Distributed Optimization and Machine Learning

Prof. Mayank Baranwal

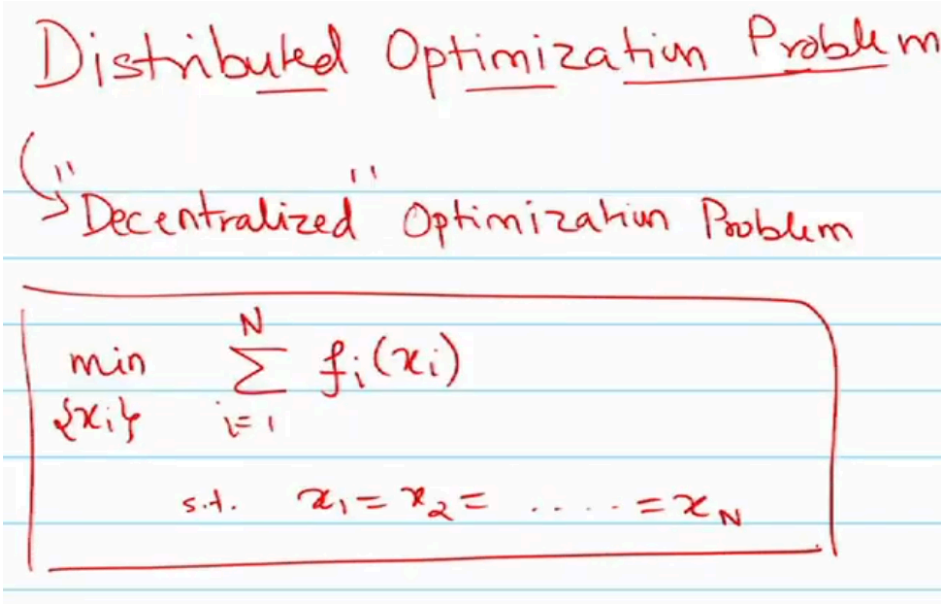
Computer Science & Engineering, Electrical Engineering, Mathematics

Indian Institute of Technology Bombay

Week-10

Lecture 35: Algorithms for Distributed Optimization

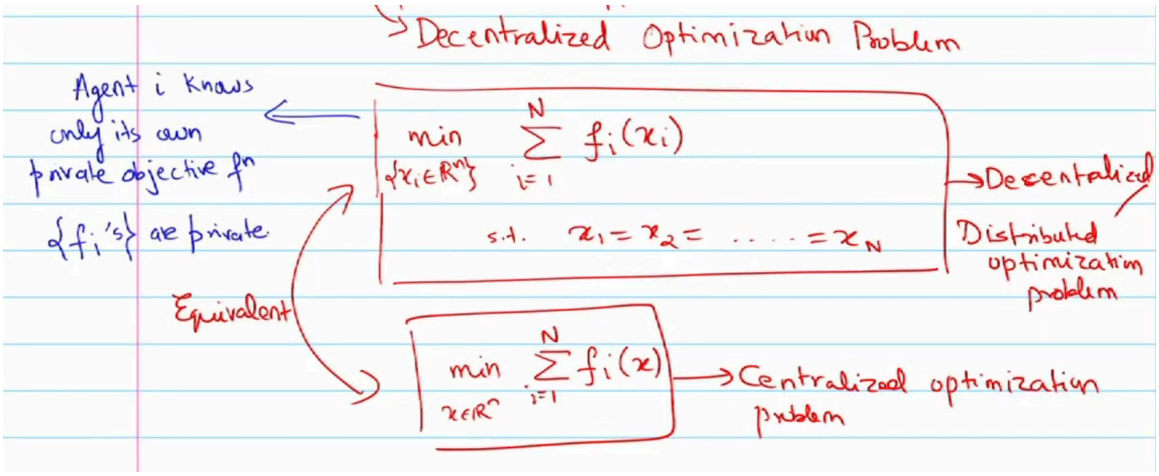
So, in the last lecture we looked at distributed economic dispatch problem and that was the sort of first introduction to a real world distributed optimization problem. So, today we are going to look at a more general form of distributed optimization problem. and we are going to look at a few algorithms first order algorithm, second order algorithm for solving distributed optimization problems as well as the fixed time equivalent of distributed optimization problem. So, in certain text you will also see it is called it is referred to as a decentralized optimization problem and when I say decentralized It is called decentralized because all the gradient computation that is going to be decentralized, but the communication is going to be distributed in the sense that you are only going to be communicating with your neighbors and there is no centralized computation or communication happening. So, in certain text you would see decentralized appearing instead of distributed, but at the core of it they are trying to solve a similar kind of problem. And the kind of problem that we are interested in is of this form.



