STOCHASTIC APPROXIMATION: THEORY AND APPLICATIONS

Dr. Gugan Thope

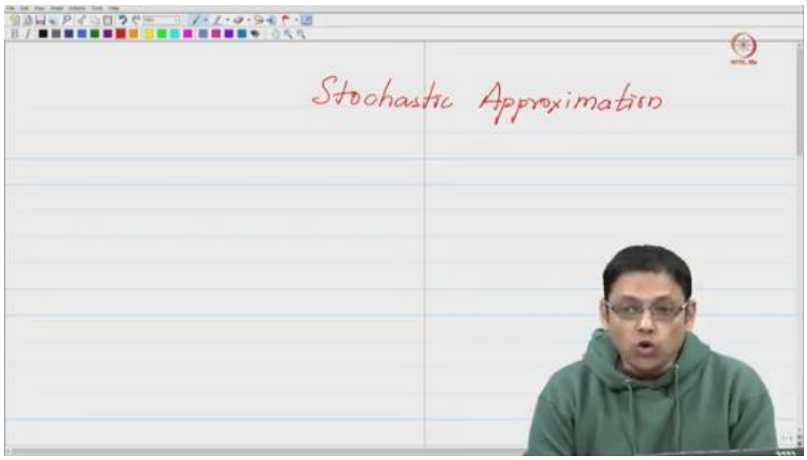
Department of Computer Science and Automation

Indian Institute of Science

Lecture 1

Overview of Stochastic Approximation

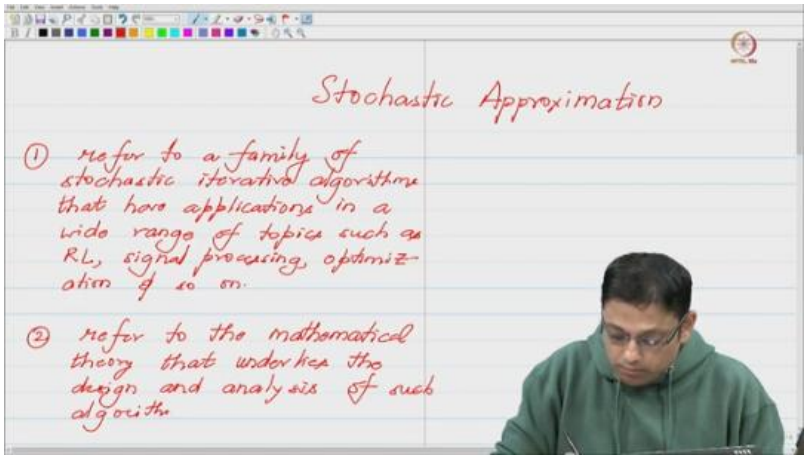
Namaste, hello, and welcome. My name is Gugan Thope, and I am an assistant professor in the Department of Computer Science and Automation at the Indian Institute of Science. One of my key research interests lies in understanding the theory and applications of reinforcement learning. For the theoretical underpinnings of reinforcement learning and many other modern learning algorithms, I rely on a beautiful and powerful mathematical framework known as stochastic approximation. In this NPTEL course, I look forward to sharing my understanding of this subject with you.



Together, we will explore both the elegance of the theory and its practical relevance across a wide range of applications. In this course, I will use stochastic approximation in two distinct but related ways. On the one hand, I will use stochastic approximation to refer to a family of stochastic iterative algorithms, that have applications in a wide range of topics such as reinforcement learning, signal processing, optimization, and so on.

Sometimes, I will use stochastic approximation to refer to a specific algorithm as well. On the other hand, I will use stochastic approximation to refer to the mathematical theory that

underlies the design and analysis of such algorithms. Without further ado, let us bring forward the fundamental form of a stochastic approximation algorithm. As I mentioned, a stochastic approximation algorithm is an iterative algorithm.



