CS344: Introduction to Artificial Intelligence (associated lab: CS386)

Pushpak Bhattacharyya
CSE Dept.,
IIT Bombay

Lecture 9: Viterbi; forward and backward probabilities

25th Jan, 2011

HMM Definition

- Set of states: S where $|S|=N$
- Start state S_0 /*$P(S_0)=1$*/
- Output Alphabet: O where $|O|=M$
- Transition Probabilities: $A= \{a_{ij}\}$ /*state i to state j*/
- Emission Probabilities : $B= \{b_j(o_k)\}$ /*prob. of emitting or absorbing o_k from state j*/
- Initial State Probabilities: $\Pi=\{p_1,p_2,p_3,\ldots p_N\}$
- Each $p_i=P(o_0=\varepsilon,S_i|S_0)$

Markov Processes

- Properties
  - Limited Horizon: Given previous *t* states, a state *i*, is independent of preceding *0* to *t-k+1* states.
    - $P(X_t=i|X_{t-1}, X_{t-2}, \ldots X_0) = P(X_t=i|X_{t-1}, X_{t-2} \ldots X_{t-k})$
    - Order *k* Markov process
  - Time invariance: (shown for $k=1$)
    - $P(X_t=i|X_{t-1}=j) = P(X_1=i|X_0=j) \ldots = P(X_n=i|X_{n-1}=j)$

Three basic problems (contd.)

- Problem 1: Likelihood of a sequence
  - Forward Procedure
  - Backward Procedure
- Problem 2: Best state sequence
  - Viterbi Algorithm
- Problem 3: Re-estimation
  - Baum-Welch (Forward-Backward Algorithm)

Probabilistic Inference

- O: Observation Sequence
- S: State Sequence

- Given O find S^* where $S^* = \arg\max_S p(S/O)$ called Probabilistic Inference

- Infer "Hidden" from "Observed"
- How is this inference different from logical inference based on propositional or predicate calculus?

Essentials of Hidden Markov Model

1. Markov + Naive Bayes

2. Uses both transition and observation probability

$$p(S_k \xrightarrow{O_k} S_{k+1}) = p(O_k / S_k)\, p(S_{k+1} / S_k)$$

3. Effectively makes Hidden Markov Model a Finite State Machine (FSM) with probability

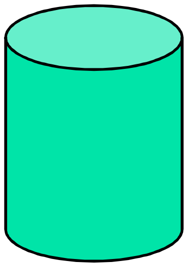
Probability of Observation Sequence

$$p(O) = \sum_{S} p(O, S)$$
$$= \sum_{S} p(S)\, p(O / S)$$

- Without any restriction,
  - Search space size= $|S|^{|O|}$

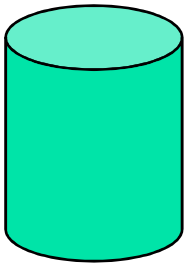
Continuing with the Urn example

Colored Ball choosing



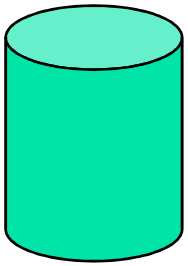
Urn 1

# of Red = 30
# of Green = 50
# of Blue = 20



Urn 2

# of Red = 10
# of Green = 40
# of Blue = 50



Urn 3

# of Red =60
# of Green =10
# of Blue = 30

Example (contd.)

Given :

