STOCHASTIC APPROXIMATION: THEORY AND APPLICATIONS

Dr. Gugan Thope

Department of Computer Science and Engineering

Indian Institute of Science, Bangalore

Week 9

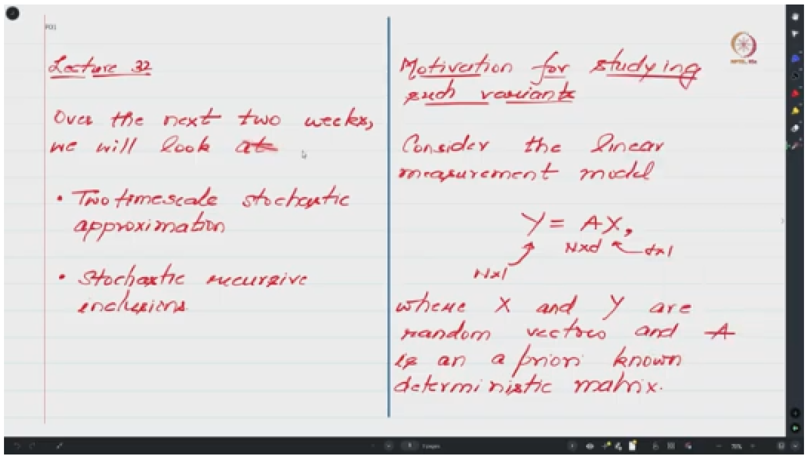
Lecture 32

Distributed Estimation of the Mean of a Random Vector

Hello and Namaste, everyone. Welcome to the NPTEL course on Stochastic Approximation. We are in week 9 of this course. During this week and the week that follows, we will look at two different variants of Stochastic Approximation. In particular, we will look at Stochastic Approximation with two time scales, and the other thing we will focus on is Stochastic Recursive Inclusions.

So, in some sense, these variants extend the applicability of stochastic approximation. To motivate the use of such variants and where they can help, what I will do in today's class is take a motivating application, and design a stochastic approximation algorithm that does not have any of these properties—that is, it is not of two-time-scale nature and does not have this stochastic recursive inclusion nature. However, at the end of today's class, I will pose a problem, and to solve that problem, we will need to invoke some of these ideas—that is, this two-time-scale nature and stochastic recursive inclusions. We will see how those concepts naturally arise and then look at the analysis of convergence of such algorithms and their convergence rates. With that, let us begin our formal discussion.

So, as I said, over the next two weeks, we will look at two-time-scale stochastic approximation and stochastic recursive inclusions. But in today's class, we will look at one application for which we will design what is known as a single-time-scale stochastic approximation. So, the problem we will look at over the next two weeks is the following. Suppose we have a linear measurement model y equals A times x, where x is a d-dimensional vector. A is an n-by-d matrix, and y, because of these dimensions, is n-by-1. In this linear measurement model, we will presume that x and y are random vectors, while A is an a priori known deterministic matrix.


Lecture 32

Over the next two weeks, we will look ~~at~~

- Two timescale stochastic approximation
- Stochastic recursive inclusions

Motivation for studying such variants

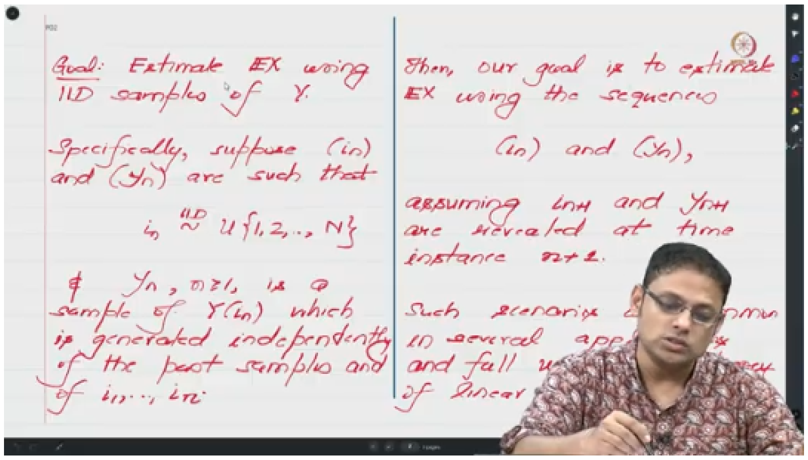
Consider the linear measurement model

$$\underset{N\times 1}{Y} = \underset{N\times d}{A}\underset{d\times 1}{X},$$

where X and Y are random vectors and A is an a priori known deterministic matrix.

$$Y = AX$$

So, A is some matrix which we know and is not random right and x and y are random vectors of suitable dimensions. And the problem that we will be looking at over the next two weeks is that of estimating the expected value of X using samples of this random vector Y. Specifically, we will presume that instead of having access to you know all coordinates of y at every time instance we will presume that at any given time instance one of the coordinate indices is randomly picked and the corresponding you know the sample of y corresponding to that coordinate is provided more formally we will presume that we have two sequences i n and y n such that IN is independent and identically distributed uniform random variable on the set of indices 1 to N. Recall that your matrix A has N rows, capital N many rows. So, this IN over here picks one row uniformly randomly at any given time instance.


Goal: Estimate $\mathbb{E}X$ using IID samples of Y.

Specifically, suppose (i_n) and (y_n) are such that

$$i_n \overset{IID}{\sim} \mathcal{U}\{1, 2, \ldots, N\}$$

& y_n, $n \geq 1$, is a sample of $Y(i_n)$ which is generated independently of the past samples and of $i_1, \ldots, i_n$.

Then, our goal is to estimate $\mathbb{E}X$ using the sequences

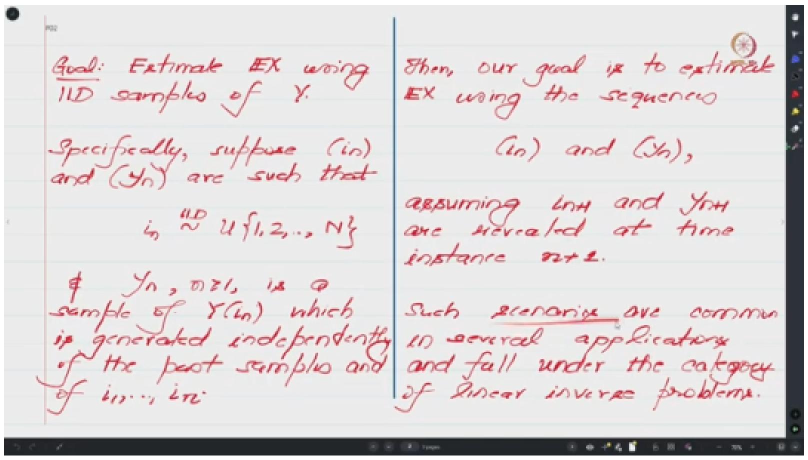
$$(i_n) \text{ and } (y_n),$$

assuming i_{n+1} and y_{n+1} are revealed at time instance $n+1$.

