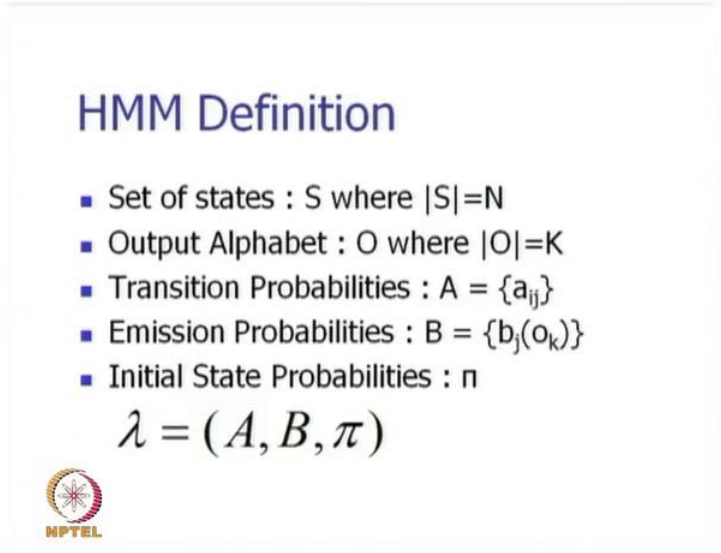
Natural Language Processing
Prof. Pushpak Bhattacharyya
Department of Computer Science and Engineering
Indian Institute of Technology, Bombay

Lecture - 18
HMM, Viterbi, Forward Backward Algorithm

Today, we will concentrate on the algorithmic aspects of HMM. Our focus will be on, how we can actually use the hidden Markov model for the purpose of tagging a sequence, labeling a sequence, doing different kinds of machine learning tasks, which depend on probabilistic techniques, so we continue with the discussion.

(Refer Slide Time: 00:47)



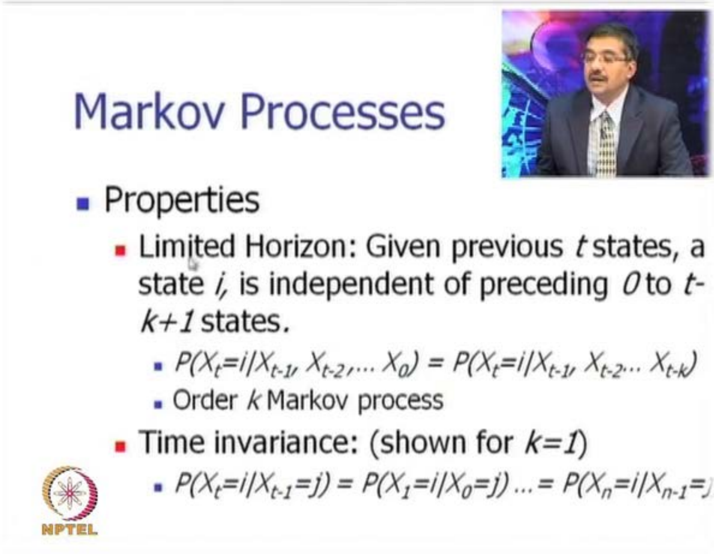
HMM Definition

- Set of states : S where |S|=N
- Output Alphabet : O where |O|=K
- Transition Probabilities : A = $\{a_{ij}\}$
- Emission Probabilities : B = $\{b_j(o_k)\}$
- Initial State Probabilities : π

$$\lambda = (A, B, \pi)$$

And we have seen in the last class that, the hidden Markov model is defined by a set of states S where, the number of states is equal to n. The output alphabet O, where the number the number of output alphabet is K, there is transition probabilities a ij. The meaning of a ij is the row column combination i and j, it is a table after all and the value they indicates the next state for a particular state. The emission probabilities is associated with the output symbols and the initial state probabilities are pi and the complete machine is defined by the initial state probability, the output probability and the transition probability.

(Refer Slide Time: 01:42)



Markov Processes

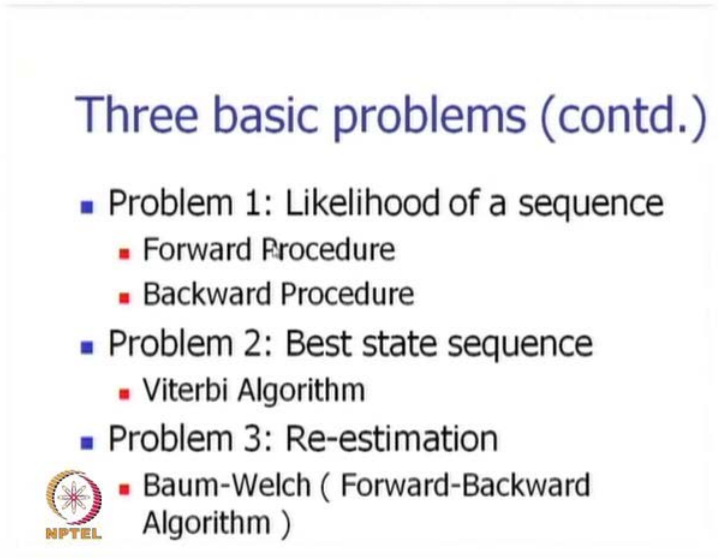
- Properties
  - Limited Horizon: Given previous *t* states, a state *i*, is independent of preceding *0* to *t-k+1* states.
    - $P(X_t=i|X_{t-1}, X_{t-2},\ldots X_0) = P(X_t=i|X_{t-1}, X_{t-2}\ldots X_{t-k})$
    - Order *k* Markov process
  - Time invariance: (shown for *k=1*)
    - $P(X_t=i|X_{t-1}=j)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

Now, there are two very important properties of Markov processes, on which the theory of hidden Markov model rests. The first properties that of limited horizon, so this says that, given previous t sates, a state i is independent of preceding 0 to t minus k plus 1 states. This is expressed mathematically as follows, the t th states is equal to i, conditioned on the fact that, they are having the states X 0, X 1 upto X t minus 1, this is equal to the condition probability X t equal to i, given that previous k states are X t minus 1, X t minus 2 up to X t minus k.

So, this shows that, beyond k states which come before the t th state, we can ignore everything, so this is the limited horizon or window property. And since k states, previous states determine a particular state, we call this Markov process or k Markov process and these possibly is the most important property about the Markov process. The time invariance property, we show it only for k equal to 1 says that, the dependence of the t th state on t minus 1 th state that means, the dependence of a particular state on the previous state is same everywhere.

That means, over the whole sequence, wherever we have X t preceded by X t minus 1, the states being i and j, this probability will not change from place to place over the sequence. Why is it called time invariance, we should remember that hidden Markov model was inspired by processing of speech signals and online speech has an association of time with it. At every time instant, we sample a part of the speech signal and that is, some kind of a wastage of that previous state or affairs, in which hidden Markov model is discussed. Now, we are essentially concentrating on sequences, this sequences may or may not be synchronized with time. But, what is important is that, there is a sequence and the time invariance essentially says that, at this probability is position invariant. This conditional probability has changed from place to place over the sequence.

(Refer Slide Time: 04:28)



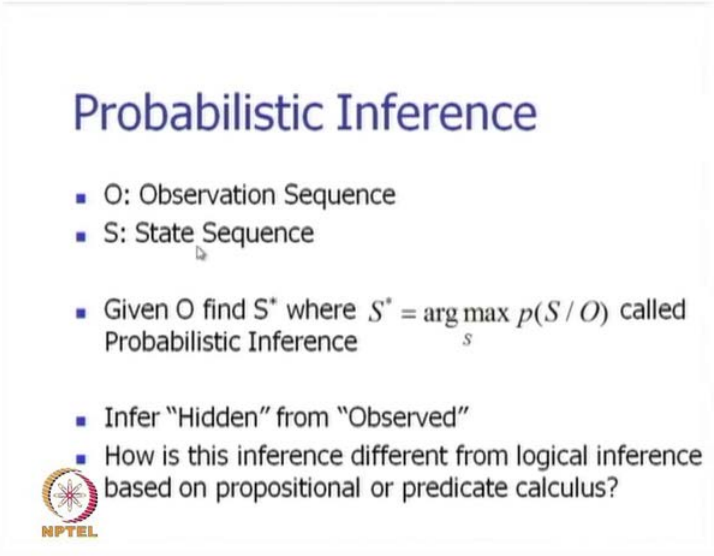
Three basic problems (contd.)

- 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)

So, the three basic problems given in the HMM are the following, first problem is likelihood of a sequence. So, given a output sequence, what is the probability of the sequence, so there is a forward procedure for this and there is a backward procedure, which we will see very soon. Problem 2 is the best state sequence probability, this the famous viterbi algorithm, which we did last time, we will mention this once again. And problem 3 is the re estimation problem, where the algorithm is called to Baum Welch algorithm or forward backward algorithm.

So here, the problem is to get the parameters of the HMM which are nothing but the transition output probabilities. And we know that, for a order 1 Markov process, the transition in output probabilities can be combined and a probability can come on the arg of the machine. So, these are the three problems and we will be concerned with, how to solve these problems algorithmically.

(Refer Slide Time: 05:31)

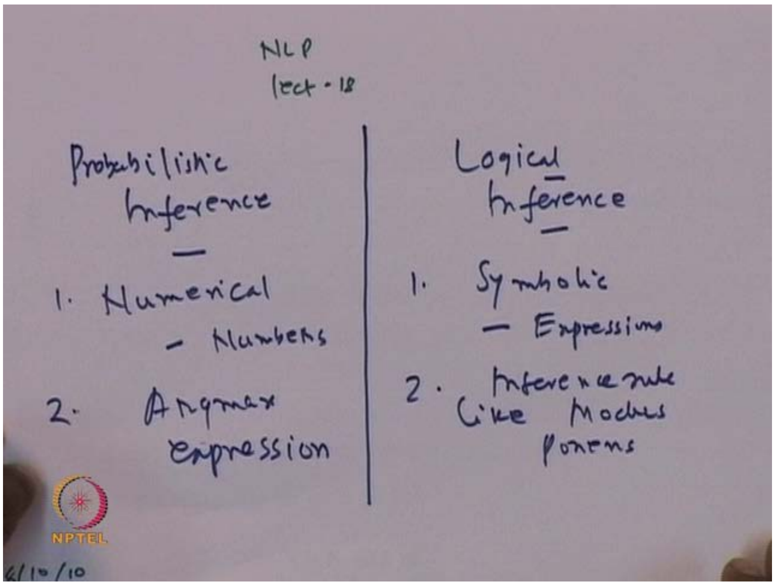


Probabilistic Inference

- 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?

Now, there is a term introduced probabilistic inference, here the observation sequence O is given and the state sequence S is to be found out. So, given O, we need to find out a S star where, S star is a arg max p S O, given O called the probabilistic inference. Now, we call this inference, because we are inferring the hidden state sequence from the observe sequence, so this is also called inference. And we have another kind of inference coming from the domain of logic, predicate calculus and proposition calculus. So, here we find that, the logical inference is based on the logical expressions and their operators there and their inferences. So, let us write down some of the differences that exists between probabilistic inference and the logical inference.

