In the name of God
the Compassionate, the Merciful

Hidden Markov Models

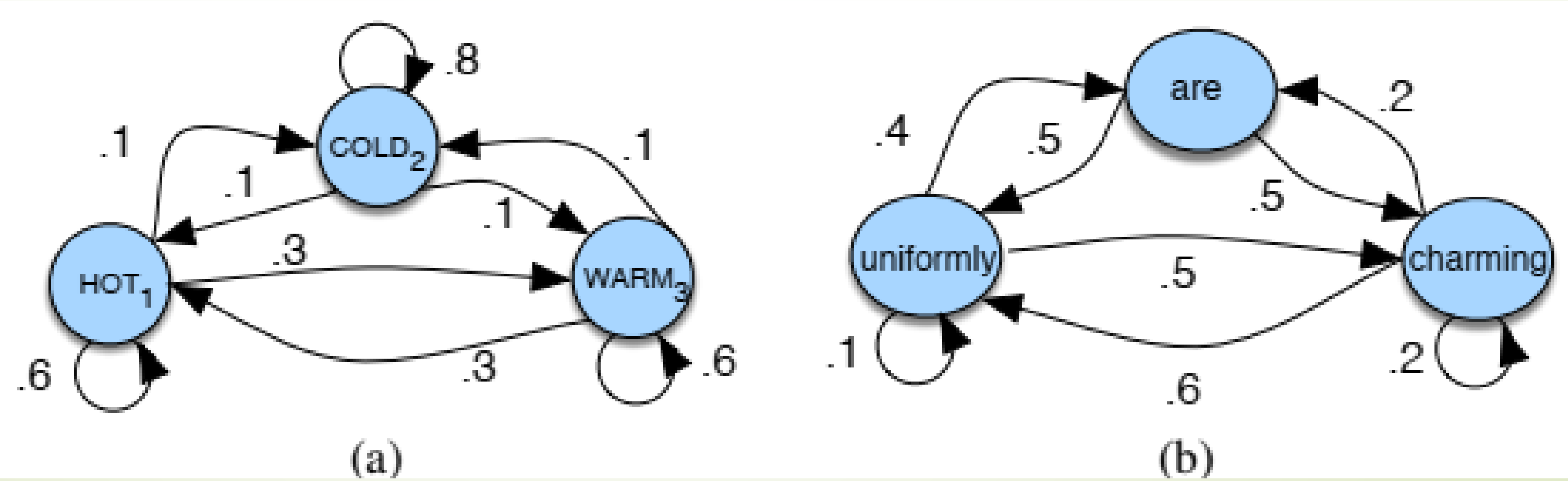
Basics and Algorithms

Roadmap

- Basics of Markov Chains
- Algorithms
  - 1 - Likelihood
  - 2 - Decoding
  - 3 - Learning

Basics of Markov Chains

- The probabilities of sequences of random variables, *states*, each of which can take on values from some set
- A Markov chain makes a very strong assumption that if we want to predict the future in the sequence, all that matters is the current state



A start distribution π is required; setting π = [0:1; 0:7; 0:2]

Continue..

- Markov Assumption $P(q_i = a|q_1 \ldots q_{i-1}) = P(q_i = a|q_{i-1})$
- The transitions are probabilities: the values of arcs leaving a given state must sum to 1
- This Markov chain should be familiar; in fact, it represents a bigram language model
- Specification:

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} a_{12} \ldots a_{N1} \ldots a_{NN}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$\pi = \pi_1, \pi_2, \ldots, \pi_N$	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, $\sum_{i=1}^{N} \pi_i = 1$

