T-61.184
Automatic Speech Recognition: From Theory to Practice

http://www.cis.hut.fi/Opinnot/T-61.184/
October 11, 2004

Prof. Bryan Pellom
Department of Computer Science
Center for Spoken Language Research
University of Colorado

pellom@cslr.colorado.edu

Today

- More about Gaussian Mixture Models (GMMs)
- Sources of Acoustic Variability
- Methods for Acoustic Modeling for Speech Recognition
- State Clustering Methods
  - Bottom-up
  - Top-Down

Resources used for Today's Meeting

- S.J. Young, J.J. Odell, P.C. Woodland, "Tree-based state tying for high accuracy acoustic modeling," *Proc. ARPA Human Language Technology Conference, Plainsboro, NJ, March, 1994*
- *J. Odell, PhD Thesis, University of Cambridge, "The use of context in large vocabulary speech recognition," March 1995*

Gaussian Mixture Model

- **Single-state HMM model**
- **Observation probability a sum of M component Gaussians,**

$$b(\mathbf{o_t}) = \sum_{k=1}^{M} w_k b_k(\mathbf{o_t}, \mu_k, \Sigma_k)$$

$$= \sum_{k=1}^{M} \frac{w_k}{(2\pi)^{d/2} |\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2} (o_t - u_k)' \Sigma_k^{-1} (o_t - u_k) \right)$$

Estimating Gaussian Mixture Models

- **Define probability of *t*th observation being generated by the *k*th mixture component,**

$$P(k \mid o_t, \lambda) = \frac{w_k b_k(o_t)}{\sum_{j=1}^{M} w_j b_j(o_t)}$$

- **Note that "k" (b_k) here refers to mixture component, not HMM state. We are assuming just 1 HMM state.**

GMM Update Equations (Full-Covariance)

Updated (new) parameter estimates

$$\boxed{\bar{w}_k} = \frac{1}{T}\sum_{t=1}^{T}\boxed{P(k \mid o_t, \lambda)}$$

$$\boxed{\bar{u}_k} = \frac{\sum_{t=1}^{T} P(k \mid o_t, \lambda)\cdot o_t}{\sum_{t=1}^{T} P(k \mid o_t, \lambda)}$$

$$\boxed{\bar{\Sigma}_k} = \frac{\sum_{t=1}^{T} P(k \mid o_t, \lambda)\cdot (o_t - \bar{u}_k)(o_t - \bar{u}_k)'}{\sum_{t=1}^{T} P(k \mid o_t, \lambda)}$$

This term is computed using model parameters from previous algorithm iteration.

GMM Update Equations (Diagonal-Covariance)

$$\overline{w}_k = \frac{1}{T}\sum_{t=1}^{T} P(k \mid o_t, \lambda)$$

scalar term

$$\overline{u}_k = \frac{\sum_{t=1}^{T} P(k \mid o_t, \lambda) \cdot o_t}{\sum_{t=1}^{T} P(k \mid o_t, \lambda)}$$

vector term

$$\overline{\sigma}_k^2 = \frac{\sum_{t=1}^{T} P(k \mid o_t, \lambda) \cdot (o_t)^2}{\sum_{t=1}^{T} P(k \mid o_t, \lambda)} - (\overline{u}_k)^2$$

vector term

Initialization of Model Parameters

- **How to initialize the means and variances and mixture weights of the model?**
- **One method: compute global mean & variance of feature vectors,**

$$\mu_1 = \frac{1}{T}\sum_{t=1}^{T} o_t \qquad \sigma_1^2 = \frac{1}{T}\sum_{t=1}^{T}(o_t - \mu_1)^2$$

- **Algorithm incrementally splits mixture into 2 components,**

$$\tilde{\mu}_1 = \mu_1 - 0.2\sigma_1^2 \qquad \tilde{\sigma}_1^2 = \sigma_1^2$$

$$\tilde{\mu}_2 = \mu_1 + 0.2\sigma_1^2 \qquad \tilde{\sigma}_2^2 = \sigma_1^2$$

Initialization of Model Parameters

- **Incremental Mixture Splitting,**
  1. Apply 1 re-estimation iteration with N Gaussians
  2. Find Gaussian with largest mixture weight
  3. Split Gaussian into 2 components, adjust mixture weights to be ½ of the original weight (weights should sum to 1). N=N+1. Go to step 1 until desired N reached.
  4. Apply re-estimation with N Gaussians for several additional iterations to ensure model parameter convergence

Other Training Issues

- **Often useful to enforce a minimum variance value during model estimation.**
  - Improves stability of the training algorithm on computers
  - Helps to avoid "nan" and "inf" during calculations

- **At each algorithm iteration, just check the variances of the model Gaussians. If below a threshold, set them equal to a threshold.**

- **For MFCC features (in speech recognition), threshold values of 0.01 or 0.001 are generally sufficient for good performance.**

Computing the Emission Probability

$$b(\mathbf{o_t}) = \sum_{k=1}^{M} \frac{w_k}{(2\pi)^{d/2} |\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2} (o_t - u_k)' \Sigma_k^{-1} (o_t - u_k) \right)$$

- **Assuming a diagonal covariance matrix,**

$$|\Sigma_k|^{1/2} = \left\| \begin{array}{|c|c|c|c|c|} \hline \sigma_k^2(1) & 0 & 0 & 0 & 0 \\ \hline 0 & \sigma_k^2(2) & 0 & 0 & 0 \\ \hline 0 & 0 & \sigma_k^2(3) & 0 & 0 \\ \hline 0 & 0 & 0 & \ddots & 0 \\ \hline 0 & 0 & 0 & 0 & \sigma_k^2(d) \\ \hline \end{array} \right\|^{1/2} = \left(\prod_{i=1}^{d} \sigma_k^2(i) \right)^{1/2}$$

T-61.184

Computing the Emission Probability

$$(o_t - u_k)' \Sigma_k^{-1} (o_t - u_k) =$$

$$= \underbrace{\begin{array}{|c|} \hline a_k(1) \\ \hline a_k(2) \\ \hline a_k(3) \\ \hline \vdots \\ \hline a_k(d) \\ \hline \end{array}}_{1xd} \times \underbrace{\begin{array}{|c|c|c|c|c|} \hline b_k(1) & 0 & 0 & 0 & 0 \\ \hline 0 & b_k(2) & 0 & 0 & 0 \\ \hline 0 & 0 & b_k(3) & 0 & 0 \\ \hline 0 & 0 & 0 & \ddots & 0 \\ \hline 0 & 0 & 0 & 0 & b_k(d) \\ \hline \end{array}}_{dxd} \times \underbrace{\begin{array}{|c|c|c|c|c|} \hline a_k(1) & a_k(2) & \cdots & \cdots & a_k(d) \\ \hline \end{array}}_{dx1}$$

$$\boxed{\begin{aligned} a_k(i) &= o_t(\mathrm{i}) - \mu_{\mathrm{k}}(i) \\ b_k(i) &= \frac{1}{\sigma_k^2(i)} \end{aligned}}$$

Inverting a diagonal matrix is equivalent to inverting the elements along the diagonal.

