Hidden Markov Models

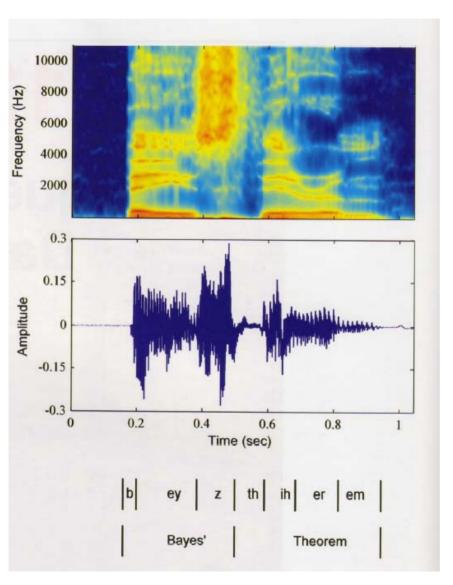
and

Sequential Data

Sequential Data

- Often arise through measurement of time series
 - Snowfall measurements on successive days in Buffalo
 - Rainfall measurements in Chirrapunji
 - Daily values of currency exchange rate
 - Acoustic features at successive time frames in speech recognition
- Non-time series
 - Nucleotide base pairs in a strand of DNA
 - Sequence of characters in an English sentence
 - Parts of speech of successive words

Sound Spectrogram of Spoken Words



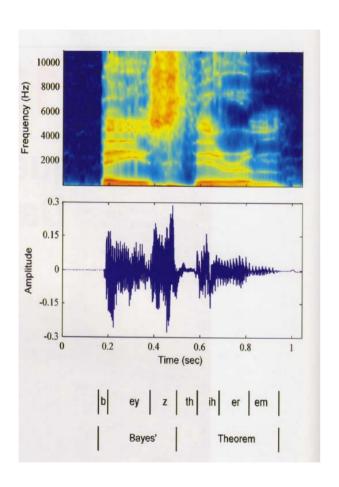
"Bayes Theorem"

- Plot of the intensity of the spectral coefficients versus time index
- Successive observations of the speech spectrum are highly correlated

Task of making a sequence of decisions

- Processes in time, states at time t are influenced by a state at time t-1
- In many time series applications, eg financial forecasting, wish to predict next value from previous values
- Impractical to consider general dependence of future dependence on all previous observations
 - Complexity would grow without limit as number of observations increases
- Markov models assume dependence on most recent observations

Model Assuming Independence



- Simplest model:
 - Treat as independent
 - Graph without links









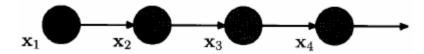
Markov Model

- Most general Markov model for observations {x_n}
- Product rule to express joint distribution of sequence of observations

$$p(x_1,..x_N) = \prod_{n=1}^{N} p(x_N \mid x_1,..x_{N-1})$$

First Order Markov Model

Chain of observations {x_n}



- Distribution $p\{x_n|x_{n-1}\}$ is conditioned on previous observation
- Joint distribution for a sequence of n variables

$$p(x_1,...x_N) = p(x_1) \prod_{n=1}^{N} p(x_n | x_{n-1})$$

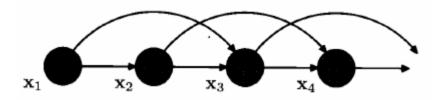
It can be verified (using product rule from above) that

$$p(x_n \mid x_1..x_{n-1}) = p(x_n \mid x_{n-1})$$

 If model is used to predict next observation, distribution of prediction will only depend on preceding observation and independent of earlier observations

Second Order Markov Model

 Conditional distribution of observation x_n depends on the values of two previous observations x_{n-1} and x_{n-2}



$$p(x_1,...x_N) = p(x_1)p(x_2 \mid x_1) \prod_{n=1}^N p(x_n \mid x_{n-1}, x_{n-2})$$

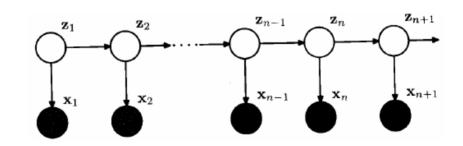
Each observation is influenced by previous two observations

Introducing Latent Variables

- For each observation x_n, introduce a latent variable z_n
- z_n may be of different type or dimensionality to the observed variable
- Latent variables form the Markov chain
- Gives the "state-space model"

