Hidden Markov Model

Dibyendu Das
M.Sc in Big Data Analytics
Ramakrishna Mission Vivekananda Educational and Research Institute

Markov Chain :

A Markov chain is specified by the following components:

1. $Q = q_1, \mathbf{q_2}, , , , \mathbf{q_n}$ A set of n states

2. **A =**

a_{11}	a_{12}	a_{13}
a_{21}	a_{22}	a_{23}
a_{31}	a_{32}	a_{33}

Transition Matrix , where each a_{ij} represents the transition between state i -> j such that $\sum a_{ij} = 1$,i = 1(1)n
j = 1(1)n

3. **$\pi = \pi_1, \pi_2, ..., \pi_n$ an initial probability distribution over states. π_i is the probability that the Markov chain will start in state i , and $\sum \pi_i = 1$, i = 1(1)n**

Markov Chain Continue...

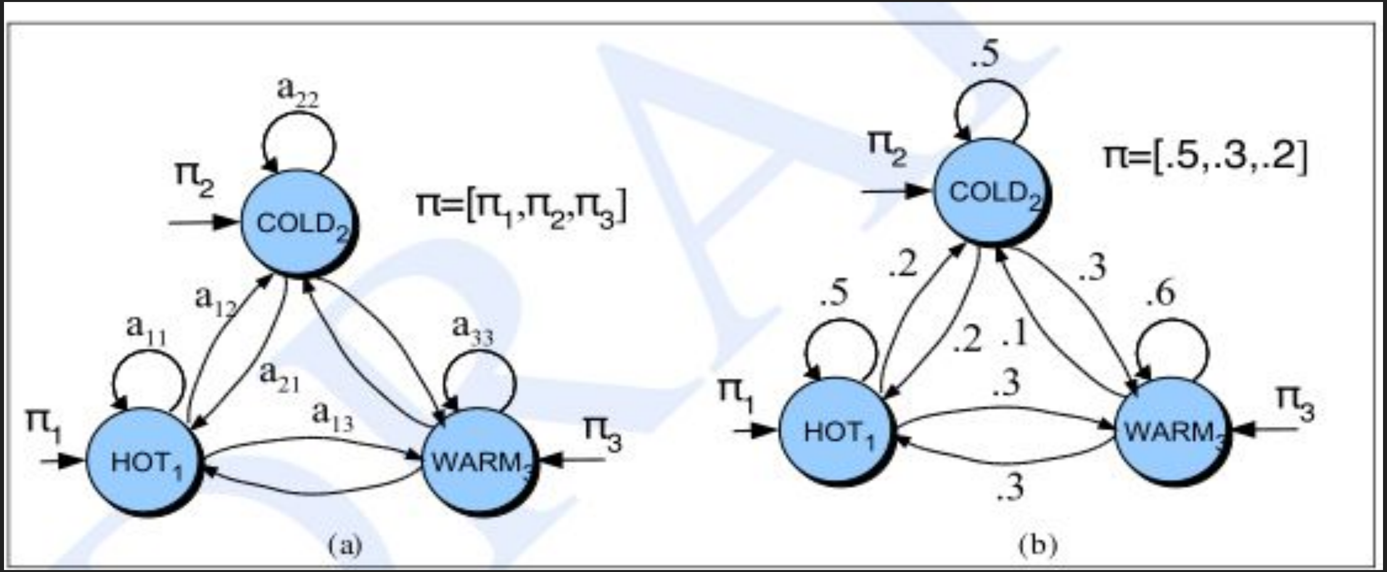
4. q_0 , q_f a special start state and end state which are not associated with observation.

<u>Markov Assumption</u> : The probability of a particular state depends only on the previous state.

$$P(q_i \mid q_1 , q_2, , , q_{i-1}) = P(q_i \mid q_{i-1})$$

Markov Chain Continue...

Markov chain for assigning a probability to a sequence of weather events,whether the vocabulary consists of HOT ,COLD, and RAINY.



The Hidden Markov Model

A Markov chain is useful when we need to compute a probability for a sequence of events that we can observe in the world. In many cases, however, the events we are interested in may not be directly observable in the world. For example, in part-of-speech tagging we didn’t observe part of speech tags in the world; we saw words, and had to infer the correct tags from the word sequence. We call the part-of- speech tags hidden because they are not observed. The same architecture will come up in speech recognition; in that case we’ll see acoustic events in the world, and have to infer the presence of ‘hidden’ words that are the underlying causal source of the acoustics. 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)

HMM

Let’s begin with a formal definition of a Hidden Markov Model, focusing on how it differs from a Markov chain. An HMM is specified by the following components:

1. $Q = q_1, \mathbf{q_2}, , , , \mathbf{q_n}$ $\longrightarrow$ A set of n states
2. $A = a_{11}\ a_{12} \ldots a_{n1} \ldots a_{nn}$ $\Rightarrow$ Transition Matrix , where each a_{ij} represents the transition between state i -> j such that $\sum a_{ij} = 1$,i = 1(1)n , j = 1(1)n
3. $O = o_1\ o_2 \ldots o_T$ $\Rightarrow$ Observation sequence
4. $B = b_i(o_t)$ $\Rightarrow$ A sequence of observation likelihoods: also called emission probabilities , each expression the probability of an observation O_t being generated from a state i .
5. q_o, q_F start and end state

HMM

6. **$\pi = \pi_1, \pi_2, ..., \pi_n$ an initial probability distribution over states. π_i is the probability that the Markov chain will start in state i , and $\sum \pi_i = 1$, i = 1(1)n**

