ELEN E6884/COMS 86884 Speech Recognition Lecture 2

Michael Picheny, Ellen Eide, Stanley F. Chen
IBM T.J. Watson Research Center
Yorktown Heights, NY, USA
{picheny,eeide,stanchen}@us.ibm.com

15 September 2005

Administrivia

- today is picture day!
- will hand out hardcopies of slides and readings for now
  - don’t take something if you don’t want it
- main feedback from last lecture
  - a little fast?
  - went through signal processing quickly
    - will try to make sure you’re OK for lab 1
- Lab 0 due tomorrow
- Lab 1 out today, due on Friday in two weeks

Outline of Today's Lecture

- Feature Extraction
- Brief Break
- Dynamic Time Warping

Goals of Feature Extraction

- Capture essential information for sound and word identification
- Compress information into a manageable form
- Make it easy to factor out irrelevant information to recognition
  - Long-term channel transmission characteristics
  - Speaker-specific information such as pitch, vocal-tract length
- Would be nice to find features that are i.i.d. and are well-modeled by simple distributions so that our models will perform well.

Figures from Holmes, HAH or R+J unless indicated otherwise.

What are some possibilities?

- Model speech signal with a parsimonious set of parameters that intuitively describe the signal
  - Acoustic-phonetic features
- Use some type of function approximation such as Taylor or Fourier series
- Ignore pitch
  - Cepstral Coefficients
  - Linear Prediction (LPC)
- Match human perception of frequency bands
  - Mel-Scale Cepstral Coefficients (MFCCs)
  - Perceptual Linear Prediction (PLP)
- Ignore other speaker dependent characteristics e.g. vocal tract length

- Vocal-tract length normalized Mel-Scale Cepstral Coefficients

- Incorporate dynamics
  - Deltas and Double-Deltas
  - Principal component analysis

Pre-processor to Many Feature Calculations: Pre-Emphasis

Purpose: Compensate for 6dB/octave falloff due to glottal-source and lip-radiation combination.

Assume our input signal is $x[n]$. Pre-emphasis is implemented via very simple filter:

$$y[n] = x[n] + ax[n-1]$$

To analyze this, let's use the "Z-Transform" introduced in Lecture 1. Since $Z(x[n-1]) = z^{-1}Z(x[n])$ we can write

$$Y(z) = X(z)H(z) = X(z)(1 + az^{-1})$$

If we substitute $z = e^{j\omega}$, we can write

$$\begin{aligned} |H(e^{j\omega})|^2 &= |1 + a(\cos\omega - j\sin\omega)|^2 \\ &= 1 + a^2 + 2a\cos\omega \end{aligned}$$

or in dB

$$10\log_{10}|H(e^{j\omega})|^2 = 10log_{10}(1 + a^2 + 2a\cos\omega)$$

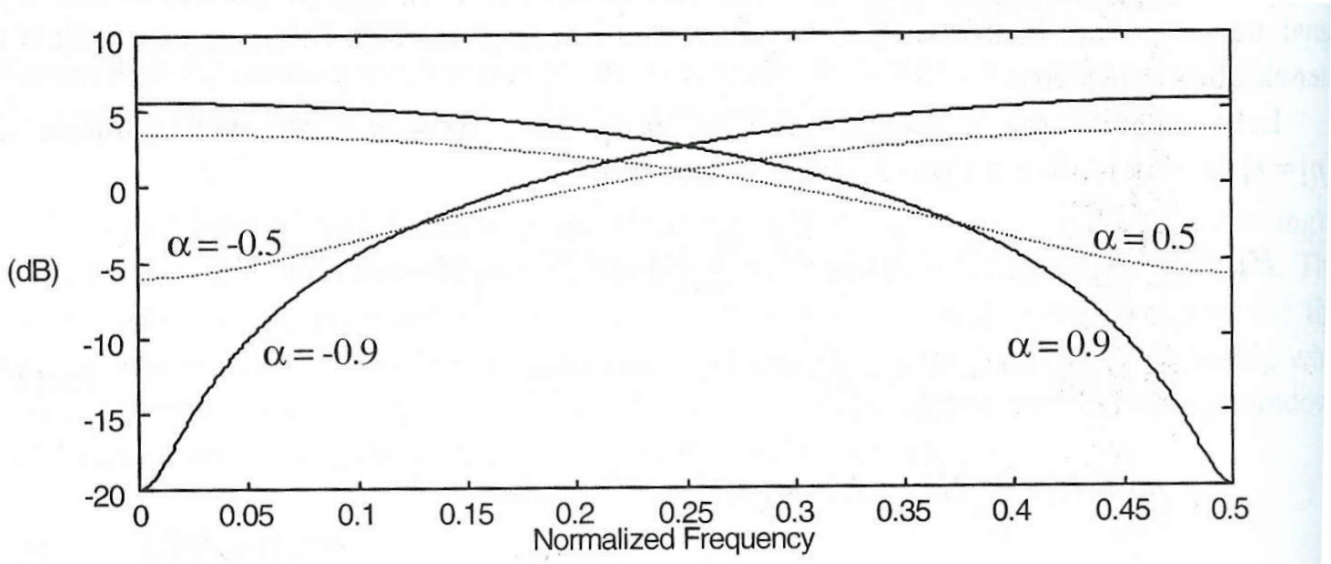


Figure 5.21 Frequency response of the first order FIR filter for various values of α.

For $a > 0$ we have a low-pass filter and for $a < 0$ we have a high-pass filter, also called a "pre-emphasis" filter because the frequency response rises smoothly from low to high frequencies.

Uses are:

- Improve LPC estimates (works better with “flatter” spectra)
- Reduce or eliminate DC offsets
- Mimic equal-loudness contours (higher frequency sounds appear “louder” than low frequency sounds for the same amplitude)

Basic Speech Processing Unit - the Frame

The speech waveform is changing over time. We need to focus on short-time segments over which the signal is more or less representing a single phoneme, since our models are phoneme-based.

Define

$$x^m[n] = x[n - mF]w[n]$$

as frame m to be processed where F is the spacing between frames and $w[n]$ is our window of length N.

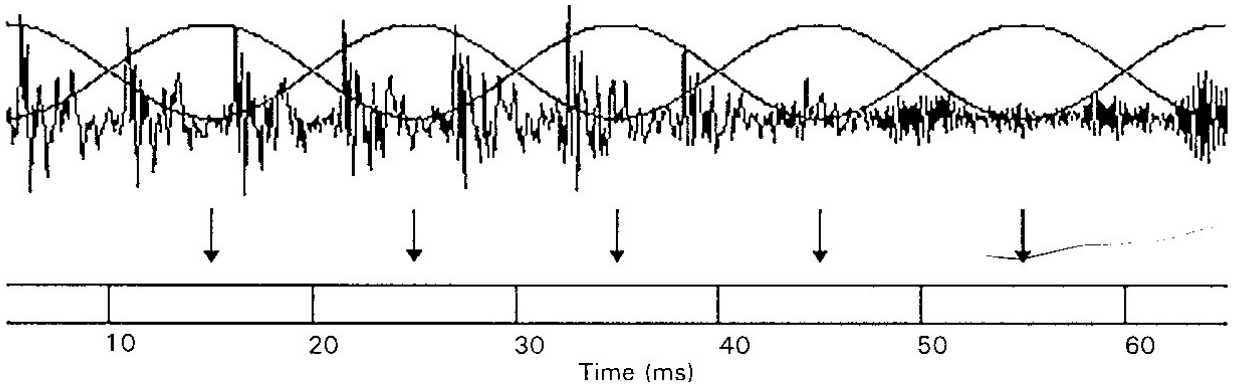


Figure 10.1 Analysis of a speech signal into a sequence of frames. This example shows a 20 ms Hanning window applied at 10 ms intervals to give a frame rate of 100 frames/s.

How do we choose the window type $w[n]$, the frame spacing, F, and the window length, N?

- Experiments in speech coding suggest that F should be around 10 msec. For F greater than 20 msec and one starts hearing noticeable distortion. Less and things do not appreciably improve.
- From last week, we know that both Hamming and Hanning windows are good.

$$h[n] = .5 - .5\cos 2\pi n/N \text{(Hanning)}$$

$$h[n] = .54 - .46\cos 2\pi n/N \text{(Hamming)}$$

So what window length should we use?

- If too long, vocal tract will be non-stationary; smooth out transients like stops.
- If too short, spectral output will be too variable with respect to window placement.

Usually choose 20-25 msec window length as a compromise.

Effects of Windowing

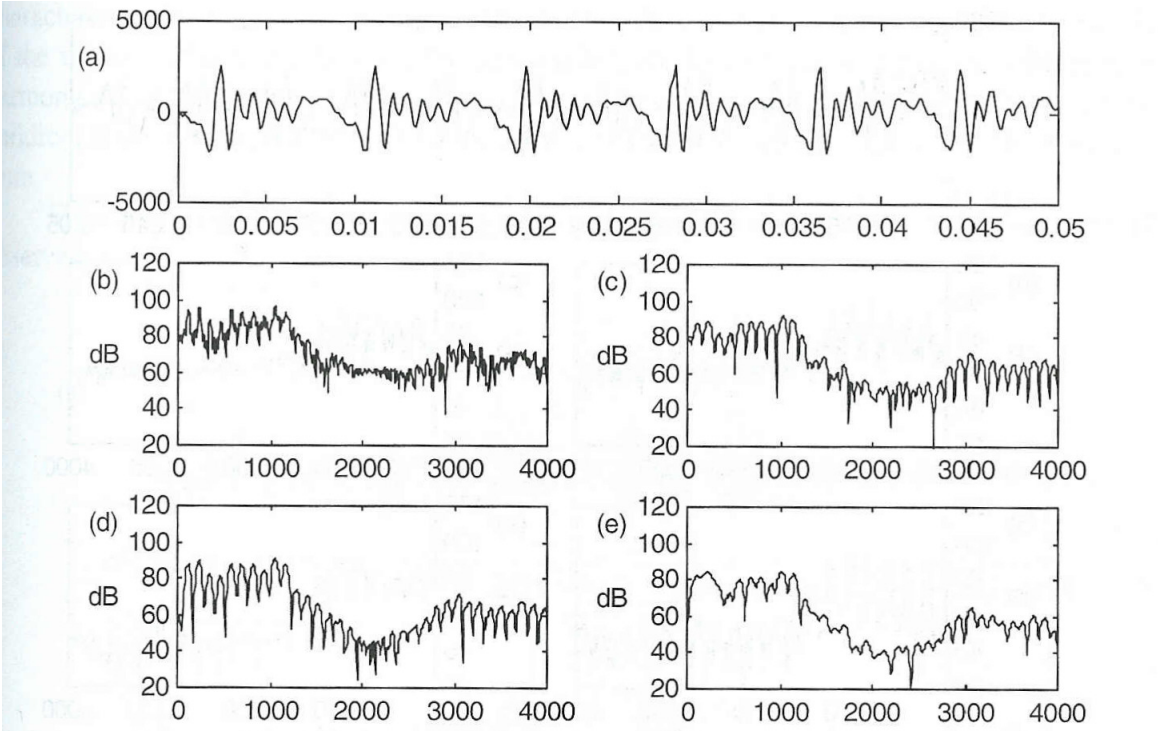


Figure 6.3 Short-time spectrum of male voiced speech (vowel /ah/ with local pitch of 110Hz): (a) time signal, spectra obtained with (b) 30 ms rectangular window and (c) 15 ms rectangular window, (d) 30 ms Hamming window, (e) 15 ms Hamming window. The window lobes are not visible in (e), since the window is shorter than 2 times the pitch period. Note the spectral leakage present in (b).

