Hidden Markov Models

Notes of https://web.stanford.edu/~jurafsky/slp3/A.pdf

Xiachong Feng

Outline

- Markov Chains
- The Hidden Markov Model
- **Likelihood Computation: The Forward Algorithm**
- **Decoding: The Viterbi Algorithm**
- **HMM Training: The Forward-Backward Algorithm**
- Summary

Markov Chains

- **What is Markov Chains？**
  - A **Markov chain** is a model that tells us something about the **probabilities of sequences of random variables**, *states*, each of which can take on values from some set.

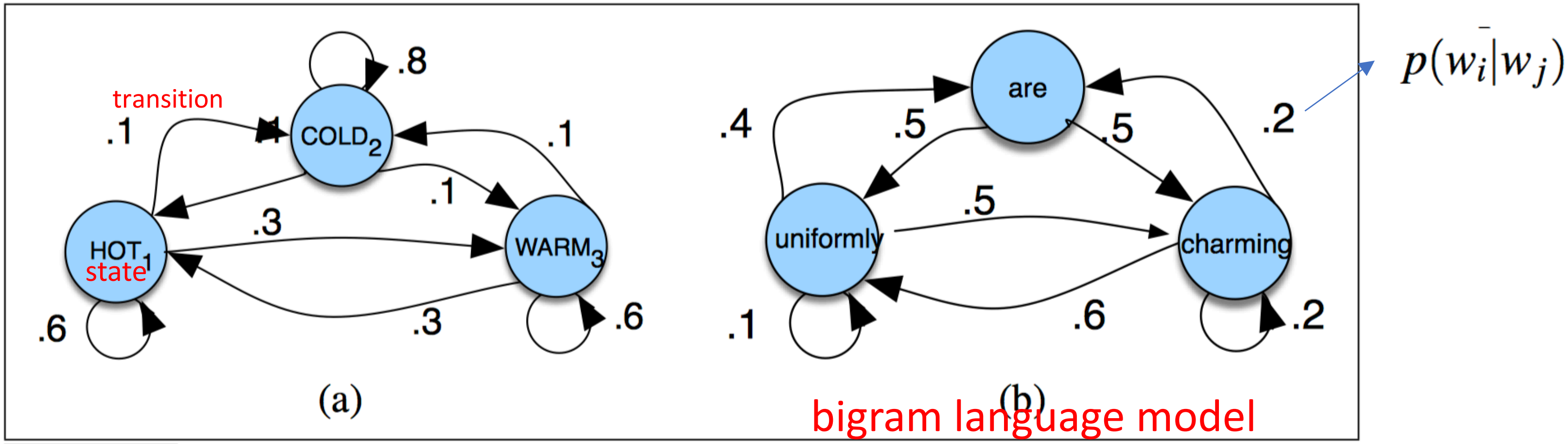

Figure A.1 A Markov chain for weather (a) and one for words (b), showing states and transitions. A start distribution π is required; setting $\pi = [0.1, 0.7, 0.2]$ for (a) would mean a probability 0.7 of starting in state 2 (cold), probability 0.1 of starting in state 1 (hot), etc.

Markov Chains

- **Markov Assumption**
  - when predicting the future, the past doesn't matter, **only the present**.

$$\boxed{\textbf{Markov Assumption:} \quad P(q_i = a | q_1 ... q_{i-1}) = P(q_i = a | q_{i-1})}$$

q: state variables

- **Components**

$Q = q_1 q_2 \ldots q_N$	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. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$\pi = \pi_1, \pi_2, ..., \pi_N$	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

- **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.
- **Components**

$Q = q_1 q_2 \ldots q_N$	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. $\sum_{j=1}^{N} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \ldots o_T$	a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, \ldots, v_V$
$B = b_i(o_t)$ P(Si->Ot)	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state i
$\pi = \pi_1, \pi_2, \ldots, \pi_N$	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

- **First-order hidden Markov model**
  - **Markov Assumption:** the probability of a particular state depends only on the previous state

下一时刻的状态只和当前状态有关

Markov Assumption: $P(q_i|q_1 \dots q_{i-1}) = P(q_i|q_{i-1})$

  - **Output Independence:** the probability of an output observation oi depends only on the state that produced the observation

观测只和生成这个观测的状态有关

Output Independence: $P(o_i|q_1 \dots q_i, \dots, q_T, o_1, \dots, o_i, \dots, o_T) = P(o_i|q_i)$

each hidden state produces only a single observation

Example-Eisner task

- **Task definition**

观测：每天吃冰激凌的数目

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.