(Refer Slide Time: 06:45)



NLP
lect - 18

Probabilistic Inference

1. Numerical
   - Numbers
2. Argmax expression

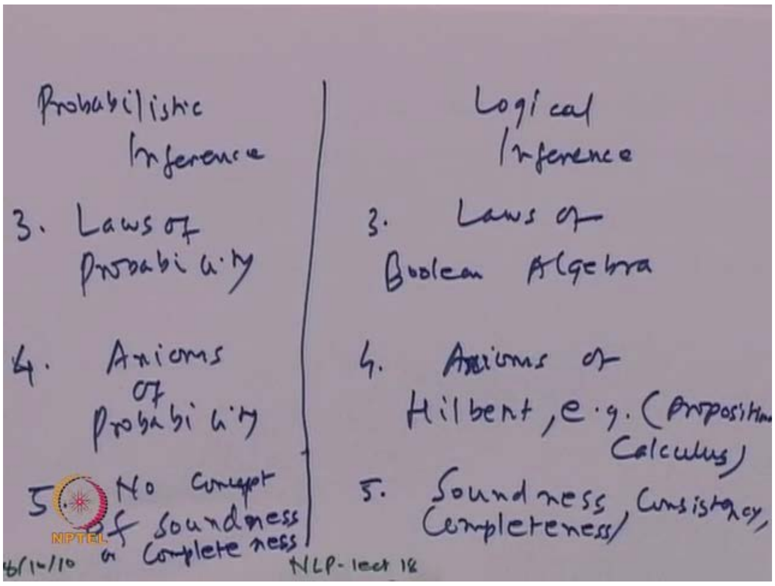
Logical Inference

1. Symbolic
   - Expressions
2. Inference rule Like Modus ponens

6/10/10

So, as we write we can see that, we are putting down probabilistic inference on one side and logical inference on the other side. Probabilistic inference first of all numerical, logical inference is a symbolic, here we work with numbers and here we work with expressions. Inference rule like modus ponens, here the inference rule is typically an arg max expression.

(Refer Slide Time: 07:26)



Probabilistic Inference

3. Laws of probability
4. Axioms of probability
5. No concept of soundness or Completeness

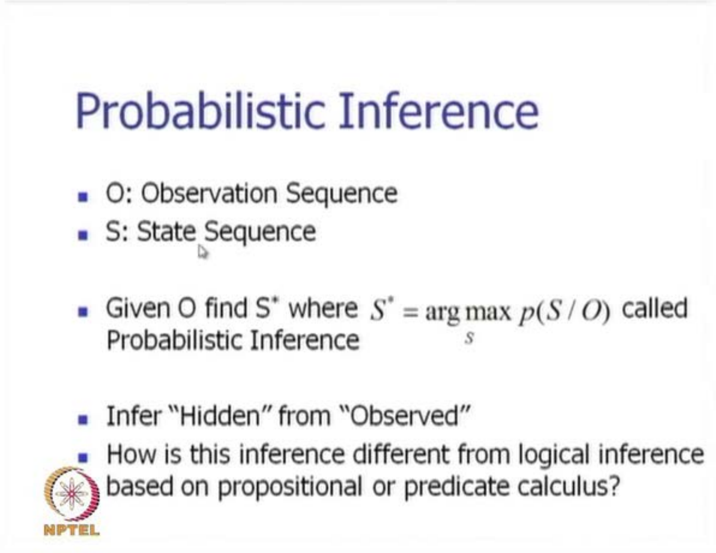
Logical Inference

3. Laws of Boolean Algebra
4. Axioms of Hilbert, e.g. (Propositional Calculus)
5. Soundness, Consistency, Completeness/

6/10/10 NLP- lect 18

We continue these differences, probabilistic inference and logical inference, so we have laws of probability, here we have laws of Boolean algebra. We have axioms of probability, here we have axioms of Hilbert, for example in propositional calculus. Here, soundness consistency completeness are important, here on the other hand no concept of soundness or completeness. So, this is the way it goes, we distinguish between probabilistic inference and logical inference.

(Refer Slide Time: 08:35)

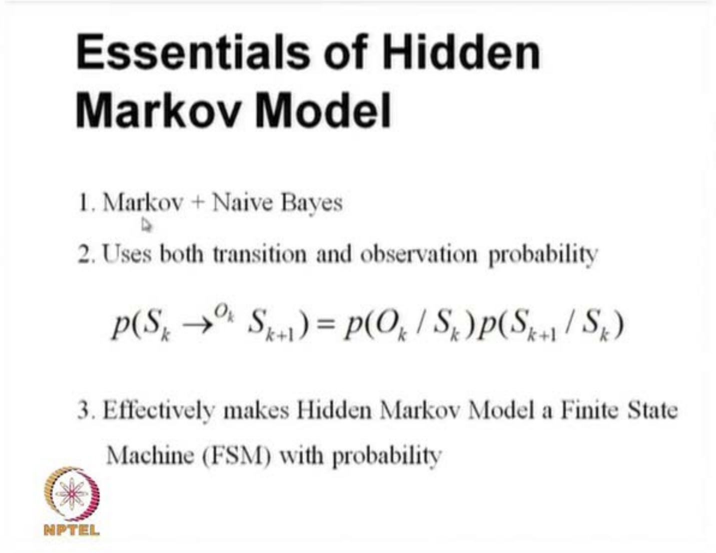


Probabilistic Inference

- 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?

So, we continue with the slides and the probabilistic inference is to be done from arg max competition.

(Refer Slide Time: 08:41)



Essentials of Hidden Markov Model

1. Markov + Naive Bayes
2. Uses both transition and observation probability

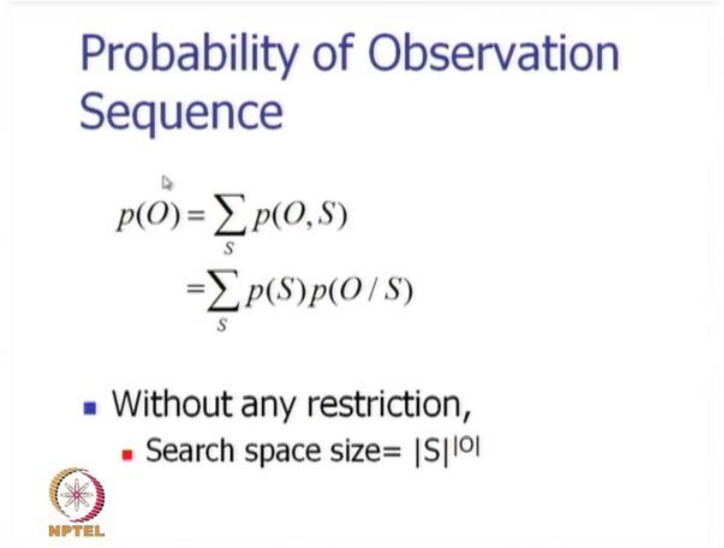
$$p(S_k \rightarrow^{O_k} S_{k+1}) = p(O_k / S_k) p(S_{k+1} / S_k)$$

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

Some very essential facts about hidden Markov model, which we described last time, but we reiterate them in a form of a summary of points. So, Markov plus Naive Bayes is a very powerful idea, very frequently used in artificial intelligence, planning, mission learning, natural language processing and so on. The hidden mark of model uses both transition and observation probability, when it is of order 1, k is equal to 1 and the probability of going from the state S k to S k plus 1 on symbol O k is equal to actually, p O k given S k into p S k plus 1 given S k.

So, this notation is very elegant, very easy to be understood by human beings, but it is mathematical foundation is this, that it is a product of two conditional probabilities, O k given S k and S k plus 1 given S k. This whole thing you can see, can be written as probability of O k comma S k plus 1 given S k. So, the condition variable is S k and the condition variable at 2 O k, the output symbol and the next state, S k plus 1. So, these particular thing when the order is k equal to 1, this effectively makes hidden Markov model a finite state machine with probability.

(Refer Slide Time: 10:13)



Probability of Observation Sequence

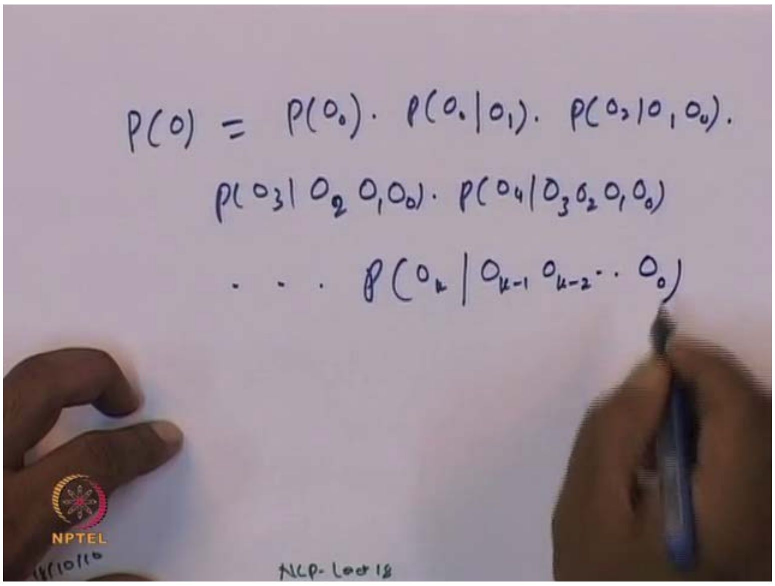
$$p(O)=\sum_{S} p(O,S)$$

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

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

The probability of the observation sequence is p O, you can very easily see cannot be computed from the transition table and the observation table directly. This p O suppose, you are playing a chain rule, the chain rule will give you p O 1 given the O 2, p O 2 given O 1 p. So, it will first p O 1 into p O 2 given p O 1, let me write it down for clarity.

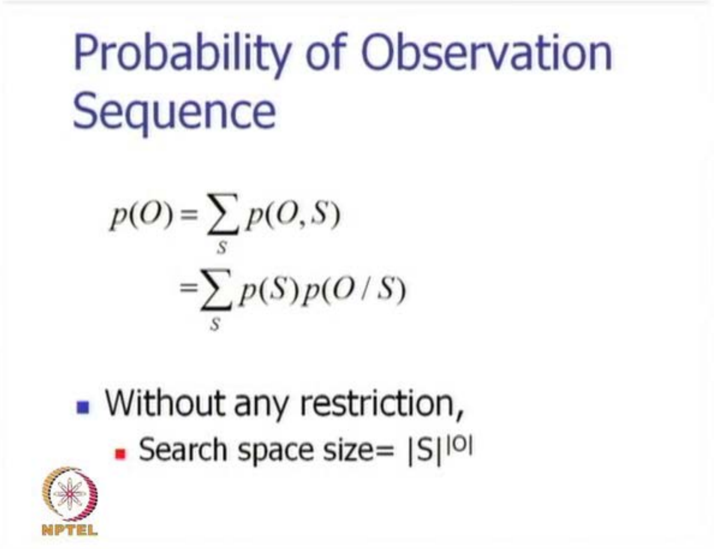
(Refer Slide Time: 10:39)



$$P(O) = P(O_0) \cdot P(O_0 | O_1) \cdot P(O_2 | O_1, O_0) \cdot$$
$$p(O_3 | O_2\, O_1 O_0) \cdot p(O_4 | O_3 O_2 O_1 O_0)$$
$$\cdots P(O_k | O_{k-1}\, O_{k-2} \cdots O_0)$$

So, p O is equal to p O 0 into p O 0 given O 1 into p O 2 given O 1 O 0, into p O 3 given O 2 O 1 O 0, into p O 4 given O 3 O 2 O 1 O 0 and so on, until p O k given O k minus 1, O k minus 2 upto O 0. So, this probability sequence which is found by the application of chain rule is not very useful, because no probability table either transition or observation gives us these parameters, they need to be computed some way.