Transition Probability

	U_1	U_2	U_3
U_1	0.1	0.4	0.5
U_2	0.6	0.2	0.2
U_3	0.3	0.4	0.3

and

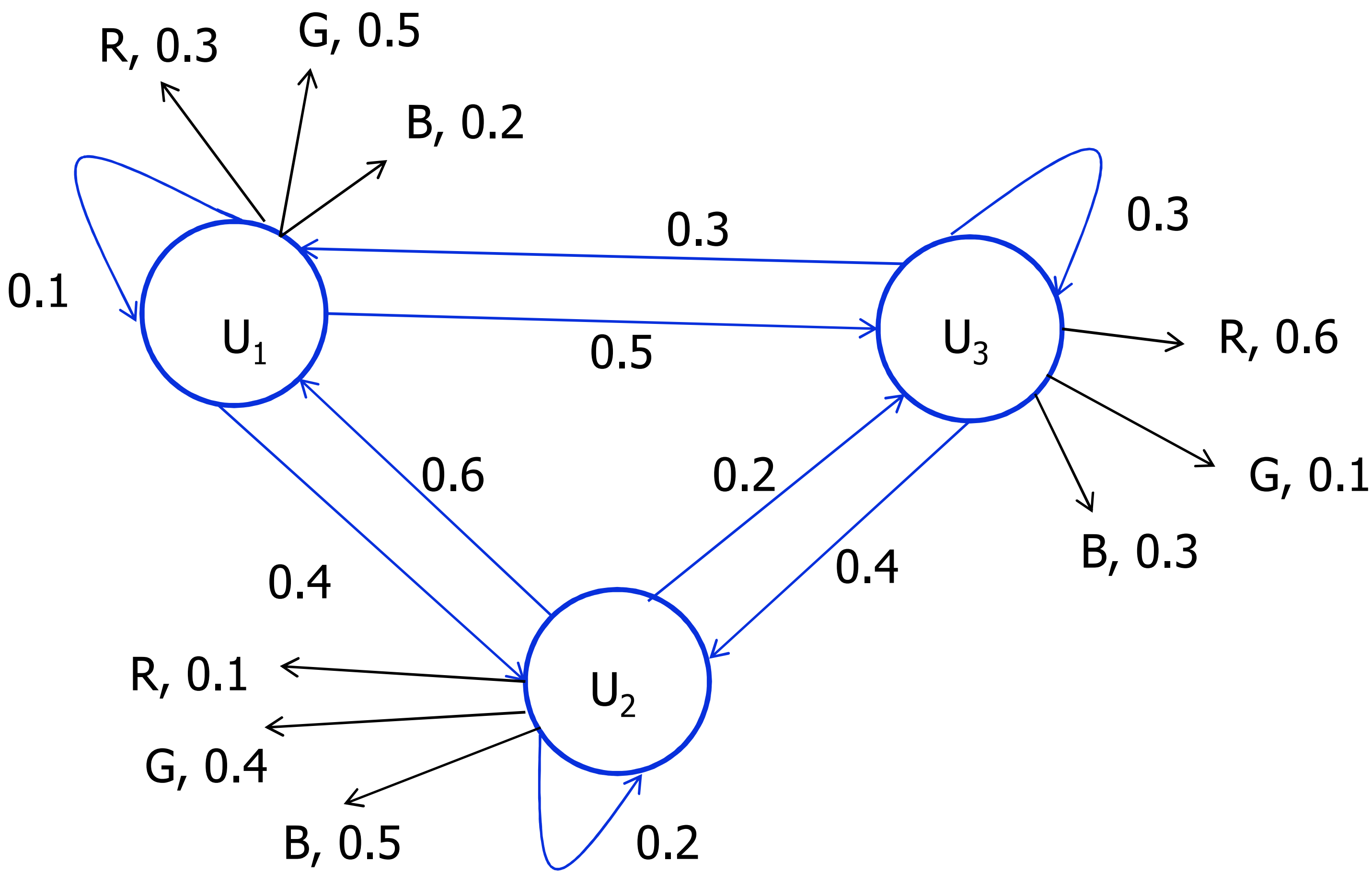
