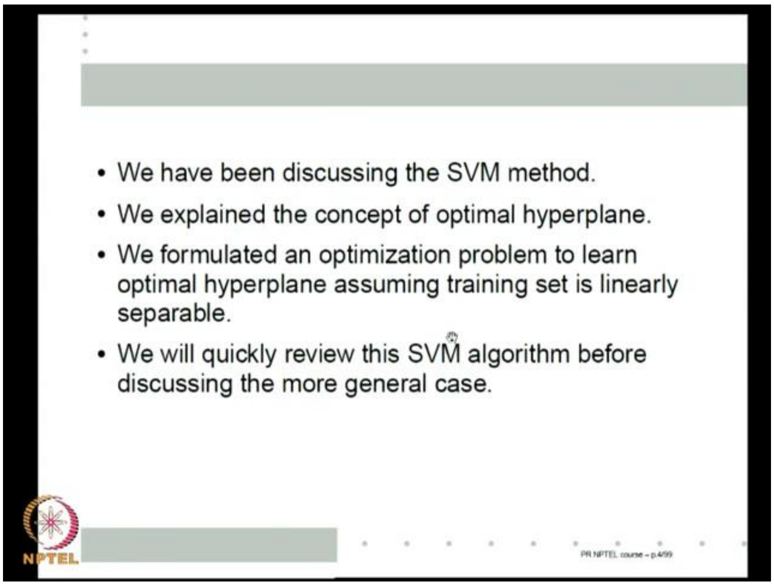
Pattern Recognition
Prof. P. S. Sastry
Department of Electronics and Communication Engineering
Indian Institute of Science, Bangalore

Lecture - 33
SVM formulation with slack variables; non-linear SVM classifiers

Hello and welcome to this next lecture on pattern recognition, we have started discussing the support vector machines from last class. As you already discussed this is an approach where the idea is that we effectively transform the feature vectors into high dimensional space and learn a linear classifier there, because you already know how to learn a linear classifier and learning linear classifier is in general more efficient. We want to learn a linear classifier in a high dimensional space, which in the original space will be a non-linear classifier that is the basic idea. We said that to for the idea to work, we actually do not learn any linear classifier, but learn what we called an optimal separating hyperplane. So, to briefly review we have been discussing the SVM method.

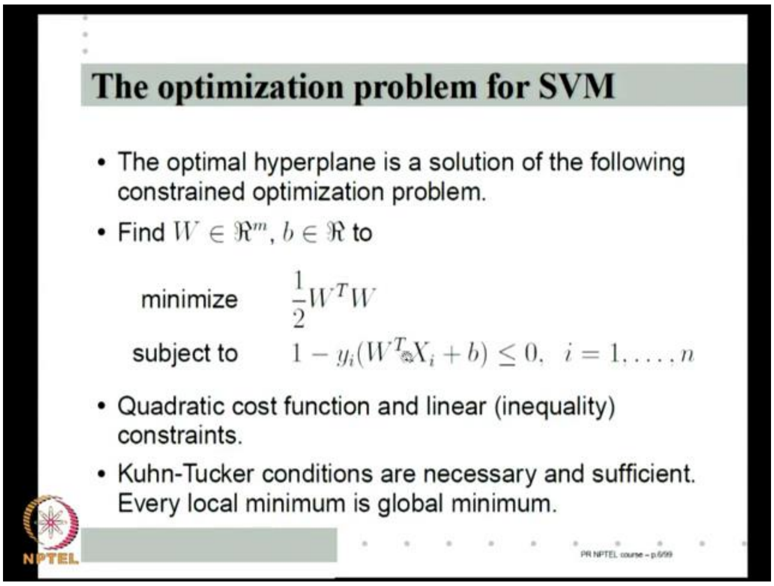
(Refer Slide Time: 01:08)



The one major new concept is the optimal hyperplane, right? We formulated the optimization problem of how one learns optimal hyperplane. We seen that learn for learning optimal hyperplane, we essentially have to solve a quadratic optimization problem with linear constraints. Basically, the optimal hyperplane is one which maximizes the separation and we looked at the formulation only under the assumption that the training set is linearly separable that is the case we have been considering so far.

So, what we will do in this class is we quickly review the, we start with the optimization formulation again quickly go through the dual and then move on and see how we can do the same formulation in general even when the training set is not linearly separable

(Refer Slide Time: 02:12)



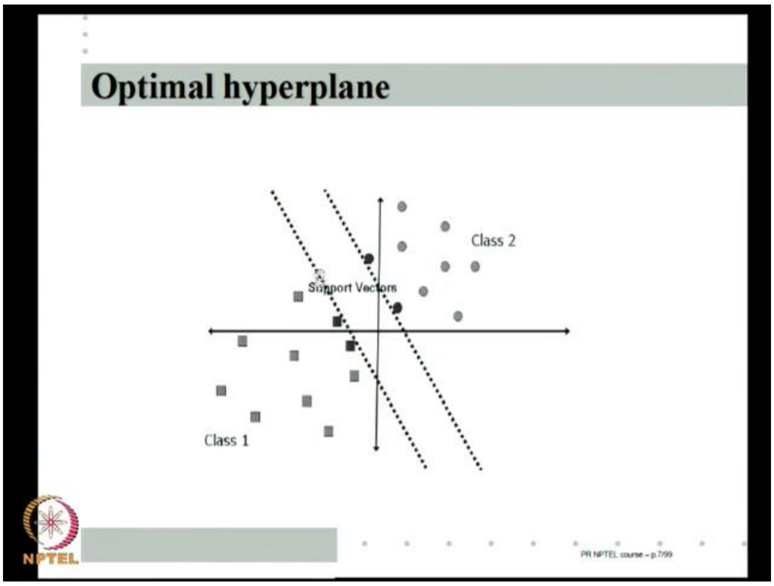
So, the optimal hyperplane is a solution to this optimization problem, what is the optimization? It is a constraint optimization problem namely to minimize transpose W subject to 1 minus y i into W transpose X i plus b less than or equal to 0. Optimization variables are W and b, find W and b to minimize half W transpose W subject to this constraints. What does this constraint imply? As we have already seen if W b, W comma b represents a separating hyperplane, then y i into W transpose X i plus b will be equal to 1, we will be greater than or equal to 1.

So, essentially y i into W transpose X i plus b greater than or equal to 1, which is same as 1 minus y i into W transpose X i plus b less than or equal to 0, tells you that this W b is a separating hyperplane and the separating hyperplane is such that W transpose X i plus b is equal to minus 1 and W transpose X i plus b is equal to plus 1. The two parallel hyperplanes that we considered last class no pattern is in between, so a separating hyperplane which, for which the margin of separation is given by 1 by norm w, that is any W b that satisfies this constraint will be a separating hyperplane, for which the closest pattern is 1 by norm W away.

Hence among all such W b if I minimize this, then I am finding a separating hyperplane, which has the maximum margin of separation meaning the closest training pattern is far away from for this hyperplane compared to any other separating hyperplane. So, that is why it is a constrained optimization problem among all separating hyperplanes that is a all W which satisfy this constraint, find the one which has the maximum margin of separation this is the optimization problem.

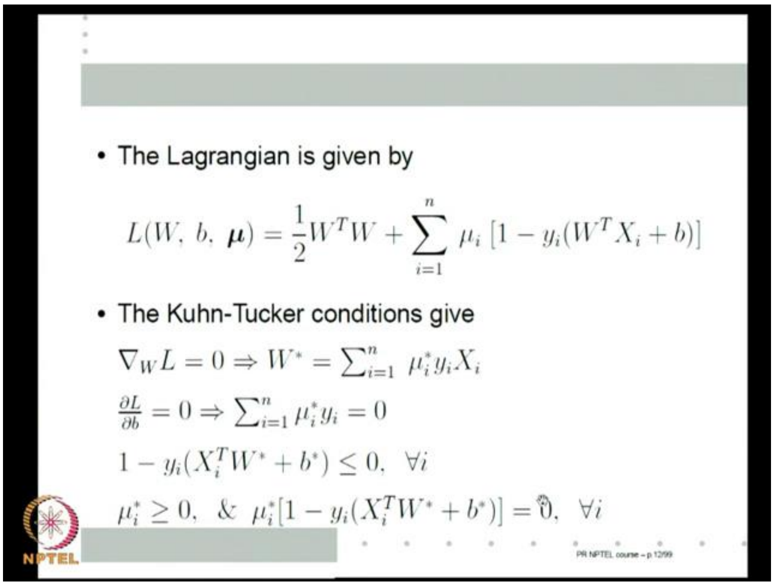
So, the optimal hyperplane is a solution to this problem and this is a, this this is the cost function, so which is quadratic, convex quadratic, all constraints are linear. So, this is the most efficient constrained optimization problem one can solve, for this problem because this is convex and constraints are linear, Kuhn Tucker conditions are (( )) and sufficient every local minimum is a global minimum. We have seen all the background of constraint optimization last class, so let us quickly write down the Kuhn-Tucker conditions for this problem.

(Refer Slide Time: 04:40)



Before that let me give you the as I have been telling you that these constraints essentially mean that you know if the hyperplane is this at the centre, then on either side i have got these two parallel hyperplanes and this is the margin of separation. So, what the optimal hyperplane is doing is that finding that hyperplane which has the highest margin of separation, that is what my optimization problem. I do not want a separating hyperplane like this, but I want a separating hyperplane like that, a separating hyperplane for which the closest pattern is more distance away and this. So, this is a non optimal hyperplane whereas, that is a optimal hyperplane. So, what is what is my Lagrangian for this problem? So, Lagrangian is my cost function plus sum of all the constraints you have to Lagrangian multiplies into constraints.

(Refer Slide Time: 05:38)



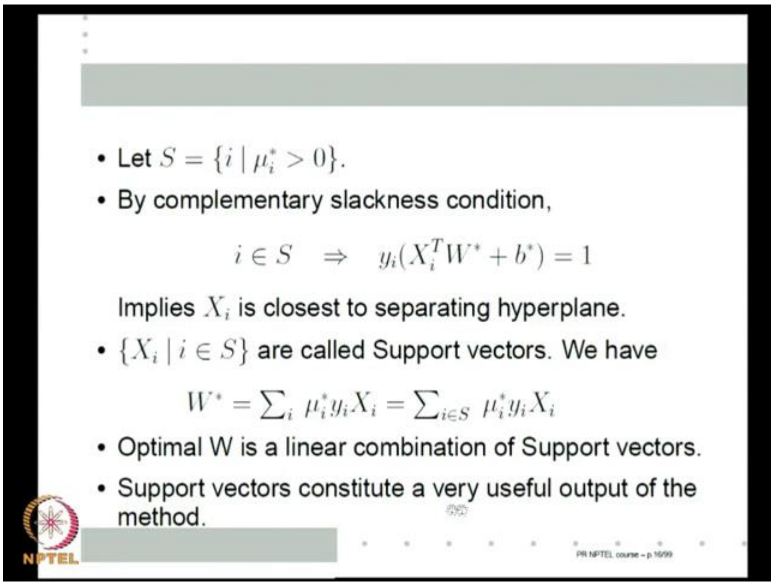
So, my Lagrangian will be half W transpose W plus sum over i mu i into 1 minus y I W transpose X i plus b, mu i are the Lagrangian multipliers. So, it is a function of W b the optimization variables and all the mu i which are the Lagrangian multipliers. So, what are my Kuhn Tucker conditions? The gradient of l with respect to the optimization variable should be 0, so if I equate gradient with respect to W to 0, if I take W k I get W here and from here there is one linear W term, so i get mu i y i X i. So, essentially at the optimal w, W star is summation mu i y i X i for all the optimal variables we put star. So, mu i, star is the optimal values of the Lagrangian multipliers, W star is the optimal value of W, this the normal to the separating hyperplane.

