NPTEL

NPTEL ONLINE COURSES

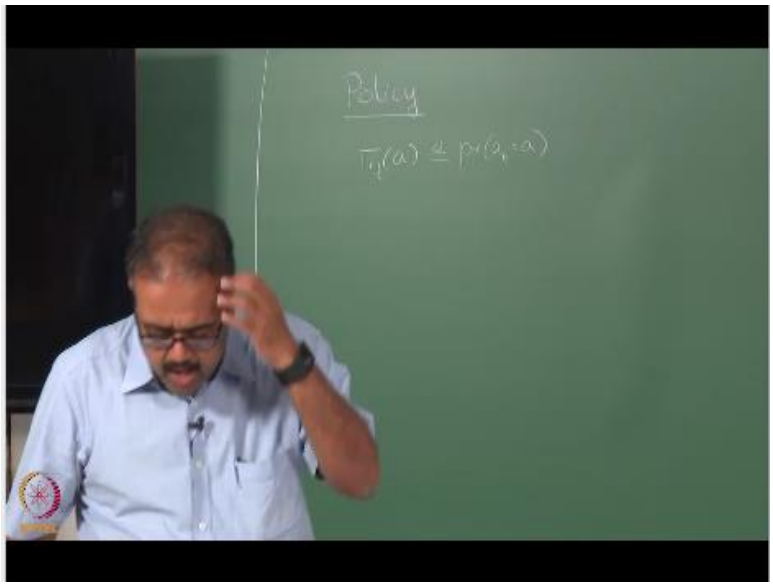
REINFORCEMENT LEARNING

Policy Search

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

Estimated expected value right and likewise at the end to recommend a specific com I also use the estimated expected, value there is a whole other class of algorithms that directly work with the directly work with the, representation of the policy right so what do I mean by a policy here, I told you what a policy was in the very first class right what is the policy, I sense a mapping from States to actions or states to action probabilities that in this case we really have no states right so for we have not considered any states.

(Refer Slide Time: 01:26)



So what would a policy be in this case sorry yeah it could be just one arm or it could be some kind of a probability distribution over the arms right it could be one or more some probability

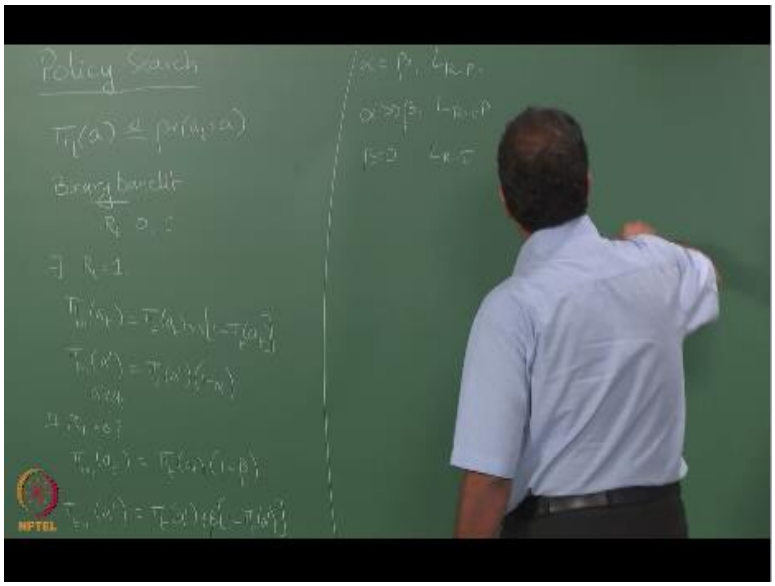
distribution over the arms so I am going to denote this by right, so which is the probability of pulling arm A right and my goal is to keep changing this policy over time, says that eventually I will be recommending with probability one to pull the best arm right.

So this policy is going to keep changing over time so I am going to denote this way so the policy is going to keep changing over time and denote this by π_t right, so now what a learning algorithm has to do in such a case is it has to give me away of changing this π_t with time right, so one way of thinking about things like epsilon greedy right or soft max is that they define this π_t in an indirect fashion right if you remember I wrote it down also right.

