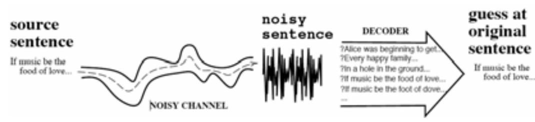
CS 294-5: Statistical Natural Language Processing

Speech Recognition II
Lecture 21: 11/29/05

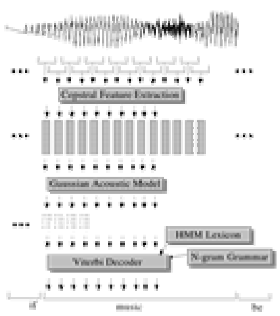
Slides directly from Dan Jurafsky, indirectly many others

The Noisy Channel Model

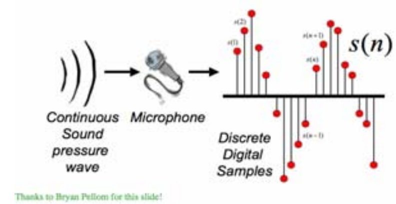


- Search through space of all possible sentences.
- Pick the one that is most probable given the waveform.

Speech Recognition Architecture



Digitizing Speech



Frame Extraction

- A frame (25 ms wide) extracted every 10 ms

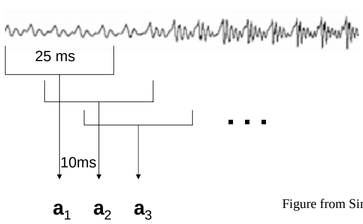


Figure from Simon Arnfield

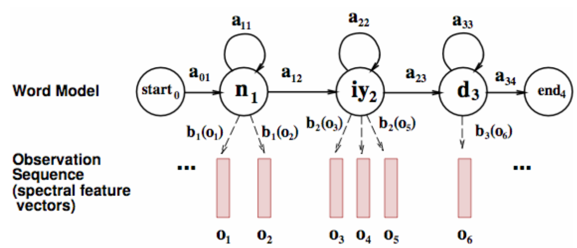
Mel Freq. Cepstral Coefficients

- Do FFT to get spectral information
  - Like the spectrogram/spectrum we saw earlier
- Apply Mel scaling
  - Linear below 1kHz, log above, equal samples above and below 1kHz
  - Models human ear; more sensitivity in lower freqs
- Plus Discrete Cosine Transformation

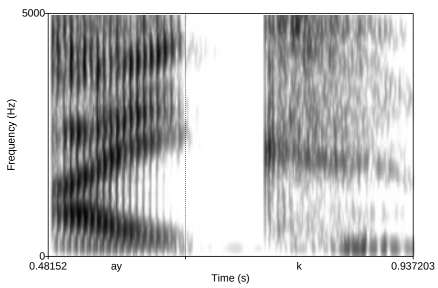
Final Feature Vector

- 39 (real) features per 10 ms frame:
  - 12 MFCC features
  - 12 Delta MFCC features
  - 12 Delta Delta MFCC features
  - 1 (log) frame energy
  - 1 Delta (log) frame energy
  - 1 Delta Delta (log frame energy)
- So each frame is represented by a 39D vector