Latent variables

Observations



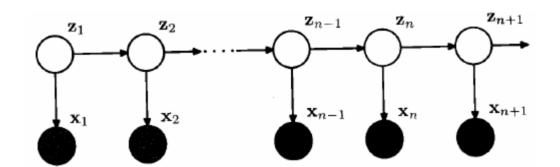
Joint distribution for this model

$$p(x_1,...x_N,z_1,...z_n) = p(z_1) \prod_{n=1}^{N} p(z_n \mid z_{n-1}) \prod_{n=1}^{N} p(x_n \mid z_n)$$

Two Models Described by this Graph

Latent variables

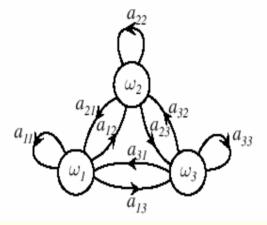
Observations



1. If latent variables are discrete: Hidden Markov Model
Observed variables in a HMM may be discrete or continuous

2. If both latent and observed variables are Gaussian then we obtain *linear dynamical systems*

Latent variable with three discrete states



 Transition probabilities a_{ij} are represented by a matrix

Not a graphical model since the nodes are not separate variables but states of a single variable

This can be unfolded over time to get trellis diagram

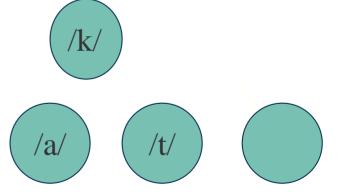
Markov model for the production of spoken words

States represent phonemes

Production of the word: "cat"

- Represented by states
 /k/ /a/ /t/
- Transitions from
 - /k/ to /a/
 - /a/ to /t/
 - /t/ to a silent state
- Although only the correct cat sound Is represented by model, perhaps other transitions can be introduced, eg, /k/ followed by /t/

Markov Model for word "cat"



First-order Markov Models (MMs)

- State at time t: ω(t)
- Particular sequence of length T:

$$\omega^{\mathsf{T}} = \{\omega(1), \, \omega(2), \, \omega(3), \, ..., \, \omega(\mathsf{T})\}\$$
e.g., $\omega^6 = \{\omega 1, \, \omega 4, \, \omega 2, \, \omega 2, \, \omega 1, \, \omega 4\}$

Note: system can revisit a state at different steps and not every state needs to be visited

Particular model a_{22} $\theta = \{a_{ij}\}$ a_{2II} a_{2II} a_{32} a_{33} a_{34} a_{35} a_{35}

Discrete states = nodes, Transition probs = links In first-order discrete time HMM at step t system is in state $\omega(t)$ State at step t +1 is a random function that depends on state at step t and transition probabilities Model for the production of any sequence is described by the *transition probabilities*

$$P(\omega_{i}(t+1) / \omega_{i}(t)) = a_{ij}$$

Which is a time-independent probability of having state ω_j at step (t+1) given at time t state was ω_i

No requirement transitional probabilities be symmetric

Given model θ probability that model generated sequence

$$\omega_6 = \{\omega_1, \omega_4, \omega_2, \omega_2, \omega_1, \omega_4\}$$
is

 $P(\omega_6 \mid \theta) = a_{14} \cdot a_{42} \cdot a_{22} \cdot a_{21} \cdot a_{14}$ Can include a priori probability of first state as

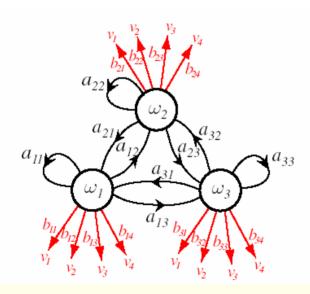
$$P(\omega(1) = \omega_i)$$

First Order Hidden Markov Models

- Perceiver does not have access to the states ω(t)
- Instead we measure properties of the emitted sound
- Need to augment Markov model to allow for visible states (symbols)

First Order Hidden Markov Models

- Visible States (symbols) $V^T = \{v(1), v(2), v(3), ..., v(T)\}$
 - For instance $V^6 = \{v_5, v_1, v_1, v_5, v_2, v_3\}$
- In any state $\omega_j(t)$ probability of emitting symbol $v_k(t)$ is b_{jk}



Three hidden units in HMM Visible states and emission probabilities of visible states in red