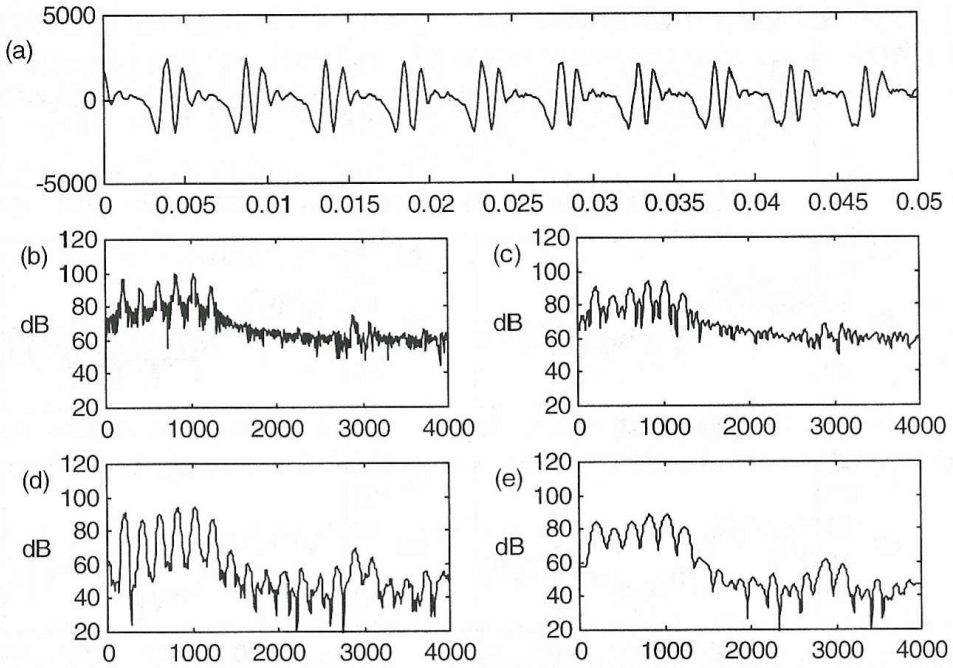


Figure 6.4 Short-time spectrum of female voiced speech (vowel /aa/ with local pitch of 200Hz): (a) time signal, spectra obtained with (b) 30 ms rectangular window and (c) 15 ms rectangular window, (d) 30 ms Hamming window, (e) 15 ms Hamming window. In all cases the window lobes are visible, since the window is longer than 2 times the pitch period. Note the spectral leakage present in (b) and (c).

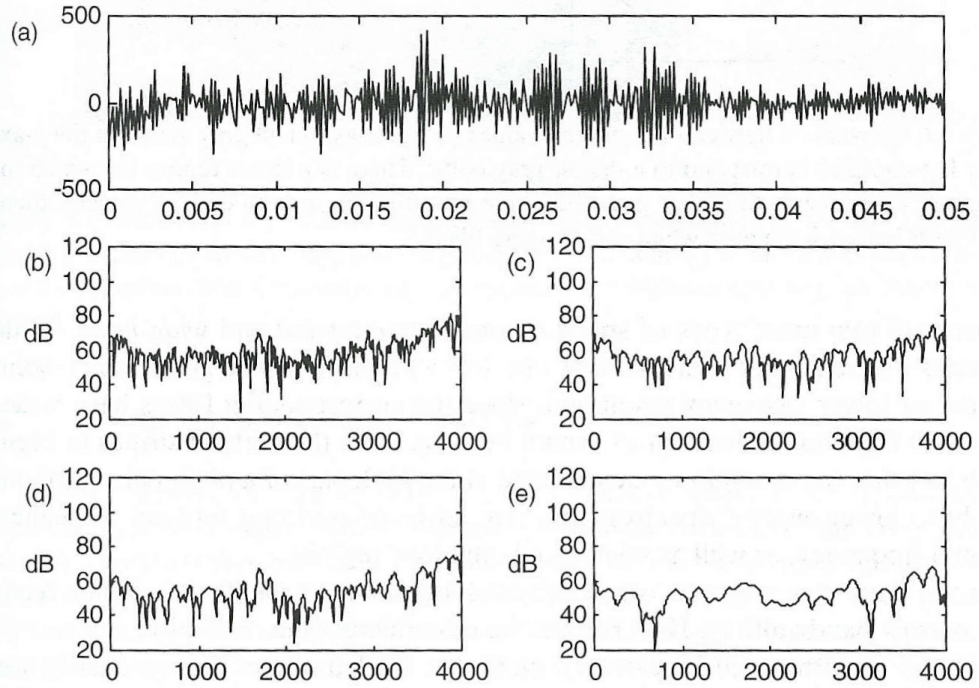


Figure 6.5 Short-time spectrum of unvoiced speech: (a) time signal, (b) 30 ms rectangular window, (c) 15 ms rectangular window, (d) 30 ms Hamming window, (e) 15 ms Hamming window.

Acoustic-phonetic features

Goal is to parameterize each frame in terms of speaker actions (nasality frication, voicing, etc.) or physical properties related to source-filter model (formant locations, formant bandwidths, ratio of high-frequency to low-frequency energy, etc.)

Haven't proven as effective as some other feature sets such as MFCC's

Conjecture: This could be because of our model's assumption that observations are independent...probably a worse fit for acoustic-phonetic features than for MFCC's.

Spectral Features

Could use features such as DFT coefficients directly, such as what is used in spectrograms.

Recall that the source-filter model says the pitch signal is convolved with the vocal tract filter

In the frequency domain, that convolution equates to multiplication

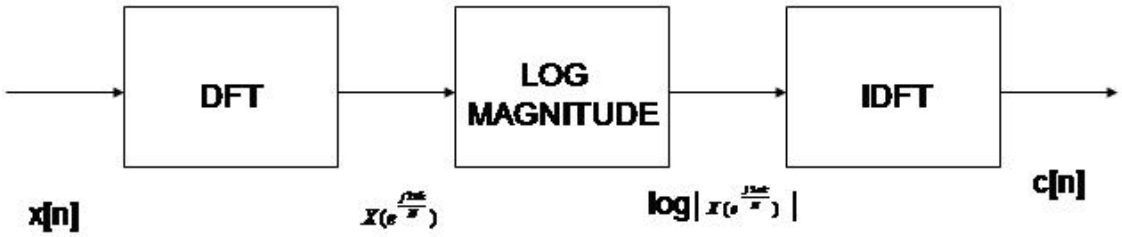
Bad aspect: pitch and spectral envelope characteristics intertwined... not easy to throw away just the pitch information

Cepstral Coefficients

Recall that the source-filter model says the pitch signal is convolved with the vocal tract filter

In the frequency domain, that convolution equates to multiplication

Taking the logarithm of the spectrum converts multiplication to addition



NOTE: Because the log magnitude spectrum of a real signal is real and symmetric, the cepstrum can be obtained by doing a discrete cosine transform (DCT) on the log magnitude spectrum rather than doing the IDFT

Fortunately the pitch signal and vocal-tract filter are easily separted after taking the logarithm ... the pitch signal corresponds to high-time part of the cepstra, the vocal tract to the low-time part.

Truncation of the cepstra results in spectral envelope without pitch info.

Aside: Truncating the cepstral vector can be used for estimating formants.

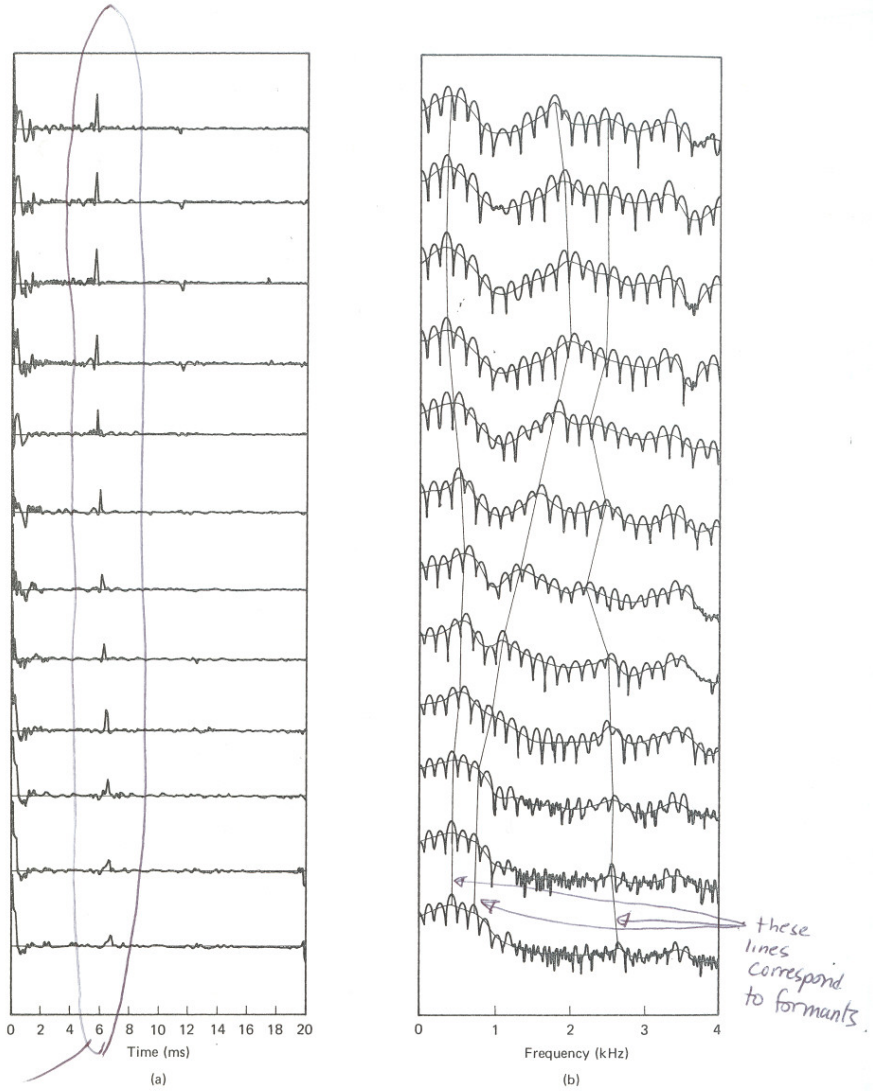


Figure 12.28 (a) Cepstra and (b) log spectra for sequential segments of voiced speech. Original with cepstrally smoothed superimposed.