天气状态热和冷

- **HMM for the ice cream task**

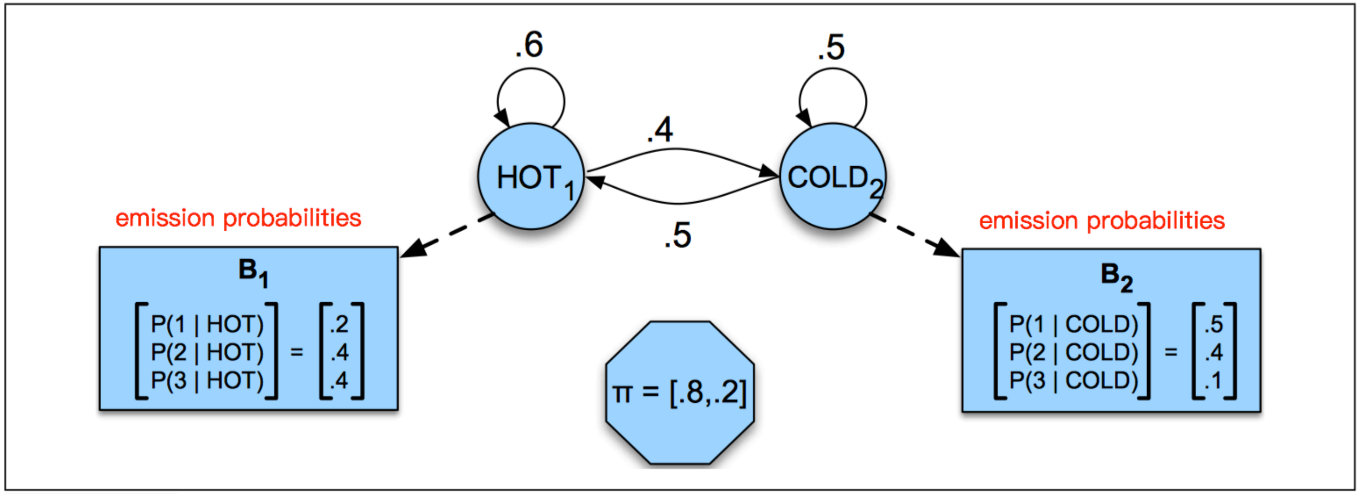

Figure A.2 A hidden Markov model for relating numbers of ice creams eaten by Jason (the observations) to the weather (H or C, the hidden variables).

The Hidden Markov Model

- **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

- **Task definition**

Computing Likelihood: Given an HMM $\lambda = (A, B)$ and an observation sequence O, determine the likelihood $P(O|\lambda)$.

- **Forward algorithm**
  - Dynamic programming algorithm
  - uses a table to store intermediate values
  - **computes the observation probability by summing over the probabilities of <u>all possible hidden state paths that could generate the observation sequence</u>**
  - Use a **single forward trellis**

Likelihood Computation

- **Each cell of the forward algorithm trellis** $\alpha_t(j)$ *For hidden state*
  - **probability** of being in **state j** after seeing the first **t observations**, given the **automaton λ**
  - **summing over** the probabilities of every path that could lead us to this cell.

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

  - For a given state q_j at time t, the value $\alpha_t(j)$ is computed as

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

Considering the current state could generate the Ot

$\alpha_{t-1}(i)$	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 P(Si->Ot)

