Distributed Optimization and Machine Learning

Prof. Mayank Baranwal

Computer Science & Engineering, Electrical Engineering, Mathematics

Indian Institute of Technology Bombay

Week-12

Lecture - 47: Generate Update Rule

In total, if I try to summarize, so even here right like when I looked at this particular algorithm, I can also view this algorithm as x t plus 1 as a function of Xt right some function of Xt plus some weighted gradient some like essentially central server even though if I look at this particular diagram here right. So every agent or every client makes multiple local updates. So let's say every client makes just one local update. Then this problem is same as parameter server approach. is same as parameter server approach where xt plus 1 is xt minus step size times the sum of all the gradient for the average of all the gradients and this is this is how the xt plus 1 is updated right. Now because you are updating because I am updating this multiple times I can still view this.

So, instead of sending the gradient for one on the entire batch client would be sending some I mean basically an accumulated gradients to the central server right. Every client would be because you are accumulating the gradient every time you are performing the local updates. So if I were to write xt j plus xt tau tau i as a function of xt something that we have already done here. So this is nothing but it is essentially accumulating the gradients with certain weights and that information is being sent to the central server.

the general update rule let me write this general update rule at the server side is xt plus 1 let me also call it xt plus 1 0 because that is the value that is communicated to or that is the parameter that is communicated to all the clients right. So, that 0 is what it indicates it is essentially means that this is the as I said over here right like we have this x t 1 0 this is nothing but x t. So, they really think of it as x t plus 1 at the server side we are calling it x t plus 1 0. So, that it is also the 0th I mean without any update without any local update this is the value that is received at each client right. So, you can show that all the fed averaging algorithm can be written as a generalized update rule like this tau effective times summation i 1 through m weights wi minus step size eta times some update direction dt.

I will tell you what dt is, but this is what it looks like. So, this is more or less like if let us say every agent performs just a single update, then we know that x t plus 1 0 is x t 0

General Update Rule

$$x_{t+1,0} = x_{t,0} + \tau_{eff} \sum_{i=1}^{m} w_i \left(-\eta\, d_t^{(i)}\right)$$

minus eta times the sum of the gradients. So, let us assume every agent performs just one local update ok. So, then what is the information that that the central server is receiving? So, it would be doing xt plus 10. xt0 minus eta times i 1 through m wi and you have the summation of basically weighted summation of gradients.

$$\overset{\checkmark}{x_{t+1,0}} = x_{t,0} - \eta \sum_{i=1}^{m} w_i\, g(x_{t,0}; \xi_i)$$

So, g x t 0 on the data set of the first i th agent right this is what it would be doing. And if I look at this particular update rule and this particular update rule they are kind of similar in the sense that tau effective in this case turns out to be 1. So, that is the effective number of local updates, everyone is performing just one local update. So, tau effective turns out to be 1 and dt turns out to be the gradient ok. But what happens if let us say agent 1 runs tau 1 number of updates agent 2 runs tau 2 number of updates, you would have certain expression for tau effective, and you would have a certain expression for dt as well.

So, dt need not be need not directly be the gradient, but it may be the sum of the gradient and so on ok. How do we say that? Let us say agent agent 1 or agent i performs two local updates. So, in that case assume agent i performs two local updates. So that means, we will have xt 1 i is going to be xt 0 i minus step size times gradient So, that is one update and the second update is xt 2 i is xt 1 i minus, is this clear? So, I can write xt 1 in terms of xt naught. So, let us write that.

$$x_{t,1}^{(i)} = x_{t,0}^{(i)} - \eta\, g(x_{t,0}^{(i)}; \xi_0)$$

$$x_{t,2}^{(i)} = x_{t,1}^{(i)} - \eta\, \underline{g}\,(x_{t,1}^{(i)}; \xi_1)$$

$$= x_{t,0}^{(i)} - \eta\, g(x_{t,0}^{(i)}; \xi_0) - \eta\, g(x_{t,1}^{(i)}; \xi_1)$$

$$x_{t,2}^{(i)} = x_{t,0}^{(i)} - \eta \underbrace{\sum_{j=0}^{1} g(x_{t,j}^{(i)}; \xi_j)}_{d}$$

