HIDDEN MARKOV MODELS IN SPEECH RECOGNITION

Wayne Ward

Carnegie Mellon University

Pittsburgh, PA

Acknowledgements

Much of this talk is derived from the paper

"An Introduction to Hidden Markov Models",

by Rabiner and Juang

and from the talk

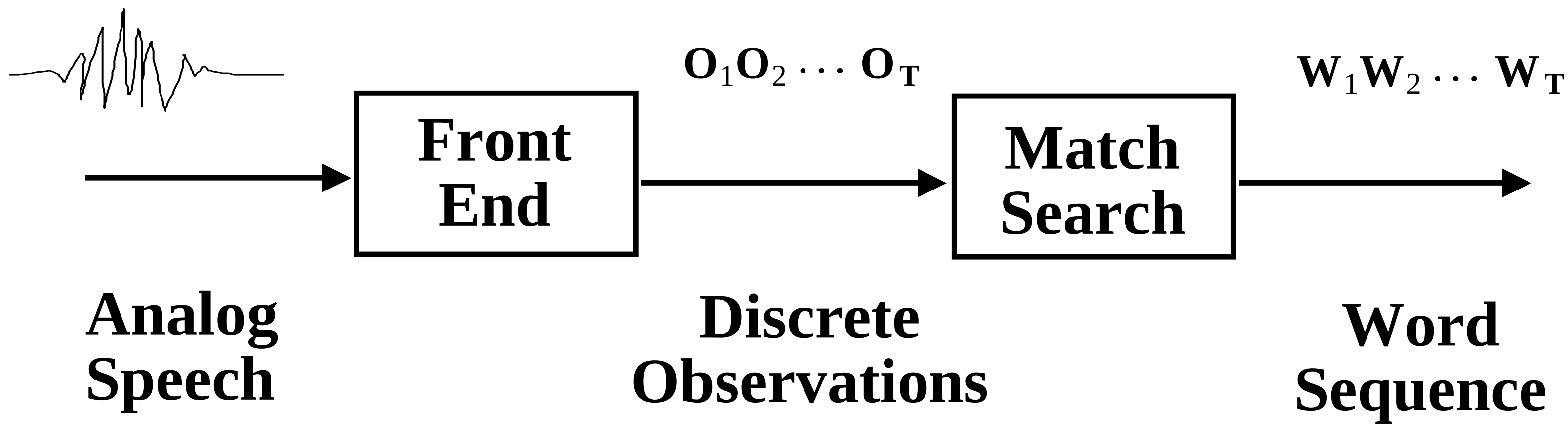
"Hidden Markov Models: Continuous Speech Recognition"

by Kai-Fu Lee

Topics

- Markov Models and Hidden Markov Models
- HMMs applied to speech recognition
  - Training
  - Decoding

Speech Recognition



ML Continuous Speech Recognition

Goal:

Given acoustic data $A = a_1, a_2, ..., a_k$

Find word sequence $W = w_1, w_2, ... w_n$

Such that $P(W \mid A)$ is maximized

Bayes Rule:

acoustic model (HMMs)

language model

$$P(W \mid A) = \frac{P(A \mid W) \bullet P(W)}{P(A)}$$

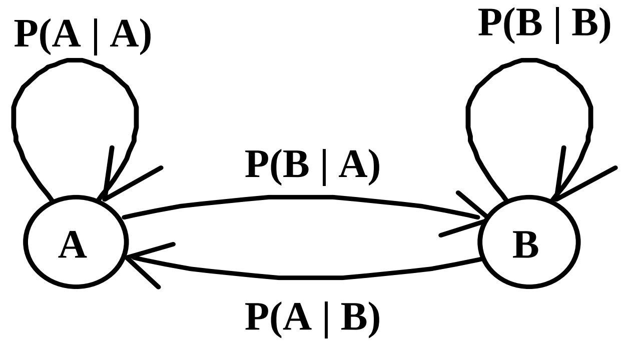
P(A) is a constant for a complete sentence

Markov Models

Elements:

States : $\mathbf{S} = \{\mathbf{S}_0, \mathbf{S}_1, \cdots \mathbf{S}_\mathbf{N}\}$

Transition probabilities : $P(q_t = S_i \mid q_{t-1} = S_j)$



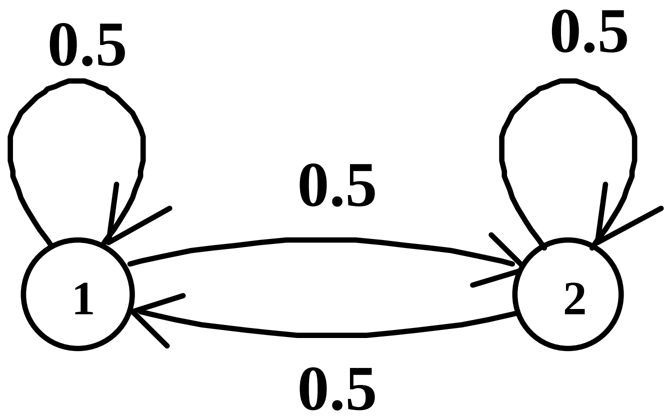
Markov Assumption:

Transition probability depends only on current state

$$P(q_t = S_i \mid q_{t-1} = S_j,\ q_{t-2} = S_k,\ \cdots) = P(q_t = S_i \mid q_{t-1} = S_j) = a_{ji}$$

$$a_{ji} \geq 0 \quad \forall\ j,i \qquad\qquad \sum_{i=0}^{N} a_{ji} = 1 \quad \forall_j$$

