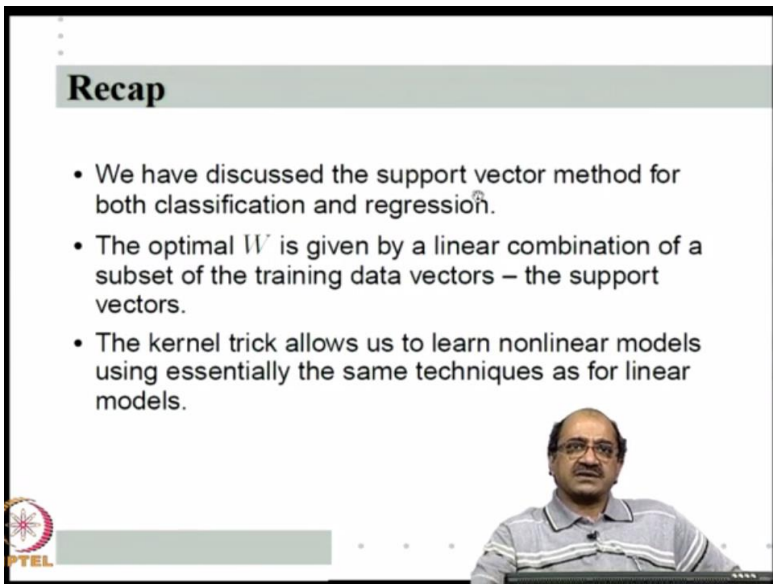
Pattern Recognition
Prof. P. S. Sastry
Department of Electronics and Communication Engineering
Indian Institute of Science, Bangalore

Lecture - 36
Overview of SMO and other algorithms for SVM; v-SVM and v-SVR; SVM as a risk minimizer

(Refer Slide Time: 00:33)

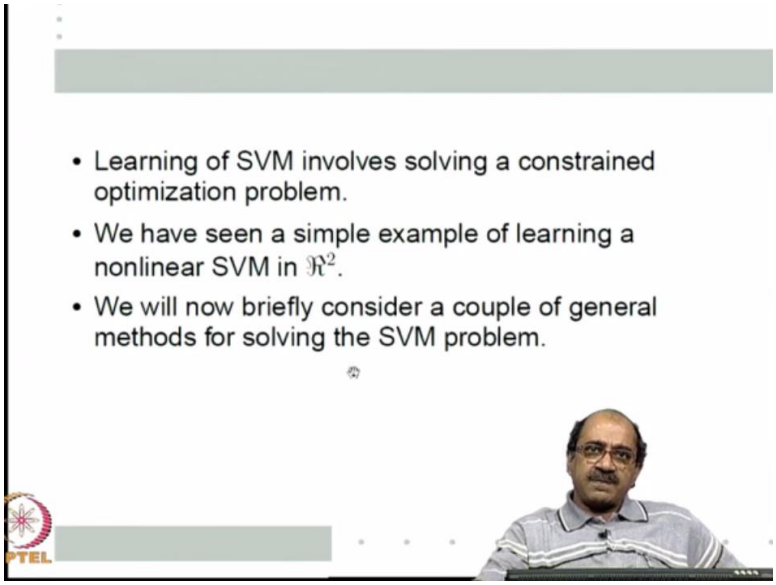


Hello and welcome to this next lecture in this course of pattern recognition, we have been essentially discussing the support vector method and the kernel function based classifiers. So, we will continue with the discussion this class also, we have looked at the support vector method both for classification and regression. We have seen the support vector machines for classification and also looked at this support vector regression in both cases. There are certain things that are interesting one is that the optimal parameter vector is given by the linear combination of a subset of the training data. The linear combinations coefficients happen to be the Lagrange multipliers or some functions of the Lagrange multipliers in the of the optimization problem.

But essentially the W is given as a linear combination of the training, training vectors this, this support vectors. And, we seen that the kernel trick allows us to learn non-linear models using essentially linear techniques this because that kernel we can kernelize the inner product. So, any algorithm that essentially has only inner products can be implicitly

be executed in high dimensional space, using the kernel function. So, these are essentially the two basic characteristics of support vector method that we seen so far.

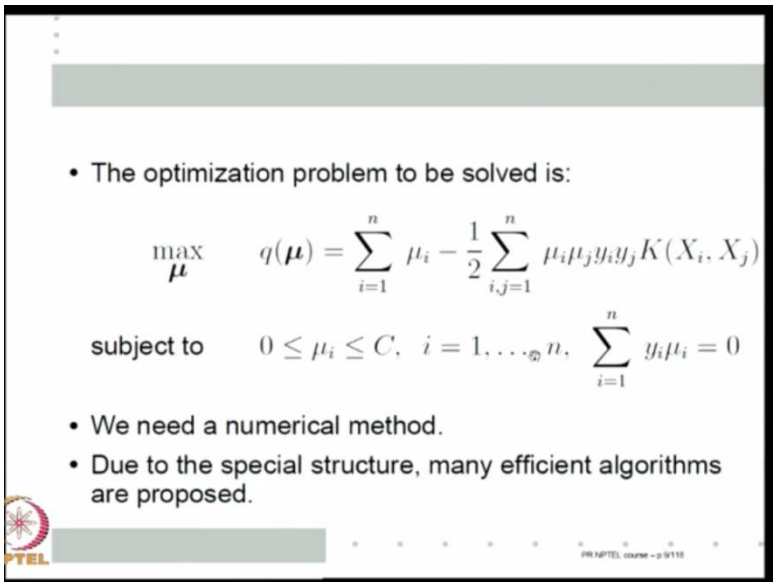
(Refer Slide Time: 01:45)



Now, what we have not fully discussed even though we mentioned it just briefly towards the end of last class is, how does one actually solve the learning problem of support vector machines? Learning of the support vector machine involves a constrained optimization problem. And last class, we seen how to an example of solving this constrained optimization problem for a very, very simple example is, is in R 2 learning a quadratic classifier with only 3 examples. Though it is simple instance, it does tell us the, the more involved the, the issues involved in solving the optimization problem.

That we need more general methods, we briefly mentioned. We just recall them once again. So, in this class, we will briefly look at the general methods that people has proposed for the solution, we are not looking at any method in any great detail, because most of these algorithms simply happen to be specialized numerical optimization techniques. So, algorithmically once you formulate the optimization problem. There is not much more there is; there is nothing much more conceptually to convey except that, these are very interesting specialized numerical technique for solving the class of optimization problems that are of interest in SVM.

(Refer Slide Time: 03:10)



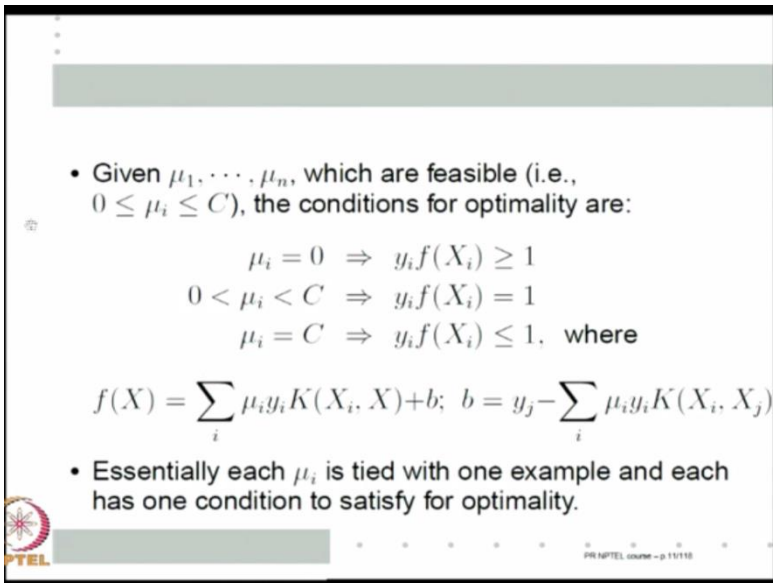
So, this is the optimization problem as you already seen this is the optimization problem that we need to solve is a quadratic optimization, quadratic programming problem. The we have a quadratic cost function, it is a linear term summation, mu i and a quadratic term mu I, mu j, y i y j K X i X j. Essentially, this is the quadratic form with respect to the mu vector of the matrix, whose I j'th element is y i y j K X i X j and R 2 constraints one is S linear constrain linear equality constrain mu i y i is equal to 0. Another, the rest of the rest of the constraints are simply V bound constraints of the variables.

So, is a standard quadratic programming problem with a quadratic cost function and linear constraints. So, what do we need is a nice numerical method of course, there are many numerical methods for general non-linear programming problems. For example, I can use constraint gradient, descent gradient projection one can use penalty method, barrier method. There are many such standard numerical algorithms for solving constraint optimization problems, but you know we can do; we can; we can certainly do such gradient descent base methods.

But this problem has lot of extra structure, and because of that many more efficient algorithms are proposed. This essentially, there are only bound constraints on the variable which are very simple to implement. There is only 1 equality constraint, this is a very nice quadratic form, so there are interesting things you can do with it to actually, I look at this structure, let us look at the conditions for optimality of this problem. We are

asking given some mu 1 mu, 2 mu, n when how can I say (( )) not there optimal. So, I it simply means whether, or not they satisfy the Kuhn tucker conditions, so by writing Kuhn tucker conditions and doing some algebra.

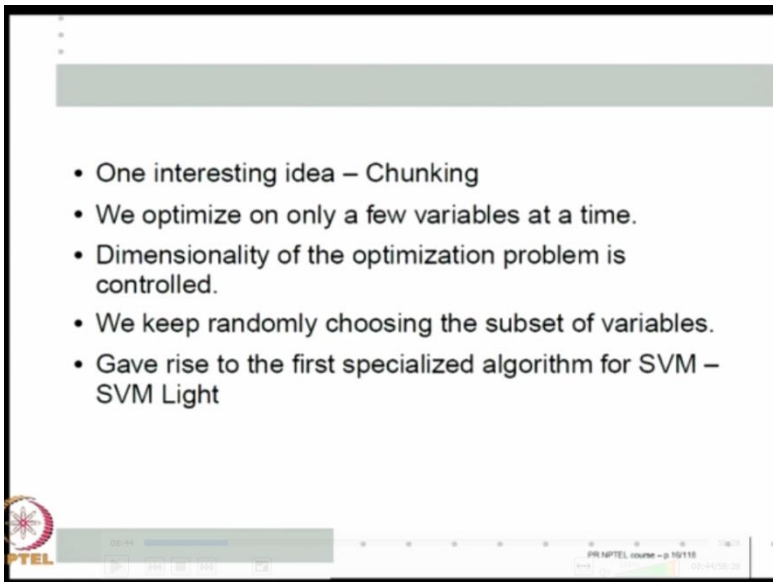
(Refer Slide Time: 05:11)



We can show that these are the optimality constraints, if you give me mu 1, mu n then I will define f X as mu i y i K X i X plus B. Where b is given by this for some mu j greater than 0, as we can see these are the standard solutions. We could have got for W on B, if mu what the optimal thing. Then with respect to such an F, the, this is what all I have to satisfy for every. I if mu i is equal to 0 then y i of X i should be greater than equal to 1, if mu i is strictly between 0 and C y i of X i is equal to 1 and mu i equal to C. Then y i of X i has to be less than equal to 1. So, these are this is the optimality constraint.

