

# Sequence Labeling with Hidden Markov Models

Kevin Duh

Intro to NLP, Fall 2019

# Sequence Labeling/Tagging Problem

- Input: a sequence of **T** words (e.g. a sentence)
- Output: a sequence of **T** labels/tags, one for each word
- In contrast, text classification:
  - Input: a sequence of **T** words
  - Output: **1** label

# A naive implementation

- Treat each **T** word as independent
- Apply a classifier to each input word independently
- But this ignores sequence structure
  - e.g. maybe some labels are more likely to follow others

# Part-of-Speech Tagging

- Input: The grand jury commented on a number of topics
- Output: DT JJ NN VBD IN DT NN IN NNS

# Named Entity Recognition

- Task: Find text spans that refer to a proper name and label its type, e.g.
  - [George Washington PERSON] was the first president.
  - [Washington ORGANIZATION] won the World Series
- Input: George Washington was the first president
- Output: B-PER I-PER O O O O

# Running Example: Ice Cream & Global Warming (courtesy of Jason Eisner)

- It's 2799. You're a climatologist studying the history of global warming. You have Jason Eisner's diary from 2007, which lists how many ice creams he ate every day.
  - Assume there are two kinds of days: Cold (C) and Hot (H)
- Task: given a sequence of diary observations  $O$ , figure out the correct hidden sequence  $Q$  of weather states.
  - e.g. Jason ate 3 icecreams on Day 1, 1 icecream on Day 2, and 3 icecreams on Day 3. What's the weather on those three days?

# This lecture

- We'll discuss in-depth **Hidden Markov Models (HMM)**
  - Later in the course, we will also cover Conditional Random Fields (CRF), etc.
- HMM is a very useful pedagogical tool to illustrate:
  - How to **probabilistically model** the sequence labeling problem
  - How to efficiently compute with **Dynamic Programming**
  - How a model's parameters can be learned by either **supervised** and **unsupervised** learning (for the latter, we'll focus on **Expectation-Maximization**)

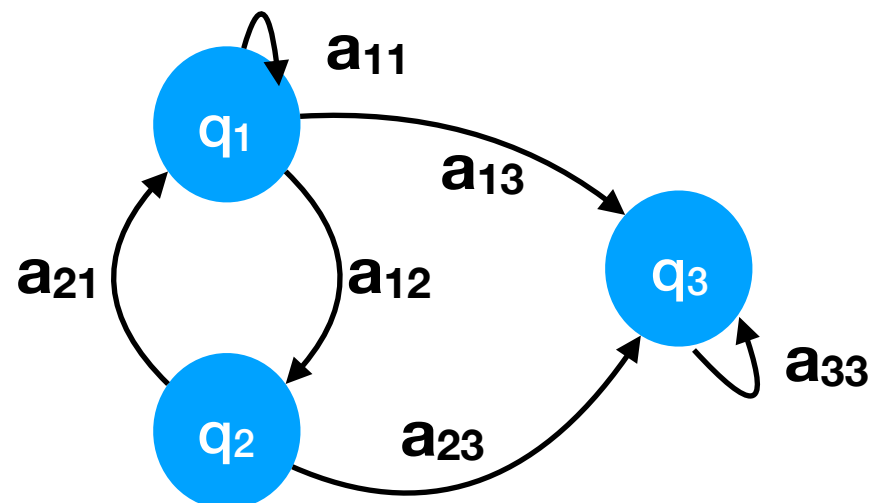
# Outline

- Motivation/Examples
- HMM: Basic Definition & Three Problems
- Problem 1: Likelihood
- Problem 2: Decoding
- Problem 3: Learning
  - Supervised
  - Unsupervised: EM

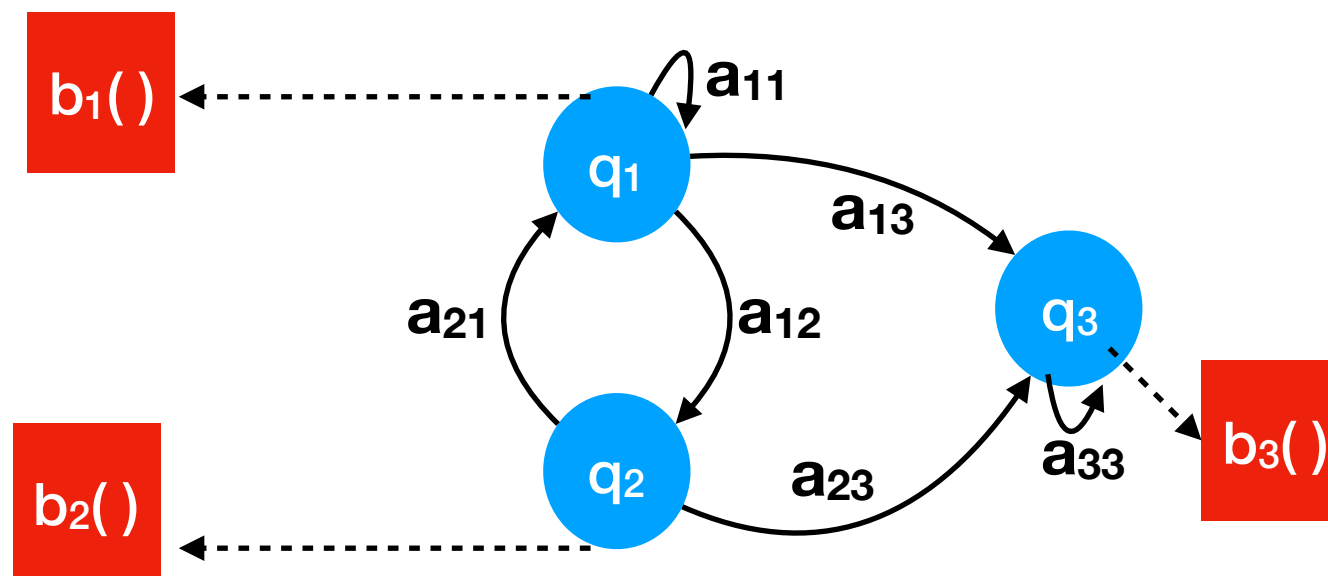


# Markov Chains

- A Markov Chain is specified by:
  - $Q = q_1 q_2 \dots q_N$ : a set of  $N$  states
  - $A = a_{11} a_{12} \dots a_{ij} \dots a_{NN}$ : a transition probability matrix
  - $\pi = \pi_1 \pi_2 \dots \pi_N$ : initial probability distribution
- Markov Assumption:  $P(q_i | q_1 \dots q_{i-1}) = P(q_i | q_{i-1})$



