Digital Speech Processing—Lecture 20

The Hidden Markov Model (HMM)

Lecture Outline

- Theory of Markov Models
  - discrete Markov processes
  - hidden Markov processes
- Solutions to the Three Basic Problems of HMM's
  - computation of observation probability
  - determination of optimal state sequence
  - optimal training of model
- Variations of elements of the HMM
  - model types
  - densities
- Implementation Issues
  - scaling
  - multiple observation sequences
  - initial parameter estimates
  - insufficient training data
- Implementation of Isolated Word Recognizer Using HMM's

Stochastic Signal Modeling

- Reasons for Interest:
  - basis for theoretical description of signal processing algorithms
  - can learn about signal source properties
  - models work well in practice in real world applications
- Types of Signal Models
  - deterministic, parametric models
  - stochastic models

Discrete Markov Processes

System of $N$ distinct states, $\{S_1, S_2, \ldots, S_N\}$

Markov Property:

$P[q_i = S_j \mid q_{i-1} = S_j, q_{i-2} = S_j, \ldots] = P[q_i = S_j \mid q_{i-1} = S_j]$  

Example of Discrete Markov Process

Once each day (e.g., at noon), the weather is observed and classified as being one of the following:

- State 1—Rain (or Snow; e.g. precipitation)
- State 2—Cloudy
- State 3—Sunny

with state transition probabilities:

$A = (a_{ij}) = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$
Discrete Markov Process

Problem: Given that the weather on day 1 is sunny, what is the probability (according to the model) that the weather for the next 7 days will be "sunny-sunny-rain-sunny-cloudy-sunny"?

Solution: We define the observation sequence, $O$, as:

$$O = \{S_3, S_3, S_1, S_1, S_2, S_2, S_3\}$$

and we want to calculate $P(O | \text{Model})$. That is:

$$P(O | \text{Model}) = P[S_3 | S_3]P[S_1 | S_1]P[S_1 | S_2]P[S_2 | S_2]P[S_3 | S_3]$$

Discrete Markov Process

Problem: Given that the model is in a known state, what is the probability it stays in that state for exactly $d$ days?

Solution:

$$P(O | \text{Model}, q_i = S_j) = (a_{ij})^d \cdot (1 - a_j) = p_i(d)$$

Exercise

b) what is the probability that the next 10 tosses will produce the sequence {H H H H H H H H H H}?

SOLUTION:

Similarly:

$$P(H H H H H H H H H H) = (1/2)^{10}$$

Thus a specified run of length 10 is equally likely as a specified run of interlaced H and T.

c) what is the probability that 5 of the next 10 tosses will be tails? What is the expected number of tails over the next 10 tosses?

SOLUTION:

The probability of 5 tails in the next 10 tosses is just the number of observation sequences with 5 tails and 5 heads (in any sequence) and this is:

$$P(5 \text{Tails}) = \frac{(10)!}{5!5!} (1/2)^{10} = \frac{252}{1024} = 0.25$$

since there are (10C5) combinations (ways of getting 5H and 5T) for 10 coin tosses, and each sequence has probability of $(1/2)^{10}$. The expected number of tails in 10 tosses is:

$$E(\text{Number of Tails}) = 10 \cdot \left( \frac{1}{2} \right)^{10} = 5$$

Thus, on average, there will be $5\text{H}$ and $5\text{T}$ in 10 tosses, but the probability of exactly $5\text{H}$ and $5\text{T}$ is only about 0.25.
Coin Toss Models

A series of coin tossing experiments is performed. The number of coins is unknown; only the results of each coin toss are revealed. Thus a typical observation sequence is:

\[ O = O_1, O_2, O_3, \ldots, O_T = HHTTTHTTH\ldots H \]

Problem: Build an HMM to explain the observation sequence.

Issues:
1. What are the states in the model?
2. How many states should be used?
3. What are the state transition probabilities?

Problem: Consider an HMM representation (model \( \lambda \)) of a coin tossing experiment. Assume a 3-state model (corresponding to 3 different coins) with probabilities:

\[
\begin{align*}
P(H) & \begin{array}{c} \text{State 1} \\ \text{State 2} \\ \text{State 3} \end{array} \\
& \begin{array}{c} 0.5 \\ 0.75 \\ 0.25 \end{array} \\
P(T) & \begin{array}{c} 0.5 \\ 0.25 \\ 0.75 \end{array}
\end{align*}
\]

and with all state transition probabilities equal to 1/3. (Assume initial state probabilities of 1/3).

a) You observe the sequence: \( O = \text{HHHHTHTTTT} \) What state sequence is most likely? What is the probability of the observation sequence and this most likely state sequence?

