

Hidden Markov Models

Hidden Markov Models



Probability Recap

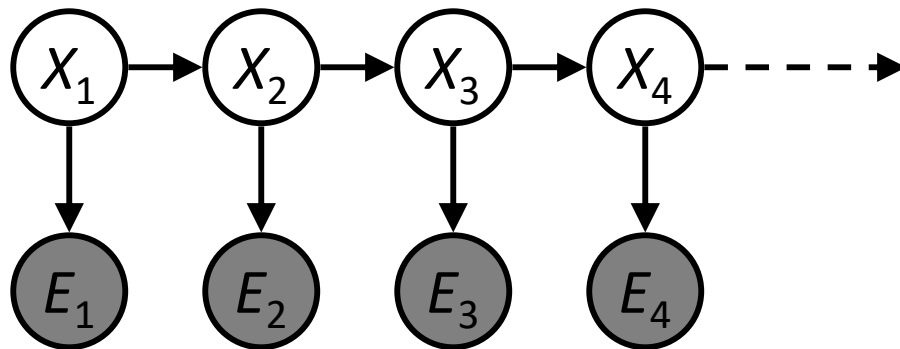
- Conditional probability $P(x|y) = \frac{P(x, y)}{P(y)}$
- Product rule $P(x, y) = P(x|y)P(y)$
- Chain rule
$$\begin{aligned} P(X_1, X_2, \dots, X_n) &= P(X_1)P(X_2|X_1)P(X_3|X_1, X_2) \dots \\ &= \prod_{i=1}^n P(X_i|X_1, \dots, X_{i-1}) \end{aligned}$$
- X, Y independent if and only if: $\forall x, y : P(x, y) = P(x)P(y)$
- X and Y are conditionally independent given Z if and only if: $X \perp\!\!\!\perp Y | Z$
$$\forall x, y, z : P(x, y|z) = P(x|z)P(y|z)$$

Hidden Markov Models

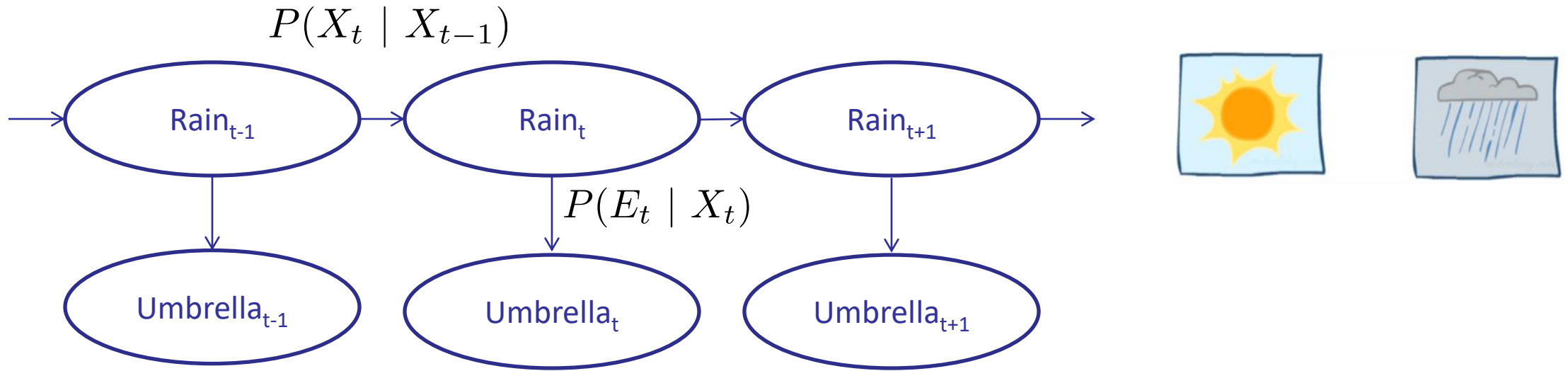


Hidden Markov Models

- Markov chains not so useful for most agents
 - Need observations to update your beliefs
- Hidden Markov models (HMMs)
 - Underlying Markov chain over states X
 - You observe outputs (effects) at each time step



Example: Weather HMM

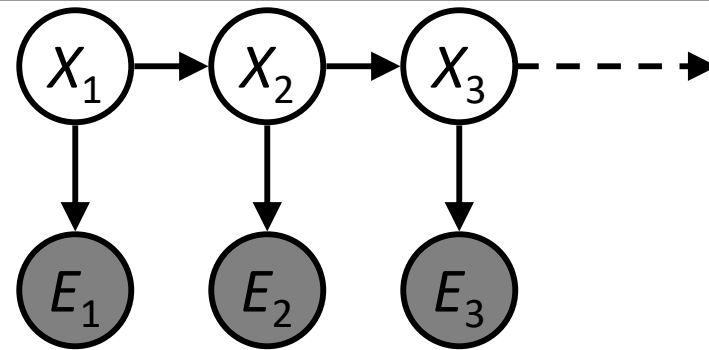


- An HMM is defined by:
 - Initial distribution: $P(X_1)$
 - Transitions: $P(X_t | X_{t-1})$
 - Emissions: $P(E_t | X_t)$

R_t	R_{t+1}	$P(R_{t+1} R_t)$
+r	+r	0.7
+r	-r	0.3
-r	+r	0.3
-r	-r	0.7

R_t	U_t	$P(U_t R_t)$
+r	+u	0.9
+r	-u	0.1
-r	+u	0.2
-r	-u	0.8

Joint Distribution of an HMM



- Joint distribution:

$$P(X_1, E_1, X_2, E_2, X_3, E_3) = P(X_1)P(E_1|X_1)P(X_2|X_1)P(E_2|X_2)P(X_3|X_2)P(E_3|X_3)$$

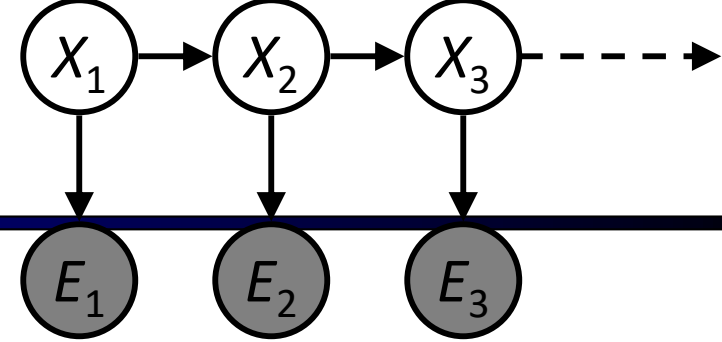
- More generally:

$$P(X_1, E_1, \dots, X_T, E_T) = P(X_1)P(E_1|X_1) \prod_{t=2}^T P(X_t|X_{t-1})P(E_t|X_t)$$

- Questions to be resolved:

- Does this indeed define a joint distribution?
- Can every joint distribution be factored this way, or are we making some assumptions about the joint distribution by using this factorization?

