

Hidden Markov Models

Material refers to Duda 3.10;

A classical reference is A Tutorial on Hidden Markov Models,
Lawrence R. Rabiner in Readings in speech recognition (1990)-
available on the web; although it uses different notation

Hidden Markov Models

So far: considered classification systems for making a single decision (e.g. discriminant functions or estimation of class-conditional densities.)

Now we consider: the problem of using dynamic probability models

Example: Automatic Speech Recognition (ASR).

In ASR, we need to determine a sequence of phonemes (like vowels and consonants) that make up the observed speech sound.

For this we will introduce **Hidden Markov Models** (HMMs):

Statistical signal models

- Hidden and observable Markov models are statistical signal models. Other examples: Gaussian processes.
- General assumption:
 - Signal can be well characterized as a parametric random signal processing, e.g., noise removal process, and the parameters of the stochastic process can be signal recognition, e.g., identification determined in a precise, well-defined manner



First-order Markov Models Revision

Markov Models describe the evolution of a finite state machine.

State at time t is denoted $\omega(t)$. A sequence of length T is denoted :

$$\omega^T = \{\omega(1), \omega(2), \dots, \omega(T)\}$$

e.g. $\omega^6 = \{\omega_1, \omega_4, \omega_2, \omega_2, \omega_1, \omega_4\}$

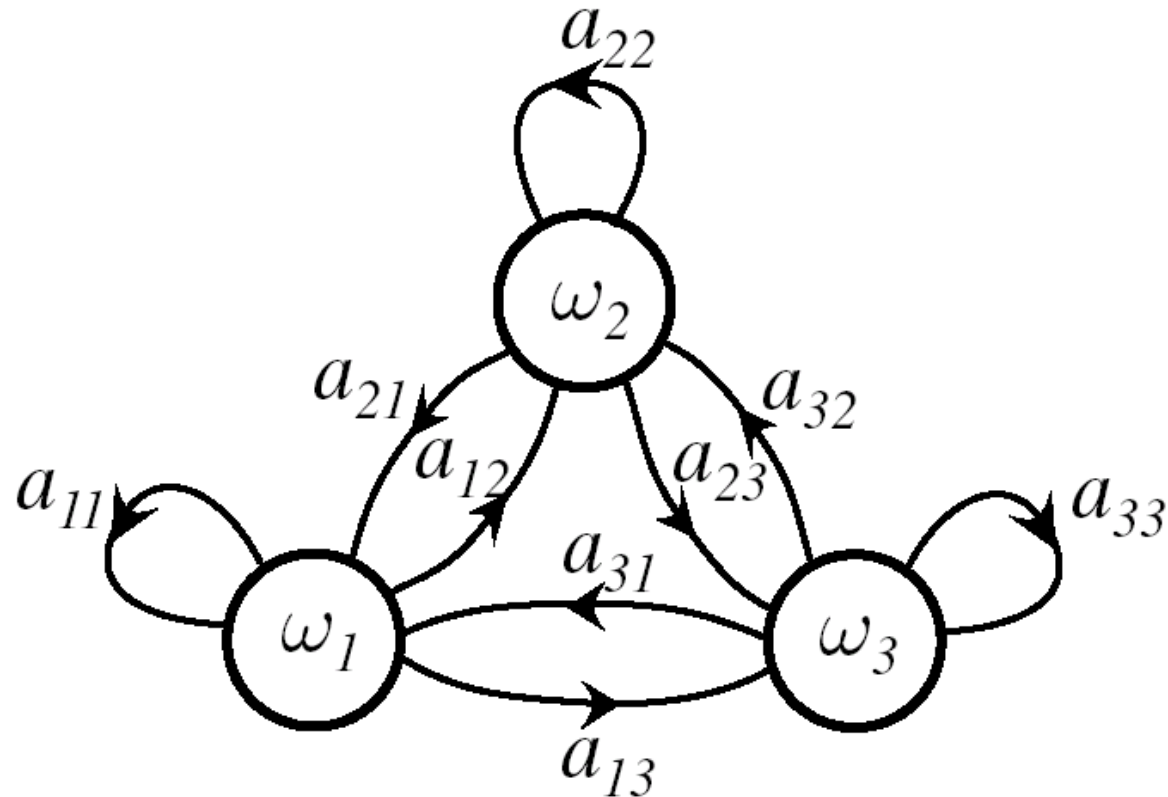
Markov Model characterised by : *transition probabilities*

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

Prob. of moving to state ω_j at time $(t+1)$
given state ω_i at time t .

NOTE: First-order
depends only on
previous state

Markov Model State Transition Graph



In general, $a_{ij} \neq a_{ji}$ (not necessarily be symmetric), a_{ij} can equal 0 and $a_{ii} \neq 0$ (non - zero prob. of staying in the same state)

Calculating the model probability

Given a_{ij} what is the probability the model will generate

a particular sequence ω^T ? Note in a first order Markov model we have

$$P(\omega_j(t) | \omega_i(t-1), \omega_k(t-2), \dots, \omega_l(1)) = P(\omega_j(t) | \omega_i(t-1)) = a_{ij}$$

(i.e. it's only dependent on the previous state). Note also that time is not involved any more. This is known as the stationary process assumption.

For a 2 sequence, assuming known initial state $\omega(0) = \omega_1$.

$$\begin{aligned} P(\{\omega_i, \omega_j\} | \omega_1(0)) &= P(\omega_j(2) | \omega_i(1)) P(\omega_i(1) | \omega_1(0)) \\ &= a_{ij} a_{1i} \end{aligned}$$