(Refer Slide Time: 11:32)



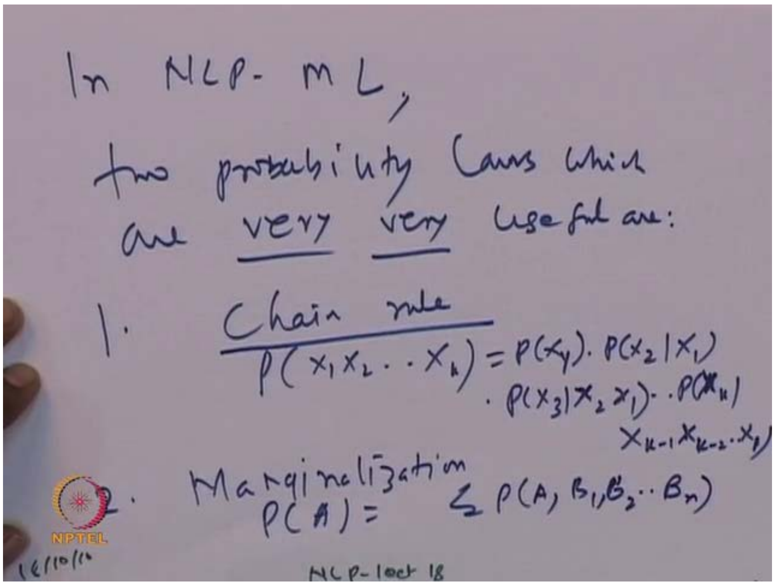
Probability of Observation Sequence

$$p(O) = \sum_S p(O,S)$$
$$= \sum_S p(S)p(O/S)$$

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

So therefore, what is done as we see on the slide, is that p O is written as p O comma S submission over S that means, S is the introduce variable, this process is called marginalization. This is the marginal variable S, we attach this with the O and S can take all possible values, we sum them up and this is equal to the probability expression shown here. Now, this is a very powerful operation from probability, which is useful throughout machine learning AINLP and so on, namely the process of marginalization.

(Refer Slide Time: 12:16)



In NLP- ML,

two probability laws which are very very useful are:

1. Chain rule

$$P(X_1 X_2 \cdots X_k) = P(X_1) \cdot P(X_2|X_1) \cdot P(X_3|X_2 X_1) \cdots P(X_k|X_{k-1} X_{k-2} \cdots X_1)$$

2. Marginalization

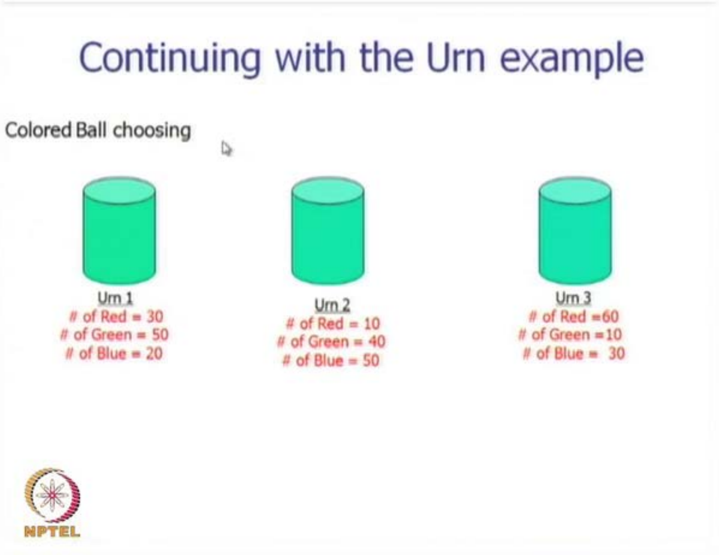
$$P(A) = \sum P(A, B_1, B_2 \cdots B_n)$$

16/10/16

NLP-lect 18

So, let us remember two things, which I would again like to write, in NLP machine learning, two probability laws which are very very useful are, one chain rule which is P X 1, X 2 upto X k is nothing but P X 1 into P X 2 given X 1 into P X 3 given X 2 X 1, P X k given X k minus 1 X k minus 2 upto X 1. This is the first rule which is very important and the second rule is marginalization where, we can say P A is equal to sigma P A comma B 1, B 2 upto B n and all B 1, B 2 upto B n take all possible values. So, these two rules are extremely important in machine learning natural language processing and we have to make use of them often times.

(Refer Slide Time: 13:44)



Continuing with the Urn example

Colored Ball choosing

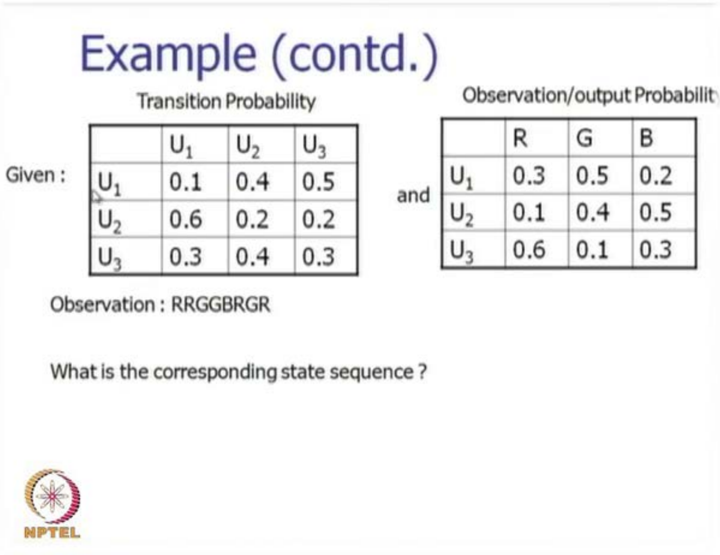
Urn 1
# of Red = 30
# of Green = 50
# of Blue = 20

Urn 2
# of Red = 10
# of Green = 40
# of Blue = 50

Urn 3
# of Red =60
# of Green =10
# of Blue = 30

So, we proceed further now and we continue with the urn example, which was discussed in the last class. So, if you remember, we had three urns urn 1, urn 2 and urn 3, these are three containers, they have distributions of red, green and blue balls within them. So, urn 1 for example, contains 30 red 50 green and 20 blue balls, urn 2 contains 10 red 40 green and 50 blue balls, urn 3 contains 60 red 10 green and 30 blue balls. So, this is the way the urns are organized, so to say, with the balls inside them of different colors.

(Refer Slide Time: 14:30)



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 Probabilit

	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

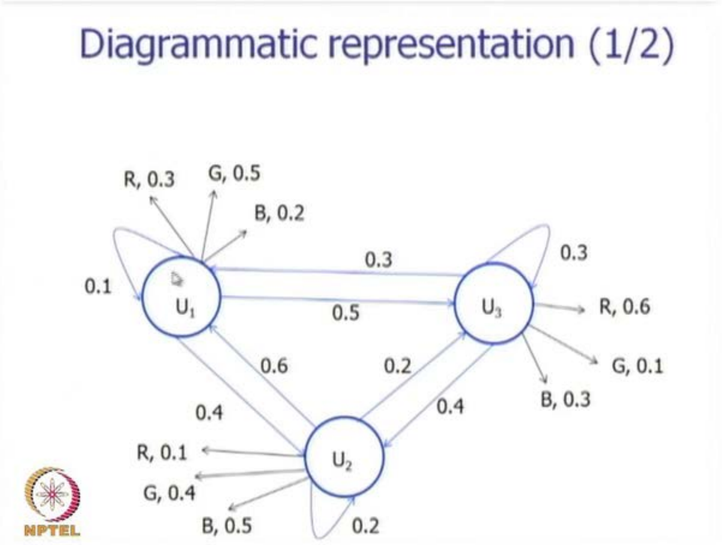
Observation : RRGGBRGR

What is the corresponding state sequence ?

And we know, that the probability of going from one urn to another, one of the urns in the row to a urn in the column is given by these numbers. So, the probability of going from urn U 1 to urn U 2 is 0.1, from U 1 to U 2 0.4, U 1 to U 3 0.5, this is the way these probabilities are organized. Similarly, probability of going to U 1 from U 2 is 0.6, coming back to U 2 is 0.2, coming to U 3 is 0.2 and so on. These are transition probability, observation or output probabilities are functions of their states only.

So, the probability of drawing a red ball from urn 1 is 0.3, because 30 percent of the balls are red, that of drawing a green ball is 0.5, that of drawing of blue ball is 0.2. Now, suppose, we have these observation sequence, we had a red ball, followed by another red ball, green ball, green ball, blue ball, red ball, green ball and then again red ball. Now, our question is, what would be the most probable urns sequence, which produce this and how do you know the best possible urn sequence in terms of probability. We have to compute it from the transition probability and the observation probability.

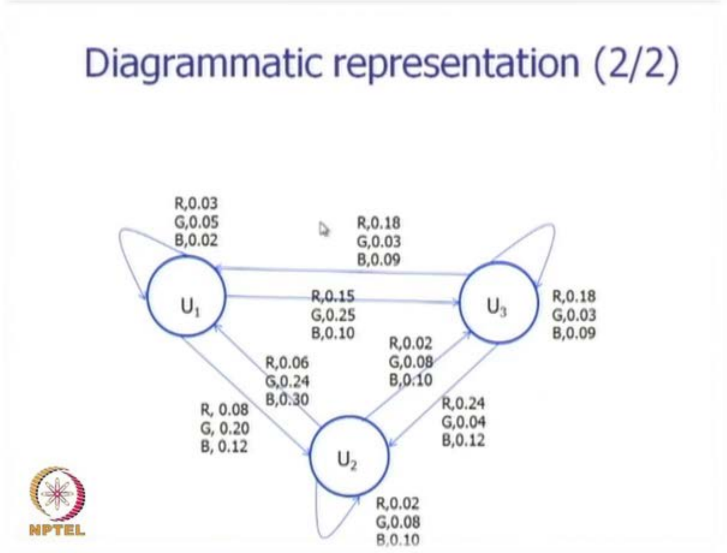
(Refer Slide Time: 15:57)



Diagrammatic representation (1/2)

So, the diagrammatic representation for the transition table probabilities and observation probabilities is this, U 1 can come back to U 1 with 0.1 probability, these are transition probability. U 1 to u 2 is 0.4 these are transition probability, U 2 to U 3 is 0.2 which is shown on the arg. Similarly, from U 1 obtaining the red ball has the probability is 0.3, obtaining green ball has probability 0.5, obtaining a blue ball 0.2. So, these are observation probabilities from each state and the transition probabilities are between the states.

