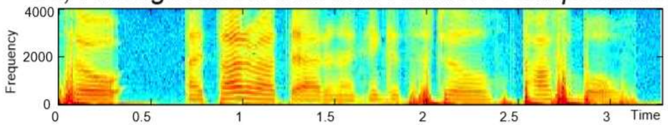
Speech Recognition

Outline

Recognizing speech

"So, I thought about that and I think it's still possible"



What kind of information might we want from the speech signal?

- words
- phrasing, 'speech acts' (prosody)
- mood / emotion
- speaker identity

What kind of processing do we need to get at that information?

- time scale of feature extraction
- signal aspects to capture in features
- signal aspects to exclude from features

Speech recognition as Transcription

Transcription = “speech to text"

- find a word string to match the utterance

Gives neat objective measure: word error rate (WER) %

- can be a sensitive measure of performance

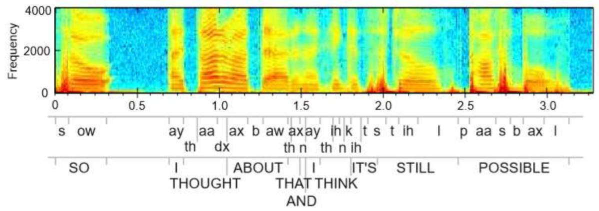
Reference:	THE	CAT	SAT	ON	THE		MAT
Recognized:	–	CAT	SAT	AN	THE	A	MAT
	Deletion			*Substitution*		*Insertion*	

Three kinds of errors:

$$WER = (S + D + I)/\mathrm{N}$$

Problems: Within-speaker variability

- Timing variation
  - word duration varies enormously



  - fast speech ‘reduces' vowels
- Speaking style variation
  - careful/casual articulation
  - soft/loud speech
- Contextual effects
  - speech sounds vary with context, role: “How **do** you **do**?”

Problems: Between-speaker variability

Accent variation

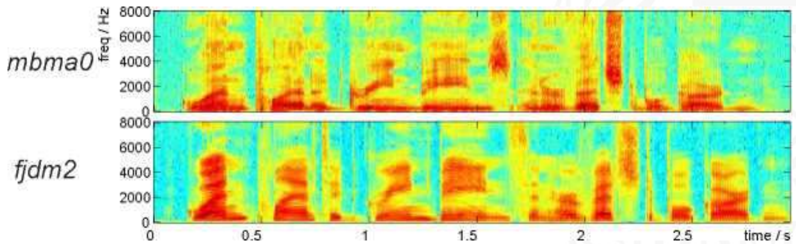
- regional / mother tongue

Voice quality variation

- gender, age, huskiness, nasality

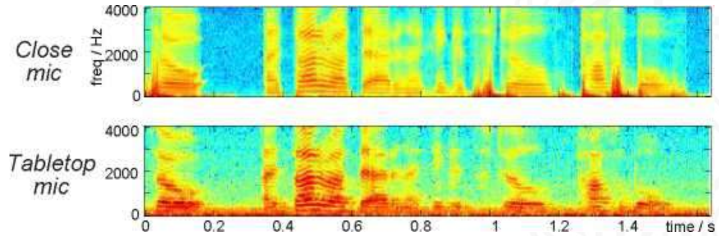
Individual characteristics

- mannerisms, speed, prosody



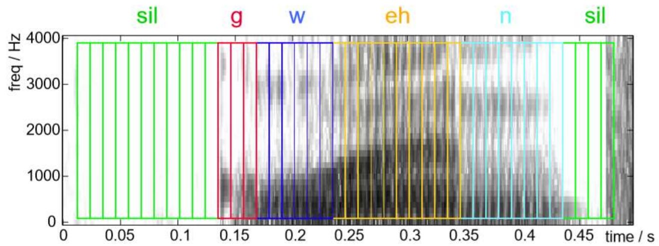
Problems: Environment variability

- Background noise
  - fans, cars, doors, papers
- Reverberation
  - 'boxiness' in recordings
- Microphone/channel
  - huge effect on relative spectral gain



How to recognize speech?

- Cross correlate templates?
  - waveform?
  - spectrogram?
  - time-warp problems
- Match short-segments & handle time-warp later
  - model with slices of ~10 ms
  - pseudo-stationary model of words:



  - other sources of variation...

Probabilistic formulation

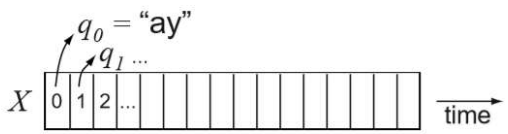
Vowel
Consonant
Word: a~z
Characters

- Probability that segment label is correct
  - gives standard form of speech recognizers
- Feature calculation: $s[n] \rightarrow X_m \quad (m = \frac{n}{H})$
  - transforms signal into easily-classified domain
- Acoustic classifier: $p(q^i \mid X)$
  - calculates probabilities of each mutually-exclusive state q^i
- 'Finite state acceptor' (i.e. HMM),

$$Q^* = \underset{\{q_0, q_1, \ldots q_L\}}{\operatorname{argmax}} \; p(q_0, q_1, \ldots q_L \mid X_0, X_1, \ldots X_L)$$

Hidden Markov Model

  - MAP match of allowable sequence to probabilities:



Standard speech recognizer structure

Fundamental equation of speech recognition:

$$
\begin{aligned}
Q^* &= \underset{Q}{\operatorname{argmax}}\, p(Q \mid X, \Theta) \\
&= \underset{Q}{\operatorname{argmax}}\, p(X \mid Q, \Theta)\, p(Q \mid \Theta)
\end{aligned}
$$

- X = acoustic features
- $p(X \mid Q, \Theta)$ = acoustic model
- *p(Q | ©)* = language model
- $p(Q \mid \Theta)$ = search over sequences

Questions:

- what are the best features?
- how do we do model them?
- how do we find/match the state sequence?

Outline

Feature Calculation

Goal: Find a representational space most suitable for classification

- waveform: voluminous, redundant, variable
- spectrogram: better, still quite variable
- ...?

