Distributed Optimization and Machine Learning

Prof. Mayank Baranwal

Computer Science & Engineering, Electrical Engineering, Mathematics

Indian Institute of Technology Bombay

Week-10

Lecture 39: Large-Scale Machine Learning

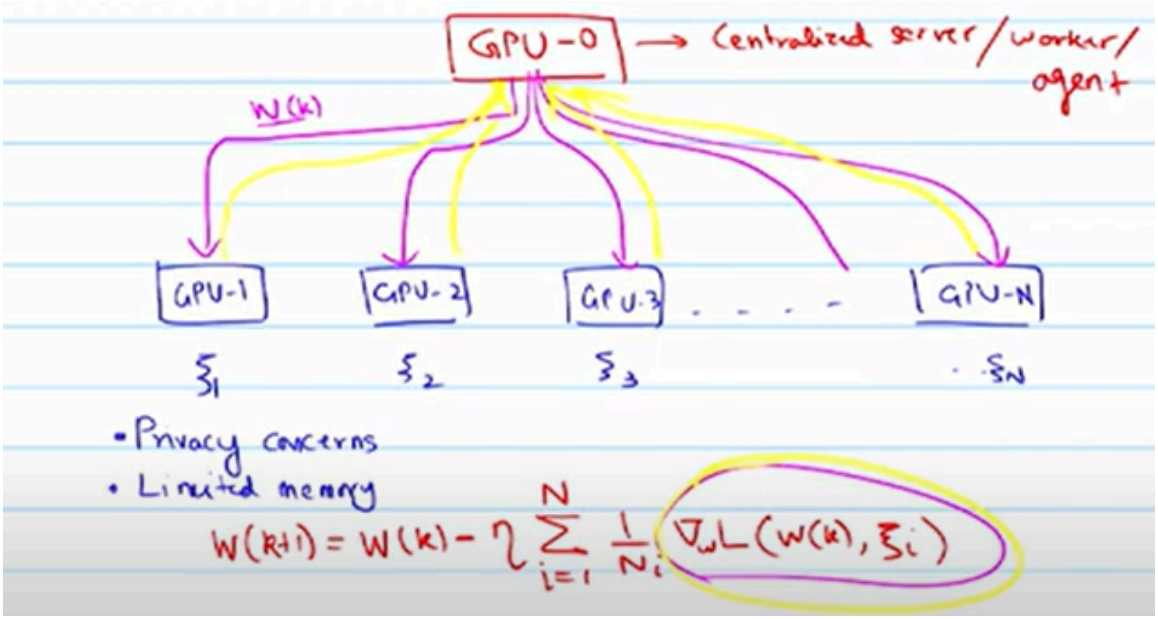
So, now this basically sort of gives us an idea as to how to train a neural network using maybe a stochastic gradient descent like this or of mini-batch gradient descent like this ok. Now think of a scenario I mean this is basically tells you how to train a neural network and basically what I mean what essentially we are trying to implement this particular thing right effectively. It can be gradient computation let's say I have these end data points distributed across multiple agents. And these agents, I mean, think of it as if you have your own private data, right? And when Google tries and sort of deploys, tries to train its global model, it would need inputs from all its users, right? So in that case, your data is going to, I mean, your data is private, so you wouldn't be sharing the data with anyone else, so there won't be a centralized training of this. But then maybe together you, I mean it's possible that all of us would aggregate our gradients together, sum it over and then send it to Google and maybe they'll train a centralized neural network. They'll get the new set of weights and this new set of weights then will be sort of relayed back to us.

So there is, like in terms of how do we work with large-scale data. So you can have a centralized architecture which is also called a parameter server approach and usually these are trained on I mean your networks are trained on GPUs let us say call GPU 0 which is like yeah. centralized server in some sense and then you have a bunch of other GPUs. So, GPU 1, by the way servers and workers these are sort of agents all these are in touch use interchangeably.

So, servers you can also call it worker or an agent. So, all these are used interchangeably. So, you will find based on what text you are referring to. So, we have a parameter server approach right? So, what happens in this particular case? So, let us say at the beginning of the first like in beginning of the training. So, first of all every agent has got its own private data.

