HMM NOTES

MARKOV CHAIN :-

Stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in previous event. (Time based sequence)

For T time instances :- $q_i = \{1 \ldots N\}$ ↳ one of the states.

$q_1, q_2, \ldots q_T \rightarrow$ state sequence

markov assumption :-

$$p(q_t \mid q_{t-1}, q_{t-2} \ldots) = p(q_t \mid q_{t-1})$$

How to characterize a markov chain?

$$p(q_t = j \mid q_{t-1} = i) = a_{ij} \rightarrow \text{Transition probabilities}$$

$i, j = \{1 \ldots N\}$

a_{ij} does not depend on time, only the states involved

Markov chain parameters :

$$A = \begin{bmatrix} & & \\ & a_{ij} & \\ & & \end{bmatrix}_{N \times N}$$

a_{ij} = prob. of transitioning from i to j.

$\pi = \left[\ p(q_0 = i)\ \right]_{1 \times N}$ — starting probability (probability of some state being the first $(t=1)$)

HIDDEN MARKOV MODEL :-

- Statistical markov model in which the system being modeled is assumed to be a markov process with unobservable (hidden) states.

- We have emissions associated with each state. These emissions are observed.

- Let O_i be observation at time instance i

$O = O_1 O_2 \dots O_T$ is one observation seq.

$O_i \in \{ S_1 \dots S_M \}$ — symbols

- Along with A & π defined in markov chain, we also have B :-

$$B = \begin{bmatrix} & b_{jk} & \end{bmatrix}_{N \times M}$$

$b_{jk} = b_j(K)$
i.e. probability of emitting symbol K at state j

- Thus an HMM (say λ) can be characterized by A, B and π $[\lambda = (A, B, \pi)]$

Three classic problems of HMM :-

Problem 1 - Evaluation :-

→ How likely / probable is an observation sequence under a given HMM λ.

→ Using a naive solution we encounter a time complexity of $2TN^T$ if we consider an observation sequence for T time instances and N states

→ N^T possible sequences
→ each sequence needs T multiplications

→ As a solution we utilize some of the key properties and come up with forward method & backward method.

Forward Method :- $O(N^2T)$

- define :-

$\alpha_t(i)$ — forward variable

where :-

$\alpha_t(i) = P(O_1, O_2, \ldots O_t, q_t = i/\lambda)$

↳ partial sequence upto t such the state at time instance t is i

Initialization :-

$\alpha_1(i) = P(O_1, q_1 = i/\lambda)$

$= \pi_i b_i(O_1)$ (over all i)

