

TTIC 31110

Speech Technologies

April 16, 2020

Announcements

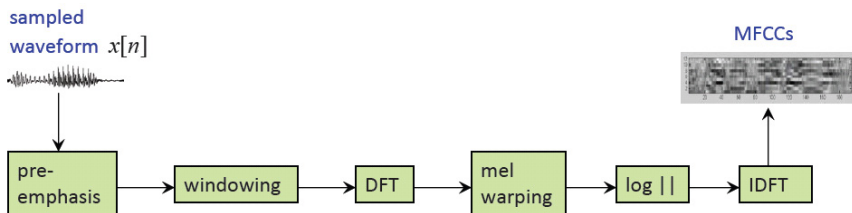
- Tutorial 4/20 3:30pm (machine learning concepts)
- HW1 due Friday 4/17 7pm
- A few HW1 submissions already, so far so good
- Note: Feel free to use external resources, but all the needed information should be in the lecture notes and/or assigned readings (let me know if not!)
- HW2 released, due Monday 4/27 7pm

Outline

Signal processing and acoustic features wrap-up

Dynamic time warping (DTW)

MFCCs



Questions/clarifications from last time

- For each frame of length N samples starting at sample t , the spectrum is computed using the discrete Fourier transform (DFT):

$$X[k] = \sum_{n=t}^{t+N-1} x[n]e^{-j2\pi kn/N}, \quad k = 0, \dots, N-1$$

- $X[k]$ is in general complex-valued, but we will only use its magnitude
- $X[k]$ is the value of the spectrum at the k^{th} frequency
- The term “spectrum” refers to any frequency-domain representation of a signal (which can include DFT, Fourier series, discrete-time Fourier transform, etc.); we will only use DFT
- Equivalently, we can consider $k = -N/2, \dots, 0, \dots, N/2$
- The choice of N values of k is arbitrary and different conventions exist

Questions/clarifications from last time

- For each frame of length N samples starting at sample t , the spectrum is computed using the discrete Fourier transform (DFT):

$$X[k] = \sum_{n=t}^{t+N-1} x[n]e^{-j2\pi kn/N}, \quad k = 0, \dots, N-1$$

- The complex sinusoids of length N are a basis for all signals of length N
- $X[k]$ is the projection of $x[n]$ onto the k^{th} basis function
- Inverse DFT makes this explicit:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X_{\text{mel}}[k]e^{j2\pi kn/N}, \quad n = 0, \dots, N-1$$

- Many other choices of basis are possible; the DFT corresponds to a particular choice

Questions/clarifications from last time

What does zero-padding in FFT do to the spectrum?

$$X[k] = \sum_{n=t}^{t+N-1} x[n]e^{-j2\pi kn/N}, \quad k = 0, \dots, N-1$$

- The DFT can be viewed as samples of the *continuous* spectrum given by the *discrete-time Fourier transform (DTFT)*:

$$X(\omega) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}, \text{ where } \omega \text{ is now any frequency}$$

- When using FFT algorithm to compute DFT, we pad the signal with zeros to length $M = 2^m$ for some m and compute length- M DFT
- It turns out: Zero padding in the time domain results in “ideal” interpolation in the frequency domain
- Ideal interpolation = sampling the DTFT at M samples instead of the original N
- Roughly speaking, this is “doing the right thing”

Questions/clarifications from last time:

Periodic signals

- Spectra of periodic signals with period λ seconds consist of delta functions at multiples of the fundamental frequency $\frac{1}{\lambda}$ Hz
- What do we mean by “periodic” or “delta function” for finite-length sequences?
- Answer:
 - Consider the periodic extension of the finite-length signal
 - By “periodic” we mean there is some periodicity with period $< N$
 - If so, the DFT will be zero except at multiples of the signal's fundamental frequency
 - In practice, we see an approximate version of this, because the window often contains a *non-integer* number of periods of the original signal, and the signal is often not *exactly* periodic

Questions/clarifications from last time:

Periodic signals

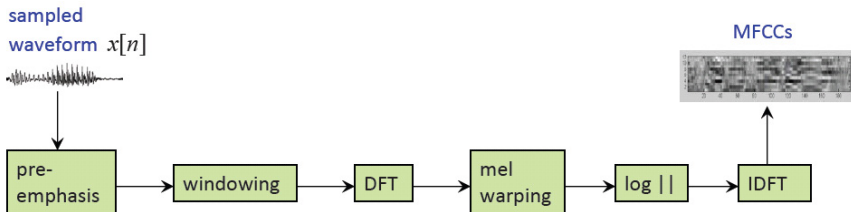
Terminology notes

- For those who are familiar with Fourier series...
- The DFT is identical to the Fourier series of the periodic extension of the signal
- For those who are familiar with the *short-time Fourier transform*... it is \approx spectrogram
- “Spectrogram” is sometimes used to refer to the magnitude only

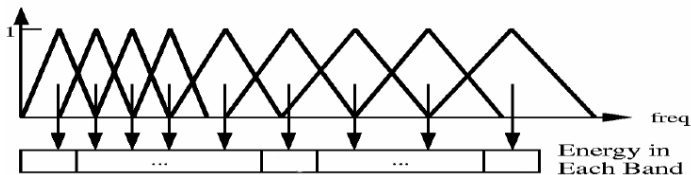
Signal processing demos

- <http://www.falstad.com/fourier/>
- <http://www.falstad.com/dfilter/>

Note: Mel warping filters



The mel warping step is typically done using a set of 40-80 filters that simulate the responses of the cochlear hair cells



(Fig. from <https://social.technet.microsoft.com>)

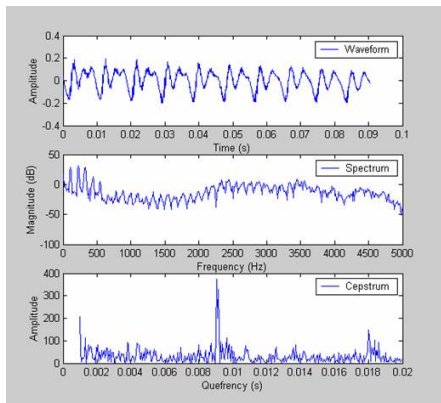
Inverse DFT

- Finally, we take the inverse DFT (IDFT) to obtain the

$$\text{cepstrum: } c[n] = \frac{1}{N} \sum_{k=0}^{N-1} X_{mel}[k] e^{j2\pi kn/N}, \quad n = 0, \dots, N-1$$

- n is called “quefrency”
- The log spectrum magnitude was real and symmetric, so this IDFT will also be real and symmetric, and is therefore equivalent to taking another DFT
- (Some people use the discrete *cosine* transform (DCT), which is equivalent here)
- The cepstrum is the “spectrum of the (log magnitude) spectrum”!
- Can view quefrency as samples of time in seconds

Example



(fig. from UCL Dept. of Phonetics and Linguistics)

- Here no mel scaling has been done
- The peak at 0.009s corresponds to the fundamental period
- In practice, much fewer samples are used in mel spectrum and cepstrum

Other details

Typically:

- The first 13 MFCCs are retained (this keeps most of the filter and little of the source)
- Estimates of the first and second derivatives (“deltas” and “delta-deltas”) are often appended
- Final feature vector is 39 dimensions computed every 10ms
- Whew!

Optional details

Cepstral mean subtraction/normalization (CMS or CMN) for removing the effect of (stationary) channels

- The mean of each cepstral coefficient (over a “long” segment, e.g. an utterance) is subtracted, $\hat{c}[n] = c[n] - \text{mean}(c[n])$
- If the speech was passed through an unknown channel with frequency response $H[k]$, then the cepstra of original speech + channel are additive, $c[n] = c_{true}[n] + h[n]$, where $h[n] = \text{IDFT}\{\log_{10} |H[k]|\}$
- Then $\hat{c}[n] = c[n] - \text{mean}(c[n]) = c_{true}[n] - \text{mean}(c_{true}[n])$
- This is just the true cepstrum shifted by a fixed amount; the channel is gone
- Makes MFCCs immune to frequency warping channels!