Hidden Markov Model Computation

- Finite State Machines with transitional probabilities— called Markov Networks
- Strictly causal: probabilities depend only on previous states
- A Markov model is ergodic if every state has non-zero probability of occuring given some starting state
- A final or absorbing state is one which if entered is never left

Hidden Markov Model Computation

$$a_{ij} = P(\omega_j(t+1)|\omega_i(t))$$

$$b_{jk} = P(v_k(t)|\omega_j(t)).$$

$$\sum_{j} a_{ij} = 1 \quad \text{for all } i$$

$$\sum_{k} b_{jk} = 1 \quad \text{for all } j,$$

Three Basic Problems for HMMs

- Given HMM with transition and symbol probabilities
- Problem 1: The evaluation problem
 - ullet Determine the probability that a particular sequence of symbols V^T was generated by that model
- Problem 2: The decoding problem
 - Given a set of symbols V^T determine the most likely sequence of hidden states ω^T that led to the observations
- Problem 3: The learning problem
 - Given a coarse structure of the model (no of states and no of symbols) but not the probabilities a_{ij} and b_{jk}
 - determine these parameters

Problem 1: Evaluation Problem

• Probability that model produces a sequence V^T of visible states:

$$P(V^{T}) = \sum_{r=1}^{r_{max}} P(V^{T}/\omega_{r}^{T})P(\omega_{r}^{T})$$
Visible sequence

Hidden states

where each r indexes a particular sequence

$$\omega_r^T = \{\omega(1), \omega(2), ..., \omega(T)\}\$$
 of T hidden states

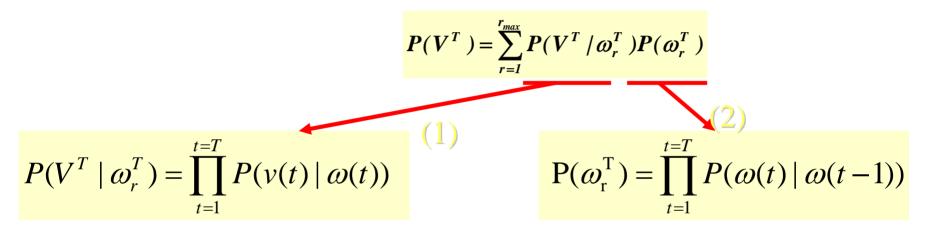
In the general case of c hidden states there will be

$$\mathbf{r}_{\text{max}} = \mathbf{c}^{\mathbf{T}}$$

possible terms

Evaluation Problem Formula

Probability that model produces a sequence V^T of visible states:



Because (1) output probabilities depend only upon hidden states and (2) first order hidden Markov process. Substituting,

$$P(V^T) = \sum_{t=1}^{r_{max}} \prod_{t=1}^{t=T} P(v(t)/\omega(t)) P(\omega(t)/\omega(t-1))$$

Interpretation: Probability of sequence V^T is equal to the sum over all rmax possible sequences of hidden states of the conditional probability that the system has made a particular transition multiplied by the probability that it then emitted the

visible symbol in the target sequence

Computationally Simpler Evaluation Algorithm

- Calculate $P(V^T) = \sum_{t=1}^{r_{max}} \prod_{t=1}^{t=T} P(v(t)/\omega(t)) P(\omega(t)/\omega(t-1))$
- recursively because each term involves only v(t), $\omega(t)$ and $\omega(t-1)$

Define:

tial state tial state
$$b_{jk} = P(\omega_j(t+1)|\omega_i(t))$$

$$b_{jk} = P(v_k(t)|\omega_j(t)).$$

 $\mathbf{P}(\mathbf{v}(\mathbf{t}) | \omega(\mathbf{t})) \mathbf{P}(\omega(\mathbf{t}) | \omega(\mathbf{t}-1))$

$$\alpha_j(t) = \begin{cases} 0 & t = 0 \text{ and } j \neq \text{initial state} \\ 1 & t = 0 \text{ and } j = \text{initial state} \\ \left[\sum_i \alpha_i(t-1)a_{ij}\right]b_{jk}v(t) & \text{otherwise,} \end{cases}$$

where: $b_{ik}v(t)$ means the symbol probability b_{ik} corresponding to v(t)

Therefore

is the probability that the model is in state $\omega_j(t)$ and has generated the target sequence upto step t

