Distributed Optimization and Machine Learning

Prof. Mayank Baranwal

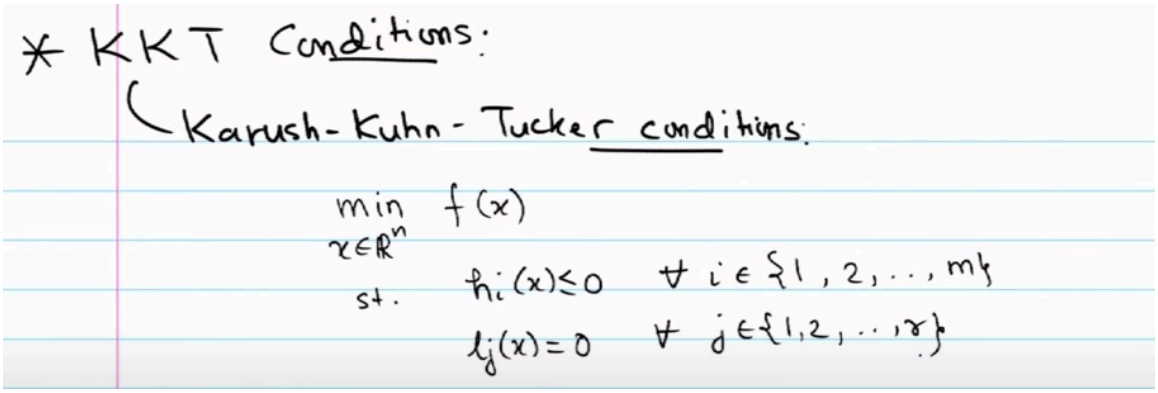
Computer Science & Engineering, Electrical Engineering, Mathematics

Indian Institute of Technology Bombay

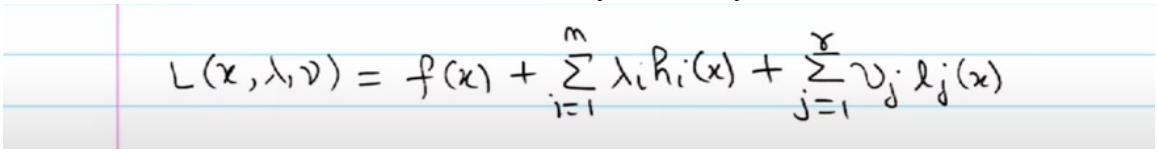
Week-3

Lecture - 11: KKT conditions

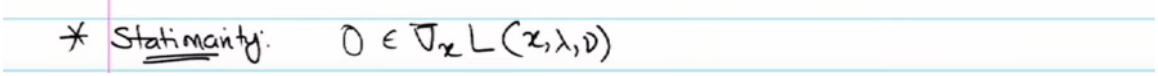
We are now going to look at something called KKT conditions or Karush-Kuhn-Tucker condition. So, again we have this primal problem minimize f of x subject to h i x less than equal to zero for every i in 1, 2, .. m. and lj x equal to zero for all j in 1 , 2 ..



. r. So, this is the primal optimization problem and we define the Lagrangian for this primal problem in terms of lambda and nu where lambda is are the Lagrange variables are the dual variables like with respect to the inequality constraints and nu are with respect to the equality constraint and this was defined to be f of x plus summation i from 1 to m lambda i h i of x plus summation j from 1 to r nu j l j of x. So, what are KKT conditions? So, the first condition is stationarity which says that.

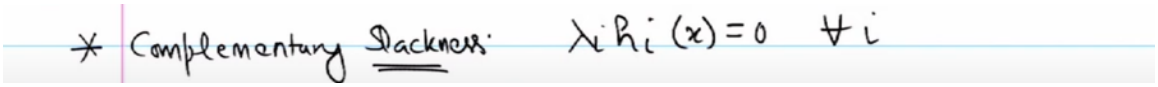


So, why do we need Lagrangian and why do we use Lagrangian? What does Lagrangian help it converts the constrained optimization problem into an unconstrained optimization problem right. So, that means if for let us say for a point x star lambda star and nu star to be optimal point. So, 0 must belong to this particular set of right or x lambda nu. So, the gradient must be like you be 0. So, that is the stationarity condition.

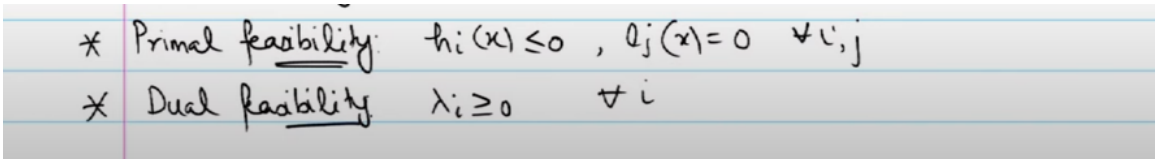


then there is something called complementary slackness which says lambda i h i of x this is equal to 0 for every i. So, either the constraint is 0 or the corresponding Lagrange multiplier is 0 right why because the optimal this term you want this to be equal to f of x right. We know that lg of x is anyway 0 for feasible point. If the point is not feasible I mean you will have these inequality constraints as well right. So, if like if let us say you because you want this to be equal to this, this anyway you know that this is less than equal to 0 and this particular term is greater than equal to 0.

