Lecture 2

Signal Processing and Dynamic Time Warping

Michael Picheny, Bhuvana Ramabhadran, Stanley F. Chen

IBM T.J. Watson Research Center
Yorktown Heights, New York, USA
{picheny,bhuvana,stanchen}@us.ibm.com

17 September 2012

Administrivia

- Students are 75% EE, 25% CS.
- Top three goals:
  - General understanding of ASR theory.
  - Learn about ASR implementation/practice.
  - Learn about ML/AI/pattern recognition.
- Feedback (2+ votes):
  - Signal processing fast/muddy.
  - Hard to hear/speak too fast.
  - Stan shouldn't read slides.
- Thank you for comments!!!

Demo of Web Site

`www.ee.columbia.edu/~stanchen/fall12/e6870/`

- Will provide hardcopies of readings + slides.
- PDF readings up on web site by Friday before lecture.
  - Username: *speech*, password: *pythonrules*
- PDF slides on web site by 8pm day before lecture (usually).

A Word on Programming Languages

- Everyone (not including auditors) knows C, C++, or Java.
  - Will support C++ and Java (not as well).
  - Only basic C++ used; will document stuff outside of C.
- C++ is the international language of speech recognition.
  - Speed! (Have you heard of Sphinx 4?)
  - Java users will suffer a little in a couple labs.
- Why not Matlab?
  - Can implement signal processing algorithms quickly . . .
  - But not as good for later labs.

Review: A Very Simple Speech Recognizer

- Training data: audio sample A_w for every word $w \in$ vocab.
- Given test sample A_{test}, pick word w^*:

$$w^* = \arg\min_{w \in \text{vocab}} \text{distance}(A_{\text{test}}, A_w)$$

Today's Lecture

$$w^* = \arg\min_{w \in \text{vocab}} \text{distance}(A_{\text{test}}, A_w)$$

- *signal processing* — Extract *features* from audio . . .
  - So simple distance measure works.
- *dynamic time warping* — Handling time/rate variation in the distance measure.

Part I

Signal Processing

Goals of Feature Extraction

- Capture essential information for word identification.
- Make it easy to factor out irrelevant information.
  - *e.g.*, long-term channel transmission characteristics.
- Compress information into manageable form.

Figures in this section from **[Holmes]**, **[HAH]** or **[R+J]** unless indicated otherwise.

What Has Actually Worked?

- 1950s–1960s — Analog filterbanks.
- 1970s — Linear Predictive Coding (LPC).
- 1980s — LPC Cepstra.
- 1990s — Mel-Scale Cepstral Coefficients (MFCC) and Perceptual Linear Prediction (PLP).
- 2000s — Posteriors and multistream combinations.

Concept: The Frame

- Raw 16kHz input: sample every $\frac{1}{16000}$ sec.
- What should output look like?
  - Point: speech phenomena aren't *that* short.
  - *e.g.*, output *frame* of features every, say, $\frac{1}{100}$ sec ...
  - Describing what happened in that $\frac{1}{100}$ sec.
- How wide should feature vector be?
  - Empirically: 40 or so.
- *e.g.*, 1s of audio: 16000×1 nums in $\Rightarrow$ 100×40 nums out.

LPC Ceptra, MFCC, and PLP: The Basic Idea

- For each frame:
  - Step 1: Compute short-term spectrum.
  - Step 2: From spectrum, compute *cepstrum*.
  - Step 3: Profit!
- Each method does these steps differently.
  - LPC inspired by human production.
  - MFCC, PLP inspired by human perception.

What is a Short-Term Spectrum?

- Extract out *window* of samples for that frame.
- Compute energy at each frequency using discrete Fourier transform.
  - Look at signal as decomposition of its frequency components.
- Lots more gory details in next section.

Why the Short-Term Spectrum?

- Matches human perception/physiology?
  - This sounds like what the cochlea is doing?
- Frequency information distinguishes phonemes.
  - Formants identify vowels; *e.g.*, Pattern Playback machine.
  - Humans can “read” spectrograms.
- Speech is not stationary signal.
  - Want information about small enough region . . .
  - Such that spectral information is useful feature.

What is a Cepstrum?

- (Inverse) Fourier transform of . . .
  - Logarithm of the (magnitude of the) spectrum.
  - Homomorphic transformation
- In practice, spectrum is “smoothed” first.
  - *e.g.*, via LPC and/or Mel binning.

What is a Cepstrum?

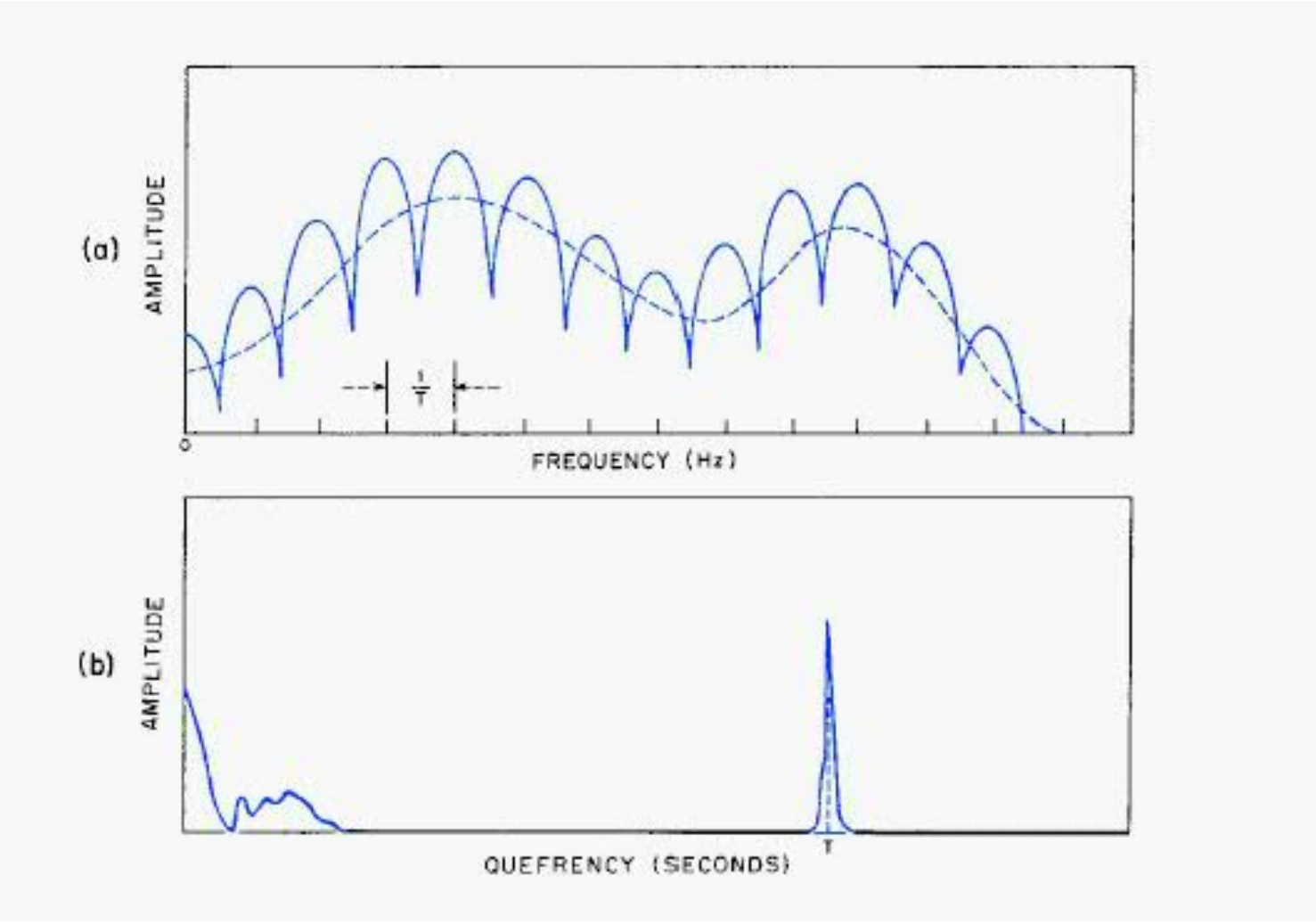
- (Inverse) Fourier transform of . . .
  - Logarithm of the (magnitude of the) spectrum.
  - Homomorphic transformation
- In practice, spectrum is “smoothed” first.
  - *e.g.*, via LPC and/or Mel binning.

Why the Cepstrum?

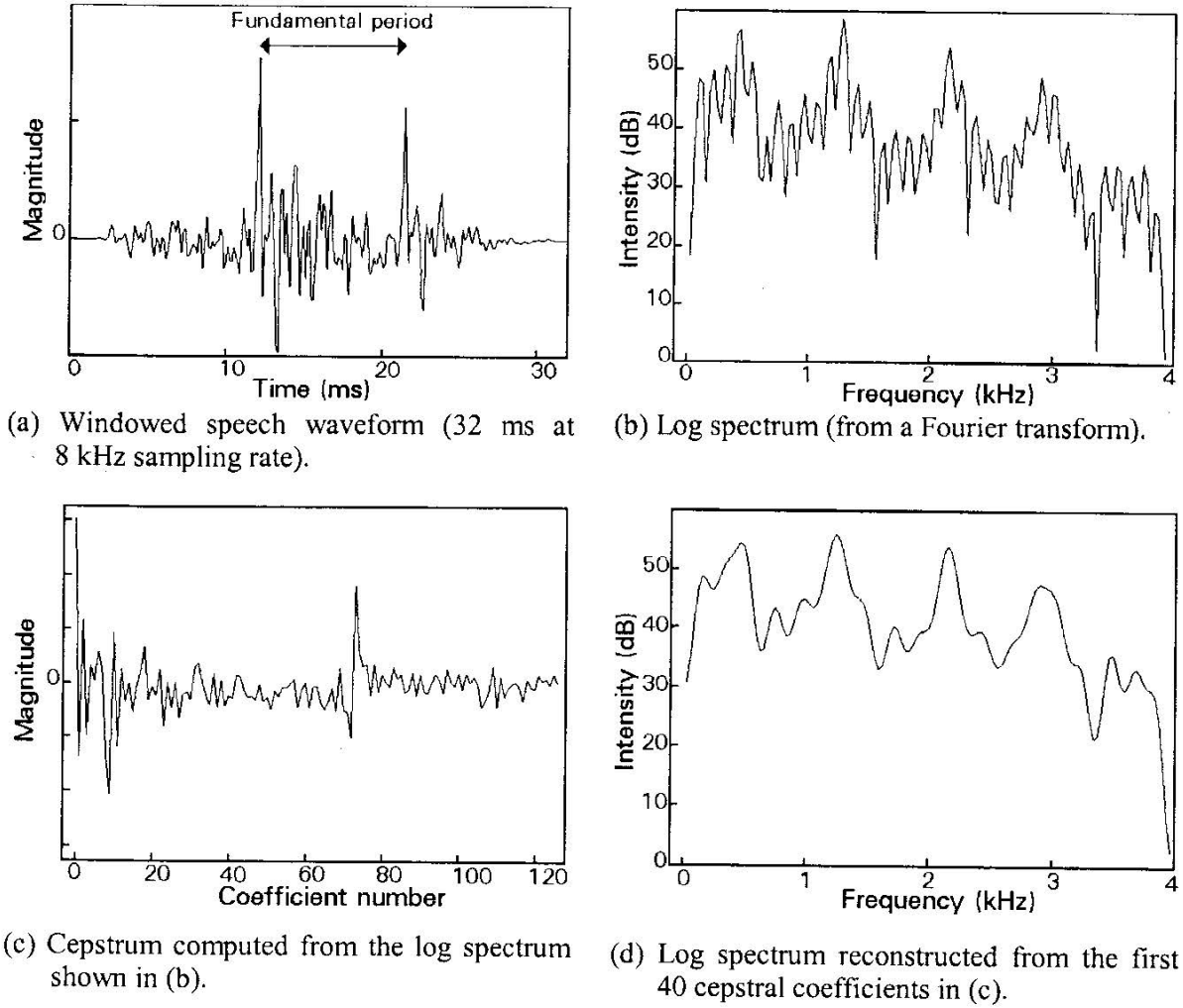
- Lets us separate excitation (source; don't care) . . .
- From vocal tract resonances (filter; do care).
  - Vocal tract changes shape slowly with time.
  - Assume fixed properties over small interval (10 ms).
  - Its natural frequencies are formants (resonances).
- Low *quefrencies* correspond to vocal tract.

View of the Cepstrum (Voiced Speech)

- Cepstrum contains peaks at multiples of pitch period.



Cepstrum of a speech signal



(a) Windowed speech waveform (32 ms at 8 kHz sampling rate).

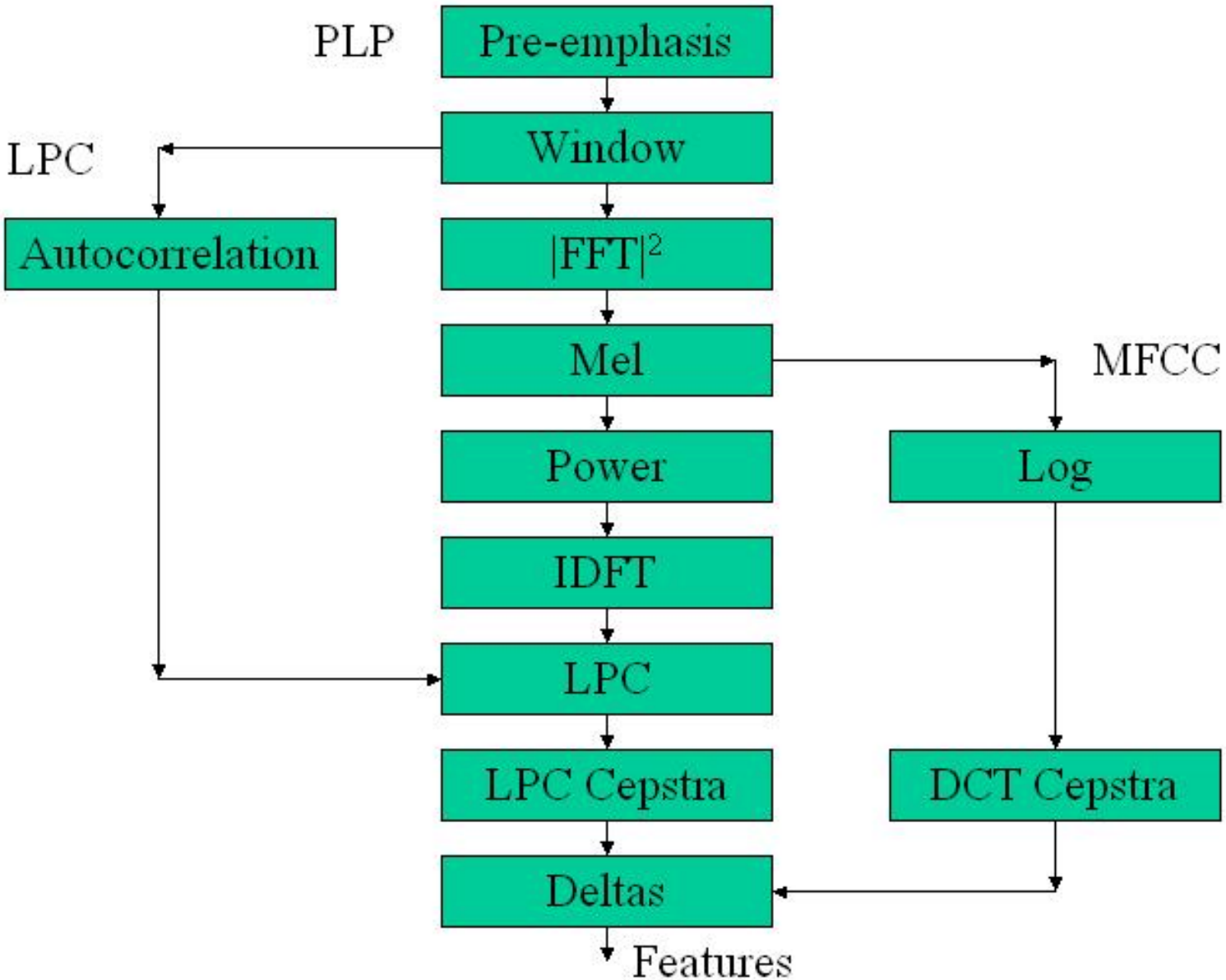
(b) Log spectrum (from a Fourier transform).

(c) Cepstrum computed from the log spectrum shown in (b).

(d) Log spectrum reconstructed from the first 40 cepstral coefficients in (c).

Figure 10.3 Analysing a section of speech waveform to obtain the cepstrum and then to reconstruct a cepstrally smoothed spectrum.

LPC Ceptra, MFCC, and PLP: Overview



Where Are We?

