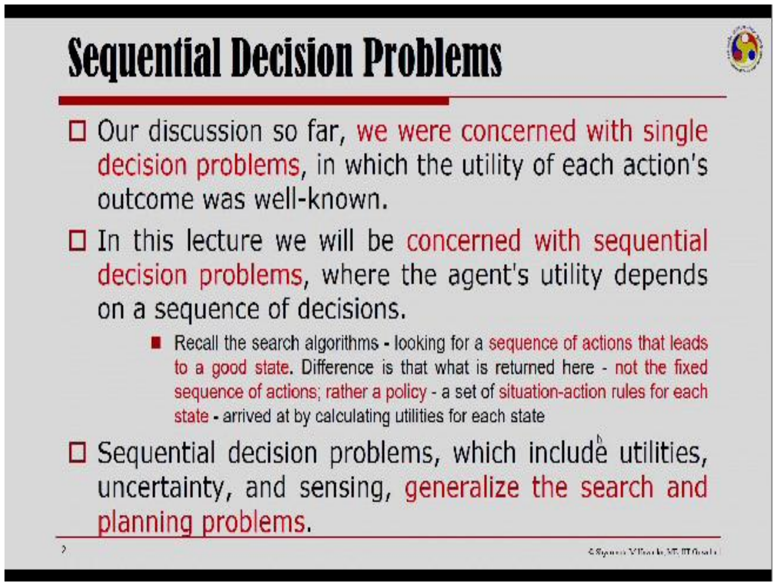
Fundamentals of Artificial Intelligence
Prof. Shyamanta M Hazarika
Department of Mechanical Engineering
Indian Institute of Technology- Guwahati

Lecture-25
Sequential Decision Problems

Welcome to fundamentals of artificial intelligence, today we look at sequential decision problems. Sequential decision problems which involve utility and uncertainty, generalize the search and planning problems that we have looked so far. Recall that in our discussion on reasoning under uncertainty we had looked at utility theory, wherein the agent was concerned about making a single decision based on the outcome of the actions utility value.

Here today in sequential decision problems we will look at utility being computed for a sequence of actions and we will see how we can go about making decisions under such a scenario.

(Refer Slide Time: 01:45)

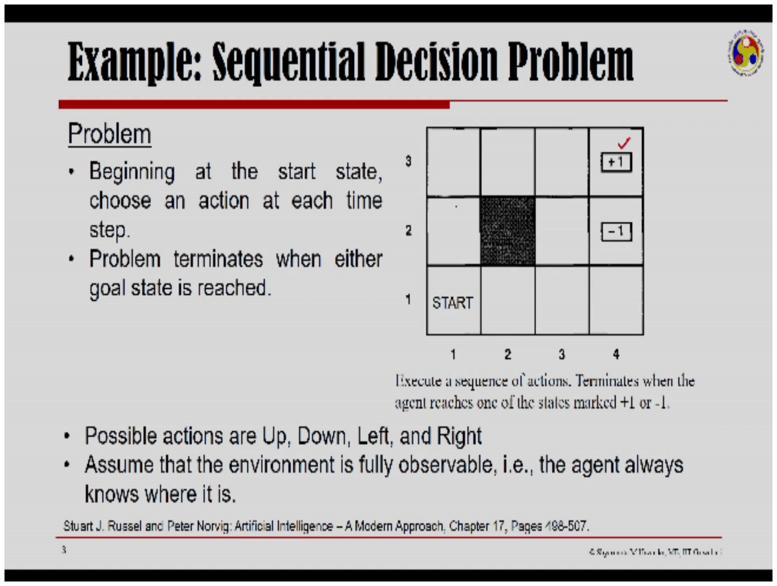


Sequential Decision Problems

- Our discussion so far, we were concerned with single decision problems, in which the utility of each action's outcome was well-known.
- In this lecture we will be concerned with sequential decision problems, where the agent's utility depends on a sequence of decisions.
  - Recall the search algorithms - looking for a sequence of actions that leads to a good state. Difference is that what is returned here - not the fixed sequence of actions; rather a policy - a set of situation-action rules for each state - arrived at by calculating utilities for each state
- Sequential decision problems, which include utilities, uncertainty, and sensing, generalize the search and planning problems.

Sequential decision problems are concerned with making decisions when a sequence is involved. Now if you think of the search algorithms that we have discussed previously we were looking for a sequence of actions that could lead to a good state for us. However here there is a basic difference, the difference being instead of returning as a sequence of actions what would be returned by a sequential decision problem is a policy, a set of situation action rules for each state. And that would be arrived by calculating utilities for each table.

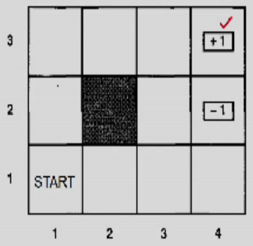
(Refer Slide Time: 02:36)



Example: Sequential Decision Problem

Problem

- Beginning at the start state, choose an action at each time step.
- Problem terminates when either goal state is reached.



Execute a sequence of actions. Terminates when the agent reaches one of the states marked +1 or -1.

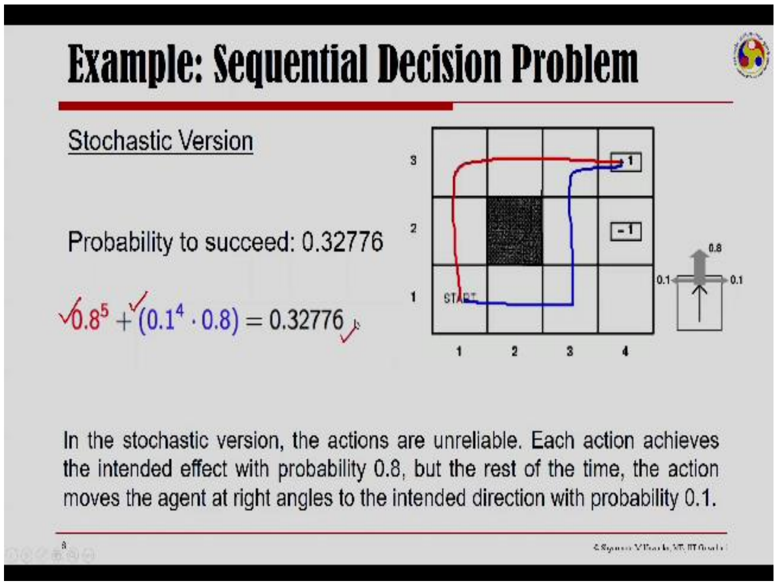
- Possible actions are Up, Down, Left, and Right
- Assume that the environment is fully observable, i.e., the agent always knows where it is.

Stuart J. Russel and Peter Norvig: Artificial Intelligence – A Modern Approach, Chapter 17, Pages 498-507.

So let us look at an example to understand sequential decision problems, here is a 4 cross 3 grid and an agent needs to start at 1, 1 and it much reach idly the state marked with +1. Now the problem terminates when either of the goal state given by + 1 or - 1 is reached. This example that we are taken here today for discussion is from the book by Russell and Norvig chapter 17 pages 498 to 507.

Now the possible actions for this agent in this scenario are to either go up or in a given scenario come down or go left and right. So the agent needs to execute a sequence of actions and we assume that the environment is fully observable, that is the agent always knows where it is.

(Refer Slide Time: 03:57)



Example: Sequential Decision Problem

Stochastic Version

Probability to succeed: 0.32776

$0.8^5 + (0.1^4 \cdot 0.8) = 0.32776$

3
2
1
START
1
2
3
4
+1
-1
0.8
0.1
0.1

In the stochastic version, the actions are unreliable. Each action achieves the intended effect with probability 0.8, but the rest of the time, the action moves the agent at right angles to the intended direction with probability 0.1.

