Distributed Optimization and Machine Learning

Prof. Mayank Baranwal

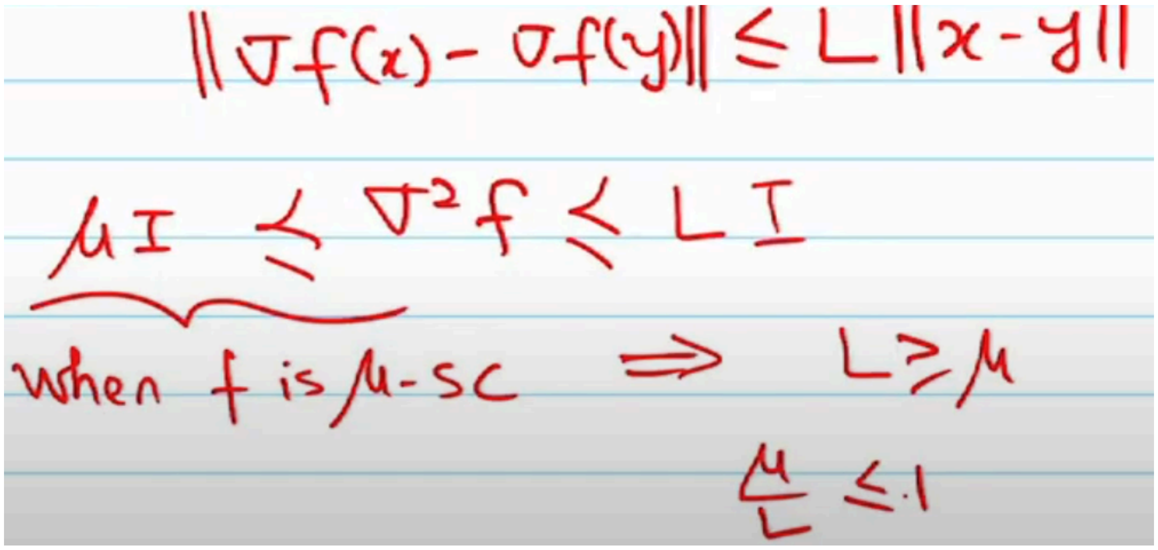
Computer Science & Engineering, Electrical Engineering, Mathematics

Indian Institute of Technology Bombay

Week-3

Lecture - 10: Analysis of gradient descent algorithm

So just a quick recap of what we have looked at in this course so far. So now everyone knows what convex functions are. So we have looked at convex functions, we have looked at variants of convex functions namely strongly convex is something that we looked at in rather extensively. The reason being you can accelerate the optimization of such strongly convex functions. For a general convex function you may not be able to accelerate the convergence. The other thing is strongly convex function also strictly convex function? Yes right and we know it has a unique minimizer.



$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x-y\|$$

$$\underbrace{\mu I \preceq \nabla^2 f}_{\text{when } f \text{ is } \mu\text{-sc}} \preceq LI \quad \Rightarrow \quad L \geq \mu$$

$$\frac{\mu}{L} \leq 1$$

So, unlike with convex functions while it is fine that you I mean whatever minima you arrive it is going to be the global minima as well. With strongly convex function there is going to be just one global minima. So, and we looked at another I mean we kind of briefly touched upon something called L smooth function. does anyone remember what else smooth function is? So, the gradient is Lipschitz right.

So, just to recap what else smooth functions are. So, smooth functions if I consider the gradient of this function. So, gradient has this Lipschitz property. So, another way to say this is hessian of f that is going to be upper bounded by L times identity that is your L smooth function. And we know that if the function is strongly convex then it is lower bounded by mu times identity.

So, this is when f is mu strongly convex. So, what does this tell you? if the function is both strong mu strongly convex and l smooth what does this tell you? Yeah. So, from here you can conclude that L is always greater than equal to mu ok or rather mu over L for a strongly convex function which is a L smooth as well. So, this number is always going to be smaller than or equal to 1 ok. So, this is one thing that we are going to look at when we derive convergence rate for different types of algorithms all right.

So, we have looked at so far we have looked at convex functions certain variants of convex functions as I said the strongly convex functions strictly convex functions and so on. In the last class in particular we also looked at Slater's condition and how it sort of relates to strong duality. And basically we looked at the dual formulation of primal optimization problems. ok alright. So, does everyone know what gradient descent algorithm, how it works? So, we simply, so if I look at something like gradient descent, it simply has this kind of update right x k plus 1 is x k minus step size times gradient of f evaluated at x k.