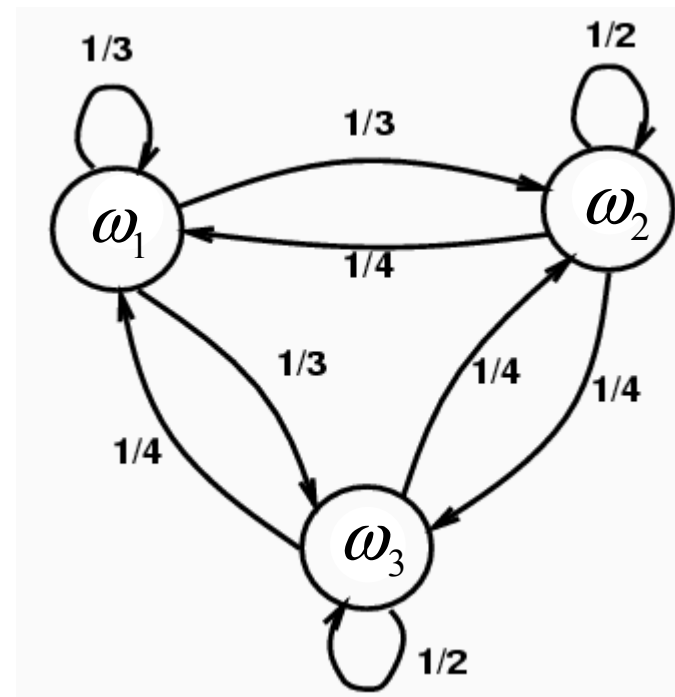
Basic Markov Model: Example

Weather example:

ω_1 = sunny, ω_2 = cloudy, ω_3 = rainy.

Suppose we have transition matrix

$$A = [a_{ij}] = \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/4 & 1/2 & 1/4 \\ 1/4 & 1/4 & 1/2 \end{bmatrix}$$



e.g. $P(\text{"sunny tomorrow given it is cloudy today"}) = a_{21} = 1/4$.

Pr. of sequence " $\omega_1\omega_2\omega_3$ " (given we start in state ω_1) is:

$$P(\{\omega_2, \omega_3\} | w(0) = \omega_1) = a_{12}a_{23} = (1/3) \times (1/4) = 1/12.$$

Markov: Example 2

Using same model, this time we want

Prob of sequence " $\{\omega_1, X, \omega_3\}$ " where "X" can be anything.

This is:

$$\begin{aligned} p_{aXb} &= P(\{\omega_1, \omega_1, \omega_3\} \text{ or } \{\omega_1, \omega_2, \omega_3\} \text{ or } \{\omega_1, \omega_3, \omega_3\}) \\ &= P(\{\omega_1, \omega_1, \omega_3\}) + P(\{\omega_1, \omega_2, \omega_3\}) + P(\{\omega_1, \omega_3, \omega_3\}) \end{aligned}$$

(since the sequences are mutually exclusive).

Hence this is

$$\begin{aligned} p_{aXb} &= a_{11}a_{13} + a_{12}a_{23} + a_{13}a_{33} \\ &= (1/3)(1/3) + (1/3)(1/4) + (1/3)(1/2) \\ &= 13/36 \approx 0.36 \end{aligned}$$

Hidden Markov Model

Generally (e.g. speech recognition) we do not have access to the states $\omega(t)$. Instead, we observe some visible symbols (features), $v(t)$, that are in some way dependent on the current state $\omega(t)$.

In HMMs this relationship is also modelled by probabilities :

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

is the probability of emitting symbol v_k at time t given the state w_j at time t . Instead of observing a state sequence ω^T we observe a visible symbol sequence given $\mathbf{V}^T = \{v(1), v(2), \dots, v(T)\}$
Hence we have a *hidden* Markov model.

The ice cream introduction to HMM

- let's borrow the setup proposed by Jason Eisner in 2002 [1], “Ice Cream Climatology.”
 - **The situation:**

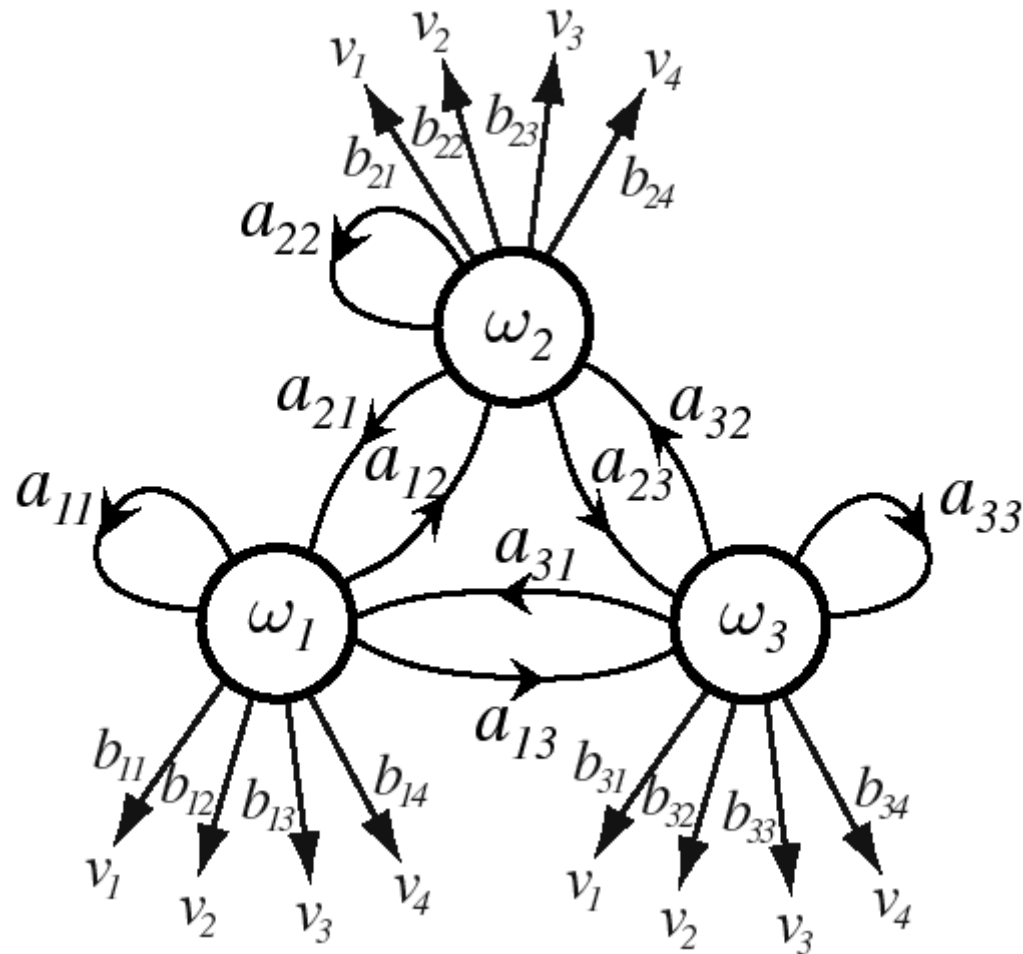
You are a climatologist in the year 2799, studying the history of global warming. You can't find any records of Baltimore weather, but you do find my (Jason Eisner's) diary, in which I carefully recorded how much ice cream I ate each day. What can you figure out from this about the weather that summer?
 - What are the visible states?
 - What are the hidden states (which really interests us)?

