

CS 224S / LINGUIST 236 Speech Recognition and Synthesis

Dan Jurafsky

Lecture 9: Acoustic Modeling

IP Notice: Some of these slides were derived from Andrew Ng's CS 229 notes, as well as lecture notes from Chen, Picheny et al, and Bryan Pellom. I'll try to give credit on each slide, but may not be completely successful.

1/31/05

CS 224S Winter 2005

1

Summary from Last Time

- 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.

1/31/05

CS 224S Winter 2005

2

Outline for Today

- Baum-Welch (EM) training of HMMs
- The ASR component of course
 - 1/24: Hidden Markov Models, Forward, Viterbi Decoding
 - 1/26: Baum-Welch (EM) training of HMMs
 - Start of acoustic model: Vector Quantization, Gaussians
 - 2/1: [Acoustic Model estimation: Gaussians, triphones, etc](#)
 - 2/3: Advanced Issues in Acoustic Mod.: Guest Lecture
 - 2/8: Language Modeling: Lecture by Rion
 - 2/10: Advanced Issues in Decoding Search

1/31/05

CS 224S Winter 2005

3

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?
 - Decoding: Given o_t , how to compute $P(o_t|q)$
 - Learning: How to modify EM to deal with real-valued features

1/31/05

CS 224S Winter 2005

4

Outline for Today

- Increasingly sophisticated models
- Acoustic Likelihood for each state:
 - Gaussians
 - Multivariate Gaussians
 - Mixtures of Multivariate Gaussians
- Where a state is progressively:
 - CI Subphone (3ish per phone)
 - CD phone (=triphones)
 - State-tying of CD phone
- Forward-Backward Training
- Viterbi training

1/31/05

CS 224S Winter 2005

5

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.

1/31/05

CS 224S Winter 2005

6

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

1/31/05

CS 224S Winter 2005

7

VQ

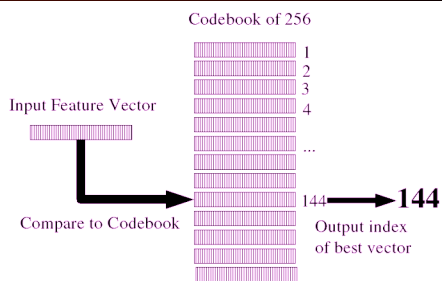
- We'll define a
 - Codebook, which lists for each symbol
 - A prototype vector, or codeword
- If we had 256 classes ('8-bit VQ'),
 - A codebook with 256 prototype vectors
 - Given an incoming feature vector, we compare it to each of the 256 prototype vectors
 - We pick whichever one is closest (by some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

1/31/05

CS 224S Winter 2005

8

VQ



1/31/05

CS 224S Winter 2005

9

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.

1/31/05

CS 224S Winter 2005

10

Distance metrics

- **Simplest:**
 - Euclidean distance
$$d(x, y) = \sum_{i=1}^D (x_i - y_i)^2$$
 - Also called 'sum-squared error'

1/31/05

CS 224S Winter 2005

11

Distance metrics

- **More sophisticated:**
 - Mahalanobis distance
 - Assume that each dimension of feature vector has variance σ^2

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

1/31/05

CS 224S Winter 2005

12

Summary: VQ

- To deal with real-valued input
- Convert the input to a symbol
- By choosing closest prototype vector in a preclustered codebook
- Where 'closest' is defined by:
 - Euclidean distance
 - Mahalanobis distance
- Learning:
 - Do VQ and then use Baum-Welch to assign probabilities to each symbol
- Decoding:
 - Do VQ and then use the symbol probabilities in decoding

1/31/05

CS 224S Winter 2005

Directly Modeling Continuous Observations

- Gaussians
 - Univariate Gaussians
 - Baum-Welch for univariate Gaussians
 - Multivariate Gaussians
 - Baum-Welch for multivariate Gaussians
 - Gaussian Mixture Models (GMMs)
 - Baum-Welch for GMMs