Chain Rule and HMMs



- From the chain rule, *every* joint distribution over $X_1, E_1, X_2, E_2, X_3, E_3$ can be written as:

$$P(X_1, E_1, X_2, E_2, X_3, E_3) = P(X_1)P(E_1|X_1)P(X_2|X_1, E_1)P(E_2|X_1, E_1, X_2) \\ P(X_3|X_1, E_1, X_2, E_2)P(E_3|X_1, E_1, X_2, E_2, X_3)$$

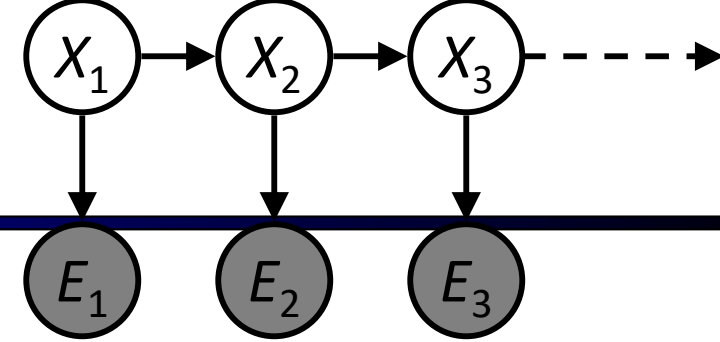
- Assuming that*

$$X_2 \perp\!\!\!\perp E_1 \mid X_1, \quad E_2 \perp\!\!\!\perp X_1, E_1 \mid X_2, \quad X_3 \perp\!\!\!\perp X_1, E_1, E_2 \mid X_2, \quad E_3 \perp\!\!\!\perp X_1, E_1, X_2, E_2 \mid X_3$$

gives us the expression posited on the previous slide:

$$P(X_1, E_1, X_2, E_2, X_3, E_3) = P(X_1)P(E_1|X_1)P(X_2|X_1)P(E_2|X_2)P(X_3|X_2)P(E_3|X_3)$$

Chain Rule and HMMs



- From the chain rule, *every* joint distribution over $X_1, E_1, \dots, X_T, E_T$ can be written as:

$$P(X_1, E_1, \dots, X_T, E_T) = P(X_1)P(E_1|X_1) \prod_{t=2}^T P(X_t|X_1, E_1, \dots, X_{t-1}, E_{t-1})P(E_t|X_1, E_1, \dots, X_{t-1}, E_{t-1}, X_t)$$

- Assuming* that for all t :

- State independent of all past states and all past evidence given the previous state, i.e.:

$$X_t \perp\!\!\!\perp X_1, E_1, \dots, X_{t-2}, E_{t-2}, E_{t-1} \mid X_{t-1}$$

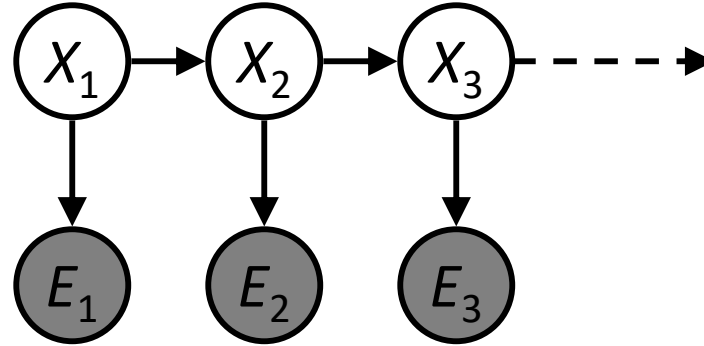
- Evidence is independent of all past states and all past evidence given the current state, i.e.:

$$E_t \perp\!\!\!\perp X_1, E_1, \dots, X_{t-2}, E_{t-2}, X_{t-1}, E_{t-1} \mid X_t$$

gives us the expression posited on the earlier slide:

$$P(X_1, E_1, \dots, X_T, E_T) = P(X_1)P(E_1|X_1) \prod_{t=2}^T P(X_t|X_{t-1})P(E_t|X_t)$$

Implied Conditional Independencies



- Many implied conditional independencies, e.g.,

$$E_1 \perp\!\!\!\perp X_2, E_2, X_3, E_3 \mid X_1$$

- To prove them

- Approach 1: follow similar (algebraic) approach to what we did in the Markov models lecture
- Approach 2: directly from the graph structure (3 lectures from now)
 - Intuition: If path between U and V goes through W, then $U \perp\!\!\!\perp V \mid W$ [Some fineprint later]

HMM components: Here ($X_i = Q_i$) and ($E_i = O_i$)







$Q = q_1 q_2 \dots q_N$	a set of N states
$A = a_{11} \dots a_{ij} \dots a_{NN}$	a transition probability matrix A , each a_{ij} representing the probability of moving from state i to state j , s.t. $\sum_{j=1}^N a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \dots o_T$	a sequence of T observations , each one drawn from a vocabulary $V = v_1, v_2, \dots, v_V$
$B = b_i(o_t)$	a sequence of observation likelihoods , also called emission probabilities , each expressing the probability of an observation o_t being generated from a state i
$\pi = \pi_1, \pi_2, \dots, \pi_N$	an initial probability distribution over states. π_i is the probability that the Markov chain will start in state i . Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^n \pi_i = 1$