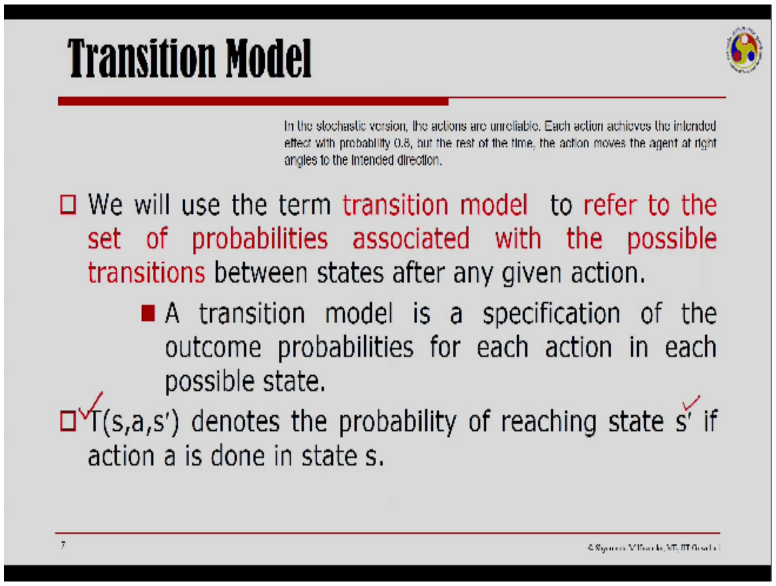
In a deterministic version of this problem if I am interested to get to the goal the objective would be to get the maximum reward. So each action would be like moving the one square in the intended direction, except when I move into your wall resulting in no change in my position. In a deterministic version if I know the initial state and the effects of my actions then this problem can be directly solved by search algorithms that we have described before in this course.

Now this is true regardless of where whether the environment is accessible or inaccessible. In such a scenario the solution for this sequential decision problem is to move up once and thereafter move again move to the right one time, a second time and the third time would bring me to the goal. Now if I am looking at a stochastic version of the same problem. Let us assume that I can go forward with a probability of 0.8 and the rest of the time it either goes to the cells to its right or to its left with probability of 0.1.

And like the previous case if my agent bombs into the wall it stays in place. Now under such a scenario one cannot create a plan ahead of time, in fact if you look at whether my old plan would succeed, I can see what would be the probability of the old plan succeeding. So because the actions are unreliable each action has the intended effect with certain probabilities and that is 0.8 when it goes forward and 0.1 when the agent goes at right-angles.

So if I now compute the probability to succeed by taking the one version which is the one that is marked in red or a very unlikely other version which is marked in blue where the total probability to succeed would be 0.32776.

(Refer Slide Time: 06:52)



Transition Model

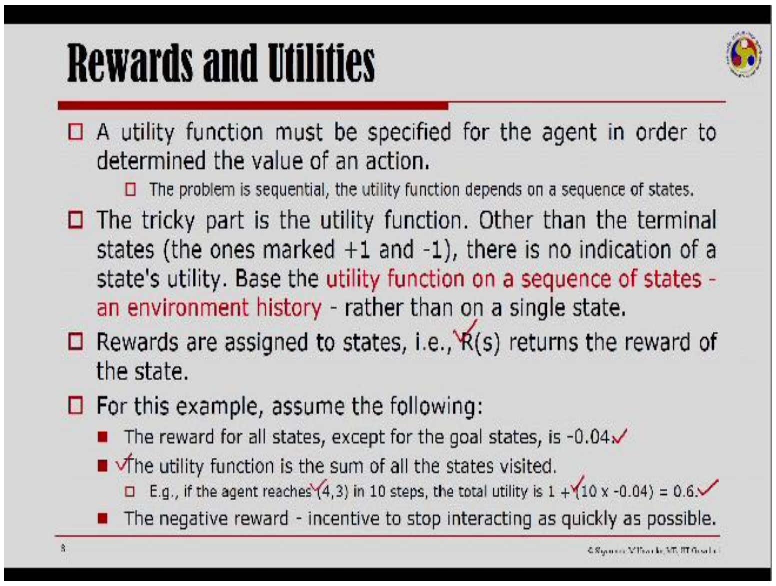
In the stochastic version, the actions are unreliable. Each action achieves the intended effect with probability 0.8, but the rest of the time, the action moves the agent at right angles to the intended direction.

- We will use the term transition model to refer to the set of probabilities associated with the possible transitions between states after any given action.
  - A transition model is a specification of the outcome probabilities for each action in each possible state.
- T(s,a,s′) denotes the probability of reaching state s′ if action a is done in state s.

Now we should understand that in the stochastic version that we were talking off the actions are unreliable, each action actually achieves the intended effect with a probability of 0.8. But the rest of the time the action moves the agent at right angles to the intended direction and therefore we will need to use a transition model. This transition model refer to the set of probabilities associated with the possible transitions between states after any given action.

The transition model is a specification of the outcome probabilities for each action, in each possible state. In our case the transition model is given as T of s a s prime which denotes the probability of reaching the state s prime, if I do an action a in the state s. So under such a transition model I would know now what would be the likelihood of a given action in a given state to arrive at a new state?

(Refer Slide Time: 08:25)



Rewards and Utilities

- A utility function must be specified for the agent in order to determined the value of an action.
  - The problem is sequential, the utility function depends on a sequence of states.
- The tricky part is the utility function. Other than the terminal states (the ones marked +1 and -1), there is no indication of a state's utility. Base the utility function on a sequence of states - an environment history - rather than on a single state.
- Rewards are assigned to states, i.e., R(s) returns the reward of the state.
- For this example, assume the following:
  - The reward for all states, except for the goal states, is -0.04.
  - The utility function is the sum of all the states visited.
    - E.g., if the agent reaches (4,3) in 10 steps, the total utility is 1 + (10 x -0.04) = 0.6.
  - The negative reward - incentive to stop interacting as quickly as possible.

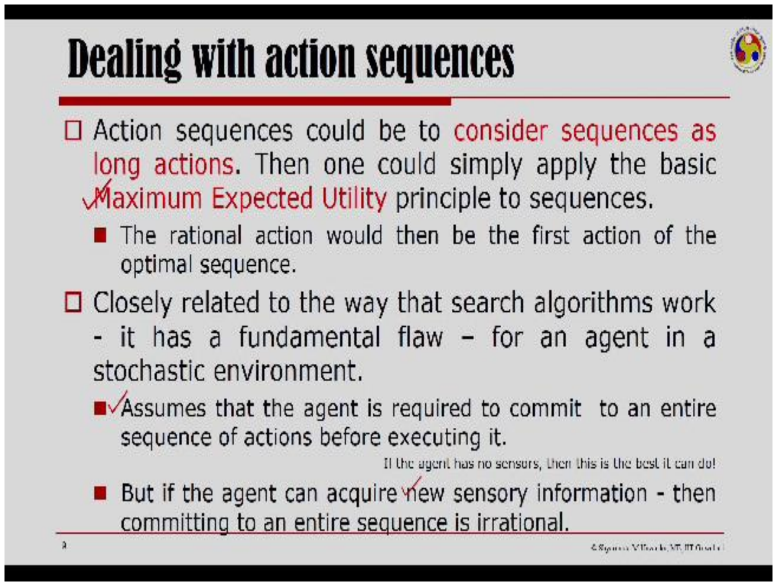
A utility function would need to be specified for the agent in order to determine the value of an action. Now in our earlier discussion each agent was involved in a single decision and when I was looking for the utility I was computing it based on what is the final result of a given action, here the problem is sequential and therefore the utility function depends not on a single state that I arrive at but on a sequence of states under such as scenario it is very tricky to compute the utility function.

In our case except for the terminal states which is marked with + 1 or - 1 no other state has been given an indication of what its utility is. So when we base the utility function on sequence of states we need to look at the history or what we call the environment history rather than on a single state. So we do it in the following way rewards are assigned to the states that is each state returns a reward which is given as Rs.

For this example we can assume the following that the reward for all states except for the goal state is - 0.04 and then if we look at the utility function the utility function is the sum of all the states visited so far. So for example if my agent reaches 4, 3 in 10 steps then the total utility would be 1 + 10 steps it has visited which each had a utility of - 0.4. So that would be reduced from 1 and I would have a total utility of 0.6.