So minimize x i with respect to x i f i x i subject to x 1 equal to. So, this is the kind of optimization problem that we are interested in. So, this problem is same as say x i is in R n. So, let me also say x i is in.

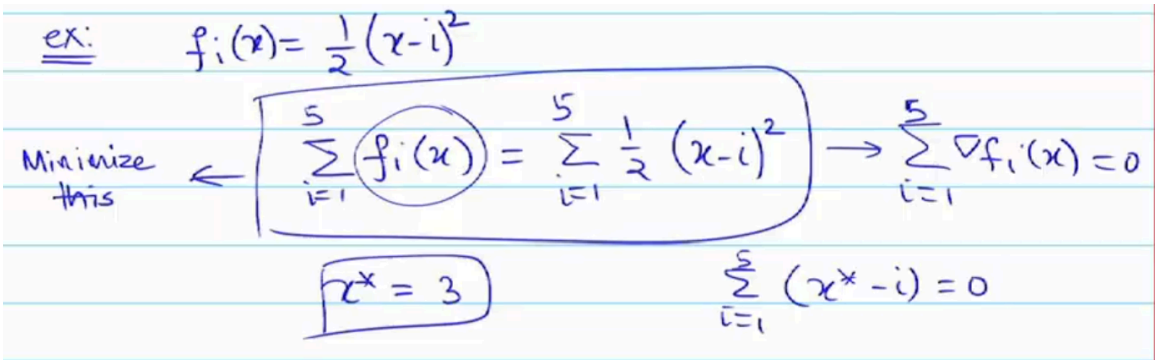
So, this is same as solving this centralized optimization problem which is i equal 1 through n. So, this is centralized optimization problem. and this is decentralized or distributed optimization problem. When I say distributed optimization problem, so that means, so first of all is this clear that the two optimization problems are equivalent. So, minimizing the summation of f i.

where x is your decision variable is same as minimizing the summation of fi of xi with the constraint that x1 is equal to x2 is equal to x3. So these two problems are equivalent. So equivalent formulation or these formulations are equivalent. But the reason we call it distributed optimization problem is because we are going to make certain restrictive assumptions on what is known. So agent i knows only its own private objective function.



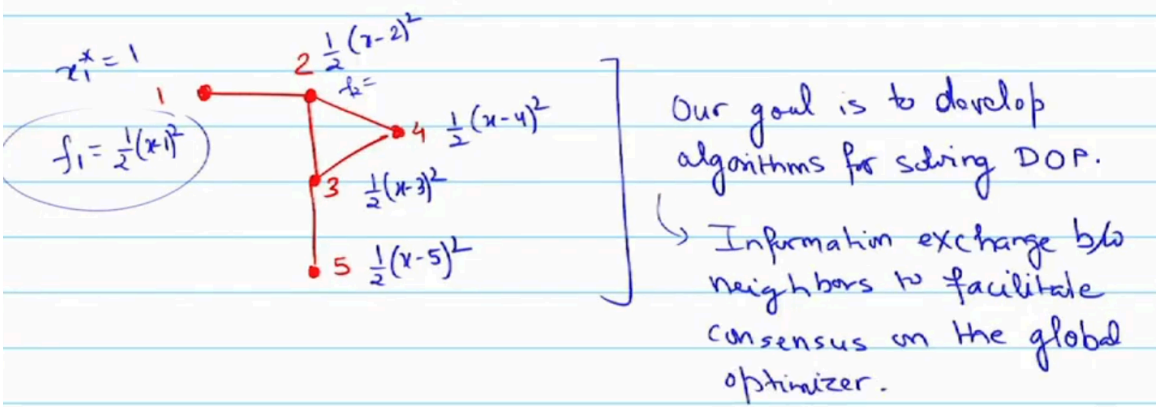
So, what do we mean by that? So, these fi's are private. So, agent j must be having some other objective function in mind, agent i has some other optimization problem to deal with. but together they are trying to solve this cooperative or a sort of common together they are trying to minimize this common objective function right. So, let us say I look at a problem of this form. So, example would be f i of x given by half of x minus i whole square ok.