Three Problems of HMM

An influential tutorial by [Rabiner \(1989\)](#), based on tutorials by Jack Ferguson in the 1960s, introduced the idea that hidden Markov models should be characterized by **three fundamental problems**:

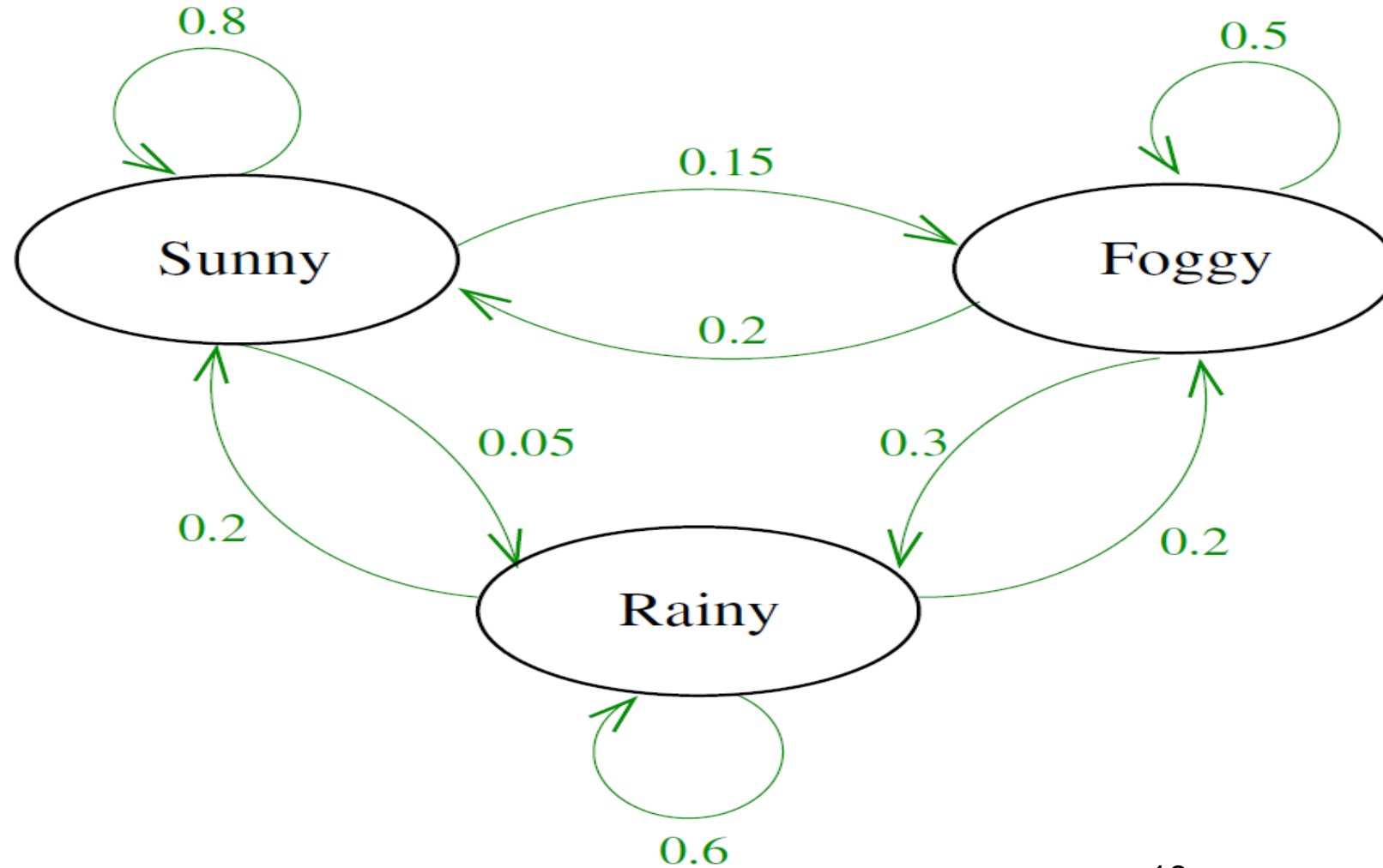
- Problem 1 (Likelihood):** Given an HMM $\lambda = (A, B)$ and an observation sequence O , determine the likelihood $P(O|\lambda)$.
- Problem 2 (Decoding):** Given an observation sequence O and an HMM $\lambda = (A, B)$, discover the best hidden state sequence Q .
- Problem 3 (Learning):** Given an observation sequence O and the set of states in the HMM, learn the HMM parameters A and B .

HMM Example

Today's weather	Tomorrow's weather		
			
	0.8	0.05	0.15
	0.2	0.6	0.2
	0.2	0.3	0.5

Probabilities $p(q_{n+1}|q_n)$ of tomorrow's weather based on today's weather

Markov model with Graz weather with state transition probabilities



Probability of Umbrella

Weather	Probability of umbrella
Sunny	0.1
Rainy	0.8
Foggy	0.3

Example1

1. Suppose the day you were locked in it was sunny. The next day, the caretaker carried an umbrella into the room. You would like to know, what the weather was like on this second day.

First we calculate the likelihood for the second day to be sunny:

$$\begin{aligned} L(q_2 = \text{☀}|q_1 = \text{☀}, x_2 = \text{☂}) &= P(x_2 = \text{☂}|q_2 = \text{☀}) \cdot P(q_2 = \text{☀}|q_1 = \text{☀}) \\ &= 0.1 \cdot 0.8 = 0.08, \end{aligned}$$

then for the second day to be rainy:

$$\begin{aligned} L(q_2 = \text{☁}|q_1 = \text{☀}, x_2 = \text{☂}) &= P(x_2 = \text{☂}|q_2 = \text{☁}) \cdot P(q_2 = \text{☁}|q_1 = \text{☀}) \\ &= 0.8 \cdot 0.05 = 0.04, \end{aligned}$$

and finally for the second day to be foggy:

$$\begin{aligned} L(q_2 = \text{☁}|q_1 = \text{☀}, x_2 = \text{☂}) &= P(x_2 = \text{☂}|q_2 = \text{☁}) \cdot P(q_2 = \text{☁}|q_1 = \text{☀}) \\ &= 0.3 \cdot 0.15 = 0.045. \end{aligned}$$

Thus, although the caretaker did carry an umbrella, it is most likely that on the second day the weather was sunny.