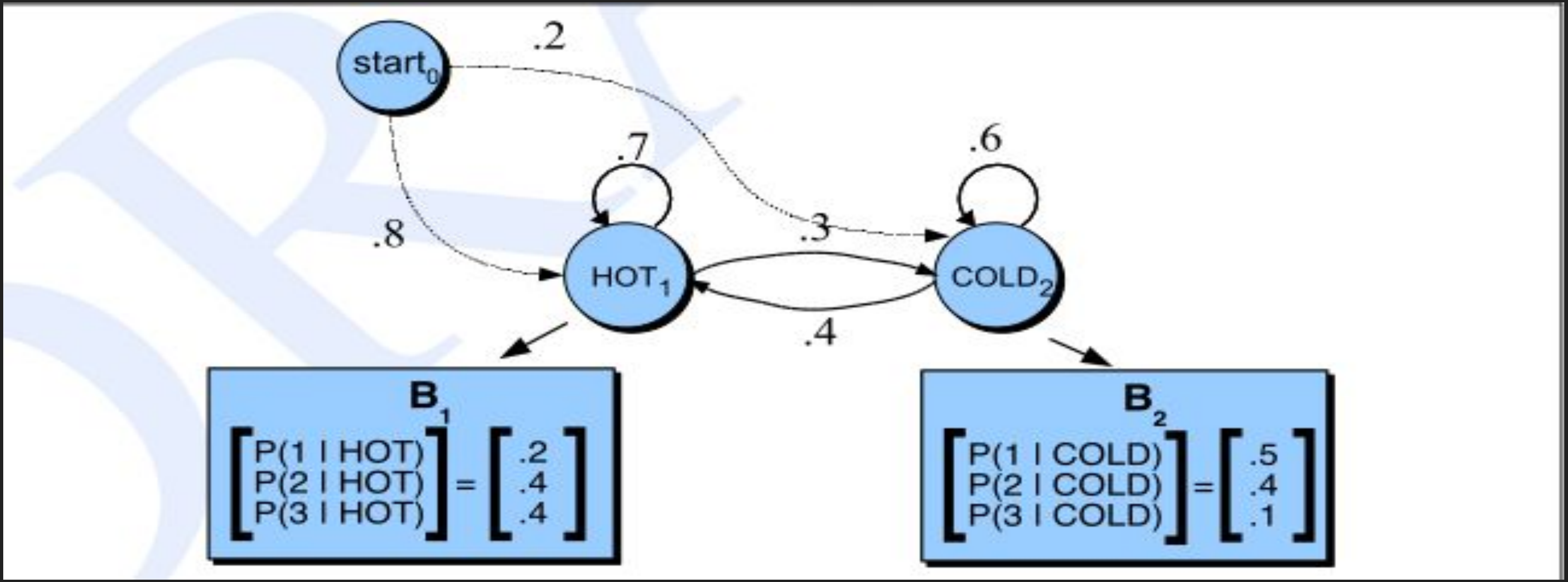
Markov Assumption : $P(q_i \mid q_1, q_2, , , q_{i-1}) = P(q_i \mid q_{i-1})$

Output Independence :

$$P(O_i \mid q_1, q_2, , , q_T, O_1, O_2, \dots, O_T) = P(O_i \mid q_i)$$

HMM Example :

Fig Shows a simple HMM for the ice cream task. The two hidden states(H and C) Corresponds to hot and cold weather ,while the observation O = { 1,2,3 } Correspond to number of ice cream eaten by Dibyendu on a given day.


start$_0$
.2
.8
.7
.6
.3
.4
HOT$_1$
COLD$_2$
B_1
$\begin{bmatrix} P(1 \mid HOT) \\ P(2 \mid HOT) \\ P(3 \mid HOT) \end{bmatrix} = \begin{bmatrix} .2 \\ .4 \\ .4 \end{bmatrix}$
B_2
$\begin{bmatrix} P(1 \mid COLD) \\ P(2 \mid COLD) \\ P(3 \mid COLD) \end{bmatrix} = \begin{bmatrix} .5 \\ .4 \\ .1 \end{bmatrix}$

Three fundamental Problems:

<u>Problem 1 (Likelihood)</u> : Given an HMM λ = (A, B) and a observation sequence O , determine the likelihood P(O | λ) .

<u>Problem 2 (Decoding)</u> : Given an observation sequence O nas an HMM λ = (A, B) , discover the best hidden sequence Q

<u>Problem 3 (Learning)</u> : Given an observation sequence O and the set of states in the HMM , Learn HMM parameter A and B

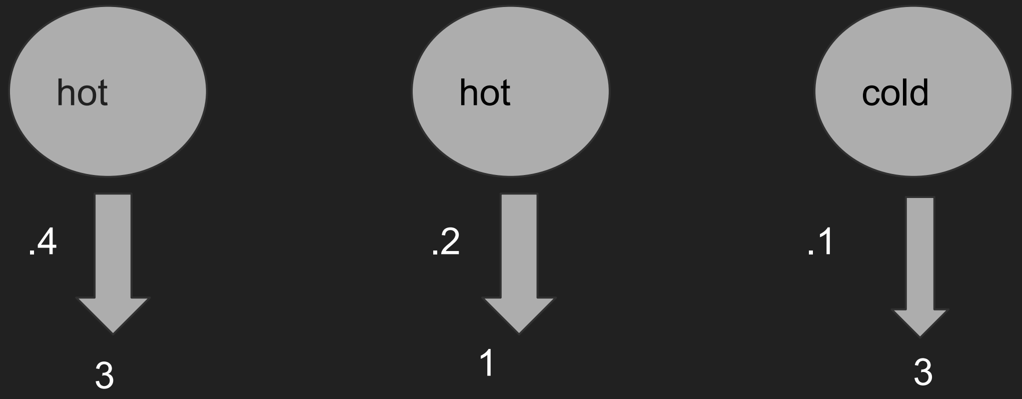
Computing Likelihood : The Forward Algorithm

