B) = P(A)P(B)$$

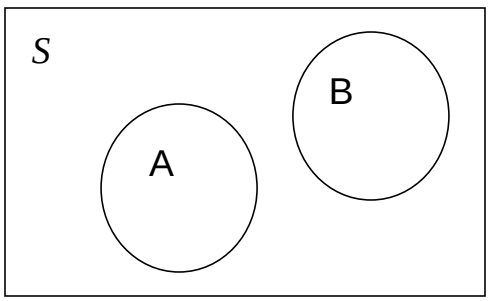
The conditional probability of B given A is:

$$P(B|A) = \frac{P(B \cap A)}{P(A)}$$

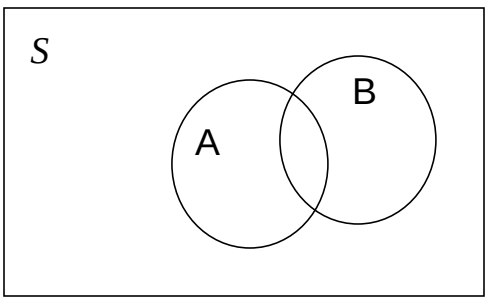
Hence,

$$P(B \cap A) = P(B|A)P(A)$$

Mutually Exclusive



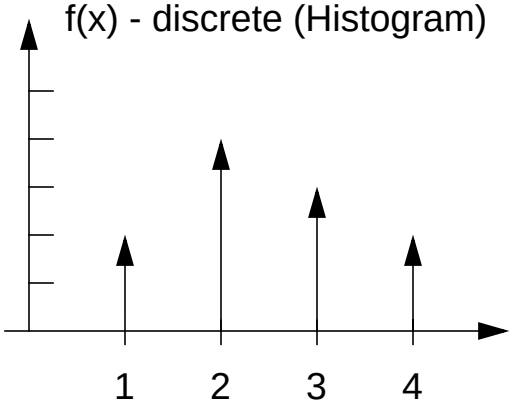
$P(B \cap A) = P(B|A)P(A)$



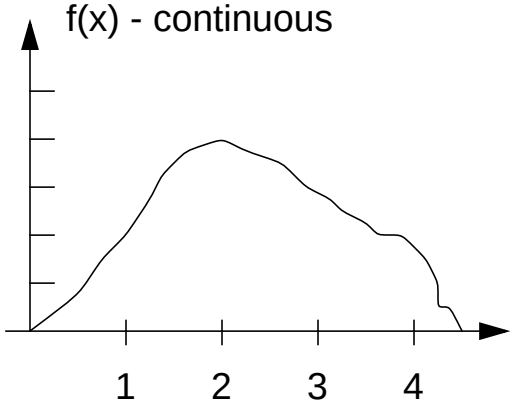
Functions of Random Variables

Probability Density Functions:

f(x) - discrete (Histogram)

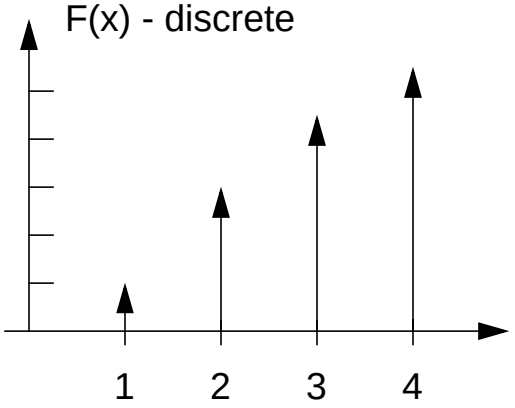


f(x) - continuous

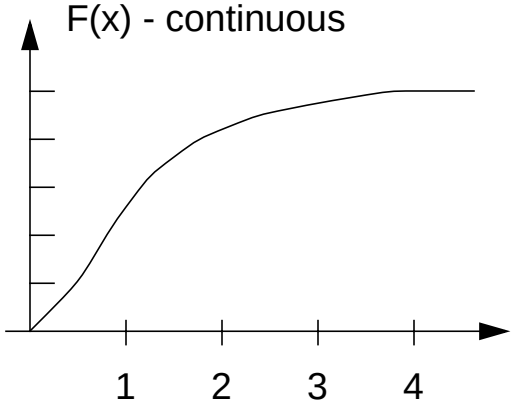


Cumulative Distributions:

F(x) - discrete

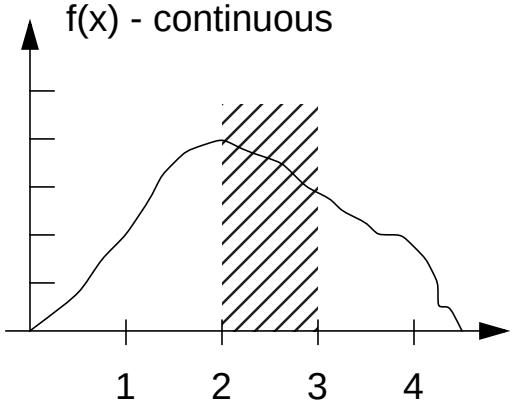


F(x) - continuous



Probability of Events:

f(x) - continuous



$$P(2 < x \leq 3) = \int_{2}^{3} f(x)dx = F(3) - F(2)$$

Important Probability Density Functions

Uniform (Unix rand function):

$$f(x) = \begin{cases} \dfrac{1}{b-a} & a < x \le b \\ 0 & elsewhere \end{cases}$$

Gaussian:

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

Laplacian (speech signal amplitude, durations):

$$f(x) = \frac{1}{\sqrt{2\sigma^2}} \exp\left\{\frac{-\sqrt{2}|x|}{\sigma}\right\}$$

Gamma (durations):

$$f(x) = \frac{\sqrt{k}}{2\sqrt{\pi|x|}} \exp\{-k|x|\}$$

We can extend these concepts to N-dimensional space. For example:

$$P(A \in A_x | B \in A_y) = \frac{P(A,B)}{P(B)} = \frac{\int_{A_x}\int_{A_y} f(x,y)\,dy\,dx}{\int_{A_y} f(y)\,dy}$$

Two random variables are statistically independent if:

$$P(A,B) = P(A)P(B)$$

This implies:

$$f_{xy}(x,y) = f_x(x)f_y(y) \quad \text{and} \quad F_{xy}(x,y) = F_x(x)F_y(y)$$

Mixture Distributions

A simple way to form a more generalizable pdf that still obeys some parametric shape is to use a weighted sum of one or more pdfs. We refer to this as a mixture distribution.

If we start with a Gaussian pdf:

$$f_i(x) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left\{\frac{-(x-\mu_i)^2}{2\sigma_i^2}\right\}$$

The most common form is a Gaussian mixture:

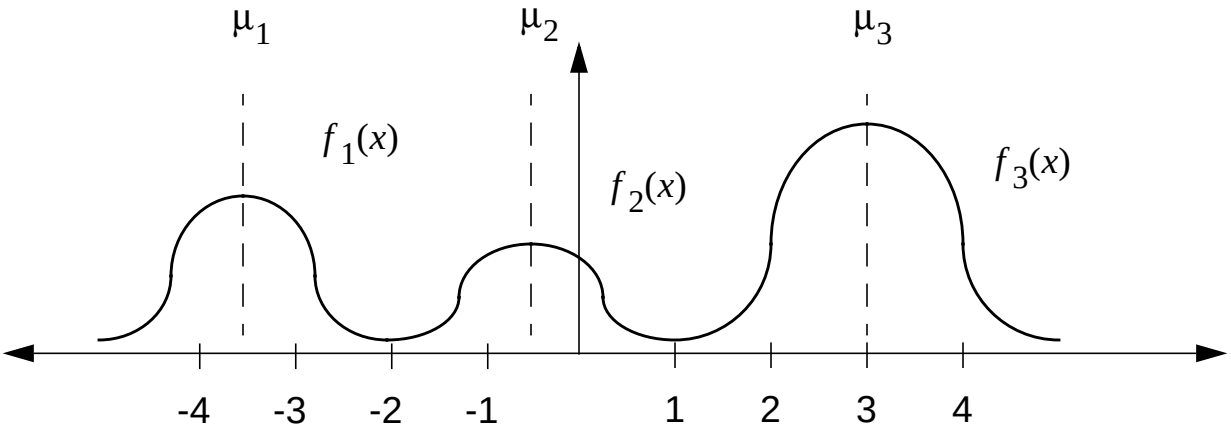
$$f(x) = \sum_{i=1}^{N} c_i f_i(x)$$

where

$$\sum_{i=1}^{N} c_i = 1$$

Obviously this can be generalized to other shapes.

Such distributions are useful to accommodate multimodal behavior:



Derivation of the optimal coefficients, however, is most often a nonlinear optimization problem.

Expectations and Moments

The statistical average of a scalar function, $g(x)$, of a random variable is:

$$E[g(x)] = \sum_{i=1}^{\infty} x_i P(x = x_i) \quad \text{and} \quad E[g(x)] \equiv \int_{-\infty}^{\infty} g(x) f(x) dx$$

The central moment is defined as:

$$E[(x-\mu)^i] = \int_{-\infty}^{\infty} (x-\mu)^i f(x) dx$$

The joint central moment between two random variables is defined as:

$$E[(x-\mu_x)^i (y-\mu_y)^k] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x-\mu_i)^i (y-\mu_y)^k f(x,y) dx dy$$

We can define a correlation coefficient between two random variables as:

$$\rho_{xy} = \frac{c_{xy}}{\rho_x \rho_y}$$

We can extend these concepts to a vector of random variables:

$$\bar{x} = [x_1, x_2, \ldots, x_N]^T$$

$$f_{\bar{x}}(\bar{x}) = \frac{1}{N/2\sqrt{2\pi}\sqrt{|\boldsymbol{C}|}} \exp\left\{-\frac{1}{2}(\bar{x}-\bar{\mu})^T \boldsymbol{C}_{\bar{x}}^{-1}(\bar{x}-\bar{\mu})\right\}$$

What is the difference between a random vector and a random process?

What does wide sense stationary mean? strict sense stationary?

What does it mean to have an ergodic random process?

How does this influence our signal processing algorithms?

Correlation and Covariance of WSS Random Processes

For a signal, $x(n)$, we can compute the following useful quantities:

Autocovariance:

$$\begin{aligned} c(i,j) &= E[(x(n-i)-\mu_i)(x(n-j)-\mu_j)] \\ &= E[x(n-i)x(n-j)] - \mu_i\mu_j \\ &= \frac{1}{N}\sum_{n=0}^{N-1} x(n-i)x(n-j) - \left(\frac{1}{N}\sum_{n=0}^{N-1} x(n-i)\right)\left(\frac{1}{N}\sum_{n=0}^{N-1} x(n-j)\right) \end{aligned}$$

If a random process is wide sense stationary:

$$c(i,j) = c(|i-j|, 0)$$

Hence, we define a very useful function known as the autocorrelation:

$$r(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)x(n-k)$$

If $x(n)$ is zero mean WSS:

$$c(i,j) = r(|i-j|)$$

What is the relationship between the autocorrelation and the spectrum:

$$DFT\{r(k)\} = |X(k)|^2$$

For a linear time-invariant system, $h(n)$:

$$DFT\{r_y(k)\} = |DFT\{h(n)\}|^2 DFT\{r_x(k)\}$$

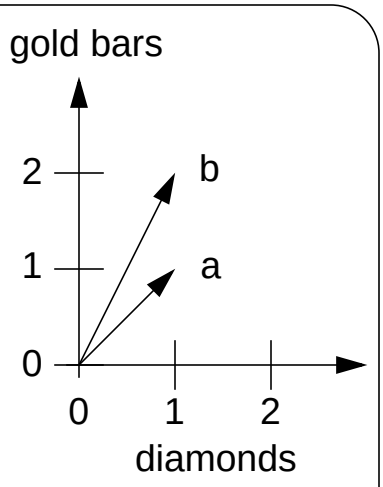
The notion of random noise is central to signal processing:

- white noise?
- Gaussian white noise?
- zero-mean Gaussian white noise?
- colored noise?

Therefore, we now embark upon one of the last great mysteries of life:

- How do we compare two random vectors?

Distance Measures



What is the distance between pt. a and pt. b?

The N-dimensional real Cartesian space, denoted $\Re^N$ is the collection of all N-dimensional vectors with real elements. A metric, or distance measure, is a real-valued function with three properties:

$\forall \bar{x}, \bar{y}, \bar{z} \in \Re^N$:

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,

$$d_\infty(\bar{x}, \bar{y}) = \max_k |x_k - y_k|$$

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:

$$\bar{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_k \end{bmatrix}, \bar{y} = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_k \end{bmatrix}, \text{ and } \underline{W} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1k} \\ w_{21} & w_{22} & \cdots & w_{2k} \\ \cdots & \cdots & \cdots & \cdots \\ w_{k1} & w_{k2} & \cdots & w_{kk} \end{bmatrix}.$$

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):

$$\begin{aligned} d_2(\bar{x}_m, \bar{y})^2 &= \sum_{j=1}^{k} (x_{m_j} - y_j)^2 = \sum_{j=1}^{k} x_{m_j}{}^2 - 2x_{m_j}y_j + y_j^2 \\ &= \left|\left|\bar{x}_m\right|\right|^2 - 2\bar{x}_m \bullet \bar{y} + ||\bar{y}||^2 \\ &= C_m + C_y - 2\bar{x}_m \bullet \bar{y} \end{aligned}$$

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 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:

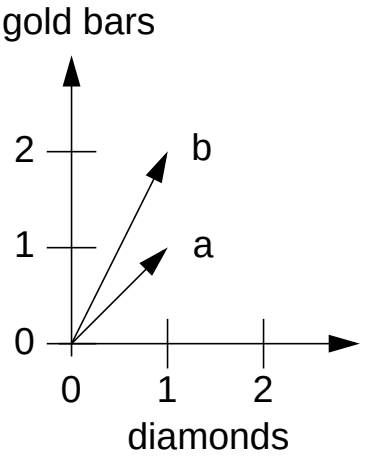
$$\bar{z} = \underline{V}\bar{x}$$

System 1:

$$\beta_1 = 1\hat{i} + 0\hat{j} \text{ and } \beta_2 = 0\hat{i} + 1\hat{j}$$

$$\bar{a} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad \bar{b} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

$$d_2(\bar{a}, \bar{b}) = \sqrt{0^2 + 1^2} = 1$$

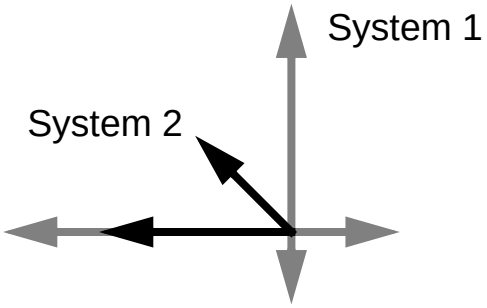


System 2:

$$\gamma_1 = -2\hat{i} + 0\hat{j} \text{ and } \gamma_1 = -1\hat{i} + 1\hat{j}$$

$$\bar{a} = \begin{bmatrix} -2 & 0 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} -1 \\ 1 \end{bmatrix} \qquad \bar{b} = \begin{bmatrix} -2 & 0 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} -\frac{3}{2} \\ 2 \end{bmatrix}$$

$$d_2(\bar{a}, \bar{b}) = \sqrt{\left(-1 - \left(-\frac{3}{2}\right)\right)^2 + (1-2)^2} = \sqrt{\frac{5}{4}}$$



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?

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):

$$\underline{C} = \underline{\Phi}\underline{\Lambda}\underline{\Phi}^T$$

where Φ denotes a matrix of eigenvectors of $\underline{C}$ and $\underline{\Lambda}$ denotes a diagonal matrix whose elements are the eigenvalues of $\underline{C}$. Consider:

$$\bar{z} = \underline{\Lambda}^{-1/2}\underline{\Phi}\bar{x}$$

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, $\underline{\Lambda}$. Why? This is known as variance-weighting.

"Noise-Reduction"

The prewhitening transform, $\bar{z} = \underline{\Lambda}^{-1/2}\underline{\Phi}\bar{x}$, is normally created as a $k \times k$ matrix in which the eigenvalues are ordered from largest to smallest:

$$\begin{bmatrix} z_1 \\ z_2 \\ \dots \\ z_k \end{bmatrix} = \begin{bmatrix} \lambda_1^{-1/2} & & & \\ & \lambda_2^{-1/2} & & \\ & & \dots & \\ & & & \lambda_k^{-1/2} \end{bmatrix} \begin{bmatrix} v_{11} & v_{12} & \dots & v_{13} \\ v_{21} & v_{22} & \dots & v_{2k} \\ \dots & \dots & \dots & \dots \\ v_{k1} & v_{k2} & \dots & v_{kk} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_k \end{bmatrix}$$

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 Procedures

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.

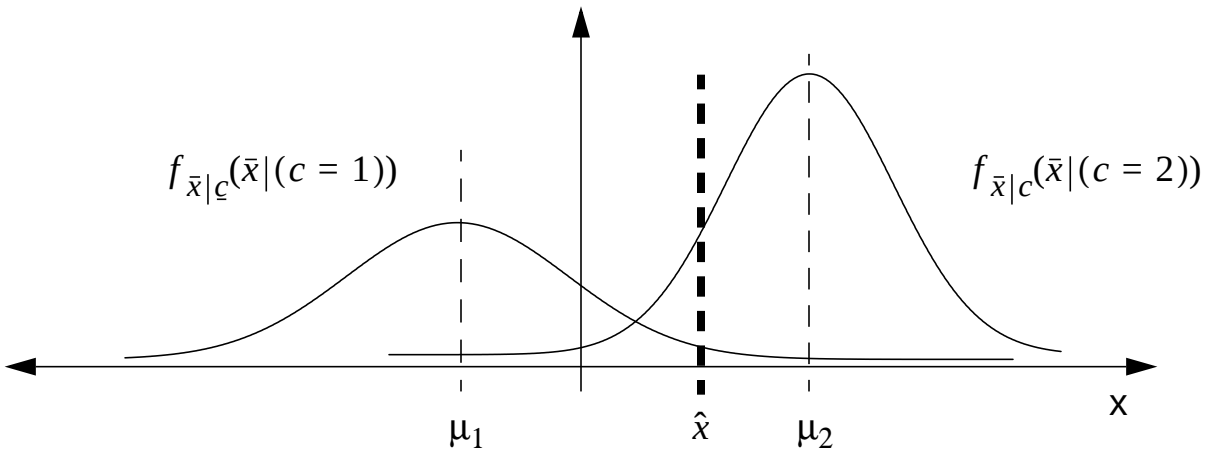
Another method, known as the QR decomposition, factors the covariance matrix into a series of transformations:

$$\underline{C} = \underline{Q}\underline{R}$$

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:

$$c^* = \underset{c}{\text{argmax}} \; P((c = \hat{c})|(\bar{x} = \hat{\bar{x}}))$$

However, we can't estimate this probability directly from the training data. Hence, we consider:

$$c^* = \underset{c}{\text{argmax}} \; P((\bar{x} = \hat{\bar{x}})\big|(c = \hat{c}))$$

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

$$P((\bar{x} = \hat{\bar{x}})\big|(c = \hat{c})) = \frac{P((c = \hat{c}),(\hat{x} = \hat{\bar{x}}))}{P(c = \hat{c})}$$

from which we have

$$P((c = \hat{c})|(\bar{x} = \hat{\bar{x}})) = \frac{P((\bar{x} = \hat{\bar{x}})\big|(c = \hat{c}))P(c = \hat{c})}{P(\bar{x} = \hat{\bar{x}})}$$

Clearly, the choice of c that maximizes the right side also maximizes the left side. Therefore,

$$c^* = \underset{c}{\text{argmax}} \left[P((\bar{x} = \hat{\bar{x}}) \middle| (c = \hat{c}))\right]$$

$$= \underset{c}{\text{argmx}}[P((\bar{x} = \hat{\bar{x}}) \middle| (c = \hat{c}))P(c = \hat{c})]$$

if the class probabilities are equal,

$$c^* = \underset{c}{\text{argmx}}[P((\bar{x} = \hat{\bar{x}}) \middle| (c = \hat{c}))]$$

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:

$$f_{\bar{x}|c}(x_1,\ldots,x_N \middle| c) = f_{\bar{x}|c}(\hat{\bar{x}} \middle| \hat{c})$$

$$= \frac{1}{\sqrt{2\pi \left| C_{\bar{x}|c} \right|}} \exp\left\{ -\frac{1}{2} (\hat{\bar{x}} - \bar{\mu}_{\bar{x}|c})^T \underline{C}_{\bar{x}|c}^{-1} (\hat{\bar{x}} - \bar{\mu}_{\hat{\bar{x}}|c}) \right\}$$

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:

$$c^* = \underset{c}{\text{argmin}} \left[(\hat{\bar{x}} - \bar{\mu}_{\bar{x}|c})^T \underline{C}_{\bar{x}|c}^{-1} (\hat{\bar{x}} - \bar{\mu}_{\hat{\bar{x}}|c}) + \ln\left\{ \left| \underline{C}_{\bar{x}|c}^{-1} \right| \right\} \right]$$

(Note that the maximization became a minimization.)

We can define a distance measure based on this as:

$$d_{ml}(\bar{x}, \bar{\mu}_{\bar{x}|c}) = (\hat{\bar{x}} - \bar{\mu}_{\bar{x}|c})^T \underline{C}_{\bar{x}|c}^{-1} (\hat{\bar{x}} - \bar{\mu}_{\hat{\bar{x}}|c}) + \ln\left\{ \left| \underline{C}_{\bar{x}|c}^{-1} \right| \right\}$$

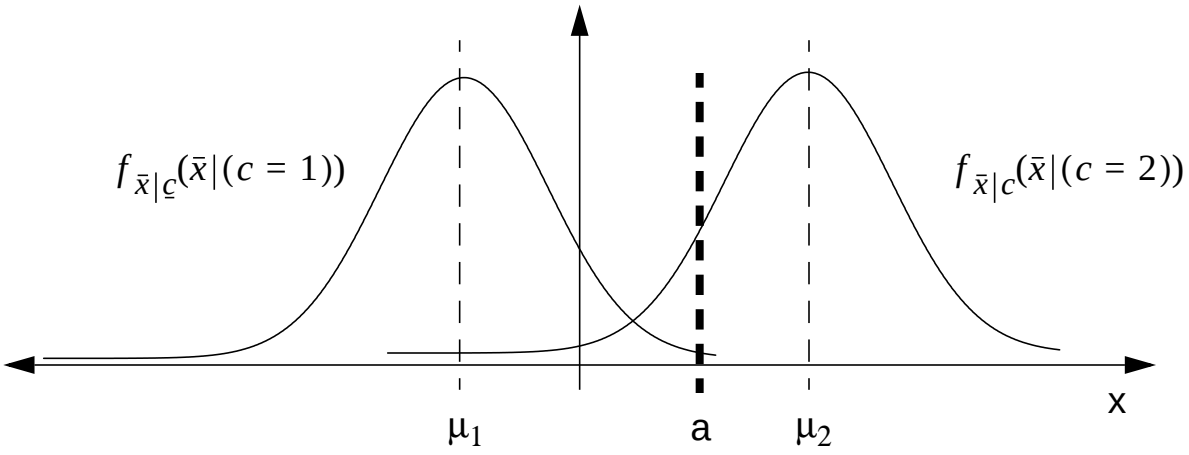
Note that the distance is conditioned on each class mean and covariance. This is why “generic” distance comparisons are a joke.

If the mean and covariance are the same across all classes, this expression simplifies to:

$$d_M(\bar{x}, \bar{\mu}_{\bar{x}|c}) = (\hat{\bar{x}} - \bar{\mu}_{\bar{x}|c})^T \underline{C}_{\bar{x}|c}^{-1} (\hat{\bar{x}} - \bar{\mu}_{\hat{\bar{x}}|c})$$

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,

$$\begin{array}{lll} if & x < a & x \in (c = 1) \\ else & & x \in (c = 2) \end{array}$$

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 equiprobable?

Probabilistic Distance Measures

How do we compare two probability distributions to measure their overlap?

Probabilistic distance measures take the form:

$$J = \int_{-\infty}^{\infty} g\{ f_{\bar{x}|c}(\hat{\bar{x}}|\hat{c}), P(c = \hat{c}), \hat{c} = 1, 2, \ldots, K \} d\hat{\bar{x}}$$

where

1. J is nonnegative
2. J attains a maximum when all classes are disjoint
3. J=0 when all classes are equiprobable

Two important examples of such measures are:

(1) Bhattacharyya distance:

$$J_B = -\ln\left[\int_{-\infty}^{\infty} \sqrt{f_{\bar{x}|c}(\hat{\bar{x}}|1) f_{\bar{x}|c}(\hat{\bar{x}}|2)}\, d\hat{\bar{x}} \right]$$

(2) Divergence

$$J_D = \int_{-\infty}^{\infty} [f_{\bar{x}|c}(\hat{\bar{x}}|1) - f_{\bar{x}|c}(\hat{\bar{x}}|2)] \ln\left[\frac{f_{\bar{x}|c}(\hat{\bar{x}}|1)}{f_{\bar{x}|c}(\hat{\bar{x}}|2)} \right] d\hat{\bar{x}}$$

Both reduce to a Mahalanobis-like distance for the case of Gaussian vectors with equal class covariances.

Such metrics will be important when we attempt to cluster feature vectors and acoustic models.

Probabilistic Dependence Measures

A probabilistic dependence measure indicates how strongly a feature is associated with its class assignment. When features are independent of their class assignment, the class conditional pdf's are identical to the mixture pdf:

$$f_{\bar{x}|c}(\hat{\bar{x}}|\hat{c}) = f_{\bar{x}}(\hat{\bar{x}}) \qquad \forall c$$

When their is a strong dependence, the conditional distribution should be significantly different than the mixture. Such measures take the form:

$$J = \int_{-\infty}^{\infty} g\{ f_{\bar{x}|c}(\hat{\bar{x}}|\hat{c}), f_{\bar{x}}(\hat{\bar{x}}), P(c = \hat{c}), \hat{c} = 1, 2, \ldots, K \} d\hat{\bar{x}}$$

An example of such a measure is the average mutual information:

$$M_{avg}(c, \hat{\bar{x}}) = \sum_{c=1}^{K} P(c = \hat{c}) \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} f_{\bar{x}|c}(\hat{\bar{x}}|\hat{c}) \log \frac{f_{\bar{x}|c}(\hat{\bar{x}}|\hat{c})}{f_{\bar{x}}(\hat{\bar{x}})} d\hat{\bar{x}}$$

The discrete version of this is:

$$M_{avg}(c, \hat{\bar{x}}) = \sum_{c=1}^{K} P(c = \hat{c}) \sum_{i=1}^{L} P(\bar{x} = \bar{x}_l) \log_2 \frac{P(\bar{x} = \bar{x}_l | c = \hat{c})}{P(\bar{x} = \bar{x}_l)}$$

Mutual information is closely related to entropy, as we shall see shortly.

Such distance measures can be used to cluster data and generate vector quantization codebooks. A simple and intuitive algorithms is known as the K-means algorithm:

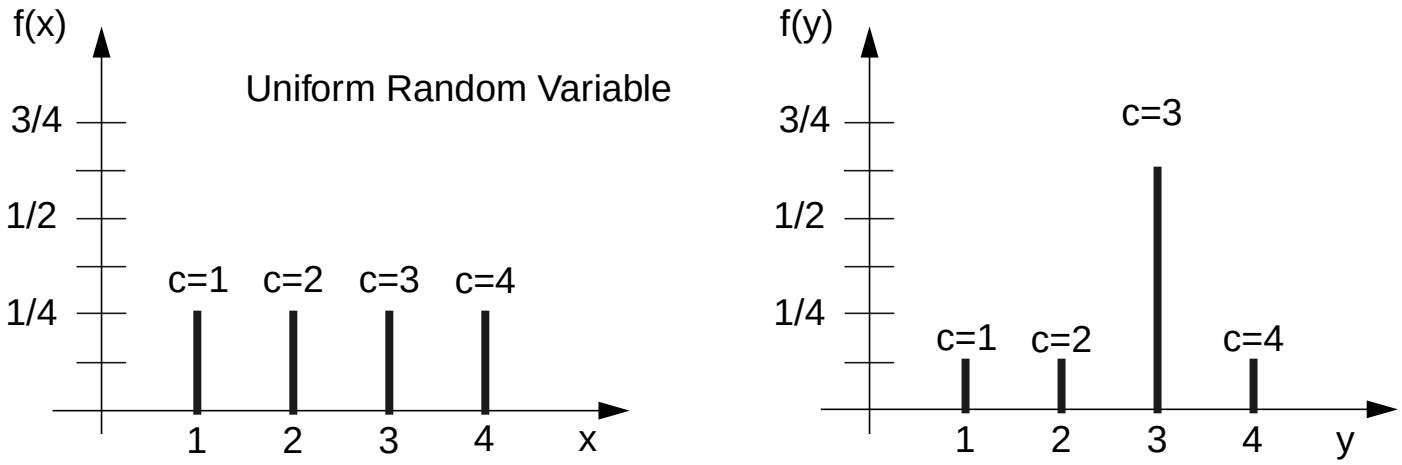
Initialization: Choose K centroids

Recursion:

1. Assign all vectors to their nearest neighbor.
2. Recompute the centroids as the average of all vectors assigned to the same centroid.
3. Check the overall distortion. Return to step 1 if some distortion criterion is not met.

What Is Information? (When not to bet at a casino...)

Consider two distributions of discrete random variables:



Which variable is more unpredictable?

Now, consider sampling random numbers from a random number generator whose statistics are not known. The more numbers we draw, the more we discover about the underlying distribution. Assuming the underlying distribution is from one of the above distributions, how much more information do we receive with each new number we draw?

The answer lies in the shape of the distributions. For the random variable **x**, each class is equally likely. Each new number we draw provides the maximum amount of information, because, on the average, it will be from a different class (so we discover a new class with every number). On the other hand, for **y**, chances are, **c=3** will occur 5 times more often than the other classes, so each new sample will not provide as much information.

We can define the information associated with each class, or outcome, as:

$$I(c = \hat{c}) \equiv \log_2 \frac{1}{P(c = \hat{c})} = -\log_2 P(c = \hat{c})$$

Since $0 \le P(x) \le 1$, information is a positive quantity. A base 2 logarithm is used so that K discrete outcomes can be measured in $2^M = K$ bits. For the distributions above,

$$I_x(c = 1) = (-1)\log_2 (1/4) = 2 \; bits \qquad I_y(c = 1) = (-1)\log_2 \left(\frac{1}{8}\right) = 3 \; bits$$

Huh??? Does this make sense?

What is Entropy?

Entropy is the expected (average) information across all outcomes:

$$H(c) \equiv E[I(c)] = -\sum_{k=1}^{K} P(c = c_k) \log_2 P(c = c_k)$$

Entropy using $\log_2$ is also measured in bits, since it is an average of information.

For example,

$$H_x = -\left[\sum_{k=1}^{4} \left(\frac{1}{4}\right) \log_2 \left(\frac{1}{4}\right)\right] = 2.0 \text{ bits} \qquad H_y = -\left[\sum_{k=1}^{3} \left(\frac{1}{8}\right) \log_2 \left(\frac{1}{8}\right) + \left(\frac{5}{8}\right) \log_2 \left(\frac{5}{8}\right)\right] = 0.8 \text{ bits}$$

We can generalize this to a joint outcome of N random vectors from the same distribution, which we refer to as the *joint entropy*:

$$H[\bar{x}(1),\ldots,\bar{x}(N)] = -\sum_{l_1=1}^{N} \cdots \sum_{l_N}^{N} P[(\bar{x}(1) = \bar{x}_{l_1}), \ldots, (\bar{x}(N) = \bar{x}_{l_N})] \times \log_2 P[(\bar{x}(1) = \bar{x}_{l_1}), \ldots, (\bar{x}(N) = \bar{x}_{l_N})]$$

If the random vectors are statistically independent:

$$H[\bar{x}(1),\ldots,\bar{x}(N)] = \sum_{n=1}^{N} H[\bar{x}(n)]$$

If the random vectors are independent and indentically distributed:

$$H[\bar{x}(1),\ldots,\bar{x}(N)] = NH[\bar{x}(1)]$$

We can also define conditional entropy as:

$$H(\bar{x}|\bar{y}) = \sum_{k=1}^{K} \sum_{l=1}^{L} P((\bar{x} = \bar{x}_l)\big|(\bar{y} = \bar{y}_k)) \log_2 [P((\bar{x} = \bar{x}_l)\big|(\bar{y} = \bar{y}_k))]$$

For continuous distributions, we can define an analogous quantity for entropy:

$$H = -\int_{-\infty}^{\infty} f(x) \log_2 f(x)\, dx \quad \text{(bits)}$$

A zero-mean Gaussian random variable has maximum entropy $(\frac{1}{2}\log_2(2\pi e\sigma^2)$.

