STOCHASTIC APPROXIMATION: THEORY AND APPLICATIONS

Dr. Gugan Thope

Department of Computer Science and Automation

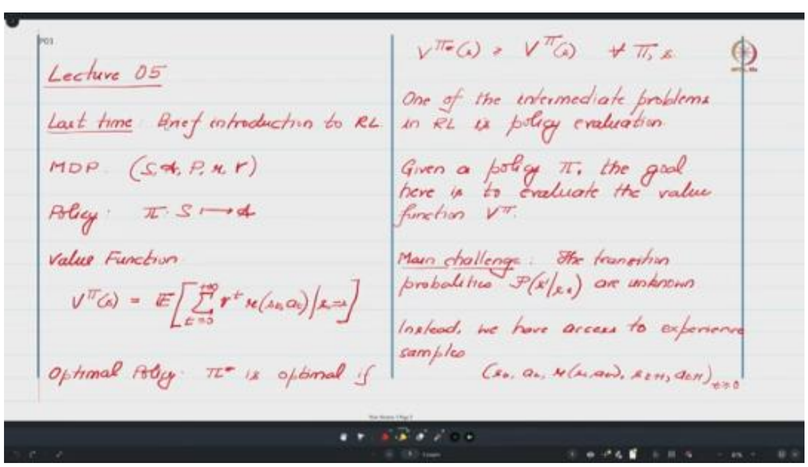
Indian Institute of Science

Lecture 5

Policy Evaluation in Reinforcement Learning

Hello and Namaste, everyone. Welcome back to Lecture 5. So, let us do a quick recap of what we have covered so far. We have looked at two examples of stochastic approximation algorithms, and both of them turned out to be special cases of SGD itself. And then we said stochastic approximation can actually go beyond SGD methods. Towards that, we said that these examples are quite common in the context of reinforcement learning.

In the previous class, we gave a very brief introduction to reinforcement learning. In particular, we spoke about what is called a Markov decision process. We said that an MDP is characterized by five things, which include the state space denoted by S, the action space denoted by calligraphic A. The transition probability kernel denoted by P specifies the probability of moving from state S to state S prime under action A. Then you have the reward function denoted by R, which tells you the immediate reward that you would get when you take action A at a particular state S.



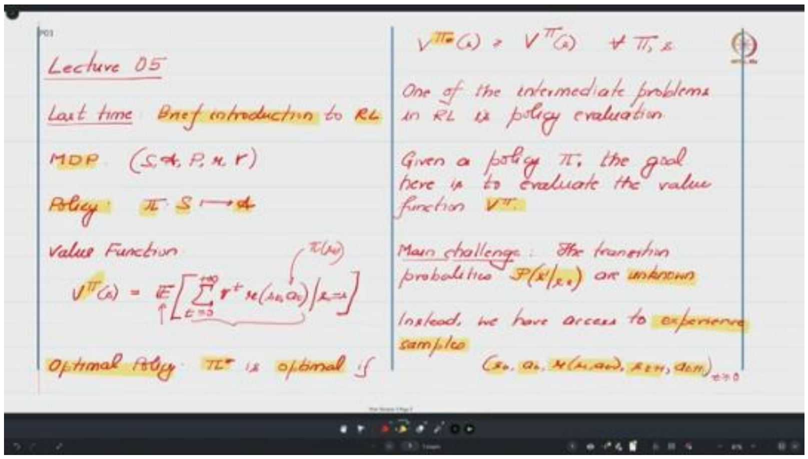
And finally, you have the discount factor gamma, which in some sense quantifies the relative value of a unit reward obtained at different time instances, right? Then, we discussed something called a policy. A policy, at a very high level, tells us how to act—in particular, at a given state, what action to take. Formally, a policy is a function that goes from the state space to the action space. To keep things simple, I am only focusing on policies that are deterministic in nature.

However, we could also have more sophisticated policies, which, on the one hand, are stochastic in nature and could also depend on history, and so on and so forth. Now, once we have a policy, we would like to know how good that policy is. And one way to quantify how good a policy is, is by looking at its value function, which is a vector and which we denote by V^{π}. This is a vector whose size equals the number of states in your state space, and the s-th coordinate of this value function is given by the expected value of this cumulative discounted sum of rewards starting from state s, and the key thing to note here is that the action a t is chosen. As per the rule specified by your policy pi, that is AT equals pi of ST.

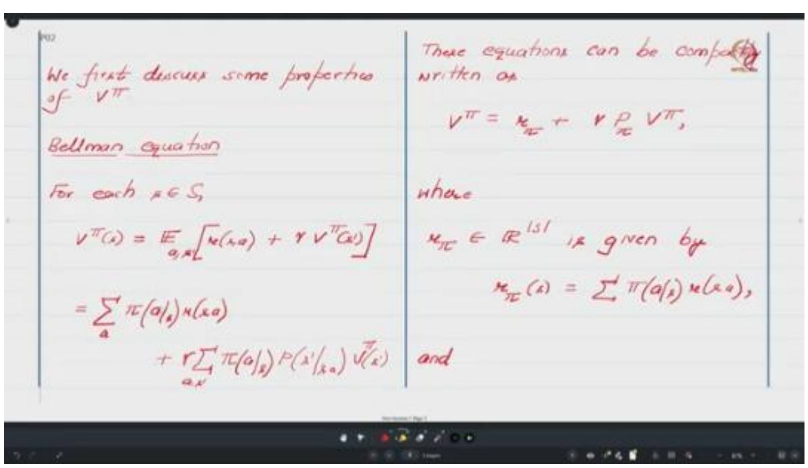
Then we discussed this concept of an optimal policy. We said that a policy pi star is optimal if V pi star of S is greater than or equal to V pi of S for all pi and all S. This means that your optimal policy would give you the highest returns in this cumulative discounted sum sense starting from any state S. So, of course, one of the key problems of reinforcement learning is finding such an optimal policy pi star. However, one of the intermediate goals also includes policy evaluation, which means that given a policy pi, what is V^{π}?

So, we will now come up with an algorithm to estimate V^{π}, and then we will see that this algorithm cannot be viewed as an SGD, that is, a stochastic gradient descent method, right? But before we go over there, let us first understand the challenges in estimating V^{π}. The main challenge in estimating V^{π} is that this expectation that you have over here is dependent on these transition probabilities, which are unknown. And the goal, in some sense, is to make use of the experience samples, that is, ST, AT, R of ST, AT, ST plus 1, and AT plus 1, right? So, which basically means you start at the state, you take an action, you see an immediate reward, and then the environment makes a transition to the next state, which is ST plus 1, and then again you take the next action.