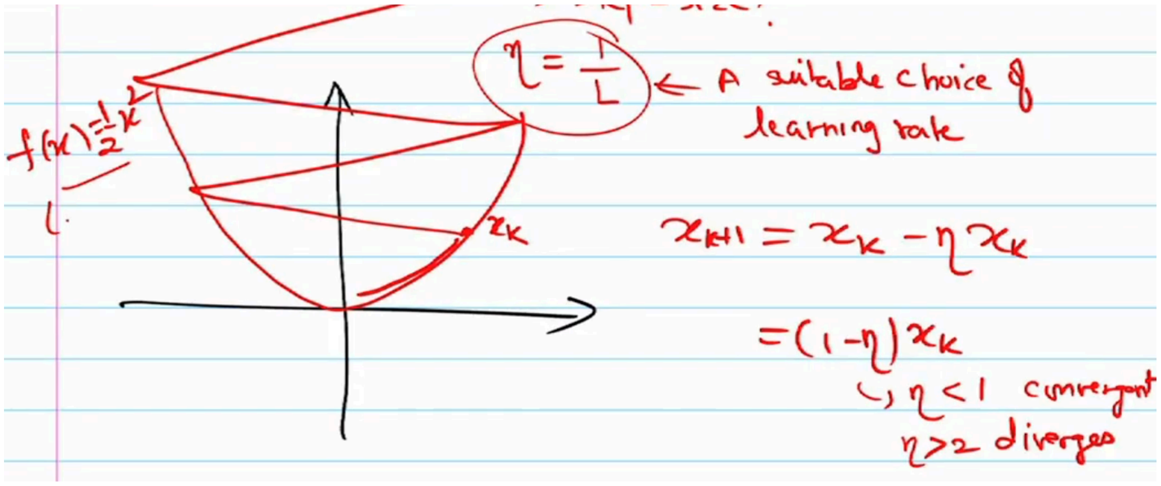
Now, how do you choose this particular step size? Anyone who has worked with optimization algorithm? Hessian as in? Yeah, so if Hessian inverse rather if you, but that is for Newton's method right for simple gradient descent which does not use second order information. Let us say we are looking at simple gradient descent. You can do line search. So, let us say, let us add a little bit more information about this function. So, let us say the function is L smooth.

$$\underline{\text{GD:}} \quad x_{k+1} = x_k - \underbrace{\eta}_{\hookrightarrow \text{ Step-size?}} \nabla f(x_k) \quad \longleftarrow \ f \text{ is } L\text{-smooth}$$

Can we say something about and let us also assume that we know the value of L. We need not know the optimal solution, but we may know the value of L right. So, does this give us some way to come up with an optimal learning rate? So, the answer will to some sense in some sense the answer is yes and today I mean the focus for today's class is going to be analyzing the naive sort of gradient descent algorithm and we will show that for L smooth functions which are convex need not be strongly convex for L smooth functions which are convex a good choice of the learning rate would be 1 over L and we are going to be deriving that today ok. So, this is So, we are going to look at that criteria how we basically come up with 1 over L and first of all I mean to start with would it even converge right. Let us say if I consider a problem of this form.

Let us say I am trying to minimize x square right. ok. So, I have x k here, x k here and then I want to arrive at an x k plus 1 which will improve the function value right. So, what is the gradient for let us say f of x is half x square ok. What is the gradient of f? x right.

So, if I look at the for just for this specific problem, if I look at x k minus eta times gradient of f of x k which is also going to be x k, this is 1 minus eta times x k right ok. If I choose eta to be more than 1, let us say I choose eta to be 2, I learning a rate of 2 what happens? it just keeps oscillating between plus x minus x plus x minus like basically whatever let us say you start with x naught equal to 2, it will keep oscillating between 2 and minus 2 right. So, it does not even converge. If I choose eta to be more than 2, it diverges right. So, what would happen is this is for eta more than 2, it will just keep on like basically diverging eventually.