Likelihood Computation

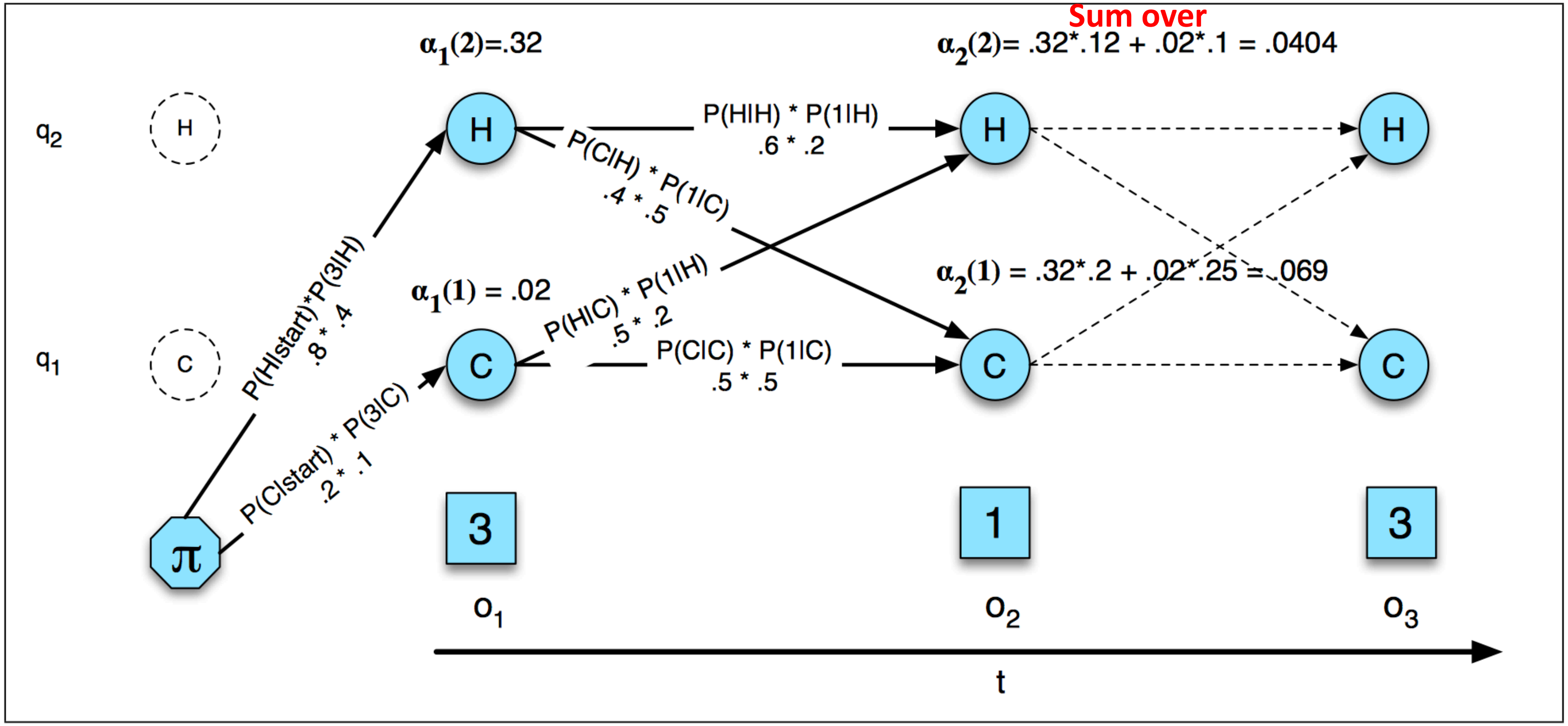

Figure A.5 The forward trellis for computing the total observation likelihood for the ice-cream events *3 1 3*. Hidden states are in circles, observations in squares. The figure shows the computation of $\alpha_t(j)$ for two states at two time steps. The computation in each cell follows Eq. A.12: $\alpha_t(j)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