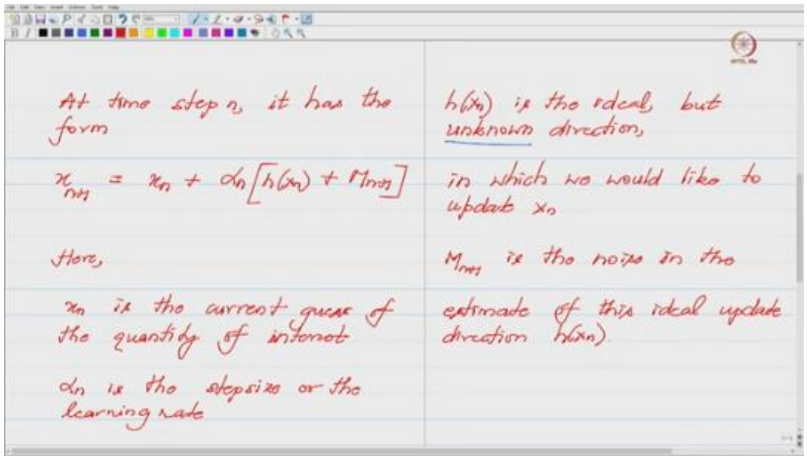
At time step n, it has the form. Xn plus 1 equals Xn plus alpha n H of Xn plus Mn plus 1.

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

Here, xn is the current guess of the quantity of interest. For example, you may be interested in figuring out the mean of a random variable. In that case, xn would be the current estimate of that mean.

Alpha n is the step size or the learning rate. H of Xn is the ideal but unknown direction in which we would like to update xn. So, observe that I said H of Xn is the ideal but unknown. So, this is very important. If H of Xn was known, then in some sense, we would not need stochastic approximation.

Stochastic approximation becomes useful because there are several applications where this ideal update direction is unknown. And finally, Mn plus 1 is the noise in the estimate of this ideal update direction H of xn. So, with this update rule in place, let us try to understand what stochastic approximation means. If we had access to the ideal update direction, then we could have as well used the algorithm Xn plus 1 equals Xn plus alpha n times H of Xn. In that update rule, we would not have had Mn plus 1.



However, we will soon see there are several scenarios where we do not have access to this ideal update direction. In such scenarios, we instead have access to a noisy approximation of this ideal update direction. We shall denote this noisy update direction often as H of Xn plus Mn plus 1. So, you can see that ideally, we wanted H of Xn, but we do not have access to it. Hence, we approximate it using a noisy version of it, which we denote by H of Xn plus Mn plus 1.

And this motivates the naming of the algorithm as stochastic approximation. The word stochastic comes because this Mn plus 1 will often be noisy or random. And it is an approximation because we ideally wanted H of Xn. We do not have access to it, and hence we replace it with H of Xn plus Mn plus 1. If H of X equals grad F of X for all X.

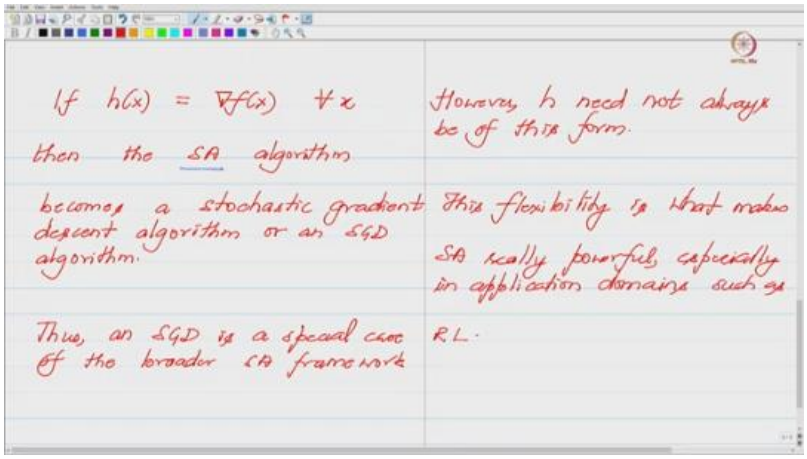
$$h(x) = -\nabla f(x) \forall x$$

In other words, suppose this ideal update direction equals the gradient of some function F for all X, then the stochastic approximation algorithm So, notice that I use the abbreviation SA to mean stochastic approximation, and henceforth whenever I write SA, I will imply stochastic approximation. So, let us go back to our discussion. If H of X equals grad F of X for all X, then the stochastic approximation algorithm becomes a stochastic gradient descent algorithm or an SGD algorithm in short.

