

**NPTEL**

**NPTEL ONLINE CERTIFICATION COURSE**

**Introduction to Machine Learning**

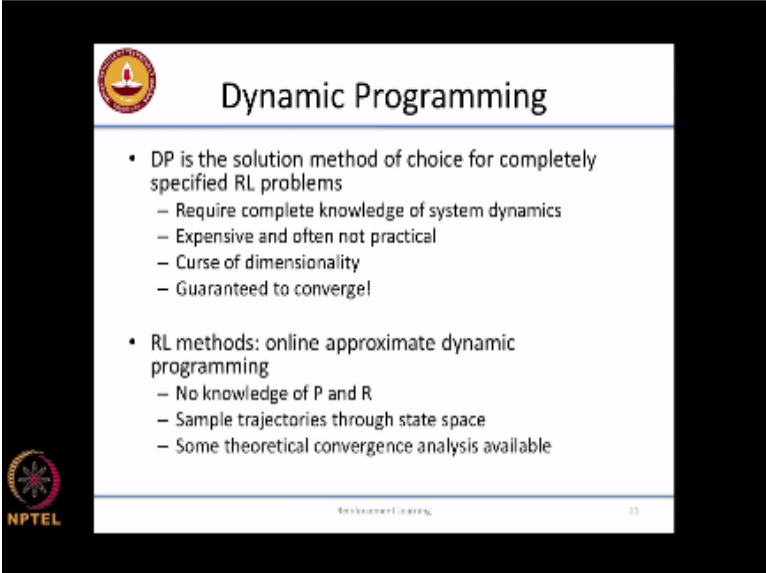
**Lecture-85**

**Solution Methods & Applications**

**Prof. Balaraman Ravindran  
Computer Science and Engineering  
Indian Institute of Technology Madras**

The one thing which I wanted to tell you was this right, there are you know two classes of algorithms that we will be talking about. So one of them is based on what is called dynamic programming right.

(Refer Slide Time: 00:29)



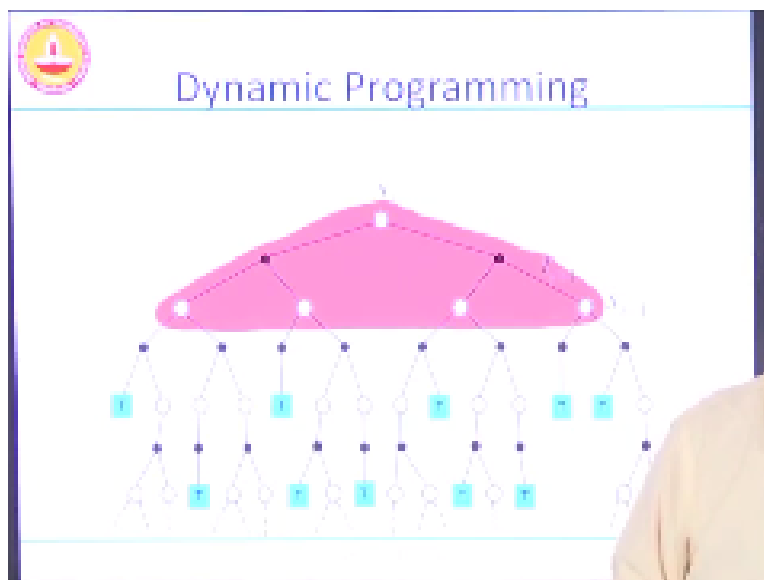
The slide is titled "Dynamic Programming" and features the IIT Madras logo in the top left corner. It contains two main bullet points. The first bullet point states that DP is the solution method of choice for completely specified RL problems, with sub-points: "Require complete knowledge of system dynamics", "Expensive and often not practical", "Curse of dimensionality", and "Guaranteed to converge!". The second bullet point discusses RL methods: online approximate dynamic programming, with sub-points: "No knowledge of P and R", "Sample trajectories through state space", and "Some theoretical convergence analysis available". The NPTEL logo is in the bottom left, and the text "Reinforcement Learning" and the number "23" are in the bottom right.

- DP is the solution method of choice for completely specified RL problems
  - Require complete knowledge of system dynamics
  - Expensive and often not practical
  - Curse of dimensionality
  - Guaranteed to converge!
- RL methods: online approximate dynamic programming
  - No knowledge of P and R
  - Sample trajectories through state space
  - Some theoretical convergence analysis available

So dynamic programming, so how many of you knows what dynamic programming is? I expect at least 23 hands up and I having mechanical engineers putting their heads have not the CS guys really, you know what dynamic programming is something wrong with your hand then. Now the essential idea behind dynamic programming is you will be using some kind of repeated structure in the problem right. So I am to solve the problem more efficiently right, suppose I have a solution that I can give for a problem of say size  $n$  right.

Then I will try to see if I can use that for defining the solution for the problem of size  $n+1$ , so some kind of a repeated sub structure in the problems right. The very rough way of describing what dynamic programming is right. So for example one way of thinking about dynamic programming is I have this game tree right, so I look at the values of winning or the expectations of winning from all of these steps right. I will use these in order to compute the value of winning or the probability of winning from the state right.