So, this is x t naught i minus eta g x t and then you have this additional term and this would be x t. Is this clear? So, if I were to write x t plus 2 i, comma 2 i, it is nothing but x t 0 i minus eta times summation, let us say j equals 1 through in this case it is just 2. So, it is 1 through 0 to 1. Is this clear? So, you can effectively think of this as your dt. The information that is, so essentially the sum, basically the sum of accumulation of gradient in each local update is what your dti looks like ok.

Because this is the information after two updates, this is the information that is being communicated by the client to the central server right. So, this is the information that is being communicated you essentially this is what is being communicated. So, in a sense if I were to write this, so this is x t tau sub i is going to be x t 0 i minus step size eta times j equals 0 to tau i minus 1 g x t j i z ok. So, this is after tau a updates alright is this clear. So, if I accumulate all of this. So, this is just from the ith agent right. So, if I look at the

$$x^{(i)}_{t,\tau_i} = x^{(i)}_{t,0} - \eta\left(\sum_{j=0}^{\tau_i-1} g(x^{(i)}_{t,j};\xi_j)\right)$$

server side x t plus 1 0 at the server side xt plus 1 0 is essentially going to be summation pi i from 1 through m xt tau sub i ok with summation pi equal to 1. This is the server-side update and if I look at it like this i 1 through m p sub i and I just write it in terms of x t 0 i which is by the way same which is same as xt0 because for every client at the beginning of the communication round they would have received the same value. So, let me simply write this as x t 0 rather minus eta times summation j equals 0 through tau i minus 1 j x t ok. So, I can this implies that I can write xt plus 1 0 as xt 0 minus summation i 1 through m pi or let us say plus and then you have minus 3rd times in this particular term So, if I compare this particular update with my general update rule like this, that means I am using tau effective to be equal to 1, wi is same as pi and dti is this accumulated gradient.

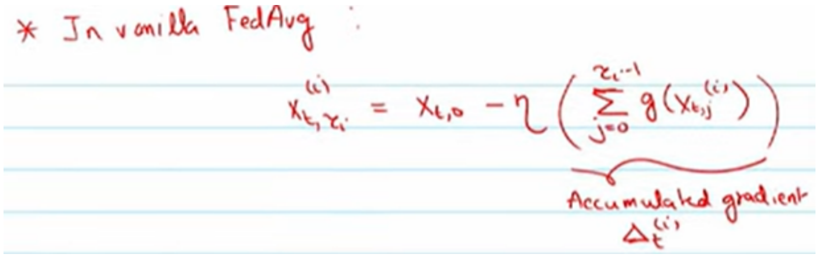
$$x_{t+1,0} = \sum_{i=1}^{m} p_i\, x^{(i)}_{t,\tau} \qquad \text{with } \sum p_i = 1$$

$$= \sum_{i=1}^{m} p_i \left(x_{t,0} - \eta\left(\sum_{j=0}^{\tau_i-1} g(x^{(i)}_{t,j};\xi_j)\right)\right)$$

$$\Rightarrow \left|\, x_{t+1,0} = x_{t,0} + \sum_{i=1}^{m} p_i\left(-\eta\left(\sum_{j=0}^{\tau_i-1} g(x^{(i)}_{t,j};\xi_j)\right)\right)\right.$$

So, fed averaging can be fit into this particular general update rule. Is this clear? So, Fed averaging can be thought of. So, the reason why we came up with this general update rule is because in fact there is this NeurIPS 2020 paper which proposes algorithm called FedNova. and they showed that there are other algorithms like fedprox fedaverage fednova all of this can be represented at as a general update rule like this with different value of let's say tau effective with different value of dti with different value of wis. So, all of this I mean wi can be a function of pi in this case they turned out to be exactly same as pi, but depending on different algorithm as I said FedNova, FedProx all of this can be represented like a family of these federated learning algorithms can be represented using this general update.