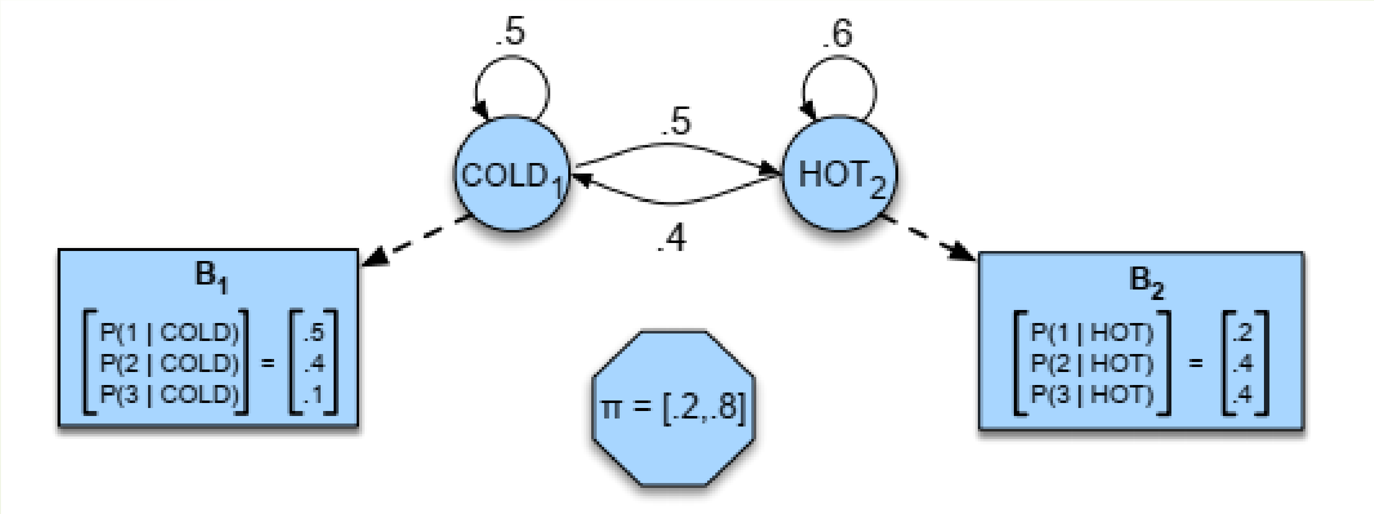
The Hidden Markov Model

- A hidden Markov model (HMM) allows us to talk about both observed events (like words that we see in the input) and hidden events (like part-of-speech tags) that we think of as causal factors in our probabilistic model
- Specification

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} \ldots a_{ij} \ldots a_{NN}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{N} a_{ij} = 1 \quad \forall i$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t (drawn from a vocabulary $V = v_1, v_2, \ldots, v_V$) being generated from a state q_i
$\pi = \pi_1, \pi_2, \ldots, \pi_N$	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, $\sum_{i=1}^{n} \pi_i = 1$

HMM

- Two simplifying assumptions
  - Markov Assumption: $$P(q_i|q_1 \ldots q_{i-1}) = P(q_i|q_{i-1})$$
  - Output Independence: $$P(o_i|q_1 \ldots q_i, \ldots, q_T, o_1, \ldots, o_i, \ldots, o_T) = P(o_i|q_i)$$
- Example:



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.

Likelihood Computation: The Forward Algorithm

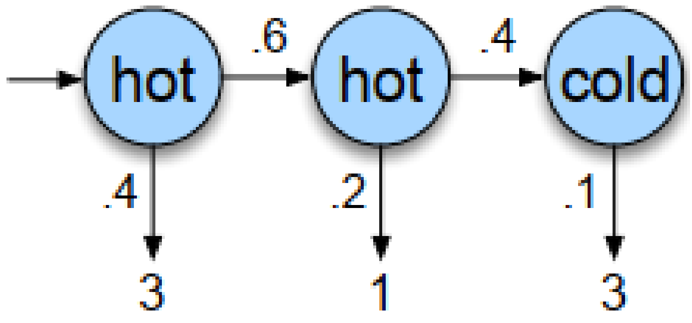
- Computing Likelihood: Given an HMM λ = (A;B) and an observation sequence O, determine the **likelihood** $P(O \mid \lambda)$
- First, recall that for hidden Markov models, each hidden state produces only a single observation. Thus, the sequence of hidden states and the sequence of observations have **the same length**.
- For a particular hidden state sequence $Q = q_0;q1;q2;\ldots q_T$ and an observation sequence $O = o_1;o_2;\ldots;o_T$, the likelihood of the observation sequence