So, they have their own private data. So, let us call it zeta 1, zeta 2, zeta 3. So, they have their own private data to work with right and ideally what should happen is. So, the

algorithm that needs to be implemented is wk plus 1 is wk minus let us say the m data points every agent has or every agent has got ni data points I should i equal 1 through n sorry ok. So, this is what.



I mean ideally let's say you have all these n times like basically n1 plus n2 plus n3 and so on and n times the data like basically these many data points right and if you had enough memory to store all of it and if you have enough computation then everything can be trained on a single GPU right and also assuming that there are no privacy issues. So everything can be trained on a single GPU but usually we have either privacy concerns or you have let us say limited memory that it is very difficult to store all the data points at a single place right. And that is when we want to train this neural network what happens is that the first iteration this information is basically sort of relay to all the servers. So the current like at the kth iteration after the end of kth iteration the centralized server basically tells every agent that the current weights are wk ok. Now on this wk every agent computes this on their own data, this gradient on their own data and then they basically relay this gradient information.

So this gradient is computed, this gradient information is relayed back. to the server who just adds all the incoming entries instead of storing their gradients separately. So, that I mean if let us say the weight vector is d dimensional instead of working with n times d dimensional vector or storing them separately with every d dimensional vector it receives it just adds the previous one right because of this additive nature of the loss function ok. So, it just adds them once everything is added then it this centralized server it performs this gradient update or essentially the parameter update which is wk plus 1 is the current estimate wk and the added the all the entries that it has gotten. So, this gradient information that it has got aggregated gradient information, it performs this parameter update and this new value wk plus 1 is then relayed back to the GPUs and then they

compute this loss derivative of the loss function on this new updated value wk plus 1 and this process goes on right.

So,t using a centralized server you will still have a distributed training of neural network, but then we assume an information we assume the presence of a centralized server like this. So, what is the shortcoming of this particular approach? Something that is very evident. Yeah. So, one thing is. One thing is, I mean, there's a single point of failure, right? If the central server doesn't, like, I mean, if it stops working, then, I mean, the total communication is broken.

So there's a single point of failure. And what is the communication bandwidth requirement on the centralized server? This is N, if there are N servers, this is N times D, right? So the communication bandwidth requirement is basically, it scales with the number of Scales with the number of agents and number of workers, right? Communication. Scales with the number of servers. Right, so this is not a viable architecture, a viable setup if you have very large number of agents in the network, okay. Because you have a huge communication required bandwidth requirement over here.

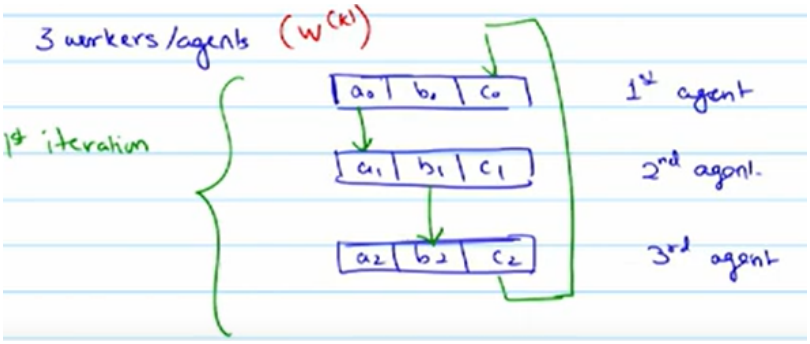
It would have been much ideal that if this bandwidth requirement is distributed between the servers and not necessarily towards one particular entity, right. These are two sort of challenges. And in order to account for, in order to sort of elevate these challenges, So you guys know this company Baidu, right? Baidu, which is Google equivalent in China, Baidu. No? You guys know Andrew Yang? So Andrew Yang, in fact, for a brief period he had worked at Baidu as well. Anyway, so Baidu sort of, I mean it's the name of the company, but they came up with this algorithm called ring all reduce algorithm ok.