Pattern Recognition:

representation is upper bound on performance

- maybe we *should* use the waveform...
- or, maybe the representation can do *all* the work

Feature calculation is intimately bound to classifier

- pragmatic strengths and weaknesses

Features develop by slow evolution

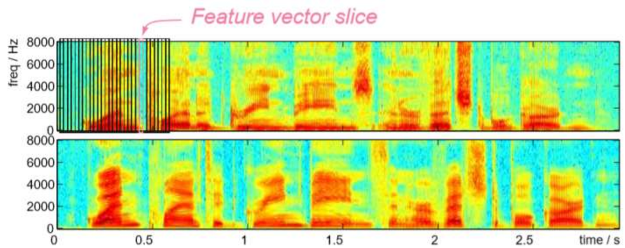
- current choices more historical than principled

Features (1): Spectrogram

Plain STFT as features *e.g.*

$$X_m[k] = S[mH, k] = \sum_n s[n + mH]\, w[n]\, e^{-j2\pi kn/N}$$

Consider examples:



Similarities between corresponding segments

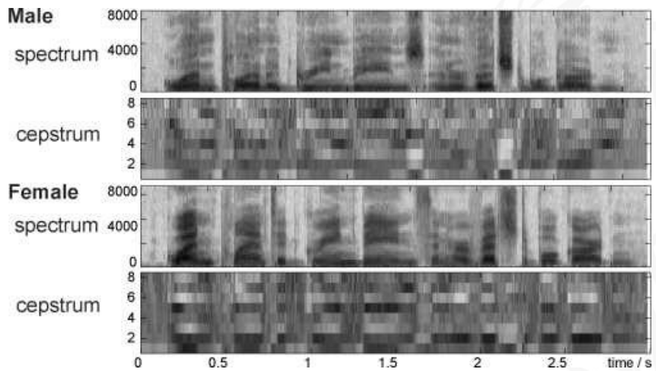
- but still large differences

Features (2): Cepstrum

Idea: Decorrelate, summarize spectral slices:

$$X_m[\ell] = \mathrm{IDFT}\{\log |S[mH, k]|\}$$

- good for Gaussian models
- greatly reduce feature dimension



Features (3): Frequency axis warp

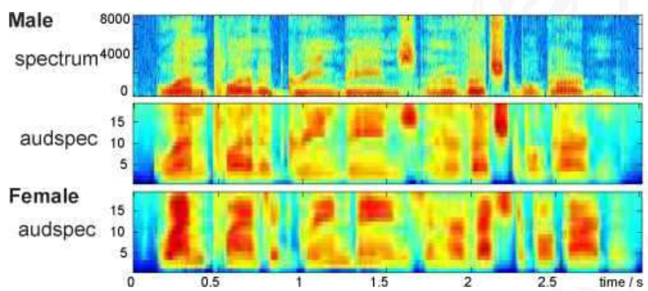
Linear frequency axis gives equal 'space' to 0-1 kHz and 3-4 kHz

- but perceptual importance very different

Warp frequency axis closer to perceptual axis

- mel, Bark, constant-Q ...

$$X[c] = \sum_{k=\ell_c}^{u_c} |S[k]|^2$$



Features (4): Spectral smoothing

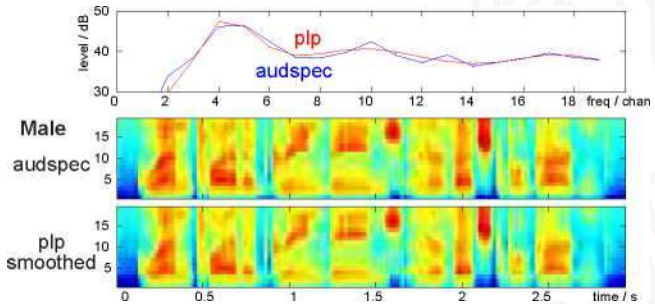
Generalizing across different speakers is helped by smoothing *(i.e. blurring)* spectrum

Truncated cepstrum is one way:

- MMSE approx to log |S[*k*]|

LPC modeling is a little different:

- MMSE approx to |S[k]| — prefers detail at peaks



Features (5): Normalization along time

Idea: feature variations, not absolute level

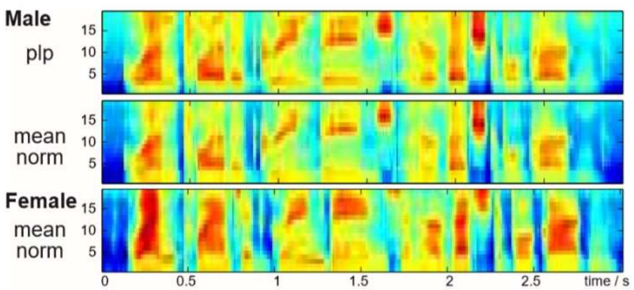
Hence: calculate average level and subtract it:

$$\hat{Y}[n,k] = \hat{X}[n,k] - \operatorname*{mean}_{n}\{\hat{X}[n,k]\}$$

Factors out fixed channel frequency response

$$x[n] = h_c * s[n]$$

$$\hat{X}[n,k] = \log|X[n,k]| = \log|H_c[k]| + \log|S[n,k]|$$



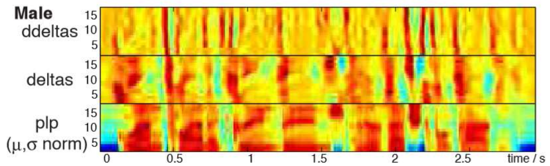
Delta features

Want each segment to have ‘static' feature vals

- but some segments intrinsically dynamic!

— calculate their derivatives—maybe steadier?

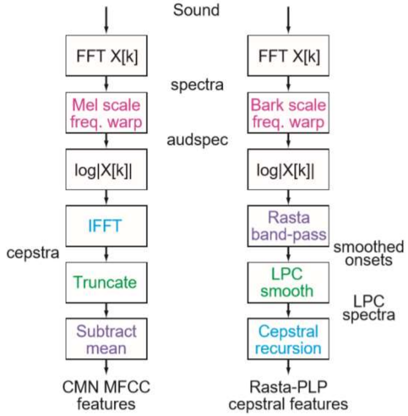
Append dX/dt (+ d^2X/dt^2) to feature vectors



Relates to onset sensitivity in humans?