Optional details (cont'd)

Vocal tract length normalization (VTLN)

- To first order, difference between vocal tracts can be represented by a linear warping of the frequency scale
- VTLN techniques estimate the warping factor using various approaches (e.g., pitch-based, formant-based, maximum likelihood-based)

Beyond MFCCs: Other popular features

- The main competitor to MFCCs: *Perceptual linear prediction* coefficients (PLPs), based on the coefficients of a linear regression model for each sample $x[n]$ given the previous L samples:

$$x[n] \approx \sum_{m=1}^L a_m x[n-m] \text{ [Hermansky 1990]}$$

- Other techniques: more explicit auditory models (e.g. [Seneff 1990]) and articulatory models (e.g. [Kirchhoff+ 2002])
- State-of-the-art systems now often use a truncated MFCC “pipeline”, and use the log mel spectrogram directly
- Neural networks are good at learning the rest of the pipeline

Measuring distance between speech signals

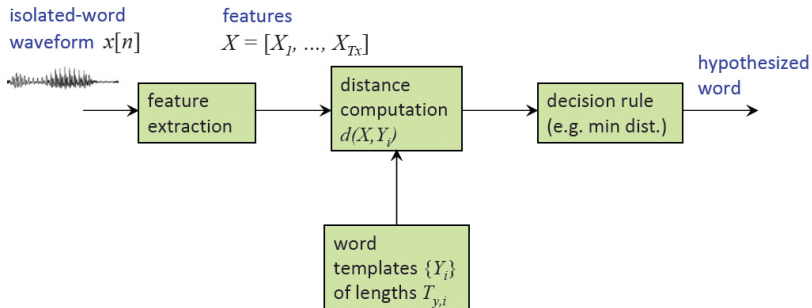
Measuring distance between two short speech frames of equal length

- Pretty easy: compute acoustic feature vectors (MFCCs, spectra) and compute a vector distance
- e.g., Euclidean or cosine distance is reasonable

But we also want to measure distances between pairs of *arbitrary-length* speech signals

- In the 1950s-60s, this was the dominant approach for speech recognition
- At least up to the 1980s it was still used for voice dialing on mobile phones

Isolated-word recognition by template matching



Measuring distance between speech signals

Modern applications

- Query-by-example search
- Automatic spoken term discovery
- Some modern ASR research systems are bringing back template matching as a part of the model
- Other applications: music, medical applications, ...

What do we mean by "distance"?

- Raw distance between the acoustics? (including noise, channel, etc.)
- Do we want to detect speaker differences? Accent differences?
- It depends on the task...
- E.g. for ASR or query-by-example, we typically want the same word/phrase *types* to have low distance, regardless of speaker/accent/acoustic condition

Dynamic time warping (DTW)

- Most common approach for measuring distance between two different-length signals
- A generic measure of distance based on aligning (warping) the signals and computing distance between aligned frames
- Note: there are several variations on DTW; we will follow a popular textbook by Rabiner and Juang (reading will be provided)

But first: Linear time warping

$$d(X, Y) = \sum_{i_x}^{T_x} d(i_x, i_y),$$

where $d(i_x, i_y)$ is distance in feature space between X_{i_x} and Y_{i_y}

and $i_y = \frac{T_y}{T_x} i_x$ (with some appropriate roundoff)

- Assumes that speaking rate variation is uniform (independent, e.g., of linguistic content)
- Not a great assumption... speaking rate variation is typically greater at ends of words, focused on stressed syllables, depends on phones being spoken...

Nonlinear time warpings (from [R+J])

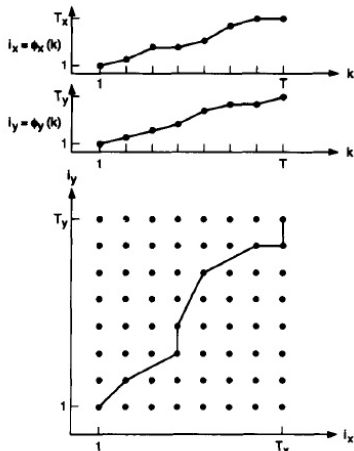


Figure 4.37 An example of time normalization of two sequential patterns to a common time index; the time warping functions ϕ_x and ϕ_y map the individual time index i_x and i_y , respectively, to the common time index k .