The Short-Time Spectrum

- Extract out *window* of N samples for that frame.
- Compute energy at each frequency using fast Fourier transform.
  - Standard algorithm for computing DFT.
  - Complexity $N \log N$; usually take $N = 512, 1024$ or so.
- What's the problem?
  - The devil is in the details.
  - *e.g.*, frame rate; window length; window shape.

Windowing

- Samples for mth frame (counting from 0):

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

- $w[n]$ = window function, *e.g.*,

$$w[n] = \begin{cases} 1 & n = 0, \ldots, N-1 \\ 0 & \text{otherwise} \end{cases}$$

- N = window length.
- F = frame spacing, e.g., $\frac{1}{100}$ sec $\Leftrightarrow$ 160 samples at 16kHz.

How to Choose Frame Spacing?

- Experiments in speech coding intelligibility suggest that F should be around 10 msec ($= \frac{1}{100}$ sec).
- For $F > 20$ msec, one starts hearing noticeable distortion.
- Smaller F and no improvement.
  - The smaller the F, the more the computation.

How to Choose Window Length?

- If too long, vocal tract will be non-stationary.
  - Smears out transients like stops.
- If too short, spectral output will be too variable with respect to window placement.
- Time *vs.* frequency resolution (Fig. from [4]).
- Usually choose 20-25 msec window as compromise.

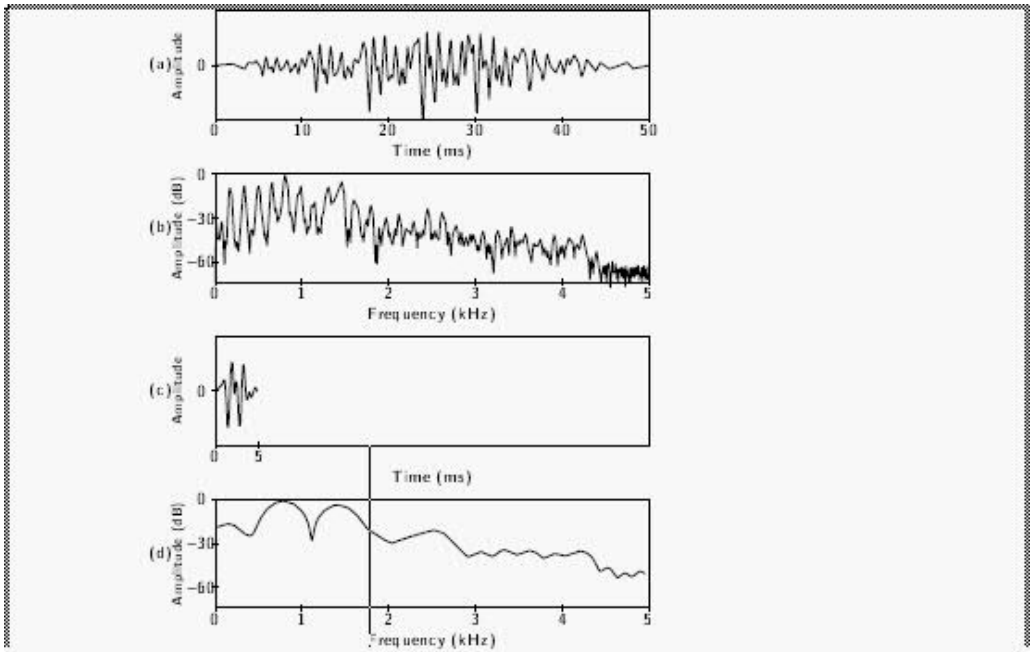
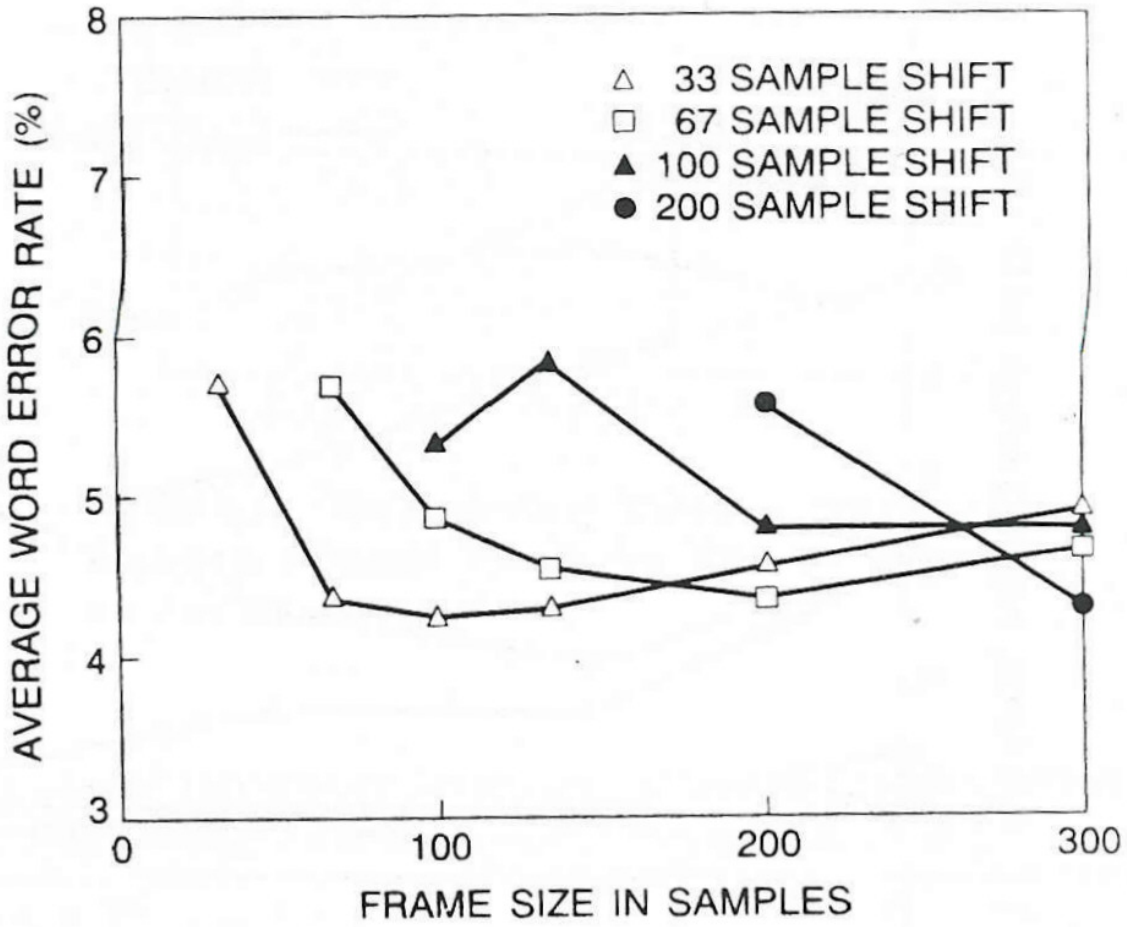


Figure 2.6: Time signals and spectra of a vowel: (a) signal multiplied by a 50-ms Hamming window; (b) the corresponding spectrum (note that harmonic structure is strongest at low frequencies); (c) signal multiplied by a 5-ms Hamming window; (d) its corresponding spectrum.

Optimal Frame Rate



- Few studies of frame rate vs. error rate.
- Above curves suggest that the frame rate should be one-third of the frame size.

Analyzing Window Shape

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

- Convolution theorem: multiplication in time domain is same as convolution in frequency domain.
  - Fourier transform of result is $X(\omega) * W(\omega)$.
- Imagine original signal is periodic.
  - Ideal: after windowing, $X(\omega)$ remains unchanged $\Leftrightarrow$ $W(\omega)$ is delta function.
  - Reality: short-term window cannot be perfect.
  - How close can we get to ideal?

Rectangular Window

$$w[n] = \begin{cases} 1 & n = 0, \ldots, N-1 \\ 0 & \text{otherwise} \end{cases}$$

- The FFT can be written in closed form as

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

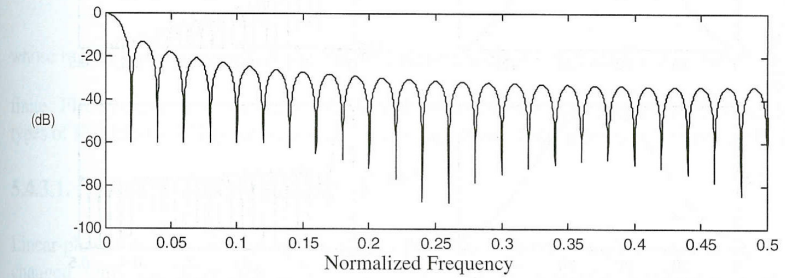


Figure 5.19 Frequency response (magnitude in dB) of the rectangular window with $N = 50$, which is a digital sinc function.

- High sidelobes tend to distort low-energy spectral components when high-energy components present.

Hanning and Hamming Windows

- Hanning: $w[n] = .5 - .5\cos 2\pi n/N$
- Hamming: $w[n] = .54 - .46\cos 2\pi n/N$

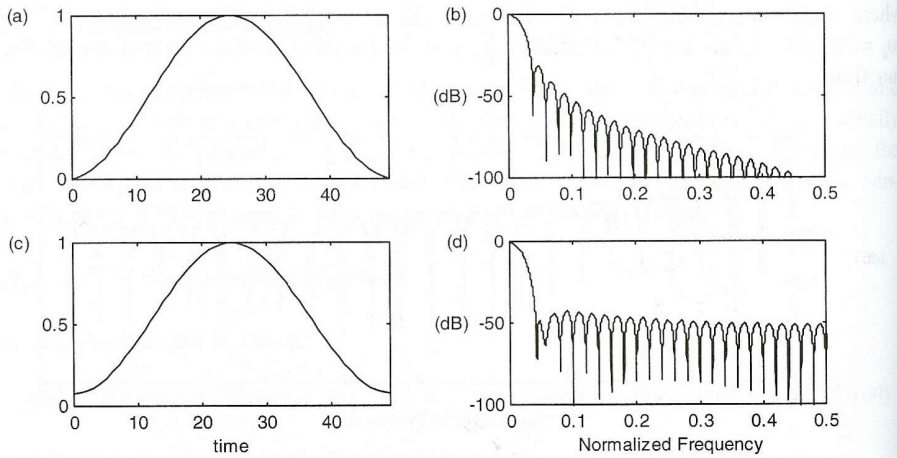


Figure 5.20 (a) Hanning window and (b) the magnitude of its frequency response in dB; (c) Hamming window and (d) the magnitude of its frequency response in dB for $N = 50$.

- Hanning and Hamming have slightly wider main lobes, much lower sidelobes than rectangular window.
- Hamming window has lower first sidelobe than Hanning; sidelobes at higher frequencies do not roll off as much.

Effects of Windowing

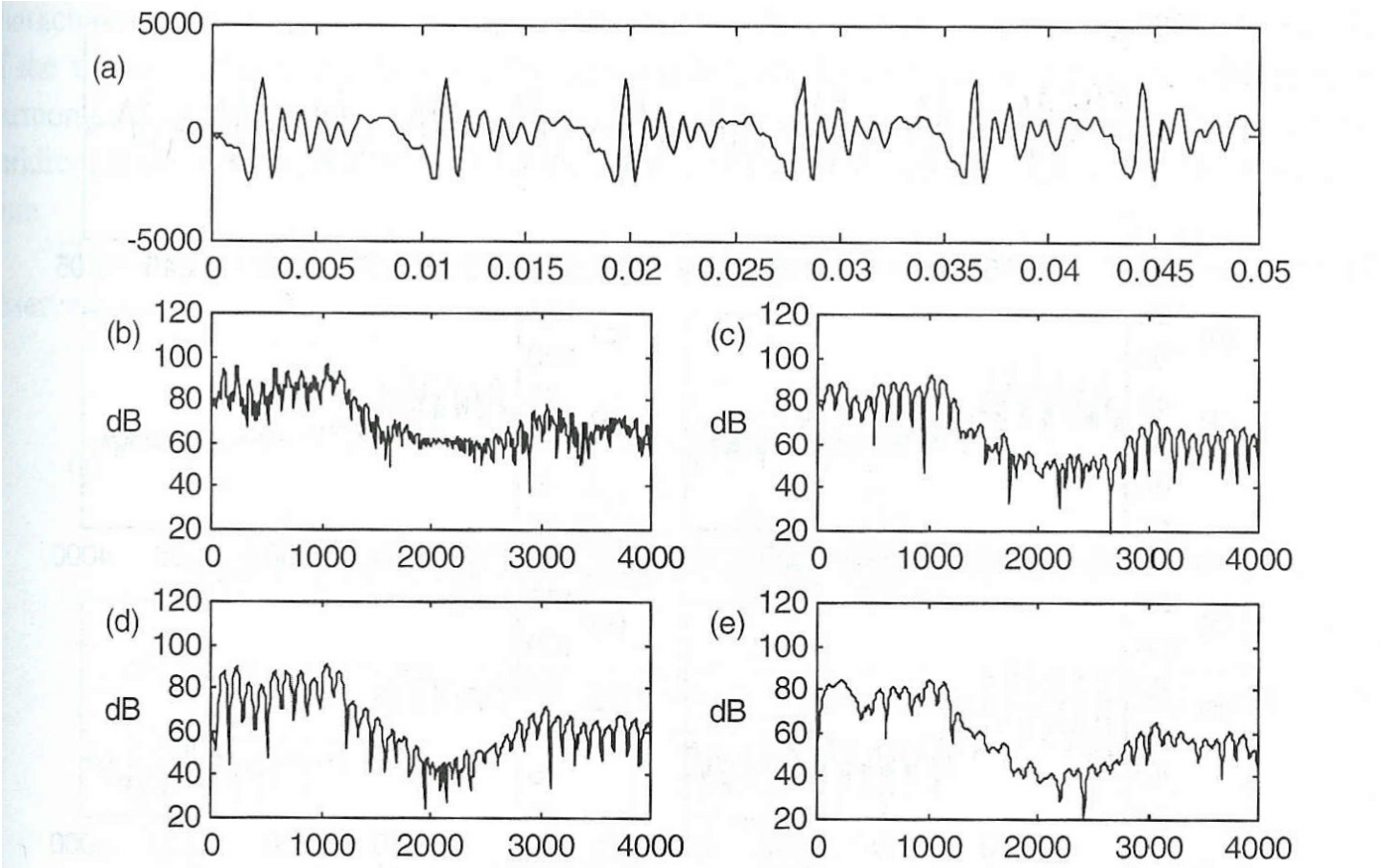


Figure 6.3 Short-time spectrum of male voiced speech (vowel /ah/ with local pitch of 110Hz): (a) time signal, spectra obtained with (b) 30 ms rectangular window and (c) 15 ms rectangular window, (d) 30 ms Hamming window, (e) 15 ms Hamming window. The window lobes are not visible in (e), since the window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

Effects of Windowing

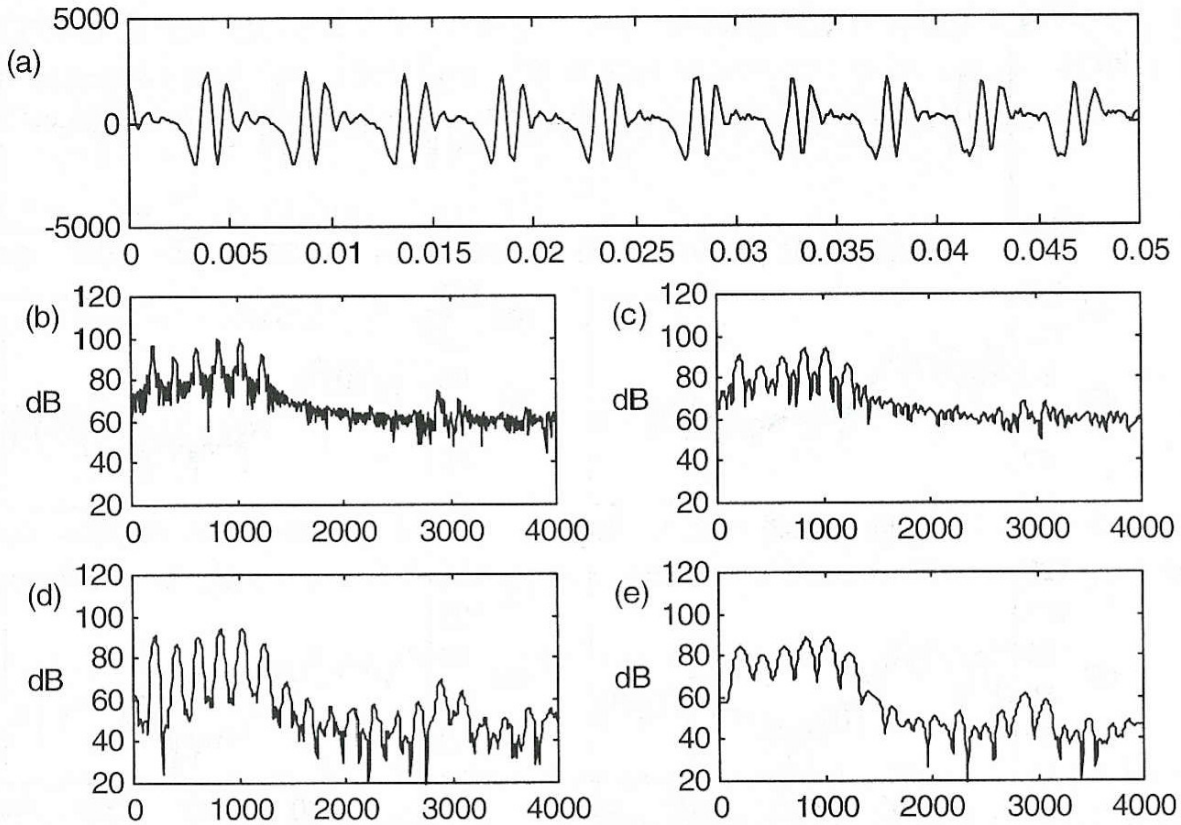


Figure 6.4 Short-time spectrum of female voiced speech (vowel /aa/ with local pitch of 200Hz): (a) time signal, spectra obtained with (b) 30 ms rectangular window and (c) 15 ms rectangular window, (d) 30 ms Hamming window, (e) 15 ms Hamming window. In all cases the window lobes are visible, since the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