HMM Forward

$$\alpha_j(t) = \left\{ \begin{array}{ll} 0 & t = 0 \text{ and } j \neq \text{initial state} \\ 1 & t = 0 \text{ and } j = \text{initial state} \\ \left[\sum_i \alpha_i(t-1)a_{ij} \right] b_{jk} v(t) & \text{otherwise,} \end{array} \right.$$

```
Algorithm 2. (HMM Forward)

1 initialize t \leftarrow 0, a_{ij}, b_{jk}, visible sequence \mathbf{V}^T, \alpha_j(0)

2 for t \leftarrow t + 1

3 \alpha_j(t) \leftarrow b_{jk}v(t) \sum_{i=1}^c \alpha_i(t-1)a_{ij}

4 until t = T

5 return P(\mathbf{V}^T) \leftarrow \alpha_0(T) for the final state

6 end
```

where: $b_{jk}v(t)$ means the symbol probability b_{jk} corresponding to v(t)

Computational complexity of $O(c^2T)$

Time-reversed Version of Forward Algorithm

```
Algorithm 3. (HMM Backward)

1 initialize \beta_j(T), t \leftarrow T, a_{ij}, b_{jk}, visible sequence \mathbf{V}^T

2 for t \leftarrow t - 1;

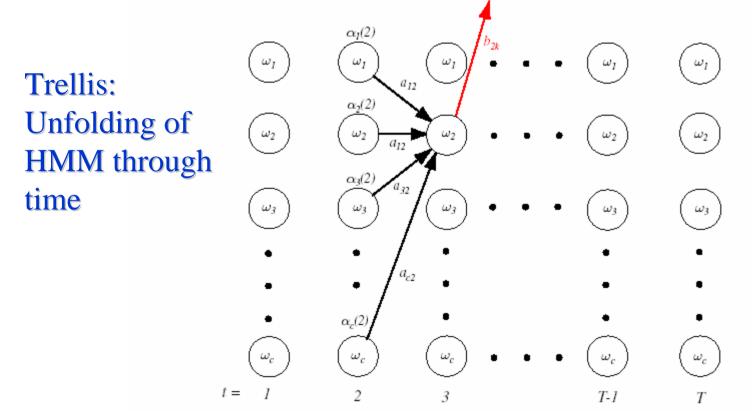
3 \beta_i(t) \leftarrow \sum_{j=1}^c \beta_j(t+1)a_{ij}b_{jk}v(t+1)

4 until t = 1

5 return P(\mathbf{V}^T) \leftarrow \beta_i(0) for the known initial state

6 end
```

Computation of Probabilities by Forward Algorithm

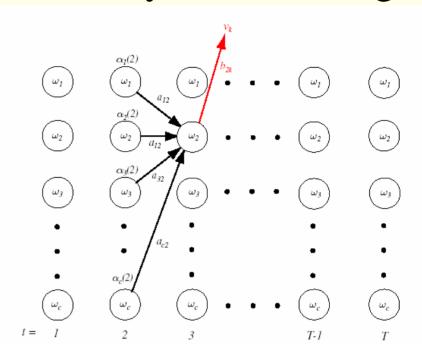


we seek the probability that the HMM was in state ω_2 at t=3 and generated the observed visible symbol up through that step (including the observed visible symbol v_k). The probability the HMM was in state $\omega_j(t=2)$ and generated the observed sequence through t=2 is $\alpha_j(2)$ for $j=1,2,\ldots,c$. To find $\alpha_2(3)$ we must sum these and multiply the probability that state ω_2 emitted the observed symbol v_k . Formally, for this particular illustration we have $\alpha_2(3)=b_{2k}\sum_{j=1}^c\alpha_j(2)a_{j2}$.