So, this is known as an experience tuple. And you get hold of these experience tuples by interacting with the environment, right? So, these are stochastic or noisy in nature. And one would like to make use of these experience samples to estimate V^{π}, right? So, let me again summarize.



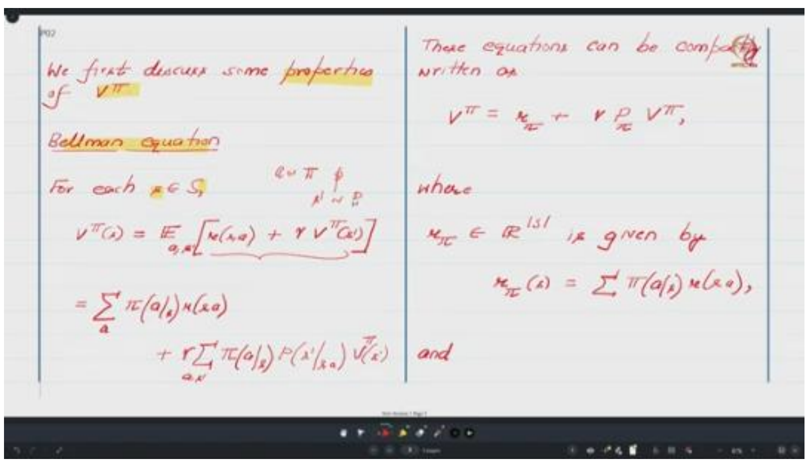
The challenge in estimating V^{π} is that the transition probabilities, and hence this expectation that is there in V pi's definition, is unknown. Instead, we have to make use of these experience tuples to get hold of V^{π}. So, before we understand how to estimate V pi or go about estimating V^{π}, let us first look at some key properties that V^{π} satisfies. So, why V^{π} satisfies these properties we would not be discussing in this course; for that, I request you to take a look at a course on reinforcement learning or maybe you can directly look at some textbooks on reinforcement learning. So, one of the key properties that V^{π} satisfies is what is called the Bellman equation.



So, what is the Bellman equation? Well, for each state S in the state space S, your V^{π} of S satisfies an equation of the following form. So, let us look at what is there on the left-hand side. On the left-hand side, you have the Sth coordinate of V^{π}. And on the right-hand side, you have an expectation of this random variable, right?

$$V^{\pi}(s) = \mathbb{E}_{a,s'}[r(s,a) + \gamma V^{\pi}(s')]$$

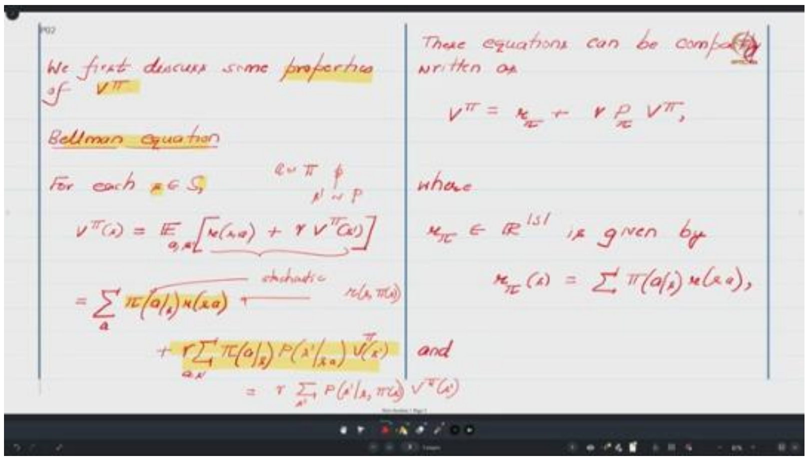
In this random variable, the things that are random are the action A and the next state S prime, right? And how is this action A chosen? Well, this action A is chosen according to your policy pi, right? And your S prime comes from your transition function P, right? So, this expression over here that is this expectation can be expressed in the following way that is this expectation of RSA.



Notice that it does not depend on S prime. Hence, it is sum over A pi of A given S RSA. Now, here I have presumed that your policy pi is actually stochastic. However, in the deterministic case, this expression over here would just be r of s comma pi of s. And the second quantity that is gamma times V pi of s prime its expectation would be this quantity over here.

$$= \sum_{a} \pi\,(a|_s) r(s,a) + \gamma \sum_{a,s'} \pi(a|_s)\, P\big(s'|_{s,a}\big) V^{\pi}(s')$$

So, you can see that this is sum over a comma s prime pi of a given s p of s prime comma s given a times V pi of s prime. So, again in the deterministic case, this expression would equal gamma times sum over S prime P of S prime given S comma pi of S times V pi of S prime. So, V pi actually satisfies these Bellman equations and you can see that we have an equation for each little state s in the state space. And these equations can be compactly written as V pi equals r pi plus gamma times V pi times s. sorry, gamma times p pi times v pi, right, where r pi is a vector whose size equals the number of states in the state space and its s-th coordinate is given by this expression over here, which is sum over, sorry, which is the sum over a pi of a given s times r sa.



Again, in the deterministic case, this will basically be r of s comma pi of s right.

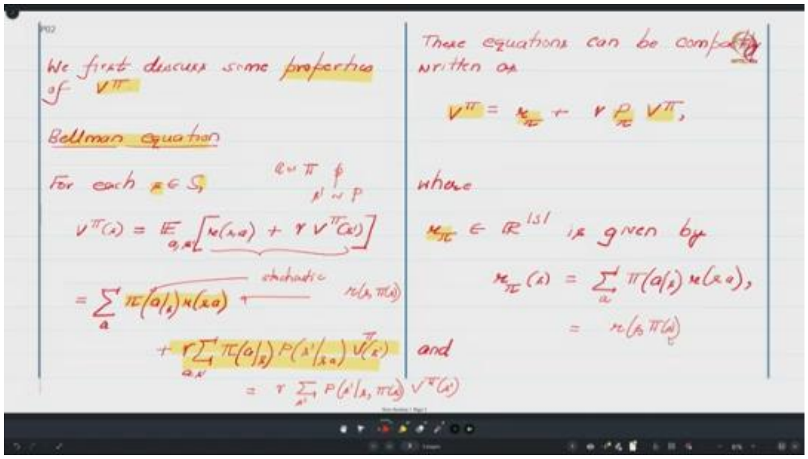
$$V^{\pi} = r_{\pi} + V P_{\pi} V^{\pi},$$

