Distributed Optimization and Machine Learning

Prof. Mayank Baranwal

Systems and Control Engineering

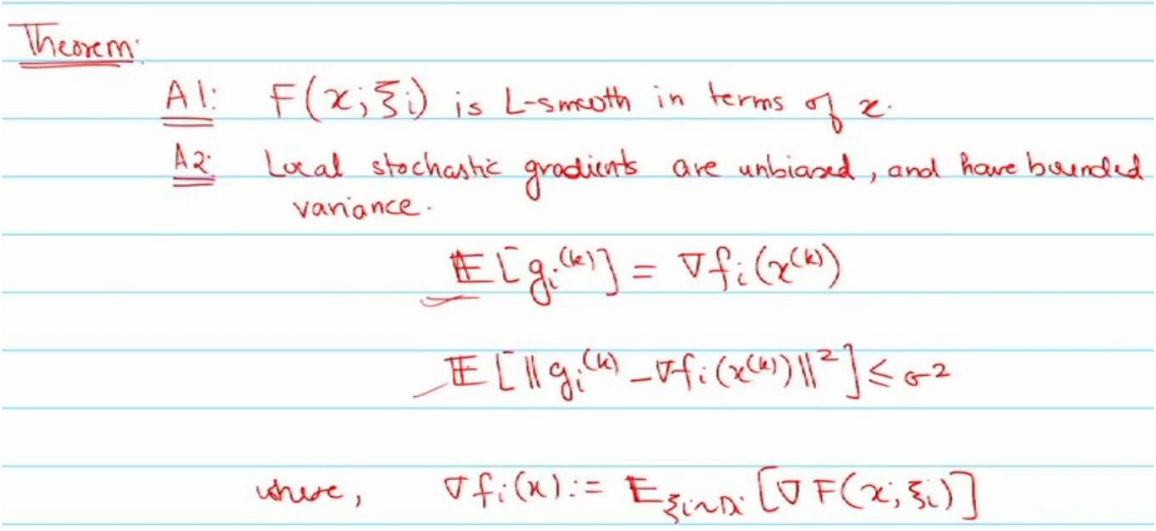
Indian Institute of Technology Bombay

Week-11

Lecture 41: Decentralized Stochastic Gradient Descent -2

So, let us write down a theorem. this is again going to be similar to the decent the SGD theorem that we had looked at, but now it is applied to decentralize SGD. So, we are not going to prove this theorem, but we are just going to look at the consequences of this theorem. So, first of all let me write down the key assumptions which are going to be very similar to the ones that we looked at in the context of simple SGD. So, we assume that the function the loss function or the objective function that you are trying to minimize it is going to be a function of x as well as the data points. So, this is l smooth in terms of x that is assumption 1.

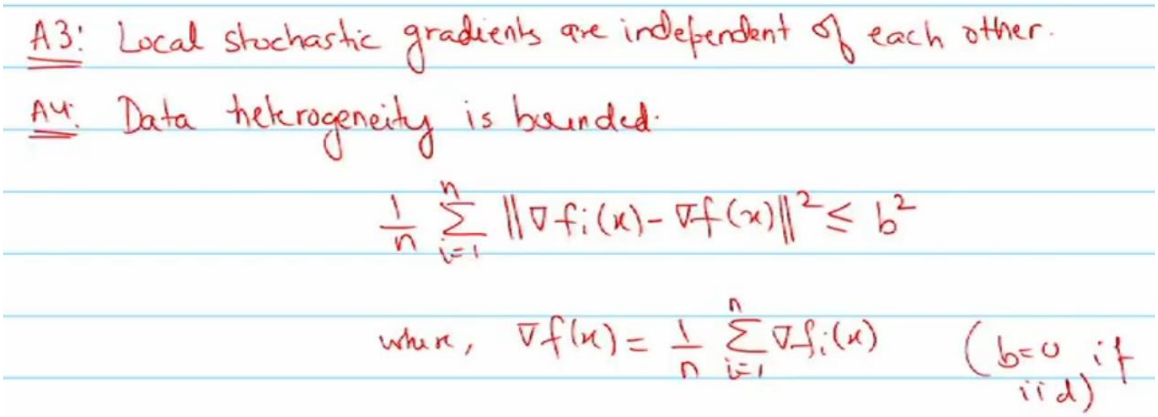
Assumption 2 is local stochastic gradients are unbiased and have bounded variance. So, meaning, so if the gradient is unbiased, so which is saying that expected value of g is simply the and for bounded variance we have g i k minus where this particular quantity gradient f i of x is defined as the expected value of the gradient of the. So, if you have data points So, you have an unbiased estimator. So, those stochastic gradients are basically unbiased and the gradients essentially they have bounded variance.