And if I look at this particular centralized optimization problem i 1 through f i of x which is same as let us say n is equal to 5. So, there are 5 such 5 suppose this is what we are trying to minimize. So, what is the objective optimal value for this particular or what is the optimizer for this particular optimization problem. So, we want to minimize this. So, what is x star? what are the objective functions here or fys half x minus 1 whole square half x minus 2 whole square x minus 3 x minus 4 and x minus 5 whole square.



So, you can show x star turns out to be 3 which is the middle of like the midpoint of 1 2 3 4 5 right. So, you can in fact show that x square turns out to be 3 and as I mean simply you can set the gradient. So, let us call this So if you set this gradient to 0, 1 through 5, gradient fi of x, if you set this to be equal to 0, so what do you get? You get x star minus i, 1 through 5, this is equal to 0. That means basically x star is the average of 1, 2, 3, 4 and 5 which happens to be 3. So, this is the centralized optimization problem, but it may happen that these may these objective functions individually are being or individually are being held private by different agents in the network.

So, you can imagine a scenario for instance, you have a network like this 1, 2, 3, 4 and and for the first agent the objective function is half x minus 1 whole square, for the second agent it is half x minus 2 whole square, x minus 4 whole square. So, agent 1 only knows about its own private objective function which is f 1. Agent 2 knows about its own private objective function which is f 2 and so on, f 2, f 3 and so on. Now, they are trying to minimize the sum of this cumulative objective function and if I mean if they want to minimize this without having to know what fi's for other agents look like, then they need to exchange certain information. Because if every agent greedily minimizes their own objective function, so in this case x1 star turns out to be 1, x2 star turns out to be 2, x3 star would be 3, 4 and 5.

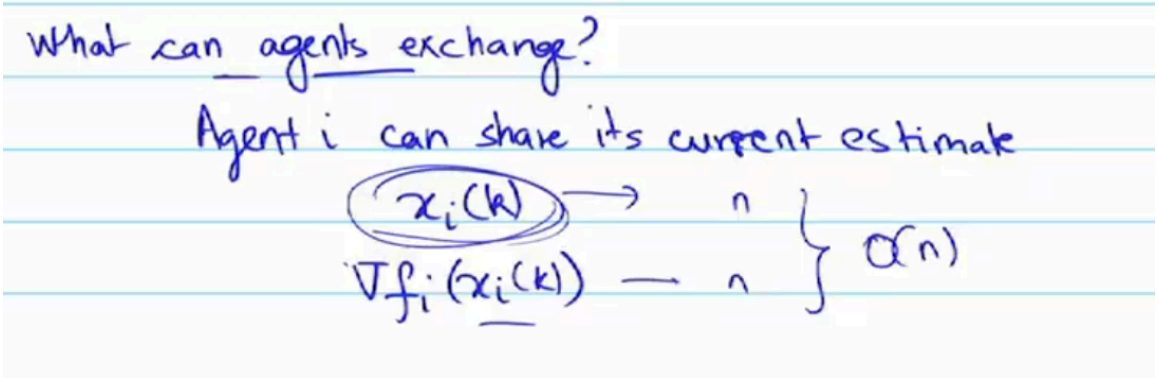


So everyone's belief about this global optimization problem. or the global optimal value is going to be different right because they do not know what other agents are trying to minimize because agent i is unaware of agent j's objective function. So that means they

need to exchange certain information and therefore we also need to run consensus on x i's. So that eventually every agent not just they end up minimizing a part of their own objective function, but also a part of the global objective function or some their neighbors objective function and eventually they arrive at the same optimal value right. So, our goal today is to develop algorithms for solving distributed optimization problems DOP ok.

And one thing that we are going to I mean we are going to be agreeing with is that you need certain kind of information exchange between neighbors to facilitate consensus on the optim global optimizer right that is the only way they can. So, maybe agents can exchange information about x star or maybe they can exchange information about their gradient for instance. So, let us say agent 1 has its current estimate. So, what can agents exchange or what information can agents exchange so maybe let's say agent i can share its current estimate current estimate xik right let's say the kth iteration it can share its current estimate of what the optimal optimizer looks like or it can also exchange maybe information about So what is the current value of the gradient at its current estimate of the, current value of its own gradient. I mean again because it does not have information about fj's, so it can only evaluate its own gradient.