(Refer Slide Time: 01:49)

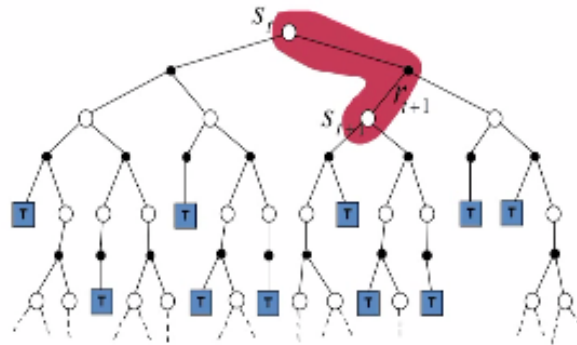


So if you think about it if from here if I am going to take say  $n$  steps, from here how many steps will be expect me to take? And minus 1 steps right, so I look at the probability of winning when I when I have only  $n - 1$  steps left right, I will estimate that first I will use that solution for estimating the probability of winning when I have  $n$  steps only  $n - 1$  steps left right I will estimate that first, I will use that solution for estimating the probability of winning when I have  $n$  steps left right. So that is essentially the idea behind dynamic programming and so all you do now is instead of having the entire outcome.

(Refer Slide Time: 02:20)



## Simplest TD Method



And using that for estimating the probability of winning here right I am going to just use one step that I take through the tree, right I use this what happens in this one step I will use that in order to update the probability of winning here all rights instead of using the entire outcome right as a dynamic programming in reinforcement learning methods we will be using samples that you are getting through the state space okay this is the TD method in the other method I explained to in for tic-tac-toe what would you do your sample will run all the way to the end right and you use that to update.

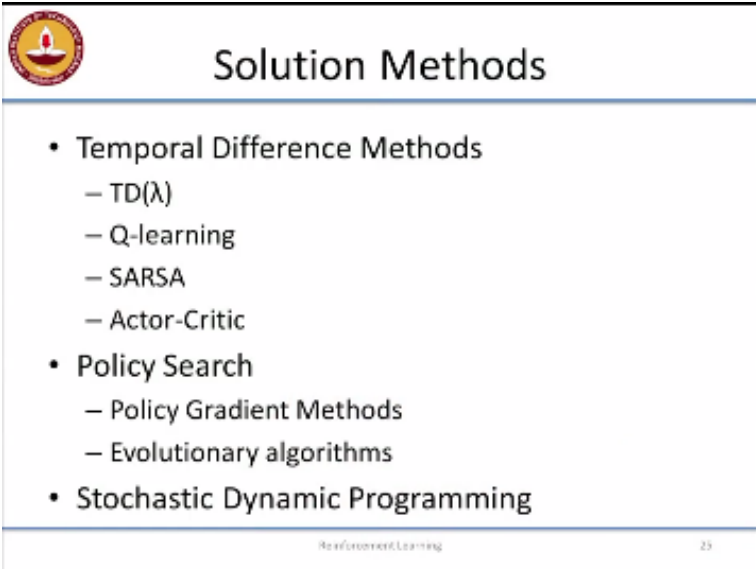
So this is the two different ways of using samples here, so if T will be determined by the value of  $s_{t+1}$  right so the value of  $s_{t+1}$  should be computed for all and so depends on the steepest to again that should be computed first so would not this be the same thing as exploring the hole all the way down and then computing this if you are doing it for the first time right the first time down the tree there will be no updates actually because  $s_{t+1}$  is also be 0  $s_t$  will also be 0 the first time we go down.

There will be no updates but once you reach an end then from there you start updating the previous day, so the next time you go down the tree then it will keep going further up whether in what the game have lost the game I actually I am taking the exact outcome that happened in that particular trail right, at that particular game and I used that to change my probabilities right but here I am not just taking the outcome of that particular game right I am looking at the expected value of winning from the next board position right. So if I wait there all the way here and I say I

won and it take this in update  $s_t$  then I will be only updating it with the fact whether I won or not okay but if I am updating it from  $s_{t+1}$  see I could have reached a steeper so multiple ways before right.

When I am doing the updating from my  $s_{t+1}$  is essentially the average accumulated over all the previous trends that I will be using right, so if I play all the way to the end and update it will be with a 1 or a 0 but  $s_{t+1}$  could be anywhere between 1 and 0 depending on what is the probability of finding so I will be using that value for updation that is a crucial difference I so there are many different solution methods so there are all this which are called temporal difference methods so these are all different algorithms.

(Refer Slide Time: 05:03)



The slide is titled "Solution Methods" and features a logo in the top left corner. The content is organized into three main bullet points, each with sub-bullets. The first bullet point is "Temporal Difference Methods", which includes "TD( $\lambda$ )", "Q-learning", "SARSA", and "Actor-Critic". The second bullet point is "Policy Search", which includes "Policy Gradient Methods" and "Evolutionary algorithms". The third bullet point is "Stochastic Dynamic Programming". At the bottom of the slide, there is a footer that reads "Reinforcement Learning" and the number "23".