Why?

Mutual Information

The pairing of random vectors produces less information than the events taken individually. Stated formally:

$$I(\bar{x}, \bar{y}) \leq I(\bar{x}) + I(\bar{y})$$

The shared information between these events is called the mutual information, and is defined as:

$$M(\bar{x}, \bar{y}) \equiv [I(\bar{x}) + I(\bar{y})] - I(\bar{x}, \bar{y})$$

From this definition, we note:

$$\begin{aligned} M(\bar{x}, \bar{y}) &= \log_2 \left[\frac{P(\bar{x}, \bar{y})}{P(\bar{x})P(\bar{y})} \right] \\ &= \log_2 \left[\frac{1}{P(\bar{x})} \right] + \log_2 \left[\frac{1}{\frac{P(\bar{x}, \bar{y})}{P(\bar{y})}} \right] = \log_2 \left[\frac{1}{P(\bar{x})} \right] - \log_2 \left[\frac{1}{P(\bar{x}|\bar{y})} \right] \\ &= I(\bar{x}) - I(\bar{x}|\bar{y}) \\ &= I(\bar{y}) - I(\bar{y}|\bar{x}) \end{aligned}$$

This emphasizes the idea that this is information shared between these two random variables.

We can define the *average mutual information* as the expectation of the mutual information:

$$\begin{aligned} \overline{M}(\bar{x}, \bar{y}) &= E\left[\log_2 \left(\frac{P(\bar{x}, \bar{y})}{P(\bar{x})P(\bar{y})} \right) \right] \\ &= \sum_{k=1}^{K} \sum_{l=1}^{L} P((\bar{x} = \bar{x}_l), (\bar{y} = \bar{y}_k)) \log_2 \left[\frac{P((\bar{x} = \bar{x}_l), (\bar{y} = \bar{y}_k))}{P(\bar{x} = \bar{x}_l)P(\bar{y} = \bar{y}_k)} \right] \end{aligned}$$

Note that:

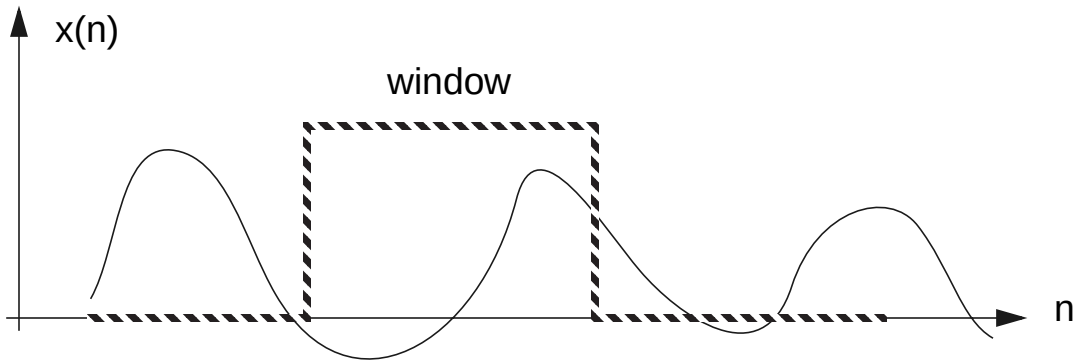
$$\overline{M}(\bar{x}, \bar{y}) = H(\bar{x}) - H(\bar{x}|\bar{y}) = H(\bar{y}|\bar{x}) - H(\bar{y})$$

Also note that if $\bar{x}$ and $\bar{y}$ are independent, then there is no mutual information between them.

Note that to compute mutual information between two random variables, we need a joint probability density function.

How Does Entropy Relate To DSP?

Consider a window of a signal:



What does the sampled z-transform assume about the signal outside the window?

What does the DFT assume about the signal outside the window?

How do these influence the resulting spectrum that is computed?

What other assumptions could we make about the signal outside the window? How many valid signals are there?

How about finding the spectrum that corresponds to the signal that matches the measured signal within the window, and has maximum entropy?

What does this imply about the signal outside the window?

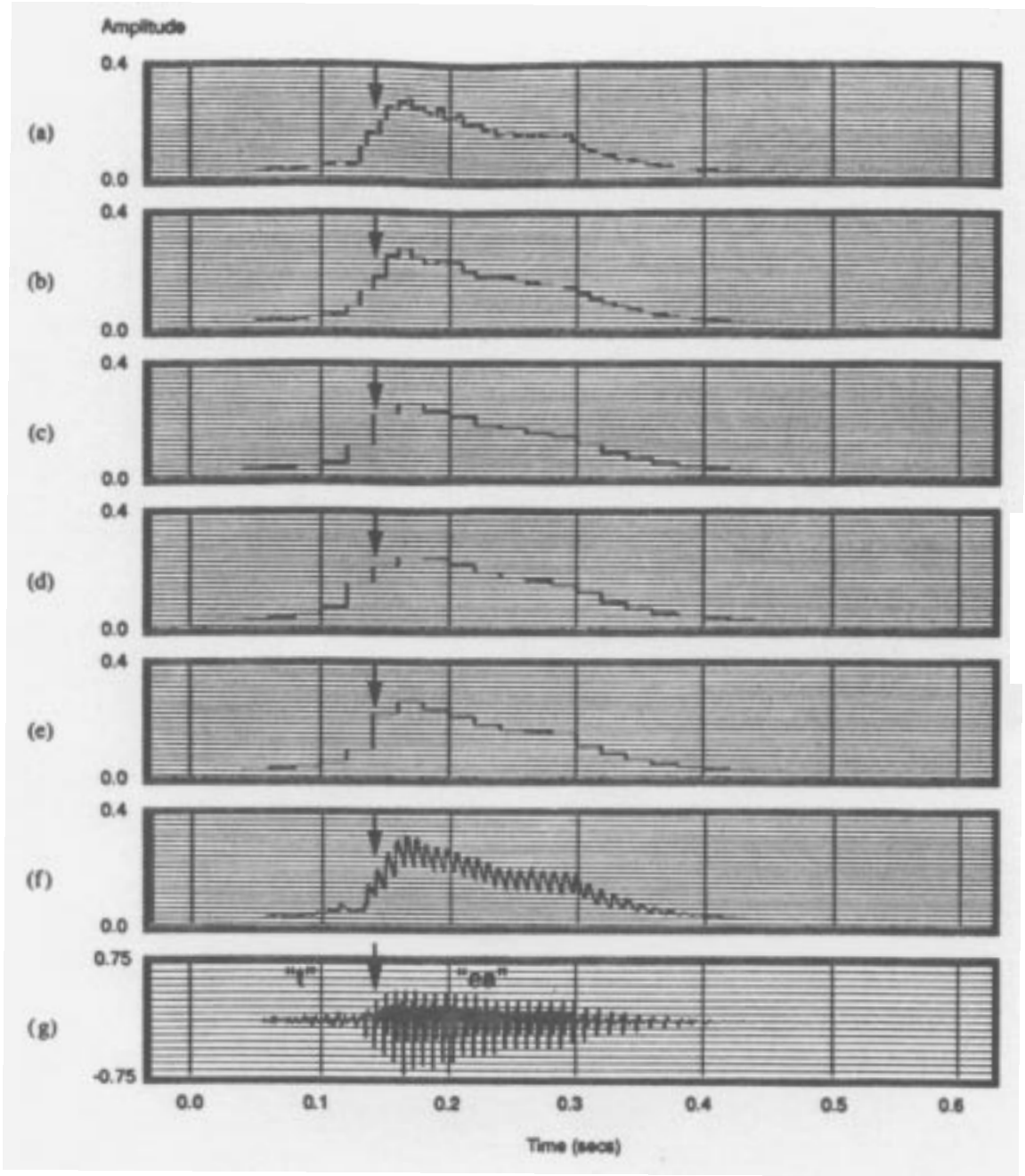
This is known as the principle of maximum entropy spectral estimation. Later we will see how this relates to minimizing the mean-square error.

Session III:

Signal Measurements

Short-Term Measurements

What is the point of this lecture?



$T_f = 5\ ms$
$T_w = 10\ ms$

$T_f = 10\ ms$
$T_w = 20\ ms$

$T_f = 20\ ms$
$T_w = 30\ ms$

$T_f = 20\ ms$
$T_w = 30\ ms$
$Hamm.\ Win.$

$T_f = 20\ ms$
$T_w = 60\ ms$
$Hamm.\ Win.$

Recursive
50 Hz LPF

Speech Signal

Time/Frequency Properties of Windows

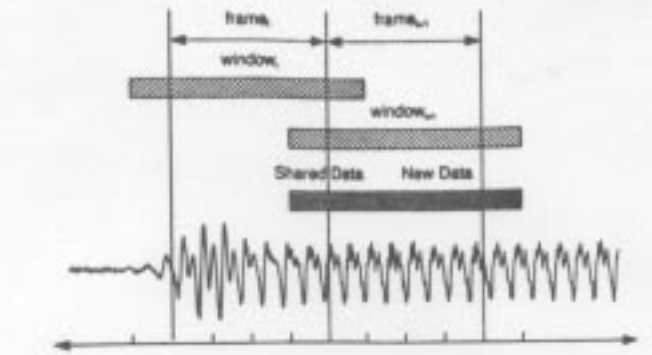
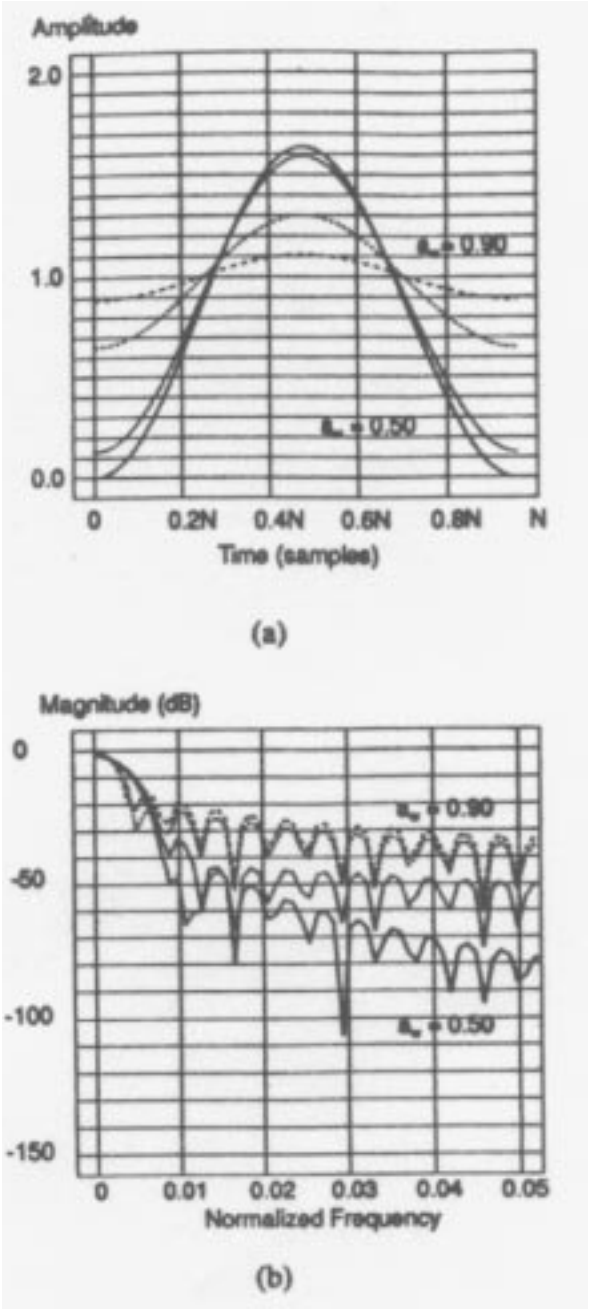


Fig. 6. A frame-based overlapping analysis is depicted. In this case, a 33% overlap is shown. One-third of the data used in each analysis frame is shared with the previous frame. Note that only one-third of the data are unique to the current frame—the remaining two-thirds are shared between adjacent frames.

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



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

$\alpha = 0.54$, *Hamming window*

$\alpha = 0.50$, *Hanning window*

Recursive-in-Time Approaches

Define the short-term estimate of the power as:

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

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:

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

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

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

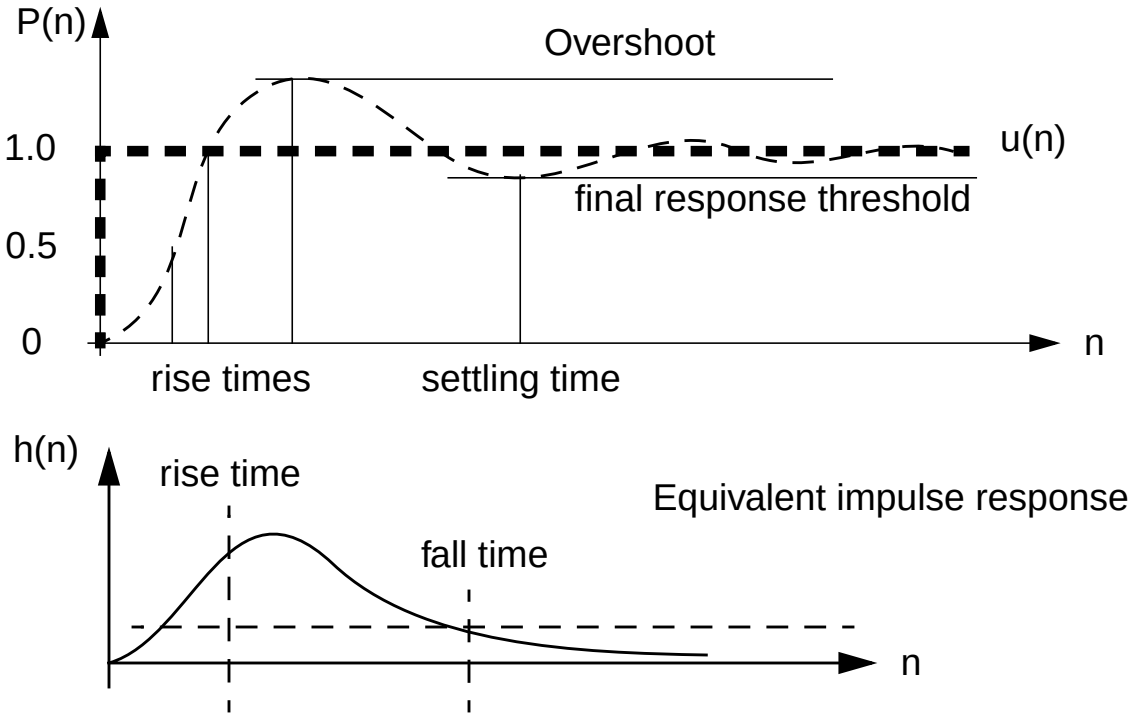
The first-order systems can be related to physical quantities by observing that the system consists of one real pole:

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

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.

Short-Term Autocorrelation and Covariance

Recall our definition of the autocorrelation function:

$$r(k) = \frac{1}{N}\sum_{n=k}^{N-1} s(n)s(n-k) = \frac{1}{N}\sum_{n=0}^{N-1-k} s(n)s(n+k) \qquad 0 < k < p$$

Note:

- this can be regarded as a dot product of $s(n)$ and $s(n+i)$.
- let's not forget preemphasis, windowing, and centering the window w.r.t. the frame, and that scaling is optional.

What would C++ code look like:

array-style:

```
for(k=0; k<M; k++) {
  r[k] = 0.0;
  for(n=0; n<N-k; n++) {
    r[k] += s[n] * s[n+k]
  }
}
```

pointer-style:

```
for(k=0; k<M; k++) {
  *r = 0.0; s = sig; sk = sig + k;
  for(n=0; n<N-i; n++) {
    *r += (s++) * (sk++);
  }
}
```

We note that we can save some multiplications by reusing products:

$$r[3] = s[3](s[0] + s[6]) + s[4](s[1] + s[7]) + \ldots + s[N]s[N-3]$$

This is known as the *factored autocorrelation* computation.
It saves about 25% CPU, replacing multiplications with additions and more complicated indexing.

Similarly, recall our definition of the covariance function:

$$c(k,l) = \frac{1}{N}\sum_{n=p}^{N-1} s(n-k)s(n-l) \qquad 0 < l < k < p$$

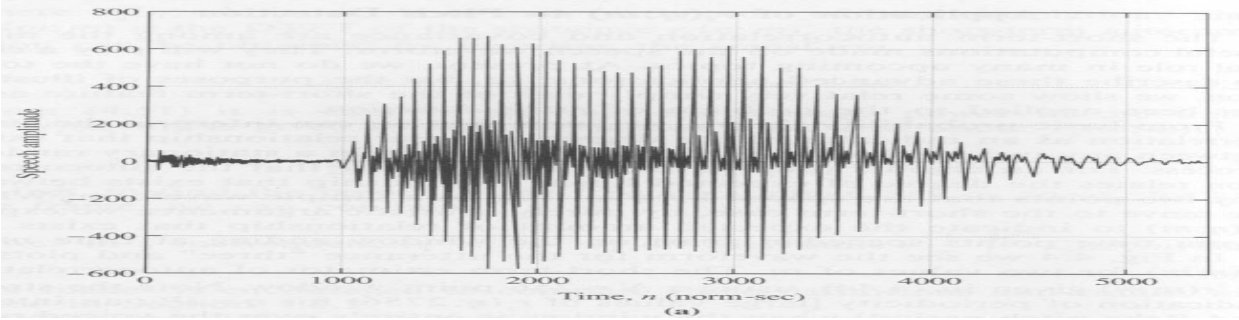
Note:

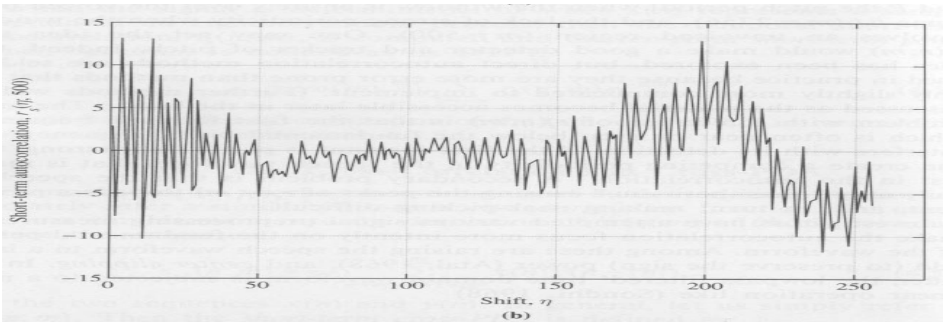
- we use N-p points
- symmetric so that only the $k \geq l$ terms need to be computed

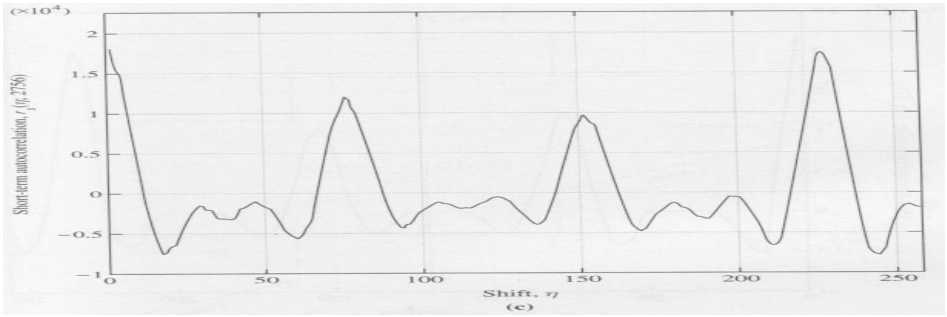
This can be simplified using the recursion:

$$c(k,l) = c(k-1, l-1) + s(p-k)s(p-l) - s(N-k)s(N-l)$$

Example of the Autocorrelation Function







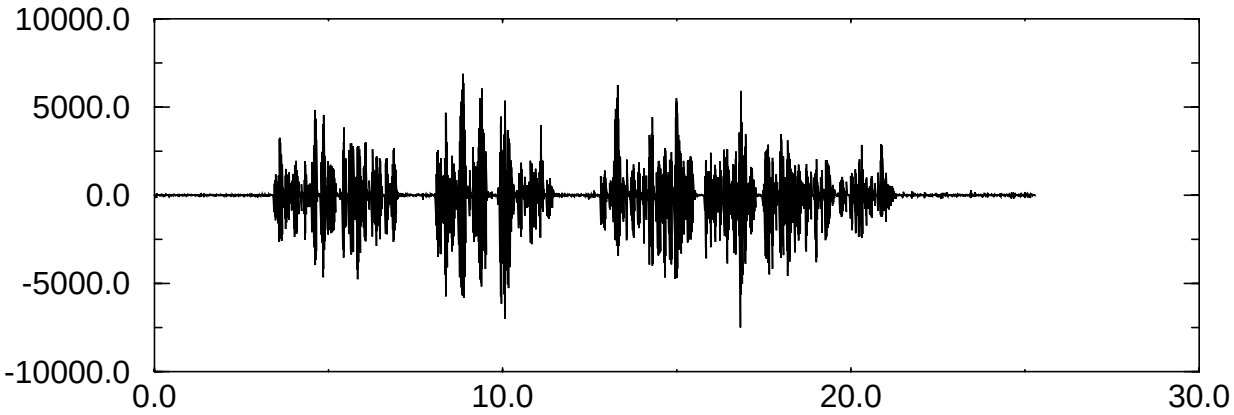
Autocorrelation functions for the word “three” comparing the consonant portion of the waveform to the vowel (256-point Hamming window).

Note:

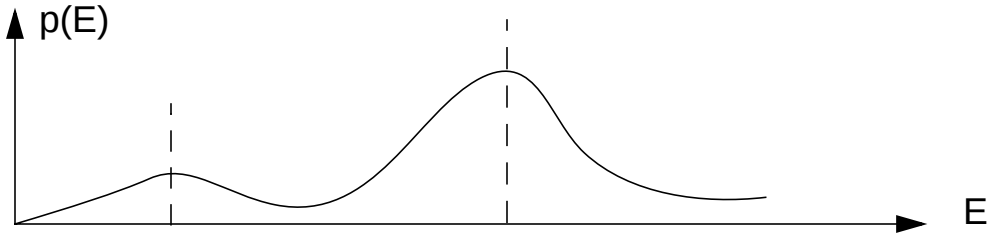
- shape for the low order lags - what does this correspond to?
- regularity of peaks for the vowel — why?
- exponentially-decaying shape — which harmonic?
- what does a negative correlation value mean?

An Application of Short-Term Power: Speech SNR Measurement

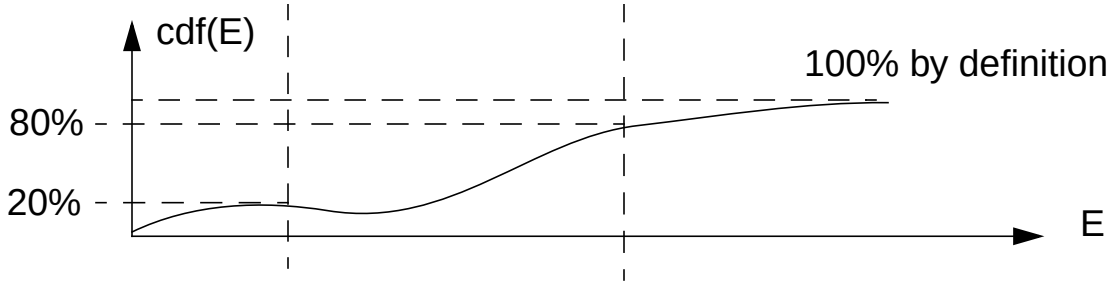
Problem: Can we estimate the SNR from a speech file?



Energy Histogram:



Cumulative Distribution:



Nominal Noise Level Nominal Signal+Noise Level

The SNR can defined as:

$$SNR = 10\log_{10}\frac{E_s}{E_n} = 10\log_{10}\frac{(E_s + E_n) - E_n}{E_n}$$

What percentiles to use?

Typically, 80%/20%, 85%/15%, or 95%/15% are used.

Linear Prediction (General)

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 $1 \le l \le p$ by differentiating E and setting the result equal to zero:

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

Linear Prediction (Cont.)

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 and 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,

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

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,

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

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.

Levinson-Durbin Recursion

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

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

This recursion gives us great insight into the linear prediction process. First, we note that the intermediate variables, $\{k_i\}$, are referred to as *reflection coefficients*.

Example: p=2

$$
\begin{aligned}
E_0 &= r(0) \\
k_1 &= r(1)/r(0) \\
a_1(1) &= k_1 = r(1)/r(0) \\
E_1 &= (1 - k_1^2) E_0 \\
&= \frac{r^2(0) - r^2(1)}{r(0)} \\
k_2 &= \frac{r(2)r(0) - r^2(1)}{r^2(0) - r^2(1)}
\end{aligned}
\qquad
\begin{aligned}
a_2(2) &= k_2 = \frac{r(2)r(0) - r^2(1)}{r^2(0) - r^2(1)} \\
a_2(1) &= a_1(1) - k_2 a_1(1) \\
&= \frac{r(1)r(0) - r(1)r(2)}{r^2(0) - r^2(1)} \\
\alpha_1 &= a_2(1) \\
\alpha_2 &= a_2(2)
\end{aligned}
$$

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

Linear Prediction Coefficient Transformations

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

Predictor to reflection coefficient transformation:

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

$$k_i = a_i(i)$$

$$a_{i-1}(j) = \frac{a_i(j) + k_i a_i(i-j)}{1 - k_i^2} \qquad 1 \le j \le i-1$$

Reflection to predictor coefficient transformation:

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

$$a_i(i) = k_i$$

$$a_i(j) = a_{i-1}(j) - k_i a_{i-1}(i-j) \qquad 1 \le j \le i-1$$

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

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

(1) $|k_i| < 1$

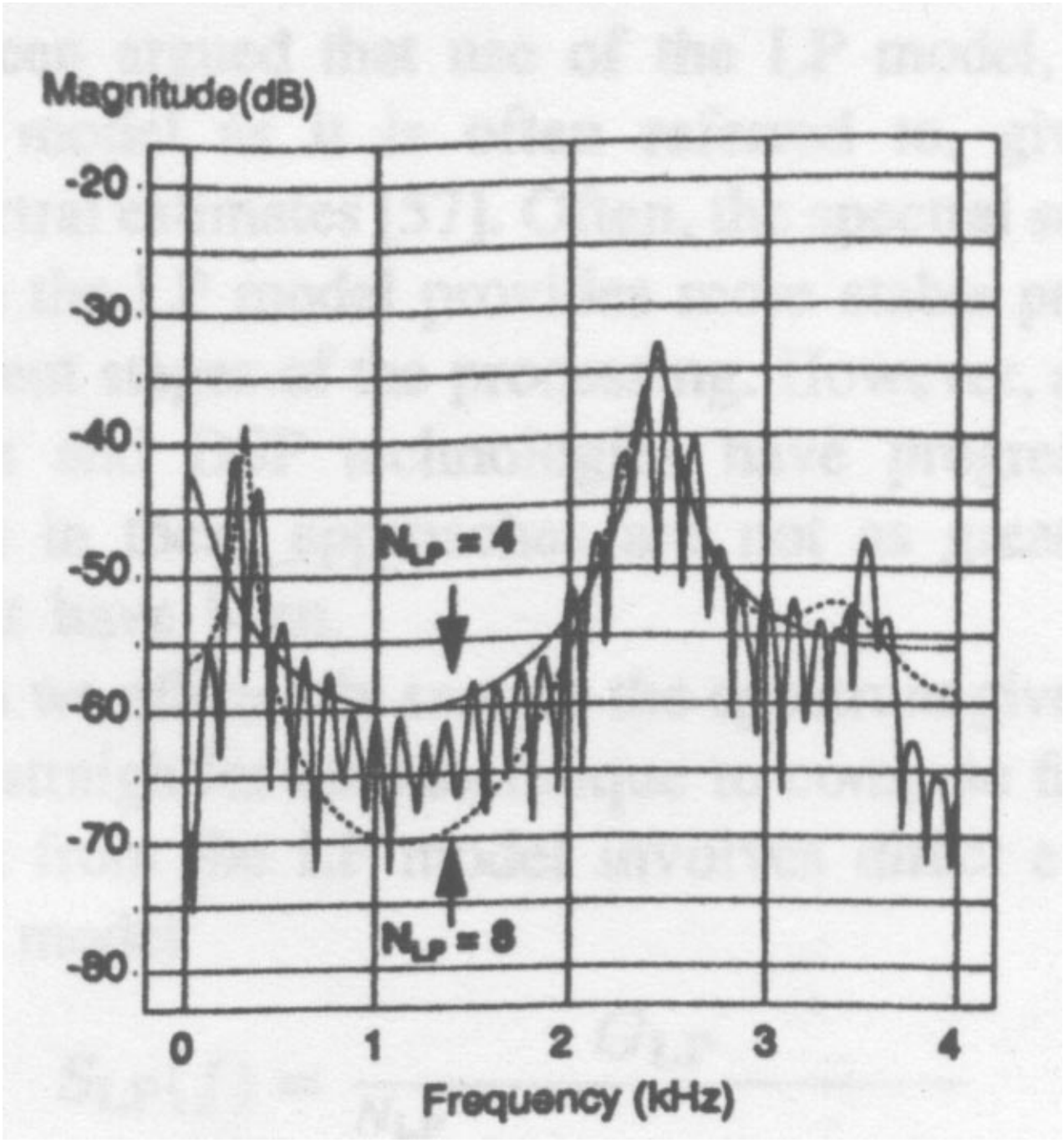
(2) $|k_i| = 1$, implies a harmonic process (poles on the unit circle).