Such scenarios [illegible] in several app[illegible] and fall u[illegible] of linear [illegible]

In particular, probability IN equals let us say 2 is 1 over capital N. Separately, Yn, this calligraphic Yn for any n bigger than equal to 1 is a sample of Y of In. Recall that capital Y is a vector of size capital N. And we will presume that at any time instance n, once In has been sampled, calligraphic Yn is a sample of the Inth coordinate of Y. And we will presume that this Yn is generated independently of past samples and also of how these indices are chosen. Using this data, our goal then is to estimate the expected value of X using In and Yn.

And we want to estimate this expected value of X in an online fashion. what that means is that your index i n plus 1 and this sample calligraphic y n plus 1 is are presumed to be revealed exactly at time instance n plus 1 right. And I would like to highlight that such scenarios are quite common in several applications and fall under the category of what is known as linear inverse problems. Now, one such motivating application is that of network tomography. So, let me try to illustrate within the context of network tomography, how do you end up with this linear measurement model.



Goal: Estimate $\mathbb{E}X$ using IID samples of Y.

Specifically, suppose (i_n) and (y_n) are such that

$$i_n \overset{IID}{\sim} \mathcal{U}\{1, 2, \dots, N\}$$

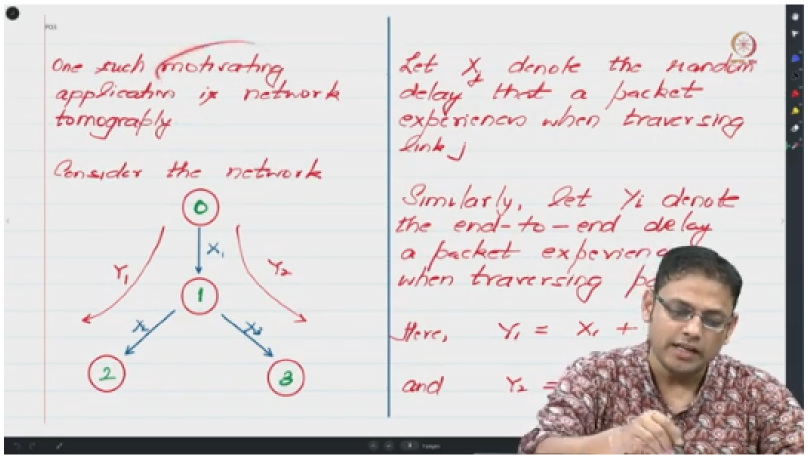
& y_n, $n \geq 1$, is a sample of $Y(i_n)$ which is generated independently of the past samples and of $i_1, \dots, i_n$.

Then, our goal is to estimate $\mathbb{E}X$ using the sequences

(i_n) and (y_n),

assuming i_{n+1} and y_{n+1} are revealed at time instance $n+1$.

Such scenarios are common in several applications and fall under the category of linear inverse problems.



One such motivating application is network tomography

Consider the network

Let X_j denote the random delay that a packet experiences when traversing link j

Similarly, let Y_i denote the end-to-end delay a packet experien when traversing p

Here, $Y_1 = X_1 +$

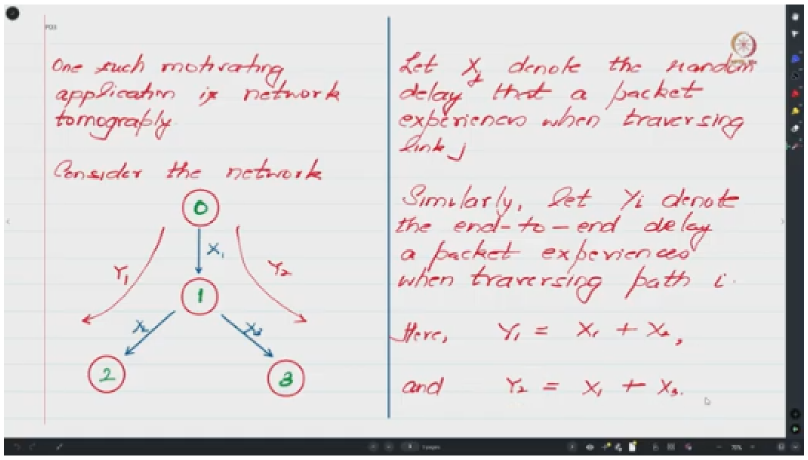
and $Y_2 =$

So, in network tomography, we typically have a network. This network could represent either a road network, a computer network, or any network of that kind. And let us say these are your different nodes. For example, in the context of a computer network, this could represent your routers, your local machines, and so on and so forth. And this blue arrow here denotes a link.

It could either be a wired link or a wireless link. And X1 denotes the delay a packet experiences while traversing link J. So, X1 denotes the delay a packet experiences when traversing link 1. Similarly, X2 denotes the delay a packet experiences when traversing link 2, and so on. And we will presume that these delays are random, and XJ denotes that random variable. Right.

And we will also presume we have a path. A path is basically a sequence of links that takes you from one node to another node. For example, we have a path consisting of this link and this link. That enables us to send a packet from node 0 all the way to node 2. And we will presume that Y1 is the end-to-end delay that the packet experiences on path 1.

Similarly, we will presume that Y2 is the end-to-end delay that the packet experiences while traversing path 2. So, here keep in mind that we have 3 links and 2 paths. And of course, you know, since X1, X2, X3 are random variables, Y1 and Y2 will also be random variables. And, you know, if we presume that the end-to-end delays are additive, then one can see that Y1 satisfies the relation Y1 equals X1 plus X2 and Y2 satisfies the relation X1 plus X3. And such a system of equations can be put in compact form in the following way.