Effects of Windowing

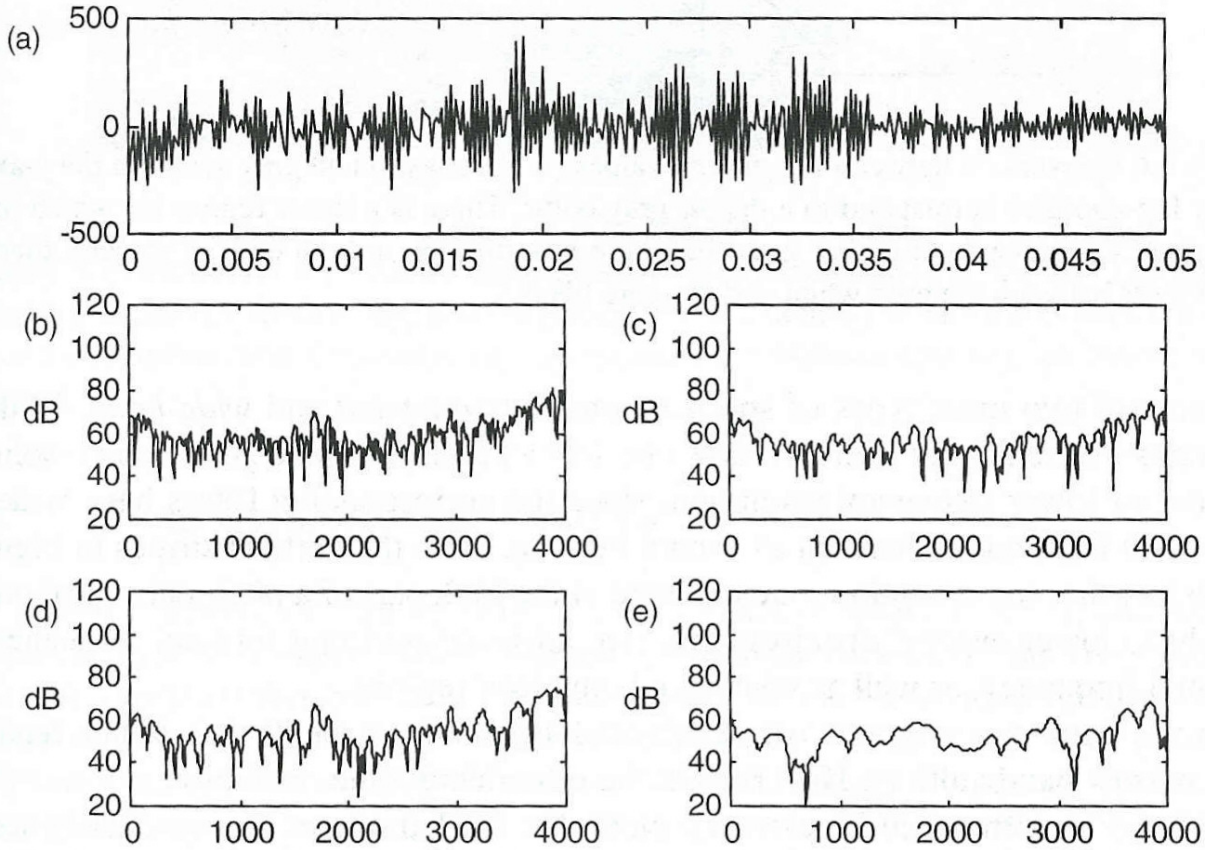
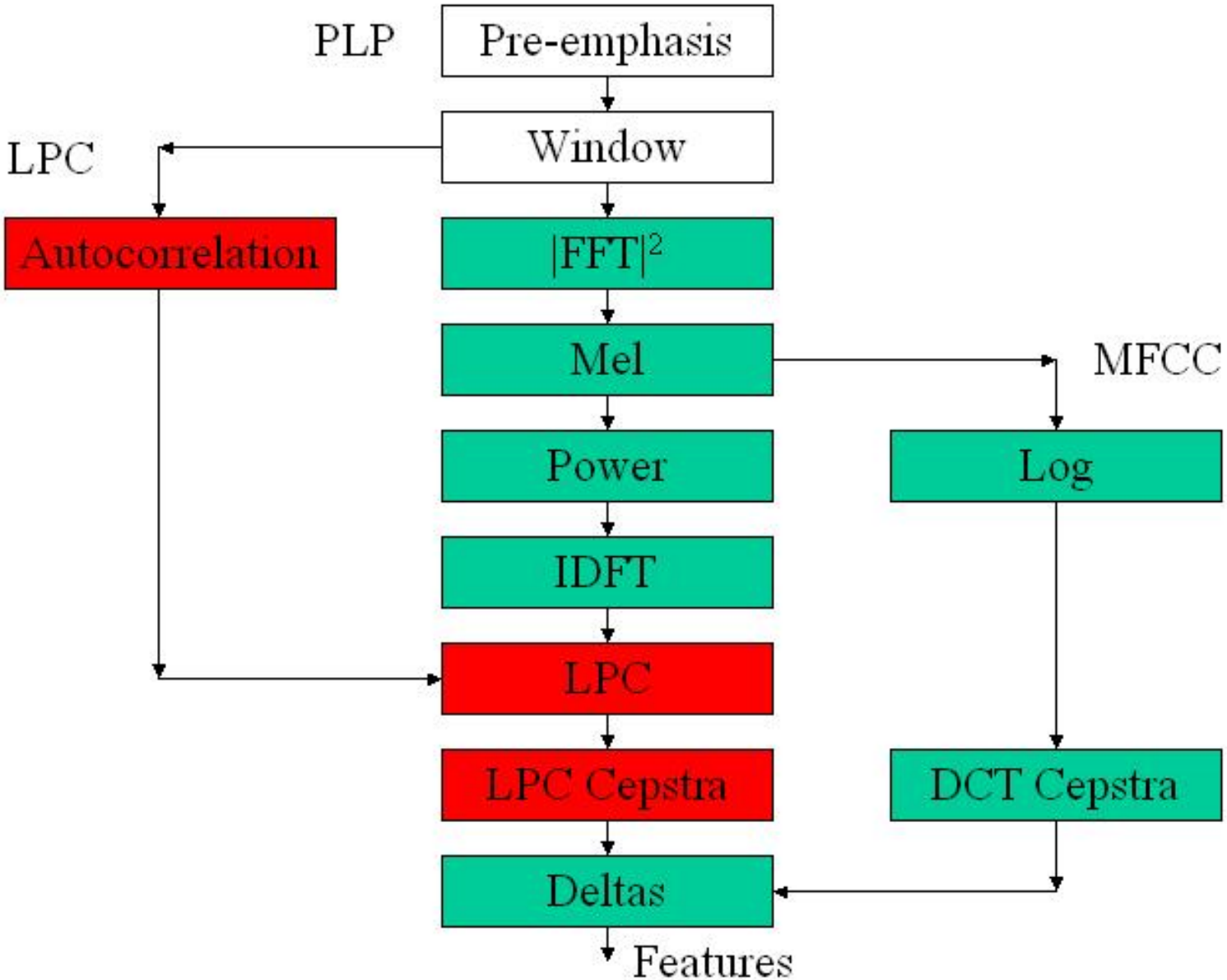


Figure 6.5 Short-time spectrum of unvoiced speech: (a) time signal, (b) 30 ms rectangular window, (c) 15 ms rectangular window, (d) 30 ms Hamming window, (e) 15 ms Hamming window.

- What do you notice about all these spectra?

Where Are We?

Linear Prediction



Linear Prediction: Motivation

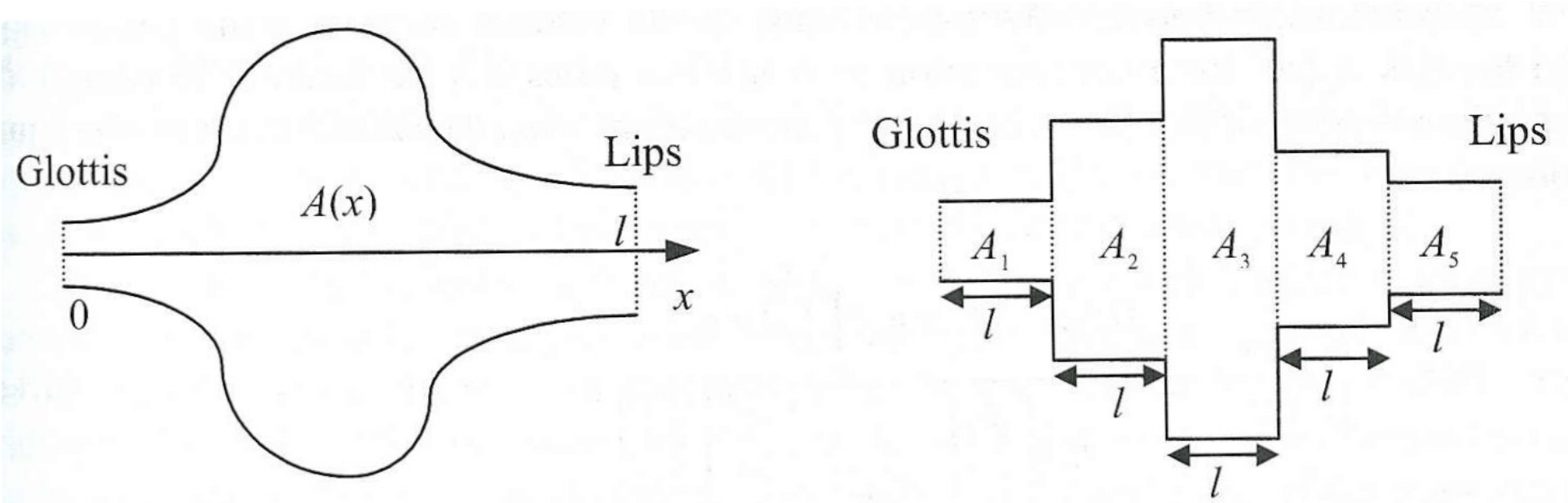
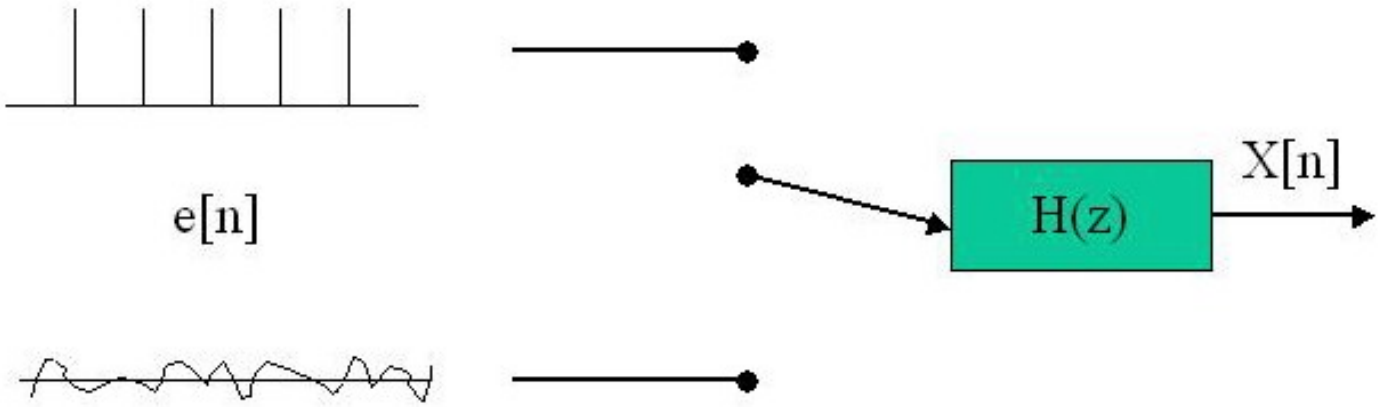


Figure 6.9 Approximation of a tube with continuously varying area $A(x)$ as a concatenation of 5 lossless acoustic tubes.

- Above model of vocal tract matches observed data well.
- Can be represented by filter $H(z)$ with simple time-domain interpretation.

Linear Prediction



- The *linear prediction* model assumes output $x[n]$ is linear combination of p previous samples and excitation $e[n]$ (scaled by *gain* G).

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

- $e[n]$ is impulse train representing pitch (voiced) . . .
  - Or white noise (for unvoiced sounds).

The General Idea

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

- Given audio signal $x[n]$, solve for $a[j]$ that . . .
  - Minimizes prediction error.
- Ignore $e[n]$ term when solve for $a[j] \Rightarrow$ unknown!
  - Assume $e[n]$ will be approximated by prediction error!
- The hope:
  - The $a[j]$ characterize shape of vocal tract.
  - May be good features for identifying sounds?
  - Prediction error is either impulse train or white noise.

Solving the Linear Prediction Equations

- Goal: find $a[j]$ that minimize prediction error:

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

- Take derivatives w.r.t. $a[i]$ and set to 0:

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

  where $R(i)$ is *autocorrelation* sequence for current window of samples.
- Above set of linear equations is *Toeplitz* and can be solved using *Levinson-Durbin recursion* ($O(n^2)$ rather than $O(n^3)$ as for general linear equations).

Analyzing Linear Prediction

- Recall: Z-Transform is generalization of Fourier transform.
- The Z-transform of associated filter is:

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

- $H(z)$ with $z = e^{j\omega}$ gives us LPC spectrum.

The LPC Spectrum

- Comparison of original spectrum and LPC spectrum.

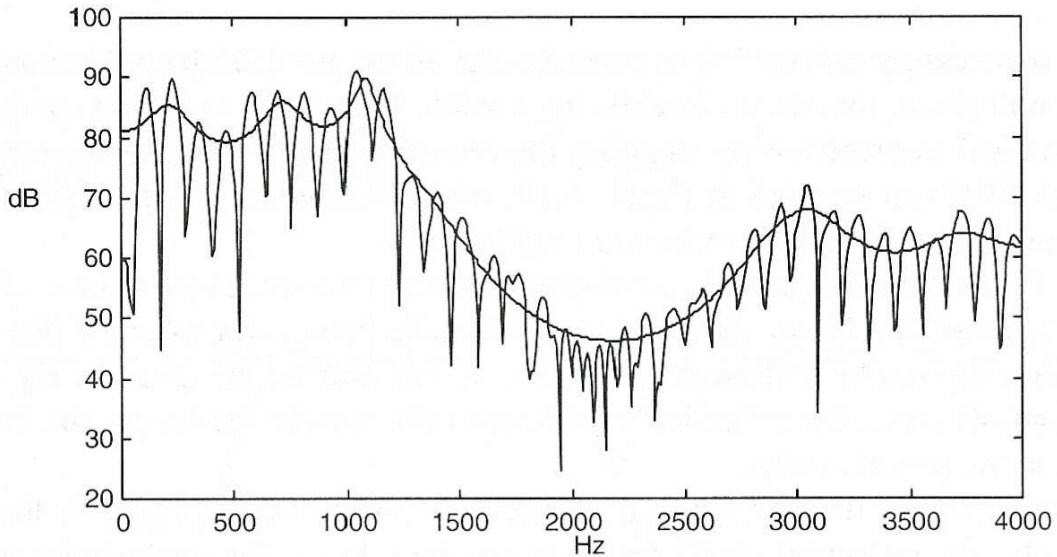


Figure 6.20 LPC spectrum of the */ah/* phoneme in the word *lives* of Figure 6.3. Used here are a 30-ms Hamming window and the autocorrelation method with $p = 14$. The short-time spectrum is also shown.

- LPC spectrum follows peaks and ignores dips.
  - LPC error $\int E(z) = X(z)/H(z)dz$ forces better match at peaks.