$$d_\phi(X, Y) =$$

$$\frac{1}{M_\phi} \sum_{k=1}^T d(\phi_x(k), \phi_y(k)) m(k)$$

$$d(X, Y) = \min_{\phi} d_\phi(X, Y),$$

where:

$d(\phi_x(k), \phi_y(k))$: distance
between test feature vector at
time $\phi_x(k)$ and template vector
at time $\phi_y(k)$

$m(k)$: path weighting
coefficients

M_ϕ : normalizing constant

DTW constraints

- Endpoint constraints:

$$\phi_x(1) = 1, \phi_y(1) = 1,$$

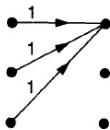
$$\phi_x(T) = T_x, \phi_y(T) = T_y$$

- Monotonicity: No backwards moves (no negative slopes)

$$\phi_x(k+1) \geq \phi_x(k),$$

$$\phi_y(k+1) \geq \phi_y(k)$$

- Local continuity: Only some types of path steps allowed with some weights, e.g.:



Bright idea: Dynamic programming (DP)

Dynamic programming algorithms take advantage of

- Overlapping subproblems: The problem can be broken up into subproblems that can be reused (e.g. using recursion)
- Optimal substructure: An optimal solution can be found efficiently from optimal solutions to subproblems

Typical DP algorithms involve a recursion in which optimal solutions to subproblems are kept and other information is discarded

DP shows up everywhere in speech problems (e.g. hidden Markov model algorithms, word error rate computation)

DP for DTW

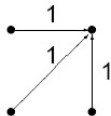
- Key insight: Minimum distance from start state $(1, 1)$ to end state (T_x, T_y) is min distance from $(1, 1)$ to some penultimate state (T'_x, T'_y) plus $\text{dist}((T'_x, T'_y), (T_x, T_y))$...
- ... which in turn is the minimum distance to some prior penultimate state (T''_x, T''_y) , plus $\text{dist}((T''_x, T''_y), (T'_x, T'_y))$, plus $\text{dist}((T'_x, T'_y), (T_x, T_y))$...
- ... and so on.
- So for each state, we need only keep track of the *best* way of getting there from predecessor states: Main recursion step

OK, let's do it!

- Warning: DTW is a lot easier to *do* than to *write down*

Example recursion step

Example move set:



$$\text{let } D(i_x, i_y) = \min_{\phi, T'} \sum_{k=1}^{T'} d(\phi_x(k), \phi_y(k)) m(k)$$

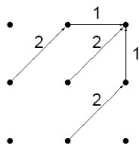
$$\text{s.t. } \phi_x(T') = i_x,$$

$$\phi_y(T') = i_y$$

$$\begin{aligned} \text{then } D(i_x, i_y) = \min[& D(i_x - 1, i_y - 1) + d(i_x, i_y), \\ & D(i_x - 1, i_y) + d(i_x, i_y), \\ & D(i_x, i_y - 1) + d(i_x, i_y)] \end{aligned}$$

Example recursion step

Another example move set:



$$D(i_x, i_y) = \min \begin{bmatrix} D(i_x - 1, i_y - 1) & + & 2d(i_x, i_y), \\ D(i_x - 2, i_y - 1) & + & 2d(i_x - 1, i_y) + d(i_x, i_y), \\ D(i_x - 1, i_y - 2) & + & 2d(i_x, i_y - 1) + d(i_x, i_y) \end{bmatrix}$$

In general:

$$D(i_x, i_y) = \min_{i'_x, i'_y} [D(i'_x, i'_y) + \theta((i'_x, i'_y), (i_x, i_y))], \text{ where}$$

$$\theta((i'_x, i'_y), (i_x, i_y)) = \sum_{l=0}^L d(\phi_x(T' - l), \phi_y(T' - l)) m(T' - l),$$

where L is the number of steps from (i'_x, i'_y) to (i_x, i_y)