So, essentially it is as if every mu i is tied with one example. Even in the our constraints with the primal every mu i tied with because of every constraint in the primal is tied with one example. Every mu i is tied with one example. So, I can say which examples mu this is and each mu i is essentially has it is own condition because, it is not complete because, of f X depends on all the other mu is, but really I can ask which mu satisfy the constraint, which mu's do not given the current f X if I think of f X as given. And do not worry about it being a function of mu, I can individually ask which mu satisfies and which mu's do not satisfy the optimality constraints. So, given this kind of a structure, there are many interesting tricks you can do in optimization

(Refer Slide Time: 06:48)

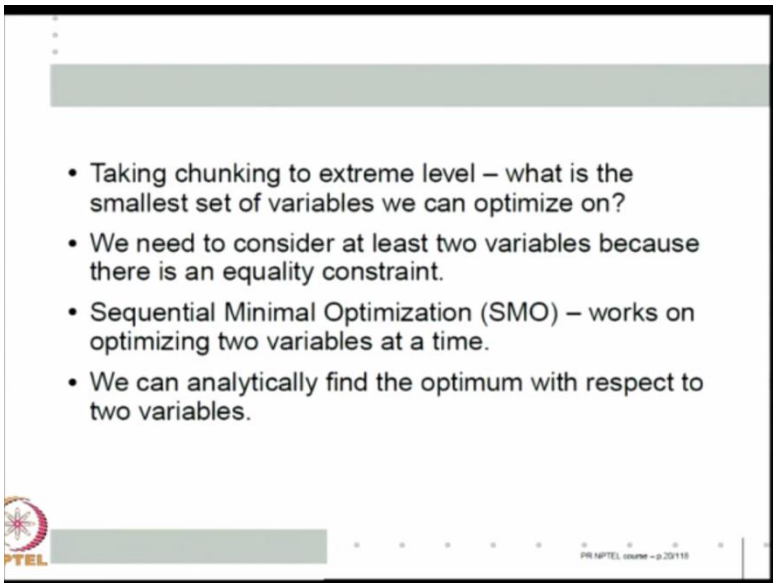


And one standard such trick is what is called a chunking. Chunking is a very generic idea in optimization, where even though I have a optimization problem with large variables. We would optimize only on a few variables at a time, this standard chunking that most of us know about is the coordinate wise optimization. If I have an unconstraint optimization RNA simple thing to do is to first optimize with respect to first variable X 1 then with respect to X 2. Then with respect to X 3 and so on the idea is that all one dimensional optimizations are simple. So, in general chunking the idea of chunking consists of optimizing on only few variables at a time. It is utility is that the dimensionality of the optimization problem is controlled.

Because we may not want to do one at a time as in the coordinate wise optimization so may be, I have optimization problem over thousand variables. I may be between 25 at a time, or 50 at a time, or 20 at a time. the idea is because the quadratic programming problem if I take only 20 variables. I have a quadratic form with 20 by 20 matrix which is much easier to solve than a cost function which involves quadratic form of 1000 by 1000 matrix. Basically the idea of chunking is that at our iteration, I choose let us say a 20 or 25 variables to optimize on and only for that, that many variables are in the optimization algorithm and I keep randomly or chosen, keep choosing subsets of variable to optimize and normally this speeds up the entire optimization, and this idea gave rise to the first specialized algorithm for SVM which is called SVM light. Even today possibly one of the most popular SVM algorithms easily downloadable, if just from any web

browser SVM light will tell you how to download SVM light. Freely available software and very good for experimenting with so, SVM light essentially uses a chunking idea to solve the SVM optimization problem.

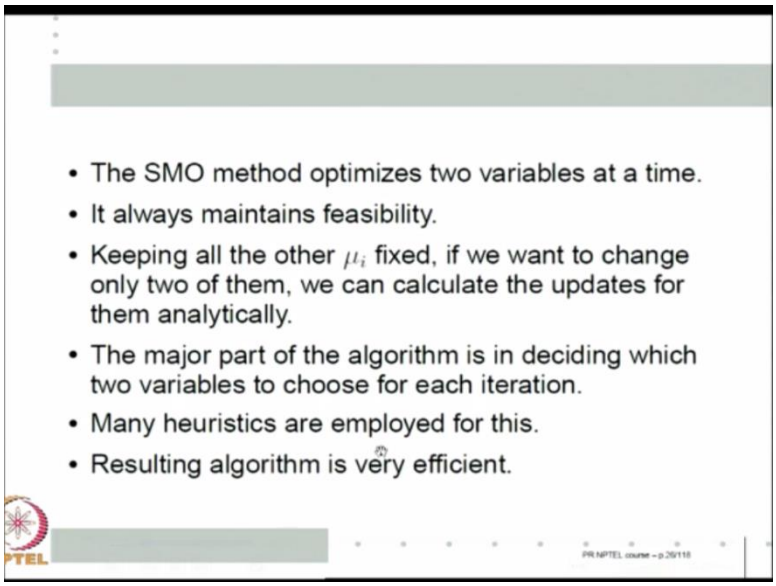
(Refer Slide Time: 09:00)



Now, as we seen chunking is basically do a few variables at a time. So, you can what is the extreme chunking I can do what is smallest set of variables, I can optimize on can I do coordinate wise optimization in this problem, I cannot because the mu i is have to satisfy one equality constraint mu i y i summation. mu i y i is equal to 0, what it means is if I am currently at a feasible point. So, I am at some mu 1, mu 2, mu N, which is feasible it satisfies all the constraints. If I change only 1 mu keeping everybody else same, obviously; I will go out of the feasibility region because, I can no longer satisfy the equality constraint. So, because a higher equality constraint if at every iteration I want to move from a feasibility point to a better feasibility point and so on.

I cannot get rid of feasibility. So, I have to at least consider 2 variables at time, so this smallest chunked that I can optimize on is 2. So, there is an algorithm called sequential minimal optimization. It is called minimal, because it works only on 2 variables at a time, the earlier we said that, by during chunking, by reducing the number of variables to optimize. We are improving the, the I mean improving the numerically over speeding of the algorithm, so using only 2 at a time is very nice because you know, we do not even need a numerical optimization technique to actually with respect to 2 variables.

(Refer Slide Time: 10:26)



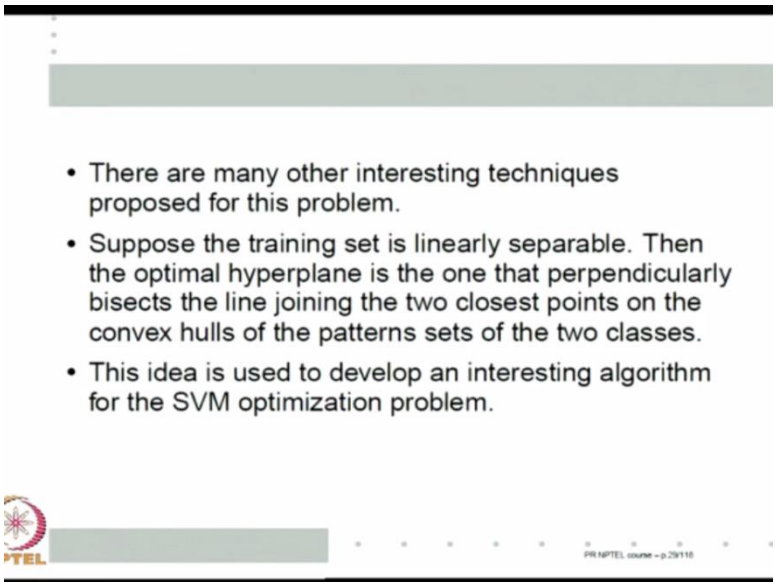
So, this is the algorithm SMO so SMO method optimizes 2 variables at a time it always, maintains feasibility at any given point. It is at a feasible at any given time in the algorithm is at a feasible point and then, it uses some 2 variables to optimize and comes to a new sort of a values for those 2 variables in such a way that terms still in the feasible division. It always maintains feasibility basically, what it does is keeping all other mu i fix, if we it changes only 2 of them let us say mu1, mu 2 have been changed. So, mu 3 to mu n are all fixed. Then my, my quadratic form of the; of the objective function right.

If I can always take, away all the terms involving only mu 3 to mu N, then I just get some quadratic expression in mu 1 and mu 2. Now, that is very easy to optimize, I can analytically find the optimum with respect to those 2. So, keeping all the other mu, I fixed if we want to change only 2 of them, we can calculate the updates for them analytically very easily which means, I do not have to spend my time doing numerical optimization. So, I can spend most of my time in deciding which 2 variables to choose at each iteration.

Now, I have separate optimality conditions to be satisfied by them. So, I know which all variables are not satisfying optimality. Now, out of them I can choose 2 based on some heuristics. So, there are lot set of heuristics in SMO, which are employed to decide which 2 variables to choose and the resulting algorithm is extremely efficient. It can solve tons of 1000 of variable SVM problem very, very fast even on a simple computer.

So, SMO is for a long time or the fastest algorithms for solving SVM, but too there are many, many other recent additions to this, but possibly even today SMO is one of the; one of the simplest in the sense of conceptually simple simple to explain. One of the simplest algorithms that is still very efficient to solve most of the freely available packages used at the SMO or SVM light for solving. The SVM optimization problem cause.

(Refer Slide Time: 12:57)

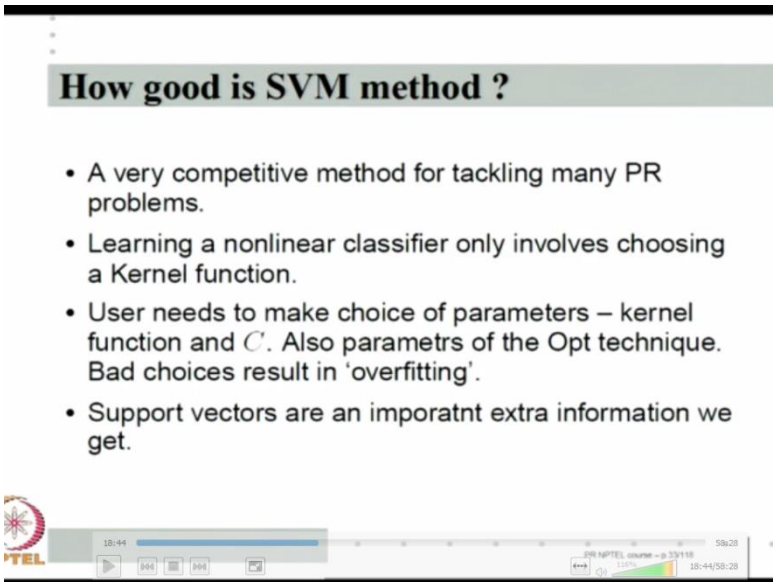


There are many other techniques for example, here is another technique which proposes also quite efficient compared to SMO is a just a little bit involved to actually create the details. The basic idea is the following suppose you have a 2 class problem with linearly separable patterns. When we did the; when we did the V C dimension proof for hyperplanes actually saw that 2 sets of points are linearly separable if and only if their convex hulls are separable. So, basically if I have 2 sets of points which are linearly separable, because 2 class problem where which is linearly separable then if I take convex hull of class 1 and convex hull of class 2 then those will be separable 2.

Then I ask what are the 2 closest points on the 2 convex hulls that is I want to pick a pair of points one on each convex hull, which are closest to each other. Now, they are called nearest point methods and there are, some very efficient algorithm since computational geometry for such things. And one can show that the optimal hyperplane as we have defined earlier will be the line that perpendicularly bisects the line that is joining the 2

closest points. Now, this idea can be used of course, you still need this is only in the linear case I have to make this thing work with kernel function, but this idea has been used to develop a very interesting and highly efficient algorithm for SVM optimization. So, like this there are many other techniques, so we stop there.

(Refer Slide Time: 14:32)



And so we have considered the entire SV support vector method for both classification and regression. We will see some generalizations of the kernel idea, but before that let us ask the question now, that we have considered one more method for learning a non-linear classifiers, how good it is. As it turns out so say very competitive method for tackling most PR problems today, the state of what is such that if your trying out a for the algorithm on your problem of the first, (( )) unless you have very good reason. You have to first try SVM for example, if I am a researcher in your writing, your paper and suggesting a new method the one thing against, which you have to benchmark your method, is a SVM.