Hidden Markov model

States denoted
by ω_i .

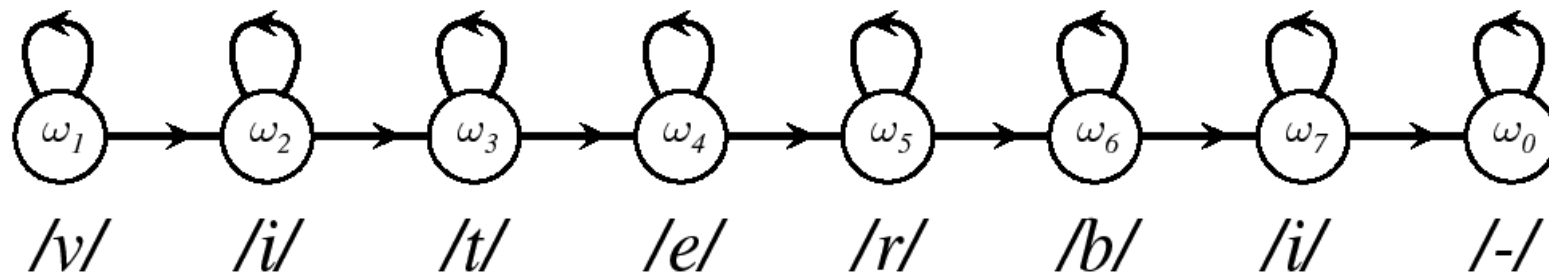
Visible symbols
denoted by v_j .

This model shows all
state transitions as
being possible: not
always the case.



Left-to-Right Models

In speech recognition, most HMMs are *left - to - right* models, i.e. we have $a_{ij} = 0$ if $j < i$ (unless $j = 0$, the absorbing state), so the state sequence always goes "left to right".



In a left - to - right model we can also specify "no state skips".

This means that

$a_{ij} = 0$ unless $j = i$ or $j = i + 1$ or $i = i_{\max}$ and $j = 0$.

Probability Parameters

To summarize, HMMs are defined by the following:

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

$b_{jk} = P(v_k(t) | w_j(t))$ matrix of emission probabilities

Further, we require that

(1) some transition must occur from step t to $t+1$ (even if to the same state)

(2) some visible symbol must be emitted at each step.

Hence we have

$$\sum_j a_{ij} = 1 \text{ for all } i \quad \text{and} \quad \sum_k b_{jk} = 1 \text{ for all } j$$

We will also assume a given starting state e.g. $\omega(0) = \omega_1$

Back to our silly example with ice creams

- Let's simplify and suppose there are only two kinds of days: C (cold) and H (hot).
- You have a record of how many ice creams I eat per day
- C and H are the hidden states
- How many ice creams the observable state.
- Lets make some assumptions on probabilities

	B	C	D	E	F	G
10		p(... C)	p(... H)	p(... START)		
11	p(1 ...)	0.7	0.1		If today is cold (C) or hot (H), how many cones did I prob. eat?	
12	p(2 ...)	0.2	0.2			
13	p(3 ...)	0.1	0.7			
14	p(C ...)	0.8	0.1	0.5	If today is cold or hot, what will tomorrow probably be?	
15	p(H ...)	0.1	0.8	0.5		
16	p(STOP ...)	0.1	0.1	0		

3 central issues

(1) The Evaluation Problem

Given an HMM (i.e. a_{ij} and b_{jk} and $\omega(0)$), what is the probability of generating the sequence \mathbf{V}^T ?

(2) The Decoding Problem

Given an HMM and observation sequence \mathbf{V}^T , determine the most likely hidden state sequence ω^T .

(3) The Learning Problem

Given the structure of a HMM (i.e. no. of states, connections and no. of visible symbols), determine the best model (a_{ij} and b_{jk}) for a set of training observations.

In our silly ice-cream example

(1) The evaluation problem:

Given an HMM what is the probability that I ate 3, 2, 1 ice-creams on 3 days consecutive in this order?

(2) The decoding problem:

Given the ice-creams recorded what was likely the weather in those days, assuming that I know the HMM parameters?

(3) The learning problems:

Given just the ice-creams recorded what are HMM parameters?

Evaluation

Prob that the model produces visible sequence \mathbf{V}^T :

$$P(\mathbf{V}^T) = \sum_{r=1}^{r_{\max}} P(\mathbf{V}^T | \boldsymbol{\omega}_r^T) P(\boldsymbol{\omega}_r^T)$$

where each r indexes a particular sequence of T states

$$\boldsymbol{\omega}_r^T = \{\omega(1), \omega(2), \dots, \omega(T)\}$$

For c hidden states we have $r_{\max} = c^T$ sequences.

Since we have a Markov model, 2nd prob term is

$$P(\boldsymbol{\omega}_r^T) = \prod_{t=1}^T P(\omega(t) | \omega(t-1))$$

(we've assumed $\omega(0)$ is given).

Evaluation (cont)

Since emission probs depend only on hidden state,

$$P(\mathbf{V}^T | \boldsymbol{\omega}_r^T) = \prod_{t=1}^T P(v(t) | \omega(t))$$

giving

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

This can be evaluated since we know

$$P(\omega_j(t) | \omega_i(t-1)) = a_{ij} \quad \text{and} \quad P(v_k(t) | \omega_j(t)) = b_{jk}.$$

A straightforward interpretation:

- Probability that we observe the particular sequence of T visible states \mathbf{V}^T is equal to the sum over all 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 our target sequence. All these are captured in our parameters a_{ij} and b_{jk} , and thus Eq. 91 can be evaluated directly.