I can also make provisions for negative reward that is an incentive to stop interacting as quickly as possible, after we have looked at the rewards we can now deal with action sequences .So we can look at action sequences as long actions themselves, then we could apply the basic maximum expected utility principle to the complete sequence.

(Refer Slide Time: 11:11)



Dealing with action sequences

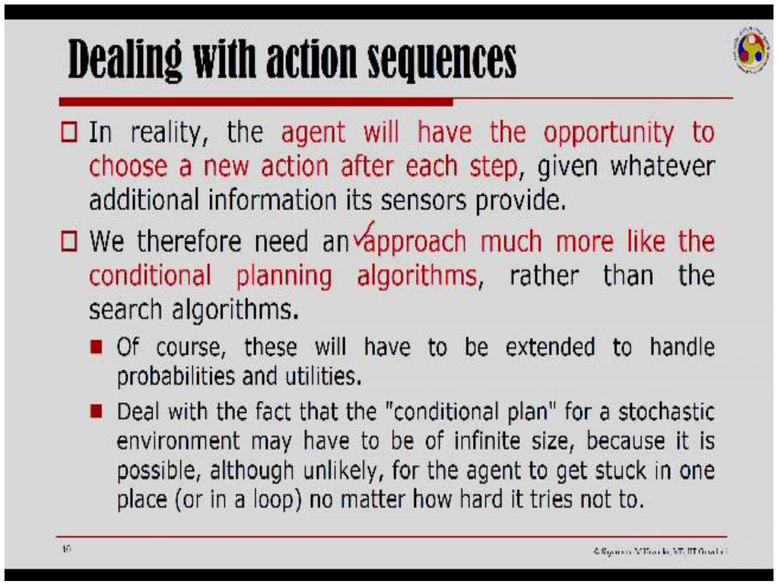
- Action sequences could be to consider sequences as long actions. Then one could simply apply the basic ✓Maximum Expected Utility principle to sequences.
  - The rational action would then be the first action of the optimal sequence.
- Closely related to the way that search algorithms work - it has a fundamental flaw – for an agent in a stochastic environment.
  - ✓Assumes that the agent is required to commit to an entire sequence of actions before executing it.

    If the agent has no sensors, then this is the best it can do!
  - But if the agent can acquire ✓new sensory information - then committing to an entire sequence is irrational.

And the rational action would then be the first action of the optimal sequence, but then there is a flaw in this approach even if it is closely related to the way that search algorithms work for a stochastic environment the flaw is that here I assume the agent to commit to an entire sequence of action before executing it. Now in a case where the agent has no senses this is the best it can do. But let us assume that the agent can acquire new sensory information.

Then committing to an entire sequence is actually irrational because after I reach a state new percepts that I have may enable me to take a better decision rather than committing to a sequence of actions a priori. So even if this idea of action sequences to be considered as long actions and just using maximum expected utility looks attractive but it is not the best thing to do. In reality as I have been emphasizing the agent will have the opportunity to choose a new action after each step.

(Refer Slide Time: 12:44)



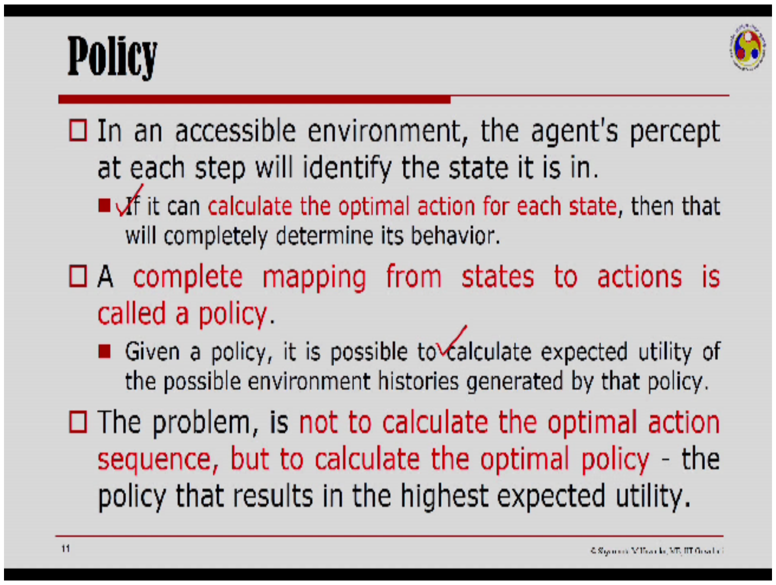
Dealing with action sequences

- In reality, the agent will have the opportunity to choose a new action after each step, given whatever additional information its sensors provide.
- We therefore need an approach much more like the conditional planning algorithms, rather than the search algorithms.
  - Of course, these will have to be extended to handle probabilities and utilities.
  - Deal with the fact that the "conditional plan" for a stochastic environment may have to be of infinite size, because it is possible, although unlikely, for the agent to get stuck in one place (or in a loop) no matter how hard it tries not to.

Given whatever new information it gets from the sensors, we therefore need an approach which is more like conditional planning algorithms rather than search algorithms. So this will of course have to be extended to handle probabilities and utilities we can deal with the fact that the conditional plan for a stochastic environment may have to be also infinite in size, because one needs to understand that though very unlikely there is a possibility that a agent may get stuck in one place or in a loop no matter how hard it tries not to when dealing with a stochastic environment.

So when we are dealing with action sequences we would rather be using an approach more like conditional planning instead of search as a sequence. So in an accessible environment we have the agents percept which at each step will identify the state the agent is in.

(Refer Slide Time: 14:02)



Policy

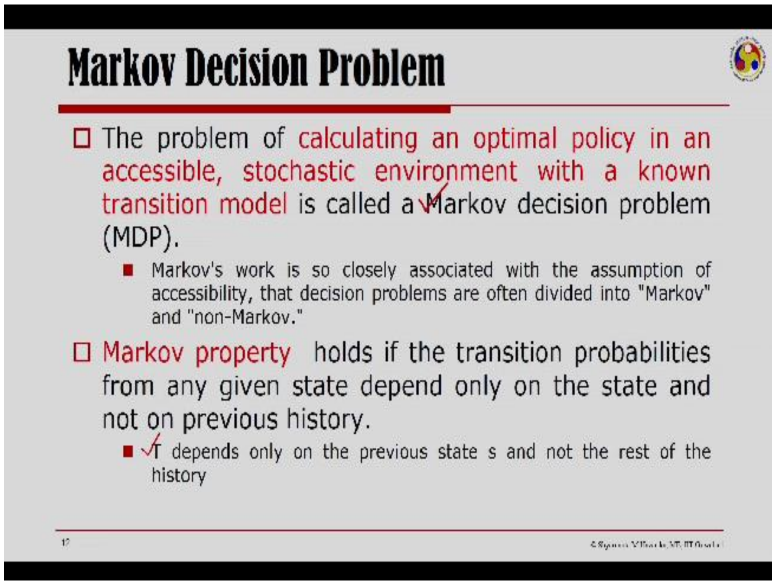
- In an accessible environment, the agent's percept at each step will identify the state it is in.
  - If it can calculate the optimal action for each state, then that will completely determine its behavior.
- A complete mapping from states to actions is called a policy.
  - Given a policy, it is possible to calculate expected utility of the possible environment histories generated by that policy.
- The problem, is not to calculate the optimal action sequence, but to calculate the optimal policy - the policy that results in the highest expected utility.

11

If it can calculate an optimal action for each step then that will completely determine its behavior, a complete mapping from the states to action is what I need and this is called a policy. So given a policy it is possible to calculate the expected utility of the possible environment histories generated by the policy and the problem is therefore not to calculate the optimal action sequence, what I am more interested in sequential decision problems is to calculate the optimal policy.