So, from gradient of l with respect to W is 0, gives me that W star is this, gradient of l with respect to b 0, gives me there is only one b term, but that is linear. So, I get i is equal to 1 to n mu i y i into b, so differentiate it I get summation i is equal to 1 to n mu i y i is equal to 0. This is the condition that the gradient of the Lagrangian with respect to optimization variables is 0, then we need feasibility, so all constraints should be satisfied. So, 1 minus y i into X i transpose W star plus b star is less than or equal to 0, for all i and then I have complementary slackness, so i i first I want all the lagragian multipliers to be positive or non negative and mu i star into the constraint.

As I have already mentioned last class our optimization problem is such that there is one constraint for each example, essentially because I want W b to be a separating hyperplane. For each X i y, each example X i y i it should be on the right side of the W hyperplane, so each constraint is straight to one example, each example has one constraint, so each constraint is a corresponding X i. In the, in my Lagrangian each constraint will have its own Lagrange multiplier, so my my each Lagrange multiplier is tied to a particular example.

So, I can talk of Lagrange multiplier for that example, because each Lagrange multiplier is for a constraint and each constraint is for an example. Now, this complementary slackness gives us something very interesting as we seen last class. Let us say s is the set of all indices for which these corresponding Lagrange multipliers are strictly positive, you know Lagrangian multipliers have to be greater than or equal to 0, some of them might be 0, some of them might be strictly positive, s may be the index such that Lagrange multipliers are strictly positive.

(Refer Slide Time: 08:13)



Then the complementary slackness says if this is strictly, this strictly greater than 0 then this has to be 0, because the product has to be 0. So, if i belongs to s then y i into X i transpose W star plus b star is equal to 1, what is that this mean? This means that the the

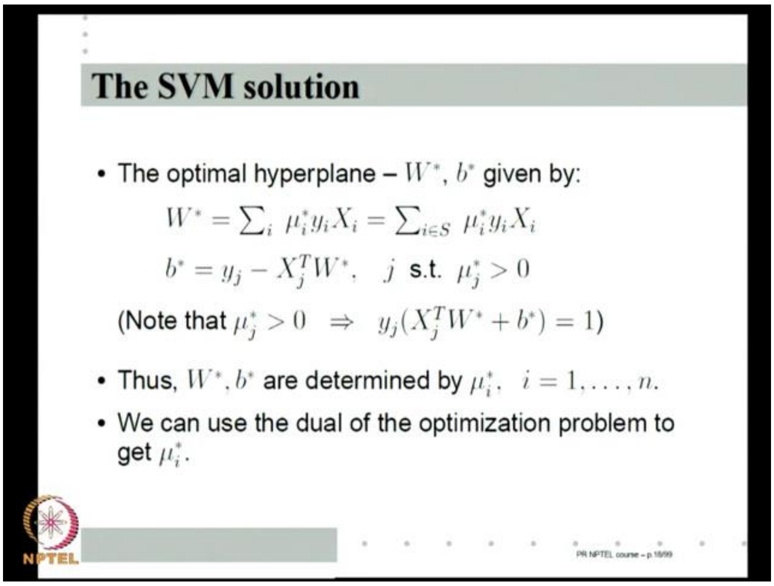
X i is on the hyperplane W star transpose X i, X plus b is equal to plus 1 or minus 1 depending on whether y is plus 1 or minus. So, is on one of the two parallel hyperplanes, which means that X i is a pattern that is closest to the separating hyperplane. So, the X i corresponding to the non non zero Lagrangian multipliers or the training patterns which are closest to the separating hyperplane, right? The complementary slackness says if mu i star is greater than 0 then this has to be 0, right?

So, if I think of s as the set of all Lagrange multipliers, in this of all Lagrange multipliers which are strictly positive, then for all those i I know this is satisfied which means all the corresponding X i are closest to the separating hyperplane. We will call such X i X i corresponding to positive Lagrangian multipliers as support vectors, these are training patterns they are closest to the separating hyperplane. These are the the my optimal W star I already know is summation or i mu i star y i X i, so if this summation need be taken only over those mu i star which are strictly greater than 0. So, essentially my W star is a linear combination of these support vectors, if I define this as the support vectors.

Also another thing that should remembered is that for my optimization problem, if mu i star is equal to 0 then the constraint is not active, so which means essentially these X i, X i corresponding to the support vectors are the critical X i. Because if I take away all the other examples and leave only these I still get the same separating hyperplane, the optimization problem will not change. That is why W star is a linear combination of support vectors, we have already seen while discussing perceptrons that any separating hyperplane will be a linear combination of X i, but for the optimal separating hyperplane is a linear combination of some of the X i not all, only the what we call the support vectors and the linear combination involves the corresponding Lagrange multipliers.

So, optimal W is a linear combination of support vectors and in this sense support vectors constitute a very useful output of the method, right? For this problem so these are the support vectors, the pattern that are darken they are closest to the supporting. So, as you can see geometrically if I throw away all the remaining data and just keep the support vectors right for that pattern classification problem also this will be the best hyperplane. So, in some sense support vectors are the most critical examples of this pattern set of this example set, thus as we shall see later on much more clearly, support vectors constitute a very useful extra output of the method in addition to getting the W star like this.

(Refer Slide Time: 11:57)

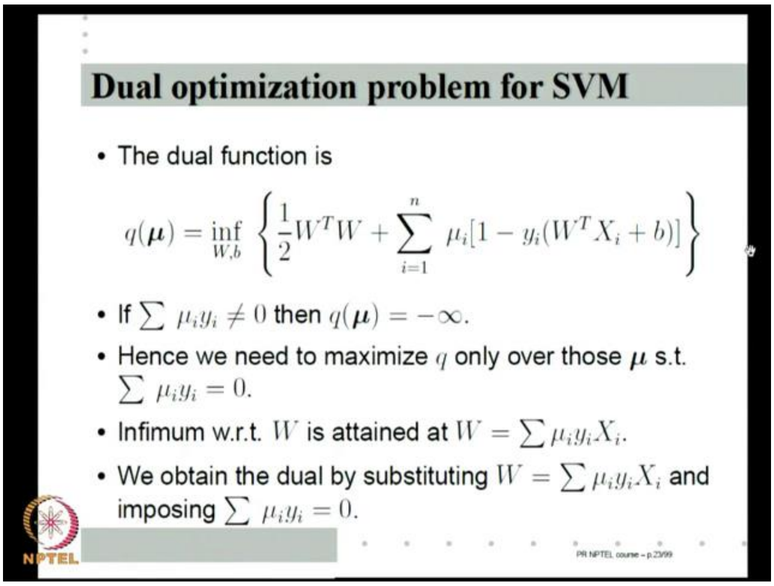


So, let us sum up the SVM solution, SVM tries to learn the optimal hyperplane, optimal hyperplane is defined to be one that maximizes the separation, is a separating hyperplane which has the maximum separation. The optimal hyperplane is given by W star transpose X plus b is equal to 0, then the W star and b star are given by this, W star is summation over i, mu i star y i X i and b star how do I know b star? From the complementary slackness, whenever mu j star is greater than 0 I know y j into X j transpose W star plus b star is equal to 1. Because my complimentary slackness tells me that whenever the mu is Lagrange multiplier is 0 this is equal to 1.

When this is 1 b multiply both side y j, I remember y j square is equal to 1 because y j is plus 1 or minus 1, so that gives me b star is y j minus X j transpose W star right. So, once I have all the mu i stars I can calculate W star and second calculate W star and have all the mu mu stars I can take any j at that mu j star is greater than 0 and b star will be this. Theoretically does not matter which j you take you should get the same value of b star, so that W star and b star are determined by your mu i stars your Lagrangian multipliers.

So, if I can get the optimal Lagrangian multipliers for this problem I am done and because I essentially want to get Lagrangian multipliers one way of doing solving this problem, one way of getting the optimal hyperplane is to find mu i star by solving the dual of the optimization problem. We have seen in last class how to find the dual, by the dual we find the dual function which is the infimum of the Lagrangian and then maximize the dual function.

(Refer Slide Time: 13:50)



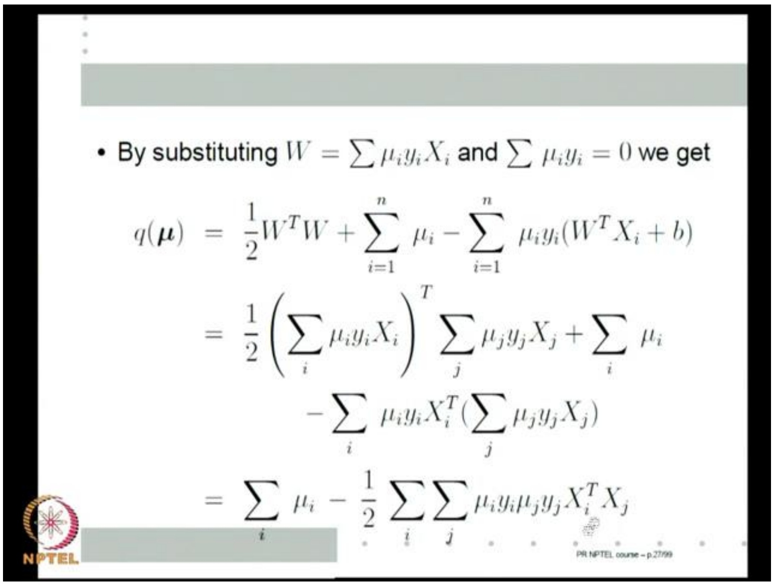
So, what is the dual function for this problem? It is a function of the Lagrangian multipliers q of mu, which is infimum over the optimization variables W b of the Lagrangian of W transpose W plus i is equal to 1 to n mu i into 1 minus y i into W transpose X i plus b. I have to find infimum of this with respect to both W and mu, so for a given mu I have to find infimum of this with respect to W and mu that becomes the value of q mu. So, ultimately I get a function of mu and that is what I have to maximize subject to some constraint the constraint that mu i are positive that is what the dual is.

Now, if I want to find infimum, first note is that there is a term here summation i is equal to 1 to n W i, summation i is equal to 1 to n mu i y i b with a negative sign in front, but that does not matter. So, there is a term here that is b times summation i is equal to 1 to n mu i y i. Now, I am finding infimum with respect to b so if summation mu i y i is not equal to 0, then infimum with respect to minus infinity I can take b to be sufficiently large positive or negative as the case may be to keep decreasing q mu, right?

So, if summation for a given mu I am finding the value of q at a given mu for a given mu if summation mu i y i is not equal to 0, then for that mu the value of q mu will be minus infinity, because the infimum minus infinity I am not interested on those mu. Because I am maximizing q mu so a mu at which q mu takes value minus infinity is not a place where maximum will be attained. Hence we need to maximize q only over those mu which satisfy summation mu i y i is equal to 0. I am not interested in finding value of q mu for those mu for which mu i y i is not equal to 0 for those mu any way i know is minus infinity.

So, we need to maximize q only over these over mu are there that summation mu i y i is equal to 0, so once I take summation mu i y i is equal to 0 the b term drops out. Now, I need to only find the infimum of this with respect to W, that means I have to differentiate with find the gradient with respect to W equal to 0 which we already done earlier. So, I know infimum with respect to W is attained at W is equal to summation mu i y i X i because this X differentiate with respect to W and equate with 0. So, how do I get q mu for a given mu now? For that mu using that mu I write W is equal to mu i y i X i and now I substitute in this expression W is equal to that mu i y i X i and impose mu i commissioned mu i y i is equal to 0 then i get q mu.