For a particular hidden state sequence $Q = q_0 , q_1 , q_2 , ..., q_T$ and an observation sequence $O = o_1 , o_2 , ..., o_T$, the likelihood of the observation sequence is:

$$P(O \mid Q) = \prod P (o_i \mid q_i) \quad , i = 1(1)T$$

The computation of forward probability for our ice-cream observation 3 1 3 from one possible hidden state sequence { hot , hot , cold } as follows

P (3 1 3 | hot hot cold) = P (3 | hot) * P (1 | hot) * P (3 | cold)



Forward Algorithm Continue..

But of course, we don’t actually know what the hidden state (weather) sequence was. We’ll need to compute the probability of ice-cream events 3 1 3 instead by summing over all possible weather sequences, weighted by their probability.

$P (O, Q) = P(O \mid Q) \times P(Q) = \prod P (o_i \mid q_i) \times \prod P (q_i \mid q_{i-1})$

For our particular case, we would sum over the 8 three-event sequences cold cold cold, cold cold hot, i.e

P(3 1 3) = P(3 1 3, cold cold cold)+P(3 1 3, cold cold hot)+P(3 1 3, hot hot cold)+...

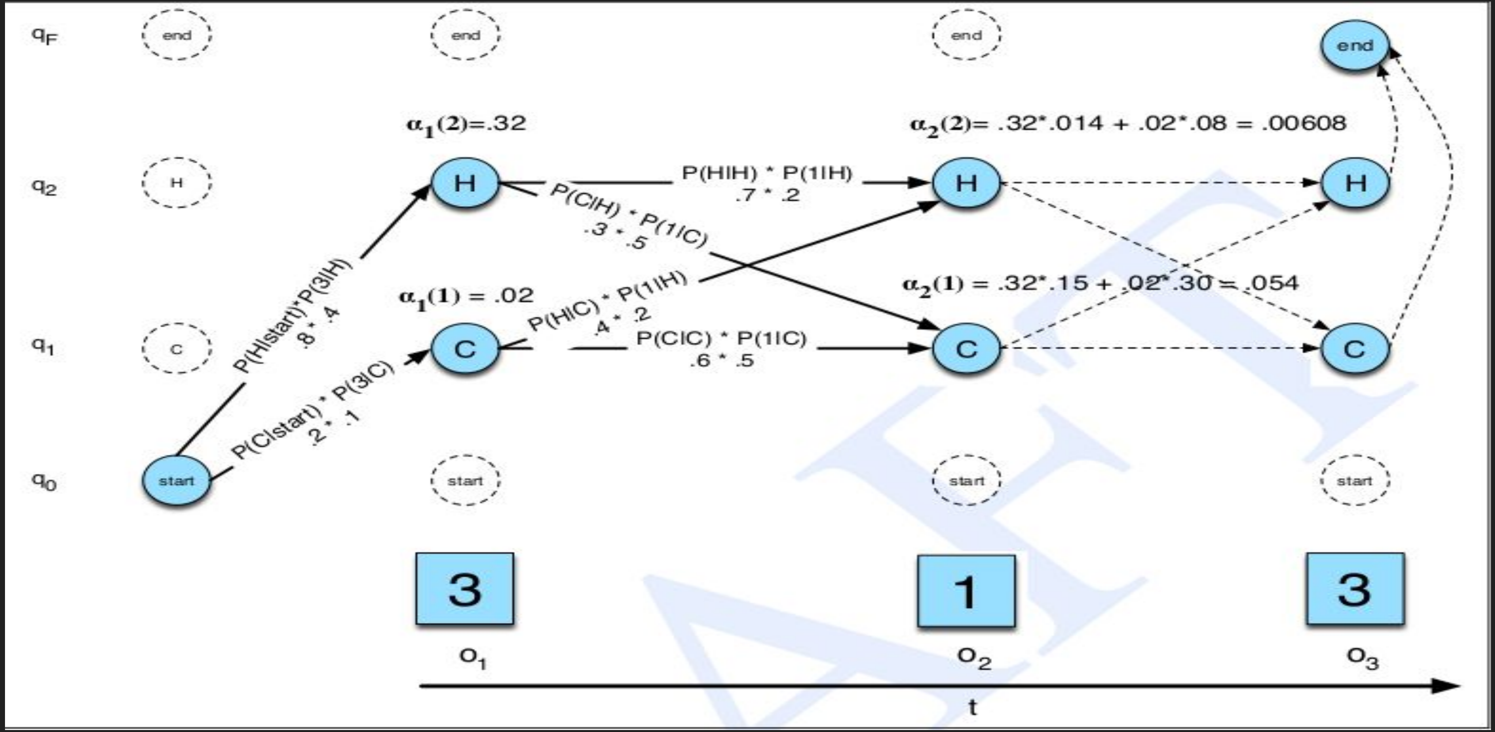
For an HMM with N hidden states and an observation sequence of T observations, here are N^T possible hidden sequences

Forward Algorithm Continue ...