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

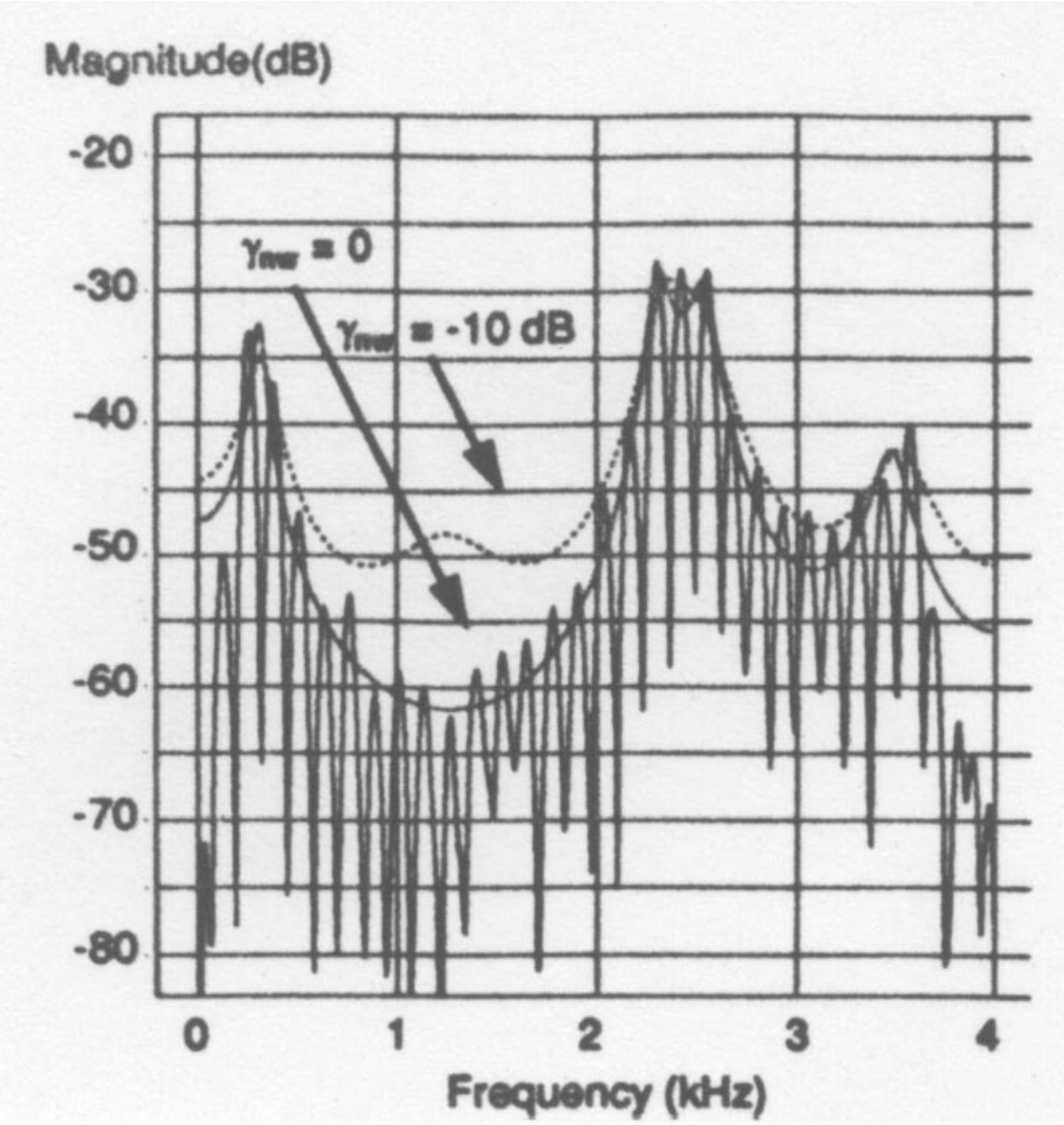
(4) $E_0 > E_1 > \ldots > E_p$

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!).

Spectral-Matching Interpretation of Linear Prediction



Noise-Weighting



The Real Cepstrum

Goal: Deconvolve spectrum for multiplicative processes

In practice, we use the "real" cepstrum:

$$\begin{aligned} C_s(\omega) &= \log|S(f)| \\ &= \log|V(f)U(f)| \\ &= \log|V(f)| + \log|U(f)| \end{aligned}$$

$V(f)$ and $U(f)$ manifest themselves at the low and high end of the "quefrency" domain respectively.

We can derive cepstral parameters directly from LP analysis:

$$\ln A(z) = C(z) = \sum_{n=1}^{\infty} c_n z^{-n}$$

To obtain the relationship between cepstral and predictor coefficients, we can differentiate both sides is taken with respect to z^{-1}:

$$\frac{d}{dz^{-1}}\{\ln A(z)\} = \frac{d}{dz^{-1}}\left\{\ln\left[\frac{1}{1-\sum_{k=1}^{p}\alpha_k z^{-k}}\right]\right\} = \frac{d}{dz^{-1}}\sum_{n=1}^{\infty} c_n z^{-n}$$

which simplifies to

$$c_1 = a_1$$

$$c_n = \sum_{k=1}^{n-1}(1-k/n)\alpha_k c_{n-k} + a_n$$

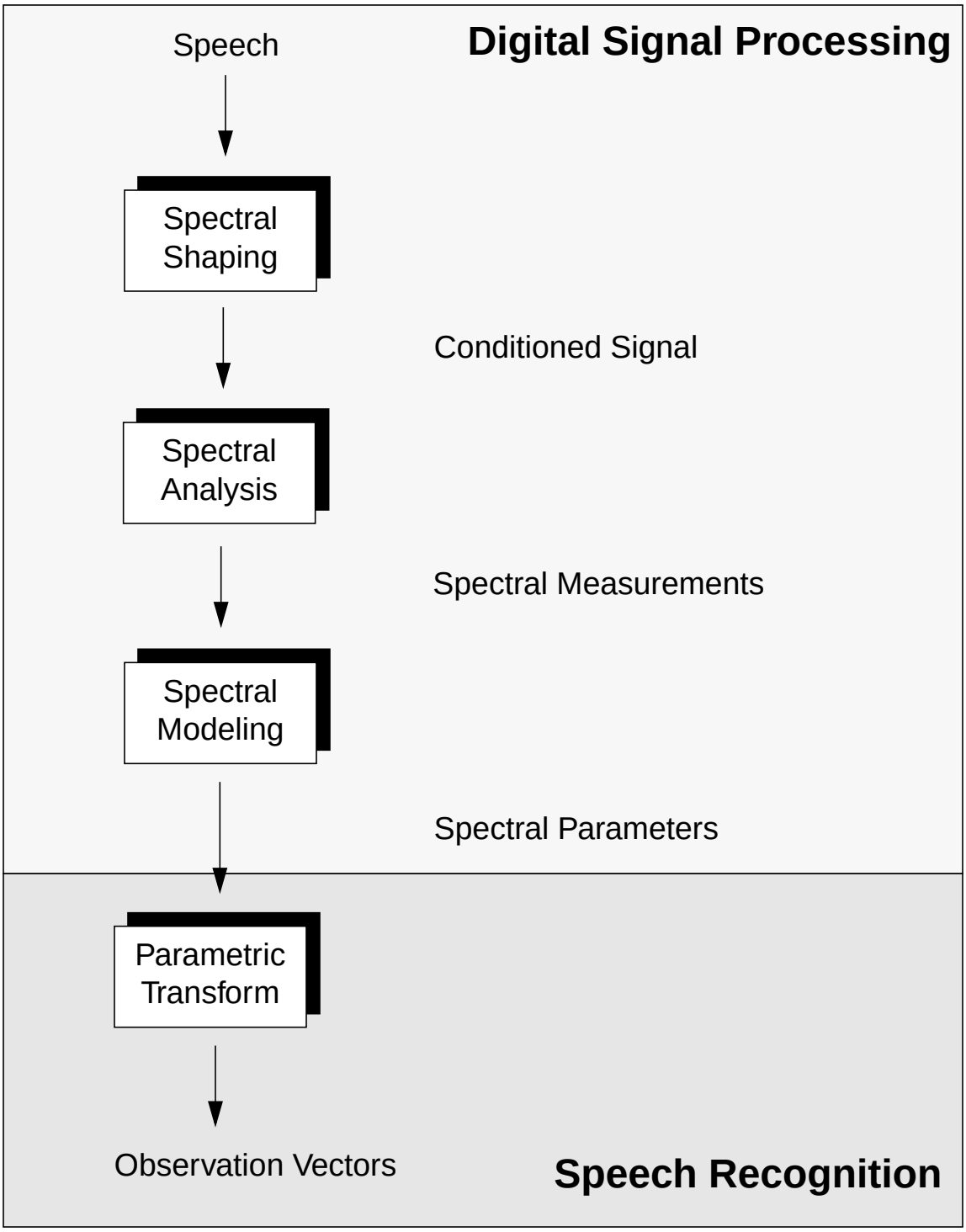
Note that the order of the cepstral coefficients need not be the same as the order of the LP model. Typically, 10-16 LP coefficients are used to generate 10-12 cepstral coefficients.

Session IV:

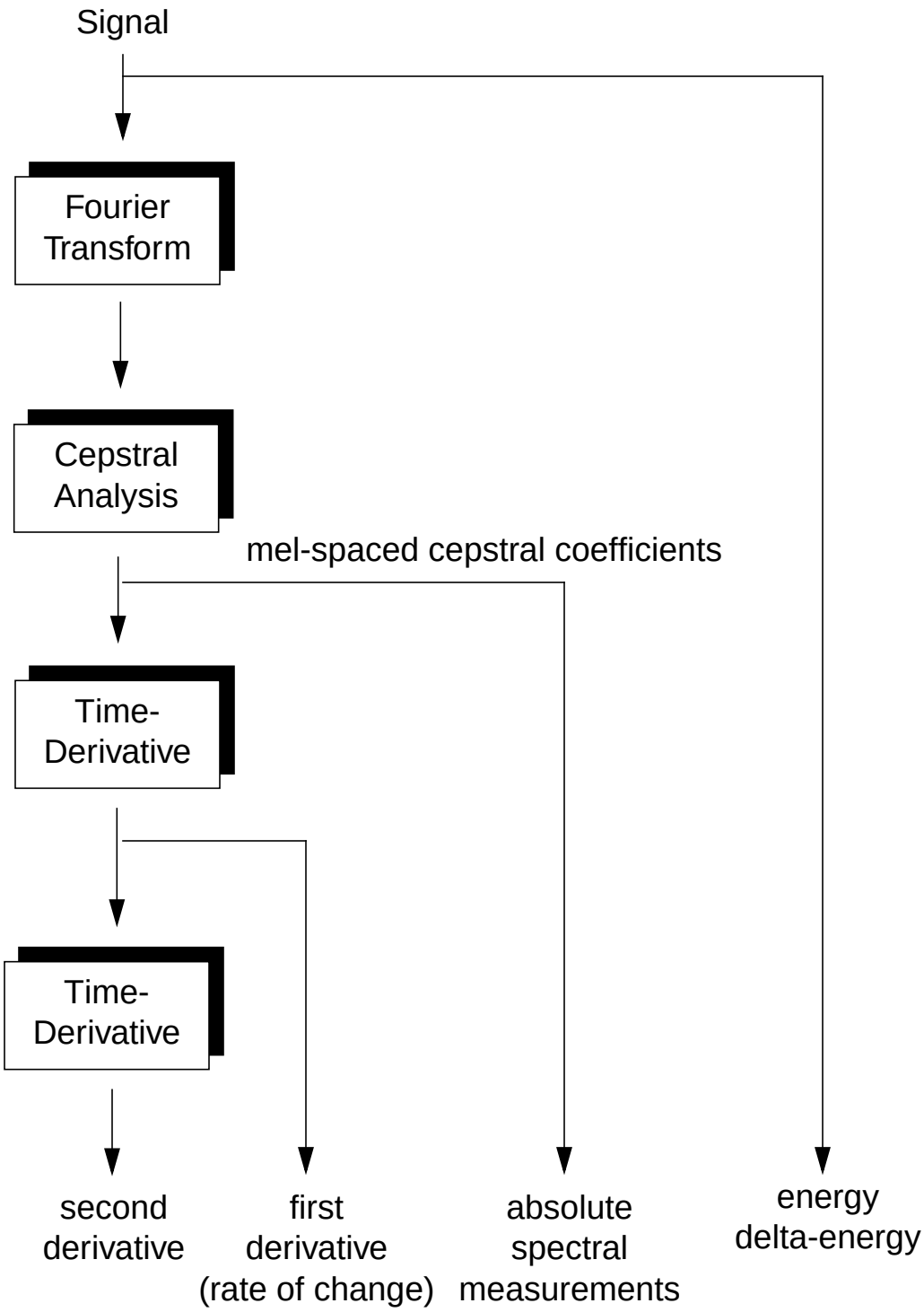
Signal Processing

In Speech Recognition

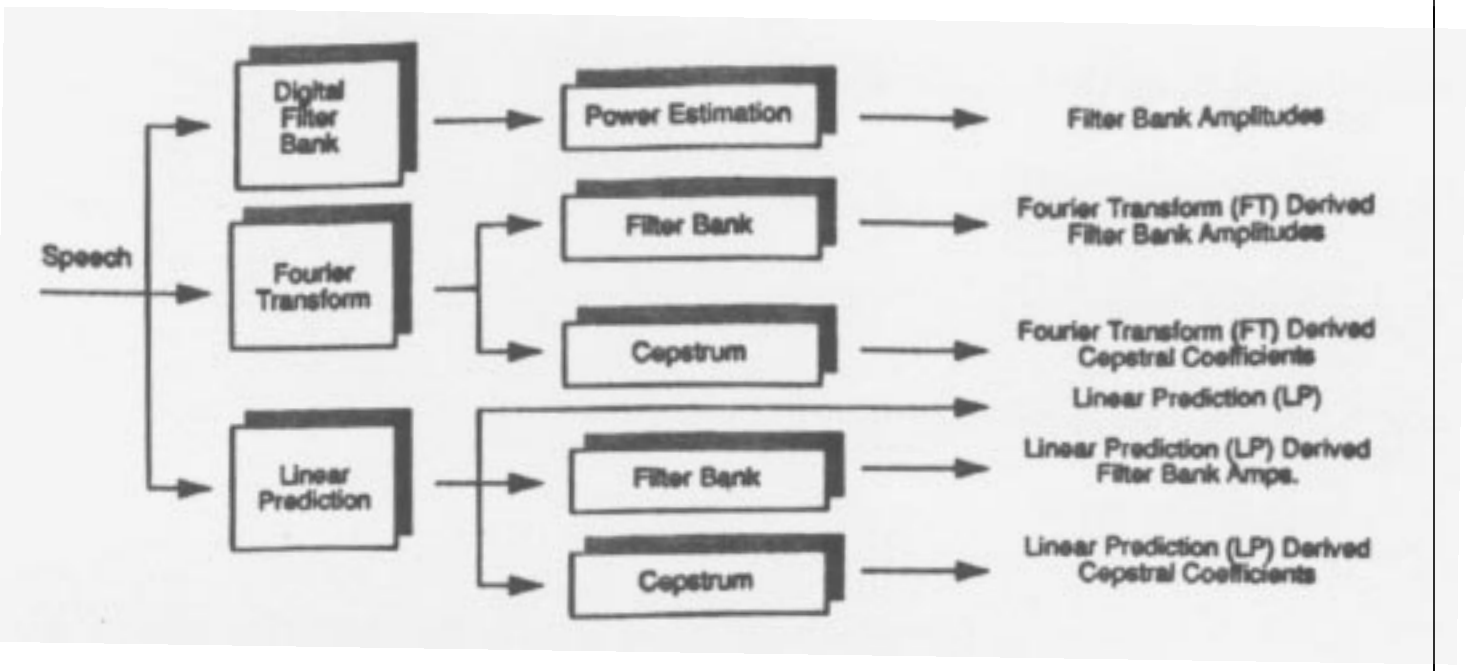
THE SIGNAL MODEL ("FRONT-END")



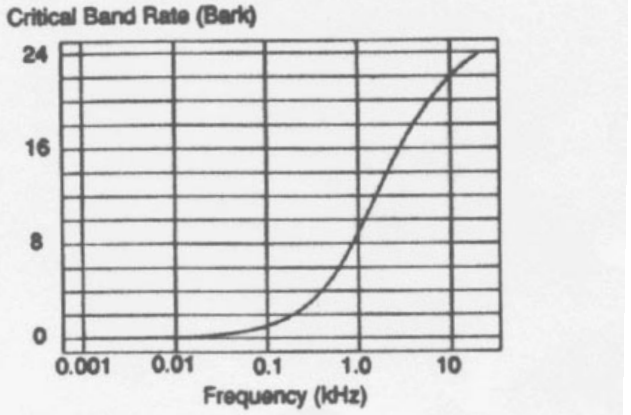
A TYPICAL “FRONT-END”



Putting It All Together

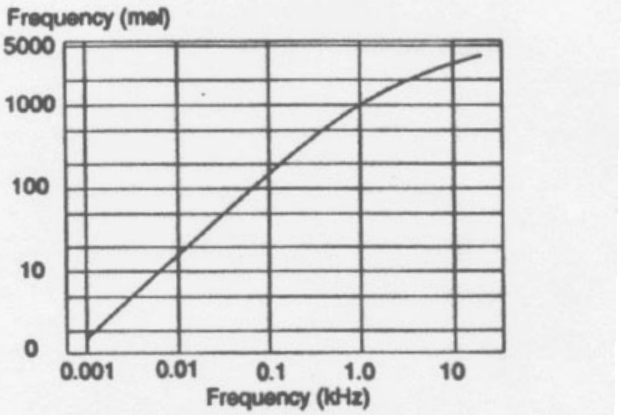


Mel Cepstrum



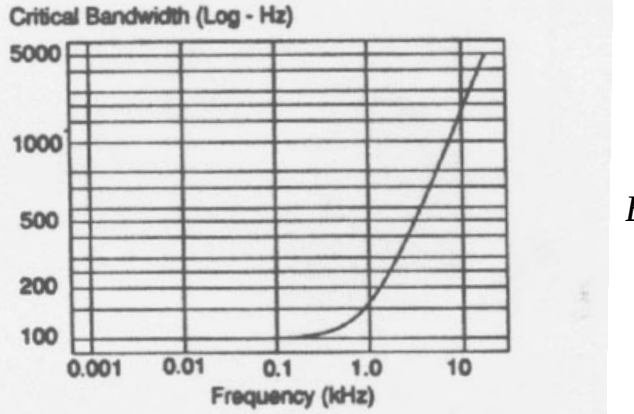
(a)

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



(b)

$$mel\ f = 2595\log 10\left(1 + \frac{f}{7000}\right)$$



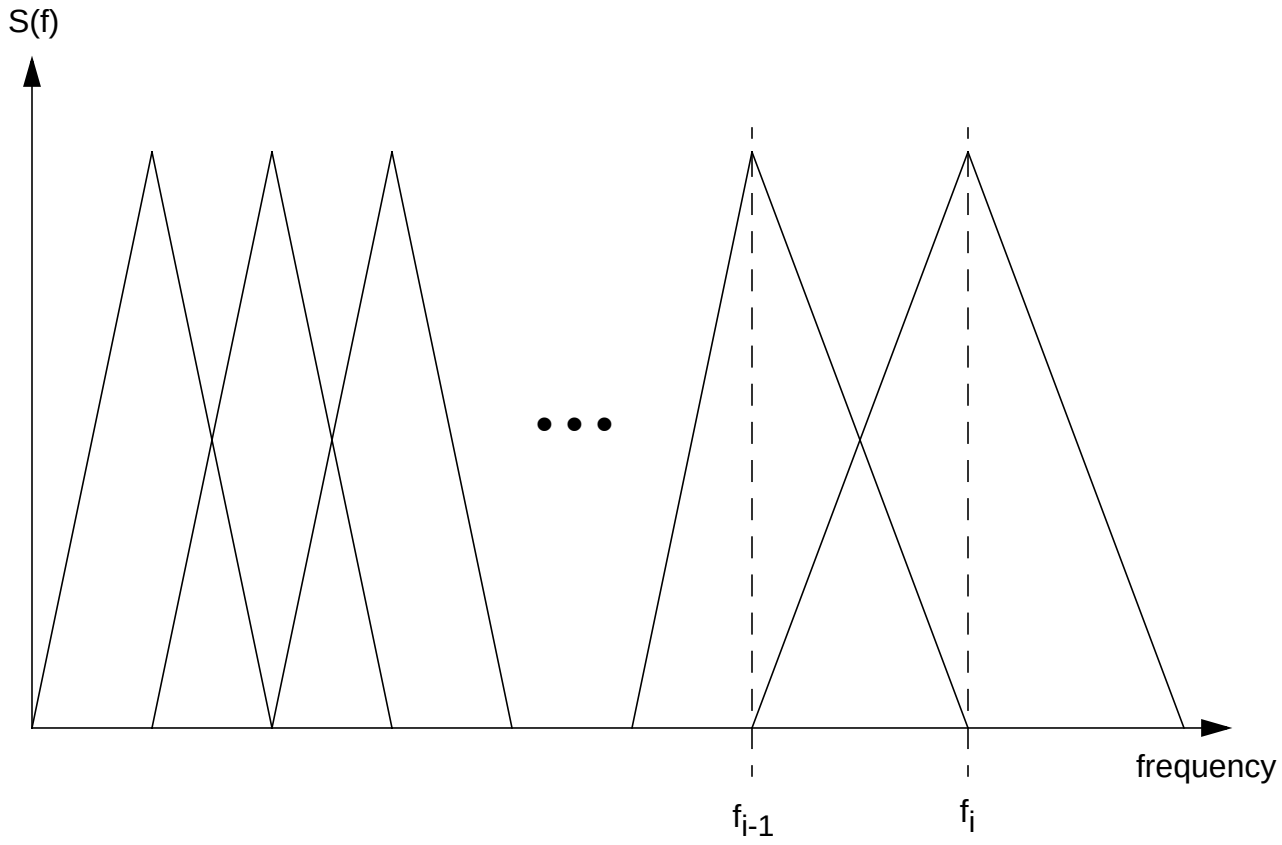
(c)

$$BW_{crit} = 25 + 75\left[1 + 1.4\left(\frac{f}{1000}\right)^2\right]^{0.69}$$

Mel Cepstrum Computation Via A Filter Bank

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

Mel Cepstrum Computation Via Oversampling Using An FFT



- Note that an FFT yields frequency samples at $\left(\frac{k}{N}\right) f_s$
- Oversampling provides a smoother estimate of the envelope of the spectrum
- Other analogous techniques efficient sampling techniques exist for different frequency scales (bilinear transform, sampled autocorrelation, etc.)

Perceptual Linear Prediction

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

Claims:

- Improved speaker independent recognition performance
- Increased robustness to noise, variations in the channel, and microphones

Linear Regression and Parameter Trajectories

Premise: Time differentiation of features is a noisy process

Approach: Fit a polynomial to the data to provide a smooth trajectory for a parameter; use closed-loop estimation of the polynomial coefficients

Static feature:

$$s(n) = c_k(n)$$

Dynamic feature:

$$\dot{s}(n) \approx c_k(n+\Delta) - c_k(n-\Delta)$$

Acceleration feature:

$$\ddot{s}(n) \approx \dot{s}(n+\Delta) - \dot{s}(n-\Delta)$$

We can generalize this using an r^{th} order regression analysis:

$$R_{rk}(t, T, \Delta T) = \frac{\sum_{X=1}^{L} P_r(X, L) C_k\left(t + \left(X - \frac{L+1}{2}\right)\Delta T\right)}{\sum_{X=1}^{L} P_r^2(X, L)}$$

where $L = T/\Delta T$ (the number of analysis frames in time length T) is odd, and the orthogonal polynomials are of the form:

$$P_0(X, L) = 1$$

$$P_1(X, L) = X$$

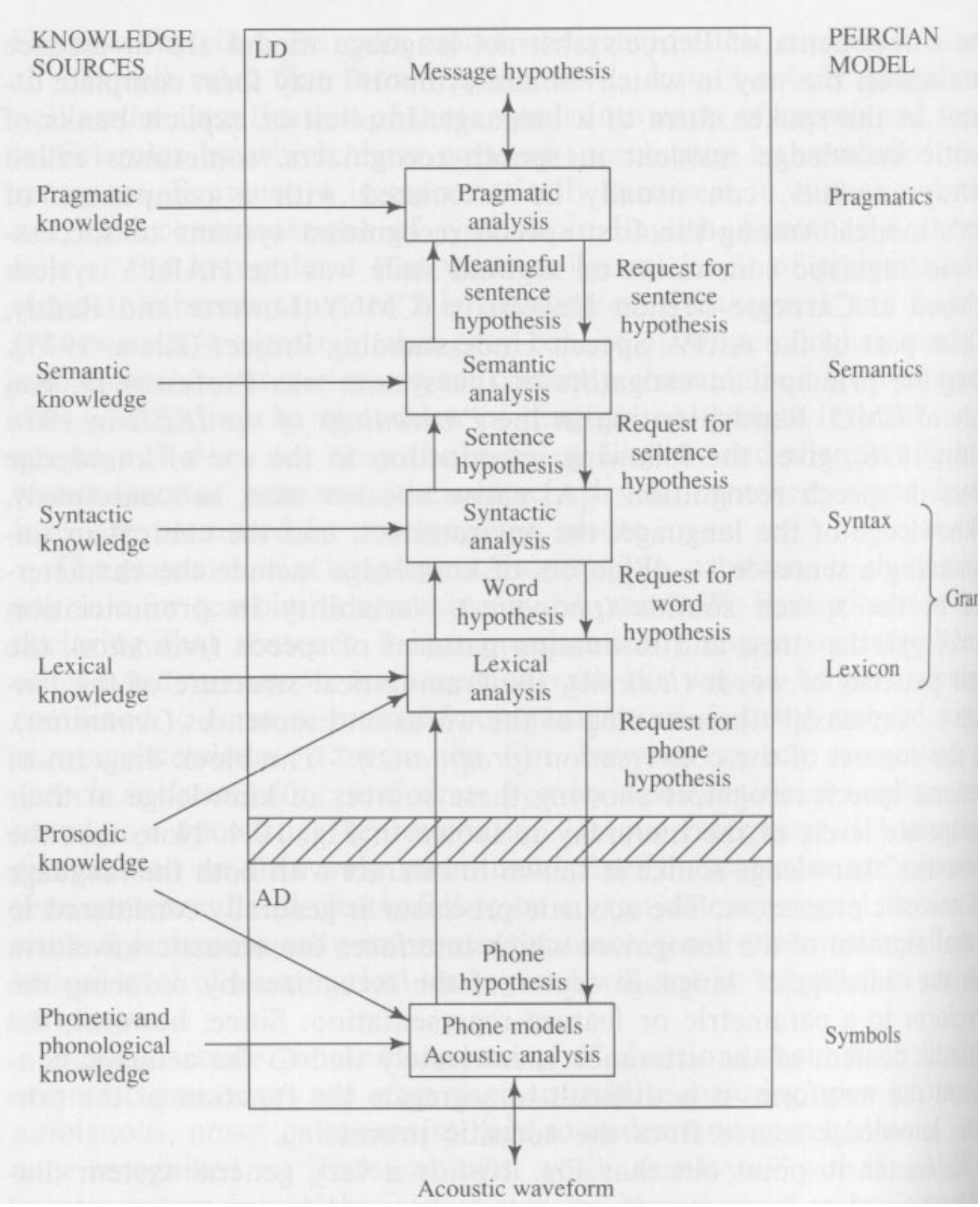
$$P_2(X, L) = X^2 - \frac{1}{12}(L^2 - 1)$$

$$P_3(X, L) = X^3 - \frac{1}{20}(3L^2 - 7)X$$

This approach has been generalized in such a way that the weights on the coefficients can be estimated directly from training data to maximize the likelihood of the estimated feature (maximum likelihood linear regression).

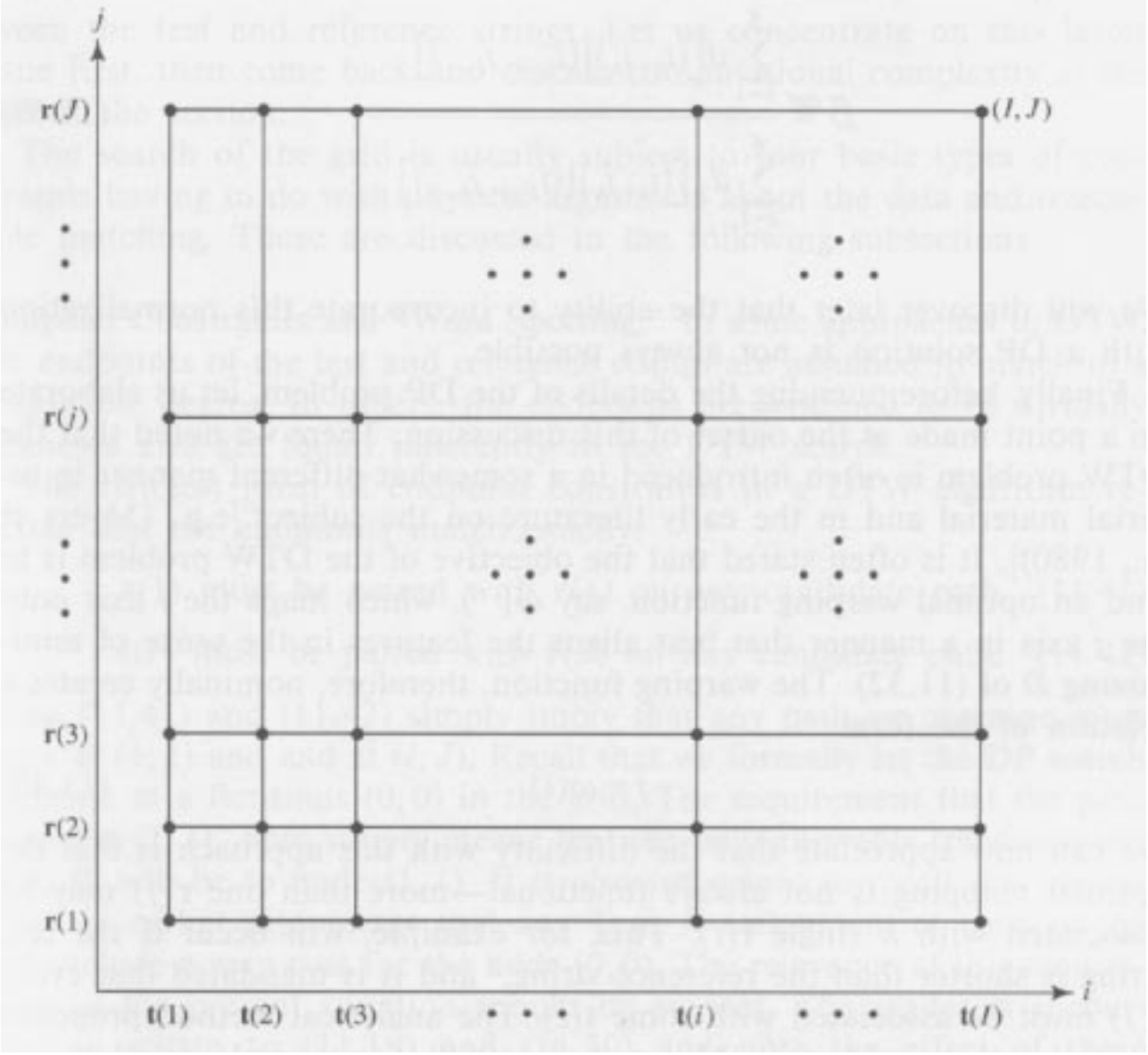
Session V:

Dynamic Programming


KNOWLEDGE SOURCES
LD
PEIRCIAN MODEL
Message hypothesis
Pragmatic knowledge
Pragmatic analysis
Pragmatics
Meaningful sentence hypothesis
Request for sentence hypothesis
Semantic knowledge
Semantic analysis
Semantics
Sentence hypothesis
Request for sentence hypothesis
Syntactic knowledge
Syntactic analysis
Syntax
Word hypothesis
Request for word hypothesis
Lexical knowledge
Lexical analysis
Lexicon
Request for phone hypothesis
Prosodic knowledge
AD
Phone hypothesis
Phonetic and phonological knowledge
Phone models Acoustic analysis
Symbols
Acoustic waveform