Feature calculation

MFCCs and/or RASTA-PLP



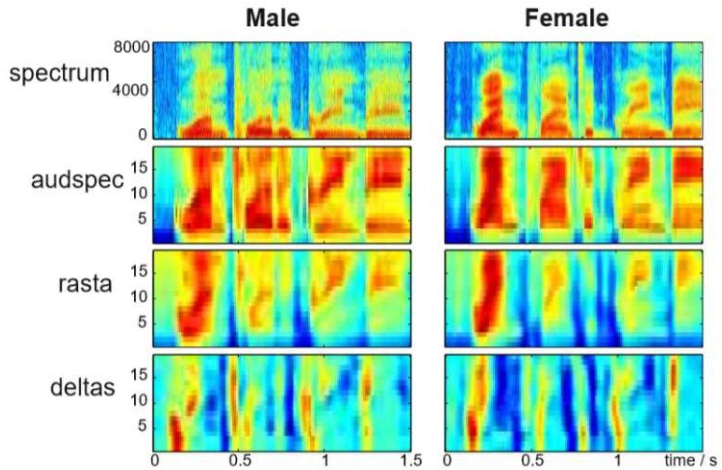
Mel-Frequency Cepstral Coefficients (MFCC)

- Spectrum → Mel-Filters → Mel-Spectrum
- Say log X[k] = log (Mel-Spectrum)
- NOW perform Cepstral analysis on log X[k]
  - log X[k] = log H[k] + log E[k]
  - Taking IFFT
  - x[k] = h[k] + e[k]
- Cepstral coefficients h[k] obtained for Mel-spectrum are referred to as Mel-Frequency Cepstral Coefficients often denoted by *MFCC*

Mel-Frequency Analysis

- Mel-Frequency analysis of speech is based on human perception experiments
- It is observed that human ear acts as filter
  - It concentrates on only certain frequency components
- These filters are non-uniformly spaced on the frequency axis
  - More filters in the low frequency regions
  - Less no. of filters in high frequency regions

Features summary



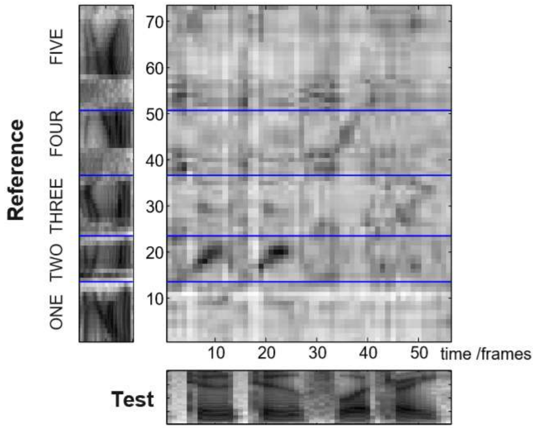
- Normalize same phones
- Contrast different phones

Outline

1. Recognizing speech
2. Feature calculation
3. Sequence recognition
4. Large vocabulary, continuous speech recognition (LVCSR)

Sequence recognition: Dynamic Time Warp (DTW)

Frame-wise comparison with stored templates:

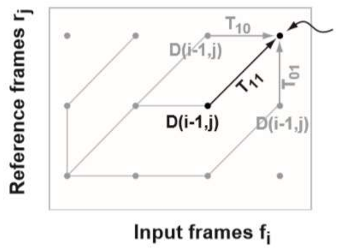


- distance metric?
- comparison across templates?

Dynamic Time Warp (2)

Find lowest-cost constrained path:

- matrix $d(/, j)$ of distances
  between input frame f and reference frame r_j
- allowable predecessors and transition costs T_{xy}



Lowest cost to (i,j)

$$D(i,j) = d(i,j) + \min\begin{pmatrix} D(i-1,j) + T_{10} \\ D(i,j-1) + T_{01} \\ D(i-1,j-1) + T_{11} \end{pmatrix}$$

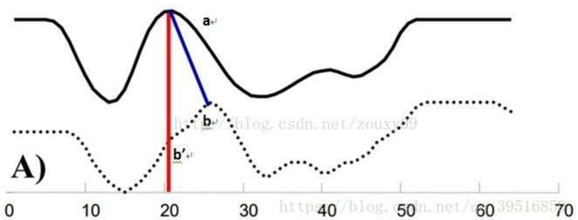
Local match cost

Best predecessor (including transition cost)

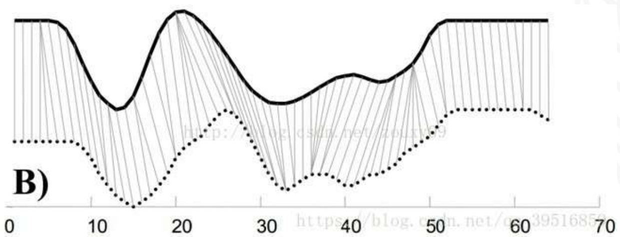
Best path via traceback from final state

- store predecessors for each (/, j)

Dynamic Time Warp (example)

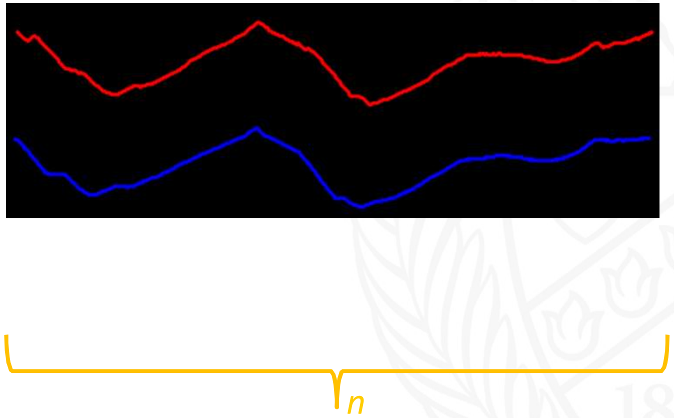


$$\gamma(i,j) = d(q_i,c_j) + \min\{\, \gamma(i\text{-}1,j\text{-}1),\ \gamma(i\text{-}1,j\,),\ \gamma(i,j\text{-}1)\,\}$$

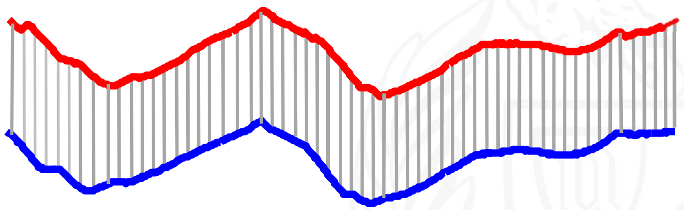


A Visual Intuition of Distance Measures

We have two time series, what is the distance between them?
Equivalently, how similar are they?