I know I wrote down what π_t would be for epsilon greedy and soft max I started off by writing in force soft max then it came back and wrote it down for epsilon greedy right people remember that I said what will be the probability of picking I don't know if I use the π notation that they use the π notation yes, I use P the probability of picking action race I didn't because I didn't introduce the π notation I did not think I used it but essentially the probability of picking an action I have written probability at=a right, that is exactly what $\pi_{t(}$a) is right.

So this is, it is a probability that at=a to this is additional notation we will be using the notation π throughout right, so later we will use will condition this on this states also that lessons the probability that at=a given st=s right, here since there's only one state we do not have to do that great, so some of the earliest known approaches for solving bandits use this kind of a direct policy search right.

(Refer Slide Time: 03:53)



So some kind of us policy search approach right so in that what we do is, so let us let us keep it simple I will talk about a two action case right and then we can talk about other things right so the very very simplified the situation which is called the binary bandit where there are only two outcomes either you get a 0 or you get a 1 right, so is a good outcome 0 is a bad outcome this just makes me and this is going to give you like the historic perspective here okay.

So in fact it is so historic that in addition to they got rid of all of this discussion in the book, right so if you want to know more about the kind of things I am talking about right now we'll have to go back and read addition one of the RL book, so let us say I'm talk about binary banditos Rt can be either 0 or 1 okay, and let us say I have a two armed bandit case here so I have only armed one or arm too just to make my thing easy.

So if right at some time t Rt=1 right, then what I will do is I will make, π t+1 of at essentially the action I took at time T right at is action at took at time T and the reward I got was 1 so what should I do so increases all right so how will I do that right, so if α is small enough ok, so the best action will eventually converge to a probability of 1 if after α is very large I will start oscillating this will this might not be probabilities all those problems are the right.

And what about the other case if this is this if I am doing this, the action some other action a prime what should I do for that right, I mean ideally it should be +α times 0-π so this will work if it is two actions which multiple actions what happens, it not be a probability distribution I will have to make sure that everything stays probability distribution, where is the multiple actions have taken away α from everybody then I should take away α/n-1 from the other actions okay.

So that when I sum up everything after that also it will still be 1, starting off with assumption that π_t sums to 1, so I need to make sure $\pi_{t+}1$ also sums to 1, so I will have to adjust these numbers accordingly okay, so I am pretty sure all of you can do the character adjustment, so this is essentially a one update rule one sec what happens if I got 0 the reward of 0, this one has nothing to do with the reward okay.

In fact this one here has nothing to do with the reward the one here it has to do with the fact that if the probability I would like this probability to be one you remember the form of the rule I told you right, current value target minus the current value, so the error right so current value plus α times error so that is the thing it's the same way here the probability I want is 0 okay, so it is the current value+α times 0-π, so I simplified that I wrote it as 1-α right.

So I'm always writing it in the same stochastic averaging form right, so the one is the target I want this not the reward right, so what i got reward of zero what do, I do to flip it around right it turns out that well that is one valid option there are many options that you can have right, so just look this around right, I reduce the probability of the action at took at time T I increase the probability of faction I took I didn't take at time T right.

So why do I have to do this just to make sure it is a probability distribution right when I reduce this probability this will automatically have to go up, it is not like I am rewarding the action I didn't take right, but the fact we have this constraint of the probabilities have to sum to 1 we will automatically reward the other action, okay so but notice i wrote α and β, so depending on the relationship between α and β.

So you have different kinds of algorithms right so if $\alpha = \beta$, so we have something called so they called linear reward penalty LRP algorithm okay, so it's this these things have been around for many many many decades ok, so the LRP algorithm is a linear because the relationships there is are linear so you can see there are no higher order terms in the updates that you right, so you have you do something you change the parameters on a reward and you also change the parameters when you get a penalty right.

So the reward is +1 penalty is 0 and you are changing them by equal amounts right, that is when $\alpha = \beta$, so when α much greater than β right, remember these are relative terms okay, when I say α much greater than β still α is very small and I expect α to be like 10^{-3} or 10^{-2} or something like that right even though I write alp α much far greater than β that means beta should be 10^{-5} or 10^{-7} something much smaller right this is called epsilon P.