Likelihood Computation

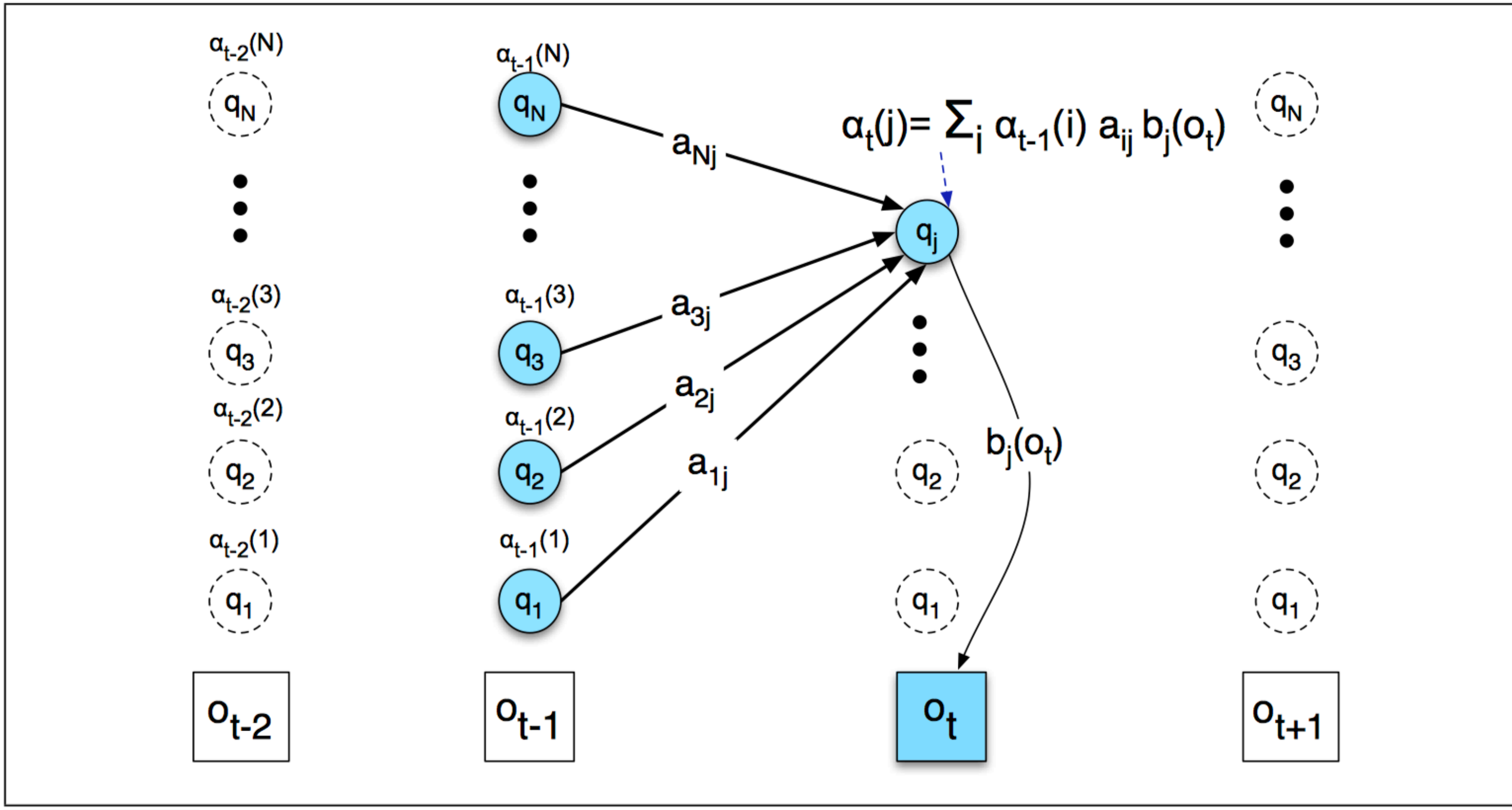

Figure A.6 Visualizing the computation of a single element $\alpha_t(i)$ in the trellis by summing all the previous values α_{t-1}, weighted by their transition probabilities a, and multiplying by the observation probability $b_i(o_{t+1})$. For many applications of HMMs, many of the transition probabilities 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 $\alpha_t(i)$.

Likelihood Computation

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

create a probability matrix *forward[N,T]* N个状态，T个观测

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

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

$forward[s,1] \leftarrow \pi_s \, * \, b_s(o_1)$ 第一列是初始概率*发射概率

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

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

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

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

forward[N,T]已经算好

$forwardprob \leftarrow \sum_{s=1}^{N} forward[s,T]$; termination step *Sum of the last column*

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

return *forwardprob*

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

Decoding: The Viterbi Algorithm

- **Task definition**

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

- **Viterbi algorithm**
  - Dynamic programming algorithm
  - $v_t(j)$
    - represents the probability that the HMM is in state j after seeing the first t observations and **passing through the most probable state sequence q1 ,...,qt−1** , given the automaton λ
    - computed by recursively taking the **most probable path** that could lead us to this cell

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

    - For a given state qj at time t, the value $v_t(j)$ is computed as

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

Forward vs Viterbi algorithm

Likelihood : Forward

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

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

Sum over

summing over the probabilities of every path that could lead us to this cell

Decode : Viterbi

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

*represent the most probable path by taking the **maximum** over all possible <u>previous state sequences</u>*

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

Max

most probable path that could lead to this cell

Note that the Viterbi algorithm is identical to the forward algorithm except that it takes the <u>max over</u> the previous path probabilities whereas the forward algorithm takes the <u>sum</u>.