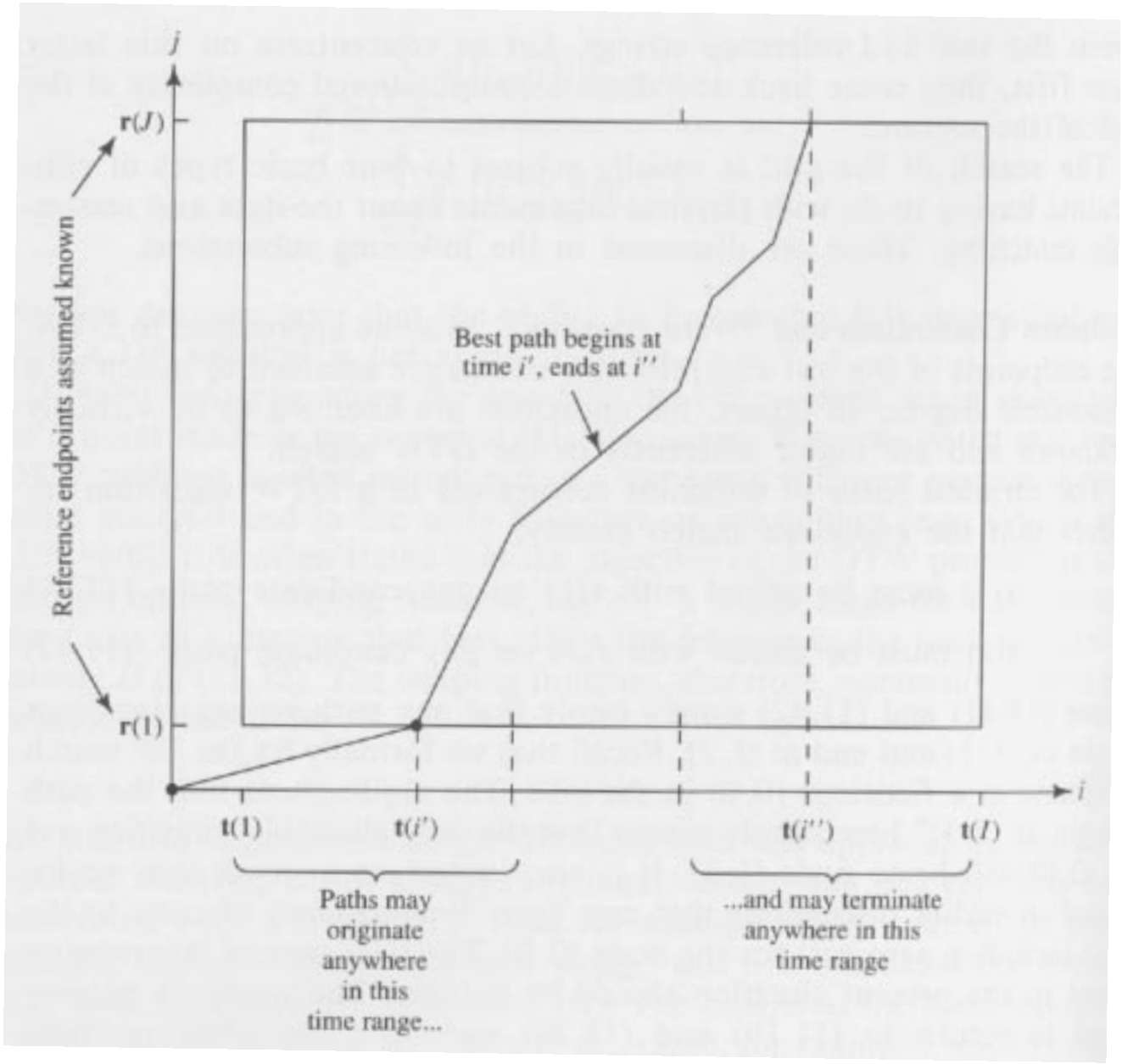
The Principle of Dynamic Programming

- An efficient algorithm for finding the optimal path through a network

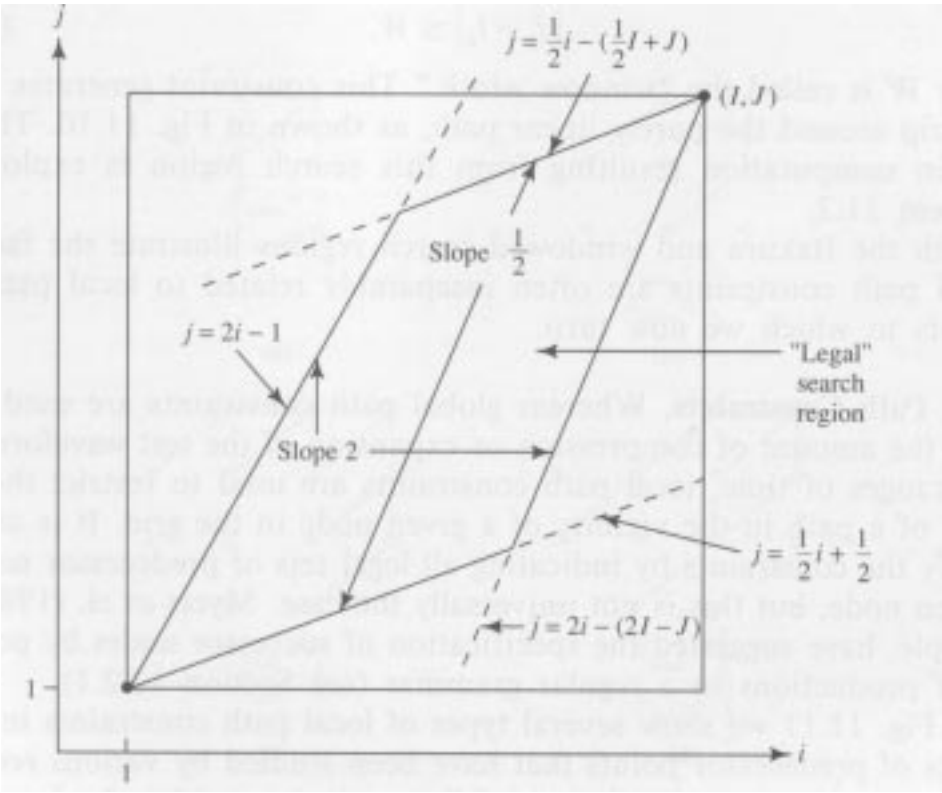


Word Spotting Via Relaxed and Unconstrained Endpointing

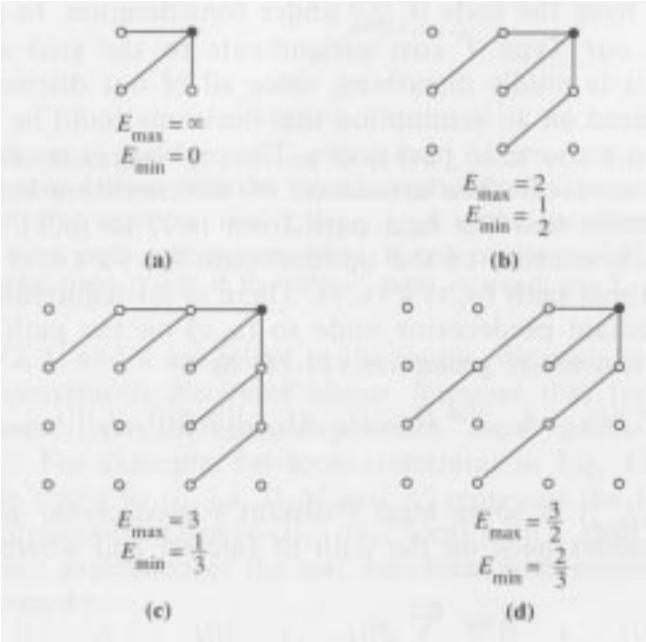
- Endpoints, or boundaries, need not be fixed — numerous types of constraints can be invoked

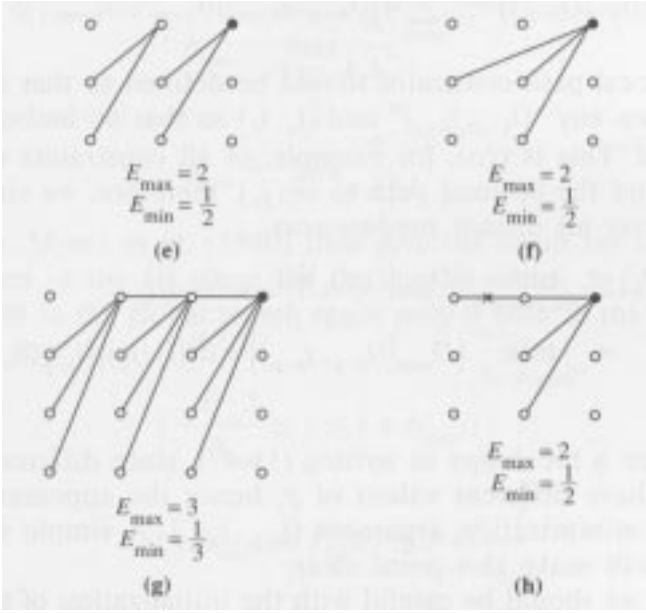


Slope Constraints: Increased Efficiency and Improved Performance



- Local constraints used to achieve slope constraints

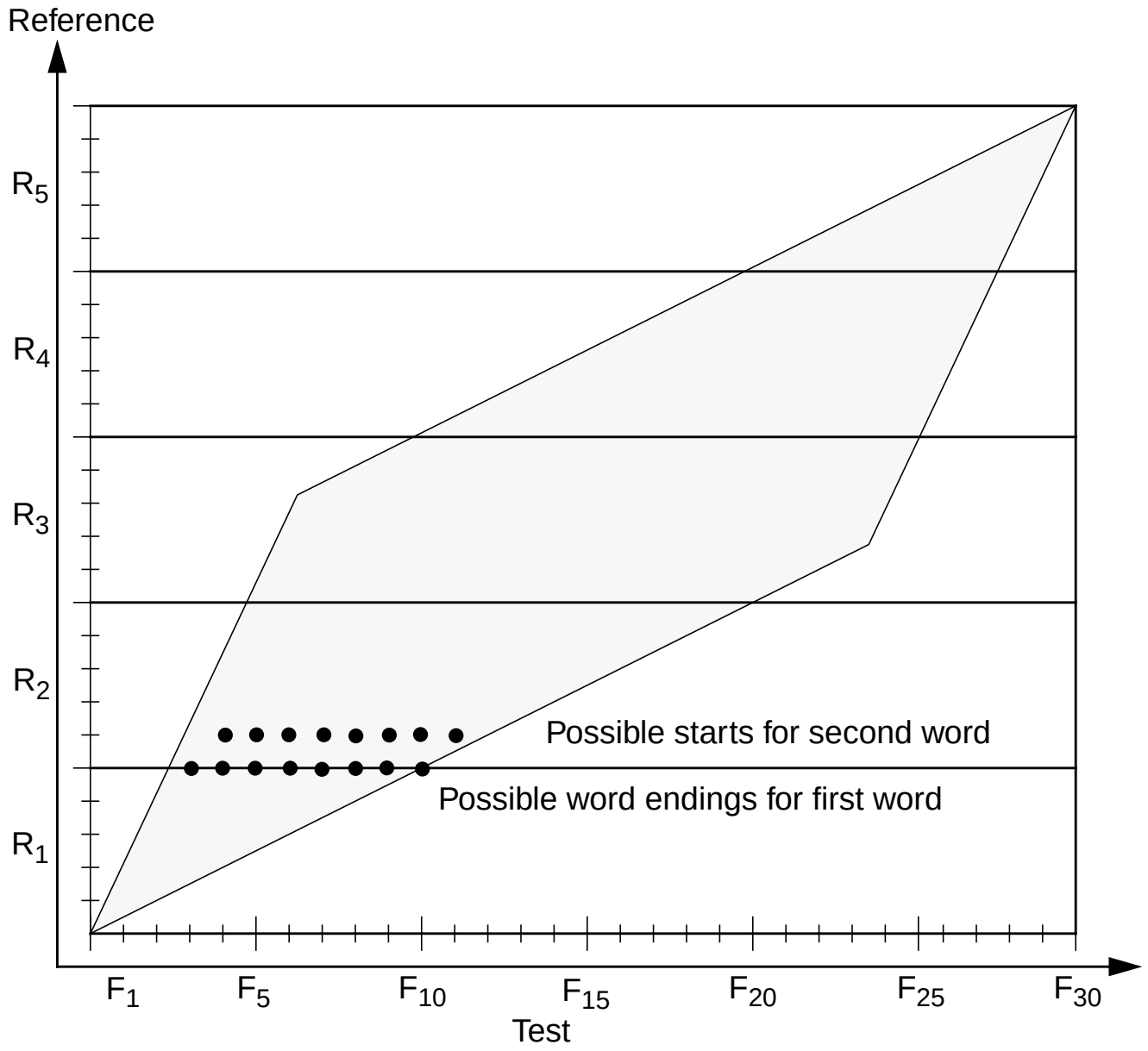




DTW, Syntactic Constraints, and Beam Search

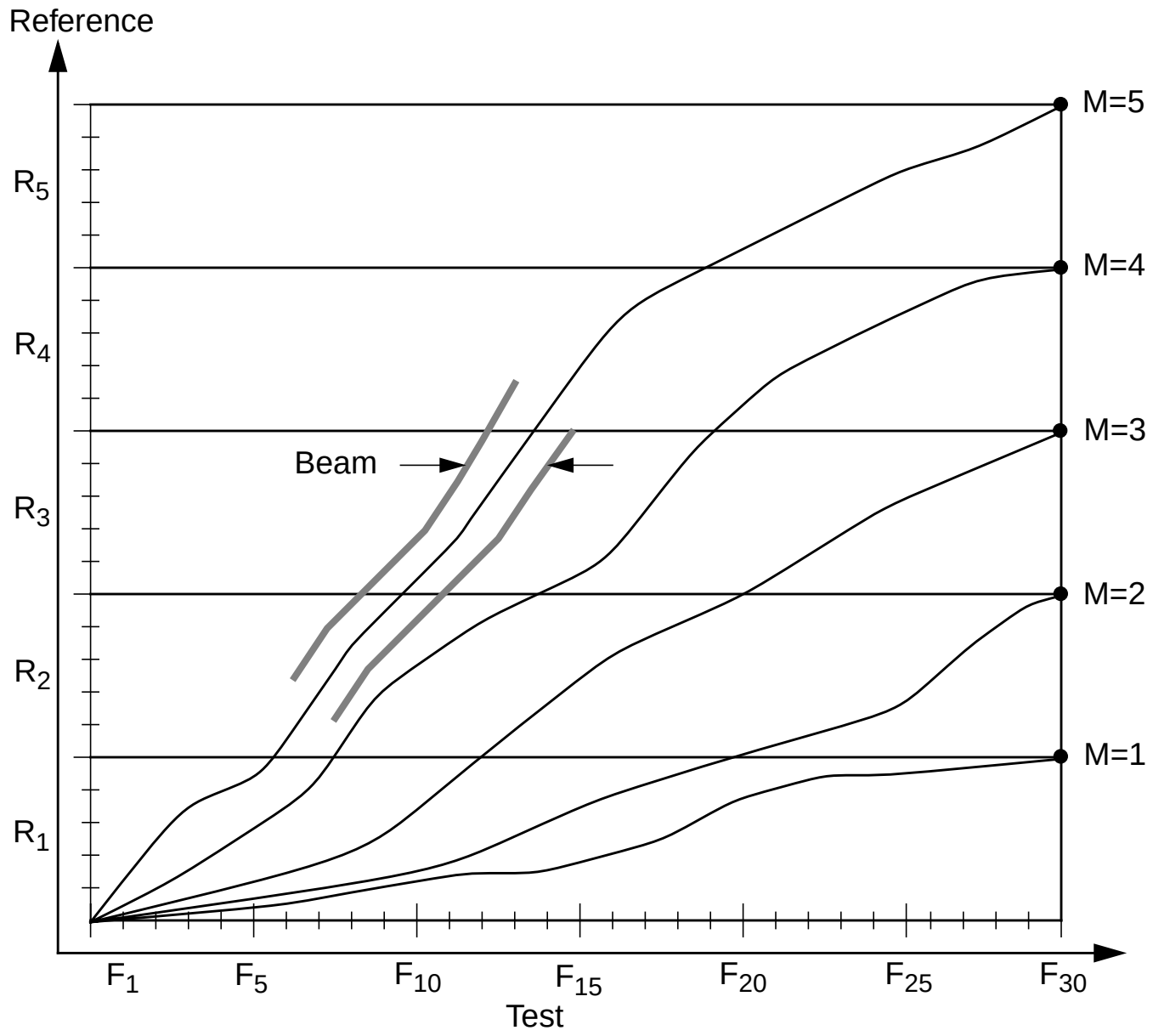
Consider the problem of connected digit recognition: “325 1739”. In the simplest case, any digit can follow any other digit, but we might know the exact number of digits spoken.

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.

Level Building For An Unknown Number Of Words

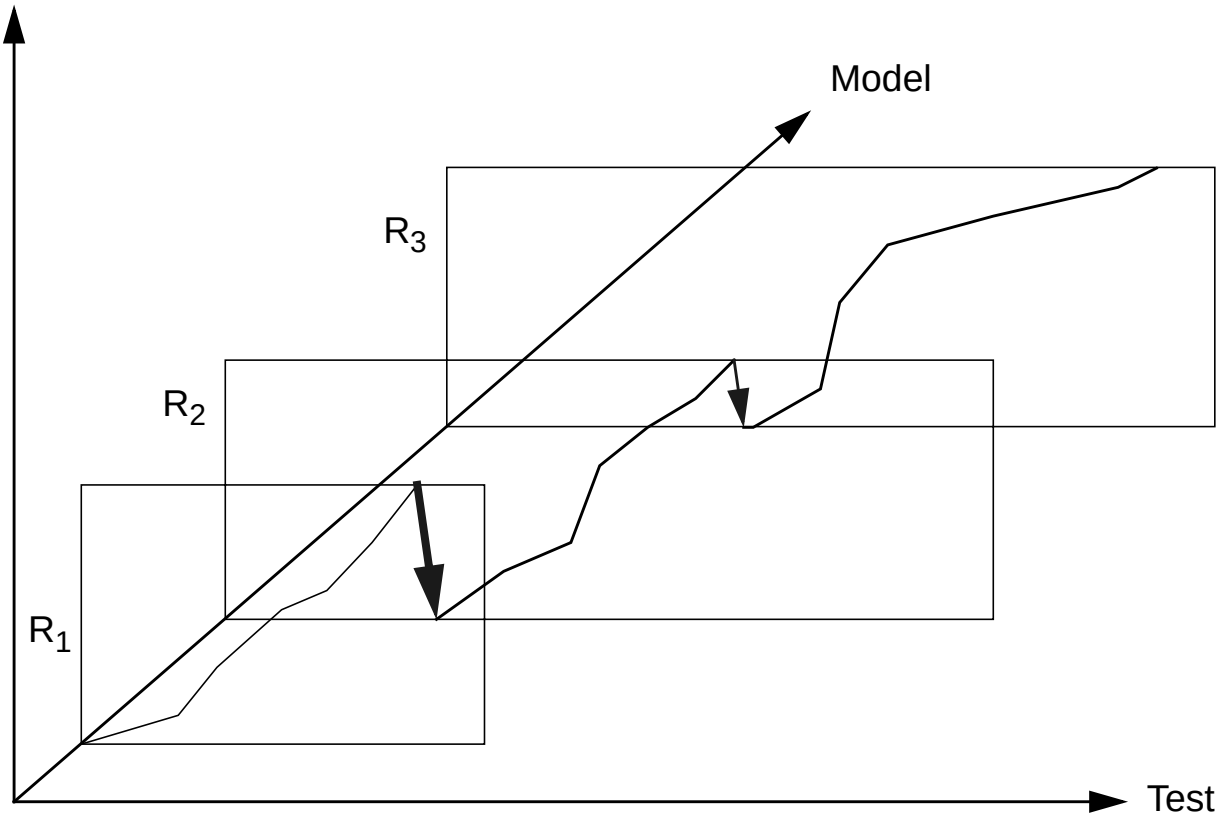


- Paths can terminate on any level boundary indicating a different number of words was recognized (note the significant increase in complexity)
- A search band around the optimal path can be maintained to reduce the search space
- Next-best hypothesis can be generated (N-best)
- Heuristics can be applied to deal with free endpoints, insertion of silence between words, etc.
- Major weakness is the assumption that all models are the same length!

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).

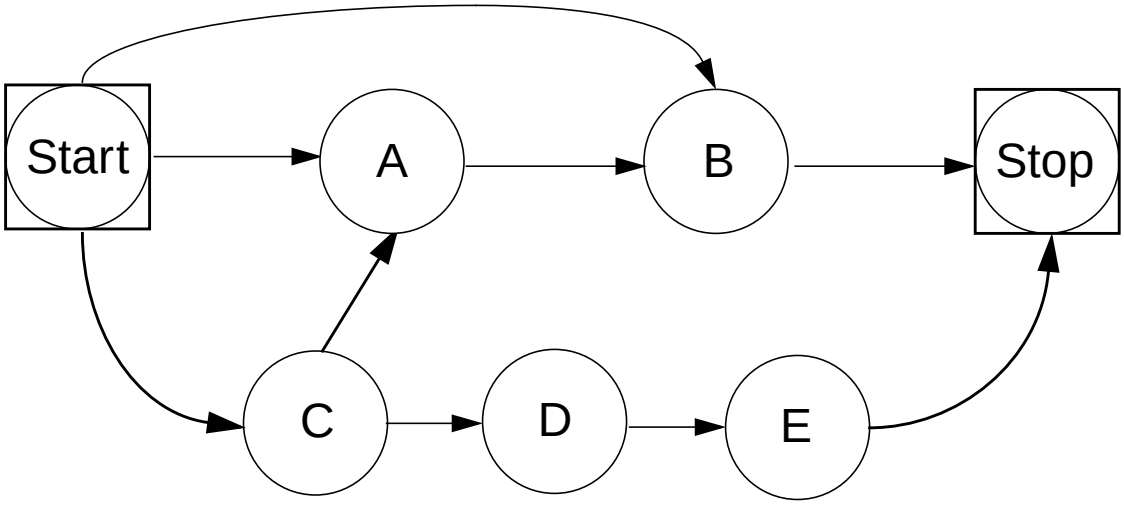
An elegant algorithm to perform this search in one pass is demonstrated below:



- 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?

Introduction of Syntactic Information

- 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 a via a state machine:



For example, when you enter state C, you output one of the following words: {daddy, mommy}.

If:

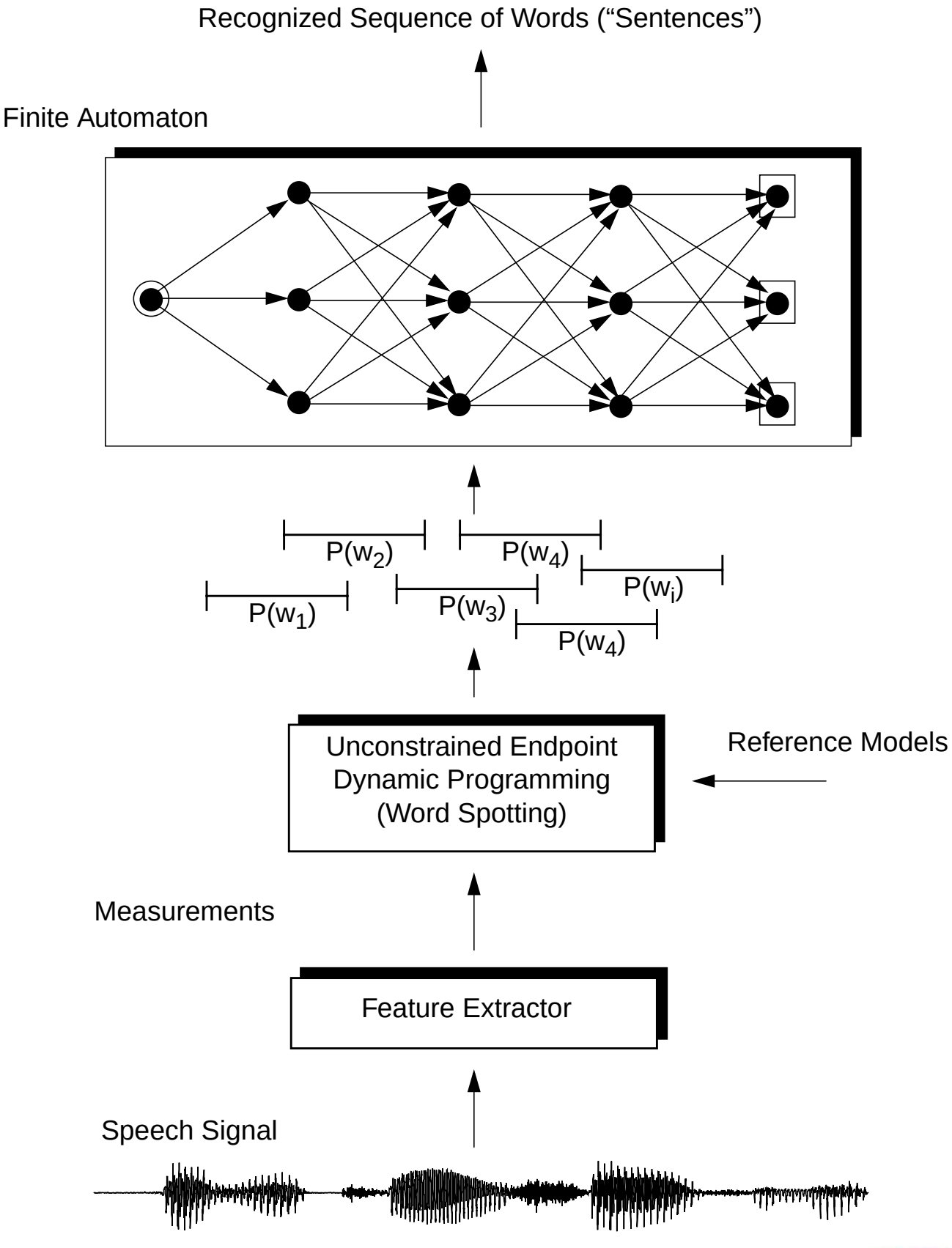
state A:	give
state B:	me
state C:	{daddy, mommy}
state D:	come
state E:	here

We can generate phrases such as:

Daddy give me

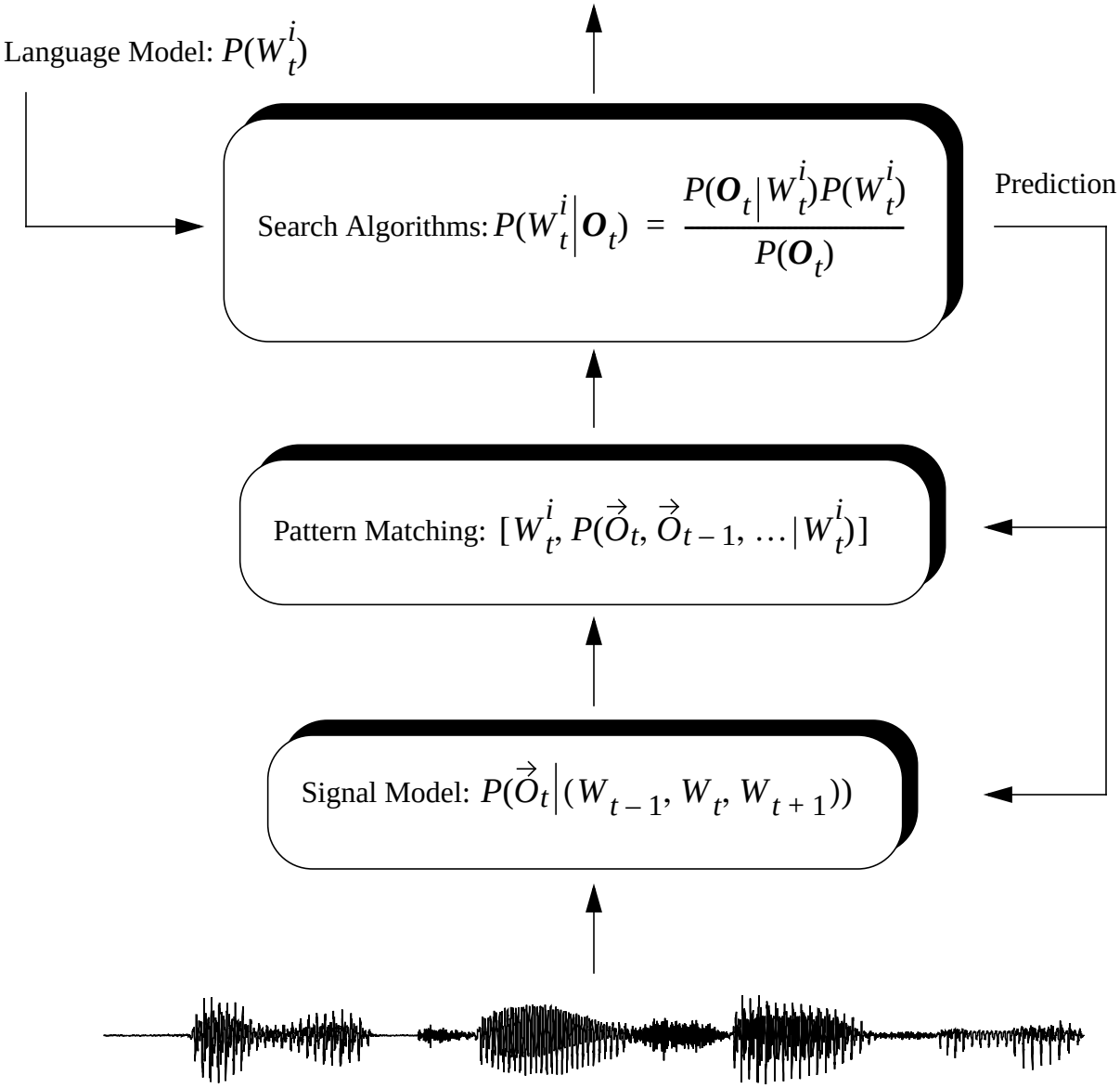
- We can represent such information numerous ways (as we shall see)

Early Attempts At Introducing Syntactic Information Were “Ad-Hoc”



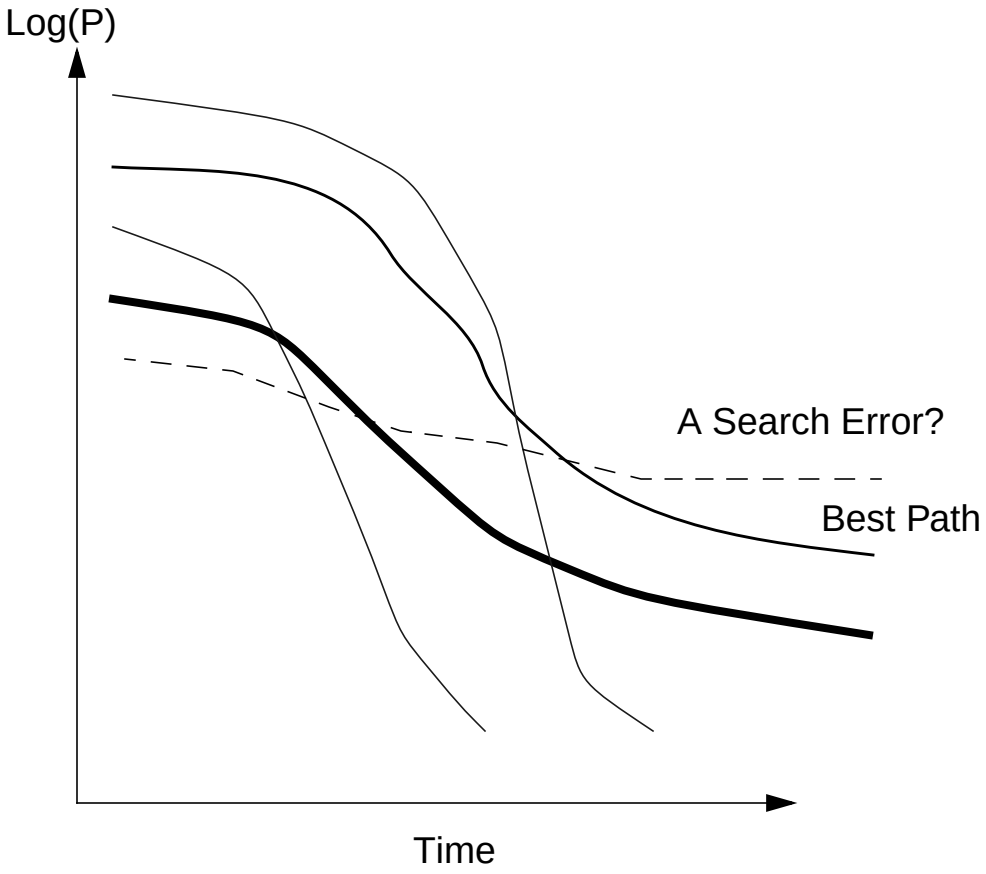
BASIC TECHNOLOGY — A PATTERN RECOGNITION PARADIGM BASED ON HIDDEN MARKOV MODELS

Recognized Symbols: $P(S|\boldsymbol{O}) = \arg max|_T \prod_i P(W_t^i \Big| (\vec{O}_t, \vec{O}_{t-1}, \ldots))$



BEAM SEARCH

- The modern view of speech recognition is a problem consisting of two operations: signal modeling and search.
- Finding the most probable sequence of words is an optimization problem that can be solved by dynamic programming
- Optimal solutions are intractable; fortunately, sub-optimal solutions yield good performance
- Beam search algorithms are used to trade-off complexity vs. accuracy



Session VI:

Hidden Markov Models

A Simple Markov Model For Weather Prediction

What is a first-order Markov chain?

$$P[q_t = j \,|\, (q_{t-1} = i, q_{t-2} = k, \ldots)] = P[q_t = j \,|\, q_{t-1} = i]$$

We consider only those processes for which the right-hand side is independent of time:

$$a_{ij} = P[q_t = j \,|\, q_{t-1} = i] \qquad 1 \le i, j \le N$$

with the following properties:

$$a_{ij} \ge 0 \qquad \forall j, i$$

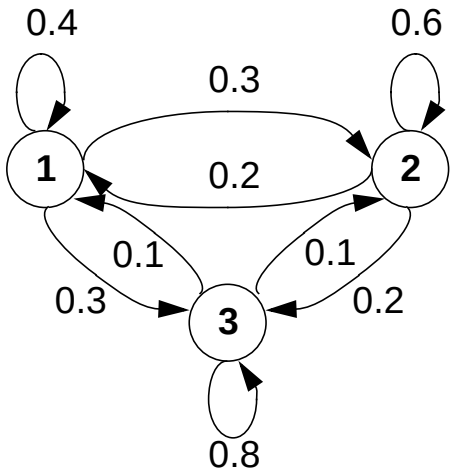
$$\sum_{j=1}^{N} a_{ij} = 1 \qquad \forall i$$

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.