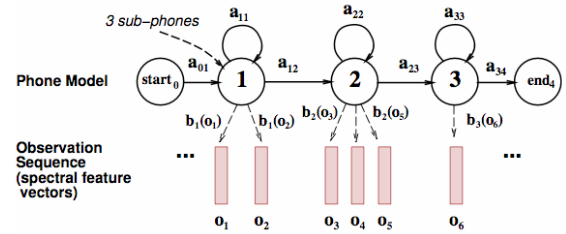
HMMs for Speech



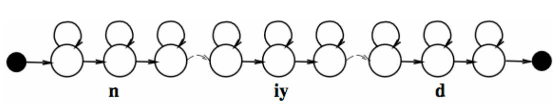
Phones Aren't Homogeneous



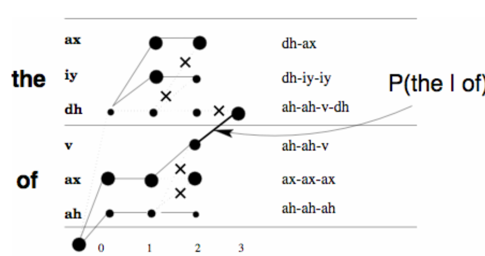
Need to Use Subphones



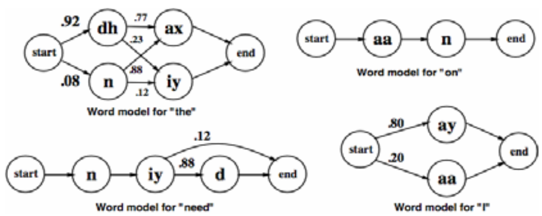
A Word with Subphones



Viterbi Decoding



ASR Lexicon: Markov Models

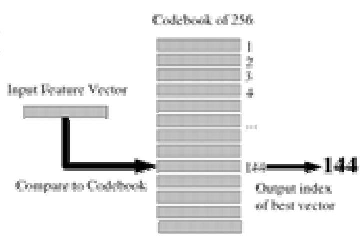


HMMs for Continuous Observations?

- Before: discrete, finite set of observations
- Now: spectral feature vectors are real-valued!
- Solution 1: discretization
- Solution 2: continuous emissions models
  - Gaussians
  - Multivariate Gaussians
  - Mixtures of Multivariate Gaussians
- A state is progressively:
  - Context independent subphone (~3 per phone)
  - Context dependent phone (=triphones)
  - State-tying of CD phone

Vector Quantization

- Idea: discretization
  - Map MFCC vectors onto discrete symbols
  - Compute probabilities just by counting
- This is called Vector Quantization or VQ
- Not used for ASR anymore; too simple
- Useful to consider as a starting point



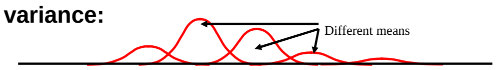
Gaussian Emissions

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

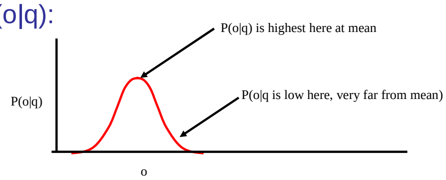
$$f(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{(x-\mu)^2}{2\sigma^2})$$

Gaussians for Acoustic Modeling

A Gaussian is parameterized by a mean and a variance:



- P(o|q):



Multivariate Gaussians

- Instead of a single mean μ and variance σ:

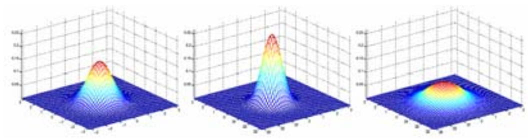
$$f(x \mid \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{(x-\mu)^2}{2\sigma^2})$$

- Vector of means μ and covariance matrix Σ

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

- Usually assume diagonal covariance
  - This isn't very true for FFT features, but is fine for MFCC features

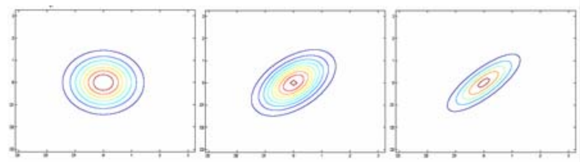
Gaussian Intuitions: Size of Σ



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

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

Gaussians: Off-Diagonal

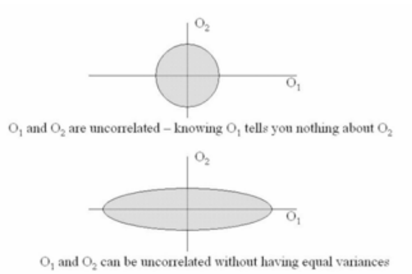


$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}; \quad .\Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