Recursion :-

$$\alpha_{t+1}(j) = \sum_{i=1}^{N} \alpha_t(i)\, a_{ij}\, b_j(O_{T+1})$$

$(1 \le j \le N \text{ and } 1 \le t \le T-1)$

Final result :-

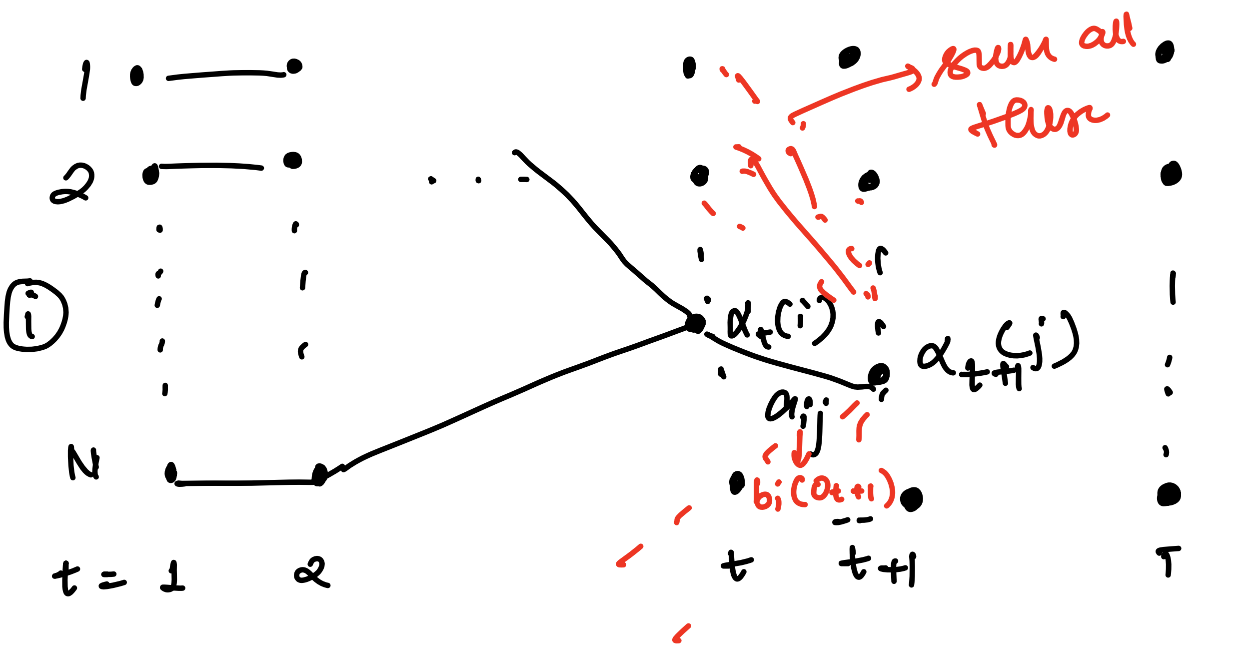
$$P(O/\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

Time complexity :- $O(N^2T)$

for each time instance we have N multiplication for each state i.e. N^2 computaions for T time instances.

For visualization of Forward method :-

Trellis



For ith state :

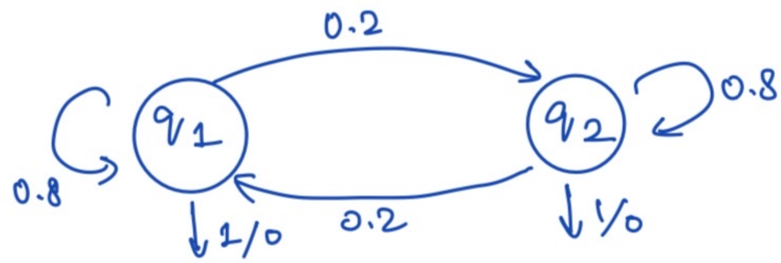
(similar to dp)

$\alpha_t(i)$ find all ways of reaching i state at t & emitting $O_1, O_2 \dots O_t$

Example of forward method:-

(Question in slides)

The discrete HMM having two states:-



We have initial probabilities

$$\pi = \begin{array}{cc} \left[\,0.6\right. & \left.0.4\,\right] \\ q_1 & q_2 \end{array}$$

Transition probabilities

$$A = \begin{array}{c|cc} & q_1 & q_2 \\ q_1 & 0.8 & 0.2 \\ q_2 & 0.2 & 0.8 \end{array}$$

Emission probabilities (binary emission)

$$B = \begin{array}{c|cc} & 0 & 1 \\ q_1 & 1 & 0 \\ q_2 & 0 & 1 \end{array}$$

We have HMM λ characterized by A, B, π

$\because$ it is already observed $o_3 = 0$, $o_4 = 0$, $o_5 = 1$

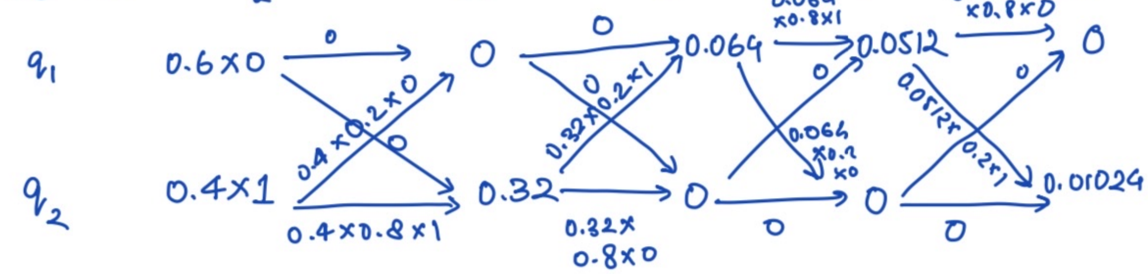
we have probable sequences by varying o_1 and o_2 as 00, 01, 10, 11 :-

$$\{S_1, S_2, S_3, S_4\} = \{11001, 00001, 10001, 01001\}$$

Finding probability of each sequences,

For S_1, initial probabilities are $\pi_i \times b_i(O_1)$ then on we sum over $\alpha_t(i) \times a_{ij} \times b_j(O_t)$ for all i's.