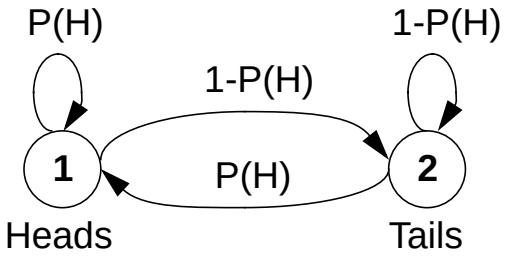
What have we learned from this example?

Why Are They Called "Hidden" Markov Models?

Consider the problem of predicting the outcome of a coin toss experiment. You observe the following sequence:

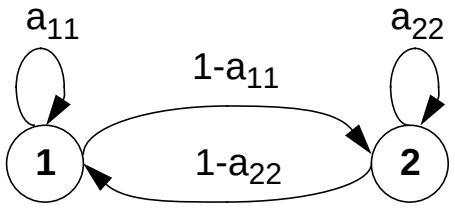
$$\bar{O} = (HHTTTHTTH \ldots H)$$

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 ...

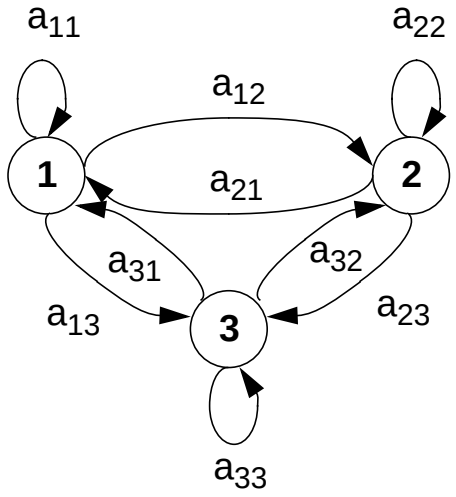


2-Coins Model
(Hidden Markov Model)

O =	H	H	T	T	H	T	H	H	T	T	H ...
S =	1	1	2	2	1	2	1	1	2	2	1 ...

$P(H) = P_1$ $\quad$ $P(H) = P_2$

$P(T) = 1-P_1$ $\quad$ $P(T) = 1-P_2$



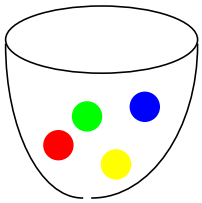
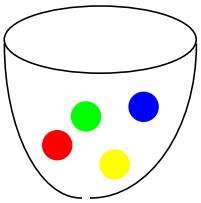
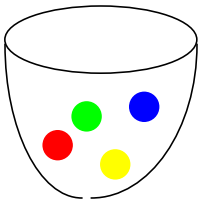
3-Coins Model
(Hidden Markov Model)

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):	$1-P_1$	$1-P_2$	$1-P_3$

Why Are They Called Doubly Stochastic Systems?

The Urn-and-Ball Model

P(red) = $b_1(1)$	P(red) = $b_2(1)$	P(red) = $b_3(1)$
P(green) = $b_1(2)$	P(green) = $b_2(2)$	P(green) = $b_3(2)$
P(blue) = $b_1(3)$	P(blue) = $b_2(3)$	P(blue) = $b_3(3)$
P(yellow) = $b_1(4)$	P(yellow) = $b_2(4)$	P(yellow) = $b_3(4)$
...	...	...

$\overline{O}$ = {green, blue, green, yellow, red, ..., blue}

How can we determine the appropriate model for the observation sequence given the system above?

Elements of a Hidden Markov Model (HMM)

- 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)?

Formalities

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:

$$b(y(t) \,|\, i) \equiv P(\underline{y}(t) = y(t) \,|\, \underline{x}(t) = i)$$

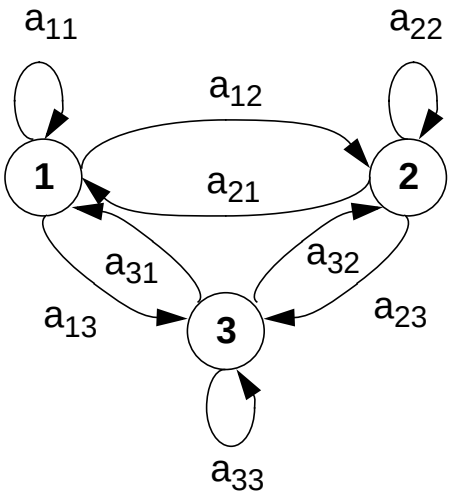
The observation probability distribution can be represented as a matrix whose dimension is K rows x S states.
We can define the observation probability vector as:

$$\boldsymbol{p}(t) = \begin{bmatrix} P(\underline{y}(t) = 1) \\ P(\underline{y}(t) = 2) \\ \cdots \\ P(\underline{y}(t) = K) \end{bmatrix}, \quad \text{or,} \quad \boldsymbol{p}(t) = \boldsymbol{B}\pi(t) = \boldsymbol{B}\boldsymbol{A}^{t-1}\pi(1)$$

The mathematical specification of an HMM can be summarized as:

$$\boldsymbol{M} = \{S, \pi(1), \boldsymbol{A}, \boldsymbol{B}, \{\boldsymbol{y}_k, 1 \le k \le K\}\}$$

For example, reviewing our coin-toss model:



P(H): P_1 P_2 P_3
P(T): $1\text{-}P_1$ $1\text{-}P_2$ $1\text{-}P_3$

$$S = 3$$

$$\pi(1) = \begin{Bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{Bmatrix}$$

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$$\boldsymbol{B} = \begin{bmatrix} P_1 & P_2 & P_3 \\ 1-P_1 & 1-P_2 & 1-P_3 \end{bmatrix}$$

Recognition Using Discrete HMMs

Denote any partial sequence of observations in time by:

$$y_{t_1}^{t_2} \equiv \{y(t_1), y(t_1+1), y(t_1+2), \ldots, y(t_2)\}$$

The forward partial sequence of observations at time t is

$$y_1^t \equiv \{y(1), y(2), \ldots, y(t)\}$$

The backward partial sequence of observations at time t is

$$y_{t+1}^T \equiv \{y(t+1), y(t+2), \ldots, y(T)\}$$

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 $\vartheta = \{i_1, i_2, \ldots, i_T\}$ denote a specific state sequence. The probability of a given observation sequence being produced by this state sequence is:

$$P(y|\vartheta, M) = b(y(1)|i_1)b(y(2)|i_2)\ldots b(y(T)|i_T)$$

The probability of the state sequence is

$$P(\vartheta|M) = P(\underline{x}(1) = i_1)a(i_2|i_1)a(i_3|i_2)\ldots a(i_T|i_{T-1})$$

Therefore,

$$P(y,(\vartheta|M)) = P(\underline{x}(1) = i_1)a(i_2|i_1)a(i_3|i_2)\ldots a(i_T|i_{T-1})$$
$$x \; b(y(1)|i_1)b(y(2)|i_2)\ldots b(y(T)|i_T)$$

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!

The "Any Path" Method (Forward-Backward, Baum-Welch)

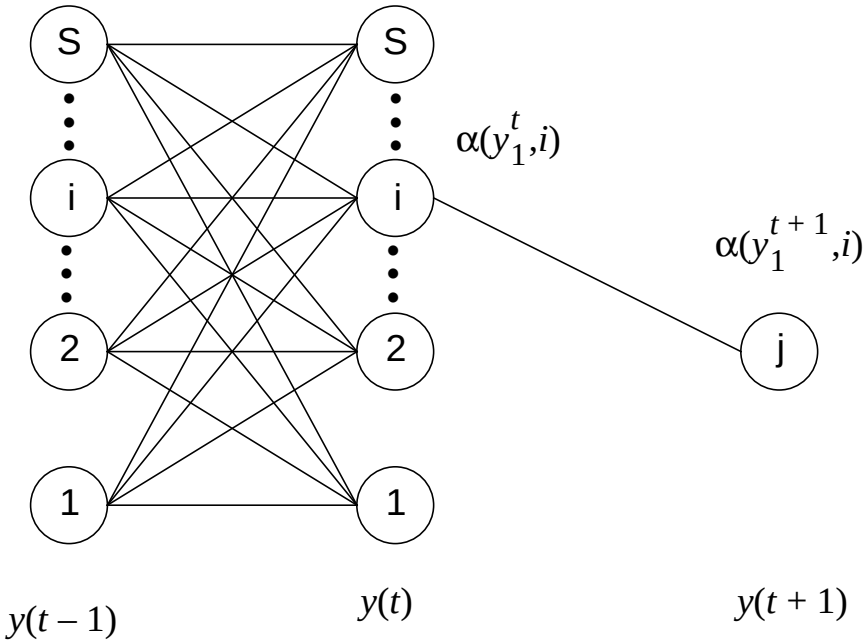
The *forward-backward* (F-B) *algorithm* begins by defining a "forward-going" probability sequence:

$$\alpha(y_1^t) \equiv P(\underline{y}_1^t = y_1^t, \underline{x}(t) = i \,|\, M)$$

and a "backward-going" probability sequence:

$$\beta(y_{t+1}^T \,|\, i) \equiv P(\underline{y}_{t+1}^T = y_{t+1}^T \,|\, \underline{x}(t) = i, M)$$

Let us next consider the contribution to the overall sequence probability made by a single transition:



$$\begin{aligned}\alpha(y_1^{t+1}, j) &= \alpha(y_1^t, i) P(\underline{x}(t+1) = j \,|\, \underline{x}(t) = i) \times \\ &\quad P(\underline{y}(t+1) = y(t+1) \,|\, \underline{x}(t+1) = j) \\ &= \alpha(y_1^t, i) a(j \,|\, i) b(y(t+1) \,|\, j)\end{aligned}$$

Summing over all possibilities for reaching state "j":

$$\alpha(y_1^{t+1}, j) = \sum_{i=1}^{S} \alpha(y_1^t, i) a(j \,|\, i) b(y(t+1) \,|\, j)$$

Baum-Welch (Continued)

The recursion is initiated by setting:

$$\alpha(y_1^t, j) = P(\underline{x}(1) = j) b(y(1) | j)$$

Similarly, we can derive an expression for β:

$$\beta(y_{i+1}^T \Big| i) = \sum_{j=1}^{S} \beta(y_{t+2}^T \Big| j) a(j|i) b(y(t+1)|j)$$

This recursion is initialized by:

$$\beta(y_{T+1}^T \Big| i) \equiv \begin{cases} 1, & \text{if } i \text{ is a legal final state} \\ 0, & \text{otherwise} \end{cases}$$

We still need to find $P(y|M)$:

$$P(y, \underline{x}(t) = i | M) = \alpha(y_1^t, i) \beta(y_{t+1}^T \Big| i)$$

for any state i. Therefore,

$$P(y|M) = \sum_{i=1}^{S} \alpha(y_1^t, i) \beta(y_{t+1}^T \Big| i)$$

But we also note that we should be able to compute this probability using only the forward direction. By considering $t = T$, we can write:

$$P(y|M) = \sum_{i=1}^{S} \alpha(y_1^T, i)$$

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 $t = T$, $P(y|M)$ is computed by summing over ALL states.

The complexity of this algorithm is $O(S^2 T)$, or for $S = 5$ and $T = 100$, approximately 2500 flops are required (compared to 10^{72} flops for the exhaustive search).

The Viterbi Algorithm

Instead of allowing any path to produce the output sequence, and hence, creating the need to sum over all paths, we can simply assume only one path produced the output. We would like to find the single most likely path that could have produced the output. Calculation of this path and probability is straightforward, using the dynamic programming algorithm previously discussed:

$$D(t, i) = a(i, j^*)b(k|i)D(t-1, j^*)$$

where

$$j^* = \arg \max_{valid\ j} \{D(t-1, j)\}$$

(in other words, the predecessor node with the best score). Often, probabilities are replaced with the logarithm of the probability, which converts multiplications to summations. In this case, the HMM looks remarkably similar to our familiar DP systems.

Beam Search

In the context of the best path method, it is easy to see that we can employ a beam search similar to what we used in DP systems:

$$D_{min}(t, i) \geq D_{min}(t, i^*_t) - \delta(t)$$

In other words, for a path to survive, its score must be within a range of the best current score. This can be viewed as a time-synchronous beam search. It has the advantage that, since all hypotheses are at the same point in time, their scores can be compared directly. This is due to the fact that each hypothesis accounts for the same amount of time (same number of frames).

Training Discrete Observation HMMs

Training refers to the problem of finding $\{\pi(1), \boldsymbol{A}, \boldsymbol{B}\}$ such that the model, M, after an iteration of training, better represents the training data than the previous model. The number of states is usually not varied or reestimated, other than via the modification of the model inventory. The apriori probabilities of the likelihood of a model, $\pi(1)$, are normally not reestimated as well, since these typically come from the language model.

The first algorithm we will discuss is one based on the **Forward-Backward** algorithm (Baum-Welch Reestimation):

$$u_{j|i} \equiv \text{label for a transition from state } i \text{ to state } j$$

$$u_{\bullet|i} \equiv \text{set of transitions exiting state } i$$

$$u_{j|\bullet} \equiv \text{set of transitions entering } j$$

Also, $\underline{u}(t)$ denotes a random variable that models the transitions at time t and $\underline{y}_j(t)$ a random variable that models the observation being emitted at state j at time t. The symbol "•" is used to denote an arbitrary event.

Next, we need to define some intermediate quantities related to particular events at a given state at a given time:

$$\begin{aligned}\zeta(i, j; t) &\equiv P(\underline{u}(t) = u_{j|i} \,|\, y, M) \\ &= P(\underline{u}(t) = u_{j|i}, y \,|\, M) / P(y|M) \\ &= \begin{cases} \dfrac{\alpha(y_1^t, i)\, a(j|i)\, b(y(t+1)|j)\, \beta(y_{t+2}^T \,|\, j)}{P(y|M)}, & t = 1, 2, \ldots, T \\ 0, & \text{other } t \end{cases}\end{aligned}$$

where the sequences α, β, a, and b were defined previously (last lecture). Intuitively, we can think of this as the probability of observing a transition from state i to state j at time t for a particular observation sequence, y, (the utterance in progress), and model M.

We can also make the following definition:

$$\gamma(i;t) \equiv P(\underline{u}(t) \in u_{\bullet|i} \,|\, y,M) = \sum_{j=1}^{S} \zeta(i,j;t)$$

$$= \left\{ \begin{array}{ll} \dfrac{\alpha(y_1^t, i)\beta(y_{t+1}^T \,|\, i)}{P(y|M)}, & t = 1, 2, \ldots, T \\ 0, & \text{other } t \end{array} \right\}$$

This is the probability of exiting state i. Also,

$$\nu(j;t) \equiv P(\underline{x}(t) = j \,|\, y,M)$$

$$= \left\{ \begin{array}{ll} \gamma(j;t), & t = 1, 2, \ldots, T \\ \alpha(y_1^T, j), & t = T \\ 0, & \text{other } t \end{array} \right\}$$

$$= \left\{ \begin{array}{ll} \dfrac{\alpha(y_1^t, j)\beta(y_{t+1}^T \,|\, j)}{P(y|M)}, & t = 1, 2, \ldots, T \\ 0, & \text{other } t \end{array} \right\}$$

which is the probability of being in state j at time t. Finally,

$$\delta(j,k;t) \equiv P(\underline{y}_j(t) = k \,|\, y,M)$$

$$= \left\{ \begin{array}{ll} \nu(j;t), & \text{if } y(t) = k \text{ and } 1 \le t \le T \\ 0, & \text{otherwise} \end{array} \right\}$$

$$= \left\{ \begin{array}{ll} \dfrac{\alpha(y_1^t, j)\beta(y_{t+1}^T \,|\, j)}{P(y|M)}, & \text{if } y(t) = k \text{ and } 1 \le t \le T \\ 0, & \text{otherwise} \end{array} \right\}$$

which is the probability of observing symbol k at state j at time t.

Note that we make extensive use of the forward and backward probabilities in these computations. This will be key to reducing the complexity of the computations by allowing an interactive computation.

From these four quantities, we can define four more intermediate quantities:

$$\zeta(i,\, j;\bullet) \;=\; P(\underline{u}(\bullet) \in u_{j|i} \,\big|\, y, M) \;=\; \sum_{t\,=\,1}^{T} \zeta(i,\, j;t)$$

$$\gamma(i;\bullet) \;=\; P(\underline{u}(\bullet) \in u_{\bullet|i} \,\big|\, y, M) \;=\; \sum_{t\,=\,1}^{T} \gamma(i;t)$$

$$\nu(j;\bullet) \;=\; P(\underline{u}(\bullet) \in u_{j|\bullet} \,\big|\, y, M) \;=\; \sum_{t\,=\,1}^{T} \nu(j;t)$$

$$\delta(j,\, k;\bullet) \;=\; P(\underline{y}_j(\bullet) = k \,\big|\, y, M) \;=\; \sum_{t\,=\,1}^{T} \delta(j,\, k;t) \;=\; \sum_{\substack{t\,=\,1 \\ y(t)\,=\,k}}^{T} \nu(j;t)$$

Finally, we can begin relating these quantities to the problem of reestimating the model parameters. Let us define four more random variables:

$$\underline{n}(u_{j|i}) \equiv \text{number of transitions of the type } u_{j|i}$$

$$\underline{n}(u_{\bullet|i}) \equiv \text{number of transitions of the type } u_{\bullet|i}$$

$$\underline{n}(u_{j|\bullet}) \equiv \text{number of transitions of the type } u_{j|\bullet}$$

$$\underline{n}(\underline{y}_j(\bullet) = k) \equiv \text{number of times the observation } k \text{ and state } j \text{ jointly occur}$$

We can see that:

$$\zeta(i,\, j;\bullet) \;=\; E\{\underline{n}(u_{j|i}) \,\big|\, y, M\}$$

$$\gamma(i;\bullet) \;=\; E\{\underline{n}(u_{\bullet|i}) \,\big|\, y, M\}$$

$$\nu(j;\bullet) \;=\; E\{\underline{n}(u_{j|\bullet}) \,\big|\, y, M\}$$

$$\delta(j,\, k;\bullet) \;=\; E\{\underline{n}(\underline{y}_j(\bullet) = k) \,\big|\, y, M\}$$

What we have done up to this point is to develop expressions for the estimates of the underlying components of the model parameters in terms of the state sequences that occur during training.

But how can this be when the internal structure of the model is **hidden**?

Following this line of reasoning, an estimate of the transition probability is:

$$\bar{a}(j|i) = \frac{E\{\underline{n}(u_{j|i})\big|y,M\}}{E\{\underline{n}(u_{\bullet|i})\big|y,M\}} = \frac{\underline{\zeta}(i,\,j;\bullet)}{\gamma(i;\bullet)}$$

$$= \frac{\sum_{t=1}^{T-1} \alpha(y_1^t,i)a(j|i)b(y(t+1)|j)\beta(y_{t+2}^T\big|j)}{\sum_{t=1}^{T-1} \alpha(y_1^t,i)\beta(y_{t+1}^T\big|i)}$$

Similarly,

$$\bar{b}(k|j) = \frac{E\left\{\underline{n}(\underline{n}(\underline{y}_j(\bullet)=k)\big|y,M)\big|y,M\right\}}{E\{\underline{n}(u_{j|\bullet})\big|y,M\}} = \frac{\underline{\zeta}(i,\,j;\bullet)}{\gamma(i;\bullet)}$$

$$= \frac{\sum_{t=1}^{T} \alpha(y_1^t,j)\beta(y_t^T\big|j)}{\sum_{y(t)=k}^{T} \alpha(y_1^t,j)\beta(y_{t+1}^T\big|j)}$$

Finally,

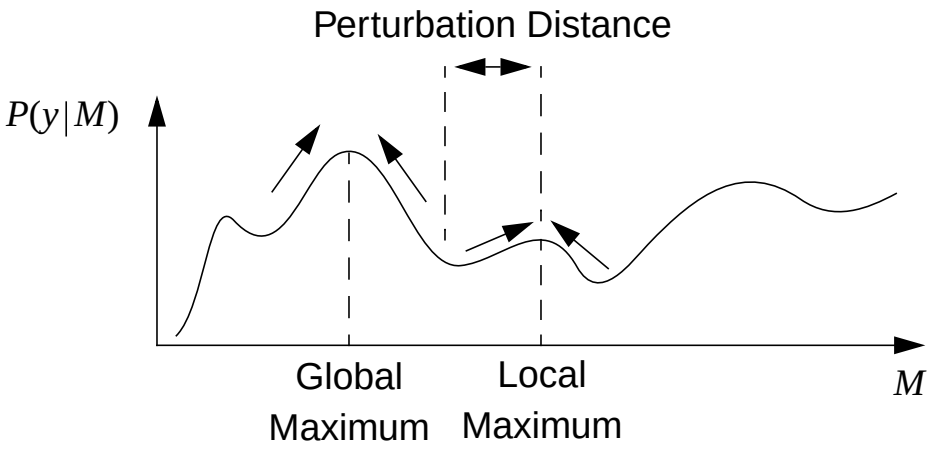
$$P(\underline{x}(1)=i) = \frac{\alpha(y_1^1,i)\beta(y_2^T\big|i)}{P(y|M)}$$

This process is often called reestimation by recognition, because we need to recognize the input with the previous models in order that we can compute the new model parameters from the set of state sequences used to recognize the data (hence, the need to iterate).

But will it converge? Baum and his colleagues showed that the new model guarantees that:

$$P(y|\overline{M}) \geq P(y|M)$$

Since this is a highly nonlinear optimization, it can get stuck in local minima:



We can overcome this by starting training from a different initial point, or "bootstrapping" models from previous models.

Analogous procedures exist for the **Viterbi algorithm**, though they are much simpler and more intuitive (and more DP-like):

$$\bar{a}(j|i) = \frac{E\{\underline{n}(u_{j|i})\big|y,M\}}{E\{\underline{n}(u_{\bullet|i})\big|y,M\}}$$

and,

$$\bar{b}(k|j) = \frac{E\{\underline{n}(u_{j|i})\big|y,M\}}{E\{\underline{n}(u_{\bullet|i})\big|y,M\}}$$

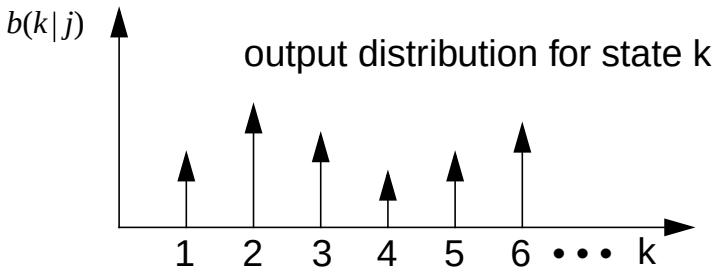
These have been shown to give comparable performance to the forward-backward algorithm at significantly reduced computation. It also is generalizable to alternate formulations of the topology of the acoustic model (or language model) drawn from formal language theory. (In fact, we can even eliminate the first-order Markovian assumption.)

Further, the above algorithms are easily applied to many problems associated with language modeling: estimating transition probabilities and word probabilities, efficient parsing, and learning hidden structure.

But what if a transition is never observed in the training database?

Continuous Density HMMs

The discrete HMM incorporates a discrete probability density function, captured in the matrix $\boldsymbol{B}$, 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, recently, the continuous density model has become widely accepted.
Let us assume a parametric model of the observation pdf:

$$M = \left\{ S, \pi(1), \boldsymbol{A}, \left\{ f_{\underline{y}|\underline{x}}(\xi|i), 1 \le i \le S \right\} \right\}$$

The likelihood of generating observation $\boldsymbol{y}(t)$ in state j is defined as:

$$b(\boldsymbol{y}(t)|j) \equiv f_{\underline{y}|\underline{x}}(\boldsymbol{y}(t)|j)$$

Note that taking the negative logarithm of $b(\ )$ will produce a log-likelihood, or a Mahalanobis-like distance. But what form should we choose for $f(\ )$? Let's assume a Gaussian model, of course:

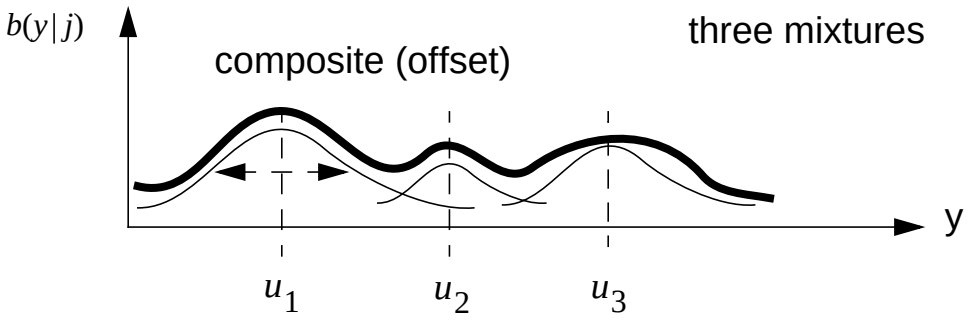
$$f_{\underline{y}|\underline{x}}(\boldsymbol{y}|i) = \frac{1}{\sqrt{2\pi|\boldsymbol{C}_i|}} \exp\left\{ -\frac{1}{2}(\boldsymbol{y}-\mu_i)^T \boldsymbol{C}_i^{-1}(\boldsymbol{y}-\mu_i) \right\}$$

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 Distributions

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:

$$f_{\underline{y}|\underline{x}}(\boldsymbol{y}|i) = \sum_{m=1}^{M} c_{im}\,\aleph(\boldsymbol{y};\mu_{im},\boldsymbol{C}_{im})$$

In order for this to be a valid pdf, the mixture coefficients must be nonnegative and satisfy the constraint:

$$\sum_{m=1}^{M} c_{im} = 1, \qquad 1 \le i \le S$$

Note that mixture distributions add significant complexity to the system: m means and covariances at each state.

Analogous reestimation formulae can be derived by defining the intermediate quantity:

$$\nu(i;t,l) \equiv P(\underline{x}(t) = i \,|\, \boldsymbol{y}(t) \text{produced in accordance with mixture } l)$$

$$= \frac{\alpha(\boldsymbol{y}_1^t,i)\beta(\boldsymbol{y}_{t+1}^T \,|\, i)}{\sum_{j=1}^{S} \alpha(\boldsymbol{y}_1^t,j)\beta(\boldsymbol{y}_{t+1}^T \,|\, j)} \times \frac{c_{il}\,\aleph(\boldsymbol{y}_{tl}^t;\mu_{il},\boldsymbol{C}_{il})}{\sum_{m=1}^{M} c_{im}\,\aleph(\boldsymbol{y}_t^t;\mu_{im},\boldsymbol{C}_{im})}$$

The mixture coefficients can now be reestimated using:

$$\bar{c}_{il} = \frac{\nu(i;\bullet,l)}{\sum_{m=1}^{M} \nu(i;\bullet,m)}$$

the mean vectors can be reestimated as:

$$\bar{\mu}_{il} = \frac{\sum_{t=1}^{T} \nu(i;t,l)\boldsymbol{y}(t)}{\nu(i;\bullet,l)}$$

the covariance matrices can be reestimated as:

$$\bar{\boldsymbol{C}}_{il} = \frac{\sum_{t=1}^{T} \nu(i;t,l)[\boldsymbol{y}(t)-\mu_{il}][\boldsymbol{y}(t)-\mu_{il}]^T}{\nu(i;\bullet,l)}$$

and the transition probabilities, and initial probabilities are reestimated as usual.

The Viterbi procedure once again has a simpler interpretation:

$$\mu_{il} = \frac{1}{N_{il}} \sum_{\substack{t=1 \\ y(t) \sim il}}^{T} \boldsymbol{y}(t)$$

and

$$\boldsymbol{C}_{il} = \frac{1}{N_{il}} \sum_{\substack{t=1 \\ y(t) \sim il}}^{T} [\boldsymbol{y}(t)-\mu_{il}][\boldsymbol{y}(t)-\mu_{il}]^T$$

The mixture coefficient is reestimated as the number of vectors associated with a given mixture at a given state:

$$c_{il} = \frac{N_{il}}{N_i}$$

Session VII:

Acoustic Modeling and Training

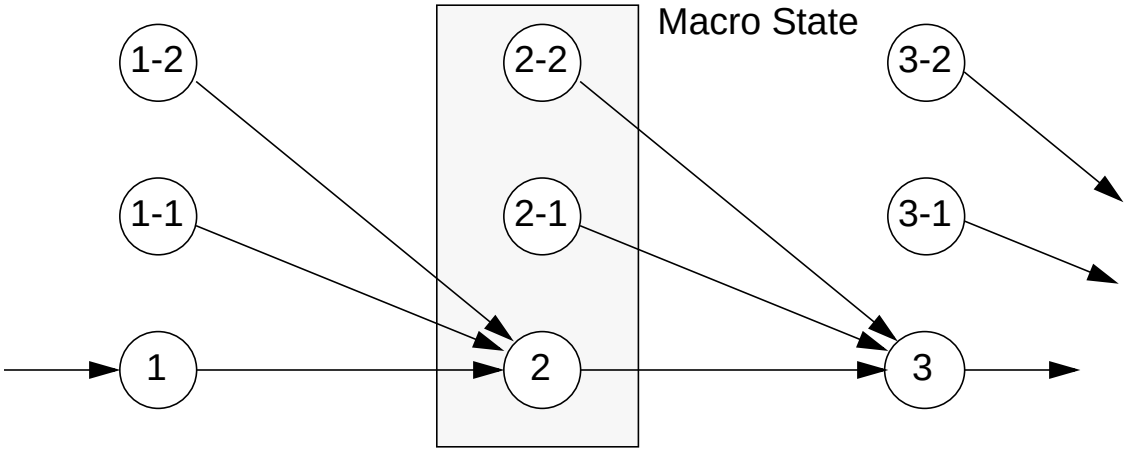
State Duration Probabilities

Recall that the probability of staying in a state was given by an exponentially-decaying distribution:

$$P(\bar{O}|Model, q_1 = i) = P(\bar{O}, q_1 = i|Model)/P(q_1 = i) = a_{ii}^{\,d-1}(1 - a_{ii})$$

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)

$$P(\underline{d}_i = d) = \tau_d \qquad 1 \le d \le D$$

- 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.