Computing the Emission Probability

- **Essentially we can just compute,**

$$b(\mathbf{o_t}) = \sum_{k=1}^{M} w_k b_k(\mathbf{o_t}, \mu_k, \Sigma_k)$$

$$= \sum_{k=1}^{M} \frac{w_k}{(2\pi)^{d/2} |\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2} (o_t - u_k)' \Sigma_k^{-1} (o_t - u_k) \right)$$

$$= \sum_{k=1}^{M} C_k \exp\left(-\frac{1}{2} \sum_{i=1}^{d} \frac{(o_t(i) - \mu_k(i))^2}{\sigma_k^2(i)} \right)$$

Computing the Emission Probability

- **How much computation is required?**

$$b(\mathbf{o_t}) = \sum_{k=1}^{M} C_k \exp\left(-\frac{1}{2} \sum_{i=1}^{d} \frac{\left(o_t(i) - \mu_k(i)\right)^2}{\sigma_k^2(i)} \right)$$

where

$$C_k = \frac{w_k}{(2\pi)^{d/2} \left|\Sigma_k\right|^{1/2}}$$

C_k is pre-computed.
"Md" subtractions,
"2M(d+1)" multiplies,
"M" exp operations

Computing the Emission Probability

- **Let's assume M=16, d=39,**
  - ❑ 624 subtractions
  - ❑ 1,280 multiplies
  - ❑ 16 exp(.) calls

C_k is pre-computed.
"Md" subtractions,
"2M(d+1)" multiplies,
"M" exp operations

- **Typical large vocabulary speech recognizer will have > 5,000 states and maybe M=16…32 mixtures per state. Remember, 100 frames per second!**

Ways to Improve Performance

- **Divides are more expensive than multiplies on most computer architectures. So, store pre-multiplied inverted variances,**

$$b(\mathbf{o_t}) = \sum_{k=1}^{M} C_k \exp\left(-\frac{1}{2} \sum_{i=1}^{d} \frac{\left(o_t(i) - \mu_k(i)\right)^2}{\sigma_k^2(i)} \right)$$

$$= \sum_{k=1}^{M} C_k \exp\left(\sum_{i=1}^{d} \frac{\left(o_t(i) - \mu_k(i)\right)^2}{-2\sigma_k^2(i)} \right)$$

$$= \sum_{k=1}^{M} C_k \exp\left(\sum_{i=1}^{d} \theta_k(i) * \left(o_t(i) - \mu_k(i)\right)^2 \right)$$

$$\boxed{\theta_k(i) = \frac{1}{-2\sigma_k^2(i)}}$$

T-61.184

Nearest-Neighbor Approximation

- **Consider this approximation in the log-domain,**

$$\log b(\mathbf{o_t}) = \log\left\{\sum_{k=1}^{M} C_k \exp\left(\sum_{i=1}^{d} \theta_k(i) * \left(o_t(i) - \mu_k(i)\right)^2\right)\right\}$$

$$\approx \max_{k=1..M}\left\{\log C_k + \sum_{i=1}^{d} \theta_k(i) * \left(o_t(i) - \mu_k(i)\right)^2\right\}$$

$$\approx \max_{k=1..M}\left\{\widetilde{C}_k + \sum_{i=1}^{d} \theta_k(i) * \left(o_t(i) - \mu_k(i)\right)^2\right\}$$

Nearest-Neighbor Approximation

$$\Theta_k = \tilde{C}_k + \sum_{i=1}^{d} \theta_k(i) * \left(o_t(i) - \mu_k(i)\right)^2$$

$$\log b(\mathbf{o_t}) \approx \max_{k=1..M} \{\Theta_k\}$$

Θ_1

- **Computing log-probability under nearest-neighbor assumption, implies finding "best mixture component"**
- **Assumes only 1 Gaussian contributes to the final log-likelihood (the reminder components assumed to have small probability). Allows "sum" to be replaced with "max".**
- **Can further speed up computation by partially computing summation (i=1..d).**

Efficient Nearest Neighbor Computation

$1)\ \text{compute}\ \Theta_1$

$2)\ \log b(o_t) = \Theta_1$

$3)\ \text{for k} = 2..\text{M},$

$\quad \Theta_\text{k} = \tilde{C}_k; \quad i = 1;$

$\quad \text{while}\left(\Theta_\text{k} > \log b(o_t)\right) \& (i <= d)$

$\quad\quad \Theta_\text{k} = \Theta_\text{k} + \theta_k(i) * (o_t(i) - \mu_k(i))^2$

$\quad\quad \text{i} = \text{i} + 1;$

$\quad \text{end}$

$\quad \text{if (i} == \text{d} + 1) \quad \log b(o_t) = \Theta_k$

end

Example Uses of GMMs

- **Speaker Identification & Verification**
  - Model each speaker with a GMM
  - Distribution of parameters (MFCCs) used to model acoustic space of each speaker.
- **Music Recognition**
  - Can model each song with a GMM
  - Model is time-independent. Can start playing the song at any position and GMM will correctly classify song
  - Group songs by artist into a single GMM. Can detect the artist in most cases!
  - What about a "genre" GMM? (classical, rock, jazz, pop?)
- **Speech Recognition**

Practical HMM Training (for Speech Recognition)

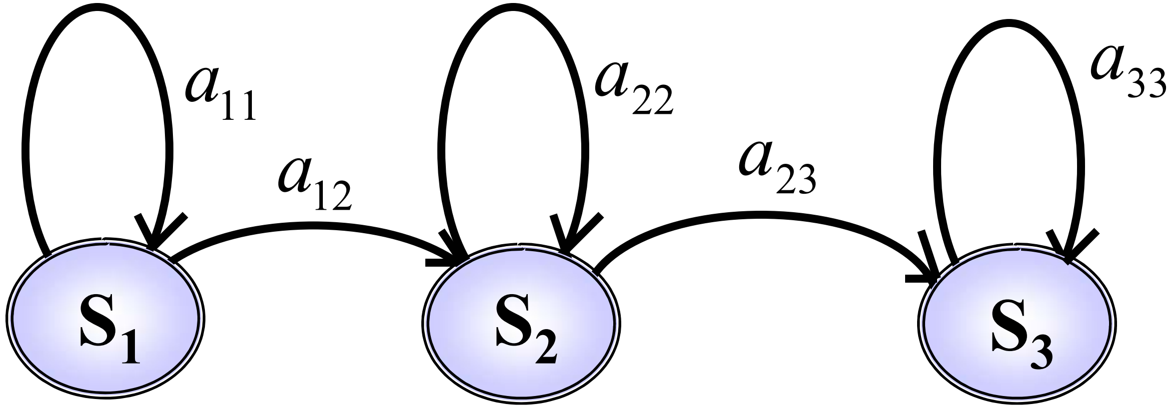
- **Our goal is to "assign" extracted feature vectors to HMM states**

- **Two popular methods for training,**
  - Forward-Backward training assigns a probability that each vector was emitted from each HMM state (fuzzy labeling)

  - Viterbi training just assigns a feature vector to a particular state (most likely state from the best path).

Phoneme HMM

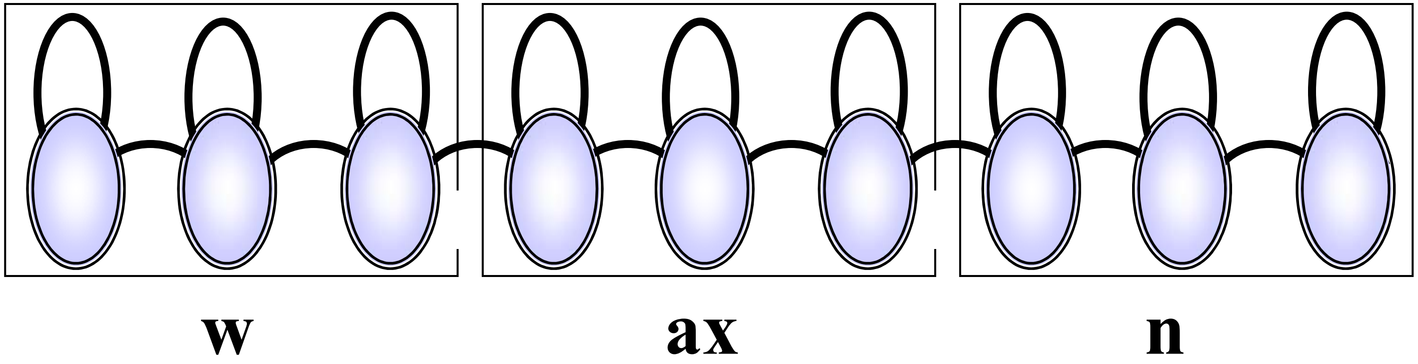
- **Let's assume each phoneme is represented by 3 HMM states connected with forward transitions,**



- **S1 models the beginning part of the sound, S2 the middle, and S3 the end-part of the sound unit.**