Example: Prediction Error

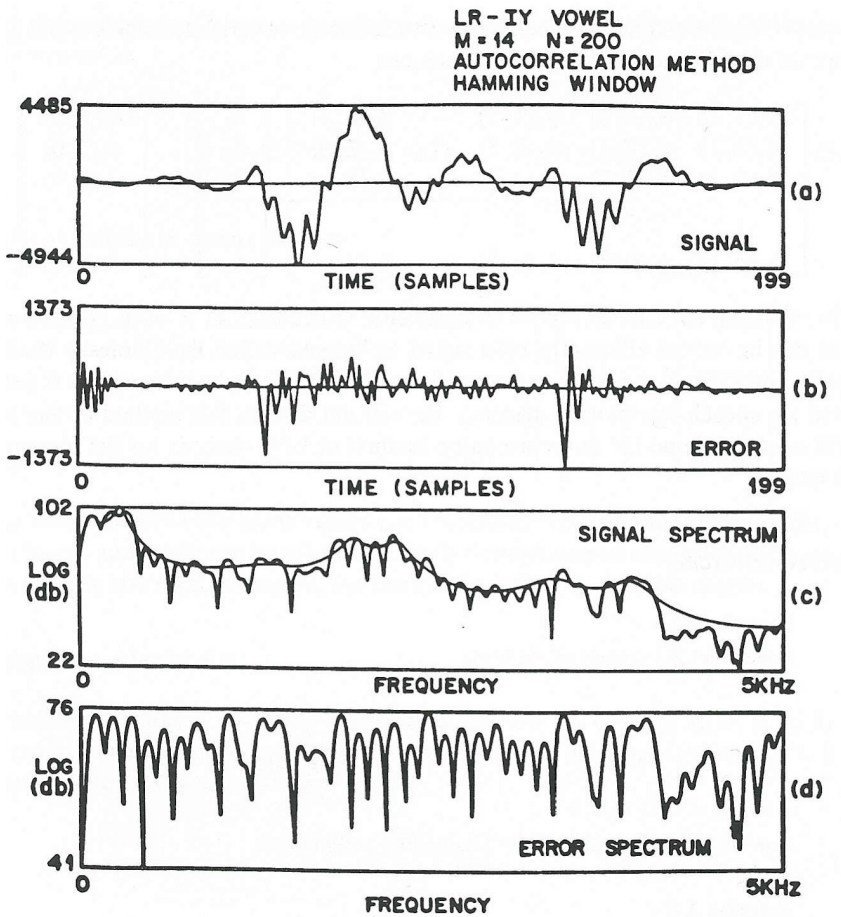


Figure 3.32 Typical signals and spectra for LPC autocorrelation method for a segment of speech spoken by a male speaker (after Rabiner et al. [8]).

- Does the prediction error look like single impulse?
- Error spectrum is *whitened* relative to original spectrum.

Example: Increasing the Model Order

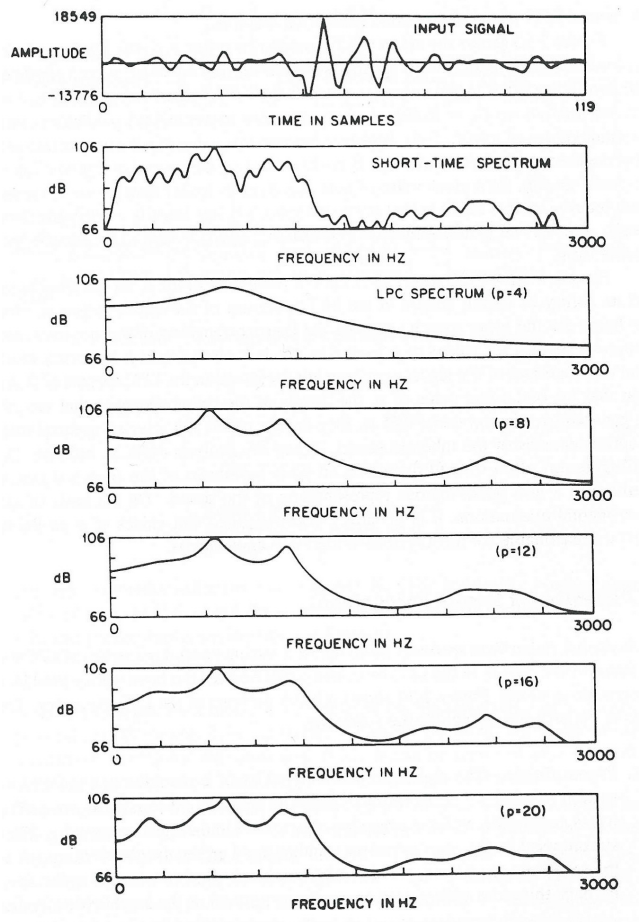


Figure 3.36 Spectra for a vowel sound for several values of predictor order, p.

- As p increases, LPC spectrum approaches original. (Why?)
- Rule of thumb: set p to (sampling rate)/1kHz + 2–4.
  - *e.g.*, for 10 KHz, use $p = 12$ or $p = 14$.

Are *a*[*j*] Good Features for ASR?

- Nope.
  - Have enormous dynamic range and are very sensitive to input signal frequencies.
  - Are highly intercorrelated in nonlinear fashion.
- Can we derive good features from LP coefficients?
  - Use LPC spectrum? Not compact.
  - Transformation that works best is LPC *cepstrum*.

The LPC Cepstrum

- The *complex cepstrum* $\tilde{h}[n]$ is the inverse DFT of ...
  - The logarithm of the spectrum.

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

- Using Z-Transform notation:

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

- Substituting in $H(z)$ for a LPC filter:

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

The LPC Cepstrum (cont'd)

- After some math, we get:

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

  - *i.e.*, given $a[j]$, easy to compute LPC cepstrum.
- In practice, 12–20 cepstrum coefficients are adequate for ASR (depending upon the sampling rate and whether you are doing LPC or PLP).

What Goes In, What Comes Out

- For each frame, output 12–20 feature values . . .
  - Which characterize what happened during that frame.
- *e.g.*, for 1s sample at 16 kHz; 10ms frame rate.
  - Input: 16000$\times$1 values.
  - Output: 100$\times$12 values.
- For MFCC, PLP, use similar number of cepstral coefficients.
- We'll say how to get to $\sim$40-dim feature vector in a bit.

Recap: Linear Predictive Coding Cepstrum

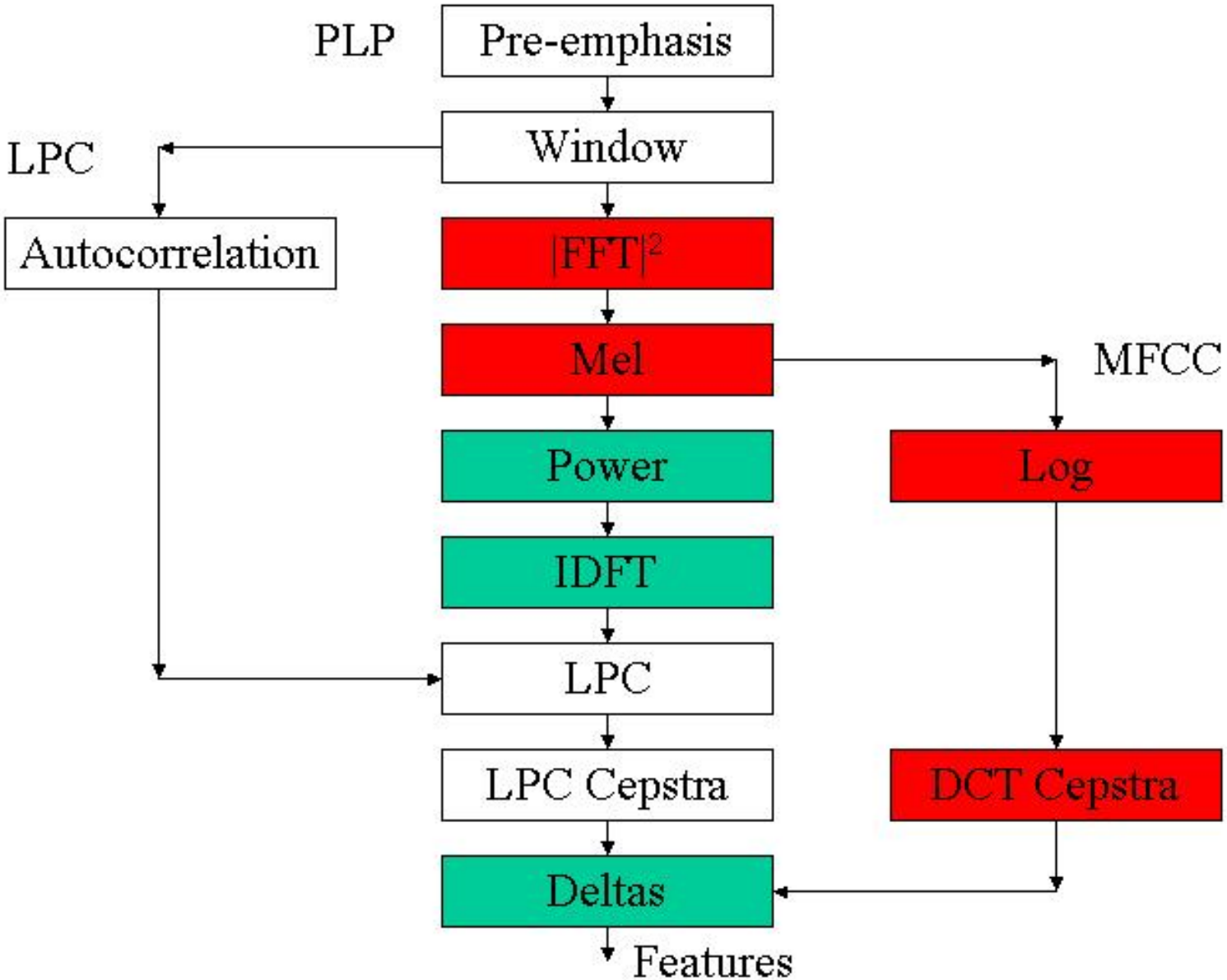
- Motivated by source-filter model of human production.
- For each frame . . .
- Step 1: Compute short-term LPC spectrum.
  - Compute autocorrelation sequence $R(i)$.
  - Compute LP coefficients $a[j]$ using Levinson-Durbin.
  - LPC spectrum is smoothed version of original:

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

- Step 2: From LPC spectrum, compute complex cepstrum.
  - Simple to compute cepstral coeffs given $a[j]$.

Where Are We?

Mel-Frequency Cepstral Coefficients



Motivation: Psychophysics

- Mel scale — frequencies equally spaced in Mel scale are equally spaced according to human perception.
- Recall human hearing is not equally sensitive to all frequency bands.

$$\text{Mel freq} = 2595 \log_{10}(1 + \text{freq}/700)$$

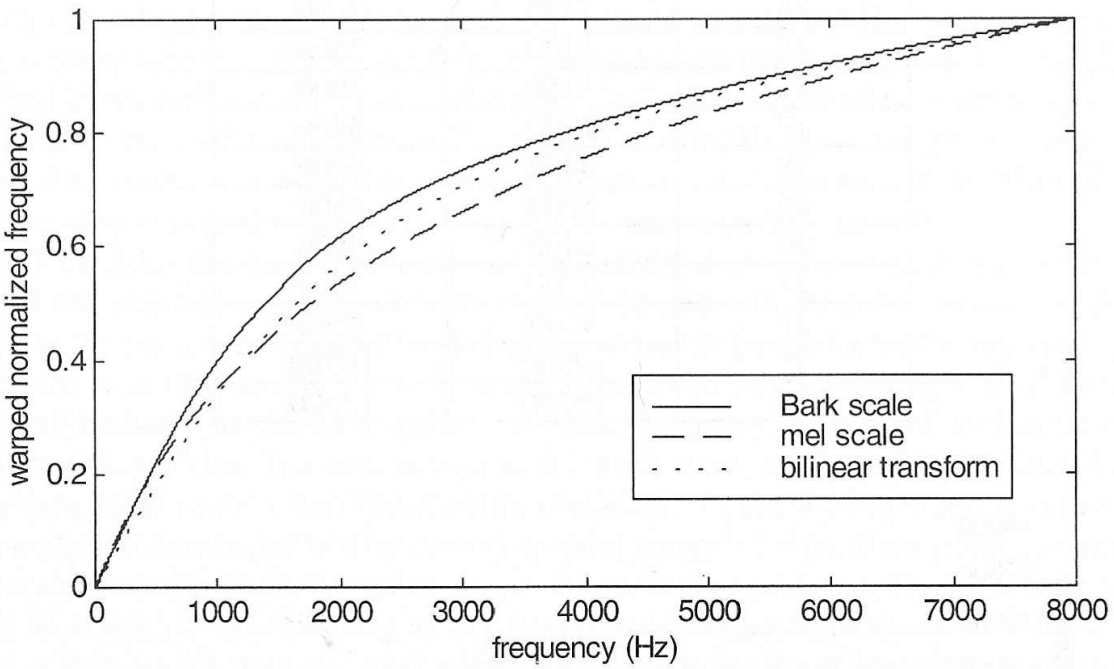


Figure 2.13 Frequency warping according to the Bark scale, ERB scale, mel-scale, and bilinear nsform for $\alpha = 0.6$: linear frequency in the x-axis and normalized frequency in the y-axis.

The Basic Idea: Mel Binning

- Goal: develop perceptually-based set of features.
- Cochlea is one big-ass *filterbank*?
  - FFT can be viewed as computing . . .
  - Instantaneous outputs of uniform filterbank.
- How to mimic frequency warping of Mel scale?
- Original spectrum $X(k)$:
  - Energy at equally-spaced frequencies.
- Created warped spectrum $S(m)$ by bucketing $X(k)$ . . .
  - Using non-uniform buckets.

Mel Binning

- Divide frequency axis into *m* triangular filters . . .
  - Spaced in equal *perceptual* increments.
  - Filters are uniformly spaced below 1 kHz and logarithmically spaced above 1 kHz.

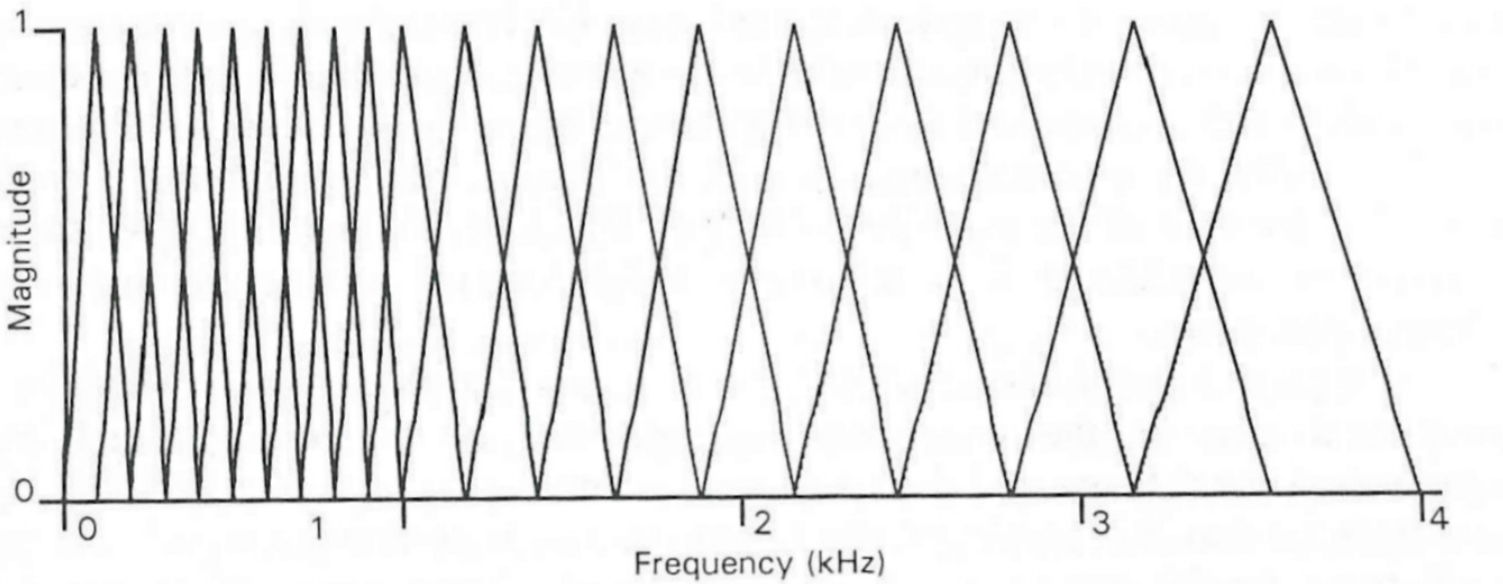


Figure 10.2 Triangular filters of the type suggested by Davis and Mermelstein (1980) for transforming the output of a Fourier transform onto a mel scale in both bandwidth and spacing.