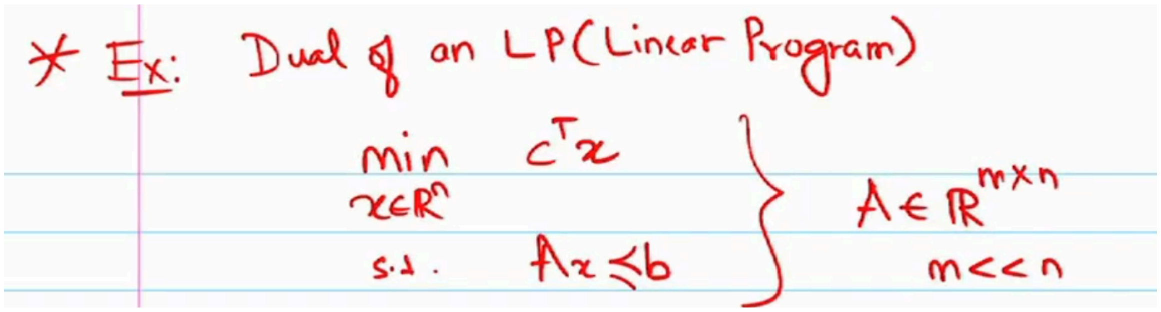
$f(x)=\frac{1}{2}x^2$
L.
x_k
$\eta = \frac{1}{L}$ ← A suitable choice of learning rate
$x_{k+1} = x_k - \eta x_k$
$=(1-\eta)x_k$
$\eta<1$ convergent
$\eta>2$ diverges

So, the choice of learning rate matters right as you can see when you are talking about discrete discretized implementation or discrete time algorithms the choice of learning rate matters. So, eta less than 1 we know that it is sort of slowly converges right. So, eta less than 1 it is convergent and of course, you are going to assume that the learning rate is positive that is an that is a lower bound and learning rate anyway, but eta less than 1 it is going to be converging because the new value of x k is plus 1 or new value of x is going to be smaller like basically something lesser than x k right. So, it is going to be converging eta equal to 1 what happens? It instantly arrives at the optimal solution right in one step. If eta equal to 1, in one step you arrive at the optimal solution and eta greater than 1, in fact greater than 2 it starts diverging, greater than 2 it diverges.

So, the choice of learning rate plays a key role and if let us say, so what is the value of L in this case? 1 right. So, in some sense you see that 1 over L earning rate plays a key role right. 1 over L is also 1 and you see that in one step you kind of converge at least in this problem. In one step you converge to the optimal solution. Now yeah it is not necessary that it will converge in one step that is true.

So, that is why I said in this problem at least we see that for l equal to 1 in one step we converge right. So, in some sense it tells you that it has this kind of optimality or like optimal behavior when you choose the learning rate to be 1 over l ok. So, that is something that we are going to be deriving today. Alright so, let us start with today's lecture. So, we looked at ways to convert a primal optimization problem to a dual optimization problem right.

So, let us look at few a simple example. So, we consider a linear program. So, dual of a linear program ok and the problem is of the form minimize c transpose x subject to. So, let me just minimize c transpose x, let us say x is in R n subject to a x less than equal to b. And why do we need to work with dual of like dual problem here, you can assume that a is matrix from m cross n and you can assume that m is much much smaller than n ok.



$*$ Ex: Dual of an LP (Linear Program)

$$\left.\begin{aligned} &\min_{x \in \mathbb{R}^n} \quad c^T x \\ &\text{s.t.} \quad Ax \preceq b \end{aligned}\right\} \quad \begin{aligned} &A \in \mathbb{R}^{m \times n} \\ &m << n \end{aligned}$$

So, how do we dualize this? Yeah. So, this is less than or equal to rho wise. Yeah, element yeah rho wise ok. So, how do we dualize this? We define the Lagrange dual. So, we consider like in this case we have m inequality constraints.

So, we are going to consider lambda which is m dimensional. And we are going to be defining this lambda as, now it is going to be an unconstrained minimization over x basically f of x plus lambda transpose whatever inequality constraints you have right which is h x. Let us say you had the inequality constraints h x less than equal to 0. So, in this case it turns out to be minimize c transpose x just lambda transpose A x minus b. Now, what does this evaluate to? So, let me, so you get minus lambda transpose b, just let us collect all terms.

Is this clear? So, g lambda by definition is going to be minus lambda transpose b if this term is 0 right. If this term is 0, if a transpose lambda plus c turns out to be 0 then it gets a value of minus lambda transpose b otherwise it can be negative infinity right. If this is not equal to 0, you can choose an x. If this is not equal to 0, let us say for one particular row, this value a transpose lambda plus c, I mean you can always choose x such that you can make things like negative infinity right. You can just keep increasing your x and x to a point, you choose a sign accordingly such that this particular element becomes negative infinity.

$$g(\lambda) := \min_{x \in \mathbb{R}^n} \left\{ f(x) + \lambda^T h(x) \right\}$$