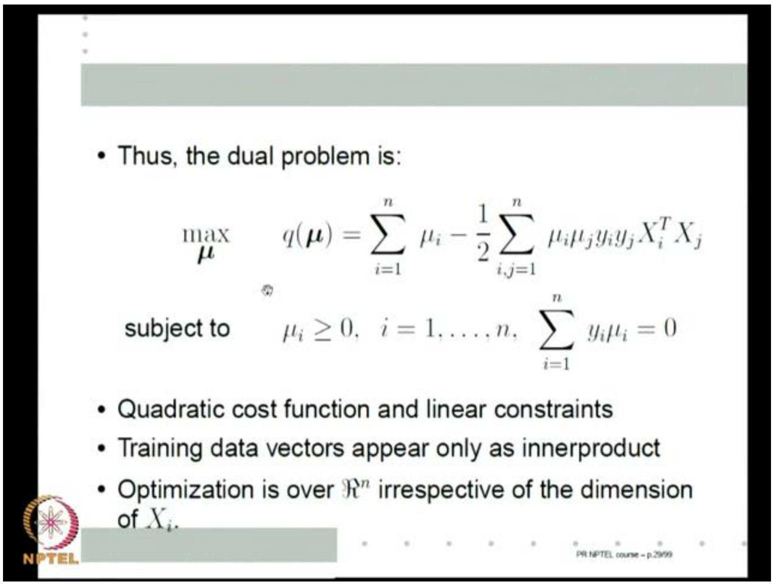
(Refer Slide Time: 16:27)



Let us calculate q mu, this is what we have to do, we have to substitute W is equal to this and my q mu is this, now I am taking an infimum, because I know how what to do for infimum or just to do the substitutions. My q mu is W transpose W mu i into 1 minus this I have written as summation mu i and the second term because this term mu i y i W transpose t X i plus b. So, this mu i y i b will go to 0 because I have to put mu i y i is equal to 0, so i have to put W is summation mu i y i X i. So, this W is summation mu i y i X i whole transpose, this W i once again this i is a dummy index. So, I have put this W as summation over j mu j y j X j plus summation mu i minus mu i y i b has gone, mu i y i i write this as X i transpose W x i transpose W is once again summation mu j y j X j.

So, both these summations are same I have this summation of i and j here mu i y i mu j y j X i transpose X j here also mu i y i mu j y j X i transpose X j this is minus 1 this is half. So, I will get a minus half. So, q mu is summation mu i minus half summation over i and j mu i y i mu j y j X i transpose X j that is my q mu.

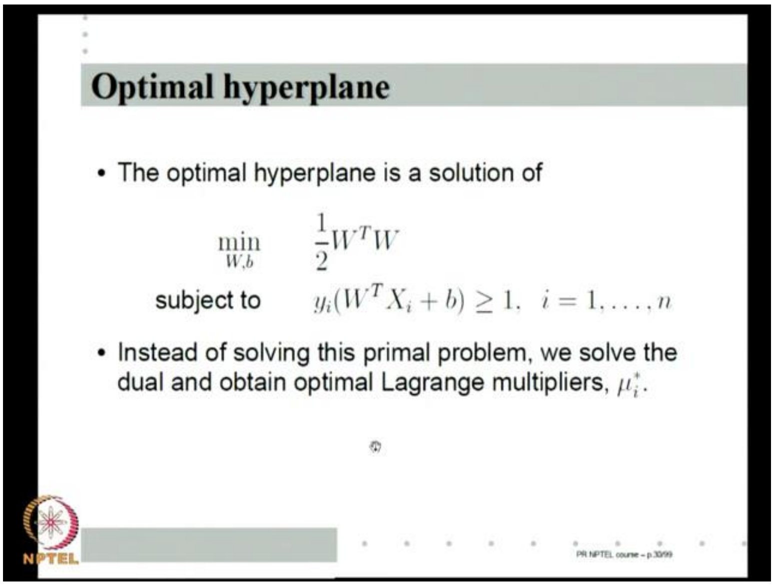
(Refer Slide Time: 17:55)



So, my dual now turns out to be maximize q mu which is given by this as we have just now seen subject to mu is greater than or equal to 0 and we have to impose the extra condition mu i y i is equal to 0, right? So, this is my dual problem this is also a constraint optimization problem, the optimization variables are mu, right? The cost function is this because my variables are mu this is a quadratic function; this is this term is linear in mu; this term is quadratic in mu. So, this is a quadratic function, constraints are linear mu i is greater than or equal to 0 summation of y i mu mu i y i is equal to 0.

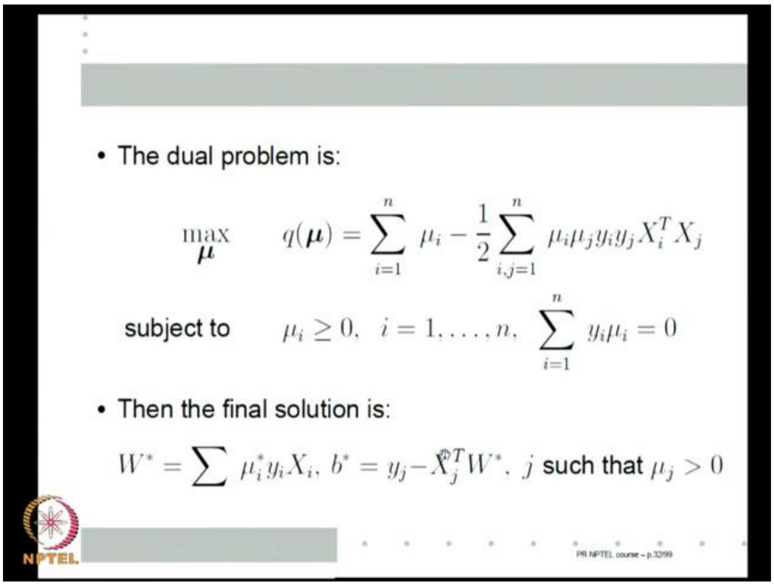
So, it is once again a constrained optimization problem with quadratic cost function and linear constraints. The training data vectors X i X j unlike in the primal way they appear inside the constraint they appear only in the objective function, there also only as a inner product X i transpose X j, right? The optimization variables here are mu how many are there n, right? As many examples 1 mu per example so there are n examples, so there are n Lagrange multipliers and so the dimensionally the optimization problem is n irrespective of the dimension of x, no matter what the dimension of X is dimension of this optimality is optimization problem is equal to the number of examples. We will just remember these things for later use.

(Refer Slide Time: 19:19)



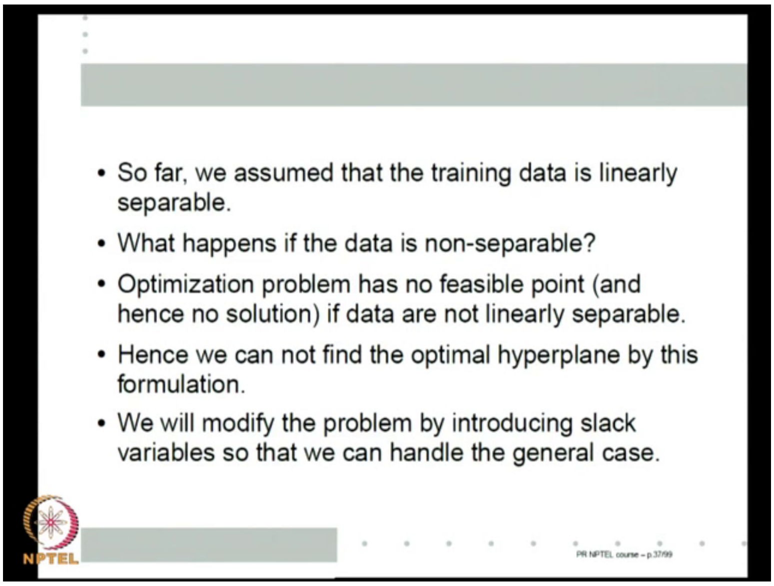
So, sum up the optimal hyperplane is a solution of this problem, this optimization problem we call this the primal problem. Instead of solving this primal problem we solve the dual and obtained optimal Lagrange multiplier.

(Refer Slide Time: 19:32)



This is the dual problem, we solve this dual problem so when we solve this we get all the mu i and once we get all the mu I, this is my final solution. What do you mean by W star b star is my solution? Now, you give me any X then I calculate W star transpose X plus b if this is positive I put it in class plus 1, if it is negative I put it in class minus 1. So, W star b star gives me my optimal hyperplane, so from now on give me any new feature vector X I will say I will calculate W star transpose X plus b star and sign of that will give me the class. As I said we will we are considering only two class problem, so this is the full SVM method for linearly separable case. So, we define the optimal hyperplane to be this, then that is the dual and that is my final solution.

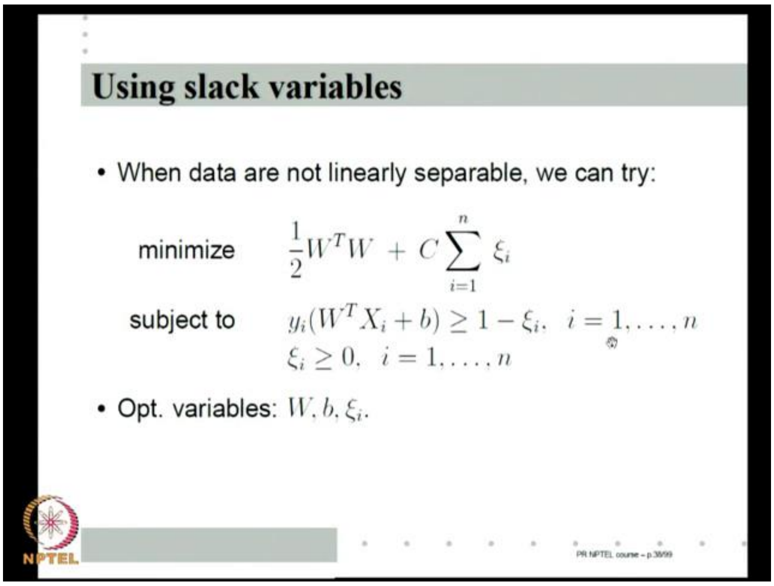
(Refer Slide Time: 20:23)



Now, in all this we have assumed that the training data is linearly separable, what happen in training data if it is not separable? What do you mean by training data not separable? If the training data is not separable, what it means is there will not epsist a W on b to satisfy these constraints. These constraints are satisfied by a W b only if W b a separating hyperplane, if there does not epsist a separating hyperplane because training data is not linearly separable, then there will not be a W b that satisfies the constraint. That means there is no feasible, so feasible point for this optimization problem.

There is no feasible point for the optimization problem obviously there is no optimal point. So, if the data is not separable the optimization problem is no feasible point and hence no optimal, so no solution, right? Which means in general if the data is not linearly separable we cannot find the optimal hyperplane by this formulation, so we need to change this formulation, so that we can. Basically, why are we not able to find a solution? Because my constraints may not be satisfied, whenever constraints are not fully satisfied the standard technique in constraint optimization we used to use what are called slack variables. So, that we will keep our constraints not hard, but soft so to say and then we we we mange to get a feasible point, so that is what we will do next.

(Refer Slide Time: 21:42)

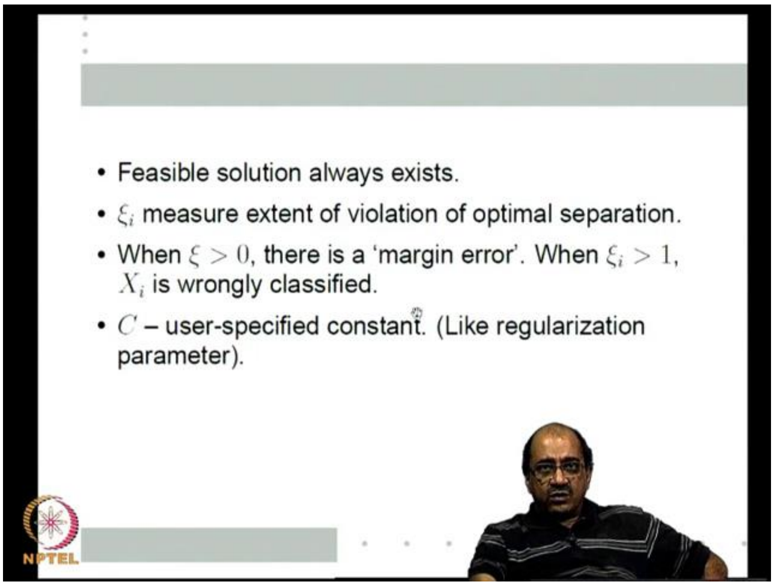


So, what we do is for general case, normally what is that we want y i into W transpose X i plus b greater than or equal to 1, if that is satisfied forget about the psi i right now, we actually wanted y i into W transpose X i plus b greater than or equal to 1 for separation, but suppose there is no W b that can satisfy this. Then I say if you cannot get it greater than 1 get it at least greater than 1 minus psi i, right? Where psi i some positive number, so essentially psi i measures a penalty of how much you are unable to satisfy your required separability constraint.

