

# CS 224S / LINGUIST 281 Speech Recognition and Synthesis

Dan Jurafsky

## Lecture 8: Learning HMM parameters: The Baum-Welch Algorithm

IP Notice: Some slides on VQ from John-Paul Hosom at OHSU/OGI

2/5/06

CS 224S Winter 2006

1

## Outline for Today

- **Baum-Welch (EM) training of HMMs**
- **The ASR component of course**
  - 1/30: Hidden Markov Models, Forward, Viterbi Decoding
  - 2/2: **Baum-Welch (EM) training of HMMs**
    - **Start of acoustic model: Vector Quantization**
  - 2/7: Acoustic Model estimation; Gaussians, triphones, etc
  - 2/9: Dealing with Variation: Adaptation, MLLR, etc
  - 2/14: Language Modeling
  - 2/16: More about search in Decoding (Lattices, N-best)
  - ...
  - 3/2: Disfluencies

2/5/06

CS 224S Winter 2006

2

## Reminder: Hidden Markov Models

- **a set of states**
  - $Q = q_1, q_2, \dots, q_N$ : the state at time  $t$  is  $q_t$
- **Transition probability matrix  $A = \{a_{ij}\}$** 

$$a_{ij} = P(q_t = j | q_{t-1} = i) \quad 1 \leq i, j \leq N$$
- **Output probability matrix  $B = \{b_i(k)\}$** 

$$b_i(k) = P(X_t = o_k | q_t = i)$$
- **Special initial probability vector  $\pi$** 

$$\pi_i = P(q_1 = i) \quad 1 \leq i \leq N$$
- **Constraints:**

$$\sum_{j=1}^N a_{ij} = 1; \quad 1 \leq i \leq N \quad \sum_{k=1}^M b_i(k) = 1 \quad \sum_{j=1}^N \pi_j = 1$$

2/5/06

CS 224S Winter 2006

3

## The Three Basic Problems for HMMs

- (From the classic formulation by Larry Rabiner after Jack Ferguson)
- L. R. Rabiner. 1989. A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proc IEEE 77(2), 257-286. Also in Waibel and Lee volume.

2/5/06

CS 224S Winter 2006

4

## The Three Basic Problems for HMMs

- **Problem 1 (Evaluation):** Given the observation sequence  $O = (o_1, o_2, \dots, o_T)$ , and an HMM model  $\Phi = (A, B, \pi)$ , how do we efficiently compute  $P(O | \Phi)$ , the probability of the observation sequence, given the model
- **Problem 2 (Decoding):** Given the observation sequence  $O = (o_1, o_2, \dots, o_T)$ , and an HMM model  $\Phi = (A, B, \pi)$ , how do we choose a corresponding state sequence  $Q = (q_1, q_2, \dots, q_T)$  that is optimal in some sense (i.e., best explains the observations)
- **Problem 3 (Learning):** How do we adjust the model parameters  $\Phi = (A, B, \pi)$  to maximize  $P(O | \Phi)$ ?

2/5/06

CS 224S Winter 2006

From Rabiner

5

## The Learning Problem: Baum-Welch

- **Baum-Welch = Forward-Backward Algorithm (Baum 1972)**
- **Is a special case of the EM or Expectation-Maximization algorithm (Dempster, Laird, Rubin)**
- **The algorithm will let us train the transition probabilities  $A = \{a_{ij}\}$  and the emission probabilities  $B = \{b_i(o_j)\}$  of the HMM**

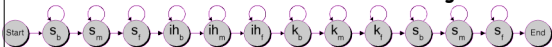
2/5/06

CS 224S Winter 2006

6

## The Learning Problem: Caveats

- Network structure of HMM is always created by hand
  - no algorithm for double-induction of optimal structure and probabilities has been able to beat simple hand-built structures.
  - Always Bakis network = links go forward in time
  - Subcase of Bakis net: beads-on-string net:



- Baum-Welch only guaranteed to return local max, rather than global optimum

2/5/06

CS 224S Winter 2006

7

## Starting out with Observable Markov Models

- How to train?
- Run the model on the observation sequence  $O$ .
- Since it's not hidden, we know which states we went through, hence which transitions and observations were used.
- Given that information, training:
  - $B = \{b_k(o_t)\}$ : Since every state can only generate one observation symbol, observation likelihoods  $B$  are all 1.0
  - $A = \{a_{ij}\}$ :

$$a_{ij} = \frac{C(i \rightarrow j)}{\sum_{q \in Q} C(i \rightarrow q)}$$

2/5/06

CS 224S Winter 2006

8

## Extending Intuition to HMMs

- For HMM, cannot compute these counts directly from observed sequences
- Baum-Welch intuitions:
  - Iteratively estimate the counts.
    - Start with an estimate for  $a_{ij}$  and  $b_k$ , iteratively improve the estimates
  - Get estimated probabilities by:
    - computing the forward probability for an observation
    - dividing that probability mass among all the different paths that contributed to this forward probability

2/5/06

CS 224S Winter 2006

9

## Review: The Forward Algorithm

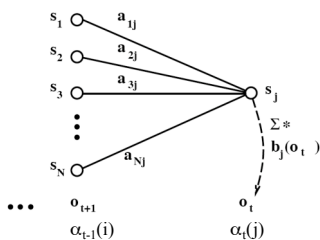
- The Idea: Fold these exponential paths into a simple trellis, so that all possible paths will remerge into  $N$  states at every time slice.
- We define the forward probability as follows:  $\alpha_t(i) = P(o_1 o_2 \dots o_t, q_t = i | \Phi)$
- this is the probability that the HMM  $\Phi$  is in state  $i$  at time  $t$  having generated partial observation  $O_t$ .
- We compute it by induction:
  - Initialization:  $\alpha_1(i) = \pi_i P(o_1 | q_1), 1 \leq i \leq N$
  - (equivalently:  $\alpha_1(i) = \pi_i b_i(o_1), 1 \leq i \leq N$ )
  - Induction:

$$\alpha_t(j) = \left[ \sum_{i=1}^N \alpha_{t-1}(i) a_{ij} b_j(o_t) \right], \quad 2 \leq t \leq T, 1 \leq j \leq N \quad (3)$$

- Termination:  $P(O | \Phi) = \sum_{i=1}^N \alpha_T(i)$

## The inductive step, from Rabiner and Juang

- Computation of  $\alpha_t(j)$  by summing all previous values  $\alpha_{t-1}(i)$  for all  $i$



2/5/06

CS 224S Winter 2006

11

## The Backward algorithm

- We define the backward probability as follows:

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, \dots, o_T, | q_t = i, \Phi)$$

- This is the probability of generating partial observations  $O_{t+1}^T$  from time  $t+1$  to the end, given that the HMM is in state  $i$  at time  $t$  and of course given  $\Phi$ .

- We compute it by induction:

- Initialization:

$$\beta_T(i) = \frac{1}{N}, \quad 1 \leq i \leq N$$

$$\beta_t(i) = \sum_{j=1}^N a_{ij} b_j(o_{t+1}) \beta_{t+1}(j), \quad t = T-1, \dots, 1, 1 \leq i \leq N$$

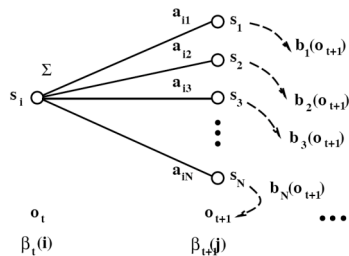
2/5/06

CS 224S Winter 2006

12

## Inductive step of the backward algorithm (figure after Rabiner and Juang)

Computation of  $\beta_t(i)$  by weighted sum of all successive values  $\beta_{t+1}$



2/5/06

CS 224S Winter 2006

13

## Intuition for re-estimation of $a_{ij}$

• We will estimate  $\hat{a}_{ij}$  via this intuition:

$$\hat{a}_{ij} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i}$$

• Numerator intuition:

- Assume we had some estimate of probability that a given transition  $i \rightarrow j$  was taken at time  $t$  in observation sequence.
- If we knew this probability for each time  $t$ , we could sum over all  $t$  to get expected value (count) for  $i \rightarrow j$ .