- Temporal Difference Methods
  - TD( $\lambda$ )
  - Q-learning
  - SARSA
  - Actor-Critic
- Policy Search
  - Policy Gradient Methods
  - Evolutionary algorithms
- Stochastic Dynamic Programming

TD  $\lambda$  q learning starts actor critic so on so forth and then there is a soul search of algorithms which called policy search algorithms and then there is dynamic programming and their whole bunch of other applications for Aaron it so we could they are all over the place as you could see.

(Refer Slide Time: 05:23)



## Other Applications

- Optimal Control
  - Robot Navigation
  - Chemical Plants
- Combinatorial Optimization
  - Elevator Dispatching
  - VLSI placement and routing
  - Job-shop scheduling
  - Routing algorithms
  - Call admission control
- More
  - Intelligent Tutoring Systems
- Computational Neuroscience
  - Primary mechanism of learning
- Psychology
  - Behavioral and operant conditioning
  - Decision making
- Operations Research
  - Approximate Dynamic Programming
- More
  - Dialogue systems

Optimal control optimization company too common a total optimization for psychology neuroscience so that is not a theory since I was asking is there anyone from biotech because biotech people do use reinforcement learning a lot and usually there are one or two people in the adult class so this is a surprise or maybe that is there was a bear trick with this according to the economic website maybe this is gave up in CS 36 and went back I do not know right so here is the most recent.

(Refer Slide Time: 05:59)



## Game Playing – Arcade Games

Minh et al., Nature 2015



- Learnt to play from video input
  - from scratch
- Used a complex *neural network!*
  - Considered one of the hardest learning problems solved by a computer.
- More importantly *reproducible!!*

Reinforcement Learning

31

The hot thing that comes from RL right more game playing and for a change is not from IBM and it is from Google but the company that actually built the first this arcade game playing engine was called deep mind and as soon as deep mine built a successful engine Google bought them right and so now it is Google deep mind but it is a separate entity right it is not part of Google deep mind operate out of London and they bring all kinds of interesting stuff many of the hot advances very recent advances in the last year or.

So in reinforcement learning seem to be coming out of deep mind so what they did was how many if you know about this Atari games right, everyone knows about Atari games people are getting tired really no one has played pac-man yeah, how about how about the pong break outs pace invaders come on yeah anyway so what happened was this is team in University of Alberta okay, which put out this their what they call the le the arcade learning about the Atari learning environment on arcade learning environment which essentially they allowed computers to play these games right.

These artery games and what the deep mine fellows came up with is a reinforcement learning agent that learn to play this game from scratch I just by looking at the screen okay that is all the input it was getting just the pixels from the screen they are all pixels on the screen but given as inputs to it used to very complex neural network so it is a deep learning deep network right and it is considered of the hardest learning problem solved by a computer and I think I believe it is a one the only computational reinforcement learning paper ever appear in nature right, so usually

is very hard for non natural science people to publish in nature and kind of obviously it is usually Hartford computer science to publish in nature but this was totaled out as a next step in trying to understand.

How humans process in post blah essence of all kinds of marketing jargon right but more importantly than anything else about this it is reproducing them so I told you about I think that is a warning sign form to stop so I told you about the helicopter right so there is basically Stanford and Berkeley or the two people who get the helicopter to fly I told me about the backgammon player that is like Jerry is to are is the one person who gets the backgammon player to work right.

Partly because all the input features he uses in there or proprietary but partly because has a very hard problem to solve right what is the amazing thing about this Atari game engine is that this case our release of code right you can if you have enough powerful GPUs right you can set it up here and get it to play and get a reasonably engine right that plays those places Atari games that is the amazing thing about it that is reproducible as opposed to many of the other things other success stories other success stories you had in the past so I do believe I had just one more slide after this so let us see if this will work.

(Refer Slide Time: 09:20)



Okay.

(Refer Slide Time: 09:24)





Oh so if you are really doubts the green one is the learning is it no, no this just sped up for you to see you mean it is not like the game is progressing at the same rate but you can see the score slow mind you it was not given a reward okay it is just given the screen never got to reward for winning ideas and understand that the pixels on the top or rewarding and if we give it roars it becomes cheating right yeah which is what they did they did they did add a game over which is misty the a heuristics considered as cheating but they did not a game over sign so the longer you keep it going the better.

It is basically nothing this is getting boring right so this is learnt here so this is a sea quest you have to swim and then sink down get some things and come up so sequence is a game that it never learned to do greatly on and sequence is not something which did learn to solve well so there are a few games like this, so they initially published the nature paper I think they had liked like 45 or 46 out of the 50 ale games they were able to play well and I think in 43 of them it had better than human performance and I think the current state is they have like one game that does not play well right and have better than human performance in like 48 of those.

### **IIT Madras Production**

Funded by  
Department of Higher Education  
Ministry of Human Resource Development  
Government of India

[www.nptel.ac.in](http://www.nptel.ac.in)

Copyrights Reserved