Single Fair Coin



P(H) = 1.0   **P(H) = 0.0**

P(T) = 0.0   **P(T) = 1.0**

Outcome head corresponds to state 1, tail to state 2

Observation sequence uniquely defines state sequence

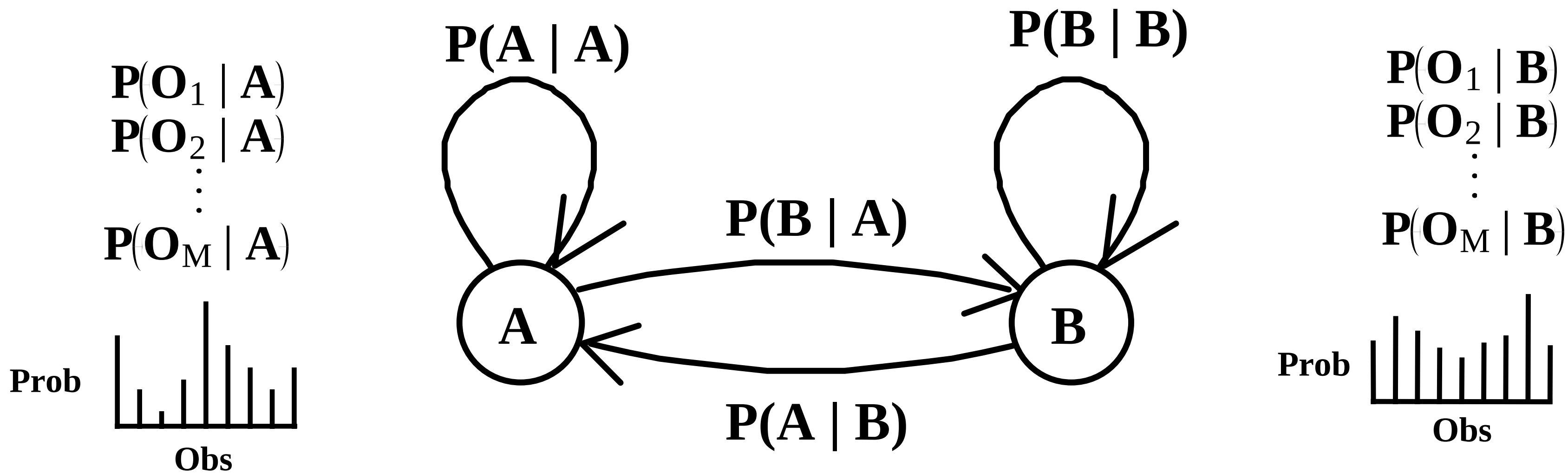
Hidden Markov Models

Elements:

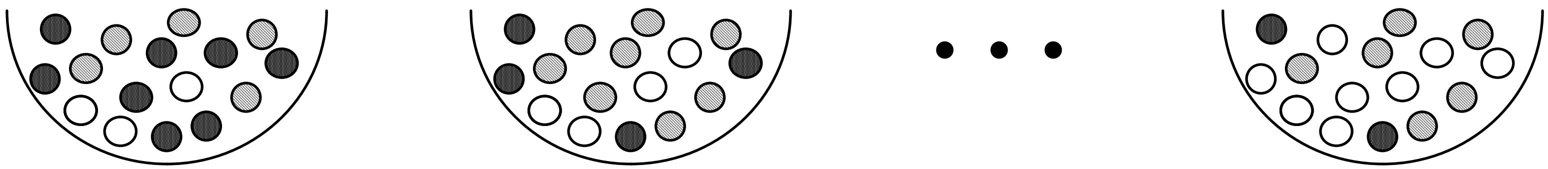
States $S = \{S_0, S_1, \cdots S_N\}$

Transition probabilities $P(q_t = S_i \mid q_{t-1} = S_j) = a_{ji}$

Output prob distributions (at state j for symbol k) $P(y_t = O_k \mid q_t = S_j) = b_j(k)$