Word & Sentence HMMs

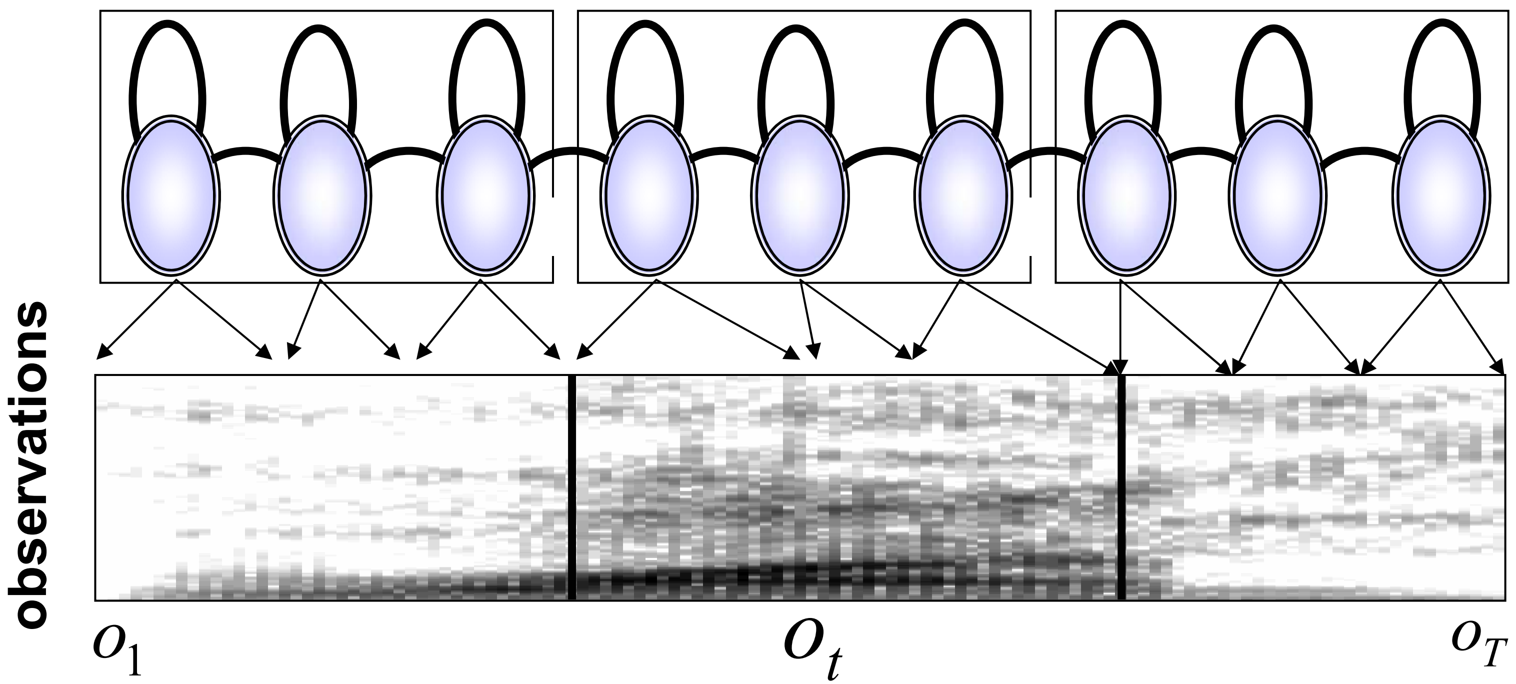
- Construct word & sentence-level HMMs from the phoneme-level units. For example, “ONE” with pronunciation “W AX N”:



- For simplification, let’s assume each state is a Gaussian Mixture Model (GMM). We also have transition probabilities between states.

Viterbi Training

- **Given an utterance, we can construct the composite HMM from the phone units and use the Viterbi algorithm to find the best state-sequence (assignment of feature-vectors to HMM states):**



Viterbi Algorithm in Log-Domain

1. **Initialization** $\tilde{\delta}_1(i) = \tilde{\pi}_i + \tilde{b}_i(\mathbf{o}_1) \quad \psi_1(i) = 0$

2. **Recursion**

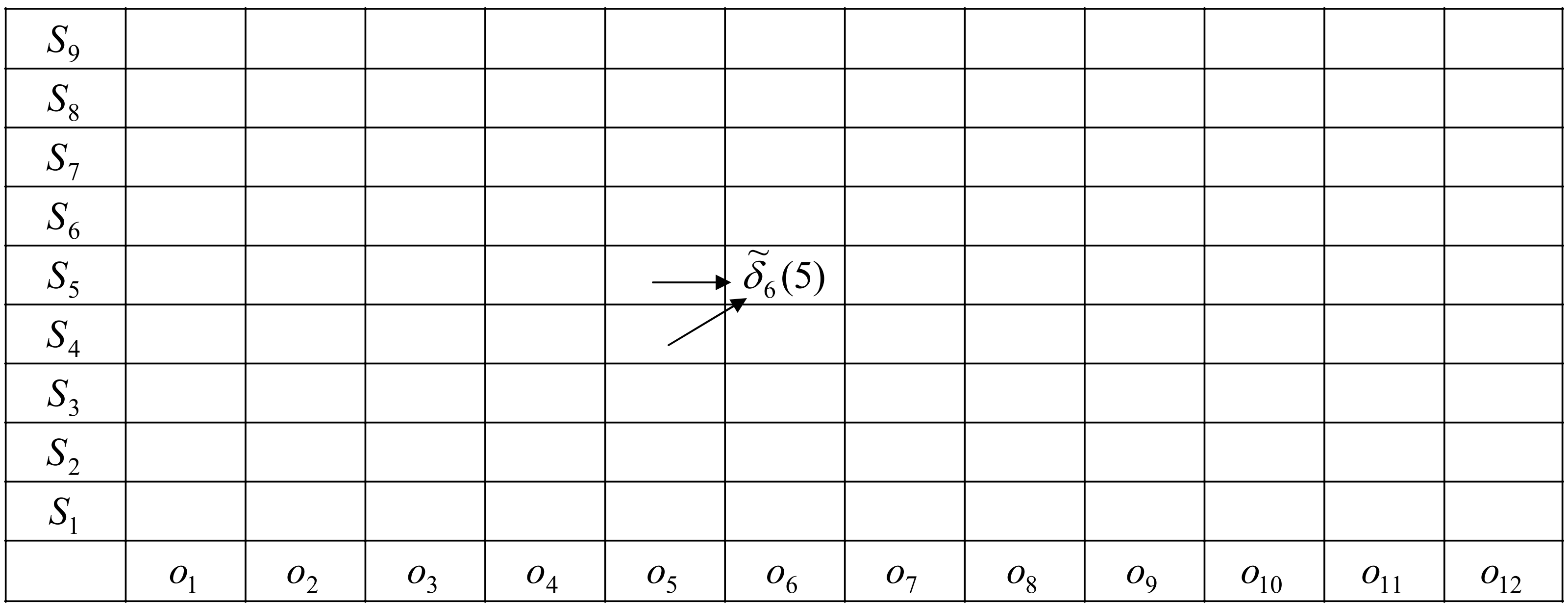
$$\tilde{\delta}_t(j) = \max_{1 \le i \le N} [\tilde{\delta}_{t-1}(i) + \tilde{a}_{ij}] + \tilde{b}_j(\mathbf{o}_t)$$

$$\psi_t(j) = \arg\max_{1 \le i \le N} [\tilde{\delta}_{t-1}(i) + \tilde{a}_{ij}]$$

3. **Termination** $\tilde{P}^* = \max_{1 \le i \le N} \left[\tilde{\delta}_T(i)\right] \quad q_T^* = \arg\max_{1 \le i \le N} \left[\tilde{\delta}_T(i)\right]$

4. **Path Back trace** $q_t^* = \psi_{t+1}(q_{t+1}^*)$

Viterbi Algorithm Illustration for Feed-Forward HMM Topology



$$\tilde{\delta}_{t=6}(s=5) = \max\left\{\left[\tilde{\delta}_{t=5}(s=5)+\tilde{a}_{55}\right],\left[\tilde{\delta}_{t=5}(s=4)+\tilde{a}_{45}\right]\right\} + \tilde{b}_{s=5}(t=6)$$

$$\psi_{t=6}(s=5) = \arg\max\left\{\underbrace{\left[\tilde{\delta}_{t=5}(s=5)+a_{55}\right]}_{self-loop},\underbrace{\left[\tilde{\delta}_{t=5}(s=4)+a_{45}\right]}_{forward-transition}\right\}$$