Figure from Oppenheim + Schafer. "Discrete-Time Signal Processing"

Linear Prediction - Motivation

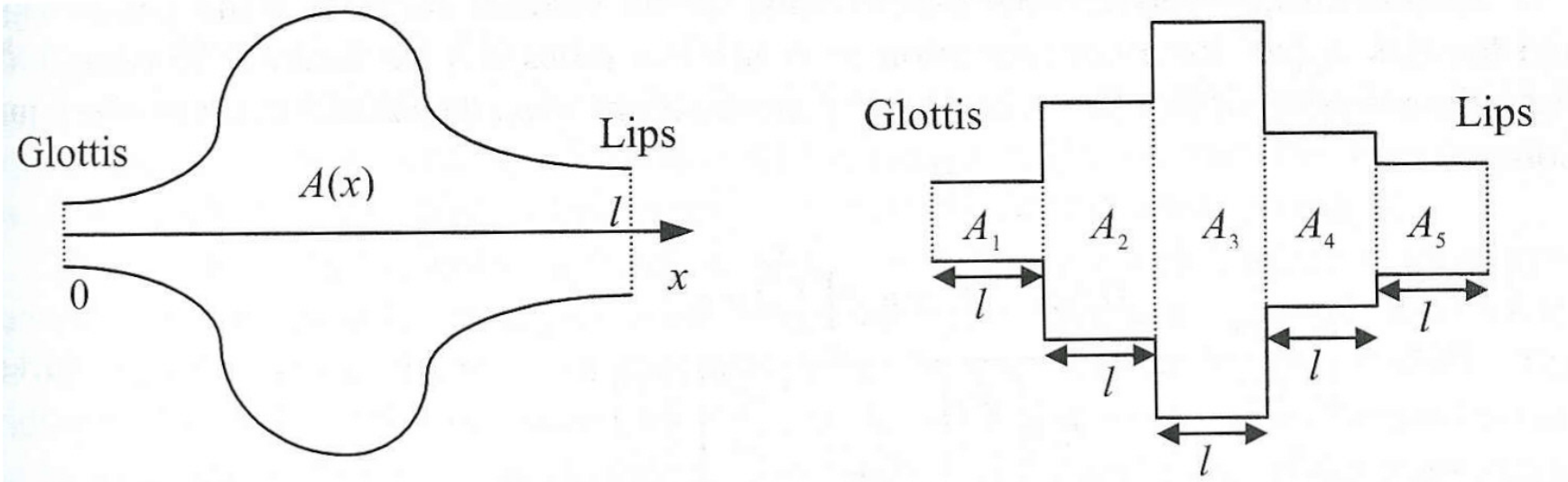
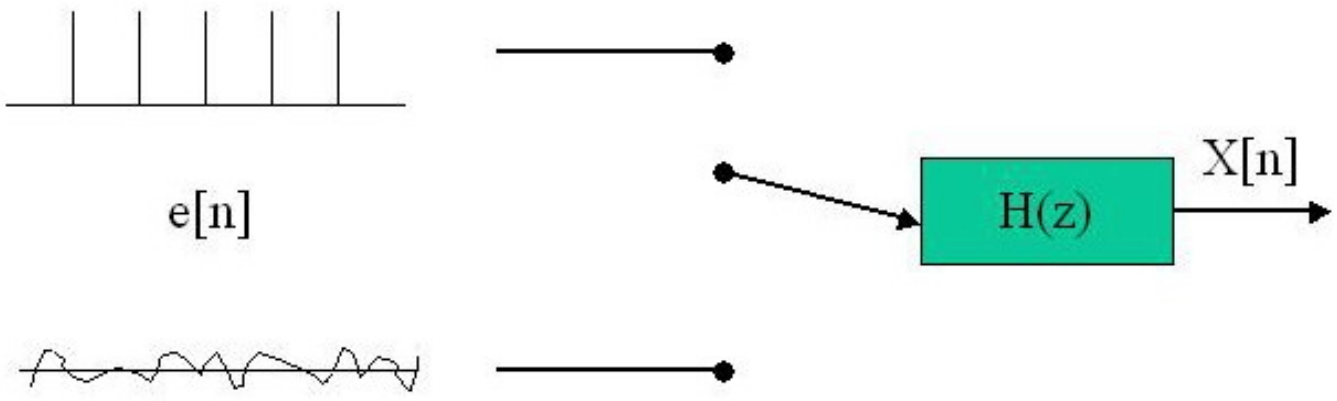


Figure 6.9 Approximation of a tube with continuously varying area $A(x)$ as a concatenation of 5 lossless acoustic tubes.

The above model of the vocal tract matches observed data quite well, at least for speech signals recorded in clean environments. It is associated with a filter $H(z)$ with a particularly simple time-domain interpretation.

Linear Prediction



The linear prediction model assumes that $x[n]$ is a linear combination of the p previous samples and an excitation $Gu[n]$

$$x[n] = \sum_{j=1}^{p} a[j]x[n-j] + Gu[n]$$

$u[n]$ is either a string of (unit) impulses spaced at the fundamental

frequency (pitch) for voiced sounds such as vowels or (unit) white noise for unvoiced sounds such as fricatives.

Taking the Z-transform,

$$X(z) = U(z)H(z) = U(z)\frac{G}{1 - \sum_{j=1}^{p} a[j]z^{-j}}$$

where $H(z)$ can be associated with the (time-varying) filter associated with the vocal tract and an overall gain G.

Solving the Linear Prediction Equations

It seems reasonable to find the set of $a[j]$s that minimize the energy in the prediction error:

$$\sum_{n=-\infty}^{\infty} e^2[n] = G^2 \sum_{n=-\infty}^{\infty} u^2[n] = E$$

Why is it reasonable to assign $Gu[n]$ to the prediction error?

Hand-wave 1: For voiced speech, u is an impulse train so it is small most of the time

Hand-wave 2: Doing this leads to a nice solution

$$E = \sum_{n=-\infty}^{\infty} (x[n] - \sum_{j=1}^{p} a[j]x[n-j])^2$$

If we take derivatives with respect to each $a[i]$ in the above equation and set the results equal to zero we get a set of p equations indexed by i:

$$\sum_{j=1}^{p} a[j]R(i,j) = R(i,0), 1 \leq i \leq p$$

where $R(i,j) = \sum_n x[n-i]x[n-j]$.

In practice, we would not use the potentially infinite signal $x[n]$ but the individual windowed frames $x^m[n]$. Since $x^m[n]$ is zero outside the window, $R(i,j) = R(j,i) = R(|i-j|)$ where $R(i)$ is just the *autocorrelation* sequence corresponding to $x^m(n)$. This allows us to write the previous equation as

$$\sum_{j=1}^{p} a[j]R(|i-j|) = R(i), 1 \leq i \leq p$$

a much simpler and regular form known as “Toeplitz.”

$$\begin{bmatrix} a_0 & a_{-1} & a_{-2} & \cdots & a_{-n+1} \\ a_1 & a_0 & a_{-1} & \ddots & \vdots \\ a_2 & a_1 & a_0 & \ddots & a_{-2} \\ \vdots & \ddots & \ddots & \ddots & a_{-1} \\ a_{n-1} & \cdots & a_2 & a_1 & a_0 \end{bmatrix}.$$

The Levinson-Durbin Recursion

The Toeplitz matrix associated with the previous set of equations can easily be solved using the "Levinson-Durbin recursion"

Initialization. $E^0 = R(0)$ Iteration. For $i = 1, \ldots, p$ do:

$$
\begin{aligned}
k[i] &= (R(i) - \sum_{j=1}^{i-1} a^{i-1}[j] R(|i-j|))/E^{i-1} \\
a^i[i] &= k[i] \\
a^i[j] &= a^{i-1}[j] - k[i] a^{i-1}[i-j], 1 \leq j < i \\
E^i &= (1 - k[i]^2) E^{i-1}
\end{aligned}
$$

End. $a[j] = a^p[j]$ and $G^2 = E^p$.

LPC Examples

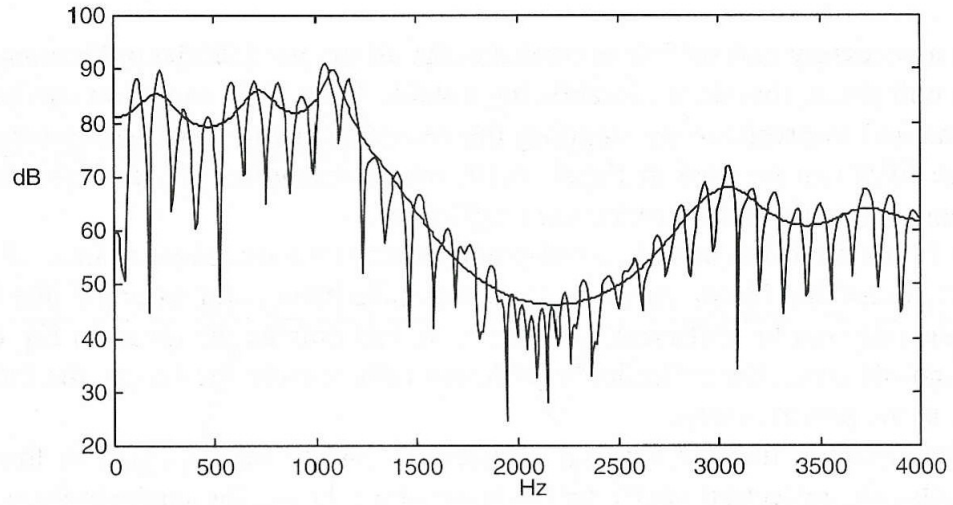


Figure 6.20 LPC spectrum of the */ah/* phoneme in the word *lives* of Figure 6.3. Used here are a 30-ms Hamming window and the autocorrelation method with $p = 14$. The short-time spectrum is also shown.

Here the spectra of the original sound and the LP model are compared. Note how the LP model follows the peaks and ignores the “dips” present in the actual spectrum of the signal as computed from the DFT.