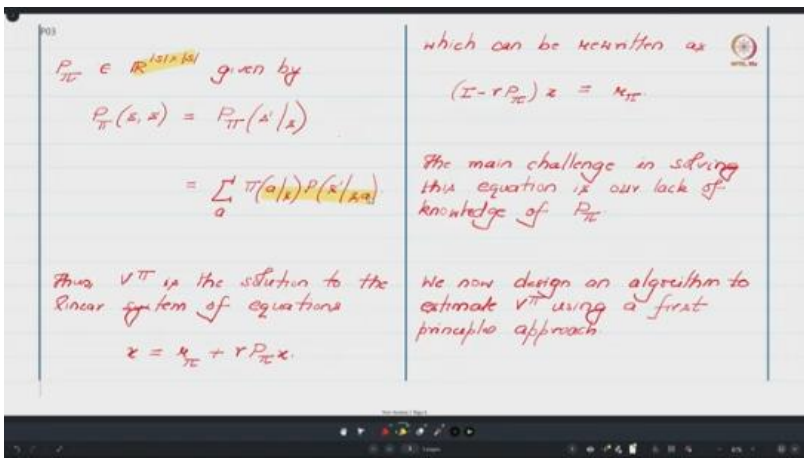
$$r_{\pi} \in \mathbb{R}^{|s|}$$

$$r_{\pi}(s) = \sum_{a} \pi(a|_{s})\, r(s,a),$$

$$= r\big(s, \pi(s)\big)$$

So, this will be the s th coordinate of this vector r pi and then you have gamma times p pi. Now, p pi is matrix of size s cross s and its s s prime entry is given by this expression that we have over here right. So, you can see that in the stochastic case it is sum over A pi of A given S times P of S prime given S comma A.



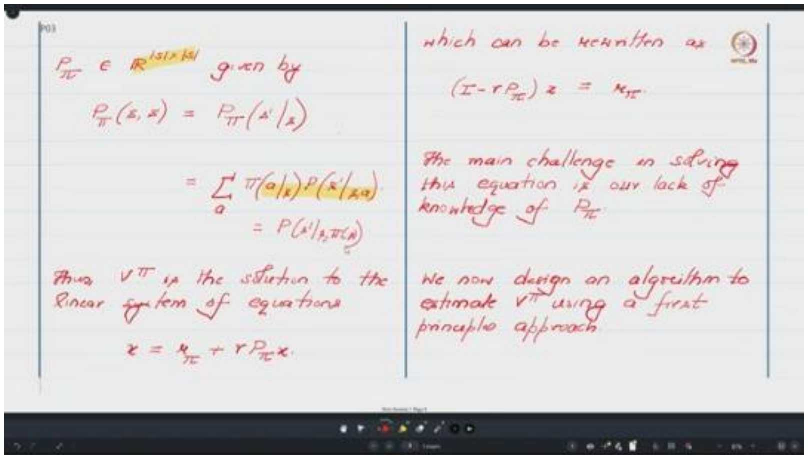


$$P_\pi \in \mathbb{R}^{|s|\times|s|}$$

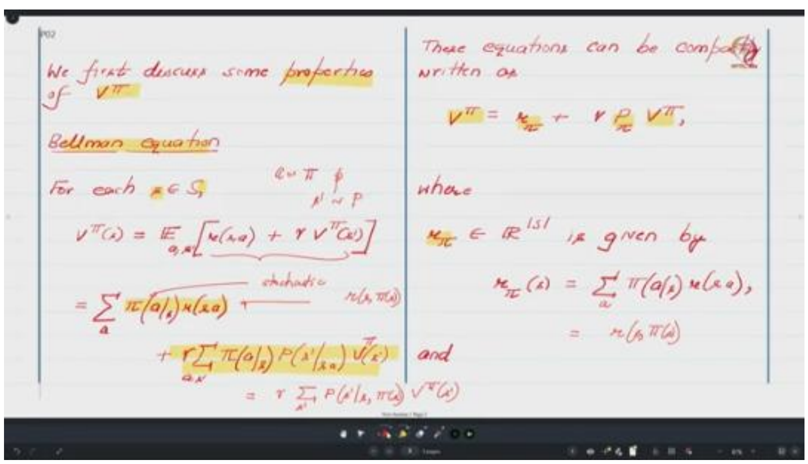
$$P_\pi(s, s') = P_{\pi(s'|_s)}$$

$$= \sum_a \pi(a|_s)\, P(s'|_{s,a})$$

So, basically you can think of this as all possible ways of starting from S and going to S prime. So, you start at S, pick an action according to the rule specified by pi, right? And then once you pick an action, you go to S prime with probability S prime given S A. Again in the deterministic case, this expression would basically be P of S prime given S comma pi of S, right? So, when I say deterministic case, I mean the policy is deterministic.



So, in this sense, so let us go back to this equation. So, we have this compact way of representing the Bellman equation. And one can see that because of this reason, one can interpret V pi as the solution for to a linear system of equations given by x equals r pi plus gamma p pi x. So, notice that r pi and p pi I have retained as it is and wherever v pi was there I have replaced it with x. So, we want to find V pi.

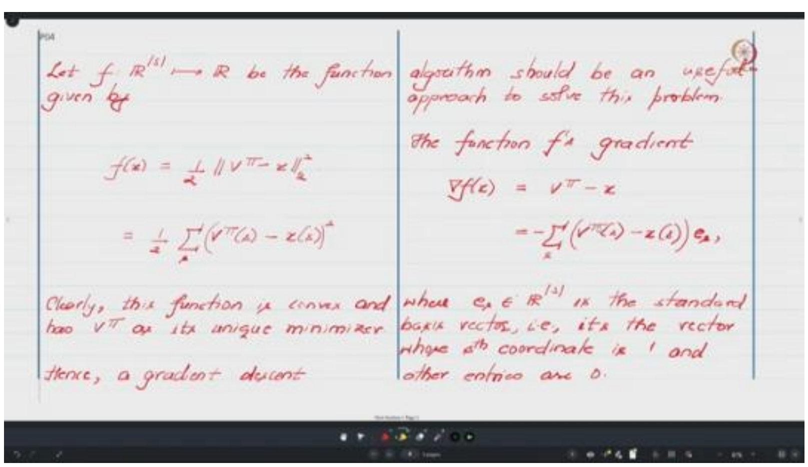


So, we think of that as the unknown, and once you know pi, you can figure out what r pi and p pi are when the reward function and the transition matrix are known. So, this system of equations can now be rewritten as I minus gamma p pi times x equals r pi. So, now we want to solve this linear system of equations so that we can find V pi. So, let me summarize what we have done so far. Our goal was to do policy evaluation, that is, given a policy pi, figure out how good it is. One way to quantify how good a policy is to estimate V pi, and then we saw that V pi.

satisfies some Bellman equation, which can be reinterpreted as some linear system of equations given by I minus gamma p pi x equals r pi. So, x here is the unknown, and V pi is the solution to this linear system of equations. So, if you solve this linear system of equations, we can get hold of V pi. So, under standard conditions, one can show that this system of equations has a unique solution, which is V pi. So, I will not go into all those details.