Instead of using such an extremely exponential algorithm, we use an efficient ($O(N^2 T)$) algorithm called the forward algorithm. The forward algorithm is a kind of dynamic programming algorithm, i.e., an algorithm that uses a table to store intermediate values as it builds up the probability of the observation sequence. The forward algorithm computes the observation probability by summing over the probabilities of all possible hidden state paths that could generate the observation sequence, but it does so efficiently by implicitly folding each of these paths into a single forward trellis.

Forward Algorithm Continue...

Fig. shows an example of the forward trellis for computing the likelihood of 3 1 3 given the hidden state sequence hot hot cold.



Forward Algorithm Continue ..

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 .

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

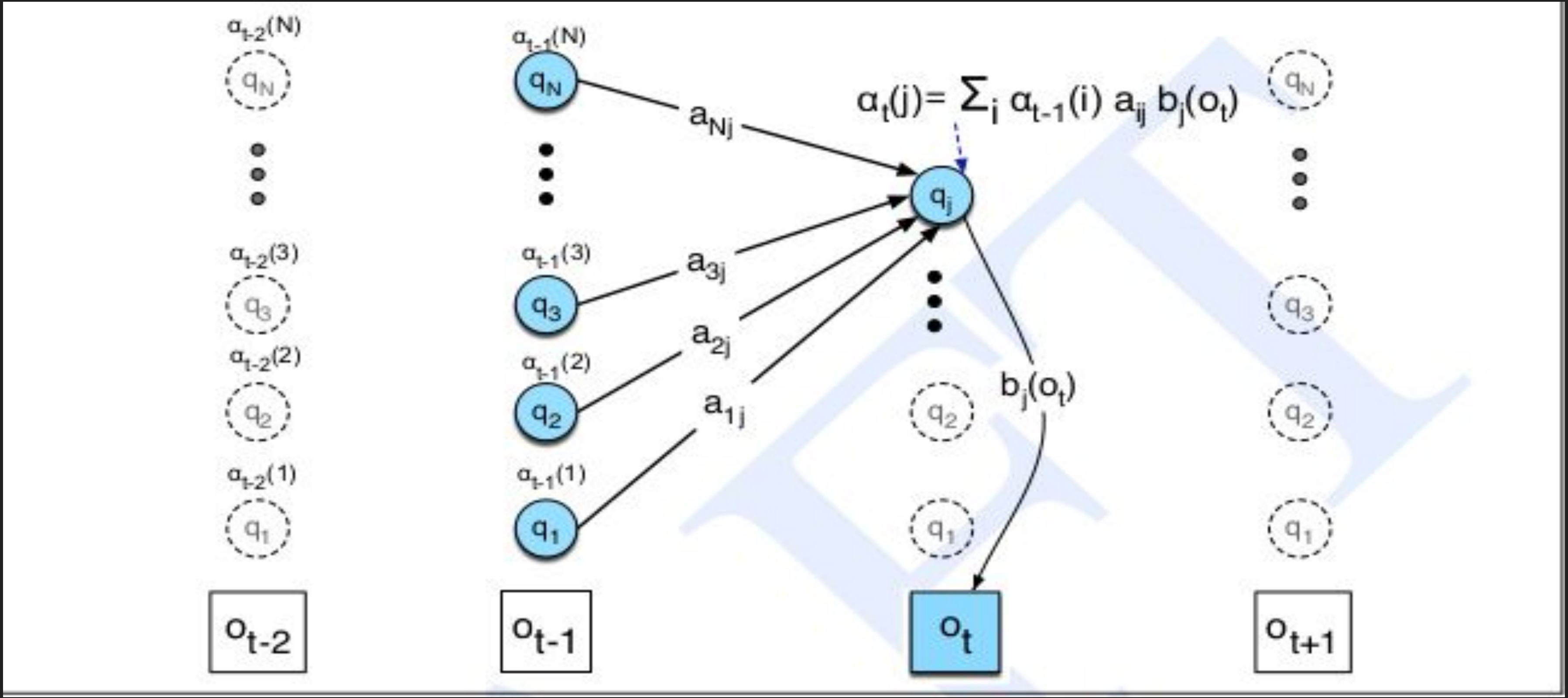
$$= \sum \alpha_{t-1}(i) * a_{ij} * b_j(o_t) \quad i = 1(1)n$$

$\alpha_{t-1}(i)$ = the previous forward path probability from the previous time step