(Refer Slide Time: 16:39)



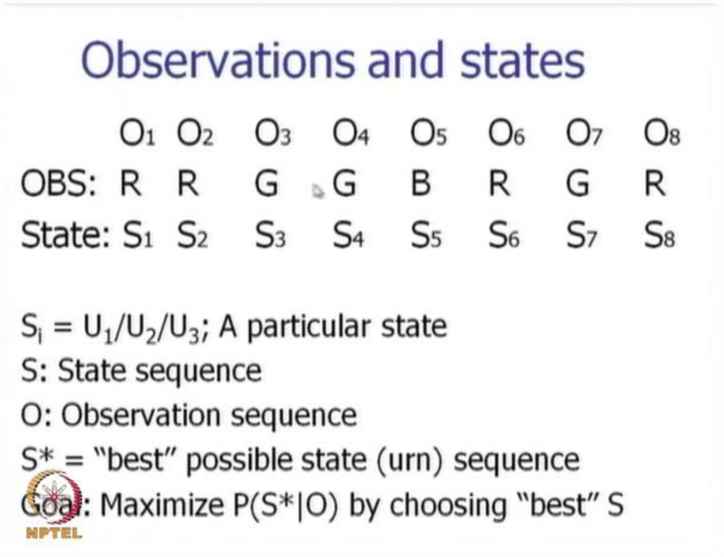
Diagrammatic representation (2/2)

Now, from our mathematical treatment last time where, we applied chain rule, applied Markov assumption, applied base theorem. Applying all these, we could group terms together and we could see that, we did not show the observation probabilities anymore, rather we can observe them into the transition probabilities. And we see that, this U 1 to U 1 with red ball, green ball and blue ball have this probabilities, this probabilities are found by multiplying the transition probability from U 1 to U 1 and the probability of red ball coming from U 1 or a green ball coming from U 1.

So, these probabilities are multiplied, which is a consequence of base theorem Markov assumption and chain rule and so on and then the whole probability of observation is combined with transition probabilities to give these picture. We consider this state transition diagram has only probabilities with respect to the observations marked on the args, so this is like a finite state machine and has these probabilities marked on this.

And actually you can see that, these particular situation of three values coming on an arg is actually equivalent to three args coming back to U 1, U 1 to U 1 on arg U 1 to U 1 on G, U 1 to U 1 on B. So, these are actually three urns similarly, there are these three args for R, G and B and so on. So, we have got this transition, got this probabilistic finite state machine which is actually a hidden Markov model. Let us remember, once again that, it was possible to draw this machine, because of the Markov assumption. The fact that, any state depends only on the previous state, this automated cannot be drawn in the way shown if the dependence of state is on previous two states.

(Refer Slide Time: 18:43)



Observations and states

	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8
OBS:	R	R	G	G	B	R	G	R
State:	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8

$S_i = U_1/U_2/U_3$; A particular state

S: State sequence

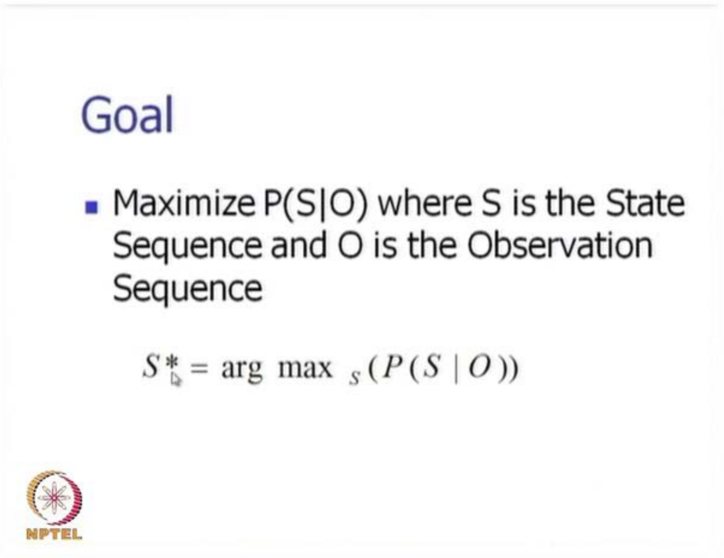
O: Observation sequence

S* = "best" possible state (urn) sequence

Goal: Maximize P(S*|O) by choosing "best" S

Now, we have see that, this is our observation sequence, R R G G B R G R and the states are S 1, S 2, S 3 upto S eight. Each S i can take either U 1 or U 2 or U 3 as the value, S is the state sequence, O is the observation sequence, now S star is the best possible arg sequence.

(Refer Slide Time: 19:05)



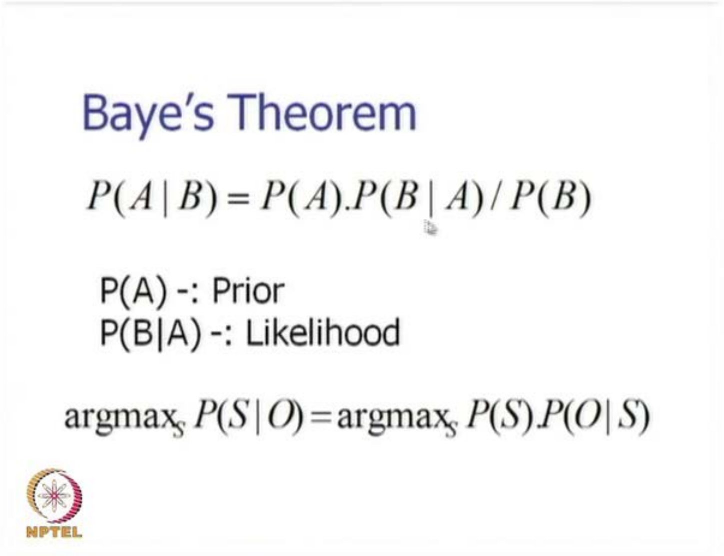
Goal

- Maximize P(S|O) where S is the State Sequence and O is the Observation Sequence

$$S^* = \arg\ \max\ _S(P(S\mid O))$$

Now, the goal is to maximize P S given O where, S is the state sequence and O is the observation sequence. Now, the mathematical expression for this is S star is equal to arg max of P S given O where, it varies over S.

(Refer Slide Time: 19:22)



Baye’s Theorem

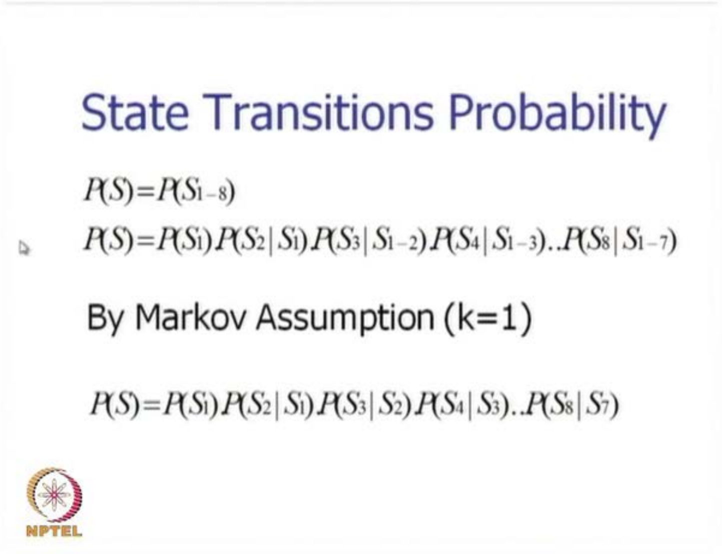
$$P(A \mid B) = P(A).P(B \mid A) / P(B)$$

P(A) -: Prior
P(B|A) -: Likelihood

$$\text{argmax}_S\, P(S \mid O) = \text{argmax}_S\, P(S).P(O \mid S)$$

The Baye’s theorem is applied on this and P A given B is nothing but P A into P B given A into P B where, P A is prior, P B given A is likelihood. And arg max expression now becomes P S given O is equal to nothing but arg max P S into P O given S.

(Refer Slide Time: 19:44)



State Transitions Probability

$$P(S) = P(S_{1-8})$$

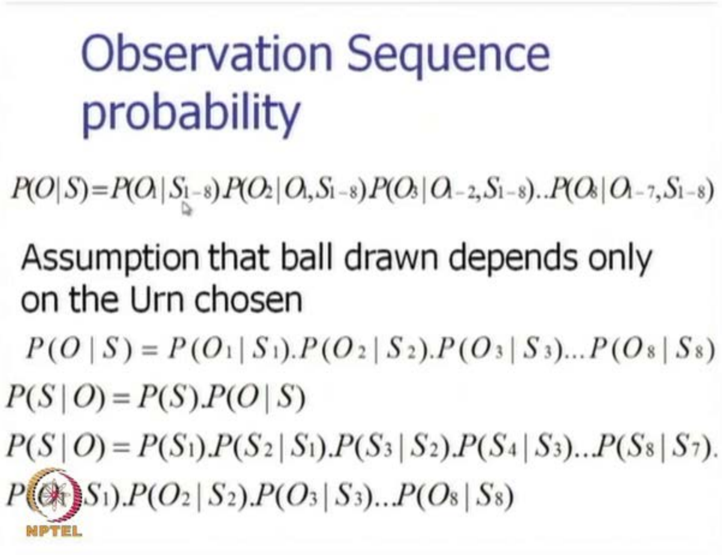
$$P(S) = P(S_1)P(S_2 \mid S_1)P(S_3 \mid S_{1-2})P(S_4 \mid S_{1-3})..P(S_8 \mid S_{1-7})$$

By Markov Assumption (k=1)

$$P(S) = P(S_1)P(S_2 \mid S_1)P(S_3 \mid S_2)P(S_4 \mid S_3)..P(S_8 \mid S_7)$$

The state transition probability can be used for simplifying this expression P S, computing P S, P S is P S 1 to 8, this is equal to P S 1 into P S 2 given S 1, P S 3 given S 2 S 1, P S 4 given S 3 S 2 S 1, P S 8 given S 7 to S 1. By Markov assumption where, k equal to 1, P S equal to P S 1 into P S 2 given S 1, P S 3 given S 2, P S 4 given S 3 upto P S 8 given S 7.

(Refer Slide Time: 20:19)



Observation Sequence probability

$$P(O|S)=P(O_1|S_{1-8})P(O_2|O_1,S_{1-8})P(O_3|O_{1-2},S_{1-8})..P(O_8|O_{1-7},S_{1-8})$$

Assumption that ball drawn depends only on the Urn chosen

$$P(O\mid S)=P(O_1\mid S_1).P(O_2\mid S_2).P(O_3\mid S_3)...P(O_8\mid S_8)$$

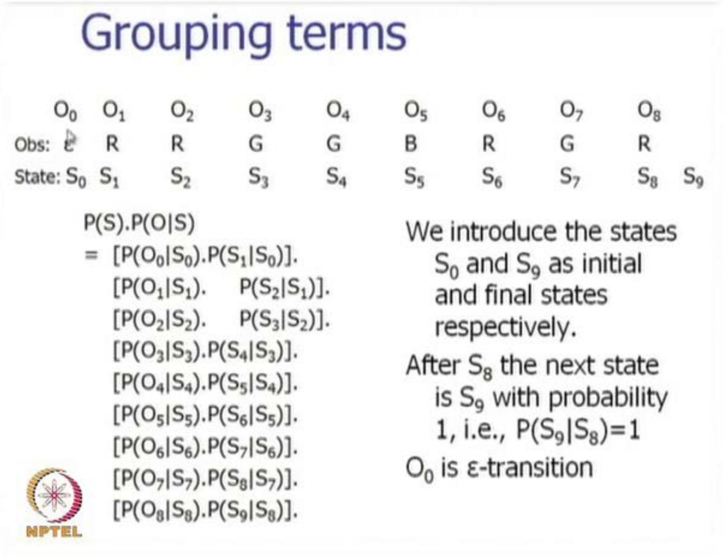
$$P(S\mid O)=P(S).P(O\mid S)$$

$$P(S\mid O)=P(S_1).P(S_2\mid S_1).P(S_3\mid S_2).P(S_4\mid S_3)...P(S_8\mid S_7).$$

$$P(O_1\mid S_1).P(O_2\mid S_2).P(O_3\mid S_3)...P(O_8\mid S_8)$$

The observation sequence probability can be found as P O given S is equal to P O 1 given S 1 to 8, P O 2 given O 1 and S 1 to 8, P O 3 given O 1 to O 2 given S 1 to 8, P O 8 given O 1 to O 7 and S 1 to S 8. Now, we assume that, a ball is drawn independent of any other draw or any urn that is not it is own urn, so P O given S can be written as P O 1 given S 1, P O 2 given S 2 upto P O 8 given S 8. So, P S given O can finally be written as P S 1 into P S 2 given S 1 upto P S 8 given S 7 multiplied by P O 1 given S 1 into P O 2 given S 2, P O 3 given S 3, P O 8 given S 8.