Viterbi Training

- **For each training example, use current HMM models to assign (align) feature vectors to HMM states.**
  - ❑ Assignment is made by using the Viterbi algorithm.
  - ❑ Assignment is based on most-likely path through composite HMM model
  - ❑ We refer to this as “Viterbi forced-alignment”
- **Group feature vectors assigned to each HMM state and estimate new HMM state parameters (e.g., using GMM update equations).**
- **Repeat alignment / retraining process**

Forward-Backward Training

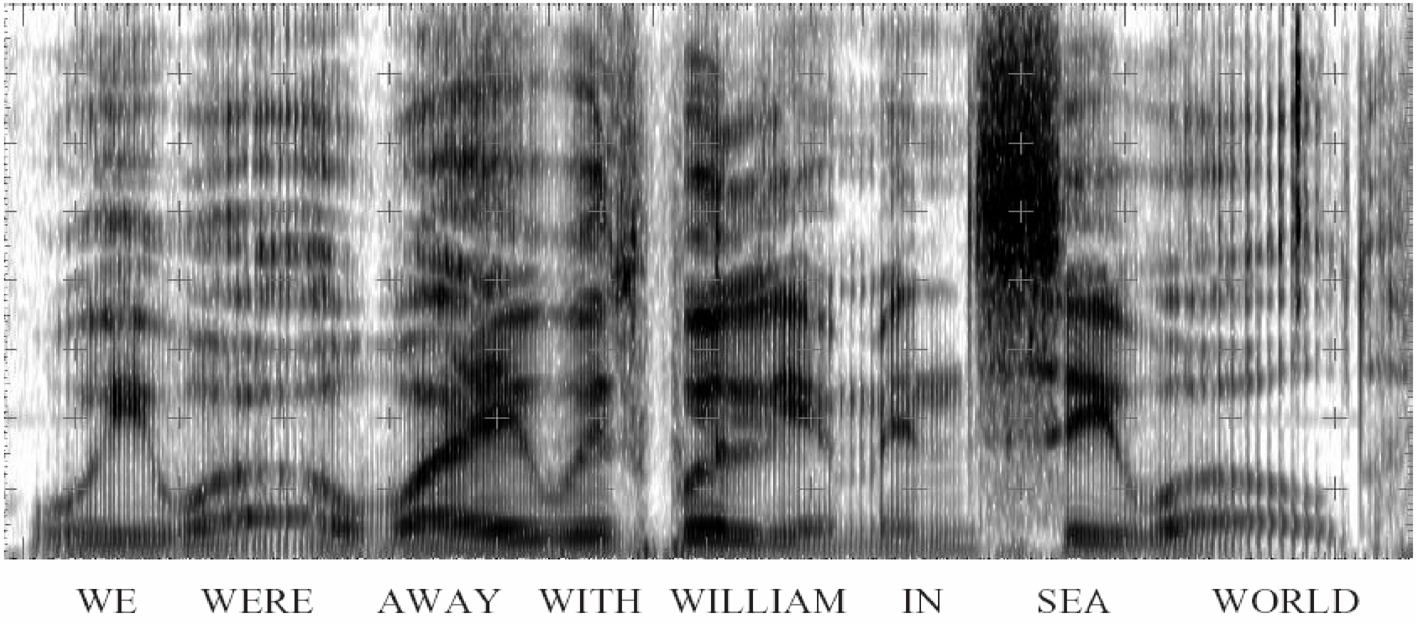
- **Rather than assigning each feature vector to a particular HMM state, we compute a "fuzzy-assignment".**
- **"Fuzzy-assignment" is based on the probability of being in state *i* at time *t*,**
  - Requires computing the forward and backward variables.
  - FB training is more expensive than Viterbi training

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}$$

T-61.184

Acoustic Modeling Issues

- **How to take into account variabilities in the acoustic signal? For example, the context-dependency of “w” in this example,**



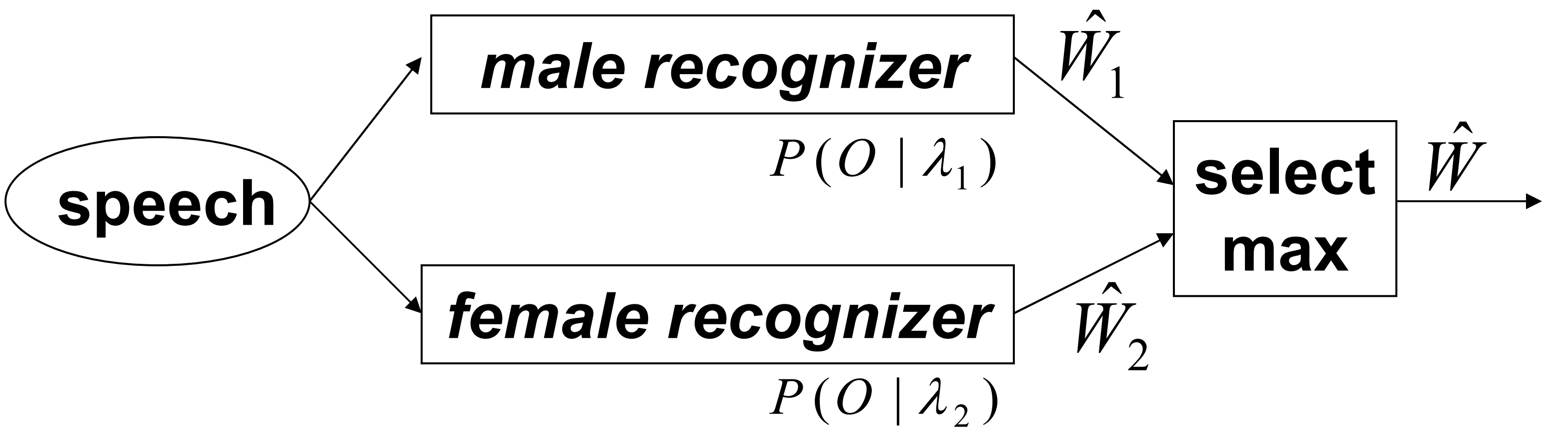
T-61.184

Types of Acoustic Variability

- **Environmental Variability**
- **Between-Speaker Variability**
  - Gender, Age
  - Dialect
  - Speaking Style
    - formal vs. informal
    - Planned vs. spontaneous
- **Within-Speaker Variability**
  - Variations within an utterance (could be due to prosody)
  - Speaker-specific co-articulation

Variability-Dependent Recognition

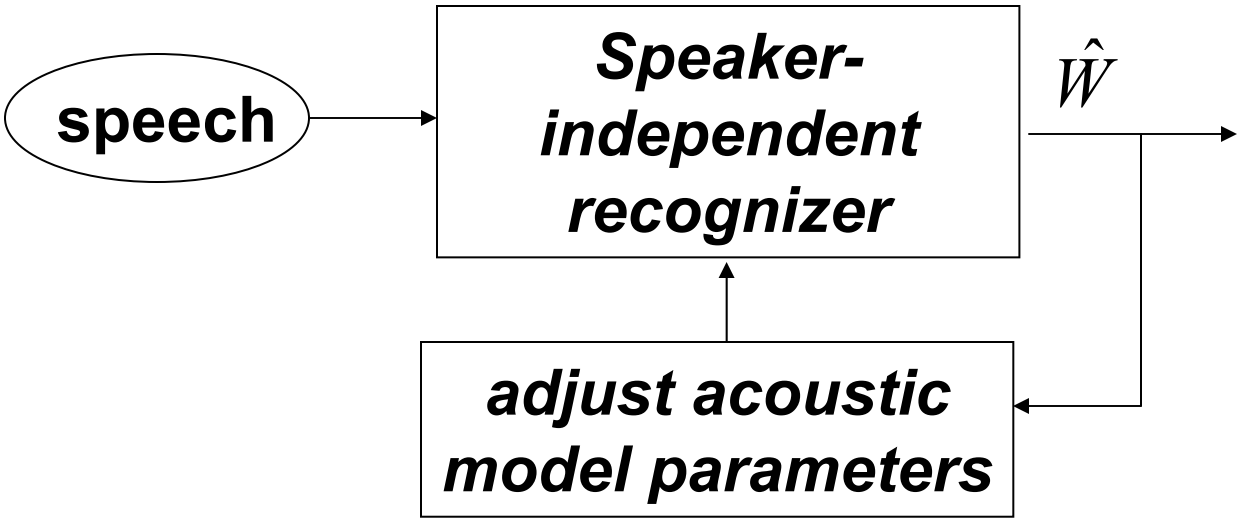
- One simple method might be to estimate model parameters under each condition,



- Can not account for all factors (data-sparse). Also very inefficient.