Example 2

2. Suppose you do not know how the weather was when you were locked in. The following three days the caretaker always comes without an umbrella. Calculate the likelihood for the weather on these three days to have been $\{q_1 = \text{☀}, q_2 = \text{☁}, q_3 = \text{☀}\}$. As you do not know how the weather is on the first day, you assume the 3 weather situations are equi-probable on this day (cf. footnote on page 2), and the *prior probability* for sun on day one is therefore $P(q_1 = \text{☀}|q_0) = P(q_1 = \text{☀}) = 1/3$.

$$\begin{aligned} L(q_1 = \text{☀}, q_2 = \text{☁}, q_3 = \text{☀} | x_1 = \text{☂}, x_2 = \text{☂}, x_3 = \text{☂}) &= \\ P(x_1 = \text{☂} | q_1 = \text{☀}) \cdot P(x_2 = \text{☂} | q_2 = \text{☁}) \cdot P(x_3 = \text{☂} | q_3 = \text{☀}) \cdot \\ P(q_1 = \text{☀}) \cdot P(q_2 = \text{☁} | q_1 = \text{☀}) \cdot P(q_3 = \text{☀} | q_2 = \text{☁}) &= \\ 0.9 \cdot 0.7 \cdot 0.9 \cdot 1/3 \cdot 0.15 \cdot 0.2 &= 0.0057 \end{aligned}$$

ICE Cream Problem

- Imagine that you are a climatologist in the year 2799 studying the history of global warming. You cannot find any records of the weather in Amaravati, for the summer of 2020, but you do find Jason Eisner's diary, which lists how many ice creams Jason ate every day that summer.
- Our goal is to use these observations to estimate the temperature every day.
- We'll simplify this weather task by assuming there are only two kinds of days: cold (C) and hot (H).
- So the Eisner task is as follows: Given a sequence of observations O (each an integer representing the number of ice creams eaten on a given day) find the 'hidden' sequence Q of weather states (H or C) which caused Jason to eat the ice cream.
- Figure A.2 shows a sample HMM for the ice cream task.
- The two hidden states (H and C) correspond to hot and cold weather, and the observations (drawn from the alphabet $O = \{1,2,3\}$) correspond to the number of ice creams eaten by Jason on a given day

ICE Cream Problem

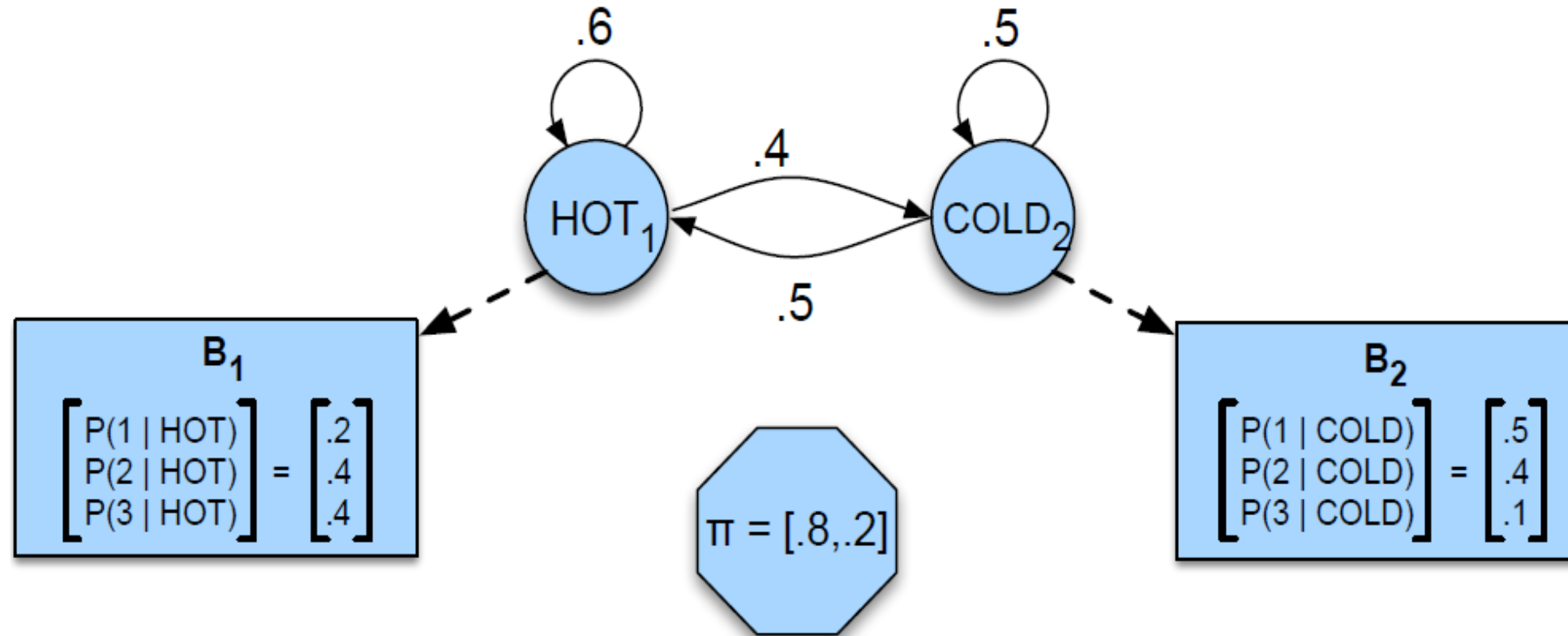


Figure A.2 A hidden Markov model for relating numbers of ice creams eaten by Jason (the observations) to the weather (H or C, the hidden variables).

Question ???

- Compute the probability of ice-cream events 3 1 3 instead by summing over all possible weather sequences, weighted by their probability.
- First, let's compute the joint probability of being in a particular weather sequence Q and generating a particular sequence O of ice-cream events.