(Refer Slide Time: 21:16)



Grouping terms

	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

$$\begin{aligned}
&P(S).P(O|S)\\
&= [P(O_0|S_0).P(S_1|S_0)].\\
&\quad [P(O_1|S_1).\quad P(S_2|S_1)].\\
&\quad [P(O_2|S_2).\quad P(S_3|S_2)].\\
&\quad [P(O_3|S_3).P(S_4|S_3)].\\
&\quad [P(O_4|S_4).P(S_5|S_4)].\\
&\quad [P(O_5|S_5).P(S_6|S_5)].\\
&\quad [P(O_6|S_6).P(S_7|S_6)].\\
&\quad [P(O_7|S_7).P(S_8|S_7)].\\
&\quad [P(O_8|S_8).P(S_9|S_8)].
\end{aligned}$$

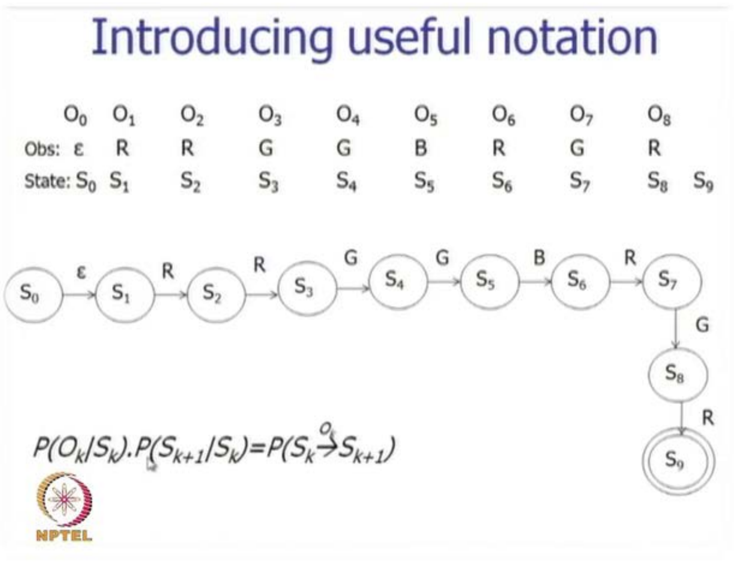
We introduce the states S_0 and S_9 as initial and final states respectively.

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

O_0 is ε-transition

Now, at this stage, we introduce the states S 0 and S 9 as initial and final states respectively, so S 0 is the initial state within epsilon output or input and S 9 is the final state. So, from S 0, the system goes to S 1 with the output of epsilon, so we have here P S into P O given S as P O 0 given S 0 into P S 1 given S 0, P O 1 given S 1 into P S 2 given P S O 2 given S 2, P S 3 given S 02 and so on. And this way, we can arrange a terms and group them together and after a state, the next state is P S 9, so P S 9 given S 8 is 1 therefore, we are not changing the equation at all. And P O naught given S naught also it is fine, because it is value is always 1 and P S naught is again 1, so P S 1 given S naught also can be written.

(Refer Slide Time: 22:22)



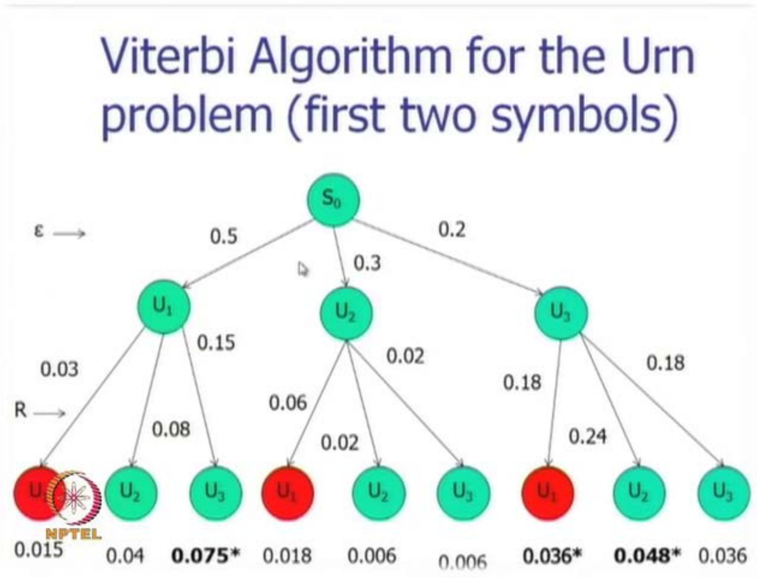
Introducing useful notation

	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

$P(O_k/S_k).P(S_{k+1}/S_k)=P(S_k \xrightarrow{O_k} S_{k+1})$

Now, after grouping up these terms and introducing these notation here, P O k given S k into P S k plus 1 given S k is P S k going to S k plus 1 with symbol O k. So, here what happens is that, the whole transition probability and observation probability, all these are combined and the whole observation sequence and the states sequence can be looked upon as a finite state machine, which states S O to S 9 and on each arg we have the symbols epsilon R R G G B R and G R. This is actually the observation sequence and our goal is to find out the best possible state sequence or the best possible urn sequence.

(Refer Slide Time: 23:12)

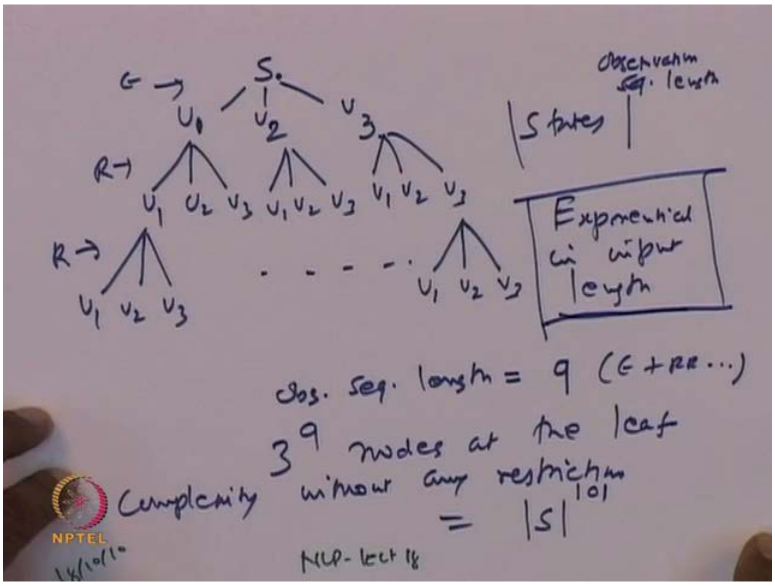


Viterbi Algorithm for the Urn problem (first two symbols)

So, now that we know that, this finite state machine with probabilities is in place, we can make use of this tree diagram, which expresses the famous viterbi algorithm for the urn problem. Here, we show the tree only for the first two symbols, the tree gets further developed as more and more symbol is coming. So, our first symbol is epsilon, second symbol is a R, third symbol is another R, fourth symbol is G and so on, so the tree will gradually grow.

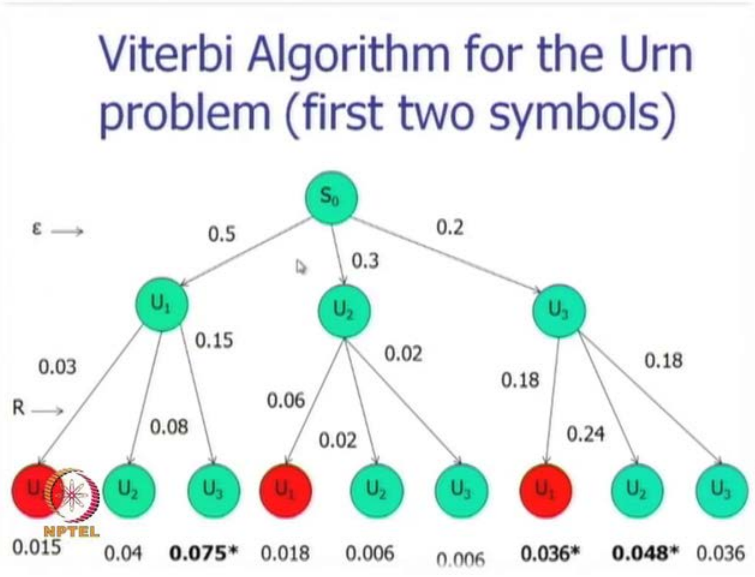
Now, as you can see on the tree from the initial state, the three possibilities are U 1, U 2 and U 3. Similarly, from U 1 you have three state, U 1 U 2 and U 3, from U 2, U 1 U 2 U 3 and U 3 again U 1 U 2 U 3. So, if this is the way the tree is grown then let us see how much of calculation is required. Let us write it down on paper and see, how much of calculation we need.

(Refer Slide Time: 24:16)


ε →
S₀
U₀
U₂
U₃
R →
U₁ U₂ U₃ U₁ U₂ U₃ U₁ U₂ U₃
R →
U₁ U₂ U₃
U₁ U₂ U₃
|States| observation seq. length
Exponential in input length
Obs. seq. length = 9 (ε + RR...)
3⁹ nodes at the leaf
Complexity without any restriction
= |S|^|O|
NPTEL
18/10/10
NLP- lect 18

So, from S 0 we have U 0 U 1 U 2 epsilon symbol, from U 0 again U 1 U 2, from U 1 you have U 1 U 2 U 3, from U 2 again U 1 U 2 U 3, from U 3 U 1 U 2 U 3. From here, again three children U 1 U 2 and U 3 and so on until at the last child we have again, U 1 U 2 U 3 and here, the symbol was R, another symbol is R here. So, if we grow the tree this way, suppose the observation sequence length is now 9 in our case, this is epsilon plus R R G R so on, 9. So, after epsilon, we have R level, after R we have second level and after third, we have the third level. So, at each level, we multiply the number of node in the previous level by 3, so first level, at the 0 th level we have only one node S 0. At the first level, we have 3, so 3 to the power 0, 3 to the power 1, here we have 3 to the power 2, here we have 3 to the power 3. So, after the final symbol, we will have 3 to the power 9 nodes at the leaf, so we can say that, complexity without any restriction is equal to the number of states to the power length of the observation sequence. So, number of states to the power length of observation sequence, so states to the power observation sequence length. So, this is a enormous complexity, exponential in the length of input in absence of the Markov assumption and viterbi.