One such motivating application is network tomography

Consider the network

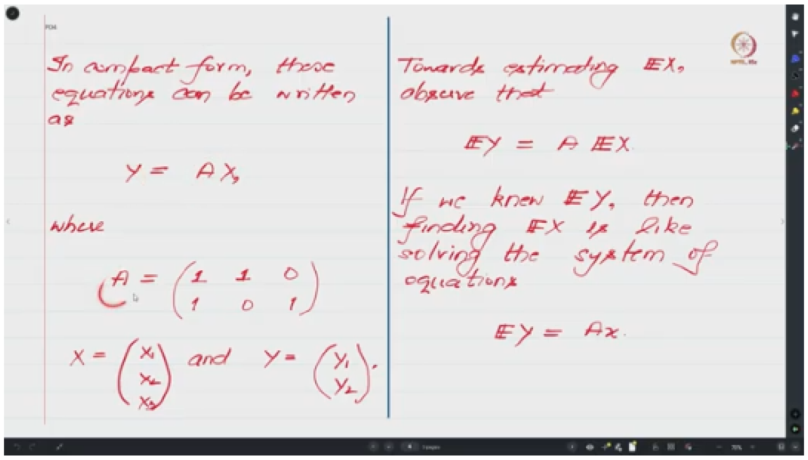
Let X_j denote the random delay that a packet experiences when traversing link j

Similarly, let Y_i denote the end-to-end delay a packet experiences when traversing path i.

Here, $Y_1 = X_1 + X_2$,

and $Y_2 = X_1 + X_3$.

We can say Y equals A times X where A is the matrix 1 1 0 and its second row is 1 0 1. So, if you let x equal x1, x2, x3 and y equal you know the vector y1, y2 then one can see that you know this system of equations that we have indeed can be compactly written in the following way. So, I would like to just highlight one thing. So, often I will be using y of i and yi if the context is clear to denote the ith vector. coordinate of y.


In compact form, these equations can be written as

$$Y = AX,$$

where

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

$$X = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} \text{ and } Y = \begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix}.$$

Towards estimating $\mathbb{E}X$, observe that

$$\mathbb{E}Y = A\,\mathbb{E}X$$

If we knew $\mathbb{E}Y$, then finding $\mathbb{E}X$ is like solving the system of equations

$$\mathbb{E}Y = Ax.$$

$$Y = AX$$

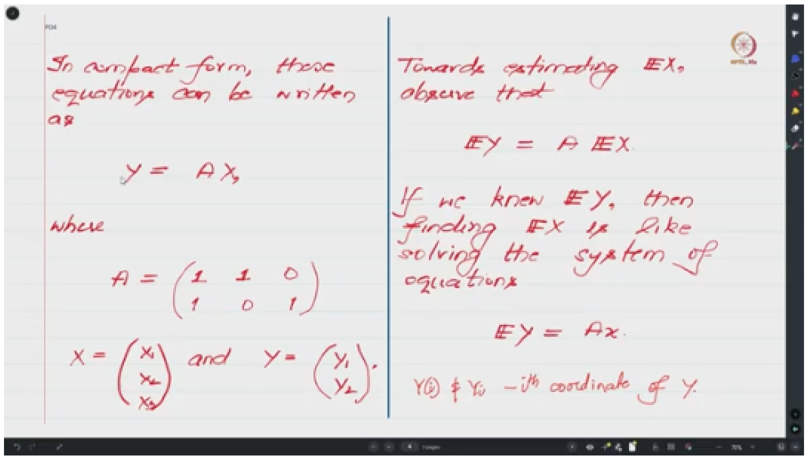
$$A = (1\ 1\ 0\ 1\ 0\ 1)$$

$$X = \left(X_1 X_2 X_3\right), \quad Y = \left(Y_1 Y_2 Y_3\right)$$

$$EY = A\,EX$$

$$EY = AX$$

So, you can see in this way that one can come up with a linear system of equations. So, this was just an example. Now, let us presume that we have some generic system in this fashion and you know recall that our goal was to estimate expected value of X. So, if you have such a system of equations and you take expectation using the linearity of expectation property, one can conclude that expected value of Y equals A times expected value of X. Now, if we knew expected value of Y, then the goal of finding expected value of X is equivalent to solving a system of equations of this form. That is a little x equals expected value of Y. So, everything here is deterministic.



In compact form, these equations can be written as

$$Y = AX,$$

where

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

$$X = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} \text{ and } Y = \begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix}.$$

Towards estimating $\mathbb{E}X$, observe that

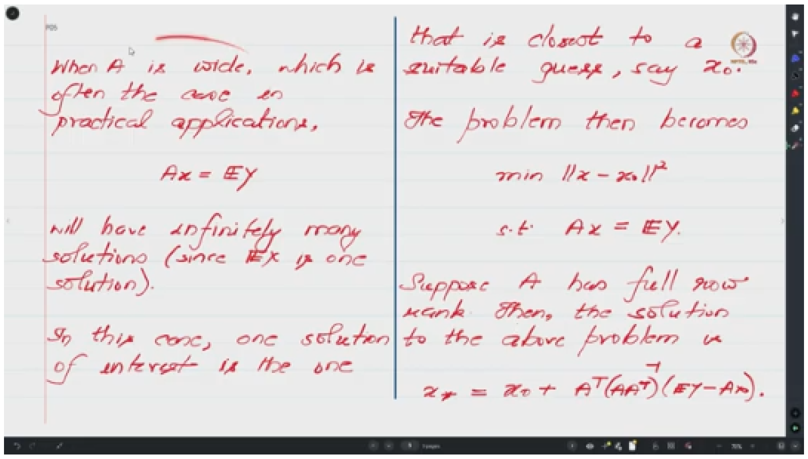
$$\mathbb{E}Y = A\,\mathbb{E}X.$$

If we knew $\mathbb{E}Y$, then finding $\mathbb{E}X$ is like solving the system of equations

$$\mathbb{E}Y = Ax.$$

$Y(i)$ & Y_i – i^{th} coordinate of Y.

The expected value of Y is a deterministic vector. A is a known matrix. X is unknown, and we want to solve this system of equations. When A is wide, right—wide meaning it has more columns than rows, and this is usually the case in many practical applications, right.


When A is wide, which is often the case in practical applications,

$$Ax = \mathbb{E}Y$$

will have infinitely many solutions (since $\mathbb{E}X$ is one solution).

So this case, one solution of interest is the one that is closest to a suitable guess, say x_0.