Computation of Probabilities by Forward Algorithm

In the *evaluation* trellis we only accumulate values

In the *decoding* trellis we only keep maximum values

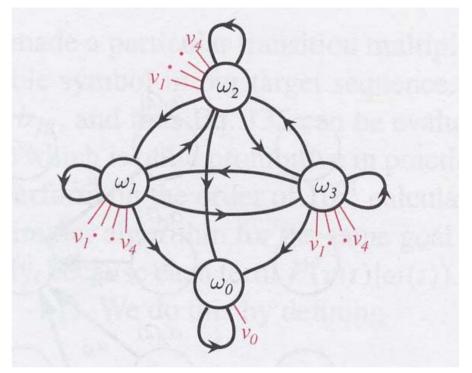


Example of Hidden Markov Model

Four states with explicit absorber state

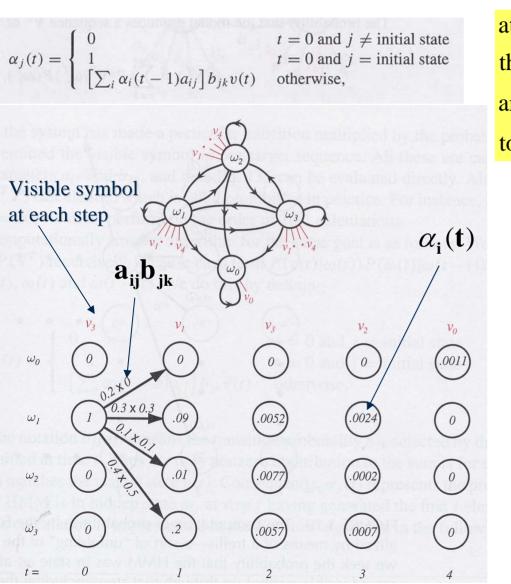
		$\omega_{\scriptscriptstyle 0}$	$\omega_{\scriptscriptstyle 1}$	$\omega_{\scriptscriptstyle 2}$	ω_3
	ω_{0}	1	0	0	0
a _{ij} =	$\omega_{\scriptscriptstyle 1}$	0.2	0.3	0.1	0.4
	ω_{2}	0.2	0.5	0.2	0.1
	ω_3	8.0	0.1	0	0.1

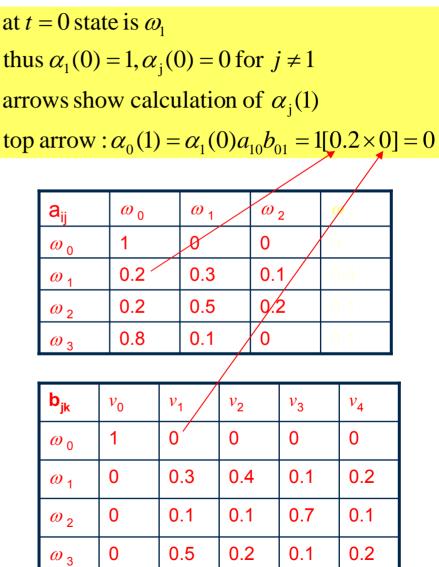
		v_0	v_1	v_2	v_3	v_4
	$\omega_{\scriptscriptstyle 0}$	1	0	0	0	0
b _{jk} =	$\omega_{\scriptscriptstyle 1}$	0	0.3	0.4	0.1	0.2
	ω_{2}	0	0.1	0.1	0.7	0.1
	$\omega_{\scriptscriptstyle 3}$	0	0.5	0.2	0.1	0.2



Five symbols with unique null symbol

Example Problem: compute probability of generating sequence $V^4 = \{v_1, v_3, v_2, v_0\}$ Assume ω_1 is the start state





Problem 2: Decoding Problem

- Given a sequence of visible states V^T, the decoding problem is to find the most probable sequence of hidden states.
- Expressed mathematically as:
 - find the single "best" state sequence (hidden states)

```
\hat{\omega}(1), \hat{\omega}(2), ..., \hat{\omega}(\mathbf{T}) \text{ such that }:
\hat{\omega}(1), \hat{\omega}(2), ..., \hat{\omega}(\mathbf{T}) = \underset{\omega(1), \omega(2), ..., \omega(\mathbf{T})}{\arg \max} \mathbf{P} \left[ \omega(1), \omega(2), ..., \omega(\mathbf{T}), \mathbf{v}(1), \mathbf{v}(2), ..., \mathbf{v}(\mathbf{T}) \mid \theta \right]
```