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

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

Example

$$
\begin{aligned}
P(3\ 1\ 3, \text{hot hot cold}) &= P(\text{hot}|\text{start}) \times P(\text{hot}|\text{hot}) \times P(\text{cold}|\text{hot}) \\
&\times P(3|\text{hot}) \times P(1|\text{hot}) \times P(3|\text{cold})
\end{aligned} \tag{A.9}
$$



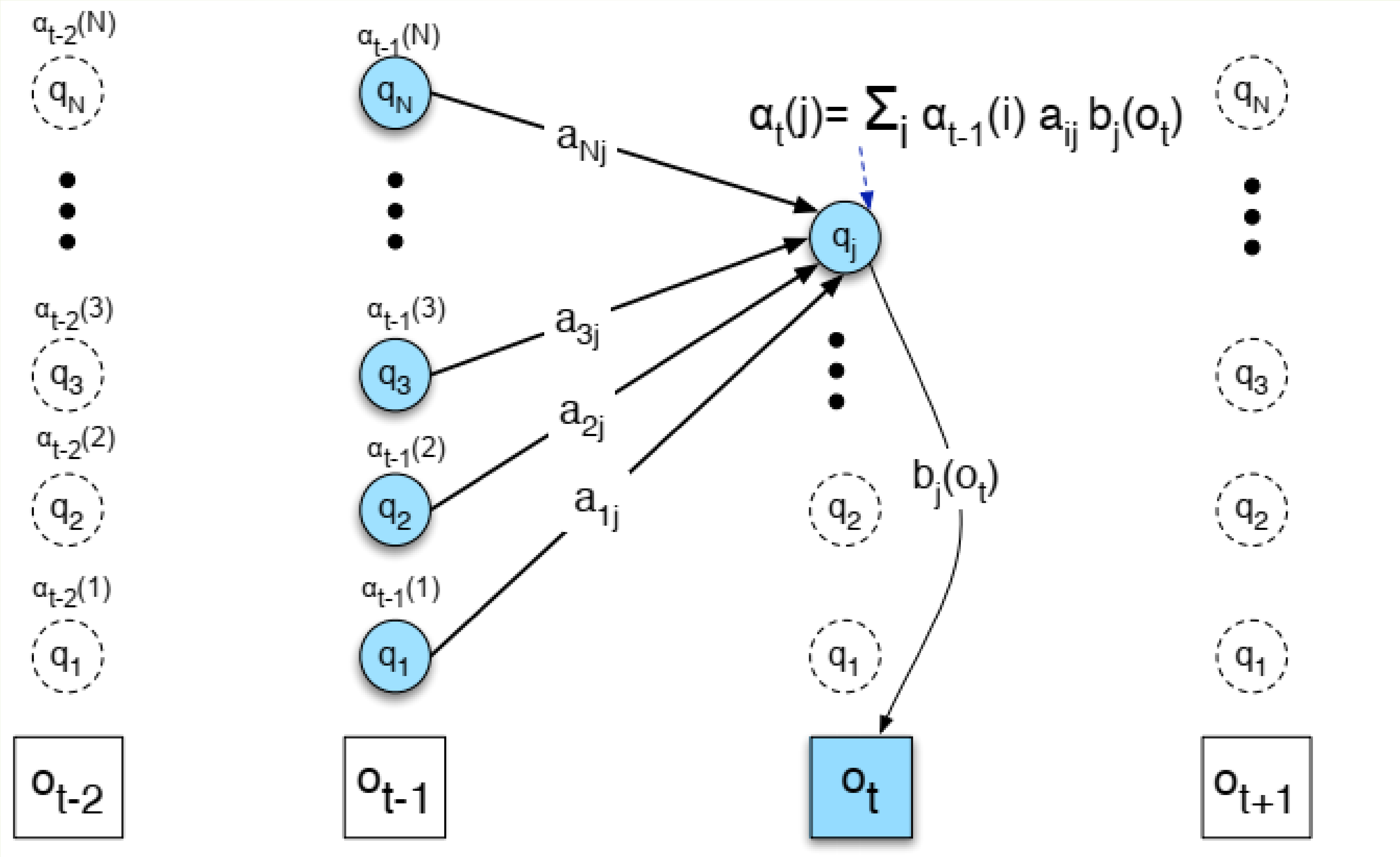
Continued..