Decoding: The Viterbi Algorithm

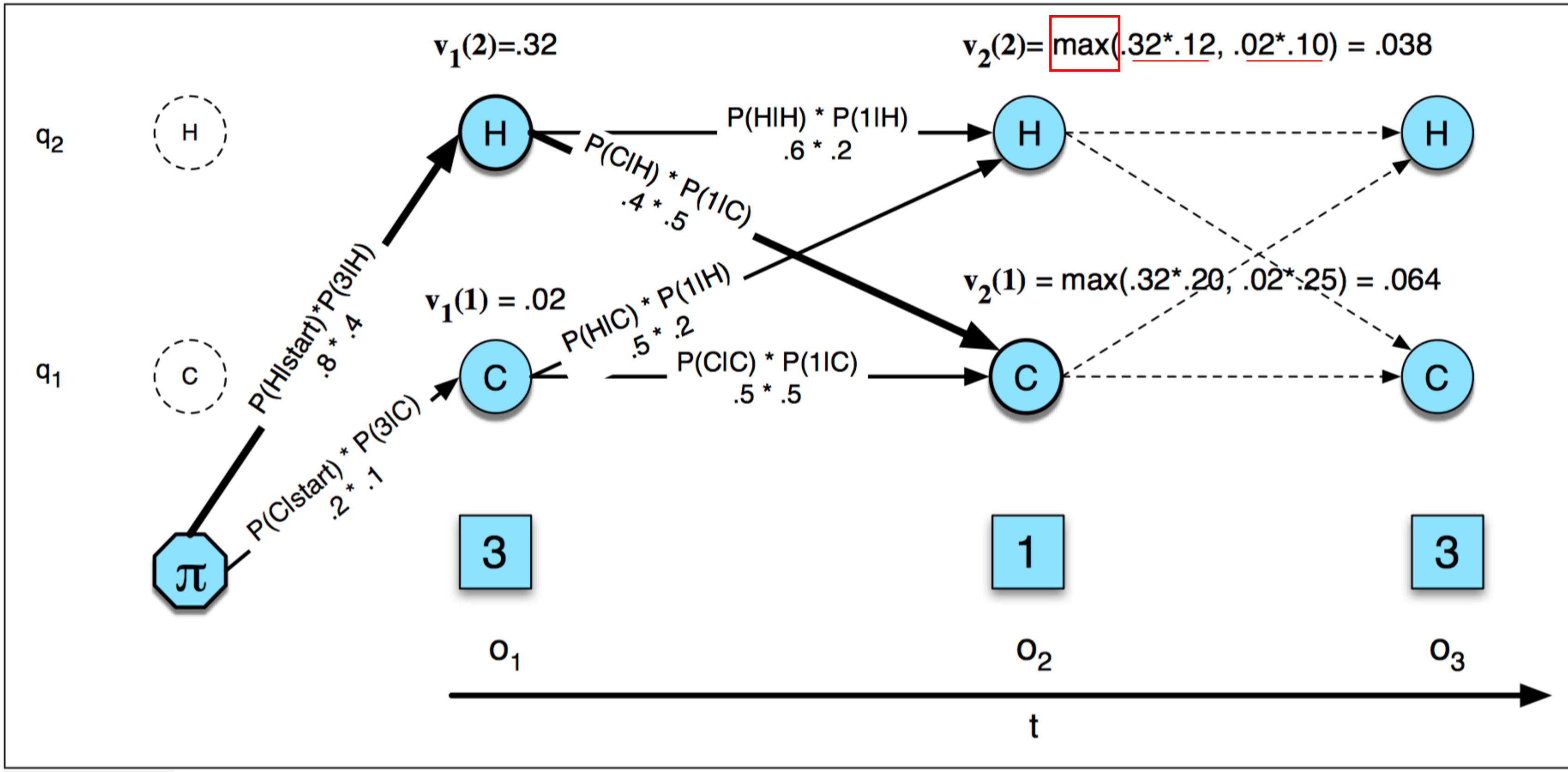

Figure A.8 The Viterbi trellis for computing the best path through the hidden state space for the ice-cream eating events *3 1 3*. Hidden states are in circles, observations in squares. White (unfilled) circles indicate illegal transitions. The figure shows the computation of $v_t(j)$ for two states 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: $v_t(j) = P(q_0, q_1, \ldots, q_{t-1}, o_1, o_2, \ldots, o_t, q_t = j|\lambda)$.

Decoding: The Viterbi Algorithm

- **The Viterbi Backtrace**
  - **Forward algorithm** needs to produce an observation likelihood,
  - **Viterbi algorithm** must produce a probability and **also the most likely state sequence.**
  - keeping track of the path of hidden states that led to each state, and then at the end **backtracing** the best path to the beginning