t =>	1	2	3	4	5
obs =>	1	1	0	0	1

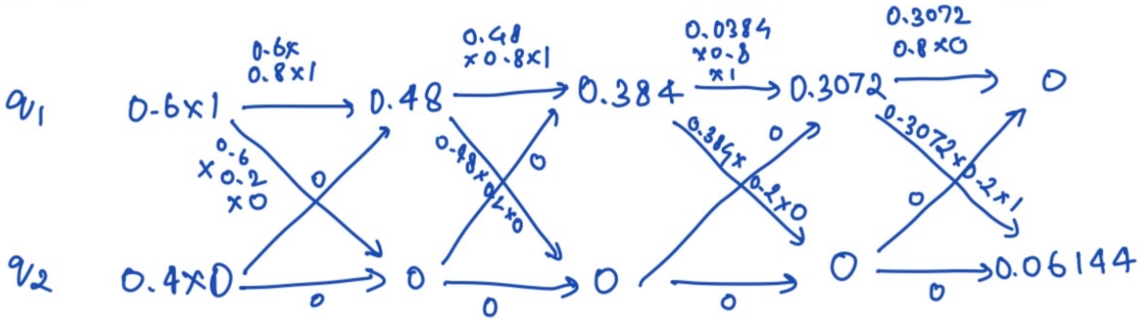


$\Rightarrow P(S_1/\lambda) = 0 + 0.01024$

$= 0.01024$

Similarly for S_2 :-

t =>	1	2	3	4	5
obs =>	0	0	0	0	1

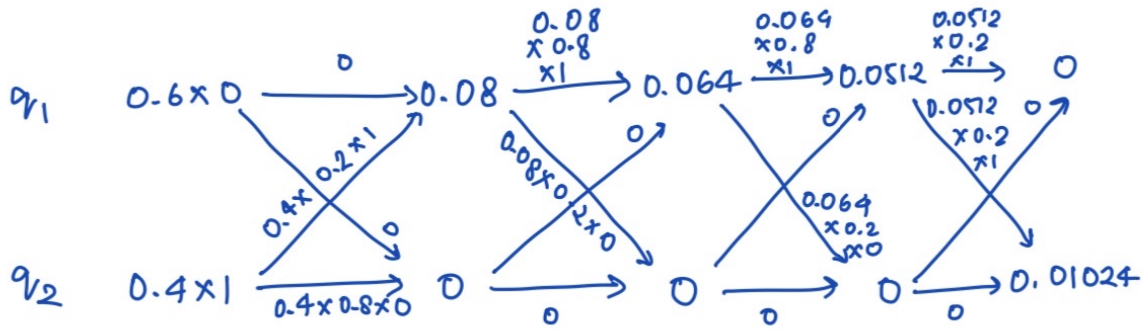


$\Rightarrow P(S_2/\lambda) = 0 + 0.06144$

$= 0.06144$

For S_3:

$t \Rightarrow$	1	2	3	4	5
obs $\Rightarrow$	1	0	0	0	1

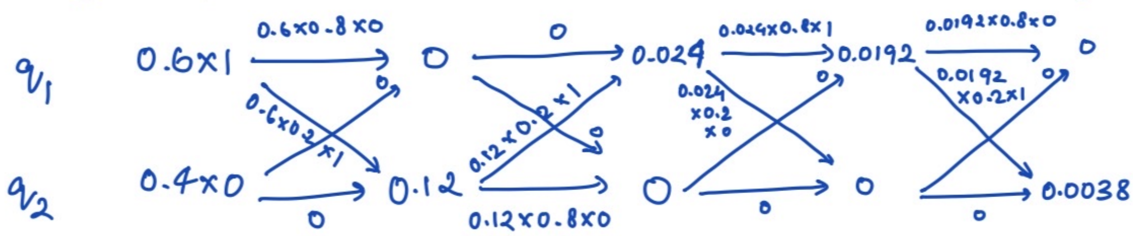


$\Rightarrow P(S_3/\lambda) = 0 + 0.01024$

$= 0.01024$

For S_4:

$t \Rightarrow$	1	2	3	4	5
obs $\Rightarrow$	0	1	0	0	1



$\Rightarrow P(S_4/\lambda) = 0 + 0.0038$

$= 0.0038$

$\Rightarrow$ $\boxed{\begin{array}{l}\text{Total probability of observing } O_3 = 0,\ O_4 = 0 \text{ and } O_5 = 1 \\ = P(S_1/\lambda) + P(S_2/\lambda) + P(S_3/\lambda) + P(S_4/\lambda) \\ = 0.08576\end{array}}$

Problem 2 : Decoding :

→ Given an observation sequence, find the best probable state sequence.

VITERBI ALGORITHM :-

→ <u>Recurrent property</u> :

$$\delta_{t+1}(j) = \max_i \left(\delta_t(i)\, a_{ij} \right) b_j(O_{t+1})$$

↳ for which previous state i, the probability of transitioning to j and emitting O_{t+1} is the maximum

→ Algorithm :-

1. Initialize :

↳ probability of emitting O_1 in state i at time instance 1

$\delta_1(i) = \pi_i\, b_i(O_1)$

↳ $\delta_{N \times T}$ $\quad 1 \leq i \leq N$

$\psi_1(i) = 0$

↳ no need of a back track for first step

↳ ψ will be used for backtracking

↳ $N \times T$

2. Recursion :

among all i's we pick the max value of RHS

$$\delta_t(j) = \max_i \left(\delta_{t-1}(i)\, a_{ij}\right) b_j(O_t)$$

$$\psi_t(j) = \underset{1 \le i \le N}{\arg\max} \left(\delta_{t-1}(i)\, a_{ij}\right)$$

$2 \le t \le T,\ 1 \le j \le N$

storing the state for which we were able to maximize the prob.

3. Termination :-

$$p^* = \max_{1 \le i \le N} \delta_T(i)$$ won't be used

$$q_T^* = \underset{1 \le i \le N}{\arg\max}\ \delta_T(i)$$

picking the last state using max probability

4. Backtracking state sequence :-

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \qquad t = T-1, T-2, \ldots 1$$

Code for viterbi :-

```
#t is the total time instances; o is the observation sequence; A, B, p characterize HMM
def viterbi(t,o,A,B,p):
    delta=np.zeros([len(p),t])
    psy=np.zeros([len(p),t])
    #initialize
    for i in range(len(p)):
        delta[i,0]=p[i]*B[i,o[0]]
        psy[i,0]=0
    #scaling
    temp1=np.sum(delta[:,0])
    delta[:,0]=delta[:,0]/temp1

    #recursion
    for i in range(1,t):
        for j in range(len(p)):
            for k in range(len(p)):
                if k==0:
                    temp_d = delta[k,i-1]*A[k,j]
                    temp_p = k
                else:
                    val=delta[k,i-1]*A[k,j]
                    if(val>temp_d):
                        temp_d = val
                        temp_p = k
            delta[j,i] = temp_d * B[j,o[i]]
            psy[j,i] = temp_p
        temp1=np.sum(delta[:,i])
        delta[:,i]=delta[:,i]/temp1
    #termination
    p_star = np.max(delta[:,t-1])
    q_t_star = np.argmax(delta[:,t-1])
    #print(q_t_star)

    #backtracking

    sequence=np.zeros([t])
    sequence[t-1]=int(q_t_star)
    for i in range(t-2,0,-1):
        #print(sequence[i+1])
        sequence[i]=psy[int(sequence[i+1]),i+1]

    return sequence
```