Distributed Optimization and Machine Learning

Prof. Mayank Baranwal

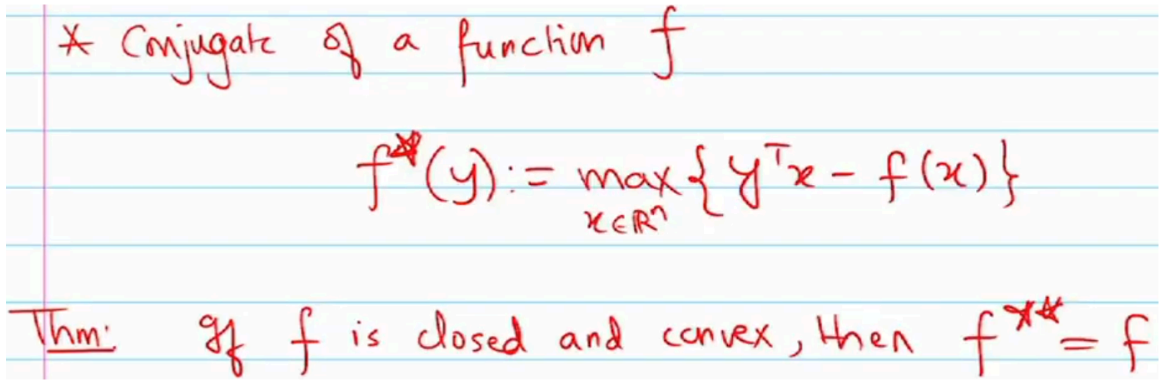
Computer Science & Engineering, Electrical Engineering, Mathematics

Indian Institute of Technology Bombay

Week-7

Lecture - 24: Dual Ascent and Dual Decomposition

So, we have already looked at conjugate of a function or a French dual of a function right. So, it is essentially going to, I mean so today we are going to look at something called dual decomposition. In fact, dual assent is something that we have already looked at right. in the context of fixed like fixed time gradient flow when we looked at the dual function like dual of the or the essential dual of a function we define the gradient ascent on that right. So, dual ascent is something that we have already looked at. So, let us revisit dual ascent and we are going to look at something called dual decomposition.



* Conjugate of a function f

$$f^*(y) := \max_{x \in \mathbb{R}^n} \{ y^T x - f(x) \}$$

Thm: If f is closed and convex, then $f^{**} = f$

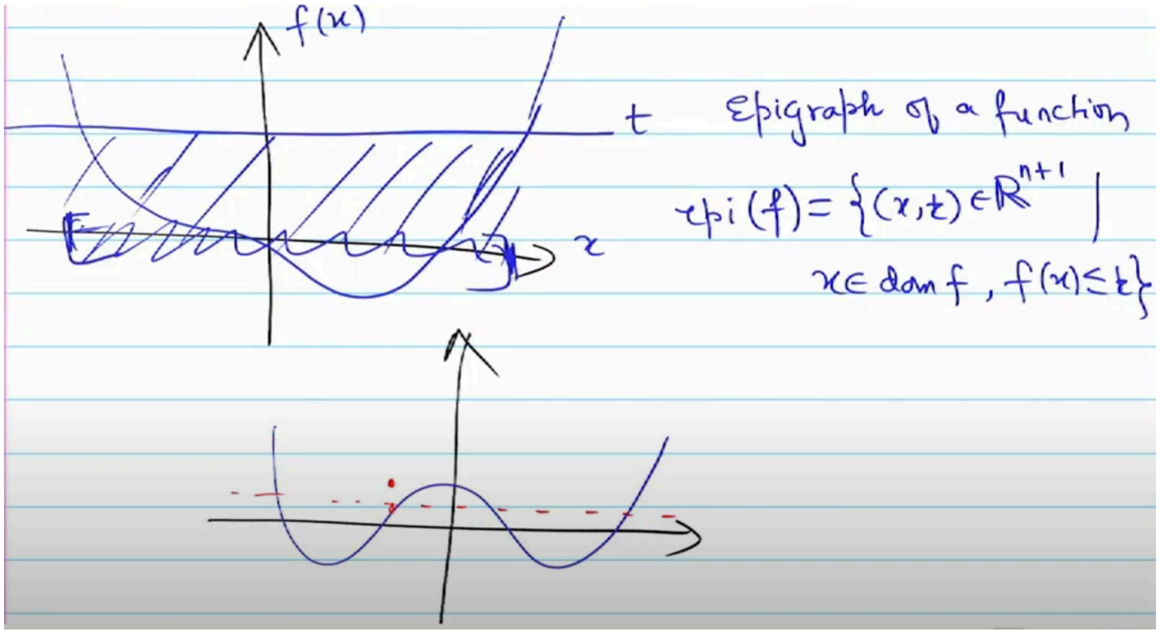
And today you are going to get the first glimpses of distributed optimization through this dual So just to briefly recall, so what is conjugate of a function f? How do we define conjugate of a function? f star of y is defined to be, so what is the definition of conjugate of a function? You can show, so there is a theorem. If f is, if your function f is closed and convex, then f star star conjugate of the conjugate is the original function f itself. So, everyone knows what convex function is. What is the closed function? Yeah epigraph of the function.