SOLUTION:
Given \( O = \text{HHHHTHTTTT} \), the most likely state sequence is the one for which the probability of each individual observation is maximum. Thus for each H, the most likely state is S2 and for each T the most likely state is S3. Thus the most likely state sequence is:

\( S = S_2, S_2, S_2, S_1, S_1, S_1, S_1, S_3, S_3, S_3, S_3, S_3 \)

The probability of \( O \) and \( S \) (given the model) is:

\[
\lambda \left( O, S \mid \lambda \right) = \left( \frac{1}{3} \right)^{10} \left( 0.75 \right)^{10} \left( 0.5 \right)^{10} \]

b) What is the probability that the observation sequence came entirely from state 1?

SOLUTION:
The probability of \( O \) given that \( S \) is of the form:

\[
\hat{S} = S_1, S_1, S_1, S_1, S_1, S_1, S_1, S_1, S_1, S_1, S_1, S_1
\]

is:

\[
P(O \mid \hat{S} \mid \lambda) = (0.5)^{10} \left( \frac{1}{3} \right)^{10}
\]

The ratio of \( P(O \mid S \mid \lambda) \) to \( P(O \mid \hat{S} \mid \lambda) \) is:

\[
R = \frac{P(O \mid S \mid \lambda)}{P(O \mid \hat{S} \mid \lambda)} = \frac{1}{57.67}
\]
**Coin Toss Models**

c) Consider the observation sequence:
\[ \hat{O} = HTTHHTTH \]
How would your answers to parts a and b change?

**SOLUTION:**
Given which has the same number of H's and T's, the answers to parts a and b would remain the same as the most likely states occur the same number of times in both cases.

---

**Coin Toss Problem Solution**

**SOLUTION:**
The new probability of O and S becomes:
\[ P(O,S | \lambda') = (0.75)^3 \cdot \left( \frac{1}{3} \right) \cdot \left( \frac{1}{3} \right) \cdot \left( \frac{1}{3} \right) \]
The new probability of O and \( \hat{S} \) becomes:
\[ P(O,\hat{S} | \lambda') = (0.50)^3 \cdot \left( \frac{1}{3} \right) \cdot \left( \frac{1}{3} \right) \]
The ratio is:
\[ R = \left( \frac{3}{2} \right)^3 \cdot \left( \frac{1}{3} \right) \cdot \left( \frac{1}{3} \right) = 1.36 \cdot 10^{-3} \]

---

**Elements of an HMM**

1. \( N \), number of states in the model
   - states, \( S = \{S_1, S_2, \ldots, S_N\} \)
   - state at time \( t, q_t \in S \)
2. \( M \), number of distinct observation symbols per state
   - observation symbols, \( V = \{v_1, v_2, \ldots, v_M\} \)
   - observation at time \( t, O_t \in V \)
3. State transition probability distribution, \( A = \{a_{ij}\} \),
   \[ a_{ij} = P(q_{t+1} = S_j | q_t = S_i), \quad 1 \leq j \leq N \]
4. Observation symbol probability distribution in state \( j \)
   \[ B = \{b_j(k)\} \]
   \[ b_j(k) = P(O_t = v_k | q_t = S_j), \quad 1 \leq j \leq N, 1 \leq k \leq M \]
5. Initial state distribution, \( \Pi = \{\pi_i\} \)
   \[ \pi_i = P(q_1 = S_i), \quad 1 \leq i \leq N \]
HMM Generator of Observations

1. Choose an initial state, \( q_0 \), according to the initial state distribution, \( \pi \).
2. Set \( t = 1 \).
3. Choose \( q_t = v_t \) according to the symbol probability distribution in state \( q_t \), namely \( b(q_t) \).
4. Transit to a new state, \( q_{t+1} \), according to the state transition probability distribution for state \( q_t \), namely \( \lambda(q_t, q_{t+1}) \).
5. Set \( t = t + 1 \); return to step 3 if \( t = T \); otherwise terminate the procedure.

Three Basic HMM Problems

Problem 1—Given the observation sequence, \( O = \{O_t \}_{t=1}^T \), and a model \( \lambda = \{A, B, \pi\} \), how do we (efficiently) compute \( P(O|\lambda) \), the probability of the observation sequence?

Problem 2—Given the observation sequence, \( O = \{O_t \}_{t=1}^T \), how do we choose a state sequence \( Q = \{q_t \}_{t=1}^T \) which is optimal in some meaningful sense?

Problem 3—How do we adjust the model parameters \( \lambda = \{A, B, \pi\} \) to maximize \( P(O|\lambda) \)?