2/5/06

CS 224S Winter 2006

14

## Re-estimation of $a_{ij}$

• Let  $\gamma_t$  be the probability of being in state  $i$  at time  $t$  and state  $j$  at time  $t+1$ , given  $O_{1..T}$  and model  $\Phi$ :

$$\gamma_t(i, j) = P(q_t = i, q_{t+1} = j | O, \Phi)$$

• We can compute  $\gamma$  from not-quite- $\gamma$ , which is:

$$\text{not\_quite\_}\gamma_t(i, j) = P(q_t = i, q_{t+1} = j_s) | \Phi$$

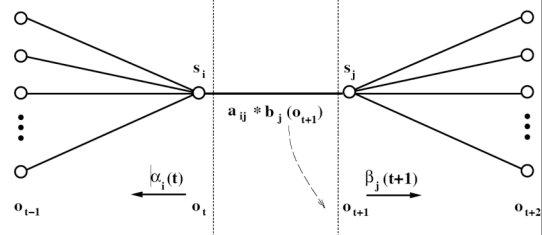
2/5/06

CS 224S Winter 2006

15

## Computing not-quite- $\gamma$

The four components of  $P(q_t = i, q_{t+1} = j_s) | \Phi$ :  $\alpha, \beta, a_{ij}$  and  $b_j(o_t)$



2/5/06

CS 224S Winter 2006

16

## From not-quite- $\gamma$ to $\gamma$

$$\text{not-quite-}\gamma_t(i, j) = \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \quad (8)$$

$$\gamma_t(i, j) = P(q_t = i, q_{t+1} = j | O, \Phi) \quad (9)$$

$$\text{not-quite-}\gamma_t(i, j) = P(q_t = i, q_{t+1} = j, O | \Phi) \quad (10)$$

$$P(X|O, \Phi) = \frac{P(X, O | \Phi)}{P(O | \Phi)} \quad (11)$$

$$P(O | \Phi) = \alpha_T(N) = \beta_T(1) = \sum_{j=1}^N \alpha_t(j) \beta_{t+1}(j) \quad (12)$$

$$\gamma_t(i, j) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\alpha_T(N)} \quad (13)$$

## From $\gamma$ to $a_{ij}$

•  $\hat{a}_{ij} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i}$

• The expected number of transitions from state  $i$  to state  $j$  is the sum over all  $t$  of  $\gamma$ .

• The total expected number of transitions out of state  $i$  is the sum over all transitions out of state  $i$ .

• Final formula for reestimated  $a_{ij}$ :

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \gamma_t(i, j)}{\sum_{t=1}^{T-1} \sum_{j=1}^N \gamma_t(i, j)} \quad (14)$$

## Re-estimating the observation likelihood b

- This is the probability of a given symbol  $v_k$  from the observation vocabulary  $V$ , given a state  $j$ :  $\hat{b}_j(v_k)$ .

$$\hat{b}_j(v_k) = \frac{\text{expected number of times in state } j \text{ and observing symbol } v_k}{\text{expected number of times in state } j}$$

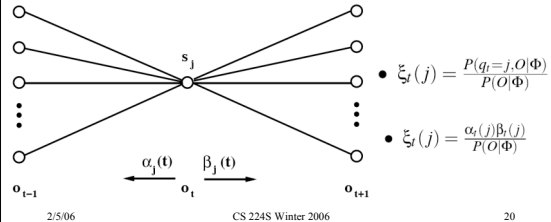
- For this we will need to know the probability of being in state  $j$  at time  $t$ , which we will call  $\xi_t(j)$  ( $\xi$  for state):
- $\xi_t(j) = P(q_t = j | O, \Phi)$
- We compute this by including the observation sequence in the probability and then normalizing:

$$\xi_t(j) = \frac{P(q_t = j, O | \Phi)}{P(O | \Phi)}$$

19

## Computing $\xi$

Computation of  $\xi_j(t)$ , the probability of being in state  $j$  at time  $t$ .