- For an HMM with N hidden states and an observation sequence of T observations, there are N^T possible hidden sequences.
- Forward Algorithm
  - N^2T
  - Dynamic Programming

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

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



Hidden states are in circles, observations in squares.

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*

1. Initialization:

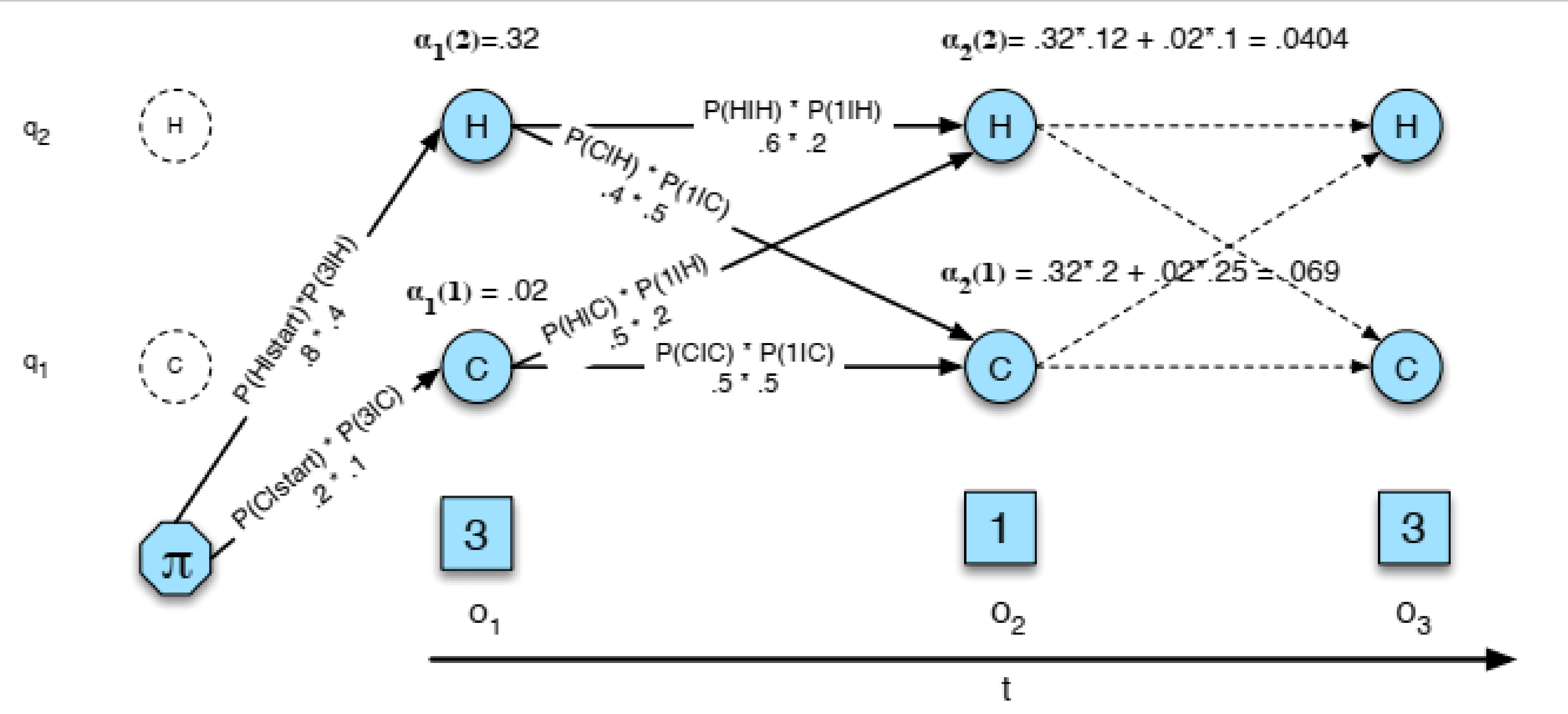
$$\alpha_1(j) \;=\; \pi_j b_j(o_1) \;\; 1 \le j \le N$$