So what is the current value of its own gradient evaluated at the current estimate xij. So maybe these are the kind of information that agent i can exchange with its neighbors. and if agent i only exchanges information about x i's. So, that means we are going to be working with first order algorithms where they are going to be running consensus on x i's and at the same time trying to minimize the global objective function. As we had seen in the context of Nestrov's accelerated gradient and heavy ball method that Sometimes you also add momentum term which is basically dependent gradient of the function.



So, you use the information at the previous step to essentially accelerate the convergence and in that case you also need I mean the agents also need to exchange information about the gradient of f y. So, if x is n dimensional gradient is also n dimensional. So, in total you are going to be exchanging order n dimensional information. which is in this case 2 n, but I mean you are not sharing information which is basically order n square or order n cube and so on, it is still order n. So, this is the total amount of information they will be

sharing if they end up sharing both, but at least this amount of information needs to be shared.

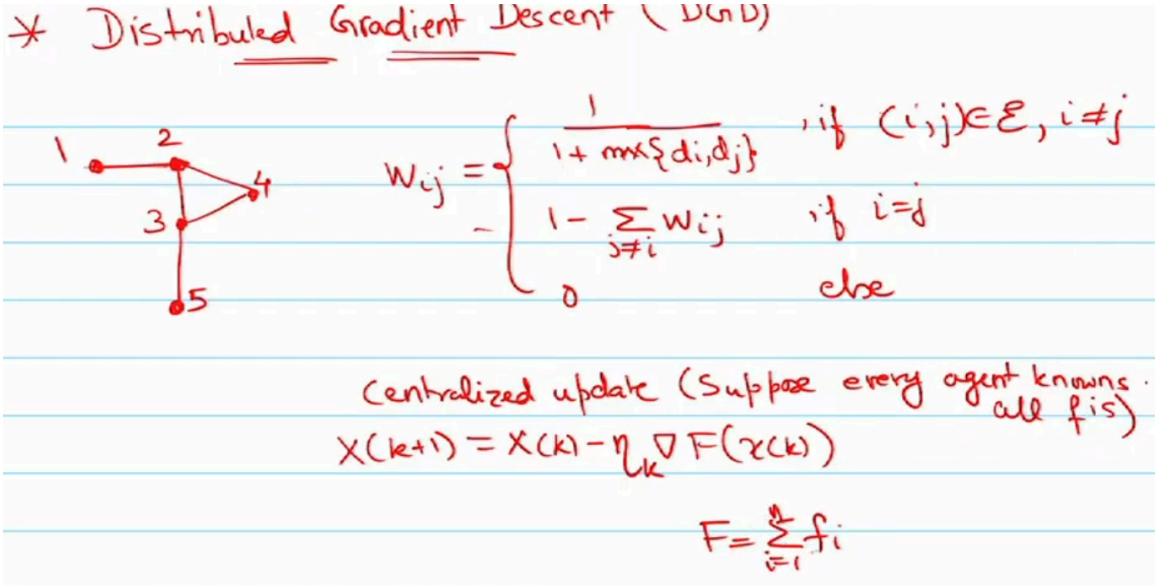
Is this clear? So, let us look at our first algorithm. f to be, f is going to be, it is a function, x need not be scalar, f is a scalar, f is a function. So, the first algorithm that we are going to be looking at is called distributed gradient descent So, imagine you have a graph like this, so let us also redraw this. So, you have one say you have 5 agents in the network. So, as we had seen in the context of consensus, we define basically mixing matrix or the doubly stochastic matrix, right.

And we had used this formula that w ij is going to be 1 over 1 plus max of d i d j if basically this pair i j belongs to your edge set and i not equal to j its 1 minus summation j not equal to i w i j if i equal to j and its 0 else right. So, if there are no edges then its 0. So, we had designed this doubly stochastic matrix right we had looked at this particular formula for designing doubly stochastic matrix. and which basically comes from metropolis rating. So, we are going to be using like in that when designing the discrete time algorithm, we are going to be use making use of this particular wij here right.