## Reestimating the observation likelihood b

$$\hat{b}_j(v_k) = \frac{\text{expected number of times in state } j \text{ and observing symbol } v_k}{\text{expected number of times in state } j}$$

- For numerator, sum  $\xi_j(t)$  for all  $t$  in which  $o_t$  is symbol  $v_k$ :

$$\hat{b}_j(v_k) = \frac{\sum_{t=1}^T 1_{s.t. O_t = v_k} \xi_j(t)}{\sum_{t=1}^T \xi_j(t)}$$

2/5/06

CS 224S Winter 2006

21

## Summary

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \gamma_t(i, j)}{\sum_{t=1}^{T-1} \sum_{j=1}^N \gamma_t(i, j)}$$

The ratio between the expected number of transitions from state  $i$  to  $j$  and the expected number of all transitions from state  $i$

$$\hat{b}_j(v_k) = \frac{\sum_{t=1}^T 1_{s.t. O_t = v_k} \xi_j(t)}{\sum_{t=1}^T \xi_j(t)}$$

The ratio between the expected number of times the observation data emitted from state  $j$  is  $v_k$ , and the expected number of times any observation is emitted from state  $j$

2/5/06

CS 224S Winter 2006

22

## Summary: Forward-Backward Algorithm

- 1) Initialize  $\Phi = (A, B, \pi)$
- 2) Compute  $\alpha, \beta, \xi$
- 3) Estimate new  $\Phi' = (A, B, \pi)$
- 4) Replace  $\Phi$  with  $\Phi'$
- 5) If not converged go to 2

2/5/06

CS 224S Winter 2006

23

## Some History

- First DARPA Program, 1971-1976
- 3 systems were similar
  - Initial hard decision making
    - Input separated into phonemes using heuristics
    - Strings of phonemes replaced with word candidates
    - Sequences of words scored by heuristics
    - Lots of hand-written rule
- 4th system, Harpy (Jim Baker) was different
  - Simple finite-state network
  - That could be trained statistically!

2/5/06

CS 224S Winter 2006  
Thanks to Picheny/Chen/Noek/Eide

24

## 1972-1984: IBM and related work: 3 big ideas that changed ASR

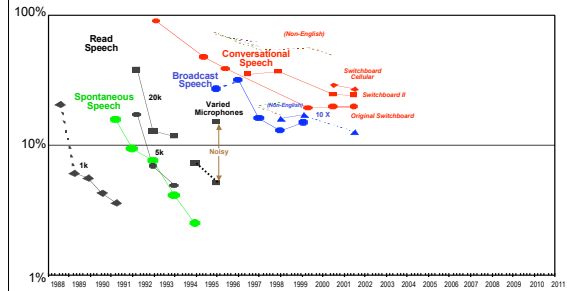
- 1) Idea of HMM
  - IBM (Jelinek, Bahl, etc)
  - independently, Baker in Dragon at CMU
  - Big idea: optimize system parameters on data!
- 2) Idea to eliminate hard decisions about phones: instead, frame-based and soft decisions
- 3) Idea to capture all language information by simple sequences of bigram/trigram rather than hand-constructed grammars

2/5/06

CS 224S Winter 2006

25

## Second DARPA program 1986-1998: NIST benchmarks



2/5/06

CS 224S Winter 2006

## New ideas each year

(table from Chen/Nock/Picheny/Ellis)

1988	1989	1990	1991	1992	1993	1994	1995
•CD HMM	•Multiple Codebooks	•Tied Mixtures	•Double Deltas	•PTMs	•STMs	•MLLR	•SAT
•Modeling	•Modeling	•Modeling	•Sig Proc	•Modeling	•Data	•Adaptation	•Adaptation
							•Sig Proc