Evaluation (cont)

A straightforward interpretation:

- Probability that we observe the particular sequence of T visible states V^T is equal to the sum over all 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 our target sequence.
- All these are captured in our parameters a_{ij} and b_{kj} , and thus this equation can be evaluated directly.
- BUT:

Naively this is an $O(c^T T)$ calculation, e.g. if $c = 10$ and $T = 20$, we need $O(10^{21})$ operations! Can we do better?

Recursive calculation of $P(V^T)$

Let us look again at $P(V^T)$:

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

However we don't have to do the calculation all at once!

Observe that each term $P(v(t) | \omega(t)) P(\omega(t) | \omega(t-1))$
involves only $v(t), \omega(t), \omega(t-1)$

This hints that I can build the product in time and put the sum over the limited time horizon (just the past!) inside!

HMM Forward Algorithm

The Forward Algorithm can be written in terms of $\alpha_j(t) = P(\omega_j, \mathbf{V}^t)$:

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

2. Increment $t \leftarrow t + 1$

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

4. If $t < T$, repeat from 2.

5. Return $P(\mathbf{V}^T) = \sum_{i=1}^c \alpha_i(T)$

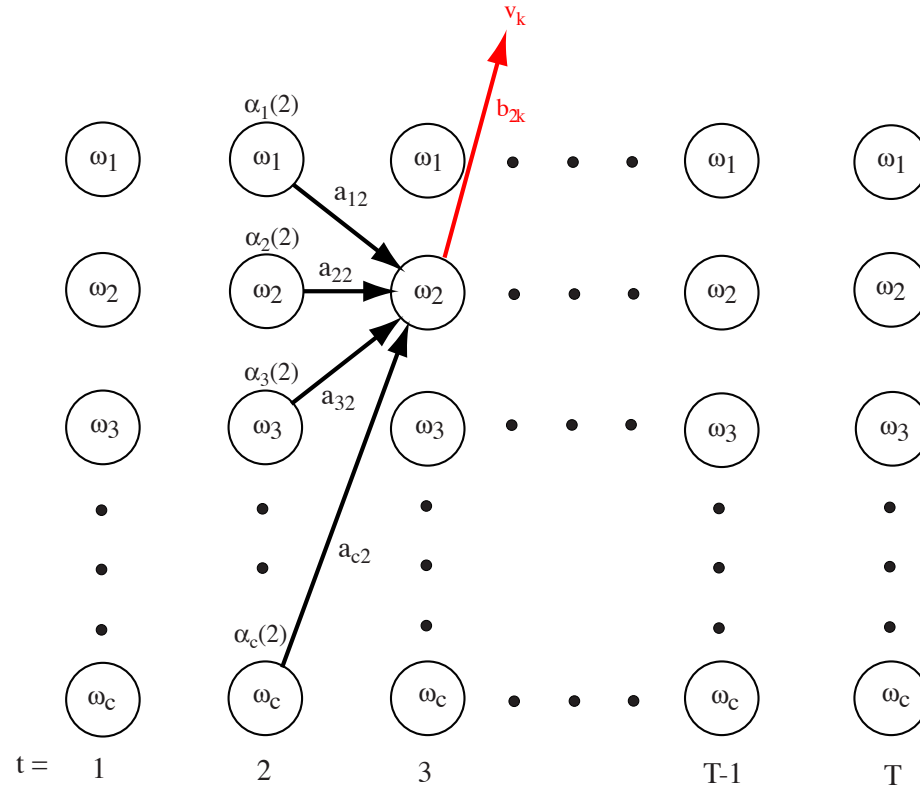
$b_{j,v(t)}$: the transition probability b_{jk} **selected** by the visible state emitted at time t . Thus the only non-zero contribution is for the index k which matches $v(t)$.

$a_{ij}(t)$: probability that HMM is in hidden state ω_j at step t having generated the first t elements in \mathbf{V}^T

(if there is a final absorbing state ω_0 then we have $P(\mathbf{V}^T) \leftarrow \alpha_0(T)$)

The Forward Algorithm has complexity $O(c^2T)$.

e.g. for $c = 10, T = 20$, needs $O(2000)$ calculations.



The computation of probabilities by the Forward algorithm can be visualized by means of a trellis — a sort of “unfolding” of the HMM through time. Suppose we seek the probability that the HMM was in state ω_2 at $t = 3$ and generated the observed visible 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 $j = 1, 2, \dots, c$. 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 $\alpha_2(3) = b_{2k} \sum_{j=1}^c \alpha_j(2) a_{j2}$.