So, in this case wij I mean we basically will have values for edges connecting 1 to 2, 2 to 4, 2 to 3, 3 to 5 and so on as well as self loop, self edges which is essentially w 1 1, w 2 2 because of this. Alright, so let us say there was no I mean everything was known right like let us say the you know like every agent knows every other agents private objective function. So, it is not private anymore let us say. So, then what would have been the dynamics for x? Let us say it is a centralized optimization problem. So, this centralized optimization problem that we are trying to solve.

So, what should be the dynamics of x? A simple gradient descent scheme would look like what? So, you have x k plus 1. So, this is centralized update. Suppose every agent knows every other agents knows all f i's right. So, that is when you are going to run a centralized update like this which is x k minus some step size eta k times gradient of f evaluated at x k with this capital F is nothing, but summation i 1 through n f i. So, this was this would have been the centralized dynamics.

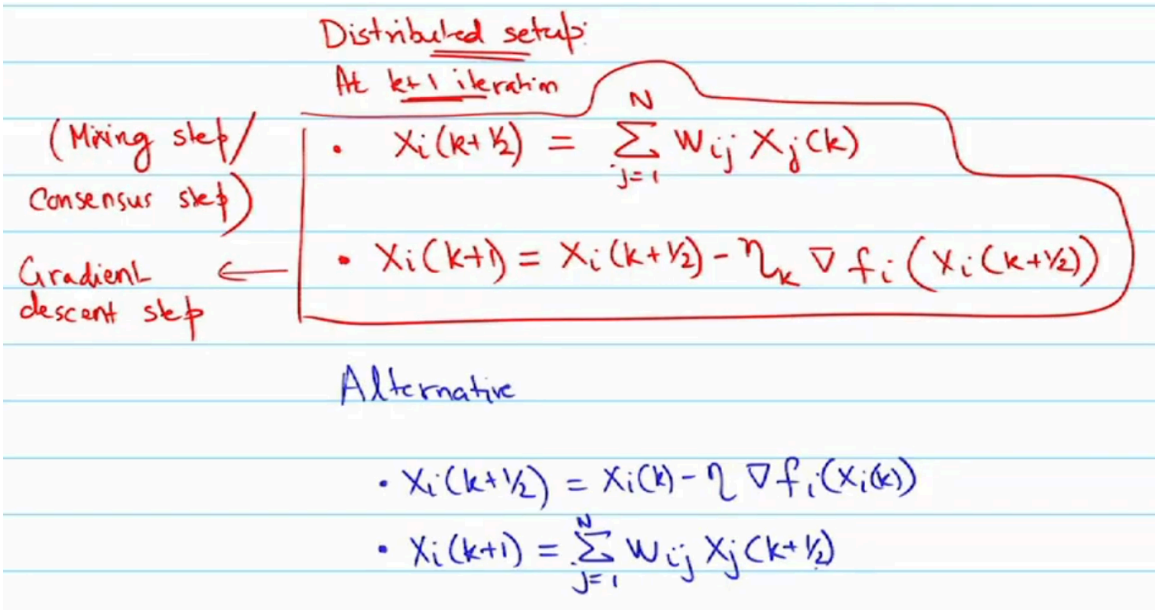
So, let me write this and capital F is basically the summation of this. So this would have been the update rule for update. But then in case of distributed optimization, so when you have distributed setup, certainly this is not possible. So we define, so every agent is going to have its first of all own copy of what x would look like. So there is no common x connecting them.



So agent i is going to updates let us say something called x i k plus half which is going to be a simple consensus step. So, j 1 through n w i j x j k. So, let us say at the end of k iterations every agent has a value x 1 k x 2 k x 3 k and so on. So, this is called mixing step or consensus step.