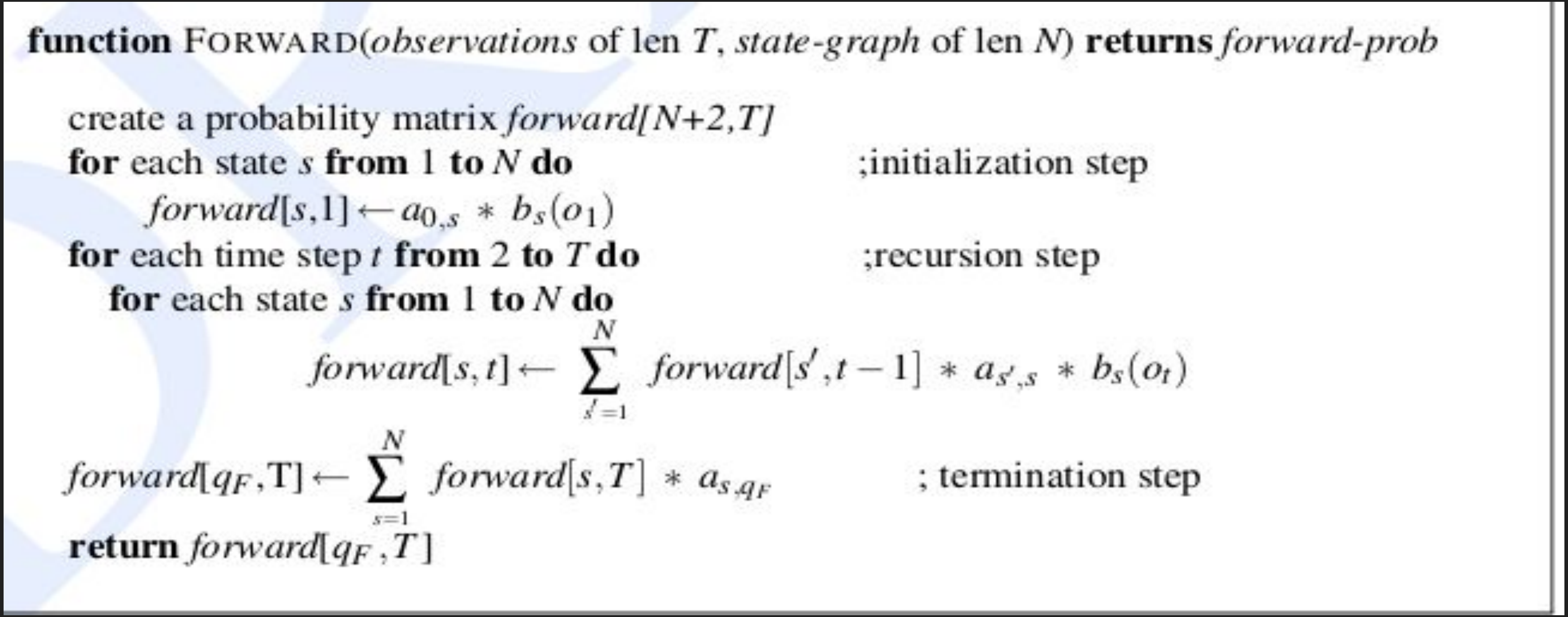
a_{ij} = the transition probability from previous state q_i to current state q_j

$b_j(o_t)$ = the state observation likelihood of the observation symbol o_t given current state j

Forward Algorithm Continue...



Forward Algorithm :