Variability-Adapted Recognition

- **Another solution adapts the parameters of the recognizer to better match the input,**



- **Adaptation can be supervised or unsupervised**

An Ideal Acoustic Model also…

- **Accounts for context-dependency**
  - ❑ A phoneme produced in one phonetic context may be similar to the same phoneme produced in another phonetic context. (the converse is also true!)
- **Provides a compact & trainable representation**
  - ❑ Which is trainable from finite amounts of data
- **Provides a general representation**
  - ❑ Allows new words to be modeled which may not have been seen in the training data

Whole-Word HMMs

- **Assign a number of HMM states to model a word as a whole.**
- **Passes the test?**
  - Accurate – Yes, if you have enough data and your environment consists of a small vocabulary. No, if you are trying to model context changes between words.
  - Compact – No, need too many states as vocabulary increases. Probably not enough training data to model *every* word. What about infrequent words???
  - General – No, can't build new words using this representation.

Context-Independent Phoneme HMMs

- **Context-independent models consist of a single *M*-state HMM (e.g., *M*=3), one for each phoneme unit**

- **Also referred to as “monophone” models**

- **Passes the test?**
  - Accurate – No, does not accurately model coarticulation
  - Compact – Yes, *M* states x *N* phonemes leads to only a few parameters which need to be estimated.
  - General – Yes, you can construct new words by stringing together the units.

Context-Dependent Triphone HMMs

- **Context-dependent models which consist of a single 3-state HMM, one for each phoneme unit modeled with the immediate left-and-right phonetic context**

- **Passes the test?**
  - Accurate – Yes, takes coarticulation into account
  - Compact – Yes, Trainable – No: For N phonemes, there exists NxNxN triphone models. Too many parameters to estimate!
  - General – Yes, you can construct new words by stringing together the units.

Describing Context-Dependent Phonetic Models

- **Monophone:**
  - A single model used to represent phoneme in all contexts.
- **Biphone:**
  - Each model represents a particular left or right context.
  - Left-context biphone notation: (a-b)
  - Right-context biphone notation: (b+c)
- **Triphone:**
  - Each model represents a particular left & right context.
  - (a-b+c) refers to phoneme “b” with “a” preceding and “c” immediately following.

Context-Dependent Model Examples

- **Monophone:**
  - ❑ BRYAN → **B R AY AX N**
- **Biphone**
  - ❑ Left-Context: → SIL-**B** B-**R** R-**AY** AY-**AX** AX-**N**
  - ❑ Right-Context: → **B**+R **R**+AY **AY**+AX **AX**+N **N**+SIL
- **Triphone**
  - ❑ → SIL-**B**+R B-**R**+AY R-**AY**+AX AY-**AX**+N AX-**N**+SIL

Word-Boundary Modeling

- ***Word-internal Context-Dependent Model Sequence*** **(backs off to left and right biphone models at word boundaries):**

  BRYAN PELLOM → `SIL` **`B`**`+R B-`**`R`**`-AY R-`**`AY`**`+AX AY-`**`AX`**`+N AX-`**`N`** **`P`**`+EH P-`**`EH`**`+L EH-`**`L+`**`AX L-`**`AX`**`+M AX-`**`M`** `SIL`

- ***Cross-Word Context-Dependent Triphone Sequence***

  BRYAN PELLOM → `SIL-`**`B`**`+R B-`**`R`**`+AY R-`**`AY`**`+AX AY-`**`AX`**`+N AX-`**`N`**`+P N-`**`P`**`+EH P-`**`EH`**`+L EH-`**`L`**`+AX L-`**`AX`**`+M AX-`**`M`**`+SIL`

Triphone Acoustic Models

- **Provide nice trade-off**
  - Compact, General, Accurate
  - Assumes dependency on just previous and following phoneme.
- **Modeling and Estimation Issues:**
  - Not all triphone contexts appear in training data. We call these “unseen” triphones.
  - Many triphone contexts occur infrequently in the training data (data-sparse modeling problem)
- **Solution**
  - Cluster HMM states which share similar statistical distributions
  - Estimate HMM parameters using resulting pooled data
  - How to cluster the data?????

Trainability of Acoustic Models

- **Tradeoff exists between the level of detail of the acoustic model and our ability to adequately estimate the parameters of the model**

- **Methods for improving trainability,**
  - ❑ Backing-off : triphones → biphones → monophones
  - ❑ Smoothing : interpolate parameters of more specific models with those of less specific (better trained) models
  - ❑ Sharing : cluster similar contexts

Basic Idea behind State-Clustering

- **For each phoneme example in the training data,**
  - Segment data into HMM states (S1, S2, S3)
  - Assign a triphone context to each “chunk” of features
- **Cluster “chunks” so that each cluster has similar acoustic properties**
- **Possible clustering methods**
  - Heuristic
  - Bottom-up
  - Top-down

Heuristic Clustering

- **Define a set of equivalence classes,**
  - C1: Stop = {B,D,G,P,T,K}
  - C2: Fricative = {S, SH, F, Z, ZH}
  - C3: Nasal = {M, N, NG}
  - C4: Vowel = {AX, AE, AY, IY, IX, IH, AA, AO, …}
  - C5: Semivowel = {L,R,W}
  - C6: Silence = {SIL}
- **Cluster data by class. For example,**
  - {C1}-AX+{C2} → {B-**AX**+S}, {D-**AX**+Z}, … ,{T-**AX**+SH}
  - {C5}-IY+{C6} → {L-**IY**+SIL},{R-**IY**+SIL},{W-**IY**+SIL}

Bottom-Up Clustering

- **Compare triphones of differing contexts and merge those that are most similar,**
  - Estimate Gaussians for each "seen" triphone context
  - Compute Distance between triphones,

$$d(i,j) = \left[\frac{1}{d} \sum_{k=1}^{d} \frac{\left(\mu_i[k] - \mu_j[k]\right)^2}{\sigma_i^2[k]\sigma_j^2[k]} \right]^{\frac{1}{2}}$$

  - Merge triphone contexts based on distance and number of training examples.
- **Can not predict "unseen" triphones**

Bottom-Up Clustering

- **K. F. Lee, "Context-Dependent Phonetic Hidden Markov Models for Speaker-Independent Continuous Speech Recognition, IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. 38, No. 4, pp. 599-609, 1990.**

- **Proposed "Generalized Triphones"**
  - "Seen" triphone contexts merged using an information theoretic measure (entropy).
  - Similar to heuristic clustering, but clusters found in a more principled manner.
  - Merging done at the model-level, not HMM state level.
  - "unseen" triphone contexts can not be predicted