So, a particular advantage of this particular algorithm is, so communication bandwidth requirement is constant in the number of agents, which is a significant thing. I mean it is, so the bandwidth requirement is going to be constant in the number of agents. So, no matter how many agents you add in the network. it is basically that required bandwidth requirement is going to be constant and also there is there is no centralized ok. So, this particular algorithm has two steps just like ADMM you had seen right. So, this algorithm also has two steps it has scatter reduce and all gather this algorithm has two steps and the way this works is now let us say to start with Every agent let us say they have at the kth iteration at the end of the kth iteration they have they know the current weights wk ok. Let us say this is the case. Now let us just for example consider you have 3 agents 3 workers. So, what every agent does is? So, they divide the data set into 3 parts.

and let us call it A naught, B naught and C naught for the first agent. Similarly, second agent would divide this as. So, there are n agents essentially you divide your dataset into

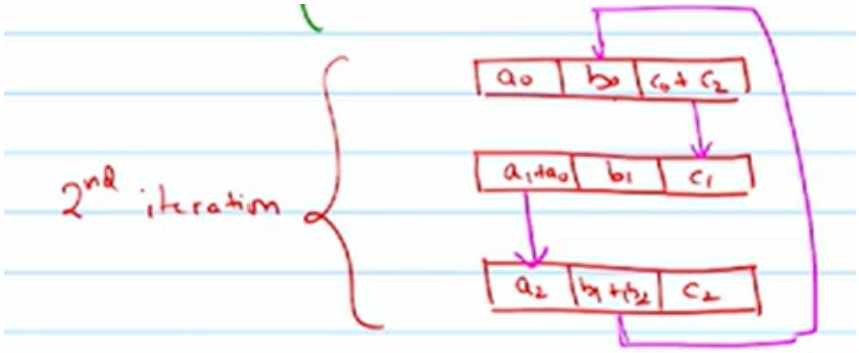
n points like this and then So, this is for the second agent and for the third agent. So, in the first iteration, so let us see. So, how do the iterations of this particular algorithm work? So, again to start with we have the current set of weights W k.

I mean there is an outer iteration which is basically the weights the kth iteration where the weights of the networks are like Wk current set of weights. So idea is to be able to sum all the entries right because that is what we want. We want like the again if I look at here we essentially want to be able to sum these entries. So if on these Wk's I can on like given the Wk on my data points I can compute the gradients But I need to be able to sum this and exchange this eventually. So let's see how this works.



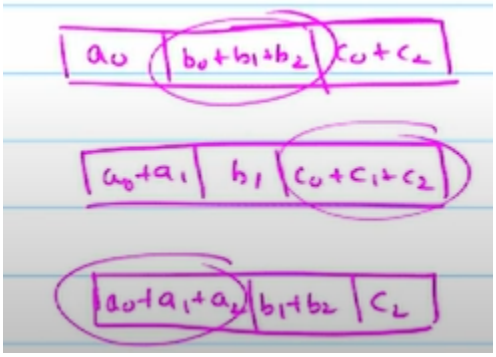
So in the first iteration, let's see the first iteration of scatter reduce. So, the first agent relays the first block information on the first block to the second agent, second agent relays the information of the second block to the third agent, and likewise this agent and when it relays that information the incoming information is simply added to the previous entry. So, what happens is after first iteration what happens? Is everyone following this? Three different sets of data points. If there are n data points then you basically have n different sets of data points.

The first block. Think of it as the first block, second block, third block. If you have n agents then you have n blocks per agent. So if let's say you have 4 agents and you have 50 data let us say 40 data points. So, it will be blocks of 10 each. So, there will be 4 such



blocks per agent. So, A naught B naught and the third entry would be because it has an incoming entry from C 2. So, it will be C naught plus C 2. Likewise you have a 1 plus a naught ok, b 1 and c 1 and similarly you have a 2, b 1 plus b 2, c 2. Is this clear? They are extending the gradients. All we are trying to get to is the sum of these gradients right.