Discrete Observation HMM



P(R) = 0.31
P(B) = 0.50
P(Y) = 0.19

P(R) = 0.50
P(B) = 0.25
P(Y) = 0.25

P(R) = 0.38
P(B) = 0.12
P(Y) = 0.50

Observation sequence: R B Y Y • • • R

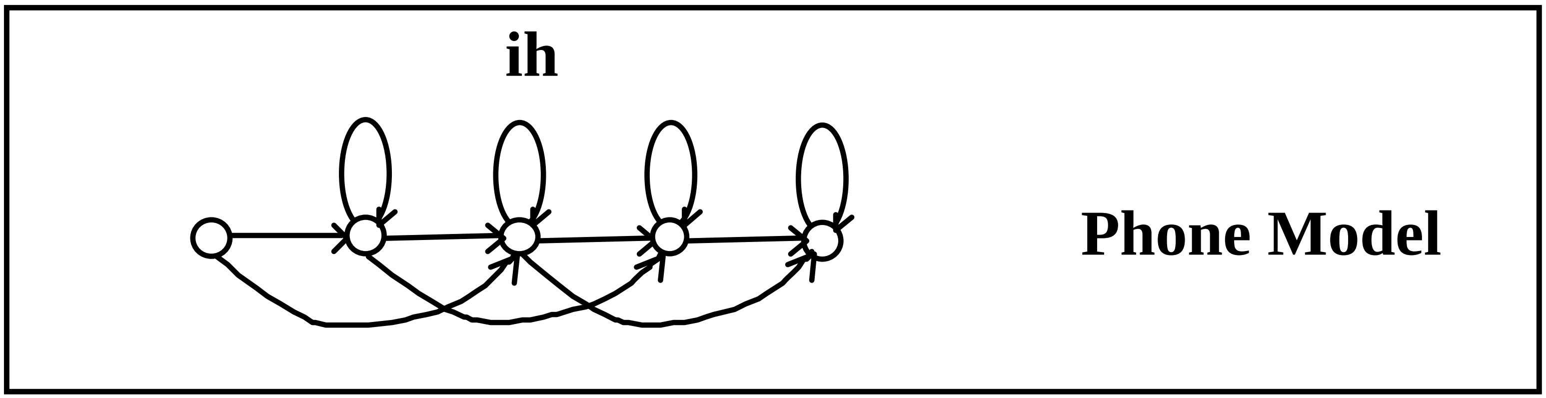
not unique to state sequence

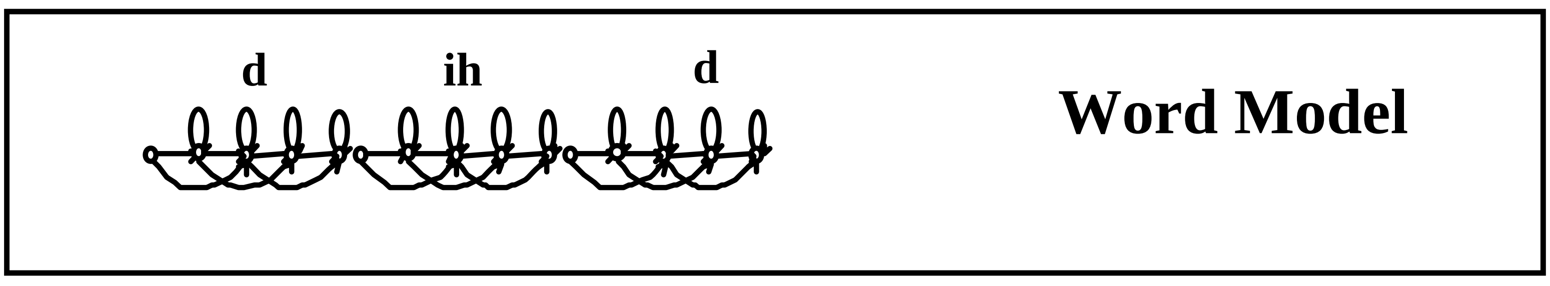
HMMs In Speech Recognition

Represent speech as a sequence of observations

Use HMM to model some unit of speech (phone, word)

Concatenate units into larger units





HMM Problems And Solutions

Evaluation:

- Problem - Compute Probabilty of observation sequence given a model
- Solution - **Forward Algorithm** and **Viterbi Algorithm**

Decoding:

- Problem - Find state sequence which maximizes probability of observation sequence
- Solution - **Viterbi Algorithm**

Training:

- Problem - Adjust model parameters to maximize probability of observed sequences
- Solution - **Forward-Backward Algorithm**

Evaluation

Probability of observation sequence $\mathbf{O} = O_1\, O_2 \cdots O_T$

given HMM model λ is :

$$P(O \mid \lambda) = \sum_{\forall Q} P(O, Q \mid \lambda)$$

$Q = q_0 q_1 \ldots q_T$ is a state sequence

$$= \sum a_{q0q1} b_{q1}(O_1) \times a_{q1q2} b_{q2}(O_2) \cdots a_{qT-1qT} b_{qT}(O_T)$$

Not practical since the number of paths is $O(N^T)$

N = number of states in model
T = number of observations in sequence

The Forward Algorithm

$$\alpha_t(j) = P(O_1\ O_2 \cdots O_t,\ q_t = S_j \mid \lambda)$$

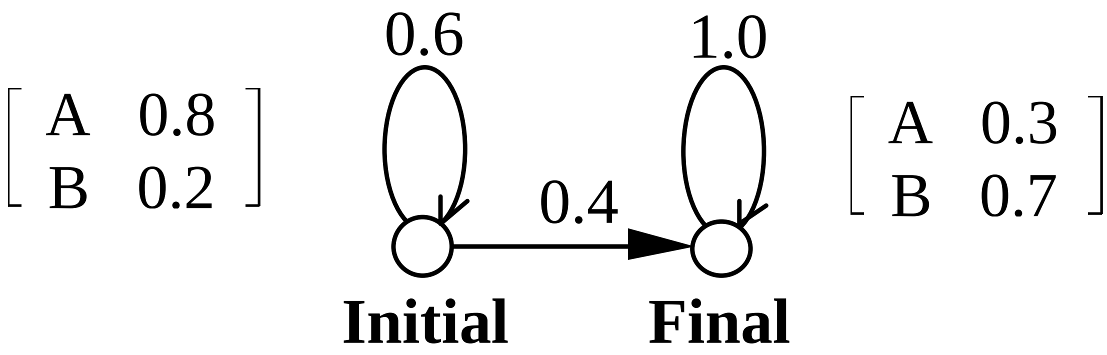
Compute α recursively:

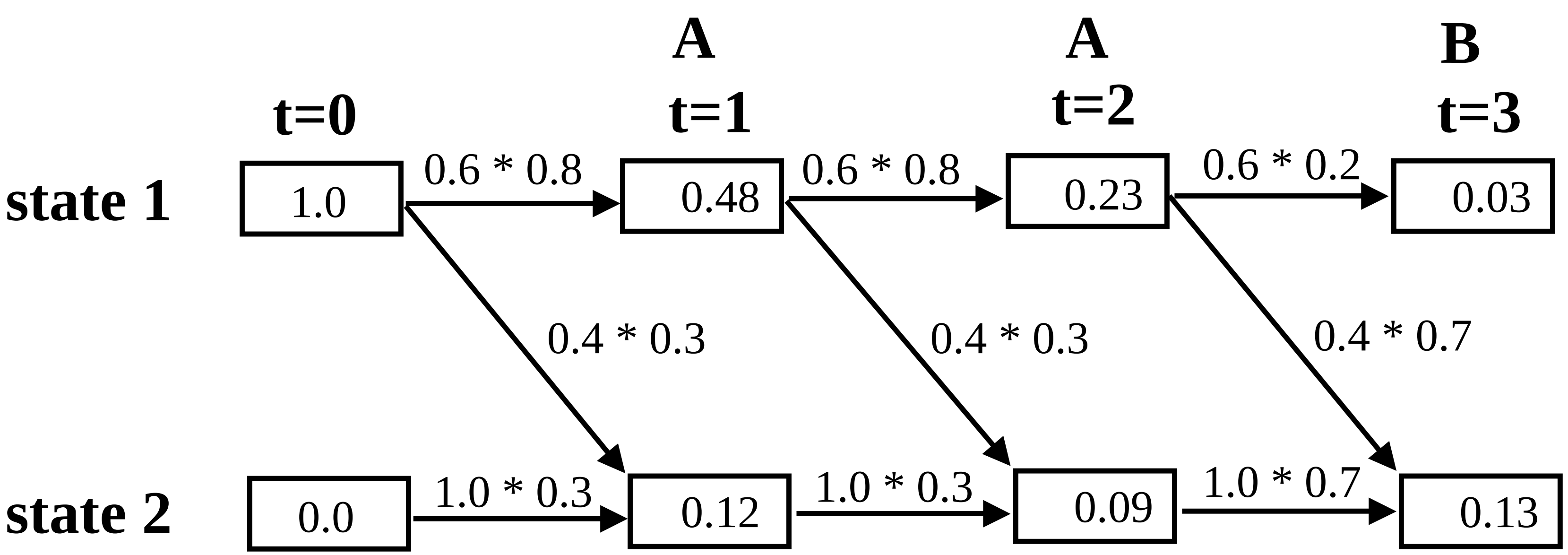
$$\alpha_0(j) = \begin{cases} \mathbf{1} \text{ if j is start state} \\ \mathbf{0} \text{ otherwise} \end{cases}$$

$$\alpha_t(j) = \left[\sum_{i=0}^{N} \alpha_{t-1}(i)\, a_{ij}\right] b_j(O_t) \qquad t > 0$$