- A **Hidden Markov Model (HMM)** is specified by:
  - $Q = q_1 q_2 \dots q_N$ : a set of  $N$  states
  - $A = a_{11} a_{12} \dots a_{ij} \dots a_{NN}$ : a transition probability matrix
  - $\pi = \pi_1 \pi_2 \dots \pi_N$ : initial probability distribution
  - $B = b_i(o_t)$ : emission probabilities
  - $O = o_1 o_2 \dots o_T$ : a sequence of  $T$  observations
- Markov Assumption:  $P(q_i | q_1 \dots q_{i-1}) = P(q_i | q_{i-1})$
- Output Independence:  $P(o_i | q_1 \dots q_T, o_1 \dots o_T) = P(o_i | q_i)$



# Three Problems for HMM

	Problem 1: Likelihood	Problem 2: Decoding	Problem 3: Learning
Given	HMM parameters $\lambda = (A,B)$ Observation $O$	HMM parameters $\lambda = (A,B)$ Observation $O$	Supervised: $O$ and $Q$ Unsupervised: $O$
Goal	Likelihood $P(O \lambda)$	Most likely hidden sequence $Q$	HMM parameters $\lambda = (A,B)$ that maximize likelihood
Method	Forward Algorithm	Viterbi Algorithm	Supervised: Count Unsupervised: Forward- Backward Algorithm

# Running Example: Ice Cream

- HMM has two states: Cold (C) and Hot (H)
- We have a sequence of integer observations, e.g.
  - **3 1 3** (3 icecream on day 1, 1 icecream on day 2,...)
- Likelihood: What's the probability of 3 1 3 occurring under our HMM model?
- Decoding: What's the most likely weather sequence for 3 1 3? Hot Cold Hot?
- Learning: How to estimate the HMM parameters?
  - Supervised: given 3 1 3 and H C H
  - Unsupervised: given 3 1 3 only

# Outline

- Motivation/Examples
- HMM: Basic Definition & Three Problems
- Problem 1: Likelihood
- Problem 2: Decoding
- Problem 3: Learning
  - Supervised
  - Unsupervised: EM

# Likelihood: $P(O|\lambda = (A,B))$

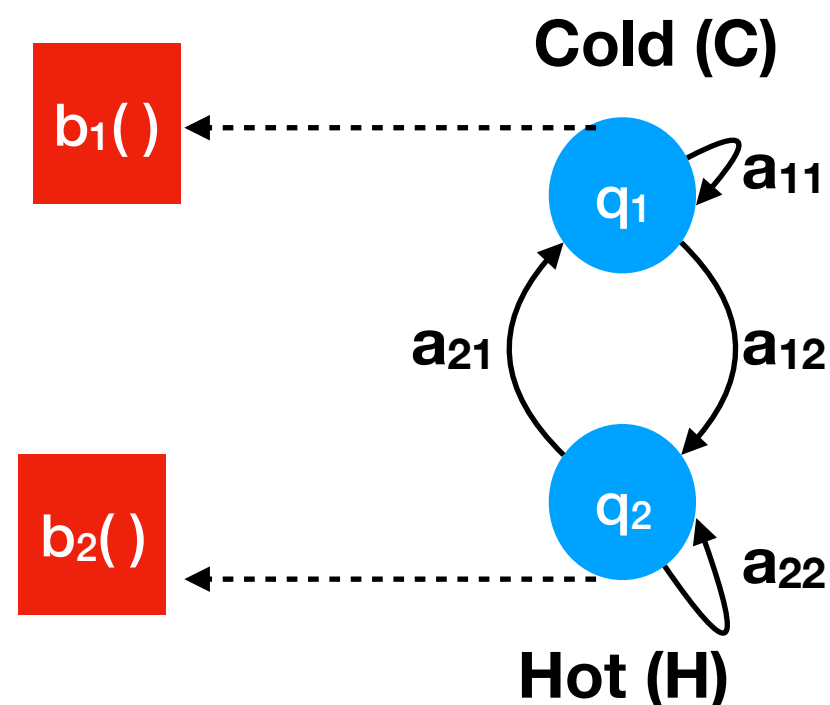
- Joint probability is easy:

$$P(O, Q) = P(O|Q)P(Q) = \prod_{t=1}^T P(o_t|q_t) \times \prod_{t=1}^T P(q_t|q_{t-1})$$

**Note:**  $\lambda = (A,B)$  is implicit; dropped for notational simplicity

- For example:  $Q = H H C$ ,  $O = 3 1 3$

$$\begin{aligned} \text{Then } P(O, Q) &= P(3|H)P(1|H)P(3|C) \times P(H|\text{start})P(H|H)P(C|H) \\ &= b_2(3) \ b_2(1) \ b_1(3) \times \pi_2 \ a_{22} \ a_{23} \end{aligned}$$



# Likelihood: $P(O|\lambda = (A,B))$

- Joint probability is easy:

$$P(O, Q) = P(O|Q)P(Q) = \prod_{t=1}^T P(o_t|q_t) \times \prod_{t=1}^T P(q_t|q_{t-1})$$

**Note:**  $\lambda = (A,B)$  is implicit; dropped for notational simplicity

- Since we don't know  $Q$ , sum over it

$$P(O) = \sum_{all Q} P(O, Q)$$

e.g.  $P(O=313) = P(O=313, Q=HHH) + P(O=313, Q=CCC) + \dots$

For  $N$  states and  $T$ -length sequence, there are  $N^T$  hidden sequences!