So, does everyone know what epigraph is? So, let us say I have a function which looks something like this. And I choose a value of t, I choose a value t. So, y equal to t in some sense. and we look for all points for which f of x is less than equal to t. So, this is my t.

So, for all points for which f of x is less than equal to t. So, in this case it is this interval right. So, this particular interval f of x is less than equal to t. Is this interval closed? So, this is called epigraph of f. So, if epigraph of f is closed, then the function is closed.

Yeah, for any t. So, let me define epigraph. It is given by defined to be set of points rn plus 1 because you are also choosing scalar t such that x lies in domain of the function and f of x is less than equal to t. This is how you define the epigraph of f. So, if epigraph of the function is closed then the function is closed.

So, what could be an example of a function that is not closed? Let us say I choose a function which looks something like this. and I choose t to be a point over here. So, you may have like let us say you may have a jump kind of discontinuity in the function maybe the function value is defined like this for instance here. So, then you can say that the function is not closed or yeah function is not closed. So, if f is closed and convex then f star star is f by the way does everyone remember what subgradients are.

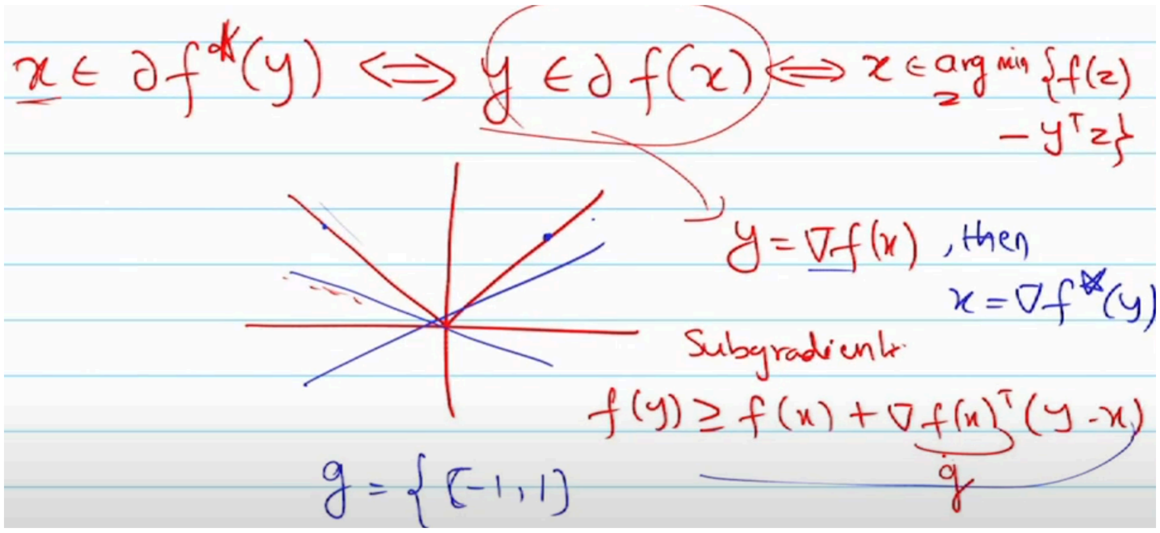


$f(x)$
t
x
Epigraph of a function
$\text{epi}(f) = \{(x,t) \in \mathbb{R}^{n+1} \mid x \in \text{dom} f, f(x) \leq t\}$

So, if x lies in sub gradient of f star at a point y, if and only if y belongs to sub So, what does this say? If you look at your dual function f star or the eventual dual or the conjugate of f, f star. So, what does this say? Like if y is a sub gradient of f at x. So, that means, so the point at which the gradient vanishes. So, y is a point at which gradient vanishes. So, if function if the gradient is not well defined then you can use sub gradient, but otherwise you should really view this as y is equal to gradient of f of x.