Scaling in HMMs

As difficult as it may seem to believe, standard HMM calculations exceed the precision of 32-bit floating point numbers for the simplest of models. The large numbers of multiplications of numbers less than one leads to underflow. Hence, we must incorporate some form of scaling.

It is possible to scale the forward-backward calculation (see Section 12.2.5) by normalizing the calculations by:

$$c(t) = \frac{1}{\sum_{i=1}^{S} \tilde{\alpha}(y_1^t, i)}$$

at each time-step (time-synchronous normalization).

However, a simpler and more effective way to scale is to deal with log probabilities, which work out nicely in the case of continuous distributions. Even so, we need to somehow prevent the best path score from growing with time (increasingly negative in the case of log probs). Fortunately, at each time step, we can normalize all candidate best-path scores, and proceed with the dynamic programming search. More on this later...

Similarly, it is often desirable to trade-off the important of transition probabilities and observation probabilities. Hence, the log-likelihood of an output symbol being observed at a state can be written as:

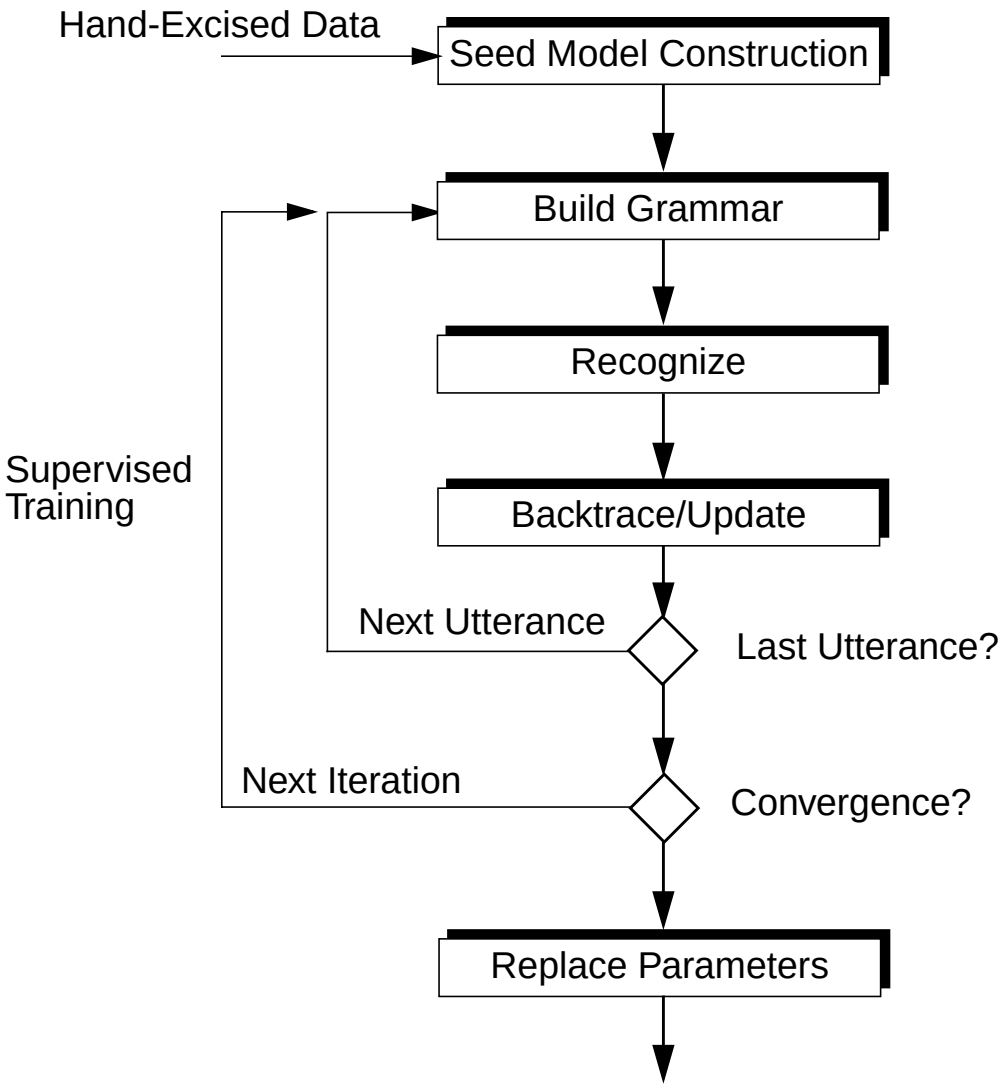
$$P(y_1^t, (\underline{x}(t) = i) \,|\, y, M) = P(\underline{x}(t-1) = j \,|\, y, M)(a_{ij})^{\alpha} + b_i(\mathbf{y}(t))^{\beta}$$

or, in the log prob space:

$$\log\left\{P(y_1^t, \ldots)\right\} = \log\{P(\underline{x}(t-1) = j)\ldots\} + \alpha\log(a_{ij}) + \beta\log D_{y,\mu}$$

This result emphasizes the similarities between HMMs and DTW. The weights, α and β can be used to control the importance of the “language model.”

An Overview of the Training Schedule



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?

Alternative Criteria For Optimization

As we have previously seen, an HMM system using the standard Baum-Welch reestimation algorithm learns to emulate the statistics of the training database. We refer to this training mode as “representation.” This is not necessarily equivalent to minimizing the recognition error rate. It depends to a great deal on the extent to which the statistics of the training database match the test database. Often, especially in open-set speaker independent cases, the test database contains data never seen in the training database.

A potential solution to this problem is to attempt to force the recognizer to learn to discriminate (reduce recognition errors explicitly). This is analogous to developing a system to make an M-way choice in an N-dimensional space by learning decision boundaries rather than clustering the data and modeling the statistics of each cluster.

One approach to modifying the HMM structure to do this is to use a different cost function. For example, we can minimize the discrimination information (or equivalently the cross-entropy) between the data and the statistical characterizations of the data implied by the model.
Recall the definition of the discrimination information:

$$J_{DI} = \int_{-\infty}^{\infty} f(\underline{\boldsymbol{y}}|y)\log\frac{f(\underline{\boldsymbol{y}}|\boldsymbol{y})}{f(\underline{\boldsymbol{y}}|M)}d\underline{\boldsymbol{y}}$$

Note that a small DI is good, because it implies a given measurement matches both the signal statistics and the model statistics (which means the relative mismatch between the two is small).

However, such approaches have proven to be intractable — leading to highly nonlinear equations. A more successful approach has been to maximize the average mutual information. Recall our definition for the average mutual information:

$$\overline{M}(\underline{y},\underline{M}) = \sum_{l=1}^{L}\sum_{r=1}^{R} P(\underline{y} = y^{l},\underline{M} = M_r)\log\left[\frac{P(\underline{y} = y^{l},\underline{M} = M_r)}{P(\underline{y} = y^{l})P(\underline{M} = M_r)}\right]$$

This can be written as:

$$\begin{aligned}\overline{M}(\underline{y},\underline{M}) &= \sum_{l=1}^{L}\sum_{r=1}^{R} P(\underline{y}=y^l,\underline{M}=M_r)\times \\ &\quad \log[P(\underline{y}=y^l,\underline{M}=M_r)]-\log[P(\underline{y}=y^l)] \\ &= \sum_{l=1}^{L}\sum_{r=1}^{R} P(\underline{y}=y^l,\underline{M}=M_r)\times \\ &\quad [\log P(\underline{y}=y^l,\underline{M}=M_r)- \\ &\quad \log\sum_{m=1}^{R} P(\underline{y}=y^l\,|\,\underline{M}=M_m)P(\underline{M}=M_m)\ ]\end{aligned}$$

Note that the last term constitutes a rewrite of $P(\underline{y}=y^l)$.

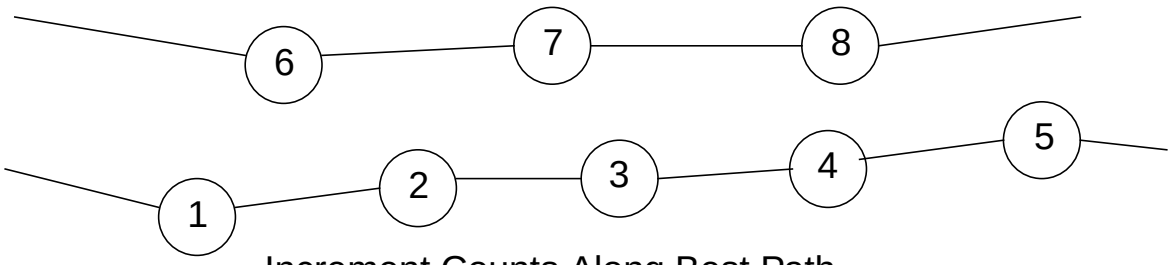
If we assume there is exactly one training string (which is the error we want to correct), and it is to be used to train M_l, then, if we assume $P(\underline{y}=y^l\,|\,\underline{M}=M_r)\approx\delta(l-r)$, we can approximate $\overline{M}(\underline{y},\underline{M})$ by:

$$\overline{M}(\underline{y},\underline{M})\approx \sum_{l=1}^{L} P(\underline{y}=y^l,\underline{M}=M_l)\ -\log\sum_{m=1}^{R} P(\underline{y}=y^l\,|\,\underline{M}=M_m)P(\underline{M}=M_m)$$

The first term in the summation corresponds to the probability of correct recognition, which we want to maximize. The second term corresponds to the probability of incorrect recognition, which we want to minimize.

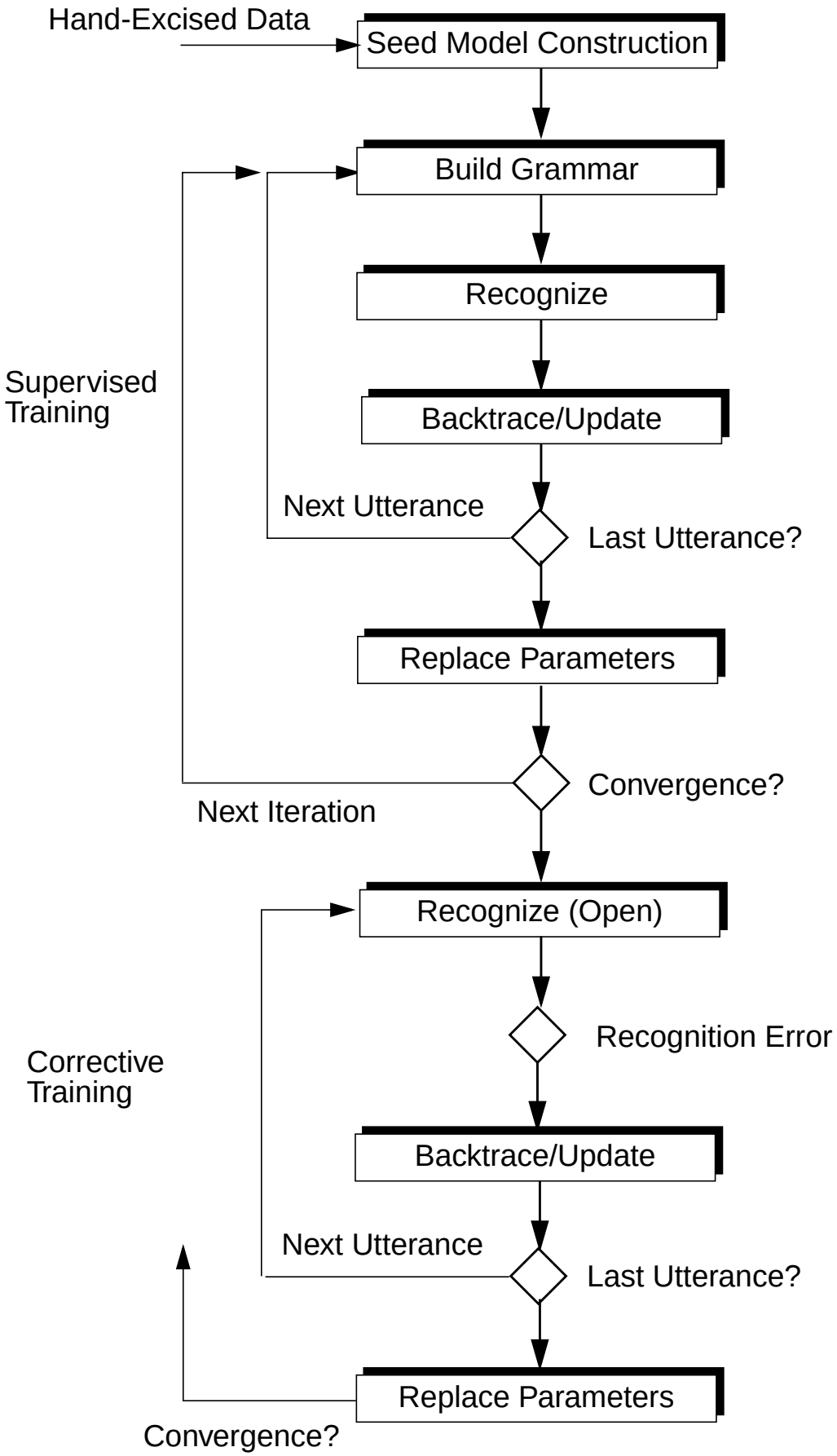
This method has a rather simple interpretation for discrete HMMs:

Decrement Counts Along Competing Incorrect Paths



Increment Counts Along Best Path

An Overview of the Corrective Training Schedule



Unfortunately, this method is not readily extensible to continuous speech, and has proved inconclusive in providing measurable reductions in error rate. However, discriminative training algorithms continues to be an important are of research in HMMs.

Later, when we study neural networks, we will observe that some of the neural network approaches are ideally suited towards implementation of discriminative training.

Distance Measures for HMMs

Recall the KMEANS clustering algorithm:

Initialization: Choose K centroids

Recursion:
1. Assign all vectors to their nearest neighbor.
2. Recompute the centroids as the average of all vectors assigned to the same centroid.
3. Check the overall distortion. Return to step 1 if some distortion criterion is not met.

Clustering of HMM models is often important in reducing the number of context-dependent phone models (which can often approach 10,000 for English) to a manageable number (typically a few thousand models are used). We can use standard clustering algorithms, but we need some way of computing the distance between two models.

A useful distance measure can be defined as:

$$D(M_1,M_2) \equiv \frac{1}{T_2}[\log P(y^2 \mid M_1) - \log P(y^2 \mid M_2)]$$

where y^2 is a sequence generated by M_2 of length T_2. Note that this distance metric is not symmetric: $D(M_1,M_2) \neq D(M_2,M_1)$

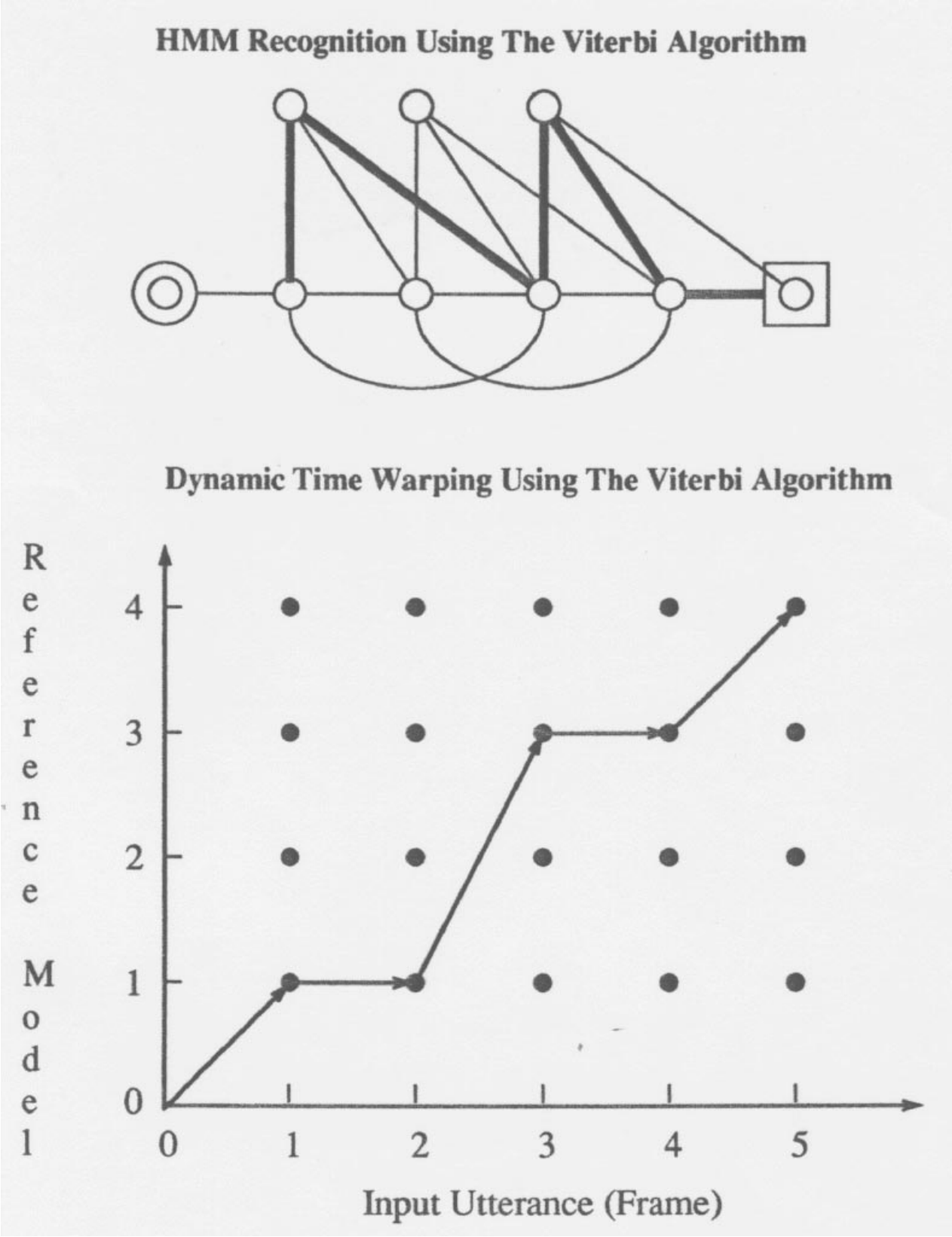
A symmetric version of this is:

$$D'(M_1,M_2) = \frac{D(M_1,M_2) + D(M_2,M_1)}{2}$$

The sequence y^2 is often taken to be the sequence of mean vectors associated with each state (typically for continuous distributions).

Often, phonological knowledge is used to cluster models. Models sharing similar phonetic contexts are merged to reduce the complexity of the model inventory. Interestingly enough, this can often be performed by inserting an additional network into the system that maps context-dependent phone models to a pool of states.

The DTW Analogy



Alternative Acoustic Model Topologies

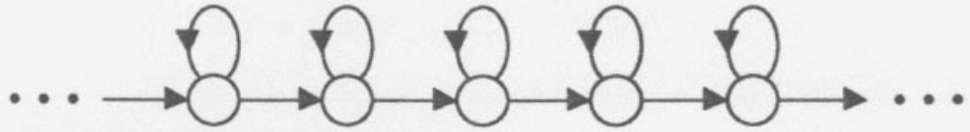


Figure 8(a). A simple progressive HMM topology. In general, the duration probability density function at a state has an exponential behavior.

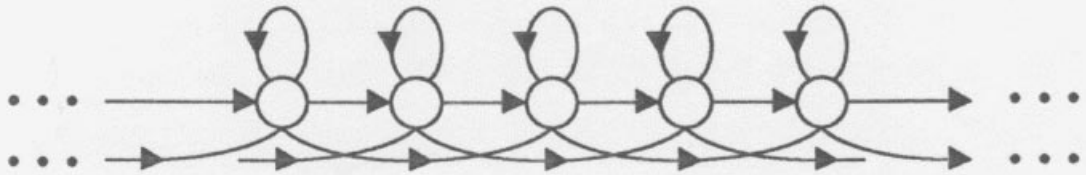


Figure 8(b). The Bakis topology (a progressive model with skip states).

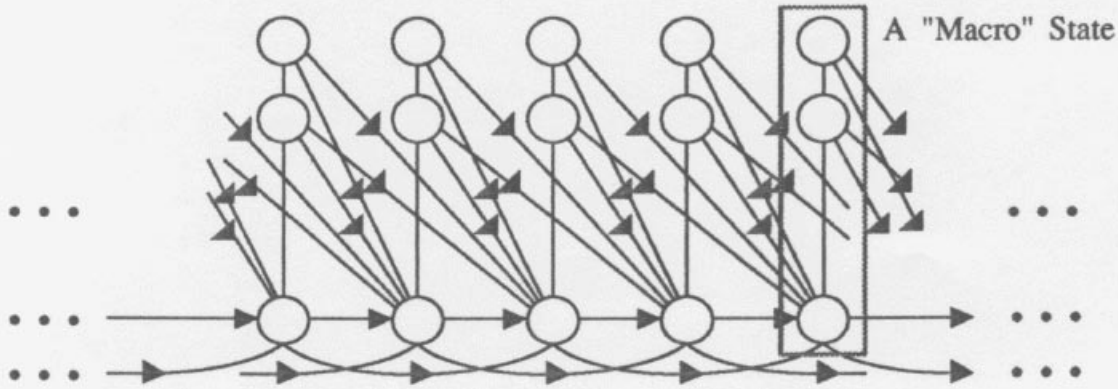


Figure 8(c). A finite duration topology. This topology is most analogous to DTW.

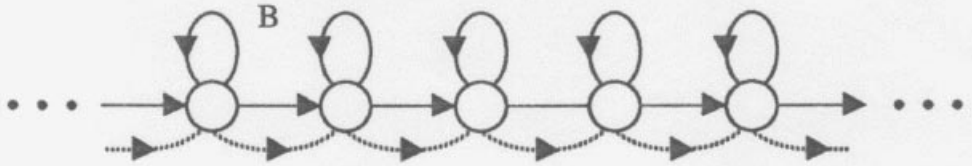


Figure 8(d). A fenonic baseform topology. The dashed line indicates a transition that produces no output.

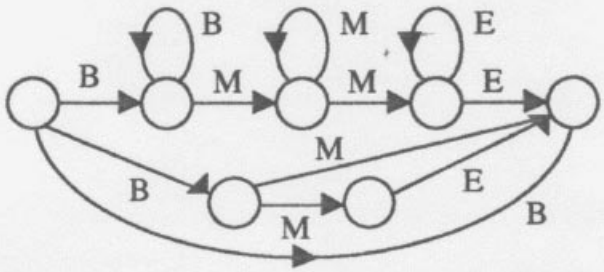
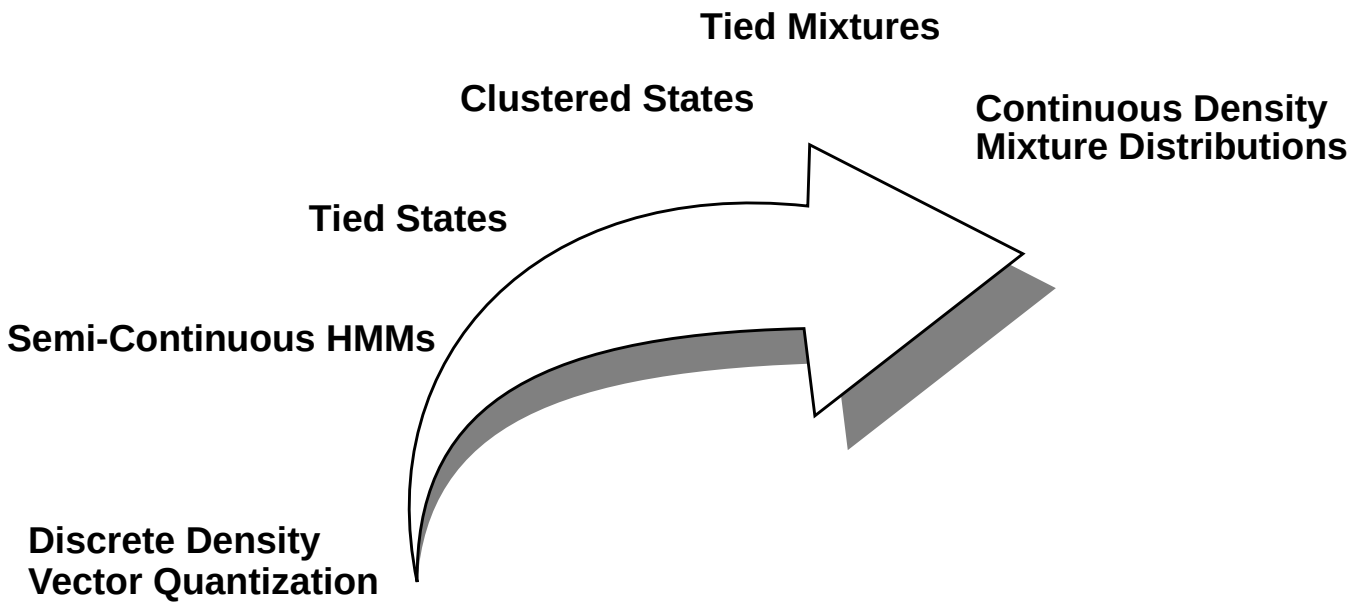
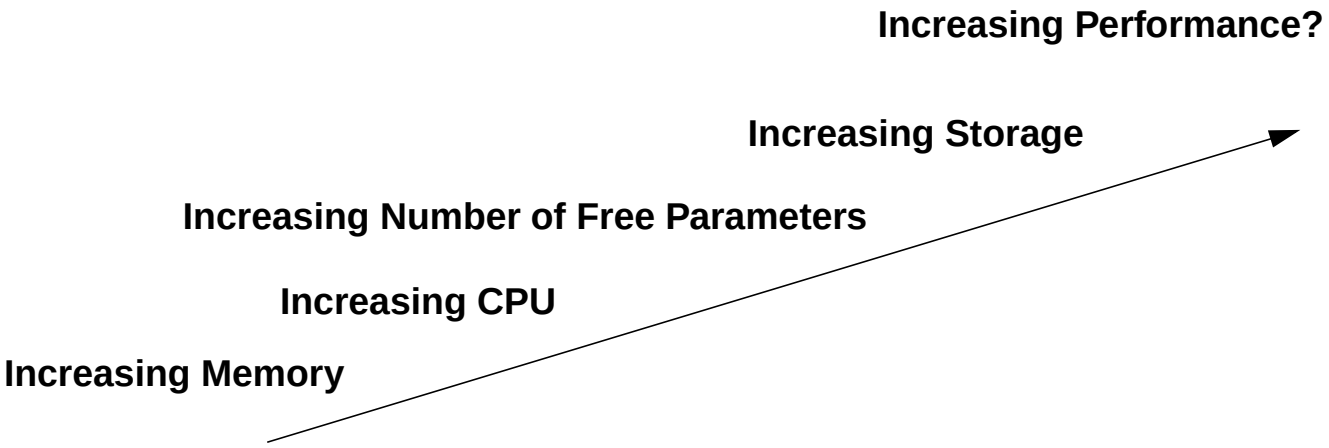


Figure 8(e). A modified fenonic baseform with tied transitions. Transitions in the same group share output probabilities.