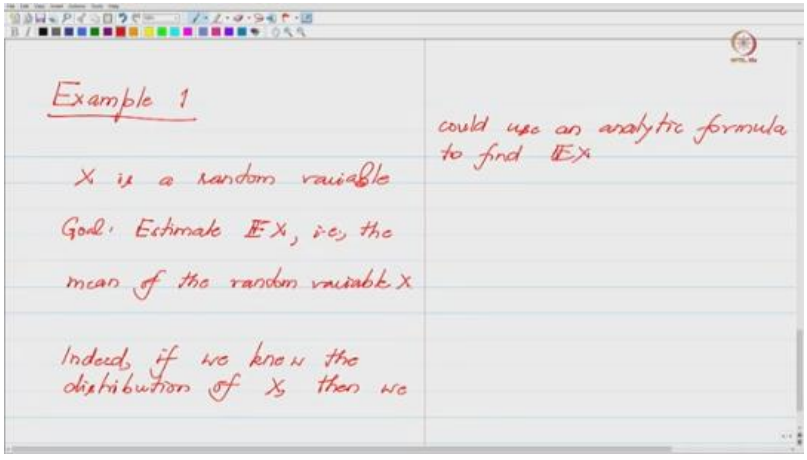
I am sure, with the hype and impact of machine learning, everyone must have heard about stochastic gradient descent methods in the context of machine learning. And what I am saying here is that when h of x equals grad f of x, then indeed a stochastic approximation

algorithm becomes an SGD algorithm. Thus, SGD is a special case of the broader stochastic approximation framework. However, H need not H need not always be of this form.

This flexibility is what makes stochastic approximation really powerful, especially in application domains such as reinforcement learning. I now discuss one easy example to understand stochastic approximation. In future classes, we will see many more examples. Example 1.



So, I hope this is an easy example for everyone. Let us say X is a random variable. And our goal is to estimate the expected value of X, that is, the mean of the random variable X. Indeed, if we knew the distribution of X, then we could have, we could use an analytic formula to find the expected value of X.



For instance, suppose X is a discrete random variable. It takes values in 1, 2, ..., m with probability mass function Px. That is, suppose P subscript x of i equals the probability of the event X equals i for all i in 1, 2, ..., m.

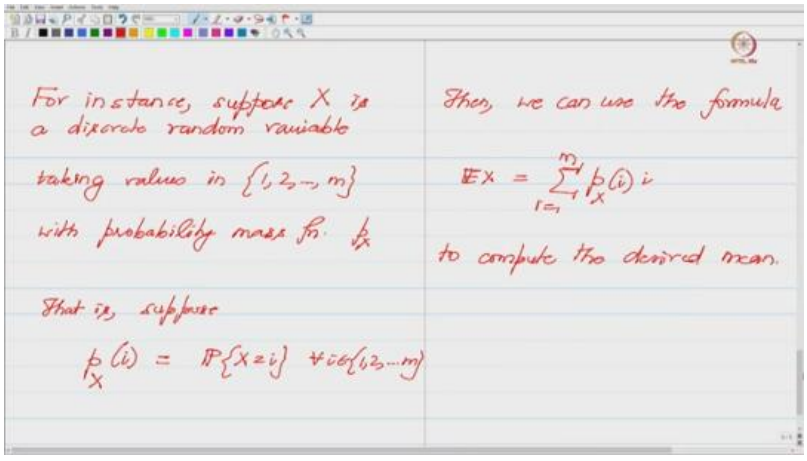
$$p_X(i) = \mathbb{P}\{X = i\} \forall\, i \in \{1,2 \ldots m\}$$

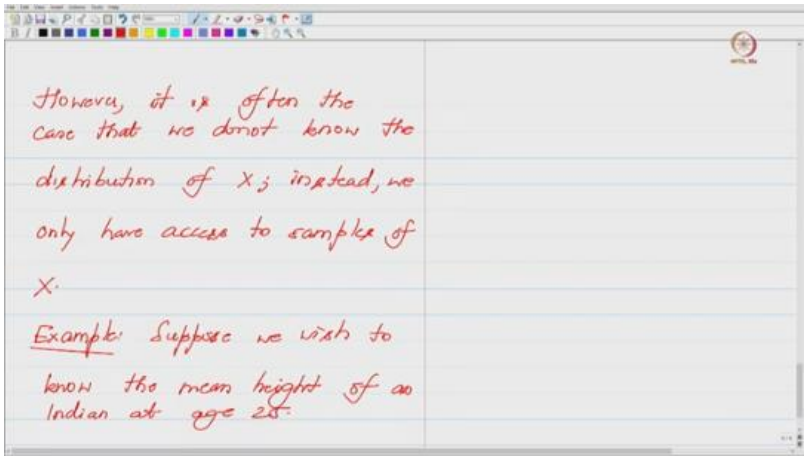
Then, we can use the formula: expected value of X equals summation from i equals 1 to m of P subscript X of i times i.