No sub gradient is not same as sub gradient is. So, when the gradient is not defined then we can. So, when we talk about convexity. So, just to briefly recap what sub gradients are. when we talk about convexity we have f of y is greater than equal to f of x ok.

But then when this thing is not defined so we can use something called sub gradient g. So, for this is one particular example the gradient is not defined at x equal to 0 right. So, in this case we can use g to be any number from minus 1 through 1. So, and this would hold whereas the value of g is exactly equal to 1 and exactly equal to minus 1 here because the gradients are well defined. So, what this result says is if y is this thing then x is gradient of f star of y.

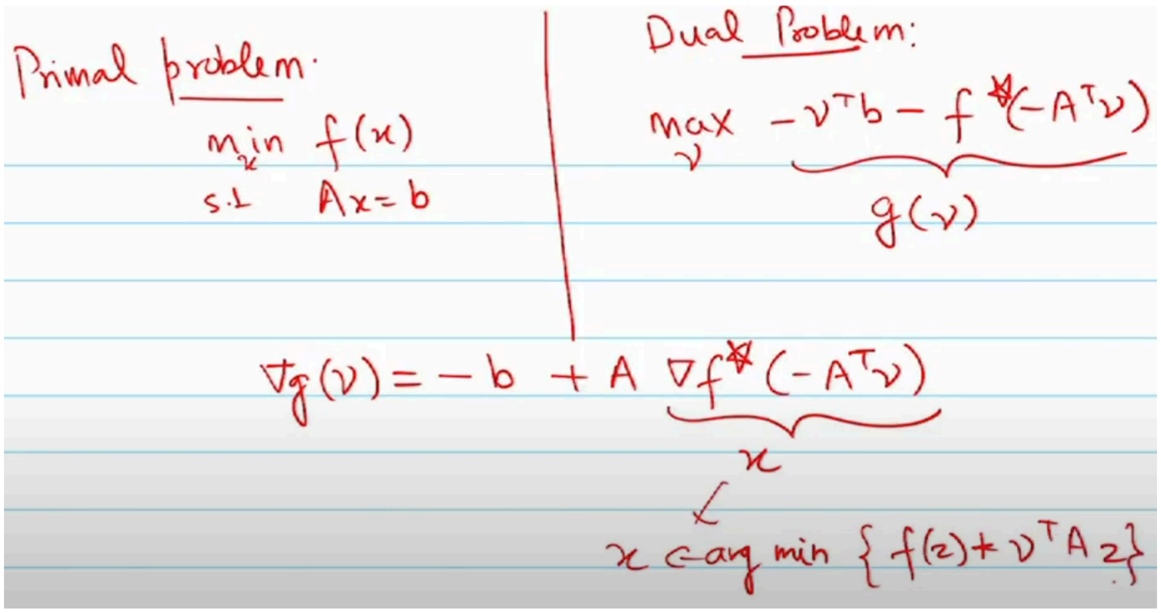

$x \in \partial f^*(y) \iff y \in \partial f(x) \iff x \in \arg\min_z \{f(z) - y^T z\}$
$y = \nabla f(x)$, then
$x = \nabla f^*(y)$
Subgradients
$f(y) \geq f(x) + \nabla f(x)^T (y - x)$
g
$g = \{[-1, 1]$

So, remember we said like if we know the closed form expression for the dual of a function or the conjugate of a function, then we can do gradient descent on the dual itself right, but what if we do not know the dual of that function. So, in that case what can we do? So, that is what this dual descent is about that we are going to look at all right. So, let us look at the primal problem. So, every convex function which is lower semi-continuous will be closed kind of thing if and only if it is closed. So, that then it would not be like if again that if you enforce convexity to it, then so every convex function which is lower semi-continuous is going to be closed.