If I did not have this minus psi i y i into W transpose X i plus b greater than 1 is my separability constraint, I am unable to get W b so that for each i it is definitely greater than or equal to 1, I said at least get it greater than or equal to 1 minus psi i for some slack psi i. Now, I do not want too much slack if I have too much slack obviously you know you will take W to be 0, right? Because you want to minimize this you take W to be 0 and put everything into this slack, so I do not want too much slack, so the psi i are the slack variables I will penalise psi i. So, if psi i are large this term will be large, I am minimizing not only half W transpose w, but also summation i is equal to1 to n psi i, right?

So, my optimization variables now are W b and psi i, find W b and psi i, psi i is psi 1, psi 2, psi n. So, that I added the psi i here, but I do not want too much slack, so I added a penalty here. So, the first thing about this is that a feasible solution always is this you give me any W b, I put a corresponding y i and X i here for each of the i. If it is greater than 1 I am fine I will take psi i to be 0, if it is not greater than 1 whatever is this slack I need, I take that into psi i, right? So, any W b is feasible now, because I can always find psi i to satisfy this, right?

(Refer Slide Time: 23:28)

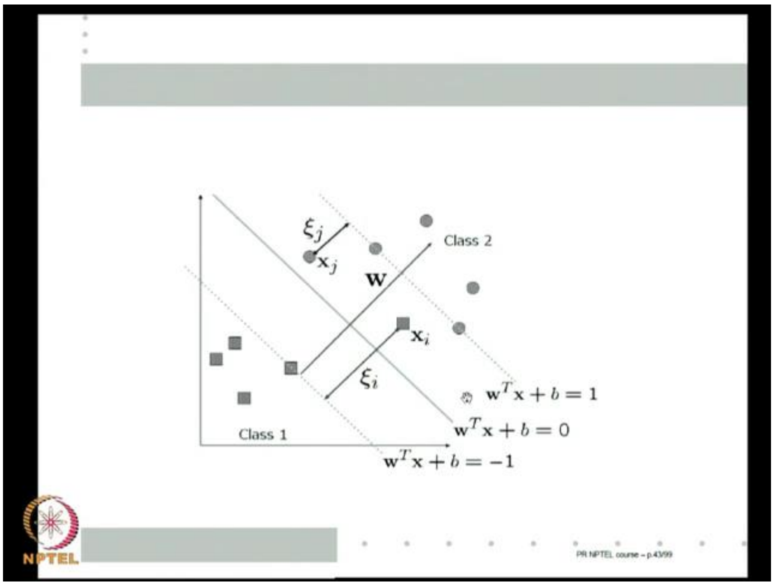


So, there is no problem about finding the feasible point, feasible solution always exist any W b is feasible. psi i measures the extent of violation of the optimal separation that also we seen. So, basically if psi i is greater than 0 then there is a margin error, if psi is greater than 1 X i is wrongly classified you see. If psi i is greater than 0 what it means, W y i into W transpose X i plus b is is not greater than or equal to (( )) it might be only greater than or equal to 0.5, even if it is greater than or equal to 0.5 the sign of W transpose X i plus b is same as sign of y, right?

So, I am not getting the optimal separation because we wanted so much separation, we do not, did not want any training pattern between W transpose X plus b is equal to plus 1 and minus 1. That much separation I am not getting if psi i is greater than 0, but if psi i is greater than 1, that means this sign is changing which means I have not even classifying it correctly, right? That is that is how psi i measure the X under violation.

Now, finally, what we are minimizing is my margin plus this some of these slack variables they tells you how much violation of separation I am doing and C is a kind of a exchange rate constant. How much of increased margin of separation am I willing to take at the expense of you know making so many errors in classification to say. So, this tells me some measure of the error in optimal separation, so this C tells you how much of error I am willing to trade with how much of increase in margin, C is used as a different parameter much like regularisation constant later on we will see that it is actually regularisation constant, C is user specified.

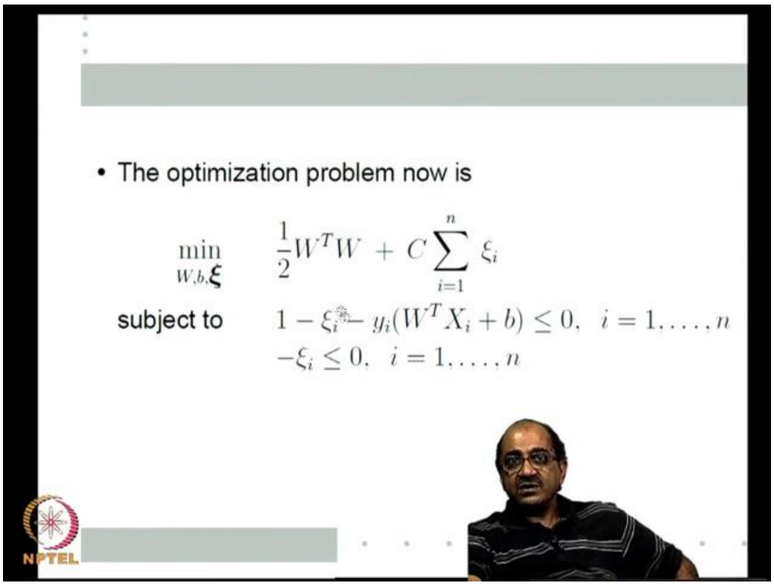
(Refer Slide Time: 25:34)



So, we can see this formulation what we are saying is this is W transpose X plus b is equal to 0, this is W transpose X plus b is equal to 1 W transpose X plus b is equal to minus 1. So, I take this to be my w, W on b then if a class two pattern came here, so for all the patterns which are beyond this anyway my psi will be 0. If a class two pattern came here between these two this psi j would be greater than 0, but less than 1 still on the right side. But I am actually counting this as my margin right because I have taken this W on b, I am counting this as my margin with respect to this much margin, this is one pattern that is not giving me that much margin, right?

On this side, on the other hand this pattern is actually on the wrong side of the hyperplane, right? If I want to take this as the separating, so here this psi I would be greater than 1 right, but essentially what it means is given this given this pattern is here and this pattern here, normally I would not have been able to find any separating hyperplane. What I decided to do is call these two patterns errors then I get a separating hyperplane with such a nice merge, that is what this formulation is. Separate bit a penalty by misclassifying a few patterns and then separate the rest of it; that is what I am trying to do.

(Refer Slide Time: 26:56)

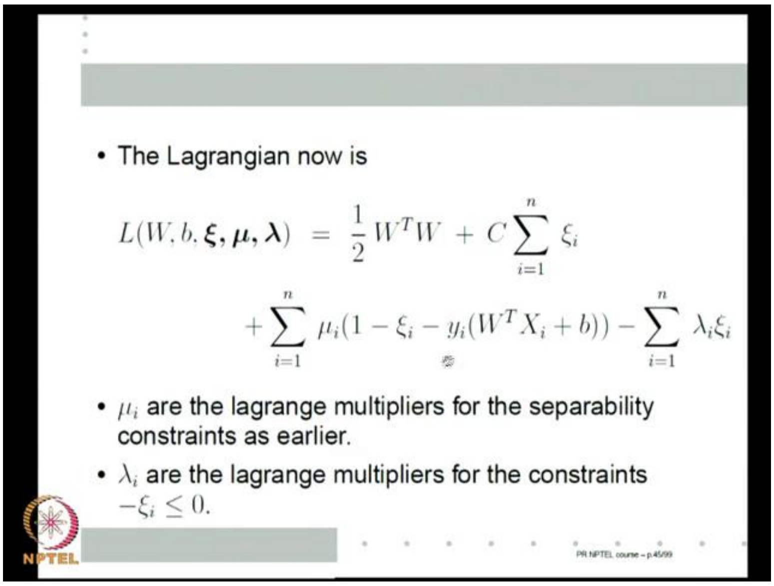


So, that is my optimization problem, right? Minimize over W b and psi i, so I put that as the all the psi i as the bold psi, half W transpose W plus C times is sum of slack for I have written this in now my standard optimization thing in optimization all my constraints have to be written as something less than or equal to 0. So, I wrote it as instead of writing y i into W transpose X i plus b greater than or equal to 1 minus psi i, I wrote as 1 minus psi i minus phi i into W transpose X i plus b less than or equal to 0.

Similarly, instead of writing psi i greater than or equal to 0 I wrote minus psi i less than or equal to 0, this is my optimization problem now. Now, what is my Lagrangian? Once again my Lagrangian will be a function of the optimization variable W b psi and there will be 1 Lagrange multiplier per constraint so there will be n Lagrange multipliers for these n constraints, call them mu 1 to mu n. There will be another n Lagrange multiplier for these n constraints, call them lambda 1 to a lambda n, then my Lagrangian will be a function of W b all the psi i and then mu i and lambda i, right?

Lagrange will be my cost function plus sum over mu i into this constraint plus sum over lambda into this constraint, right? So, my Lagrange will be W b psi mu lambda this is my cost function plus mu i times my 1 minus psi i into y i W transpose X i plus b plus lambda i times minus psi i, so I wrote it as minus lambda i psi.

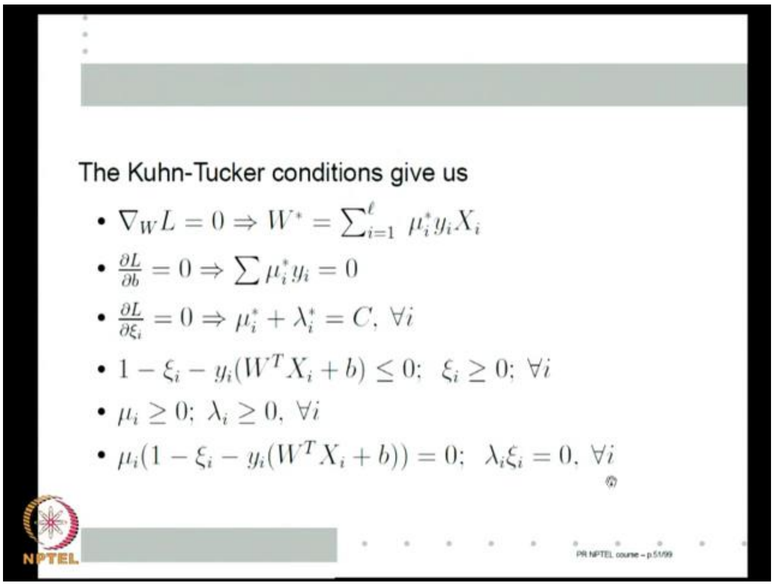
(Refer Slide Time: 28:37)



So, mu i are the Lagrange multipliers for this separability constraints, as earlier for each examples. Lambda is the Lagrange multipliers for the constraints minus psi i less than or equal to 0. So, this is my Lagrangian, so what will be my Kuhn-Tucker conditions? I have to get derivative with derivative with respect to W is equal to 0, derivative with respect to b is equal to 0, gain with respect to psi that means partial derivative with respect to each of psi i is equal to 0, that is the derivative constraints.

Then I need feasibility, all constraints should be satisfied and mu i should be greater than or equal to 0, lambda i should be greater than or equal to 0 and there are complimentary slackness. So, let us write all our Kuhn Tucker conditions, first is gradient of W gradient of l with respect to W is equal to 0, so as W the function still the same it has not changed. So, I get one W from here and I get mu i y i X i here right? So, gradient of W is equal to 0 still gives me the same old thing W star is summation over i mu i y i X i.