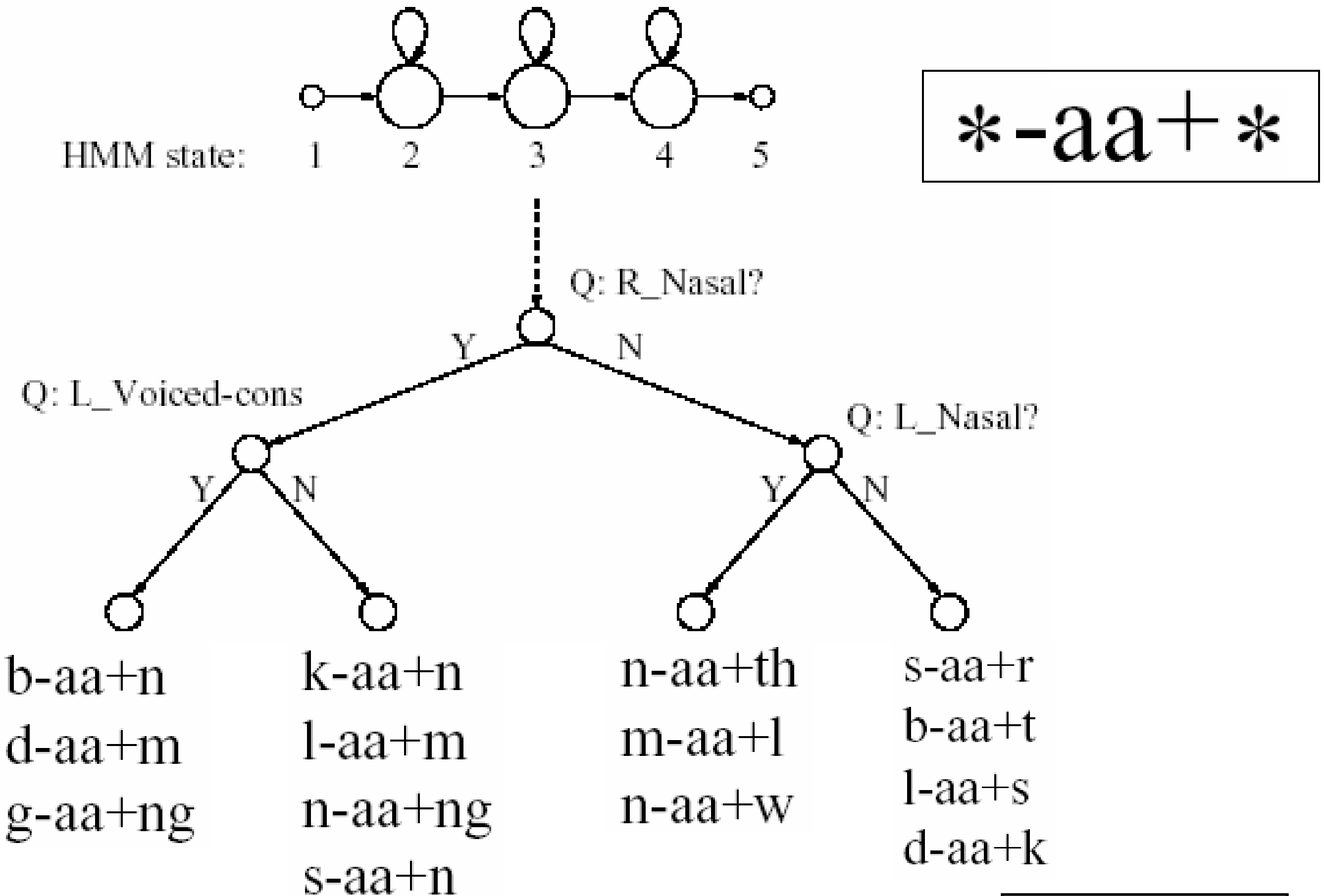
Top-Down Clustering

- **Place all training tokens at a root-node**
- **Sequentially partition data into children nodes which share similar phonetic contexts**
- **Split-data to ensure,**
  - ❑ Sufficient difference between clustered states
  - ❑ Sufficient training data exists to estimate model parameters
- **Allows for prediction of “unseen” triphones…**

Decision-Tree State Clustering Approach

- **A decision tree is constructed by asking phonetically motivated questions about the left/right context of the training data**
- **Binary {yes/no} questions are asked about the data. E.g., "is the phoneme to the immediate left a nasal?"**
- **Questions which maximize the likelihood of the training data being generated by the model are selected.**
- **Each sequential split of the data partitions the acoustic space into similar phonetic contexts.**

Decision-Tree State Clustering (one tree built for each state-position)



Example Splitting Questions

`$silence`	`SIL br ls lg ga`
`$aspiration`	`HH`
`$dental`	`DH TH`
`$l_w`	`L W`
`$s_sh`	`S SH`
`$s_z_sh_zh`	`S Z SH ZH`
`$affricate`	`CH TS JH`
`$nasal`	`M N NG`
`$schwa`	`AX IX AXR`
`$voiced_fric`	`DH Z ZH V`
`$voiceless_fric`	`TH S SH F`

"Is the Left-Context an "L" or "W"?
"Is the Right-Context an "L" or "W"?

Advantages of Decision-Tree Clustering

- **Hierarchical structure ensures context-dependent model is built for <u>all</u> contexts**
- **Expert linguistic knowledge incorporated via splitting questions**
- **Splitting can be controlled to ensure enough data exists to model the particular context**
- **Greater degrees of context-dependency can be incorporated through the question sets**

Algorithm Constraints

- **Each leaf should have a minimum number of examples to ensure that the models are adequately estimated**
- **A finite set of questions can be used to split each node. Question set selected to incorporate knowledge of similar articulatory events.**

Splitting Tree Nodes

- At each node-position in the tree, questions assigned to the node are evaluated according to their "goodness of split". Best question selected to split the parent node into 2 children nodes.
- "Goodness of split" based on maximizing the likelihood of the training data
- Children nodes are also checked to ensure that a minimum amount of data are assigned for improved trainability.

Summary of Decision-Tree Process

1. **All of the states to be clustered are placed initially at the root node of the tree and the likelihood of the training data is calculated**

2. **Node is split by finding the question which partitions the states in the parent node to give the maximum increase in log-likelihood.**

3. **Splitting process repeated until:**
   - Log-likelihood change due to a split falls below a threshold,
   - Number of training examples (or occupation count) within children nodes fall below a threshold

Evaluating the Goodness of Split

- **It can be shown that assuming Gaussian PDFs of dimension 'd', the approximate log-likelihood of generating the training data is given by,**

$$L(\mathbf{S}) = -\frac{1}{2}\left(\log[(2\pi)^d |\Sigma(\mathbf{S})|] + d\right) \sum_{t \in T} \sum_{s \in S} \gamma_s(\mathbf{o}_t)$$

- **Where,**

$\mathbf{S}$ **: Set of tied states**

$\Sigma(\mathbf{S})$ **: Variance of tied states**

Variance of the Tied-States

- **Assuming each seen triphone context is modeled using a single mean vector and diagonal covariance matrix,**

$$\Sigma(\mathbf{S}) = \frac{\sum_{c \in C(S)} \gamma_c \left(\Sigma_c + \mu_c \mu_c' \right)}{\sum_{c \in C(S)} \gamma_c} - \left(\frac{\sum_{c \in C(S)} \gamma_c \mu_c}{\sum_{c \in C(S)} \gamma_c} \right) \left(\frac{\sum_{c \in C(S)} \gamma_c \mu_c}{\sum_{c \in C(S)} \gamma_c} \right)'$$

- **Thus, the pooled variance is estimated from the individual means and variances of the seen triphone contexts (c) which comprise the tied-state. Weighted by counts (gamma)**
- **Splitting is therefore based on training data statistics, not the data itself. Makes the algorithm efficient.**

Evaluating “Goodness of Split”

- **Assuming a question “q” partitions the states “S” into 2 <u>y</u>es/<u>n</u>o subsets:**

$$\begin{tikzcd}[column sep=small] & \mathbf{S} \arrow[dl] \arrow[dr] & \\ \mathbf{S}_{\mathbf{y}}(q) & & \mathbf{S}_{\mathbf{n}}(q) \end{tikzcd}$$

- **Question is selected such that the change in log-likelihood is maximized,**

$$\Delta L_q = L(\mathbf{S}_{\mathbf{y}}(q)) + L(\mathbf{S}_{\mathbf{n}}(q)) - L(\mathbf{S})$$

Practical Viterbi Training using DT's

- Step 1: Use Viterbi Algorithm to determine alignment of frames of training data to HMM states by building sentence-level HMMs. Requires an initial HMM model.

- Step 2: Place all frames of training data at the root node of the tree (there are 3 trees built for each phoneme, one for each HMM state). Split the tree into leaf nodes using the decision tree algorithm

- Step 3: Re-estimate the clustered state Gaussians using frames assigned to each leaf. Repeat 1-3 until model converges. Gaussians constructed from final set of tree leaves will model all possible triphone contexts.