$$\mathbb{E}X = \sum_{i=1}^{m} p_X(i)i$$

This computes the desired mean.

However, it is often the case that we do not know the distribution of X. Instead, we only have access to samples of X. For example, suppose we wish to know the mean height of an Indian at age 25, okay. A general person would not have access to nationwide health records, and we may not know the distribution of the heights of Indian people at age 25.





So, one can clearly see that in this case, we do not know the distribution of X. However, we can measure the height of random 25-year-old individuals that we encounter. So, here is a scenario where we do not have access to the distribution. Rather, we can obtain samples of this quantity of interest and use those samples to determine the mean of the random variable. So, let us go back to that problem.

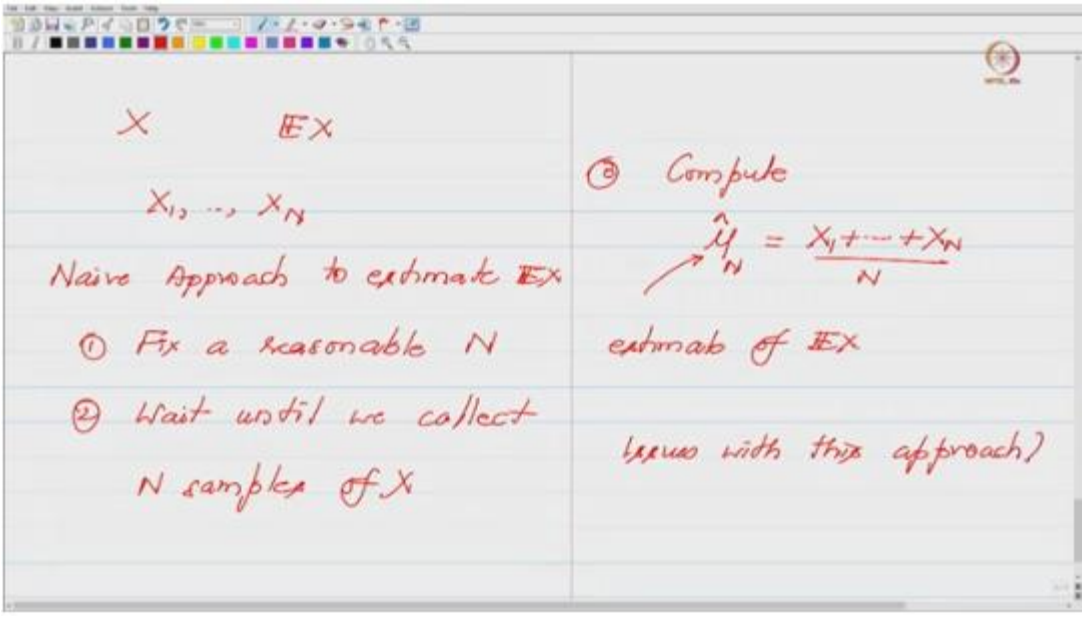
So, recall the problem was we have a random variable x, and we are just a minute—let me change the pen back. So, recall the example. So, recall our problem. We have our random variable x, and our goal is to find the expected value of x, and we are considering the scenario where we do not know or have access to the distribution of x. Instead, we are in a scenario where we have access to samples of x, and let us denote these samples by x1, x2, all the way up till xn.

Now, what is a naive way to determine the expected value of X? Well, the different steps in this naive approach could be: fix a reasonable N. Okay, so, this N could be, for example, a thousand, 1 million, 1 billion, and so on. The next step would be to wait until we collect N samples of X.

And finally, we use these samples to compute an estimate of the expected value of X using the formula: mu hat n equals (X1 to Xn) divided by n.

$$\hat{\mu}_N = \frac{X_1 + \cdots + X_N}{N}$$

So, this quantity over here we will use to denote the estimate of the expected value of X.