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

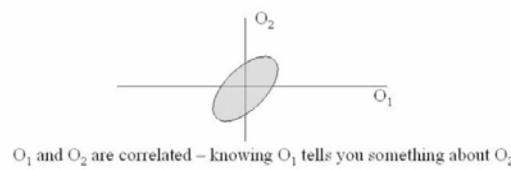
Text and figures from Andrew Ng's lecture notes for CS229

In two dimensions



From Chen, Picheny et al lecture slides

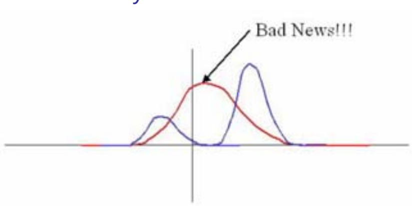
In two dimensions



From Chen, Picheny et al lecture slides

But we're not there yet

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



- Solution: Mixtures of Gaussians

Figure from Chen, Picheney et al slides

Mixtures of Gaussians

- M mixtures of Gaussians:

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

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

- For diagonal covariance:

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

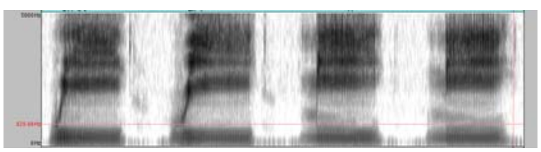
GMMs

- 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

Training Mixture Models

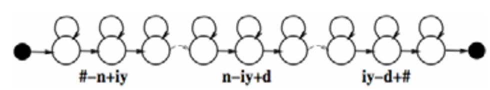
- Forced Alignment
  - Computing the "Viterbi path" over the training data is called "forced alignment"
  - We know which word string to assign to each observation sequence.
  - We just don't know the state sequence.
  - So we constrain the path to go through the correct words
  - And otherwise do normal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

Modeling phonetic context



W iy r iy m iy n iy

"Need" with triphone models

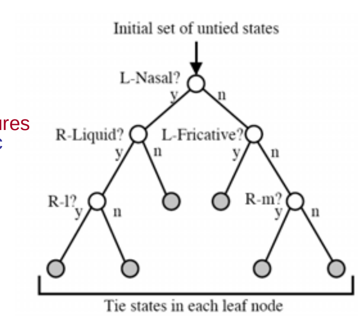


Implications of Cross-Word Triphones

- Possible triphones: 50x50x50=125,000
- How many triphone types actually occur?
- 20K word WSJ Task (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.

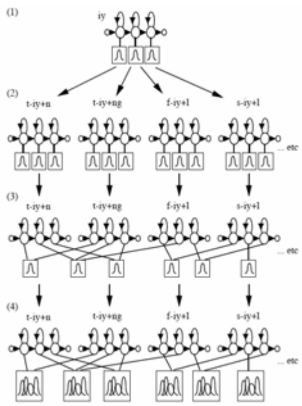
State Tying / Clustering

- [Young, Odell, Woodland 1994]
- How do we decide which triphones to cluster together?
- Use phonetic features (or 'broad phonetic classes')
  - Stop
  - Nasal
  - Fricative
  - Sibilant
  - Vowel
  - lateral



State Tying

- Creating CD phones:
  - Start with monophone, do EM training
  - Clone Gaussians into triphones
  - Build decision tree and cluster Gaussians
  - Clone and train mixtures (GMMs



Viterbi with 2 Words + Unif. LM

- Null transition from the end-state of each word to start-state of all (both) words.