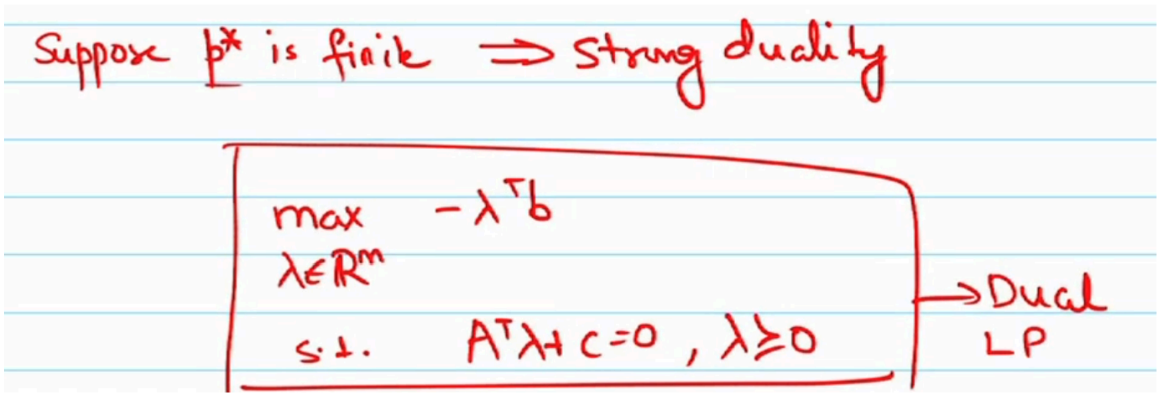
$$= \min_{x \in \mathbb{R}^n} \left\{ c^T x + \lambda^T (Ax - b) \right\}$$

$$= \min_{x \in \mathbb{R}^n} \left\{ -\lambda^T b + \underline{(A^T \lambda + c)}^T x \right\}$$

$$g(\lambda) = \begin{cases} -\lambda^T b & \text{if } A^T \lambda + c = 0 \\ -\infty & \text{else} \end{cases}$$

So, suppose So first of all is this problem let's say suppose p star is finite or the primal optimal is finite. So does the strong duality hold here? Or Slater's condition does it hold here? So for linear inequality constraints do we need strict infeasibility or strict feasibility? no right. So, these are linear inequality constraints. So, we do not even need strict feasibility status condition hold true by default. So, I mean if p star is finite I mean you in fact have linear function to start with which is convex right.

So, you have the assumption star that we started with and p star is finite. So, you have strong duality here. So, that means this is not true. So, the corresponding dual problem is maximize minus lambda transpose b subject to you have constraint a transpose lambda plus c greater than 0 and lambda is greater than equal to 0 and this is your dual problem.



Suppose p^* is finite $\Rightarrow$ Strong duality

$$\boxed{\begin{aligned} &\max_{\lambda \in R^m} \quad -\lambda^T b \\ &\text{s.t.} \quad A^T\lambda + c = 0, \ \lambda \geq 0 \end{aligned}} \rightarrow \text{Dual LP}$$

This is a dual linear program. Is this clear? So, dual of an LP is also an LP like you start if you start with linear program in primal variable you come up with a dual optimization problem which is also an LP right, it is a linear program in lambda. So, dual of an LP is also an LP, but the good thing is if you I mean in earlier case we had started with inequality constraints. for the dual problem we all we get the equality constraint plus lambda greater than equal to 0. So, I mean manageable inequality constraints, but then you get an corresponding equality constraint here right. So, that that is one advantage of working with the dual formulation of an LP ok.

because I mean let's say you are solving the primal right and if m is much much smaller than let's say you are using some kind of gradient descent or something and let's say m is much much smaller than n and n is very large let's say 100,000 dimensional. So you are storing an x which is 100,000 dimensional and you are sort of like updating it every time right. So that becomes memory inefficient. Whereas when you work with dual if lambda let us say m is much smaller let us say m is equal to 10 or m is equal to 20 you are now working with the much smaller m right much smaller dimension like storing smaller lambda is much easier than storing a larger x. But now we have a lot of constraints.

Not a lot of constraints we had like lot of constraints even earlier as well right ax less than equal to. sure but then these are equal these are equality constraints that I mean that makes things easier right like first of all I mean equality constraints is something has to lie on the hyperplane versus something that can be anywhere like less than equal like I mean in the half space that is separated by that hyperplane right yeah I mean I mean both are linear programs in some sense the difficulty at like at some level is kind of in terms of algorithmic from algorithmic view point I think they are going to be similar it is largely about like are you going to be working with an n dimensional vector versus an m dimensional vector and that makes things much more efficient if you are going to be working with the smaller dimensional vector. And also think of it in terms of again I mean like at the back of your mind keep the distributed optimization aspect like I mean we are looking all of this because eventually we are going to be working with dual algorithms when we look at distributed optimization problem. And if you are going to be exchanging information with your neighbors you want to exchange vectors or information which are smaller dimensional than like higher dimensional right or larger dimensional. So, it makes sense to work with dual optimization problem than primal ones.