1996	1997	1998	1999	2000	2001	2002	2003
•Multiple Models	•MLLT, BIC	•ROVER	•IMLLR-SAT	•MMI	FSTs	•MPE	•Data
•VTLN	•Data	•Data					
•Data							
•Modeling	•Modeling	•Decoding	•Adaptation	•Training	•Modeling	•Training	•Data
•Adaptation	•Data	•Data					
•Data							

2/5/06

CS 224S Winter 2006

27

## Databases

- Read speech (wideband, head-mounted mike)
  - Resource Management (RM)
    - 1000 word vocabulary, used in the 80s
  - WSJ (Wall Street Journal)
    - Reporters read the paper out loud
    - "Verbalized punctuation" or "non-verbalized punctuation"
- Broadcast Speech (wideband)
  - Broadcast News ("Hub 4")
    - English, Mandarin, Arabic
- Conversational Speech (telephone)
  - Switchboard
  - CallHome
  - Fisher

2/5/06

CS 224S Winter 2006

28

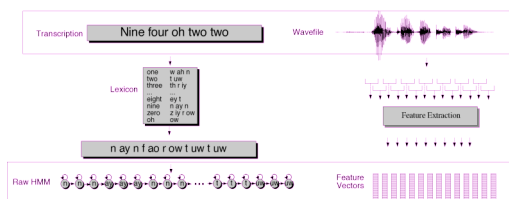
## Summary

- We learned the Baum-Welch algorithm for learning the A and B matrices of an individual HMM
- It doesn't require training data to be labeled at the state level; all you have to know is that an HMM covers a given sequence of observations, and you can learn the optimal A and B parameters for this data by an iterative process.

2/5/06

CS 224S Winter 2006

29



2/5/06

CS 224S Winter 2006

30

## Now: HMMs for speech continued

- How can we apply the Baum-Welch algorithm to speech?
- For today, we'll show some strong simplifying assumptions
- On Tuesday, we'll relax these assumptions and show the general case of learning GMM acoustic models and HMM parameters simultaneously

2/5/06

CS 224S Winter 2006

31

## Problem: how to apply HMM model to continuous observations?

- We have assumed that the output alphabet  $V$  has a finite number of symbols
- But spectral feature vectors are real-valued!
- How to deal with real-valued features?

2/5/06

CS 224S Winter 2006

32

## Vector Quantization

- Idea: Make MFCC vectors look like symbols that we can count
- By building a mapping function that maps each input vector into one of a small number of symbols
- Then compute probabilities just by counting
- This is called Vector Quantization or VQ
- Not used for ASR any more; too simple
- But is useful to consider as a starting point.

2/5/06

CS 224S Winter 2006

33

## Vector Quantization

- Create a training set of feature vectors
- Cluster them into a small number of classes
- Represent each class by a discrete symbol
- For each class  $v_k$ , we can compute the probability that it is generated by a given HMM state using Baum-Welch as above

2/5/06

CS 224S Winter 2006

34

## VQ

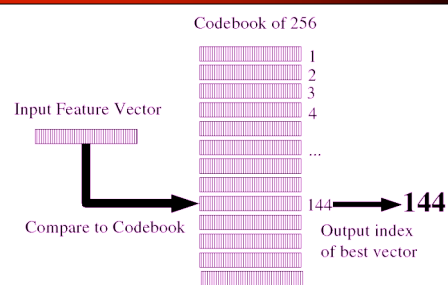
- We'll define a
  - Codebook, which lists for each symbol
  - A prototype vector, or codeword
- If we had 256 classes ('8-bit VQ'),
  - A codebook with 256 prototype vectors
  - Given an incoming feature vector, we compare it to each of the 256 prototype vectors
  - We pick whichever one is closest (by some 'distance metric')
  - And replace the input vector by the index of this prototype vector

2/5/06

CS 224S Winter 2006

35

## VQ



2/5/06

CS 224S Winter 2006

36

## VQ requirements

- A **distance metric** or **distortion metric**
  - Specifies how similar two vectors are
  - Used:
    - to build clusters
    - To find prototype vector for cluster
    - And to compare incoming vector to prototypes
- A **clustering algorithm**
  - K-means, etc.