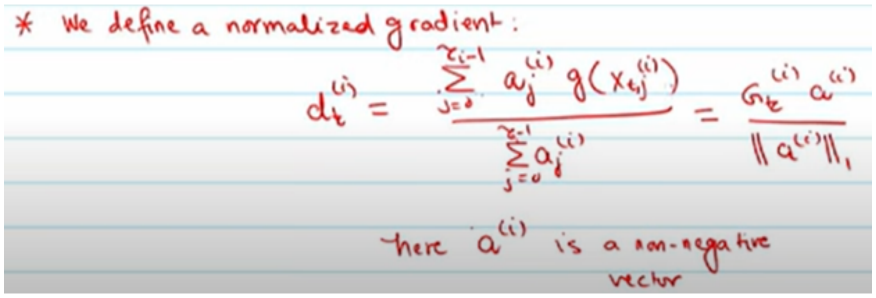
And we are now going to look at FedNova and with primarily because of this objective inconsistency problem and how FedNova tries to eliminate that. Is this clear? So, in vanilla fed averaging, so in vanilla fed averaging x t tau i, this is going to be x t 0 which is same across all clients minus eta times your accumulated gradient which turns out to be j is equal to 0 to tau i minus 1. I am just like for the sake of simplicity, I am just dropping of eta j, but that is clear that it is the gradient computed on that mini-batch. So, this is your accumulated gradient. and let us call this delta t i ok.



* In vanilla FedAvg :

$$x^{(i)}_{t,\tau_i} = x_{t,0} - \eta \underbrace{\left(\sum_{j=0}^{\tau_i-1} g(x^{(i)}_{t,j})\right)}_{\substack{\text{Accumulated gradient}\\ \Delta_t^{(i)}}}$$

This is your accumulative gradient delta t i. Now, instead we define a normalized we define a normalized gradient which looks something like this delta t i or d t i is what you have here. So, the dti vector that we are going to be sending right, this is going to be of the form j equal 0 to tau i minus 1. So, it is defined through another vector let me first write this aj i.



* We define a normalized gradient :

$$d_t^{(i)} = \frac{\sum_{j=0}^{\tau_i-1} a_j^{(i)} g(x^{(i)}_{t,j})}{\sum_{j=0}^{\tau_i-1} a_j^{(i)}} = \frac{G_t^{(i)} a^{(i)}}{\|a^{(i)}\|_1}$$

here $a^{(i)}$ is a non-negative vector

tau y minus 1. So, instead of the idea is instead of sending this accumulated gradient g, summation of g right that is the accumulated gradient. So, the fed averaging was sending this accumulated gradient summation g a and we saw that there is objective inconsistency issue with the fed averaging. So, instead of sending this accumulated gradient which is summation of g, we are instead sending a weighted basically a weighted gradient right and here this vector a. So, a has this components in let us call it a i basically a i 0 a i 1 a i 2. So, essentially we are like we are waiting each gradient in each batch right the gradient g in each batch through this a is a non-negative and you can essentially if it is a non-negative vector you can write this as gt in the vector form.

So, one norm essentially some I mean gti ai is nothing, but it is essentially the this particular term over here. normalized by the value I mean basically one norm of A. Why one norm of A? Because A is anyway a non-negative non-negative vector. So, essentially absolute value of A j is same A j i is same as A j i since it is a non-negative vector. So, in basically in the jth round in the jth local update.