The computation of the joint probability of our ice-cream observation 3 1 3 and one possible hidden state sequence **hot hot cold** is shown below

$$P(O, Q) = P(O|Q) \times P(Q) = \prod_{i=1}^T P(o_i|q_i) \times \prod_{i=1}^T P(q_i|q_{i-1})$$

$$P(3 \ 1 \ 3, \text{hot hot cold}) = P(\text{hot}|\text{start}) \times P(\text{hot}|\text{hot}) \times P(\text{cold}|\text{hot}) \\ \times P(3|\text{hot}) \times P(1|\text{hot}) \times P(3|\text{cold})$$

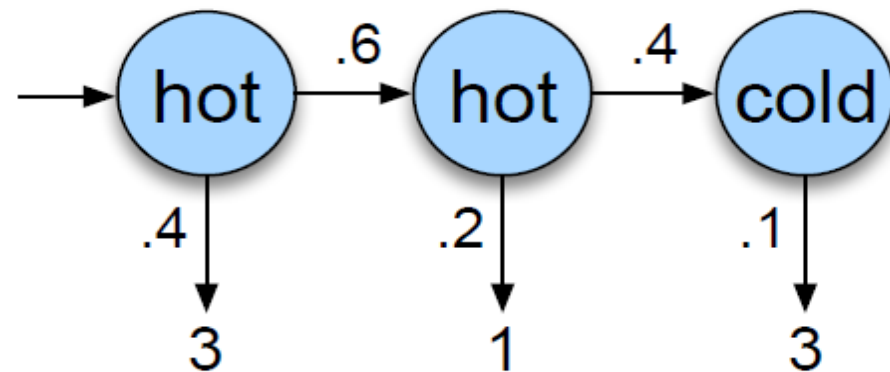


Figure A.4 The computation of the joint probability of the ice-cream events 3 1 3 and the hidden state sequence *hot hot cold*.

-
- $N = \text{No. of hidden states} = 2$
 - $T = \text{Given no.of observations} = 3$
 - $M = N^T = 2^3 = 8$ possibilities (HHC, CCC, HCH...)

- $N = 7$
- $T = 10$
- $N^T = 7^{10}$

How to solve this ?

Each cell of the forward algorithm trellis $\alpha_t(j)$ represents the probability of being in state j after seeing the first t observations, given the automaton λ . The value of each cell $\alpha_t(j)$ is computed by summing over the probabilities of every path that could lead us to this cell. Formally, each cell expresses the following probability:

$$\alpha_t(j) = P(o_1, o_2 \dots o_t, q_t = j | \lambda) \quad (\text{A.11})$$

Here, $q_t = j$ means “the t^{th} state in the sequence of states is state j ”. We compute this probability $\alpha_t(j)$ by summing over the extensions of all the paths that lead to the current cell. For a given state q_j at time t , the value $\alpha_t(j)$ is computed as

$$\alpha_t(j) = \sum_{i=1}^N \alpha_{t-1}(i) a_{ij} b_j(o_t) \quad (\text{A.12})$$

The three factors that are multiplied in Eq. A.12 in extending the previous paths to compute the forward probability at time t are

$\alpha_{t-1}(i)$	the previous forward path probability from the previous time step
a_{ij}	the transition probability from previous state q_i to current state q_j
$b_j(o_t)$	the state observation likelihood of the observation symbol o_t given the current state j

Forward Trellis

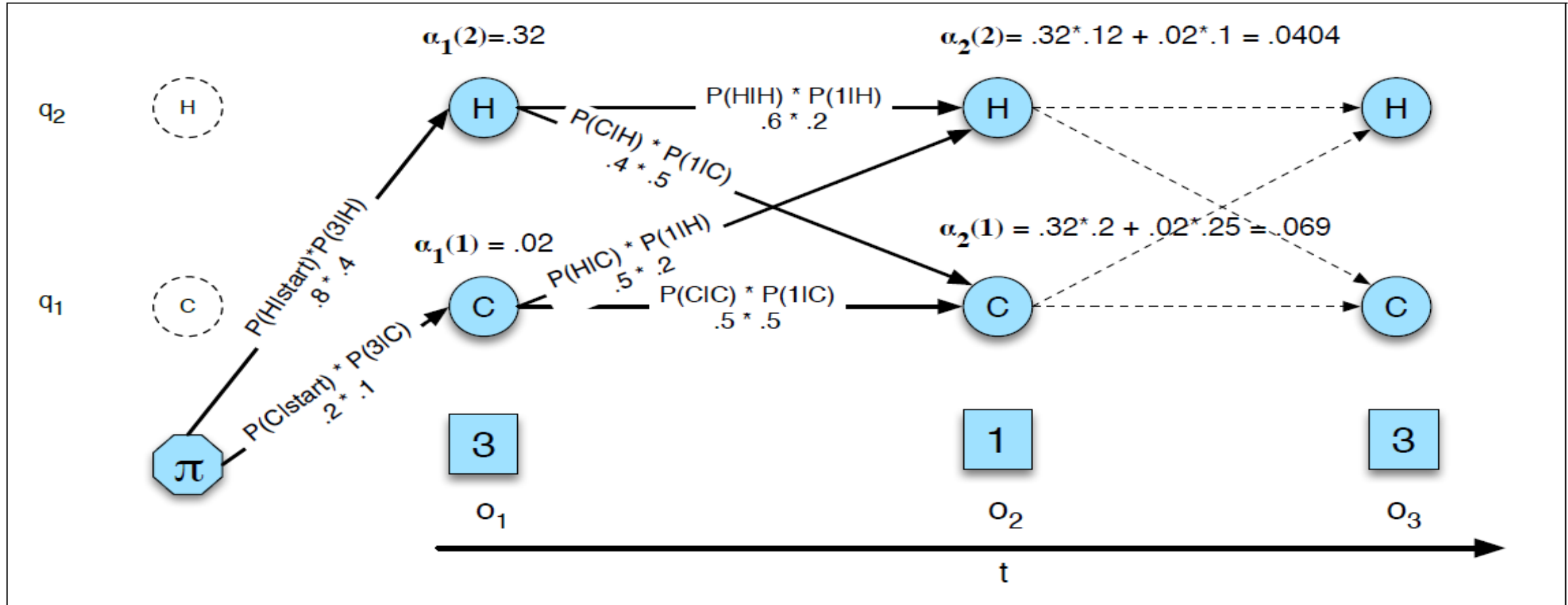


Figure A.5 The forward trellis for computing the total observation likelihood for the ice-cream events 3 1 3. Hidden states are in circles, observations in squares. The figure shows the computation of $\alpha_t(j)$ for two states at two time steps. The computation in each cell follows Eq. A.12: $\alpha_t(j) = \sum_{i=1}^N \alpha_{t-1}(i) a_{ij} b_j(o_t)$. The resulting probability expressed in each cell is Eq. A.11: $\alpha_t(j) = P(o_1, o_2 \dots o_t, q_t = j | \lambda)$ 26