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

  create a probability matrix forward[N+2,T]
  for each state s from 1 to N do                          ;initialization step
      $forward[s,1] \leftarrow a_{0,s} * b_s(o_1)$
  for each time step t from 2 to T do                      ;recursion step
    for each state s from 1 to N do
          $forward[s,t] \leftarrow \sum_{s'=1}^{N} forward[s',t-1] * a_{s',s} * b_s(o_t)$
  $forward[q_F,T] \leftarrow \sum_{s=1}^{N} forward[s,T] * a_{s,q_F}$            ; termination step
  return $forward[q_F,T]$
```

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.

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 $Q = q_1 \, q_2 \; . \; . \; . \; q_T$.

Viterbi Algorithm Continue….

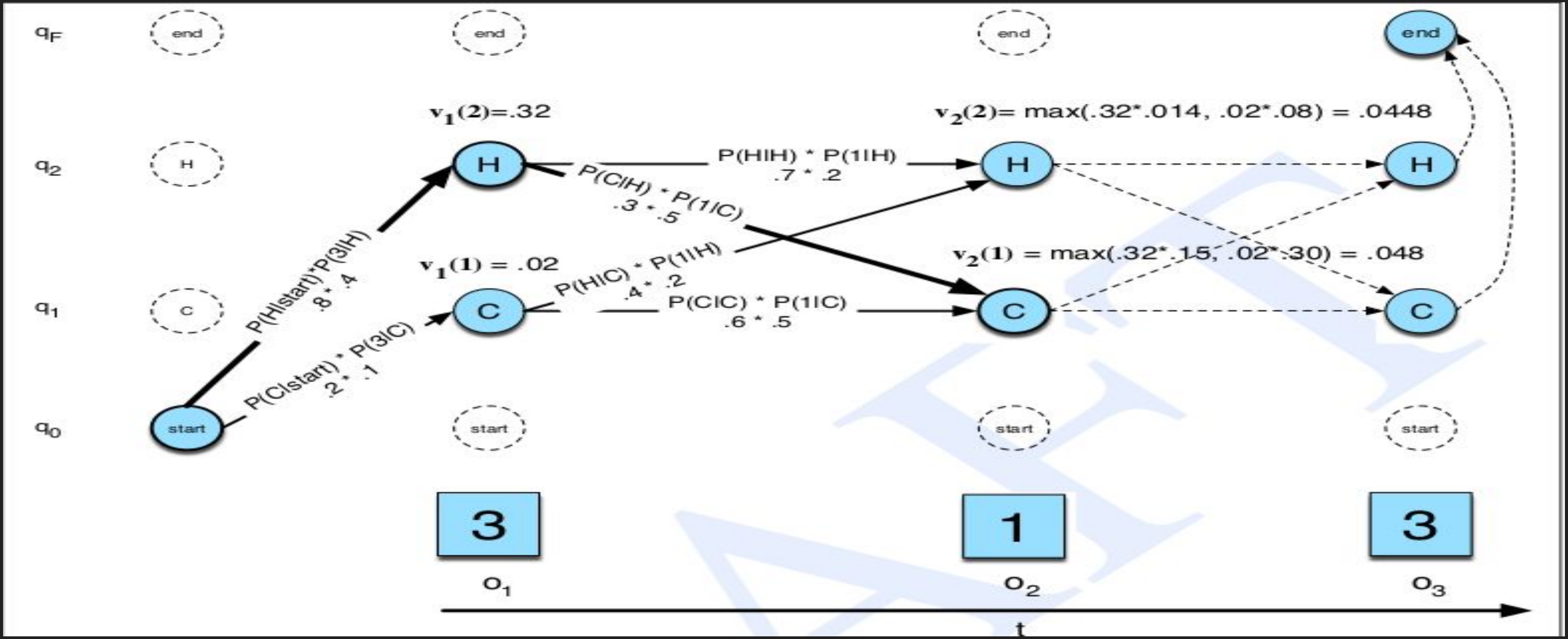
Each cell of the Viterbi trellis, v_t (j) represents the probability that the HMM is in state j after seeing the first t observations and passing through the most probable state sequence q_0 , q_1, ..., q_{t-1} , given the automaton λ . The value of each cell v_t (j) is computed by recursively taking the most probable path that could lead us to this cell. Formally, each cell expresses the following probability:

$$v_t(j) = \max_{q_0,q_1 \ldots, q_{t-1}} P(q_0, q_1 \ldots q_{t-1}, o_1, o_2 \ldots o_t, q_t = j \mid \lambda)$$

$$v_t(j) = \max v_t(j) * a_{ij} * b_j(o_t) \quad \text{for } i = 1\ (1)\ N$$

Viterbi Algorithm Continue….

The Viterbi trellis for computing the best path through the hidden state space for the ice-cream eating events 3 1 3 .



Training HMMs: 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.

In order to understand the algorithm, we need to define a useful probability related to the forward probability, called the backward probability.