Managing Model Complexity

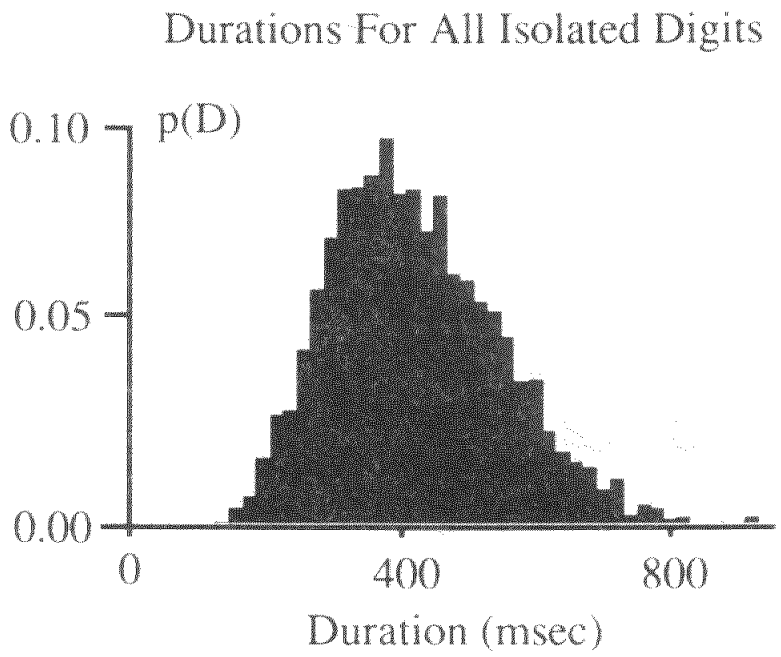




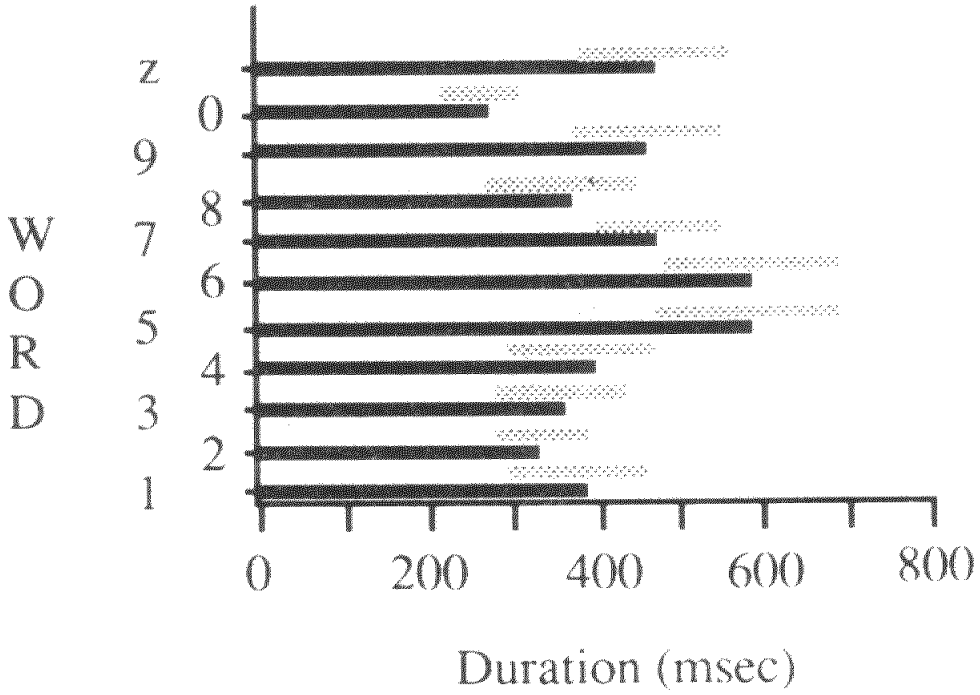
- Numerous techniques to robustly estimate model parameters; among the most popular is deleted interpolation:

$$\boldsymbol{A} = \varepsilon_t \boldsymbol{A}_t + (1-\varepsilon_t)\boldsymbol{A}_u$$
$$\boldsymbol{B} = \varepsilon_t \boldsymbol{B}_t + (1-\varepsilon_t)\boldsymbol{B}_u$$

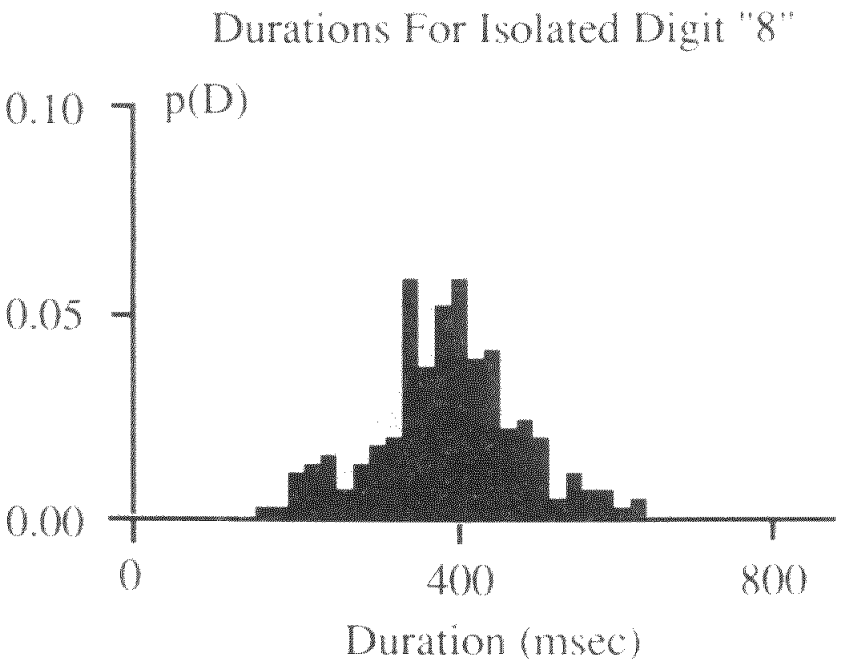
Seed Model Construction: Duration Distributions



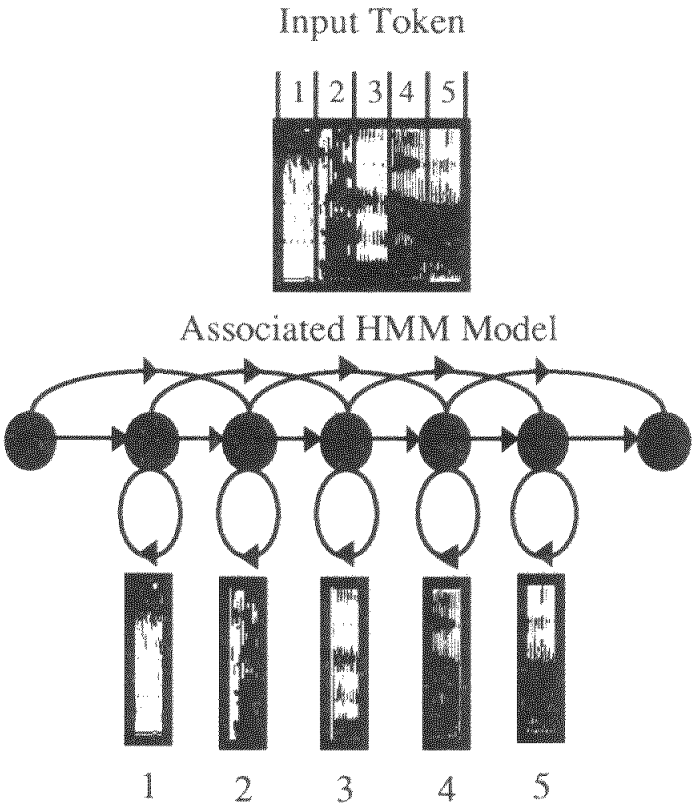
Mean: 408 msec
Standard Deviation: 125 msec



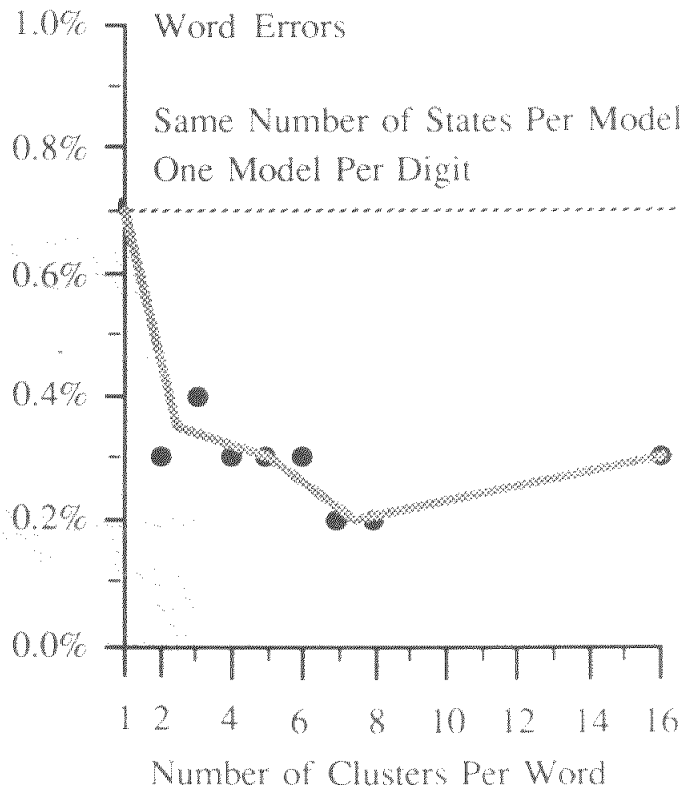
Seed Model Construction: Number of States Proportional to Duration

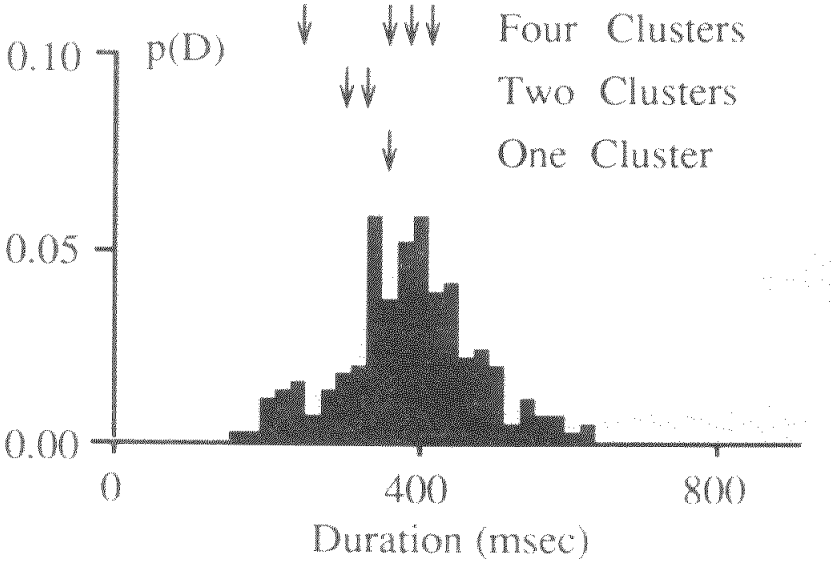


Mean: 366 msec
Standard Deviation: 88 msec



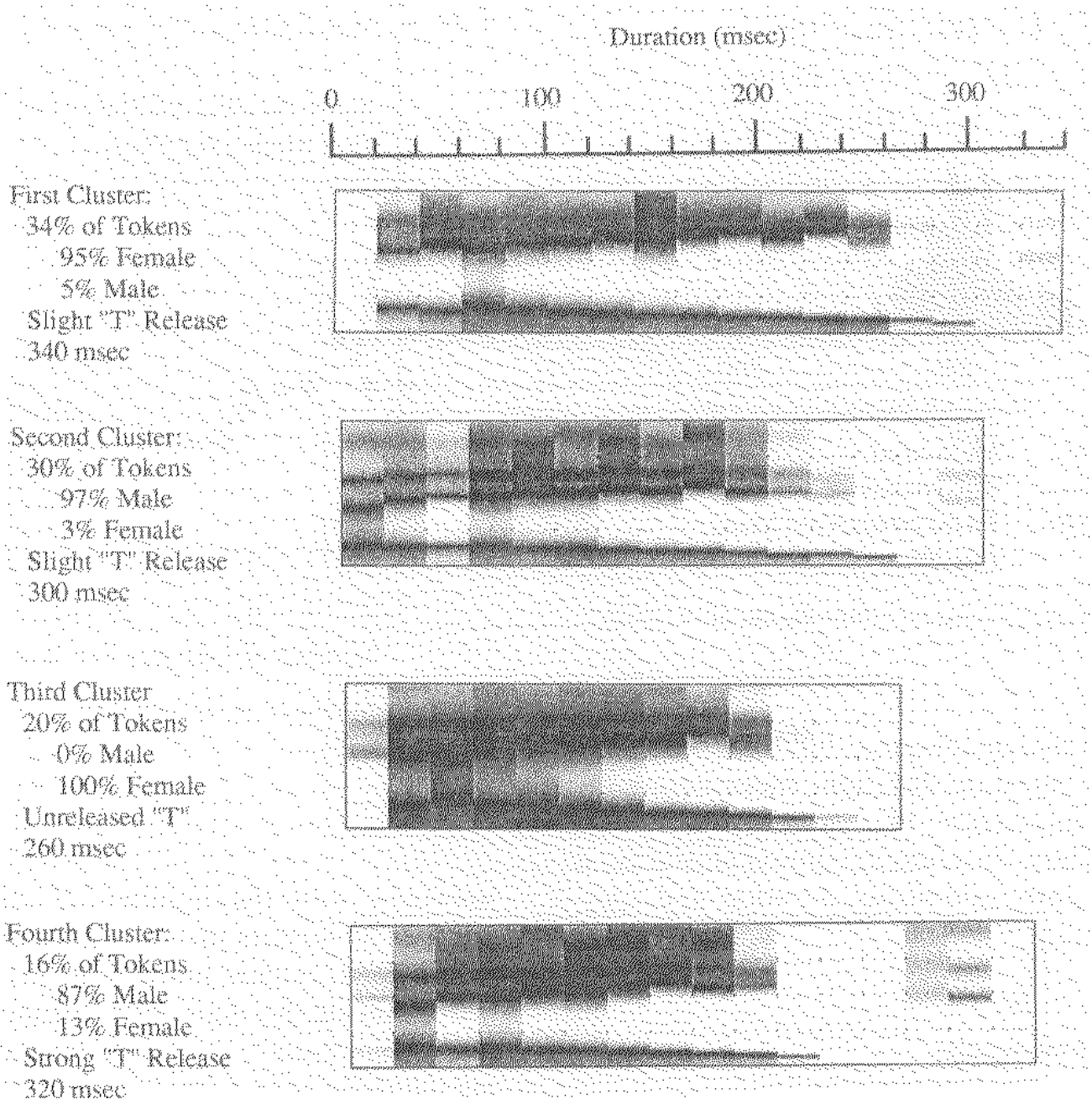
Duration in Context Clustering





One Cluster: Male Seed
Two Clusters: One Male/One Female
Four Clusters: Two Female/One Male/
One Short Model

Examples of Word Models



Session VIII:

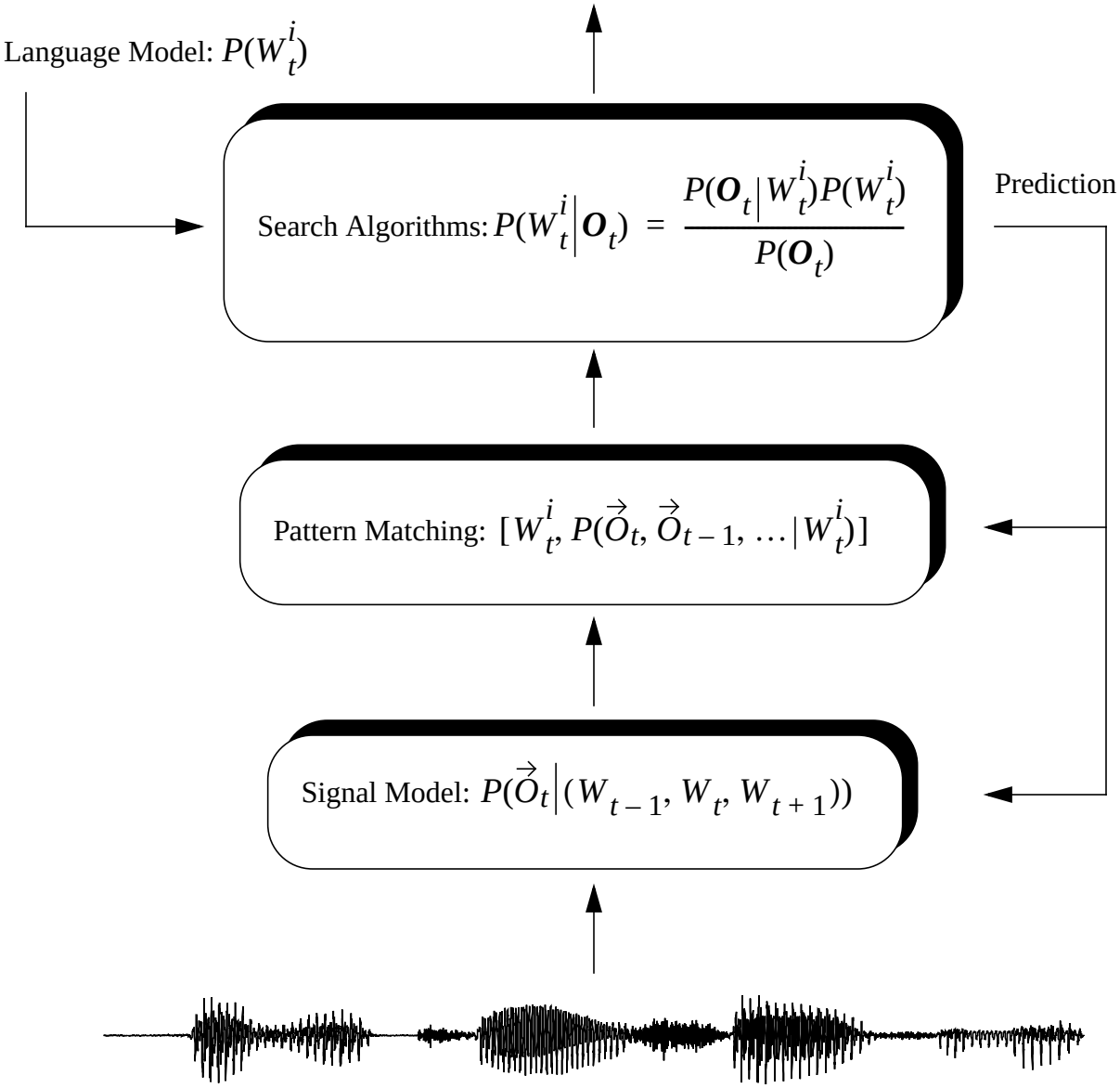
Speech Recognition Using Hidden Markov Models

BASIC TECHNOLOGY:

A PATTERN RECOGNITION PARADIGM

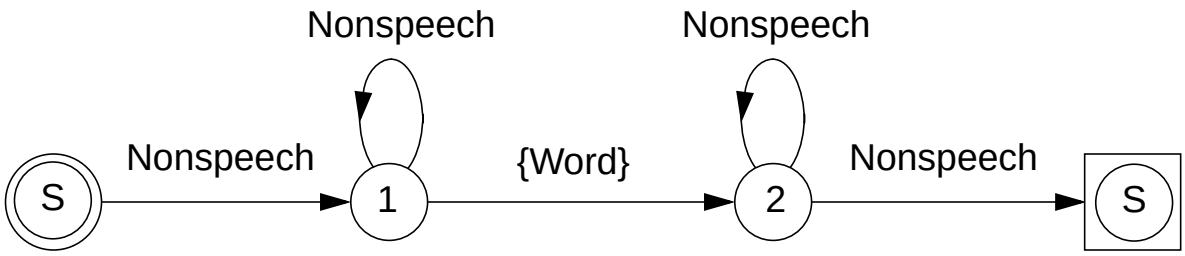
BASED ON HIDDEN MARKOV MODELS

Recognized Symbols: $P(S|\boldsymbol{O}) = \arg max|_T \prod_i P(W_t^i \Big| (\vec{O}_t, \vec{O}_{t-1}, \ldots))$



High Performance Isolated Word Recognition Using A Continuous Speech Recognizer

Isolated Word Recognition:

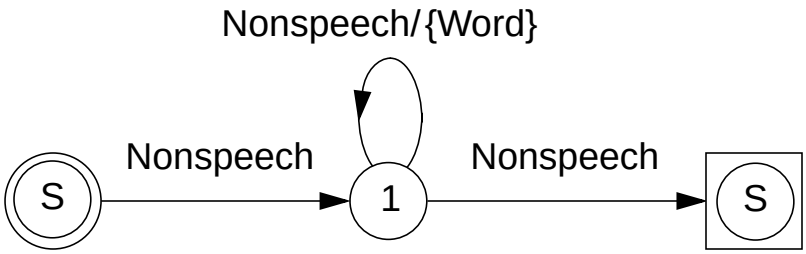


Nonspeech: typically an acoustic model of one frame in duration that models the background noise.

{Word}: any word from the set of possible words that can be spoken

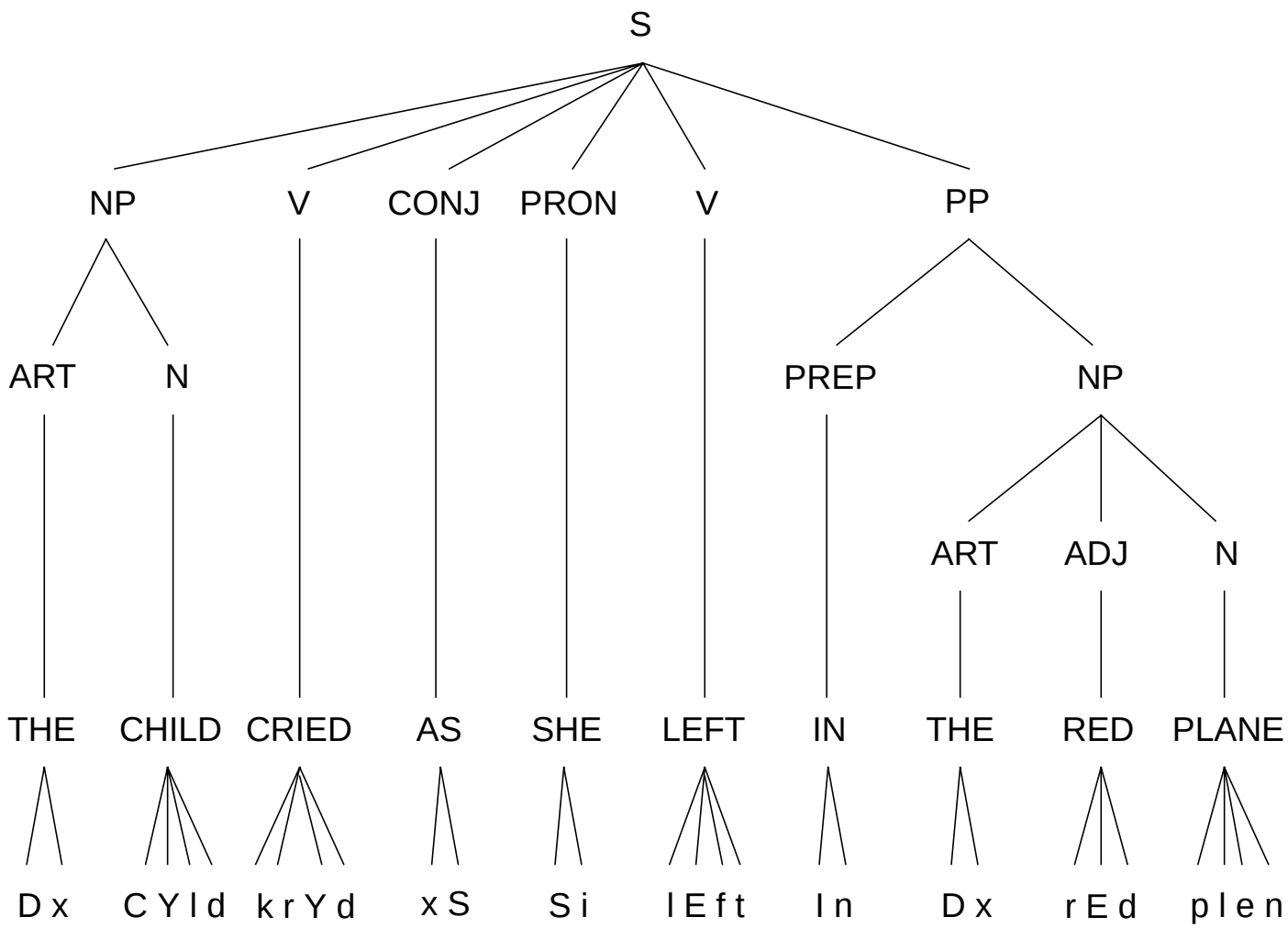
- The key point here is that, with such a system, the recognizer finds the optimal start/stop times of the utterance with respect to the acoustic model inventory (a hypothesis-directed search)

Simple Continuous Speech Recognition ("No Grammar"):



- system recognizes arbitrarily long sequences of words or nonspeech events

The Great Debate: “Bottom-Up” or “Top-Down”



Parsing refers to the problem of determining if a given sequence could have been generated from a given state machine.

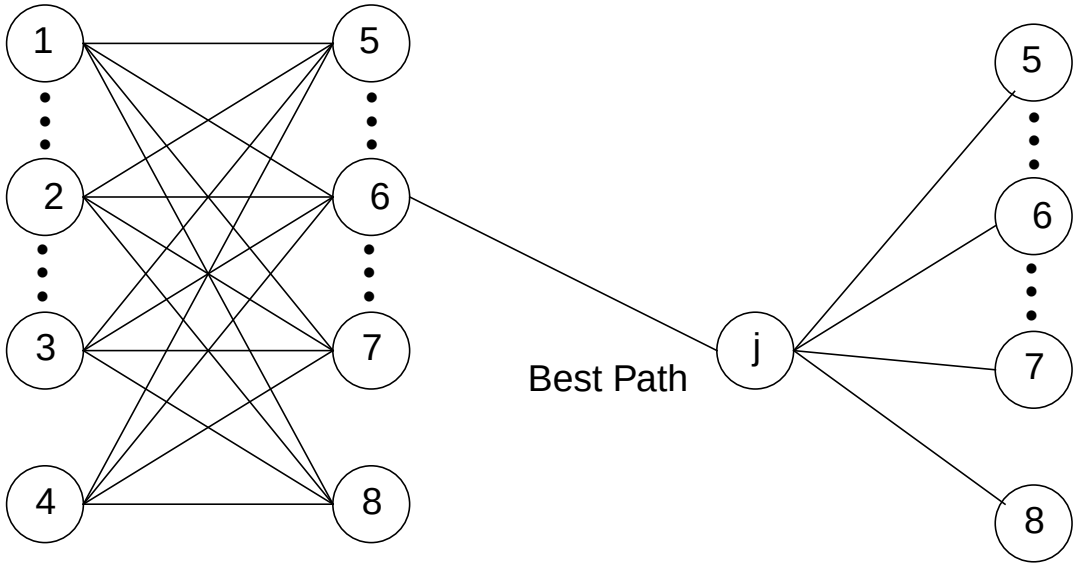
This computation, as we shall see, typically requires an elaborate search of all possible combinations of symbols output from the state machine.

This computation can be efficiently performed in a “bottom-up” fashion if the probabilities of the input symbols are extremely accurate, and only a few symbols are possible at each point at the lower levels of the tree.

If the input symbols are ambiguous, “top-down” parsing is typically preferred.

Network Searching and Beam Search

Premise: Suboptimal solutions are useful; global optimums are hard to find.



The network search considers several competing constraints:

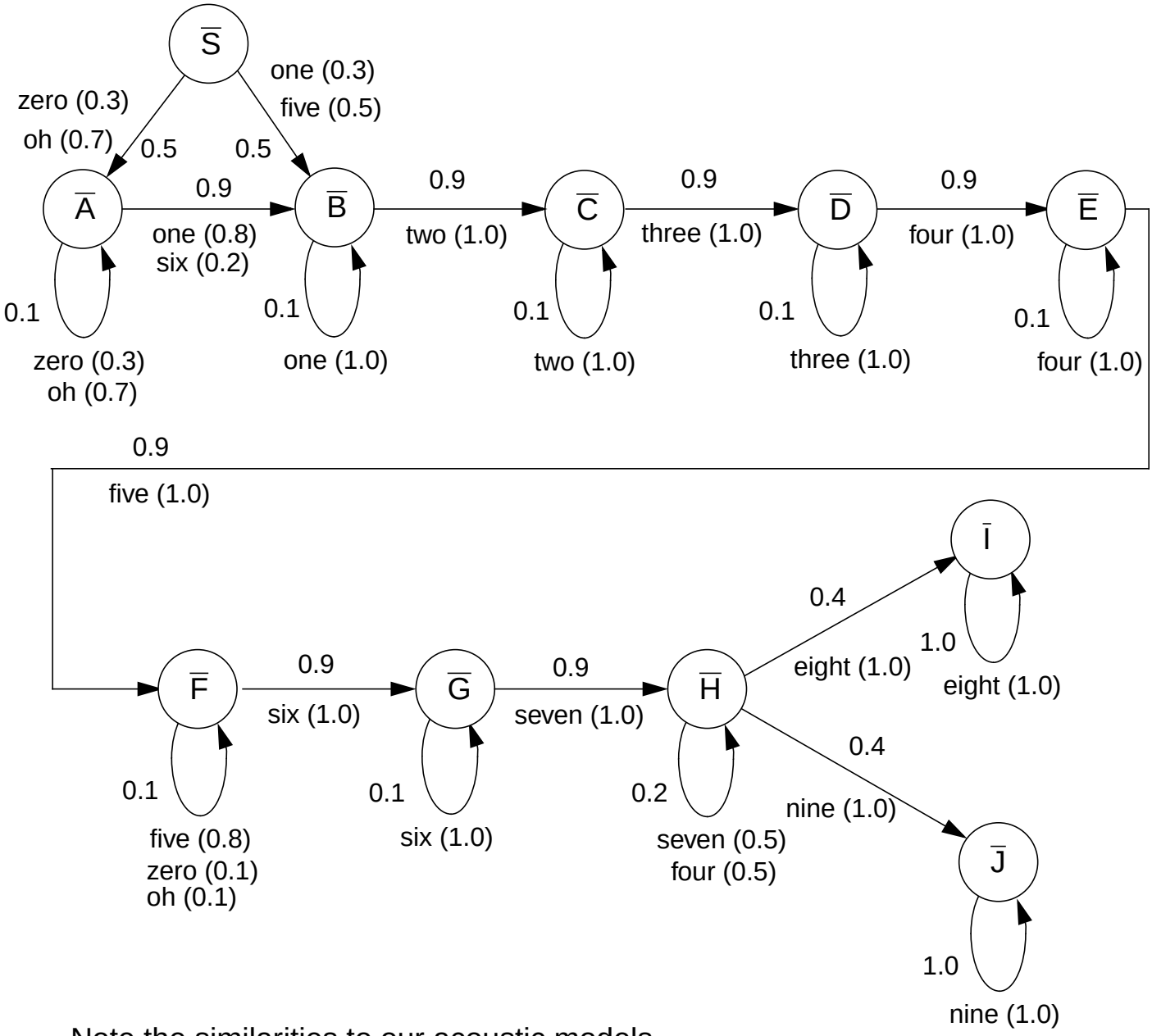
- The longer the hypothesis, the lower the probability
- The overall best hypothesis will not necessarily be the best initial hypothesis
- We want to limit the evaluation of the same substring by different hypotheses (difficult in practice)
- We would like to maintain as few active hypotheses as possible
- All states in the network will not be active at all times

Popular Search Algorithms

- Time Synchronous (Viterbi, DP)
- State Synchronous (Baum-Welch)
- Stack Decoding (N-Best, Best-First, A^*)
- Hybrid Schemes
  - — Forward-Backward
  - — Multipass
  - — Fast Matching

Generalization of the HMM

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 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 more *linguistic* information.

We can also express this system as a *regular grammar*:

$\overline{S} \xrightarrow{p_1}$ zero, $\overline{A}$
$\overline{S} \xrightarrow{p_2}$ oh, $\overline{A}$
$\overline{S} \xrightarrow{p_3}$ one, $\overline{A}$
$\overline{S} \xrightarrow{p_4}$ five, $\overline{A}$
$\overline{A} \xrightarrow{p_5}$ zero, $\overline{A}$
$\overline{A} \xrightarrow{p_6}$ oh, $\overline{A}$
$\overline{A} \xrightarrow{p_7}$ one, $\overline{B}$
$\overline{A} \xrightarrow{p_8}$ six, $\overline{B}$
$\overline{B} \xrightarrow{p_9}$ one, $\overline{B}$
$\overline{B} \xrightarrow{p_{10}}$ two, $\overline{C}$
$\overline{C} \xrightarrow{p_{11}}$ two, $\overline{C}$
$\overline{C} \xrightarrow{p_{12}}$ three, $\overline{D}$
$\overline{D} \xrightarrow{p_{13}}$ three, $\overline{D}$
$\overline{D} \xrightarrow{p_{14}}$ four, $\overline{E}$
$\overline{E} \xrightarrow{p_{15}}$ four, $\overline{E}$
$\overline{E} \xrightarrow{p_{16}}$ five, $\overline{F}$
$\overline{F} \xrightarrow{p_{17}}$ zero, $\overline{F}$
$\overline{F} \xrightarrow{p_{18}}$ oh, $\overline{F}$
$\overline{F} \xrightarrow{p_{19}}$ five, $\overline{F}$
$\overline{F} \xrightarrow{p_{20}}$ six, $\overline{G}$
$\overline{G} \xrightarrow{p_{21}}$ six, $\overline{G}$
$\overline{G} \xrightarrow{p_{22}}$ seven, $\overline{H}$
$\overline{H} \xrightarrow{p_{23}}$ seven, $\overline{H}$
$\overline{H} \xrightarrow{p_{24}}$ four, $\overline{H}$
$\overline{H} \xrightarrow{p_{25}}$ eight, $\bar{I}$
$\overline{H} \xrightarrow{p_{26}}$ nine, $\overline{J}$
$\bar{I} \xrightarrow{p'_{27}}$ eight, $\bar{I}$
$\bar{I} \xrightarrow{p''_{27}}$ eight.
$\bar{I} \xrightarrow{p'_{28}}$ nine, $\overline{J}$
$\bar{I} \xrightarrow{p''_{28}}$ nine.

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.