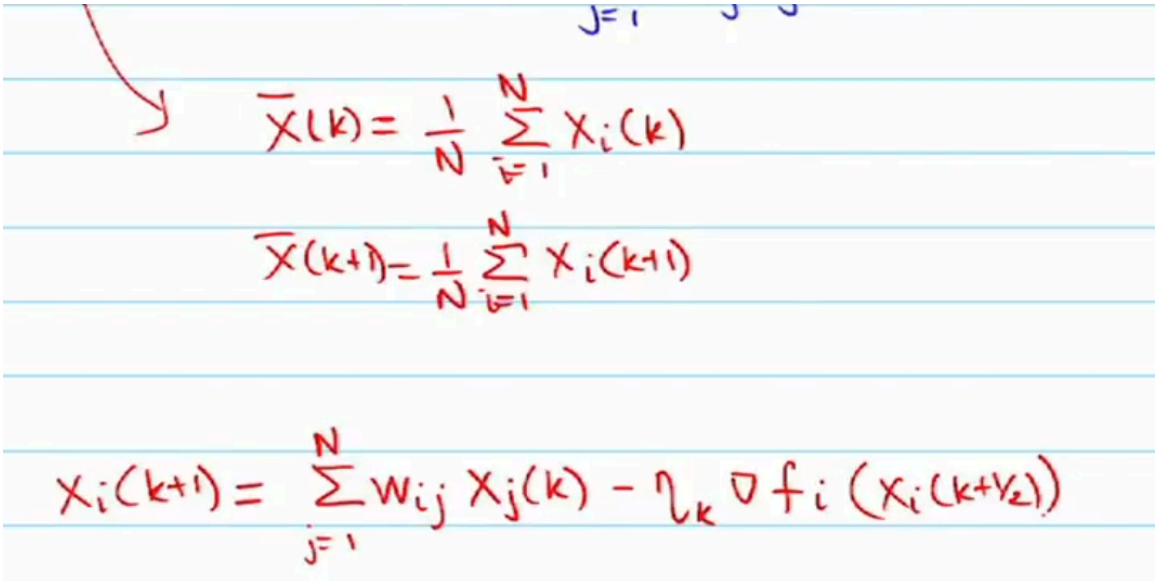
or consensus step. So, this is so at k plus 1 iteration this is what happens. So, we compute the x i k plus half let us call it an intermediate value and then based on this we compute an updated x i k plus 1 which is going to be a gradient descent on this. minus eta k times gradient of fi. Is this scheme clear to everyone? So, this is what this particular algorithm does. Essentially you have a consensus step or the mixing step followed by the gradient step or gradient descent step.

There are other ways to I mean it is possible that you can run the gradient descent step first followed by the consensus and that would work too. So, an alternative would be simple alternative would be you define x i k plus f to be x i k minus and then you define x i k plus 1 as j 1 through n w i j x j k plus half. So, that is also possible. So, you can do the gradient descent step first followed by the mixing step and this would also be fine thing to do. Is this algorithm clear to everyone? So, we are running a mixing of the consensus step followed by the gradient step and with while we are not going to go into the proof of why this distributed gradient descent algorithm or the DGD algorithm works.



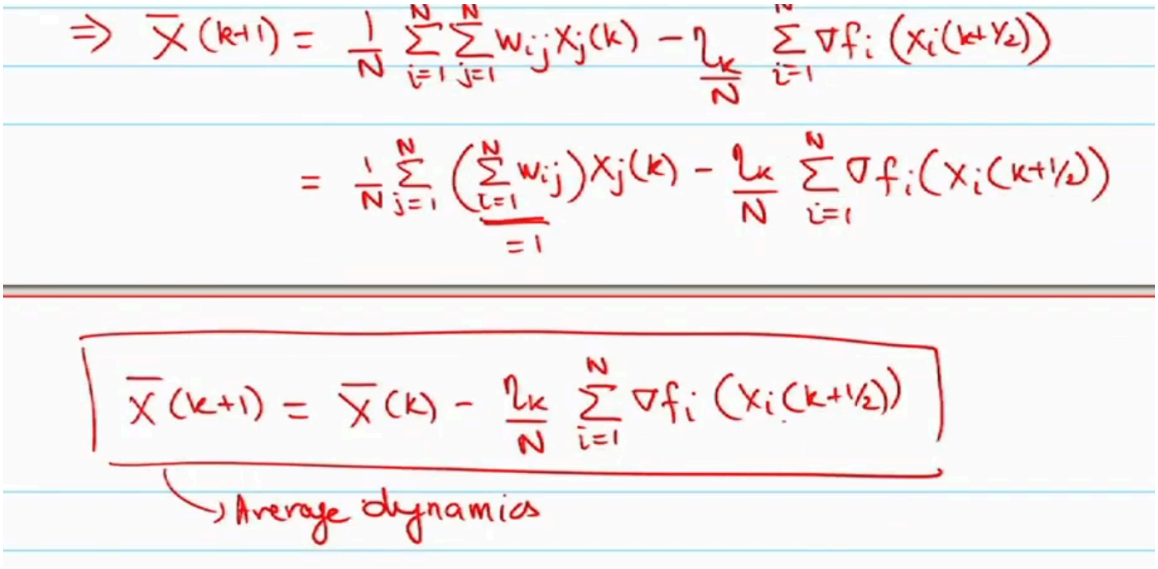
Let us try to develop some intuition as to why this would work. So, let us try and analyze this particular algorithm. So, we are going to define x bar k. So, we want the average dynamics to converge to the optimal solution right. So, what is x bar k? We are going to be defining x bar k as summation i 1 through n x i k and likewise you have x bar k plus 1 defined which is going to be 1 over n summation i equal 1 through n x i k plus 1.