So, overall this term is less than equal to 0. Complementary slackness says that the individually each of these terms the product of lambda i's and the h i of x this is going to be 0 for every i. So, either the constraints are going to be active meaning constraints are going to be satisfied with the equality, the inequality constraint or the lambda i's are going to be 0. So, that is complementary slackness. Then you have something called primal feasibility.

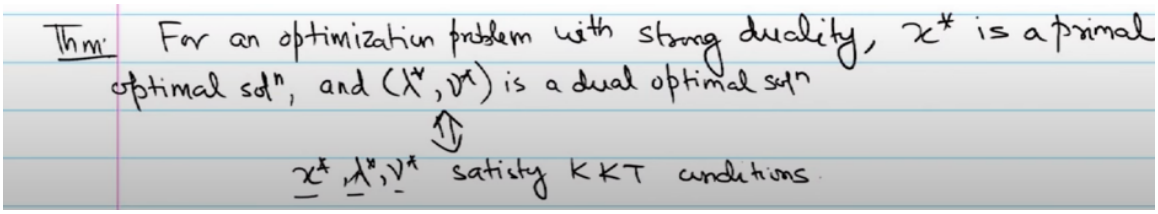


So, primal feasibility is I mean the primal problem should be feasible. So, you have h i of x less than equal to 0 and l j of x equal to 0 for every i j. So, these are just I am just writing on the condition so far and then you have dual feasibility. So, dual feasibility is that lambda i should be greater than equal to 0 right that is the that is the constraint that we have on the dual variables for every i. You can have, but then I mean let us say one of the constraints is trivially satisfied right? So, in general like so basically it holds for every constraint and not just the sum of it right.



I mean in fact I am going to have like have you guys prove this the next term that I am going to write, but so these like are the four constraints clear to everyone. We did not show anything I mean I am just saying that as of now I have just written the four constraints ok. So, let me just write down the theorem statement. So, again we are considering convex optimization. So, everything you assume that f, h and l are the convex functions.

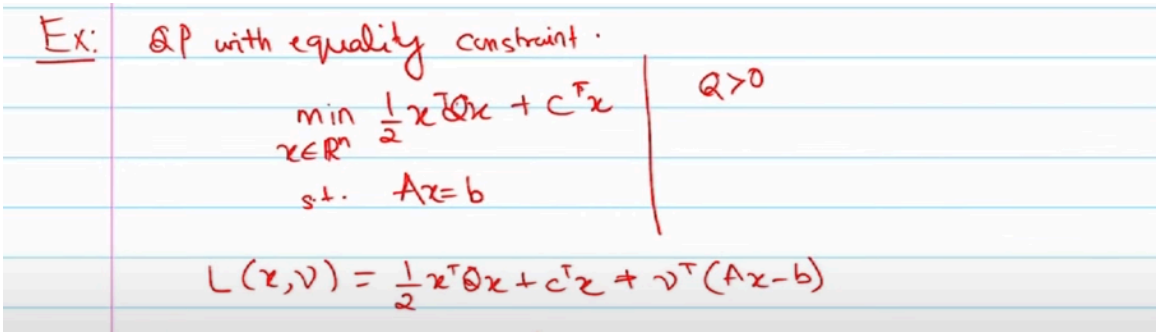
So, for an optimization problem or a primal problem rather or optimization problem with strong duality So, we say x star is a primal optimal solution and the pair lambda star and nu star is a dual optimal solution if and only if x star lambda star nu star satisfy the KKT condition. So, remember when we looked at strong duality we had one statement that if the strong duality holds then KKT conditions which are always sufficient also become necessary. So, this is if and only if condition. So KKT conditions are both sufficient and necessary for let us say x. So what this particular theorem says is that if x star lambda star and nu star, if you can find these points which basically satisfy the KKT conditions, these are also going to be primal and dual optimal solutions.



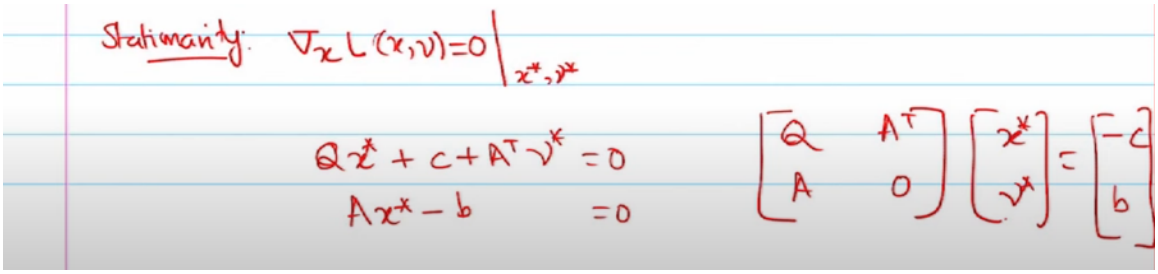
Is the statement clear to everyone? sigma is also 0 and that is when you get I mean the optimal value of the Lagrangian is same as the optimal function value. So, let us look at an example. So, we will consider the quadratic QP with equality constraint, quadratic program with equality constraint. So, the problem is minimize x half x transpose Qx plus c transpose x subject to Ax equal to b. You can assume Q to be positive definite, it is fine.