2/5/06

CS 224S Winter 2006

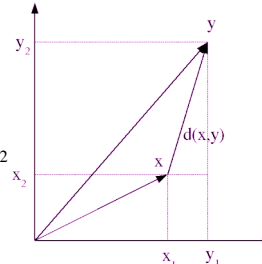
37

## Distance metrics

- **Simplest:**
  - (square of) Euclidean distance

$$d^2(x, y) = \sum_{i=1}^D (x_i - y_i)^2$$

- Also called 'sum-squared error'



2/5/06

CS 224S Winter 2006

38

## Distance metrics

- **More sophisticated:**
  - (square of) Mahalanobis distance
  - Assume that each dimension of feature vector has variance  $\sigma^2$

$$d^2(x, y) = \sum_{i=1}^D \frac{(x_i - y_i)^2}{\sigma_i^2}$$

- Equation above assumes diagonal covariance matrix; more on this later

2/5/06

CS 224S Winter 2006

39

## Training a VQ system (generating codebook): K-means clustering

1. **Initialization**  
choose  $M$  vectors from  $L$  training vectors (typically  $M=2^B$ ) as initial code words... random or max. distance.
2. **Search:**  
for each training vector, find the closest code word, assign this training vector to that cell
3. **Centroid Update:**  
for each cell, compute centroid of that cell. The new code word is the centroid.
4. **Repeat (2)-(3) until average distance falls below threshold (or no change)**

2/5/06

CS 224S Winter 2006

40

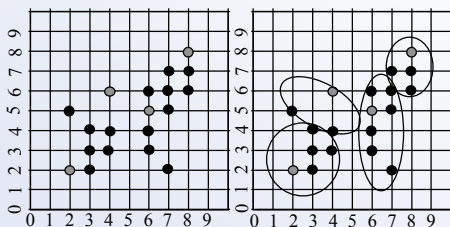
Slide from John-Paul Hosum, OHSU/OGI

### Vector Quantization

Slide thanks to John-Paul Hosum, OHSU/OGI

#### • Example

Given data points, split into 4 codebook vectors with initial values at (2,2), (4,6), (6,5), and (8,8)



2/5/06

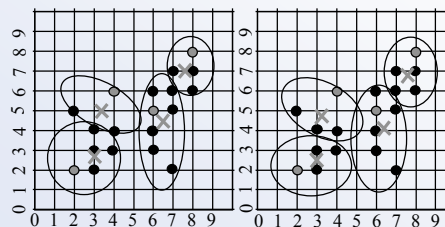
41

### Vector Quantization

Slide from John-Paul Hosum, OHSU/OGI

#### • Example

compute centroids of each codebook, re-compute nearest neighbor, re-compute centroids...



2/5/06

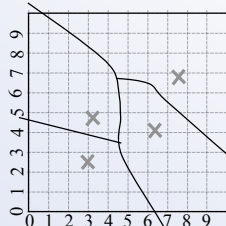
42

## Vector Quantization

Slide from John-Paul Hosum, OHSU/OGI

### • Example

Once there's no more change, the feature space will be partitioned into 4 regions. Any input feature can be classified as belonging to one of the 4 regions. The entire codebook can be specified by the 4 centroid points.