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 .

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

1. ***Initialization:*** $\beta_T (i) = a_{I\,,\,F}\ , 1 \le i \le N$

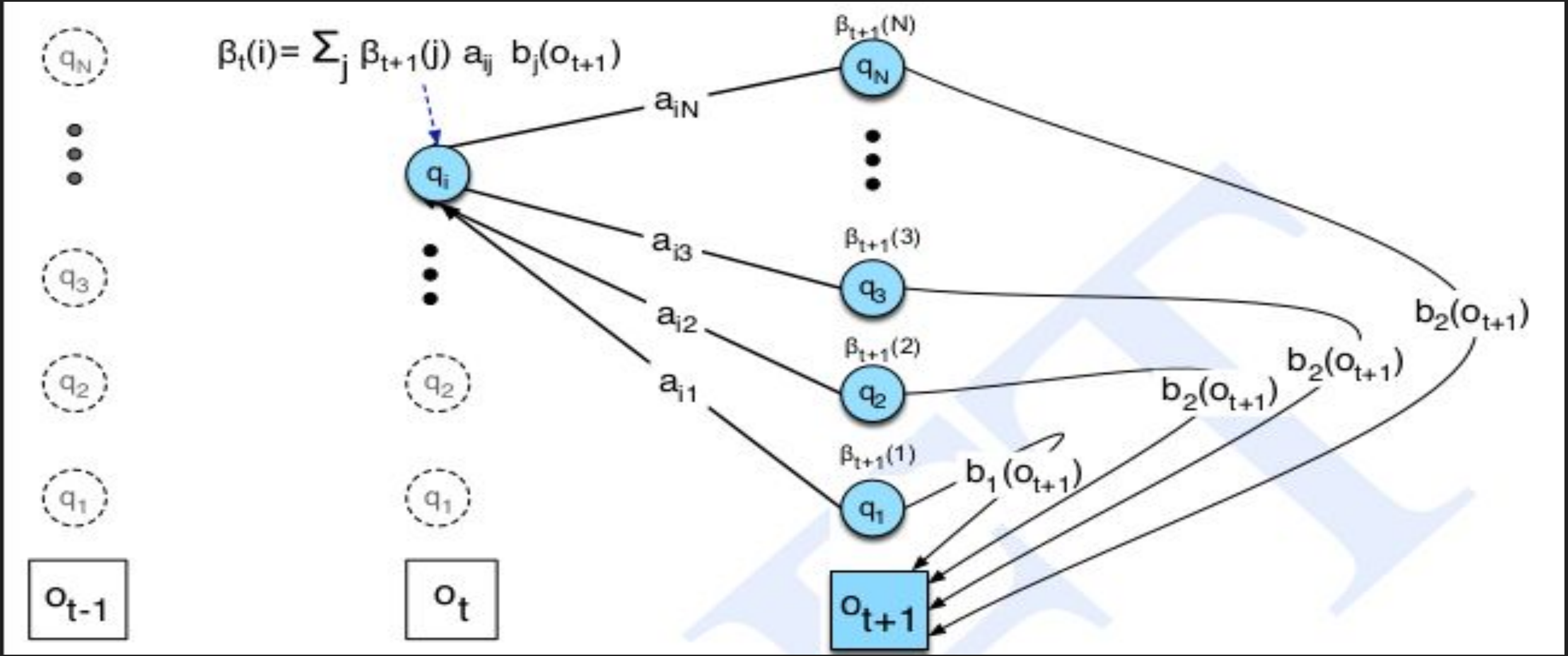
2. Recursion (again since states 0 and q F are non-emitting):

$$\beta_T (i) = \sum a_{ij}\ b_j (o_{t+1})\ \beta_{t+1} (j) \quad 1 \le i \le N, 1 \le t < T , j = 1(1) N$$

3. ***Termination:***

$$P(O| \lambda) = \alpha_T (q_F) = \beta_1 (0) = \sum a_{oj}\ b_j (o_j)\ \beta_1 (j) \quad , j = 1(1)N$$

Backward probability Continue ….



Estimation Of a_{ij} :

Let's begin by showing how to estimate a_{ij} ,

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

How do we compute the numerator? Here's the intuition. Assume we had some estimate of the probability that a given transition $i \rightarrow j$ was taken at a particular point in time t in the observation sequence. If we knew this probability for each particular time t, we could sum over all times t to estimate the total count for the transition $i \rightarrow j$.

Estimation Of A :

let’s define the probability ξ_t as the probability of being in state i at time t and state j at time t + 1, given the observation sequence and of course the model:

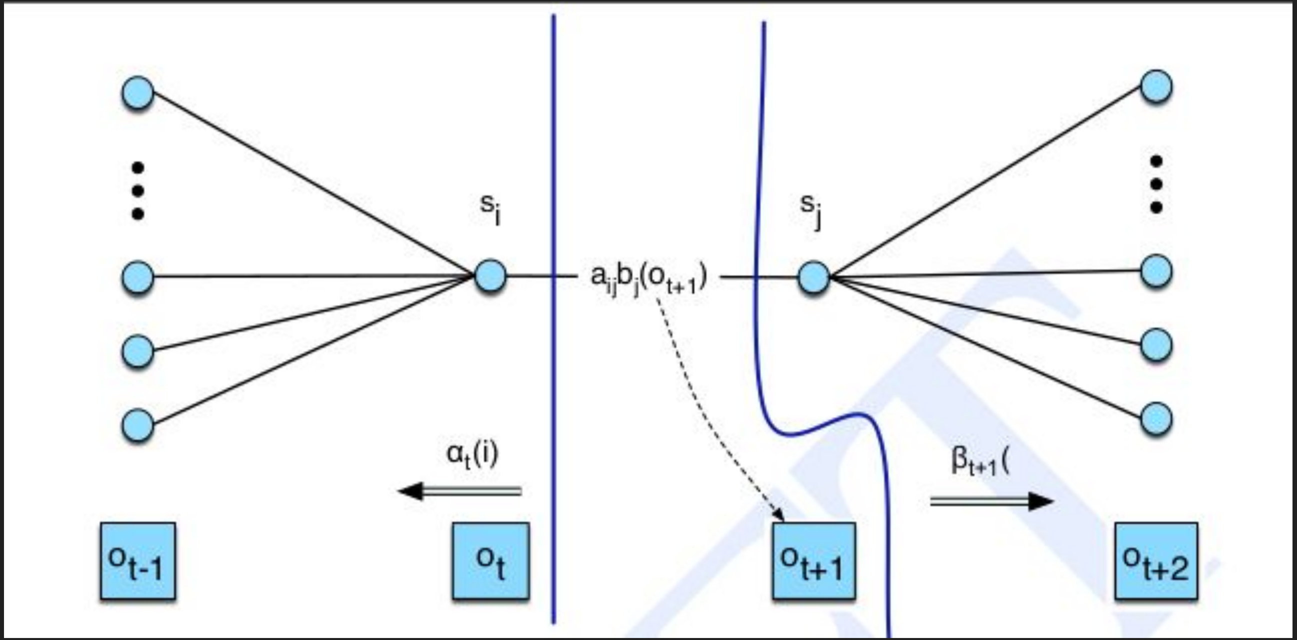
$\xi_t(i, j) = P(q_t = i, q_{t+1} = j \mid O, \lambda)$ ------- (1)

In order to compute ξ_t , we first compute a probability which is similar to ξ_t , but differs in including the probability of the observation; note the different conditioning of O from Eq. 1

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

not-quite- ξ_t (i, j) :