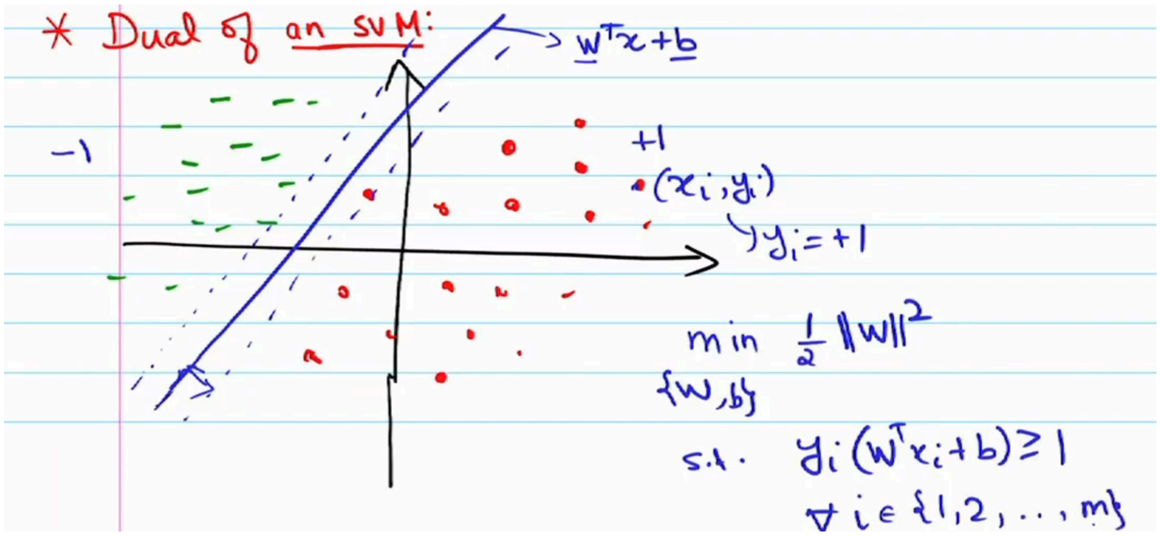
Sir. Yeah, so then I mean so then there is this constraint which sort of relates I mean in case of here I mean yeah, so that is a good question. In the earlier case usually we So you know the value of for instance once you get the optimal value right maximum value. So lambda star transpose b plus a transpose lambda star c plus x star that would give you the that would basically hold with the equality right and then from there you can solve for x star. So eventually you need to find x and you can do it I mean once you have sort of again think of a distributed optimization setting right. you work with lambdas and once you have your consensus on lambda among all the agents then I mean they can simply sort of get the original primal variable using this equality right.

So, you are right I mean this basically gives you the same optimal value not the value of x star directly, but then you can solve for x star from here. So, we are going to look at another advantage of working with dual problems and this time through this support vector machine ok. So, dual of an SVM. So, does everyone remember what SVM is support vector machines.

So, let just recall. So, you have bunch of positively labeled and negatively labeled data points. And the goal is to find a separating hyperplane or a hyperplane, let me use a different color that separates these two sets of points right. So, let us let us say the labels for these points is plus 1 and for these set of point that is minus 1. So, any point is represented by x i y i. So, x i is a like basically coordinate of that point and y i is essentially the label of that point.

So, y i in this case is plus 1 ok. Now, you want to find the hyperplane that maintains the maximum margin with respect to the positively and the negatively labeled data points and margin is in some sense it is related to how much you can perturb on either side before like you incorrectly sort of classify a particular point right. And you want to maximize this margin in some sense because there are multiple solutions that separates these two sets of points. So, which one do you choose you choose the one which maximizes this margin robustness margin. ok. And equation of this hyperplane was given by w transpose x plus b.

So, remember in this case the decision variables are w and b not the x, x are the data points then y are the label ok. And we derived the this particular optimization for the primal optimization problem and this was minimize with respect to w and b. subject to let us say there are m points. So, this was the primal problem that we had derived I think in the few lectures back right.