So, this is assumption 2. Assumption 3 is that local stochastic gradients are independent

of each other. So, for agent i, it is independent of agent j and so on. So, local stochastic gradients are independent of each other. And finally, so data heterogeneity is bounded.

So, essentially every agent has its own private data and the data distribution that agent i has may be different from the data distribution that agent j has right. So, the heterogeneity in the data between agents that is again bounded. So, these are sort of minimalistic assumptions also practical enough, but what we are saying is that data heterogeneity is bounded. So, that means 1 over n make gradient where ok. So, this is the foundedness on the data heterogeneity.

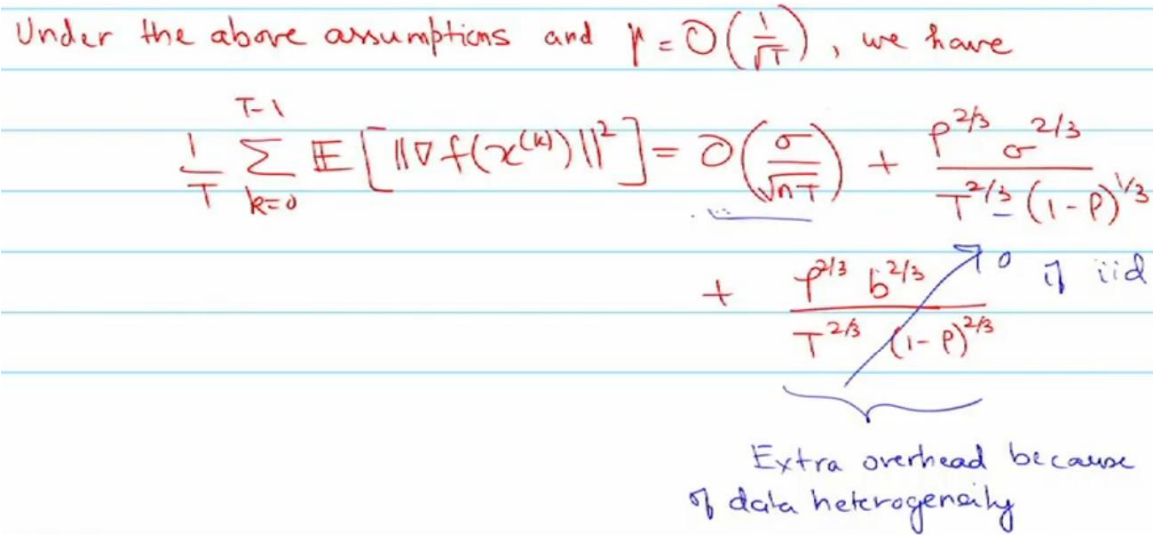


So, essentially if you have iid samples, if data is iid then b is going to be 0 essentially, for non-IoD when the data is heterogeneous then you are you are going to have some non-zero. So, b is equal to 0 if you have IoD. So, then under the above assumptions and assuming your learning rate gamma. So, this particular this particular learning rate or the step size gamma is order square root 1 over t we have. So, the expected value essentially similar kind of result that we had in the context of n node training and single node training and so on.

So, first of all this is a n node training. So, you would assume that the this particular error for this particular quantity will at least be this much right. So, if I look at the beginning of this lecture. So, we had this particular result right when you have n node parallel training this quantity is order sigma over square root of n t right. So, in this case you are going to have at least this much error plus the error because you had data heterogeneity and you had no global synchronization right.

So, these terms would also contribute. So, this particular term is going to be rho where rho is your spectral gap rho 2 third sigma 2 third plus So, this particular term is your extra overhead because of data heterogeneity, you have the B term appearing it right, extra overhead because of data heterogeneity. And this particular term is there because we do not have a centralized setting, we have a decentralized setting right. So, we need some iterations because of, we need few iterations just to make sure that there is also

synchronization between agents estimates. Had it been a centralized training, we know that for n node parallel training this is the, this is when if you want to get epsilon close.



you have to have like sigma square over t square number of iterations, sigma square over t times epsilon square number of iterations. But in this case you need more iterations because you need synchronization as well as you have data heterogeneity. So, this term is going to be 0 if you have iid setting right, if there is no data heterogeneity then you have this term is going to be 0. but this particular term is still going to be there by decentralized SGD and it is entirely related to not having access to a globally synchronized estimate x. So, let us say we look at this example of MNIST one.