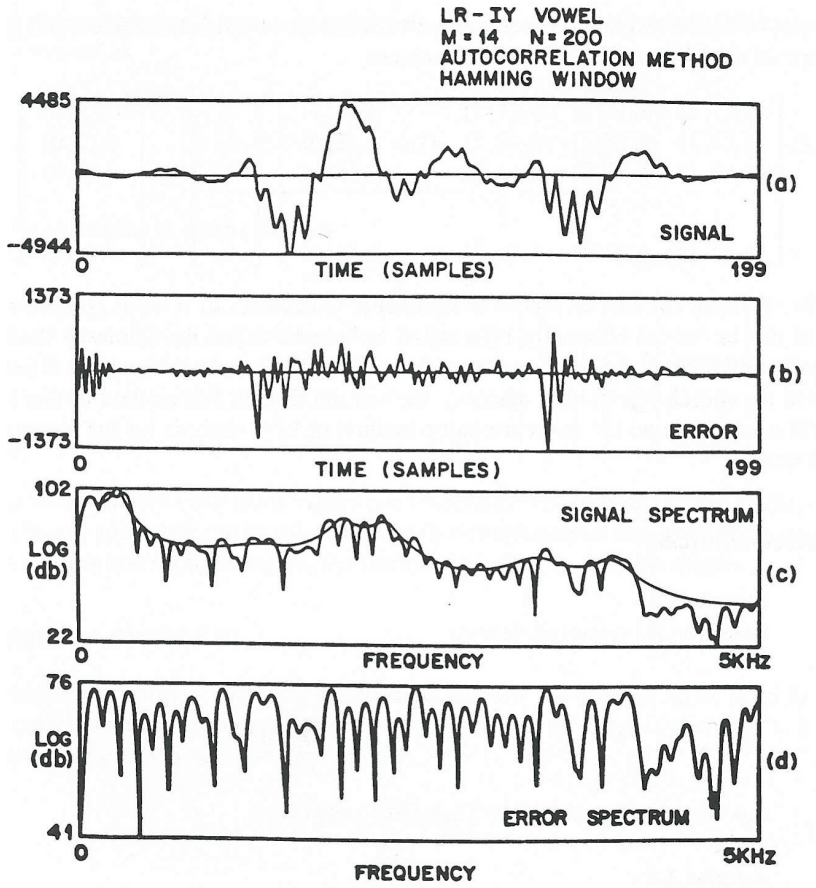


Figure 3.32 Typical signals and spectra for LPC autocorrelation method for a segment of speech spoken by a male speaker (after Rabiner et al. [8]).

Observe the prediction error. It clearly is NOT a single impulse.

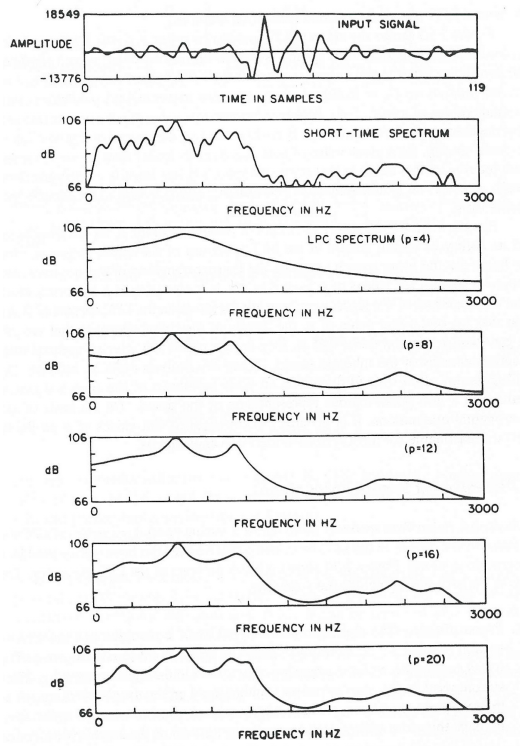


Figure 3.36 Spectra for a vowel sound for several values of predictor order, p.

As the model order p increases the LP model progressively approaches the original spectrum. As a rule of thumb, one typically sets p to be the sampling rate (divided by 1 KHz) + 2-4, so for a 10 KHz sampling rate one would use $p = 12$ or $p = 14$.

LPC and Speech Recognition

How should one use the LP coefficients in speech recognition?

- The $a[j]$s themselves have an enormous dynamic range, are highly intercorrelated in a nonlinear fashion, and vary substantially with small changes in the input signal frequencies.
- One can generate the spectrum from the LP coefficients but that is hardly a compact representation of the signal.
- Can use various transformations, such as the reflection coefficients $k[i]$ or the log area ratios $\log(1-k[i])/(1+k[i])$ or LSP parameters (yet another transformation related to the roots of the LP filter).
- The transformation that works best is the LPC *Cepstrum*.

LPC Cepstrum

The complex cepstrum is defined as the IDFT of the logarithm of the spectrum:

$$\tilde{h}[n] = \frac{1}{2\pi} \int \ln H(e^{j\omega}) e^{j\omega n} d\omega$$

Therefore,

$$\ln H(e^{j\omega}) = \sum \tilde{h}[n] e^{-j\omega n}$$

or equivalently

$$\ln H(z) = \sum \tilde{h}[n] z^{-n}$$

Let us assume correponding to our LPC filter is a cepstrum $\tilde{h}[n]$. If so we can write

$$\sum_{n=-\infty}^{\infty} \tilde{h}[n] z^{-n} = \ln G - \ln(1 - \sum_{j=1}^{p} a[j] z^{-j})$$

Taking the derivative of both sides with respect to z we get

$$-\sum_{n=-\infty}^{\infty} n\tilde{h}[n]z^{-n-1} = \frac{-\sum_{l=1}^{p} la[l]z^{-l-1}}{1-\sum_{j=1}^{p} a[j]z^{-j}}$$

Multiplying both sides by $-z(1-\sum_{j=1}^{p} a[j]z^{-j})$ and equating coefficients of z we can show with some manipulations that $\tilde{h}[n]$ is

$$\begin{array}{cc} 0 & n<0 \\ \ln G & n=0 \\ a[n]+\sum_{j=1}^{n-1}\frac{j}{n}\tilde{h}[j]a[n-j] & 0<n\leq p \\ \sum_{j=n-p}^{n-1}\frac{j}{n}\tilde{h}[j]a[n-j] & n>p \end{array}$$

Notice the number of cepstrum coefficients is infinite but practically speaking 12-20 (depending upon the sampling rate and whether you are doing LPC or PLP) is adequate for speech recognition purposes.

Simulating Filterbanks with the FFT

A common operation in speech recognition feature extraction is the implementation of filter banks.

The simplest technique is brute force convolution. Assuming i filters $h_i[n]$

$$x_i[n] = x[n] * h_i[n] = \sum_{m=0}^{L_i-1} h_i[m]x[n-m]$$

The computation is on the order of L_i for each filter for each output point n, which is large.

Say now $h_i[n] = h[n]e^{j\omega_i n}$, a fixed length low pass filter heterodyned up (remember, multiplication in the time domain is the same as convolution in the frequency domain) to be centered

at different frequencies. In such a case

$$\begin{aligned} x_i[n] &= \sum h[m] e^{j\omega_i m} x[n-m] \\ &= e^{j\omega_i n} \sum x[m] h[n-m] e^{-j\omega_i m} \end{aligned}$$

The last term on the right is just $X_n(e^{j\omega})$, the Fourier transform of a windowed signal, where now the window is the same as the filter. So we can interpret the FFT as just the instantaneous filter outputs of a uniform filter bank whose bandwidths corresponding to each filter are the same as the main lobe width of the window.

Notice that by combining various filter bank channels we can create non-uniform filterbanks in frequency.

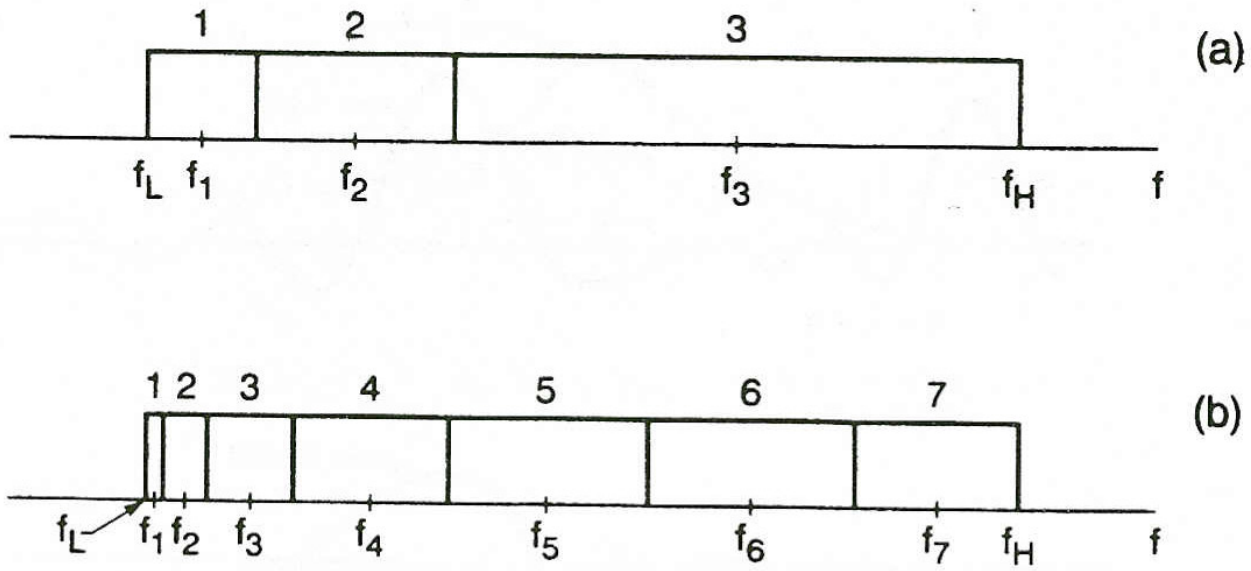


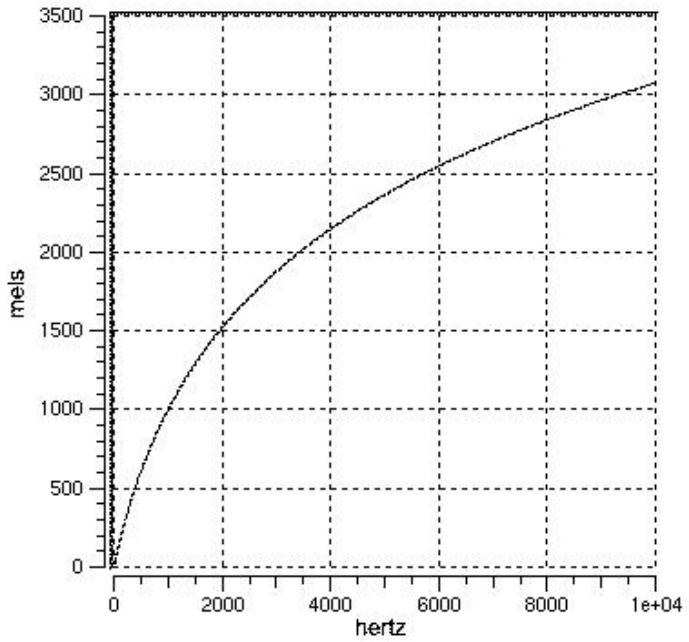
Figure 3.18 Two arbitrary nonuniform filter-bank ideal filter specifications consisting of either 3 bands (part a) or 7 bands (part b).