ok. So, we are trying to understand how the average dynamics kind of behaves ok. This is what we are trying to do. So, if I look at this particular algorithm and I combine the two steps because I know what x i k plus half looks like. So, let us do that. So, if I combine both steps x i k plus half is nothing but summation j 1 through w ij xj k right and then you have minus eta k in gradient fi and then you have x i let me write it as x i k plus half for now ok.



Now if I try to write down this dynamics in terms of this average dynamics which is x bar k so that means I need to add basically sum it from 1 through n and take 1 over n. So, this implies x bar k plus 1 is going to be 1 over n summation i 1 through n summation j 1 through n w ij x j k minus theta k over n summation i 1 through n. is this clear? Now what is this quantity? So, if I look at this particular quantity 1 over n summation j 1 through n summation i 1 through n w ij x j k. So what is this quantity? What is this equal to? So this is essentially the row sum right and row sum for a doubly stochastic matrix is 1.

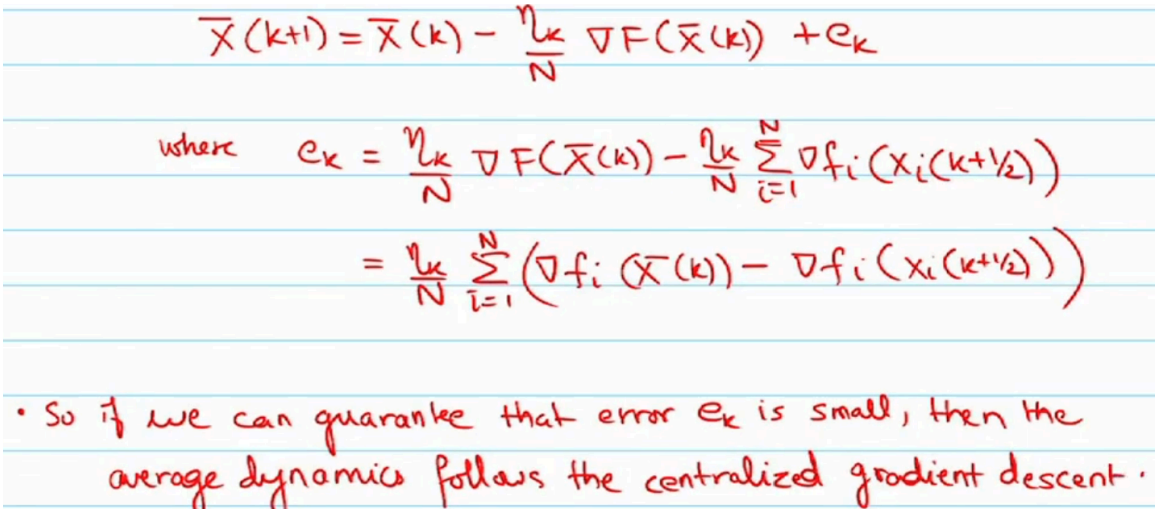
So this is equal to 1. So that means I can write x bar k plus 1 as 1 over n summation x j k right. So this becomes x bar k minus eta k over n times i equal 1 through n gradient fi x i k plus half. So, this is what we end up getting for the average dynamics. So, this is average dynamics And if I look at the centralized update, so centralized update was xk plus 1 which is for the average dynamics is xk minus eta k times gradient of f of xk right, capital F. We are almost in that, so we can, so you can equivalently see that I can write this as xk plus 1 is x bar k minus eta k over n gradient of f of x, x bar k.



plus some ek error term, why because this is defined in terms of x i k plus half whereas, this centralized update was defined in terms of in terms of a common x k right ok. So, therefore, where ek is ek is eta k over n gradient of x power k minus eta k over n summation i 1 through n gradient fi of x i k plus half or I can write this as eta k over n summation i equal 1 through n gradient fi x bar k minus gradient fi x i k plus half. ok is this clear. So, this is the as long as this error is small. So, that means your average dynamics is going to follow your centralized dynamics right.