Likelihood Computation: The forward algorithm

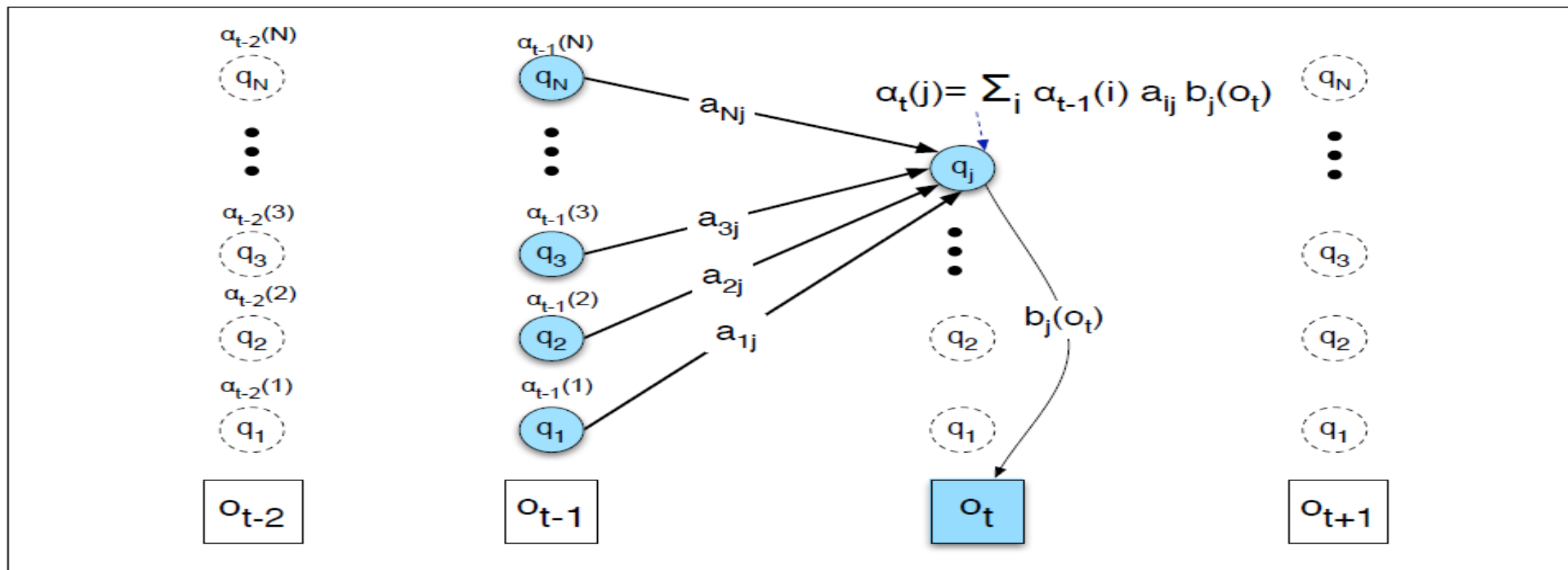


Figure A.6 Visualizing the computation of a single element $\alpha_t(i)$ in the trellis by summing all the previous values α_{t-1} , weighted by their transition probabilities a , and multiplying by the observation probability $b_i(o_t)$. For many applications of HMMs, many of the transition probabilities are 0, so not all previous states will contribute to the forward probability of the current state. Hidden states are in circles, observations in squares. Shaded nodes are included in the probability computation for $\alpha_t(i)$.

The Forward Algorithm

```
function FORWARD(observations of len  $T$ , state-graph of len  $N$ ) returns forward-prob

  create a probability matrix forward[ $N, T$ ]
  for each state  $s$  from 1 to  $N$  do                                ; initialization step
    forward[ $s, 1$ ]  $\leftarrow \pi_s * b_s(o_1)$ 
  for each time step  $t$  from 2 to  $T$  do                            ; recursion step
    for each state  $s$  from 1 to  $N$  do
      forward[ $s, t$ ]  $\leftarrow \sum_{s'=1}^N \textit{forward}[s', t-1] * a_{s',s} * b_s(o_t)$ 
  forwardprob  $\leftarrow \sum_{s=1}^N \textit{forward}[s, T]$                         ; termination step
  return forwardprob
```

Figure A.7 The forward algorithm, where *forward*[s, t] represents $\alpha_t(s)$.

Steps in Forward Algorithm

1. Initialization:

$$\alpha_1(j) = \pi_j b_j(o_1) \quad 1 \leq j \leq N$$

2. Recursion:

$$\alpha_t(j) = \sum_{i=1}^N \alpha_{t-1}(i) a_{ij} b_j(o_t); \quad 1 \leq j \leq N, 1 < t \leq T$$

3. Termination:

$$P(O|\lambda) = \sum_{i=1}^N \alpha_T(i)$$

Decoding: The Viterbi Algorithm

For any model, such as an HMM, that contains hidden variables, the task of determining which sequence of variables is the underlying source of some sequence of observations is called the **decoding** task. In the ice-cream domain, given a sequence of ice-cream observations $3\ 1\ 3$ and an HMM, the task of the **decoder** is to find the best hidden weather sequence ($H\ H\ H$). More formally,

Decoding: Given as input an HMM $\lambda = (A, B)$ and a sequence of observations $O = o_1, o_2, \dots, o_T$, find the most probable sequence of states $Q = q_1 q_2 q_3 \dots q_T$.