So, SVM has become that kind of a standard in pattern recognition today. SVM meaning SVM and all the kernel, kernel based methods, it is possibly more today is much more, much more if a competitive method than neural network methods. We considered earlier, but there is not to say that neural network methods will not work or for every problem. SVM is good obviously; it does not generate magic classifiers but all the same learning a non-linear classifier only, involves choosing a kernel function here.

And choosing a kernel function is much easier because, I just asking the kind of separation. I want as I want to choosing whether I want one hidden layer or 2 hidden layer how many hidden nodes, I want in a in a neural network case and also if your using feed forward sigmidal networks. I have the problem of local minima here the SVM optimization problem does not have the problem of optimal minima. So, learning non-linear classifiers with SVM is certainly very, very attractive because, all it involves is choosing kernel of course, it is not as simple as that uses still needs to make choice of parameters. I have to choose kernel function, but I also have to choose parameters in the kernel function for example, if I am using Gaussian kernel, I have to choose what the sigma in the Gaussian kernel should be.

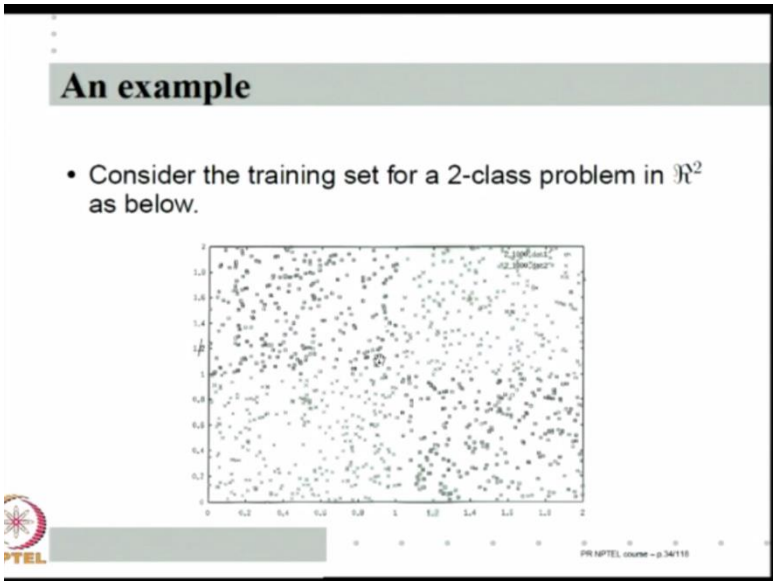
Second thing, I have to choose is the C parameter, the penalty parameter in the optimization problem. I have to also choose parameters in the optimization problem for example, in any numerical optimization. I have to choose an epsilon which is equal to 0 So, what I am trying to say for checking optimality, if I have to say y i f X i is equal to 1 obviously, I have to ask to what precision should be one in numerical technique. So, to also choose an epsilon so all these choices are still there bad choices can still result in overfitting, but given that still SVM is possibly one of the most competitive methods for tackling PR problems today. The second very interesting thing about SVM is are the support vectors, as we have seen the support vectors or he training data which are closest to the class boundary as well as closest to the optimal separating hyperplane.

Of course, this hyperplane is in high dimension space that essentially closest to the class boundary in the original space. So, these themselves have very important extra information that this support vector method gives me because, if I know that these are the patterns that are closest to the boundary in some sense. I know these are the hardest samples to learn to classify as a matter of fact in the SVM method, as we saw once we get our optimal hyperplane. If now I keep only the support vectors, and throw away the rest of the examples. And once again solve the optimization problem, new optimization problem only on this and learn this exactly the same optimal hyperplane. Because these support vectors are the most important the once they classify support vectors. The rest of the things will be automatically classified.

So, in that sense support vectors at the end of thing, I am not only just getting a classifier. I am also getting support vectors, which is a very useful extra information that the

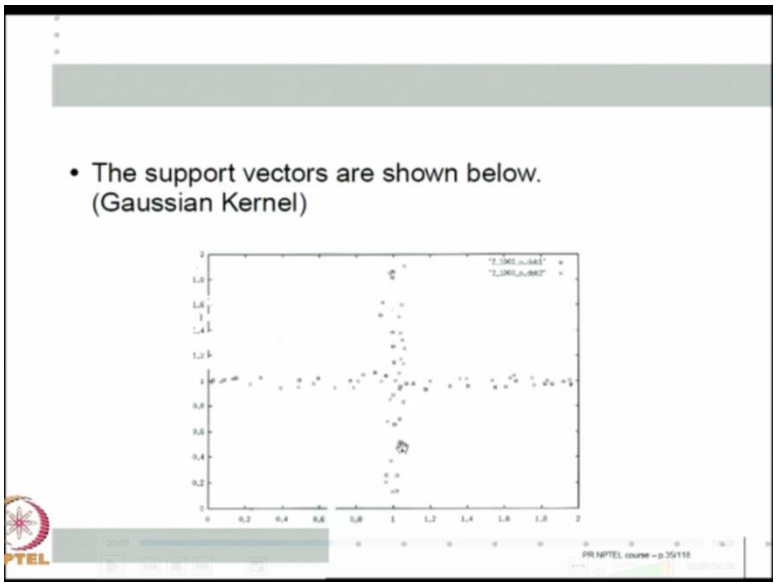
method gives me matter of fact in the sense that they are they, they define the class boundary may be never about classifier. We just use a support vectors as we will do very well as the matter of fact in many applications such things are tried. I will just show you a 2 true interesting cases of which illustrate both how nicely the SV method classifies as well as utility of the support vectors.

(Refer Slide Time: 18:55)



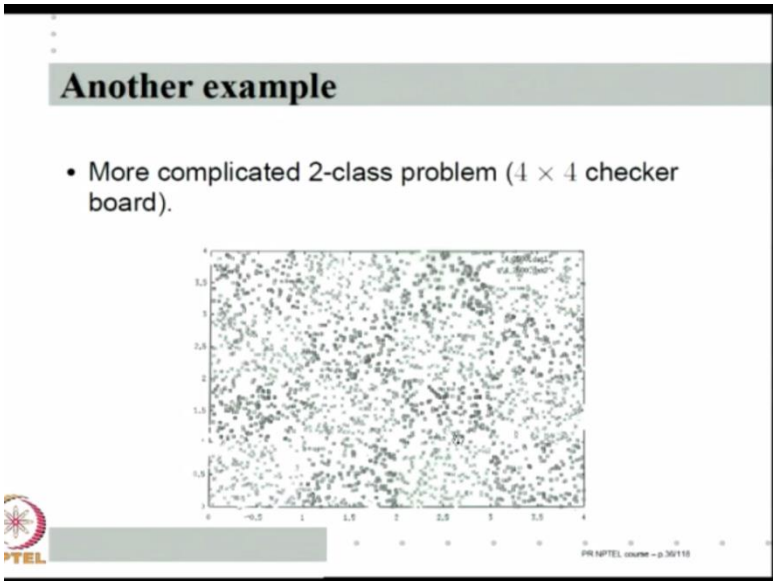
Here is the first example is a 2 class problem, one class is shown with slightly darker things than the other, actually we have classes and squares. I suppose all of you can see that it is a 2 by 2 checker board problem we have at the centre 2 lines. So, this quadrant and this quadrant is one class this quadrant and this quadrant is what is shown here are thousand samples uniformly drawn in the square and the, the classification is also given. So, these are the sample set and, if I did in SVM with Gaussian kernel and after learning the SVM, if I flag only the support vectors.

(Refer Slide Time: 19:38)



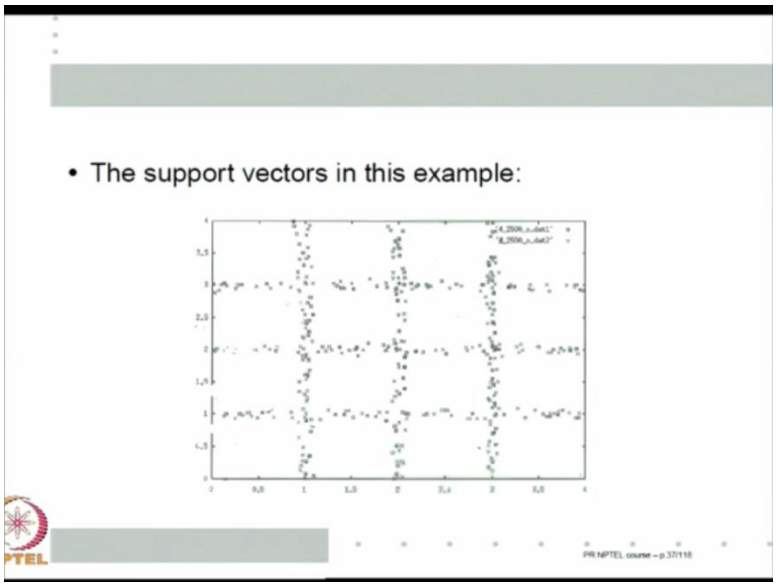
That is what I get as you can see these support vectors essentially, define the class boundary. These are all the vector that are closest to the class boundary of course, I use a Gaussian kernel. So, I am working in a very higher dimensional space learning hyperplane there, but then the patterns are closest to that hyperplane or this matters. So, essentially, the support vectors give me a very good idea of the geometry of the class boundary of course, pictorially um.

(Refer Slide Time: 19:38)



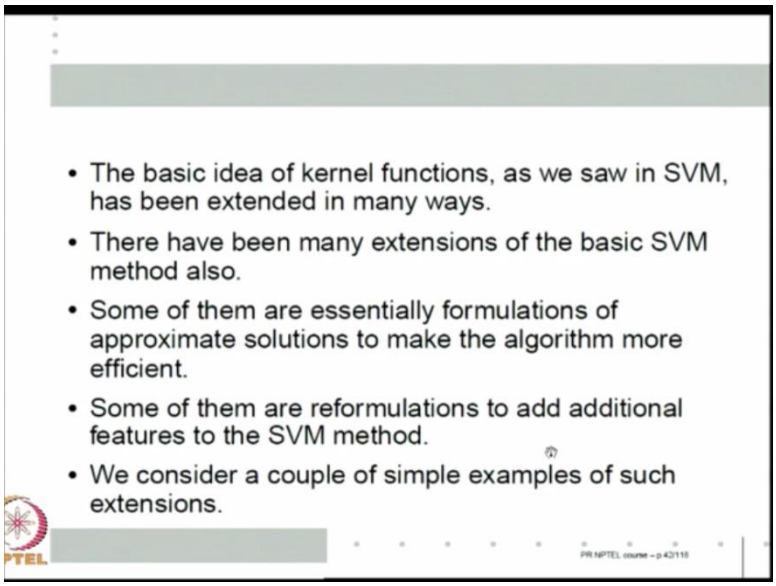
Here is another example, certainly more complicated actually it looks like a mess here, I plotted 2500 sample points. This is actually a 4 by 4 checker board. That is a, a 4 by 4 grid, where alternate squares are different classes. So, this is one class that is another class that is one class that is another class and so on. This is one class that is another class once again, this is class 1 and that is class 2 and so on. These are 4 by 4 square once again, using Gaussian kernel, if I learn SVM,

(Refer Slide Time: 20:47)



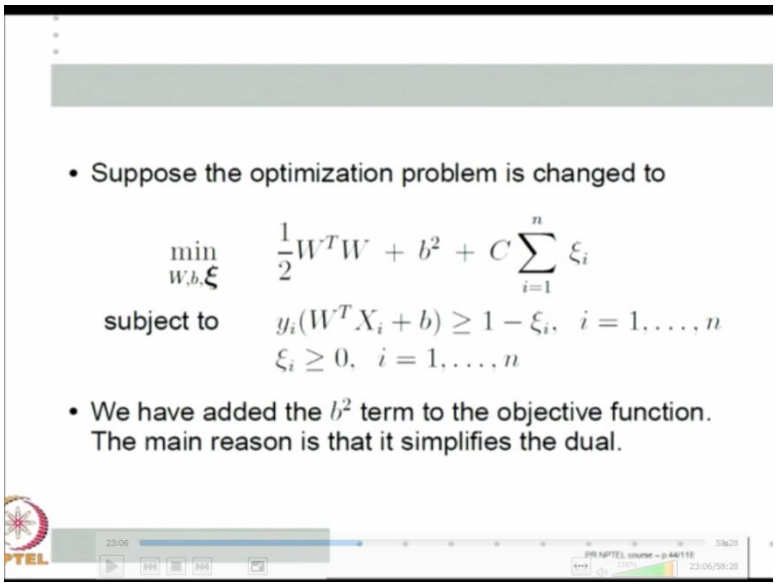
And then plot the support vectors. Those are the support vectors as you can once again see, I have neatly pulled out only the vectors, which are closest to the class boundary, rest of them can all be thrown out now. So, in this sense not only SVM is do a good classification in, in both these cases. The SVM classification is very has very high accuracy, but in addition the support vector themselves are a very useful extra information that we get out of this method.