Why Triangular Filters?

- Crude approximation to shape of tuning curves . . .
  - Of nerve fibers in auditory system.

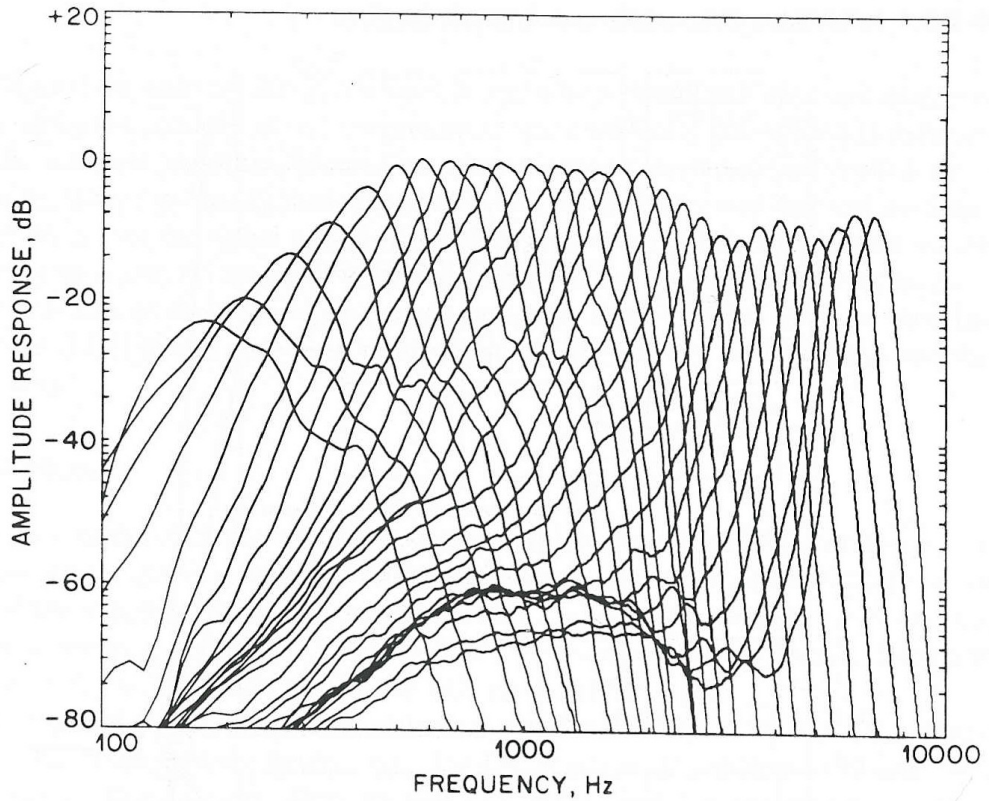


Figure 3.50 Frequency response curves of a cat's basilar membrane (after Ghitza [13]).

Equations, Please

- $H_m(k)$ = weight of energy at frequency k for mth filter.

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

- $f(m-1)/f(m)/f(m+1)$ is left boundary/middle/right boundary of mth filter.

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

  - f_l/f_h are lowest/highest frequencies of filterbank.
  - F_S is sampling frequency; M is number of filters.
  - N is length of FFT; B is Mel scale:

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

Equations (cont'd)

- Output of mth filter $S(m)$:

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

  - $X_m(k) =$ N-Point FFT of $x^m[n]$, the mth window frame.
- N is chosen as smallest power of two ...
  - Greater or equal to window length.
  - Rest of input to FFT padded with zeros.

The Mel Frequency Cepstrum

- (Real) cepstrum is inverse DFT of log magnitude of spectrum.
  - Log magnitude spectrum is symmetric so . . .
  - DFT simplifies to *discrete cosine transform* (DCT).
- Why log energy?
  - Logarithm compresses dynamic range of values.
  - Human response to signal level is logarithmic.
  - Less sensitive to slight variations in input level.
  - Phase information not helpful in speech.
- The *Mel cepstrum* is DCT of filter outputs $S(m)$:

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

The Discrete Cosine Transform

- DCT $\Leftrightarrow$ DFT of symmetrized signal.
  - There are many ways of creating this symmetry.

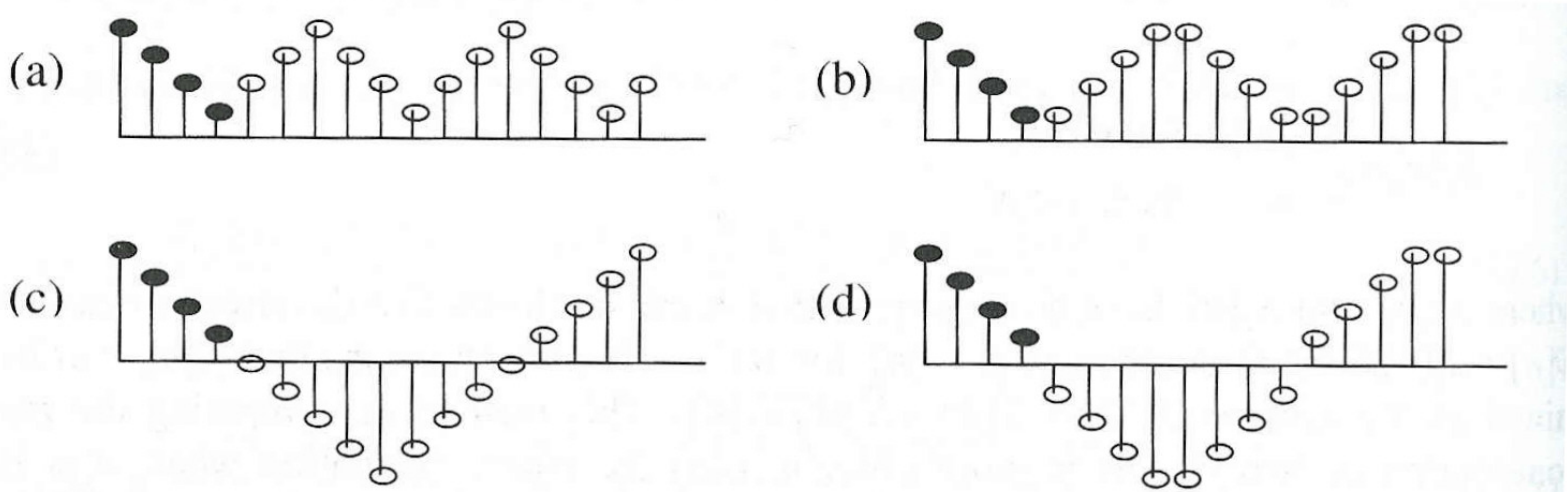


Figure 5.17 Four ways to extend a four-point sequence $x[n]$ to make it both periodic and have even symmetry. The figures in (a), (b), (c) and (d) correspond to the DCT-I, DCT-II, DCT-III and DCT-IV respectively.

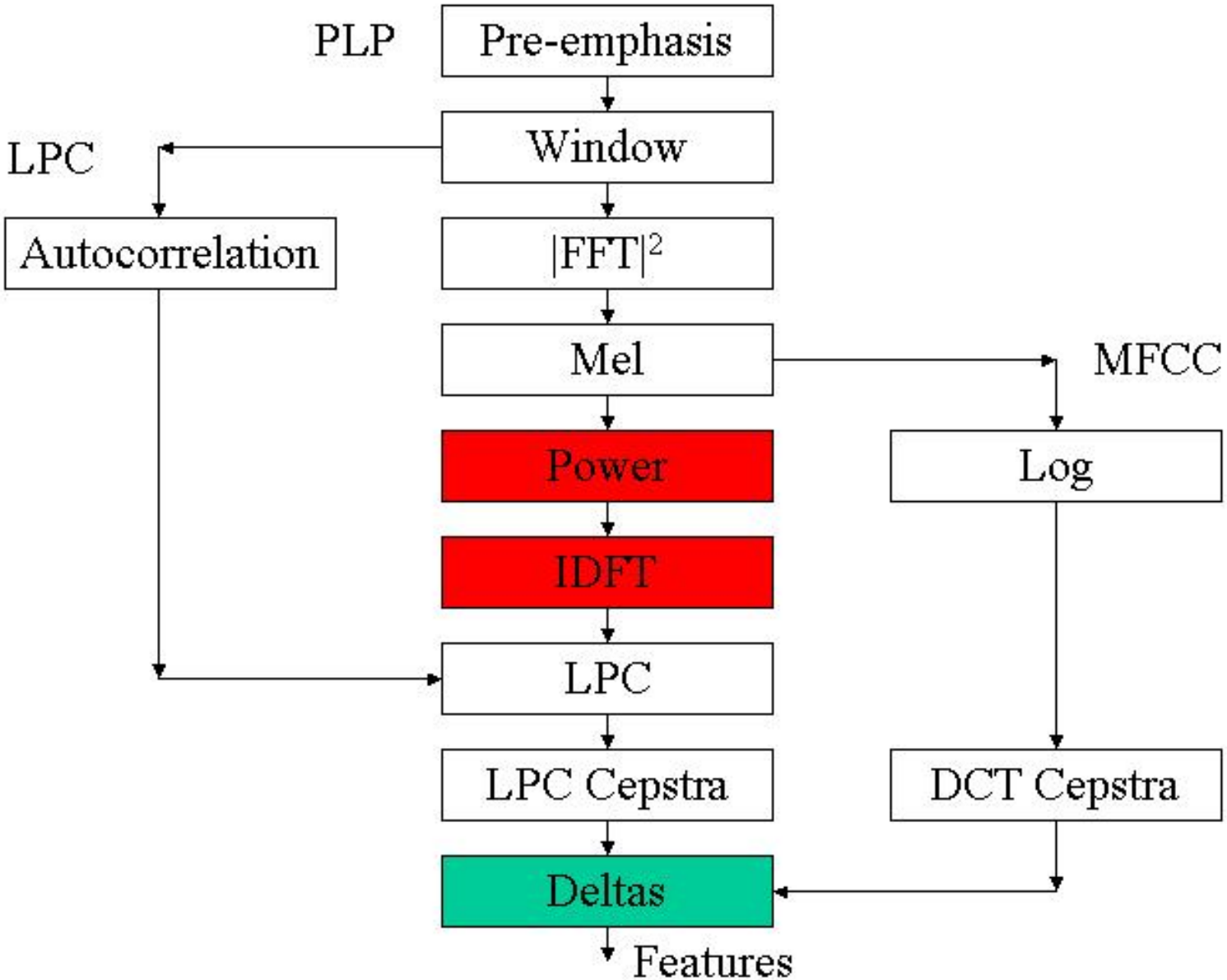
- DCT-II has better *energy compaction*.
  - Less of discontinuity at boundary.
  - Energy concentrated at lower frequencies.
  - Can represent signal with fewer DCT coefficients.

Mel Frequency Cepstral Coefficients

- Motivated by human perception.
- For each frame . . .
- Step 1: Compute frequency-warped spectrum $S(m)$.
  - Take original spectrum and apply Mel binning.
- Step 2: Compute cepstrum using DCT . . .
  - From warped spectrum $S(m)$.

Where Are We?

Perceptual Linear Prediction



Practical Perceptual Linear Prediction [2]

- Merges best features of Linear Prediction and MFCC's.
- Start out like MFCC: apply Mel binning to spectrum.
  - When compute output of mth filter $S(m)$ ...
  - Take cube root of power instead of logarithm:

$$S(m) = \left(\sum_{k=0}^{N-1} |X_m(k)|^2 H_m(k)\right)^{\frac{1}{3}}$$

- Take IDFT of symmetrized version of $S(m)$ (will be real):

$$R(m) = \text{IDFT}([S(:), S(M-1:-1:2)])$$

  - Pretend $R(m)$ are autocorrelation coeffs of real signal.
  - Given $R(m)$, compute LPC coefficients and cepstra ...
  - As in "normal" LPC processing.

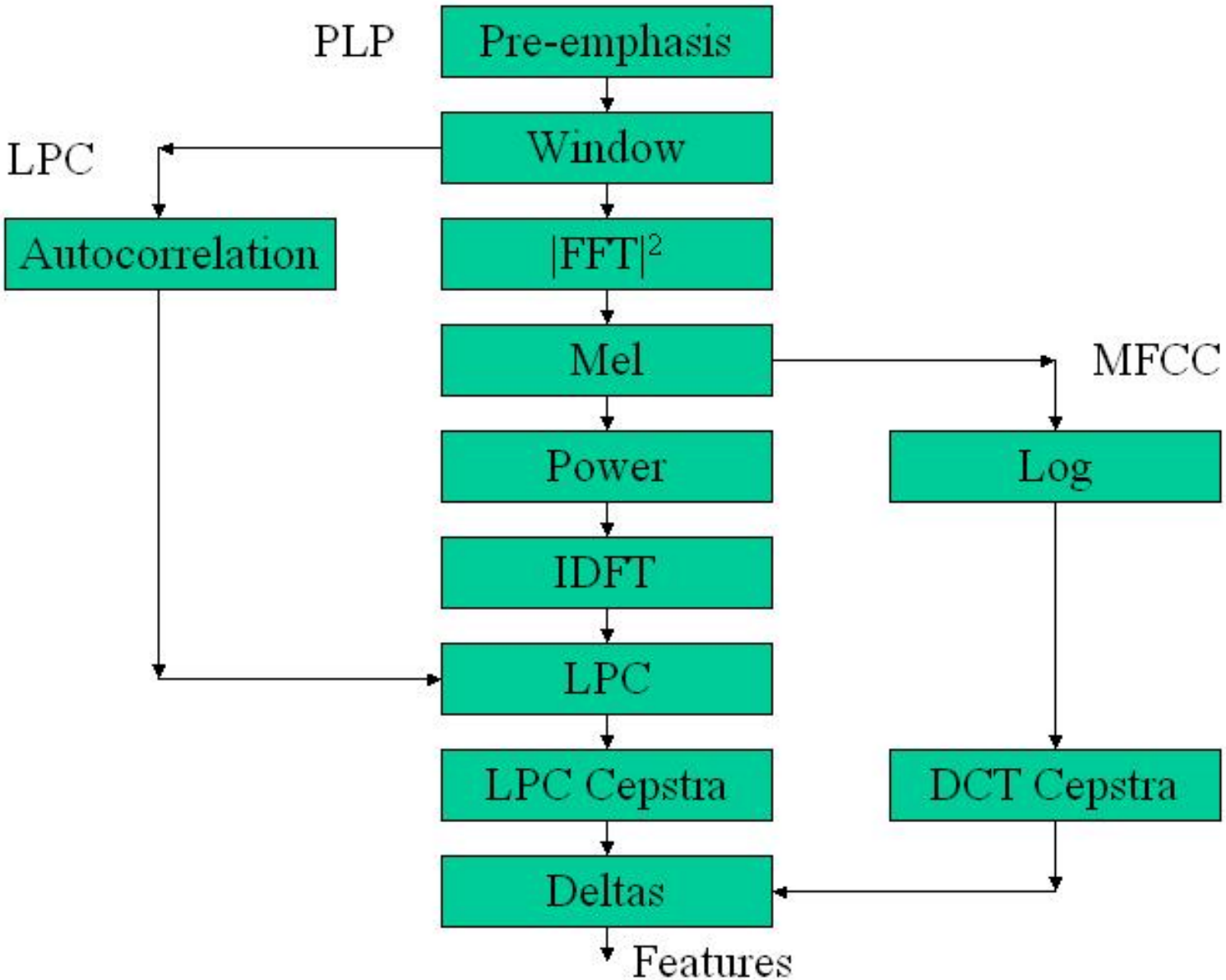
Recap: Perceptual Linear Prediction

- Smooth spectral fit that matches higher amplitude components better than lower amplitude components (LP).
- Perceptually-based frequency scale (Mel binning).
- Perceptually-based amplitude scale (cube root).

- For each frame ...
- Step 1: Compute frequency-warped spectrum $S(m)$.
  - Take original spectrum and apply Mel binning.
  - Use cube root of power instead of logarithm.
- Step 2: Compute LPC cepstrum from ...
  - Fake autocorrelation coeffs produced by IDFT of $S(m)$.

Where Are We?

Heads or Tail?