1/31/05

CS 224S Winter 2005

14

Better than VQ

- VQ is insufficient for real ASR
- Instead: Assume the possible values of the observation feature vector o_t are normally distributed.
- Represent the observation likelihood function $b_j(o_t)$ as a Gaussian with mean μ_j and variance σ_j^2

$$f(x | \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

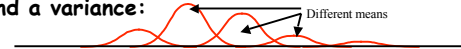
1/31/05

CS 224S Winter 2005

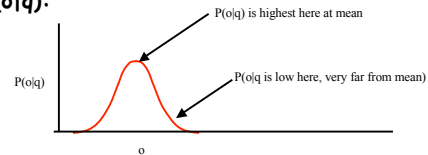
15

Gaussians for Acoustic Modeling

A Gaussian is parameterized by a mean and a variance:



- $P(o|q)$:



1/31/05

CS 224S Winter 2005

16

Gaussian PDFs

- A Gaussian is a probability density function; probability is area under curve.
- We will be using point estimates; value of Gaussian at point;
- Technically these are not probabilities, since need to be multiplied by dx
- As we will see later, this is ok since same value is omitted from all Gaussians, so argmax is still correct.

1/31/05

CS 224S Winter 2005

17

Training a Univariate Gaussian

- A (single) Gaussian is characterized by a mean and a variance
- Imagine that we had some training data in which each state was labeled
- We could just compute the mean and variance from the data:

$$\mu_i = \frac{1}{T} \sum_{t=1}^T o_t \text{ s.t. } o_t \text{ is state } i$$

$$\sigma_i^2 = \frac{1}{T} \sum_{t=1}^T (o_t - \mu_i)^2 \text{ s.t. } o_t \text{ is state } i$$

1/31/05

CS 224S Winter 2005

18

Training Univariate Gaussians

- But we don't know which observation was produced by which state!
- What we want: to assign each observation vector o_t to every possible state i , prorated by the probability the HMM was in state i at time t .
- The probability of being in state i at time t is $\xi_t(i)$!

$$\bar{\mu}_i = \frac{\sum_{t=1}^T \xi_t(i) o_t}{\sum_{t=1}^T \xi_t(i)} \quad \bar{\sigma}_i^2 = \frac{\sum_{t=1}^T \xi_t(i) (o_t - \mu_i)^2}{\sum_{t=1}^T \xi_t(i)}$$

1/31/05

CS 224S Winter 2005

19

Multivariate Gaussians

- Instead of a single mean μ and variance σ :

$$f(x | \mu, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right)$$

- Vector of means μ and covariance matrix Σ

$$f(x | \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)\right)$$

1/31/05

CS 224S Winter 2005

20

Multivariate Gaussians

- Defining μ and Σ

$$\mu = E(x)$$

$$\Sigma = E[(x - \mu)(x - \mu)^T]$$

- So the i - j th element of Σ is:

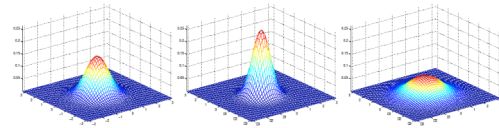
$$\sigma_{ij}^2 = E[(x_i - \mu_i)(x_j - \mu_j)]$$

1/31/05

CS 224S Winter 2005

21

Gaussian Intuitions: Size of Σ



- $\mu = [0 \ 0]$ $\mu = [0 \ 0]$ $\mu = [0 \ 0]$
- $\Sigma = \mathbf{I}$ $\Sigma = 0.6\mathbf{I}$ $\Sigma = 2\mathbf{I}$
- As Σ becomes larger, Gaussian becomes more spread out; as Σ becomes smaller, Gaussian more compressed

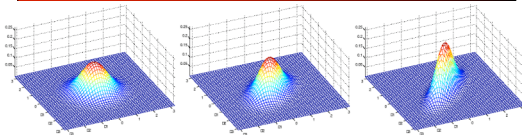
1/31/05

CS 224S Winter 2005

22

Text and figures from Andrew Ng's lecture notes for CS229

Gaussian Intuitions: Off-diagonal



$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}; \quad \Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

- As we increase the off-diagonal entries, more correlation between value of x and value of y