$P(O \mid \lambda) = \alpha_T(S_N)$ Computation is $O(N^2 T)$

Forward Trellis





The Backward Algorithm

$$\beta_t(i) = P(O_{t+1}\ O_{t+2} \cdots O_T \mid q_t = S_i , \lambda)$$

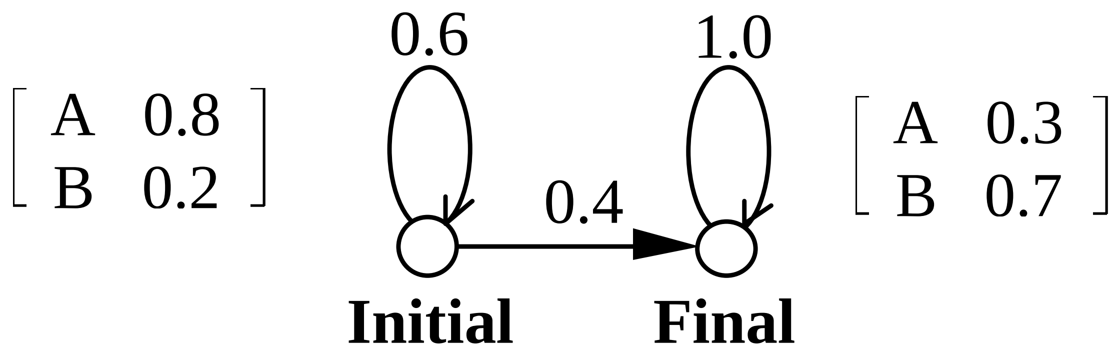
Compute β recursively:

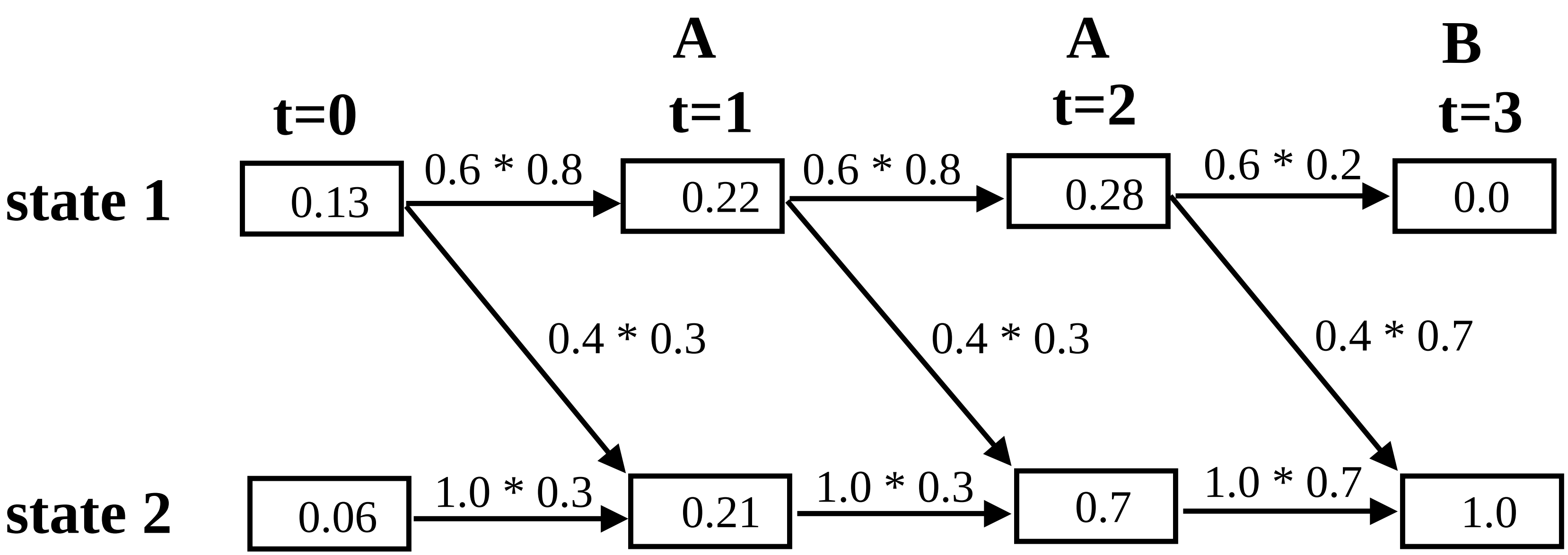
$$\beta_T(i) = \begin{cases} \mathbf{1} & \textbf{if i is end state} \\ \mathbf{0} & \textbf{otherwise} \end{cases}$$

$$\beta_t(i) = \sum_{j=0}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \qquad t < T$$

$$P(O \mid \lambda) = \beta_0(S_0) = \alpha_T(S_N) \qquad \text{Computation is } O(N^2 T)$$

Backward Trellis





The Viterbi Algorithm

For decoding:

Find the state sequence **Q** which maximizes **P(O, Q | λ)**

Similar to Forward Algorithm except **MAX** instead of **SUM**

$$VP_t(i) = MAX_{q_0, \cdots q_{t-1}} \; P(O_1 O_2 \cdots O_t, q_t = i \mid \lambda)$$