Interpretation:

Problem 1—Evaluation or scoring problem.
Problem 2—Learn structure problem.
Problem 3—Training problem.

Solution to Problem 1—\( P(O|\lambda) \)

Consider the state sequence (there are \( N^T \) such sequences):

\[ Q = \{q_t \}_{t=1}^T \]

Then

\[ P(O|Q) = b_1(O_1) b_2(O_2) \cdots b_T(O_T) \]

and

\[ P(O|Q) = \sum_{Q'} P(Q|Q') P(O|Q') \]

Finally

\[ P(O|\lambda) = \sum_{Q} P(Q|\lambda) \]

Calculations required = \( 27 \cdot N^T \cdot N \cdot 5 \cdot T = 100 \approx 2 \cdot 100 \cdot 5^{18} \)

\( \approx 10^{23} \) computations!

The “Forward” Procedure

Consider the forward variable, \( \alpha(i) \), defined as the probability of the partial observation sequence (until time \( t \)) and state \( q_i \) at time \( t \), given the model, i.e.,

\[ \alpha(i) = P(O_1, O_2, \ldots, O_t | q_i = S_t) \]

Inductively solve for \( \alpha(i) \) as:

1. Initialization

\[ \alpha(0) = \pi_0 b_1(O_1) \text{ for } 1 \leq i \leq N \]

2. Induction

\[ \alpha(i) = \sum_{j \in Q} \lambda(j, i) \alpha(j) b_i(O_i) \text{ for } 1 \leq t \leq T - 1 \text{ and } 1 \leq j \leq N \]

3. Termination

\[ P(O_1, O_2, \ldots, O_T | q_i = S_T) = \sum_{j \in Q} \lambda(j, i) \alpha(j) \]

Computation: \( N^T \) versus \( 27 N^2 \cdot N \cdot 5 \cdot T = 100 \approx 2500 \) versus \( 10^{23} \)

The “Backward” Algorithm

Consider the backward variable, \( \beta(i) \), defined as the probability of the partial observation sequence from \( t = 1 \) to the end, given state \( q_i \) at time \( t \), and the model, i.e.,

\[ \beta(i) = P(O_{t+1}, O_{t+2}, \ldots, O_T | q_i = S_t) \]

Inductive Solution:

1. Initialization

\[ \beta(i) = 1 \text{ for } 1 \leq i \leq N \]

2. Induction

\[ \beta(i) = \sum_{j \in Q} \lambda(j, i) b_j(O_j) \beta(j) \text{ for } t = T - 1, t - 2, \ldots, 1 \text{ and } 1 \leq i \leq N \]

- \( N^T \) calculations, same as in forward case.
Solution to Problem 2—Optimal State Sequence

1. Choose states, $q_i$, which are individually most likely to maximize expected number of correct individual states
2. Choose states, $q_i$, which are pair-wise most likely to maximize expected number of correct state pairs
3. Choose states, $q_i$, which are triple-wise most likely to maximize expected number of correct state triples
4. Choose states, $q_i$, which are T-wise most likely to find the single best state sequence which maximizes $P(O, q_i)$

This solution is often called the Viterbi state sequence because it is found using the Viterbi algorithm.

Maximize Individual States

We define $\gamma_t(i)$ as the probability of being in state $S_i$ at time $t$, given the observation sequence, and the model, i.e.,

$$\gamma_t(i) = \frac{P(q_i = S_i | O, \lambda) \cdot P(q_i = S_i | O, \lambda)}{P(O, \lambda)}$$

then

$$\gamma_t(i) = \frac{P(q_i = S_i | O, \lambda) \cdot P(O, \lambda)}{\sum_{q_i} P(q_i = S_i | O, \lambda) \cdot P(O, \lambda)}$$

with $\sum_{q_i} \gamma_t(i) = 1 \forall t$

then

$$q^*_t = \arg\max_{q_i} \gamma_t(i), \quad 1 \leq t \leq T$$

Problem: $q^*_t$ need not obey state transition constraints.

Best State Sequence—The Viterbi Algorithm

Define $\delta_t(i)$ as the highest probability along a single path, at time $t$, which accounts for the first $t$ observations, i.e.,

$$\delta_t(i) = \max_{q_i} P(q_t, q_{t-1}, q_{t-2}, \ldots, q_1 = i, O_t | O_{t-1}, \lambda)$$

We must keep track of the state sequence which gave the best path, at time $t$, to state $i$. We do this in the array $\psi_t(i)$.