Evaluation Example

Consider HMM with absorber state, null symbol v_0 , and

Absorbing state

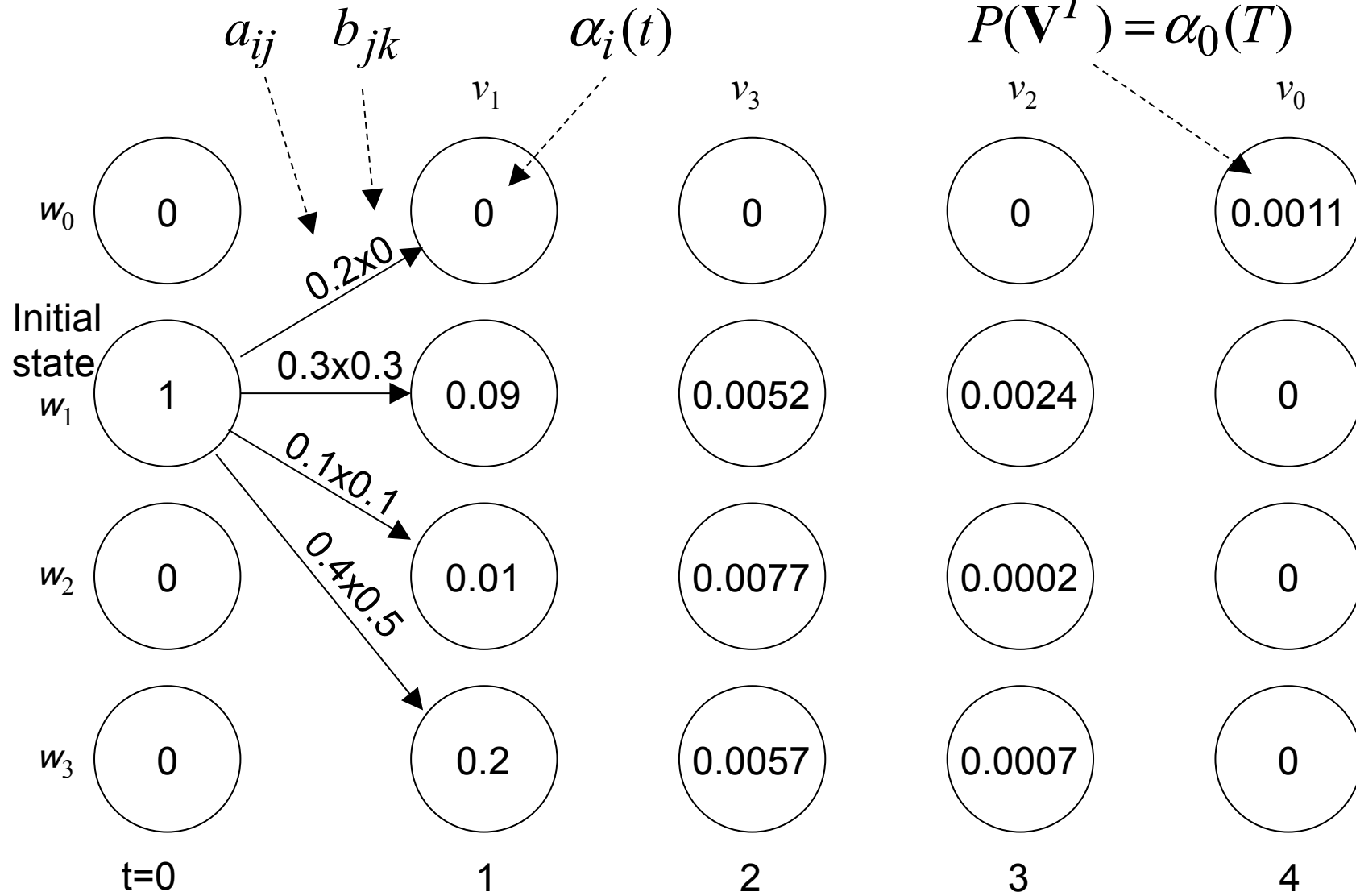
$$a_{ij} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.2 & 0.3 & 0.1 & 0.4 \\ 0.2 & 0.5 & 0.2 & 0.1 \\ 0.8 & 0.1 & 0.0 & 0.1 \end{pmatrix} \quad b_{jk} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0.3 & 0.4 & 0.1 & 0.2 \\ 0 & 0.1 & 0.1 & 0.7 & 0.1 \\ 0 & 0.5 & 0.2 & 0.1 & 0.2 \end{pmatrix}$$

(matrix indexes start at 0, so top row is for w_0)

What is the probability this generates the sequence

$\mathbf{V}^4 = \{v_1, v_3, v_2, v_0\}$, given initial state at $t = 0$ is ω_1 ?

Evaluation Example (cont)

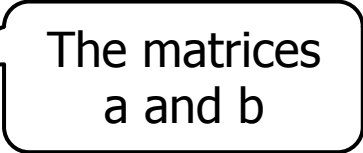


- The visible symbol at each step is shown above, and $\alpha_i(t)$ in each unit. Circles show the value for $\alpha_i(t)$ as we progress left to right. Product $a_{ij}b_{jk}$ is shown along each transition link for the step $t = 1$ to $t = 2$.
- In each node we show $\alpha_i(t)$ — the probability the model generated the observed visible sequence up to t .
- For instance, we know that the system was in hidden state ω_1 at $t = 1$, and thus $\alpha_1(0) = 1$ and $\alpha_i(0) = 0$ for $i \neq 1$. The arrows show the calculation of $\alpha_i(1)$. For instance, since visible state v_1 was emitted at $t = 1$, we have $\alpha_0(1) = \alpha_1(0)a_{10}b_{01} = 1[0.2 \times 0] = 0$, as shown by the top arrow.
- Likewise the next highest arrow corresponds to the calculation $\alpha_1(1) = \alpha_1(0)a_{11}b_{11} = 1[0.3 \times 0.3] = 0.09$.
- In this example, the calculation of $\alpha_i(1)$ is particularly simple, since only transitions from the known initial hidden state need be considered; all other transitions have zero contribution to $\alpha_i(1)$. For subsequent times, however, the calculation requires a sum over all hidden states at the previous time, as given by line 3 in the Forward algorithm. The probability shown in the final (absorbing) state gives the probability of the full sequence observed, $P(V^T) = 0.0011$.

Making Decisions

Given the ability to calculate the probability of an observed sequence. We can now compare different HMMs. This is just Bayesian Decision theory revisited!

Recall:

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


The matrices
a and b

Hence given model θ_1 and θ_2 we select θ_1 if:

$$P(\theta_1 | \mathbf{V}^T) > P(\theta_2 | \mathbf{V}^T)$$

or equivalently

$$P(\mathbf{V}^T | \theta_1)P(\theta_1) > P(\mathbf{V}^T | \theta_2)P(\theta_2)$$