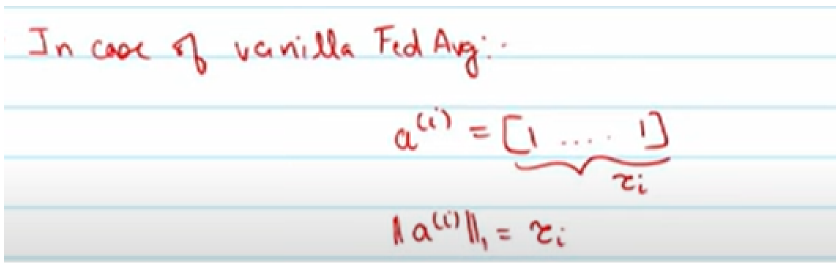
So, what we are saying is instead of just like equally weighting all the gradients which what which is what fed averaging we are going to be weighting the gradient in each in the jth round using this quantity aji ok. And we will see how this has a role to play in the context of objective inconsistency. So, when you have fed averaging as your algorithm what is a ok. So, before we go there let us see what happens if we start sending this normalized gradient. So, if we if we instead if the agents share normalized gradient like this or if the let us say client basically yeah if the agents share this normalized gradient, we can show that on the server side the update rule the general update rule looks like xt 0. plus summation i 1 through m. So, this is essentially the dti that each client is sending. So, it would be minus w is called wi times minus eta times dti . So, is everyone following this? Is everyone following this? So, essentially this is a general update rule right. This is the general update rule.

$$x_{t+1,0} = x_{t,0} + \sum_{i=1}^{m} w_i \left(-\eta \frac{G_t^{(i)} a^{(i)}}{\|a^{(i)}\|_1}\right)$$

So earlier like in case of the Vanilla Fed averaging dti is nothing but the accumulated gradient. Now we are saying that we are not going to be like clients are not going to be sending accumulated gradient. They are going to be sending some kind of weighted gradient or normalized gradient right. So, dti turns out to be I mean since this is what the clients are going to be sending. So, this is how the update for the server update would look like at the end of Tth communication round.

This is how the update would look like. Is this clear? So, if I want to use FedAveraging what should be the a that I should be working with? So, in FedAveraging what is the what do we send? Sum of gradients right. In flat averaging we send sum of gradients. So, if you want just the sum of gradients to be sent out, you should choose a is to be 1, all a's to be 1 right. If all a's are 1, then what is the 1 norm of this particular vector? How many elements are there in this vector a i? Tau i number vectors right.

So, the 1 norm is simply tau i. So, let us see what happens. So, this is so, in case of vanilla fed averaging, a i is a vector of 1's ok, there are tau i number of 1's here. So, norm the 1 norm of a i is essentially tau_i. Gti is nothing but, so what is Gti? It is a gradient, vector of gradients. So, what would be Gti times ai? It is just the accumulated gradient.

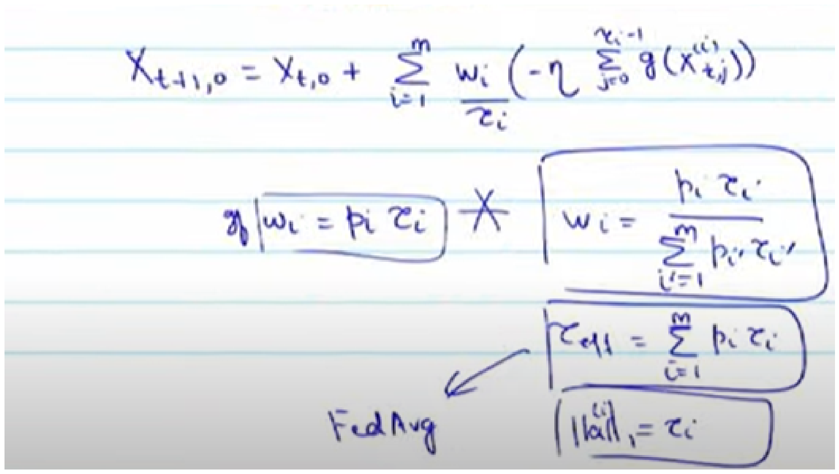


In case of vanilla Fed Avg:

$$a^{(i)} = \underbrace{[1 \ldots 1]}_{\tau_i}$$

$$\|a^{(i)}\|_1 = \tau_i$$

Gti times ai would be accumulated gradients and the tau effective would be, so essentially what would you get? there is a yes there is a tau in the denominator. So, what would what would end up happening ok. So, let us let us try and fit make sense of this. So, this part is clear to everyone. So, fed like for fed averaging you have x t plus 1 0 is x t 0 plus summation wi 1 through wi minus eta times accumulated gradient j 0 through tau i minus 1 g of psi vj divided by tau.



$$X_{t+1,0} = X_{t,0} + \sum_{i=1}^{m} \frac{w_i}{\tau_i}\left(-\eta \sum_{j=0}^{\tau_i-1} g(x_{t,j}^{(i)})\right)$$

If $\boxed{w_i = p_i \tau_i}$ $\times$ $\boxed{w_i = \frac{p_i \tau_i}{\sum_{i'=1}^{m} p_{i'} \tau_{i'}}}$

$\boxed{\tau_{eff} = \sum_{i=1}^{m} p_i \tau_i}$ $\rightarrow$ FedAvg

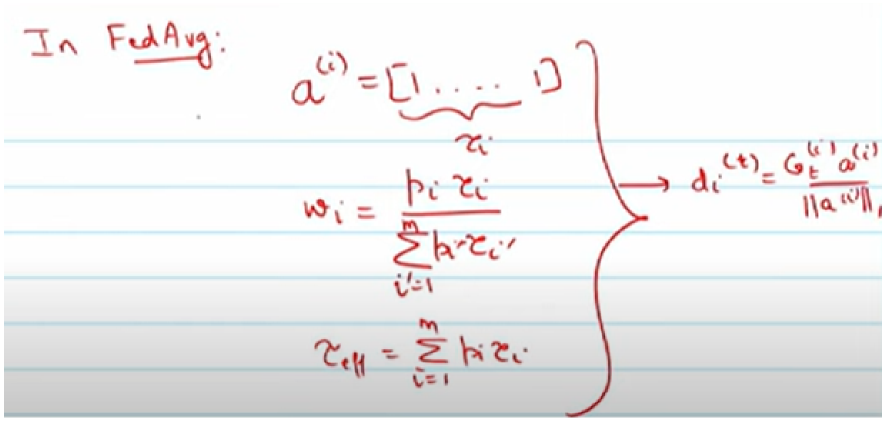
$\boxed{\|a^{(i)}\|_1 = \tau_i}$

And we know this is what. for fed averaging this is the server update rule, this is what the server update rule looks like. So, if I choose pi to be wi times tau i or rather if let us say wi is chosen to be pi times tau i. So, then this tau i and tau i would cancel, cancel each other out. and what you would be left with is the fed averaging. So, in fed averaging if you want to use the same update rule.

So, wi is chosen to be pi times tau i and this vector a turns out to be just vector of 1s. So, that that is when you will recover the fed average. So, from the general from the generalized gradient or the normalized gradient rather you would recover the fed averaging if you use vector of a is to be 1s and wi to be pi times tau is this clear ok. So, that is or I mean other way. So, essentially there is one more constraint we want w i's to be also equal to 1 right that is one more constraint.

So, in that case we rather normalize this. So, this is instead of choosing w i to be this we choose w i to be p i tau i this i prime 1 through pi prime tau i prime. So, that this basically adds up to 1 and then you choose tau effective to be summation i equal 1 through m pi tau i. If I choose this to be my tau effective, this to be my w and a to be ai to be essentially vector of 1.