# Efficient enumeration by Dynamic Programming: Forward Algorithm

- Prepare a trellis data structure
- Each cell represents probability of being in state  $j$  after seeing the first  $t$  observations:

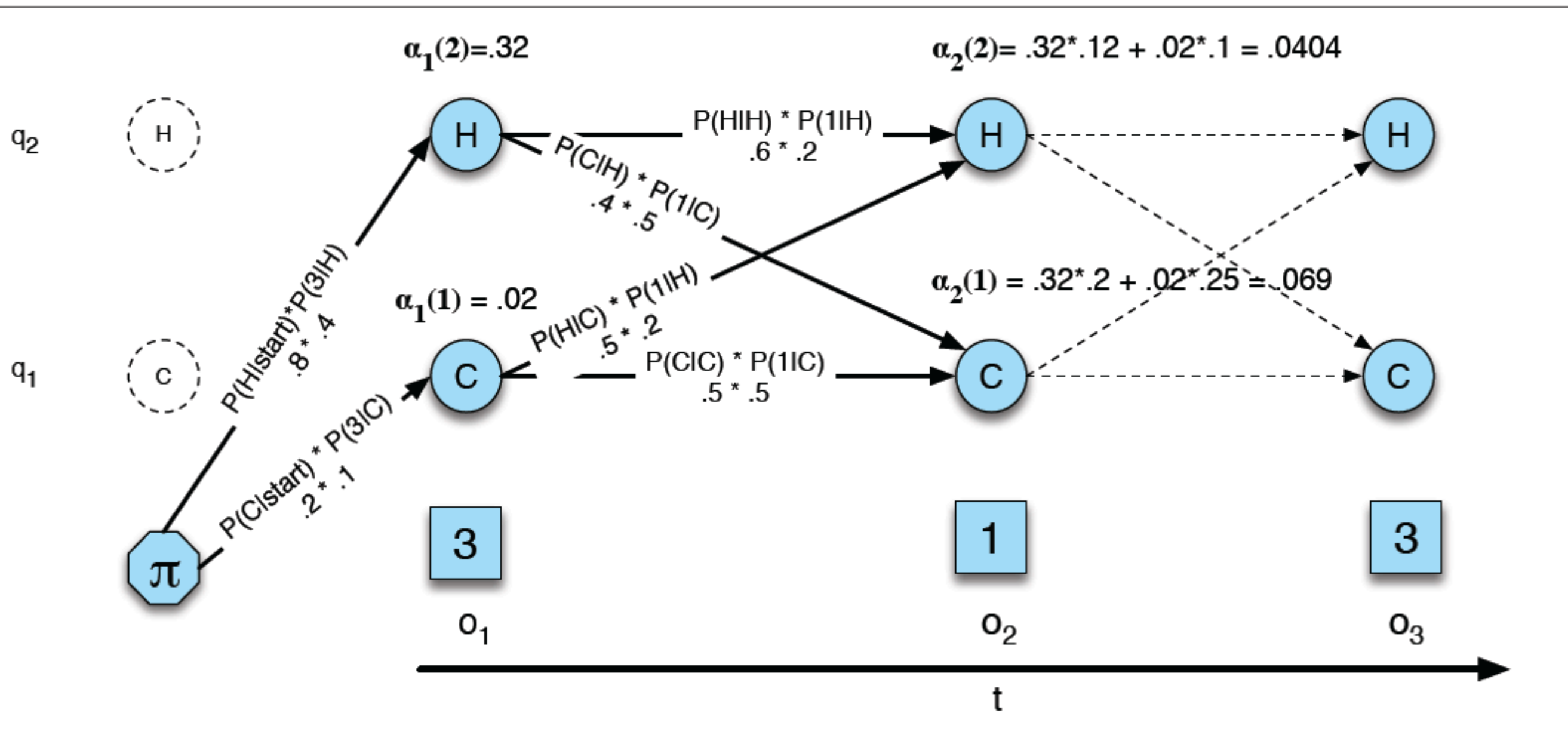
$$\alpha_t(j) = P(o_1, o_2, \dots, o_t, q_t = j | \lambda)$$

- DP subproblem recursion:

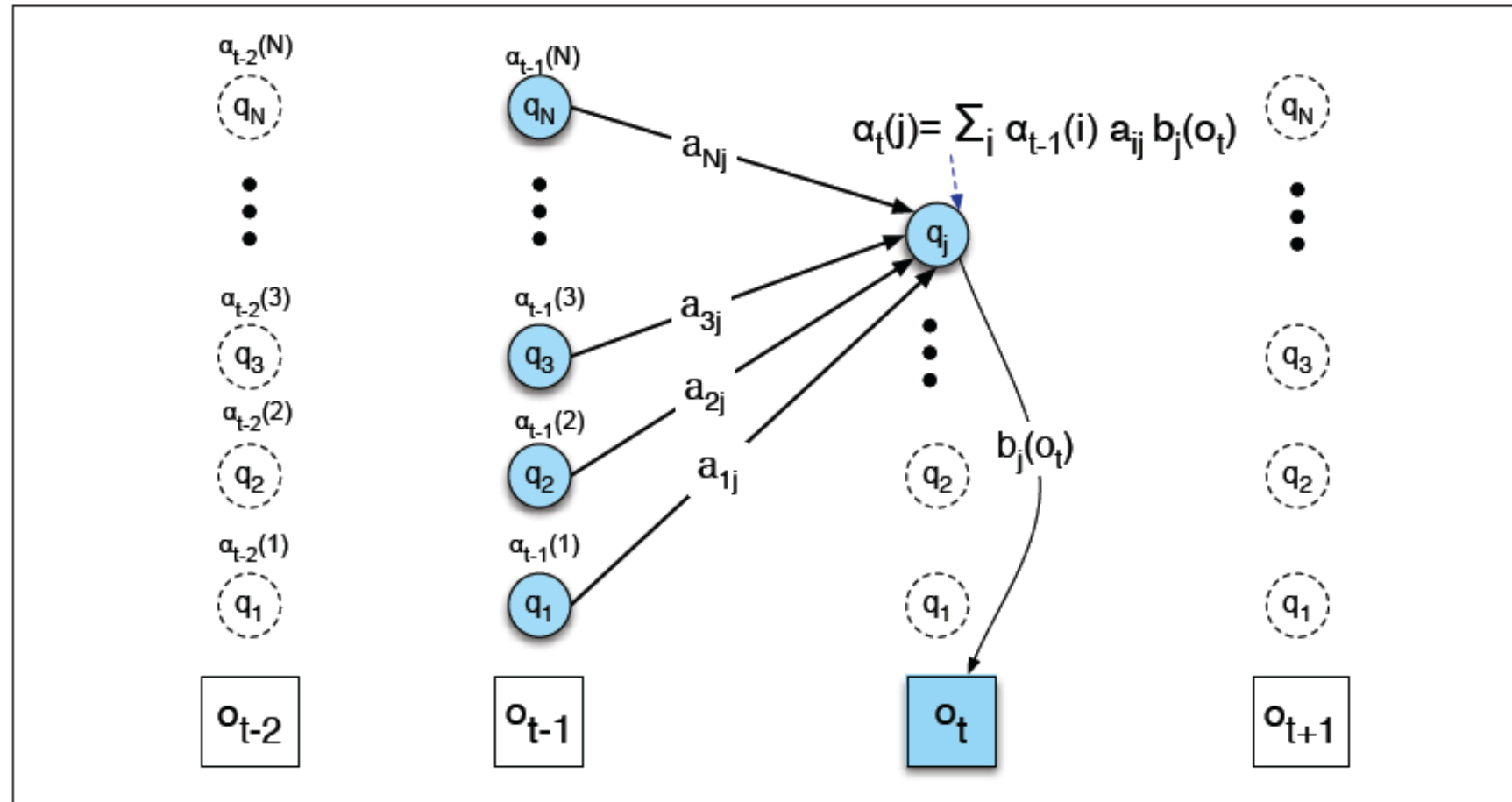
$$\alpha_t(j) = \sum_{i=1}^N \alpha_{t-1}(i) a_{ij} b_j(o_t)$$



$$\alpha_t(j) = P(o_1, o_2, \dots, o_t, q_t = j | \lambda)$$



**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:  $\alpha_t(j) = P(o_1, o_2 \dots o_t, q_t = j | \lambda)$ .



**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**

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

$$forwardprob \leftarrow \sum_{s=1}^N forward[s, T]$$

; termination step

**return** *forwardprob*

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

# Outline

- Motivation/Examples
- HMM: Basic Definition & Three Problems
- Problem 1: Likelihood
- Problem 2: Decoding
- Problem 3: Learning
  - Supervised
  - Unsupervised: EM

# Decoding (Viterbi Algorithm): very similar to Likelihood Computation (Forward Algo.)

- Likelihood: Each cell in trellis represents probability of being in state  $j$  after seeing the first  $t$  observations:

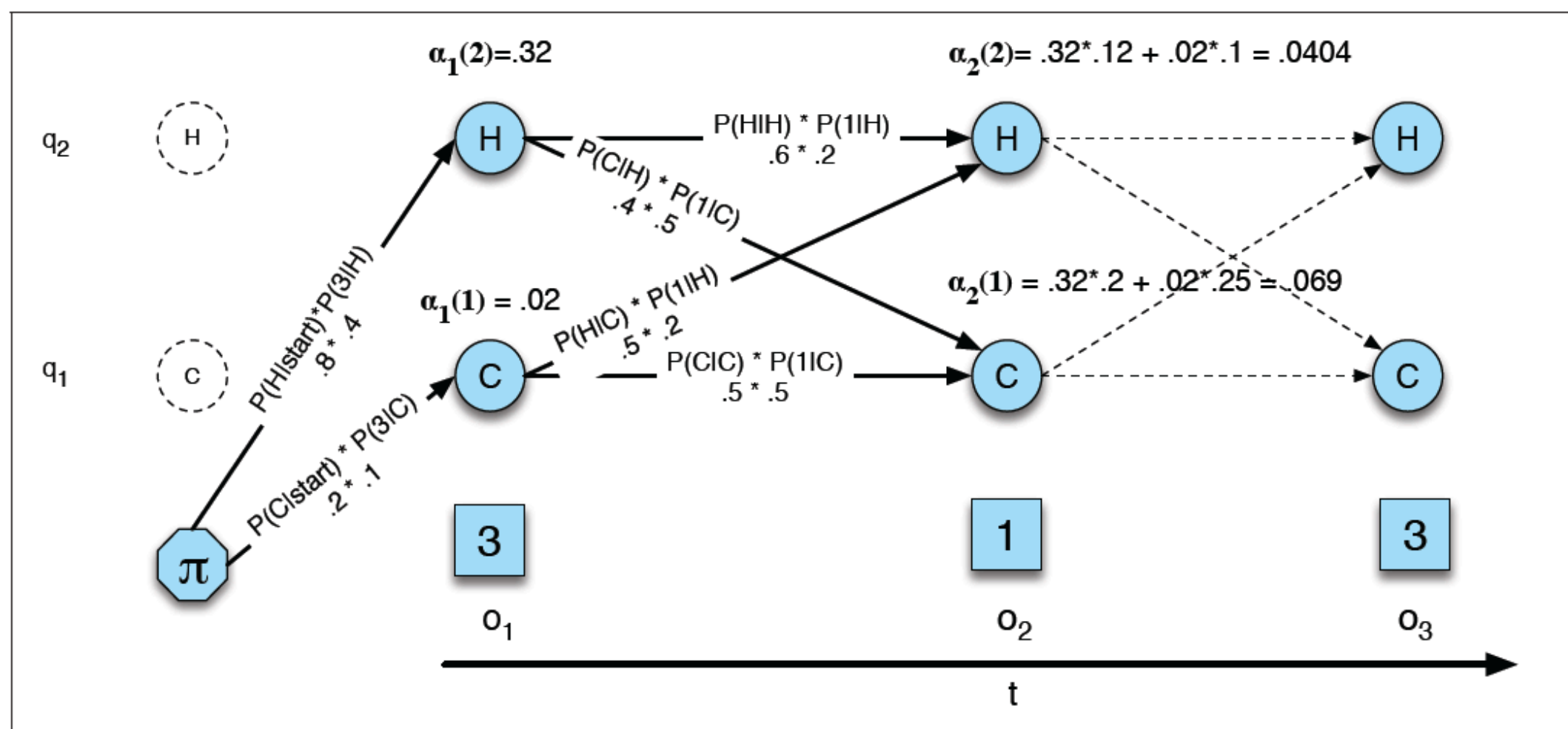
$$\alpha_t(j) = P(o_1, o_2, \dots, o_t, q_t = j | \lambda)$$