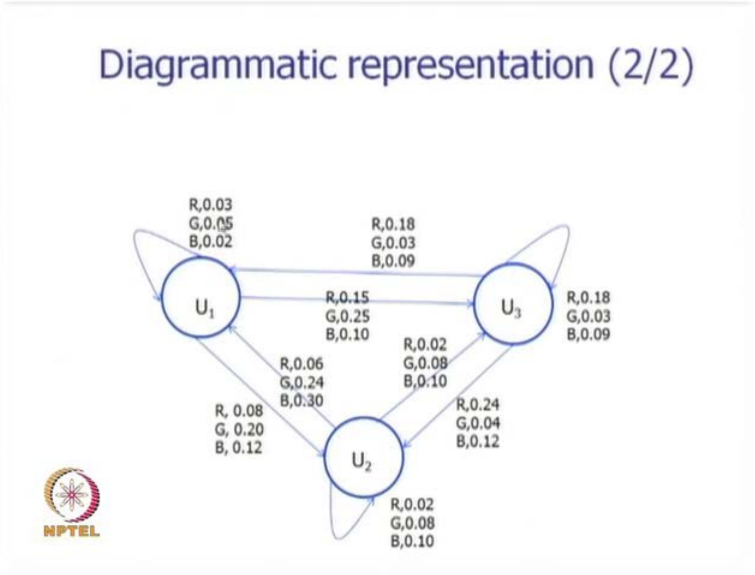
(Refer Slide Time: 26:44)



Viterbi Algorithm for the Urn problem (first two symbols)

However, we see on the slide that, this will not be the case, this is the main point of the viterbi algorithm. Because of the Markov assumption, any state will depend only on the previous state. So now, we start with S 0 go to U 1 then go to U 1, so this sequence is U 1 U 1 and the probability here is 0.03 into 0.5, which is 0.015. The probability at this leaf is 0.5 into 0.08, which is 0.04, at this leaf the probability is 0.5 into 0.15, which is 0.075. Here, the sequence is U 2 U 1 and the probability is 0.6 into 0.3, which is 0.018 and so on. So, we compute the probabilities of the sequences by multiplying the probabilities, which are obtained here. Now, these probabilities where do they come from, these probabilities actually are given by our transition table.

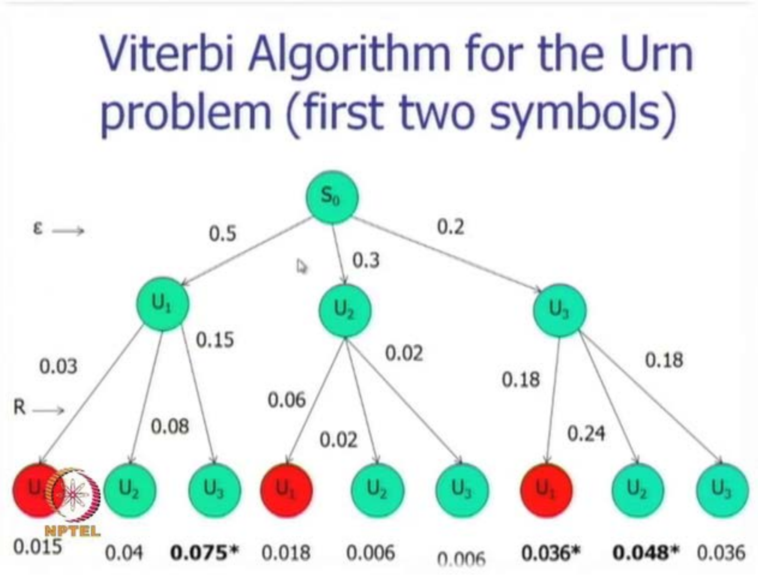
(Refer Slide Time: 28:09)



Diagrammatic representation (2/2)

This probability you want to U 1 to U 1 R, we have already seen it in one of the previous slides, which is this slide you see. So, U 1 to U 1 on R is 0.03 similarly, U 1 to U 2 on green is 0.24, so we have recorded all these in the viterbi tree.

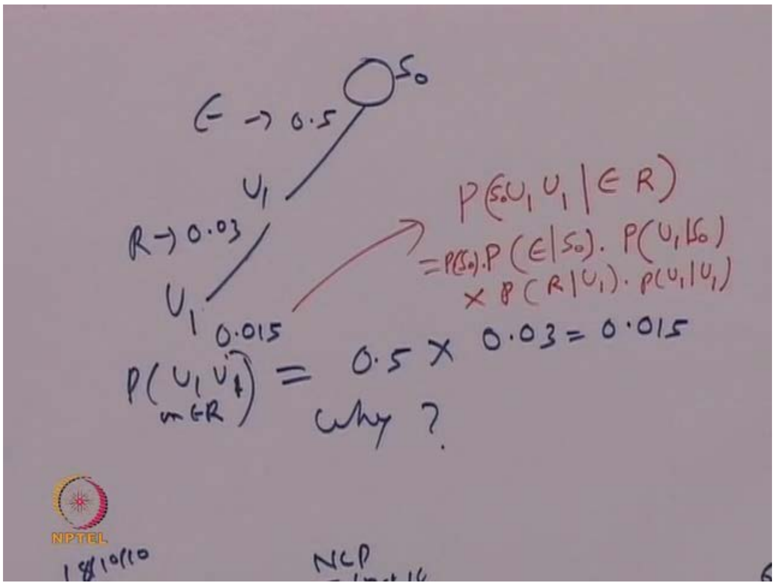
(Refer Slide Time: 28:27)



Viterbi Algorithm for the Urn problem (first two symbols)

So, R is a symbol here and everything has to be with respect to R, so U 1 to U 2 on R is 0.08. So, these things are recorded here, now why is it we can multiply these probabilities, how can we multiply these probabilities, so let us look at the mathematics for this.

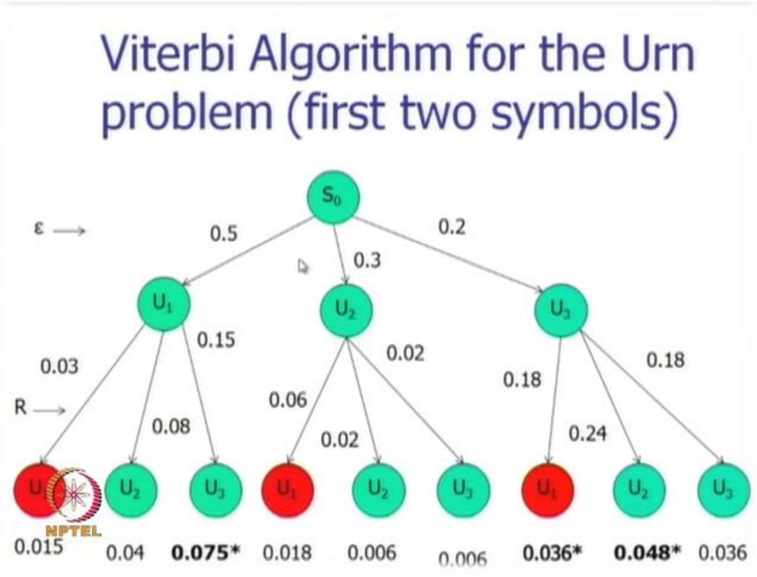
(Refer Slide Time: 28:57)


S_0
$\epsilon \rightarrow 0.5$
U_1
$R \rightarrow 0.03$
U_1
0.015
$P(S_0 U_1 U_1 | \epsilon R)$
$= P(S_0) . P(\epsilon | S_0) . P(U_1 | S_0)$
$\times P(R | U_1) . P(U_1 | U_1)$
$P(U_1 U_1 \text{ on } \epsilon R) = 0.5 \times 0.03 = 0.015$
why ?
18/10/10
NCP

So, if I take apart of the tree S 0 and we have U 1 again U 1 and this is on epsilon 0.5 and on R, 0.03. This probability of U1 U 1, which is a probability recorded here 0.15, we are saying is nothing but 0.5 into 0.03 which is 0.015. So, the question is why, why cannot we multiply these probabilities, the reason is that what is the quantity measured here, the quantity which is measured here is p U 1 U 1, p U 1 U 1 sequence on R, on epsilon R. Or in other words, this particular probability is nothing but probability of U 1 U 1, the state sequence U 1 U 1 given epsilon R.

So, this is nothing but the probability of U 1 U 1, it should be S 0 U 1 U 1 this is nothing but probability of epsilon given S 0 into probability of U 1 given S 0, probability of S 0 into probability of R given U 1 into probability of U 1 given U 1. By applying Markov assumption, base theorem, etcetera and that is why, this probability comes out to be the product of the probabilities on this R. So, we are correct in multiplying these probabilities values.

(Refer Slide Time: 30:50)



Viterbi Algorithm for the Urn problem (first two symbols)

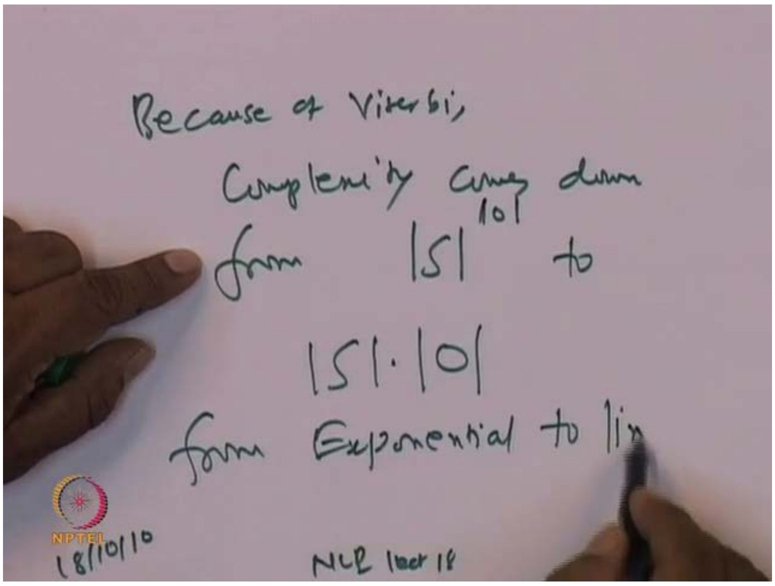
So, proceeding further, as we look at the slides, now we do the most important step in the viterbi algorithm. We notice, which are the sequences which ended U 1, here is a sequence which ends in U 1, this is S 0 U 1 U 1 and this is S 0 U 2 U 1 and S 0 U 3 U 1. Now, we take all these three sequences and find out, which probability is the highest, amongst all this sequences which ends in U 1. So, this probability is 0.015, this is 0.018, this is 0.036.

So, all those sequences which end in U 1, amongst them this is the winner sequence, so the sequence U 3 U 1 has the highest probability amongst all those sequences, which ended in U 1. So, what u will do is that, we will cancel out these particular node and this particular node, because later also when we introduce their children, the child of U 1 here will be U 1 U 2 U 3, here again it will be U 1 U 2 U 3 and again here will be U 1 U 2 U 3.

Now, you see everywhere will be multiplying the probability of U 1 given U 1 or probability of U 2 given U 1 or probability of U 3 given U 1. So, the same quantity we multiply to these subsequent probabilities and these sequences will never be able to overcome the values obtained under the sub tree here. So, everything that is the subtree of U 1 here and U 1 here will always be less than the strings produced under U 1 here. Therefore, there is no point advancing these notes, because they will never be the winner sequences.