Pre-Emphasis

- Compensate for 6dB/octave falloff due to . . .
  - Glottal-source and lip-radiation combination.
- Implement pre-emphasis by transforming audio signal $x[n]$ to $y[n]$ via simple filter:

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

- How does this affect signal?
  - Filtering $\Leftrightarrow$ convolution in time domain $\Leftrightarrow$ multiplication in frequency domain.
- Taking the Z-Transform:

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

Substituting $z = e^{j\omega}$, we get:

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

Pre-Emphasis (cont'd)

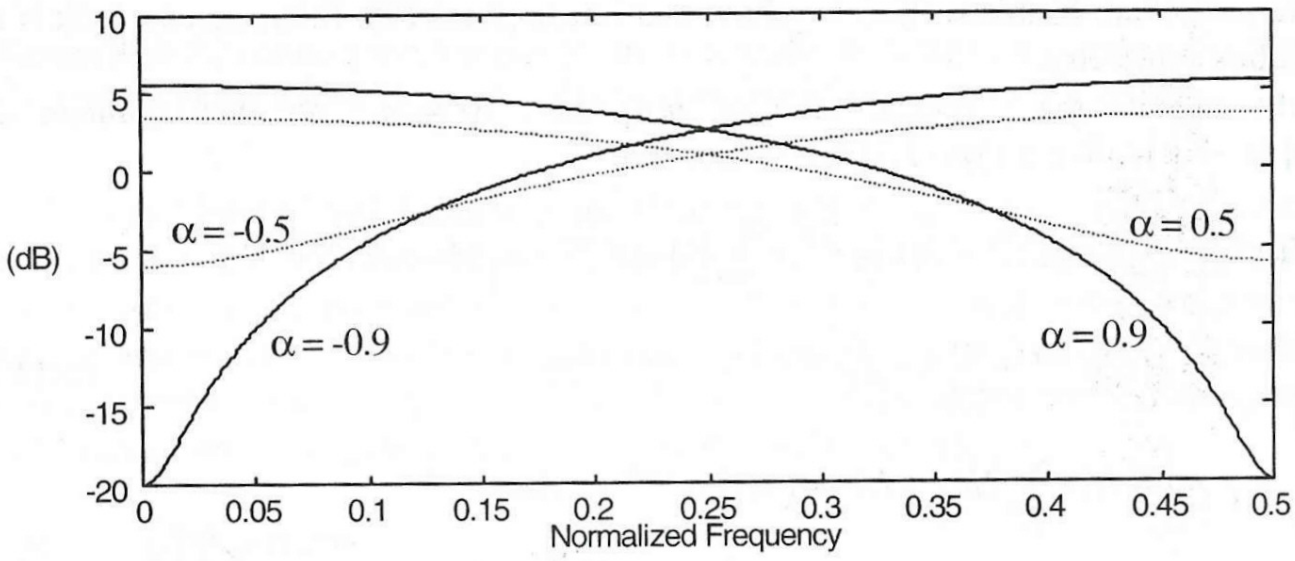


Figure 5.21 Frequency response of the first order FIR filter for various values of α.

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

- For $a < 0$ we have high-pass filter.
  - a.k.a. *pre-emphasis* filter as frequency response rises smoothly from low to high frequencies.

Properties

- Improves LPC estimates (works better with “flatter” spectra).
- Reduces or eliminates “DC” (constant) offsets.
- Mimics equal-loudness contours.
  - Higher frequency sounds appear “louder” than low frequency sounds given same amplitude.

Deltas and Double Deltas

- Story so far: use 12–20 cepstral coeffs as features to . . .
  - Describe what happened in current 10–20 msec window.
- Problem: dynamic characteristics of sounds are important!
  - *e.g.*, stop closures and releases; formant transitions.
  - *e.g.*, phenomena that are longer than 20 msec.
- One idea: directly model the trajectories of features.
- Simpler idea: *deltas* and *double deltas*.

The Basic Idea

- Augment original "static" feature vector . . .
  - With 1st and 2nd derivatives of each value w.r.t. time.
- Deltas: if y_t is feature vector at time t, take:

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

  and create new feature vector

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

- Doubles size of feature vector.
- D is usually one or two frames.
- Simple, but can help significantly.

Refinements

- Improve estimate of 1st derivative using linear regression.
  - *e.g.*, a five-point derivative estimate:

$$\Delta y_t = \sum_{\tau=1}^{D} \tau \frac{(y_{t+\tau} - y_{t-\tau})}{2\sum_{\tau=1}^{D} \tau^2}$$

- Double deltas: can estimate derivative of first derivative . . .
  - To get second derivatives.
- If start with 13 cepstral coefficients, . . .
  - Adding deltas, double deltas gets us to $13 \times 3 = 39$ dim feature vectors.

Where Are We?

Did We Satisfy Our Original Goals?

- Capture essential information for word identification.
- Make it easy to factor out irrelevant information.
  - *e.g.*, long-term channel transmission characteristics.
- Compress information into manageable form.
- Discuss.

What Feature Representation Works Best?

- Literature on front ends is weak?
- Good early paper by Davis and Mermelstein [1]:
  - 52 different CVC words.
  - 2 (!) male speakers.
  - 676 tokens in all.
- Compared these methods for generating feature vectors:
  - MFCC.
  - LFCC.
  - LPCC.
  - LPC+Itakura metric.
  - LPC Reflection coefficients.

What Feature Representation Works Best?

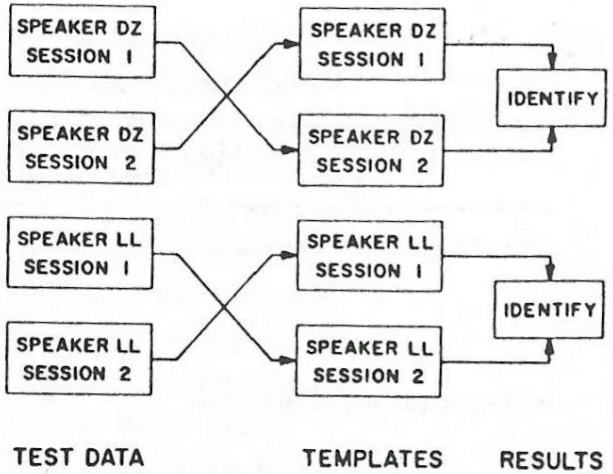


Fig. 7. Two-way speaker-dependent identification tests.

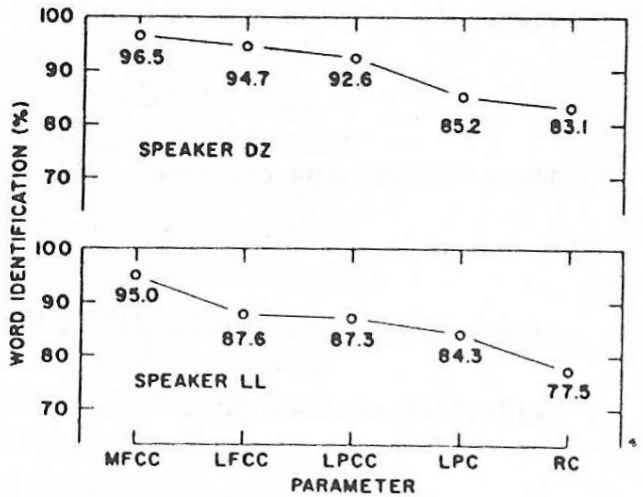


Fig. 8. Performance of parametric representations for recognition.

- Also, 6.4 msec frame rate slightly better than 12.8 msec (but lots more computation).

How Things Stand Today

- No one uses LPC cepstra any more?
- Experiments comparing PLP and MFCC are mixed.
  - Which is better may depend on task.
  - General belief: PLP is usually slightly better.
  - It's always safe to use MFCC.
  - (Can get some improvement by combining systems?)

Who Made This Stuff Up?

- No offense, but this all seems like a big hack.
- The vocal tract, like the Internet, is series of tubes!? (LPC)
- “Quefrencies” are best way to identify sounds?
  - Is there intuitive interpretation of quefrency . . .
  - Computed from Mel spectrum?
- That PLP pipeline is one ugly duck.

Points

- People have tried hundreds, if not thousands, of methods.
  - MFCC, PLP are what worked best (or at least as well).
- What about more data-driven/less knowledge-driven?
  - Instead of hardcoding ideas from speech production/perception . . .
  - Try to automatically learn transformation from data?
  - Hasn't helped yet.
- How much audio processing is hardwired in humans?

What About Using Even More Knowledge?

- Articulatory features.
- Neural firing rate models.
- Formant frequencies.
- Pitch (except for tonal languages such as Mandarin).
- Hasn't helped yet over dumb methods.
  - Can't guess hidden values (*e.g.*, formants) well?
  - Some features good for some sounds but not others?
  - Dumb methods automatically learn some of this?

This Isn't The Whole Story (By a Long Shot)

- MFCC/PLP are only starting point for acoustic features!
- In state-of-the-art systems, do many additional transformations on top.
  - LDA (maximize class separation).
  - Vocal tract normalization.
  - Speaker adaptive transforms.
  - Discriminative transforms.
- We'll talk about this stuff in Lecture 9 (Adaptation).

Hot Topic: Neural Network Bottleneck Feats

- Step 1: Build a “conventional” ASR system.
  - Use this to guess (CD) phone identity for each frame.
- Step 2: Use this data to train NN that guesses phone identity from (conventional) acoustic features.
  - Use NN with narrow hidden layer, *e.g.*, 40 hidden units.
  - Force NN to try to encode all relevant info about input in *bottleneck* layer.
- Use hidden unit activations in bottleneck layer as features.
  - Append to or replace original features.
- Will be covered more in Lecture 12 (Deep Belief networks).

References

- S. Davis and P. Mermelstein, “Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences”, IEEE Trans. on Acoustics, Speech, and Signal Processing, 28(4), pp. 357–366, 1980.
- H. Hermansky, “Perceptual Linear Predictive Analysis of Speech”, J. Acoust. Soc. Am., 87(4), pp. 1738–1752, 1990.
- H. Hermansky, D. Ellis and S. Sharma, “Tandem connectionist feature extraction for conventional HMM systems”, in Proc. ICASSP 2000, Istanbul, Turkey, June 2000.
- L. Deng and D. O’Shaughnessy, *Speech Processing: A Dynamic and Optimization-Oriented Approach*, Marcel Dekker Inc., 2003.

Part II

Dynamic Time Warping

A Very Simple Speech Recognizer

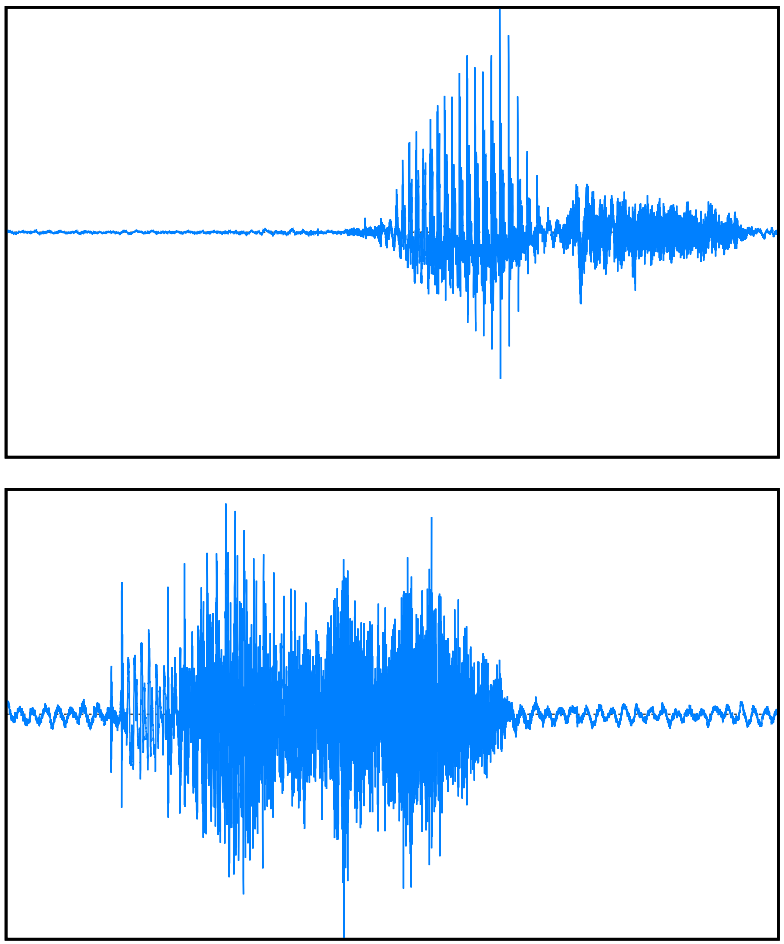
$$w^* = \arg\min_{w \in \text{vocab}} \text{distance}(A'_{\text{test}}, A'_w)$$

- *signal processing* — Extracting features A' from audio A.
  - *e.g.*, MFCC with deltas and double deltas.
  - *e.g.*, for 1s signal with 10ms frame rate $\Rightarrow \sim 100 \times 40$ values in A'.
- *dynamic time warping* — Handling time/rate variation in the distance measure.

The Problem

$$\text{distance}(A'_{\text{test}}, A'_w) \stackrel{?}{=} \sum_t \text{framedist}(A'_{\text{test},t}, A'_{w,t})$$

- In general, samples won't even be same length.



Problem Formulation

- Have two audio samples; convert to feature vectors.
  - Each x_t, y_t is $\sim$40-dim vector, say.

$$X = (x_1, x_2, \ldots, x_{T_x})$$
$$Y = (y_1, y_2, \ldots, y_{T_y})$$

- Compute distance(X, Y).

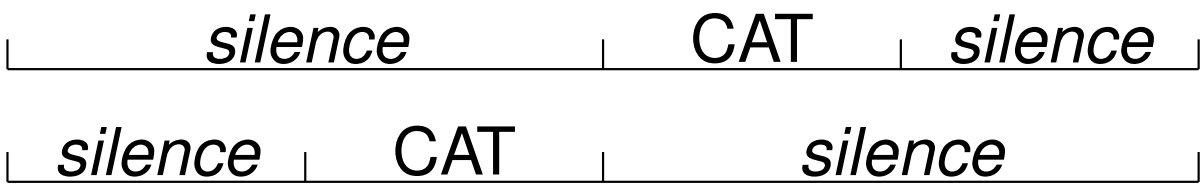
Linear Time Normalization

- Idea: omit/duplicate frames uniformly in Y . . .
  - So same length as X.

$$\text{distance}(X, Y) = \sum_{t=1}^{T_x} \text{framedist}(x_t, y_{t \times \frac{T_y}{T_x}})$$

What's the Problem?

- Handling silence.

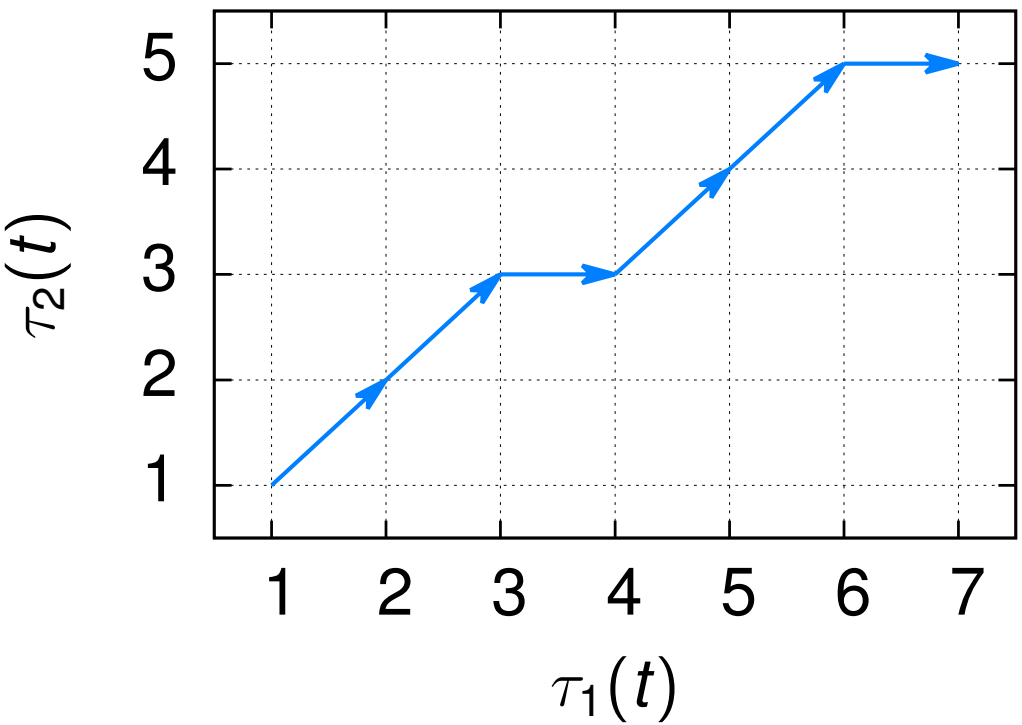


  - Solution: *endpointing*.
- Do vowels and consonants stretch equally in time?
- Want *nonlinear* alignment scheme!