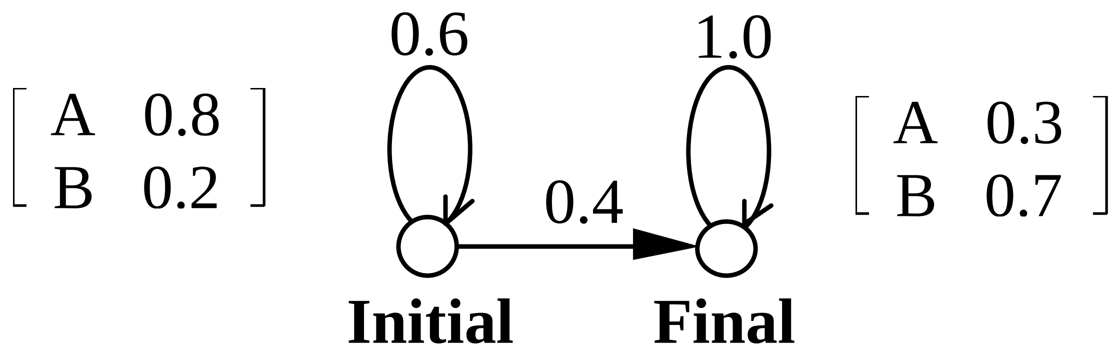
Recursive Computation:

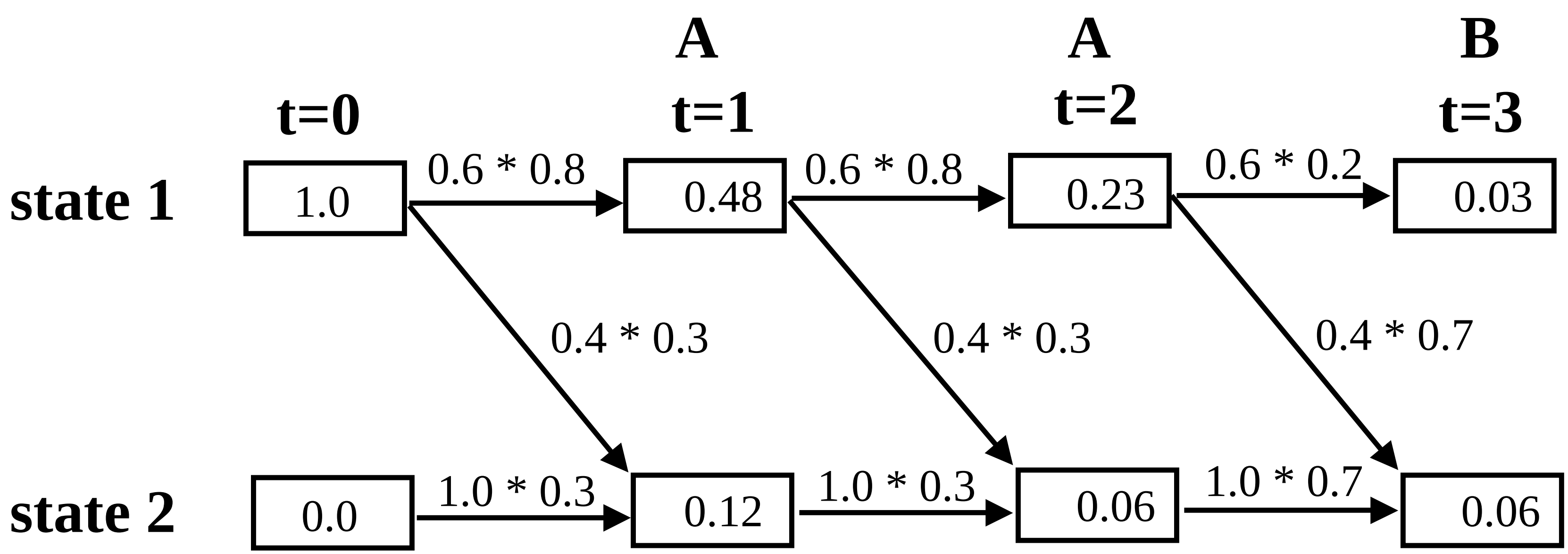
$$VP_t(j) = MAX_{i=0, \cdots, N} \; VP_{t-1}(i) \; a_{ij} b_j(O_t) \qquad t > 0$$

$$P(O, Q \mid \lambda) = VP_T(S_N)$$

Save each maximum for backtrace at end

Viterbi Trellis





Training HMM Parameters

Train parameters of HMM

- Tune λ to maximize $P(O \mid \lambda)$
- No efficient algorithm for global optimum
- Efficient iterative algorithm finds a local optimum

Baum-Welch (Forward-Backward) re-estimation

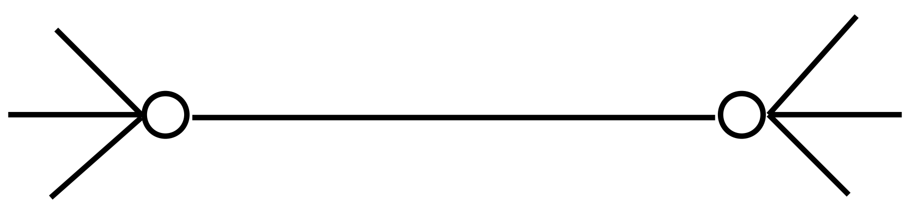
- Compute probabilities using current model λ
- Refine λ —> λ based on computed values
- Use α and β from Forward-Backward

Forward-Backward Algorithm

$\xi_t(i,j) =$ **Probability of transiting from S_i to S_j at time t given O**

$$= P(q_t = S_i,\ q_{t+1} = S_j \mid O, \lambda)$$

$$= \frac{\alpha_t(i)\ a_{ij}\ b_j(O_{t+1})\ \beta_{t+1}(j)}{P(O \mid \lambda)}$$



$\alpha_t(i)$ $\quad a_{ij}b_j(O_{t+1})$ $\quad \beta_{t+1}(j)$

Baum-Welch Reestimation

$$\bar{a}_{ij} = \frac{\text{expected number of trans from } S_i \text{ to } S_j}{\text{expected number of trans from } S_i}$$

$$= \frac{\sum_{t=0}^{T-1} \xi_t(i,j)}{\sum_{t=0}^{T-1} \sum_{j=0}^{N} \xi_t(i,j)}$$

$$\bar{b}_j(k) = \frac{\text{expected number of times in state j with symbol k}}{\text{expected number of times in state j}}$$

$$= \frac{\sum_{t:O_{t+1}=k} \sum_{i=0}^{N} \xi_t(i,j)}{\sum_{t=0}^{T-1} \sum_{i=0}^{N} \xi_t(i,j)}$$