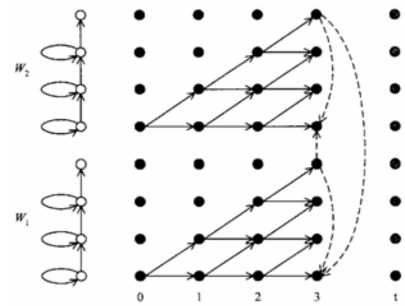


Figure from Huang et al page 612

Search space for unigram LM

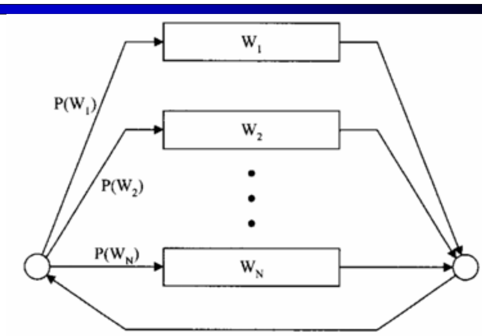


Figure from Huang et al page 617

Search space with bigrams

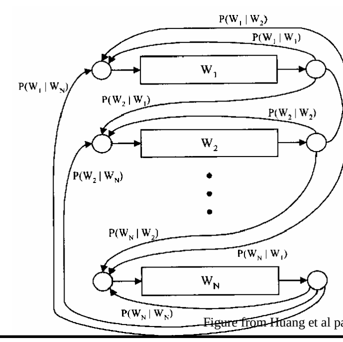


Figure from Huang et al page 618

Speeding things up

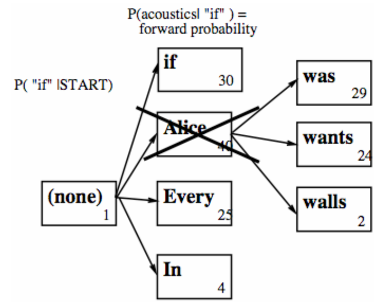
- Viterbi is $O(N^2T)$, where N is total number of HMM states, and T is length
- This is too large for real-time search
- A ton of work in ASR search is just to make search faster:
  - Beam search (pruning)
  - Fast match
  - Tree-based lexicons

Beam Search

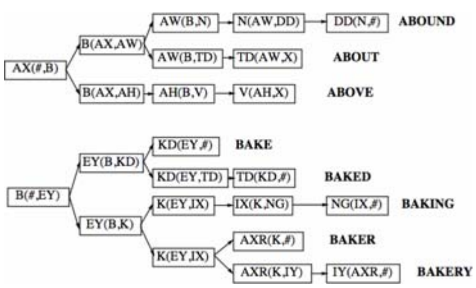
- Most common strategy (still!)
- Just like earlier in the term
- Instead of retaining all candidates at every time frame
  - Use a threshold T to keep subset
  - At each time t
    - Identify state with lowest cost D_{min}
    - Each state with cost > D_{min} + T is discarded ("pruned") before moving on to time t+1
- Empirically, beam size of 5-10% of search space
- 90-95% of HMM states don't have to be considered
- Vast savings in time

A* Decoding (2)

A* Decoding (cont.)



Tree structured lexicon



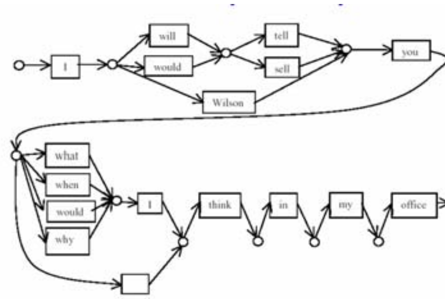
Multipass Search

N-best list

1. I will tell you would I think in my office
2. I will tell you what I think in my office
3. I will tell you when I think in my office
4. I would sell you would I think in my office
5. I would sell you what I think in my office
6. I would sell you when I think in my office
7. I will tell you would I think in my office
8. I will tell you why I think in my office
9. I will tell you what I think on my office
10. I Wilson you I think on my office

From Huang et al, page 664

Word Graph



From Huang et al, page 665

One-pass vs. multipass

- Potential problems with multipass
  - Can't use for real-time (need end of sentence)
    - (But can keep successive passes really fast)
  - Each pass can introduce inadmissible pruning
    - (But one-pass does the same w/beam pruning and fastmatch)
- Why multipass
  - Very expensive KSs. (NL parsing,higher-order n-gram, etc)
  - Spoken language understanding: N-best perfect interface
  - Research: N-best list very powerful offline tools for algorithm development
  - N-best lists needed for discriminant training (MMIE, MCE) to get rival hypotheses