So, one can show all those things, right? So, again, one can ask if you have a linear system of equations and there is a matrix and a vector. So, the matrix here is I minus gamma p pi, and the vector is r pi. So, given the matrix and r pi, why can we not solve them? So, one of the main challenges in solving such an equation is that often in practice we will not have access to this p pi.

So, without knowing P pi, how can we solve this system of equations? That is the question we would like to answer. And what we will do now is design an algorithm to estimate V pi using a first-principles approach. So, to understand this, hopefully, you will not need much background. So, what we will do is first begin with defining a function f which goes from RS to R. So, by RS, I mean it takes as input a vector x whose size equals the number of states in your state space, and this function spits out or outputs a real number.



Right? And how do we define this function? Well, this function is defined using the formula given over here, which is f of x equals half of the norm of v pi minus x squared. And for simplicity, I will be working with the L2 norm. Right?

$$f(x) = \frac{1}{2}\|V^{\pi} - x\|_2{}^2$$

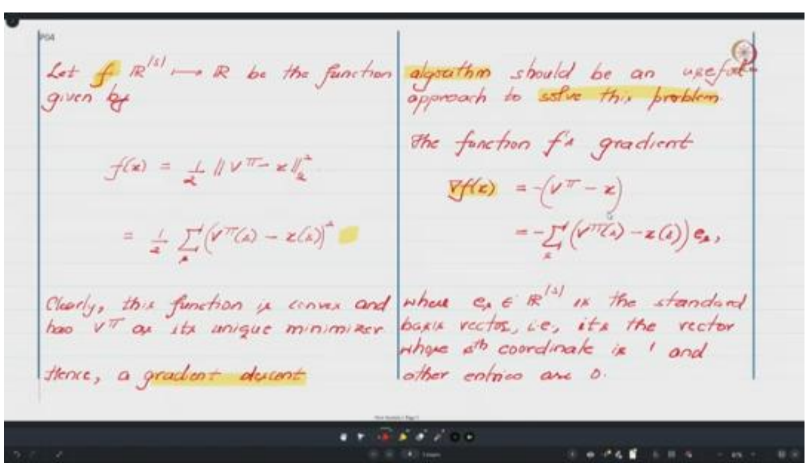
And this expression—I hope it is not difficult to see—can be expressed using the formula given over here. That is, half the sum over S of V pi of S minus X of S, the whole squared.

$$= \frac{1}{2}\sum_{s}\left(V^{\pi}(s) - x(s)\right)^2$$

And it is not difficult to see that this function is convex, and under standard conditions, one can show that V pi is the unique minimizer of this function. In particular, if you substitute x equals V pi, you will get 0. And because we have this function,

Right? One can see that the gradient descent algorithm is a useful approach to solve this problem. Right? So, let us look at the gradient of this function f. The gradient of this function, I hope it is not difficult to see that it is—sorry, I think there is a minus sign missing.

It should be minus of v pi minus x. And, this I can rewrite it in the following form: that is, minus sum over s V pi of s minus x of s times Es, where Es is the standard basis vector in the Euclidean space of dimension equal to the cardinality of the state space. So, what is the standard basis vector? In particular, what is Es? Well, it is the vector whose sth coordinate is 1 and other entries are all 0.

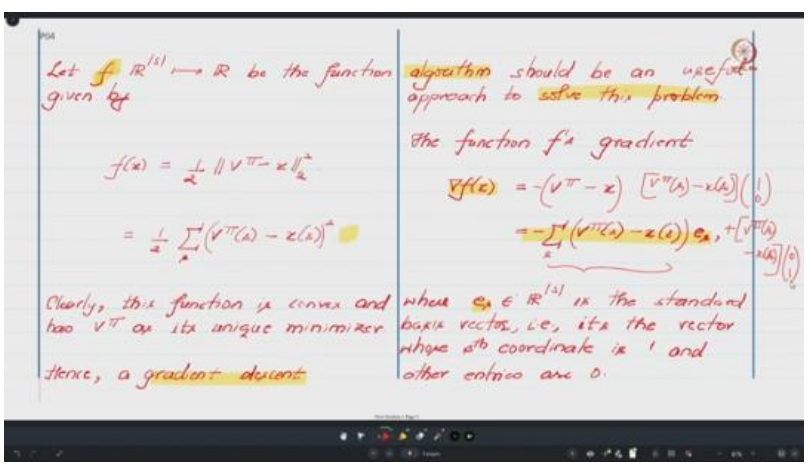


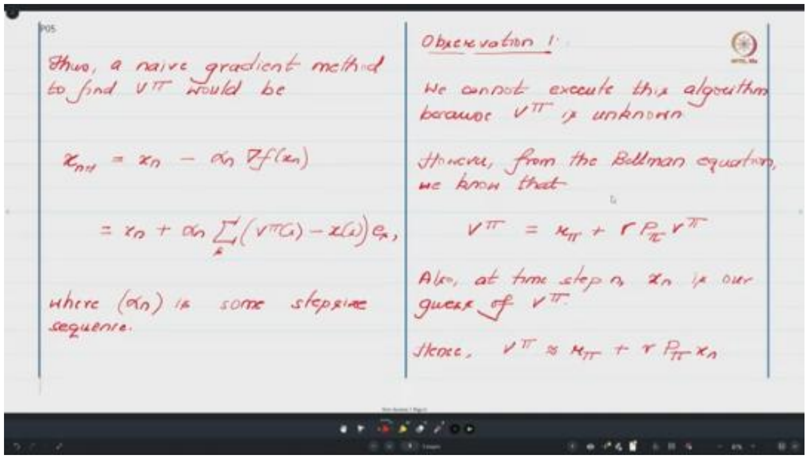
So, you will soon see why I have written the gradient in this expression. On one hand, it may look complicated, but you will soon see why we have written it in this way, and it is not difficult to see why these two expressions are the same. Like in the two-state case, what I am saying is that this expression would be V pi of, let us say, S1, the first state, minus X of S1, and in the standard basis vector would be 1 0. plus V pi of S2 minus E of—sorry, minus x of S2 times 0, 1.