(Refer Slide Time: 20:15)



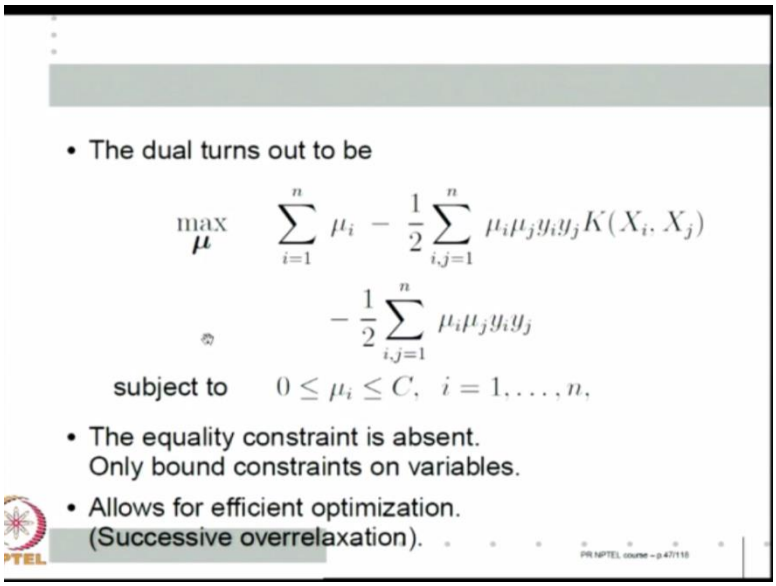
The basic ideas about vector machines as we seen is the kernel function that is what makes the method competitive the idea of kernel functions can extended in many ways. So, basically do the extensions mean basically, I can reformulate the SVM problem so, me of them are we can deformulate the SVM problems. So, that I get may be only an approximate optimal hyperplane by the algorithm will be more efficient, or I reformulate exactly, the sense I still get the optimal hyperplane that I can add some additional features to the original SVM method. There are many, many, many variations like this. Today for this course to stick to a time and other constraints, I have I am just going to illustrate it with just 2 simple examples, once of each kind.

(Refer Slide Time: 22:14)



There is, this is the way one does these things. So, I start with the basic SVM optimization problem and then, then I change the optimization problem in some ways which is useful to me. So, here is one way my original optimization problem is half W transpose, W plus time X i minimize, that subject to that supportability constrain y i W transpose X i plus P greater than 1 minus X i greater than equal to 0 that is the standard SVM problem with penalty constant. I get a b square the idea is, I am having my, my actual classifier is W transpose X plus b, but I am using only W transpose W b square now. I know why, I put transpose W because, that is the margin that I am minimizing b square has nothing to do there, but let us say we put the b square term anywhere, just like that we put the b square term then what happens,

(Refer Slide Time: 23:15)



What happens is this I what turns out to be the dual your objective function hasn't changed, not hasn't changed, this time is same as earlier, this time is same as earlier. I am just adding one more term in the objective function mu i mu j y i y j and, this is also a quadratic term, it makes no difference adding one more term into this quadratic optimum, but what I gain is that the equality constraint is lost there is no longer.

The equality constraint in the dual only bound constraints, why is such a great thing now it is no different from just minimizing unconstrained way a quadratic function for example, I can do gradient descent on this. If I do gradient descent on this normally in a constrained gradient descent, I have to do gradient projection, what do I do from the current point I calculate the variant direction move along that thing wherever I go from there I have to project myself back on the feasibly region now, the feasibility region is crazy. I do not know how to project myself.

But if the feasibility is only about bound constraints then what it means is I start from some mu travel along the gradient then at that point, if any of the mu is less than 0. I make that 0 for any of the mu is greater than C, I make it C that is all the projecting back in this constraint region is, it can be better than that because, it is a quadratic function if I take it is gradient and equate to 0, I get a similar linear equations. So, actually I have to solve a set of linear equations subject to these constraints, I can easily solve a set of linear equations by many relaxation methods. relaxation for example, relaxation simply

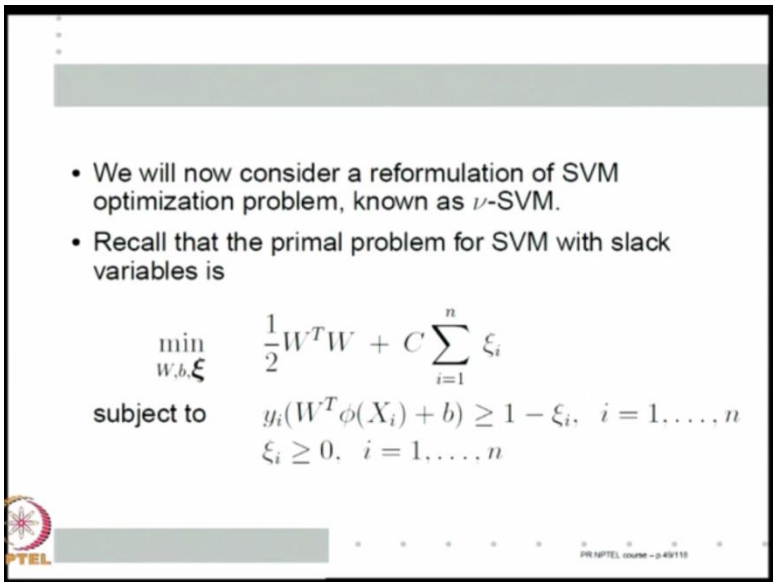
says, I start with some, some point, some solution point, then of the new value for the first variable from the first equation.

By using the old values of the other thing now, for this is new value for the second variable from the second equation, and so on. and if the it is very easy for me to implement these constraints because, after every time I calculate the update for mu i have to just clip it between 0 and C is that update turns out to be negative I make it 0. If it turns out to be greater than C i make it C right.

So, I can just take the, the, the variant which become just a set of linear equations and I can solve the linear equations under this bound constraint, using a simple algorithm such as relaxation, where iteratively I keep finding new values for variables by one using one, one equation for one variable. It is called, it is called successive over relaxation this makes very efficient algorithm. It, it is much more efficient almost 10 times more efficient that SMO.

But the problem is that, it is not solving the problem, I am interested in I am of course; it is its giving me a very efficient algorithm for solving a different problem. what is this, b square doing will that still give me optimal hyperplane of course, it does because my optimal hyperplane is a solution of the optimization problem with b square not there. But what can be shown is that if I do with a b square I would not be too far away, from the optimal hyperplane. So, I get only approximate optimal hyperplane finally, a good approximation and the (( )) I mean the, the, the advantage I am gaining for being only approximately correct on the optimal hyperplane is that solving my dual optimization problem is very, very efficient. So, this is one way one can reformulate.

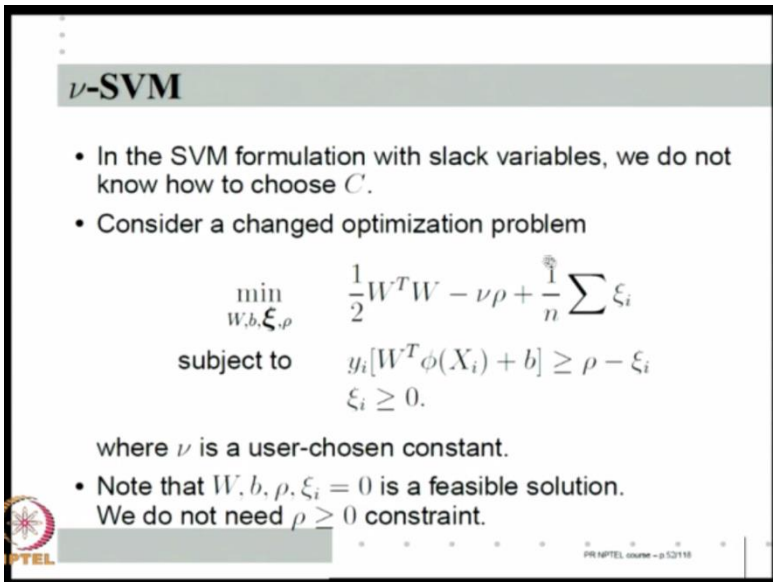
(Refer Slide Time: 26:57)



We look at another reformulation now, this is called the nu SVM and most people think that this nu is a (( )) on the SVM proposed this new SVM the idea behind the or the motivation behind new SVM is as follows. This is the primal objective problem that we that we seek to solve for SVM here, I put phi X i, instead of X i because, we are essentially learning this optimal hyperplane near high dimensional space and phi is the symbol for that transformation. Basically, this is the margin and basically, you put X i because, we do not whether things are separable or not. So, we use the slack variables X i. The problem with this optimization is, I do not know how to choose C. C is a kind of an exchange rate between margin, and some measure of error and the measure of error is very crazy here, because X i i greater than 0, does not have mean that term actually, make the error if X i i is better than 0.

But less than one the corresponding X i y i is correctly classified, but it still has 2 error and it does not tell me, how many I am incorrectly classifying. Instead of misclassifying only 1 but misclassifying by a large margin, I may settle for something that classifies 100 misclassifies, 100 patterns, but each one by small margin. Because by this the summation X i, I really do not know what it is doing. So, I I cannot give any physical meaning to C all in (( )), if C is very large, I will not tolerate any errors except, for that I have very little idea about whether, I should choose C to be 5 or 10 and what are the tradeoffs.

(Refer Slide Time: 28:45)

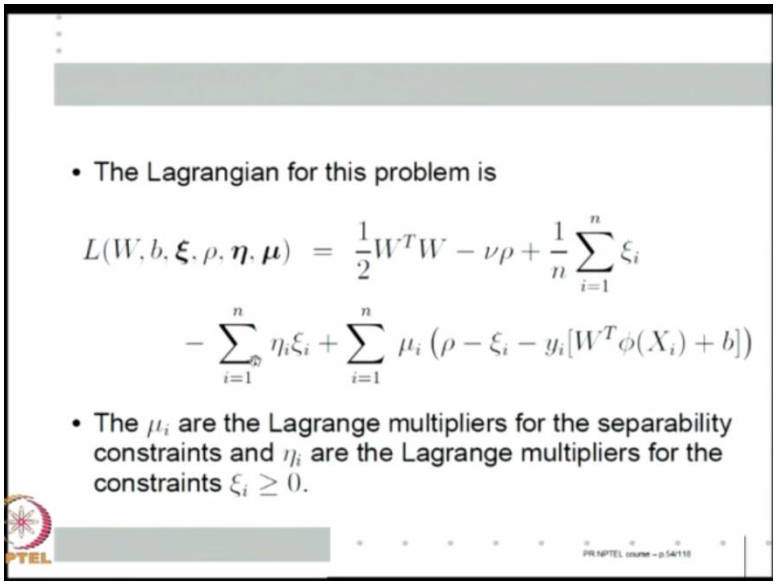


So, one problem with the SVM formulation slack variables is we do not have good guidelines on how to choose C it is very difficult to give guidelines because C does not have any simple interpretation So, let us say we we want to work with a optimization problem in which the use of constant like C has some meaning. So, I reformulate the primal objective function like this I still have are We transpose W I still have summation X i i instead of C. I put 1 by n here, where n is the number of samples and, add a one extra term minus nu rho nu is a parameter instead of C I am my new parameter is nu, nu rho is also optimization variable now in addition to W b and X i i I have added one more optimization variable rho and rho comes here in place of one

So, instead of say demanding y i and W transpose phi X i plus b greater than one minus X i. I am saying, it should be greater than rho minus X i. So, whether the margin is with respect to hyperplanes that are one distance away or rho distance away. I do not know, I am just saying take a rho distance away and optimize. Now, let us just consider this problem as it is and of course, if I want rho to play, put the place of one and, and now, that is what defines a margin then rho I need rho greater than 0. I do not need rho greater than 0, for the following thsi optimization is very cleverly chosen given this problem if the point W is all come as a W is equal to 0 or, b equal to 0 all components are X i equal to 0 and rho equal to 0. This is my optimization variables, all 0 is a feasible point and because, it satisfies all the constraints.