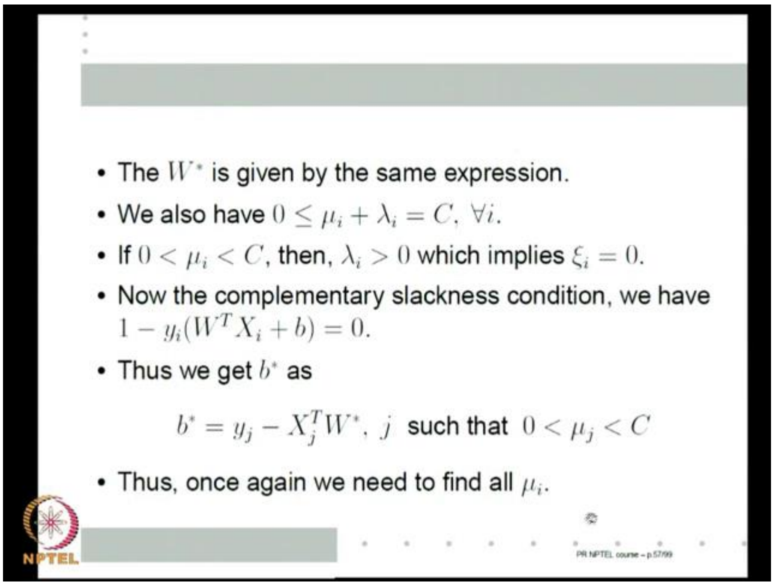
(Refer Slide Time: 29:38)



Similarly, with respect to b I get the same old thing, what will it give me if I differentiate with respect to each of these psi i? There is one psi i here, so we get a C term plus C here i get a minus mu i from here, minus lambda i from here. So, c minus mu i minus lambda is equal to 0 for each i, which is same as mu i plus lambda i is equal to C for each i. So, this this the derivative with respect to remaining optimization variables will tell me mu i star plus lambda i star is equal to C for every i, then i have the feasibility constraints all my psi i have to be positive.

These are my other constraint 1 minus psi i into y i W transpose X i plus b less than or equal to 0 the separability constraints. All my Lagrange multipliers have to be non zero, non negative and my compliment is slackness, mu i into this constraint should b equal to 0. Similarly, lambda i psi i should be equal to 0, these are my full complement of Kuhn-Tucker conditions for this optimisation problem.

(Refer Slide Time: 30:52)



Let us see what this means, W star is still given by the same expression as you have already seen, we also have the extra condition mu i plus lambda i is equal to c for every i and because both mu i and lambda i are non negative mu i plus lambda i will also be non negative, so we have this for every i. What does this mean? If mu i is strictly between 0 and C, right? My complementary slackness, remember my complementary slackness implies if mu i is is greater than 0, then this term is 0, this constraint becomes equal to 0. Similarly, if lambda is greater than 0 then psi i becomes equal to 0, right?

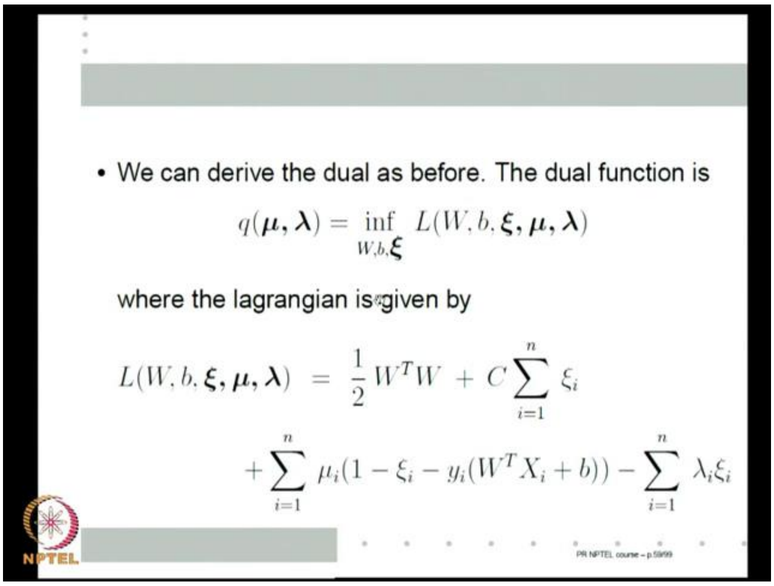
Now, because I have this, if I consider all mu i which are not only strictly greater than 0, but are strictly less than C, then mu i greater than 0, the first complementary slackness tells me that the constraint is 0, because lambda is greater than 0. Why is lambda a greater than 0? Because mu i is strictly less than C and I need mu i plus lambda is equal to C, lambda has to be strictly greater than 0, which implies psi i is equal to 0, right? So, if if I consider any mu i such that mu i is strictly between 0 and C, then lambda is greater than 0, so psi i is equal to 0 and mu i greater than 0, so this term is equal to 0.

In this term I know psi i is equal to 0, I get 1 minus phi i into W transpose a plus b is equal to 0. Now, the complementary slackness conditions give you 1 minus y i into W transpose X i plus b is equal to 0. So, earlier for any mu i greater than 0 this is true, now for all mu i which are strictly between 0 and C this is true. Now, this obviously immediately gives me like earlier my b star, right? Only thing is instead of this formula being true for any mu that is strictly greater than 0, it is only for those mu which are strictly between 0 and C.

Now, I can think of my vector that determine b to be one that are not stuck at 0 or C. See now that, see earlier I had only mu i greater than 0, now I have mu i plus lambda is equal to C and lambda is also greater than or equal to 0, which means mu i is now strictly between 0 and C anyway i need to have 0 less than or equal to mu i less than or equal to C. Similarly, 0 less than or equal to lambda i less than or equal to C. So, unlike earlier problems mu also has a strict upper bond, if mu i is strictly between their bonds not equal to either of the bonds, then that mu i, that determines these X i which are closest to the separating boundary and all.

Earlier, the X i that are closest to the separating hyperplane are those whose corresponding Lagrange multipliers are strictly positive. Now, with this slack variable formulation, I have to look at those X j for the for whom the corresponding Lagrange multipliers are not stuck at either end, they are neither stuck at 0, nor stuck at C. Those mu correspond to X j which satisfies this which means those are the X j that are closest to the separating hyperplane now. Apart from that once again if you I can (( )) a star and b star so I am done. So, once again I can use the dual to find the mu.

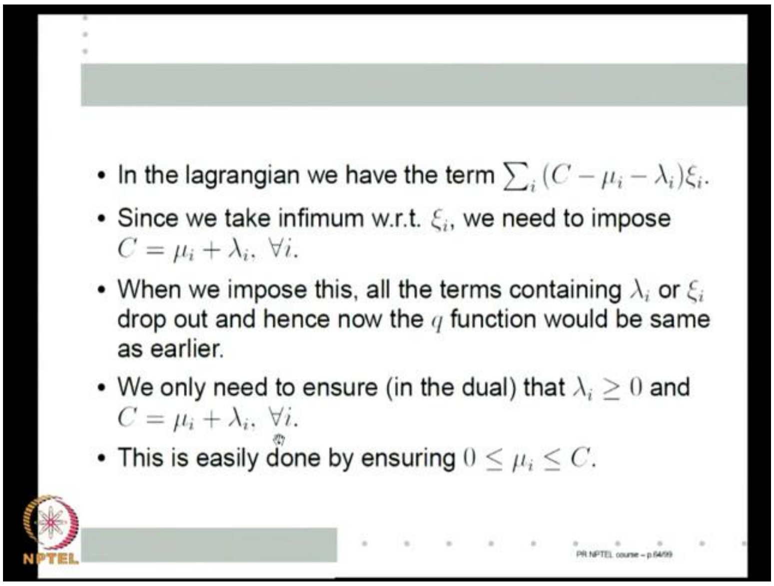
(Refer Slide Time: 34:18)



So, what will be the multipliers, now there are the mu and the lambdas, so q is a function of mu 1 lambda, it is obtained as infimum over the all the optimization variables of the primal problem namely W b psi of the Lagrangian, Lagrangian function W b psi and the two sets of Lagrange multipliers mu 1 lambda, so where the Lagrangian as we have already seen is given by this. So, basically for a given a given a particular mu 1 to mu 1 and lambda 1 to lambda n, if I want the value of q at that mu 1 lambda, I need to find infimum of this expression at that mu 1 lambda with respect to W b and all the psi i, right?

Let us first consider finding infimum with respect to psi i, what is, what is the psi i dependence of this term, for each i I have a psi i into C minus mu i minus lambda i term. So, for a given lambda and mu if C minus lambda i minus mu i is not 0, then I can take that psi i to be as negative as I want, when I am taking the infimum or as positive as I want and hence infimum will be minus infinity.

(Refer Slide Time: 35:41)

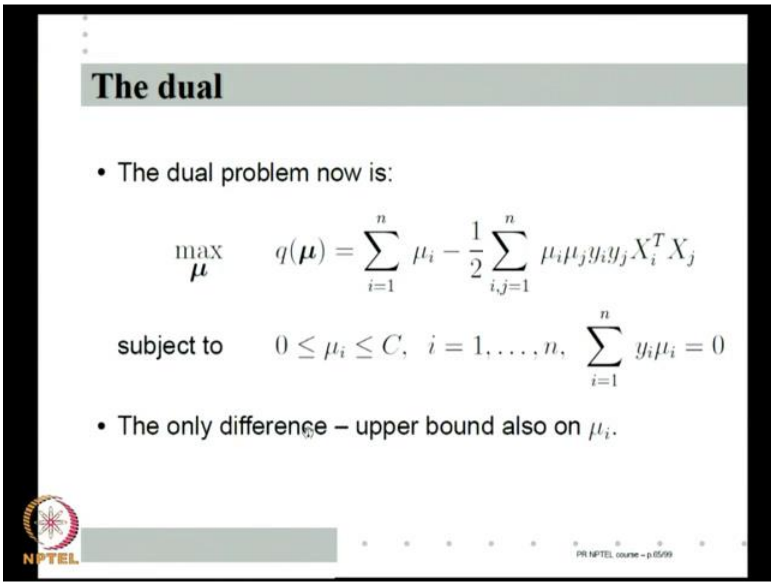


Because we have this term, because you are taking infimum with respect to psi i, unless we impose C is equal to mu i plus lambda i infimum will be minus infinity and that we are not interested. So, once again we only work, we only need to look at mu 1 lambda, so that C is equal to mu plus lambda. When we, when we consider mu and lambda so that for each i mu i plus lambda is equal to C, then all the psi i terms will drop out, for each psi i i have got a c minus mu i minus lambda i and similarly, all the lambda i terms drop out, right?

The moment I impose the c is equal to mu i plus lambda i, then all terms containing is a lambda i or psi i or dropout of the expression. Once the drop out of the expression what is left is the old expression and I need to find infimum with respect toW1 b and we already that is that old q function we know, right? So, if you want to find the new q function, all you have to do is that we get the old q function only anyway, because we have to anyway impose this. But in addition to the old constraints I have to also impose the constraints that lambda is greater than equal to 0 and C is equal to mu i plus lambda i. Because I have to impose C is equal to mu i plus lambda i, my q function will not contain any dai terms.

So, q is still only function of mu, so I do not have to actually maximize to our lambda i, as long as I can somehow ensure that these two conditions are satisfied. These two are easily satisfied by imposing this condition, and once I impose this condition on mu, I can take c minus mu i to be lambda i, because lambda i need not have to does not acquire the cost function, lambda i does not affect the cost function as long as I satisfy the constraints on lambda I am done. So, let us say I solve it only for mu with mu as this constraint, then C minus mu i I take to be lambda i, then I have to satisfy lambda i greater than or equal to 0, as well as mu i plus lambda i is equal to C.