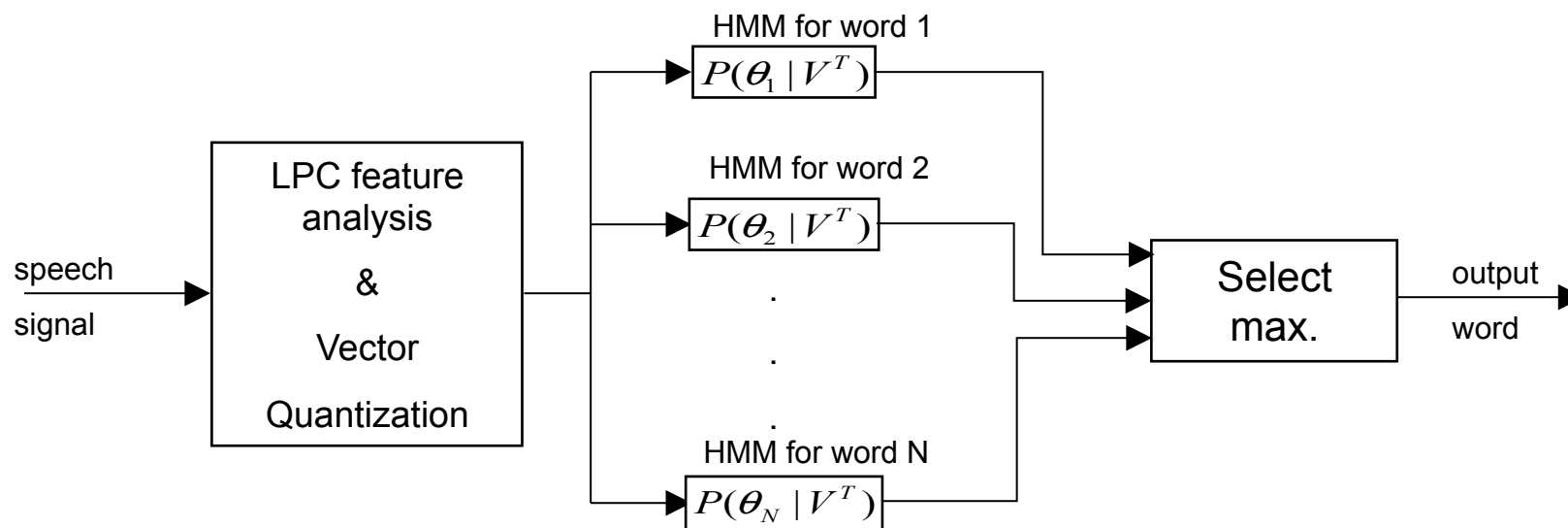
We can calculate this since Forward Alg. gives us $P(\mathbf{V}^T | \theta)$ and $P(\theta)$ is our prior belief in the model (e.g. from a language model).

Example: suppose $\theta_1 = \text{'y'-'e'-'s'}$ and $\theta_2 = \text{'n'-'o'}$. If we expect that the answer is more likely to be 'yes' we weight the priors accordingly.

HMMs for speech recognition

In ASR the observed data is usually a measure of the short term spectral properties of the speech. There are two popular approaches:

1. *Continuous Density observations* – The finite states $\omega(t)$ are mapped into a continuous feature space using a MoG density model.
2. *VQ observations* – the continuous feature space is discretized into a finite symbol set using vector quantization.



An example of an isolated word HMM recognition system:

An alternative recursion Backward Algorithm

We define $\beta_i(t)$ as prob. of generating the remaining $\{v(t+1), \dots, v(T)\}$ given that the model is in state ω_i at time t .

We therefore have the following *Backward Algorithm* :

1. Initialize $t \leftarrow T, a_{ij}, b_{jk}$, visible sequence $\mathbf{V}^T, \beta_j(T) = 1$
2. Increment $t \leftarrow t - 1$
3. $\beta_j(t) = \sum_{i=1}^c \beta_i(t+1) a_{ij} b_{j,v(t+1)}$
4. If $t > 1$, repeat from 2.
5. Return $P(\mathbf{V}^T | \boldsymbol{\theta}) \leftarrow \beta_i(0)$ for the known initial state

Decoding Problem

The problem is to choose the most likely state sequence.

Unlike the evaluation problem, it is not uniquely defined. For example at time t we could find:

$$\hat{\omega}(t) = \arg \max_{\omega(t)} P(\omega(t) | V^T)$$

since $P(\omega(t) | V^T) \propto P(\omega(t), V^T)$ it is equivalent to maximise

$$\begin{aligned} P(\omega(t), V^T) &= P(\omega(t), V^t) P(v(t+1) \dots v(T) | \omega(t)) \\ &= \alpha(t) \beta(t) \end{aligned}$$

However this only finds the states that are individually most likely – hence the sequence, ω^T , may not be viable.

Viterbi Algorithm

Alternatively find the *single* most probable state sequence, ω^T .

Define

$$\delta_i(t) = \max_{\omega(1) \cdots \omega(t-1)} P(\omega(1) \cdots \omega(t-1), \omega(t) = \omega_i, v(1) \cdots v(t))$$

and recall $P(\omega^t, V^t) = P(v(t) | \omega(t))P(\omega(t) | \omega(t-1))P(\omega^{t-1}, V^{t-1})$.