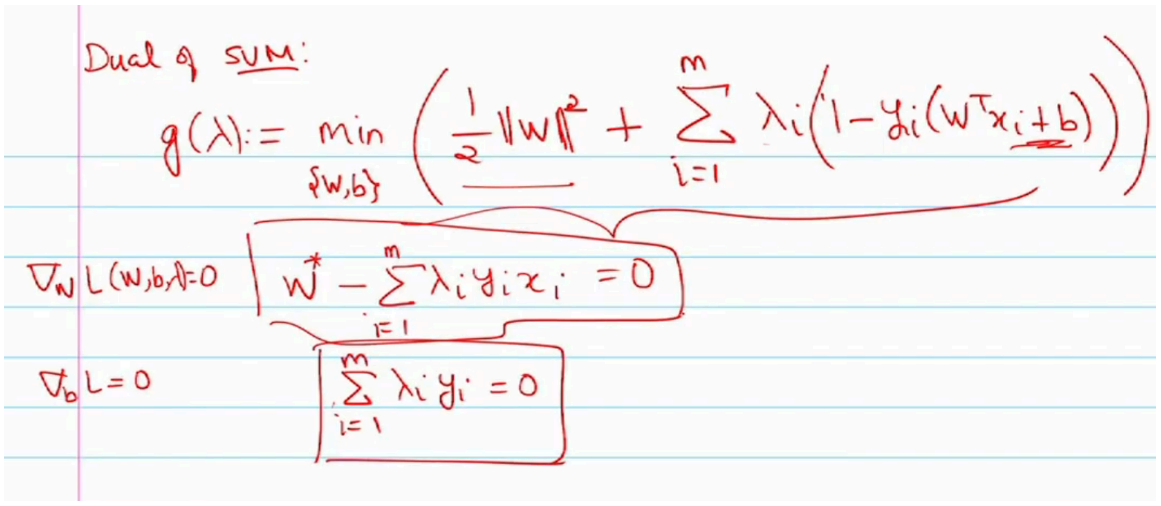

* Dual of an SVM:
w^Tx+b
-1
$+1$
(x_i, y_i)
$y_i=+1$
$\min_{\{w,b\}} \frac{1}{2}\|w\|^2$
s.t. $y_i(w^Tx_i+b) \geq 1$
$\forall i \in \{1,2,\ldots,m\}$

So, is everyone with me on this? Alright. So, let us try and come up with the dual for this optimization problem ok. So, first we define the like if you want to come up with the dual we need to define the Lagrangian dual for that right. So, that is going to be in terms of like because they are just the inequality constraints. So, we would be needing lambdas right and let us define g lambda to be So, w and b are the primal decision variables f of x which is half norm w square plus there are m inequality constraints because there are m points in the training set. So, we would have summation i equal 1 through n lambda i which is the ith coordinate.

Is everyone with me on this? Yeah. Is this clear to everyone? So, f of x plus lambda transpose h of x in this case there are m inequality constraints. So, you have m I mean basically lambda is m dimensional. So, lambda i times this particular term. Is it ok? Right. So, now we need to minimize this is an unconstrained minimization with respect to w and b.

So, how do we find the optimal w and b? Just take the derivative and set it to 0 because it is an unconstrained minimization. So, what is the minimum of when we try to minimize this with respect to w, what do we have? The derivative of the first term with respect to w is simply w and the second term is minus summation i equal. So, let us call it w star i equal 1 through lambda i y i x i transpose here. This is equal to 0, is this clear? So, it is w transpose w right, half of it. So, it will be if you take the derivative with respect to w, it will be simply w.

Is this clear? So, that is when you set the derivative of the Lagrangian with respect to w and you set it to 0 that is what you get. But the other decision variable is what? b. So, we also need to set the derivative with respect to b and set it to 0. So, that means gradient with respect to b of this Lagrangian that also needs to be set to 0 and what does this give us? So, if you set the derivative with respect to b and to 0 you get i equal 1 through m lambda i y i is equal to 0 ok. So, I have this constraint and this constraint and let us make this constraint into the let us use this to evaluate g lambda.



Dual of SVM:

$$g(\lambda) := \min_{\{w,b\}} \left(\frac{1}{2}\|w\|^2 + \sum_{i=1}^{m} \lambda_i \left(1 - y_i (w^T x_i + b)\right) \right)$$

$$\nabla_w L(w,b,\lambda) = 0 \qquad \boxed{w^* - \sum_{i=1}^{m} \lambda_i y_i x_i = 0}$$

$$\nabla_b L = 0 \qquad \boxed{\sum_{i=1}^{m} \lambda_i y_i = 0}$$

because we are trying to minimize g lambda with respect to w and b. So, let us use this to minimize g lambda. So, what does this give us? g lambda is equal to, now I am using minimum value of w star and b star. So, let us see. So, half of w transpose w right and w star by definition is this particular term.