(Refer Slide Time: 37:42)

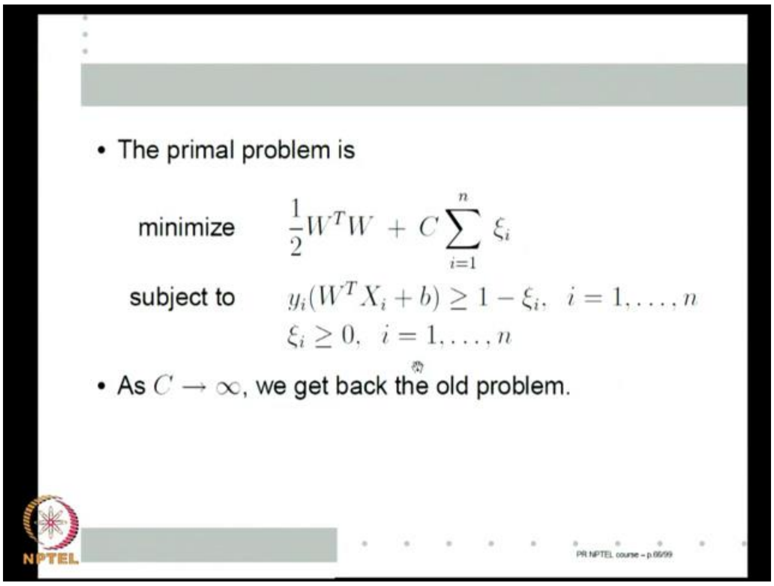


So, which means my dual now will be exactly be same as the old dual, the cost function. All the old constraint will be there, earlier I had only mu i greater than or equal to 0, now

I have mu i less than equal to C also. So, the only difference is in upper bound, right? This very nice see in the primary problem, the problem is totally different I get a plus C into summation psi i in the cost function, each of my constraints is changed I have got n extra constraints.

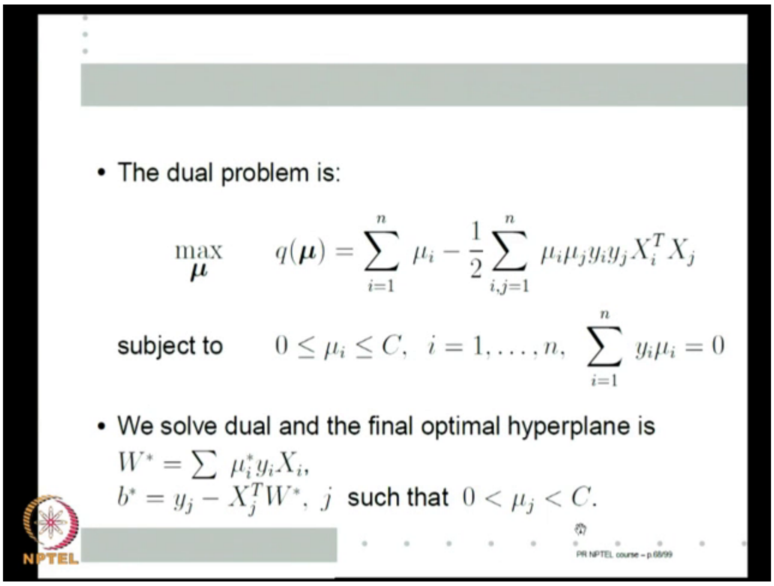
So, if you wanted to solve the primal lots of extra work needs to be done, but if you want to solve the dual whether I am thinking that the patterns or linear separable all I am willing to use slack variables, so that I have always have a feasible solution and always get a solution. The only difference is whether or not I put an extra upper bond which is anyway used as different constant. So, essentially solved this optimisation problem the dual by by just choosing some convenient upper bound and you know that is all right. The only difference is in upper bound I am doing, so that is for another reason for preferring to solve the dual, right?

(Refer Slide Time: 38:45)



Of course it is very easy to see, this is my primal problem. So, if C tends to infinity, if C very a large that even a little bit of positive psi, this term becomes so large that that will not be the where the minimum is attained. So, ultimately my minimum will be attained only for those psi for that place where psi i are all 0, if psi i are all 0 it is like the old problem. So, if C tends to infinity we get the old problem, if C tends to infinity I get the old dual obviously, right?

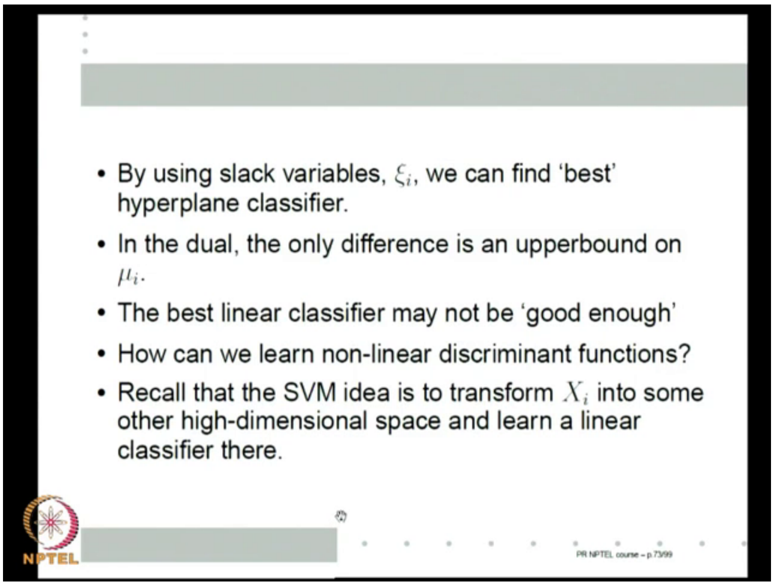
(Refer Slide Time: 39:19)



So, this is my dual, so we solve the dual and once again this is given by my, this is my optimal hyperplane. The only difference is earlier the b star is for any j of that mu j greater than 0 and now it is mu j strictly between 0 and C. So, this extra one upper bound on C is all the difference in the optimization problem between solving the problem assuming training data is linearly separable or you know taking general data and finding the best possible linear separation. Is best possible in the sense I have just decided to make a few errors on some patterns and add the others in one specific way, by adding these slack variables into the optimization problem.

So, if this is a formulation with somehow I am minimising some measure of the error in classification while maintaining a maximum margin, right? When in the linearly separable case the optimization problem is very nice to see, because it maximizes margin and the separability constraint. In this case because of this C, I am essentially adding apples and oranges I am taking some part of the margin, I am some margin plus some constant into the others are making I am minimising that. But anyway this is what one does has to do in general, because in general we have no way of knowing whether the training data is linearly separable or not.

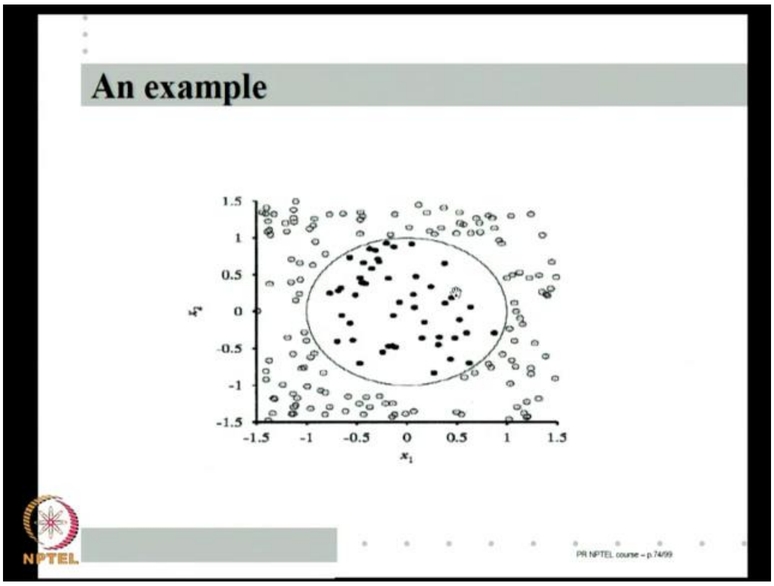
(Refer Slide Time: 40:42)



So, by using the slack variables psi i, we can find the best hyperplane classifier. So, this is the optimal separating hyperplane very good, so we can find, instead of finding any linear classifier now, we have a concept of some optimal linear classifier and we can find it by solving a nice quadratic optimization problem as a quadratic cost function and linear constraints. The only difference is when we are solving the dual as we have seen the only difference is an upper bound and mu, so I just throw in a constant c as an upper bound on mu and I always solve only the dual, so it is very nice.

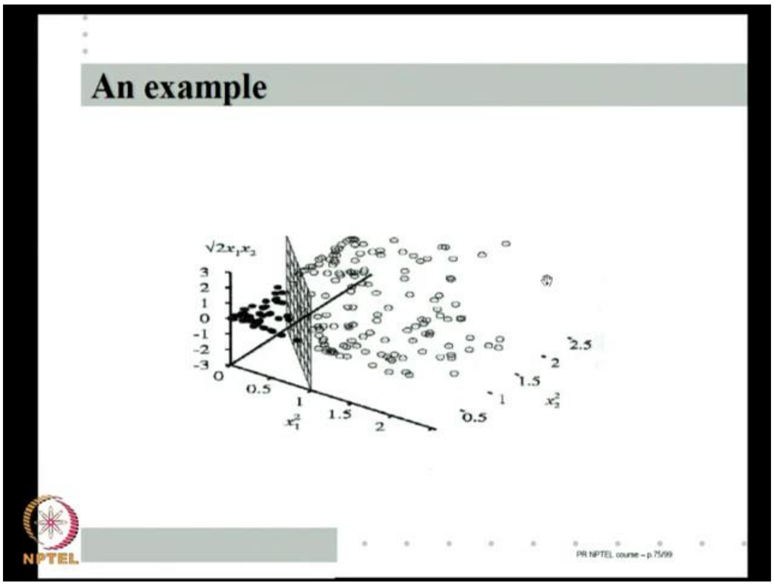
Of course, as we have already seen the best linear classifier may not be good enough, the idea is we want to learn nonlinear discriminant functions and recall that the whole idea of SVM is to transform psi into some other high dimensional space and then learn a linear classifier. The idea is psi will be transformed to z and then in that z space you learn a linear classifier, right?

(Refer Slide Time: 41:38)



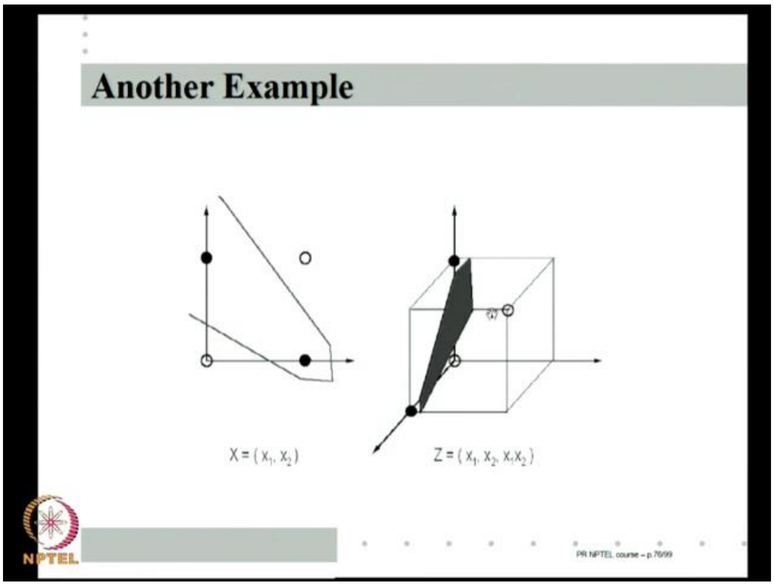
So, as you just to recall this is the idea, so if this is the original feature space and i needed a quadratic discriminant function to separate the two classes.