So, essentially this is equal to tau i. then you essentially recover fed averaging why because if i look at this general update rule here if you choose wi to be summation pi tau i divided by the normalized sum so you have to cancel the normalized sum right so you have to choose tau effective to be that normalized sum that your that sum that you can see you want to cancel this out wi chosen to be pi times tau i because we want this to look like this particular fed averaging thing right is this clear. So, in fed averaging so just to summarize in fed averaging a sub i is chosen to be a vector of all ones. So, there are tau i such an elements wi is chosen to be pi tau i. by i prime 1 through n p i tau i and tau effect p i prime tau i prime and tau effective is chosen to be summation i 1 through n pi tau i. So, if we use all these three together when you send the generalized gradient which is generalized gradient by definition is g pi a i divided by norm of a i, then you would recover the fed averaging algorithm.


In FedAvg:
$a^{(i)} = [\underbrace{1 \ldots 1}_{\tau_i}]$
$w_i = \frac{p_i \tau_i}{\sum_{i'=1}^{m} p_{i'} \tau_{i'}}$
$\tau_{eff} = \sum_{i=1}^{m} p_i \tau_i$
$d_i^{(t)} = \frac{G_t^{(i)} a^{(i)}}{\|a^{(i)}\|_1}$

So, in the fed nova algorithm, so this is your fed averaging algorithm. So, if the fed nova algorithm which is a significant improvement over fed averaging. So, in fed nova what is done is client i normalizes the total accumulated gradient. So, in fed averaging client i simply sends the total accumulated gradient, in fed averaging it normalizes the total accumulated update or which is delta ti by the number of updates, number of local updates tau i. So, essentially your negative eta dt i that looks like.

So, you want this to be equal to delta t divided by tau r, this is what we want this to look like ok. So, then the server is going to get the. So, on the server side the total update is going to be summation i 1 through m, you have pidelta t i divided by tau i and it multiplies by tau effective which is which turns out to be summation i e i tau i.

$$-\eta d_t^{(i)} = \frac{\Delta_t^{(i)}}{\tau_i}$$

$$\sum_{i=1}^{m} p_i \frac{\Delta_t^{(i)}}{\tau_i}$$ and multiplies by $\tau_{eff} = \sum_{i=1}^{m} p_i \tau_i$

$$x_{t+1,0} = x_{t,0} + \tau_{eff} \sum_{i=1}^{m} p_i \frac{\Delta_t^{(i)}}{\tau_i}$$

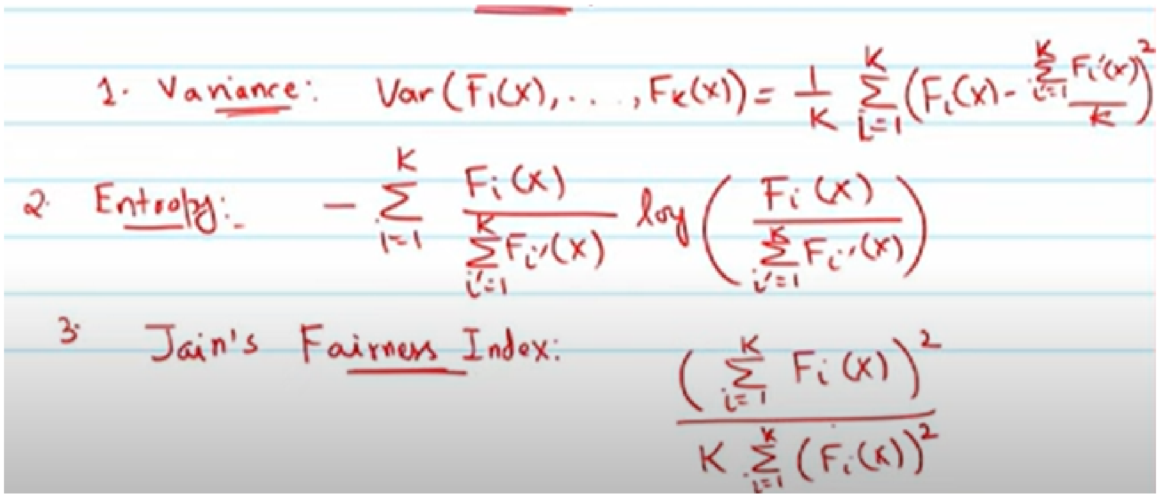
Essentially this is what your FedNova algorithms look like which is x t 0 plus tau effective times i 1 through m p i delta p i divided by tau 1 ok. And you can show that this is compared to Fed averaging, Fed NOVA is actually a much-improved version of Fed averaging.