Because all 0 is feasible point and at that point, the objective function is value 0 which means the, the actual minima cannot be positive. Because, I already knew one feasible point at with the objective functions value 0. Your strictly positive value can be it is minima now, if I choose rho greater than 0, this term will be positive, I know W transpose, W is positive. I know X i i is are positive So, if I, if I choose any rho greater than 0, this will be strictly greater than 0. Now, such a point can never be the solution of this optimization problem because, I know a feasible point at which the objective functions value 0. So, we do not need rho greater than 0 constraint at all we can just minimize unconstraint on rho. So, let us start with the this problem.

(Refer Slide Time: 31:23)

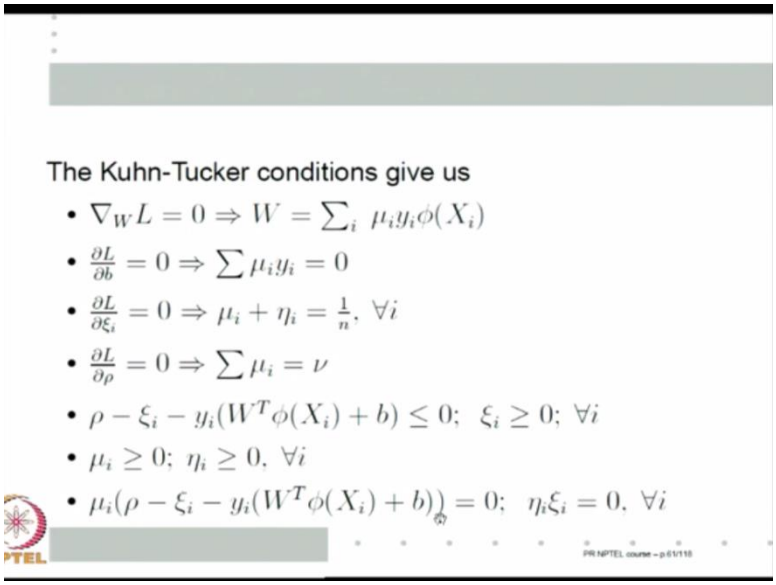


And form it is Lagrangian. So, recall that I have W b X i rho 4 optimization variables then I have n constraints like this. Another unconstraint like this, I have to write all my constraints as less or equal to 0 form so, this constrain becomes rho minus X i i minus y into W transpose phi X i plus b less or equal to 0. This constraint becomes minus X i i less than equal to 0, this constraint becomes minus X i i less than equal to 0 also, the (( )) that we are putting phi X i here so that we are already using the kernel trick and we ultimately put the kernel there that we are we are of keeping the notations.

So, that we are conscious that the formulation is in a transformed space so, with all this, this will be my L;agrangian. This is objective function half W transpose, W minus rho nu plus 1 by n summation X i i minus eta i X i i is for this constraint minus X i i less than

equal to 0, and eta are the corresponding Lagrangian variables and this is for this constraint summation mu i rho minus X i i minus y i W transpose phi X i plus b this is my Lagrangian right.

So, mu i are the Lagrangian multipliers for the separability constraints and eta is the Lagrangian multipliers for constraints X i i greater than equal to 0. Now let us put the Kuhn tucker conditions for this I have to first is so, derivative with respect to all the optimization variables should be 0 of l. So, there with expect to W 0 del by del b is equal to 0 with respect to X i is equal to 0 and del l, where del rho is equal to 0 del with respect to W is equal to 0 is such a thing because with respect to W nothing has changed, I get one W from here and I get.

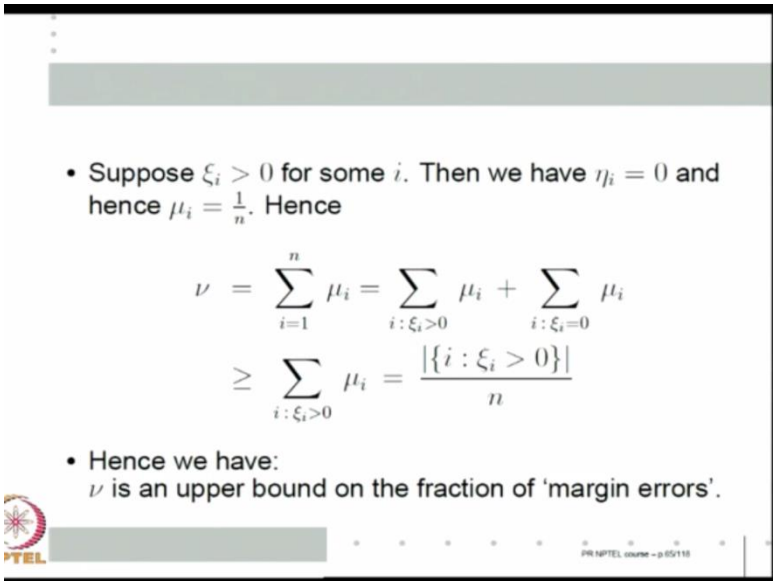


So, that will still give me this similarly, with respect to b, also nothing has changed so I put I get that with respect to X i. So, if I want to differentiate with respect to any particular X i i I get 1 by n from here minus eta i, from here and minus mu i from here so any (( )) with respect to any X i i will give me eta i plus mu i is equal to 1 by n. It is a very interesting thing the, the 2 Lagrangian multipliers or the different constraints in the primal I will (( )) like this similarly, del l by del rho will give me if differentiate with respect to rho will give me minus nu from here and summation mu i from here.

So, this nu that I have chosen as a nu is of different constant should be equal to summation, I is equal to one to n mu i interesting relationship. So, the Kuhn tucker conditions specifies that summation mu i should be equal to nu, then I have the

feasibility the way the 2 constant that should be satisfied then the Lagrange multipliers for the equality constraints shall be positive then, I have the complimental slackness because, eta i is for the constraint X i i greater than equal to 0, eta i X i i should be equal to 0. Similarly, nu I into this constraint should be equal to 0 these two are the complimental slackness condition.

(Refer Slide Time: 34:40)



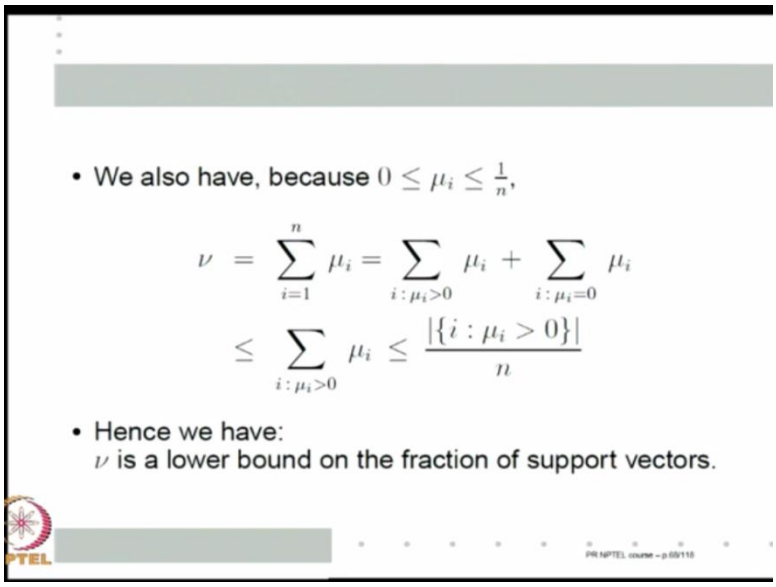
Now, why is this interesting suppose X i i greater than 0 for some I so, X i i is greater than 0. Then the corresponding eta is equal to 0 because, of the complimentary slackness once the corresponding eta is equal to 0, this tells me that the corresponding mu i should be equal to 1 by n. and I also know that nu is equal to summation mu i we can put all of these together suppose X i i is greater than 0 for some i. Then we have eta i equal to 0 and hence, mu i is equal to to 1 by n and hence nu is equal to summation mu I. Now, this summation mu i can be written as summation over all those I is such that, X i i greater than 0 and summation over all those i is, is that X i i equal to 0. For some I X i i will be X i for some i X i i will, will greater than equal to 0 the mu i is are positive.

So, I can if I (( )) this terms it is all, I am only finding a lower bound. So, I can always write this plus, this is greater than equal to only this term and this term has a very interesting structure. When X i i greater than 0 mu i is equal to 1 by n. So, each of these terms will be 1 by n how many systems are there as many i are there X i i greater than 0. So, I can say this is the, this set of I is are that X i i greater than 0. This is the number of I

for which X i i greater than 0 by n. What does X i i greater than 0 means? On that particular example, X i y i there is an margin error that is I am actually do not have as much margin, as I want that is X i i is are slack variables, X i i will be greater than 0 only if I makeshift arrangement making an error on the corresponding X i y i.

So, this numerator here is nothing but the number of examples on which I made error. So, what we have is that mu is greater than equal to the fraction of margin errors so, the new that I have chosen is such, that it has to be greater than or equal to the number of margin errors. So, if I chose mu to be point 1, when I, when I am solving this problem that means, I will not allow that means my optimization problem guarantees that any solution. I get is such that the number of i for which I have made margin errors is less than 10 percent. So, nu is an upper bound on the fraction of margin errors is not all.

(Refer Slide Time: 36:57)

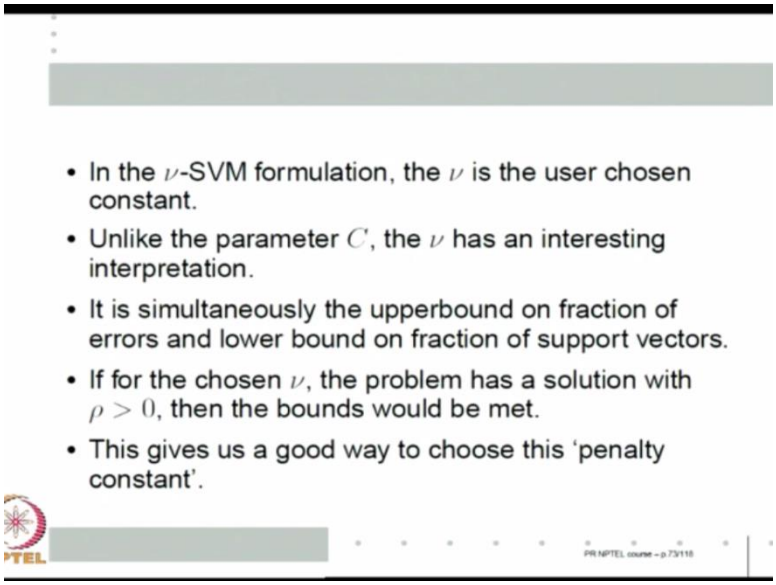


We also know that for all I, whether or not X i i greater than 0, mu i is between 0 and 1 by n and we known nu is equal to summation mu I. So, I can once again split that to mu i such that mu i greater than 0, mu i such that mu i equal to 0. This term does matter really and for all these mu I, I know it is less than equal to 1 by n So, I can write it less than equal to mu i such that mu i greater than 0, and we know what are mu i greater than 0 they are support vectors. So, nu is also, because each mu is less than equal to 1 by n summation mu i is less than equal to 1 by n times. The number of thing in the summation

and number of things in that summation is number of I is, is the mu i greater than 0 that is equal to the number of support vectors.