And let us derive the KKT conditions for this. So, first of all is this problem like does the strong duality hold here? again you have I mean there are no inequality constraints. So, I mean with the question of strict feasibility does not even come in here. You have just equality constraint no inequality constraints. I mean it is you have a convex problem to work with convex constraints.

So, it is fine strong duality holds ok. So, because strong duality holds I mean KKT conditions are going to be both sufficient and necessary. So, let us look at the KKT conditions. of all let us look at the Lagrangian, there is no lambda here, just nu. So, the Lagrangian is going to be L of x, nu equal to half x transpose Qx plus c transpose x plus nu transpose A x minus b.



So, what is the first condition, the stationarity condition? So, let us look at the stationarity condition. So, the gradient of x, gradient of Lagrangian with respect to x that should be equal to 0 at the optimal x star and nu star. So, what is the gradient of Lagrangian with respect to x? What is the gradient of the first term? Qx, let us call it Qx star because we are evaluating at x star plus c plus A transpose lambda this should be equal to 0, sorry yeah A transpose nu, thank you. Let us call it nu star right, this should be equal to 0. The second condition does not even show up.

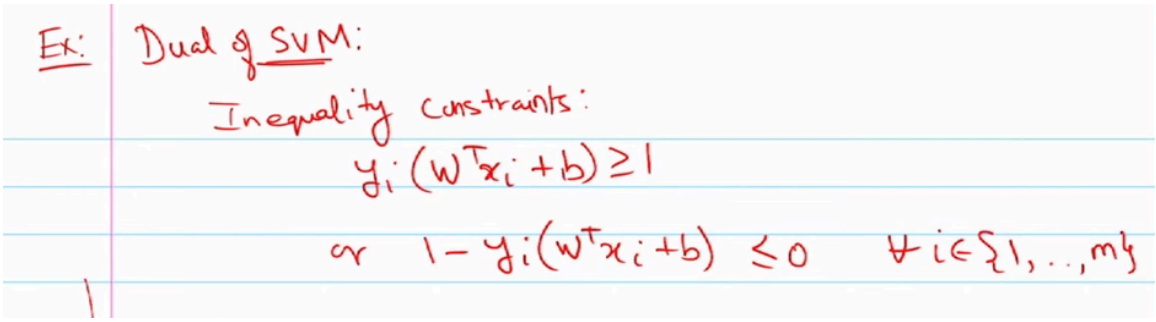


So, complementary slackness because there are no inequality constraints. So, complementary slackness is not there. What about primal feasibility? Primal feasibility would require this to be equal to 0 right. So, that means another constraint that we have is A x star minus b equal to 0 or A x star equal to b that is another constraint and again dual feasibility does not even show up here. So, the solution to this particular.

So, you have x star nu star and this you get to be minus c comma b, you have Q A transpose A 0 . So, the optimal solution is basically the solution to these set of linear equations. So, if you solve, so KKT conditions obviously help you solve an optimization problem, particularly if strong duality holds then they are both sufficient as well as necessary. But the real use case of KKT condition is to check optimality. So, when you are writing, let us say you develop an algorithm to solve an constraint optimization problem, you know that I mean you can like if these constraints are satisfied, means you have arrived at the optimal solution.

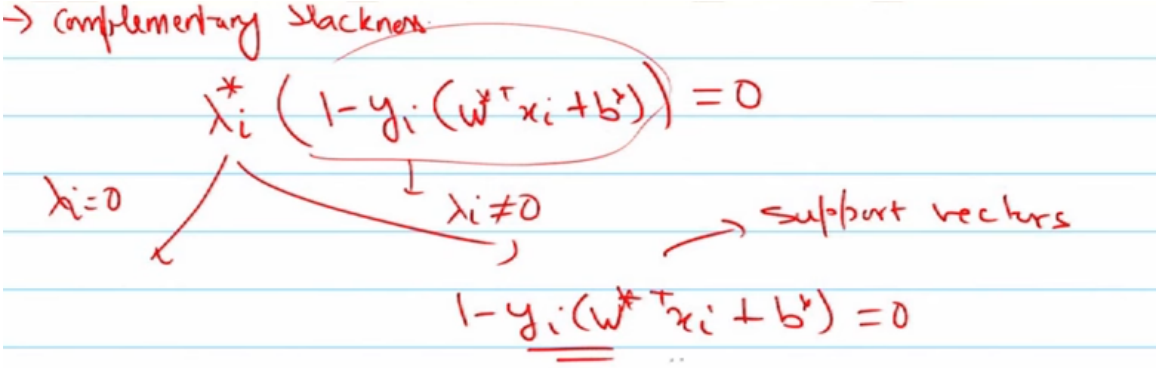
So in this case I mean it's easier to I mean this is a simple enough example you can for quadratic program you can quickly solve it like this but in general like let's say you have inequality constraints and in general it's not a quadratic program you have a more complex looking function. KKT conditions are often used to provide optimality certificates whether you have arrived at the optimal solution or not and the way to evaluate this is if you see that the at the value of x that you currently have if the gradient has almost vanished the lambdas that you have the complementary slackness condition is satisfied primal and dual feasibility are there as long as all these conditions are satisfied that means you have arrived at the optimal solution ok. So, they are often used to check optimality or sub-optimality. and only seldom they are used to solve an optimization problem like in this case, we have solved it using KKT conditions, but more often than not we use it to verify whether we have arrived at optimal solution. Yeah, so I will also look at the inequality problem as well just in fact the same SVM problem is what we are going to look at.