So, in this case I mean if you want this. So, the aggregation weights w i simply turn out to be p i right because there is 1 over tau i here. Instead of using this aggregation weight in the which is which look something like this w i simply equal to p i and then you would you would essentially end up getting this particular thing over ok all right. So, this is about the fed nova algorithm. And again you can as I said you can view the FED NOVA, FED AVERAGING all of these algorithms as the general using this general update rule. You just have to work with the appropriate choices of your tau effective wi and dti ok.

The last point or the last thing that I wanted to cover in the context of edited learning is fairness. essentially how do we evaluate fairness in federated learning. So, I will just briefly talk about measures of fairness and essentially measures of fairness in federated learning. So, some of the popular measures one is the variance of the function itself. So, let us say the client I, so what first of all what do you mean by fairness? So, fairness essentially evaluates the overall performance against the performance of the individual clients right.

So, how different like or how uniform the distribution is essentially what we are trying to see is it fair to every client or not fair to every client and so on. So, one simple approach is that you just look at the variance and the variance is nothing, but you look at the variance of the function f 1 x their k clients and this is simply measured as 1 over k summation i equal 1 through capital K f i of x minus you just subtract the mean of i prime 1 through capital K f i prime So, it is a simple variance that is one way to estimate the fairness like is everyone's objective is like essentially fair every part spreading clients objective. Second is entropy, so an entropy. So, what is the what is the definite like how do what is the definition of entropy? negative p log p kind of thing right. So, how do we get the probability distribution here? So, it again we are measuring the fairness in the functions itself.

So, what we can do is i 1 through k we can define something like this summation we can try to y prime 1 through k f i prime of x and then the log of the same quantity right. that is the entropy measure. The third is something called Jans' Fairness Index ok and it is usually measured as get the form right. this is Jans fairness or rather fi square of x whatever, but this is Jans fairness index. So, all these are different ways to measure fairness in federated learning.



1. Variance: $Var(F_1(x),\ldots,F_K(x)) = \frac{1}{K}\sum_{i=1}^{K}\left(F_i(x) - \frac{\sum_{i'=1}^{K} F_{i'}(x)}{K}\right)^2$

2. Entropy: $-\sum_{i=1}^{K} \frac{F_i(x)}{\sum_{i'=1}^{K} F_{i'}(x)} \log\left(\frac{F_i(x)}{\sum_{i'=1}^{K} F_{i'}(x)}\right)$

3. Jain's Fairness Index: $\frac{\left(\sum_{i=1}^{K} F_i(x)\right)^2}{K\sum_{i=1}^{K} (F_i(x))^2}$

So, I mean if you have like fairness in a federated learning then you would then you would as I mean as you can imagine if the objective functions are let us say roughly similar across different clients then the error convergence is much better right. Then if you have more heterogeneity in of let us say non fairness in federated learning. So, these are different ways through which you can measure fairness in federated learning. There are other topics like robustness and I mean some more advanced topics I would say, there is something called Q fair federated learning. And so, there are several variants of federated learning which further go into devising algorithms which are essentially fair and so on.

So, we would not give in the interest of this course and largely keeping it around distributed optimization and particularly around optimization. So, we would not go too deep into it and this pretty much concludes all the contents that I wanted to cover in this particular course. Thank you very much.