What is typically done in speech processing for recognition is to sum the magnitudes or energies of the FFT outputs rather than the raw FFT outputs themselves. This corresponds to a crude estimate of the magnitude/energy of the filter output over the time duration of the window and is not the filter output itself, but the terms are used interchangeably in the literature.

Mel-Frequency Scaling

Goal: Develop perceptually based set of features.

Psychophysical studies have shown that human perception of tones does not follow a linear scale.



Divide frequency axis into m filters spaced in equal perceptual

increments. Each filter is defined in terms of the FFT bins k as

$$H_m(k) \begin{cases} 0 & k < f(m-1) \\ \frac{k - f(m-1)}{f(m) - f(m-1)} & f(m-1) \le k \le f(m) \\ \frac{f(m+1) - k}{f(m+1) - f(m)} & f(m) \le k \le f(m+1) \\ 0 & k > f(m+1) \end{cases}$$

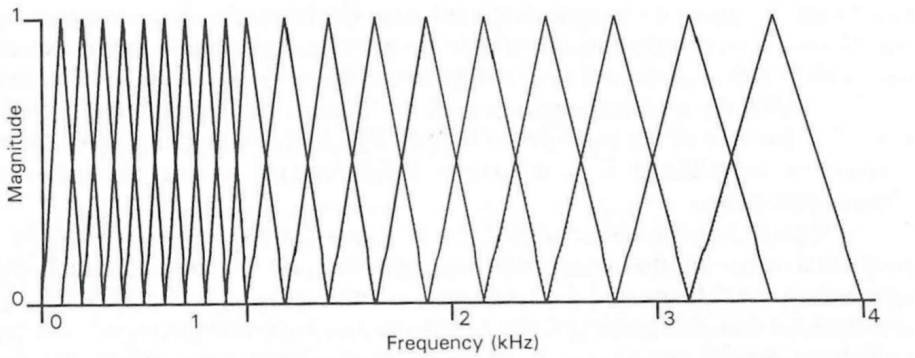


Figure 10.2 Triangular filters of the type suggested by Davis and Mermelstein (1980) for transforming the output of a Fourier transform onto a mel scale in both bandwidth and spacing.

Define f_l and f_h to be lowest and highest frequencies of the filterbank, F_s the sampling frequency, M, the number of filters, and N the size of the FFT. The boundary points $f(m)$ are spaced

in equal increments in the mel-scale:

$$f(m) = \frac{N}{F_S} B^{-1}(B(f_l) + m\frac{B(f_h) - B(f_l)}{m+1})$$

where the mel-scale, B, is given by

$$B(f) = 2595 \log_{10}(1 + f/700)$$

Some authors prefer to use $1127 \ln$ rather than $2595 \log_{10}$ but they are obviously the same thing. The filter outputs for a given frame are computed as

$$S(m) = 20\, log_{10}(\sum_{k=0}^{N-1} |X(k)| H_m(k)), 0 < m < M$$

where $X(k)$ is the N-Point FFT of the current frame of the input

signal, $x[n]$. N is chosen as the largest power of two greater than the window length; the rest of the input FFT is padded with zeros.

Mel-frequency Cepstral Coefficients

The mel-cepstrum can then be defined as the DCT of the M filter outputs

The DCT can be interpreted as the DFT of a symmetrized signal. There are many ways of creating this symmetry:

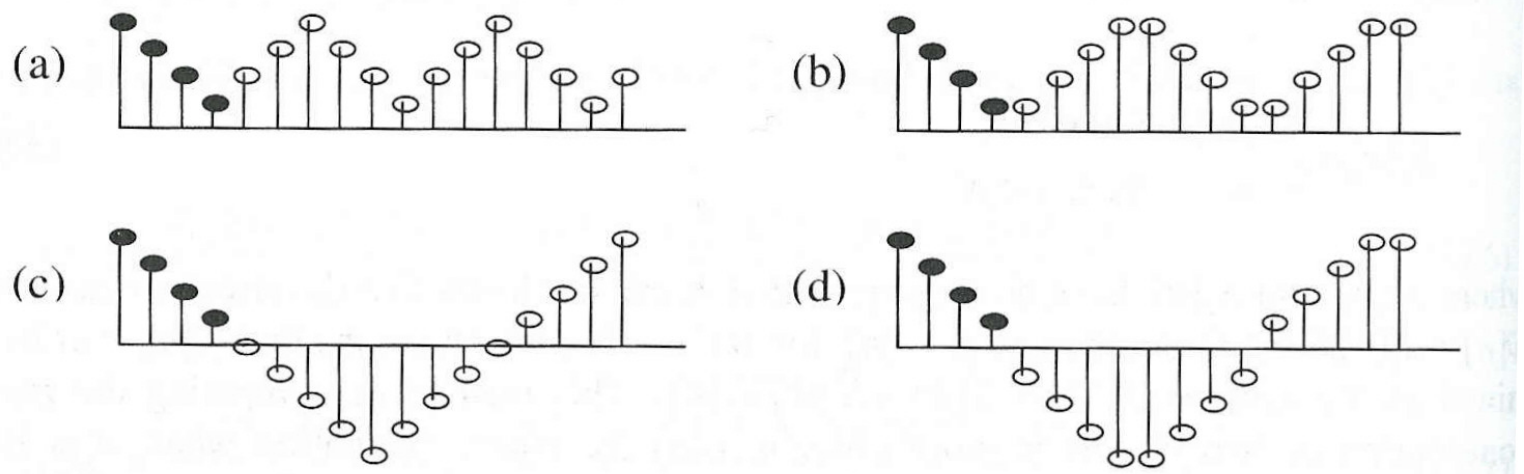


Figure 5.17 Four ways to extend a four-point sequence $x[n]$ to make it both periodic and have even symmetry. The figures in (a), (b), (c) and (d) correspond to the DCT-I, DCT-II, DCT-III and DCT-IV respectively.

The DCT-II scheme above concentrates the energy at lower frequencies thus making it somewhat easier to get by with fewer coefficients.

Taking the DCT-II yields:

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

Perceptual Linear Prediction

Reference: H. Hermansky, (1990) “Perceptual Linear Predictive Analysis of Speech”, J. Acoust. Soc. Am., 87(4) pp. 1738-1752

Perceptual linear prediction tries to merge the best features of Linear Prediction and MFCCs.

- Smooth spectral fit that matches higher amplitude components better than lower amplitude components (LP)
- Perceptually based frequency scale (MFCCs)
- Perceptually based amplitude scale (neither)

We compute the mel-warped power spectrum and take the cube root of power:

$$S(m) = (\sum_{k=0}^{N-1} |X(k)|^2 H_m(k))^{.33}$$

Then, the IDFT of a symmetrized version of $S(m)$ is taken:

$$R(m) = \text{IDFT}(S_{sym}(m))$$

This symmetrization ensures the result of the IDFT is real (the IDFT of a symmetric function is real).

We can now pretend that $R(m)$ are the autocorrelation coefficients of a genuine signal and compute LPC coefficients and cepstra as in “normal” LPC processing.

Vocal-tract length normalized features

Goal: Try to eliminate speaker-specific variability. Leave only variation due to differences in acoustics for each phoneme

In the following figure, we will see first and second formant positions for English vowels for a variety of speakers. Note that the inter-speaker variability causes overlap in the vowels, which is undesirable from a recognition point of view.

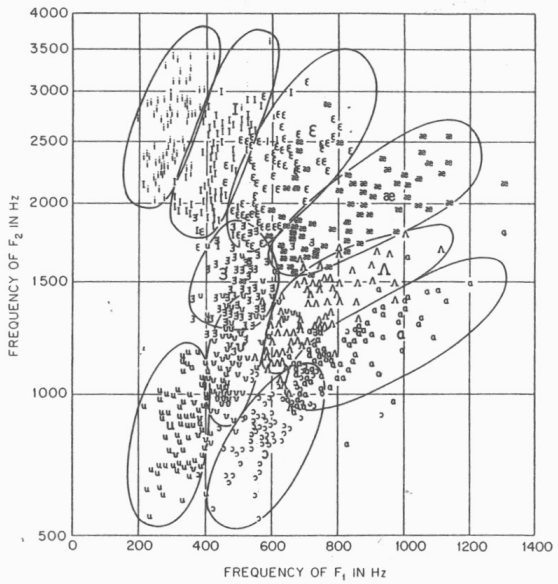
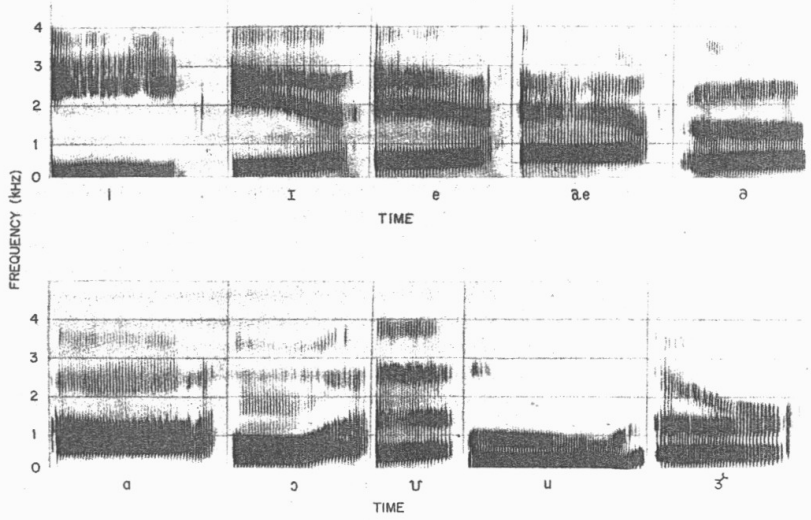


Fig. 3.4 Plot of second formant frequency versus first formant frequency for vowels by a wide range of speakers. (After Peterson and Barney [11].)

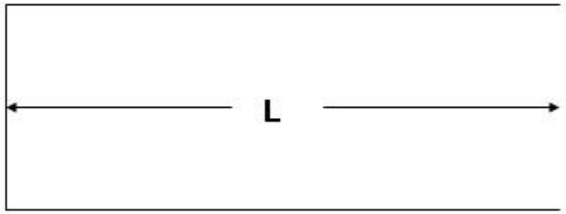
The following table indicates typical formant positions for male speakers.

FORMANT FREQUENCIES FOR THE VOWELS					
Typewritten Symbol for Vowel	IPA Symbol	Typical Word	F_1	F_2	F_3
IY	i	(beet)	270	2290	3010
I	ɪ	(bit)	390	1990	2550
E	ɛ	(bet)	530	1840	2480
AE	æ	(bat)	660	1720	2410
UH	ʌ	(but)	520	1190	2390
A	ɑ	(hot)	730	1090	2440
OW	ɔ	(bought)	570	840	2410
U	ʊ	(foot)	440	1020	2240
OO	u	(boot)	300	870	2240
ER	ɜ	(bird)	490	1350	1690

The following diagram shows what the spectra look like for these vowels as spoken by a male speaker.