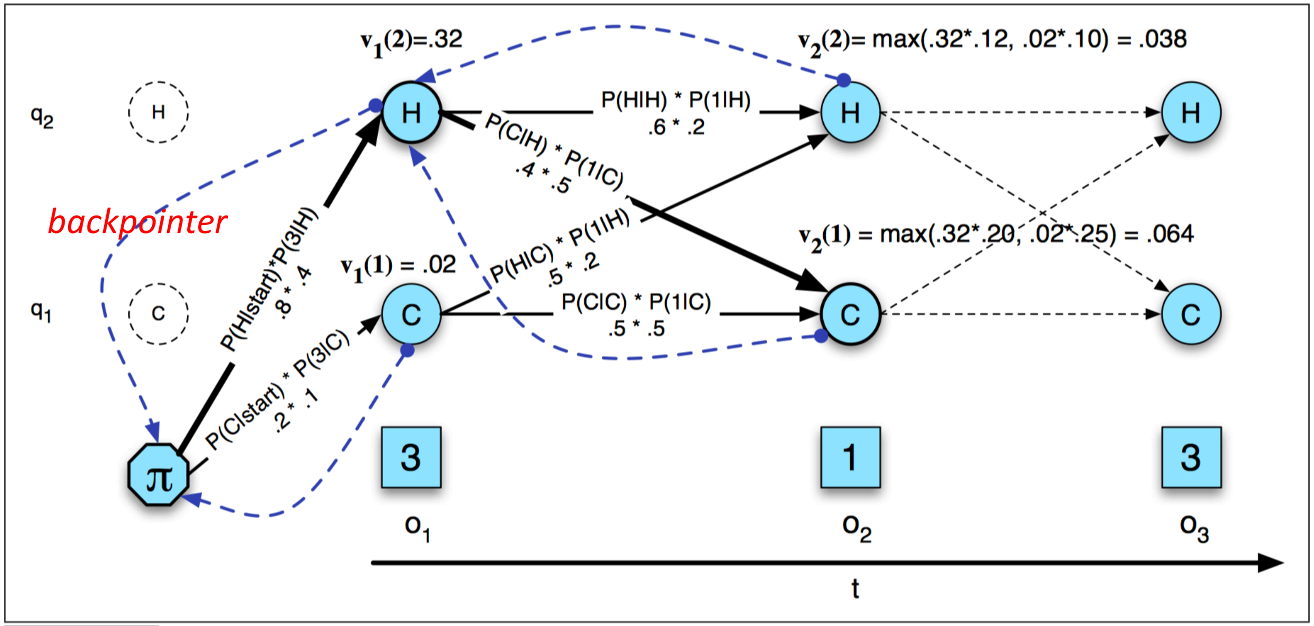

Figure A.10 The Viterbi backtrace. As we extend each path to a new state account for the next observation, we keep a backpointer (shown with broken lines) to the best path that led us to this state.

Decoding: The Viterbi Algorithm

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
$\quad viterbi[s,1] \leftarrow \pi_s * b_s(o_1)$ *init*
$\quad backpointer[s,1] \leftarrow 0$ *point to initial state*
for each time step t **from** 2 **to** T **do** ; recursion step
$\quad$ **for** each state s **from** 1 **to** N **do**
$$viterbi[s,t] \leftarrow \max_{s'=1}^{N} \; viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$
$$backpointer[s,t] \leftarrow \underset{s'=1}{\overset{N}{\operatorname{argmax}}} \; viterbi[s',t-1] * a_{s',s} * b_s(o_t)$$
$$bestpathprob \leftarrow \max_{s=1}^{N} \; viterbi[s,T] \quad \text{; termination step}$$
$$bestpathpointer \leftarrow \underset{s=1}{\overset{N}{\operatorname{argmax}}} \; viterbi[s,T] \quad \text{; termination step}$$
bestpath $\leftarrow$ the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

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

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

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

The start of backtrace: $q_T* = \underset{i=1}{\overset{N}{\operatorname{argmax}}}\, v_T(i)$

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.

HMM Training

- **Task definition**

  Learning: Given an observation sequence O and the set of possible states in the HMM, learn the HMM parameters A and B.

- **Input**
  - unlabeled sequence of observations O *example ： O = {1,3,2,...,}*
  - a vocabulary of potential hidden states Q *H and C*
- **Forward-backward or Baum-Welch algorithm**
  - a special case of **the Expectation-Maximization** or **EM algorithm**
  - The algorithm will let us train both the **transition probabilities A** and the **emission probabilities B**
  - **EM** is an **iterative algorithm**, computing an initial estimate for the probabilities, then using those estimates to computing a better estimate, and so on, iteratively improving the probabilities that it learns.

HMM Training

- **Fully visible Markov model**
- **We know**
  - Input observations
  - Aligned hidden state sequences *(labeled)*

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

- **Compute the HMM parameters just by maximum likelihood estimation**

$$\pi \qquad \pi_h = 1/3 \qquad \pi_c = 2/3$$

$$A \text{ matrix} \qquad \begin{array}{ll} p(hot|hot) = 2/3 & p(cold|hot) = 1/3 \\ p(cold|cold) = 1/2 & p(hot|cold) = 1/2 \end{array}$$

Maybe error in book

ignoring the final hidden states

$$B \text{ matrix} \qquad \begin{array}{ll} P(1|hot) = 0/4 = 0 & p(1|cold) = 3/5 = .6 \\ P(2|hot) = 1/4 = .25 & p(2|cold = 2/5 = .4 \\ P(3|hot) = 3/4 = .75 & p(3|cold) = 0 \end{array}$$

But we don't know which path of states was taken through the machine for a given input!!!

HMM Training

- **Backward probability** β
  - 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)