So, the same dual of SVM if we look at. So, here in the dual of SVM we get some inequality constraint as well right. So, the inequality or in the primal form of SVM these are the inequality constraints that we have. So, what does complementary slackness condition? So, again so inequality constraints in the primal form were let me just rewrite it. So, what were the inequality constraints lambda i or the inequality constraints are y i w transpose x i plus v they should be greater than equal to 1 or rather 1 minus y i w transpose x i plus v.



So, this is less than equal to 0 for all i ok. So, if I apply the KKT condition. So, complementary slackness is one of the conditions that we should check right. And what does complementary slackness tell you? At the optimal dual optimal solution lambda i star. So, lambda i star times 1 minus w star let us say w star transpose x i plus b star.

So, this should be equal to 0 right. So, that is the complementary slackness condition lambda i times h i of x that should be equal to 0 ok. So, how is this possible either this is so this implies that either lambda is equal to 0 or the other term is equal to 0. If lambda equal to 0 then this constraint need not be active. it can have some other value, but if lambda is not equal to 0, then that means 1 minus y i there exists one particular x i such the and y i such that this is equal to 0 for some i right.



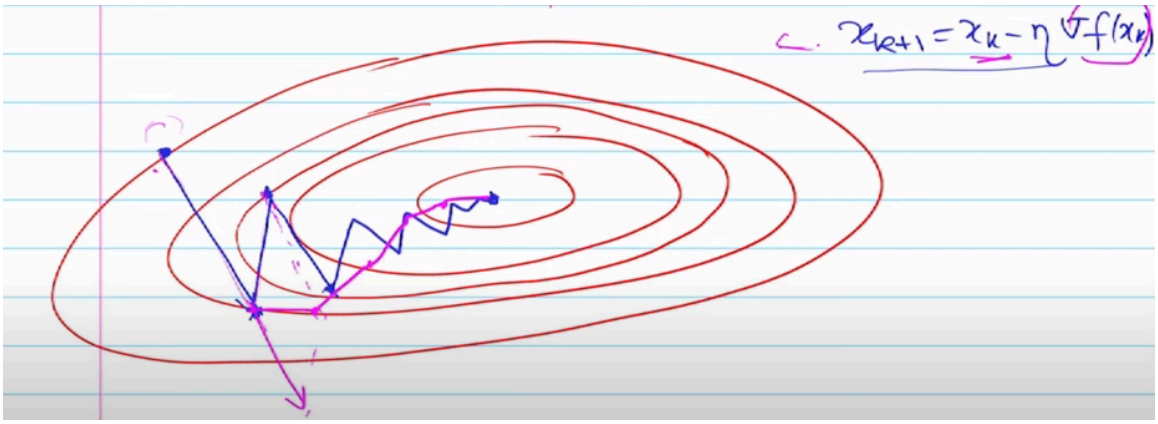
And in fact, these are the points which are called support vectors, and if I look at what support vectors are vectorially. So, these are your support vectors the point. So, this is the equation of as we said the minimum distance of this point is 1 right. So, these are the support vectors and for these set of points lambda i's are non-zero and you get the constraint that w transpose x plus b equal to 0 or equal to 1. of yi times this term is equal to 1.

Is this clear? So, this also gives you a way to, so because you know the corresponding x i star or the point i for which lambda is not equal to 0, this is how you can evaluate your b star. So if you find your support vectors you can evaluate your b star from there and this is this is where KKT conditions can be used. But the KKT the points for which lambda is not equal to 0 these are these basically define your support vectors and so in this case they I mean it has a geometrical meaning to it ok. So we are now going to be looking into

design and analysis of optimization algorithms right. So and the idea is let us let us consider a very simple say we are trying to minimize a function and we consider the level sets of the function and so on.

So, these are the level sets of the function and let us say you start somewhere over here. So, and how does gradient descent work? So, in gradient descent what do we do? x k plus 1 is x k minus some step size times this thing right. So the level set I mean so essentially the gradient would be pointing in the orthogonal direction to it right. So it would be pointing let's say in this direction like this. So wherever we are we then basically move in the direction of the gradient or in the direction of the steepest descent and let's say we arrive at this particular level curve.