The Viterbi Algorithm

Step 1 - Initialization

$\delta_0(i) = b_i(O_0), \quad 1 \leq i \leq N$

$\psi_0(i) = 0, \quad 1 \leq i \leq N$

Step 2 - Recursion

$\delta_t(i) = \max_{q_j} \{ \delta_t(j) b_j(O_t) \} , \quad 2 \leq t \leq T, 1 \leq j \leq N$

$\psi_t(i) = \arg\max_{q_j} \{ \delta_t(j) b_j(O_t) \} , \quad 2 \leq t \leq T, 1 \leq j \leq N$

Step 3 - Termination

$P^* = \max_{1 \leq i \leq N} \delta_T(i)$

$q^*_T = \arg\max_{1 \leq i \leq N} \delta_T(i)$

Step 4 - Path (State Sequence) Backtracking

$q^*_t = \psi_t(q^*_t), \quad t = T - 1, T - 2, \ldots, 1$

Calculation = $N^T$ operations ($\ast$)

Problem

Given the model of the coin toss experiment used earlier (i.e., 3 different coins) with probabilities:

<table>
<thead>
<tr>
<th></th>
<th>State 1</th>
<th>State 2</th>
<th>State 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P(H)$</td>
<td>0.5</td>
<td>0.75</td>
<td>0.25</td>
</tr>
<tr>
<td>$P(T)$</td>
<td>0.5</td>
<td>0.25</td>
<td>0.75</td>
</tr>
</tbody>
</table>

with all state transition probabilities equal to $1/3$, and with initial state probabilities equal to $1/3$. For the observation sequence $O = H H H H T T T T$, find the Viterbi path of maximum likelihood.

Alternative Viterbi Implementation

$\delta_t = \log\{\cdot\}, \quad 0 \leq t \leq T$

$\delta(O) = \log\{\cdot\}, \quad 1 \leq i \leq N$

Step 1 - Initialization

$\delta_0(i) = \log b_i(O_0), \quad 1 \leq i \leq N$

$\psi_0(i) = 0, \quad 1 \leq i \leq N$

Step 2 - Recursion

$\delta_t(i) = \max_{q_j} \{ \delta_t(j) + \log b_j(O_t) \}, \quad 2 \leq t \leq T, 1 \leq j \leq N$

$\psi_t(i) = \arg\max_{q_j} \{ \delta_t(j) + \log b_j(O_t) \}, \quad 2 \leq t \leq T, 1 \leq j \leq N$

Step 3 - Termination

$P^* = \max_{1 \leq i \leq N} \delta_T(i)$

$q^*_T = \arg\max_{1 \leq i \leq N} \delta_T(i)$

Step 4 - Backtracking

$q^*_t = \psi_t(q^*_t), \quad t = T - 1, T - 2, \ldots, 1$

Calculation = $N^T$ additions
Problem Solution
Since all $a_i$ terms are equal to 0.25, we can omit these terms (as well as the initial state probability term) giving:

The recursion for $\alpha(t)$ gives $i=1, ..., T$:

- $\alpha_1(t) = 0.5$, $\alpha_2(t) = 0.75$, $\alpha_3(t) = 0.25$

The training problem

- Solution to Problem 3—the Training Problem
  - No globally optimal solution is known
  - All solutions yield local optima
    - Can get solution via gradient techniques
    - Can use a re-estimation procedure such as the Baum-Welch or EM method
  - Consider re-estimation procedures
    - Basic idea: Given a current model estimate $\lambda$, compute expected values of model events, then refine the model based on the computed values

Define $\zeta(i,j)$, the probability of being in state $S_i$ at time $t$, and state $S_j$ at time $t+1$, given the model and the observation sequence. I.e.,

$\zeta(i,j) = P(q_t = S_i, q_{t+1} = S_j | O, \lambda)$

Re-estimation Formulas

$\pi_i$: Expected number of times in state $S_i$ at $t$ = 1

$\gamma(i)$: Expected number of observations in state $S_i$

$\delta_{ij}$: Expected number of transitions from state $S_i$ to state $S_j$

$\xi(i,j)$: Expected number of times in state $S_i$ with symbol $v_k$

$E(i,j)$: Expected number of times in state $S_i$ with symbol $v_k$

$E_j(k)$: Expected number of times in state $j$ with symbol $v_k$

Solution to Problem 3—the Training Problem

- The initial model, $\lambda$, defines a critical point of the likelihood function, in which case $\lambda = \lambda_0$.
- The re-estimated model is more likely than model $\lambda_0$ in the sense that $P(O | \lambda) > P(O | \lambda_0)$, i.e., we have found a new model $\lambda$ from which the observation sequence is more likely to have been produced.