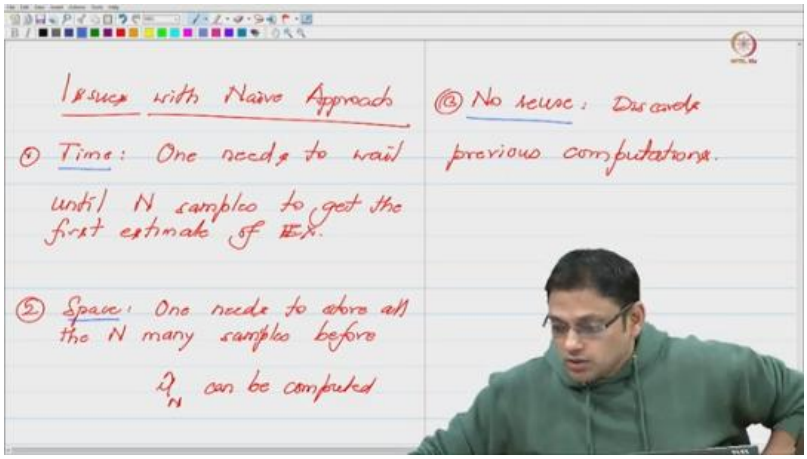
So, what do you think are the issues with this approach? 1. Time—one needs to wait until n samples to get the first estimate of the expected value of X. 2. Space—one needs to store all the n many samples before mu hat n can be computed.

And the third aspect is that this is a really inefficient approach to do the calculation in that we do not make use of any of the previous computations when new samples become available. For example, in the scenario that we considered, we had n many samples. Let us say n was 1,000. So, we had 1,000 many samples, and the naive approach suggested taking the average of those 1,000 values to get an estimate of the mean.

Now, let us consider a scenario where we, you know, get some additional 1,000 samples, right? So, the approach says, you know, take these 2,000 samples together and recompute the mean, right? So, in that sense, this naive approach discards previous computations. Right, so in this way, you can see that this naive approach doesn't, you know, effectively use the samples. It sort of needs—I mean, one needs to wait for n many samples to get the first estimate, one needs to store all those samples before one can compute this estimate of the expected value of X, and, you know, we discard all the previous computation. So, in

So, in this sense, one can see that this approach is really, really inefficient. So, the question is, can we do something efficiently? So, I am sure all of you can already guess what we are

trying to do. So, here is the stochastic approximation alternative. So, let mu hat n as before be x1 plus dot dot dot plus x little n over n.



Then, one can make this very easy observation that $\hat{\mu}_{n+1}$, which is equal to (x_1 all the way up to x_{n+1}) divided by (n + 1), satisfies the relation. n times $\hat{\mu}_n$ plus x_{n+1} over (n + 1), which, with some further algebra, can be shown to equal $\hat{\mu}_n$ plus 1 over (n + 1) times the square bracket [X_{n+1} minus $\hat{\mu}_n$]. So, one can see that the naive approach made use of calculations like this, right? It used $\hat{\mu}_n$ when it had access to n many samples and used $\hat{\mu}_{n+1}$ when it had access to n + 1 many samples. And in this—sorry, I have written 'key contribution.'

Instead, I should say 'key observation.' Sorry about that. So, the key observation is that $\hat{\mu}_{n+1}$ and $\hat{\mu}_n$ satisfy the following relation. So, based on this, the proposed stochastic approximation algorithm to find the expected value of X is the following. Where α_n equals 1 over (n + 1).