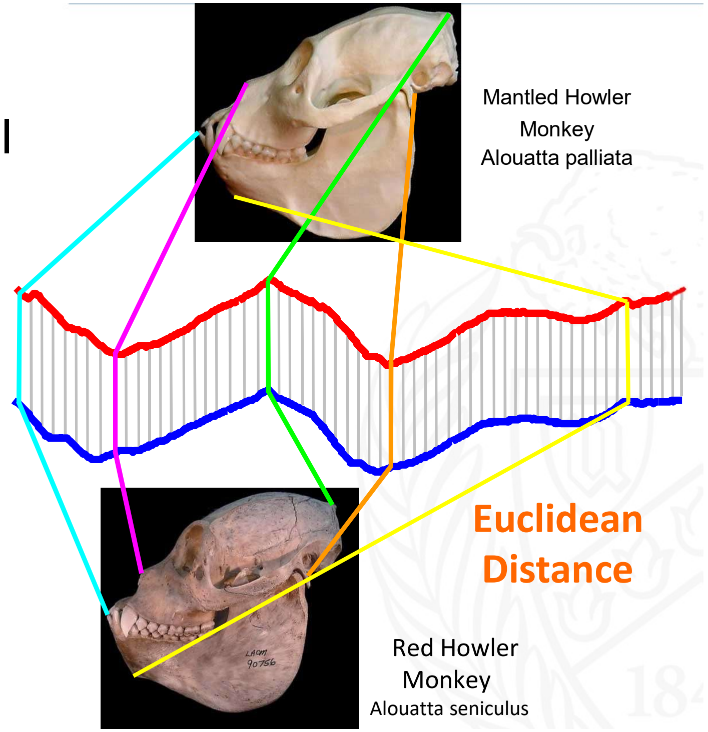
A Visual Intuition of Distance Measures

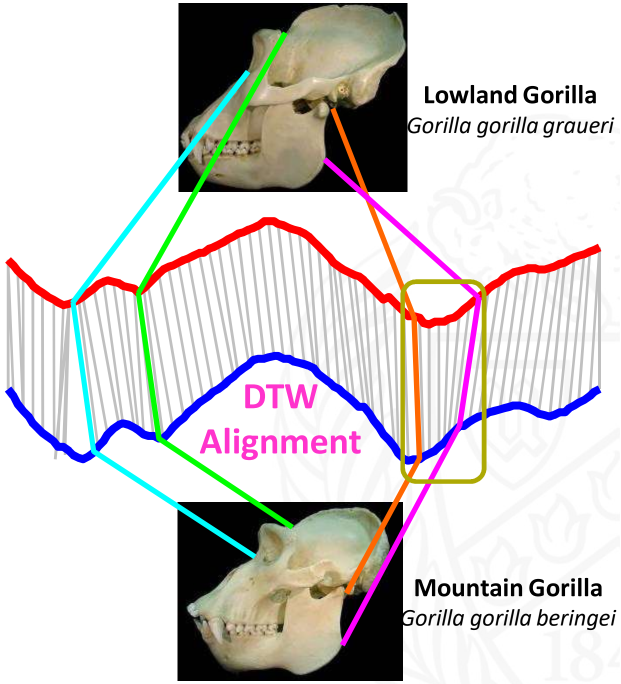


Euclidean Distance

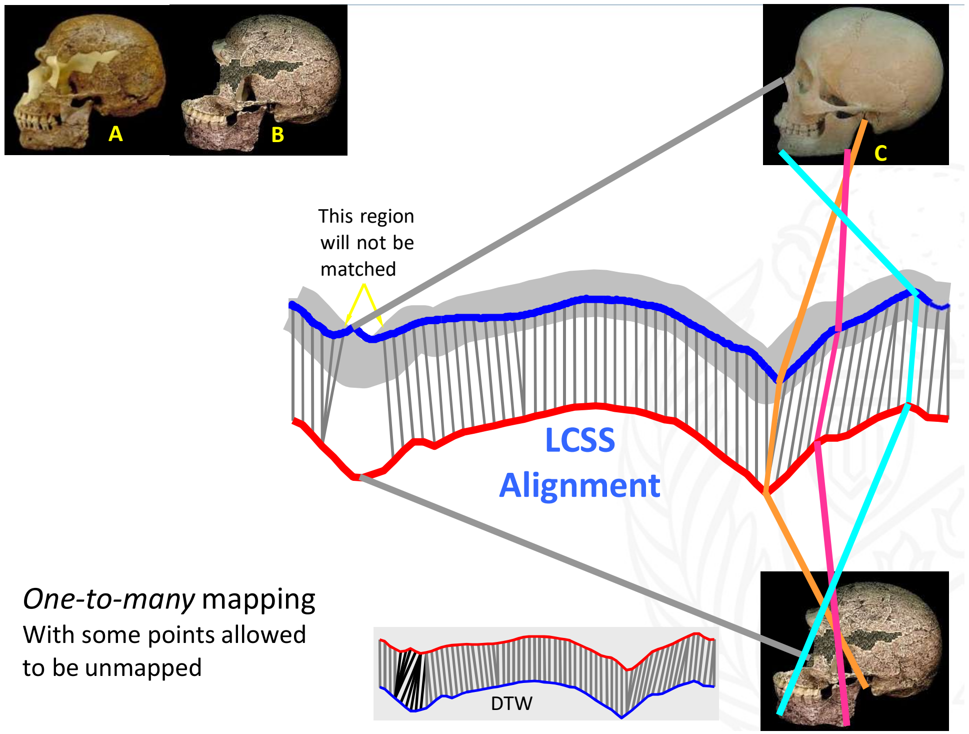
One-to-one mapping

Skull


Mantled Howler
Monkey
Alouatta palliata
Euclidean
Distance
Red Howler
Monkey
Alouatta seniculus

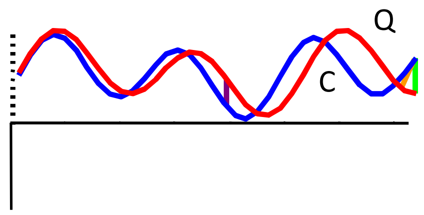


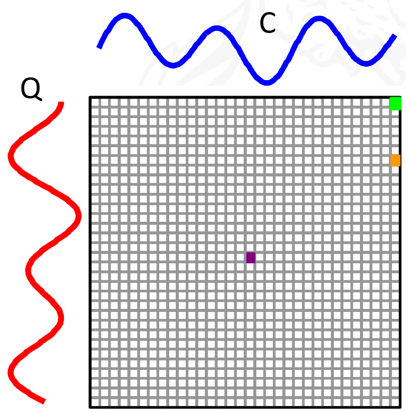
One-to-many mapping


A
B
C
This region will not be matched
LCSS
Alignment
One-to-many mapping
With some points allowed to be unmapped
DTW

How is DTW Calculated? I

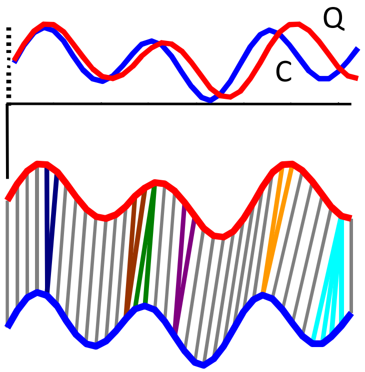
We create a matrix the size of |Q| by |C|, then fill it in with the distance between every possible pair of points in our two time series.