So, nu is a lower bound on the fraction of support vectors that is nu is always less than equal to fraction of support vectors. So, if I chose nu to be 1. I am, I am asking the optimization I am, I formulate the optimal problems such that at the optimum point, the fraction of support vector should be at least 10 percent and the number of margin errors have to be utmost 10 percent. Matter of fact, one can show that asymptotically as the number of examples goes to infinity nu will be actually equal to both. These fractions nu will be equal to the fraction of support vectors and fraction of margin errors. It is very useful so, I can decide what kind of performance I want by choosing my nu.

(Refer Slide Time: 38:36)



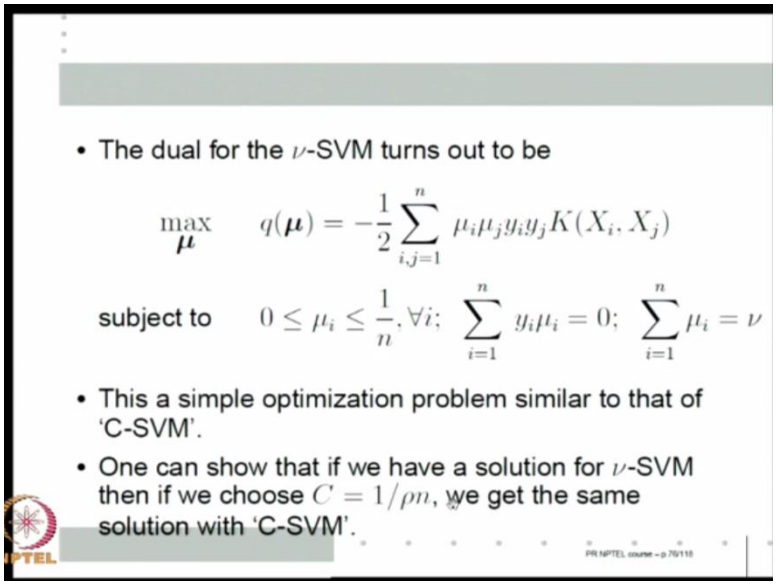
So, the nu in SVM formulation in the nu in the nu, SVM formulation the nu is a user chosen constant and unlike the parameter C. the nu has interesting interpretation nu will be between 0 and 1 and with of nu between 0 and 1. This will be, this will not have a solution because nu has to be a summation mu i and, mu i are always between 0 and 1 by n and given that nu between 0 and 1. It is simultaneously, it is a, it is a lower bound on the fraction of support vectors and is an upper bound on the fraction of margin errors.

Simultaneously, an upper bound on the fraction of errors and lower bound on the fraction of support vectors. So, what it means is that for the chosen nu the problem has a solution with rho greater than 0 of course, if I chosen an unattainable nu I cannot get a solution,

but what do you mean by I cannot get a solution it always has a feasible point. So, I will get a solution, but the solution will be trivial, I will get rho is equal to 0, but if I will get a solution with rho greater than 0, then the bounds would be met. This gives you a very good way of choosing the penalty constant unlike C, I can choose nu. And you know because, I have now good idea of so if I, if I am allowing, if I am willing to tolerate may be at most 10 percent margin errors then I choose nu to be 0.1.

I solve it suppose on this problem, I cannot get a solution, where I can only misclassify 10 percent points it turns over that the rho that I get would be 0. So, maybe I will change the 10 percent nu to 15 percent nu. But the nu has a nice interpretation and hence in that sense is much easier to choose, than the penalty constancy, this is the; this is the real utility of the nu SVM. So, can it be solved easily see my, my primal problem now looks much, much different from the old primal problem of old SVM, because I get a minus rho nu rho here and I changes to rho but much more importantly unlike the rho SVM. It has one more optimization variable, but as it turns out if I take the dual the, dual looks no different.

(Refer Slide Time: 41:10)



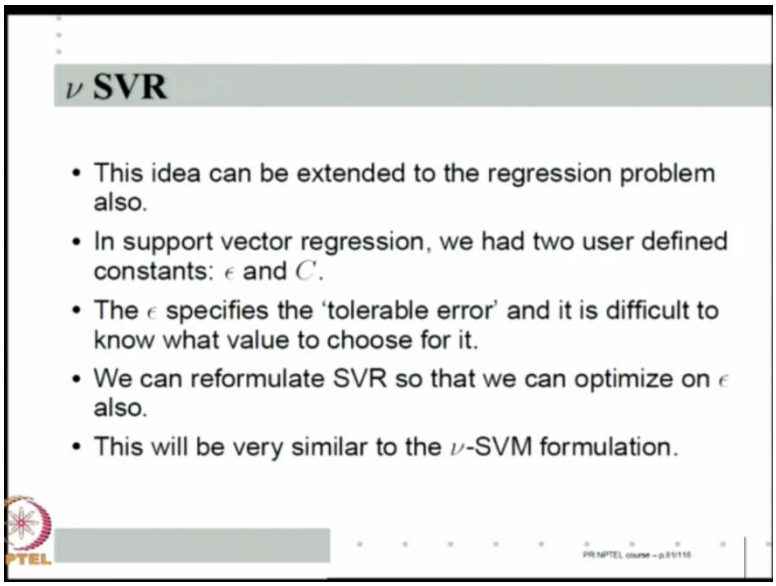
This is the dual for us all that happened is the, the quadratic term in, in the in the dual still stays mu i mu j y i y j K X i X j with a minus term the, the linear term summation mu i goes away does not matter. Now, I still have the old inequality constraint, I is equal to 1 to n mu i y i is equal to 0. The old equality constraint then I still have bound

constraint, but not there is no C. Now, as what was there before summation X i i is 1 by n.

So, I get mu i to be between 0 and 1 by n and I get this extra summation, i is equal to 1 to n mu i is equal to nu. So, it is exactly same as the old SVM, this external rho that I have put in really makes no difference. The objective function is still quadratic and, I still have similar kind of linear equality constraints and bound constraints of the variables. So, a simple optimization problem similar to that of old SVM so, it is very easy to solve once again the dimensionality is only n and so on, so forth. One can also show that this is not any approximation like the P S 1 I considered.

If suppose, you have a solution for nu SVM and ultimately once we solve we will not only get mu i is, but also get rho and get say, the rho is strictly greater than 0. That is what, we mean by solution for nu SVM. Then I go back to the old SVM formulation there I choose C is equal to 1 by rho times n, n is the number of examples, then 1 can show that you get the same solution with that old SVM algorithm as you got for nu SVM. So, this nu SVM essentially gives you a solution, you could have got for a particular value of C in that sense unlike adding b square in function I am not doing any approximation that, what I have done is, I have really cleaned up the formulation of SVM because, unlike the penalty constant C the use of different constant nu here has some interpretation.

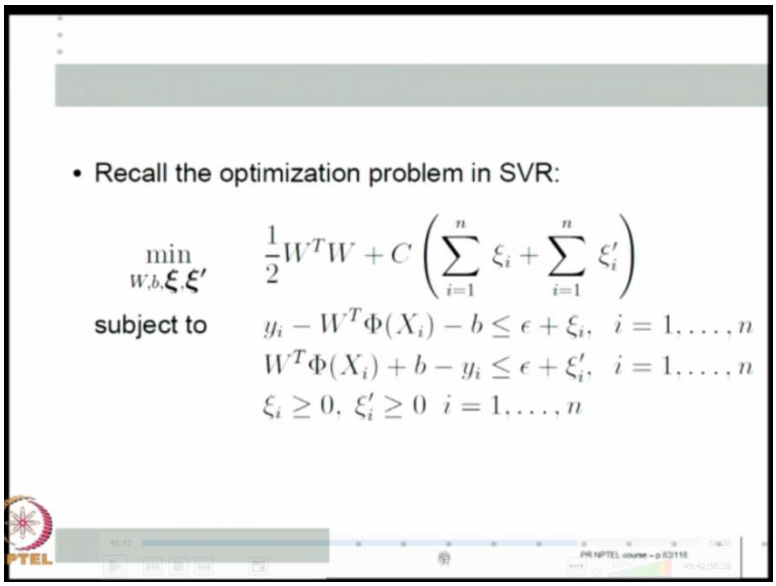
(Refer Slide Time: 43:20)



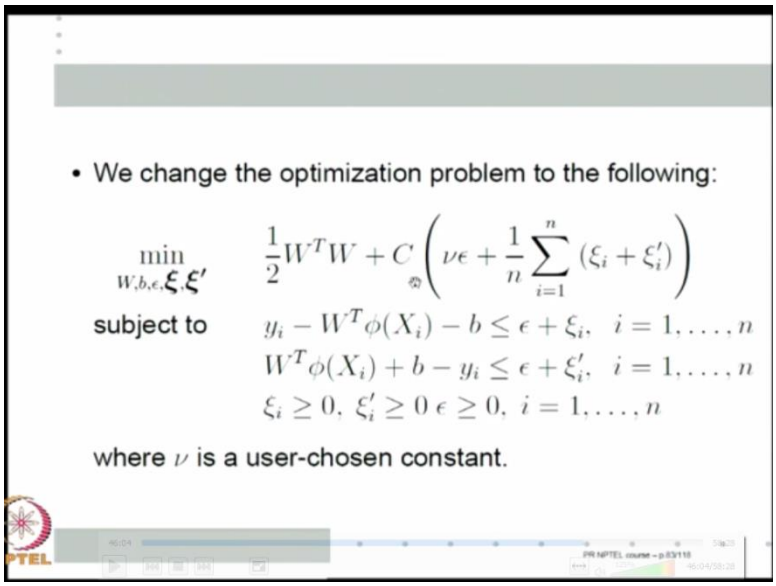
Now, this idea can be extended to regression problem also just you have got nu SVM. I can get nu support vector regression algorithm, let us call it nu SVR in the regression problem, what are the use of different constants one is epsilon, epsilon tells me see I if you recall your support vector regression algorithm. We're using the so called epsilon insensitive loss function. So, in that sense epsilon tells me the allowable error range if the actual value the difference between the actual value and my prediction is less than epsilon I suffer no loss unlike in the least squares case so to make a good guess on epsilon on this problem. If I put epsilon very large then useless predictions will come and even, if I put epsilon very small, I may not be able to find a nice function, which predicts to that level of accuracy.