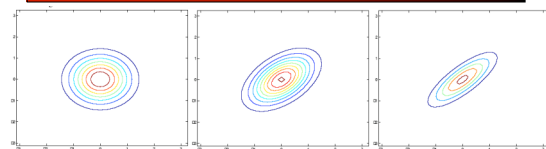
1/31/05

CS 224S Winter 2005

23

Text and figures from Andrew Ng's lecture notes for CS229

Gaussian Intuitions: off-diagonal



$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}; \quad \Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

- As we increase the off-diagonal entries, more correlation between value of x and value of y

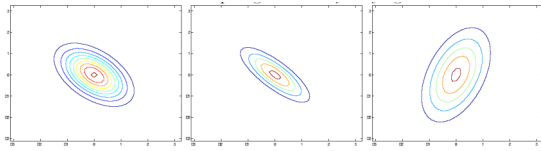
1/31/05

CS 224S Winter 2005

24

Text and figures from Andrew Ng's lecture notes for CS229

Gaussian Intuitions: off-diagonal and diagonal



$$\Sigma = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}; \Sigma = \begin{bmatrix} 1 & -0.8 \\ -0.8 & 1 \end{bmatrix}; \Sigma = \begin{bmatrix} 3 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

- Decreasing non-diagonal entries (#1-2)
- Increasing variance of one dimension in diagonal (#3)

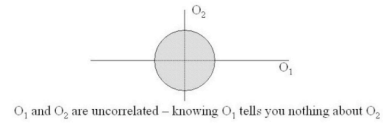
1/31/05

CS 224S Winter 2005

25

Text and figures from Andrew Ng's lecture notes for CS229

In two dimensions



O_1 and O_2 are uncorrelated – knowing O_1 tells you nothing about O_2

O_1 and O_2 can be uncorrelated without having equal variances

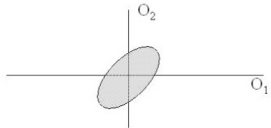
1/31/05

CS 224S Winter 2005

26

From Chen, Picheny et al lecture slides

In two dimensions



O_1 and O_2 are correlated – knowing O_1 tells you something about O_2

1/31/05

CS 224S Winter 2005

27

From Chen, Picheny et al lecture slides

But: assume diagonal covariance

- I.e., assume that the features in the feature vector are uncorrelated
- This isn't true for FFT features, but is true for MFCC features, as we will see.
- Computation and storage much cheaper if diagonal covariance.
- I.e. only diagonal entries are non-zero
- Diagonal contains the variance of each dimension σ_{ii}^2
- So this means we consider the variance of each acoustic feature (dimension) separately

1/31/05

CS 224S Winter 2005

28

Diagonal covariance

- Diagonal contains the variance of each dimension σ_{ii}^2
- So this means we consider the variance of each acoustic feature (dimension) separately

$$f(x | \mu, \sigma) = \prod_{d=1}^D \frac{1}{\sigma_d \sqrt{2\pi}} \exp\left(-\frac{1}{2} \left(\frac{x_d - \mu_d}{\sigma_d}\right)^2\right)$$

$$f(x | \mu, \sigma) = \frac{1}{2\pi^{D/2} \prod_{d=1}^D \sigma_d} \exp\left(-\frac{1}{2} \sum_{d=1}^D \frac{(x_d - \mu_d)^2}{\sigma_d^2}\right)$$

1/31/05

CS 224S Winter 2005

29

Baum-Welch reestimation equations for multivariate Gaussians

- Natural extension of univariate case, where now μ_i is mean vector for state i :

$$\bar{\mu}_i = \frac{\sum_{t=1}^T \xi_t(i) o_t}{\sum_{t=1}^T \xi_t(i)}$$

$$\bar{\Sigma}_i = \frac{\sum_{t=1}^T \xi_t(i) (o_t - \mu_i)(o_t - \mu_i)^T}{\sum_{t=1}^T \xi_t(i)}$$