So let's say we are trying to classify images of cats, dogs and tigers. And let's say two of us are the two agents or two workers. I have all the cats images and you have the images of dogs and tigers. So the kind of data distribution that you have is entirely different from the kind of data distribution that I have. So it's heterogeneous data.

Had it been the case that you have one, let's say half of the cat images, half of the dog images and half of the tiger images. And likewise, if that had been the case with me as well, then our data is kind of homogeneous. But in like if you have entirely like I mean this would be the extreme case that I have all the cat images and you have all the dogs and tiger images to work with. So, we would end up training very different neural networks had it not been the case that right. So, you would train a neural network that would be very different from the neural network that I end up training.

But we are because we are also exchanging information on the current weights. We are actually trying to train a neural network that works for all kinds of images and not just the images that you have or the images that I have right. So, that is where the data heterogeneity role part is important. What? Right. So, this is the definition exactly this is

the definition of the heterogeneity being bounded.

So, it is like. So, if you look at the average of all the gradients. So, this gradient of f of x is basically the average of all the gradients. So, this has a bounded variance. So, that is the mathematical definition of when I say data heterogeneity is bounded. Yeah, b is a bound just like we are using sigma as a bound for the bounded variance.

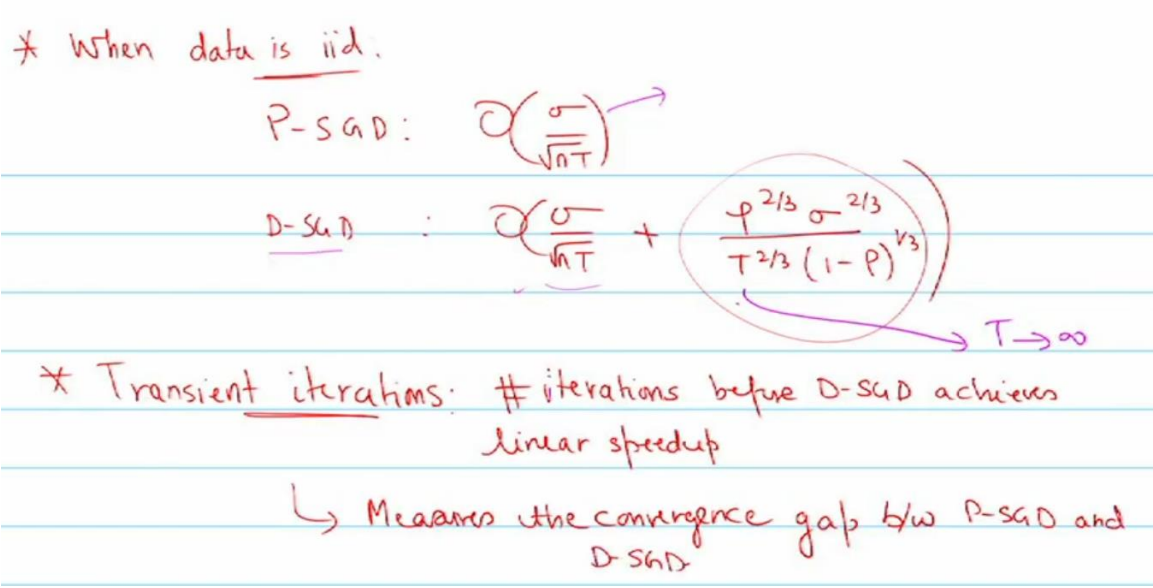
So, for the bounded variance let me just like we are using sigma for each agent this is across agent the how the gradients are going to be going to be having a bounded variance. So, let us say consider this setting when the data is iid. So, when data is iid that means b is equal to 0. So, in that case what you would have for parallel SDD if you want to get epsilon close like for getting an epsilon accurate, like let us say, so this was the error was in this order sigma over square root of nt, right. For decentralized SGD, this error is sigma over square root of nt plus this term, second term here, right, which is sigma, so rho 2 third, sigma 2 third, this particular term.