Understanding the Importance of Decision Tree Splitting Questions

Condition	Question	Total Gain
All states of all models	R-Vowel	25.9
	L-Vowel	23.3
	R-Unrounded	19.7
	L-UnFortisLenis	19.5
	R-UnFortisLenis	18.3
	R-r	17.1
Entry state of all models	L-UnFortisLenis	18.3
	L-Vowel	16.9
	L-Nasal	10.3
	L-CentralFront	7.7
	L-Unrounded	7.4
	L-Fortis	6.2
Exit state of all consonants	R-Vowel	15.2
	R-Unrounded	8.6
	R-High	4.7
	R-ee	3.9
	R-Rounded	3.7
	R-Syllabic	3.6

- **Young, HLT'94 paper**
- **Wall Street Journal Dictation Task**
- **30 hours of training data**
- **Most important U.S. English questions,**
  - "Is the right context a vowel?"
  - "Is the left context a vowel?"
  - "Is the right context an unrounded vowel?" …

Improvements Realized by Tree-Based Clustering

System Clustering Method	Word Error Rate (Wall Street Journal 5k Vocabulary Task)
Model-based Clustering (Bottom-Up)	12.17%
State-based Clustering (Bottom-Up)	10.73%
Tree-based Clustering (Top-Down)	9.67%

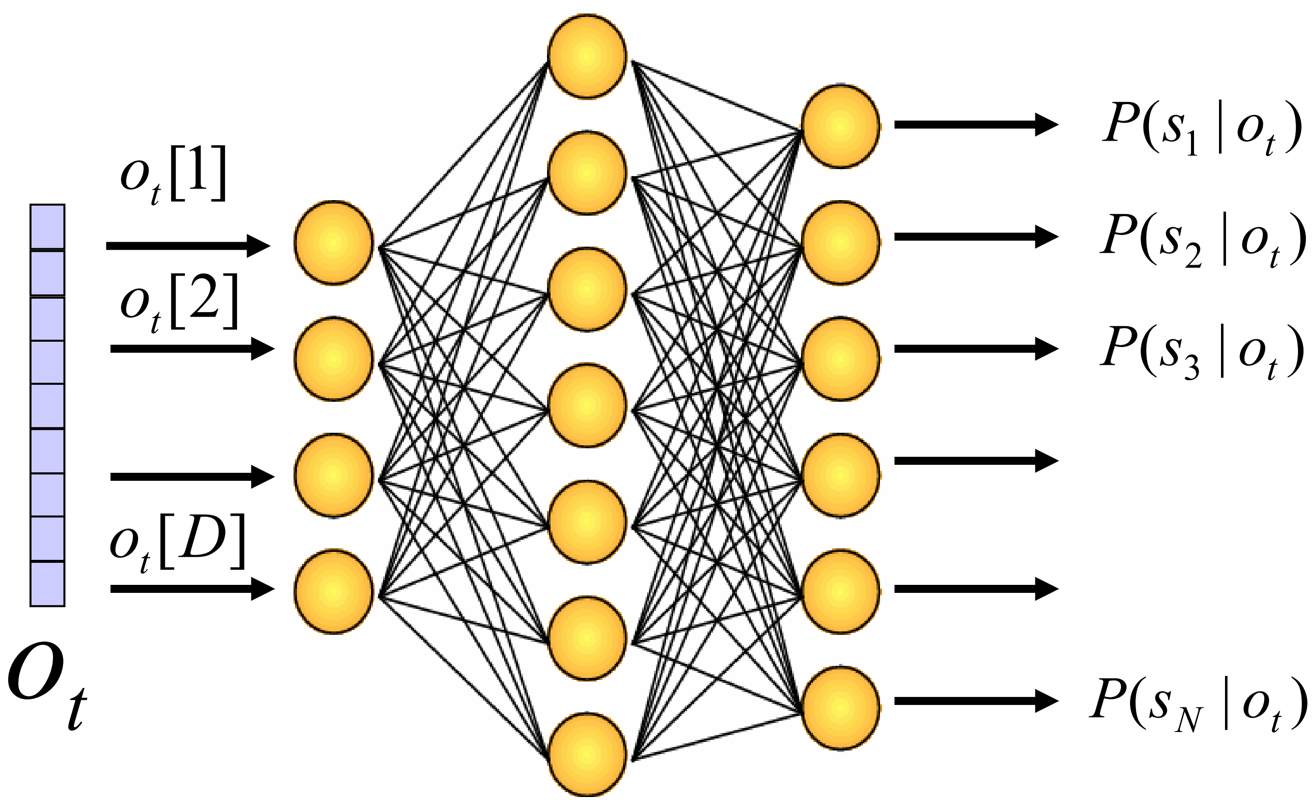
From Young HLT'94

Research Issues

- Acoustic training based on maximizing the likelihood of the training data does not ensure discrimination between units.
- Can consider “discriminative” training of acoustic model parameters
- Discriminative training methods are expensive since most require running the recognizer on the training data so that confusions can be modeled.

Alternative Methods for Modeling Emission Probabilities in an HMM system

Multi-layer Perceptron (MLP)



Multi-layer Perceptron (MLP)

- **Advantages**
  - Input a feature vector and immediately compute posterior probability of each modeled class (must divide by the priors from the training data to get the likelihoods for the HMM)
  - (1) Efficient in terms of CPU compared to Gaussians, and (2) efficient in terms of memory space

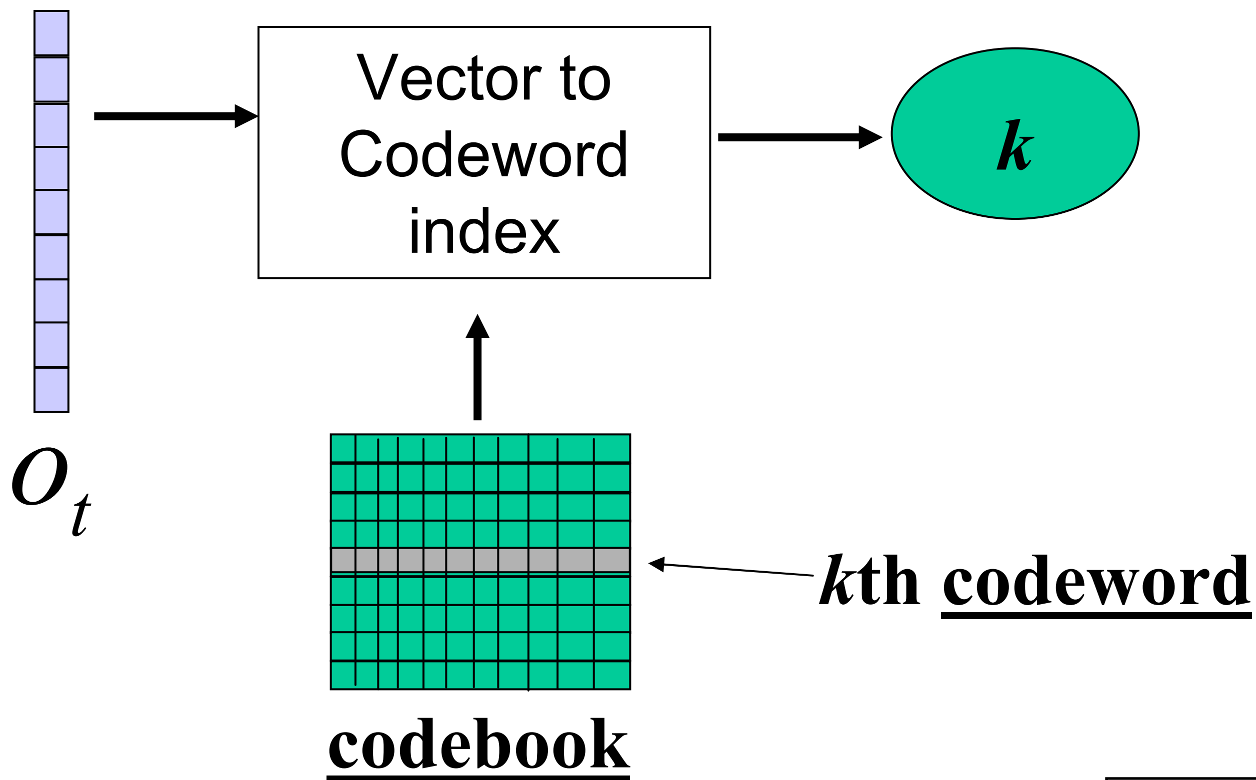
- **Disadvantages**
  - Each output node models 1 context-dependent class. Too many output nodes needed
  - Training time is very slow compared to Gaussian systems
  - Speaker Adaptation? How to adjust the weights??