Conclusion: Iteratively use $\lambda$ in place of $\lambda_0$, and repeat the re-estimation until some limiting point is reached. The resulting model is called the maximum likelihood (ML) HMM.
Re-estimation Formulas

1. The re-estimation formulas can be derived by maximizing the auxiliary function $Q(\lambda, T)$ over $T$, i.e.,

$$Q(\lambda, T) = \sum P(O, q|\lambda) \log[P(O, q|\lambda)]$$

It can be proved that:

$$\max Q(\lambda, T) \Rightarrow P(O|T) \succeq P(O|\lambda)$$

Eventually the likelihood function converges to a critical point

2. Relation to EM algorithm:
   - **E (Expectation) step** is the calculation of the auxiliary function, $Q(\lambda, T)$
   - **M (Modification) step** is the maximization over $\lambda$

$\lambda \Rightarrow \max P(O, q|\lambda) \quad \lambda \geq \lambda$ (ii)

Eventually the likelihood function converges to a critical point

Notes on Re-estimation

1. Stochastic constraints on $s_n, \lambda_n, b(k)$ are automatically met, i.e.,

$$\sum_{i,j} 1, 1 \quad 0, 1$$

2. At the critical points of $P = P(\lambda|O)$, then

$$\pi_n = \frac{\pi_n}{\sum_{i} \pi_i} \quad \lambda_n = \frac{\lambda_n}{\sum_{i} \lambda_i}$$

$\Rightarrow$ at critical points, the re-estimation formulas are exactly correct.

Types of HMM

1. Types of HMM—model structures
2. Continuous observation density models—mixtures
3. Autoregressive HMM’s—LPC links
4. Null transitions and tied states
5. Inclusion of explicit state duration density in HMM’s
6. Optimization criterion—ML, MMI, MDI

Continuous Observation Density HMM’s

Most general form of pdf with a valid re-estimation procedure is:

$$b(x) = \sum_{m,j} \pi_{jm} \mu_{jm} x^m | U_{jm}$$

$x = \text{observation vector} = [x_1, x_2, ..., x_m]$  
$M = \text{number of mixture densities}$  
$N = \text{any log-concave or elliptically symmetric density (e.g., a Gaussian)}$  
$\mu_{jm} = \text{mean vector for mixture m, state j}$  
$U_{jm} = \text{covariance matrix for mixture m, state j}$  
$c_m = \text{gain of m-th mixture in state j}$  
$\sum_{m,j} c_m = 1 \quad 1 \leq m \leq M \quad 1 \leq j \leq N$

$\sum_{j} b(j|x) = 1 \quad 1 \leq j \leq N$
Application of Autoregressive HMM

Consider an observation vector \( O = (x_1, x_2, \ldots, x_n) \) where each \( x_i \) is a waveform sample, and \( O \) represents a frame of the signal (e.g., \( K = 256 \) samples). We assume \( x_i \) is related to previous autoregressive or predictor coefficients.

Each mixture characterized by predictor vector or by autocorrelation vector from which predictor vector can be derived. Re-estimation formulas for \( r_i \) are:

\[
r_i = \frac{\sum_{j} r_{ij}(k)}{\sum_{j} r_{ij}(k)}
\]

\[
r_{ij}(k) = \frac{\alpha_i(j)(k) t_{ij}(k)}{\sum_{i} \alpha_i(j)(k) t_{ij}(k)}
\]

\[
r_i = \frac{\sum_{j} r_{ij}(k)}{\sum_{j} r_{ij}(k)}
\]

\[
r_{ij}(k) = \frac{\alpha_i(j)(k) t_{ij}(k)}{\sum_{i} \alpha_i(j)(k) t_{ij}(k)}
\]

Null Transitions and Tied States

**Null Transitions**: transitions which produce no output, and take no time, denoted by \( \phi \)

**Tied States**: sets up an equivalence relation between HMM parameters in different states

- number of independent parameters of the model reduced
- parameter estimation becomes simpler
- useful in cases where there is insufficient training data for reliable estimation of all model parameters
Null Transitions

Inclusion of Explicit State Duration Density

For standard HMMs, the duration density is:
\[ p(d) = \text{probability of exactly } d \text{ observations in state } S \]
\[ = (A_{1,d}^T)^{d-1} A_{1,d} \]

With arbitrary state duration density, \( p(d) \), observations are generated as follows:
1. an initial state, \( q_0 = S_1 \), is chosen according to the initial state distribution, \( \pi_1 \)
2. a duration \( d \), is chosen according to the state duration density \( p_d(\cdot) \)
3. observations \( O_1 \ldots O_d \) are chosen according to the joint density \( b_{i_1}(O_1) \ldots b_{i_d}(O_d) \)

Generally we assume independence, so
\[ b_{i_1}(O_1) \ldots b_{i_d}(O_d) = \prod_{i=1}^{d} b_i(O_i) \]
4. the next state, \( q_{d+1} = S_i \), is chosen according to the state transition probabilities, \( a_{i,j} \), with the constraint that \( a_{i,j} = 0 \), i.e., no transition back to the same state can occur.