FREQUENCY (KHz)
4
3
2
1
0
i
I
e
æe
ə
TIME
4
3
2
1
0
ɑ
ɔ
ʊ
u
ɝ
TIME

We can use a simple approximation of a vocal tract being a uniform tube of length L. For this model formant frequencies occur at odd multiples of $1/L$.



Scaling the tube by a factor k, so that the new length $L' = kL$,

results in formant frequencies being scaled linearly by a factor $1/k$.

Typically female speakers have formants which are roughly 20% higher than the formants for male speakers.

VTLN features try various scale factors and warp the frequency axis linearly during the FFT computation so as to fit some “canonical” speaker.

After the signal is transformed to the warped frequency domain, any feature computation e.g. MFCC’s can proceed normally.

Deltas and Double Deltas

Dynamic characteristics of sounds often convey significant information

- Stop closures and releases
- Formant transitions

Bright idea: augment normal "static" feature vector with dynamic features (first and second derivatives of the parameters). If y_t is the feature vector at time t, then compute

$$\Delta y_t = y_{t+D} - y_{t-D}$$

and create a new feature vector

$$y'_t = (y_t, \Delta y_t)$$

D is typically set to one or two frames. It is truly amazing that this relatively simple “hack” actually works quite well.

A more robust measure of the time derivative of the parameter can be computed using linear regression: A good five point derivative estimate is given by:

$$y'[t] = (y[t-2] - 8y[t-1] + 8y[t+1] - y[t+2])/12$$

A good five point estimate of the 2nd derivative is:

$$y''[t] = y[t-1] - 2y[t] + y[t+1]$$

What Feature Representation Works Best?

Evidence for the value of adding delta and delta-delta parameters are buried in old DARPA proceedings.

Many experiments comparing PLP and MFCC parameters are somewhat inconsistent - sometimes better, sometimes worse, depending on the task. The general consensus is PLP is slightly better, but it is always safe to stay with MFCC parameters.

Speech Recognition as Pattern Classification

A simple scenario

- for each word w, collect a single audio example A_w^{raw}
- to recognize audio signal A^{raw}, find word w that minimizes $\text{DISTANCE}(A^{\text{raw}}, A_w^{\text{raw}})$

Speech Recognition as Pattern Classification

Signal processing

- convert raw audio signals A^{raw} into salient features A
  - such that similar sounds have similar feature values
  - goal: want simple distance measures to work well
- example: MFCC features with Δ's and $\Delta\Delta$'s ($13 \times 3 = 39$ dimensional)
  - $A^{\text{raw}} \Leftrightarrow$ 16000 samples/sec
  - $A \Leftrightarrow$ 39 features/frame $\times$ 100 frames/sec

What Is a Reasonable Distance Measure?

- a simple scenario
  - for each word w, collect a single audio example A_w
  - to recognize audio signal A, find word w that minimizes $\text{DISTANCE}(A, A_w)$
- case 1: A, A_w are all the same length, say, T frames
  - $A(t) \equiv$ 39-dimensional feature vector for A at frame t

$$\text{DISTANCE}(A, A_w) = \sum_{t=1}^{T} \text{FRAMEDIST}(A(t), A_w(t))$$

What Is a Reasonable Frame Distance Measure?

- see what works well
- popular distances
  - Euclidean distance (L^2 norm): $\sqrt{\sum_i (A_i - A'_i)^2}$
  - L^p norm: $\sqrt[p]{\sum_i |A_i - A'_i|^p}$
  - weighted L^p norm: weight contributions from each dimension differently
    - *e.g.*, *liftering* for cepstra
  - Itakura; symmetrized Itakura
- whatever

Time is The Enemy

case 2: A, A_w have different lengths

- what to do?
- solution 1: make everything the same length
  - *e.g.*, *linear time normalization*
  - omit/duplicate frames uniformly in A_w so same length as A

$$\text{DISTANCE}(A, A_w) = \sum_{t=1}^{T} \text{FRAMEDIST}(A(t), A_w(t'))$$

where

$$t' = t \times \frac{\text{length}(A_w)}{\text{length}(A)}$$

Time is The Enemy

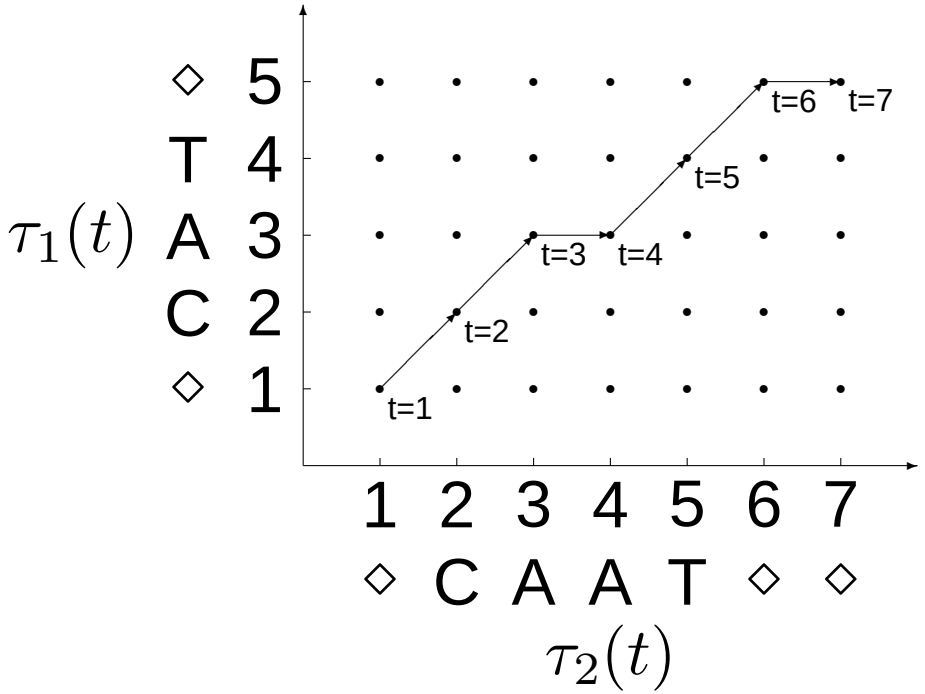
Is linear time normalization a good model of reality?

- do vowels and consonants stretch equally in time?
- handling silence
  - utterance 1: *silence* | CAT | *silence*
  - utterance 2: *silence* | CAT | *silence*
- want a *nonlinear* alignment scheme!

Dynamic Time Warping

$$\text{DISTANCE}(A_1, A_2) = \sum_{t=1}^{T} \text{FRAMEDIST}(A_1(\tau_1(t)), A_2(\tau_2(t)))$$

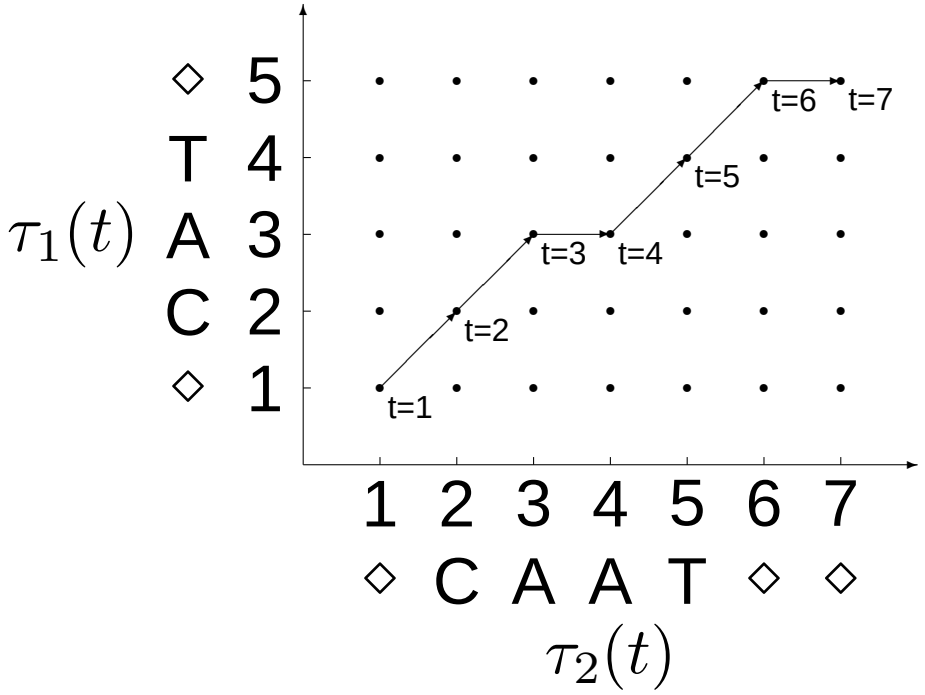
- introduce *warping* functions $\tau_1(t)$, $\tau_2(t)$
  - frame $\tau_1(t)$ in A_1 is *aligned* to frame $\tau_2(t)$ in A_2
  - a frame in A_1 can potentially align to any frame in A_2



Dynamic Time Warping

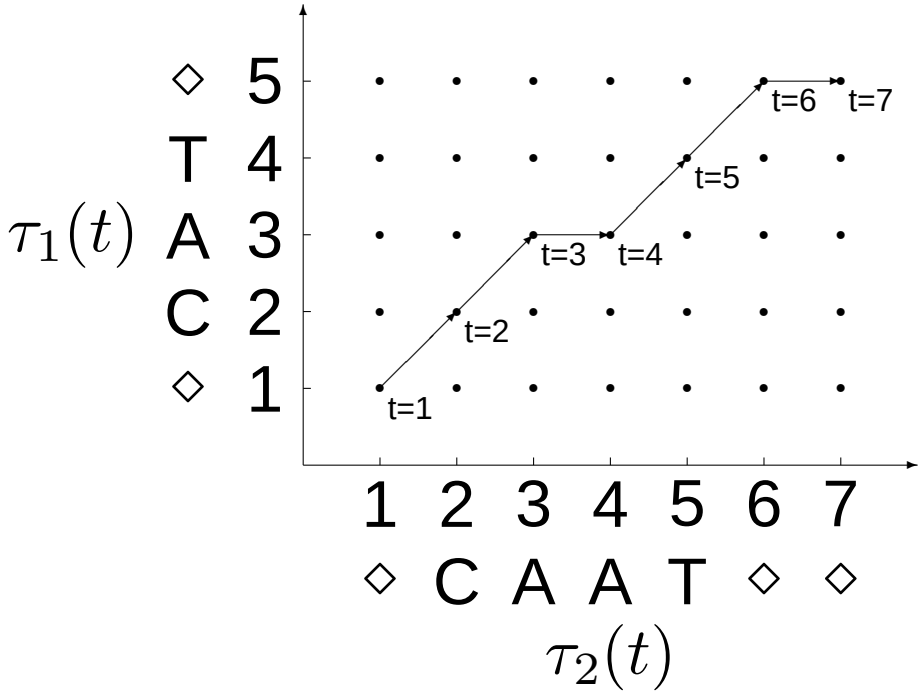
$$\text{DISTANCE}(A_1, A_2) = \sum_{t=1}^{T} \text{FRAMEDIST}(A_1(\tau_1(t)), A_2(\tau_2(t)))$$

- given a pair of warping functions $\tau_1(t)$, $\tau_2(t)$, distance is well-defined
  - how to constrain warping functions?
  - which particular pair of warping functions to pick?