The problem then becomes

$$\min \|x - x_0\|^2$$
$$\text{s.t. } Ax = \mathbb{E}Y.$$

Suppose A has full row rank. Then, the solution to the above problem is

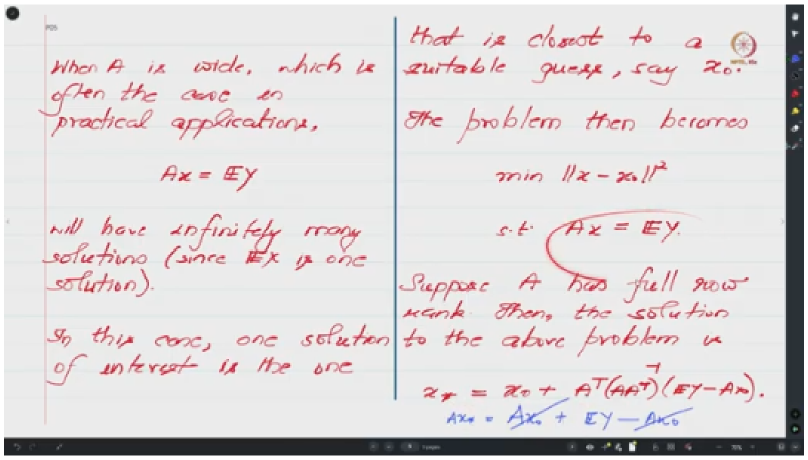
$$x_* = x_0 + A^T(AA^T)^{-1}(\mathbb{E}Y - Ax_0).$$

Then this system of equations, that is, Ax equals the expected value of y, will have infinitely many solutions since the expected value of x is already a solution, right. So, in an undetermined system of equations, whenever we have one solution, it automatically implies that there are infinitely many solutions. So, since there are infinitely many solutions, one would like to narrow down our focus on one special solution, and such a solution of interest is the one that is closest to some suitable guess, say X0. So, from our domain knowledge, we may have some intuitive idea that maybe the delays are somewhere close to this. And let us say X naught denotes that vector.

Then the problem of identifying the expected value of X becomes equivalent to solving this optimization problem. That is, minimize the distance between X and X naught. So, X naught is our initial guess. So, this is fixed. And I am looking at the square of the Euclidean distance between them.

And the constraint is that whatever X I find, it should satisfy this system of equations that is AX equals expected value of Y. So, this optimization problem says that give me an X which is close to my initial guess and at the same time satisfies this system of equations. Now, one can perhaps by using this idea of Lagrange multipliers show that if this matrix A which is wide has full row rank. Then the solution of the above problem is given by an expression which looks like this. That is, it is X0 plus A transpose, AA transpose inverse expected value of Y minus AX0. I mean, one can do a quick sanity check over here.

So, if I multiply A on both sides, one can see that the left-hand side will be AX star and the right-hand side will be AX0. plus AA transpose times AA transpose inverse. So, those things will cancel off and what I will be left with is expected value of Y minus A times X0. So, this will cancel off and one can see that indeed AX star equals expected value of Y in that it satisfies this system of equations. So, some of you perhaps may have this question of why AA transpose should be invertible.



When A is wide, which is often the case in practical applications,

$$Ax = \mathbb{E}Y$$

will have infinitely many solutions (since $\mathbb{E}X$ is one solution).

In this case, one solution of interest is the one that is closest to a suitable guess, say x_0.

The problem then becomes

$$\min \|x - x_0\|^2$$

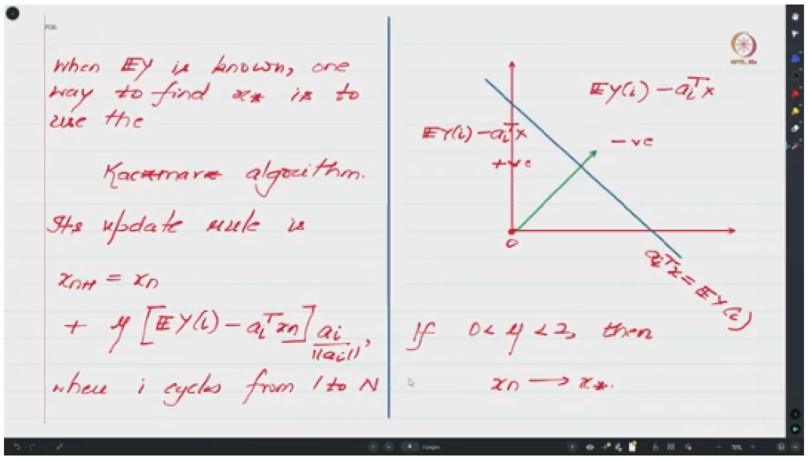
$$\text{s.t. } Ax = \mathbb{E}Y.$$

Suppose A has full row rank. Then, the solution to the above problem is

$$x_* = x_0 + A^T(AA^T)^{-1}(\mathbb{E}Y - Ax_0).$$

$$Ax_* = \cancel{Ax_0} + \mathbb{E}Y - \cancel{Ax_0}$$

Well, it should be invertible because of this full row rank condition. So, one can use some basic ideas from linear algebra to show that you know whenever your wide matrix A has full row rank then the matrix AA transpose indeed has full row rank. So, the summary is that you know the solution to this problem where everything is deterministic is X star which is given by this expression over here. Now, the question that we would like to next discuss is how to find this X star. In particular when expected value of Y is known one of the popular algorithms to find



When $\mathbb{E}Y$ is known, one way to find x_* is to use the

Kaczmarz algorithm.

Its update rule is

$$x_{n+1} = x_n + \eta\left[\mathbb{E}Y(i) - a_i^T x_n\right]\frac{a_i}{\|a_i\|},$$

where i cycles from 1 to N

If $0 < \eta < 2$, then

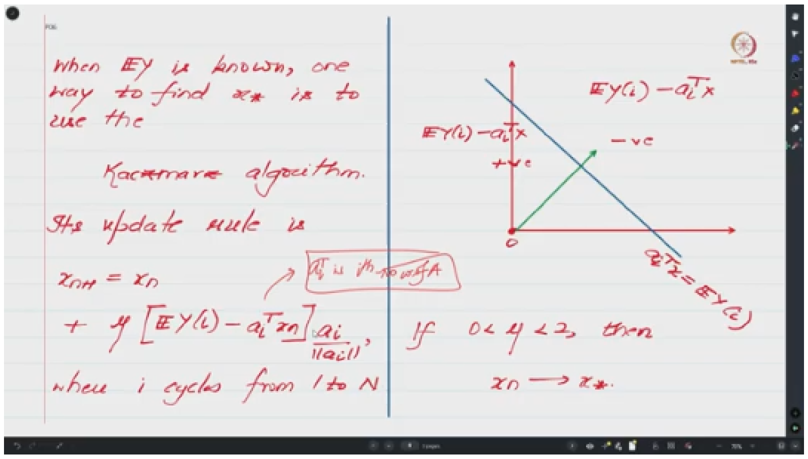
$$x_n \longrightarrow x_*.$$

$\mathbb{E}Y(i) - a_i^T x$
$-ve$
$\mathbb{E}Y(i) - a_i^T x$
$+ve$
0
$a_i^T x = \mathbb{E}Y(i)$