So, if you can make the error small. So, that is when the average dynamics would follow the centralized dynamics. So, if we can guarantee that error E k is a small, then the average dynamics follows the centralized dynamics centralized gradient descent ok. and

how can we ensure that ek is small. So, if a gradient is let us say if a gradient is L Lipschitz then you can bound this in terms of the difference between x bar and this is a standard argument that is eventually used to prove that ek is going to be small. So, if you assume that f is let us say L smooth that means a gradient is L Lipschitz and you can use those arguments to basically bound this and therefore you can show that this kind of this kind of distributed gradient descent would actually converge to the optimal solution.

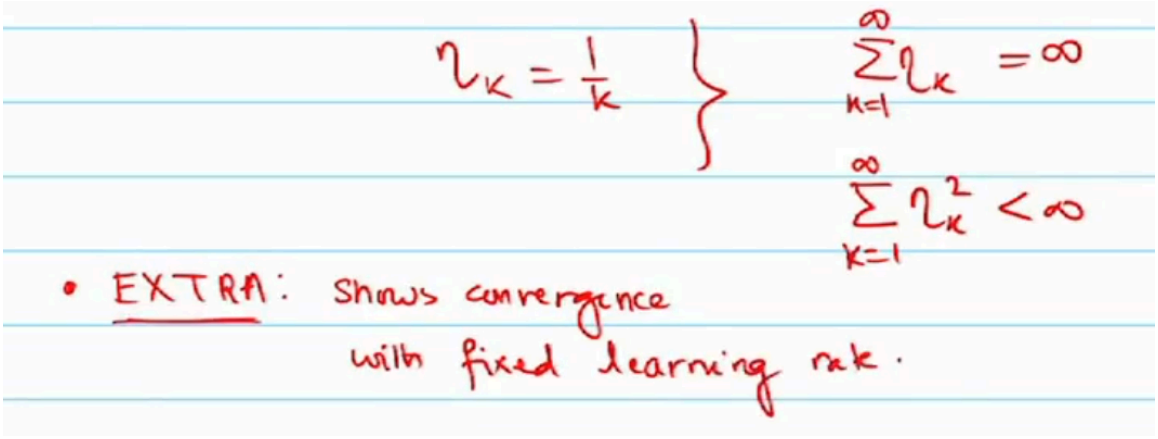


Optimal solution of the global objective function ok, not the individual objective functions. So, every agent would converge to the same value, same optimizer and or same value and that value happens to be the optimizer of the global objective function. Yeah you have to prove, so usually this is done using, so usually when I say eta k it is kind of taken to be 1 over k kind of learning rate, so that eventually anyway you make it go to 0. Yeah that is what I am saying, if you make eta k to be 1 over k it will become 0 right, it will eventually go to 0 as k goes to number of iterations go to infinity. So, then right, but then this term may also be very large right.

So, you want to also ensure that this term also becomes smaller along with eta k to be 1 over k, what if this term grows like I mean more than k. So, you also have to ensure that. So, using L Lipschitzness and let us say you also I mean if you want to expedite the rate convergence rate then you also have to assume that f is mu strongly convex and so on. But assuming f is L smooth you can still f is L smooth you can still pretty much show that this E k is going to be small. And these are the standard arguments that we had used when proving the centralized convergence of the centralized gradient descent using similar very similar arguments you can show this as well.

So, typically this the learning rate that is used is for these kinds of approaches is eta k is like 1 over k essentially you want Robbins Monroe kind of condition saying that summation eta k k 0 k essentially 1 to infinity they should diverge and summation k 1 to infinity eta k square they should be less than infinity. So, one of the criteria being 1 over

k. So this is a sufficient condition by the way. There is another algorithm. So DGD was one of the sort of primary or the algorithms I think it was a paper by Nedich and Olszewski where they showed the convergence proof of DGD.



There is I mean therefore then sort of more refined convergence proofs particularly this algorithm called EXTRA which essentially shows convergence with fixed learning rate, slightly modified version of what we have seen, but it shows convergence with fixed learning rate. Thank you very much.