DP algorithm for DTW

- Initialization: $D(1, 1) = d(1, 1)m(1)$

- Recursion: For $1 \leq i_x \leq T_x$, $1 \leq i_y \leq T_y$

$$D(i_x, i_y) = \min_{i'_x, i'_y} [D(i'_x, i'_y) + \theta((i'_x, i'_y), (i_x, i_y))], \text{ where}$$

$$\theta((i'_x, i'_y), (i_x, i_y)) = \sum_{l=0}^L d(\phi_x(T' - l), \phi_y(T' - l)) m(T' - l),$$

where L is the number of steps from (i'_x, i'_y) to (i_x, i_y) ,


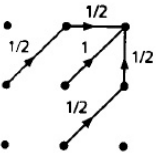
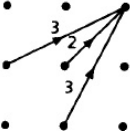
$$\phi_x(T') = i_x, \phi_y(T') = i_y, \phi_x(T' - L) = i'_x, \phi_y(T' - L) = i'_y$$

- Termination: $d(X, Y) = \frac{D(T_x, T_y)}{M_\phi}$

If we wish to find the distance as well as the best path, then keep a pointer to the best incoming path at each step, and backtrace at the end

DTW constraint examples and recursion formulas

TABLE 4.7. Summary of sets of local constraints, slope weights, and DP recursion formulas

Local Constraints & Slope Weights	DP Recursion Formula
	$\min \left\{ \begin{array}{l} D(i_x - 1, i_y) + d(i_x, i_y), \\ D(i_x - 1, i_y - 1) + 2d(i_x, i_y), \\ D(i_x, i_y - 1) + d(i_x, i_y) \end{array} \right\}$
	$\min \left\{ \begin{array}{l} D(i_x - 2, i_y - 1) + \frac{1}{2}[d(i_x - 1, i_y) + d(i_x, i_y)], \\ D(i_x - 1, i_y - 1) + d(i_x, i_y), \\ D(i_x - 1, i_y - 2) + \frac{1}{2}[d(i_x, i_y - 1) + d(i_x, i_y)] \end{array} \right\}$
	$\min \left\{ \begin{array}{l} D(i_x - 2, i_y - 1) + 3d(i_x, i_y), \\ D(i_x - 1, i_y - 1) + 2d(i_x, i_y), \\ D(i_x - 1, i_y - 2) + 3d(i_x, i_y), \end{array} \right\}$

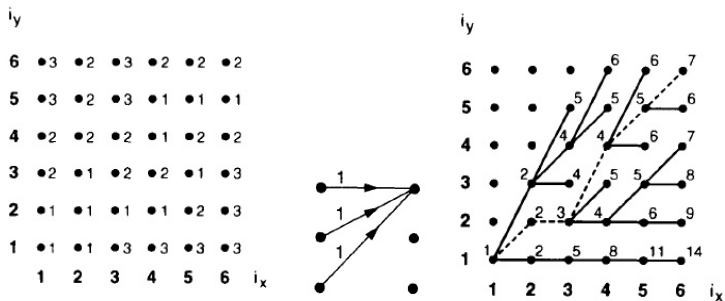
A note on path normalization

- We removed M_ϕ from the DP assuming it is independent of the warping path

- Typically, want $M_\phi = \sum_{k=1}^T m(k)$

- For some move sets, $M_\phi = T_x$ or $T_x + T_y$
- For others, M_ϕ is highly dependent on the chosen path – breaks dynamic programming
- Usually, use $M_\phi = T_x$ or $T_x + T_y$ anyway (see reading for when this might be bad)

DTW example



What about the frame distance function

$$d(i_x, i_y)?$$

- Typically, this is a distance between X_{i_x} and Y_{i_y} , which are vectors of dimensionality D
- Euclidean: $d(i_x, i_y) = \left[\sum_{d=1}^D (X_{i_x,d} - Y_{i_y,d})^2 \right]^{1/2}$
- Cosine: $d(i_x, i_y) = 1 - \frac{X_{i_x} \cdot Y_{i_y}}{\|X_{i_x}\| \|Y_{i_y}\|} = 1 - \frac{X_{i_x}^T Y_{i_y}}{\|X_{i_x}\| \|Y_{i_y}\|}$
(note: not a true distance metric)