So, that basically gives us i equal 1 through n lambda i y i x i transpose I mean i is just a dummy variable. So, I can use j lambda j y j x j half w transpose w that is the first term is this clear plus summation i equal 1 through m lambda i that is the first term over here minus i 1 through m lambda i y i and you have w transpose x i here and I can again replace w write w in terms of summation j 1 through m lambda j y j x j transpose x i ok, w transpose x i is what I am writing and then you have minus b. summation i equal 1 through m lambda i y i b i right or lambda i y i is this clear all goods of everyone ok with this. So I am just replacing the value of w star, just writing it, just replacing it over here because I need to find the optimal value of this particular term, this particular Lagrangian. So I am just replacing w star using this particular constraint.

This is going to be 0 because we have just derived this particular thing. So this term is going to be 0. What about these two terms? This is half of this term and it is minus 1 of the same term. So, this means g lambda is simply summation i 1 through m lambda i minus half of summation i comma j 1 through m, you get lambda i lambda j y i y j and you have x i transpose x j which I can write this is in a product between x i and x j.

$$g(\lambda) = \frac{1}{2}\left(\sum_{i=1}^{m} \lambda_i y_i x_i\right)^T \left(\sum_{j=1}^{m} \lambda_j y_j x_j\right) + \sum_{i=1}^{m} \lambda_i$$
$$- \sum_{i=1}^{m} \lambda_i y_i \left(\sum_{j=1}^{m} \lambda_j y_j x_j\right)^T x_i - b \cancelto{0}{\sum_{i=1}^{m} \lambda_i y_i}$$

So, this is the this is your g lambda ok. And the dual problem is maximize g lambda, lambda in R m subject to you have this constraint right summation i equal 1 through m lambda i y i equal to 0 and lambda a is greater than equal to 0. and this is your dual of SVM. Now something interesting is what you notice over here and this also gives you an idea as to why dual dualization of this particular problem is important. So, in this case here it is easier to find a linear hyperplane that separates these two sets of points, but what if what if the distribution of points is something like this. So, you have positively labeled points at the periphery and negatively labeled points like this.

$$\boxed{g(\lambda) = \sum_{i=1}^{m} \lambda_i - \frac{1}{2} \sum_{i,j=1}^{m} \lambda_i \lambda_j y_i y_j \langle x_i, x_j \rangle}$$

$$\boxed{\begin{aligned} &\max_{\lambda \in \mathbb{R}^m} \; g(\lambda) \\ &\text{s.t.} \; \sum_{i=1}^{m} \lambda_i y_i = 0 \, , \; \lambda_i \geq 0 \end{aligned}} \longrightarrow \text{Dual of SVM}$$