Recurrent Neural Network (RNN)

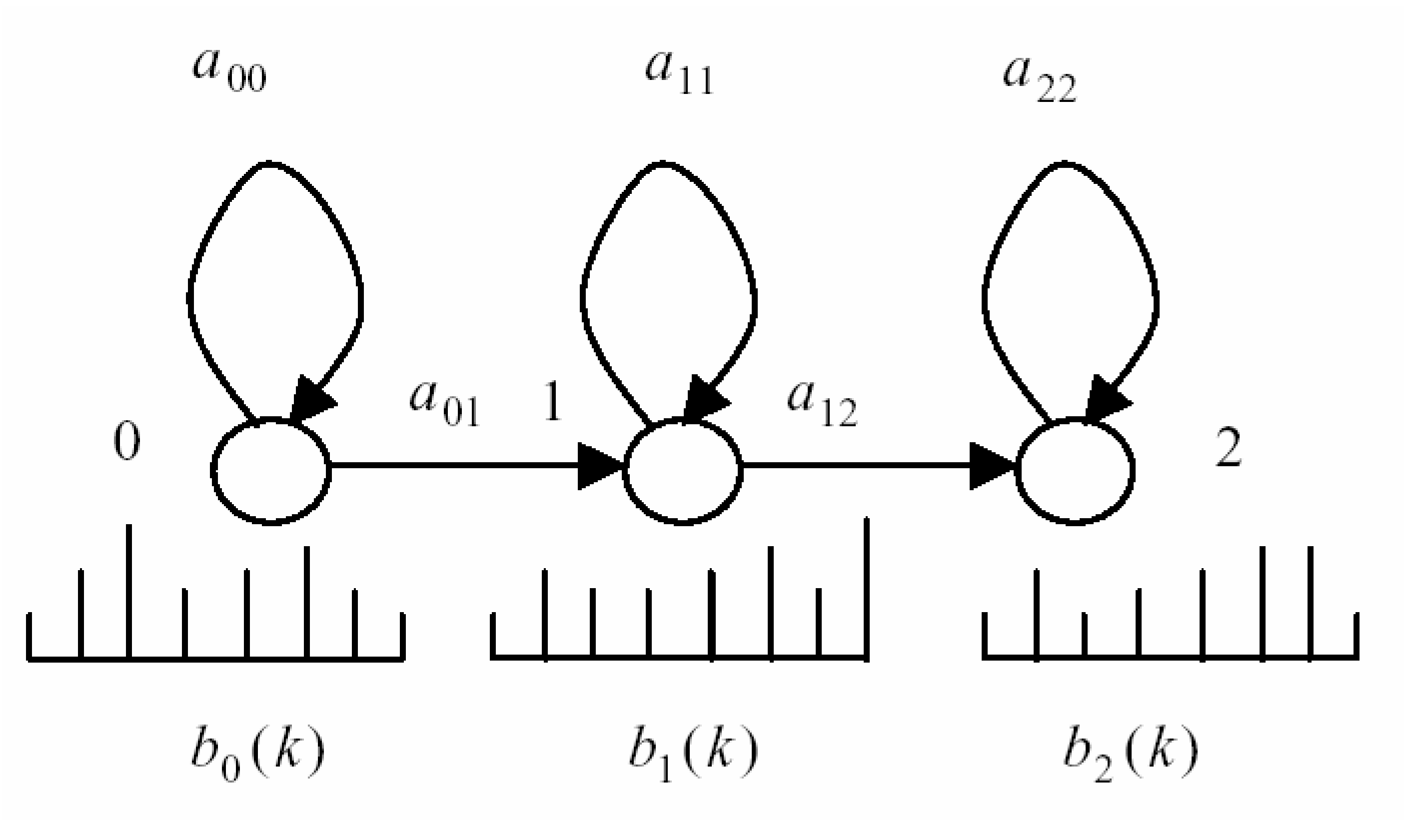
- **Advantages**
  - Feedback in this architecture allows for better modeling of context-dependency. Can use one output node per phoneme. Network generates posterior probability of phoneme. Can convert to likelihoods by scaling by the priors of each unit.
  - Memory and CPU efficient during recognition
  - NICO Toolkit: `http://www.speech.kth.se/NICO/`
- **Disadvantages**
  - Training time required to compute RNN parameters
  - Scalability to very large speech recognition tasks unknown

Discrete Symbol HMM

- **Model speech features using discrete symbols:**

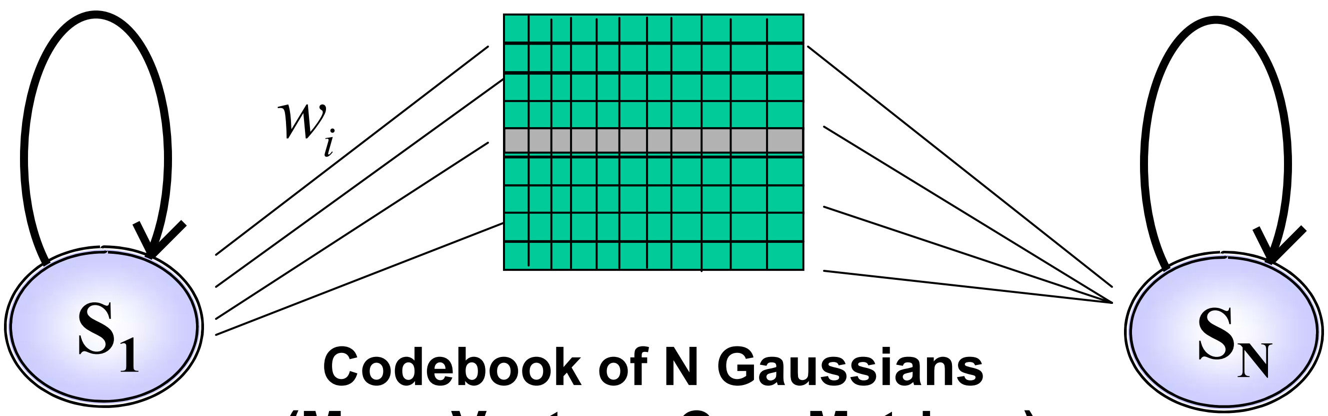


Discrete Symbol HMMs



Semi-Continuous HMM (SCHMM)

- System has a codebook of N Gaussians (e.g., N=256)
- Each clustered HMM state modeled by weighted set of Gaussians from system codebook



Codebook of N Gaussians
(Mean Vectors, Cov. Matrices)

Semi-Continuous HMM (SCHMM)

- **Advantages**
  - Speed: Compute all Gaussians within codebook for each input frame. Once computed, the state-likelihoods are efficiently computable (weighted sum).
  - Efficiency: the codebook can be stored with little memory overhead
  - Trainability: the system codebook can be robustly estimated.
- **Disadvantages**
  - Some loss in modeling accuracy due to fixed codebook
  - Fully-continuous systems essentially have 1 codebook per clustered state. SCHMMs have 1 code book shared by all clustered states.

Exercise 4: Project Midterm

- **Details posted on course webpage (see link to exercise 4)**
- **Project can be "hands-on" or "literature survey"**
  - Hands-on projects require less writing
- **Important Dates**
  - October 18th, deadline to select a group for "hands-on" projects and also to submit an abstract for your project. Must be approved by the 18th.
  - October 27th, Midterm write-up deadline
  - November 29th, In-class project presentations
  - December 8th, deadline for final project write-up

Upcoming Meetings

- **Monday, October 18th**
  - ❑ Language Modeling for Speech Recognition

- **Monday, October 25th**
  - ❑ Search Algorithms for Speech Recognition