Now at this point let's say this is after one iteration. At this point the gradient would be pointing orthogonal to it. So maybe we arrive somewhere over here. Let's draw this level curve here as well. Again we would be moving in the orthogonal direction and we arrive somewhere over let's say here, somewhere here, here, here and so on right and eventually we would converge to the optimal solution and that's how gradient descent in general works.



But one thing that you can notice here for instance is. With this kind of approach you get a very zigzaggy path to the optimal solution or the optimal value right. Like when x converges to the optimizer you have a very zigzaggy way in which x sort of moves towards the optimizer. And this is the nature of like first order optimization algorithm where you just use like let us say you just work with the x and you use the gradient information. Instead what you can use is something called momentum.

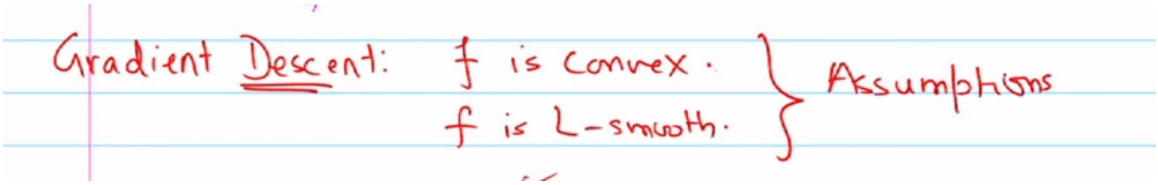
right and if you guys do not know what momentum is I mean it basically has a similar sort of connotation here as well and the idea is. So, I am going to be moving in the direction of the steepest descent, but not entirely I will be using some previous like I will be maybe maintaining some kind of momentum in which I was moving earlier and let's say at this point the gradient is suggested to be here right. So, instead of moving entirely in this direction I will probably move in a combination of like basically sort of a linear combination of these two vectors and that means next step I will move somewhere over here, then somewhere over here, somewhere over here, here, here and that's how I would

eventually converge to the optimal solution. So, you can see it gets a much smoother sort of convergence behavior right, but at the cost of also storing the value of the previous update. So if you are just using xk and if you are just using the gradient information and just storing the value of xk then I would not be able to implement this.

In order to implement this algorithm I need to know this particular direction. Gradient in the previous step and gradient in the current step to be able to move in this direction. So at the expense of storing an additional information I can have a much smoother convergence. Now think of it in terms of when you are training neural networks where you already have billions of parameters to work with.

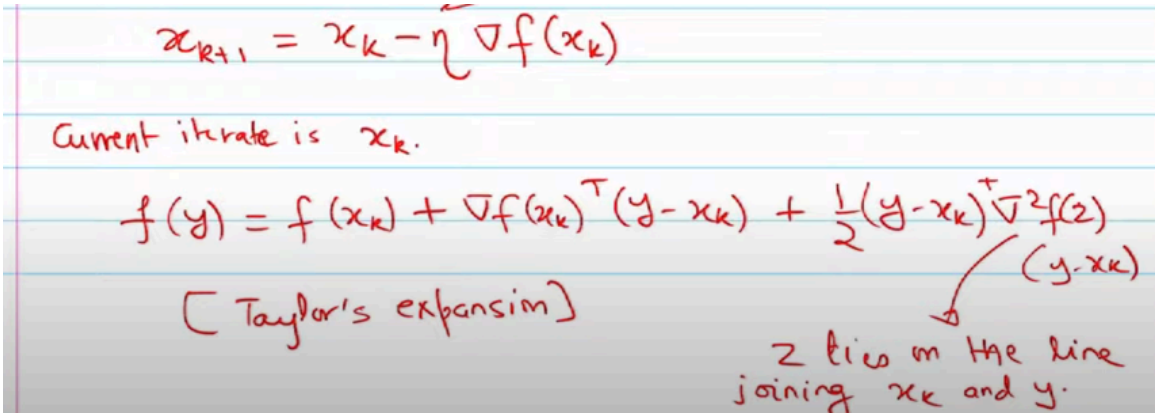
right. So, that means you are storing twice the number of parameters that I mean with gradient descent you are storing just the number of parameters that you have in a network with the with like with second-order method which also uses momentum you would be using twice storing twice the number of parameters right and that and depending on the compute that you may have. I mean it may take more time even though you may have nicer convergence and so on right. Just because it is memory inefficient it may converge faster in terms of number of iterations, but it may be memory inefficient. So the optimization algorithm like the first-order and the second-order optimization algorithm that way have a trade off. generally, with momentum you can you can have a much smoother sort of convergence behavior and in the subsequent lectures we are going to be analyzing optimization algorithms which are momentum-based as well and we will see that in fact you get accelerated convergence behavior with momentum-based algorithms then using something as simple as gradient descent ok.

So, so focus for today's lecture is going to be analyzing gradient descent algorithm entirely and then in the subsequent lectures we will move towards advanced second-order methods, is that clear all right. So, let us start a discussion on gradient descent. So, we are going to be working assuming that function f is convex right. If it is not convex then we cannot guarantee convergence to global optimum. So, we are going to be assuming that f is convex and we are also going to be assuming that f is L-smooth.