So, we have this extra iteration that we need to perform in order to make sure that the error is in the same range for the two algorithms right. So, if we want to get epsilon close then we would have to perform these extra iterations and that is what we call as transient iterations. So, let me write this down transient iterations So this is the number of iterations. So what happens when, in this particular case, what happens as t goes to infinity? So this particular term compared to this term, so you know this is 1 over square root of t, right? And this is going to dominate this particular term as t starts becoming very large. So as t goes to infinity, then you would have parallel SGD and decentralized SGD kind of matching up.

but then for fixed t for a finite t you would have I mean you would not have that case right. So, the number of, so essentially you would get a linear speedup. So, we know that parallel SGD has linear speedup right which is to say that, so we just wrote this here. So, parallel SGD has linear speed up because the number of iterations required to get epsilon close is like 1 over n essentially if you increase the number of servers essentially you would have I mean you would have the linear speed up in the. So, it is the same applies here as well you would have linear speed up only asymptotically right with the decentralized SGD.

So, the number of iterations number of iterations before decentralized std achieves linear speedup is basically your transient iterations. So, at least these many transient iterations have to be encountered before you start seeing the effect of this linear speedup. right and it basically measures the. So, transient alterations are used to measure the gap between the parallel SGD and the decentralized SGD. So, it measures the convergence

gap, measures the convergence gap between parallel SGD and decentralized SGD.



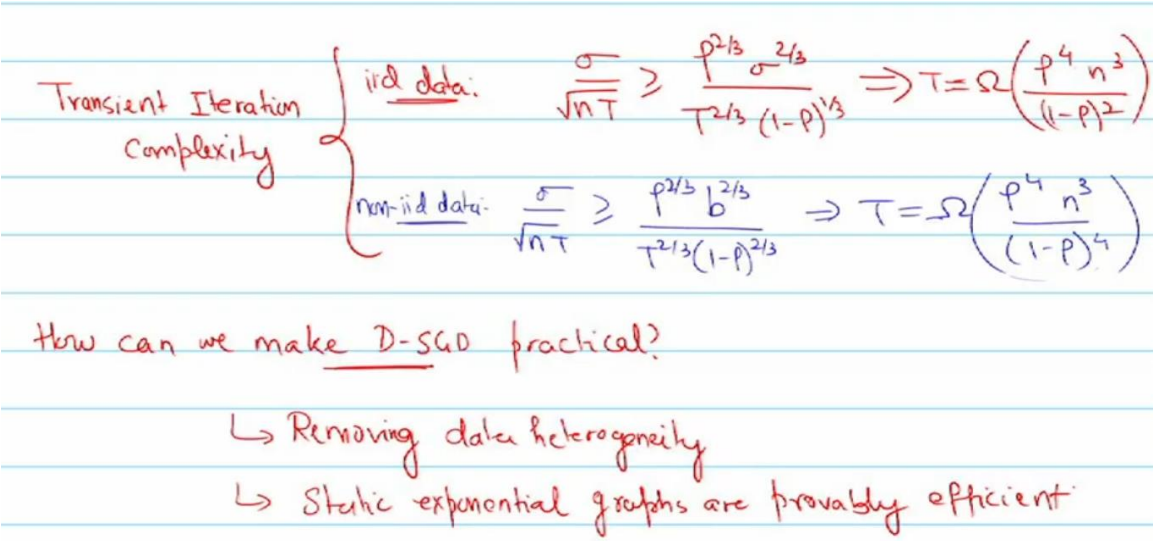
So, what is the transient iteration complexity? So, when you have iid data, so transient iterations would how do we how would we equate transient iterations? We want to get in this regime when this term starts dominating this particular term right. So if you want to get that transient iteration for id data that means sigma over square root of n t this should be greater than or equal to the other term which is rho 2 third sigma 2 third t 2 third and 1 minus rho 1 third. So this basically gives you a transient iteration complexity of order rho to the 4 n cube. upon 1 minus rho square. So, after these many iterations is when you can start seeing the effect of linear speedup due to decentralized STD ok.

So, this is just by just this is obtained by just equating these two terms. What is the case when you have non-ID data? How would we compute this? So, what is the constraint when we have non-ID data. So, when we have non-ID data then this term is also there, but if you look at a value of rho, I mean we want a certain point actually this term we put I mean in terms of capital T both these terms are they scale like T 2 third right. if you look at the value of rho. So, this scales like 1 and rho is a number between 0 and 1.

So, basically you have to compare the these two terms, because this this term is essentially going to be much smaller compared to this, this whole term is going to be larger compared to this particular term. So, we compare these two terms here, because it is 1 minus rho 2 third versus 1 minus rho 1 third. So, that means I need to ensure that this particular term is greater than equal to rho 2 third b. And this gives you a transient iteration complexity of order rho 4 n cube 1 minus rho So, when you have data heterogeneity you see it is 1 minus rho raised to the power 4 and not 1 minus rho raised to like power square 2 right. So, essentially saying that so this number because rho is a number between 0 and 1.