We might propose to find the best sequence as follows: For each possible hidden state sequence (HHH , HHC , HCH , etc.), we could run the forward algorithm and compute the likelihood of the observation sequence given that hidden state sequence. Then we could choose the hidden state sequence with the maximum observation likelihood. It should be clear from the previous section that we cannot do this because there are an exponentially large number of state sequences.

Viterbi Algorithm

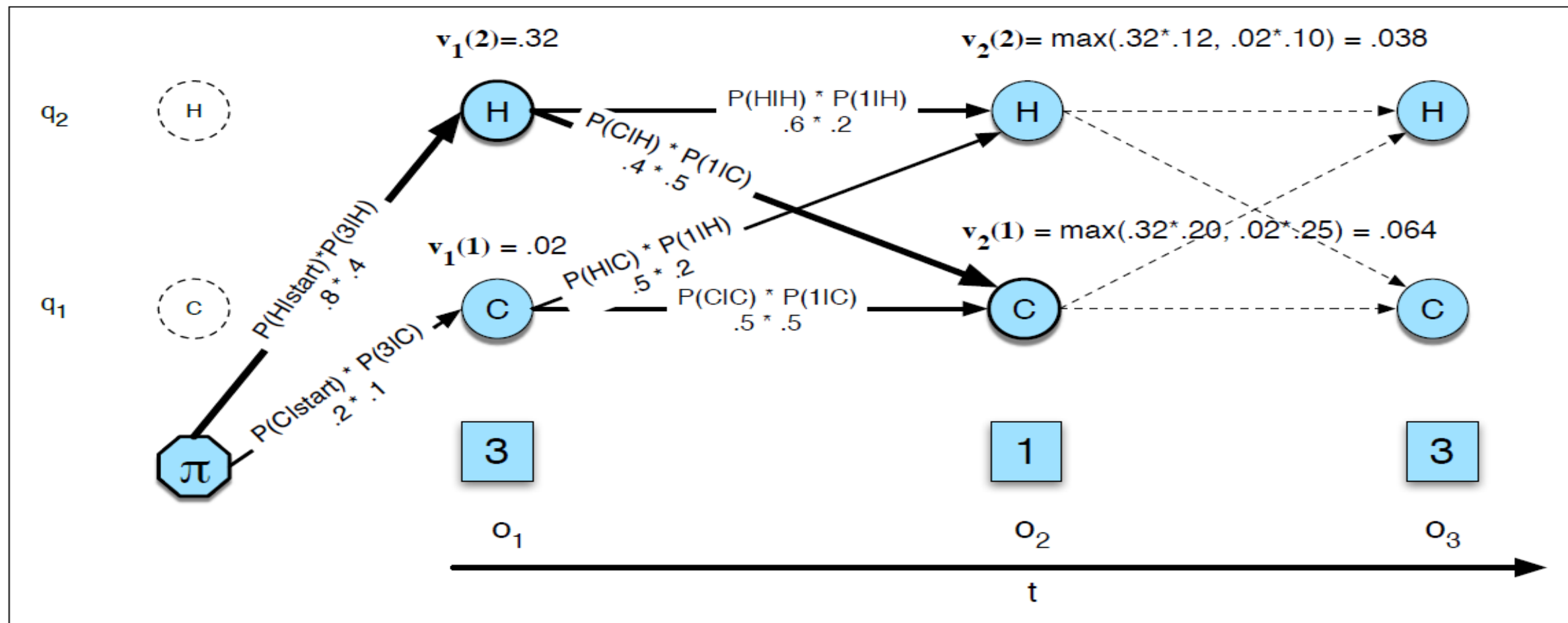


Figure A.8 The Viterbi trellis for computing the best path through the hidden state space for the ice-cream eating events 3 1 3. Hidden states are in circles, observations in squares. White (unfilled) circles indicate illegal transitions. The figure shows the computation of $v_t(j)$ for two states at two time steps. The computation in each cell follows Eq. A.14: $v_t(j) = \max_{1 \leq i \leq N-1} v_{t-1}(i) a_{ij} b_j(o_t)$. The resulting probability expressed in each cell is Eq. A.13: $v_t(j) = P(q_0, q_1, \dots, q_{t-1}, o_1, o_2, \dots, o_t, q_t = j | \lambda)$.

$$v_t(j) = \max_{q_1, \dots, q_{t-1}} P(q_1 \dots q_{t-1}, o_1, o_2 \dots o_t, q_t = j | \lambda) \quad (\text{A.13})$$

Note that we represent the most probable path by taking the maximum over all possible previous state sequences $\max_{q_1, \dots, q_{t-1}}$. Like other dynamic programming algorithms, Viterbi fills each cell recursively. Given that we had already computed the probability of being in every state at time $t - 1$, we compute the Viterbi probability by taking the most probable of the extensions of the paths that lead to the current cell. For a given state q_j at time t , the value $v_t(j)$ is computed as

$$v_t(j) = \max_{i=1}^N v_{t-1}(i) a_{ij} b_j(o_t) \quad (\text{A.14})$$

The three factors that are multiplied in Eq. A.14 for extending the previous paths to compute the Viterbi probability at time t are

$v_{t-1}(i)$	the previous Viterbi path probability from the previous time step
a_{ij}	the transition probability from previous state q_i to current state q_j
$b_j(o_t)$	the state observation likelihood of the observation symbol o_t given the current state j

function VITERBI(*observations* of len T , *state-graph* of len N) **returns** *best-path*, *path-prob*

create a path probability matrix $viterbi[N, T]$

for each state s **from** 1 **to** N **do** ; initialization step

$viterbi[s, 1] \leftarrow \pi_s * b_s(o_1)$

$backpointer[s, 1] \leftarrow 0$

for each time step t **from** 2 **to** T **do** ; recursion step

for each state s **from** 1 **to** N **do**

$viterbi[s, t] \leftarrow \max_{s'=1}^N viterbi[s', t-1] * a_{s', s} * b_s(o_t)$

$backpointer[s, t] \leftarrow \operatorname{argmax}_{s'=1}^N viterbi[s', t-1] * a_{s', s} * b_s(o_t)$

$bestpathprob \leftarrow \max_{s=1}^N viterbi[s, T]$; termination step

$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^N viterbi[s, T]$; termination step

$bestpath \leftarrow$ the path starting at state $bestpathpointer$, that follows $backpointer[]$ to states back in time

return $bestpath$, $bestpathprob$

Figure A.9 Viterbi algorithm for finding optimal sequence of hidden states. Given an observation sequence and an HMM $\lambda = (A, B)$, the algorithm returns the state path through the HMM that assigns maximum likelihood to the observation sequence.