Such an X star is what is known as the Kaksch-Marx algorithm. So, its update rule is an iterative algorithm, and keep in mind that we are in the deterministic setting, presuming that the expected value of Y is known. So, the Kaksch-Marx algorithm has the update rule that is given over here. So, let us try to digest this. Right, and let me sort of tell you what we are going to do next so that you know why we are discussing this. Recall that we want to discuss two-time-scale algorithms and stochastic recursive inclusions, and before that, I wanted to give a gist of

a simple problem where we would not require any of these ideas. Thereafter, I will perhaps modify the—I mean, I should not say perhaps—I am going to modify the problem where their uses become natural. Towards going there, I am going to first come up with a single-time-scale algorithm—by that, I mean the kind of algorithms that we have seen so far—to find x star. To come up with that algorithm, I am going to first look at some existing algorithm to solve a simplified version of the problem. The simplified version of the problem is basically the deterministic variant of this problem, where the expected value of y is known, A is known, and the goal is to find x star. So, the Cox-Marx algorithm's update rule proceeds as follows. Suppose at time instance n, we have access to xn.

Then the estimate of xn plus 1 is obtained in the following way: xn plus 1 equals xn plus some step size mu—notice that the step size over here is a constant—times the expected value of y of i, which is the ith coordinate of the expected value of y, minus ai transpose xn times a vector ai over the norm of ai. Here, your ai—or ai transpose—is the ith row of A. Ai transpose is the ith row of A. So, this vector will be a vector of size. So, Ai transpose is a row vector. So, it has dimension equal to the cardinality of X, and we have presumed that the cardinality of X is d.



When $\mathbb{E}Y$ is known, one way to find x_* is to use the

Kaczmarz algorithm.

Its update rule is

$$x_{n+1} = x_n + \mu \left[\mathbb{E}Y(i) - a_i^T x_n\right] \frac{a_i}{\|a_i\|},$$

$\boxed{a_i^T \text{ is } i\text{th row of } A}$

where i cycles from 1 to N

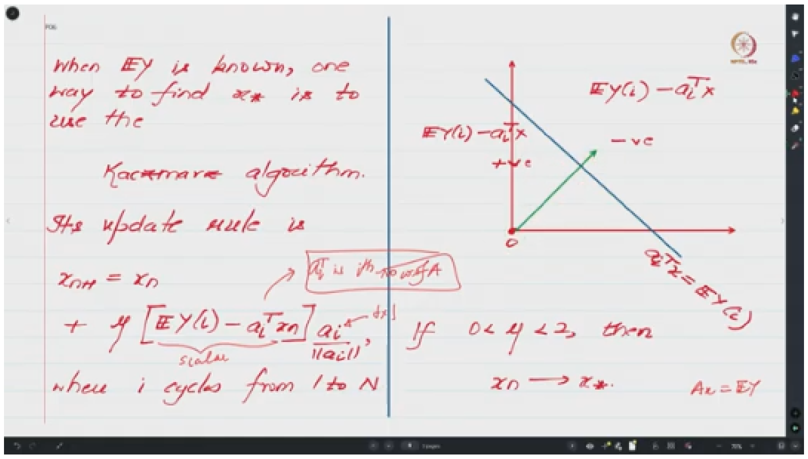
If $0 < \mu < 2$, then $x_n \longrightarrow x_*$.

$\mathbb{E}Y(i) - a_i^T x$
$-ve$
$\mathbb{E}Y(i) - a_i^T x$
$+ve$
0
$a_i^T x = \mathbb{E}Y(i)$

And hence, your a_i transpose will be a row vector of dimension d, and ai over here is basically that row vector written in column form. Hence, this will be a vector of size d cross 1. And this expression over here is a scalar. So, the update rule for the Cox-Marx algorithm is given over here. So, Xn is a d-dimensional quantity, and to that, we are adding a d-dimensional expression.

Now, the question is, how is I chosen? Well, in the classic Kaczmarz algorithm, you choose I cyclically from 1 to n. That is, you start from 1, then you go to 2, then you go all the way up till n, and once you reach n, you set I equals to 1 again, and then you cycle all the way to n, and you keep doing that. Right, and in this way, you update your XN. So, since you are going to stick with this update rule for the next two weeks, let me give a quick illustration of what this update rule is trying to do. So, towards that, what I have done is I have, you know, drawn one of the hyperplanes corresponding to one of the equations, right.

So, the ith equation in the system of equations Ax equals the expected value of y is basically Ai transpose x equals the ith coordinate of the expected value of y, right. So, this will be denoted by this hyperplane over here. So, all points on this hyperplane satisfy this equation over here. Now, this vector—that is, I do not know if you can see the color, but this is green in color. So, this vector over here is basically your Ai vector.



When $\mathbb{E}Y$ is known, one way to find x_* is to use the

Kaczmarz algorithm.

Its update rule is

$x_{n+1} = x_n$

$+ \mu \underbrace{\left[\mathbb{E}Y(i) - a_i^T x_n\right]}_{\text{scalar}} \frac{a_i}{\|a_i\|}$, where i cycles from 1 to N

$\boxed{a_i^T \text{ is } i\text{th row of } A}$

If $0 < \mu < 2$, then

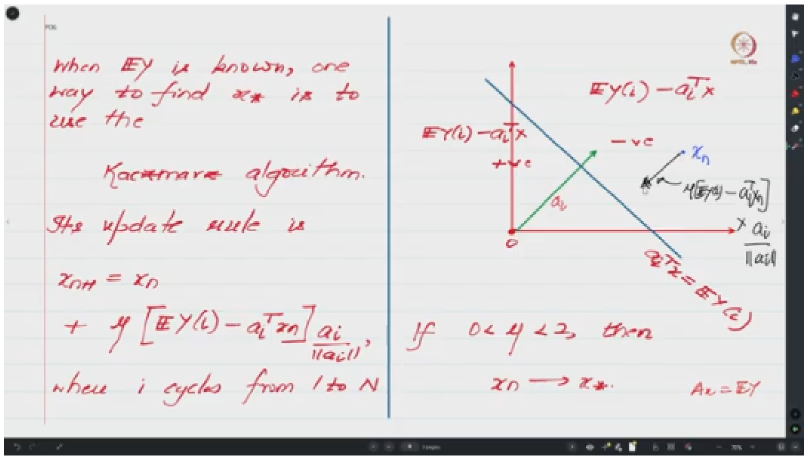
$x_n \longrightarrow x_*$.