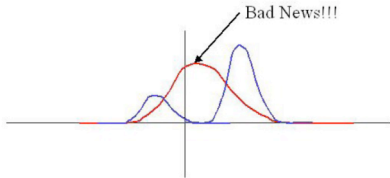
1/31/05

CS 224S Winter 2005

30

But we're not there yet

- **Single Gaussian may do a bad job of modeling distribution in any dimension:**



- **Solution: Mixtures of Gaussians**

1/31/05

CS 224S Winter 2005

31

Figure from Chen, Pichenev et al slides

Mixtures of Gaussians

- **M mixtures of Gaussians:**

$$f(x | \mu_{jk}, \Sigma_{jk}) = \sum_{k=1}^M c_{jk} N(x, \mu_{jk}, \Sigma_{jk})$$

$$b_j(o_t) = \sum_{k=1}^M c_{jk} N(o_t, \mu_{jk}, \Sigma_{jk})$$

- **For diagonal covariance:**

$$b_j(o_t) = \sum_{k=1}^M \frac{c_{jk}}{2\pi^{D/2} \prod_{d=1}^D \sigma_{jkd}^2} \exp\left(-\frac{1}{2} \sum_{d=1}^D \frac{(x_{jkd} - \mu_{jkd})^2}{\sigma_{jkd}^2}\right)$$

1/31/05

CS 224S Winter 2005

32

GMMs

- **Summary: each state has a likelihood function parameterized by:**

- M Mixture weights
- M Mean Vectors of dimensionality D
- Either
 - M Covariance Matrices of DxD
- Or more likely
 - M Diagonal Covariance Matrices of DxD
 - which is equivalent to
 - M Variance Vectors of dimensionality D

1/31/05

CS 224S Winter 2005

33

Baum-Welch for Mixture Models

- By analogy with ξ earlier, let's define the probability of being in state j at time t with the k^{th} mixture component accounting for o_t :

$$\xi_{jm}(j) = \frac{\sum_{t=1}^N \alpha_{t-1}(j) a_{ij} c_{jm} b_{jm}(o_t) \beta_j(t)}{\alpha_F(T)}$$

- Now,

$$\bar{\mu}_{jm} = \frac{\sum_{t=1}^T \xi_{jm}(j) o_t}{\sum_{t=1}^T \sum_{k=1}^M \xi_{jk}(j)} \quad \bar{c}_{jm} = \frac{\sum_{t=1}^T \xi_{jm}(j)}{\sum_{t=1}^T \sum_{k=1}^M \xi_{jk}(j)} \quad \bar{\Sigma}_{jm} = \frac{\sum_{t=1}^T \xi_{jm}(j) (o_t - \mu_j)(o_t - \mu_j)^T}{\sum_{t=1}^T \sum_{k=1}^M \xi_{jk}(j)}$$

1/31/05

CS 224S Winter 2005

34

How to train mixtures?

- Choose M (often 16; or can tune M optimally)
- Then can do various splitting or clustering algorithms
- One simple method for "splitting":
 - 1) Compute global mean μ and global variance
 - 2) Split into two Gaussians, with means $\mu \pm \epsilon$ (sometimes ϵ is 0.2σ)
 - 3) Run Forward-Backward to retrain
 - 4) Go to 2 until we have 16 mixtures