- Decoding: Each cell in trellis represents probability of being in state  $j$  after seeing the first  $t$  observations *and passing through the most probable state sequence  $q_1..q_{t-1}$*

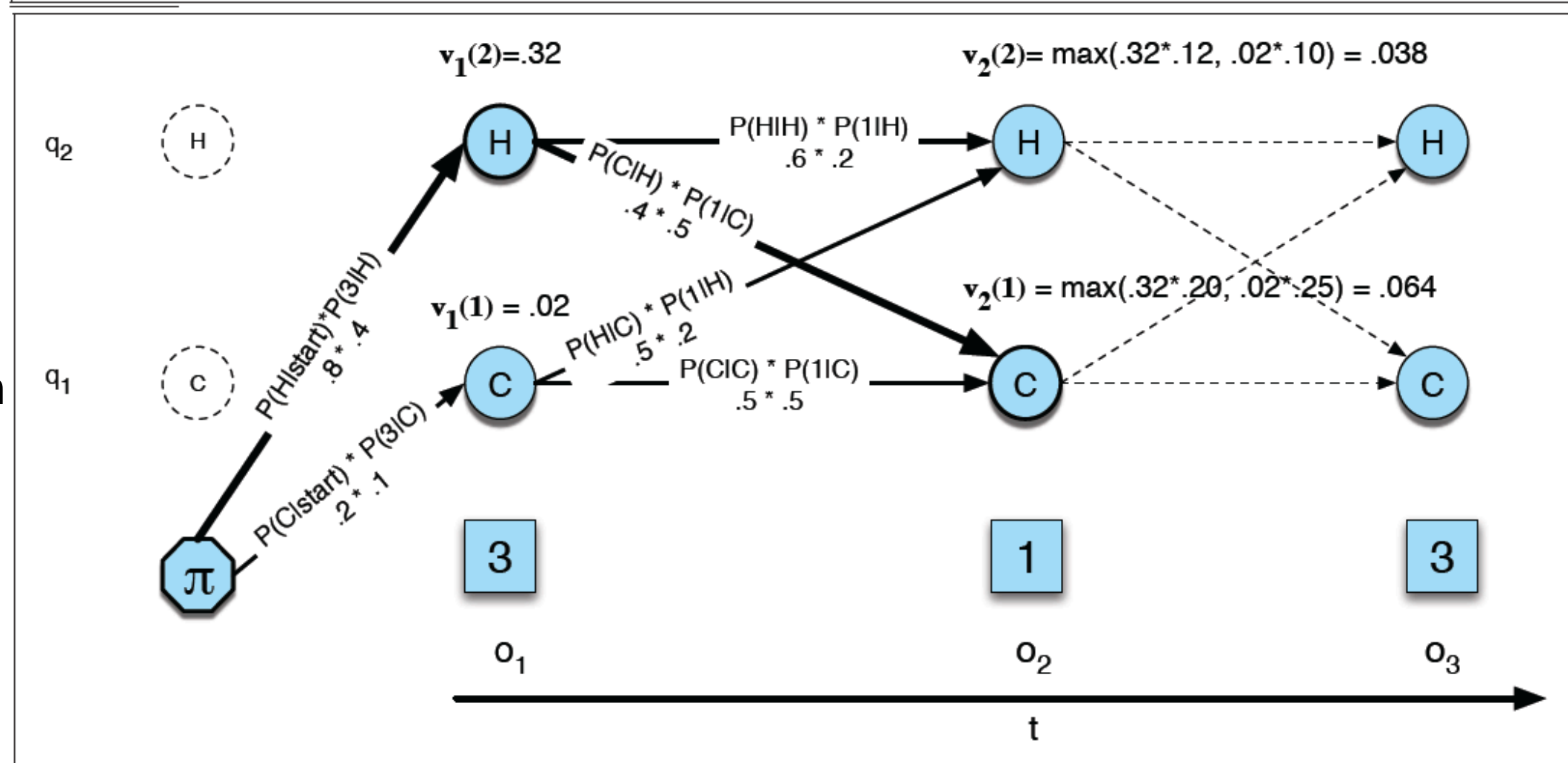
$$v_t(j) = \max_{q_1, \dots, q_{t-1}} P(o_1, o_2, \dots, o_t, q_1, \dots, q_{t-1}, q_t = j, | \lambda)$$