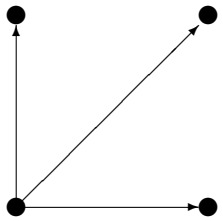
Constraining Warping Functions

- begin at the beginning; end at the end
  - $\tau_1(1) = 1$, $\tau_1(T) = \text{length}(A_1)$, $\tau_2(1) = 1$, $\tau_2(T) = \text{length}(A_2)$
- don't move backwards (monotonicity)
  - $\tau_1(t+1) \geq \tau_1(t)$, $\tau_2(t+1) \geq \tau_2(t)$
- don't move forwards too far (locality)
  - $\tau_1(t+1) \leq \tau_1(t) + 1$, $\tau_2(t+1) \leq \tau_2(t) + 1$



Constraining Warping Functions

- even better: constrain alignment to be comprised of sequence of *moves*
- *e.g.*, three possible moves
  - $\tau_1(t+1) = \tau_1(t) + 1, \tau_2(t+1) = \tau_2(t) + 1$
  - $\tau_1(t+1) = \tau_1(t) + 1, \tau_2(t+1) = \tau_2(t)$
  - $\tau_1(t+1) = \tau_1(t), \tau_2(t+1) = \tau_2(t) + 1$



- alignment must consist only of segments of these types

Selecting Warping Functions

$$\text{DISTANCE}(A_1, A_2) = \sum_{t=1}^{T} \text{FRAMEDIST}(A_1(\tau_1(t)), A_2(\tau_2(t)))$$

- which (legal) warping function to use to calculate the distance between A_1, A_2?
- consider all of them; pick the one with the smallest distance

$$\text{DISTANCE}(A_1, A_2) = \min_{\tau_1, \tau_2} \left\{ \sum_{t=1}^{T} \text{FRAMEDIST}(A_1(\tau_1(t)), A_2(\tau_2(t))) \right\}$$

Dynamic Programming

$$\text{DISTANCE}(A_1, A_2) = \min_{\tau_1, \tau_2} \left\{ \sum_{t=1}^{T} \text{FRAMEDIST}(A_1(\tau_1(t)), A_2(\tau_2(t))) \right\}$$

- wait! how can we efficiently compute the minimum distance over all warping functions?
  - there are an exponential number of warping functions τ_1, τ_2
  - computation exponential in time?
- no: dynamic programming!
  - computation quadratic in time ($\propto$ length$(A_1) \times$ length(A_2))
- why the name “dynamic programming”?
  - boss of author (Richard Bellman) had phobia of mathematics
  - author invented term to hide what he was really working on

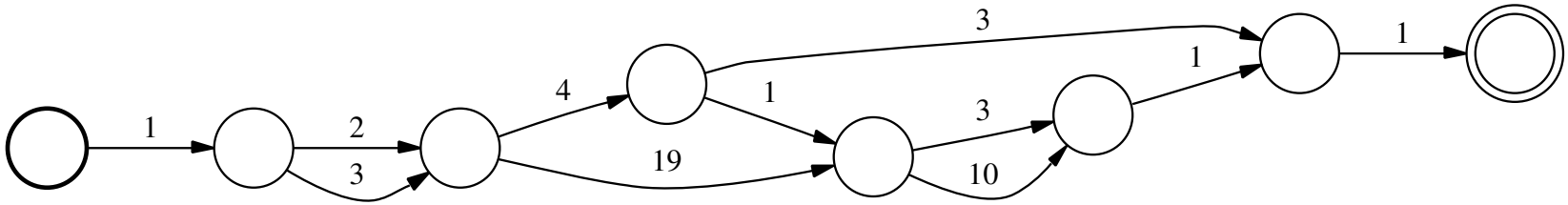
Shortest Path Problems

- DTW can be framed as instance of *shortest path* problem
- solvable using dynamic programming
- concepts useful in other speech algorithms (HMM's, finite-state machines)

Make A Bee-line for Great Taste!

Buzz along with the bee and see how many O's you can touch without flying over the same path twice. Add up your score, then go back and try for more.

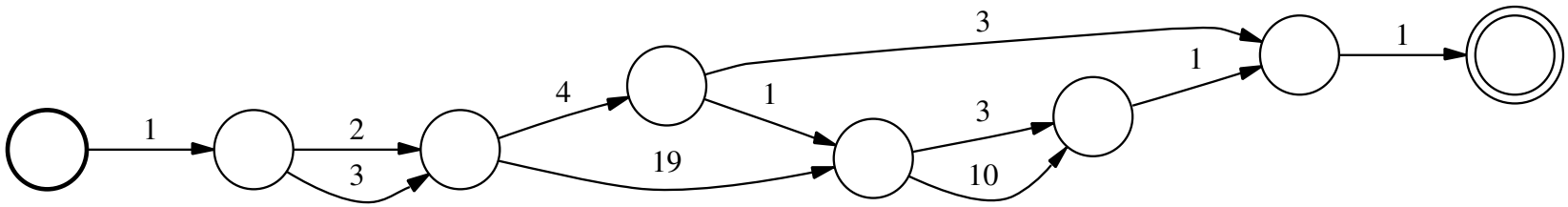
- how can we solve this baffling conundrum?
- we want *shortest* paths, so how *few* O's can you touch?



Shortest Path Problems

- key observation 1
  - shortest distance $d(S)$ from start state to state S can be expressed as ...
    - shortest distance $d(S')$ from start state to state S' plus $\textsc{distance}(S', S)$, for some immediate predecessor of S
    - if know $d(S')$ for all immediate predecessors of S, easy to compute $d(S)$

$$d(S) = \min_{S' \to S} \{d(S') + \textsc{distance}(S', S)\}$$



Shortest Path Problems

- proposed algorithm
  - loop through all states S in some order
  - compute distance to start state $d(S)$ for each state in order
    - need $d(S')$ already computed for all predecessors
  - can we always come up with such an ordering?
    - *i.e.*, an ordering such that all arcs go “forward”
- key observation 2
  - this is always possible for acyclic graphs
    - via *topological sorting*
  - in many cases, can come up with topological sorting manually

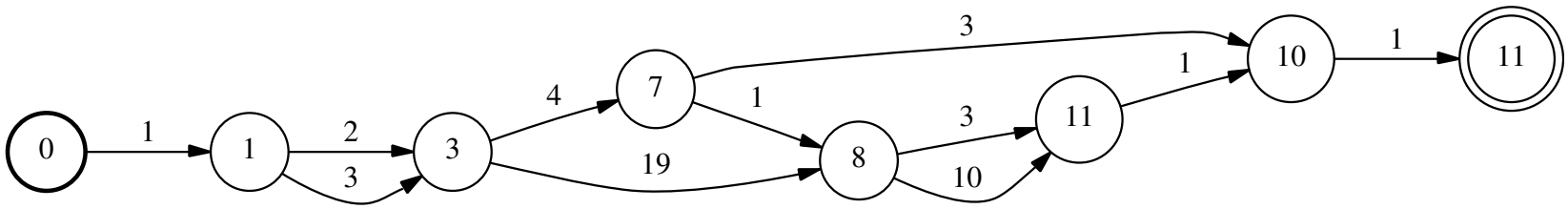
Shortest Path Problems

Algorithm

- sort states topologically: number from 1, . . . , N
  - number start state as state 1; final start as state N
  - for all arcs A, source$(A) <$ dest(A)
- $d(1) = 0$
- for $S = 2, \ldots, N$ do

$$d(S) = \min_{S' \to S} \{d(S') + \textsc{distance}(S', S)\}$$

- final answer: $d(N)$



DTW and Shortest Path Problems

$$\text{DISTANCE}(A_1, A_2) = \min_{\tau_1, \tau_2} \left\{ \sum_{t=1}^{T} \text{FRAMEDIST}(A_1(\tau_1(t)), A_2(\tau_2(t))) \right\}$$

- can we translate dynamic time warping into a shortest path problem?
- or was the whole dynamic programming discussion just our way of psyching you out?

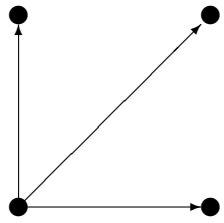
DTW and Shortest Path Problems

Consider the following problem

- align two frame utterance A_1 (A-T) with three frame utterance A_2 (A-A-T)
- frame distances FRAMEDIST$(A_1(t_1), A_2(t_2))$

	$A_2(1)$	$A_2(2)$	$A_2(3)$
$A_1(1)$	0	0	10
$A_1(2)$	10	10	0

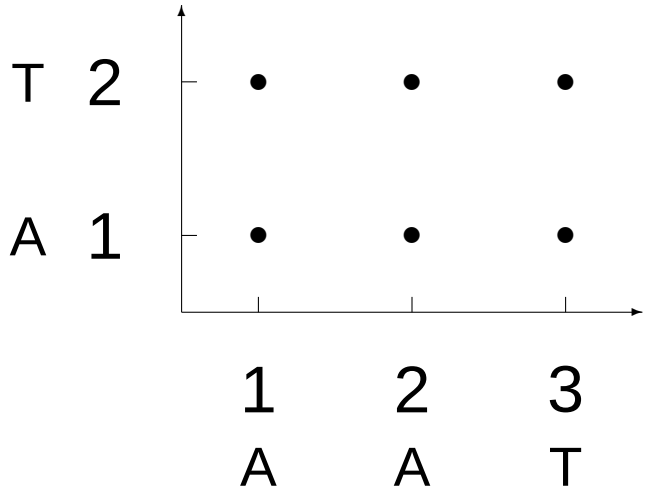
- move set
  - $\tau_1(t+1) = \tau_1(t) + 1, \tau_2(t+1) = \tau_2(t) + 1$
  - $\tau_1(t+1) = \tau_1(t) + 1, \tau_2(t+1) = \tau_2(t)$
  - $\tau_1(t+1) = \tau_1(t), \tau_2(t+1) = \tau_2(t) + 1$



DTW and Shortest Path Problems

Is this a shortest path problem?