Computation of the joint probability of being in state i at time t and state j at time t + 1.



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

Estimation of A :

In order to compute ξ_t from not-quite- ξ_t , the laws of probability instruct us to divide by P(O| λ) , since:

$$P(X|Y, Z) = P(X,Y |Z) / P(Y |Z)$$

Now $P(O| \lambda) = \alpha_T(N)$

So, the final equation for ξ_t is:

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

Therefore we get ,

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

Estimation of B matrix :

This is the probability of a given symbol v_k from the observation vocabulary V , given a state j : $b_j (v_k)$,We will do this by trying to compute:

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

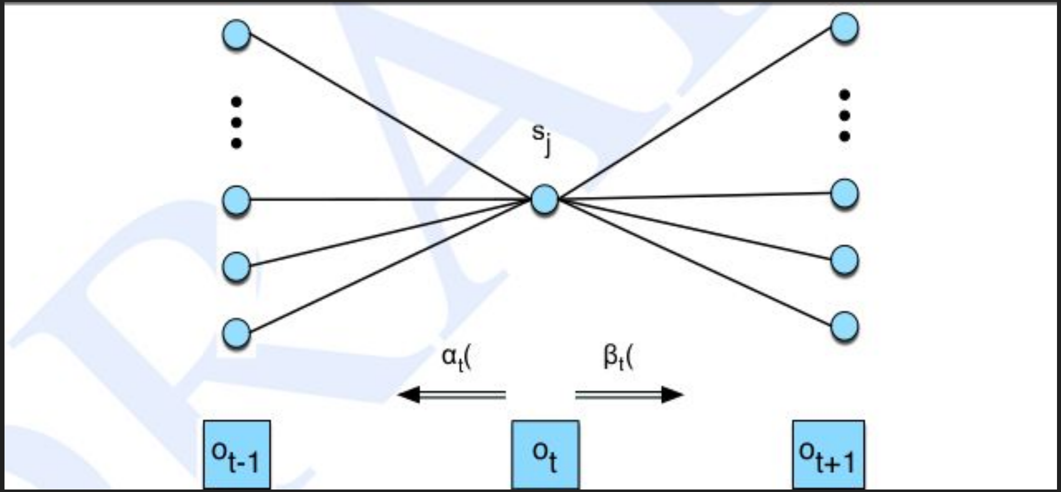
For this we will need to know the probability of being in state j at time t, which we will call $\gamma_t (j) = P(q_t = j \mid O, \lambda)$

Estimation of B matrix Continue ….

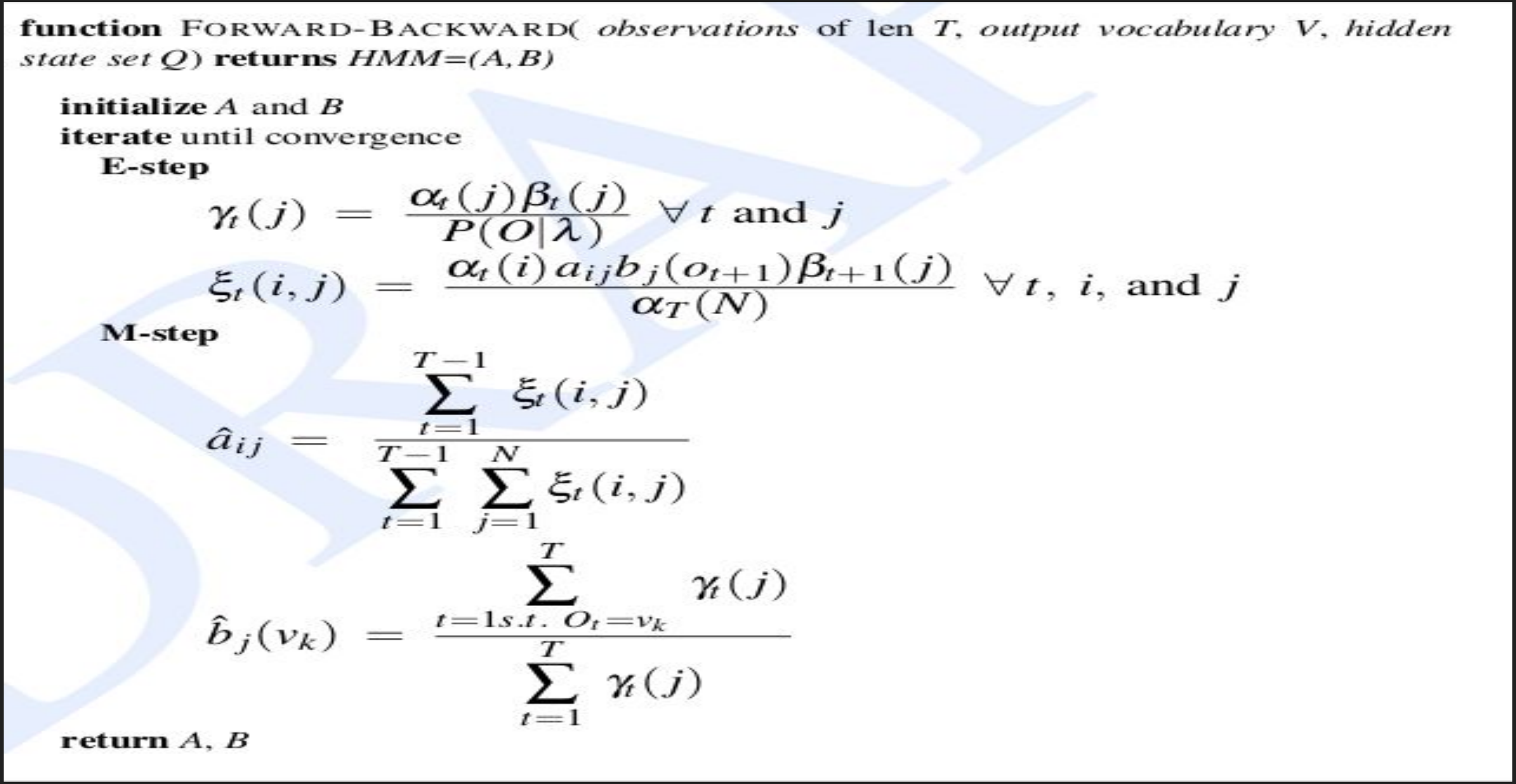
$$\gamma_t(j) = \frac{P(q_t = j, O \mid \lambda)}{P(O \mid \lambda)}$$

$$\gamma_t(j) = \frac{\alpha_t(j) * \beta_t(j)}{P(O \mid \lambda)}$$

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



Baum - Welch Algorithm :



```
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
      $\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{P(O|\lambda)} \ \forall\, t$ and $j$
      $\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij} b_j(o_{t+1})\beta_{t+1}(j)}{\alpha_T(N)} \ \forall\, t,\ i,$ and $j$
    M-step
      $\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} \xi_t(i,j)}$
      $\hat{b}_j(v_k) = \frac{\sum_{t=1 s.t.\ O_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$
  return $A$, $B$
```

Thank You