Note that summation is changed to argmax, since we want to find the best case

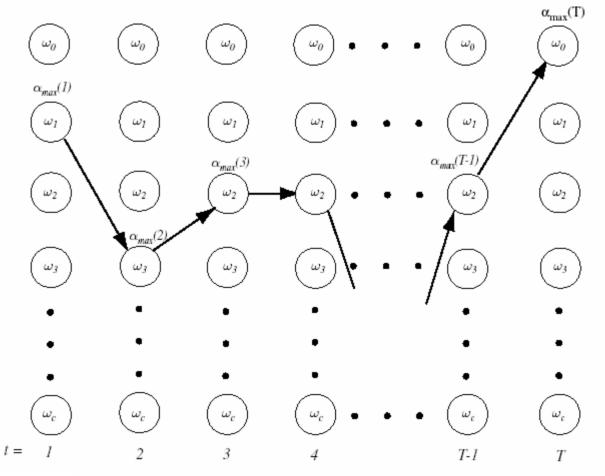
Viterbi Algorithm

```
Algorithm 4. (HMM decoding)
    begin initialize Path \leftarrow \{\}, t \leftarrow 0
               \underline{\mathbf{for}} \ t \leftarrow t+1
    j \leftarrow j + 1
    \underline{\mathbf{for}} \ j \leftarrow j+1
     \alpha_j(t) \leftarrow b_{jk} v(t) \sum_{i=1}^c \alpha_i(t-1) a_{ij}
      \mathbf{\underline{until}} \ j = c
                 j' \leftarrow \arg\max_{i} \alpha_{j}(t)
                   Append \omega_{i'} to Path
              until t = T
10
       return Path
11 end
```

Notes:

- 1. If a_{ij} and b_{jk} are replaced by log probabilities a we add terms rather than multiply them
- 2. Best path is maintained for each node
- $\mathbf{O}(\mathbf{c}^{\mathrm{T}}\mathbf{T})$

Viterbi Decoding Trellis



The decoding algorithm finds at each time step t the state that has the highest probability of having come from the previous step and generated the observed visible state v_k . The full path is the sequence of such states. Because this is a local optimization (dependent only upon the single previous time step, not the full sequence), the algorithm does not guarantee that the path is indeed allowable. For instance, it might be possible that the maximum at t=5 is ω_1 and at t=6 is ω_2 , and thus these would appear in the path. This can even occur if $a_{12}=P(\omega_2(t+1)|\omega_1(t))=0$, precluding that transition.

Example of Decoding

Four states with explicit absorber state

$$a_{ii} =$$

	ω ,	$\omega_{\scriptscriptstyle I}$	w 2	w 3
ω ,	1	0	0	0
$\mathbf{\omega}_{I}$	0.2	0.3	0.1	0.4
w 2	0.2	0.5	0.2	0.1
w 3	0.8	0.1	0	0.1

Five symbols with unique null symbol

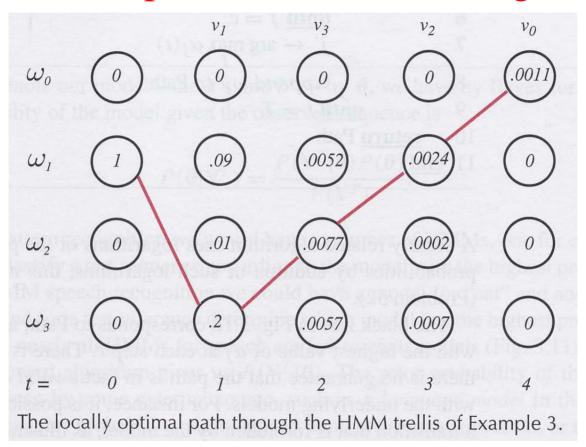
	v_{θ}	v_{I}	v_2	v_3	v_4
ω_{o}	1	0	0	0	0
ω_{1}	0	0.3	0.4	0.1	0.2
ω_{2}	0	0.1	0.1	0.7	0.1
ω ₃	0	0.5	0.2	0.1	0.2

What is the most likely state sequence that generated the particular symbol sequence

$$V^4 = \{v_1, v_3, v_2, v_0\}$$
?