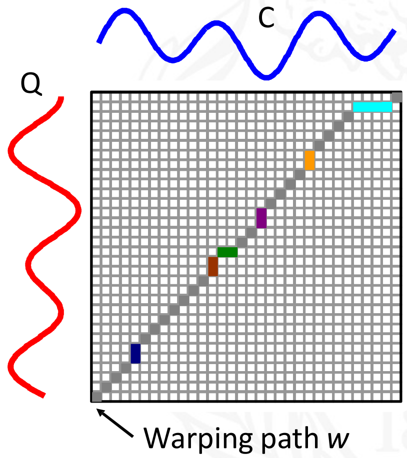
How is DTW Calculated? II

Every possible warping between two time series, is a path through the matrix. We want the *best* one…



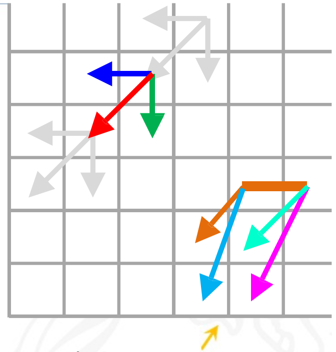
This recursive function gives us the minimum cost path

$\gamma(i,j) = d(q_i,c_j) + \min\{\ \gamma(i\text{-}1,j\text{-}1),\ \gamma(i\text{-}1,j\ ),\ \gamma(i,j\text{-}1)\ \}$



We can visualize the recursive function as a "step pattern" of allowable moves, or search operators

$$\gamma(i,j) = d(q_i,c_j) + \min\{\gamma(i-1,j-1), \gamma(i-1,j), \gamma(i,j-1)\}$$



This suggests two generalizations.

- We could weight the diagonal step, to "discourage" the warping path wandering too far from the diagonal.
- We could create other steps patterns, including asymmetric step patterns.

Both ideas were extensively studied when DTW was the dominant speech processing algorithm, but have not been investigated extensively in the data mining context.

Empirically, they seem to make little difference.

We only consider the classic symmetric1 step pattern

How is DTW Calculated? *Disclaimer!*

In practice we **don't** use *recursion* to calculate DTW. Instead we use an equivalent *iterative* method.

The iterative method is both absolutely faster, and it is allows many speed-up optimizations (early abandoning etc).

The time difference is several orders of magnitude.

This recursive function gives us the minimum cost path

$\gamma(i,j) = d(q_i,c_j) + \min\{ \gamma(i-1,j-1), \gamma(i-1,j), \gamma(i,j-1) \}$

Logically correct, but too slow and memory intensive

DTW is a Distance *Measure*, not a *Metric* 1 of 2

Requirements to be a metric

Yes for DTW

- $D(A,B) = D(B,A)$ — *Symmetry*
- $D(A,A) = 0$ — *Constancy of Self-Similarity*

No for DTW

- $D(A,B) = 0$ IIf A=B — *Positivity (Separation)*
- $D(A,B) \leq D(A,C) + D(B,C)$ — *Triangular Inequality*

Normally we prefer metrics over measures for two reasons:

- Non-Metrics can sometimes give pathological solutions when clustering or classifying data etc.
- Almost all speed-up “tricks” for high dimensional data exploit the Triangular Inequality.

DTW: Time and Space complexity

- The “off-the-shelf” DTW has
- a time complexity of O(n^2) (with a large constant factor)
- a space complexity of O(n^2)

This is the most cited reason for not using DTW.

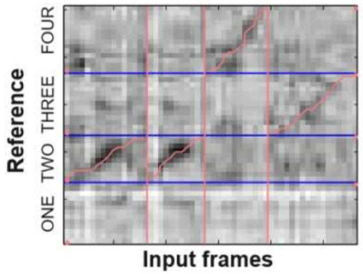
- However, as we will show in the later part of this talk. DTW can have
- a space complexity of O(n)
- an amortized time complexity of O(n) (with a very small constant factor)

DTW-based recognition

Reference templates for each possible word

For isolated words:

- mark endpoints of input word
- calculate scores through each template (+prune)



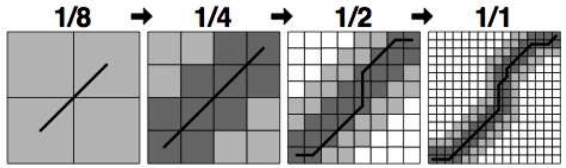


Figure 6. The four different resolutions evaluated during a complete run of the FastDTW algorithm.

- continuous speech: link together word ends

Successfully handles timing variation

- recognize speech at reasonable cost

Statistical sequence recognition

DTW limited because it's hard to optimize

- learning from multiple observations
- interpretation of distance, transition costs?

Need a theoretical foundation: Probability

Formulate recognition as MAP choice among word sequences:

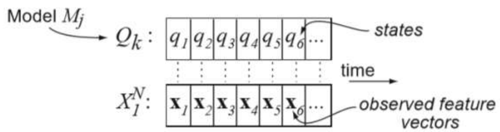
$$Q^* = \underset{Q}{\operatorname{argmax}}\, p(Q \mid X, \Theta)$$

- X = observed features
- Q = word-sequences
- Θ = all current parameters

State-based modeling

Assume discrete-state model for the speech:

- observations are divided up into time frames
- model — states — observations:



Vowel
Consonant
Word: a~z
Characters

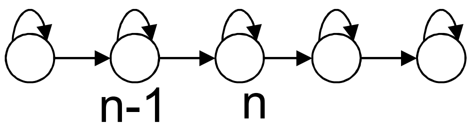
Probability of observations given model is:

$$p(X \mid \Theta) = \sum_{\text{all } Q} p(X_1^N \mid Q, \Theta)\, p(Q \mid \Theta)$$

- sum over all possible state sequences Q

How do observations X_1^N depend on states Q?