2. Recursion:

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

3. Termination:

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



Hidden states are in circles, observations in squares.

Decoding: The Viterbi Algorithm

- **Decoding**: Given as input an HMM $\lambda = (A;B)$ and a sequence of observations $O = o_1;o2;\ldots;o_T$, **find the most probable sequence of states** $Q = q_1q2q3 \ldots q_T$
- *A naïve method:* 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
- A better method: Viterbi
  - Dynamic Programming
  - Similar to “minimum edit distance”

Recursive Formula

$$v_t(j) = \max_{1 \leq i \leq N-1} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

$v_{t-1}(i)$	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 ← the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time

return *bestpath*, *bestpathprob*

Note that..

- Note that the Viterbi algorithm is identical to the forward algorithm except that it takes the **max** over the previous path probabilities whereas the forward algorithm takes the **sum**.
  - Backpointers (best path to the beginning: backtrace)

1. **Initialization:**

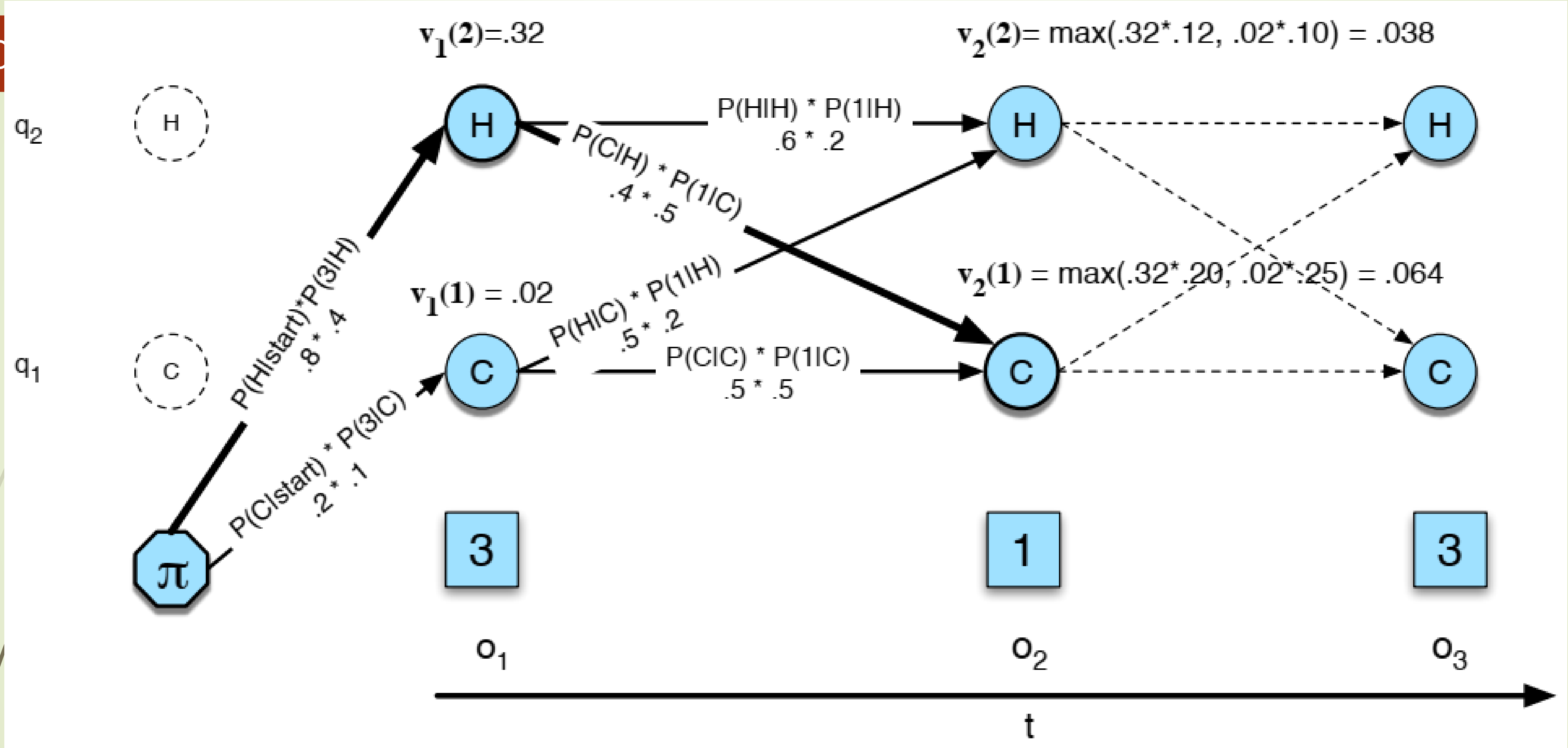
$$v_1(j) = \pi_j b_j(o_1) \qquad 1 \le j \le N$$
$$bt_1(j) = 0 \qquad 1 \le j \le N$$