Observation/output Probability

	R	G	B
U_1	0.3	0.5	0.2
U_2	0.1	0.4	0.5
U_3	0.6	0.1	0.3

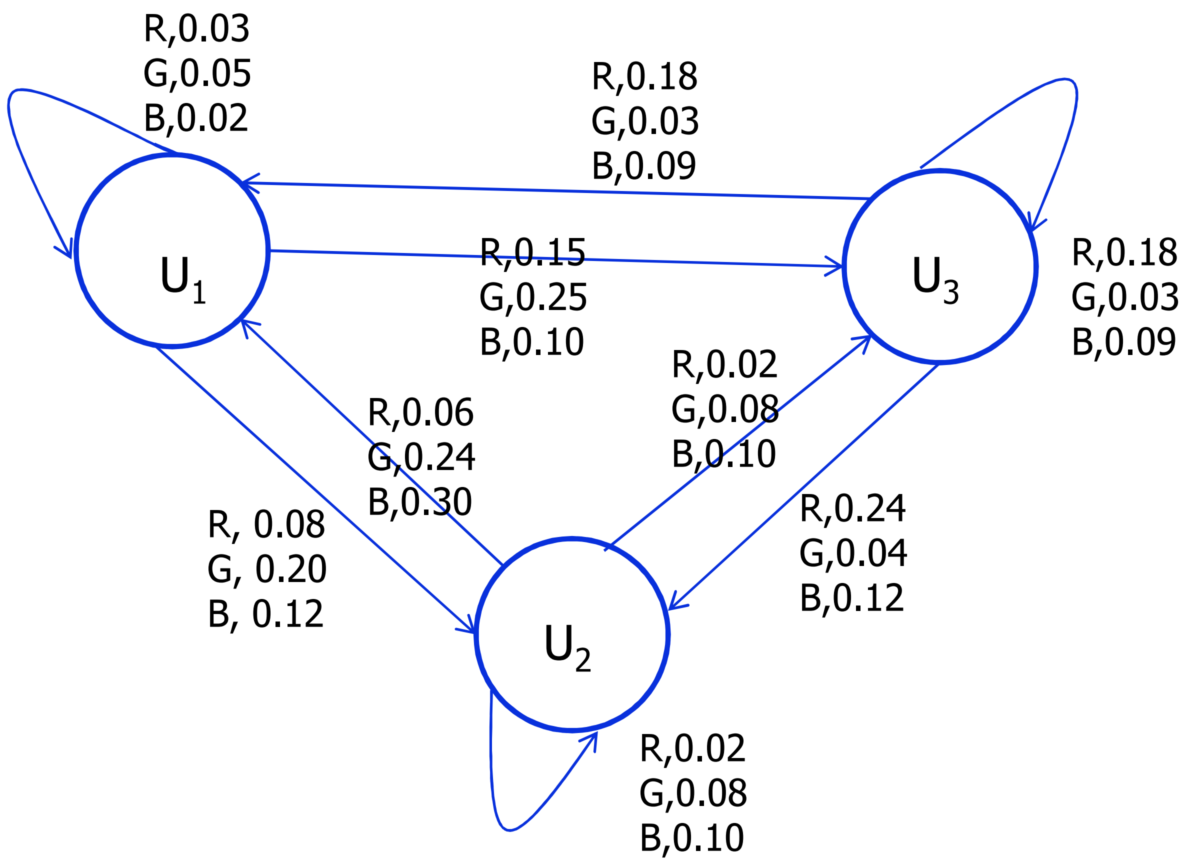