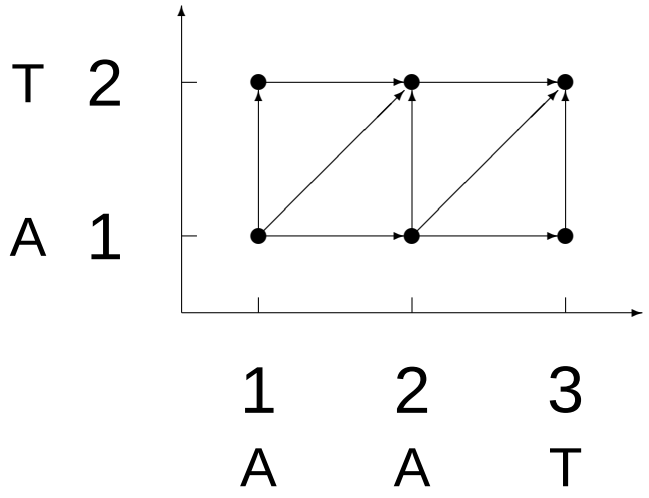
- we have the states of the graph; start state, final state
- a particular $\tau_1(t)$, $\tau_2(t)$ represents path from start to final state
- what are arcs of graph, and what are distances on each arc?



DTW and Shortest Path Problems

What are the arcs of the graph?

- at each state of the graph, you can take each move
  - these are the arcs!
  - discard arcs that go out of bounds

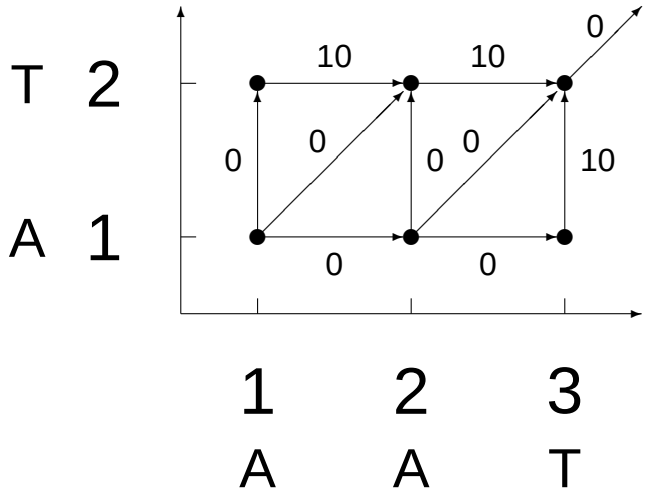


DTW and Shortest Path Problems

What are the distances on the arcs?

	$A_2(1)$	$A_2(2)$	$A_2(3)$
$A_1(1)$	0	0	10
$A_1(2)$	10	10	0

- take corresponding frame distance (at arc source)



DTW and Shortest Path Problems

Is DTW and shortest path on this graph equivalent?

$$\text{DISTANCE}(A_1, A_2) = \min_{\tau_1, \tau_2} \left\{ \sum_{t=1}^{T} \text{FRAMEDIST}(A_1(\tau_1(t)), A_2(\tau_2(t))) \right\}$$

- one-to-one correspondence between . . .
  - legal alignments $\tau_1(t)$, $\tau_2(t)$
  - paths in graph from start to final state
- distance associated with alignment is same as distance along corresponding path in graph?
- yes!
  - minimum distance alignment $\Leftrightarrow$ shortest path in graph

DTW and Shortest Path Problems

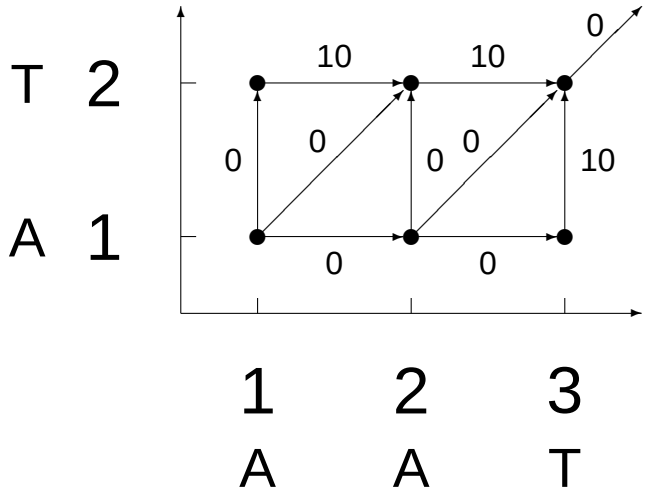
- sort states topologically
  - for all arcs A, source$(A) <$ dest(A)
  - $(t_1, t_2) \rightarrow (1,1), (1,2), (1,3), (2,1), (2,2), (2,3)$
- for $t_1 = 1, \ldots, 2$ do
  - for $t_2 = 1, \ldots, 3$ do

$$d(t_1, t_2) = \min\{$$
$$d(t_1 - 1, t_2 - 1) + \textsc{framedist}(A_1(t_1 - 1), A_2(t_2 - 1)),$$
$$d(t_1 - 1, t_2) + \textsc{framedist}(A_1(t_1 - 1), A_2(t_2)),$$
$$d(t_1, t_2 - 1) + \textsc{framedist}(A_1(t_1), A_2(t_2 - 1)) \}$$

- final answer: $d(2,3) + \textsc{framedist}(A_1(2), A_2(3))$

DTW and Shortest Path Problems

- let's simulate this algorithm on a 100Hz human brain



Recovering the Best Alignment

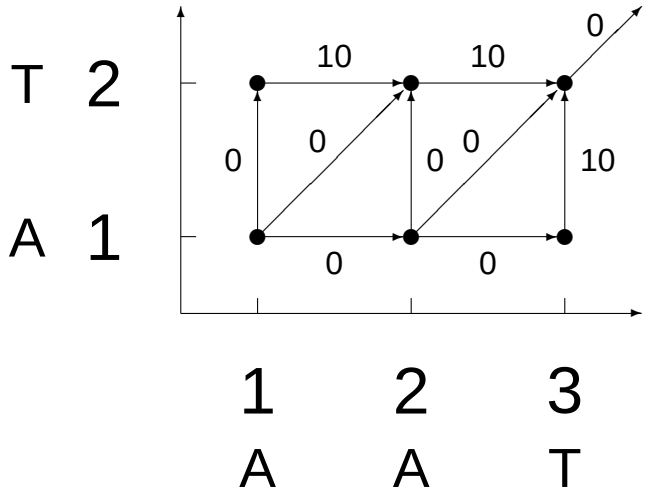
- what if we want to know which alignment produced the shortest distance?
  - keep *traceback* information

$$
\begin{aligned}
d(t_1, t_2) = \min\{ & \\
& d(t_1 - 1, t_2 - 1) + \textsc{framedist}(A_1(t_1 - 1), A_2(t_2 - 1)), \\
& d(t_1 - 1, t_2) + \textsc{framedist}(A_1(t_1 - 1), A_2(t_2)), \\
& d(t_1, t_2 - 1) + \textsc{framedist}(A_1(t_1), A_2(t_2 - 1)) \ \}
\end{aligned}
$$

traceback$(t_1, t_2) =$ which source state was best

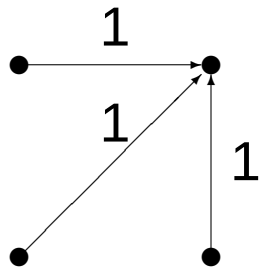
Recovering the Best Alignment

- trace back from final state to get best path: (3,2), (2,1), (1, 1)



Different Move Sets and Weighting

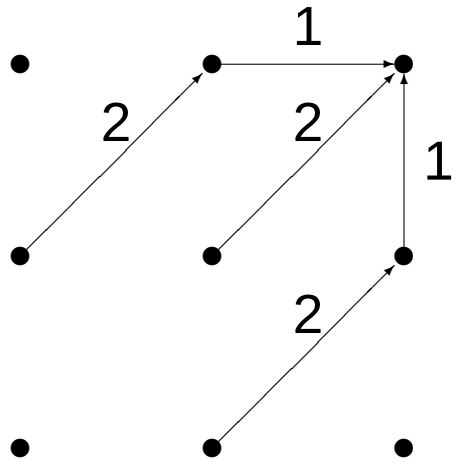
One move set



$$d(t_1, t_2) = \min\{$$
$$d(t_1 - 1, t_2 - 1) + \text{FRAMEDIST}(A_1(t_1 - 1), A_2(t_2 - 1)),$$
$$d(t_1 - 1, t_2) + \text{FRAMEDIST}(A_1(t_1 - 1), A_2(t_2)),$$
$$d(t_1, t_2 - 1) + \text{FRAMEDIST}(A_1(t_1), A_2(t_2 - 1)) \}$$

Different Move Sets and Weighting

Another move set (Sakoe and Chiba)



$$
\begin{aligned}
d(t_1, t_2) = \min\{ & \\
& d(t_1 - 1, t_2 - 1) + 2 \times \textsc{framedist}(A_1(t_1 - 1), A_2(t_2 - 1)), \\
& d(t_1 - 2, t_2 - 1) + 2 \times \textsc{framedist}(A_1(t_1 - 2), A_2(t_2 - 1)) + \\
& \qquad \textsc{framedist}(A_1(t_1 - 1), A_2(t_2 - 1)), \\
& d(t_1 - 1, t_2 - 2) + 2 \times \textsc{framedist}(A_1(t_1 - 1), A_2(t_2 - 2)) + \\
& \qquad \textsc{framedist}(A_1(t_1 - 1), A_2(t_2 - 1)) \}
\end{aligned}
$$

Normalization

$$\text{DISTANCE}(A_1, A_2) = \sum_{t=1}^{T} \text{FRAMEDIST}(A_1(\tau_1(t)), A_2(\tau_2(t)))$$

- correct bias for longer utterances to have longer distances

$$\text{DISTANCE}(A_1, A_2) = \frac{\sum_{t=1}^{T} \text{FRAMEDIST}(A_1(\tau_1(t)), A_2(\tau_2(t)))}{\text{length}(A_1) + \text{length}(A_2)}$$

Summary

- DTW is effective way to calculate distance between two signals
  - can do nonlinear time alignment
  - frame distance and move set selection is *ad hoc*
  - dynamic programming can implement DTW efficiently
- can be extended to multiple training examples per word
  - select “best” examples or do averaging
- can be extended to connected speech
  - align sequence of templates to single utterance
- signal processing and DTW are all you need for simple ASR
  - *e.g.*, cell phone name dialer

A Simple Recognizer

- training
  - for each word w, collect a single audio example A_w^{raw}, or *template*
    - do signal processing to get features A_w
- recognition
  - process audio signal A^{raw} to get features A
  - find closest word w^* from training examples

$$w^* = \arg\min_w \text{DISTANCE}(A, A_w)$$

    - $\text{DISTANCE}(A, A_w)$ can be computed using DTW, dynamic programming

The Big Picture

- signal processing and dynamic time warping
  - all you need for a simple speech recognizer (*e.g.*, Lab 1)
    - very small number of training examples
  - pat yourself on the back
- what's next
  - instead of *ad hoc* distance measures
  - start putting things on a sounder mathematical foundation
  - probabilistic modeling! (GMM's, HMM's)
    - large training sets

Course Feedback

1. Was this lecture mostly clear or unclear? What was the muddiest topic?
2. Other feedback (pace, content, atmosphere)?