$Ax = \mathbb{E}Y$

$\mathbb{E}Y(i) - a_i^T x$
$\mathbb{E}Y(i) - a_i^T x$
$+ve$
$-ve$
0
$a_i^T x = \mathbb{E}Y(i)$

Right, and one can show that this Ai vector is orthogonal to the vector joining or lying on the hyperplane, or the vector joining any two points on this solution set. One can show that that vector is going to be orthogonal to this vector over here. Is this okay? Right, and one can see that if you substitute the origin in this expression and let us presume that all coordinates of the expected value of y are positive, in which case this expression will be positive at the origin, which then implies that if you substitute any value of x on this side of this hyperplane, the value of this expression is going to be positive. And that immediately implies that the value of this expression is going to be negative for any x that lies in this region. And, for any x lying on this hyperplane, indeed, the value is 0, which implies that Ai transpose x is equal to the expected value of y. So, with this, you know, with these definitions in place, let us see what this algorithm is trying to do. So, let me erase a few things so that it becomes less cluttered. So, at time instance n, let us say we are at xn.

So, xn could either be here or it could be here. So, let us presume xn is somewhere over here. So, if xn is somewhere over here, by this fact one can see that this expression in the square bracket is going to be negative, and since it is going to be negative and we are multiplying it with this vector Ai, one can see that this expression, the whole expression, is going to be a vector that is in the negative direction of Ai, right? In other words, you know, this update would be something like this. So, this is your vector, which is mu times this expected value of yi, the expected value of yi, let me minus Ai transpose xn and times Ai by norm Ai. So, what we are doing here is you take xn, which is this vector over here, and to that you add a vector which is aligned in the negative direction relative to Ai. So, if you take a vector and add a vector that is negative along your Ai direction, I hope you can see that this xn will be brought closer to this hyperplane. Is this okay? And in the same spirit, if your xn lies on this side, then this quantity is going to be non-negative, which implies that we are taking xn.


When $\mathbb{E}Y$ is known, one way to find x_* is to use the

Kaczmarz algorithm.

The update rule is

$$x_{n+1} = x_n + \mu\left[\mathbb{E}Y(i) - a_i^T x_n\right]\frac{a_i}{\|a_i\|},$$

where i cycles from 1 to N

If $0 < \mu < 2$, then

$$x_n \longrightarrow x_*.$$

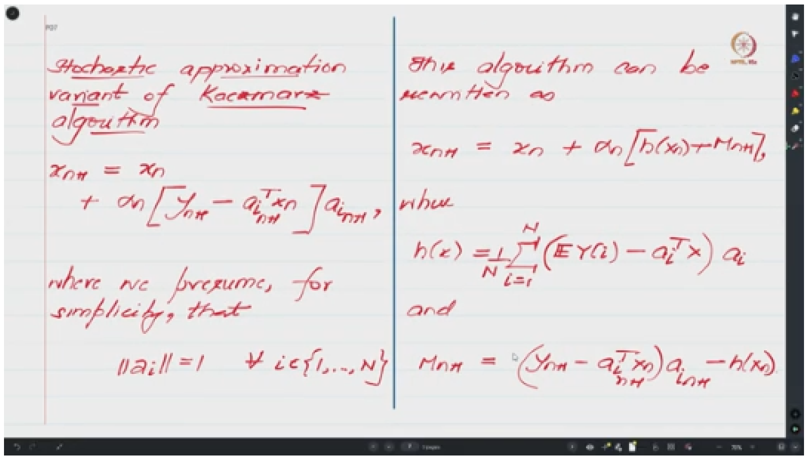
$A_x = \mathbb{E}Y$

$\mathbb{E}Y(i) - a_i^T x$
$\mathbb{E}Y(i) - a_i^T x$
$+ve$
$-ve$
x_n
$\mu[\mathbb{E}Y(i) - a_i^T x_n]$
a_i
0
x
$\frac{a_i}{\|a_i\|}$
$a_i^T x = \mathbb{E}Y(i)$

So, let us say somewhere over here is your Xn. right? And then what we are going to do is we are going to add a vector that is aligned along the AI vector. So, which implies that if you start at Xn, you would be pushed again towards the hyperplane, right? So, in this spirit, you know, the update rule or the Cox-Marx algorithm takes one hyperplane at a time and tries to push

Xn towards satisfying that hyperplane. So, this is roughly the intuition of Cox-Marx algorithm and as I said this I cycles from 1 to n. So, the idea here is at time step n or time step 0, you push X1 towards the first hyperplane then at time step 2, you push it towards the second hyperplane and so on and so forth. And iteratively you try to satisfy all the equations. And then one can show that if this step size mu if it satisfies a condition like this, right, then your Xn actually converges to X star, right.

So, recall that X star is given by this expression over here, right. So, this Cox-Marx algorithm indeed ensures that we can solve or find X star that satisfies the system of equations and is closest to the initial guess X0. So, this was some background about Kaksmark's algorithm. So, now what we are going to do is we are going to confront the original problem which is that we do not have a knowledge of expected value of y. Instead, we have access to samples of this vector y. So, of course what one could do is you know one could use these samples of Y come up with an estimate of expected value of Y and then use that estimate to you know use Kakshmat algorithm and then solve or find expected value of X or an estimate of that using this Kakshmat algorithm that I just discussed.