Observation : RRGGBRGR

What is the corresponding state sequence ?

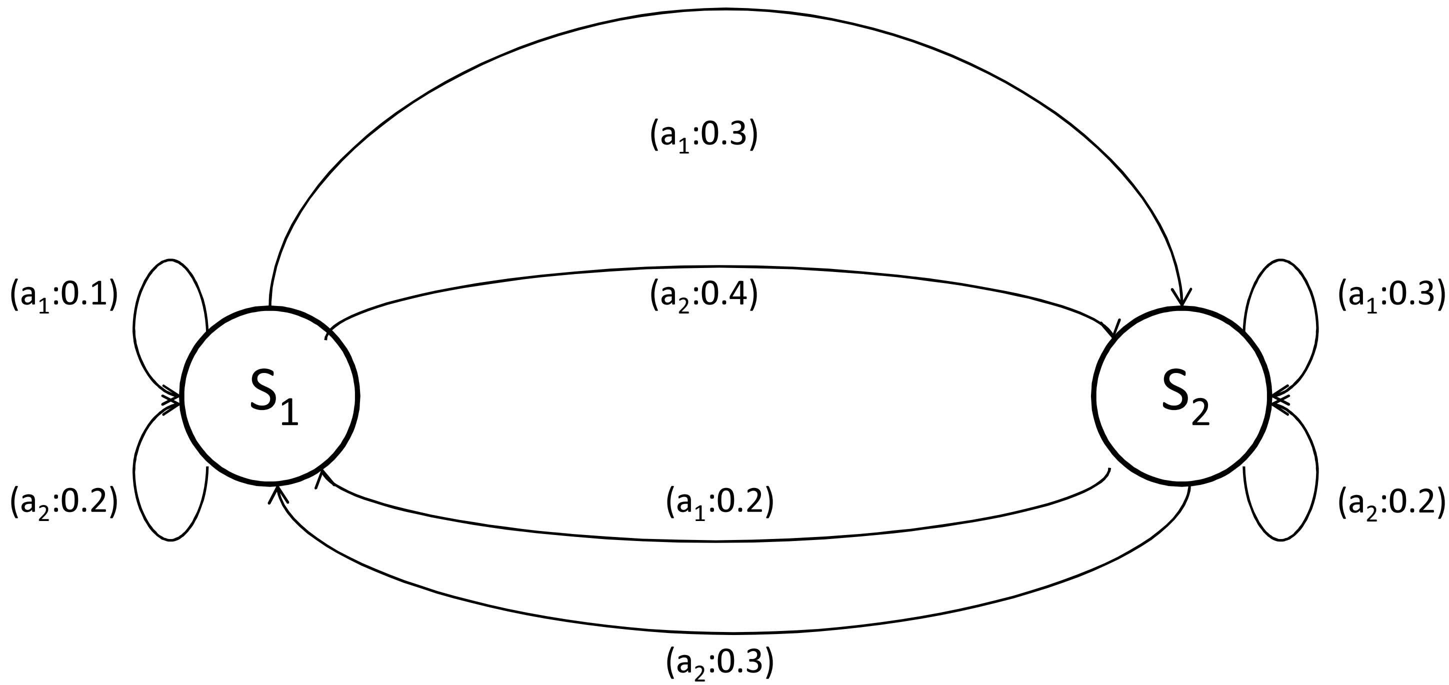
Diagrammatic representation (1/2)