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

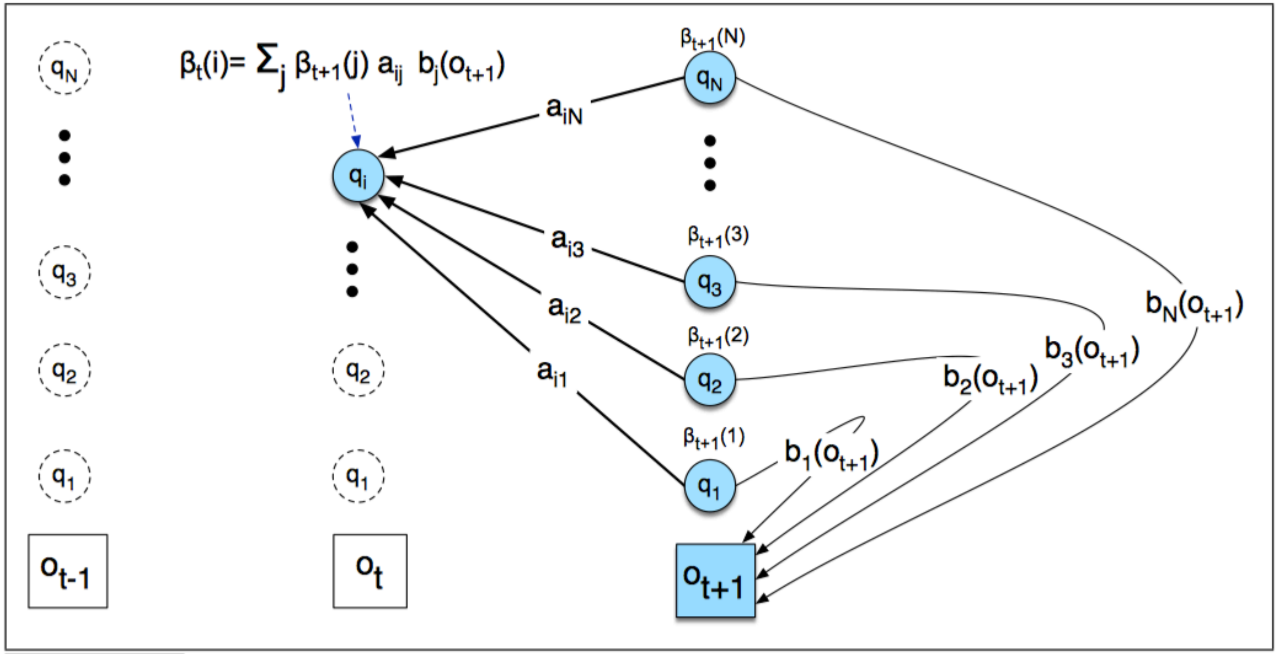

Figure A.11 The computation of $\beta_t(i)$ by summing all the successive values $\beta_{t+1}(j)$ weighted by their transition probabilities a_{ij} and their observation probabilities $b_j(o_{t+1})$. Start and end states not shown.

1. **Initialization:**

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

Final state,have already kown the whole seq

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:**

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

Given automaton, the p of O

HMM Training

- **Probability** ξ_t
  - the probability of being **in state i at time t and state j at time t +1**, **given** the observation sequence and the model

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

- **Probability** not-quite-$\xi_t(i,j)$

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

HMM Training

- **Probability** not-quite-$\xi_t(i,j)$

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

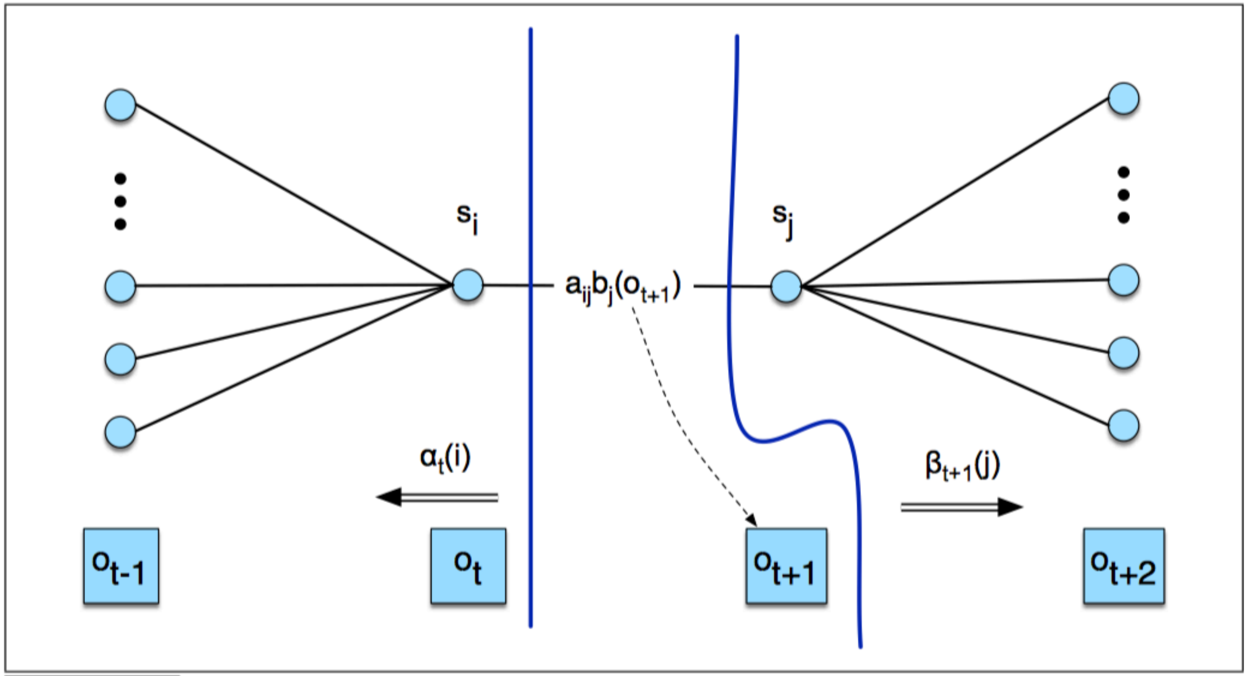

Figure A.12 Computation of the joint probability of being in state i at time t and state j at time $t+1$. The figure shows the various probabilities that need to be combined to produce $P(q_t = i, q_{t+1} = j, O|\lambda)$: the α and β probabilities, the transition probability a_{ij} and the observation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