Therefore, we will only survive the node here, similarly out of all those sequences which end in U 2 will survive only this node, others have no chance of winning later. And for U 3, this will be the surviving node, because this U 3 and this U 3 here, their probabilities values are less. So, out of these n nodes, only three will survive and we only develop those three, so that three nodes here this one, this and this one. So, again they will be advanced and they will produce 9 nodes, again we will pick up those sequences, which end in a particular state U 1 U 2 or U 3 and only survive them. So, at every stage, we keep only three nodes for further development and therefore, at the end of the observation sequence, we will have only three nodes at the leaf level. Prior to that, we had three nodes again, prior to that again three, so the total number of nodes in these tree will be 3 into 8.

(Refer Slide Time: 34:12)



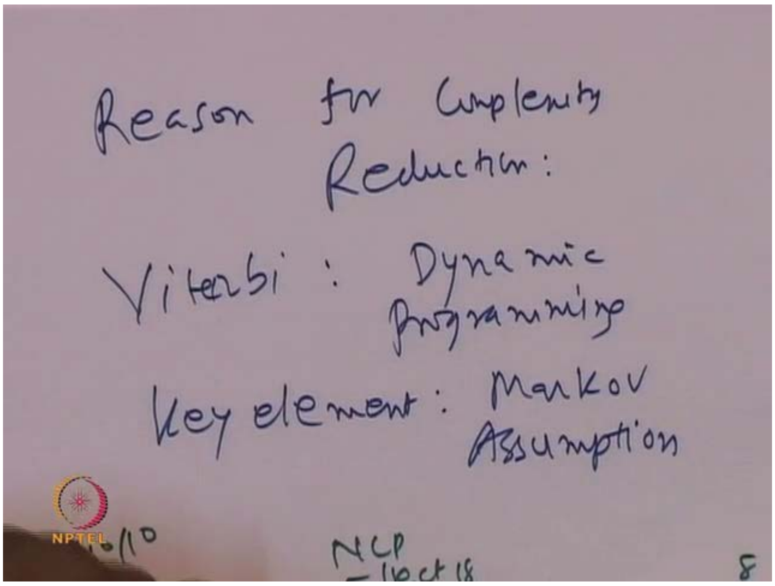
Because of Viterbi,
Complexity comes down
from $|S|^{|O|}$ to
$|S| \cdot |O|$
from Exponential to lin

18/10/10

NLP lect 18

So, what we find here is that, we have a tremendous amount of complexity saving, because of viterbi complexity comes down from S to the power O to S into O.

(Refer Slide Time: 34:36)



Reason for Complexity Reduction:

Viterbi : Dynamic Programming

Key element : Markov Assumption

16/10

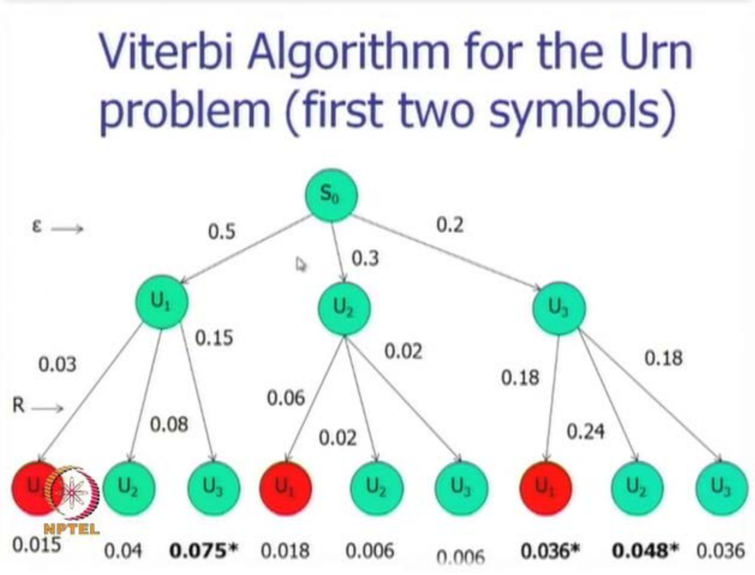
NLP
16 Oct 18

8

So, from exponential to linear and the reason, why this happens the reason for complexity degradation, reason for complexity reduction is viterbi, which is based on dynamic programming. And the key element is the Markov assumption, because of markov assumption what happens is that, a state depends only on the previous state and in S sequence, we find that only those sequences which end in a particular state, amongst them only one of them needs to be retained.

So, this is a very important idea in artificial intelligence, statistical natural language processing. Viterbi algorithm is an old algorithm, independently discovered in the 1960’s, 1970’s in many different branches of electrical engineering and even economics and what we find is that, this also has application in statistical natural language processing.

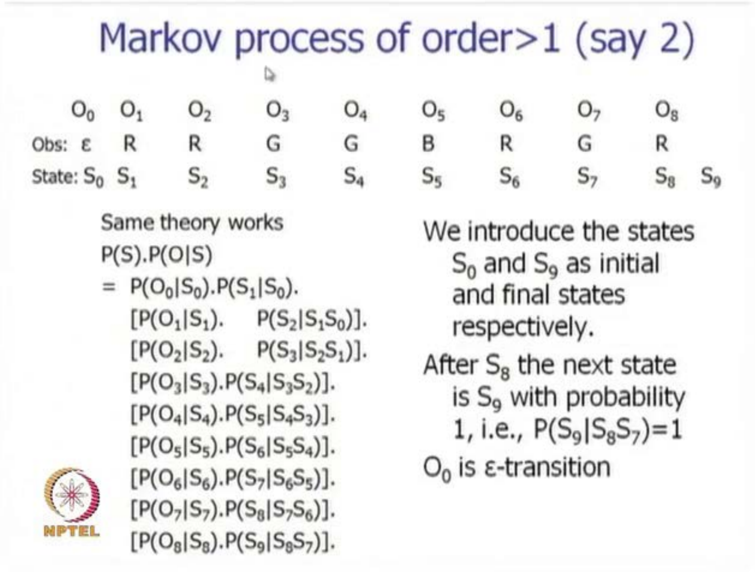
(Refer Slide Time: 35:46)



Viterbi Algorithm for the Urn problem (first two symbols)

Now, we will see that, this viterbi algorithm can be implemented quite elegantly by making use of some important data structures.

(Refer Slide Time: 36:01)



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

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

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

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

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

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

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

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

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

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

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

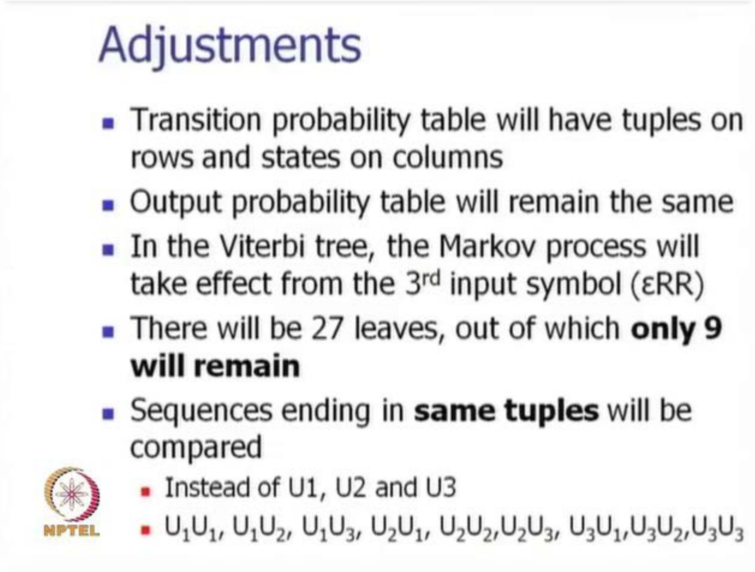
We introduce the states S_0 and S_9 as initial and final states respectively.

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

So, the same theory works when we have a Markov process of order greater than 1 say, the order is 2, the observation sequence remains the same, epsilon R R R G G B R G R, which we call O 0, O 1, O 2 upto O 8 and the state sequence is S 0, S1, S 2, S 3 upto S 8. If final state S 9 is introduced to signify termination and again we can do these application of base theorem and grouping of terms. So, P S into P O given S is P O naught given S naught into P S 1 given S naught into P O 1 given Ss 1 into P S 2 given S 1 S naught, P O 2 given S 2 into P S 3 given S 2 S 1 into P O 3 given S 3 into P S 4 given S 3 S 2. So, see how, the conditioning part now involves two states, instead of only one state and O naught is nothing but the epsilon transition. P O naught given S naught will be 1, P S 9 given S 8 and S 7 will be equal to 1, P S naught given P S 1 given S naught is the initial probability, so this is the Markov process of order greater than 1.

(Refer Slide Time: 37:27)



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

And the adjustments which are meant to the basic theory are the following, the transition probability table will have tuples on rows and states on columns. Now, since every states depends on previous two states, our transition table will not have U 1 U 2 U 3 on the row and columns U 1 U 2 U 3, on instead it will have U 1 U 2 U 3 on the column signifying the next 8, but the rows would be tuples U 1 U 2 U 1 U 2 U 2 U 3 U 3 U 1 and so on.

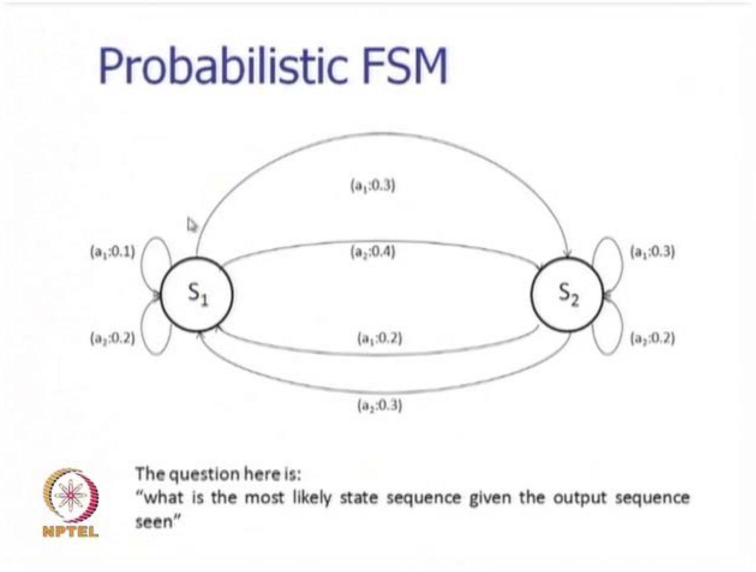
So, the transition probability table will have tuples on rows and states on columns, so if it is a Markov process of order k, the tuples will be k tuples on the rows. Output probability table however, will remain the same, because the output probability depends upon only on the state of itself. In the viterbi tree, the Markov process will take effect from the third input symbol, after epsilon R R before that, we will not be able to eliminate any of the nodes.

After these three symbols epsilon and R R, there will be 27 leaves, out of which only 9 will remain, sequences ending in same tuples will be compared. Why only 9 out of 27 leaves will survive, because sequences ending in the same tuples will be compared. Instead of U 1 U 2 and U 3, we will have to compare sequences which end in U 1 U 1, all those sequences which end in U 1 U 1, so there will be three such sequences. Similarly, all the sequences which end in U 1 U 2 and the one with the highest probability will be retrained.

Then, such tuple is U 1 U 3, will look at all the sequences which end in U 1 U 3 and only the sequence with the highest probability will be retrained. And this is the way it will go, finally there will be 9 nodes out of 27, which will survive and these nine will be again advanced giving rise to another 27 nodes, again we will look at these ending tuples and retain nine nodes out of this. So, when we do it this way, the complexity you can see will be the S square into length of the input symbol. So, the complexity will increase as per the Markov process order which is k and k will be the exponent of the states.