Diagrammatic representation (2/2)

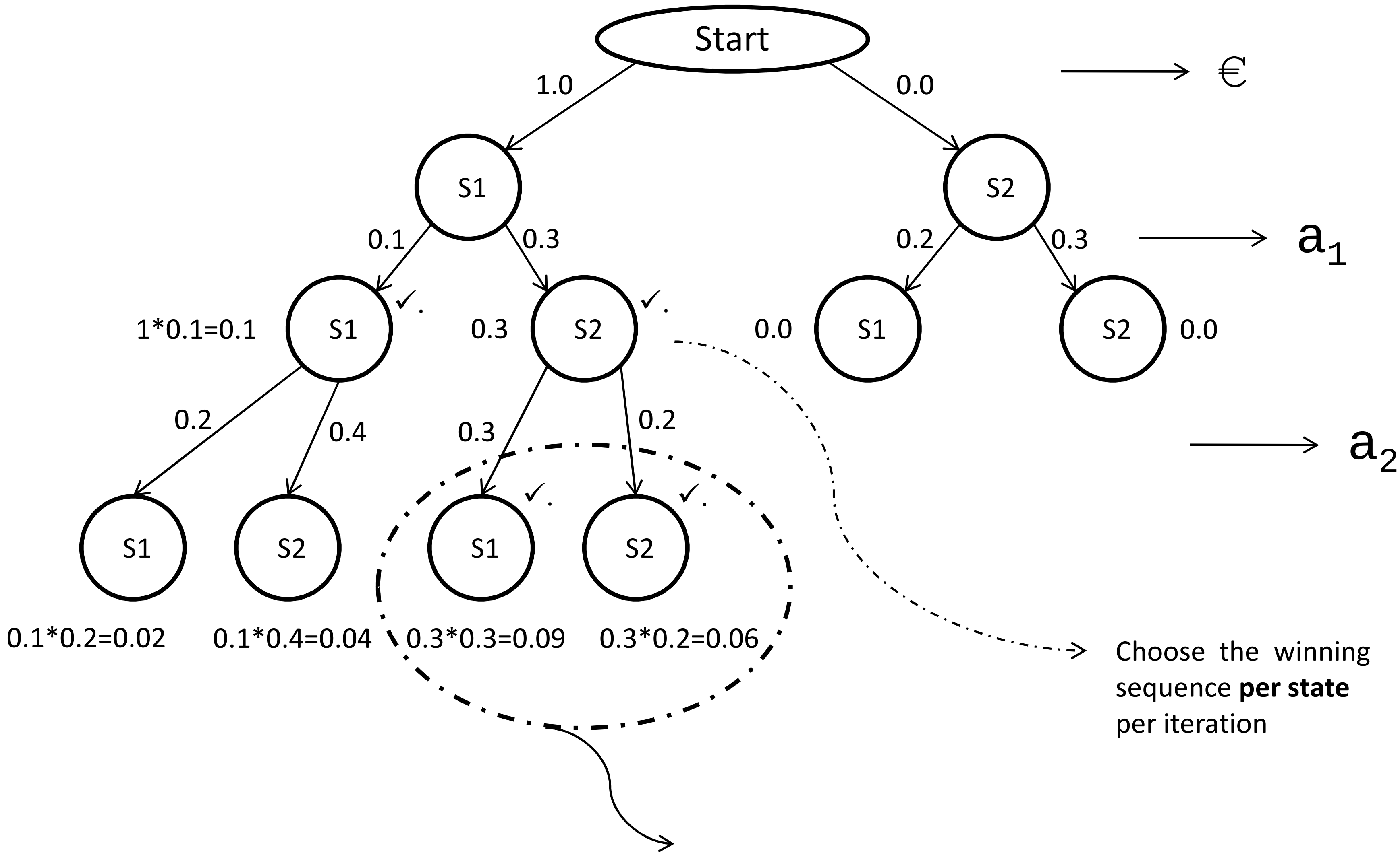


Probabilistic FSM

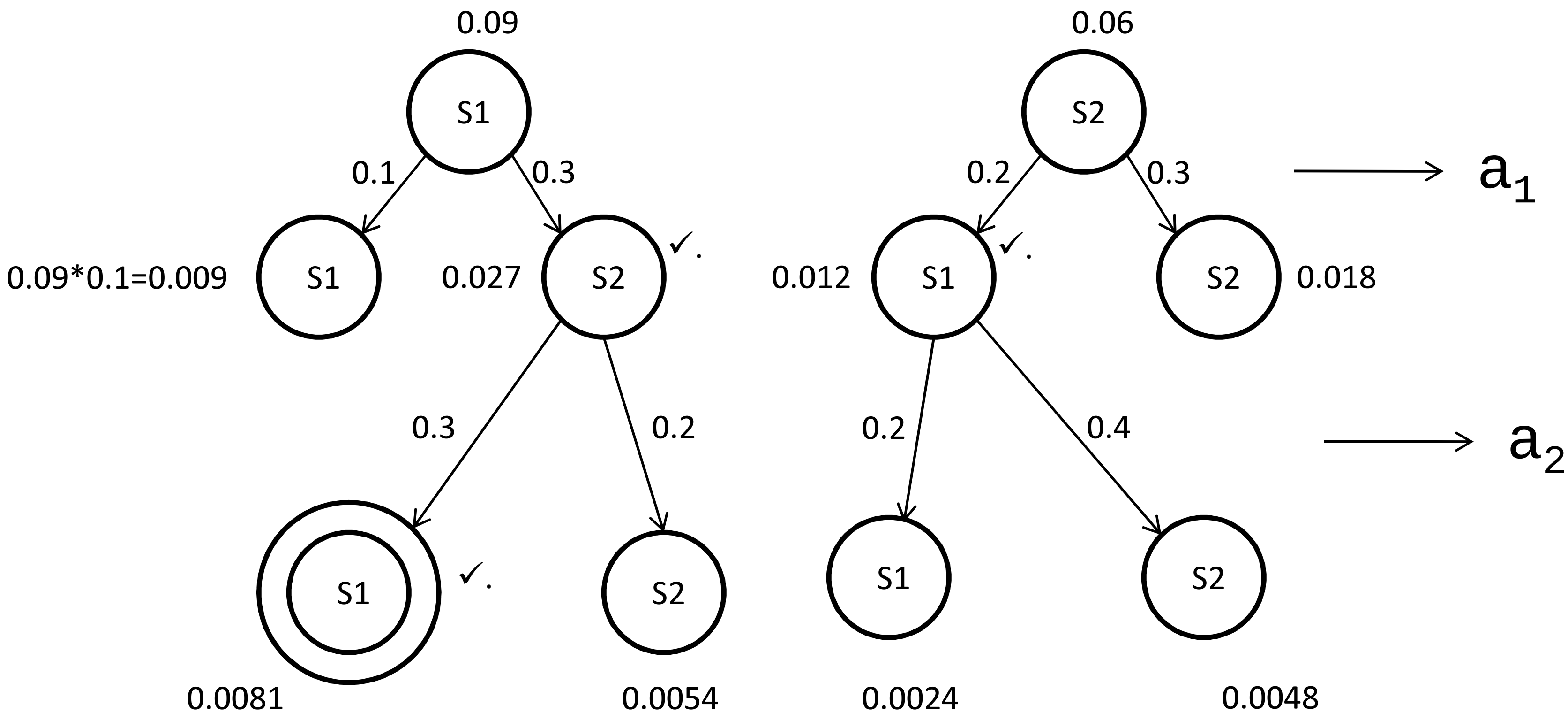


The question here is:
"what is the most likely state sequence given the output sequence seen"

Developing the tree



Tree structure contd…



The problem being addressed by this tree is $S^* = \arg\max_{S} P(S \mid a_1 - a_2 - a_1 - a_2, \mu)$

a1-a2-a1-a2 is the output sequence and μ the model or the machine

Tabular representation of the tree

Latest symbol observed / Ending state	€	a_1	a_2	a_1	a_2
S_1	1.0	(1.0*0.1,0.0*0.2)=(**0.1**,0.0)	(0.02, **0.09**)	(0.009, **0.012**)	(0.0024, **0.0081**)
S_2	0.0	(1.0*0.3,0.0*0.3)=(**0.3**,0.0)	(0.04,**0.06**)	(**0.027**,0.018)	(0.0048,0.0054)

Note: Every cell records the winning probability ending in that state

Final winner

The bold faced values in each cell shows the sequence probability ending in that state. Going backward from final winner sequence which ends in state S_2 (indicated By the 2^{nd} tuple), we recover the sequence.

Algorithm

(following James Alan, Natural Language Understanding (2nd edition), Benjamin Cummins (pub.), 1995

Given:

1. The HMM, which means:
   a. Start State: S_1
   b. Alphabet: $A = \{a_1, a_2, \dots a_p\}$