So, see we are going to be working under these assumptions that f is convex and L-smooth. Is this clear? And as I said in the beginning of the lecture that we are going to be arriving at the optimal sort of rate and let us see how we can do that all right. So, gradient descent algorithm is going to be x k plus 1 is going to be I mean you are going to be defining it as xk minus eta times gradient of f at xk that is how the gradient descent algorithm works. and the so the I mean let us say if you know that the function is convex and L-smooth the question is how you can arrive I mean how can one arrive at the optimal learning rate in let us say the current iterate is current iterate is x k ok f of y is going to be using Taylor's expansion f of x k plus gradient of f of x k transpose y minus x

k plus half y minus x k transpose hessian of f evaluated at a point z which is which lies on the line connecting x k plus x k and y times y minus x k. Everyone with me on this? So, this is using Taylor's expansion.



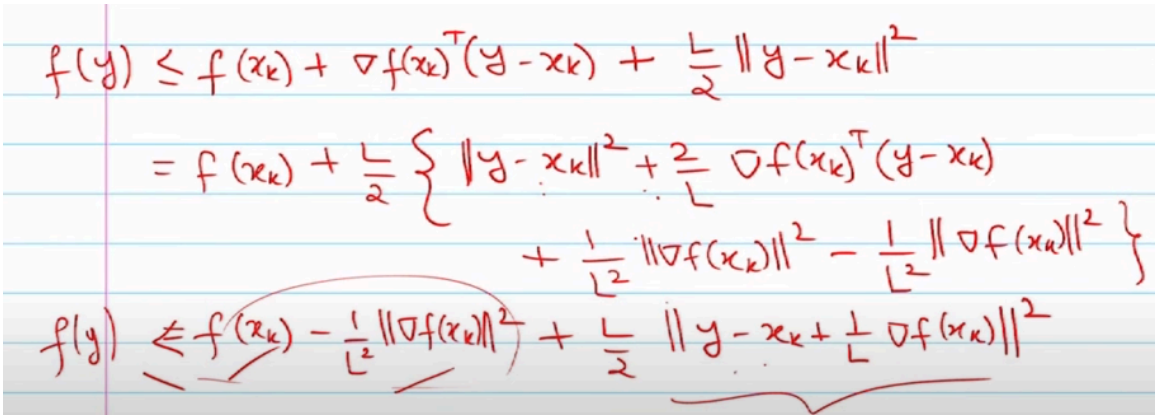
So this holds for some z that basically lies on the line joining xk and y. So this holds with equality. I mean otherwise it is a Taylor's approximation if I had used in xk instead of z. but if you if you I mean you can it holds with equality for some point z lying on the line connecting y and xk ok. So, why did we use this thing? So, we know that function is L-smooth right.

So, the function is L-smooth. What do we know about the Hessian of f? No this is just a Taylor's expansion. Yeah, so this is like Taylor's expansion is approximation, like if I truncate up to second degree, if I had used instead of z, if I had used x k, then it is an approximation, right. It is not an exact equality. It is an exact equality for some z.

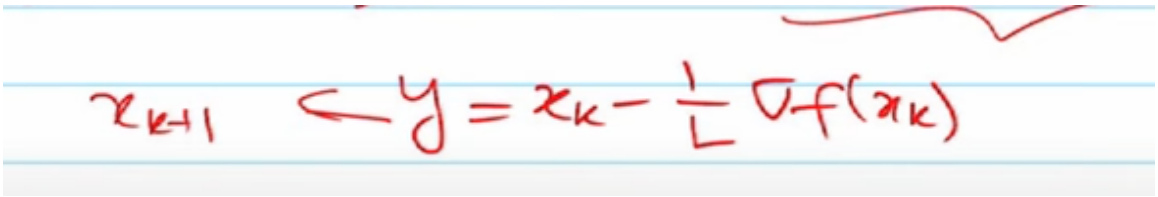
I mean we do not know what z that is. There exists some z connecting and this is true for any function, any analytical function. exists some z such that this is satisfied with equality that z lies in the line connecting these two points xk and y. So, because the function is L-smooth, what do we know about the Hessian of the function? So, norm of the Hessian is less than equal to L. So, that means f y is going to be less than f of x k plus gradient f of x k transpose y minus x k plus L over 2 times y minus x k norm square. Why? Because if I look at the last term here, norm of the Hessian is less than equal to L.

So, this becomes less than equal to L and you get y minus xk norm square. So, this becomes L by 2 norm y minus xk norm square and now you get this with inequality. So, what can we say about the right hand side? So, we know that the current iterate is xk. So, let me rearrange the term and ok and then I can add and subtract 1 over L square gradient of f xk norm square ok. So, this is equal to, so if I look at the these three terms what does this give me? So, this is equal to first let me write this.