Explicit State Duration Density

For standard HMMs, the duration density is:
\[ p(d) = \text{probability of exactly } d \text{ observations in state } S \]
\[ = (A_{1,d}^T)^{d-1} A_{1,d} \]

With arbitrary state duration density, \( p(d) \), observations are generated as follows:
1. an initial state, \( q_0 = S_1 \), is chosen according to the initial state distribution, \( \pi_1 \)
2. a duration \( d \), is chosen according to the state duration density \( p_d(\cdot) \)
3. observations \( O_1 \ldots O_d \) are chosen according to the joint density \( b_{i_1}(O_1) \ldots b_{i_d}(O_d) \)

Generally we assume independence, so
\[ b_{i_1}(O_1) \ldots b_{i_d}(O_d) = \prod_{i=1}^{d} b_i(O_i) \]
4. the next state, \( q_{d+1} = S_i \), is chosen according to the state transition probabilities, \( a_{i,j} \), with the constraint that \( a_{i,j} = 0 \), i.e., no transition back to the same state can occur.

Explicit State Duration Density

Then we have
\[ w(i) = \sum_j a_{i,j} b_j(d) P(O_1 \ldots O_d | q_j, \lambda_0) \]
\[ = a_{i,j} b_j(d) P(O_1 \ldots O_d | q_j, \lambda_0) \]
\[ = a_{i,j} b_j(d) \prod_j b_j(O_j) \]
\[ \text{Initialization of } w(i) \]
\[ w(1) = \max_j a_{1,j} \prod_j b_j(O_j) \]
\[ \text{Basic Recursion :} \]
\[ w(i) = \max_j a_{i,j} \prod_j b_j(O_j) \]
\[ \text{Storage required for } x_{i-1} \Rightarrow N \cdot D \text{ locations} \]
\[ \text{Maximization involves all terms—not just old } x_i \text{s and } a_{i,j} \text{ as in previous case} \Rightarrow \text{significantly larger computational load} \]
\[ (2^D/2)^N \Rightarrow \text{computations involving } b_i(O) \]

Example: \( N = 5, D = 20 \)

<table>
<thead>
<tr>
<th>implicit duration</th>
<th>explicit duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>storage</td>
<td>5</td>
</tr>
<tr>
<td>computation</td>
<td>2500</td>
</tr>
<tr>
<td></td>
<td>500,000</td>
</tr>
</tbody>
</table>
Issues with Explicit State Duration Density

1. quality of signal modeling is often improved significantly
2. significant increase in the number of parameters per state (D duration estimates)
3. significant increase in the computation associated with probability calculation (D² / 2)
4. insufficient data to give good (d) estimates

Alternatives:
1. use parametric state duration density
   \( f(d; \mu, \sigma) \) — Gaussian
   \( f(d; \alpha, \beta) \) — Gamma

Alternatives to ML Estimation

Assume we wish to design V different HMMs, \( \lambda_1, \lambda_2, \ldots, \lambda_V \). Normally we design each HMM, \( \lambda_i \), based on a training set of observations, \( O' \), using a maximum likelihood (ML) criterion, i.e.,

\[
\lambda_i = \arg \max P(O' | \lambda_i)
\]

Consider the mutual information, \( I_{ij} \), between the observation sequence, \( O' \), and the complete set of models, \( i = 1, 2, \ldots, V \),

\[
I_{ij} = \sum_k \log P(O'_{ik}) - \log \sum_k P(O'_{ik})
\]

Consider maximizing \( I_{ij} \) over \( \lambda_i \), giving

\[
\lambda_i = \arg \max I_{ij} = \arg \max \sum_k \log P(O'_{ik}) - \log \sum_k P(O'_{ik})
\]

• choose \( \lambda_i \) so as to separate the correct model, \( \lambda_i \), from all other models, as much as possible, for the training set, \( O' \).

Comparison of HMM’s

Thus the two models have very different \( A \) and \( B \) matrices, but are equivalent in the sense that all symbol probabilities (averaged over time) are the same.