$$\nabla f(x) = -(V^{\pi} - x)$$

$$= -\sum_{s} \left(V^{\pi}(s) - x(s)\right) e_s$$

So, you can see that because the second entry in this vector is 0 and the first entry in this vector is 0, if you add them up in the two-dimensional case, it will indeed equal this expression that we have over here. So, you can see that indeed the gradient of f for this function that we have defined on the left-hand side equals the sum of these terms. Thus, a naive gradient descent method to find V pi would be the one that is specified in this equation over here, that is, xn plus 1 equals xn minus alpha n times grad f of xn, right? The minus over here comes in because we want to do a gradient descent, that is, we want to move in the direction that is opposite to that of the gradient, hence we have this minus sign. Now, if you go back and look at this expression for grad f of x and if you substitute it over here, I hope you agree that we would end up with this expression, that is, xn plus 1 equals xn.



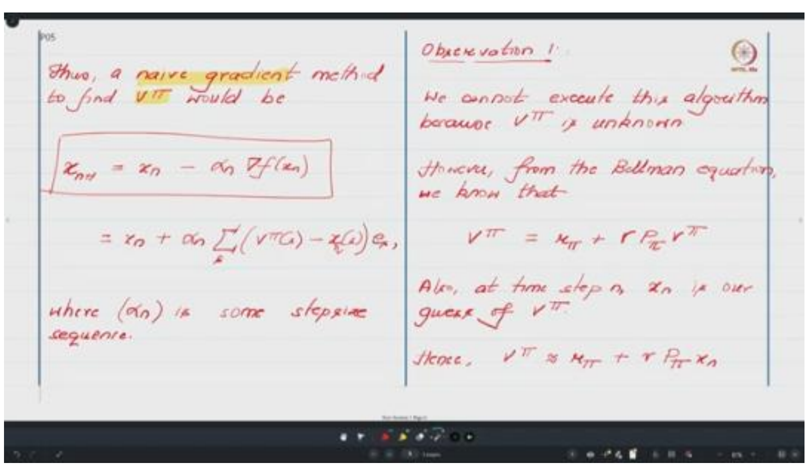


So, xn plus 1 equals xn plus alpha n sum over S V pi of S minus—there is a typo here—this should be xn of S times E of S. And alpha n over here, for the time being, we will presume is some arbitrary step size sequence.

$$x_{n+1} = x_n - \alpha_n \nabla f(x_n)$$

$$= x_n + \alpha_n \sum_s (V^\pi(s) - x_n(s))\, e_s$$

So, one can see that we have a gradient descent algorithm, and can we execute this algorithm? Of course, we cannot execute this algorithm because we do not know V pi. So, the goal of this algorithm was to find V pi.



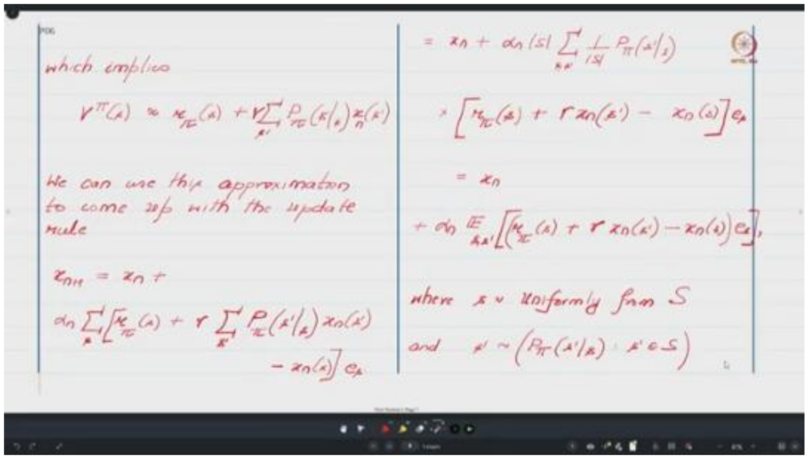
So, we cannot have an algorithm which makes use of V pi. So, of course, this algorithm cannot be executed when the goal is to find V pi, but hopefully, we can build upon this update rule by making use of some properties of V pi. In particular, as I told you before, V pi satisfies a Bellman equation, which is rewritten over here. So, V pi equals r pi plus gamma times p pi V pi, right. Now, at time step n, right, at time step n, let us say we somehow did something and figured out an estimate of V pi in the form of xn.

So, we do not know V pi, but V pi satisfies this right-hand side. Again, this right-hand side contains V pi. We do not know V pi. However, we have a guess for V pi at time step n, which is x of n. So, hence we can say that V pi at time step n can be approximated by this quantity that we have on the right-hand side, which is r pi plus gamma p pi x n.

$$V^{\pi} = r_{\pi} + \gamma \, P_{\pi} V^{\pi}$$

$$V^{\pi} \approx r_{\pi} + \gamma P_{\pi} x_n$$

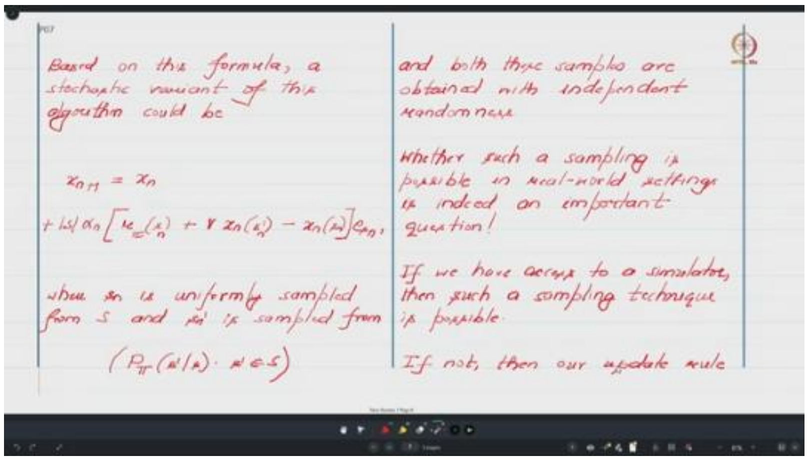
How good this approximation is will depend on how close x n is to V pi, but nevertheless, one can think of this right-hand side as an approximation. So, you know, this approximation—if you look at the s-th coordinate of it—one can see that V pi of s will then approximately equal the s-th coordinate of the right-hand side, and the s-th coordinate of the right-hand side is the s-th coordinate of r pi plus gamma times the sum over s prime p pi of s prime given s times xn of s prime. So, if you go back to our previous algorithm that we had written over here, we could not execute this algorithm because we did not have access to V pi. However, we now have an approximation to V pi, and hence, as a first guess, we may say, 'We do not know V pi; how about we replace V pi with an approximation?' And this would lead us to this update rule, which is xn plus 1 equals xn plus alpha n times the sum over s, and wherever you had V pi of s, we have now replaced it with its approximation, and the rest of the entries remain as before.