The policy that results in the highest expected utility, so instead of looking at the complete sequence of actions as a long action itself and trying to commit to that sequence a priori we rather look at a mapping from states to actions and we are interested in finding out such a policy. So the problem of calculating an optimal policy in an accessible stochastic environment with a known transition model is called a Markov decision problem.

(Refer Slide Time: 15:37)



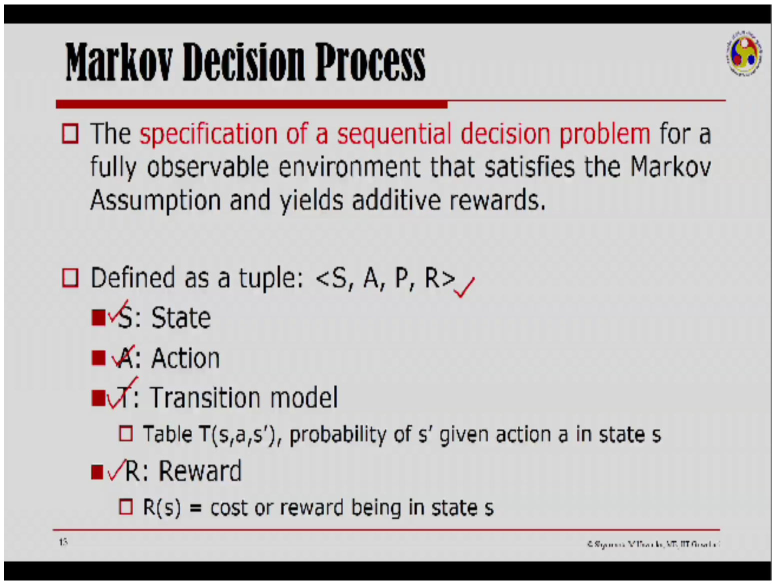
Markov Decision Problem

- The problem of calculating an optimal policy in an accessible, stochastic environment with a known transition model is called a ✓Markov decision problem (MDP).
  - Markov's work is so closely associated with the assumption of accessibility, that decision problems are often divided into "Markov" and "non-Markov."
- Markov property holds if the transition probabilities from any given state depend only on the state and not on previous history.
  - ✓T depends only on the previous state s and not the rest of the history

Now this is very important for us to realize that Markov decision problem is about getting to an optimal policy in an accessible environment under a stochastic environment, but then the transition model that is the change from one state to another state given an action in the previous set is known to me. Now Markov's work is very closely associated with the assumption of accessibility and decision problems are therefore often divided into what are called Markov and non Markov decision problems.

Closely related to the concept of a Markov decision process is the Markov property. Now Markov property holds if the transition probabilities from any given state depend only on the state and not on previous histories. In our case the transition model that we have assumed T if it depends only on the state s and not on the rest of the history, then we say it satisfies Markov problem.

(Refer Slide Time: 17:14)



Markov Decision Process

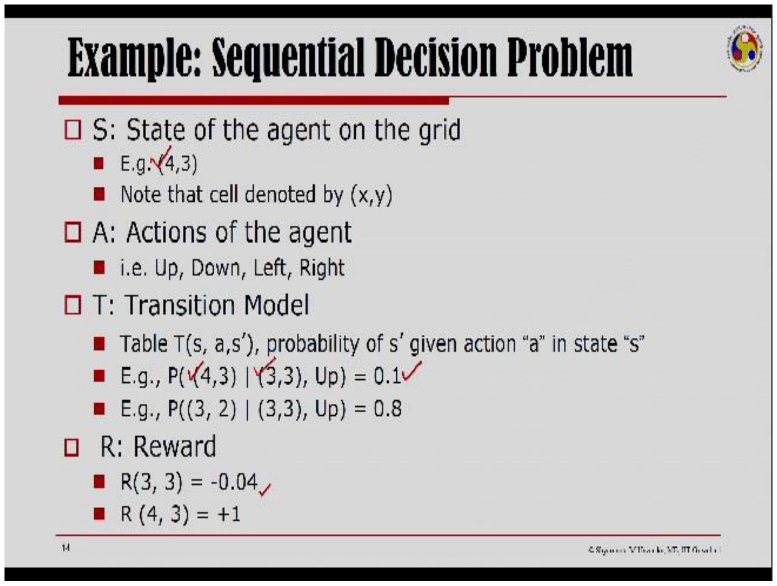
- The specification of a sequential decision problem for a fully observable environment that satisfies the Markov Assumption and yields additive rewards.
- Defined as a tuple: <S, A, P, R>✓
  - ✓S: State
  - ✓A: Action
  - ✓T: Transition model
    - Table T(s,a,s′), probability of s′ given action a in state s
  - ✓R: Reward
    - R(s) = cost or reward being in state s

15 [illegible]

Markov decision process is the specification of a sequential decision problem for a fully observable environment that satisfies the Markov assumption. So it is defined as a 4 tuple where I have this state, the action, the transition model, in fact the transition model is a table of probabilities of S prime given action a in state s and I have rewards recall that reward at every state is the cost or the reward to be in state s.

So this 4 tuple that I have under a fully observable environment satisfying the Markov assumption yielding additive rewards is a Markov decision process. Now for the example problem that we had started our discussion today let us look at what are these elements of the 4 tuple line.

(Refer Slide Time: 18:25)



Example: Sequential Decision Problem

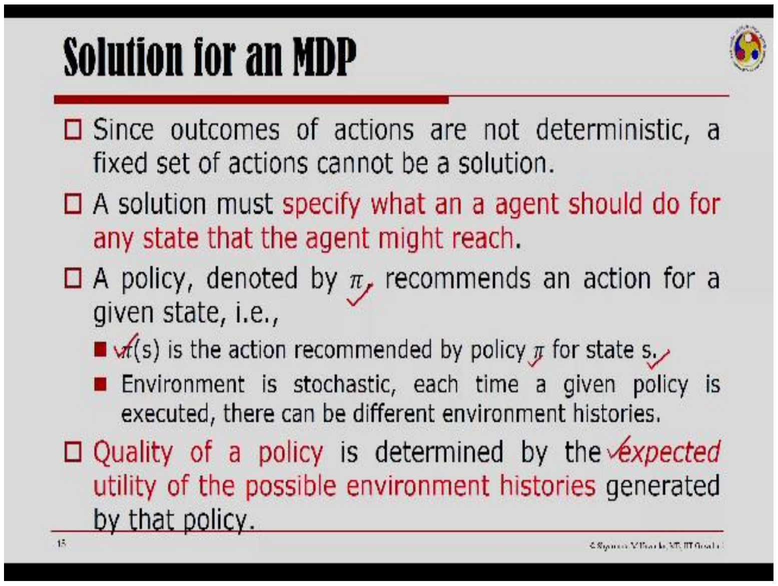
- S: State of the agent on the grid
  - E.g. (4,3)
  - Note that cell denoted by (x,y)
- A: Actions of the agent
  - i.e. Up, Down, Left, Right
- T: Transition Model
  - Table T(s, a,s'), probability of s' given action "a" in state "s"
  - E.g., P((4,3) | (3,3), Up) = 0.1
  - E.g., P((3, 2) | (3,3), Up) = 0.8
- R: Reward
  - R(3, 3) = -0.04
  - R (4, 3) = +1

14

So we have S the state of the agent on the grid, so it could be like 4, 3 the cell denoted by X , Y is this state of the agent, we have actions in our case as already noted the agent can go up down left and right, we there have to have the transition model. So we could say certain things like being in 4, 3 and the next state being 3, 3 the probability of up is 0.1. So I would have a table that gives me the probability of S prime given action a in state s.

And finally I have the rewards, so as already mentioned for our example except for the goal states the reward is assumed to be - 0.04.