So, that is I am saying that as a like sufficient condition right. As a sufficient condition, so every convex function if it is lower semi-continuous I think that will be closed that is the, but then I mean that is just that is one that is one way to see if the function is closed, but. So, we are interested in solving this primal problem subject to So, we have already looked at method of multipliers as to how to solve this right. Now, we are trying to take the dual approach that we had already taken earlier, but there we had assumed that we know the conjugate of the function right and that may not always be true. So, what was the dual problem? What is the dual problem here? Maximize g nu right.

So, where nu is your maximize with respect to nu. minus nu transpose B minus that is what we have derived right and this we called it to be g nu and we did a gradient descent on g nu ok. Is everyone with me on this? Now, this closed form expression for f star need not be known right ok. So, what is gradient of g nu here? minus b plus a times. And if I call this to be my x, so if and if you look at this particular result here.

So that means x is nothing but this particular thing right. So, x is here this is where x is argument of f of z plus nu transpose 8 z right. ok minus nu transpose x or minus y transpose x. So, it turns out to be y is in this case is minus a transpose nu right.



Primal problem:

$$\min_x f(x) \quad \text{s.t.} \quad Ax = b$$

Dual Problem:

$$\max_{\nu} \underbrace{-\nu^T b - f^*(-A^T\nu)}_{g(\nu)}$$

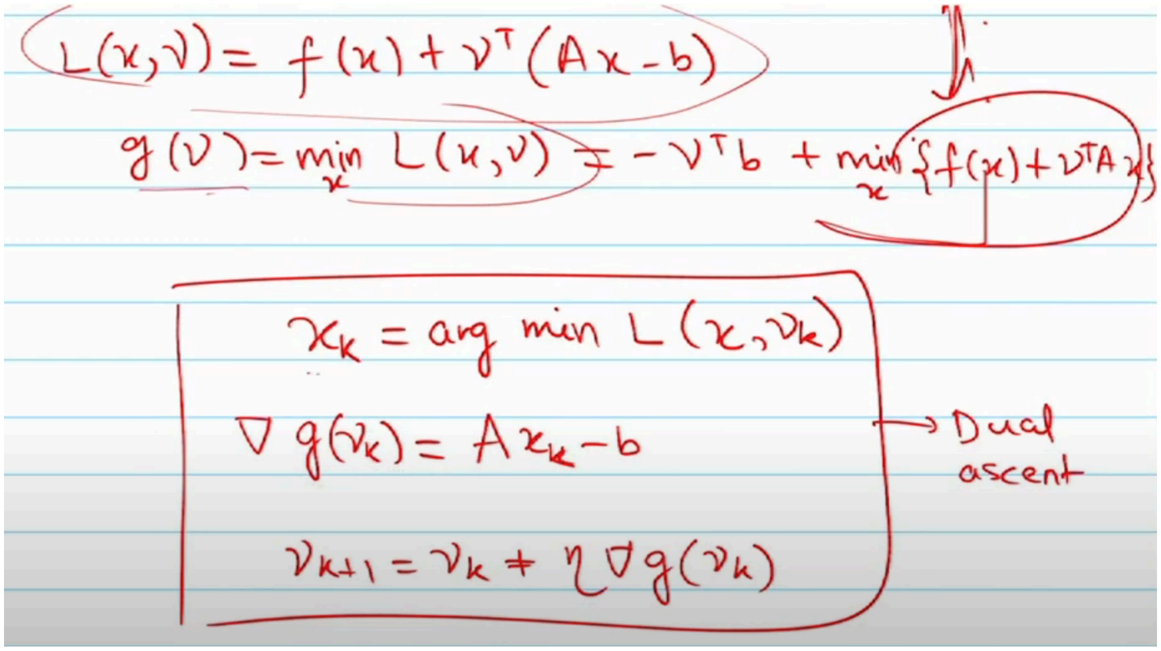
$$\nabla g(\nu) = -b + A \underbrace{\nabla f^*(-A^T\nu)}_{x}$$

$$x \in \arg\min_z \{f(z) + \nu^T A z\}$$

So, this is what. So, also look at what this algorithm I mean what is. So, why do we need this thing right ok. So, if I look at if I had looked at the Lagrangian for this particular problem what would it look like L x nu is f of x plus nu transpose a x minus b right and the definition of g nu is minimize with respect to x L x nu which is minus nu transpose B and this and this are same right. So, this is exactly what we are doing anyway. So, the dual assent algorithm basically tells you that and this is important when we do not know the closed form expression, but because of this particular result.