So, now what I will do is I will massage this update rule into a slightly different form. In particular, the first thing I will do is multiply and divide by the cardinality of the state space, right? And then I will pull this expression out and write it, you know, as a multiplying factor to this expression over here, right? So, I can pull this out. You know, despite this quantity over here not containing P pi of S prime given S, the reason being P pi, you know, is a row stochastic matrix, which means that if you look at the entries of each row, they will be non-negative and they will add up to 1, right? And since this quantity does not include S prime, when I sum this expression over S prime, okay, that will be 1, and we will get back this expression. So, the point I am making is that I can indeed pull it out in this fashion, and I am not changing what we have over here.

So, once I write it in this fashion, I can reinterpret this expression as an expectation of this expression, where the random variables are actually S and S prime. And the distribution of S is that it is uniformly sampled from the state space, and S prime is sampled from this distribution over here. So, let us go over it slowly. So, we started out with this naive gradient descent method. This algorithm cannot be implemented because the purpose of this algorithm is to estimate V pi.

However, we notice that V pi satisfies this Bellman relation, and at time step n, we would have some approximation or some knowledge of V pi in terms of Xn. Hence, we can replace this V pi by Xn, and in particular, we can get hold of an approximation to V pi of S. And once we plug in this approximation and do some massaging of the update rule, we can see that the update rule looks like Xn plus alpha n times the expectation of this quantity, where the random variables are S and S prime, and S is sampled uniformly from the state space S, and S prime is sampled from this distribution. Now, because we have an expression involving an expectation, we can drop that expectation and replace it with a sample, and that would give us a stochastic variant of the algorithm. And that algorithm is what I have written over here.

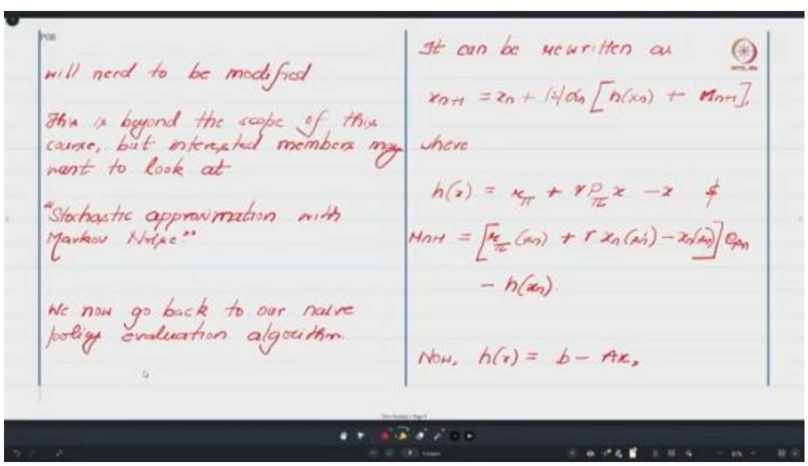


So, this is xn plus 1 is xn plus the cardinality of the state space times alpha n times the square bracket. Now, notice that the difference between this update rule and what we had before over here is that earlier we had this expectation, and now I have replaced that expectation with a sample. And what is this sample? Recall that the expectation was with respect to the current state and the next state. And we will presume in this sample that Sn is uniformly sampled from the state space S, capital S, and Sn prime is sampled from this distribution.

And for simplicity, we will presume that both these samples are obtained with independent randomness. So, of course, as a person listening to this video, a natural question that may arise in your mind is whether such a sampling strategy is possible in real-world settings. So, the short answer to that is often it will not be possible. However, in some very special cases, for example, if we have access to a simulator, then indeed such a sampling technique is possible. Like, for example, let us say you have access to a chess engine, which means that you can start the game at any particular scenario, which means some configuration of pieces.

So, that will correspond to a state S. And we can presume that the state this chess engine actually gives us is uniformly sampled from all possible states. And then, you make a move at that particular configuration of pieces, and then the computer plays its move and leads you to the next configuration of pieces, which will correspond to S prime, right? So, the way you make your move would correspond to this pi over here, and the way the computer makes its move will correspond to the unknown transition dynamics captured by this symbol capital P. So, in this special scenario where we have access to a simulator, such a sampling technique is possible.

If not, then our update rule will need to be modified. So, I want to keep this discussion very simple, and hence I would not be discussing this sophisticated case. However, if you are interested, you may want to look at something called stochastic approximation with Markov noise. So, as you get more familiar with the contents of this course, you will be able to read on your own and handle more sophisticated real-world scenarios. We will have to design more sophisticated algorithms, and one such algorithm is the one that involves Markov noise. So, we will not discuss this in this course.



So, let us now go back to our earlier discussion, which, you know, looks at this naive policy evaluation algorithm. So, the naive policy evaluation algorithm was basically Xn plus 1 equals Xn plus the cardinality of S times alpha n times something. And whatever expression was there in that square bracket, we will now rewrite it as h of xn plus mn plus 1, where h of x is this expression over here, and mn plus 1 is what we had in the update rule minus h of xn.

$$x_{n+1} = x_n + |s|\alpha_n[n(X_n) + M_{n+1}],$$

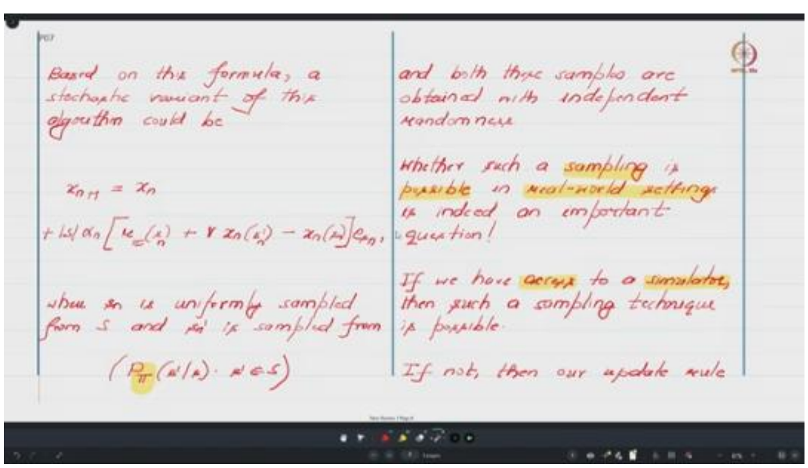
$$h(x) = r_\pi + \gamma P_\pi\, x - x$$

$$M_{n+1} = [r_\pi(s_n) + \gamma x_n(s_n') - x_n(s_n)]e_{s_n} - h(x_n)$$

So, let us go back and see what the update rule was. So, this was the update rule. This update rule is a stochastic update rule, and in some sense, what we are doing is we are taking the expectation of that noisy update rule, right? And if we take the expectation, we will get back what we had over here.