(Refer Slide Time: 40:02)

Probabilistic FSM

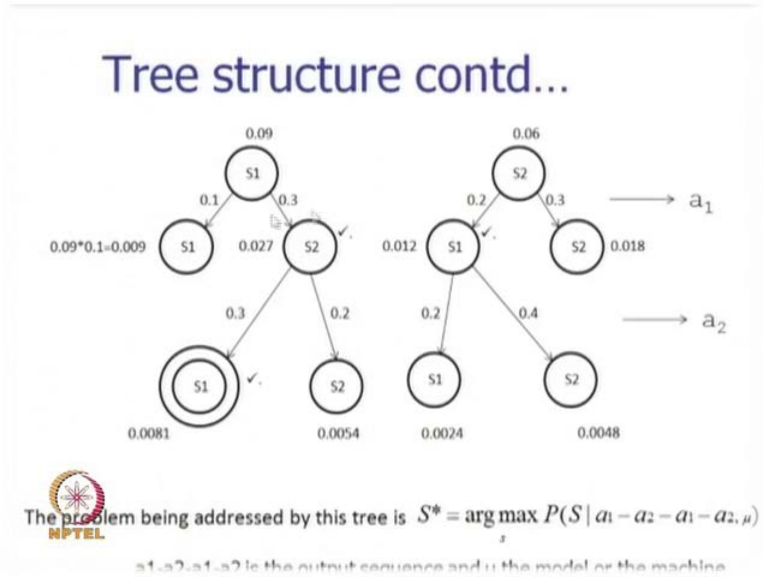


The question here is:
"what is the most likely state sequence given the output sequence seen"

Now, to illustrate the implementation of this veterbi algorithm, we take much simpler probabillistic finite state machine. We have two states S 1 and S 2 here and the two symbols are a 1 and a 2, the state S 2 goes to S 1 on symbol a 1 and the probability of that happening is 0.01. So, probability of S 1 to S 1 on symbol a 1 is 0.1, probability of S 1 to S 1 on symbol a 2 is 0.2, probability of S 1 to S 2 on symbol a 2 is 0.4, probability of S 1 to S 2 on symbol a 1 is 0.3.

Let us remember that, these probability values are actually combination probabilities, is a product of observation probability given a state into transition probability of a state given the previous state. So, here is the question is, what is the most likely state sequence given output sequence, the output sequence which is seen is epsilon a 1 a 2, a 1 a 2, once again epsilon a 1 a 2 and then again another a 1 a 2.

(Refer Slide Time: 41:09)



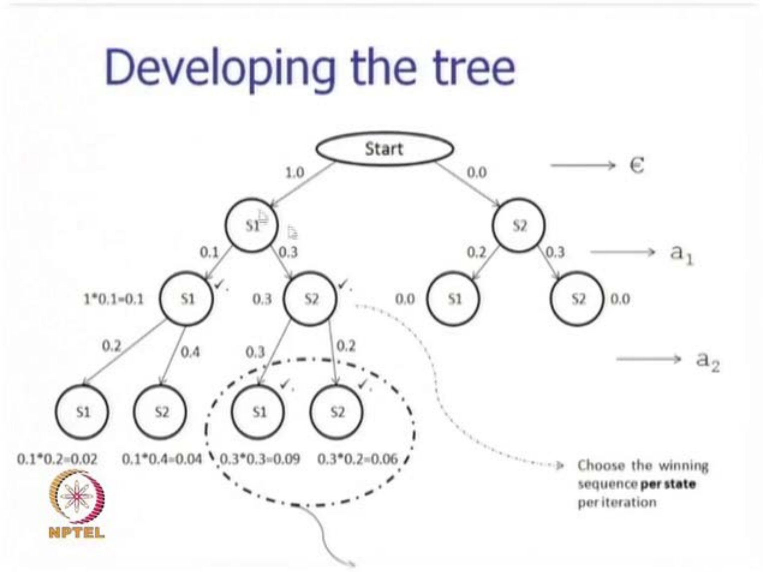
Tree structure contd...

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

The tree did not fit in the last slide, so two slides have been taken.

(Refer Slide Time: 41:17)



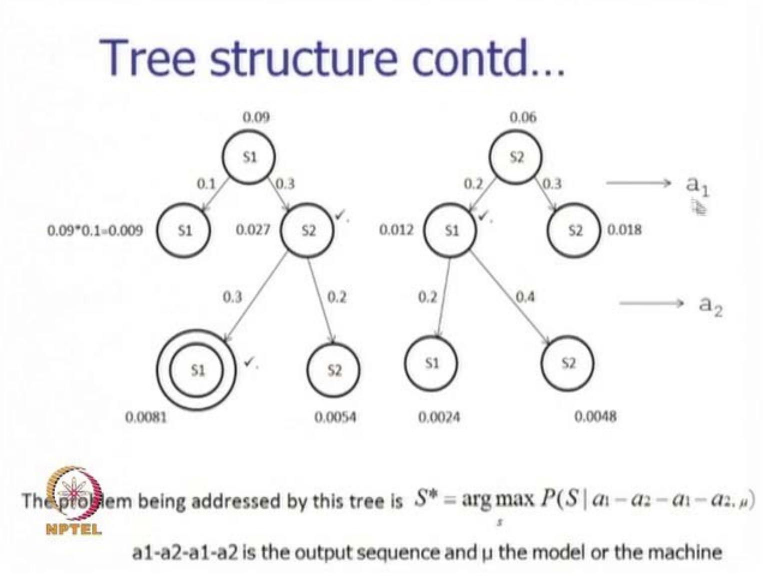
Developing the tree

Choose the winning sequence **per state** per iteration

And the observation sequences is epsilon a 1 a 2 a 1 a 2 or in other words, just a 1 a 2 a 1 a 2. So, the start state is epsilon and these state goes to S 1, S 2 is not the initial state, so therefore, their transition probability is 0. So, we have 1.0 here, this happens in epsilon then from epsilon on symbol a 1, we will go to state S 1 or S 2. The probability of S 1 to S 1 on the symbol a 1 is nothing but 0.1 and that from S 1 to S 2 on symbol a 1 is 0.3.

After that, we have advanced S 1 and S 2 and the probability of the subsequence is S 1 S 1 is noted here, which is 0.1, here the probability of the subsequence S 1 S 1, S 1 S 2 is 0.3. Now, when we advance the nodes, here S 1 goes to S 1, S 1 goes to S 2 and the symbol here is a 2 and here out of these four nodes, which are the ones we retain. So, you can see this S 1 S 1, this sequence S 1 S 1 S 1 ends in S 1, this sequence S1 S 2 S 1 ends in S 1, this is S 1 S 1 S 3 S 2 which ends in S 2, this ends in S 2. So, amongst those sequences ending in S 1 we retrain these, because this probability 0.09 is more than the other segments probability 0.02. Similarly, here we retrain S 2 here, these two nodes are discarded, they will not be expanded.

(Refer Slide Time: 43:03)



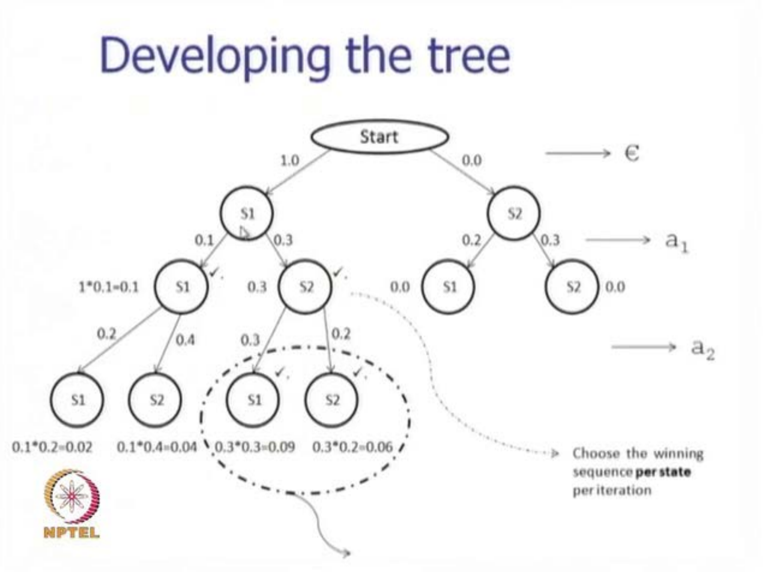
Tree structure contd...

The problem being addressed by this tree is $S^* = \underset{s}{\arg\max}\, P(S \mid a_1 - a_2 - a_1 - a_{2,\mu})$

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

Advancing the tree, we have S 1 S 2, probabilities are 0.09 and 0.06, we see another a 1, S 2 is advanced this way with S 1 and S 2. S 1 is advanced, S 1 and S2 the sequence probability are 0.009, 0.027, 0.012, 0.018. So, amongst them we retain again this node and this node, because this sequence probability is ending in S 1, is less than the sequence probability ending in S 1 here. Similarly, the sequence probability S 2 here is more than the probability sequence probability S 2 here. So, we have these two nodes surviving, the final symbol is a 2 and when we advance these again, we get S 1 S 2 S 1 S 2 here and this becomes the winner node. When we follow it backward then we get the sequence of states S 1 the final state, the previous final state is S 2, previous state is S 1.

(Refer Slide Time: 44:12)

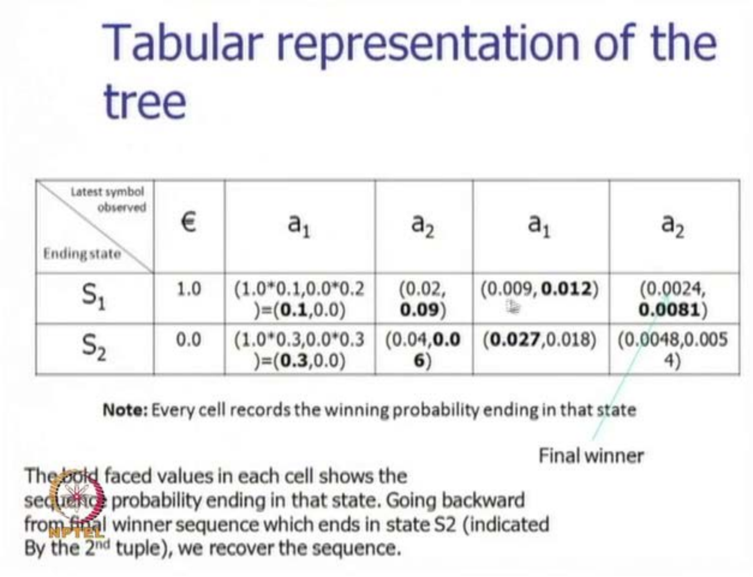


Developing the tree

Choose the winning sequence **per state** per iteration

The state previous to that is S 1 and then another S 1, so S1 S 1 S 2, S 1 S 1 S 2 and then again another S 1, this is the best possible state sequence.

(Refer Slide Time: 44:31)



Tabular representation of the tree

Latest symbol observed Ending state	ϵ	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,**0.0 6**)	(**0.027**,0.018)	(0.0048,0.005 4)

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 S2 (indicated By the 2nd tuple), we recover the sequence.

So, this whole thing can be simulated by tabular representation, which is equivalent to the tree. And these table representation is the data structure, which is useful for implementing the viterbi algorithm, this we will see in the next class.