So, what we can say is if I look at x k which is arg min of arg min or let us say no need to write the L of x nu k gradient of g nu k. is essentially then turns out to be A of x k minus b and then you define nu k plus 1 is nu k plus gradient descent on this right, some step size times and this is your dual descent algorithm. So, in this case we do not need not even know the closed form expression for the conjugate of the function, but we can still do so ok. This is your dual ascent algorithm. So, if I looked at the unconstrained Lagrangian here and I try g nu is defined to be this particular thing right, which by definition turns out to be this right.

So instead of, so if I look at the gradient of g nu, this is defined in terms of the, if I look at the gradient of g nu, this is what this particular terms looks like, right. And because of this particular result over here, so essentially we have gradient of f star evaluated at a y, right. And if that is equal to x, then x basically becomes this particular thing. and that is what we have and in fact you can see that from the Lagrangian and as I mean even otherwise this is exactly what we are doing. So, we can directly define if x k is the argument of this then gradient of g nu k is essentially a a x time a x minus b a or a x k minus b and then you can use this value gradient to essentially update your nu k plus 1.



$$L(x,\nu) = f(x) + \nu^T(Ax-b)$$

$$g(\nu) = \min_x L(x,\nu) = -\nu^T b + \min_x \{f(x) + \nu^T A x\}$$

$$\boxed{\begin{aligned} x_k &= \arg\min L(x,\nu_k) \\ \nabla g(\nu_k) &= A x_k - b \\ \nu_{k+1} &= \nu_k + \eta \nabla g(\nu_k) \end{aligned}} \rightarrow \text{Dual ascent}$$

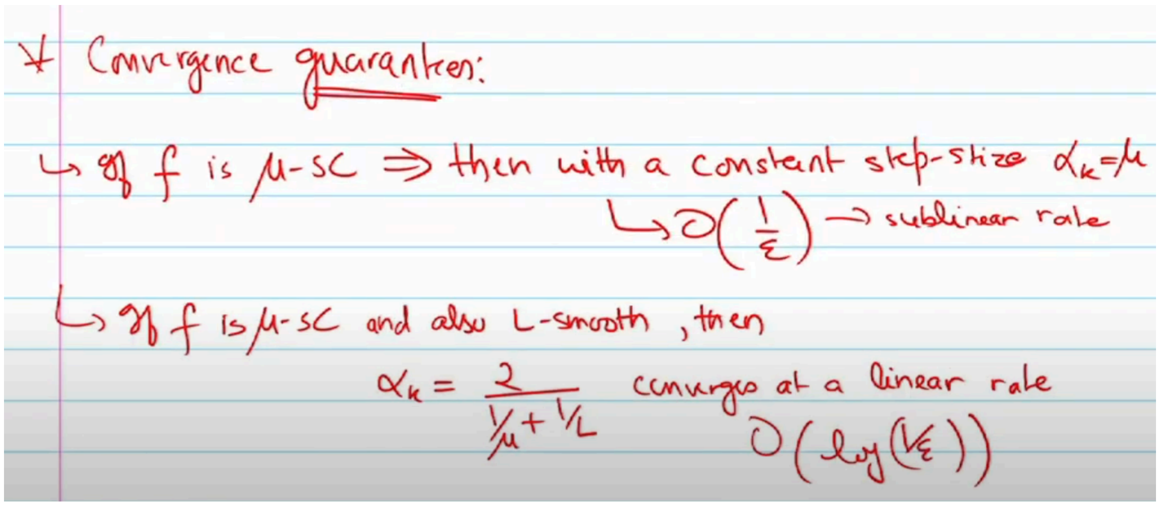
No, we are defining x k to be argument of this thing right. No, no that is not x, I am saying that let us call it x, let us call it x. So, what is then what is I mean so here right again look at this result. In this case x is equal to gradient of f star of y which is same as this argument of this particular thing right. So, if this is x then x is same as argument of this particular term.

and if you look at the Lagrangian, if you want to find g nu essentially the minimum if you g nu when you define g nu this is then this is nothing but define with respect to the argument of this particular term which is same as this right. So, that is what you have here. So, if you find x k which is argument of this Lagrangian you can directly define a gradient So, essentially here you will have to use some again unconstrained any unconstrained optimization algorithm which is your step 1, but this will automatically tell you what the gradient of this dual function is Lagrangian dual function is. Once you find this x k this is simply a x minus v that is step 2 and then step 3 simply updating u k plus 1 and then repeating this. Is this clear? Just gradient descent right on yeah ok.