We generalize the concept of model distance (dis-similarity) by defining a distance measure, \( D(\lambda_i, \lambda_j) \) between two Markov sources, \( \lambda_i \) and \( \lambda_j \), as

\[
D(\lambda_i, \lambda_j) = \frac{1}{2} [D(\lambda_i, \lambda_j) + D(\lambda_j, \lambda_i)]
\]

where \( O'_{ij} \) is a sequence of observations generated by model \( \lambda_j \), and scored by both models.

We symmetrize \( D \) by using the relation:

\[
D(\lambda_i, \lambda_j) = \frac{1}{2} [D(\lambda_i, \lambda_j) + D(\lambda_j, \lambda_i)]
\]

Implementation Issues for HMM’s

1. Scaling—to prevent underflow and/or overflow.
2. Multiple Observation Sequences—to train left-right models.
3. Initial Estimates of HMM Parameters—to provide robust models.
4. Effects of Insufficient Training Data
For left-right models, we need to use multiple sequences of observations for training.

Scaling

- \( a_i(l) \) is a sum of a large number of terms, each of the form:
  \[ \prod_{k=1}^{t} a_k(x_k, b_k(O_k)) \]
- since each \( a \) and \( b \) term is less than 1, as \( t \) gets larger, \( a_i(l) \)
exponentially heads to 0. Thus scaling is required to prevent underflow.
- consider scaling \( a_i(l) \) by the factor
  \[ c_i = \frac{1}{\sum a_i(l)} \]
  independent of \( t \)
- we denote the scaled \( a \)'s as:
  \[ \hat{a}_i(l) = c_i a_i(l) \]
  \[ \sum \hat{a}_i(l) - 1 \]

Scaling

- for scaling the \( \beta(l) \) terms we use the same scale factors as for the \( a_i(l) \) terms, i.e.,
  \[ \hat{\beta}(l) = c \beta(l) \]
  since the magnitudes of the \( a \) and \( \beta \) terms are comparable.
- the re-estimation formula for \( a \) in terms of the scaled \( a \)'s and \( \beta \)'s is:
  \[ \hat{a}_i(l) = \frac{\sum \hat{a}_i(l) s_i(b_i(\Omega_i)) \hat{\beta}_i(l)}{\sum \hat{a}_i(l) s_i(b_i(\Omega_i)) \hat{\beta}_i(l)} \]
  \[ \hat{\beta}_i(l) = \frac{\sum \hat{a}_i(l) s_i(b_i(\Omega_i)) \hat{\beta}_i(l)}{\sum \hat{a}_i(l) s_i(b_i(\Omega_i)) \hat{\beta}_i(l)} \]
- we have
  \[ \hat{a}_i(l) = \prod_{j=1}^{t} c_j a_i(l) - C_i a_i(l) \]
  \[ \hat{\beta}_i(l) = \prod_{j=1}^{t} c_j \beta_i(l) - D_i \beta_i(l) \]

Multiple Observation Sequences

For left-right models, we need to use multiple sequences of observations for training.

\[ \alpha = \{ \alpha^1, \alpha^2, \ldots, \alpha^K \} \]

We wish to maximize the probability

\[ P(O | l) \prod_{i=1}^{K} P(\alpha^i | \lambda) \prod_{t=1}^{T} P(o_t | \alpha_t^t) \]

scaling requires:

\[ \pi = \sum_{i=1}^{K} \frac{1}{\prod_{j=1}^{T} c_j (a_i^t) (b_i^t)} \]

\[ \pi = \frac{1}{\sum_{i=1}^{K} \prod_{j=1}^{T} c_j (a_i^t) (b_i^t)} \]

Initial Estimates of HMM Parameters

\( N \) -- choose based on physical considerations
\( M \) -- choose based on model fits
\( \pi \) -- random or uniform (\( \pi_i \neq 0 \))
\( a_i \) -- random or uniform (\( a_i \neq 0 \))
\( b_i(k) \) -- random or uniform (\( b_i(k) \geq \varepsilon \))
\( b_i(O) \) -- need good initial estimates of mean vectors;
need reasonable estimates of covariance matrices
Effects of Insufficient Training Data

Insufficient training data leads to poor estimates of model parameters. Possible Solutions:
1. use more training data--often this is impractical
2. reduce the size of the model--often there are physical reasons for keeping a chosen model size
3. add extra constraints to model parameters
   \[
   b_j(k) \geq \varepsilon
   \]
   \[
   U_j(r,r) \geq \delta
   \]
   - often the model performance is relatively insensitive to exact choice of \(\varepsilon, \delta\)
4. method of deleted interpolation
   \[
   \hat{\varepsilon}_{\text{rel}}^t = (1-\delta^t)\varepsilon^t
   \]

Methods for Insufficient Data

Performance insensitivity to \(\varepsilon\)