(Refer Slide Time: 41:48)



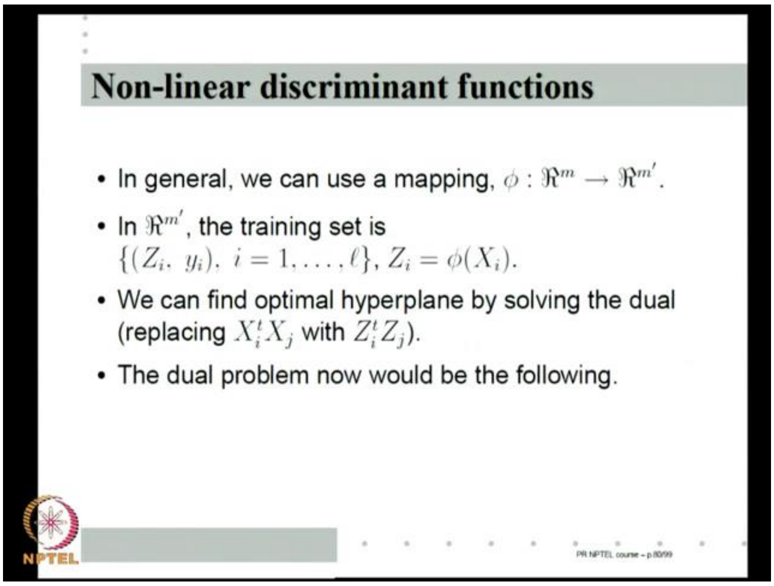
If I map the patterns into high dimensional space, I would be able to separate with the hyperplane. Essentially this is in theX1X2 space, whereas this is in theX1 squareX2 squareX1X2 space. So, a hyperplane can separate the classes now, whereas I needed a second ordered discriminant function, quadratic discriminant function in the original space.

(Refer Slide Time: 42:14)



It is another example, if I have an X r problem, right? To separate the two classes I need a nonlinear classifier a single layer line cannot separate them. But I can always map them to a higher dimensional space, which is X 1 X 2 and X 1 X 2 and in that space a hyperplane can separate them, so that is the basic idea.

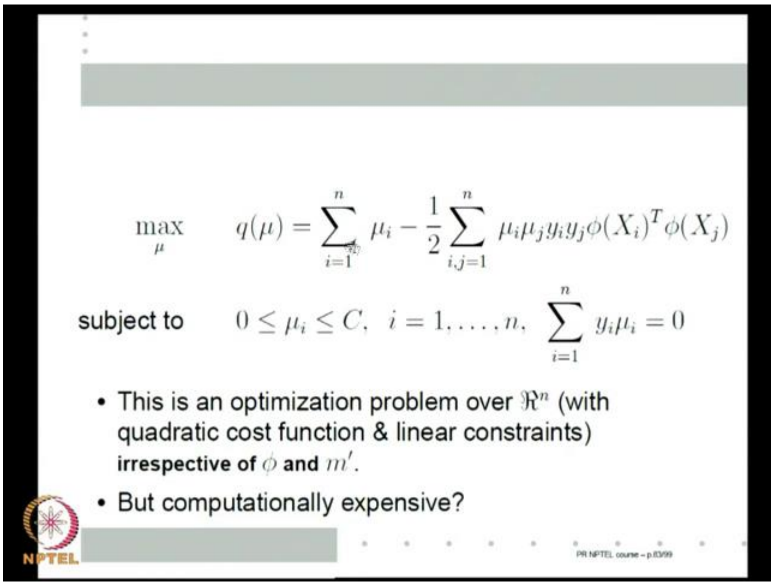
(Refer Slide Time: 42:33)



So, in general we used some mapping phi from m dimensional space to m prime dimensional space. In the new space my training set now will be z i y i where z i is phi X i. So, we will be in the next, in the rest of this class and also most of our discussions of same we will use z and phi X interchangeably, phi X i will be z i that is what the new trans, this is the, this is the vector corresponding to the training vector X i in the transform space. So, once we use this phi to transform all the training patterns from m dimensional space to m prime dimensional space, my new training set has become z i y i, so z i is equal to phi X i.

So, we can find the optimal hyperplane by solving the dual, what was our dual? If I wanted to find the best linear classifier, this is what I have to solve where X i X j is the training patterns, so nowhere else the training patterns have appeared. So, if my new training my (( )) transformed them to z i I just replace X i transpose X j by z i transpose z j that is all or phi X i transpose phi X j. So, we can find the optimal hyperplane by solving the dual just simply replacing X i transpose X j with z i transpose z j.

(Refer Slide Time: 43:51)



The dual problem will be what now? I just replace X i transpose X j with z i transpose z j z i transpose z j is same as phi X i transpose phi X j this. Am I done? The good thing about it is there is an optimization problem over r n the number of examples, it has nothing to do with the function phi r m prime. Nothing to do meaning the the dimensionality of this optimization problem does not depend on the function phi or the mu space m prime. So, originally I may have hundred dimensional space, now phi X may be ten thousand dimensional space, but in both spaces I will be solving may be a thousand dimensional optimization problem because I have thousand examples.

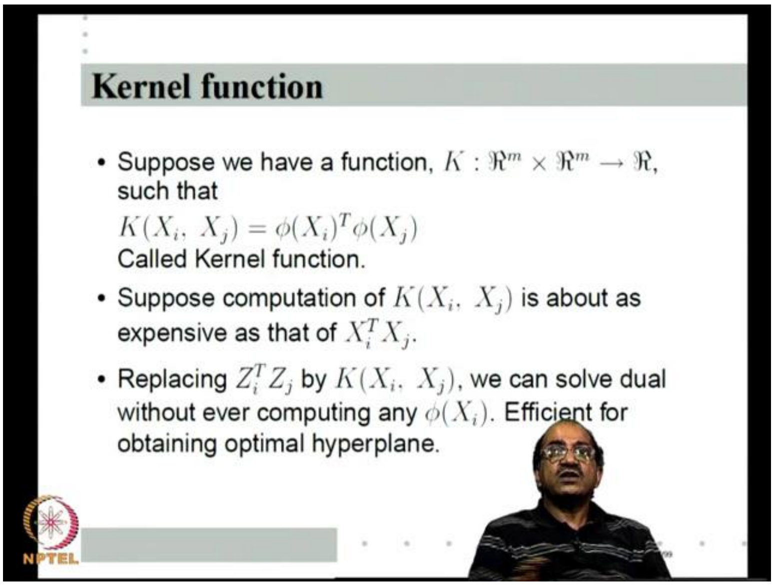
So, the optimization problem remains only of r (( )) at least earlier we said namely solved the problem in the z space, I have to solve a problem of the dimension of the z space that is m prime, but now I am not solving that, because I am solving the dual. The dimensionality of my optimizational problem remains n the number of examples, it does not become m prime. So, even if m prime is ten thousand or one million my dimensional optimization problem does not, does not increase it will stay at n. This is the good thing, but does it mean that I have really got rid of computational cost; of course, during learning I have to solve this problem to find mu i given all my training examples, so I have to calculate phi X i transpose phi X j.

Of course, I can pre compute it because these do not depend on the optimization variables here, so I can find all the n squares or the n square number phi X i transpose phi X j one time and then solve use those constants here and solve this optimization problem. Of course, still it can be expensive, because I have to first transform each of the X through phi X i, phi might be a difficult transformation and in that high dimensional space may be ten thousand dimensional space or one million dimensional space I have to calculate the inner products.

But I did it do it only once and then solve this mu obtained, is that all? No, what do I do with the mu I have to use my W stars, so the W star will will live in the phi X space so W star is the same dimension as phi x. So, how to store W star? The one million dimensional W star and every time now you give me a new X I still have to do W star transpose x, right? I have not, so every time I use the classifier I suffered the computational cost of going into high dimensional space and finding inner products there.

Of course, in addition even though the optimization problem remains after dimension n, I do not know because effectively I am learning a a hyperplane a very high dimensional space, are thousand examples enough. The second problem we will solve much later, but this SVM method as efficiently tackled that particular let us see the reasons for it later. But right now let us concentrate on the computational problem, how do I get rid of having to do this phi X i transpose phi X j?

(Refer Slide Time: 46:52)



The basic idea is the following, suppose we have a function which takes (( )) of m dimensional vectors and gives me a real number, such that k of X i X j is phi X i transpose phi X j, we call such a k as Kernel function. What do you mean by I have such a function? Of course, I have such a function once I know phi I can define k X i X j to be a phi X i transpose phi X j or basic idea is that such a function is less come to expensive than calculating phi X i transpose, right?

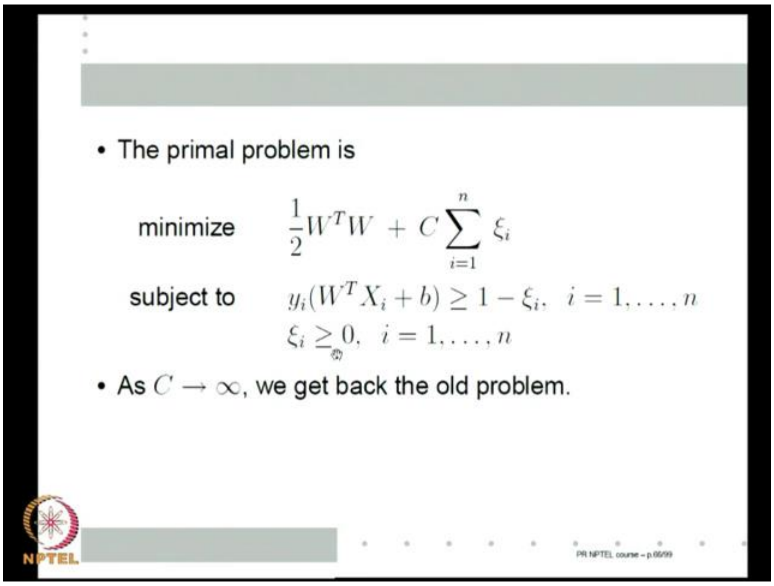
We may not have time to look at Kernel function in this class, but not to make a big secrecy of its. For example, if I take k X i X j to be 1 plus X i transpose X j whole square 1 one plus X i transpose X j whole square needs one extra multiplication compared to X i transpose X j. Suppose, X i transpose X j is 100, X i is 100 dimensional, X i transpose X j needs hundred multiplications, one addition we do not care 1 plus X i transpose X j whole square we will need one more multiplication, right?.

So, we are looking at Kernel function like that we will (( )) square as a Kernel function would be phi X i transpose phi X j for a phi which effectively learns the quadratic discriminant function in the original space, right? So what we considered right in the beginning two class ago about mapping X 1 X 2, 2 X 1 square X 2 square, X 1 X 2 X 1 X 2 and then finding a linear classifier there, where we said we need order n square dimensional z. Finding an inner product in that space would be like 1 plus X i transpose X j whole square, we will see that later.

So, in that sense instead of doing ten thousand multiplications to find phi X i transpose phi X j and also many multiplication to actually find phi X i and phi X j, we will just do one extra multiplication and X i transpose X j and all. So, we are looking for functions which are not particularly more expensive than doing X i transpose X j. Let us say such a function exists, that let us just assume for now such a function exists, then what can that function give us? Then replacing z i transpose z j by k X i X j we can solve the dual, this is my dual phi X i transpose phi X j is k X i X j, so I will just list a phi X i terms for phi X j I have put k X i X j here.

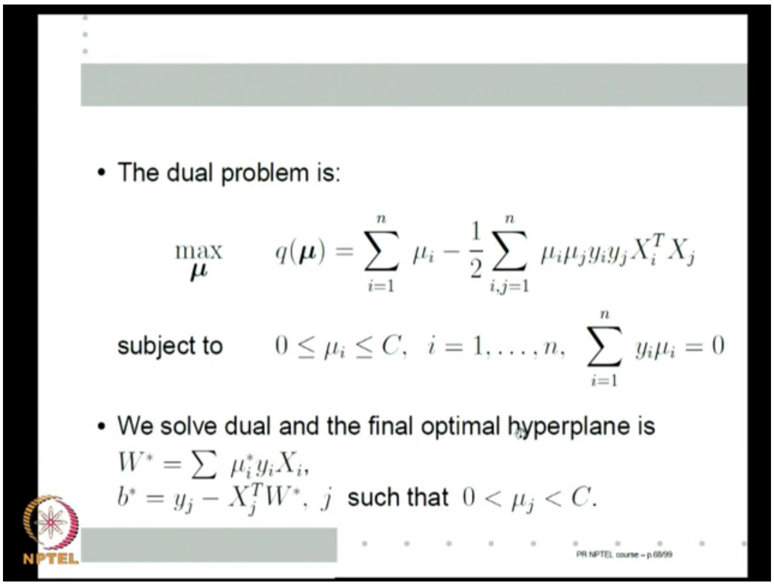
So, k psi X j is much less computationally expensive than phi X i transpose phi X j which means I have solved at least the computational problem of calculating phi X i and phi X j while learning the mu. So, replacing z i transpose z j or phi X i transpose phi X j by k X I X j we can solve the dual without ever computing any phi X i, so for obtaining the optimal hyperplane, right? So, one problem is solved now, so I can efficiently solve the dual see that is the whole idea why we went to the dual.