Stochastic approximation variant of Kaczmarz algorithm

$$x_{n+1} = x_n + \alpha_n \left[y_{n+1} - a_{i_{n+1}}^T x_n \right] a_{i_{n+1}},$$

where we presume, for simplicity, that

$$\|a_i\| = 1 \quad \forall i \in \{1, \ldots, N\}$$

This algorithm can be rewritten as

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

where

$$h(x) = \frac{1}{N} \sum_{i=1}^{N} \left(\mathbb{E} Y(i) - a_i^T x \right) a_i$$

and

$$M_{n+1} = \left(y_{n+1} - a_{i_{n+1}}^T x_n \right) a_{i_{n+1}} - h(x_n)$$

But, that would, in some sense, you know, make the algorithm offline because one would need to wait until a bunch of samples are collected, right? And then, you know, you come up with an estimate of the expected value of Y, and then once you have that estimate, you update the, you know, use Cox-Marks algorithm to find an estimate of the expected value of X, and so on and so forth, right. So, of course, one can do that, but the analysis of that algorithm is going to be very complicated because once you have additional samples of Y, one will need to additionally update the estimate of the expected value of Y and then use that additional estimate to improve upon your estimate of the expected value of X, and so on and so forth. So, in contrast, here is an algorithm that is online in nature, and it sort of takes in the new samples that appear and uses that to directly update your estimate of the expected value of X. So, this update rule will now be referred to as the stochastic approximation variant of the Cox-Marx algorithm. So, the algorithm proceeds as follows.

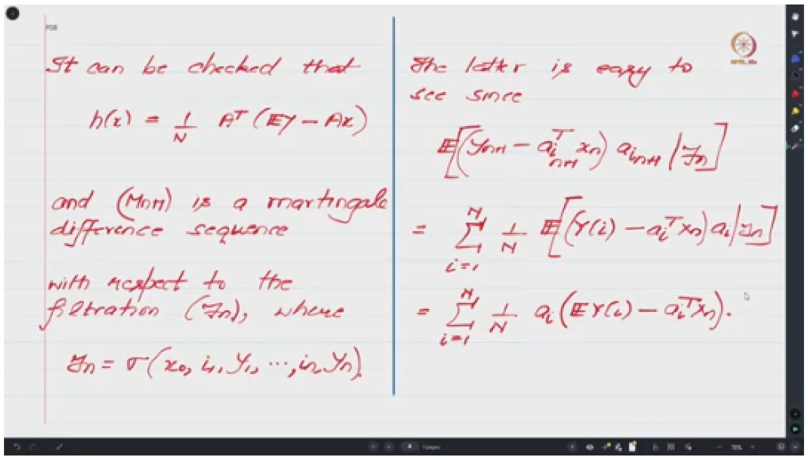
At time step n, given xn, it updates xn to xn plus 1 using the update rule xn plus 1 equals xn plus alpha n times some square bracket times a in plus 1. So, if you see the difference between this and what we had over here is that here we had a constant step size. We have now replaced it with a time-varying step size sequence, and here we had the expected value of y of i. Here we have replaced it with a sample, and the i has now been replaced with i n plus 1. Right, and also notice that here in the original Cox-Marx algorithm, you had Ai divided by the norm of Ai. Instead, here, you know, we only have Ai n plus 1. This is just to keep our analysis simple, right? And what we have done is basically we have presumed that all rows of A right, have norm 1, right.

And, you know, this is not difficult to do. We can just divide, you know, every equation by the norm of AI on both sides, and that would be one way to ensure such a condition. Is this okay? So, this is just to keep our analysis simple, right. And then,

So, this is the algorithm. So, notice here that your calligraphic Yn plus 1 are noisy estimates of the expected value of Y, and on top of that, these indices In, instead of being chosen in a cyclic fashion, are being chosen randomly. from 1 to capital N in a uniformly random fashion. So, again, this uniform assumption is for convenience. One can assume that it is being sampled from 1 to N using any arbitrary distribution.

So, this is the algorithm. Right, and as we usually do, this stochastic approximation algorithm can be rewritten in this fashion, where H of X is given by this expression, and Mn plus 1 is given by whatever you have over here minus H of Xn. Is this okay? So, one can ask, how did we cook up this H of X? Well, the strategy is as usual: we take this expression, take its conditional expectation with respect to the information available at time n. So, the information available at time n will tell you exactly what xn is. So, treating this as a constant, you then take the expectation of the remaining expression. So, if you take the expectation of the remaining expression and invoke the independence of your yi and in samples, one can easily see that the conditional expectation of

this expression with respect to the information we have at time n is what is given over here, right? More formally, one can show that if fn is the sigma field generated by these quantities—that is, your initial guess x0, your i1, calligraphic y1, I1 calligraphic Y1, I2 calligraphic Y2, all the way up to IN calligraphic YN—then one can show that with respect to this sigma field, the conditional expectation of this random variable is precisely what is given over here, which can be further simplified into this expression. It can be compactly written in the following way. So, notice that you have A i's, which, if you take their transpose, are the row vectors of A, and since there is no transpose here, it is not difficult to see that you will have A transpose here. This expression, when multiplied by something like this, results in or shows that this expression equals the expression we have over here.



It can be checked that

$$h(x) = \frac{1}{N} A^T (\mathbb{E}Y - Ax)$$

and (M_{n+1}) is a martingale difference sequence with respect to the filtration $(\mathcal{F}_n)$, where

$$\mathcal{F}_n = \sigma(x_0, i_1, Y_1, \cdots, i_n, Y_n)$$

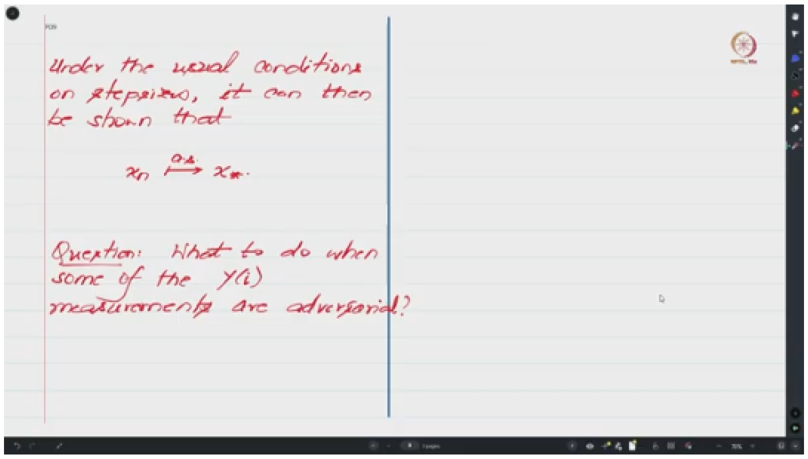
The latter is easy to see since

$$\mathbb{E}\left[\left(Y_{n+1} - a_{i_{n+1}}^T x_n\right) a_{i_{n+1}} \middle/ \mathcal{F}_n\right]$$