Assume ω_1 is the start state

Example 4. HMM Decoding



Note: transition between ω_3 and ω_2 is forbidden by model Yet decoding algorithm gives it a non-zero probability

Problem 3: Learning Problem

- Goal: To determine model parameters a_{ij} and b_{jk} from an ensemble of training samples
- No known method for obtaining an optimal or most likely set of parameters
- A good solution is straightforward:
 Forward-Backward Algorithm

Forward-Backward Algorithm

- Instance of generalized expectation maximization algorithm
- We do not know the states that hold when the symbols are generated
- Approach is to iteratively update the weights in order to better explain the observed training sequences

Backward Probabilities

- $\alpha_i(t)$ is the probability that model is in state $\alpha_i(t)$ and has generated the target sequence upto step t
 - Analogously $\beta_i(t)$ is the probability that that the model is in state $\omega_i(t)$ and will generate the remainder of the given target sequence from t+1 to T

$$\beta_i(t) = \begin{cases} 0 & \omega_i(t) \neq \omega_0 \text{ and } t = T \\ 1 & \omega_i(t) = \omega_0 \text{ and } t = T \\ \sum_j \beta_j(t+1)a_{ij}b_{jk}v(t+1) & \text{otherwise.} \end{cases}$$

Computation proceeds backward through the trellis

Backward evaluation algorithm

```
Algorithm 3. (HMM Backward)

1 initialize \beta_j(T), t \leftarrow T, a_{ij}, b_{jk}, visible sequence \mathbf{V}^T

2 for t \leftarrow t - 1;

3 \beta_i(t) \leftarrow \sum_{j=1}^c \beta_j(t+1)a_{ij}b_{jk}v(t+1)

4 until t = 1

5 return P(\mathbf{V}^T) \leftarrow \beta_i(0) for the known initial state

6 end
```

This is used in learning: parameter estimation

Estimating the a_{ij} and b_{jk}

- $\alpha_{\bf i}({\bf t})$ $\beta_{\bf i}({\bf t})$ are merely estimates of their true values since we don't know the actual values of a_{ij} and b_{jk}
- The probability of transition between $\omega_i(t-1)$ and $\omega_j(t)$ given the model generated the entire training sequence V^T by any path is:

$$\gamma_{ij}(t) = \frac{\alpha_i(t-1)a_{ij}b_{jk}\beta_j(t)}{P(\mathbf{V}^T|\boldsymbol{\theta})},$$

Calculating Improved Estimate of a_{ij}

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T} \gamma_{ij}(t)}{\sum_{t=1}^{T} \sum_{k} \gamma_{ik}(t)}.$$

- Numerator is the expected number of transitions between state $\omega_i(t-1)$ and $\omega_j(t)$
- Denominator is the total expected number of transitions from ω_i

Calculating Improved Estimate of b_{jk}

• Ratio between frequency that any particular symbol v_k is emitted and that for any symbol

$$\hat{b}_{jk} = \frac{\sum_{t=1}^{T} \sum_{l} \gamma_{jl}(t)}{\sum_{t=1}^{T} \sum_{l} \gamma_{jl}(t)}.$$

Learning Algorithm

- Start with rough estimates of a_{ij} and b_{jk}
- Calculate improved estimate using the formulas above
- Repeat until sufficiently small change in the estimated values of the parameters

Baum-Welch or Forward-Backward Algorithm

```
Algorithm 5. (Forward-Backward)

1 begin initialize a_{ij}, b_{jk}, training sequence \mathbf{V}^T, convergence criterion \theta, z \leftarrow 0

2 do z \leftarrow z + 1

3 compute \hat{a}(z) from a(z - 1) and b(z - 1) by Eq. 140

4 compute \hat{b}(z) from a(z - 1) and b(z - 1) by Eq. 141

5 a_{ij}(z) \leftarrow \hat{a}_{ij}(z - 1)

6 b_{jk}(z) \leftarrow \hat{b}_{jk}(z - 1)

7 until \max_{i,j,k} [a_{ij}(z) - a_{ij}(z - 1), b_{jk}(z) - b_{jk}(z - 1)] < \theta; (convergence achieved)

8 return a_{ij} \leftarrow a_{ij}(z); b_{jk} \leftarrow b_{jk}(z)

9 end

\hat{a}_{ij} = \frac{\sum_{t=1}^{T} \gamma_{ij}(t)}{\sum_{t=1}^{T} \sum_{t} \gamma_{jl}(t)}

\hat{b}_{jk} = \frac{\sum_{t=1}^{T} \sum_{t} \gamma_{jl}(t)}{\sum_{t=1}^{T} \sum_{t} \gamma_{jl}(t)}
```