Convergence of FB Algorithm

1. **Initialize λ = (A,B)**
2. **Compute α, β, and ξ**
3. **Estimate $\bar{\lambda} = (\bar{A}, \bar{B})$ from ξ**
4. **Replace λ with $\bar{\lambda}$**
5. **If not converged go to 2**

It can be shown that $P(O \mid \bar{\lambda}) > P(O \mid \lambda)$ unless $\bar{\lambda} = \lambda$

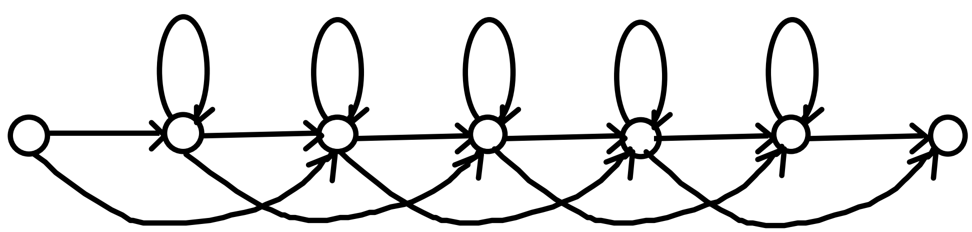
HMMs In Speech Recognition

Represent speech as a sequence of symbols

Use HMM to model some unit of speech (phone, word)