So, 1 minus rho raised to the power 4. is going to be much smaller compared to 1 minus rho square. And if it comes in the denominator that means, it would have larger iteration complexity and it would make sense right because if you have a very different data distribution than what I have, then it would require even more iterations to start seeing the effect of linear speed. So, how can we make the decentralized SGD practical? So, what do we need to do like based on what you have seen so far? One is when that I mean when it is under our control let us say we are the one who are distributing the data across agents. So, then we have to make sure the data is IID right or close to IID.

So, that is I mean that may not always be the case let us say you have your own private data to start with then it I mean we have no control, but when you have control on the data distribution then we have to ensure that removing data heterogeneity. So, that is one way to make decentralized SGD practical. The other thing is we want to optimize with respect to row here right and it turns out and when we say this when we said that we want to find the optimal topology. So, row depends on the topology that you are going to be working with and when we said that we want to find out the. So, there are two things that we are trying to optimize.



one is that per iteration complexity essentially when I say per iteration complexity you do not want to have a very large degree right. So, we want to optimize that and we want to optimize this transient iteration complexity which is basically want to ensure that the decentralized SGD is practical that is after like fewer number of iterations you start seeing the effect of this linear speedup right. So, essentially we have to optimize this row value and it turns out that the static x for something like. So, it is a very recent paper which was a static exponential graphs are probably efficient Well, this may not be the optimal topology topology per se, but this is optimal when you compare it against all popularly known topologies like ring graph, star graph, grid graph, complete graph and

line graph and so on. So, among all the commonly known topologies, something like static exponential graphs are probably efficient.

So, what are static exponential graphs? So, let us say you have 6 nodes. So first of all every node will have, so it's a directed graph. So static exponential graph, it's a directed graph. And the way you are going to place an edge is, so from node 1 to 2, you are going to place an edge. So essentially from the perspective of node 1, every 2 to the 0th node, every 2 to the first node, 2 to the second node, you are going to be placing an edge.

So from node 1, so 2 to the 0 is 1. So node 2 is essentially 1 unit apart. So you are going to place an edge. Likewise node 3 is 2 units apart and node 5 is 4 units apart. So this is when you place the edge and you do this in a round robin kind of fashion. So from node 2 you would be placing an edge here.

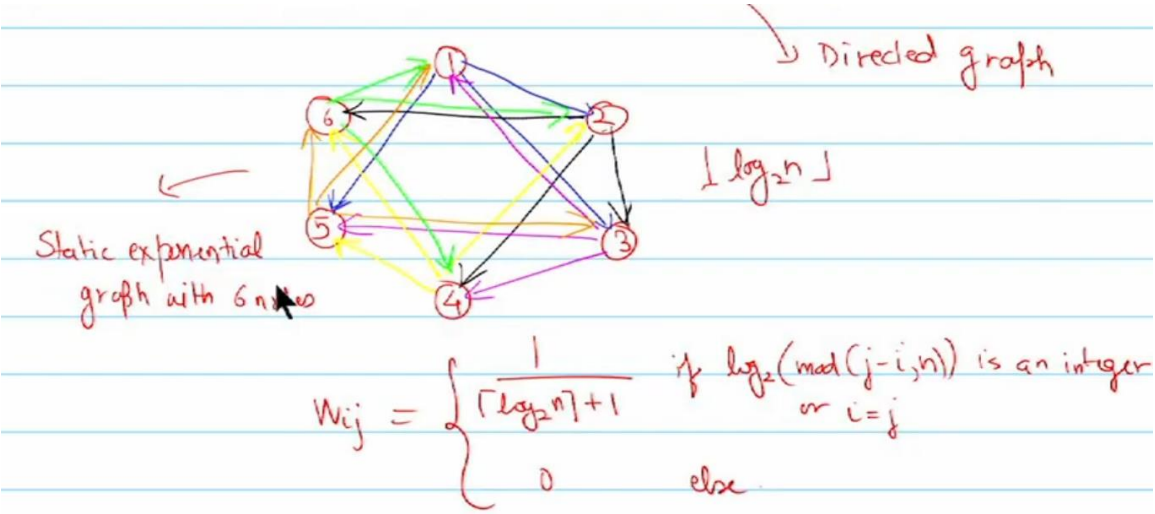
here and here. From node 3 you would be placing an edge from 3 to 4, 3 to 5 and where? Essentially you have to do it in a modular fashion. So, 3 to 1 as well. So, you are going to be placing these edges in a modular fashion. So, let us use different colors. So, from 4 to 5, 4 to 6 and 4 to 2, likewise from 5 to 6, 5 to 1 and 5 to 3 and again from 6 to 1.