(Refer Slide Time: 19:35)



Solution for an MDP

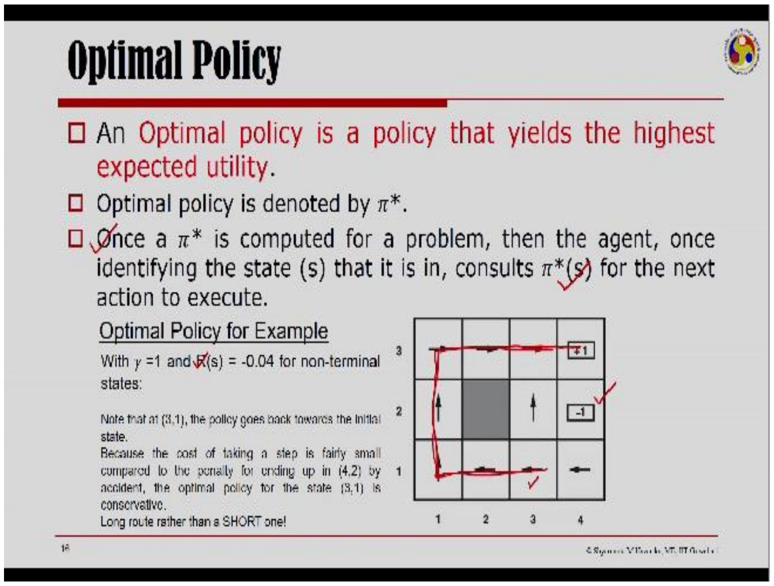
- Since outcomes of actions are not deterministic, a fixed set of actions cannot be a solution.
- A solution must specify what an a agent should do for any state that the agent might reach.
- A policy, denoted by π, recommends an action for a given state, i.e.,
  - $\pi(s)$ is the action recommended by policy π for state s.
  - Environment is stochastic, each time a given policy is executed, there can be different environment histories.
- Quality of a policy is determined by the expected utility of the possible environment histories generated by that policy.

15

So now having formulated the problem as a Markov decision problem we need to look at the solution since outcome of actions are not deterministic, a fixed set of actions cannot be a solution here. A solution here needs to specify what an agent should do for any state that the agent might reach. So we are looking for a policy and the policy is denoted by pi what the policy does is recommends an action for a given state.

So if I say pi of s this is the action recommended by the policy pi for the status. Now one needs to remember that the environment is stochastic so each time I have a given policy that is executed there can be different environment histories and the quality of a policy that I need to actually evaluate is determined by the expected utility of the possible environment histories that are generated by that policy.

(Refer Slide Time: 20:57)



Optimal Policy

- An Optimal policy is a policy that yields the highest expected utility.
- Optimal policy is denoted by π^*.
- Once a π^* is computed for a problem, then the agent, once identifying the state (s) that it is in, consults $\pi^*(s)$ for the next action to execute.

Optimal Policy for Example

With $\gamma = 1$ and $R(s) = -0.04$ for non-terminal states:

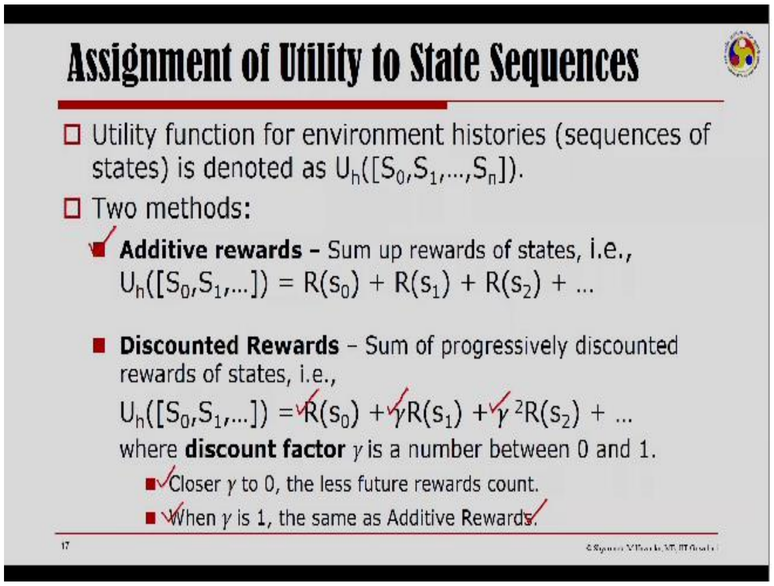
Note that at (3,1), the policy goes back towards the initial state.
Because the cost of taking a step is fairly small compared to the penalty for ending up in (4,2) by accident, the optimal policy for the state (3,1) is conservative.
Long route rather than a SHORT one!

And finally I have what is called an optimal policy that is a policy that would yield the highest expected utility. So an optimal policy is denoted by pi star and once pi star is computed for a problem then the agent identifying the state s that it is in would consult the optimal policy for the next action to execute. So what I am saying is in a sequential decision problem once we have somehow arrived at the optimal policy whenever the agent needs to take an action it will consult the optimal policy for the next action to execute at the given state.

So here is an example of the optimal policy for the problem that we were discussing. Now this is under a reward that is 0.04 for all non terminal states. Now one thing that I want to make explicit here is the point that if you look at 3, 1 even if the shortest path to the goal is about going here and getting to +1, the optimal policy here is taking along the route. Now, this is because the cost of taking a step in our case is fairly small compared to the penalty of ending up here by accident.

And therefore the optimal policy that I have is about taking a longer route and coming to the goal rather than taking a shorter one with the possibility of getting a penalty if I falling here and in order to avoid this I rather take along the route in my optimal policy.

(Refer Slide Time: 23:24)



Assignment of Utility to State Sequences

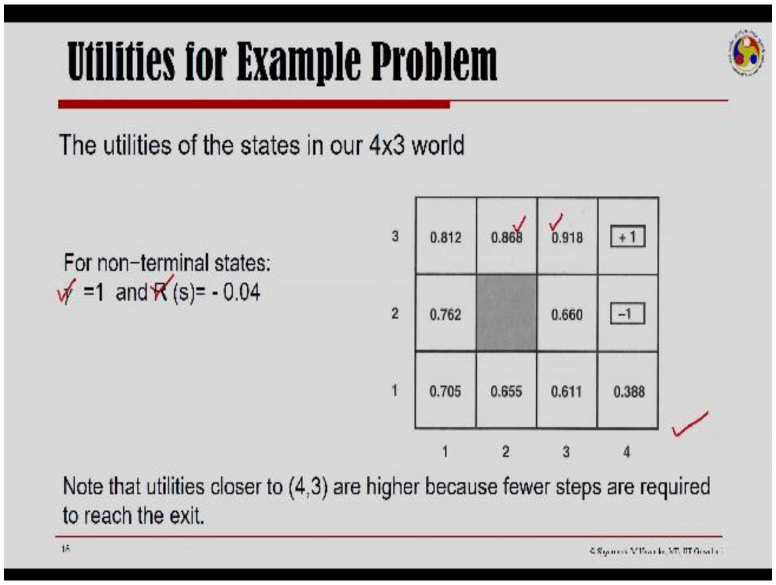
- Utility function for environment histories (sequences of states) is denoted as $U_h([S_0,S_1,\ldots,S_n])$.
- Two methods:
  - ✓ **Additive rewards –** Sum up rewards of states, i.e., $U_h([S_0,S_1,\ldots]) = R(s_0) + R(s_1) + R(s_2) + \ldots$
  - **Discounted Rewards** – Sum of progressively discounted rewards of states, i.e., $U_h([S_0,S_1,\ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots$ where **discount factor** γ is a number between 0 and 1.
    - ✓Closer γ to 0, the less future rewards count.
    - ✓When γ is 1, the same as Additive Rewards.✓

17

Once we have the environment the utility function for environment histories or sequence of states is computed under 2 methods, one is called the additive rewards. The additive rewards method is about summing up the rewards of states that comprises the environment history, the other method is called discounted rewards which is about the sum of progressively discounted rewards of states that is I have the first state reward.

The second one is multiplied by a discount factor gamma which is a number between 0 and 1 and then the next is gamma square RS 2 so on and so forth. Now the closure gamma is to 0 the last future rewards count and if you note that when I have gamma = 1 the discounted rewards is same as the additive rewards. So this is how I assign utility to state sequences and then let us look at utilities for our example problem.