Hence

$$\delta_j(t+1) = \max_i [\delta_i(t) a_{ij}] b_{j,v(t+1)}$$

To find the path, we keep track of $\max \arg i$ using

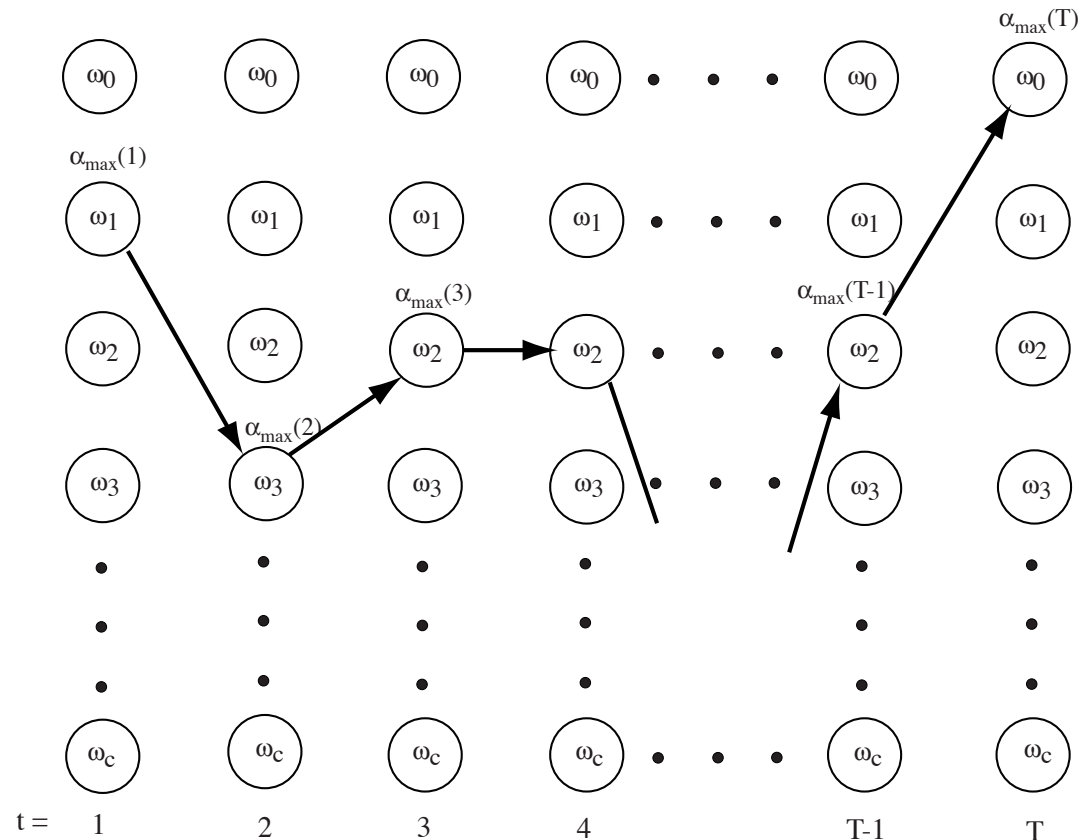
$$\psi_j(t+1) = \arg \max_i [\delta_i(t) a_{ij}].$$

Note the similarity to the forward algorithm. Here we have replaced the \sum_i operation by a \max_i operation.

Viterbi Algorithm

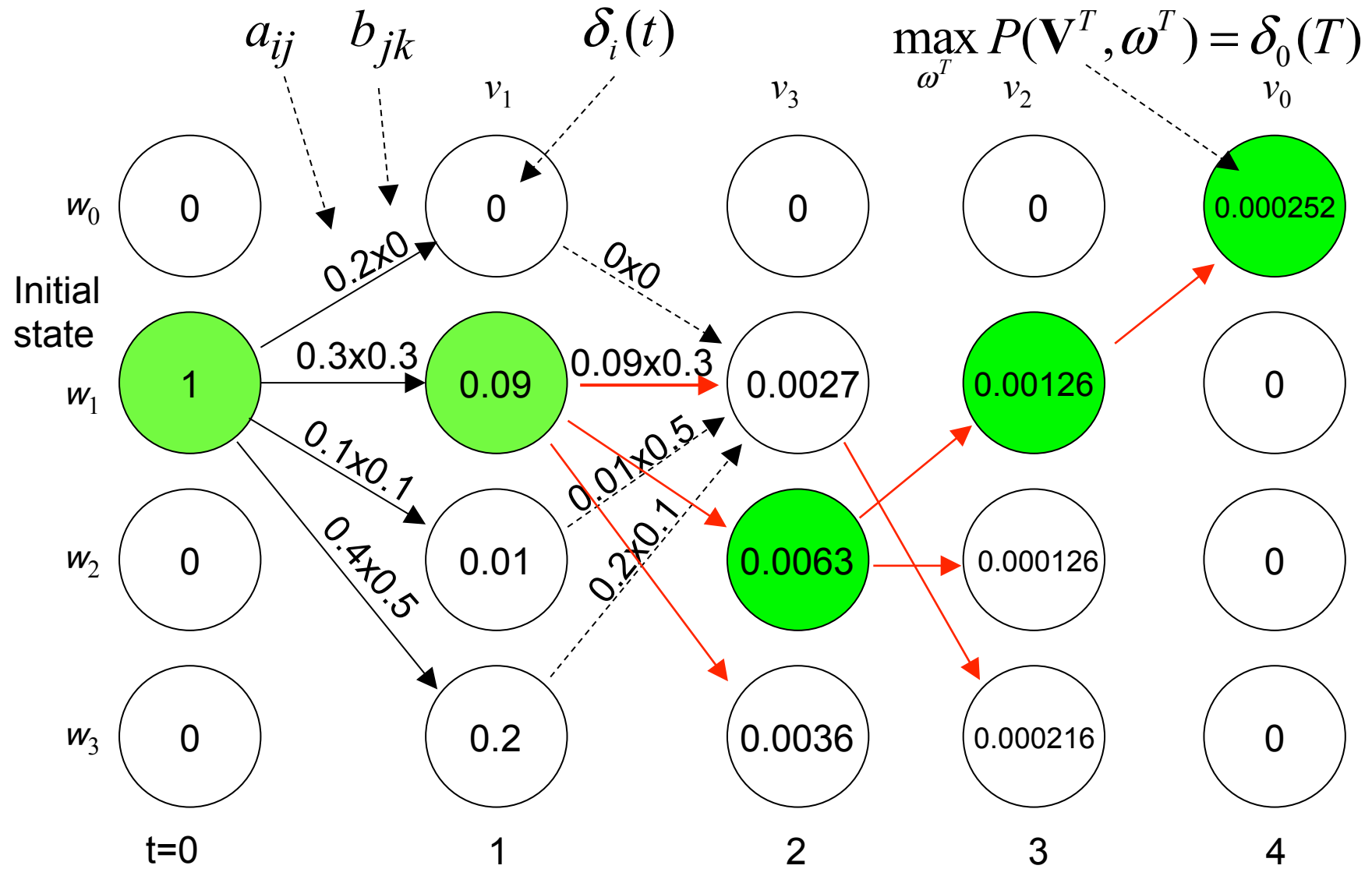
1. Initialize : $\delta_i(0) = 1$ for init state $\tilde{\omega}(0)$, 0 otherwise; $t \leftarrow 0$
2. Recursion : $t \leftarrow t + 1$
3. $\delta_j(t) \leftarrow \max_i [\delta_i(t-1)a_{ij}] b_{j,v(t)}$
4. $\psi_j(t) \leftarrow \arg \max_i [\delta_i(t-1)a_{ij}]$
5. If $t < T$ repeat from 2.
6. Termination : $\tilde{\omega}(T) \leftarrow \arg \max_i \delta_i(T)$
7. Backtracking :
8. $t \leftarrow t - 1$; $\tilde{\omega}(t) \leftarrow \psi_{\tilde{\omega}(t+1)}(t+1)$
9. If $t > 1$ repeat from 8.
10. Output state sequence $\{\tilde{\omega}(0), \dots, \tilde{\omega}(T)\}$.