You can do gradient descent you can do Newton's type method whatever you want to do. Well Newton's type you would also need the Hessian information that you cannot do, but yeah if you want to use momentum and other things that is fine, but this is this is how you can a simple gradient descent would also work. because last time we were using the closed form, so we were using this expression right g nu and we were using closed form expression for f star to find what g nu is. So, now we are not using closed form expression for f star. So, now let us venture into I mean so far we have this course is about distributed optimization.

So, let us look at first such glimpse of what distributed optimization is. yeah before that let us you had asked about the convergence rate and so on right. So, what kind of convergence? So, we can specify convergence guarantees for dual ascent methods. So, let us quickly. So, if f is mu strongly convex, then this implies that then with a constant step size.

So, this is the step size by the way let me use a more generic sort of thing. So, alpha k is your step size ok. Then with the constant step size alpha k equal to mu you get order 1 over epsilon kind of rate which is sub linear right. we are already shown that this is sub linear rate 1 over epsilon, but if f is mu strongly convex and also l smooth then a step size alpha k equal to 2 over this converges at a linear So, essentially you get order 1 log 1 over epsilon kind of number of iterations that is log 1 over epsilon ok. So, this is for the dual descent algorithm.



$\forall$ Convergence guarantees:

↳ If f is μ-SC $\Rightarrow$ then with a constant step-size $\alpha_k = \mu$

↳ $O\left(\frac{1}{\varepsilon}\right) \rightarrow$ sublinear rate

↳ If f is μ-SC and also L-smooth, then

$\alpha_k = \dfrac{2}{1/\mu + 1/L}$ converges at a linear rate

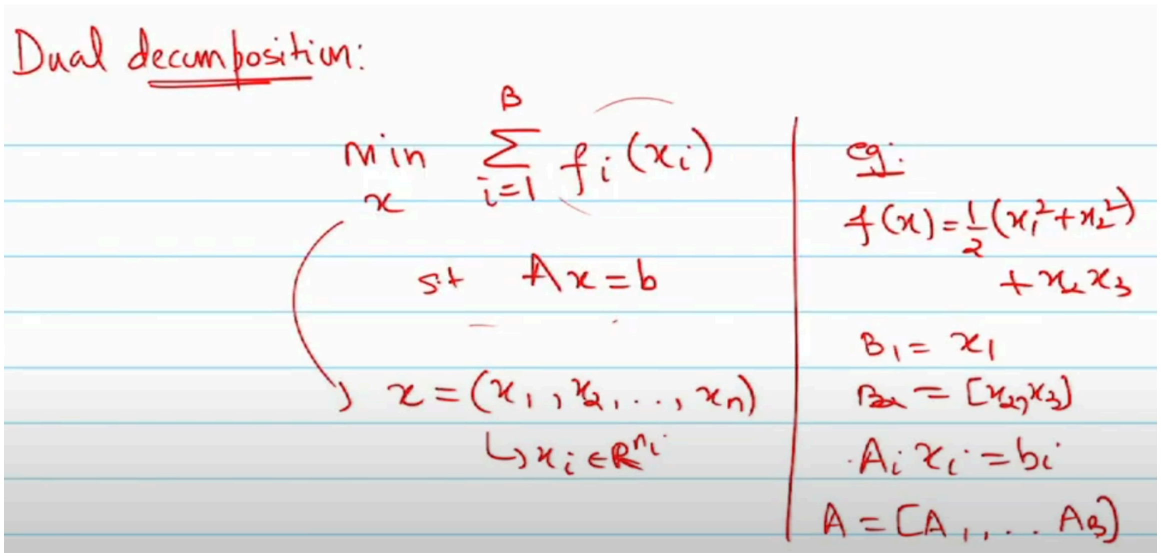
$O(\log(1/\varepsilon))$

So, these are the kind of convergence guarantees that exist in literature. Again we are not going to go into that, but just to make summarize this point particular point ok. So, what is dual decomposition? Suppose I am trying to minimize this function or some of these functions 1 to capital B, I am using B for number of blocks. So, let us say I have B blocks to work with f i of x i subject to Ax equal to sum B.