(Refer Slide Time: 24:54)



Utilities for Example Problem

The utilities of the states in our 4x3 world

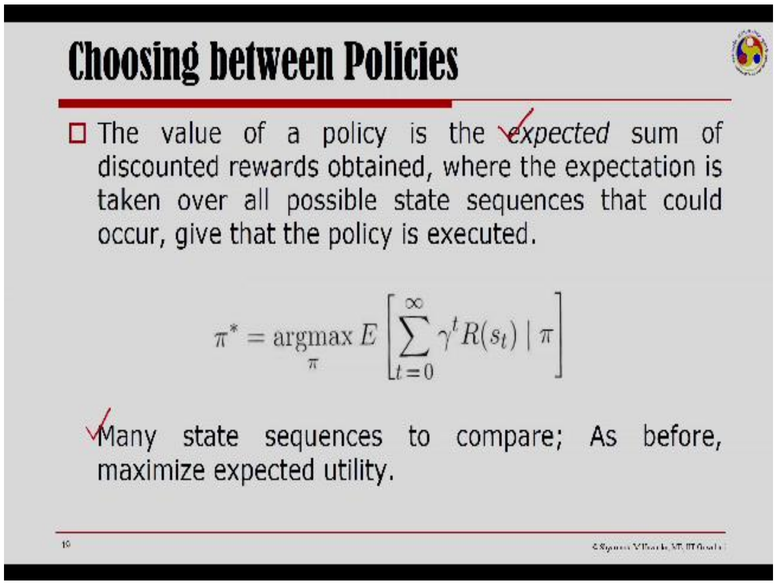
For non−terminal states:
γ =1 and R(s)= - 0.04

	1	2	3	4
3	0.812	0.868	0.918	+1
2	0.762	▨	0.660	−1
1	0.705	0.655	0.611	0.388

Note that utilities closer to (4,3) are higher because fewer steps are required to reach the exit.

So the utilities of the states in our 4 cross 3 world for our non-terminal states we take gamma = 1 and the reward being - 0.04, we have computed the utilities as shown here, now note that the utilities that are closure 2 4 3 are higher because we need fewer steps to reach the exit from these states.

(Refer Slide Time: 25:32)



Choosing between Policies

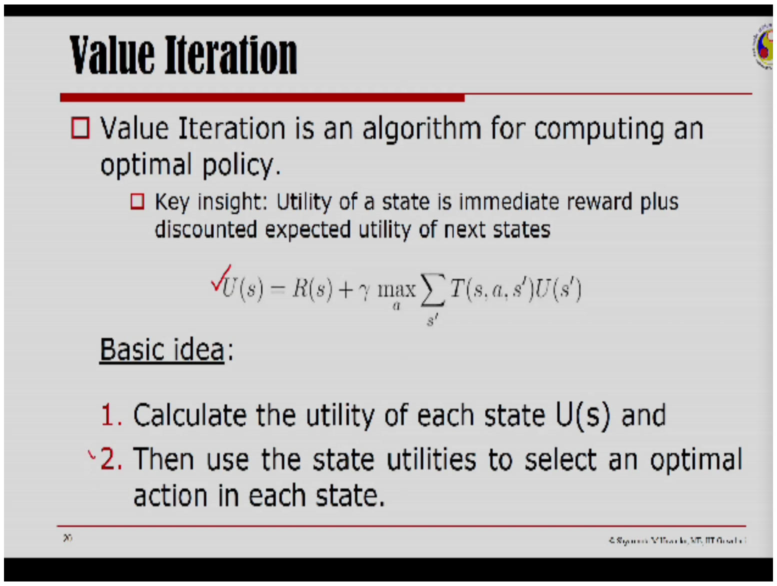
- The value of a policy is the *expected* sum of discounted rewards obtained, where the expectation is taken over all possible state sequences that could occur, give that the policy is executed.

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi\right]$$

Many state sequences to compare; As before, maximize expected utility.

Having looked at the optimal policy and the utilities of states, now the question is about choosing between policies that is how do I rank the policies: the value of a policy is the expected some of the discounted rewards obtained where the expectation is taken overall possible state sequences that could occur. So pi star that I have is about getting the maximum, so I have many state sequences to compare as before we would love to maximize the expected utility when choosing the policies. Now value iteration is an algorithm that allows me to compute the optimal policy.

(Refer Slide Time: 26:28)



Value Iteration

- Value Iteration is an algorithm for computing an optimal policy.
  - Key insight: Utility of a state is immediate reward plus discounted expected utility of next states

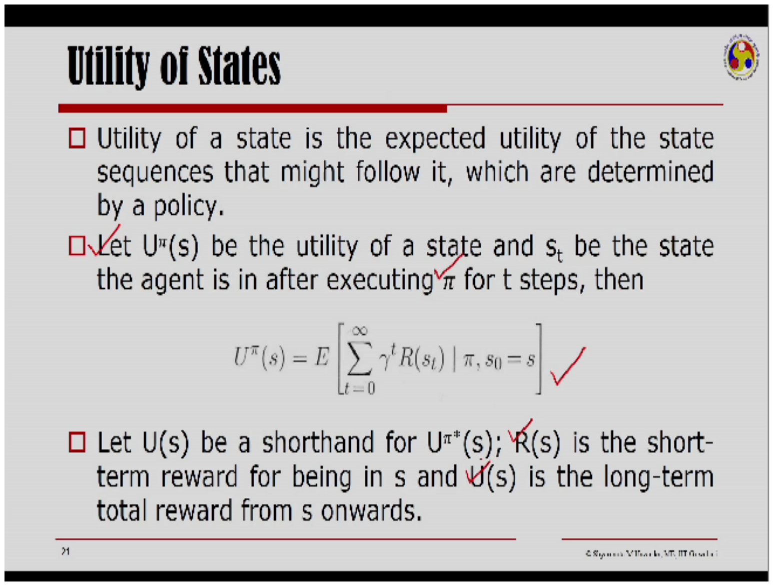
$$U(s) = R(s) + \gamma \max_{a} \sum_{s'} T(s,a,s') U(s')$$

Basic idea:

1. Calculate the utility of each state U(s) and
2. Then use the state utilities to select an optimal action in each state.

The key insight in this algorithm is that the utility of a state is immediate reward + the discounted expected utility of next states. So this is what is the key inside of the algorithm of value iteration, the basic idea is to calculate the utility of each state and then use the state utilities to select an optimal action.

(Refer Slide Time: 27:05)



Utility of States

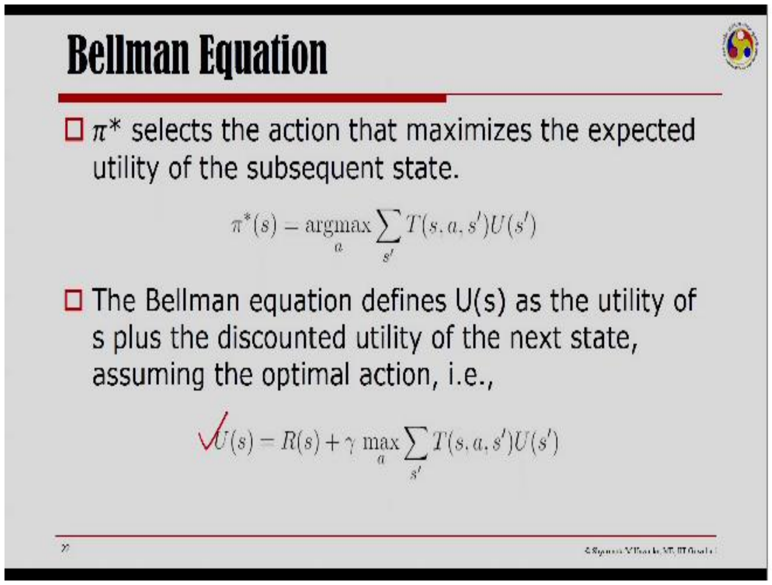
- Utility of a state is the expected utility of the state sequences that might follow it, which are determined by a policy.
- Let $U^{\pi}(s)$ be the utility of a state and s_t be the state the agent is in after executing π for t steps, then

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^{t} R(s_t) \mid \pi, s_0 = s\right]$$

- Let U(s) be a shorthand for $U^{\pi^*}(s)$; R(s) is the short-term reward for being in s and U(s) is the long-term total reward from s onwards.