6 to 2 and 6 to 4 ok. What is the largest degree of this particular in this particular graph? Log n base 2 right that is the degree you can take the let us say that is the largest degree because or rather seal function of it. So, every node is going to be connected at most to log n number of nodes. to the zeroth node, then to the first node and to the second node and so on. So that is the degree of this. But iteration communication cost is going to scale like order log n.

Like we won't be communicating with more than log n number of and we know that compared to n, log n is a much smaller like at least when the number of agents is large, log n is a much smaller number right. So, you can also show the even though this is a directed graph, this is also connected graph. So, every you can reach from one node to every other node. So, there is information flow between all the between all the agents or between all the workers that is another thing. And the weights w ij, the way these weights are defined is 1 over plus 1.

f is an integer or i equal to j and its 0 else. this is going to be a that is why I said this is going to be directed graph as you can also see from here. So, 1 is unable like 1 can exchange information with 2, but 2 cannot exchange information with 1 as you can see from this particular graph right. So, this is your static exponential graph with 6 nodes. and the edge weights are going to be following this particular formula. So, if you have these static exponential graphs, then you can show that this is I mean the result that you have

are probably better than most common topology.



So, let me share that result. So, any questions on this, how the static exponential graph is constructed? So, let me share the comparison between static exponential graphs as opposed to the common graph topologies like ring graph, grid graph, star graphs and so on. So, if you look at the per iteration communication cost and the transient complexity, you can see for I mean when you see omega tilde it is up to order log n right. So, we know that the per iteration communication cost is order log n, but if you look at the transient complexity, it is better than all other common topologies right. So, this is where like it sort of enjoys a sweet spot between minimizing the transient complexity as well as also not having a very large communication cost. So, we know that ring graphs and line graphs these are these have ring graph like every agent exchanges information with two neighboring agents right.

So, for iteration communication cost is going to be constant in the number then if you look at the transient complexity for ring graphs it turns out its order n to the 7. and something that is not scalable as increase in number of agents. Whereas if you look at this one, so this is order n cube, in fact this is n cube log n which is slightly worse than like that half random graph which is order n cube. But then if you look at the per iteration communication cost for this particular graph, this is order n. Whereas, you for static exponential graph this is order log n.

So, that is particularly attractive about this. So, this pretty much sums up the discussion that we like at least on distributed optimization. There is one more interesting thing that I wanted to show and this is on. So every agent is trying to minimize this cooperative goal. But what if there is a byzantine adversary in the network, meaning that particular adversary tries to maximize his or her own private objective function and doesn't care about what everyone else is doing.

Per-iter comm. and tran. iter. of static exp are nearly best (up to $\log_2(n)$)

Topology	Per-iter. Comm.	Trans. Iters. (iid scenario)
Ring	$\Omega(2)$	$\Omega(n^7)$
Star	$\Omega(n)$	$\Omega(n^7)$
2D-Grid	$\Omega(4)$	$\Omega(n^5 \log_2^2(n))$
2D-Torus	$\Omega(4)$	$\Omega(n^5)$
$\frac{1}{2}$-RandGraph	$\Omega(\frac{n}{2})$	$\Omega(n^3)$
Static Exp	$\tilde{\Omega}(1)$	$\tilde{\Omega}(n^3)$

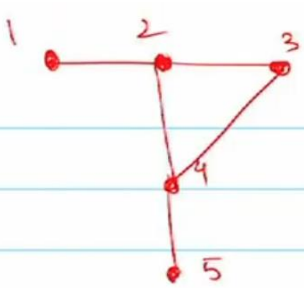
So in that case they will be exchange. So essentially the information that they will be exchanging is right in the sense that they would be exchanging their current weights or current estimate, but they would not be using a consensus to just like other agents. And in that case they can actually basically end up optimizing their own objective and they would not care about the objective of the team objective right. So, let us look at that particular example. So, the last topic on in the context of distributed optimization consensus and things like that is going to be on. So, essentially you want to have a resilient distributed optimization.

in presence of Byzantine adversary. So, as I said what is Byzantine adversary? So, they try to maximize or minimize their own objective and they do not care about the social cooperative goal, but they do not they do not share incorrect readings or incorrect estimates. So, they share the correct estimates, but they do not like locally they are not using the policy that everyone else is using. So, if you look at the same example that we looked at when we looked at the code or basically the implementation of DGD and second order accelerated aggregated gradient method. So, we looked at a graph like this. So, this was node 1, 2, 3, 4, 5 and fi of x was defined as half x minus i whole square and so gradient of fi of x minus i.