2. **Recursion**

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t); \quad 1 \le j \le N, 1 < t \le T$$
$$bt_t(j) = \operatorname*{argmax}_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t); \quad 1 \le j \le N, 1 < t \le$$

3. **Termination:**

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



Hidden states are in circles, observations in squares.

HMM Training: The Forward-Backward Algorithm

- **Learning:** Given an observation sequence *O* and the set of possible states in the HMM, learn the HMM parameters *A* and *B*.
- **Forward-backward**, or **BaumBaum-Welch Welch** algorithm
  - A special case of the **Expectation-Maximization or EM** algorithm
  - EM is an *iterative* algorithm, computing an initial estimate for the probabilities, then using those estimates to computing a better estimate, and so on
  - Compute the HMM parameters just by maximum likelihood estimation from the training data.

Foreword and backward probability

- The **backward** probability β is the probability of seeing the observations from time t +1 to the end, given that we are in state i at time t (and given the automaton λ): $\beta_t(i) = P(o_{t+1};o_{t+2} \ldots o_T \mid q_t = i;\lambda)$
- We also use **forward** probability α $(P(O \mid \lambda))$

Backward..

1. **Initialization:**

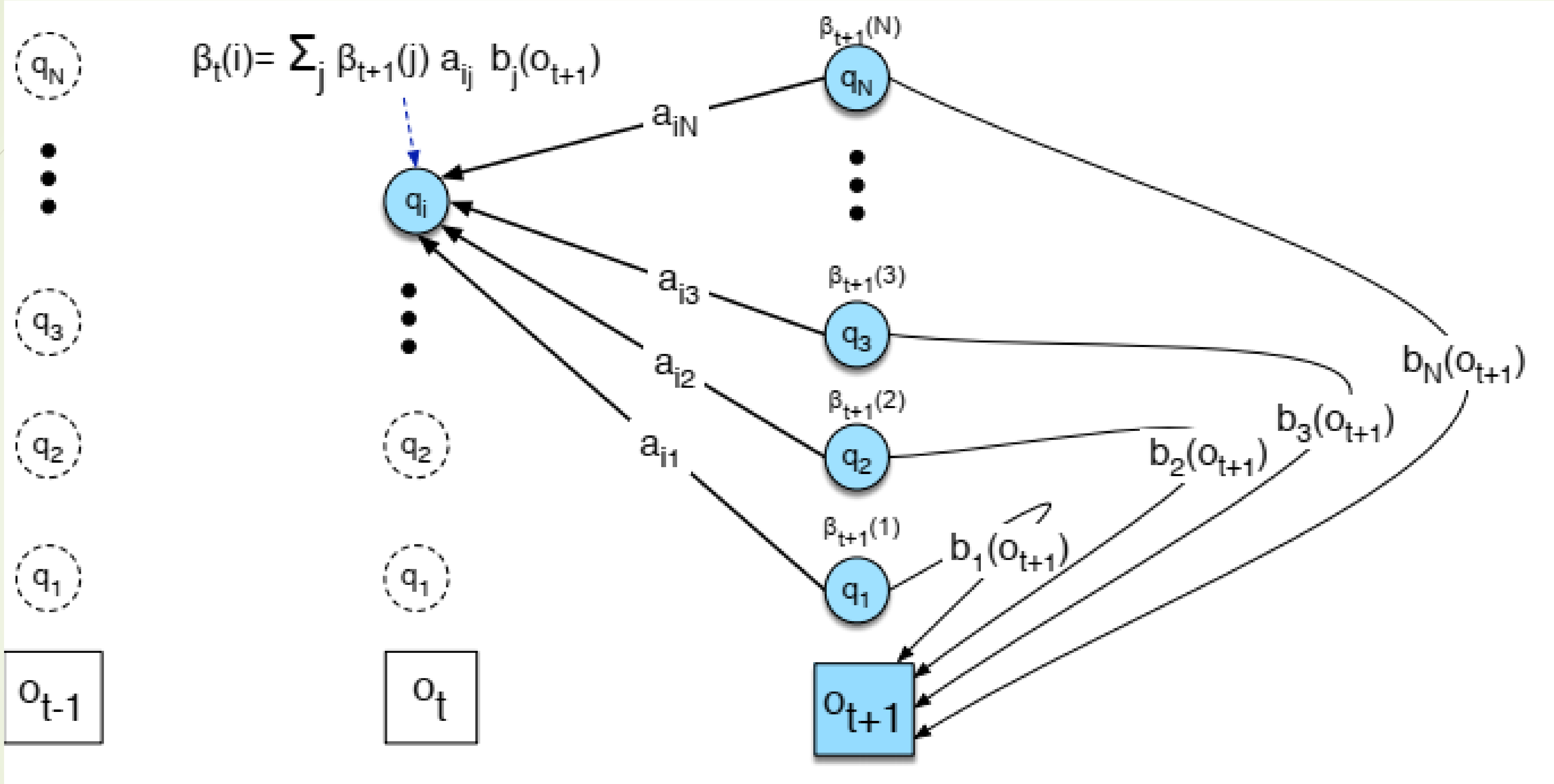
$$\beta_T(i) = 1, \quad 1 \leq i \leq N$$

2. **Recursion**

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij}\, b_j(o_{t+1})\, \beta_{t+1}(j), \quad 1 \leq i \leq N, 1 \leq t < T$$