So the utility of a state for that matter is the expected utility of the state sequence that might follow it. So here we have u the utility of a state s after execution of pi 40 steps, we are interested in finding out the expected utility as the sum of the rewards. So, here the rewards is the short-term reward for being in the state s where as the utility of s is the long-term total reward from s onwards.

(Refer Slide Time: 27:50)



Bellman Equation

- π^* selects the action that maximizes the expected utility of the subsequent state.

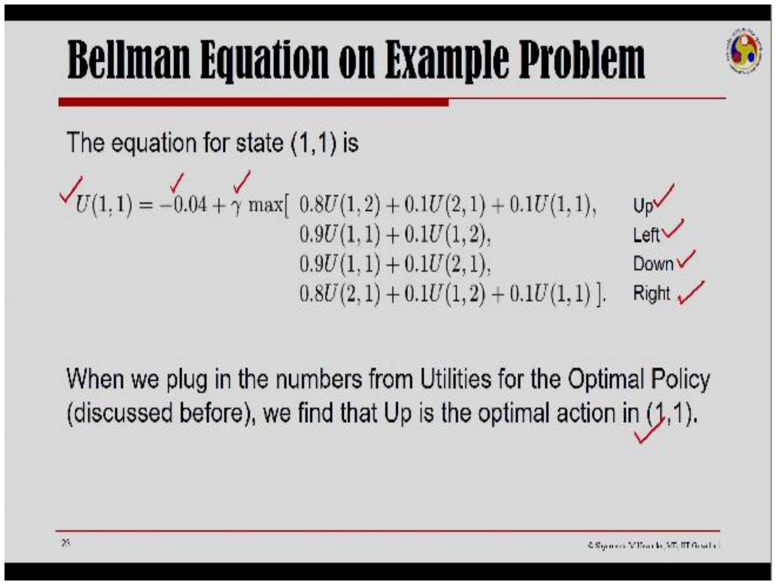
$$\pi^*(s) = \operatorname{argmax}_a \sum_{s'} T(s,a,s')U(s')$$

- The Bellman equation defines U(s) as the utility of s plus the discounted utility of the next state, assuming the optimal action, i.e.,

$$\checkmark U(s) = R(s) + \gamma \max_a \sum_{s'} T(s,a,s')U(s')$$

So pi star selects the action that maximizes the expected utility of the subsequence state and the Bellman equation defines the utility at a given state as the utility of s + the discounted utility of the next step, assuming the optimal action.

(Refer Slide Time: 28:16)



Bellman Equation on Example Problem

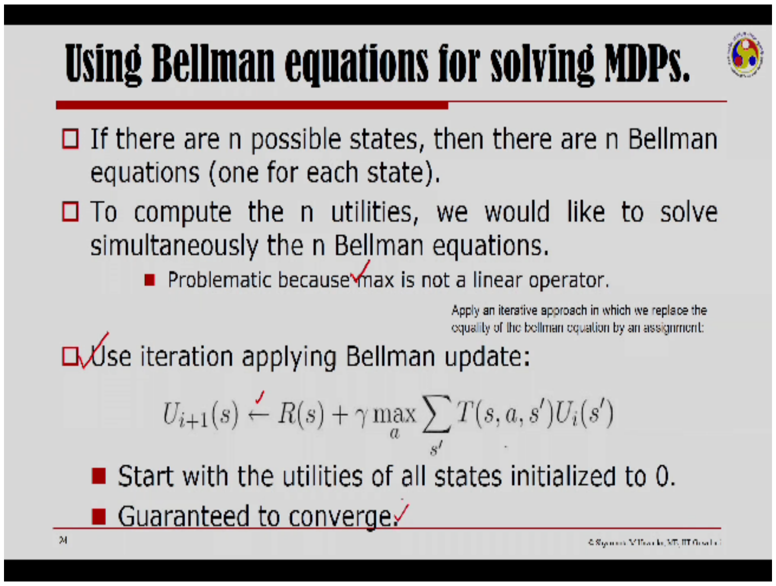
The equation for state (1,1) is

$$\begin{aligned} U(1,1) = -0.04 + \gamma \max[\ & 0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1), && \text{Up} \checkmark \\ & 0.9U(1,1) + 0.1U(1,2), && \text{Left} \checkmark \\ & 0.9U(1,1) + 0.1U(2,1), && \text{Down} \checkmark \\ & 0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1)\ ]. && \text{Right} \checkmark \end{aligned}$$

When we plug in the numbers from Utilities for the Optimal Policy (discussed before), we find that Up is the optimal action in (1,1).

So for our example problem if I want to evaluate the utility of 1, 1 given the Bellman equation, then I have its own reward + the discount factor and then I am looking at the maximum of the actions I could either do an up or a left or down or right. Now when we plug in the numbers from utilities for the optimal policy that we have discussed before we find that up is the optimal action in 1, 1 that is what comes out very clearly for us.

(Refer Slide Time: 29:03)



Using Bellman equations for solving MDPs.

- If there are n possible states, then there are n Bellman equations (one for each state).
- To compute the n utilities, we would like to solve simultaneously the n Bellman equations.
  - Problematic because max is not a linear operator.

Apply an iterative approach in which we replace the equality of the bellman equation by an assignment.

- Use iteration applying Bellman update:

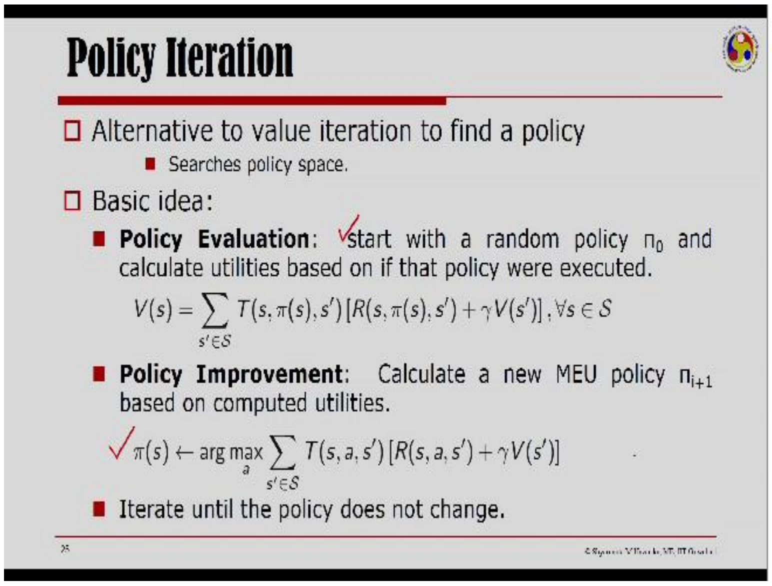
$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} T(s,a,s') U_i(s')$$

  - Start with the utilities of all states initialized to 0.
  - Guaranteed to converge.

So using Bellman equation we can look for n possible states, if there are n possible states then there are n Bellman equations one for each step and to compute the n utilities we would love to solve simultaneously the n Bellman equations. Now this is problematic, because in the Bellman equation the function max is not a linear operator. So what is done is we apply an iterative approach and in that approach we replace the equality of the Bellman equation by an assignment.

So use iteration applying Bellman update instead of an equality here I have an assignment now and we start with utilities of all states initialized to 0 and approach this iterative process which is guaranteed to converge.