So I do something when I get a reward right I do a much much smaller change when I get a penalty, any idea what the I stands for yeah, it sets or in action but yeah you get the idea right so when rewards comes I do something when the penalty side of it happens I do nothing, in action I just leave the probabilities as it is okay and it turns out that there is very very small change rate the algorithm itself is the same all I am doing is changing these things around and it turns out the convergence behaviors of these three algorithms are very different right and depending on how how rewarding your arms are right.

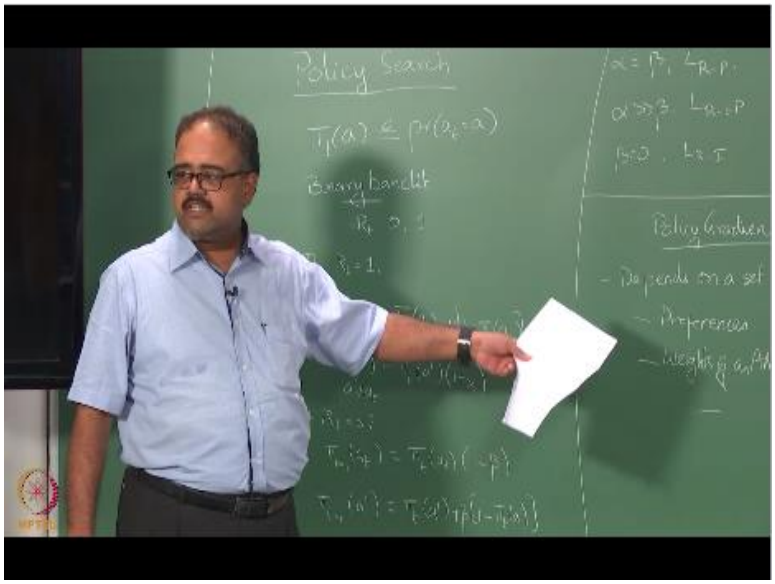
So for example my best arm let us say has a probability of a binary bandit is 0 or 1 let us say my best arm has a probability of say 0.9of coming up as one, right and the other arm has say a probability of 0.6 of coming up with one, then LRP will work well right, but suppose my best arm has probability of 0.3 of coming up of one right and my bad arm my worst arm has a probability of 0.2 of coming up as one or 0.25 as coming either one LRP is going to have a hard time infect it turns out that in such cases LRP works better because whenever a small reward comes also you whenever the reward comes right you keep hiking the probability, because the probability of getting the reward is so small right.

So if I am doing a something on the penalty side I always almost always be depressing my probabilities I will just keep decreasing the probabilities and increasing the other probability so there will be a lot of oscillations right, so in cases where rewards are scarce to come by its best to ignore the penalties so there are many many things like this so yeah, so that is still a very vibrant community that works with this kind of learning automaton ideas and this is the most basic most fundamental of these things I told you about this variable structure stochastic automata right, as I asked you about finite state machines and you all of you said you knows finite state machines right and I told you then there is this variable structure finite state machines as people use for solving.

So these are kind of the culmination of the variable structure finite state machines right so you can go down and write it more explicitly as to how it turns out to be automated right, but this is more of a compact way of representing those variables structure finite automata so I am not going to get into too much detail even though there is a vibrant community that does this, right and so this is just to tell you that this history of using this policy representations directly right goes back several decades in fact the earliest such learning automated method was proposed still the 1930s, infect these approaches even predate some of the value function value estimate based approaches we are talked about okay right.

So fine so we're all on board here so taking this further now this looks like some kind of ad hock way of coming up with the policy updates right, and it seems very natural way of doing it why I get a reward I increase the probability ok if I get a penalty I decrease the probability seems like a very natural way of doing it but is there a more systematic way I was thinking about learning the policy directly right.

(Refer Slide Time: 16:44)



So very well steady and increasingly a direction which is receiving a lot of attention call policy gradient approaches the book calls these parameterized policy representations, while the earlier one were you know value function based methods they call this parameterized policy approaches, so the basic idea here is I am going to assume that my policies right, so here I am giving you the policy has explicit probability values right.

So I'm going to assume that my policy depends on the policy depends on some set of parameters, so what do I mean by this, so policy is a probability distribution right I can specify the probability distribution in many ways for example I can specify the probability distribution like we did earlier by a soft max function right, so if I specified by a soft max function what is it that I need to give you in order to define the policy people remember soft max.