So, these are the linear. Now, this x is essentially it can be n dimensional. So, x is of the form, I will tell you what is the difference between x n and so on and every x i here it can be in R sub n i. So, an example would be let us consider an example I think it will be much clearer. So, let us consider f of x is half x 1 square plus x 2 square and then x 2 x 3.

subject to sum Ax equal to b. Now if I consider my block 1, so my first block b1 is essentially going to contain x1 and b2 will contain x2 and x3, variables x2 and x3. So, I can write this as summation of fi xi. So, f1 would be simply half x1 square and f2 would be half x2 square plus x2 times x3. the constraints if I know ax equal to b I can write this as ai xi equal to bi form right. If these are linear constraints you can always write it like this in terms of ok.

So, the point is if I write a as essentially a 1 a through all the way through a b. So, I can write down my constraints as well. It can be vector, it can be scalar. In this example these are scalars, but they can be vectors as well. But the point is you can decompose the objective function as well as the constraints to some extent.



Dual decomposition:

$$\min_{x} \sum_{i=1}^{B} f_i(x_i)$$

$$\text{s.t} \quad Ax = b$$

$$x = (x_1, x_2, \ldots, x_n)$$

$$\hookrightarrow x_i \in \mathbb{R}^{n_i}$$

eg:

$$f(x) = \frac{1}{2}(x_1^2 + x_2^2) + x_2 x_3$$

$$B_1 = x_1$$

$$B_2 = [x_2, x_3]$$

$$A_i x_i = b_i$$

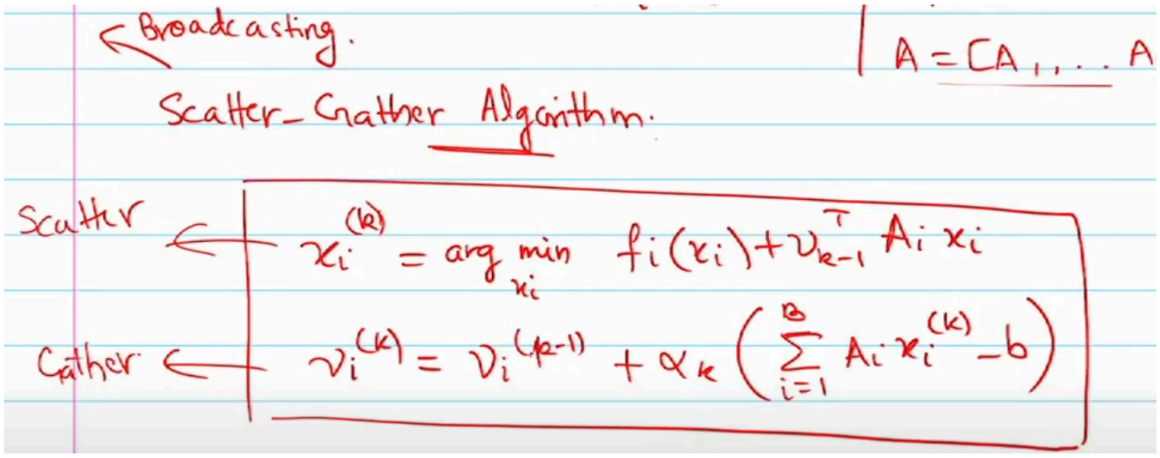
$$A = [A_1, \ldots A_B]$$

Well, constraint part is not that clear because eventually. No, x 2 and x 3 it is one block. So, agent 1 has let us say x 1, agent 2 has x 2 and x 3. So, I am just giving you an example right now I mean let us say if I mean the simple objective was half x 1 square plus half x 2 square plus half x 3 square then you could have decomposed into 3 blocks. But right now we can we are with this kind of formulation we can only decompose into 2 blocks and we have considered of the form Ax equal to b and you again have block wise this kind of thing right ok.

So, we use something called scatter gather type of approach. scatter is also known as broadcasting. So, what happens at the kth iteration is so everyone knows what the current value I mean ideally you would let us say if you have a centralized setup you would know what nu k is right. So, everyone knows what their nu k is corresponding to this particular equality constraint. So, what everyone runs is x i k plus 1 or rather x i k is essentially argmin with respect to xi because xi I mean that is that these are the only functions and where xi is going to show up right and this would be true for any xi.

So, this is what happens and then nu i k is essentially nu i k minus 1. No, but then once you take the derivative I mean does not matter right because it is not a function of x i not the derivative, but it is not a function of the constant term is not even there. So this is your, I mean what is this? This is like gradient of g nu. I mean it is the same algorithm that I am trying to write.