How do state sequences Q depend on model Θ?



Markov assumption

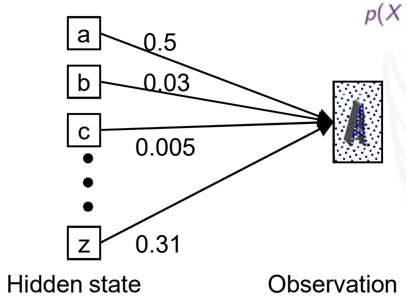
$$p(X, Q \mid \Theta) = \prod_n p(x_n \mid q_n) p(q_n \mid q_{n-1})$$

Word recognition example(a).

- Typed word recognition, assume all characters are separated.

- Character recognizer outputs probability of the image being particular character, P(image|character).

$$p(X \mid \Theta) = \sum_{\text{all } Q} p(X_1^N \mid Q, \Theta)\, p(Q \mid \Theta)$$

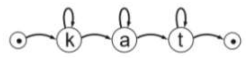


HMM review

HMM is specified by parameters Θ:

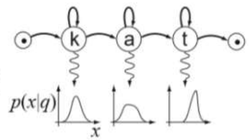
- states q^i

- transition probabilities a_{ij}



	k	a	t	•
•	1.0	0.0	0.0	0.0
k	0.9	0.1	0.0	0.0
a	0.0	0.9	0.1	0.0
t	0.0	0.0	0.9	0.1

- emission distributions $b_i(x)$

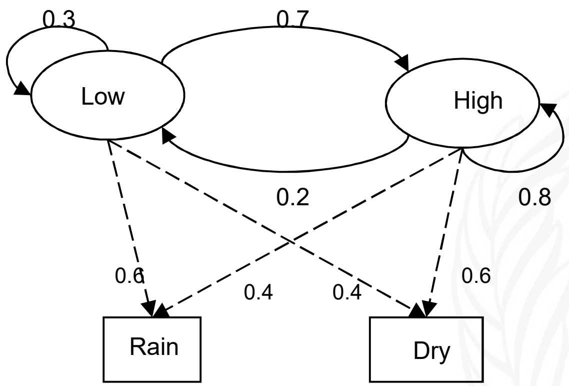


Observation probability

(+ initial state probabilities π_i)

$$a_{ij} \equiv p(q_n^j \mid q_{n-1}^i) \qquad b_i(x) \equiv p(x \mid q_i) \qquad \pi_i \equiv p(q_1^i)$$

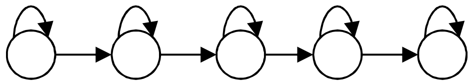
Example of Hidden Markov Model



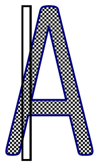
- Two states : ‘Low’ and ‘High’ atmospheric pressure.
- Two observations : ‘Rain’ and ‘Dry’.

Character recognition with HMM example.

- The structure of hidden states is chosen.



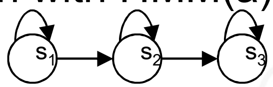
- Observations are feature vectors extracted from vertical slices.



- Probabilistic mapping from hidden state to feature vectors:
  1. use mixture of Gaussian models
  2. Quantize feature vector space.

Exercise: character recognition with HMM(a)

- The structure of hidden states:

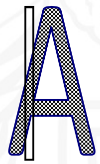


- Observation = number of islands in the vertical slice.

•HMM for character ‘A’ :

Transition probabilities: $\{q_{ij}\}= \begin{pmatrix} .8 & .2 & 0 \\ 0 & .8 & .2 \\ 0 & 0 & 1 \end{pmatrix}$

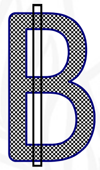
Observation probabilities: $\{b_{jk}\}= \begin{pmatrix} .9 & .1 & 0 \\ .1 & .8 & .1 \\ .9 & .1 & 0 \end{pmatrix}$



•HMM for character ‘B’ :

Transition probabilities: $\{q_{ij}\}= \begin{pmatrix} .8 & .2 & 0 \\ 0 & .8 & .2 \\ 0 & 0 & 1 \end{pmatrix}$

Observation probabilities: $\{b_{jk}\}= \begin{pmatrix} .9 & .1 & 0 \\ 0 & .2 & .8 \\ .6 & .4 & 0 \end{pmatrix}$



Exercise: character recognition with HMM(b)

- Suppose that after character image segmentation the following sequence of numbers in 4 slices was observed:
  { 1, 3, 2, 1}

- What HMM is more likely to generate this observation sequence , HMM for ‘A’ or HMM for ‘B’ ?

Exercise: character recognition with HMM(c)

Consider likelihood of generating given observation for each possible sequence of hidden states:

$$p(X, Q \mid \Theta) = \prod_n p(x_n \mid q_n) p(q_n \mid q_{n-1})$$

- HMM for character 'A':

Hidden state sequence	Transition probabilities		Observation probabilities	
$s_1 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3$	$.8 * .2 * .2$	$*$	$.9 * 0 * .8 * .9$	= 0
$s_1 \rightarrow s_2 \rightarrow s_2 \rightarrow s_3$	$.2 * .8 * .2$	$*$	$.9 * .1 * .8 * .9$	= 0.0020736
$s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_3$	$.2 * .2 * 1$	$*$	$.9 * .1 * .1 * .9$	= 0.000324