Viterbi HMM Decoding seen in a 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.

Decoding Example



Solution to this example

We can ignore ω_0 until the final time $t=4$.

$t=1$

$$\Delta_1(1) = 0.3 \cdot 0.3 = 0.09$$

$$\Delta_2(1) = 0.1 \cdot 0.1 = 0.01$$

$$\Delta_3(1) = 0.4 \cdot 0.5 = 0.2$$

$$\Phi_i(1) = 1 \text{ in each case (as } \omega_1 \text{ is known initial state)}$$

$t=2$

$$\Delta_1(2) = \max(\underline{0.09 \cdot 0.3}, 0.01 \cdot 0.5, 0.2 \cdot 0.1) \cdot 0.1 = 0.0027; \Phi_1(2) = 1;$$

$$\Delta_2(2) = \max(\underline{0.09 \cdot 0.1}, 0.01 \cdot 0.2, 0.2 \cdot 0) \cdot 0.7 = 0.0063; \Phi_2(2) = 1;$$

$$\Delta_3(2) = \max(\underline{0.09 \cdot 0.4}, 0.01 \cdot 0.1, 0.2 \cdot 0.1) \cdot 0.1 = 0.0036; \Phi_3(2) = 1;$$

$t=3$

$$\begin{aligned} \Delta_1(3) &= \max(0.0027 \cdot 0.3, \underline{0.0063 \cdot 0.5}, 0.0036 \cdot 0.1) \cdot 0.4 \\ &= \max(0.00081, \underline{0.00315}, 0.00036) \cdot 0.4 = 0.00126; \Phi_1(3) = 2; \end{aligned}$$

$$\begin{aligned} \Delta_2(3) &= \max(0.0027 \cdot 0.1, \underline{0.0063 \cdot 0.2}, 0.0036 \cdot 0) \cdot 0.1 \\ &= \max(0.00027, \underline{0.00126}, 0) \cdot 0.1 = 0.000126; \Phi_2(3) = 2; \end{aligned}$$

$$\begin{aligned} \Delta_3(3) &= \max(\underline{0.0027 \cdot 0.4}, 0.0063 \cdot 0.1, 0.0036 \cdot 0.1) \cdot 0.2 \\ &= \max(\underline{0.00108}, 0.00063, 0.00036) \cdot 0.2 = 0.000216; \Phi_3(3) = 1; \end{aligned}$$

$t=4$ (only need to consider $\Delta_0(4)$...)

$$\begin{aligned} \Delta_0(4) &= \max(0.00126 \cdot 0.2, 0.000126 \cdot 0.2, 0.000216 \cdot 0.8) \cdot 1 \\ &= \max(\underline{0.000252}, 0.0000252, 0.0001728) \cdot 1 = 0.000252; \Phi_0(4) = 1 \end{aligned}$$

Working backwards we have:

$$\Omega(4) = 0$$

$$\Omega(3) = 1; (\Phi_0(4) = 1)$$

$$\Omega(2) = 2; (\Phi_1(3) = 2)$$

$$\Omega(1) = 1; (\Phi_2(2) = 1)$$

$$\Omega(0) = 1; \text{ (known initial state)}$$

The Learning Problem

The 3rd problem is the most difficult: to learn the parameters, a_{ij} and b_{jk} from a set of training data.

Obvious approach: Maximum Likelihood Learning

However we have a familiar problem:

$$\begin{aligned}\{\hat{a}_{ij}, \hat{b}_{jk}\} &= \arg \max_{a_{ij}, b_{jk}} P(V^T \mid a_{ij}, b_{jk}) \\ &= \arg \max_{a_{ij}, b_{jk}} \sum_{\omega^T} P(\omega^T, V^T \mid a_{ij}, b_{jk})\end{aligned}$$

That is: we must **marginalize** out the state sequences, ω^T . Here we can use EM algorithm (called Baum-Welch alg. in speech recognition).

The Learning Problem (cont.)

- Similar to learning prior probability weights in MoGs, we iteratively estimate the transition probabilities, a_{ij} and the emission probabilities, b_{jk}
- The key ingredient is the joint probability of $\omega_j(t)$ and $\omega_i(t)$ given the observed data:

$$\begin{aligned}\gamma_{ij}(t) &= P(\omega_j(t), \omega_i(t-1) | V^T) \\ &= P(\omega_j(t), \omega_i(t-1), V^T) / P(V^T) \\ &= P(V^{t+1:T} | \omega_j(t)) \times P(\omega_j(t) | \omega_i(t-1)) \times P(v(t) | \omega_j(t)) \times P(\omega_i(t-1), V^{t-1}) / P(V^T) \\ &= \beta_j(t) \times a_{ij} \times b_{j,v(t)} \times \alpha_i(t-1) / P(V^T)\end{aligned}$$

- We see that it can be calculated from the backward and forward steps and the current estimates of a_{ij} and b_{jk}

The Learning Problem (cont.)

Updating a_{ij} requires the estimated prob. of moving from state i to state j , hence:

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

Expected number of transitions from $i \rightarrow \text{anywhere}$ \rightarrow

Expected number of transitions from $i \rightarrow j$ \leftarrow

Updating b_{jk} requires the estimated prob. of emitting visible symbol v_k when in state j , hence:

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

Expected number of occurrences of state j \rightarrow

Expected number of occurrences of state j emitting v_k \leftarrow

...then repeat until convergence.

The forward-backward algorithm

```
1 begin initialize  $a_{ij}, b_{jk}$ , training sequence  $V^T$ , convergence criterion  $\theta$ 
2       do  $z \leftarrow z + 1$ 
3           Compute  $\hat{a}(z)$  from  $a(z-1)$  and  $b(z-1)$ 
4           Compute  $\hat{b}(z)$  from  $a(z-1)$  and  $b(z-1)$ 
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
```