Alignments and Warping Functions

- Can specify alignment between times in X and Y using ...
  - Warping functions $\tau_x(t)$, $\tau_y(t)$, $t = 1, \ldots, T$.
  - *i.e.*, time $\tau_x(t)$ in X aligns with time $\tau_y(t)$ in Y.
- Total distance is sum of distance between aligned vectors.

$$\text{distance}_{\tau_x,\tau_y}(X, Y) = \sum_{k=1}^{T} \text{framedist}(x_{\tau_x(t)}, y_{\tau_y(t)})$$



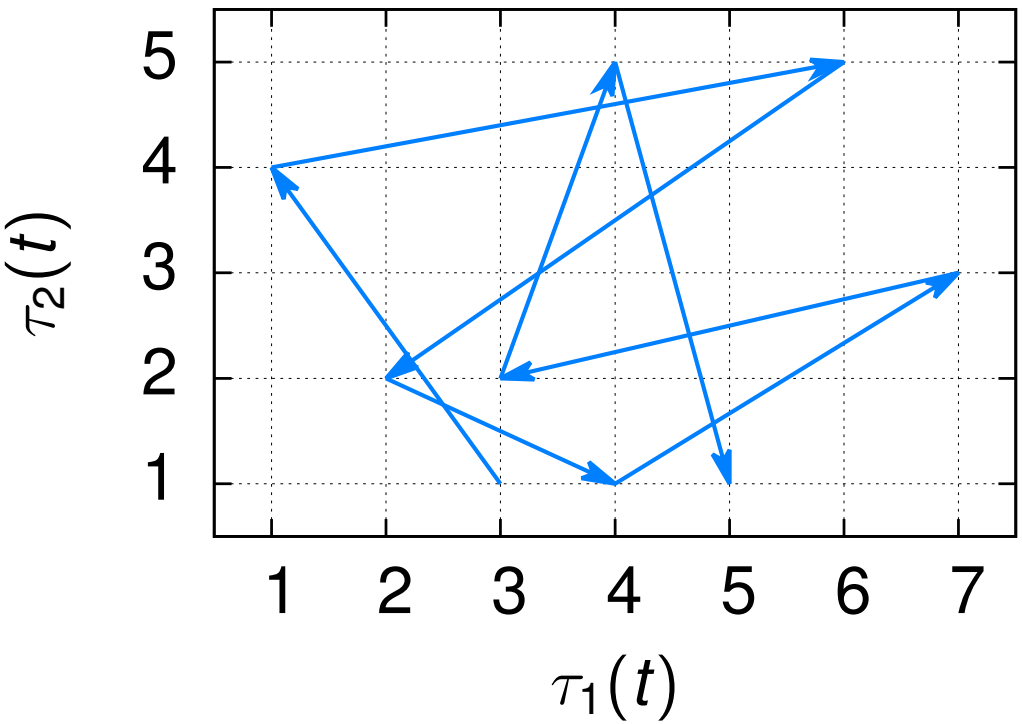
Computing Frame Distances: framedist(x, y)

- Let x_d, y_d denote dth dimension (this slide only).

Euclidean (L^2)	$\sqrt{\sum_d (x_d - y_d)^2}$
L^p norm	$\sqrt[p]{\sum_d \lvert x_d - y_d \rvert^p}$
weighted L^p norm	$\sqrt[p]{\sum_d w_d \lvert x_d - y_d \rvert^p}$
Itakura $d_I(X, Y)$	$\log(a^T R_p a / G^2)$
Symmetrized Itakura	$d_I(X, Y) + d_I(Y, X)$

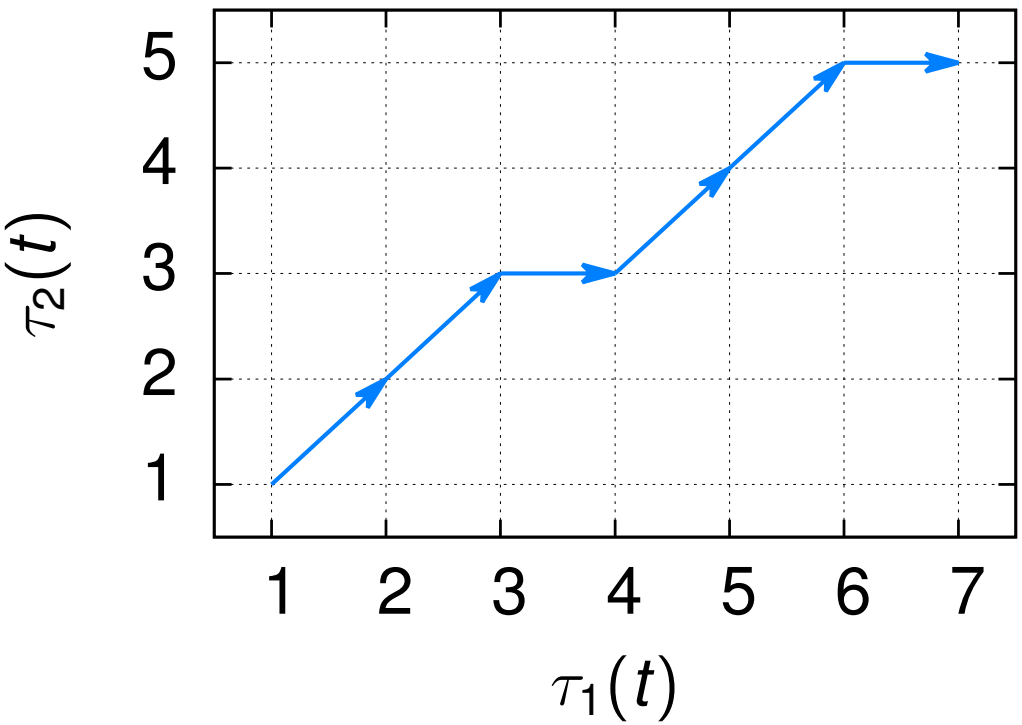
- Weighting each feature vector component differently.
  - *e.g.*, for variance normalization.
  - Called *liftering* when applied to cepstra.

Another Example Alignment



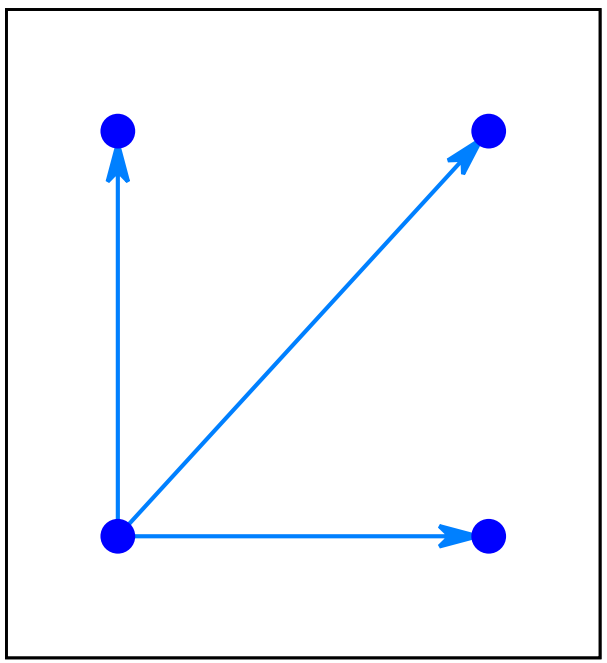
Constraining Warping Functions

- Begin at the beginning; end at the end. (Any exceptions?)
  - $\tau_x(1) = 1$, $\tau_x(T) = T_x$; $\tau_y(1) = 1$, $\tau_y(T) = T_y$.
- Don't move backwards (monotonicity).
  - $\tau_x(t+1) \geq \tau_x(t)$; $\tau_y(t+1) \geq \tau_y(t)$.
- Don't move forwards too far (locality).
  - *e.g.*, $\tau_x(t+1) \leq \tau_x(t) + 1$; $\tau_y(t+1) \leq \tau_y(t) + 1$.



Local Paths

- Can summarize/encode local alignment constraints . . .
  - By enumerating legal ways to extend an alignment.
- *e.g.*, three possible extensions or *local paths*.
  - $\tau_1(t+1) = \tau_1(t) + 1$, $\tau_2(t+1) = \tau_2(t) + 1$.
  - $\tau_1(t+1) = \tau_1(t) + 1$, $\tau_2(t+1) = \tau_2(t)$.
  - $\tau_1(t+1) = \tau_1(t)$, $\tau_2(t+1) = \tau_2(t) + 1$.



Which Alignment?

- Given alignment, easy to compute distance:

$$\text{distance}_{\tau_x,\tau_y}(X, Y) = \sum_{t=1}^{T} \text{framedist}(x_{\tau_x(t)}, y_{\tau_y(t)})$$

- Lots of possible alignments given X, Y.
  - Which one to use to calculate distance(X, Y)?
  - The best one!

$$\text{distance}(X, Y) = \min_{\text{valid } \tau_x,\, \tau_y} \left\{\text{distance}_{\tau_x,\tau_y}(X, Y)\right\}$$

Which Alignment?

$$\text{distance}(X, Y) = \min_{\text{valid } \tau_x,\, \tau_y} \left\{ \text{distance}_{\tau_x, \tau_y}(X, Y) \right\}$$

- Hey, there are many, many possible τ_x, τ_y.
  - Exponentially many in T_x and T_y, in fact.
- How in blazes are we going to compute this?

Where Are We?

Wake Up!

- One of few things you should remember from course.
  - Will be used in many lectures.
- If need to find best element in exponential space . . .
  - Rule of thumb: the answer is *dynamic programming* . . .
  - Or not solvable exactly in polynomial time.

Dynamic Programming

- Solve problem by caching subproblem solutions . . .
  - Rather than recomputing them.
- Why the mysterious name?
  - The inventor (Richard Bellman) was trying to hide . . .
  - What he was working on from his boss.
- For our purposes, we focus on *shortest path* problems.

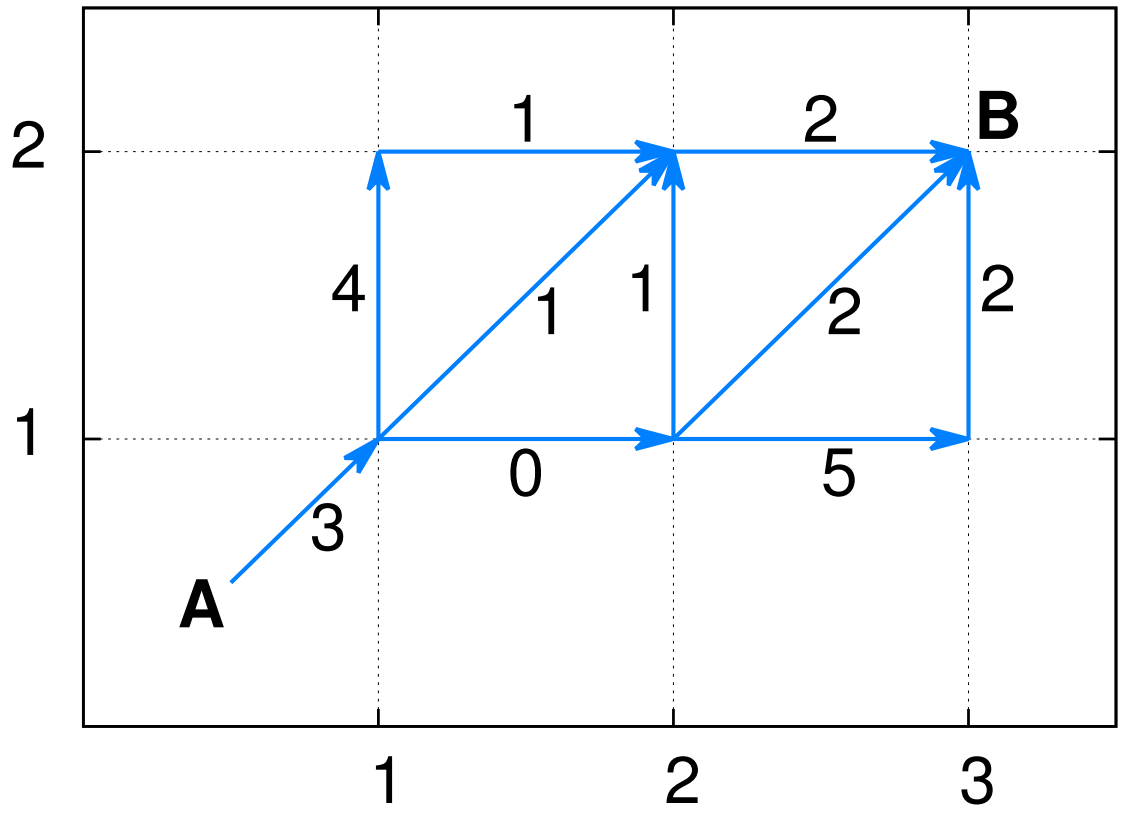
Case Study

- Let's consider small example: $T_x = 3$; $T_y = 2$.
  - Allow steps that advance t_x and t_y by at most 1.
- Matrix of frame distances: framedist($x_t, y_{t'}$).

	x_1	x_2	x_3
y_2	4	1	2
y_1	3	0	5

- Let's make a graph.
  - Label each arc with framedist($x_t, y_{t'}$) at destination.
  - Need dummy arc to get distance at starting point.

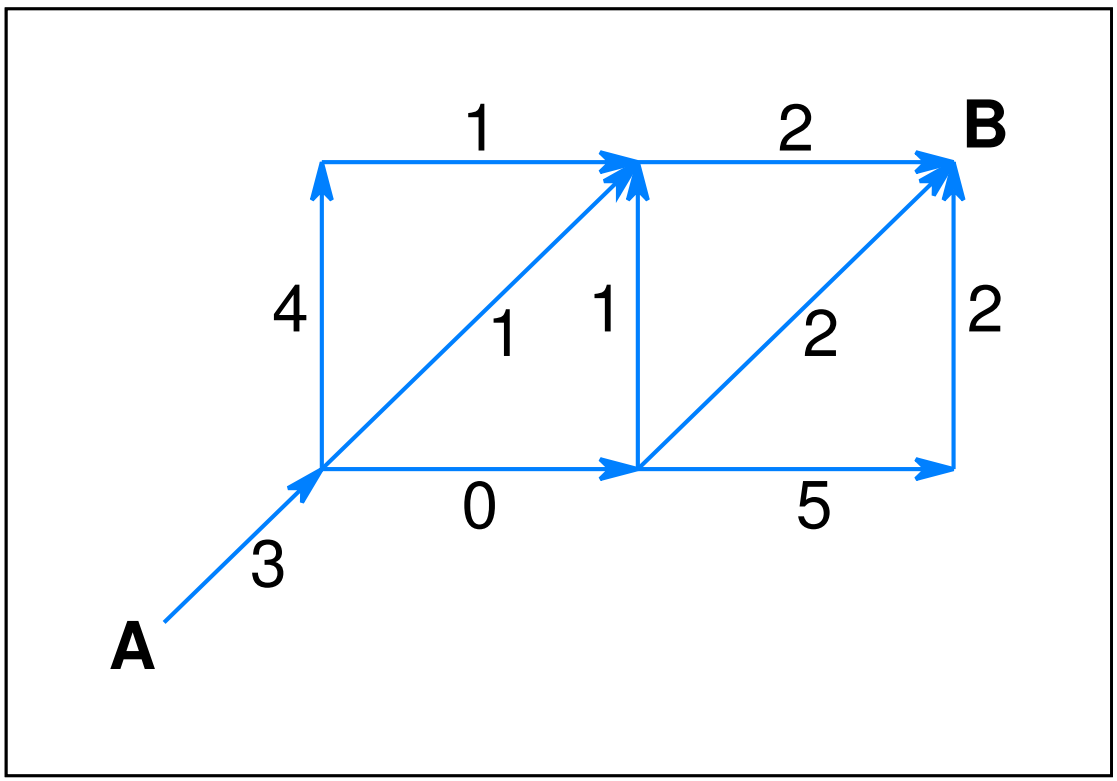
Case Study



- Each path from point A to B corresponds to alignment.
  - Distance for alignment = sum of scores along path.
  - Goal: find alignment with smallest total distance.

And Now For Something Completely Different

- Let's take a break and look at something totally unrelated.
- Here's a map with distances labeled between cities.
- Let's say we want to find the shortest route . . .
  - From point A to point B.



Wait a Second!

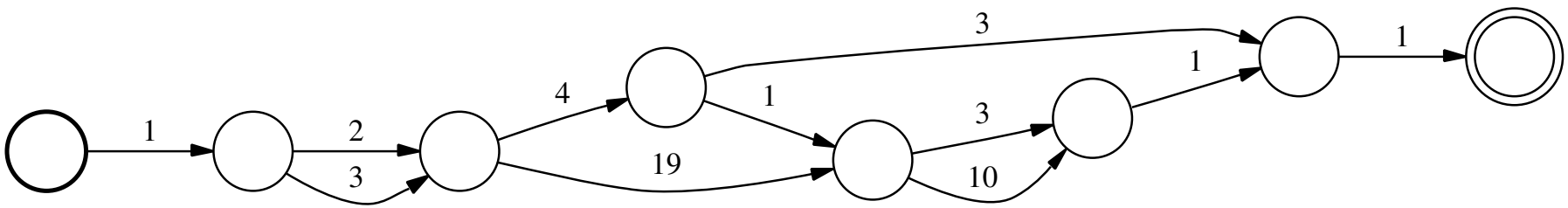
- I'm having déjà vu!
- In fact, length of shortest route in 2nd problem . . .
  - Is same as distance for best alignment in 1st problem.
- *i.e.*, problem of finding best alignment is equivalent to . . .
- *Single-pair shortest path* problem for . . .
  - *Directed acyclic graphs*.
- Has well-known dynamic programming solution.
  - Will come up again for HMM's, finite-state machines.