So, because up to epsilon the loss function is nu 0 loss, I should use epsilon very intelligently and once again like the C there here. The C is not the real issue this epsilon is the one that is telling me, how I am trading accuracy. And I have no like epsilon specify the tolerance, tolerable error and is once again difficult to know for this data set. What is a tolerable error so like in the nu SVM case, we can reformulate support vector regression. So, that just like we added rho instead of one there and rho as a optimization variable. we can add epsilon as an optimization variable here this will give me something very similar to SVM nu SVM, it is like this

(Refer Slide Time: 45:03)



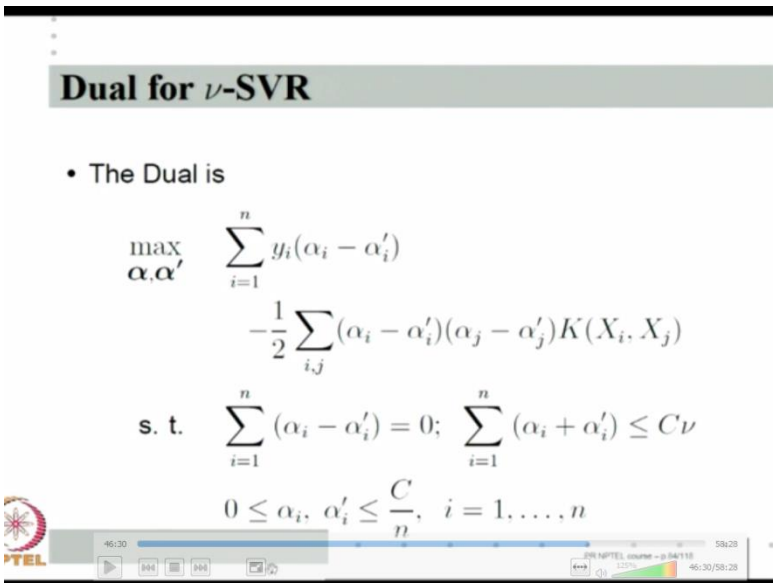
(Refer Slide Time: 45:46)



So, once again this is my old SVR optimization half W transpose W which we added as a parameter we already seen this is like the epsilon margin of a function C times, this because, this is simply a way to implement the epsilon in sense to a loss function. So, if (( )) of y i minus W transpose phi X i minus b is less than epsilon, when both X i i and X i i prime are 0, if on one side it is not less than epsilon, then the corresponding X i i will come here.

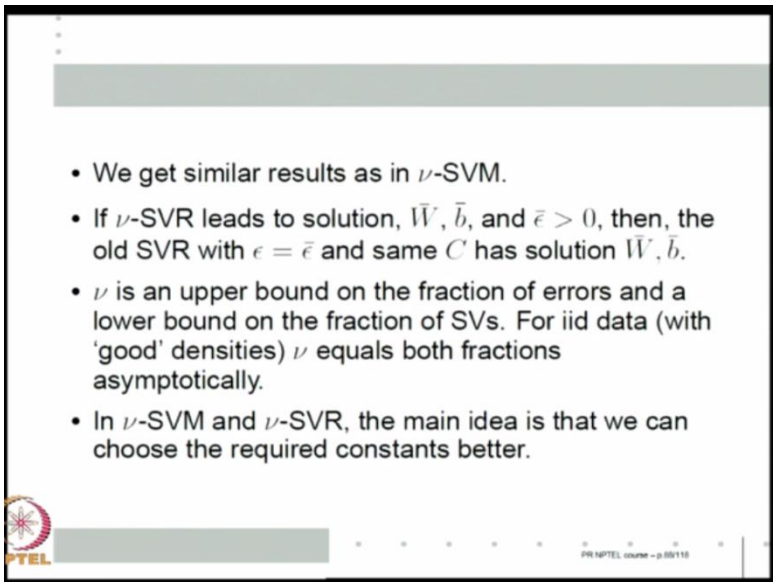
So, this is my whole formulation of the slack variables and in the nu formulation. I want to optimize an epsilon also, I add epsilon as an extra parameter for optimizing and I put nu epsilon inside here otherwise the constraints remain the same epsilon just became an optimization variables. And I am just saying on the C side in addition to your old parameters, I put nu epsilon where nu is a user chosen constant.

(Refer Slide Time: 46:11)



The dual for this once turns out to be quadratic, it is, it got once again very simple structure a, very similar to the old nu SVR. I would not go in details because, it is, its it is pointless, but the, the thing to recognise is that the, the objective function is still as one linear term and one quadratic term, and all constraints are linear. So, this is once again a very good nice optimization problem and, one can show that you get similar results as nu SVM. The whole idea of, why we did this is same as in SVM case, where we want to have a allowable accuracy. So, we want to choose nu based on how much accuracy we are looking for in our predictions. So, that you know I, I correctly optimize a my epsilon tolerance limit.

(Refer Slide Time: 47:08)



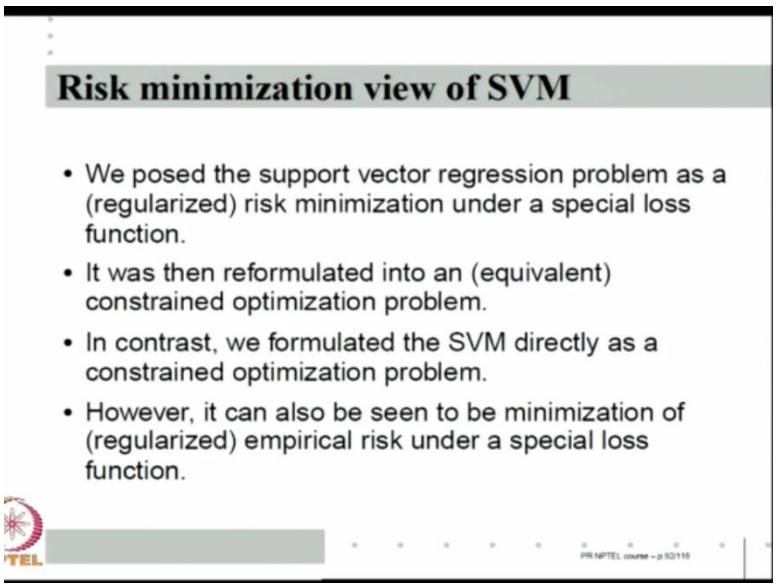
So, this is, this turns out to be true, this turns out to have similar kind of properties as the nu SVM case that is to say in the nu, SVR suppose nu SVR leads to solution, W bar b bar epsilon bar with epsilon bar strictly just like in the nu SVM. If, I choose a wrong nu I would not get a solution in the sense, I get a solution with rho is equal to 0 similarly, here so when I get a solution with epsilon strictly greater than 0. Then if I now run the old SVR with that value of epsilon and the, the, the same values for the other constant C, then I get back the same solution.

So, one like nu SVM, nu SVR also gives you a solution that old SVR could have given and at some particular parameter settings. The only thing is we beforehand do not know how to set this parameter because, for this data set we do not know, what is the tolerable error? That is why we assumed it is good except that once, once I get a particular epsilon bar for that epsilon bar. I get the; I get the same solution in the old nu SVR also, this is a actual exact formulation and once again you confer that nu is an upper bound and the fraction of errors meaning number of points on which prediction is greater than epsilon and lower bound on the fraction of support vectors. And like earlier, if you have sufficiently good data sets, sufficiently good meaning your large number of IID examples and your class conditional densities satisfies some proper conditions.

Then in such cases nu equals nu simultaneously equals both fractions asymptotically. So, as the number of examples goes to infinity nu is actually equal to the fraction of support

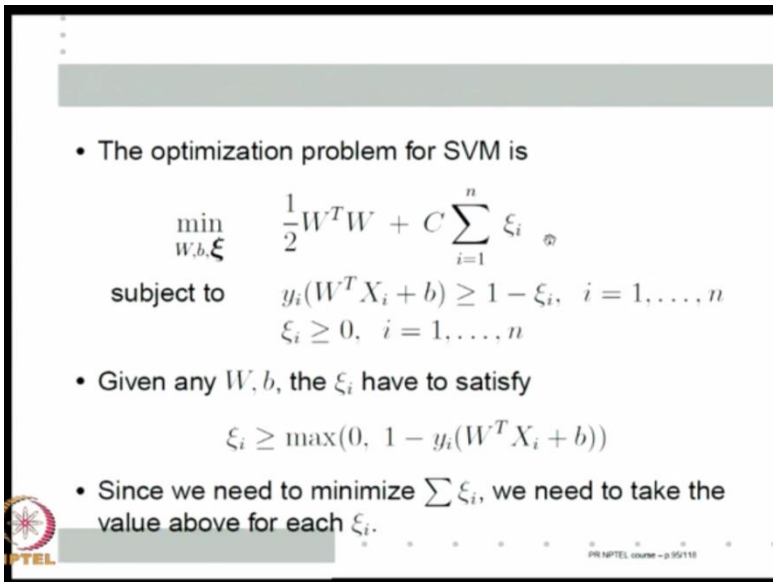
vectors, as well as the fraction of errors so, like an SVM it becomes very easy to decide, how to choose the required constants by the user. So, this is; this is an example of just reformulating the, the optimization problem for SVM. So, that the resulting optimization problem is a lot easier to choose the user defined constants while at the same time, we are still getting the same solution, as the old SVM and same solution mean, same solution at some parameters. So, we are not really sacrificing the idea of getting optimal hyperplane, but this allows us to choose our parameters much more simply much more easier to choose than the penalty constant C.

(Refer Slide Time: 49:50)



Now, I will look at one more issue with SVM is we posed SVM support vector regression problem as a risk minimization under a special loss function namely, epsilon insensitive loss function. And then reformulate, it as a equivalent constraint optimization problem for a support vector machine namely, the classification problem we directly pose the optimal hyperplane problem is a constraint optimization problem. Among all hyperplane that satisfy, the separabilty constraint, find the one with minimum margin. But know we know the connection between, constraint optimization equivalent unconstraint optimization problems at least some time, they can exist so one can ask is SVM can SVM be written like a risk minimizer, as it turns out. Yes SVM is also an empirical risk minimization algorithm under a special loss function.

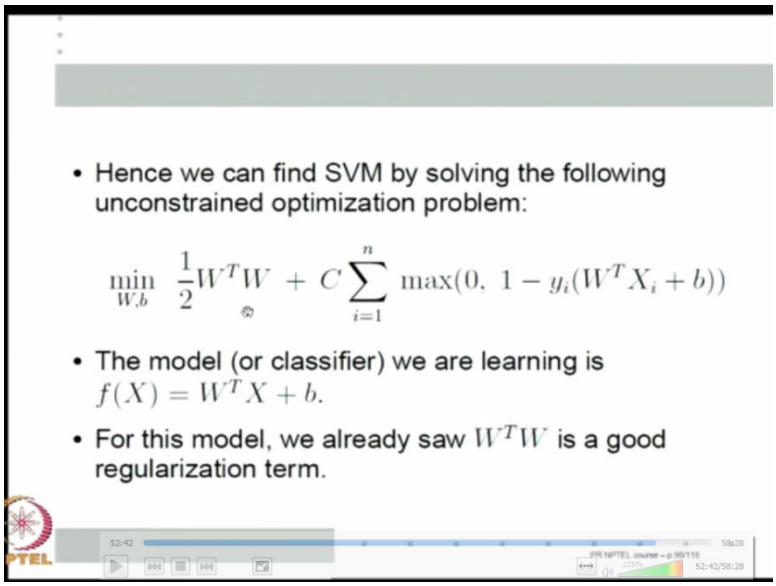
(Refer Slide Time: 50:38)



So, let us look at the SVM again this is, the SVM primal optimization problem half W transpose W plus C I with these constraints. So, what do the constraints mean the constraints mean, if you give a particular W on b these X i i is that I have to choose to maintain feasibility or such that X i i has to be greater than 0. And X i i has to greater than 1 minus y i into W transpose X i plus b.

So, X i i has to be greater than maximum of 0 comma 1 minus phi into W transpose, X i plus b ultimately. I am minimizing summation X i i so, as long X i i can be anything greater than this. So, the best X i i is to just simply choose this because, X i i has to greater than equal to this, and I am minimizing summation X i i I I just need to choose this as my X i. So, given a W b this is the best choice for X i and once I know that the best choice for X i i can put it there once, I put it there I will use up the constraints now. So, I can just do a unconstraint optimization.