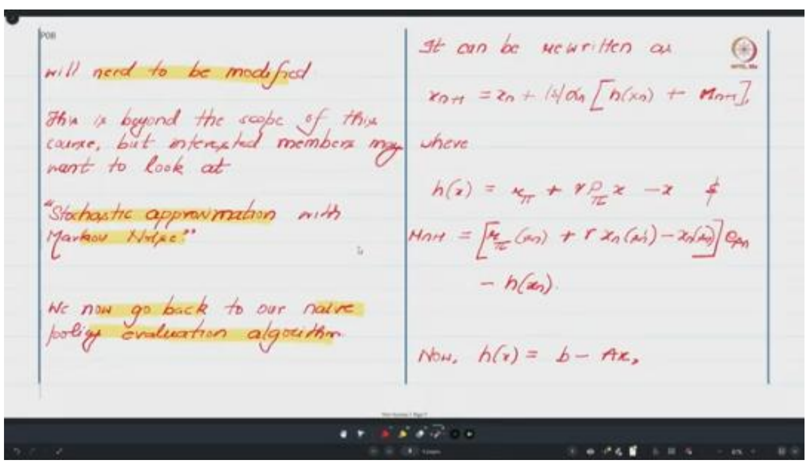
So, if you recall this expectation that we have over here, right, this expression is indeed r pi plus gamma p pi x n minus x n. So, this whole expression, along with the expectation, can be written in this form, and based on this. So, of course, based on this. So, we can rewrite the update rule in the following form, where h of x is given by this expression, and mn plus 1 is whatever we actually use to run the algorithm minus h of xn. So, we have come up with some algorithm.

So, let us just spend a few minutes appreciating this algorithm. So, notice that this algorithm can be implemented. Why can it be implemented? Because, you know, it does not require knowledge of V pi, right? So, notice that in this whole algorithm, you only have X n's, and at time step n, you have access to X n. And since you have access to X n, right, all the quantities over here, in particular X n of S n prime, X n of S n, okay, that would be known, right?



And we will presume that we also have access to the immediate reward. So, once you have access to the immediate reward and Xn, this whole expression actually becomes known. Also, notice that this stochastic algorithm does not have anything in its expression relating to pπ. So, as I told you, pπ depends on these transition probabilities, which are unknown, and in some sense, that is the reason why we prefer working with a stochastic version of that algorithm that we wrote on the previous slide. So, in this algorithm, neither do we need the knowledge of Pπ explicitly or P explicitly, nor the knowledge of Vπ. Instead, we can, you know, sample Sn and Sn′ according to the rule specified over here and then come up with this update rule.

So, if you can get access to samples like this, then indeed you can run this algorithm without the knowledge of P pi and S n. So, that is what is nice about this algorithm. So, now, I would like to take a few minutes and explain to you that while this algorithm can be implemented, it cannot be viewed as an SGD method. So, let us try and understand why that is the case. So, observe that here h of x is of the form b minus a x.



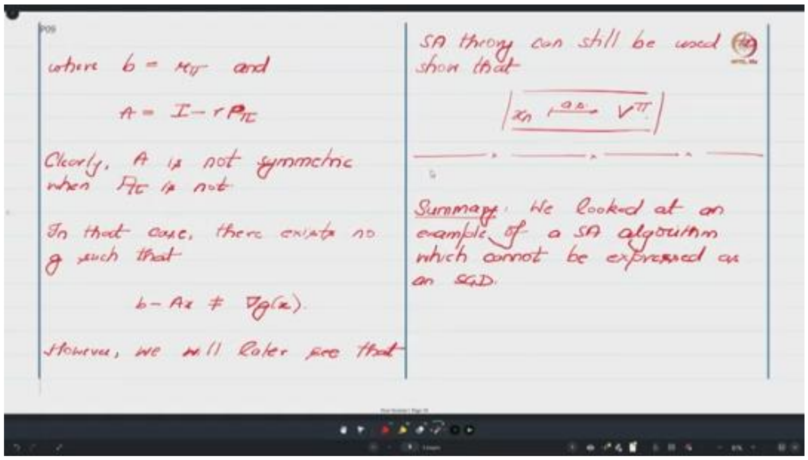
$$h(x) = b - Ax$$

So, h of x is given over here. So, this h of x has the form b minus a x, where b is r pi. And a is I minus gamma p pi, right?