$$\hat{\mu}_n := \frac{X_1 + \cdots + X_n}{n}$$

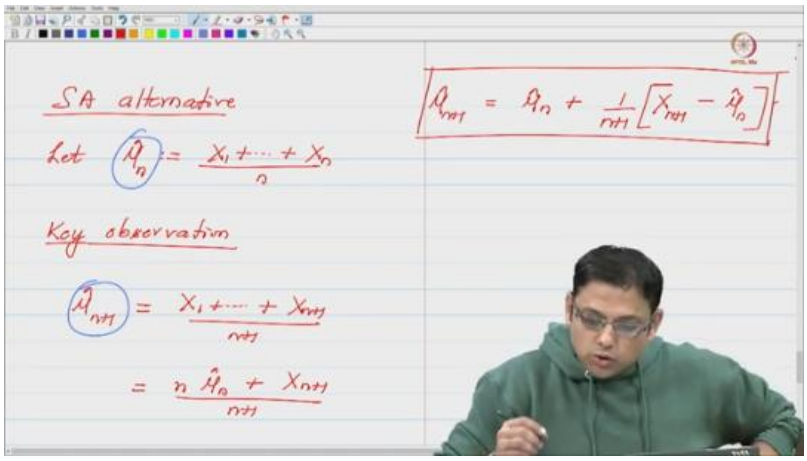
$$\hat{\mu}_{n+1} = \frac{X_1 + \cdots + X_{n+1}}{n+1}$$

$$= \frac{n\,\hat{\mu}_n + X_{n+1}}{n+1}$$

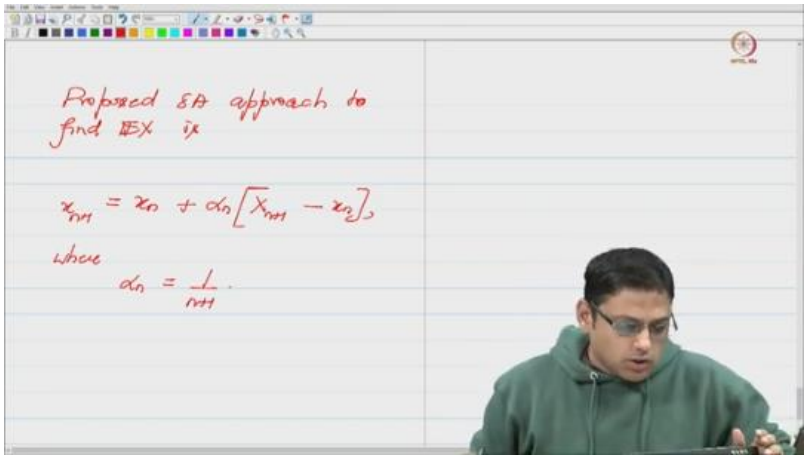
$$\hat{\mu}_{n+1} = \hat{\mu}_n + \frac{1}{n+1}[X_{n+1} - \hat{\mu}_n]$$

$$x_{n+1} = x_n + \alpha_n[X_{n+1} - x_n]$$

$$\alpha_n = \frac{1}{n+1}$$

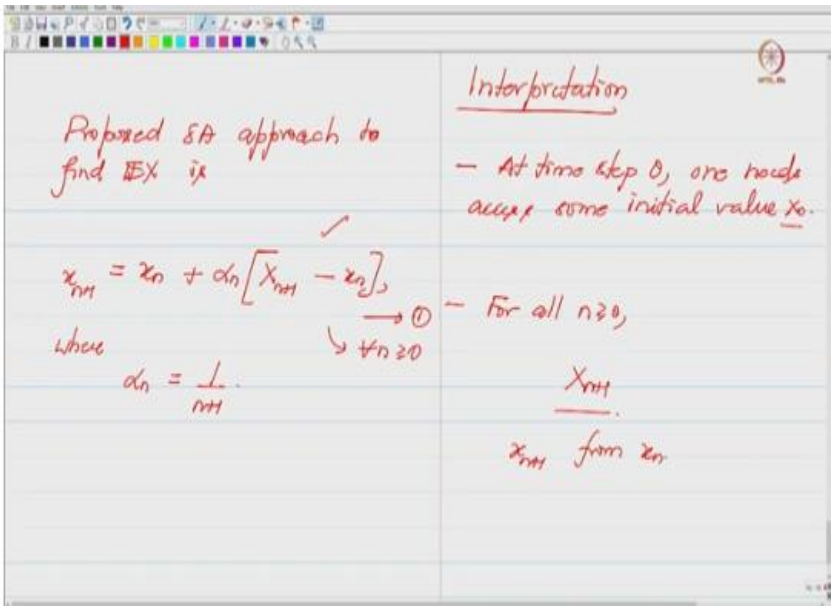


So, let us first interpret what this algorithm is trying to do. I have denoted this algorithm by Equation 1. So, let us try to interpret what this update rule is trying to do. So, first notice that at time step 0, one needs access to some initial value x_0.