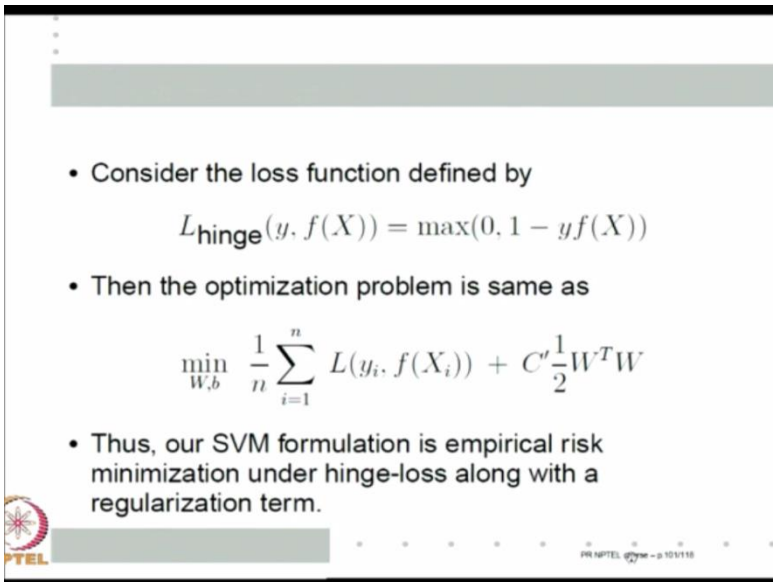
(Refer Slide Time: 51:37)



So, what this means is that solving the SVM problem this, this constraint optimization is same as solving this unconstraint optimization, instead of summation X i i. These X i i has to take only this value max of this now I do not need these constraints any more once. It takes this with respect to W and b and the corresponding X i i it satisfies these constraints. So, my SVM problem is same as solving the unconstraint optimization problem, minimize over W b of W transpose W plus C, I is equal to 1 to n max of this max of 0 comma 1 minus y i and W transpose X i plus b. Now the model we used is f X W transpose X plus b.

So, this is this term is like some, something y i and f X i this is f X i so, 1 minus y i f X i this is some function of y i and f X i. So, this can be a loss function, if this is a loss function this should be the regularization term. Of course, where that C comes is a matter of convenience. So, the end we know that is a regulation term because for this model, we already know that W transpose W is a good regularization term. So, this must be the empirical risk so, is this empirical risk.

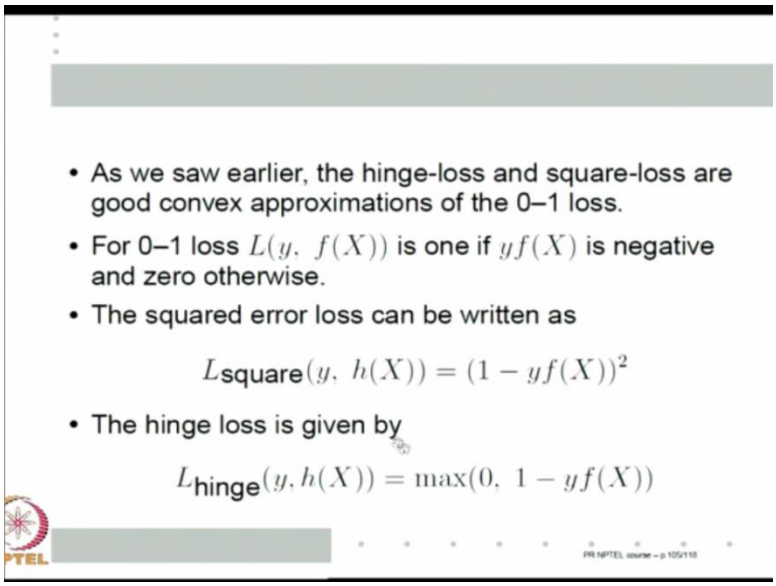
(Refer Slide Time: 52:50)



Yes of course, it is a empirical risk with respect to a particular loss function that we have seen earlier, when we studied loss functions, it is called the Hinge loss. So, the hinge loss for, for X y for a model, f is l of y comma f X is max of 0 comma 1 minus y into f X, that is what we have here, that is what exactly max of 0 comma 0 minus y into f X y i f X i. So, that is the binge loss, so with respect to hinge loss, if I want to do regularize a risk minimization. What should I do minimize over W b 1 by n because empirical risk I is equal to 1 to n l of y i f X i plus constant n is of W transpose W.

Because that is the regularization term l of y i comma f X i is nothing, but max 0 comma 1 minus y i f X i that is what, I have here. So, which means our SVM formulation is nothing but empirical risk minimization under the hinge loss along with a regularization. So, SVM is also no different from empirical risk minimization, it is just; it is just doing a empirical risk minimization under a special loss function called hinge loss. Which you have already seen as an example, loss function what we considered loss function matter of fact a truly that these are convex loss unlike the 0 1 loss function.

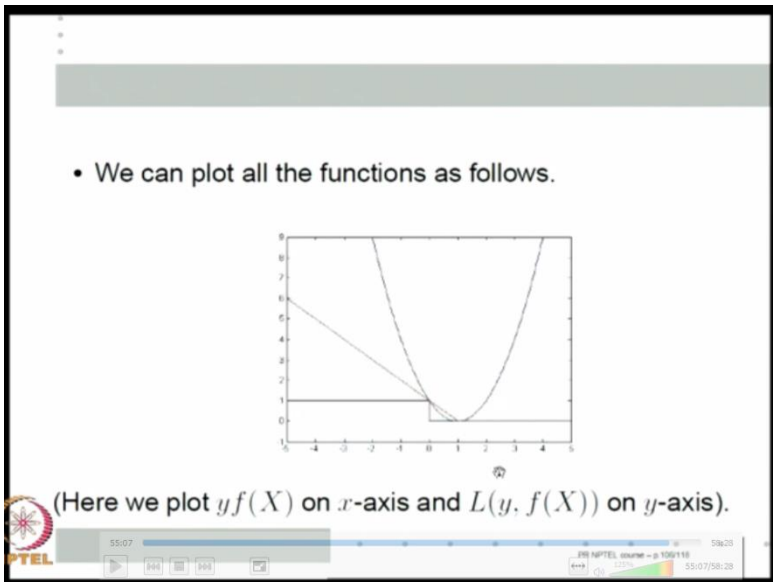
(Refer Slide Time: 54:11)



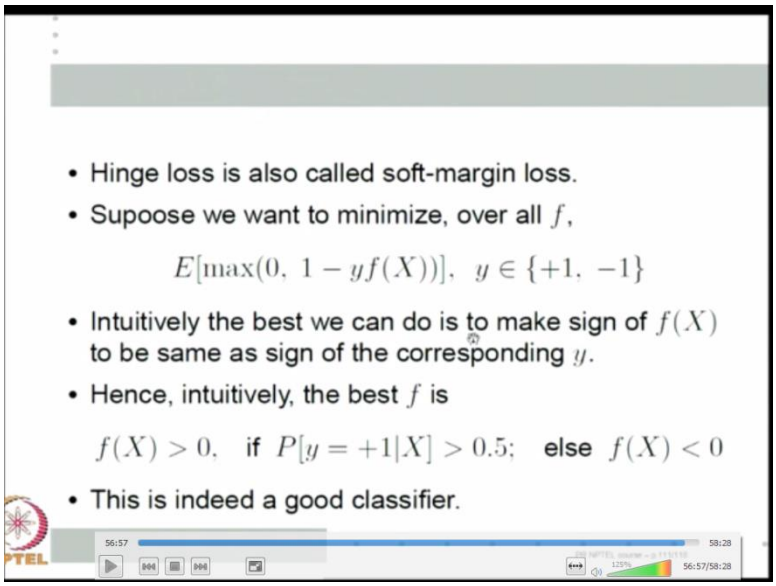
As we saw the hinge loss and square loss are good convex approximations of the 0-1 loss, when we studied loss functions, we said the 0-1 loss is nice because, minimizing 0-1 loss gives you classifier with minimum probability of misclassification. But 0 one loss is non convex and hence it is good to have some convex loss functions. So, let us just recall this 0-1 loss function is 1, we are considering y is plus 1 minus 1. So, and our classifier is sign of f X our model.

So, 0-1 loss function is a y and f X of the same sign when there is no loss otherwise, there is a loss. So, it is one if y f X is negative and 0 otherwise, as we already seen square error loss can be written like this because, when y is 1 is 1 minus f X whole square. When y is minus 1 it is minus 1 minus f X whole square, it is same as 1 plus f X whole square. So, this is the square loss function, this as we have seen the hinge loss.

(Refer Slide Time: 55:07)



(Refer Slide Time: 55:56)

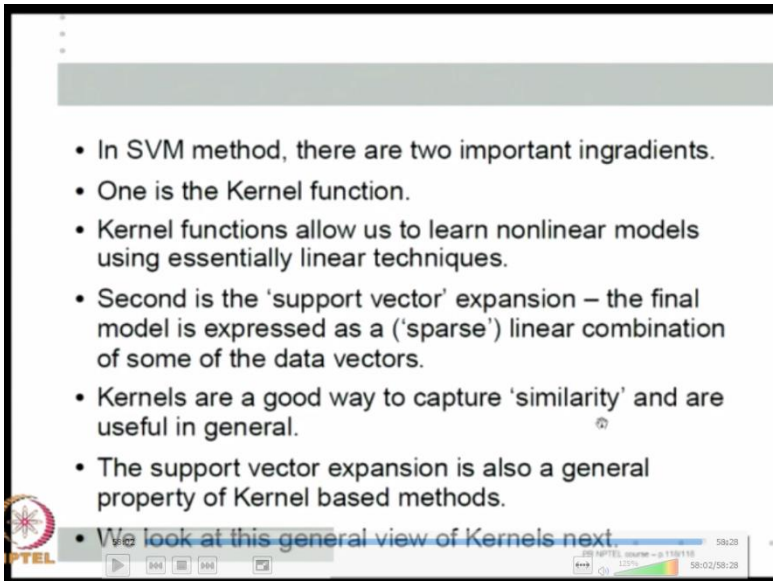


We can plot all these so, that is the 0-1 loss function plotted and the X axis, I am plotting y into f X and the y axis, I am plotting the loss function y comma f X. So, that is the 0-1 loss function, but that is non convex. So, squared error is one way of convexifying it and, the hinge loss is another way of convexifying it. Square loss is what say neural networks minimize hinge loss is, what SVM is minimize. Because, I convexified it nice in SVM because that is both pieces are linear.

I get a very nice optimization solution, the as we saw the empirical risk under regularize the empirical risk under hinge loss can be rewritten equivalently, as the constraint optimization problem, which can be solved very efficiently hinge loss. I also called soft margin loss suppose we, we are actually minimizing expectation with respect to soft margin loss say y is plus 1. So, we want to the f that models, such that we want to minimize expectation of max 0 comma 1 minus y f X what is our best. We want always this max 0 comma something to be 0.

So, we want this to be always positive, if possible that I can do if I always make y and f X of the same sign. So, essentially what it means is that, the best f is such that a probability y is equal to plus 1. It is greater than 0.5 given a particular X for that X f X should be greater than 0 and so, this is a very nice classifier. This is essentially bayes, bayes optimal classifier, if the posterior probability is greater than 0.5, I go for that class. So, essentially minimizing expectation under hinge loss, also gives us just like under 0-1 loss gives us a freely a good classifier that is that is one another way of, of looking at why SVM is perform better.

(Refer Slide Time: 57:10)



So, to go further on this, let us see that SVM method has two important ingredients; one is the kernel function, the kernel functions allow us to learn non-linear model choosing essentially, linear techniques. The second one is the support vector expansion, the final model is expressed as the linear combination of the data vectors and often sparse.

Because as the Lagrange multipliers might be except for the support vectors for the rest of them Lagrangian multipliers are 0.

So, that is the support vector expansion, the final model is expressed as a fast linear combination of the examples, though these are very useful things. Now, in general also these are good, kernels are good way to capture similarity, and are useful in general support vector expansion is also a general property of kernel based methods. So, in the next lecture, we will look at these two in detail; we will, we will look at capturing general similarities, using kernels and also when all such support vector expansions occur. So, next lecture, we will use slightly more theoretical tools and look at kernel based methods in general.

Thank you.