   c. Set of States: $S = \{S_1, S_2, \dots S_n\}$
   d. Transition probability $P(S_i \xrightarrow{a_k} S_j) \quad \forall_{i,j,k}$
      which is equal to $P(S_j, a_k \mid S_i)$
2. The output string $a_1a_2 \dots a_T$

To find:

The most likely sequence of states $C_1C_2 \dots C_T$ which produces the given output sequence, *i.e.*, $C_1C_2 \dots C_T = \arg\max_{C}[P(C \mid a_1, a_2, \dots a_T, \mu)]$

Algorithm contd…

Data Structure:

1. A N*T array called SEQSCORE to maintain the winner sequence always (N=#states, T=length of o/p sequence)
2. Another N*T array called BACKPTR to recover the path.

Three distinct steps in the Viterbi implementation

1. Initialization
2. Iteration
3. Sequence Identification

1. Initialization

SEQSCORE(1,1)=1.0

BACKPTR(1,1)=0

For(i=2 to N) do

    SEQSCORE(i,1)=0.0

[expressing the fact that first state is S_1]

2. Iteration

For(t=2 to T) do

  For(i=1 to N) do

    SEQSCORE(i,t) = $Max_{(j=1,N)}$

$$[SEQSCORE(j,(t-1)) * P(Sj \xrightarrow{a_k} Si)]$$

    BACKPTR(I,t) = index *j* that gives the MAX above

3. Seq. Identification

C(T) = i that maximizes SEQSCORE(i,T)

For i from (T-1) to 1 do

C(i) = BACKPTR[C(i+1),(i+1)]

Optimizations possible:

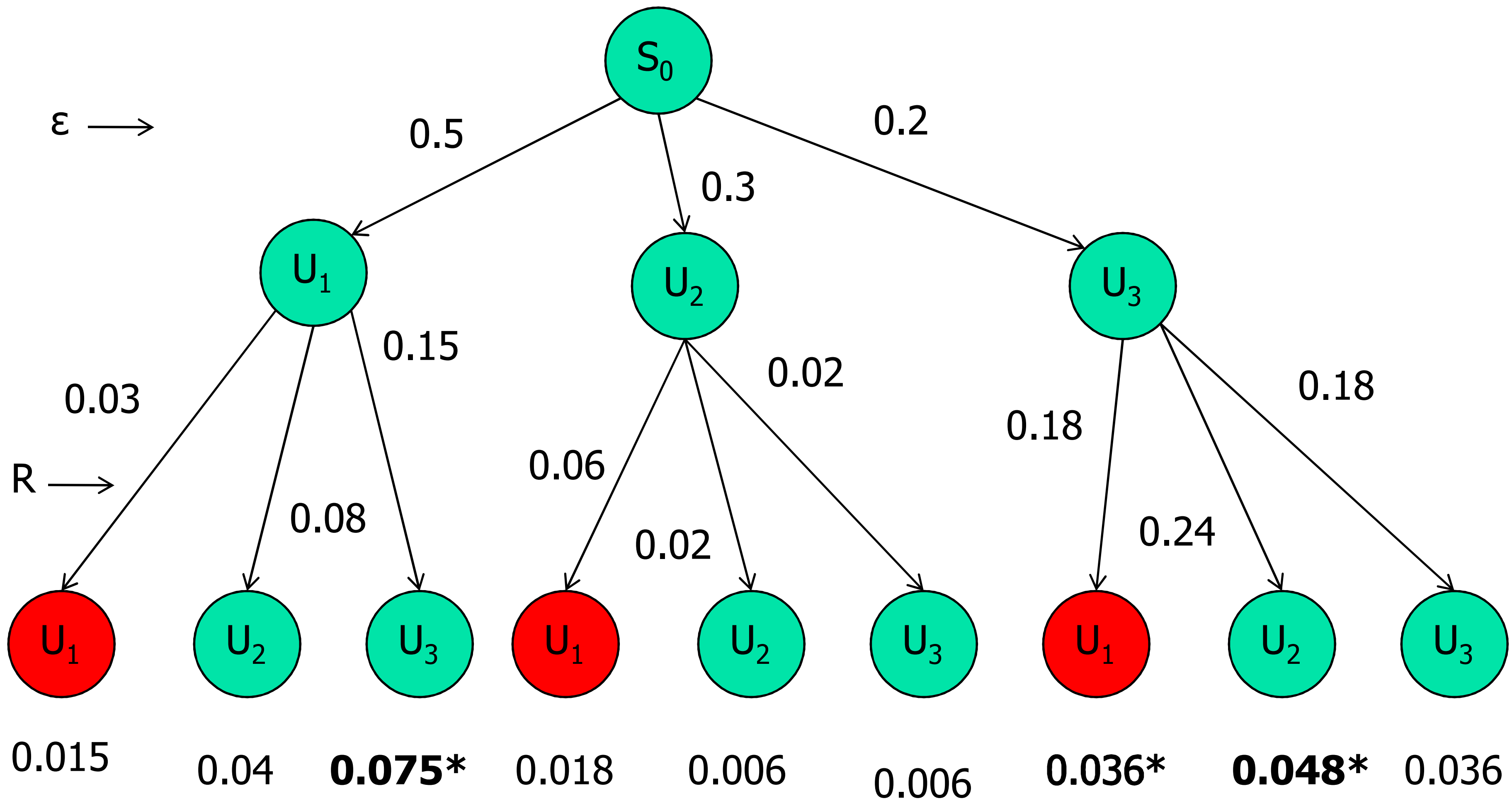
1. BACKPTR can be 1*T
2. SEQSCORE can be T*2

Homework:- Compare this with A*, Beam Search [Homework]

Reason for this comparison:

Both of them work for finding and recovering sequence

Viterbi Algorithm for the Urn problem (first two symbols)



: winner sequences

Markov process of order>1 (say 2)

	O_0	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8	
Obs:	ε	R	R	G	G	B	R	G	R	
State:	S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9

Same theory works

$P(S).P(O|S)$

$= P(O_0|S_0).P(S_1|S_0).$

$[P(O_1|S_1). \quad P(S_2|S_1S_0)].$

$[P(O_2|S_2). \quad P(S_3|S_2S_1)].$

$[P(O_3|S_3).P(S_4|S_3S_2)].$

$[P(O_4|S_4).P(S_5|S_4S_3)].$

$[P(O_5|S_5).P(S_6|S_5S_4)].$

$[P(O_6|S_6).P(S_7|S_6S_5)].$

$[P(O_7|S_7).P(S_8|S_7S_6)].$

$[P(O_8|S_8).P(S_9|S_8S_7)].$

We introduce the states S_0 and S_9 as initial and final states respectively.