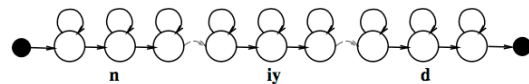
1/31/05

CS 224S Winter 2005

35

Return to Viterbi for speech

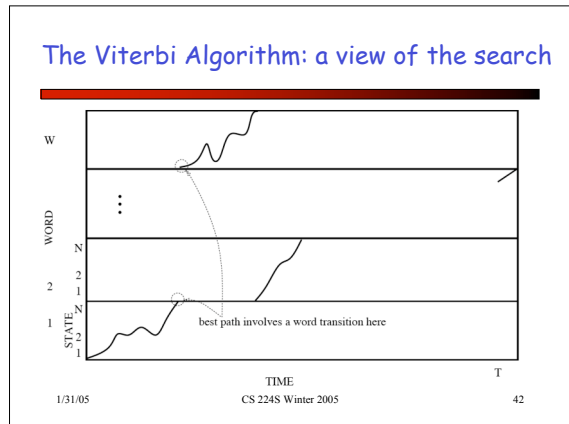
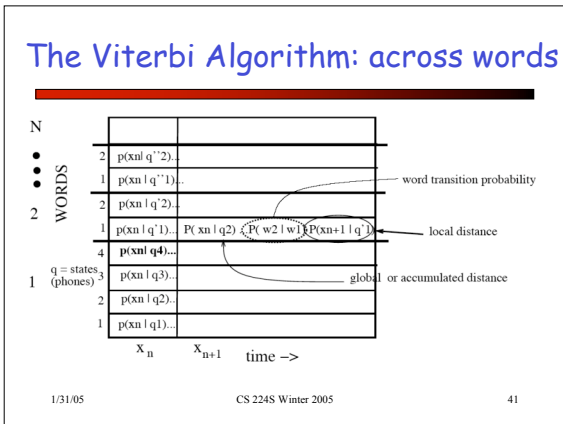
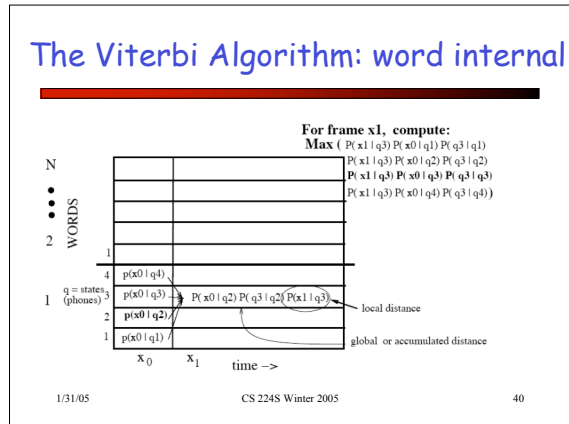
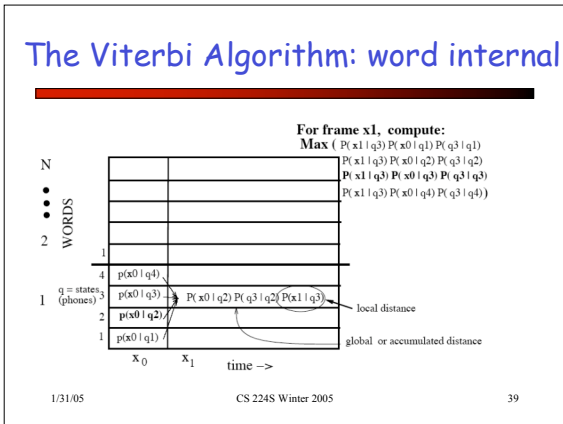
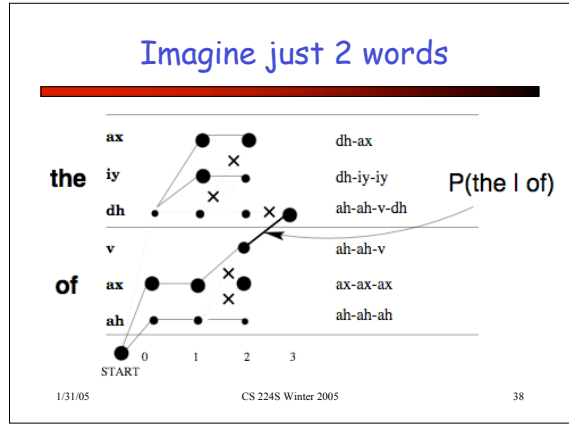
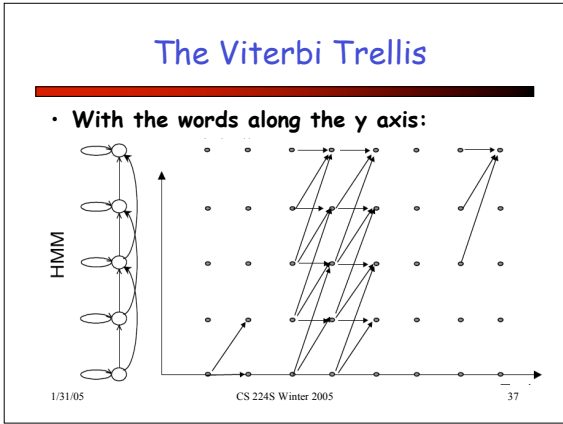
- Remember that an HMM model for a word looks like this:



1/31/05

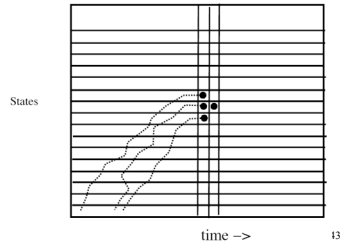
CS 224S Winter 2005

36



The Viterbi Algorithm: backtracing

- When advancing to next column, extend backtrace of most likely path



1/31/05

43

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

1/31/05

CS 224S Winter 2005

44

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!**

1/31/05

CS 224S Winter 2005

45

Viterbi training (II)

- Equations for non-mixture Gaussians

$$\bar{\mu}_i = \frac{1}{N_i} \sum_{t=1}^T o_t \quad \text{s.t. } q_t = i$$

$$\bar{\sigma}_i^2 = \frac{1}{N_i} \sum_{t=1}^T (o_t - \mu_i)^2 \quad \text{s.t. } q_t = i$$

- Viterbi training for mixture Gaussians is more complex, generally just assign each observation to 1 mixture

1/31/05

CS 224S Winter 2005

46

Initialization: "Flat start"

- Transition probabilities:**
 - set to zero any that you want to be "structurally zero"
 - The γ probability computation includes previous value of a_{ij} , so if it's zero it will never change
 - Set the rest to identical values
- Likelihoods:**
 - initialize μ and σ of each state to global mean and variance of all training data

1/31/05

CS 224S Winter 2005

47

Log domain

- In practice, we do all computation in the log domain
- Avoids underflow
 - Instead of multiplying lots of very small probabilities, we add numbers that are not so small.
- What we compute for Gaussian (for diagonal Σ)

$$-\frac{D}{2} \ln(2\pi) - \sum_{d=1}^D \ln \sigma_d - \frac{1}{2} \sum_{d=1}^D \frac{(o_t - \mu_i)^2}{\sigma_d}$$

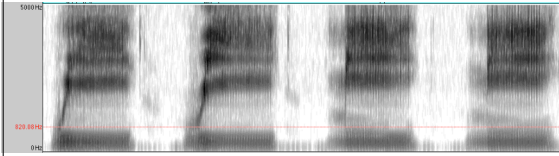
- Note that this looks like a weighted Mahalanobis distance!!!
- Also may justify why we these aren't really probabilities (since we can ignore dx values of pdf); we aren't really computing probabilities, just distances.

1/31/05

CS 224S Winter 2005

48

Modeling phonetic context



W iy r iy m iy n iy

1/31/05

CS 224S Winter 2005

49

Modeling phonetic context

- The strongest factor affecting phonetic variability is the neighboring phone
- How to model that in HMMs?
- Idea: have phone models which are specific to context.
- Instead of Context-Independent (CI) phones
- We'll have Context-Dependent (CD) phones

1/31/05

CS 224S Winter 2005

50

CD phones: triphones

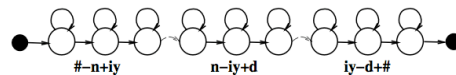
- Triphones
- Each triphone captures facts about preceding and following phone
- Monophone:
 - p, t, k
- Triphone:
 - iy-p+aa
 - a-b+c means "phone b, preceding by phone a, followed by phone c"

1/31/05

CS 224S Winter 2005

51

"Need" with triphone models



1/31/05

CS 224S Winter 2005

52

Word-Boundary Modeling

- Word-Internal Context-Dependent Models
'OUR LIST':
SIL AA+R AA-R L+IH L-IH+S IH-S+T S-T
- Cross-Word Context-Dependent Models
'OUR LIST':
SIL-AA+R AA-R+L R-L+IH L-IH+S IH-S+T S-T+SIL
- Dealing with cross-words makes decoding harder! We will return to this.