- DP subproblem recursion:  $v_t(j) = \max_{i=1, \dots, N} v_{t-1}(i) a_{ij} b_j(o_t)$

**Forward Algo:**  
Computes  
likelihood of  
observation

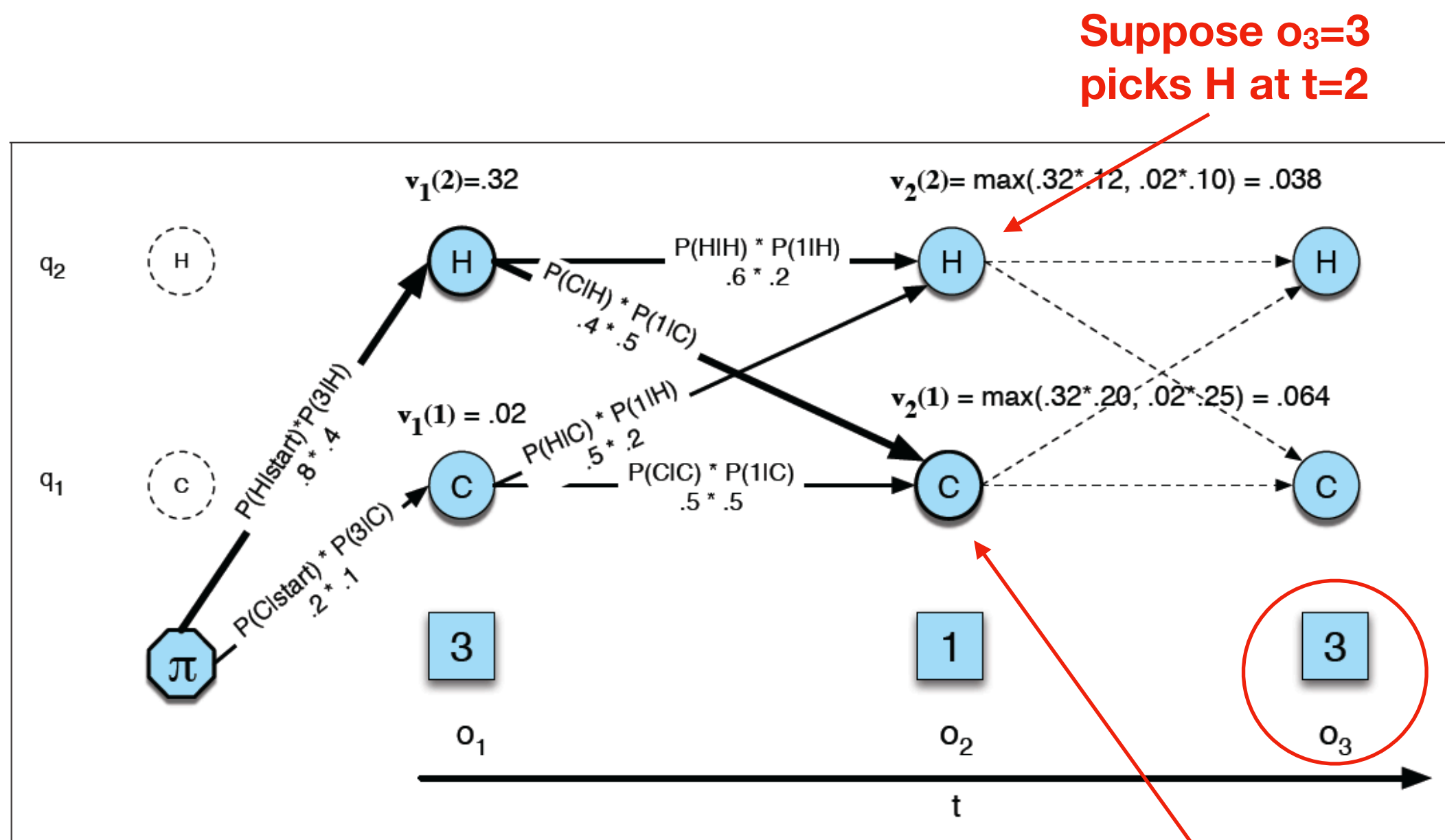


**Viterbi Algo:**  
Decode most  
probable states  
given observation



Note: we're just  
changing sum to max.  
In later lectures we'll  
discuss general  
structures & semirings

**Note: even though the HMM equation with  $P(q_t|q_{t-1})$  seems unidirectional, it actually models the whole sequence and both future/past observations matter**



**If  $o_3=1$  it's possible there may be a different best path at  $t=2$**



**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**  $bestpath$ ,  $bestpathprob$

# Outline

- Motivation/Examples
- HMM: Basic Definition & Three Problems
- Problem 1: Likelihood
- Problem 2: Decoding
- Problem 3: Learning
  - Supervised
  - Unsupervised: EM



# Supervised Learning

- Suppose we're given a dataset with both O and Q.

3	3	2	1	1	2	1	2	3
hot	hot	cold	cold	cold	cold	cold	hot	hot

- It's easy to estimate HMM parameters by counting:

e.g.

$$P(\text{hot}) = \text{Cnt}(\text{hot})/9$$

$$P(3|\text{hot}) = \text{Cnt}(3, \text{hot})/\text{Cnt}(\text{hot})$$

**Initial Probability:**

$$\pi_h = 1/3$$

$$\pi_c = 2/3$$

**Emission Probability (B):**

$$P(1|\text{hot}) = 0/4 = 0 \quad p(1|\text{cold}) = 3/5 = .6$$

$$P(2|\text{hot}) = 1/4 = .25 \quad p(2|\text{cold}) = 2/5 = .4$$

$$P(3|\text{hot}) = 3/4 = .75 \quad p(3|\text{cold}) = 0$$

**Transition Probability (A):**

$$p(\text{hot}|\text{hot}) = 2/3 \quad p(\text{cold}|\text{hot}) = 1/3$$

$$p(\text{cold}|\text{cold}) = 2/3 \quad p(\text{hot}|\text{cold}) = 1/3$$

# Unsupervised Learning

- We're only given observations  $O$ .