Deleted Interpolation

Isolated Word Recognition Using HMM’s

Assume a vocabulary of \(V\) words, with \(K\) occurrences of each spoken word in a training set. Observation vectors are spectral characterizations of the word. For isolated word recognition, we do the following:
1. for each word, \(v\), in the vocabulary, we must build an HMM, \(\lambda_v\), i.e., we must re-estimate model parameters \((A,B,T)\) that optimize the likelihood of the training set observation vectors for the \(v\)-th word. (TRAINING)
2. for each unknown word which is to be recognized, we do the following:
   a. measure the observation sequence \(O = [O_1, O_2, \ldots, O_T]\)
   b. calculate model likelihoods, \(P(O|\lambda_v), 1 \leq v \leq V\)
   c. select the word whose model likelihood score is highest
   \[
   v^* = \arg \max_{v=1}^{V} P(O|\lambda_v)
   \]
   Computation is on the order of \(V^2N^T\) required; \(V = 100, N = 5, T = 40\)
   \(\approx 10^9\) computations

Isolated Word HMM Recognizer

Choice of Model Parameters

1. Left-right model preferable to ergodic model (speech is a left-right process)
2. Number of states in range 2-40 (from sounds to frames)
   - Order of number of distinct sounds in the word
   - Order of average number of observations in word
3. Observation vectors
   - Cepstral coefficients (and their second and third order derivatives)
     derived from LPC (1-9 mixtures), diagonal covariance matrices
   - Vector quantized discrete symbols (16-256 codebook sizes)
4. Constraints on \(b_j(O)\) densities
   - \(b_j(k) \geq \varepsilon\) for discrete densities
   - \(G_j \geq 5, U_{b_j}(r,r) \geq 5\) for continuous densities
**Performance Vs Number of States in Model**

- Graph showing error rate in percent vs number of states in HMM.

**HMM Feature Vector Densities**

- Graphs illustrating HMM feature vector densities.

---

**Segmental K-Means Segmentation into States**

**Motivation:**
- Derive good estimates of the \( b_j(\cdot) \) densities as required for rapid convergence of re-estimation procedure.

**Initially:**
- Training set of multiple sequences of observations, initial model estimate.

**Procedure:**
- Segment each observation sequence into states using a Viterbi procedure.
- For discrete observation densities, code all observations in state \( j \) using the \( M \)-codeword codebook, giving

\[
 b_j(k) = \frac{\text{number of vectors with codebook index } k \text{ in state } j}{\text{number of vectors in state } j},
\]

- For continuous observation densities, cluster the observations in state \( j \) into a set of \( M \) clusters, giving

\[
 \mu_{jm} = \text{sample mean of the vectors assigned to cluster } m \text{ of state } j,
\]

\[
 U_{jm} = \text{sample covariance of the vectors assigned to cluster } m \text{ of state } j.
\]

\[
 a_{ij} = \frac{\text{number of vectors in state } i \text{ minus the number of observation sequences for the training word divided by the number of vectors in state } i}{\text{for continuous observation densities, cluster the observations in state } j \text{ into a set of } M \text{ clusters, giving}}
\]

\[
 a_{ij} = \frac{\text{number of vectors in state } i \text{ minus the number of observation sequences for the training word divided by the number of vectors in state } i}{\text{the segmenting HMM is updated and the procedure is iterated until a converged model is obtained.}}
\]

---

**Segmental K-Means Training**

- Diagram illustrating the training process for segmental K-Means.

**HMM Segmentation for /SIX/**

- Diagram illustrating HMM segmentation for the word /SIX/.
HMM PERFORMANCE ON SPEAKER INDEPENDENT, ISOLATED DIGITS

<table>
<thead>
<tr>
<th>Recognizer</th>
<th>Original Error</th>
<th>Test 1 Error</th>
<th>Test 2 Error</th>
<th>Test 3 Error</th>
<th>Test 4 Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPCTDVTW</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
<td>1.1</td>
<td>1.1</td>
</tr>
<tr>
<td>LCPCTDVTW/Q</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>HMM/CD</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>HMM/CD/AR</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
</tbody>
</table>

AVERAGE DIGIT ERROR RATES (%)

LPCTDVTW = Conventional template-based recognizer using dynamic time warping (DTW) alignment.
LCPCTDVTW/Q = Conventional recognizer with vector quantization (VQ = 64 codebooks).
HMM/CD = HMM Recognizer with M = 64 codebooks.
HMM/CD/AR = HMM recognizer using continuous density model with 5 mixtures per state.
HMM/CD/AR = HMM recognizer using mixture autoregressive observation density.