1/31/05

CS 224S Winter 2005

53

Implications of Cross-Word Triphones

- Possible triphones: $50 \times 50 \times 50 = 125,000$
- How many triphone types actually occur?
- 20K word WSJ Task, numbers from Bryan Pellom
- Word-internal models: need 14,300 triphones
- Cross-word models: need 54,400 triphones
- But in training data only 22,800 triphones occur!
- Need to generalize models.

1/31/05

CS 224S Winter 2005

54

Solution: State Tying

- Young, Odell, Woodland 1994
- Decision-Tree based clustering of triphone states
- States which are clustered together will share their Gaussians
- We call this "state tying", since these states are "tied together" to the same Gaussian.
- Previous work: generalized triphones
 - Model-based clustering ('model' = 'phone')
 - Clustering at state is more fine-grained

1/31/05

CS 224S Winter 2005

55

State tying/clustering

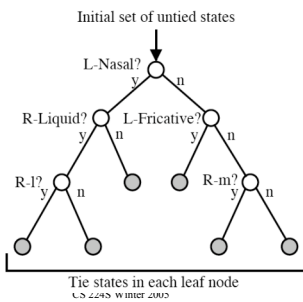
- How do we decide which triphones to cluster together?
- Use **phonetic features** (or 'broad phonetic classes')
 - Stop
 - Nasal
 - Fricative
 - Sibilant
 - Vowel
 - lateral

1/31/05

CS 224S Winter 2005

56

Decision tree for clustering triphones for tying



1/31/05

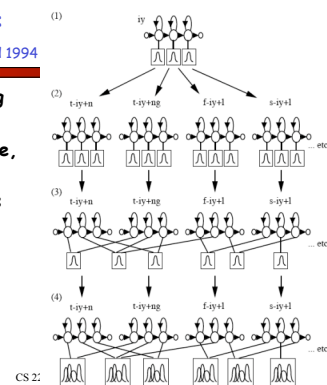
CS 224S Winter 2005

57

State Tying:

Young, Odell, Woodland 1994

- The steps in creating CD phones.
- Start with monophone, do EM training
- Then clone Gaussians into triphones
- Then build decision tree and cluster Gaussians
- Then clone and train mixtures (GMMs)



1/31/05

CS 224S

Other methods for computing $p(o|q)$

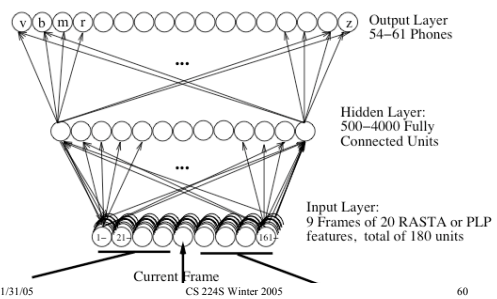
- Neural Nets
- SVMs
- Etc

1/31/05

CS 224S Winter 2005

59

Neural Nets



1/31/05

CS 224S Winter 2005

60

Neural Nets (II)

- Nets give posterior $p(q|o)$ [discriminative, good]
- But we need $p(o|q)$ to fit into HMM

$$p(q|o) = \frac{P(o|q)p(q)}{p(o)}$$

- Rearranging terms to get a likelihood:

$$\nearrow \frac{p(o|q)}{p(o)} = \frac{P(q|o)}{p(q)}$$

- scaled likelihood is ok since $p(o)$ is constant
- No improvement over Gaussians. Lately: SVMs?

1/31/05

CS 224S Winter 2005

61

Summary: Acoustic Modeling for LVCSR.

- Increasingly sophisticated models
- For each state:
 - Gaussians
 - Multivariate Gaussians
 - Mixtures of Multivariate Gaussians
- Where a state is progressively:
 - CI Phone
 - CI Subphone (3ish per phone)
 - CD phone (=triphones)
 - State-tying of CD phone
- Forward-Backward Training
- Viterbi training

1/31/05

CS 224S Winter 2005

62