(Refer Slide Time: 30:11)



Policy Iteration

- Alternative to value iteration to find a policy
  - Searches policy space.
- Basic idea:
  - **Policy Evaluation**: start with a random policy π_0 and calculate utilities based on if that policy were executed.

$$V(s) = \sum_{s' \in S} T(s,\pi(s),s')\,[R(s,\pi(s),s') + \gamma V(s')]\,, \forall s \in S$$

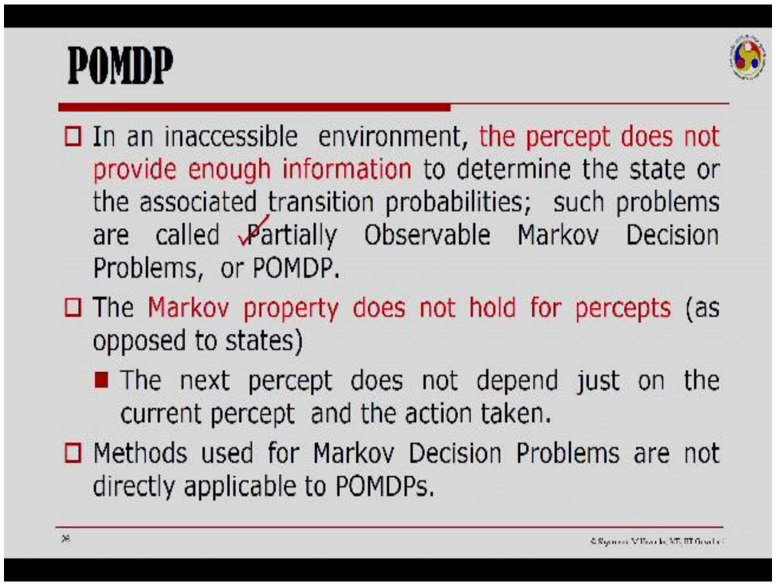
  - **Policy Improvement**: Calculate a new MEU policy π_{i+1} based on computed utilities.

$$\pi(s) \leftarrow \arg\max_a \sum_{s' \in S} T(s,a,s')\,[R(s,a,s') + \gamma V(s')]$$

  - Iterate until the policy does not change.

Another alternative to value iteration to find an optimal policy is the policy iteration. The policy iteration searches the policy space, the basic idea is to start with a random policy and calculate the utilities based on it if that policy were executed and then calculate a new maximum expected utility based on the computed utilities. That is called policy improvement and then iterate until the policy does not change.

(Refer Slide Time: 30:50)



POMDP

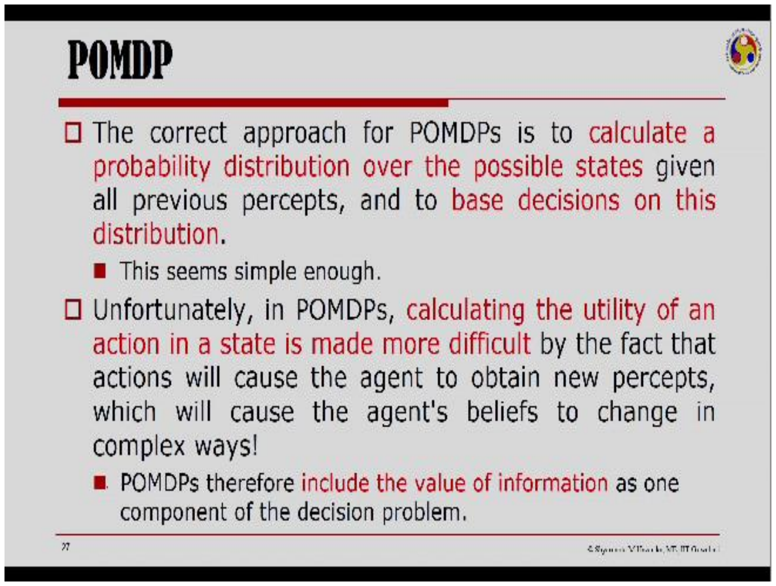
- In an inaccessible environment, the percept does not provide enough information to determine the state or the associated transition probabilities; such problems are called Partially Observable Markov Decision Problems, or POMDP.
- The Markov property does not hold for percepts (as opposed to states)
  - The next percept does not depend just on the current percept and the action taken.
- Methods used for Markov Decision Problems are not directly applicable to POMDPs.

Now so far we were talking of an accessible environment, in an inaccessible environment the percept does not provide enough information to determine the state or the associated transition probabilities, such problems are called partially observable Markov decision problems. So the

Markov property does not hold for percepts as opposed to states and the next percept does not depend just on the current percept and the action taken.

So methods used for Markov decision problems are not directly applicable to partially observable Markov decision problems.

(Refer Slide Time: 31:39)



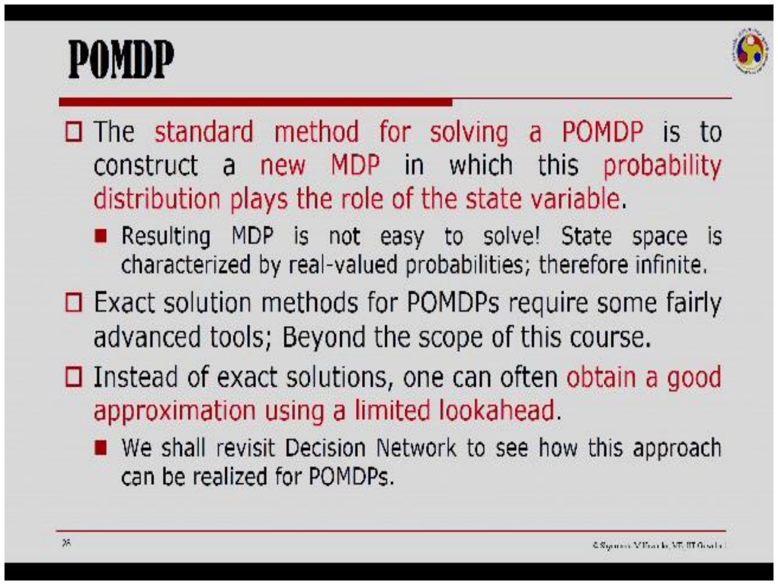
POMDP

- The correct approach for POMDPs is to calculate a probability distribution over the possible states given all previous percepts, and to base decisions on this distribution.
  - This seems simple enough.
- Unfortunately, in POMDPs, calculating the utility of an action in a state is made more difficult by the fact that actions will cause the agent to obtain new percepts, which will cause the agent's beliefs to change in complex ways!
  - POMDPs therefore include the value of information as one component of the decision problem.

The correct approach for partially observable Markov decision problems is actually to calculate a probability distribution over the possible states given all previous percepts and to base decision on this distribution. Now this seems simple enough but it is not, because in partially observable Markov decision problems calculating the utility of an action in a state is made difficult by the fact that the actions will cause the agent to obtain new percepts.

Then this will again change the agents belief to change in complex ways and therefore we can see that partially observable Markov decision problems include value of information that we have discussed when we were talking of reasoning under uncertainty as a component of the decision problem. Now the standard method of solving a partially observable Markov decision problem is to construct a new Markov decision problem in which this probability distribution plays the role of the state variable.

(Refer Slide Time: 33:01)



POMDP

- The standard method for solving a POMDP is to construct a new MDP in which this probability distribution plays the role of the state variable.
  - Resulting MDP is not easy to solve! State space is characterized by real-valued probabilities; therefore infinite.
- Exact solution methods for POMDPs require some fairly advanced tools; Beyond the scope of this course.
- Instead of exact solutions, one can often obtain a good approximation using a limited lookahead.
  - We shall revisit Decision Network to see how this approach can be realized for POMDPs.

However the resulting Markov decision problem is not easy to solve, this is because the state space is characterized by real valued probabilities and this makes it infinite. Exact solution methods for partially observable Markov decision problems required some fairly advanced tools and is beyond the scope of this fundamental course on artificial intelligence. Instead of exact solutions what one can often obtain is a good approximation using limited look-ahead as we had done while we were discussing game-playing.

Now we shall revisit decision networks and see how this approach can be realized for partially observable Markov decision problems, thank you very much.