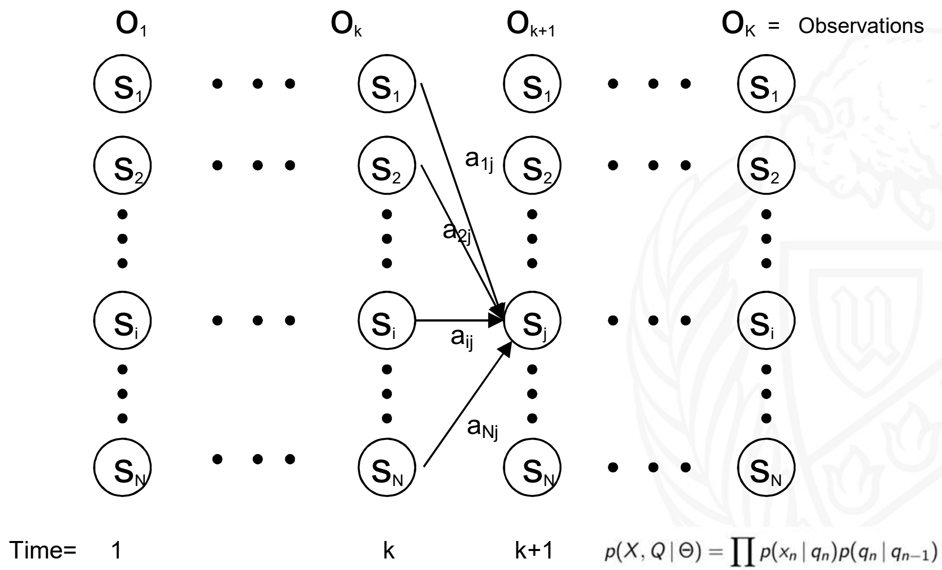
Total = 0.0023976

- HMM for character 'B':

Hidden state sequence	Transition probabilities		Observation probabilities	
$s_1 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3$	$.8 * .2 * .2$	$*$	$.9 * 0 * .2 * .6$	= 0
$s_1 \rightarrow s_2 \rightarrow s_2 \rightarrow s_3$	$.2 * .8 * .2$	$*$	$.9 * .8 * .2 * .6$	= 0.0027648
$s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_3$	$.2 * .2 * 1$	$*$	$.9 * .8 * .4 * .6$	= 0.006912

Total = 0.0096768

Trellis representation of a HMM-Viterbi



$$p(X, Q \mid \Theta) = \prod_n p(x_n \mid q_n) p(q_n \mid q_{n-1})$$

Dynamic programming

HMM summary (1)

HMMs are a generative model: recognition is inference of $p(Q \mid X)$

During generation, behavior of model depends only on current state q_n:

- transition probabilities $p(q_{n+1} \mid q_n) = a_{ij}$
- observation distributions $p(x_n \mid q_n) = b_i(x)$

Given states $Q = \{q_1, q_2, \ldots, q_N\}$

and observations $X = X_1^N = \{x_1, x_2, \ldots, x_N\}$

Markov assumption makes

$$p(X, Q \mid \Theta) = \prod_n p(x_n \mid q_n) p(q_n \mid q_{n-1})$$

HMM summary (2)

Calculate $p(X \mid \Theta)$ via forward recursion:

$$p(X_1^n, q_n^j) = \alpha_n(j) = \left[\sum_{i=1}^{S} \alpha_{n-1}(i) a_{ij} \right] b_j(x_n)$$

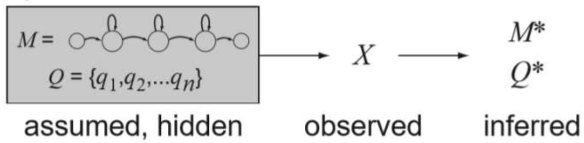
Viterbi (best path) approximation

$$\alpha_n^*(j) = \left[\max_i \left\{ \alpha_{n-1}^*(i) a_{ij} \right\} \right] b_j(x_n)$$

- then backtrace...

$$Q^* = \underset{Q}{\operatorname{argmax}}(X, Q \mid \Theta)$$

Pictorially:



Forward recursion for HMM

- Initialization:

$$\alpha_1(i) = P(o_1, q_1 = s_i) = \pi_i\, b_i(o_1)\ , \quad 1 <= i <= N.$$

- Forward recursion:

$$\alpha_{k+1}(i) = P(o_1 o_2 \ldots o_{k+1}, q_{k+1} = s_j) =$$
$$\Sigma_i\, P(o_1 o_2 \ldots o_{k+1}, q_k = s_i, q_{k+1} = s_j) =$$
$$\Sigma_i\, P(o_1 o_2 \ldots o_k, q_k = s_i)\, a_{ij}\, b_j(o_{k+1}) =$$
$$[\Sigma_i\, \alpha_k(i)\, a_{ij}]\, b_j(o_{k+1})\ , \quad 1 <= j <= N,\ 1 <= k <= K-1.$$

- Termination:

$$P(o_1 o_2 \ldots o_K) = \Sigma_i\, P(o_1 o_2 \ldots o_K, q_K = s_i) = \Sigma_i\, \alpha_K(i)$$

- Complexity :

N^2K operations.

Backward recursion for HMM

- Define the forward variable $\beta_k(i)$ as the joint probability of the partial observation sequence $o_{k+1}\, o_{k+2} \dots\, o_K$ given that the hidden state at time k is s_i : $\beta_k(i)=$ $P(o_{k+1}\, o_{k+2} \dots\, o_K | q_k = s_i)$
- Initialization:

$$\beta_K(i)= 1 \quad , \; 1<=i<=N.$$

- Backward recursion:

$$\beta_k(j)= P(o_{k+1}\, o_{k+2} \dots\, o_K | q_k = s_j) =$$
$$\Sigma_i\, P(o_{k+1}\, o_{k+2} \dots\, o_K ,\, q_{k+1} = s_i \,|\, q_k = s_j) =$$
$$\Sigma_i\, P(o_{k+2}\, o_{k+3} \dots\, o_K | q_{k+1} = s_i)\; a_{ji}\; b_i(o_{k+1}) =$$
$$\Sigma_i\, \beta_{k+1}(i)\, a_{ji}\; b_i(o_{k+1}) \;, \qquad 1<=j<=N,\; 1<=k<=K-1.$$

- Termination:

$$P(o_1\, o_2 \dots\, o_K) = \Sigma_i\, P(o_1\, o_2 \dots\, o_K ,\, q_1 = s_i) =$$
$$\Sigma_i\, P(o_1\, o_2 \dots\, o_K | q_1 = s_i)\; P(q_1 = s_i) = \Sigma_i\, \beta_1(i)\, b_i(o_1)\, \pi_i$$

Decoding problem

- **Decoding problem.** Given the HMM $M=(A, B, \pi)$ and the observation sequence $O=o_1 o_2 \ldots o_K$, calculate the most likely sequence of hidden states s_i that produced this observation sequence.
- We want to find the state sequence $Q= q_1 \ldots q_K$ which maximizes $P(Q \mid o_1 o_2 \ldots o_K)$, or equivalently $P(Q, o_1 o_2 \ldots o_K)$.
- Brute force consideration of all paths takes exponential time. Use efficient **Viterbi algorithm** instead.
- Define variable $\delta_k(i)$ as the maximum probability of producing observation sequence $o_1 o_2 \ldots o_k$ when moving along any hidden state sequence $q_1 \ldots q_{k-1}$ and getting into $q_k= s_i$.