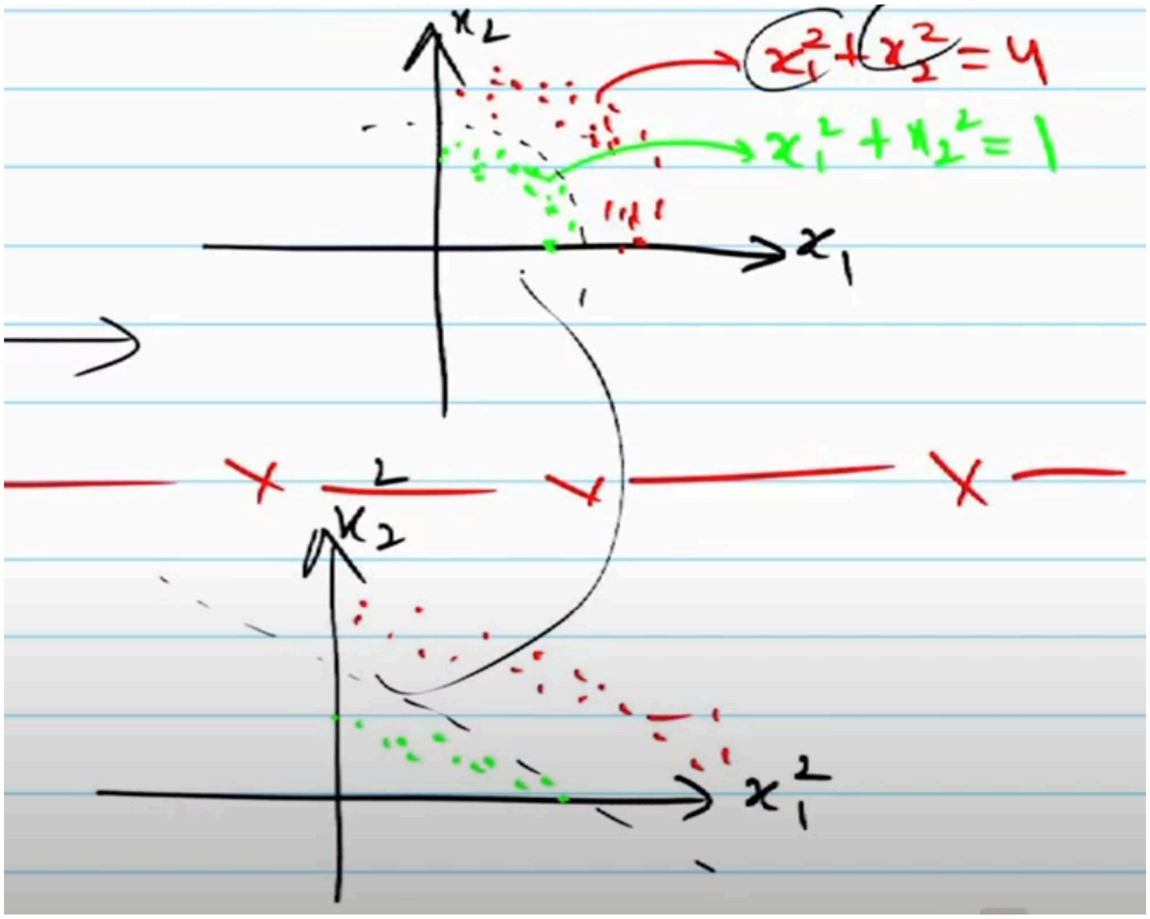
And the question is can you find a separating hyperplane that separates these two sets of points sorry a linear separating hyperplane. no right. So, in fact I mean it is in this case for instance it is impossible to separate these two set of sets of points in R 2 or in this, but using kernel function you can sort of map this to an abstract higher dimensional space where these points become linearly separable right. And you can simply replace this by kernel of x i and x j. So by choosing the appropriate kernel you can separate even non linearly separable data points like this.

So you can use polynomial kernel, RBF kernel, Gaussian kernel and so on. But you can see then in the dual formulation you just replace this inner product x i and x j with the kernel of x i and x j and then you are done. Which is not possible by the way when you are doing working with that primal formulation right. So, there is a direct advantage of working with the dual version of the SVM because you can work with non-linearly separable data points as well. And in fact in your in your assignment column I am going to give you I am going to have you basically work with a non-linear SVM essentially use

kernel to be able to separate non-linearly separable data points.
Yeah sometimes this dual view of the primal form basically it helps you with lot of such important details. All right any questions on dual formulation? So, it can be like a polynomial like depends on what kind of kernel you choose right. So, for instance the so, the idea behind kernel is for let us consider a simple setting I think it would be easier to understand through. So, you have two dimensional x 1 x 2 let us say you have two dimensional like you are working in R 2 and you have like a quarter of a circle kind of this and you have another quarter think of it as if like this particular these particular set of points are distributed across this circle x 1 square plus x 2 square is equal to 4 and may be this one is x 1 square plus x 2 square is equal to 1 something like this right. Now it is difficult to find I mean and maybe if they are really far apart you can still find a linearly separable hyper like sort of a linear separating hyperplane that separates these two set of points.

But the idea is better solution would be to find well not a hyperplane, but some like non-linear hyperplane which separates these two sets of points like this right yeah. No, solving this dual will give you lambda, right? So, you can get w from here, right? And you know that if the problem is, so you know the value of g lambda, right? optimal. So, how do you get the b? So, b would be yeah. So, we will get to. So, I have not covered something called KKT conditions which I was going to come to.

So, we will get to that point. So, as I was telling. So, you want to find this kind of hyperplane right and you see that it is not possible in the original access system. What I instead do is I define a new access system where instead of working with x 1 and I project these set of points in a new coordinate system where instead of working with x1 and x2, I have x1 square and x2 square. So if x1 square and x2 square are my variables, these are nothing but equations of lines. So these points are now going to be distributed like this in a straight line. And now in this new coordinate system, I can find a linearly separate like basically a hyperplane which separates these two sets of points a linear hyperplane and then I sort of project this back to the original coordinate system and that is what kernel does the kernel trick does ok.



And so you are trying to I mean so that kernel kind of maps it to an abstract higher dimensional space where these data points become linearly separable ok alright. Thank you very much.