Output Probabilities - Prob of observing symbol in a state

Transition Prob - Prob of staying in or skipping state

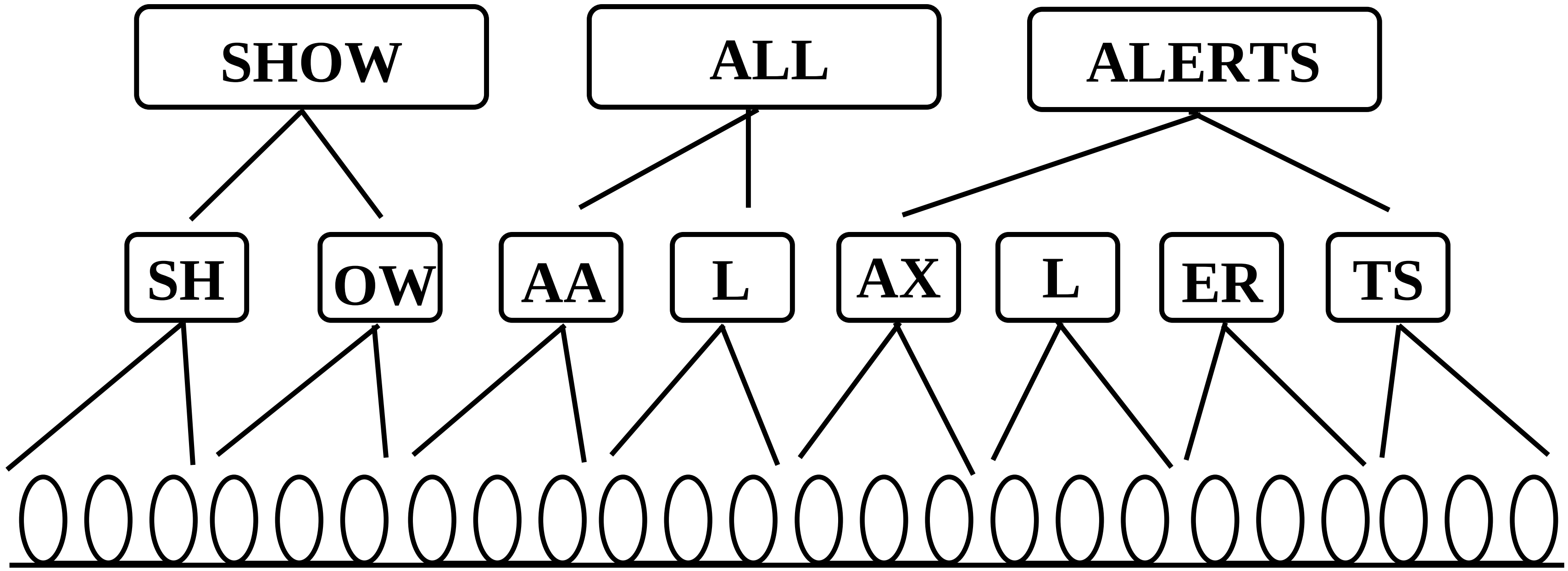


Phone Model

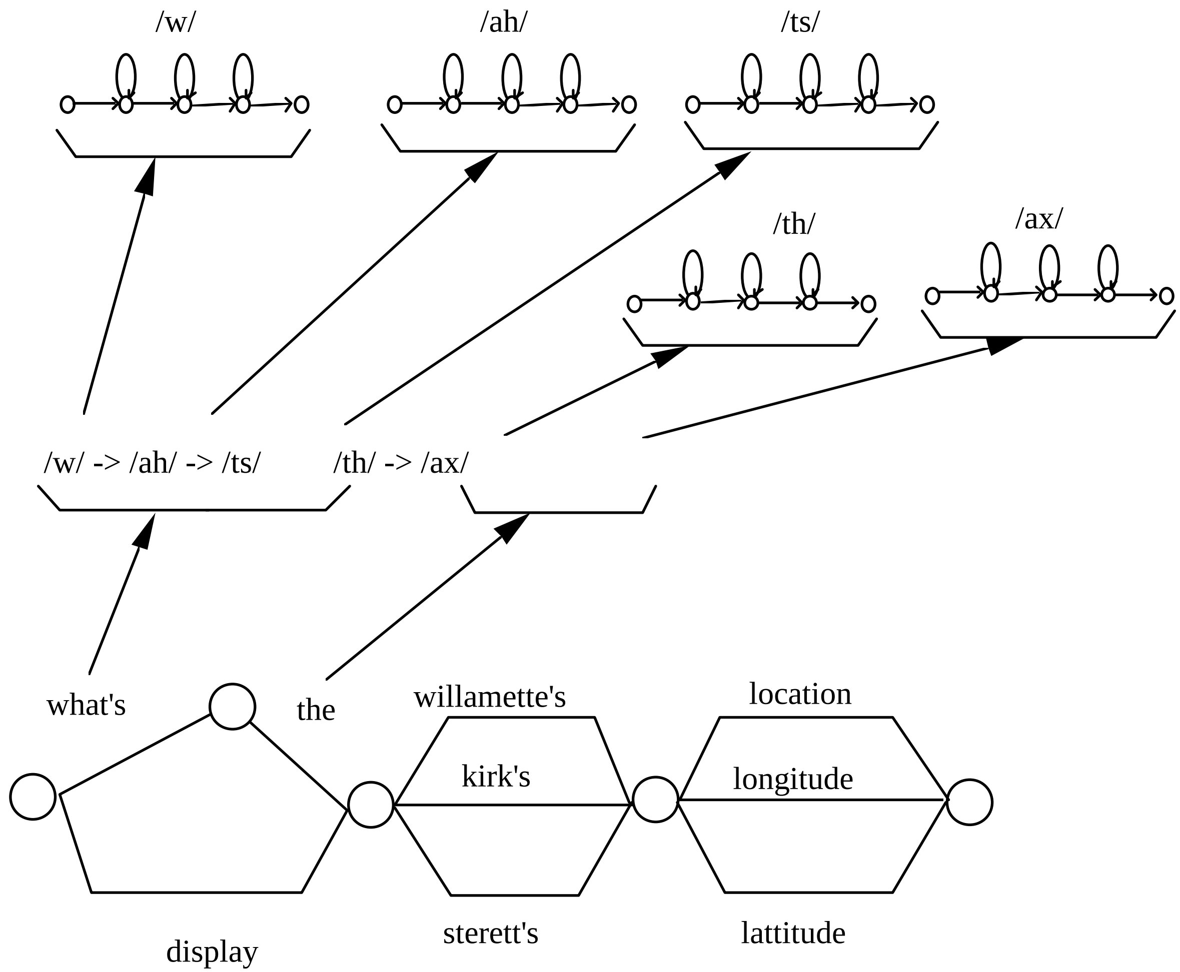
Training HMMs for Continuous Speech

- Use only orthograph transcription of sentence
  - no need for segmented/labelled data
- Concatenate phone models to give word model
- Concatenate word models to give sentence model
- Train entire sentence model on entire spoken sentence

Forward-Backward Training for Continuous Speech



Recognition Search



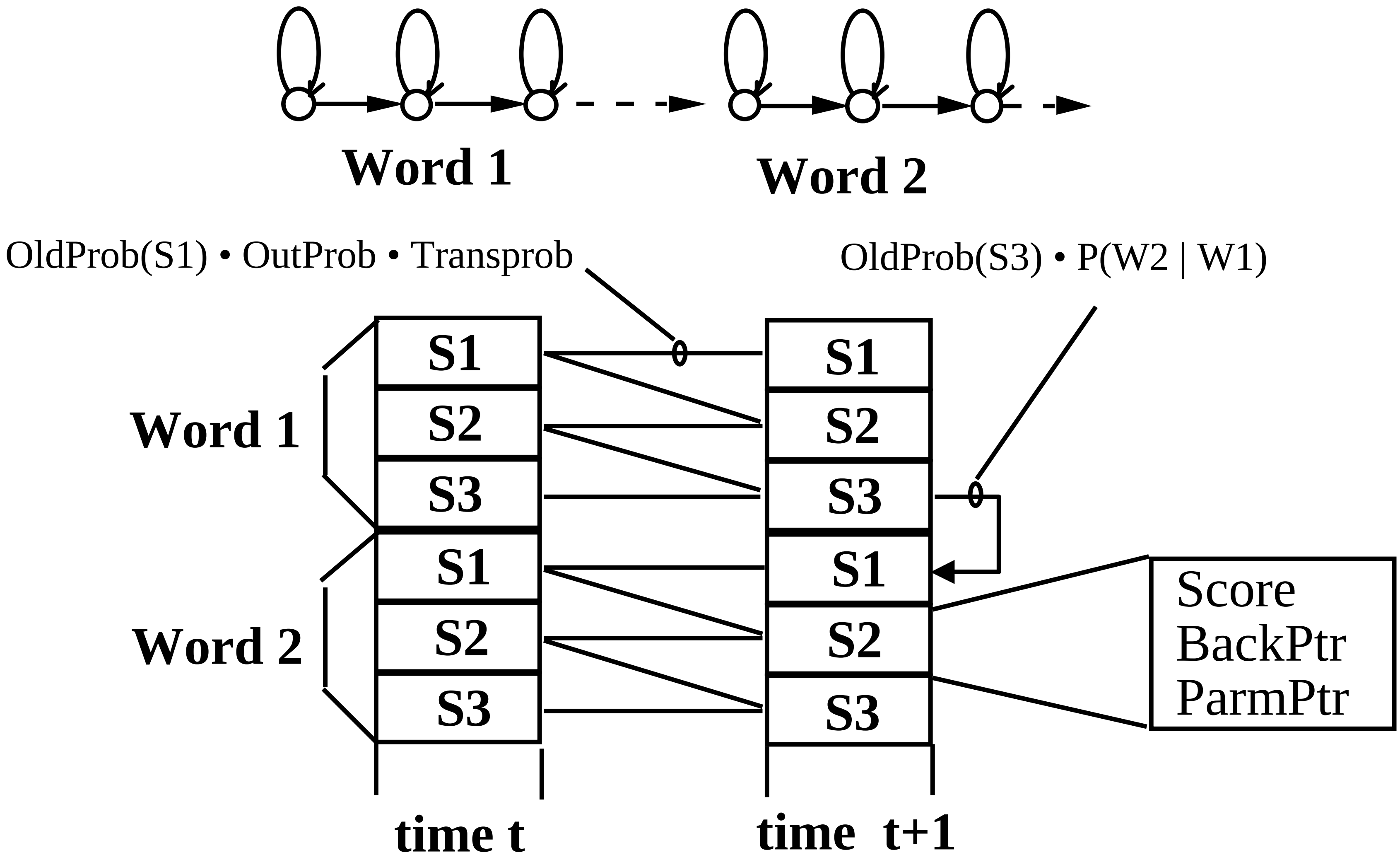
Viterbi Search

- Uses Viterbi decoding
  - Takes MAX, not SUM
  - Finds optimal state sequence $P(O, Q \mid \lambda)$ not optimal word sequence $P(O \mid \lambda)$
- Time synchronous
  - Extends all paths by 1 time step
  - All paths have same length (no need to normalize to compare scores)