So, we are now going to assume that agent 5, so by the way what is the optimal value of this summation fi, this is your cooperative goal right. So, this is equal to 3 right when the gradients are 0. So, essentially sum of the gradient is 0 essentially 5 x minus 5 x basically average of 1 2 3 4 and 5 which is 3 in this case. So, this is your global objective value ok.

Now, we assume that agent 5 is by agent n. So, what does it do? It just runs this. So, x5 k plus 1 is going to be x5 k minus step size times gradient of. So, that is all it is going to do. So, it just cares about minimizing its own objective and it does not care about what everyone else is doing in the network. While everyone else is in the network is essentially

running this DDD algorithm.



$$f_i(x) := \frac{1}{2}(x-i)^2$$

$$\nabla f_i(x) = (x-i)$$

$$\min_x \sum f_i(x) = 3$$

Agent 5 is byzantine.

$$x_5(k+1) = x_5(k) - \eta \nabla f_5(x_5(k))$$

Every other agent in the network runs the DGD algorithm:

So, every other agent in the network runs the DDD algorithm. So, agent i k plus half is going to be summation j 1 through 5 wij xjk and xik plus 1 is going to be xik plus half minus step size times. gradient of f i evaluated at x i k plus half ok. So, this is your DTD algorithm. So, because agent 5 is communicating an information based on x 5. So, you can see that you may potentially not even converge to the global objective value right and in fact you would not because agent 5 is communicating some other information, it is not running the same protocol, this is agent 5 and this is only for i in 1 through 3, 4 right.

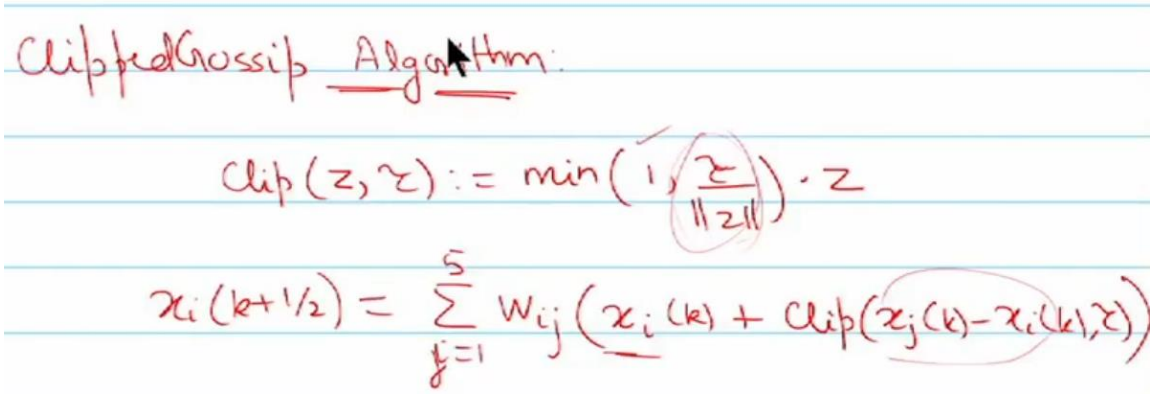
Every other agent in the network runs the DGD algorithm:

$$i \in \{1,2,3,4\} \begin{cases} x_i(k+1/2) = \sum_{j=1}^{5} W_{ij}\, x_j(k) \\ x_i(k+1) = x_i(k+1/2) - \eta \nabla f_i\left(x_i(k+1/2)\right) \end{cases}$$

So, if you have a byzantine adversary like this, who is just looking to maximize or minimize their own objective, then how do we account for such byzantine adversary. So, let us, so for this instead of using this normal consensus algorithm, we use something clipped gossip algorithm. So, in clipped gossip, so let us first define the clip function. So, you have let us say a vector z and a constant tau. So, the clip function is defined as minimum of 1 comma tau over norm of z times z.