Those are these gradient information right on the data points. Is this clear? So, this is what happens after first iteration. Then you have the second iteration. So, in the second iteration essentially you have this information, then you have, so yeah, so you will have this information and what did I miss this one here right ok. So, basically, you have moved everything over by one place.



So, it is the same operation, but we have moved everything over by one place, but what happens after the second iteration is, let us see what are the entries that we have. We have a naught here, we have b naught plus b1 plus b2 and we have c naught plus c2. a naught plus a1 b1 then you have c naught plus c1 plus c2 and the final iteration we have a naught plus a1 plus a2 for the final agent b1 plus b2 and c2. Now, if I look at the 3 agents. So, this is just this by the way after 2 iterations right.

So, if I look at the 3 agents. I had the right piece of information at across different distributed across different agents. So, let us say had this been n agents then you would have those n correct pieces of information distributed across those n agents right. So, since we divide the entire data into n blocks and this happens after how many iterations 2 iterations which is n minus 1 iterations. So, after n minus 1 iterations we actually have the right set of information right distributed across this I mean we are not done yet because if every agent I mean there is no centralized agent right so every agent if they were to be they were to ensure that their I mean their WKs are always in sync then they have to have the entire set of information and not just the one particular block correct piece of information right. So that when they do wk plus 1 everyone independently does wk plus 1 is wk minus eta times the sum of all the gradients that sum of all the gradients should be known to every agent.

In this case a part of it one part of it is known one part of it is known and one part of it is known and that information is distributed across different agents as of now. But then how many like what is the bandwidth requirement that this is basically constant in the number of agents. This bandwidth the communication bandwidth requirement the total cost communication cost is going to be constant. So, this was the scatter reduce step. Let us see what happens in the all gather step.

So let's say you have, like let's say if the first agent has gotten, has 40 data points, like every agent let's say has 40 data points and there are four such agents. So a0, b0 and c0 are the gradients computed on the first 10. So let's say for the, with respect to the- If there are four agents. If there are n agents then you have n such blocks So everyone has information about wk. Let's say they had the information about w0 to start with, the same w0. Now for the agents to run this. So everyone needs to know, because now there is no centralized entity. So everyone needs to know the entire sum of, like there are 160 data points, let's say, if there are four agents, there are 160 data points. So everyone needs to know the gradient computed on the entire 160 data points so that everyone can run the same equation and get to the same Wk plus one.

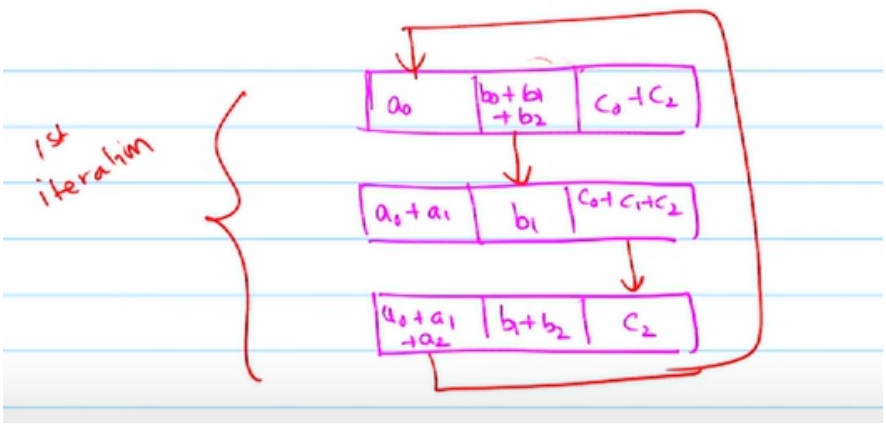
Because now there is no centralized entity that's actually helping with coordinating between the agents, right? As to relaying what Wk is. So is this scatter reduce part clear to everyone? So now let us move look at the all-gather step. So maybe use a new page. Exactly after n minus 1 iterations not some amount after 2 iterations.

In this case there were 3 agents. So 3 minus 1 which is 2 after 2 iterations the you will have the right set of information, so just for one particular let us say this would have been the information about one agent right to start with this was in fact the second agent or let us say this particular block. So this b0 plus b1 plus b2 is available, ideally I want everyone to know what is a0 plus a1 plus a2, b0 plus b1 plus b2, c0 plus c1 plus c2. Right now they only know a0, b1, b2, right now they know these respective blocks but one agent at a time.