So, the theory of stochastic approximation often does not tell you how to obtain this initial value. This initial value is often obtained through some trial-and-error simulations or domain knowledge and so on. So, in this course, we will not venture too much into how to obtain this initial value, and we will often presume that somebody has given us this initial value. And given this initial value, how do we proceed in improving upon this estimate? And if nobody has given this initial value x0, we can often set it to be the origin.

And I will discuss scenarios if necessary if we cannot do this initialization. So, let us go back and try to interpret this update rule. So, this update rule, first of all, note that it has to be executed for all n greater than or equal to 0. And at time step 0, we need access to this value X0. And thereafter, for all n greater than 0, one needs access to this sample Xn plus 1 of X, and using this sample, we use this update rule to compute Xn plus 1 from Xn.

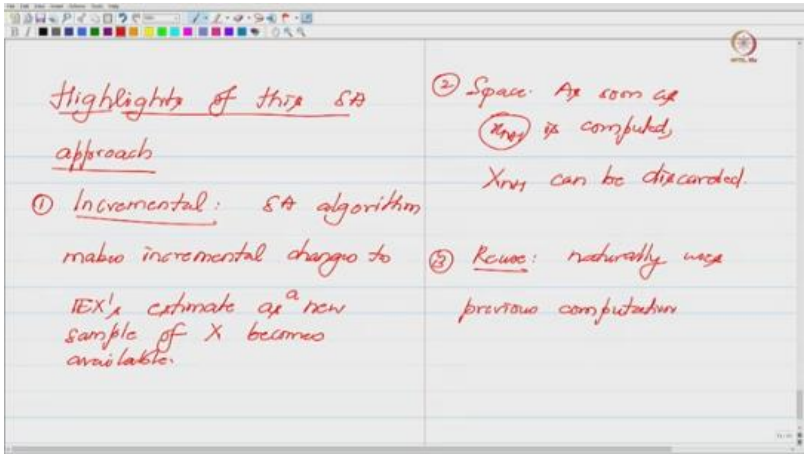


So, what are the highlights of this stochastic approximation approach? One, this approach is incremental in nature. That is, the stochastic approximation algorithm makes incremental changes to the expected value of X's estimate. as a new sample of X becomes available, right? So, notice that, as soon as you have one sample, we have an estimate of the expected value of X.

When you get the second sample, you know, you improve upon this estimate and get little x2, which is the second estimate of expected value of X and so on and so forth. So, one can see that, in an incremental way, we can make better and better estimates of the expected value of X, right, as more and more samples become available. And with regard to space, one can see that as soon as Xn plus 1 is computed, capital Xn plus 1 can be discarded. Because when you want to compute capital Xn plus 1, Sorry, when you want to compute little xn plus 1, you need little xn and capital Xn plus 1.

That's all, right? So, once you have computed little xn plus 1, that is this quantity, right? In order to compute little xn plus 2, you don't need capital Xn plus 1. You only need little xn plus 1 and capital Xn plus 2. You can verify this.

And you can see that our stochastic approximation algorithm naturally reuses previous computations. In that, if xn denotes the little xn, if it denotes the estimate of the expected value of X at time step n, then little xn plus 1 denotes the estimate of the expected value of X at time step n plus 1. So, you can see that little xn plus 1 is a function of little xn. So, in that sense, So, each new estimate of the expected value of X is based on our previous estimate of expected value of X.



So, now the last question is: why is this a stochastic approximation algorithm? Well, we can rewrite this update rule as Xn plus 1 equals Xn plus alpha n H of Xn plus Mn plus 1. where H of X equals mu minus X and Mn plus 1 equals capital Xn plus 1 minus mu, where mu equals the expected value of X.