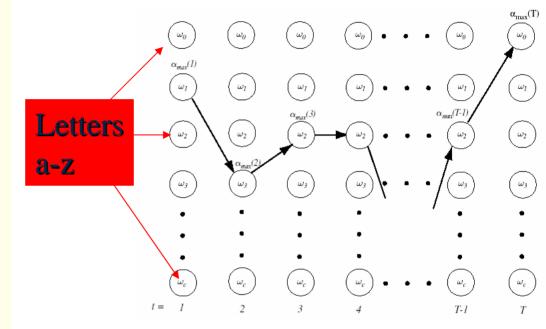
Convergence

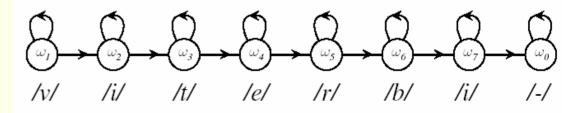
- Requires several presentations of each training sequence (fewer than 5 common in speech)
- Another stopping criterion:
 - Overall probability that learning model could have generated the training data

HMM Word Recognition

• Two approaches:

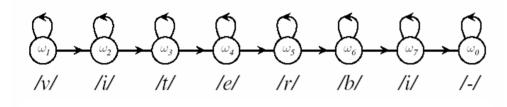
- HMM can model all possible words
 - Each state corresponds to each letter of alphabet
 - Letter transition probabilities are calculated for each pair of letters
 - Letter confusion probabilities are symbol probabilities
 - Decoding problem gives most likely word
- Separate HMMs are used to model each word
 - Evaluation problem gives probability of observation which is used as a classconditional probability





HMM spoken word recognition

- Each word, e.g., cat, dog, etc, has an associated HMM
- For a test utterance determine which model has highest probability
- HMMs for speech are left-to-right models



such a model could describe the utterance "viterbi," where ω_1 represents the phoneme /v/, ω_2 represents /i/,..., and ω_0 a final silent state. Such a left-to-right model is more restrictive than the general HMM \longrightarrow because it precludes transitions "back" in time.

HMM produces a class-conditional probability

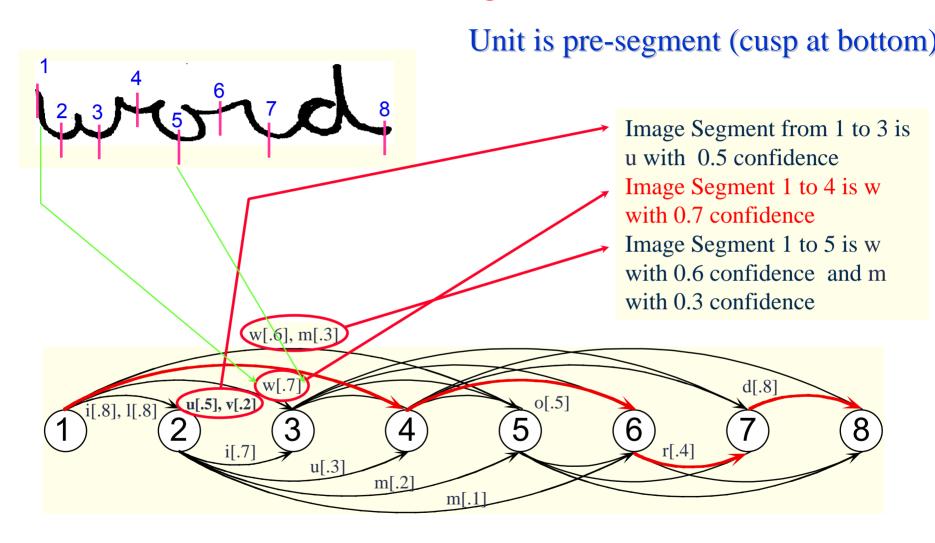
Thus it is useful to compute probability of the model given the sequence using Bayes rule

$$P(\boldsymbol{\theta}|\mathbf{V}^T) = \frac{P(\mathbf{V}^T|\boldsymbol{\theta})P(\boldsymbol{\theta})}{P(\mathbf{V}^T)}.$$

Computed by Forward algorithm

Prior probability of sequence

Cursive Word Recognition (not HMM)



Best path in graph from segment 1 to 8: w o r d

Summary: Key Algorithms for HMM

- Problem 1: HMM Forward
- Problem 2: Viterbi Algorithm
 - An algorithm to compute the optimal (most likely) state sequence in a HMM given a sequence of observed outputs.
- Problem 3: Baum-Welch Algorithm
 - An algorithm to find HMM parameters A, B, and Π with the maximum likelihood of generating the given symbol sequence in the observation vector