3    3    2                    1    1    2                    1    2    3

- How to estimate HMM parameters? Don't know hidden states associated with observation, can't get counts.
- The Baum-Welch / Forward-Backward Algorithm does this by iteratively estimating the counts, and using this to re-derive initial/transition/emission probabilities.
- This is an instance of Expectation-Maximization (EM) Algo.

- Forward Probability

$$\alpha_t(j) = P(o_1, o_2, \dots, o_t, q_t = j | \lambda)$$

- Backward Probability

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, \dots, o_T | q_t = i, \lambda)$$

- Recursion for Backward Probability

$$\beta_t(i) = \sum_{j=1}^N a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \quad P(O | \lambda) = \sum_{j=1}^N \pi_j b_j(o_1) \beta_1(j)$$

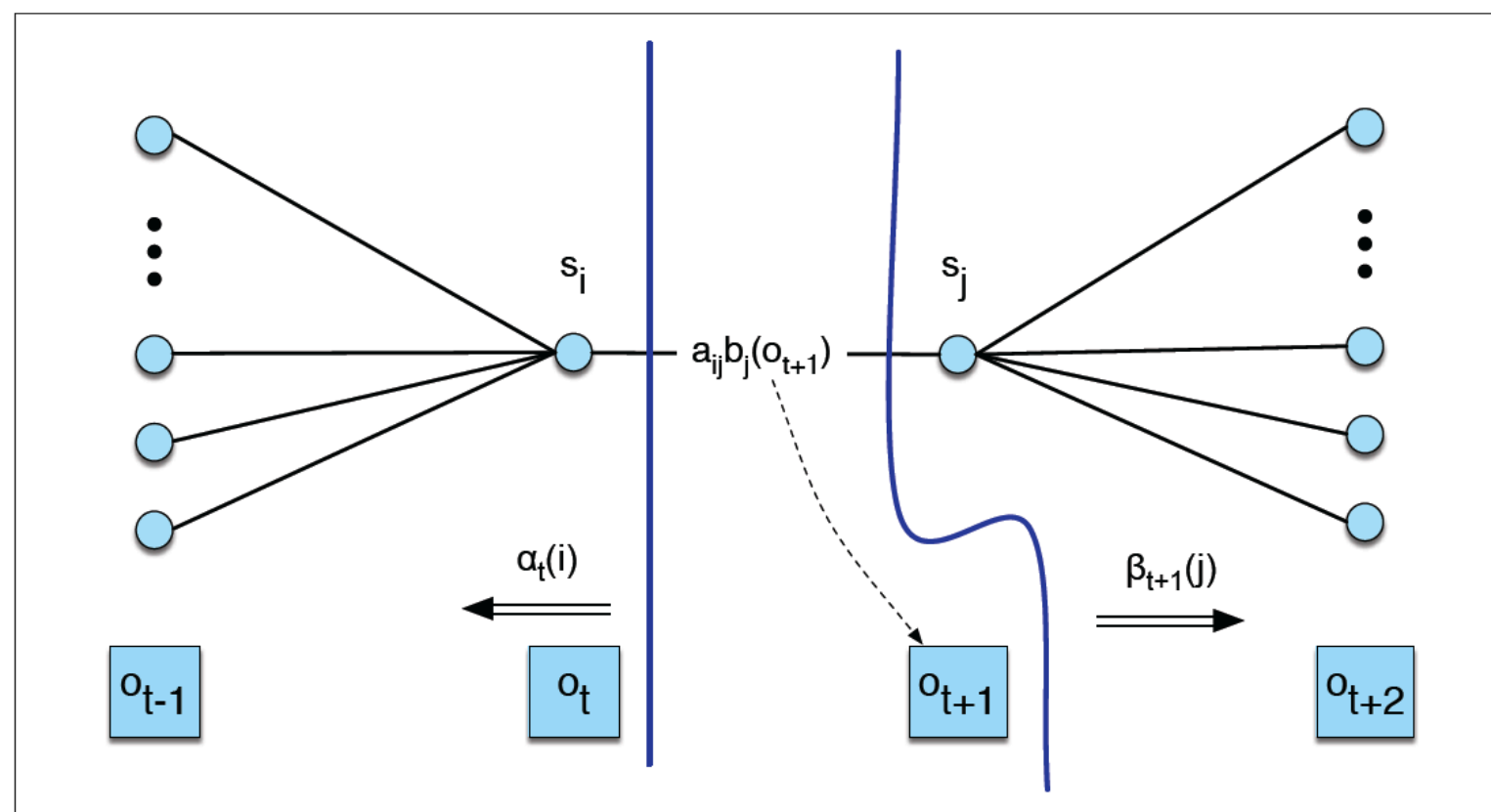
$$\hat{a}_{ij} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i}$$

$$P(O|\lambda) = \sum_{j=1}^N \alpha_t(j) \beta_t(j)$$

$$\xi_t(i, j) = P(q_t = i, q_{t+1} = j | O, \lambda) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{j=1}^N \alpha_t(j) \beta_t(j)}$$

$$\text{not-quite-}\xi_t(i, j) = P(q_t = i, q_{t+1} = j, O | \lambda) = \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)$$