(Refer Slide Time: 50:33)



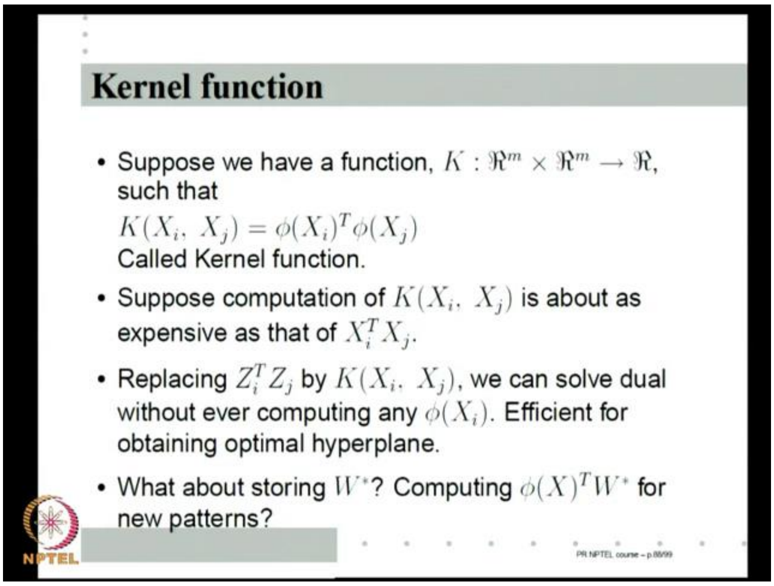
If I want to do the same thing in the primal, this X i will become phi X i, right? So, I cannot use the Kernel function here if and this is not even linear constraints anymore because I have a phi X i there, right? I mean occurs will be linear constraint in the sense X phi X i are constants linear in W and b even though it is a linear constraint, this phi X i because it comes into the constraints, right? Every time I do any, I will use any numerical algorithm in every iteration I have to calculate the corresponding phi X i in the primal.

(Refer Slide Time: 51:07)



On the other hand in the dual, because it comes only with X i transpose X j and nowhere else I am, I am able to substitute this by the Kernel right.

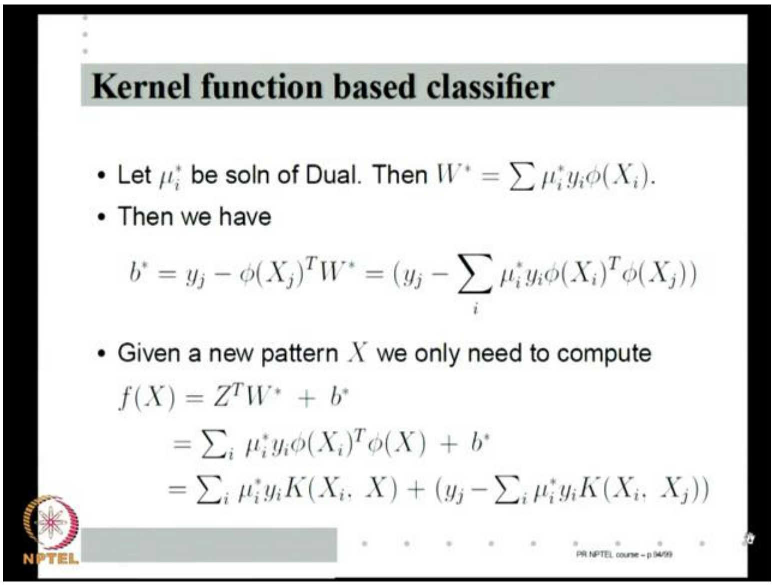
(Refer Slide Time: 51:30)



So, what the kernel has given me is that by replacing this phi X i transpose phi X j by k X i X j I have been able to efficiently solve the dual problem without actually calculating phi X i and obtain all my hyperplanes, all my Lagrange multipliers. So, far so good, but does it mean that I have solved the problem of inefficiency in inherent with this idea, one of the inefficiencies inherent with the idea is every time we may use new pattern to classify. I have to transform it in this high dimensional space and then classify it with the corresponding name.

So, even though for obtaining my mu I do not have to calculate phi X i what do I do with my mu? Ultimately, obtain my mu so that I can calculate W star and I calculate W star, because next time you give me an X I have to calculate phi X transpose W star to classify it. So, W star stays in my million dimensional space so I have to store W star and I have to calculate phi X every time you give me a mu X and do this, what do I do about this?

(Refer Slide Time: 52:34)



Well as it turns out I do not have to do that also, it is very very interesting thing. Let us say mu i star are the solutions of the dual, then we know that W star is summation mu i star y i phi i X i. So, for example what will be b star we seen b star, will be y j minus phi X i transpose W star, now W star is given by this, so it will be y j minus summation over i mu i star y i phi X i transpose phi X j. Wherever phi X i transpose phi X j comes I know I do not have to do any phi X phi j computation, I can convert it into Kernel function and k psi X j is much less expensive than phi X i transpose phi X j.

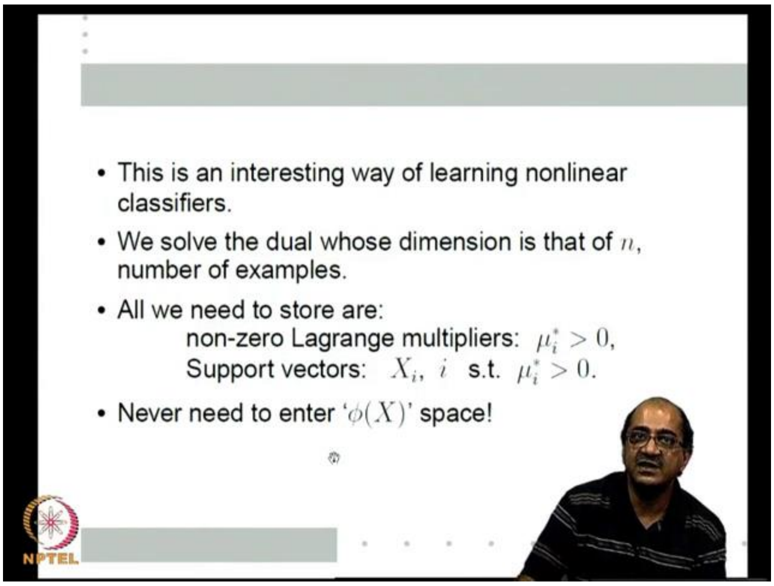
So, for example to compute b star I do not need to calculate phi X at all, if I know all the mu stars, then I can just put k X i X j here and calculate b star of course b star is not the only thing I need, I need much more than b star. What you have to do give a new pattern, what you have to compute? I have to compute f X right ultimately my classification is depend on the sign of this f x, what is this f x? z transpose W star plus b star, what is z phi x? So, phi X transpose W star plus b star, that is what I have to calculate and find its sign.

What will be that be, so W star I write it as W star transpose phi x, W star is mu i y i phi X i , so this is summation over i mu i star y i phi X i transpose z is phi X plus b star, now I know how to write b star. So, this is summation over i mu i star y i phi X i transpose phi X can be written as k X i comma X plus y j minus summation y mu i star y i k X i comma X j. This is for a particular j which have for which the corresponding mu j star is between 0 and c. As you can see while finding the mu I do not have to calculate any phi x, after obtaining the mu you give me any mu X and I want to calculate its actual phi X transpose W star plus b star.

Once again I do not have to calculate phi X at all I can do with the Kernel functions I just have to calculate this, right? This is a really interesting way of learning non linear classifiers. Now, no matter what phi X is, no matter what how high dimensional m prime is I am solving n, a constraint optimization problem and that to a quadratic cost function linear constraints constrained optimization problem. Whose dimension is simply a number of examples and to solve this I do not have to actually calculate my phi x, phi is only in the background is only effectively, because I Kernelise my inner product.

So, in by while solving the dual, where phi X i transpose phi X j would have come I would have put k X i X j, I solve the dual get all my mu, once I get my mu I have to effectively calculate this, that I can calculate like this using my mu and the Kernel function. So, I never actually calculate W star, never actually calculate phi X for any x, right?

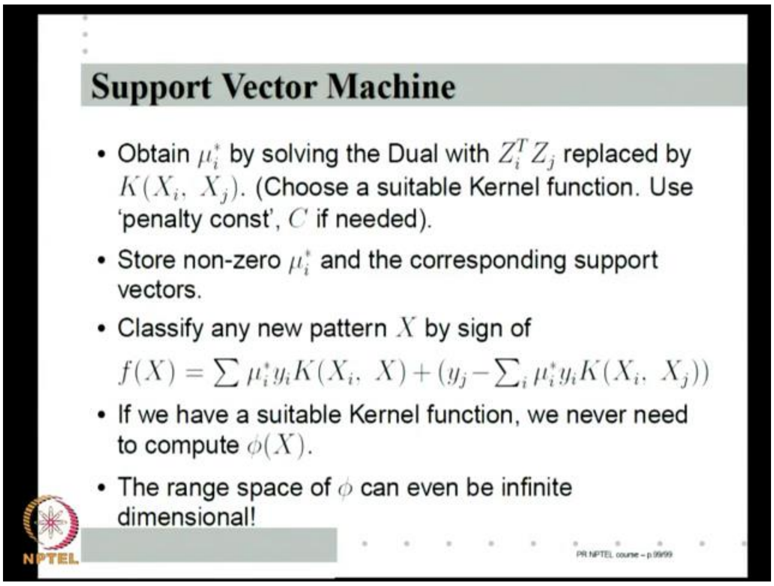
(Refer Slide Time: 55:54)



It is a interesting way of learning non linear classifiers, we solved the dual whose dimension is that of n, the number of examples and all we need to store or all the non zero Lagrange multiplier is mu i star and the corresponding support vectors as X i, right? All X i corresponding to mu i star greater than 0 I have to solve. If I solve them, if I store them you give me any mu X I calculate this and then I am done. I can classify here X by doing this, I never need to enter the phi X space, even though I am actually using some transformation inside phi, which maps my m dimensional space to some other high dimensional m prime dimensional space, I am finding a linear classifier there, I never calculate phi X of for any x, I never ever enter that m prime dimensional space, right?

The only connection between that space and me is the Kernel function, somehow to find the right Kernel function. We will see next class how we can find Kernel functions, but provided that I can find the kernel function, k X i X j is equal to phi X i transpose phi X j such a way that k X i is a much simpler to calculate the phi X i transpose phi X j then I am done. So, no matter what are the transformation phi I am interested in, if I can properly find a Kernel function to represent it then I am done, right?

(Refer Slide Time: 57:18)



What is the final support vector machine idea? Is that I obtain mu i stars by solving duals where that X i transpose X j is replaced by z i transpose z j, which is replaced by k X i X j that is essentially it chooses a suitable Kernel function and use a penalty constant C if needed. Using a penalty constant is simply putting an upper bound in my in solving my dual optimization problem, so I will just throw in a C and choose some suitable Kernel we will see what are suitable next class and I solve the dual.

Once I solve the dual, I store all the non zero mu i stars and their corresponding support vectors. Now, that is my support vector machine this represents my classifier, how do I represent my classifier? Once I solve them, I am now ready to work, you give me any new pattern x, this is what I calculate and I classify it by the sign of this. So, if you have a suitable Kernel function we never need to calculate phi X for any x, what does this mean?

For all I care for the range space of phi can be infinite dimension, I never go there. So, this is a very, very interesting idea of effectively finding a linear classifier in a high dimensional space. So, we next class we will look at the Kernel functions, what kernel functions are suitable and how different kernel functions will allow you to find different non linear classifiers.

Thank you.