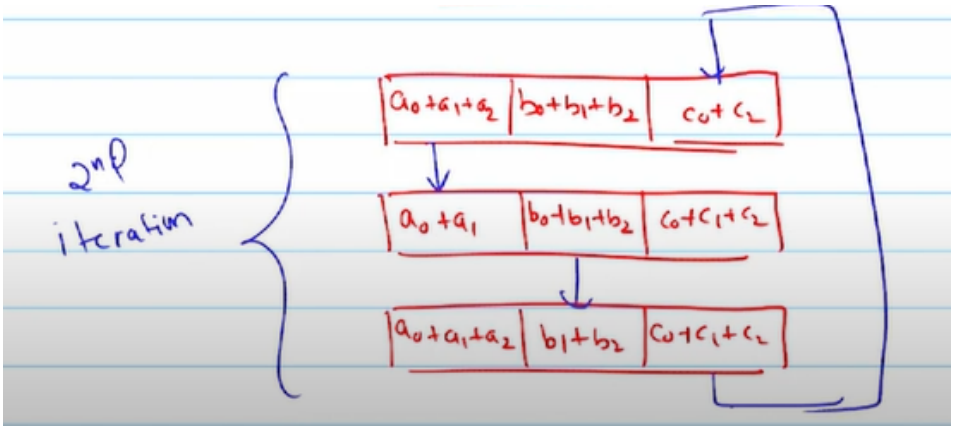
Then you have all gather step. and what happens in all gather. So, let us see what is the state after the scatter reduce step. So, let me sort of redraw it a naught c naught plus c 2. So, this is the information the current state of information after the scatter reduce step. Now, in all gather steps so unlike in scatter reduce step when the incoming information is simply added.

In this case the incoming information is just going to be I mean basically you are going to sort of rewrite your current information with the incoming information. So, what happens is this information is relayed let us say here in the first iteration, this information is relayed here and this information is relayed here. So, this is the first

iteration of let me here let me do it over here I think this is. we want this to be rewritten by the incoming information, we want this block to be rewritten by the incoming information and likewise this to be rewritten by the incoming information.



This is the first iteration of all gather. So, we are now just gathering the information. So, after this, after this particular step what happens? because we want the current information to be sort of written right. So, now what would happen after this let us see I mean I think you will get a better sense once you. So, after this first round what happens? For the first agent you have a naught plus a 1 plus a 2 in place you have b naught plus b 1 plus b 2 in place and c naught plus c 2 still needs to be updated. Likewise, for the second agent you have a naught plus a1 here, the second in a naught plus b1 plus b2 and you anyway have c naught plus c1 plus c2 and for the third agent you have A naught plus a1 plus a2, b1 plus b2 and then you have c naught plus c1 plus c2.



So every agent has gotten two blocks of correct information right after this first round. Then you have the second iteration and in the second iteration what needs to be done? So this needs to go here, this needs to go here and this needs to go here. After this round every agent has gotten the same amount of exact same piece of information. So how many iterations did it take? N minus one iterations here as well. So in total you have two

times N minus one iterations times K by N which is still constant in the number of agents.

So this is the total amount of information exchanged and this is independent of the number of agents. So the communication bandwidth requirement with this ring all reduce algorithm, it's actually independent of the number of agents and it doesn't have a centralized entity either. So, these are certain examples of I mean something I would not still call it distributed optimization or decentralized optimization because you assume that every agent starts with the same initial condition wk or w0. So there is this difference here and in the next lecture we are going to look at something called decentralized SGD which is essentially very similar to the distributed optimization algorithm or the DGD algorithm that we looked at in the previous lecture. So this algorithm is still not fully decentralized because we assume that to start with every agent has the same W0 right and that is usually not the case.