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \sum_{k=1}^N \xi_t(i, k)}$$

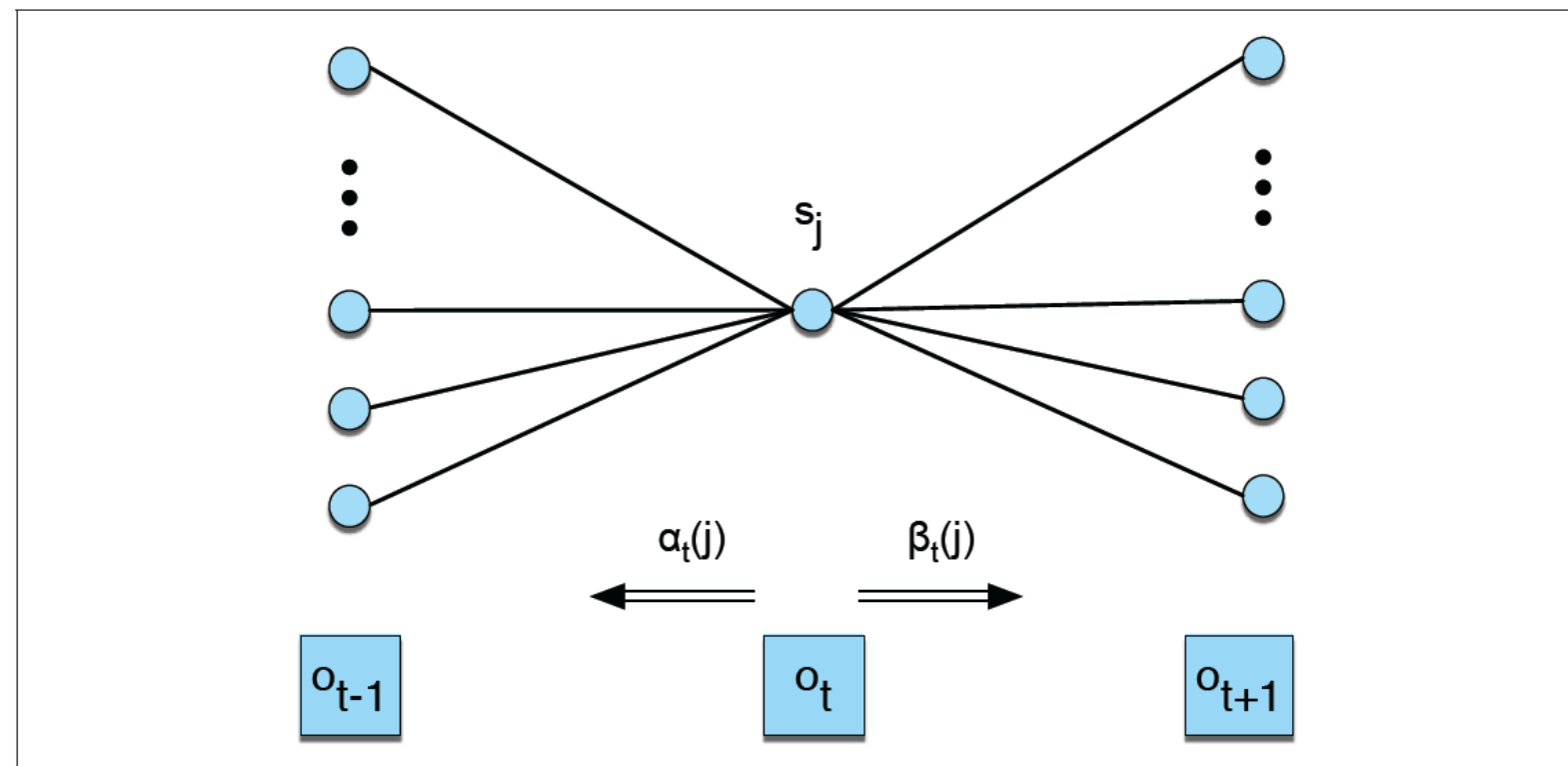


**Figure A.12** Computation of the joint probability of being in state  $i$  at time  $t$  and state  $j$  at time  $t+1$ . The figure shows the various probabilities that need to be combined to produce  $P(q_t = i, q_{t+1} = j, O | \lambda)$ : the  $\alpha$  and  $\beta$  probabilities, the transition probability  $a_{ij}$  and the observation probability  $b_j(o_{t+1})$ . After Rabiner (1989) which is ©1989 IEEE.

$$\hat{b}_j(v_k) = \frac{\text{expected number of times in state } j \text{ and observing symbol } v_k}{\text{expected number of times in state } j}$$

$$\gamma_t(j) = P(q_t = j | O, \lambda) = \frac{P(q_t = j, O | \lambda)}{P(O | \lambda)} = \frac{\alpha_t(j) \beta_t(j)}{P(O | \lambda)}$$

$$\hat{b}_j(v_k) = \frac{\sum_{t=1}^T \text{s.t. } O_t = v_k \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)}$$



**Figure A.13** The computation of  $\gamma_t(j)$ , the probability of being in state  $j$  at time  $t$ . Note that  $\gamma$  is really a degenerate case of  $\xi$  and hence this figure is like a version of Fig. A.12 with state  $i$  collapsed with state  $j$ . After Rabiner (1989) which is ©1989 IEEE.

**function** FORWARD-BACKWARD(*observations* of len  $T$ , *output vocabulary*  $V$ , *hidden state set*  $Q$ ) **returns**  $HMM=(A,B)$

**initialize**  $A$  and  $B$

**iterate** until convergence

**E-step**

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{\alpha_T(q_F)} \quad \forall t \text{ and } j$$

$$\xi_t(i, j) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\alpha_T(q_F)} \quad \forall t, i, \text{ and } j$$

**M-step**

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \sum_{k=1}^N \xi_t(i, k)}$$
$$\hat{b}_j(v_k) = \frac{\sum_{t=1, s.t. O_t=v_k}^T \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)}$$

**return**  $A, B$

**Notes:**

- Each iteration increases data likelihood
- In practice, initialization may be important

**Figure A.14** The forward-backward algorithm.

# Summary

- Motivation/Examples
- HMM: Basic Definition & Three Problems
- Problem 1: Likelihood
- Problem 2: Decoding
- Problem 3: Learning
  - Supervised
  - Unsupervised: EM