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

HMM Training

- The probability of the observation given the model is simply the forward probability of the whole utterance (or alternatively, the backward probability of the whole utterance):

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

Note that backward pro at time t does not including Ot,it refers Ot+1

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

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

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

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

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

HMM Training

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

$$\Downarrow$$

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

HMM Training

- **Probability** $\gamma_t(j)$
  - **the probability of being in state j at time t**

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

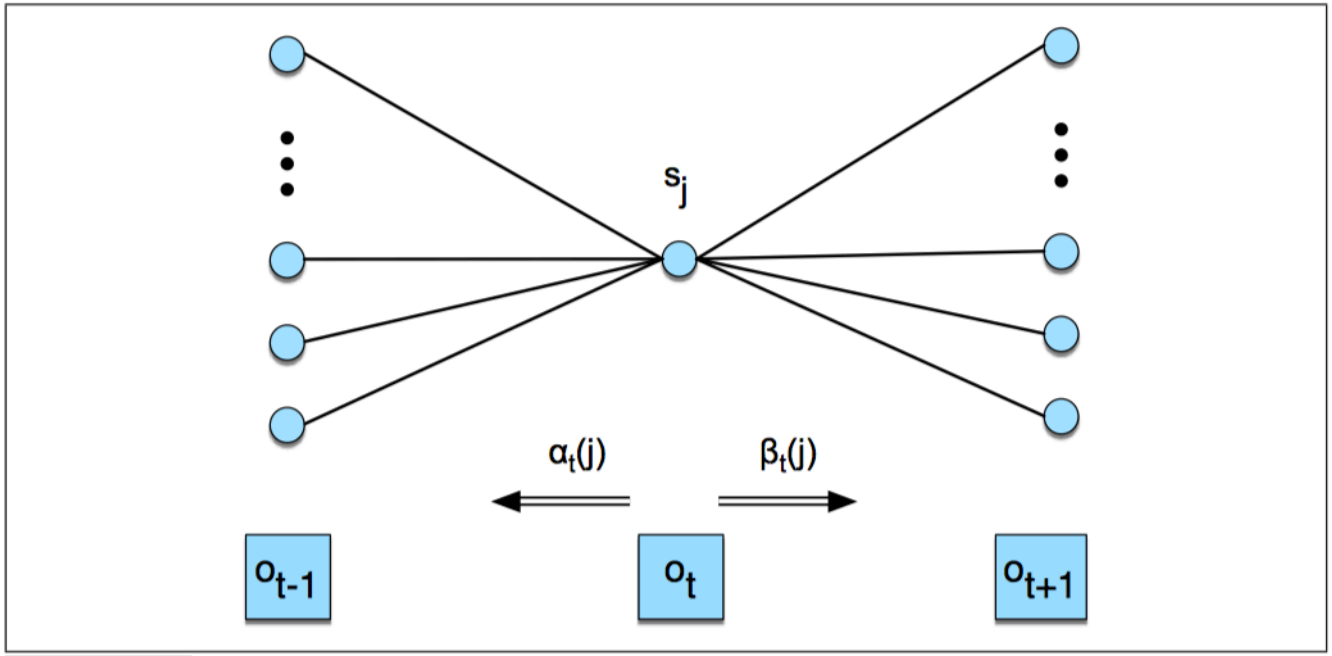

Figure A.13 The computation of $\gamma_t(j)$, the probability of being in state j at time t. Note that γ is really a degenerate case of ξ and hence this figure is like a version of Fig. A.12 with state i collapsed with state j. After Rabiner (1989) which is ©1989 IEEE.

HMM Training

Emission score P(j->Vk)

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

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

HMM Training

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

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

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

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*

Summary

- **Hidden Markov models (HMMs**) are a way of relating a sequence of observations to a sequence of hidden classes or hidden states that explain the observations.

- The process of discovering the sequence of hidden states, given the sequence of observations, is known as **decoding** or **inference**. The **Viterbi algorithm** is commonly used for decoding.

- The parameters of an HMM are the **A transition probability matrix** and the **B observation likelihood matrix**. Both can be trained with the **Baum-Welch or forward-backward algorithm**.

Thanks!