$$x_{n+1} = x_n + \alpha_n[h(x_n) + M_{n+1}]$$

$$h(x) = \mu - x$$

$$M_{n+1} = X_{n+1} - \mu$$

$$\mu = \mathbb{E}[x]$$

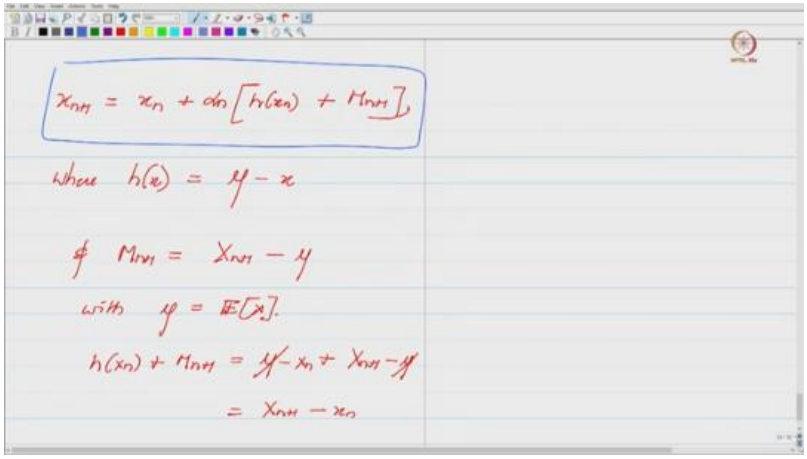
So, you can see that the update rule we had before can be written in this generic form. And this generic form, if you recall, matches the fundamental form of the stochastic approximation that I mentioned before.

We only need to interpret what H of X and Mn plus 1 mean for this specific algorithm we designed. And it is very easy to see that H of Xn plus Mn plus 1 equals mu minus Xn plus Xn plus 1 minus mu. These mu and this mu cancel out, and we are left with Xn plus 1 minus Xn, which is exactly what we had before.

$$h(X_n) + M_{n+1} = \mu - X_n + X_{n+1} - \mu$$

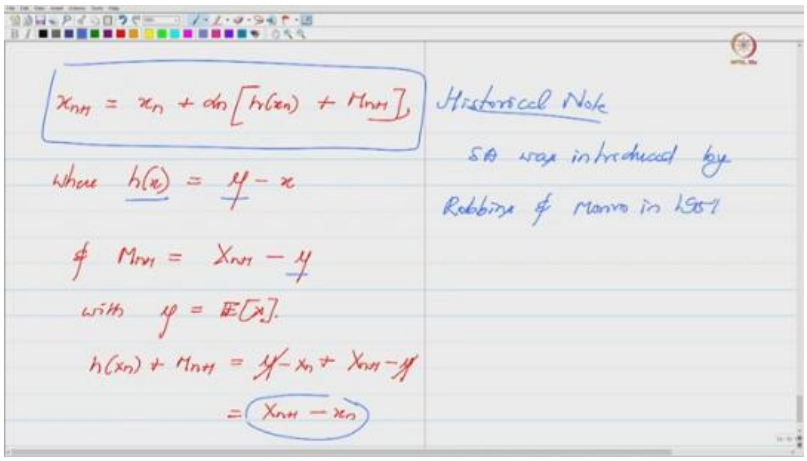
$$= X_{n+1} - X_n$$

You can see that this is exactly what we had in the square bracket over here. So, in summary, one can see that the update rule we designed can be put into the fundamental form we saw before.



And one can interpret this H of X as the ideal direction in which one would like to move Xn. And H of Xn plus Mn plus 1, which is this quantity, as the noisy estimate of this ideal update direction. So, one may ask: why have I subtracted mu over here or, sorry, added mu over here and subtracted mu over here? More generally, why have I defined H of X and Mn plus 1 in this way? Well, in the classes that will come next, I will explain the purpose of having this H and why Mn plus 1 is defined this way. At this point, all I will say is that if you take the expected value of Mn plus 1, you will find that its mean is 0.

So, with this, I would like to end my class. I would just like to add one historical note. Stochastic approximation was introduced by Robbins and Monroe in 1951. So, you can see that there is a theory that was built almost 75 years ago, and we are still using it—not just for the sake of studying, but also to analyze some of the state-of-the-art algorithms used in areas such as reinforcement learning, machine learning, and so on. So, in this class, we saw the definition of stochastic approximation.



We saw a very simple example of stochastic approximation. In the next class, we will see more sophisticated examples of stochastic approximation. And in the classes that follow, we will see how to design such stochastic approximation algorithms and analyze them. By analyze, I mean: where would these algorithms converge? What would be their convergence rates, which factors affect their convergence rates, and so on and so forth.

With this, let me stop my first lecture. Thank you.