So e power something divided by summation e power something right, so suppose I am saying that I need to tell you what the soft max function is corresponding to my current policy, right so what is information that I should communicate to you, so all those exponents that I am putting in there right for each action I have something I put in the exponent I will have to tell you that right, it need not be values, just remember very much so I am just talking about the soft max function

right however I derive the value that goes into the exponent it can be a value function it can be something else right.

I could have just arbitrarily pulled out some numbers I could have said action one put a 3 in the exponent ok action two to put a 15 in the exponent action three put a 22 and the exponent okay this gave you those numbers arbitrarily but this defines, a policy however i arrived at those numbers this defines a policy right, so this is what we call as a parameterize representation, so i have the set of parameters ok that defined the policy for me so I the parameters are somehow used to generate these numbers for me right.

So this is essentially what we mean by parameters so we have a set of parameters said there could be anything there could be those values that I plug into the exponents which sometimes are called preferences okay, so that e power thing right, so I can just say I prefer action a to action b right and then I can have some way of converting that into actual probabilities, and I would say my preference for action is 25 preference for action B is 13 right and from that I can convert it into probability.

So far because the value has a separate semantic associated with it right the value is the expected payoffs that I am going to get on pulling the arm when I say the preference for the arm is 23 does not have need necessarily have any semantics so 23 verses 15 so 23 higher than 15 there is only semantics I have, right not necessarily that 23 means that this is the payoff i am going to get so this gives me a lot more flexibility and how I change those values, right. it need not necessarily come from the estimation of the expectation okay right.

So it could be preferences it could be say the weights of a neural network if you have so inclined, right so this is something which all of you should be thinking about how many of you read this article about the go playing isn't one two therefore five six seven not enough people everybody go read the go playing agent about α go go go go go to Google search for α go it is there in the Hindu for crying out loud right, anyway.

So this is this machine learning agent which they do not actually specify exactly what is the machine learning algorithm that they use it's a machine learning agent that learns to play this game of Go which fora long long long time was considered one of the hardest games for computers to play because the number of patterns that you could have or just huge, right and rules are very simple but then it turned out to be a very very hard game for our computers to play certainly not at human levels, right but then Google deep mind has come up as soon as say Google deep mind you should know what algorithm that they use their what at least what how they formulated the problem right.

So they use reinforcement learning and the deep neural network to train a go agent that actually beat the European champion 50, I mean they played a series of five matches and it beat the champion or all five of them, right and they are hoping to have a matchup with the world champion go player, in March I know that mentioned some dates March yeah and maybe they still do not hope to beat the world champion because apparently the world champion player is at a much higher level than, the all the other players out there but still it so that is hope right.

So the point is RL plus neural networks seems to be the hot thing nowadays right so everybody wants to do that, right so this is not hesoteric example if you are going to gout there and do reinforcement learning this is probably what you will end up doing right, so where the parameters would be given by a weights of a neural network, right and I'll give you other examples as we go along right so even this you can think of as a parameter is representation right okay here is question for you.

So what kind of a probability distribution are we talking about here, binomial or baronial no limit depending on whether it is repeated trials are one trail right, so Bernoulli if I had multiple actions should have been multinomial right yeah so binomial multinomial so what is the equivalent version of Bernoulli, actually mentioned this in ml plus guys you know whether the ml should be able to tell them I'm not talking about the conjugate price verse I am saying for repeated experiments to outcome repeated experiments is binomial single trail is Bernoulli only multiple outcome repeated experience the ex-experiments is multinomial single experiment is

categorical, okay distribution called the categorical distribution which is essentially multiple outcomes single single experiment probability distributions okay but anyway.

So what are the parameters that describe binomial distribution probability of one or something so that is essentially that so this is again one way to think about this is this is also a parameterize distribution, where I am specifying directly the parameters of the multinomial or the binomial in this case I just said a prime i could say says all a prime not equal to at and then I have addressed this α then I basically made it into a multinomial case right.

So this is also parameter thing so I could specify any kind of distribution like this right so now what I am going to do is, just like we did here i am going to modify the policy parameters directly, right instead of modified the value functions i will modify the policy parameters directly what would be a systematic way of doing this we can come up with many things right so but I am talking about gradient based approach.

IIT Madras Production

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

www.nptel.ac.in