After S_8 the next state is S_9 with probability 1, i.e., $P(S_9|S_8S_7)=1$

O_0 is ε-transition

Adjustments

- Transition probability table will have tuples on rows and states on columns
- Output probability table will remain the same
- In the Viterbi tree, the Markov process will take effect from the 3rd input symbol (εRR)
- There will be 27 leaves, out of which **only 9 will remain**
- Sequences ending in **same tuples** will be compared
  - Instead of U1, U2 and U3
  - U_1U_1, U_1U_2, U_1U_3, U_2U_1, U_2U_2,U_2U_3, U_3U_1,U_3U_2,U_3U_3

Forward and Backward Probability Calculation

Forward probability *F(k,i)*

- Define *F(k,i)*= Probability of being in state S_i having seen $o_0 o_1 o_2 \ldots o_k$
- $F(k,i)=P(o_0 o_1 o_2 \ldots o_k , S_i)$
- With *m* as the length of the observed sequence
- $P(\text{observed sequence})=P(o_0 o_1 o_2 .. o_m)$

$$=\Sigma_{p=0,N} P(o_0 o_1 o_2 .. o_m , S_p)$$

$$=\Sigma_{p=0,N} F(m , p)$$

Forward probability (contd.)

$F(k, q)$

$= P(o_0 o_1 o_2 .. o_k , S_q)$

$= P(o_0 o_1 o_2 .. o_k , S_q)$

$= P(o_0 o_1 o_2 .. o_{k-1} , o_k , S_q)$

$= \Sigma_{p=0,N} P(o_0 o_1 o_2 .. o_{k-1} , S_p , o_k , S_q)$

$= \Sigma_{p=0,N} P(o_0 o_1 o_2 .. o_{k-1} , S_p).$

$P(o_m , S_q | o_0 o_1 o_2 .. o_{k-1} , S_p)$

$= \Sigma_{p=0,N} F(k-1,p). P(o_k , S_q | S_p)$

$= \Sigma_{p=0,N} F(k-1,p). P(S_p \xrightarrow{o_k} S_q)$

	O_0	O_1	O_2	O_3 ...		O_k	O_{k+1}	...	O_{m-1}	O_m	
S_0 $\longrightarrow$	S_1	S_2	S_3	...	S_p $\longrightarrow$	S_q		...		S_m	S_{final}

Backward probability *B(k,i)*

- Define *B(k,i)*= Probability of seeing $o_k o_{k+1} o_{k+2} \ldots o_m$ given that the state was S_i
- $B(k,i)=P(o_k o_{k+1} o_{k+2} \ldots o_m \mid S_i)$
- With *m* as the length of the observed sequence
- *P(observed sequence)*$=P(o_0 o_1 o_2 .. o_m)$

  $= P(o_0 o_1 o_2 .. o_m / S_0)$

  $=B(0,0)$

Backward probability (contd.)

$B(k, p)$

$= P(o_k o_{k+1} o_{k+2} \ldots o_m \mid S_p)$

$= P(o_{k+1} o_{k+2} \ldots o_m, o_k / S_p)$

$= \Sigma_{q=0,N} P(o_{k+1} o_{k+2} \ldots o_m, o_k, S_q / S_p)$

$= \Sigma_{q=0,N} P(o_k, S_q / S_p)$
$P(o_{k+1} o_{k+2} \ldots o_m / o_k, S_q, S_p)$

$= \Sigma_{q=0,N} P(o_{k+1} o_{k+2} \ldots o_m / S_q). P(o_k, S_q / S_p)$

$= \Sigma_{q=0,N} B(k+1, q). P(S_p \xrightarrow{o_k} S_q)$

	O_0	O_1	O_2	O_3 …		O_k	O_{k+1}	…	O_{m-1}	O_m		
$S_0 \rightarrow$	S_1	S_2	S_3	…	$S_p \rightarrow$	S_q		…		S_m	S_{final}	