I am directly substituting a x minus b here. So I mean nothing fancy. This is the same dual descent algorithm that I am trying to write. But what happens is this particular approach or this particular step is called scatter. So, everyone is our broadcast step and there is a centralized aggregator that just receives a x from their a i x i from the corresponding agents simply adds them and then updates the new and then broadcast it back to the agent.



$\leftarrow$ Broadcasting.

$A = [A_1, \ldots A$

Scatter–Gather Algorithm.

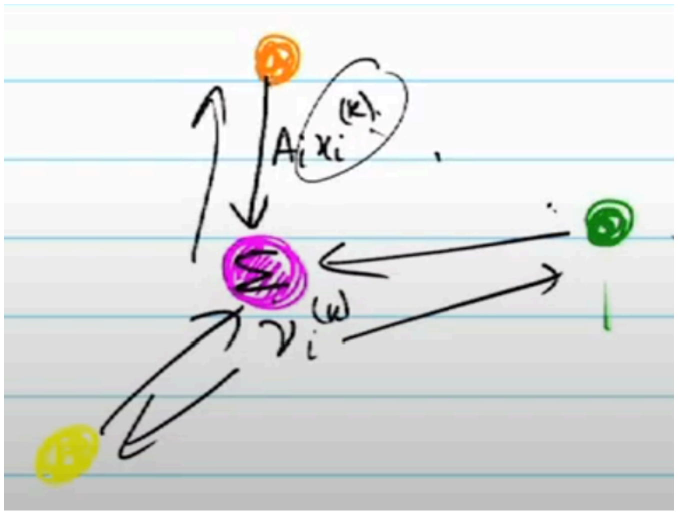
Scatter $\leftarrow$ $$x_i^{(k)} = \arg\min_{x_i} f_i(x_i) + \nu_{k-1}^T A_i x_i$$

Gather $\leftarrow$ $$\nu_i^{(k)} = \nu_i^{(k-1)} + \alpha_k \left(\sum_{i=1}^{B} A_i x_i^{(k)} - b \right)$$

So, this is gather step. So, the way this algorithm works is now you can see like the sort of decentralization happening to some extent not fully, but I mean you still need a centralized entity, but the centralized entity. So, this is let us say a centralized entity. and then you have agent 1, agent 2 and maybe agent 3 here and what is happening is. So, everyone broadcasts or broadcasts their ai xik to the centralized. So, this agent simply sums these in quantities a i x i k subtract the b or the centralized entity subtract b from it updates get a new value of new i k and that sends broadcast it back to all the agents.

So, my data x i k is going to be private first of all because I am sending a i times x i k this is also I mean the other agents are also not going to be knowing what is happening to my data. or my updates and this way you can run this algorithm right. So, this is called dual decomposition. So, it is called dual decomposition because it is built on the dual descent algorithm, but this is this algorithm is known as dual decomposition. And this type of approach where you like you I mean basically broadcast and then you gather this is called scatter gather or the broadcast gather algorithm.

So, this is the first glimpse of decentralization that you have seen in this course. I mean we are going to see a perfect decentralization as well where there is no centralized entity and in that case how do you ensure certain constraints and constraint satisfaction and also minimization, but this is how this particular algorithm works here. Because I mean the consensus would rather have, so consensus would be x 1 equal to x 2 equal to x 3 equal to x n. So, this is also you can model that as also an equality linear equality constraint like this, but I mean this is generally x equal to b kind of constraint.



So, you do not need a consensus in this one. No right, how would they be dependent because if your cost function is decomposable. But x2, x3 are just one agent that is what I am saying the agent is defined by a particular block. So, block 1 is one agent block 2 is another agent. had the objective been half x 1 square plus I mean had the objective been decomposable into three different objectives separately then yeah. So, one agent can have multiple variables that is a possibility, but this is how like.

So, these are. So, every agent is one block here. that too with the centralized entity not with the first of all not with the neighboring agents only with the centralized entity and only interacting through AIX IK and the centralized entity is also interacting with them through new IK. it is not exactly decentralized. That is what I am saying. It is not decentralized, but it is first glimpse of decentralized computation. Every agent is updating their own value and just sending the updated value to this centralized entity which sends back the updated new I for the agents to work out the next iteration and so on. Thank you.