$$= \sum_{i=1}^{N} \frac{1}{N} \mathbb{E}\left[\left(Y(i) - a_i^T x_n\right) a_i \middle/ \mathcal{F}_n\right]$$

$$= \sum_{i=1}^{N} \frac{1}{N} a_i \left(\mathbb{E}Y(i) - a_i^T x_n\right).$$

So, now you know—I hope by now you can use your prints, I mean these characterization in terms of the limiting ODE. Right, and then use our four assumptions and then verify that those assumptions are satisfied. Then conclude that if those assumptions are satisfied, the algorithm should track the solution trajectories of the limiting ODE. Right? And if the limiting ODE is well-behaved—and in this case, one can show that the limiting ODE has a unique globally asymptotically stable equilibrium point, which is X star itself. Hence, one can show that since all solution trajectories of this limiting ODE equal to X star, and all the four conditions that we have been discussing throughout this course hold, one can show that in this case, X n converges almost surely to X star. Now, so far—so far, so good—we use the single time-scale stochastic approximation algorithm. I have not explained what a single time scale is, but whatever you are seeing so far is a single time-scale stochastic approximation algorithm, and one can see that it is good enough to estimate this quantity X star over here.



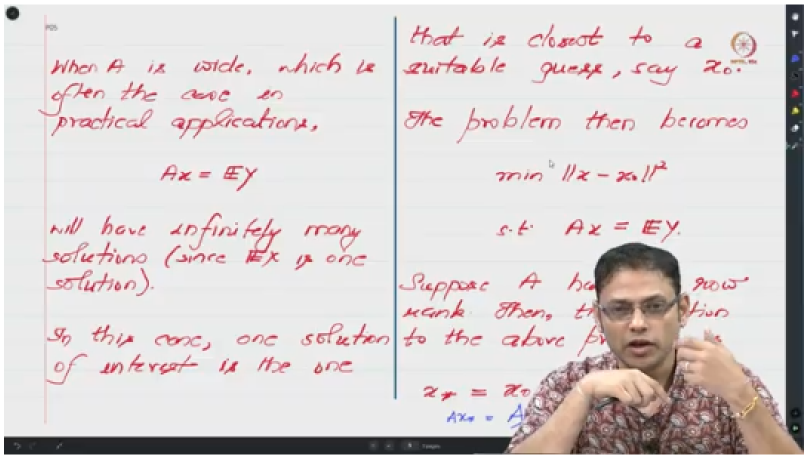
Under the usual conditions on stepsizes, it can then be shown that

$$x_n \xrightarrow{a.s.} x_*.$$

Question: What to do when some of the Y(i) measurements are adversarial?

So, now in the next couple of lectures, we are going to go beyond this setup, which will necessitate the use of slightly more sophisticated algorithms—in particular, the use of two time-scale algorithms as well as stochastic recursive inclusions. And the setup that we will be looking at in future classes is the following. Now, suppose some of the coordinates of Y—so, recall that Y is this vector made up of capital N coordinates—and our goal is to estimate the expected value of X. Now, suppose some of the coordinates of Y are controlled by adversaries. Adversaries are nodes or sensors that either have become faulty or are being controlled by agents who want to derail our estimation algorithm. So, now we do not know which measurements are being controlled by adversaries.

If we knew them, then of course we could throw away those samples and work with the rest. However, the challenge is that which of the estimates are adversarially controlled is unknown, and without knowledge of this, we would still like to come up with a robust estimation of the expected value of X. So, in this case, one cannot make use of this optimization algorithm that we saw over here. One will need to go beyond—in particular, in the next class, we will see that one way to deal with adversaries is to look at the L1 minimization of the distance between them. In which case, one will see that the simple stochastic approximation algorithm that we have designed is not good enough, and one will need to do something different—namely, make use of two time-scale nature and also this stochastic recursive inclusion idea. So, with that, let me end today's lecture, and I hope to see you in the next lecture.



When A is wide, which is often the case in practical applications,

$$Ax = \mathbb{E}Y$$

will have infinitely many solutions (since $\mathbb{E}X$ is one solution).

In this case, one solution of interest is the one

that is closest to a suitable guess, say x_0.

The problem then becomes

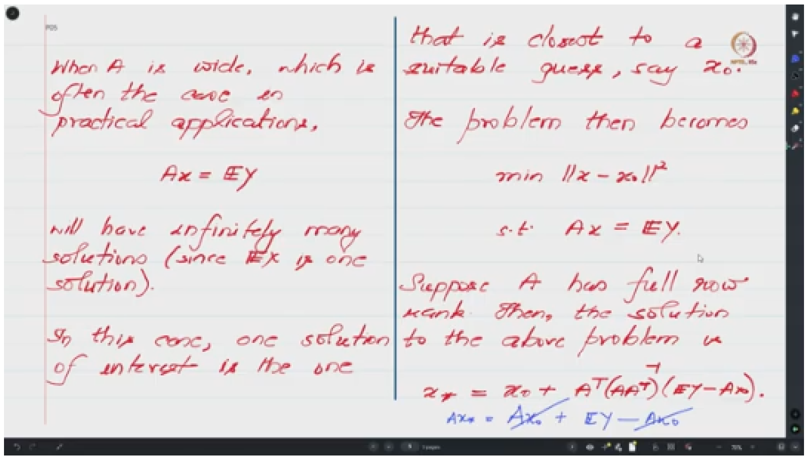
$$\min \|x - x_0\|^2$$

$$\text{s.t. } Ax = \mathbb{E}Y.$$

Suppose A ha[illegible] row rank. Then, t[illegible] to the above pr[illegible]

$$x_* = x_0 \ldots$$

$$Ax_* = A\ldots$$



When A is wide, which is often the case in practical applications,

$$Ax = \mathbb{E}Y$$

will have infinitely many solutions (since $\mathbb{E}X$ is one solution).

In this case, one solution of interest is the one

that is closest to a suitable guess, say x_0.

The problem then becomes

$$\min \|x - x_0\|^2$$

$$\text{s.t. } Ax = \mathbb{E}Y.$$

Suppose A has full row rank. Then, the solution to the above problem is

$$x_* = x_0 + A^T(AA^T)^{-1}(\mathbb{E}Y - Ax_0).$$

$$Ax_* = \cancel{Ax_0} + \mathbb{E}Y - \cancel{Ax_0}$$

Bye, goodbye, and namaste.