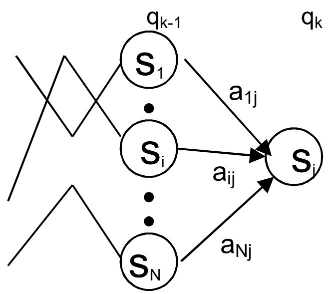
$$\delta_k(i) = \max P(q_1 \ldots q_{k-1}, q_k= s_i, o_1 o_2 \ldots o_k)$$

where max is taken over all possible paths $q_1 \ldots q_{k-1}$.

Viterbi algorithm (1)

- General idea:

  if best path ending in $q_k = S_j$ goes through $q_{k-1} = S_i$ then it should coincide with best path ending in $q_{k-1} = S_i$.



- $\delta_k(i) = \max P(q_1 \ldots q_{k-1}, q_k = s_j, o_1 o_2 \ldots o_k) =$
  $\max_i [\, a_{ij}\, b_j(o_k) \ \max P(q_1 \ldots q_{k-1} = s_i, o_1 o_2 \ldots o_{k-1})\,]$

- To backtrack best path keep info that predecessor of S_j was S_i.

Viterbi algorithm (2)

- Initialization:

$$\delta_1(i) = \max P(q_1 = s_i \,,\, o_1) = \pi_i \, b_i(o_1) \,, \quad 1<=i<=N.$$

- Forward recursion:

$$\delta_k(j) = \max P(q_1 \ldots q_{k-1} \,, q_k = s_j \,,\, o_1 o_2 \ldots o_k) =$$

$$\max_i [\, a_{ij} \, b_j(o_k) \, \max P(q_1 \ldots q_{k-1} = s_i \,,\, o_1 o_2 \ldots o_{k-1}) \,] =$$

$$\max_i [\, a_{ij} \, b_j(o_k) \, \delta_{k-1}(i) \,] \,, \qquad 1<=j<=N,\ 2<=k<=K.$$

- Termination: choose best path ending at time K

$$\max_i [\, \delta_K(i) \,]$$

- Backtrack best path.

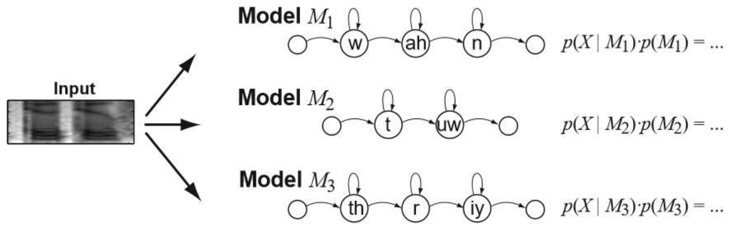
This algorithm is similar to the forward recursion of evaluation problem, with Σ *replaced by max and additional backtracking.*

Outline

Recognition with HMMs

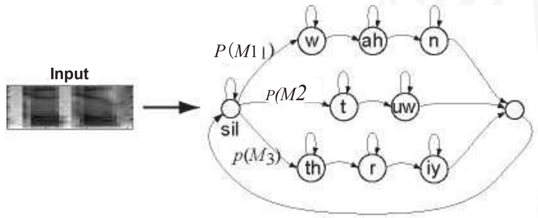
Isolated word

- choose best $p(M \mid X) \propto p(X \mid M)p(M)$



Continuous speech

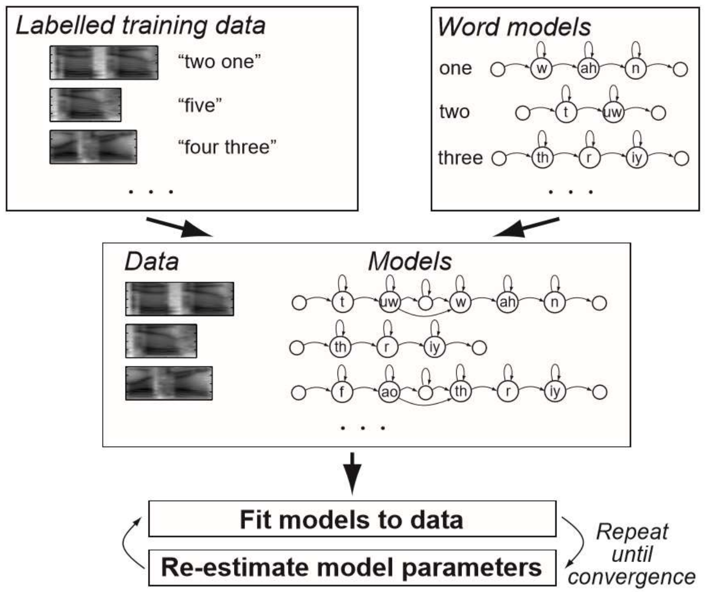
- Viterbi decoding of one large HMM gives words: Dynamic programming



Training HMMs

- Probabilistic foundation allows us to train HMMs to 'fit' training data
  - *i.e.* estimate a_{ij}, $b_i(x)$ given data
    - better than DTW...
- Algorithms to improve $p(\Theta \mid X)$ are key to success of HMMs
    - maximum-likelihood of models...
- State alignments Q for training examples are generally unknown
    - ... else estimating parameters would be easy
- Viterbi training:
    - 'Forced alignment'
    - choose 'best' labels (heuristic)
- EM training
    - 'fuzzy labels' (guaranteed local convergence)

Overall training procedure



References

Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE,* 77(2):257—286, 1989.

Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted finite-state transducers in speech recognition. *Computer Speech & Language,* 16(1):69—88, 2002.

Wendy Holmes. *Speech Synthesis and Recognition.* CRC, December 2001. ISBN 0748408576.

Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition.* Prentice Hall PTR, April 1993. ISBN 0130151572.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall, January 2000. ISBN 0130950696.

Frederick Jelinek. *Statistical Methods for Speech Recognition (Language, Speech, and Communication).* The MIT Press, January 1998. ISBN 0262100665.

Xuedong Huang, Alex Acero, and Hsiao-Wuen Hon. *Spoken Language Processing: A Guide to Theory, Algorithm and System Development*. Prentice Hall PTR, April 2001. ISBN 0130226165.