2/5/06

43

## Summary: VQ

- To compute  $p(o_t|q_j)$ 
  - Compute distance between feature vector  $o_t$ 
    - and each codeword (prototype vector)
    - in a preclustered codebook
    - where distance is either
      - Euclidean
      - Mahalanobis
  - Choose the vector that is the closest to  $o_t$ 
    - and take its codeword  $v_k$
  - And then look up the likelihood of  $v_k$  given HMM state  $j$  in the B matrix
- $B_j(o_t) = b_j(v_k)$  s.t.  $v_k$  is codeword of closest vector to  $o_t$
- Using Baum-Welch as above

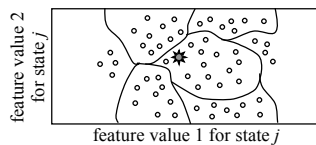
2/5/06

CS 224S Winter 2006

44

## Computing $b_j(v_k)$

Slide from John-Paul Hosum, OHSU/OGI



$$b_j(v_k) = \frac{\text{number of vectors with codebook index } k \text{ in state } j}{\text{number of vectors in state } j} = \frac{14}{56} = \frac{1}{4}$$

2/5/06

CS 224S Winter 2006

45

## Viterbi training

- Baum-Welch training says:
  - We need to know what state we were in, to accumulate counts of a given output symbol  $o_t$ .
  - We'll compute  $\xi_i(t)$ , the probability of being in state  $i$  at time  $t$ , by using forward-backward to sum over all possible paths that might have been in state  $i$  and output  $o_t$ .
- Viterbi training says:
  - Instead of summing over all possible paths, just take the single most likely path
  - Use the Viterbi algorithm to compute this "Viterbi" path
  - Via "forced alignment"

2/5/06

CS 224S Winter 2006

46

## Forced Alignment

- Computing the "Viterbi path" over the training data is called "forced alignment"
- Because we know which word string to assign to each observation sequence.
- We just don't know the state sequence.
- So we use  $a_{ij}$  to constrain the path to go through the correct words
- And otherwise do normal Viterbi
- Result: state sequence!

2/5/06

CS 224S Winter 2006

47

## Viterbi training equations

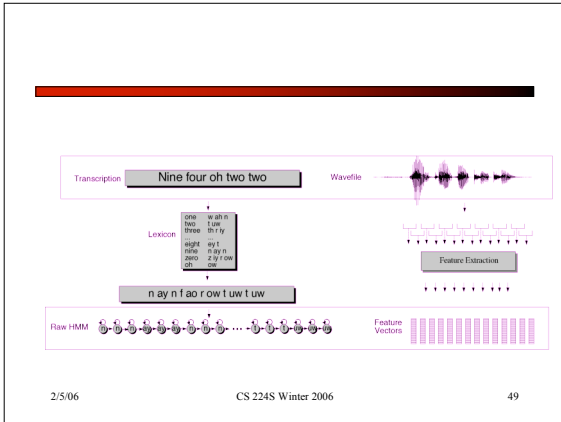
Viterbi	Baum-Welch
$\hat{a}_{ij} = \frac{n_{ij}}{n_i}$ <p style="font-size: small;">For all pairs of emitting states, <math>1 \leq i, j \leq N</math></p>	$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \gamma(i, j)}{\sum_{t=1}^{T-1} \sum_{j=1}^N \gamma(i, j)}$
$\hat{b}_j(v_k) = \frac{n_j(s.t. O_t = v_k)}{n_j}$	$\hat{b}_j(v_k) = \frac{\sum_{t=1}^T 1_{s.t. O_t = v_k} \xi_j(t)}{\sum_{t=1}^T \xi_j(t)}$

Where  $n_{ij}$  is number of frames with transition from  $i$  to  $j$  in best path  
And  $n_j$  is number of frames where state  $j$  is occupied

2/5/06

CS 224S Winter 2006

48



## Viterbi Training

---

- Much faster than Baum-Welch
- But doesn't work quite as well
- But the tradeoff is often worth it.

2/5/06 CS 224S Winter 2006 50

## Summary

---

- Baum-Welch for learning HMM parameters
- Acoustic Modeling:
  - VQ doesn't work well for ASR; I mentioned it only because it is useful to think of pedagogically.
  - What we actually do is using GMMs; Gaussian Mixture Models.
  - We will learn these, how to train them, and how they fit into EM on Tuesday

2/5/06 CS 224S Winter 2006 51