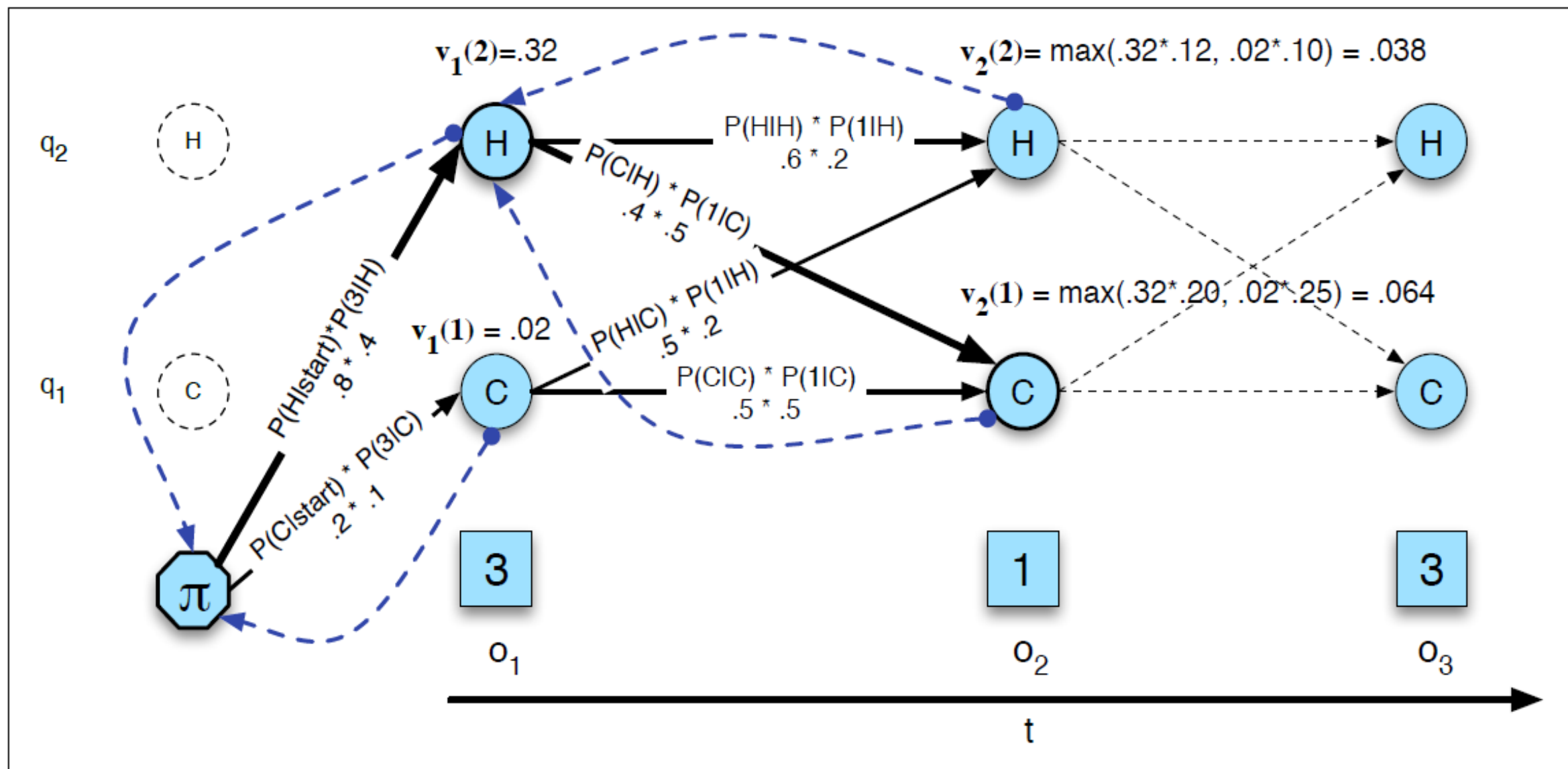


Figure A.10 The Viterbi backtrace. As we extend each path to a new state account for the next observation, we keep a backpointer (shown with broken lines) to the best path that led us to this state.

Finally, we can give a formal definition of the Viterbi recursion as follows:

1. Initialization:

$$\begin{aligned}v_1(j) &= \pi_j b_j(o_1) & 1 \leq j \leq N \\bt_1(j) &= 0 & 1 \leq j \leq N\end{aligned}$$

2. Recursion

$$\begin{aligned}v_t(j) &= \max_{i=1}^N v_{t-1}(i) a_{ij} b_j(o_t); & 1 \leq j \leq N, 1 < t \leq T \\bt_t(j) &= \operatorname{argmax}_{i=1}^N v_{t-1}(i) a_{ij} b_j(o_t); & 1 \leq j \leq N, 1 < t \leq T\end{aligned}$$

3. Termination:

$$\text{The best score: } P^* = \max_{i=1}^N v_T(i)$$

$$\text{The start of backtrace: } q_T^* = \operatorname{argmax}_{i=1}^N v_T(i)$$

Real HMM Examples

- **Speech recognition HMMs:**
 - Observations are acoustic signals (continuous valued)
 - States are specific positions in specific words (so, tens of thousands)
- **Machine translation HMMs:**
 - Observations are words (tens of thousands)
 - States are translation options
- **Robot tracking:**
 - Observations are range readings (continuous)
 - States are positions on a map (continuous)