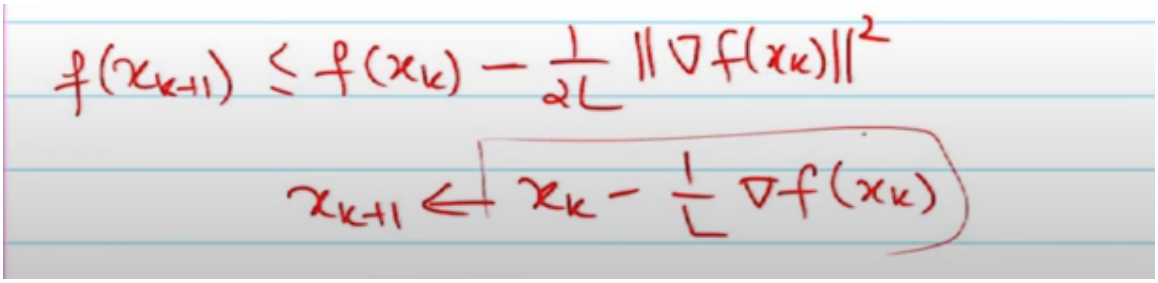
So, this is a function of x k, this is a function of x k. this term is a function of y right. Now I want to, so what do we get? f y is less than equal to this particular thing. Now I want to minimize the, I want to make this bound tighter. So I want to minimize this with respect to y.



Choose the y such that the right hand side is minimized. And how do we do that? If this term, if you can set this term to 0, so that means if I choose y to be x k minus 1 over l gradient of f of x k. then we know that this basically this is anyway going to be independent of y. So, the only term that you can like minimize is just by making this particular term equal to 0 and this essentially tells you that f of x and so, this is what I going what I am going to be calling it as x k plus 1.



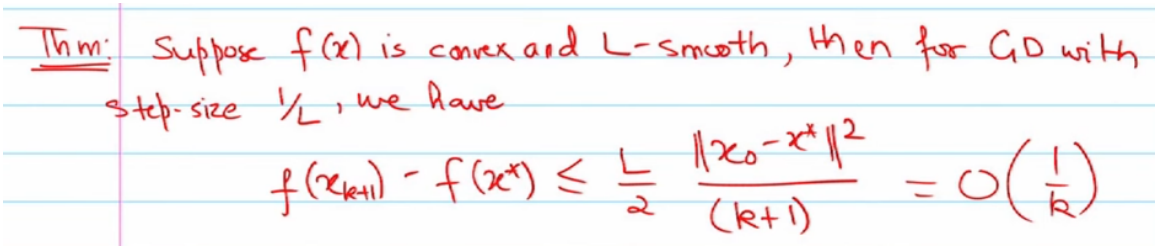
1 by 2L. Sorry, yeah 1 by 2L, thank you. So, my so what do we get f of xk plus 1 in that case is less than equal to f of xk minus 1 by 2 L times xk square and the definition of xk essentially is xk minus 1 by L gradient of f at xk ok. And this is how you get this 1 over L kind of learning rate. because you want to minimize this term on the right-hand side, the bound on the right-hand side and this bound is minimize if I choose if I basically make this term to be equal to 0. It is a constant for the L smoothness.



Yeah, but then if I look at it now this a particular update rule. 1 over L is the learning rate

right. So, you get that learning rate interpretation from this particular equivalence. So, that eta is equal to 1 over L that we talked about in the beginning of the lecture that is how you are sort of getting it over here. This one? So, this also tells you how your f of xk is decreasing right? one thing is this is how you do changing your x but then how basically what is the bound on the function value. So, you can clearly see for instance I mean because the function is the sequence first of all you are always decreasing.

I mean it also tells you that f of xk is monotonically non-increasing right because you subtract a positive value or a non-negative value every time. So, f of x is monotonically non-increasing or rather non-increasing not monotonically, but unless the gradient becomes 0 right. So, it is only when the gradient becomes 0 is when you basically arrive at the optimal solution you cannot decrease any further. So, you arrive at the optimal solution, but otherwise the yeah. So, it basically tells you by how much amount the function value decreases ok.



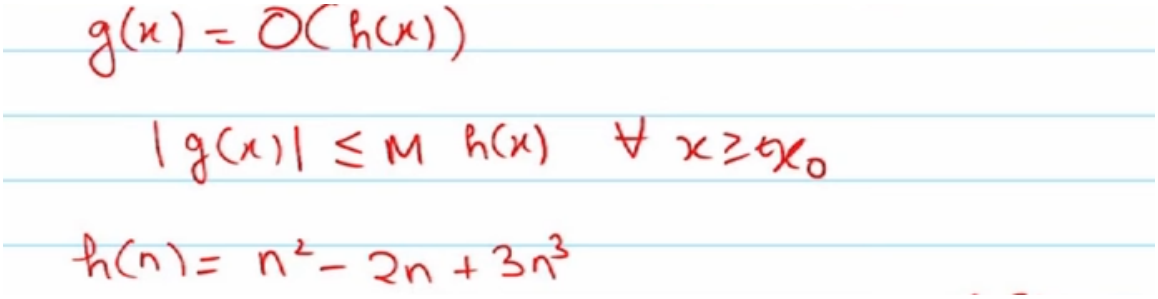
So, as long as the gradient is non 0 f of xk plus 1 is strictly less than f of x k. ok. So, this is your gradient descent and for L-smooth function the sort of eta good suitable choice for eta is 1 over L. In general you do not know the smoothness of like the Lipschitz coefficient of the gradient or Lipschitz constant for the gradient of the function or the L smoothness coefficient you do not know in practice. So, in practice we try different learning rates for functions at least for the functions for which we do not know the value of L.