Make A Bee-line for Great Taste!

- On box of Honey Nut Cheerios, circa 2005:

  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.

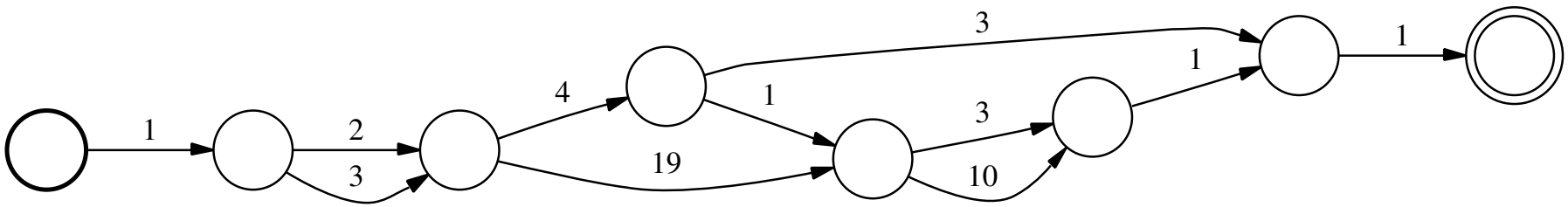
- How can we solve this baffling conundrum?
- We want *shortest* paths, so how *few* O's can you touch?



Key Observation 1

- Shortest distance $d(S)$ from start to state S is ...
  - $d(S') + \text{distance}(S', S)$ for some predecessor S' of S.
- If know $d(S')$ for all predecessors of S ...
  - Easy to compute $d(S)$.

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



Proposed Algorithm

- Loop through all states S in some order.
  - For each state, compute distance to start state $d(S)$.
  - $d(S')$ must already be known for all predecessors S'.
- Is there always an ordering on states such that . . .
  - If there is an arc $S' \rightarrow S$, . . .
  - $d(S')$ is computed before $d(S)$?

Key Observation 2

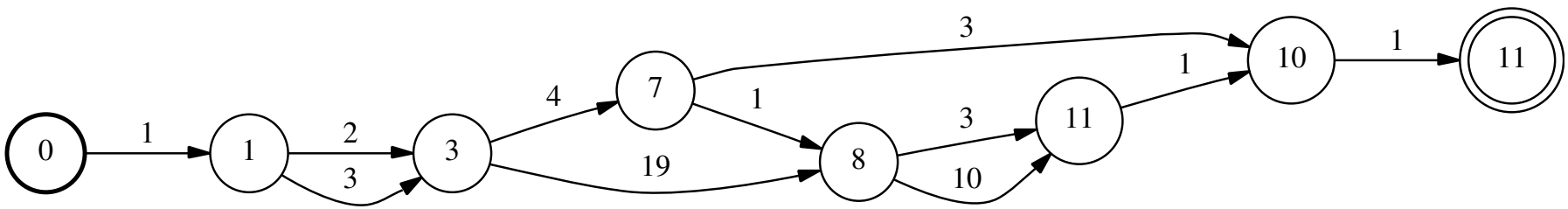
- For all acyclic graphs, there is a *topological sorting* . . .
  - Such that all arcs go from earlier to later states.
- In many cases, topological sorting is obvious by inspection.
- Otherwise, can be computed in time $O(\text{states} + \text{arcs})$.

The Shortest Path Algorithm

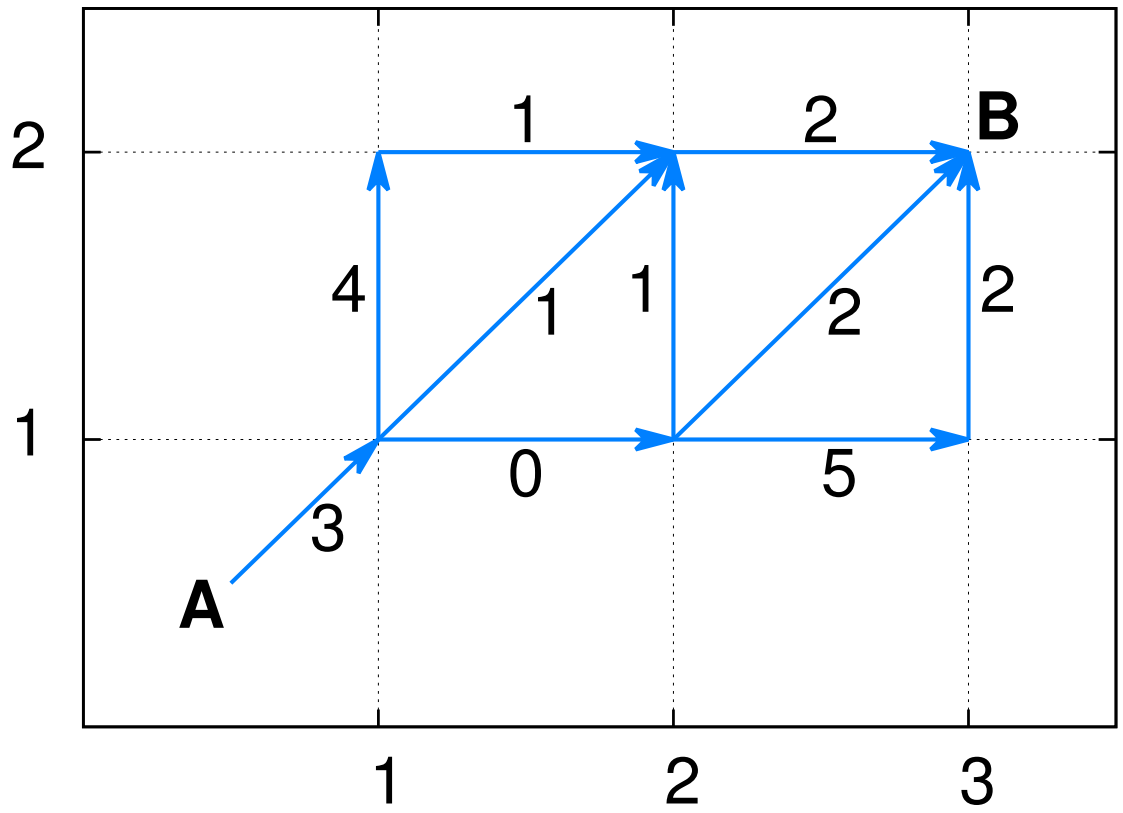
- Sort states topologically: number from 1, . . . , N.
  - Start state = state 1; final start = state N.
- $d(1) = 0$.
- For $S = 2, \ldots, N$ do

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

- Final answer: $d(N)$.
- Total time: $O(\text{states} + \text{arcs})$.



Shortest Path and DTW



- What are the states of the graph?
- What is a topological sorting for the states?
- What are the predecessors for each state?

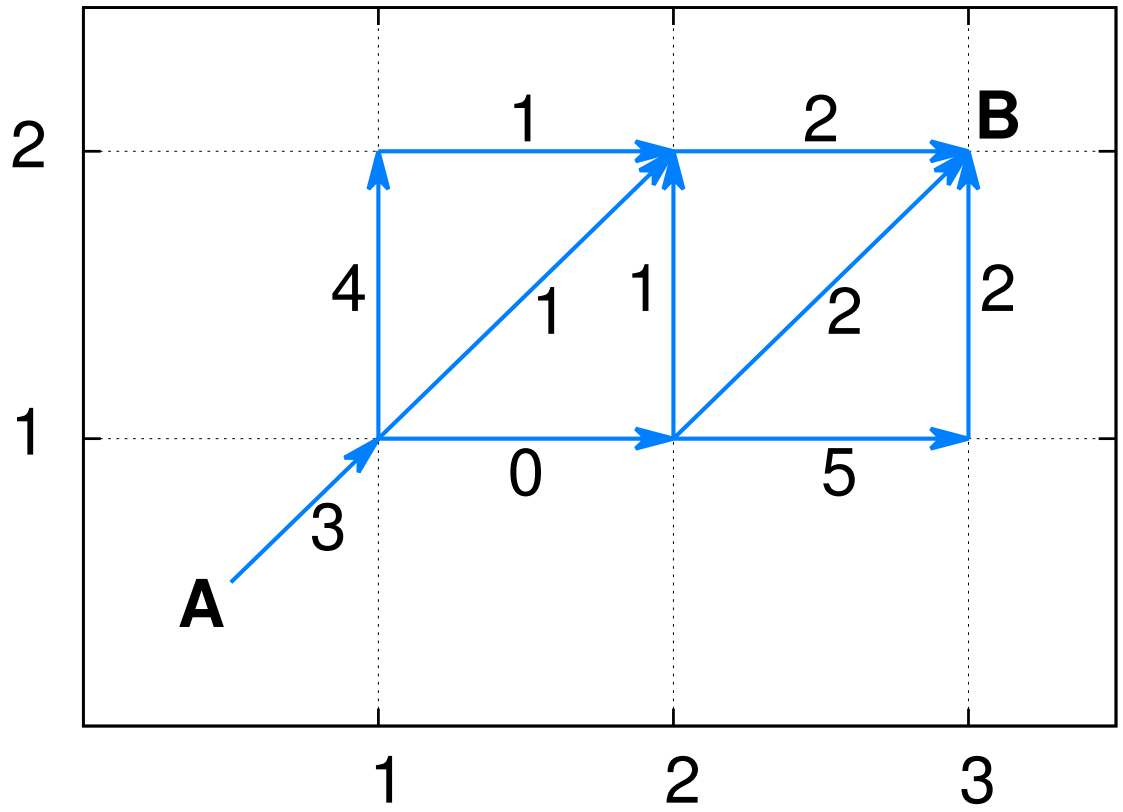
Shortest Path and DTW

- $d(1,1) = \text{framedist}(x_1, y_1)$.
- For $t_1 = 1, \ldots, 3$ do.
  - For $t_2 = 1, \ldots, 2$ do.

$$
\begin{aligned}
d(t_1, t_2) = \min\{ & \\
& d(t_1 - 1, t_2 - 1) + \text{framedist}(x_{t_1}, y_{t_2}), \\
& d(t_1 - 1, t_2) + \text{framedist}(x_{t_1}, y_{t_2}), \\
& d(t_1, t_2 - 1) + \text{framedist}(x_{t_1}, y_{t_2}) \ \}
\end{aligned}
$$

- Final answer: $d(3,2)$.

DTW: Example

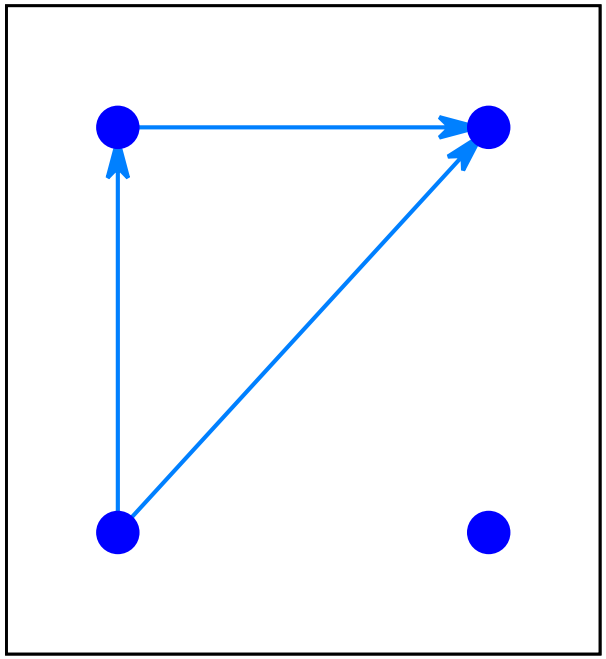


	x_1	x_2	x_3
y_2	4	1	2
y_1	3	0	5

$d(\cdot,\cdot)$	1	2	3
2	7	4	**5**
1	3	3	8

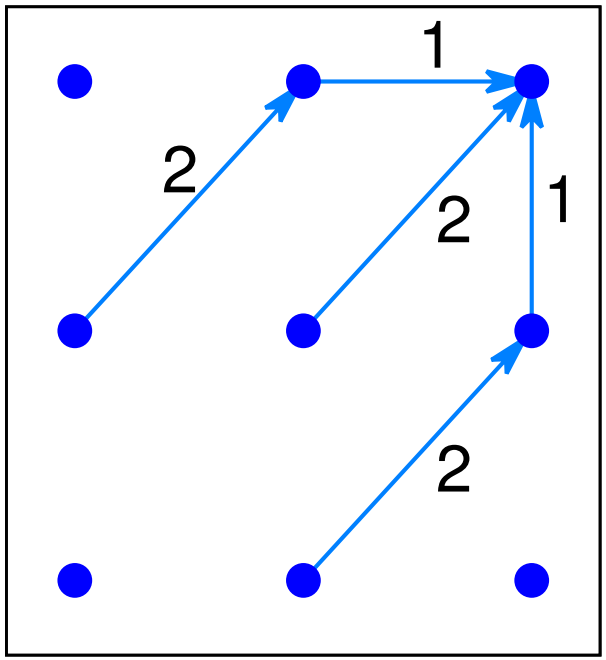
Where Are We?

Normalizing Distances



- Two-step path incurs two frame distances, while . . .
  - One-step path incurs single frame distance.
  - Biases against two-step path.
- Idea: use weights to correct for these types of biases.

Weighted Local Paths (Sakoe and Chiba)



$$
\begin{aligned}
d(t_1, t_2) = \min\{ & \\
& d(t_1 - 1, t_2 - 1) + 2 \times \text{framedist}(x_{t_1}, y_{t_2}), \\
& d(t_1 - 2, t_2 - 1) + 2 \times \text{framedist}(x_{t_1 - 1}, y_{t_2}) + \\
& \qquad \text{framedist}(x_{t_1}, y_{t_2}), \\
& d(t_1 - 1, t_2 - 2) + 2 \times \text{framedist}(x_{t_1}, y_{t_2 - 1}) + \\
& \qquad \text{framedist}(x_{t_1}, y_{t_2}) \ \}
\end{aligned}
$$

Speeding Things Up

- What is time complexity of DTW?

$$O(\text{states} + \text{arcs})$$

- For long utterances, can be expensive.
- Idea: put ceiling on maximum amount of warping, *e.g.*,

$$|\tau_x(t) - \tau_y(t)| \leq T_0$$

- Another idea: beam pruning or rank pruning.
- Can make computation linear.

A DTW Recognizer: The Whole Damn Thing

$$w^* = \arg\min_{w \in \text{vocab}} \text{distance}(A'_{\text{test}}, A'_w)$$

- Training: collect audio A_w for each word w in vocab.
  - Apply signal processing $\Rightarrow A'_w$.
  - Called *template* for word w.
  - (Can collect multiple templates for each word.)
- Test time: given audio A_{test}, convert to A'_{test}.
  - For each w, compute distance(A'_{test}, A'_w) using DTW.
  - Return w with smallest distance.

Recap

- DTW is effective for computing distance between signals . . .
  - Using nonlinear time alignment.
- *Ad hoc* selection of frame distance and local paths.
- Finds best path in exponential space in quadratic time . . .
  - Using dynamic programming.
- Signal processing, DTW: all you need for simple ASR.
  - *e.g.*, old-school cell phone name dialer.

Back to the Future

- Some recent work has revisited DTW.
  - Can extend algorithm to connected speech.
- Word models (as opposed to *phonetic* models).
- Don't average word instances together; keep separate!
- Can model longer-distance acoustic dependencies . . .
  - As compared to conventional GMM/HMM systems.
- Hasn't gone anywhere yet.

References

- R. Bellman, “Eye of the Hurricane”. World Scientific Publishing Company, Singapore, 1984.
- H. Sakoe and S. Chiba, “Dynamic Programming Algorithm Optimization for Spoken Word Recognition”, IEEE Trans. on Acoustics Speech and Signal Processing, vol. ASSP-26, pp. 43–45, Feb. 1978.

Part III

Epilogue

Lab 1

- Handed out today; due Wednesday, Oct. 3, 6pm.
- Write parts of DTW recognizer and run experiments.
  - Write most of MFCC front end (except for FFT).
  - Optional: write DTW.
  - Optional: try to improve baseline MFCC front end.

The Road Ahead

- DTW doesn't scale: linear in number of templates.
  - 10 word voc $\Rightarrow$ 100k words?
  - 1 training sample per word $\Rightarrow$ 100?
- How to deal?
  - Lecture 3: Gaussian mixture models.
  - Lecture 4: Hidden Markov models.

Course Feedback

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