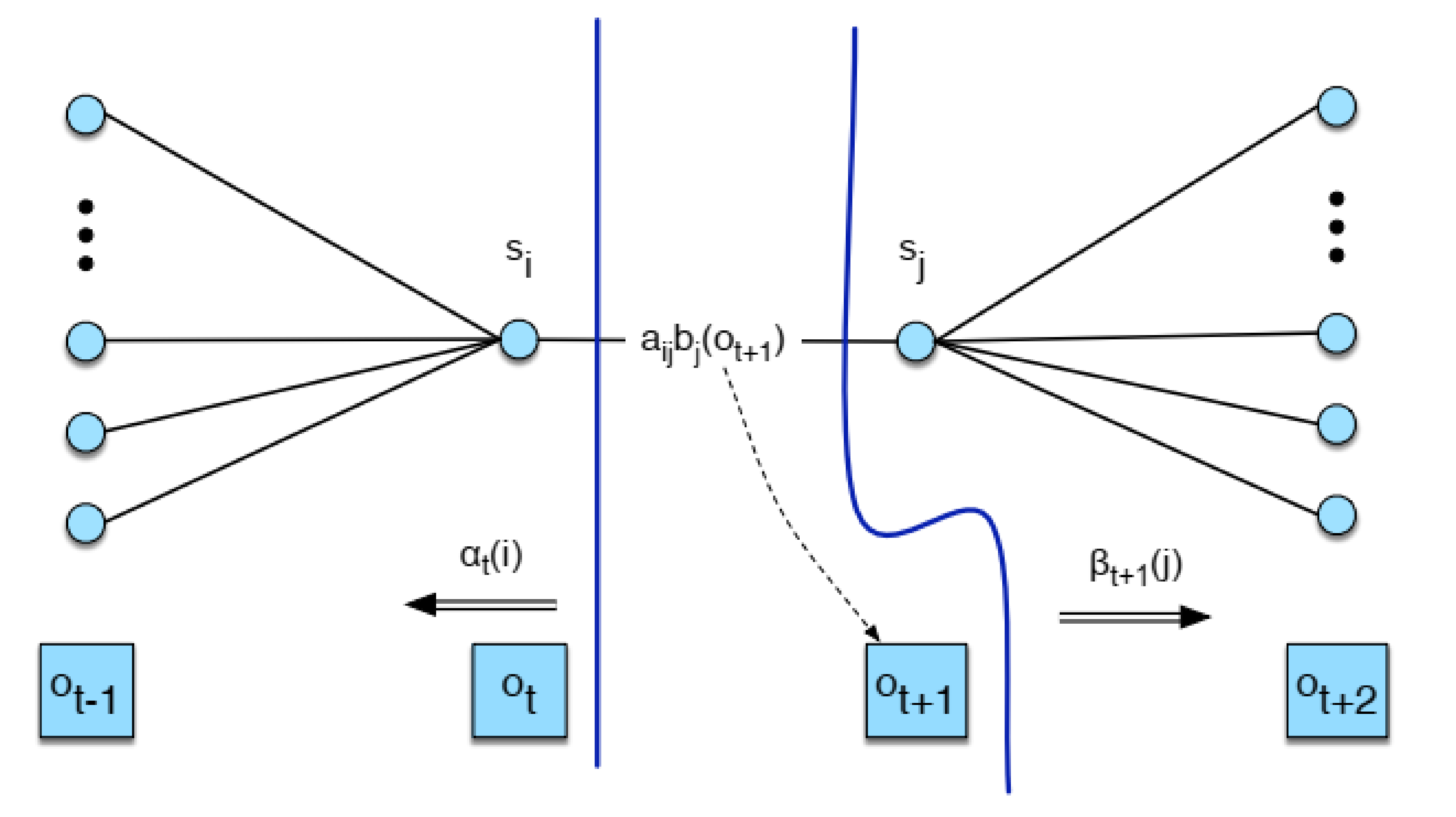
3. **Termination:**

$$P(O|\lambda) = \sum_{j=1}^{N} \pi_j\, b_j(o_1)\, \beta_1(j)$$



the transition probability a_{ij} and observation probability $b_i(o_t)$

$P(q_t=i, q_{t+1}=j, O \mid \lambda)$



Auxiliary Variables

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

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

$$P(X|Y,Z) = \frac{P(X,Y|Z)}{P(Y|Z)}$$

Computation

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

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

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

Expectation Maximization..

- In the E-step, we compute the expected state occupancy count γ and the expected state transition count ξ from the earlier A and B probabilities.
- In the M-step, we use γ and ξ to recompute new A and B probabilities.

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

Conclusion

- We introduced Markov property and Markov Chains
- Next, we introduced hidden Markov models
- Next we talked about three problems and their solutions
  - Likelihood: Foreword algorithm
  - Decoding: Viterbi algorithm
  - Learning: Foreword-backward algorithm


THE
END