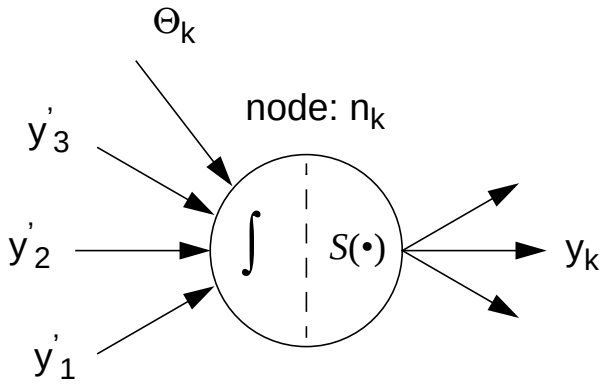
The Artificial Neural Network (ANN)

- 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

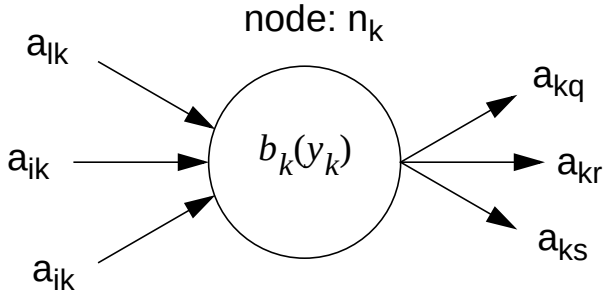
The Artificial Neuron — Nonlinear



vector input **scalar output**

$$y_k \equiv S\left(\sum_{n=1}^{N} \boldsymbol{w}_{ki}\boldsymbol{y}'_i - \theta_k\right)$$

The HMM State — Linear



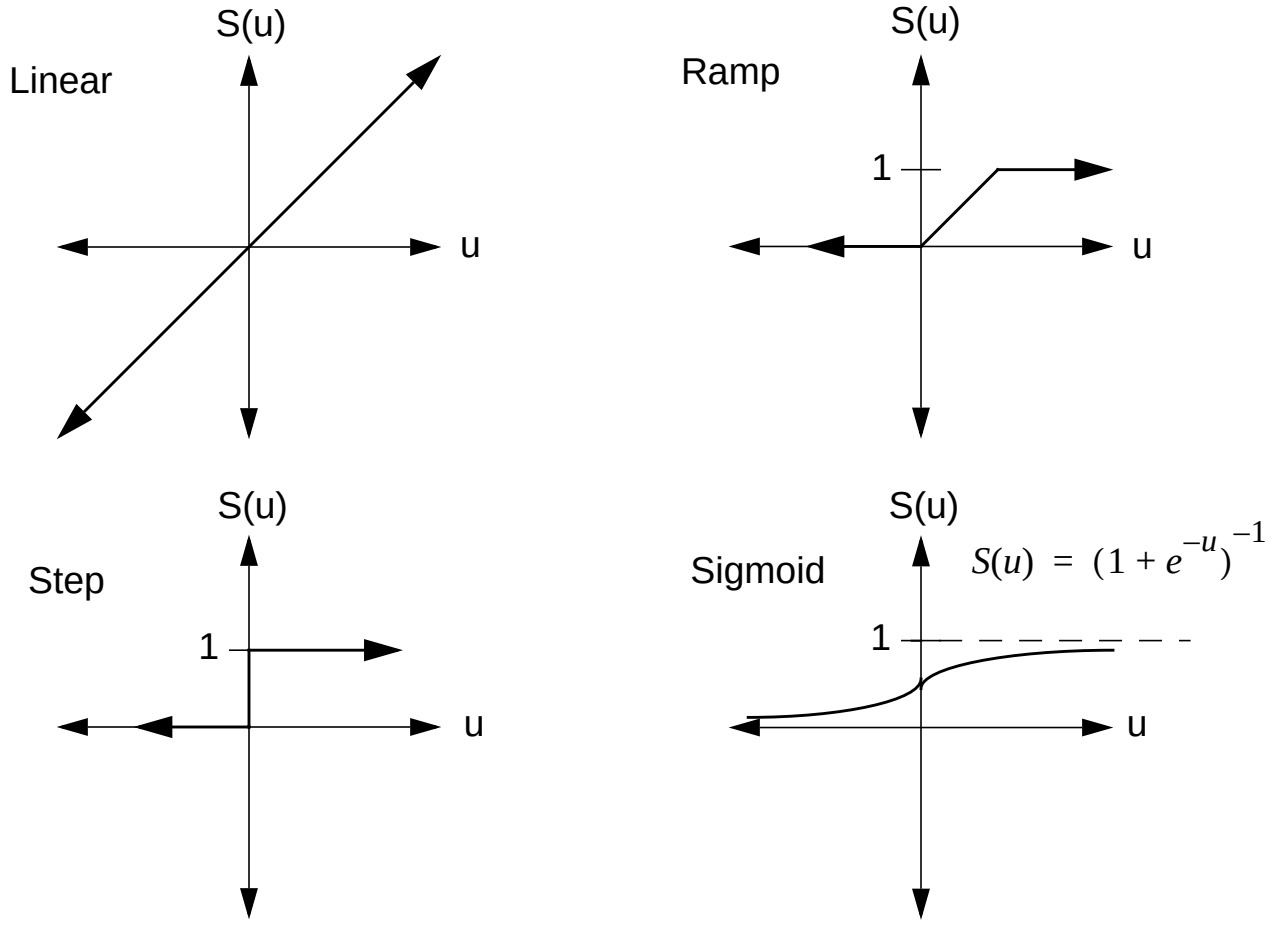
$$\alpha(y_1^{t+1}, j) = \alpha(y_1^t, i)a(j|i)b(y(t+1)|j)$$

Typical Thresholding Functions — A Key Difference

The input to the thresholding function is a weighted sum of the inputs:

$$u_k \equiv \boldsymbol{w}_k^T \boldsymbol{y}'$$

The output is typically defined by a nonlinear function:



Sometimes a bias is introduced into the threshold function:

$$y_k \equiv S(\boldsymbol{w}_k^T \boldsymbol{y}' - \theta_k) = S(u_k - \theta_k)$$

This can be represented as an extra input whose value is always -1:

$$y'_{N+1} = -1 \qquad w_{k,\,N+1} = \theta_k$$

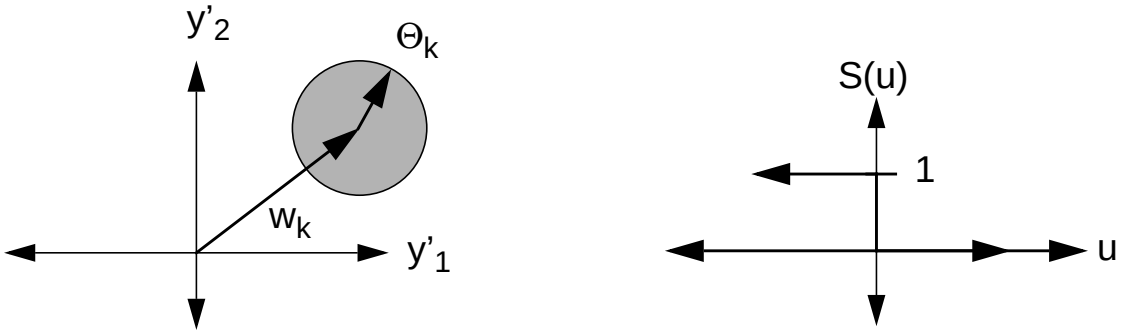
Radial Basis Functions

Another popular formulation involves the use of a Euclidean distance:

$$y_k = S(\sqrt{\sum_{i=1}^{N} (w_{ik} - y'_i)^2} - \theta_k) = S(\left\|\boldsymbol{w}_k - \boldsymbol{y}'\right\|_2 - \theta_k)$$

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?

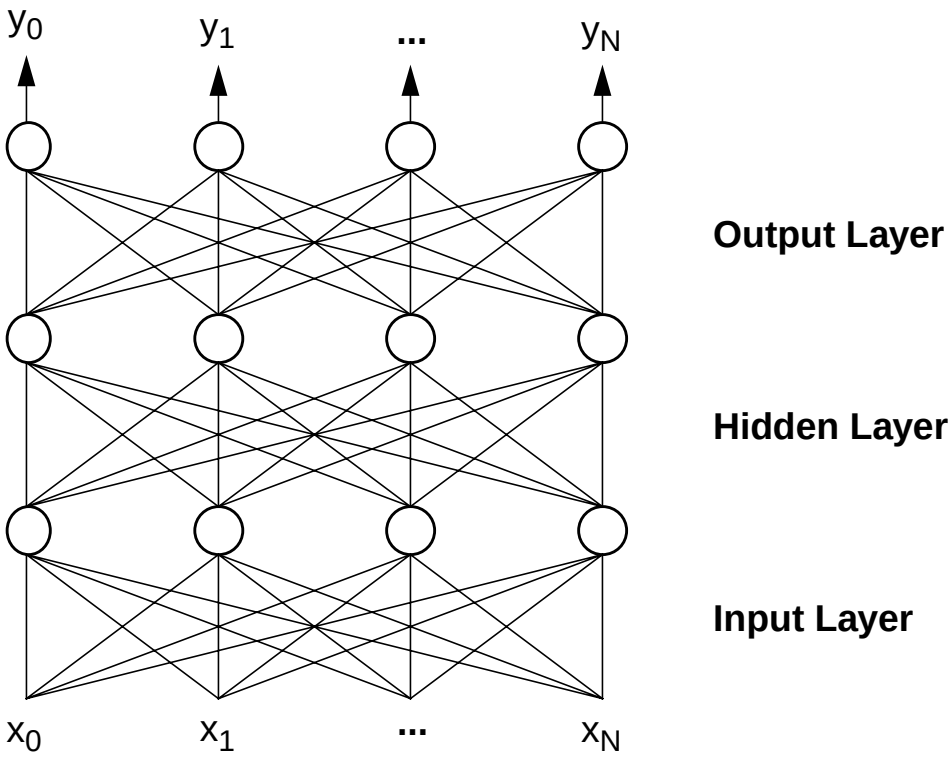
- pattern classification (N—way choice; vector quantization)
- associative memory (generate an output from a noisy input; character recognition)
- feature extraction (similarity transformations; dimensionality reduction)

We will focus on multilayer perceptrons in our studies. These have been shown to be quite useful for a wide range of problems.

Multilayer Perceptrons (MLP)

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)



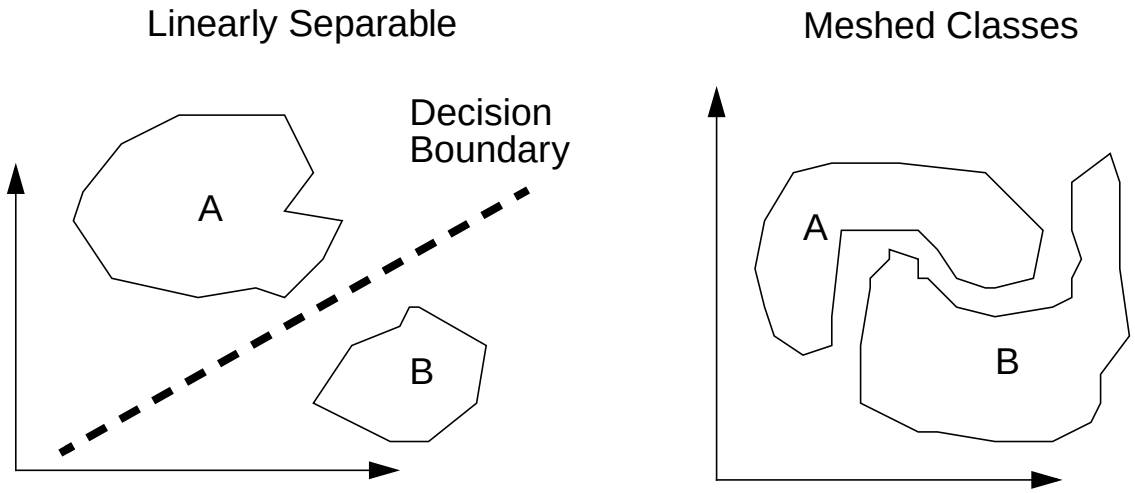
An alternate formulation of such a net is known as the learning vector quantizer (LVQ) — to be discussed later.

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?

- 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

- Temporal relationships not explicitly modeled

And, of course, both of these are extremely important to the speech recognition problem.

Session IX:

Language Modeling

Perplexity

How do we evaluate the difficulty of a recognition task?

- Instantaneous/local properties that influence peak resource requirements: maximum number of branches at a node; acoustic confusability
- Global properties that influence average difficulty: perplexity

If there are W possible words output from a random source, and the source is not statistically independent (as words in a language tend to be), the entropy associated with this source is defined as:

$$H(\underline{w}) = -\lim_{N\to\infty}\frac{1}{N}\sum_{w_1^N} -E\left\{\log P(\underline{w}_1^N = w_1^N)\right\}$$

$$= -\lim_{N\to\infty}\frac{1}{N}\sum_{w_1^N} P(\underline{w}_1^N = w_1^N)\log P(\underline{w}_1^N = w_1^N)$$

For an ergodic source, we can compute temporal averages:

$$H(\underline{w}) = -\lim_{N\to\infty}\frac{1}{N}\log P(\underline{w}_1^N = w_1^N)$$

Of course, these probabilities must be estimated from training data (or test data).

Perplexity, a common measure used in speech recognition, is simply:

$$Q(\underline{w}) = 2^{H(\underline{w})}$$

and represents the average branching factor of the language.

Perplexities range from 11 for digit recognition to several hundred for LVCSR. For a language in which all words are equally likely for all time,

$$Q(\underline{w}) = W$$

Types of Language Models

Common Language Models:

- 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 Out-Of-Vocabulary words?

- Garbage Models
- Monophone and Biphone Grammars with optional Dictionary Lookup
- Filler Models

Can we adapt the language model?

Other Formal Languages

- Unrestricted Grammars
- Context Sensitive (N-Grams, Unification)

$$\omega_1 A \omega_2 \rightarrow \omega_1 \beta \omega_2$$

- Context Free (LR)

$$A \rightarrow \beta$$

- Regular (or Finite State)

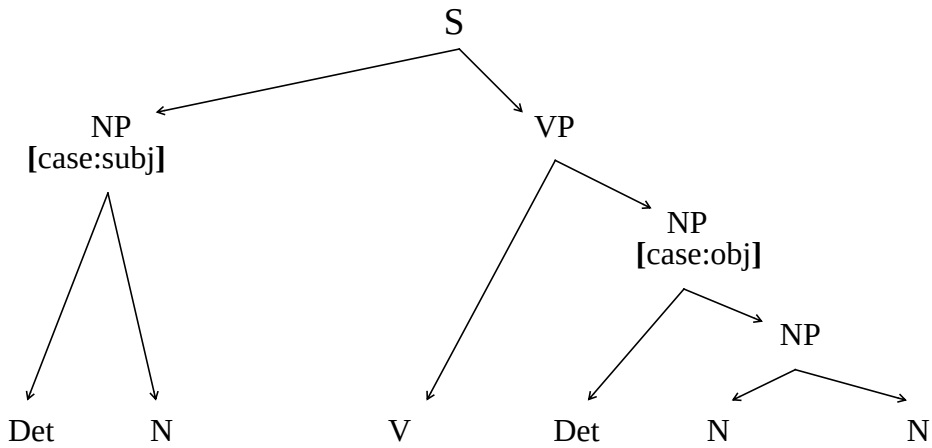
$$A \rightarrow aB$$
$$A \rightarrow b$$

- Issues
  - — stochastic parsing
  - — memory, pruning, and complexity

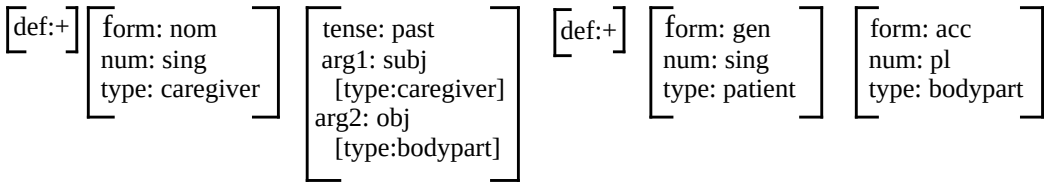
What can you do with all of this?

Logical: ∃(X) & ∃(Y) & ∃(Z) & doctor(X) & patient(Y) & knees(Z) & part-of(Y,Z) & examined(X,Z)

Syntactic:



Lexical:

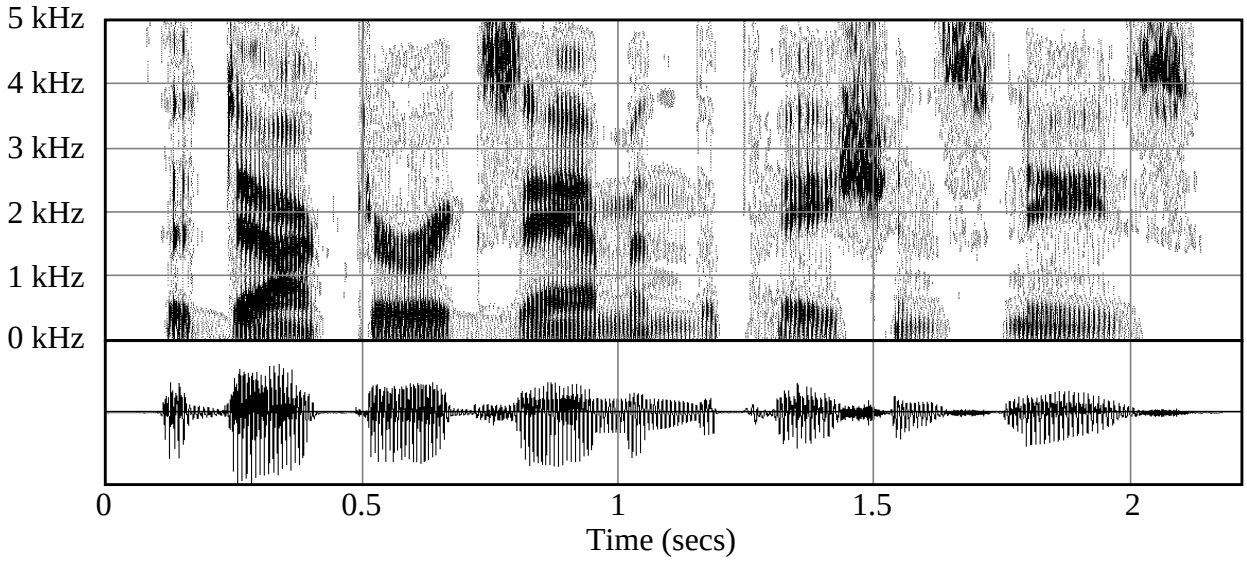


Orthographic: The doctor examined the patient's knees.

Phonemic: /# dh i # d A k t exr # I g z ae m ex n + d # dh i # p e sh ex n t + z # n i + z #/

Phonetic: dh ex d A k t exr I g z ae m I n d dhex pH e-I sh I nt s n i: z

Phonetic: dh ex d A k t exr I g z ae m I n d dhex pH e-I sh I nt s n i: z



Session IX:

State of the Art

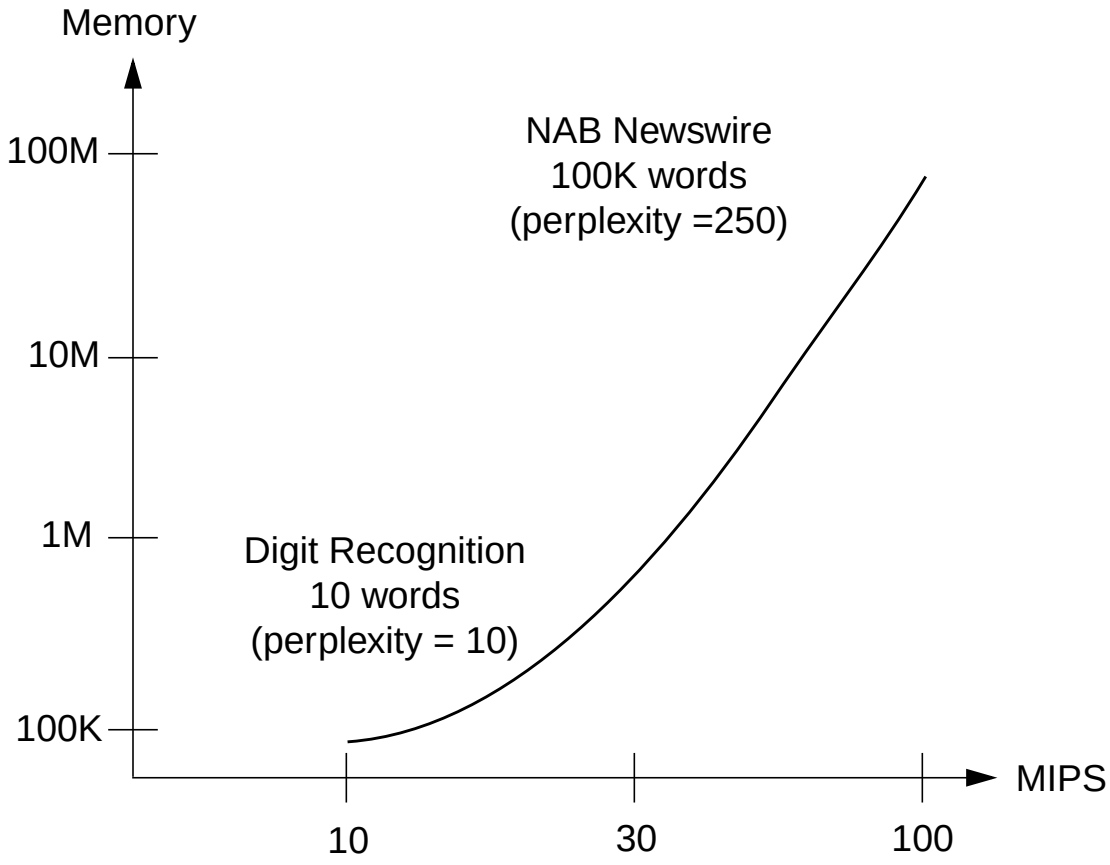
WHAT IS SPEECH UNDERSTANDING?

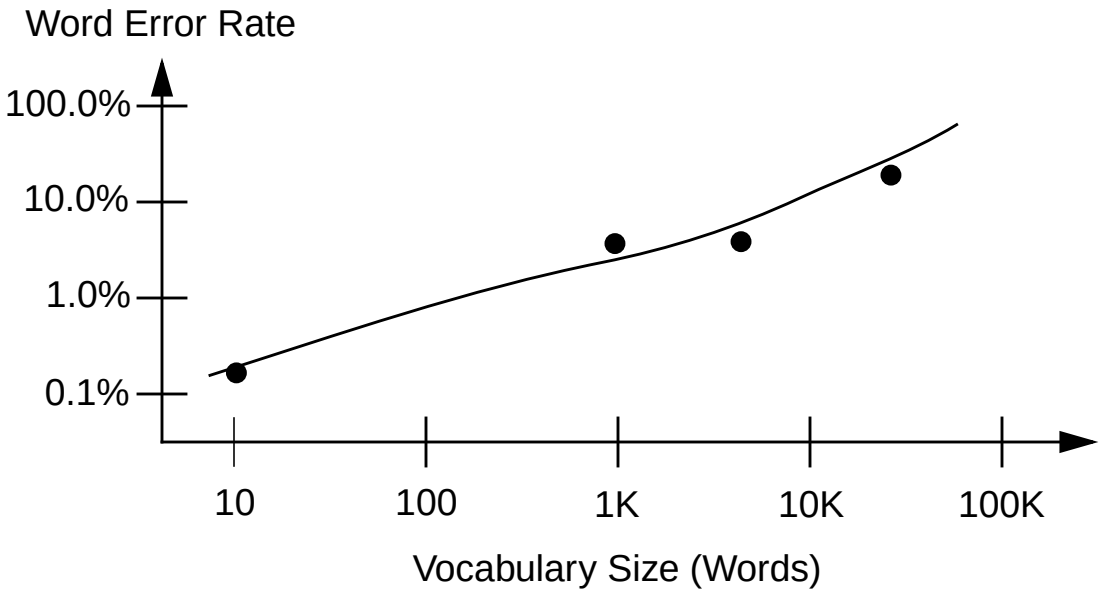
- Speech Recognition: transcription of words
  - performance measured by word error rate
- Speech understanding: acting on intentions
  - performance measured by the number of “queries” successfully answered (adds a natural language dimension to the problem)

DIMENSIONS OF THE PROBLEM

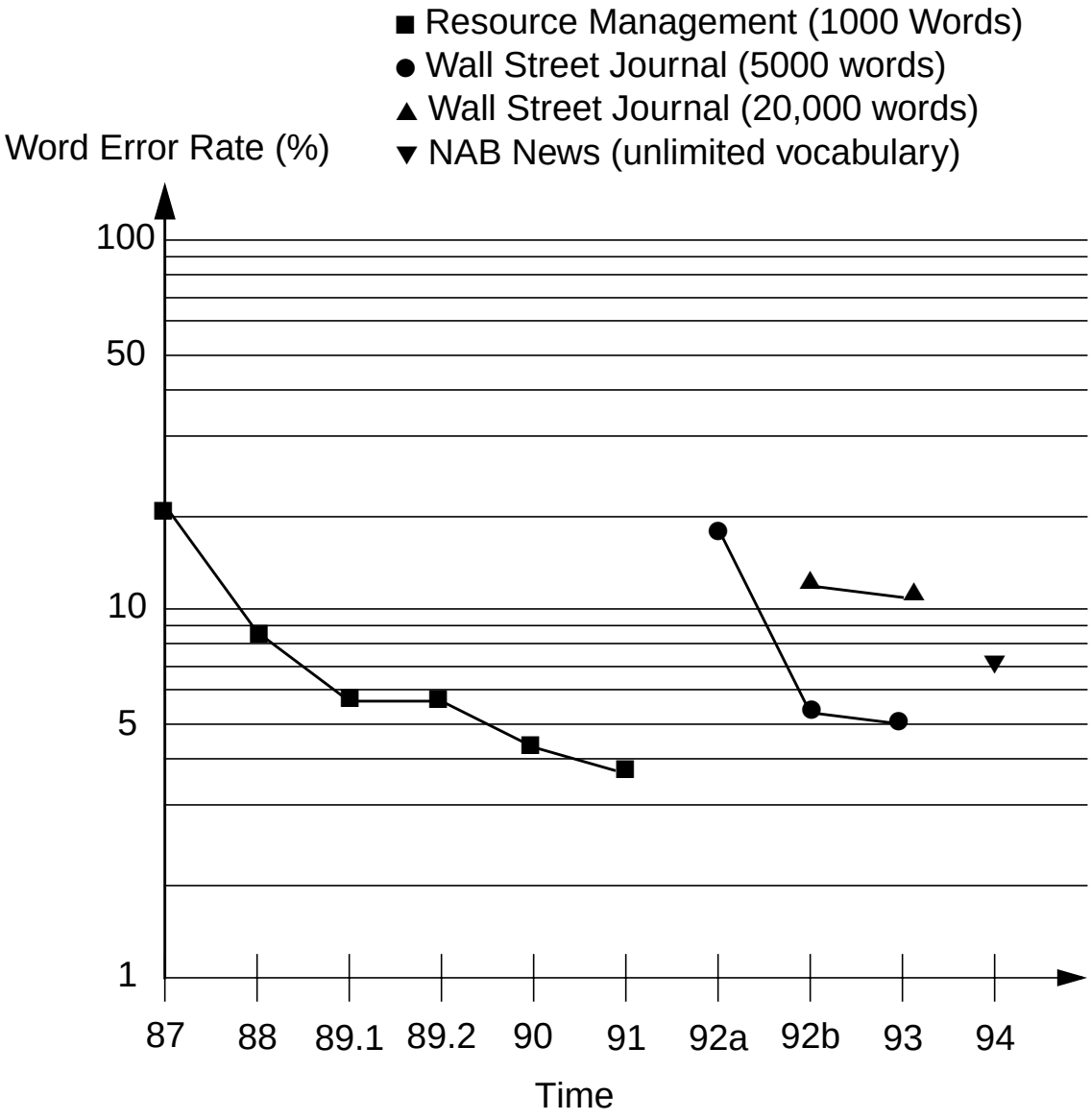
- Performance is a function of the vocabulary
  - how many words are known to the system
  - how acoustically similar are the competing choices at each point
  - how many words are possible at each point in the dialog
- Vocabulary size is a function of memory size
- Usability is a function of the number of possible combinations of vocabulary words
  - restrictive syntax is good for performance and bad for usability
- Hence, performance is a function of memory size

TODAY'S TECHNOLOGY





PROGRESS IN CSR PERFORMANCE



Note: Machine performance is still at least two orders of magnitude lower than humans

(Data from George Doddington, ARPA HLT Program Manager, HLT'95)

WHAT IS THE CONTEXT FOR SPEECH UNDERSTANDING RESEARCH IN THE 1990's?

- The Application Need

  Interaction with on-line data (Internet)

  Automated information agents ("24-hour Help Desk")

  Global multilingual interactions

- The Technology Challenge

  Create natural transparent interaction with computers ("2001," "Star Trek")

  Bring computing to the masses (vanishing window)

  Intelligent presentation of information ("hands/eyes busy applications")

- Application Areas

  Command/Control (Telecom/Workstations)

  Database Query (Internet)

  Dictation (Workstations)

  Machine Translation (Workstations)

  Real-Time Interpretation (Telecom)

THE TECHNICAL CHALLENGE

❒ Barriers

Speech understanding is a vast empirical problem:

- Hierarchies of hidden representations (part-of-speech, noun phrases, units of meaning) produce immensely complex models
- Training of complex models requires huge and dynamic knowledge bases

Interconnected and interdependent levels of representation:

- Correct recognition and transcription of speech depends on understanding the meaning encoded in speech
- Correct understanding and interpretation of text depends on the domain of discourse

❒ Approach

Capitalize on exponential growth in computer power and memory:

- Statistical modeling and automatic training
- Shared resources and infrastructure

Application-focused technical tasks

- New metrics of performance based on user feedback and productivity enhancement (human factors)

(G. Doddington, ARPA HLT Program Manager, “Spoken Language Technology Discussion,” ARPA Human Language Technology Workshop, January 1995)

Example No. 1: HTK (Cambridge University)

Signal Processing:

- Sample Frequency = 16 kHz
- Frame Duration = 10 ms
- Window Duration = 25 ms (Hamming window)
- FFT-Based Spectrum Analysis
- Mel-Frequency Filter Bank (24)
- Log-DCT Cepstral Computation (Energy plus 12 coefficients)
- Linear Regression Using A 5 Frame Window For Δ and Δ^2 Parameters

Acoustic Modeling:

- Simple Left-To-Right Topology With Three Emitting States
- 10 component Gaussian Mixtures at Each State
- Baum-Welch Reestimation
- Cross-Word Context-Dependent Phone Models
- State-Tying Using Phonetic Decision Trees (Common Broad Phonetic Classes)
- +/- 2 Phones Used In Phonetic Decision Trees Including Word Boundary

Language Modeling:

- N-Gram Language Model (3, 4, and 5-Grams Have Been Reported)
- Discounting (De-Weight More Probable N-Grams)
- Back-Off Model Using Bigrams
- Single-Pass Time-Synchronous Tree Search

Notes:

- Approx. 3 to 8 Million Free Parameters
- State-of-the-Art Performance
- Relatively Low Complexity
- Excellent Real-Time Performance

Example No. 2: Abbot Hybrid Connectionist HMM (Cambridge Univ.)

Signal Processing:

- Sample Frequency = 16 kHz
- Frame Duration = 16 ms
- Window Duration = 32 ms
- FFT-Based Spectrum Analysis
- Mel-Freq. Filter Bank (20) + 3 Voicing Features (replaced by PLP)
- Each Feature Normalized To Zero Mean and Unit Variance (Grand Covar.)

Acoustic Modeling:

- Recurrent Neural Network
- Single Layer Feed-Forward Network With Four Frame Delay
- Sigmoidal Nonlinearities
- A Single Network Computes All Posterior Phone Probabilities
- Viterbi-Like Back-Propagation-Through-Time Training
- Parallel Combination of Backward-In-Time and Forward-In-Time Networks
- Context-Dependent Probabilities Computed Using A Second Layer
- Linear Input Mapping For Speaker Adaptation (Similar to MLLR)

Language Modeling:

- Single-Pass Stack Decoding Algorithm
- N-Gram Language Model
- Phone-Level and Word-Level Pruning
- Tree-Based Lexicon
- NO Context-Dependent Cross-Word Models

Notes:

- Approx. 80K Free Parameters
- Performance Comparable To Best HMMs
- Relatively Low Complexity
- Excellent Real-Time Performance

Additional Resources

Web:

CMU:	http://www.ri.cmu.edu/speech
comp.speech:	ftp://svr-ftp.eng.cam.ac.uk/pub/pub/comp.speech/
UCSB:	http://mambo.ucsc.edu/psl/speech.html
IEEE SP (Rice):	http://spib.rice.edu/spib.html

Public Domain:

Oregon Graduate Institute

Cambridge Connectionist Group — AbbotDemo

... and many more ...

ISIP (about one year away)

Commercial:

Entropic Waves / HTK

Matlab (coming soon - also Entropic)