this is the clip function. So, agent i what it would do is it would run this x i k plus half is going to be summation j 1 through 5 w i j times x i k plus clip of x j k minus x i k comma So, if this clip value is 1, then you essentially recover the same gossip algorithm or the

consensus algorithm right, if the clip value is 1. But what we are saying is that, so by using a small value of tau. what we are saying is that I am going to rely to like on my difference a lot on my estimate a lot and I am going to play pay a smaller attention to your like the difference from the difference of my estimate to the difference of your estimate essentially this difference x i x j minus x i. So, for the other agents that are cooperating this difference is essentially going to be somewhat smaller value right. So, then in that case you would have one to be this to be activated, but if this difference is going to be large then you would see that this difference this quantity becomes activated.



And this is going to be the case for the byzantine adversary right that this particular value is going to be activated and that is how you are going to be robust to what they are essentially proposing. So, let us look at the implementation of this script gossip on the same example. this particular example and see how having a clipped gossip actually helps with. So, no one knows who the bygentine adversary is by the way right, no one knows who the bygentine adversary is, they are just exchanging information with the neighbors and they are running this clipped gossip algorithm that is it. So, depending on the value difference I mean this value this algorithm or this gossip algorithm would either take 1 or tau z norm z times this.

No, agent 5 won't. Agent 5 is just running this. Because agent 5 wants to optimize his or her own objective. So agent 5 doesn't care about what everyone else is doing. So even though everyone is supposed to work towards a social or a cooperative goal, There is a byzantine adversary who just wants to optimize their own objective. They do not care about what everyone else is doing. But they are truthful in the sense when they exchange information with their neighbors, they are not exchanging corrupted information.

They are exchanging their own current x5 value or in this case x5. They are not exchanging the corrupted information. So is the setup clear to everyone? So let us look at the implementation. So again this clip gossip paper is very recent, but it is an interesting paper that as I said most of the content that you are learning in this course it has been developed in the last 5 to 7 years. So, if you understand most of the concepts fine then you should be really proud of your learning.

So, you expect the difference to be relatively larger. So, this is the same example, exactly same example that we looked at last time. So, again just import unnecessary libraries. So, we are trying to minimize the sum of these convex functions f i. and we know that the gradient is x minus i. So, number of iterations we for now we keep it to 10000 step size is 10 to the negative 3 and this is where we define the weights essentially.

So, this was the algorithm that we looked at in the last class which was or not the last class, but when we had this implementation. So, the only difference now is. So, agents 1 through So, we are saying all values of x up to last agent and like excluding the last agent they essentially run this same consensus step. So, they run the same consensus step and then they have the gradient update on the updated z value and the last agent simply runs this ok.

The last agent is a greedy agent or a byzantine adversary and simply runs this. So, let us see what this algorithm converges to. So, if I run, you can see that every agent has converged to 5, close to 5, whereas the social objective was or the global objective was 3. So, just a single bygentile adversary has managed to like basically throw off all other agents and the consensus is happening really because agent 5 is not going to be, in this case what is happening is the agent 5 is always going to be converging to 5. And because you also have consensus on top of it, then everyone is forced to converge to value 5. So, this is what happens when one of when you have a sort of a greedy kind of adversary in the network, who just like looks to maximize or minimize their own objective depending on its.

It is like a leader follower. So, you can think of agent 5 as the leader. and everyone is a follower becomes a follower. So, now, so we define this clip function. So, I wrote written this my clip function.

The default value of tau is 0.1, but you can try a smaller value as well. So, I have just added a smaller small check because I am dividing it by the norm of z. So, if z is going to be 0 or norm of z is going to be 0 then you essentially return 0 right. So, essentially for this to be well defined I am have this if else condition, but really this is what we are returning which is the clip function that we just looked at right. So, and now in I have changed the gossip algorithm now it is a clip gossip.

So, w ij times x i plus my clip of x j minus x i let us start with tau equal to 0.1 ok. and if I let me run this algorithm and let us see what it converges to yeah. So, you can see that still everyone is still has converged to point that phi value even though you expect them to converge at least perform slightly differently and that is because we have used a

relatively large value of tau. So, if we really want to make sure that every like the agents they are able to reject the adversarial agents behavior or account for adversarial agent behavior. So, you can actually run with the smaller tau and once I do that and run this again you can see every other agent has now converged to 3 and agent 5 is anyway it is supposed to converge to 5. So, even in presence of adversary you can still try to minimize the social or the cooperative using the script causal.