Viterbi Search Algorithm

0. Create state list with one cell for each state in system
1. Initialize state list with initial states for time t= 0
2. Clear state list for time t+1
3. Compute within-word transitions from time t to t+1
   - If new state reached, update score and BackPtr
   - If better score for state, update score and BackPtr
4. Compute between word transitions at time t+1
   - If new state reached, update score and BackPtr
   - If better score for state, update score and BackPtr
5. If end of utterance, print backtrace and quit
6. Else increment t and go to step 2

Viterbi Search Algorithm



Viterbi Beam Search

Viterbi Search

All states enumerated

Not practical for large grammars

Most states inactive at any given time

Viterbi Beam Search - prune less likely paths

States worse than threshold range from best are pruned

From and To structures created dynamically - list of active states

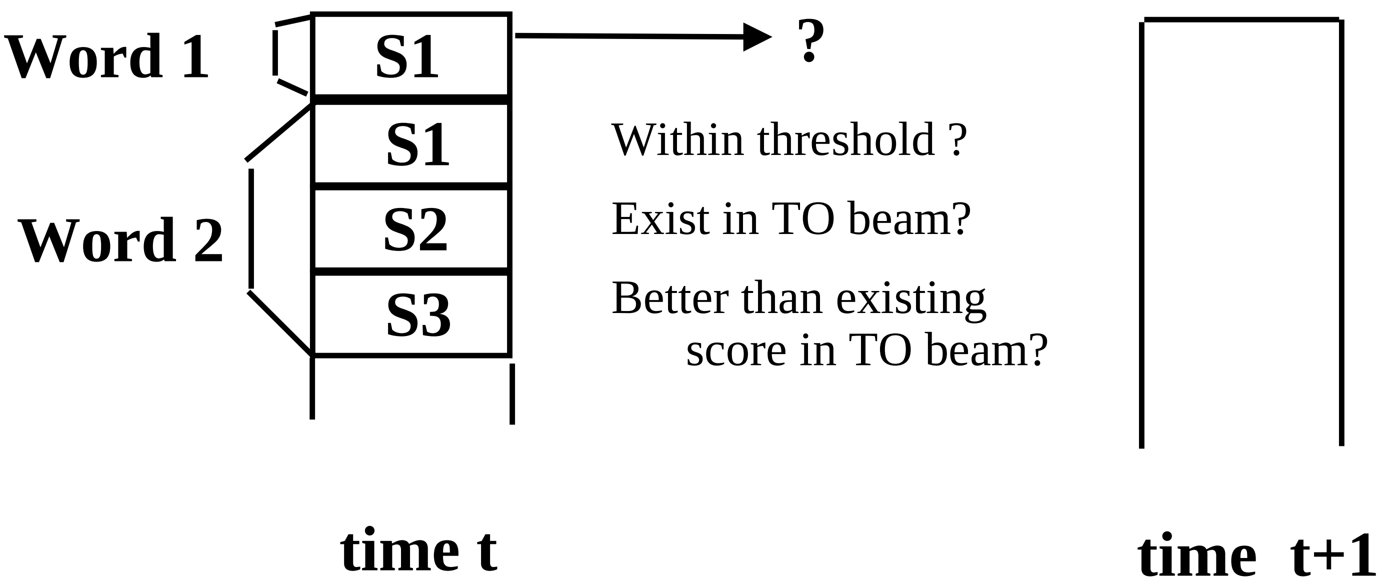
Viterbi Beam Search

FROM BEAM

States within threshold from best state

TO BEAM

Dynamically constructed



Continuous Density HMMs

Model so far has assumed discete observations,
each observation in a sequence was one of a set of M discrete symbols

Speech input must be Vector Quantized in order to provide discrete input.

VQ leads to quantization error

The discrete probability density $b_j(k)$ can be replaced with the continuous probability density $b_j(\mathbf{x})$ where $\mathbf{x}$ is the observation vector

Typically Gaussian densities are used

A single Gaussian is not adequate, so a weighted sum of Gaussians is used to approximate actual PDF

Mixture Density Functions

$b_j(\mathbf{x})$ is the probability density function for state j

$$\boxed{b_j(x) = \sum_{m=1}^{M} c_{jm} N\left[x, \mu_{jm}, U_{jm}\right]}$$

$\mathbf{x}$ = Observation vector $x_1, x_2, \cdots, x_D$

M = Number of mixtures (Gaussians)

c_{jm} = Weight of mixture m in state j where $\sum_{m=1}^{M} c_{jm} = 1$

N = Gaussian density function

μ_{jm} = Mean vector for mixture m, state j

U_{jm} = Covariance matrix for mixture m, state j

Discrete Hmm vs. Continuous HMM

- **Problems with Discrete:**
  - **quantization errors**
  - **Codebook and HMMs modelled separately**

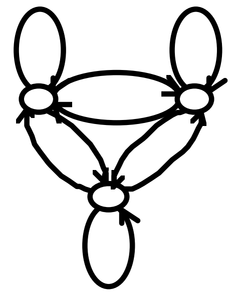
- **Problems with Continuous Mixtures:**
  - **Small number of mixtures performs poorly**
  - **Large number of mixtures increases computation and parameters to be estimated**

    c_{jm}, μ_{jm}, U_{jm} for $j = 1, \cdots, N$ and $m = 1, \cdots, M$

- **Continuous makes more assumptions than Discrete, especially if diagonal covariance pdf**
- **Discrete probability is a table lookup, continuous mixtures require many multiplications**

Model Topologies

**Ergodic - ** Fully connected, each state has transition to every other state



**Left-to-Right - ** Transitions only to states with higher index than current state. Inherently impose temporal order. These most often used for speech.