But let us say if you happen to know the value of L maybe you do not know the optimal solution, but you happen to know the value of L this would be I mean basically this would be in some sense an optimal choice for the learning rate. So, let me write down a theorem. Suppose f of x is convex and L-smooth. So, in the next lecture we are also going to be looking at like I mean implementation and we are going to gain some insights for different types of function, but yeah L-smooth. Then for gradient descent with step size 1 over L, we have f of xk plus 1 minus f of x star where x star is the optimal solution or optimize like yeah.

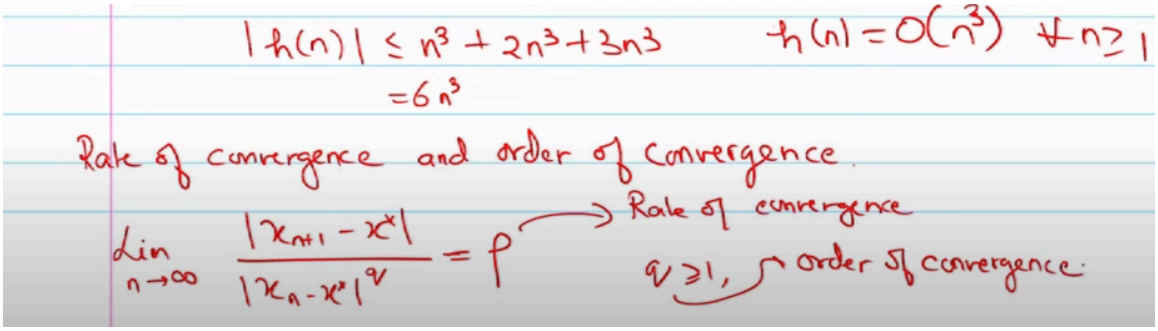
So, this is less than equal to L by 2 times norm of xo minus x star square by k+1 ok. So, this basically tells you how much the function would have decreased. So, let us say you start at xo. and x star is the optimal solution. So, depending on how far you start from xo or the x star, this basically tells you that after k plus 1 iteration, this is how your function value is going to be decreasing.

So, this is nothing but order 1 over k. So, does everyone understand this Big O notation?

Is anyone here who does not understand Big O notation? So, big O like let us say you say g of x is big O of h of x what does that mean? Yeah, so g it is less than equal to some m times h of x for some x for all x greater than equal to some x naught that is what big O notation is right. So, let us say if I define h n to be n square minus 2 n plus 3 n cube. So, what is the order of this like we go if I try to define that. So, for n greater than n greater than 1 or I mean in this case yeah.



So, this is going to be less than equal to n cube plus 2 n cube. plus 3 n cube. So, this is equal to 6 n cube. So, h of n is basically big O n cube for n greater than equal to 1. Is this clear? So, we also define something called rate of convergence and order of convergence. So when we talk about a sequence, let us say a sequence x, so which converges to some x star, suppose this is equal to rho for q greater than equal to 1.



So this rho is called rate of convergence. and this q greater than equal to 1 this thing is called order of convergence. For a sequence x which converges to x star this is how we define the rate and the order. So, we are going to be comparing different algorithms in terms of the rates of convergence and the order of convergence and so on. So, just to keep this in mind this is the context that we are going to be using.

this one. So, absolute value of hn it is going to be upper bounded by this right for n greater than equal to 1 n square is less than equal to n cube minus 2 n is less than equal to 2 n cube and so on right. Yeah, for n greater than equal to 1. So, the definition of the order is that there exists some x naught which in this case and n greater than equal to 1 and some constant like this.

So, in this case m is equal to 6. So, then you can say that h of n is order n cube. Now, how are these results useful these kind of results. So, the idea is suppose I want. So, essentially I want closeness like let us say I want to make sure that after k like I want to

arrive epsilon close to the optimal solution. So, then 1 over k is essentially 1 over k is you set it to be to be equal to epsilon or k is basically 1 over epsilon right. So, basically tells you that you if you want to get epsilon close to the optimal solution then you need to have 1 over epsilon order of number of iterations of the algorithm in order to guarantee that.

And that is how you are going to be reading or at least interpreting these kind of results. So, suppose you want to get epsilon close. So, this term you want to make it equal to epsilon which is almost saying that 1 over k or some like order 1 over k is what which is I mean will be close to epsilon. So, basically your number of iterations required to get epsilon close to the optimal value that would be order 1 over epsilon. So, you need as many iterations to converge to the or get close to the optimal solution epsilon close to the optimal solution.

the smaller the number of iterations the faster the algorithm is and then we for different types of algorithm or maybe what we are going to see is if we assume f to be strongly convex then we can have faster rates than what we have with simply convex right. So, these are some of the things that we are going to be looking at in the next class. Thank you very much.