So in order to ensure that as well you would need a consensus kind of step on top of the gradient descent step. So is this clear? N minus 1 right, for both of them it will be n minus 1, so it will be Then followed by? Yeah, so no wait, so you have an outer iteration which is like the kth iteration of the updating the weights. So let's say after the end of k iterations you have the current set of weights wk. Right now your focus is just to be able to compute this sum.

That is what you are trying to get at. You are trying to compute this particular sum and in order to compute that sum you run one round of scatter, basically ring all reduce algorithm. So if there are n agents, so there will be n-1 iterations of scatter reduce and n-1 iterations of all gather. and after that you would be able to compute that sum every agent would be able to compute that sum independently and then they would be able to update the weights independently, but it will be the same weights wk. So, this would basically conclude the k plus 1th iteration of the wk plus 1 algorithm. So, now if you look at it while you have I mean this algorithm is attractive in the sense that the communication bandwidth requirement is much smaller.

The time it takes to train neural network because you have multiple iterations of ring all reduce involved, right? Compared to this parameter server approach, they're definitely larger, right? You have two times and n minus one iterations. which is not the case over here. So that is one difference. I mean one particular advantage of parameter server approach but then again in parameter server, I mean there are other disadvantages like you assume a huge communication bandwidth requirement for the centralized server, right? So that is completely sort of elevated here.

You can, yeah, I mean you can start with... Yeah, so in this case, I mean, the reason we

do that is because, I mean, you know that this is the correct piece of block and you don't want this to be written, right? So it really, I mean, it really sort of depends on how you, I mean, how you have set it up. So the point is how, I mean, as long as you start correctly, you just have to move over by one place and you can keep doing this in a round robin fashion. If there are four agents, then you have to do it like three iterations and you have to do it in a round-robin fashion. Yeah, as long as the start is correct you just keep shifting by one unit to the left same with the scattering like scatter reduce you have started here you then shifted by one unit to the left ok. So, is this algorithm clear? So I will conclude today's lecture with the result on gradient descent in general or stochastic gradient descent and this would be useful in the next lecture when we are going to look at something called decentralized SGT.

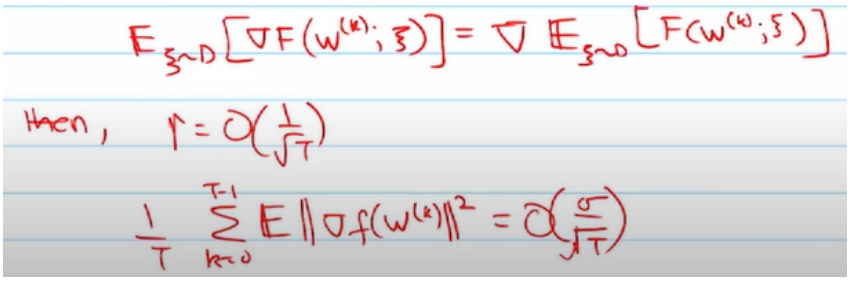
So far we have been in the context of developing algorithms we have been looking at ways we have like particularly in the context of distributed optimization. So what did we look at? We looked at algorithms which are first-order versus second-order. We looked at maybe a fixed-time variant of it. So there are these ways to accelerate the optimization process. But one thing which we haven't looked at so far is the role of the topology.

So, first of all we know that the graph diameter is going to play an important role because it tells you how quickly you are going to reach the consensus. So, the role of the topology is also important in terms of dictating how quickly you can converge. So, if we were to choose the underlying graph topology. what is an optimal choice of graph topology. So it is certainly not a line graph because it takes a lot of I mean if the graph diameter is going to be d n minus 1 and it would take that much amount of steps for the information to propagate from one end to another.

It cannot be star graph because star graph while the step propagation information propagation happens in two steps from any node to any other node. the communication bandwidth requirement on the star node, that is going to be significantly large. It's N minus one. So if you want to have a nice trade-off between communication requirement as well as how quickly the information flow should happen from one node to another, So if you want to have this nice trade-off, what is the optimal topology to work with? While this still remains an open question, but there exists a topology that beats all common topologies that we know and that is called static exponential graphs. So this is something that we are going to look at in the subsequent lectures, but I will end today's lecture with the simple result on gradient descent or stochastic gradient descent.