$$b = r_\pi$$

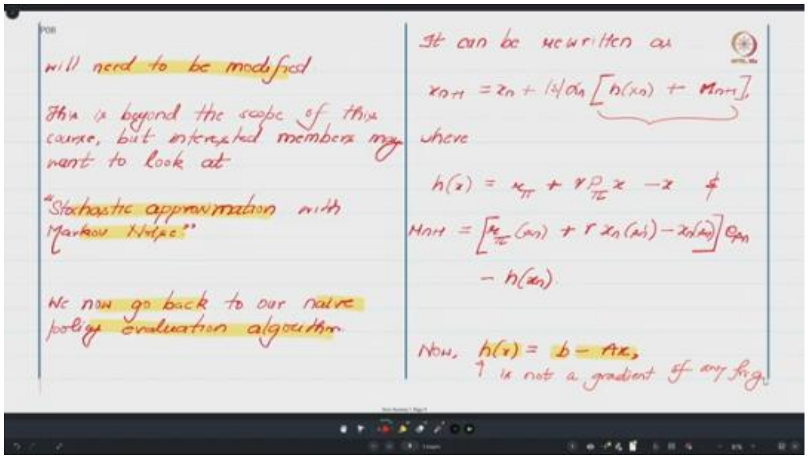
$$A = I - \gamma P_\pi$$

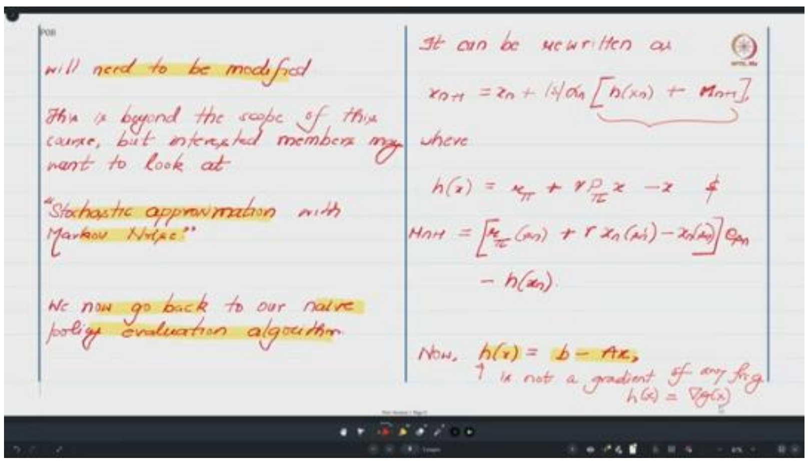
So, again, one may wonder, 'Oh, you said the ablate rule does not need the knowledge of p pi, but we can see this p pi over here.' Well, this is the form that we use to analytically analyze our algorithm.



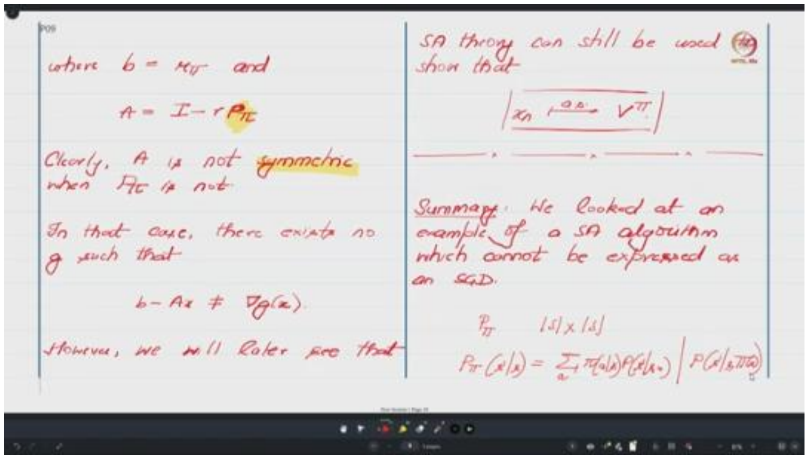
However, when we implement the algorithm, we would actually be making use of the form that is given over here. So, in this form, notice that we do not need the knowledge of P pi or V pi; we just need access to samples S n and S n prime to run this algorithm. However, when we want to analyze the algorithm, we will write it in a suitable form, which is given over here. So, in this case, h of x has this form: b minus ax, where b is r pi and a is i minus gamma p pi. So, now my claim is that this h of x that we have over here—this h of x—is not a gradient of any function g.

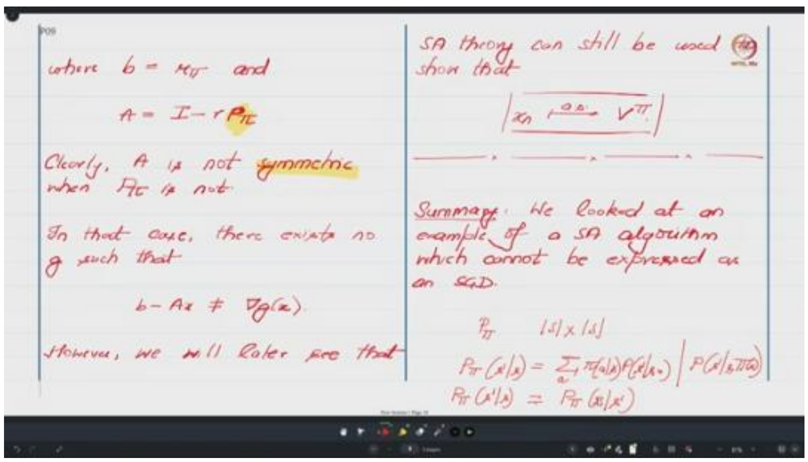
That is, you cannot find any function g such that h of x equals grad g of x. So, the claim is that you cannot find a function like g. So, why is that true? It is true because this P pi matrix, in general, will not be symmetric. Why will it not be symmetric? So, recall: what is P pi?





P pi is the matrix of size cardinality of S times cardinality of S, which means that it has the number of rows equal to the number of states and the number of columns also equal to the number of states. And P pi of S prime given S is basically the summation over A pi of A given S times P of S prime given S A, or if the policy is deterministic, this is basically P of S prime given S comma pi of S. Now, for this matrix to be symmetric, what we would require is that P pi of S prime given S, which is the S S prime entry of P pi, should equal P pi of S given S prime. That is, starting from S, the probability of moving to S prime and starting from S prime and moving to state S—these two probabilities should be the same for any combination S and S prime. Now, in general, you know Markov decision processes—such a relation need not be true. Moving from some state to another state may be easy or possible, and moving from the other state back to the previous state may or may not even be possible.





So, hence, in general, your P pi matrix will not be symmetric, and because P pi is not symmetric, your matrix A will also not be symmetric. And because your matrix A is not symmetric, one can easily see from calculus that you have a linear function where the quantity is not symmetric. Hence, you cannot find a quadratic function whose gradient will be B minus AX. However, we will see that the stochastic approximation theory can still be used to analyze such algorithms. In fact, one can use stochastic approximation theory to show that the stochastic approximation algorithm we designed will indeed converge to V pi.

That is, Xn will almost surely converge to V pi. Now, notice that because we use these samples at every state and every time step, this will be a stochastic algorithm. But nevertheless, under some very simple mild conditions, one can show that this Xn, which is now a random variable, will indeed converge to V pi. With this, I am at the end of this class. So, let me quickly summarize what we have discussed.

In lecture 5, we looked at an example of a stochastic algorithm that cannot be expressed as an SGD. So, as I said, the true potential of stochastic approximation lies in the fact that we can have algorithms beyond SGD, and these algorithms are extremely popular in reinforcement learning and related applications. With that, I would like to thank you. Hope you will join again in lecture 6. Thank you and Namaste.