So under the assumption that the loss function using capital F here is L-smooth in terms of the in terms of w and the second assumption is because we are talking about stochastic gradient descent. So, the stochastic gradient. So, when we use a few data points to get an

estimate of the gradient we call it a stochastic gradient right because it is not a true estimate of the gradient it is just a so is unbiased. it is an unbiased estimate and has bounded variance. So, you choose a different set of points you will get one variance right like of the gradient you choose another set of points you will get another variance and so on.



So, this has bounded variance sigma square. So, what do we mean by unbiased estimate? So, unbiased estimator is like let us say if I am computing. if I want to compute the expectation of the gradient, expectation with respect to how the data points are distributed or not distributed, but let us say data points that are sampled. So, I mean this is still talking about centralized case. So, this is same as the gradient of So, then we say it is an unbiased estimator. So, the result says then if these two assumptions are satisfied then and if you use a step size of the order let us say 1 over square root of T then you have 1 over T summation is running some of the running average of the gradient knobs.

So, that is going to be order sigma over square root of T. So, as t becomes larger and larger eventually you mean you have order 1 over square root of T kind of convergence, but it also depends on the variance data variance. So, in the number of iterations it is going to require if the variance is large then obviously you are going to require larger and larger number of iterations. So, this is by the way what is this little f is this particular quantity. So this is your f, the average of this is your f. So this is the result on stochastic gradient descent and in the next lecture we are going to look at decentralized gradient descent or decentralized stochastic gradient descent where results like these would be useful in at least I mean we are not going to look at the proof of why static exponential graphs are better or the worse, but then we are going to make use of these results. to basically analyze these algorithms.

So, any questions on today's lecture? So, when does stochastic gradient descent works? So, we assume that the loss function is L-smooth. I mean first of all when we talk about gradient descent or any algorithm for that matter, a either if you want to provide global guarantees and you would have to assume that the function is convex. if the function is non-convex then the guarantees are only going to be local. So, there are no guarantees of convergence to global minima.

So, I have not mentioned that f is convex or not convex. So, I mean even if it is non-convex the convergence guarantee that we are going to have is essentially local convergence guarantee. So, we assume that the loss function is L-smooth in terms of the weights w. Like let's say you choose a mean square loss or cross entropy loss. So whatever loss you end up choosing, it's going to be L-smooth in the weights of the neural network.

The other result is stochastic gradient is unbiased. So this is your stochastic gradient and it's an unbiased estimator. So I can, expected value of the gradient you can write it in terms of the gradient of the expected value and the expected value of the function is what I call as little f. So if the stochastic gradient is unbiased and the variance is bounded by some sigma square, then this is the convergence kind of rate that we get. And as number of iterations capital T becomes very large. that is when you have I mean this expectation basically starts the average of this expectation starts becoming almost close to 0, but it scales it also scales with the number with the variance or with the standard deviation right.

So, the variance is very large I mean it can still be bounded, but if it is very large then it would also require larger number of iterations. So, these are the number of like I mean this is the So, when what do we mean by number of iterations required? So, essentially when if you want to say the number of I mean if you want to get epsilon close then number of iterations required are essentially sigma over square root of t is essentially going to be your epsilon right. This is for a centralized case. This is for a centralized case.

There is no distributed or decentralized part as yet. In the next lecture we will look at the how the role of the topology plays a role I mean the topology plays a role or In fact, the graph diameters and certain parameters would also be important in that case. So, that is what we are going to look at. So, much of today's lecture was largely about large scale machine learning and this basically gives you an idea. I mean that is where these stochastic gradient descent are mostly used. So the algorithms, the parameter server approach or the ring all reduce, they are relevant mostly in the context of large-scale machine learning.

If you have very few parameters then you won't even need this kind of fancy information exchange and if you have like multiple servers and certain bandwidth requirement on those. But then this stochastic gradient descent is true for any setup. I mean right now we are not looking at particularly a decentralized setup in this case. and that would be the topic for the next subsequent lectures.