Introduction to Data Analytics

Prof. Nandan Sudarsanam and Prof. B. Ravindran

Department of Management Studies and

Department of Computer Science and Engineering

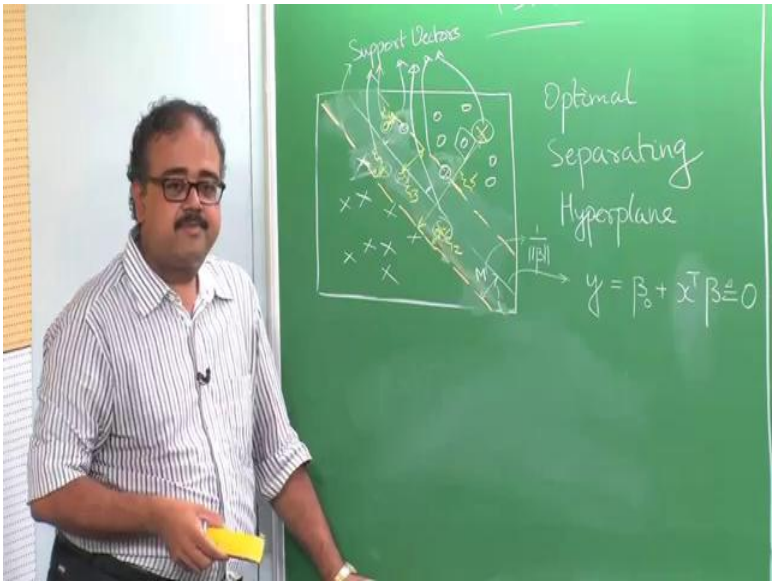
Indian Institute of Technology, Madras

Module - 05

Lecture – 32

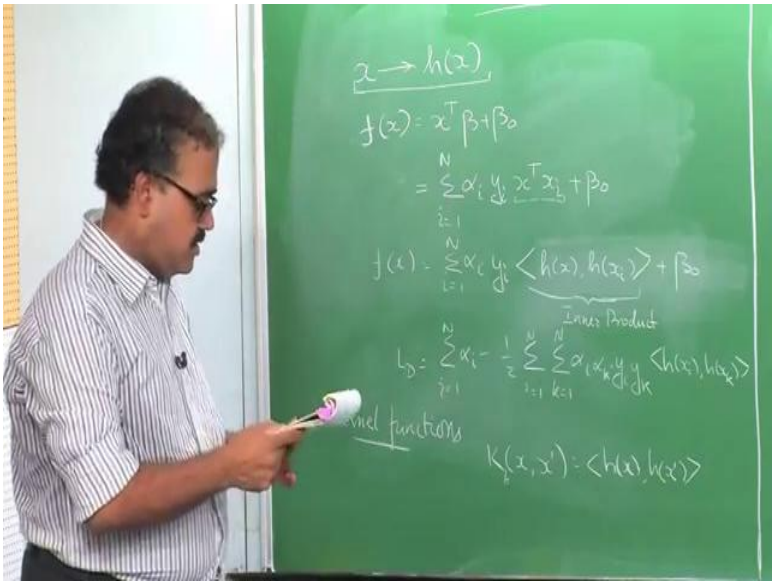
Support Vector Machines

(Refer Slide Time: 00:13)



So far, we have been looking at the problem of the optimal separating hyper-plane in the previous two modules, but then the idea of Support Vector Machines is to be able to use it in data which is not nearly linearly separable or only working in that space of linear hyper-planes, right.

(Refer Slide Time: 00:42)



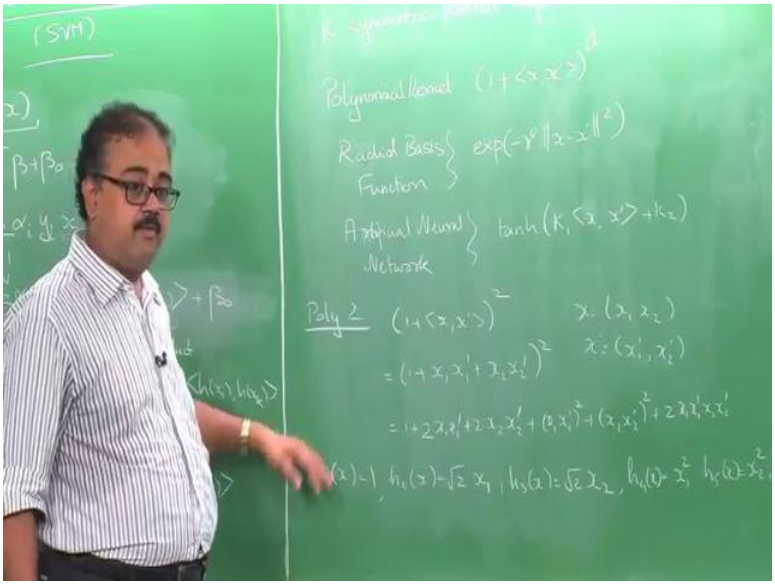
One way of looking at extending this problem to complex settings is to think of taking your original data, right and transforming it into something else, and then trying to apply the same idea to the transformed function. This idea should not be new to you because you have always seen this in linear regression where you can look at transforming the input variables into some other kind of basis function, and then trying to do linear regression on that. So, the idea is similar to that, but we are going to make use of a very powerful technique here, right. So, look at how predictor is going to work. So, we are going to have f of x equal x transpose beta plus beta naught. So, that is your predictor and if your prefix is lesser than 0, I will predict it as class minus 1. If your prefix is greater than 0, I will predict it as being of class plus 1, right.

As on this is, so this is separating hyper-plane that we have, fine. So, if you think about it, we said we have solution betas, right or going to be of this form. Therefore, I can rewrite this, right. So, if you look at it, interestingly so x, all the x's here appear as x transpose x i. This is the inner product of xi and similarly, if you look at how you are solving the optimization problem, this is what we wrote down last time. So, the duel is essentially going to have again x i transpose s k, right. So, you can see that the x's appears in our problem always as inner products, right and if somehow you are able to compute this inner products efficiently, then you should be able to solve the problem more efficiently. In particular, given that you are going to be looking at this kind of transformation, right. So, I can now write this as (Refer Time: 3:18), where this denotes

the inner products. This is what we can say inner product, right. So, likewise the dual also can be written as (Refer Time: 04:05).

So, if you stop and think about it, it essentially tells you that I really don't need to know h of x, right. If you have an efficient way of computing the inner product of the transformed function, right, then you really don't need to know what the actual transformation itself is, right. So, there is a class of function which I am going to call as kernel functions here. It is called as kernel function which allows you to compute this inner product efficiently. So, essentially it is going to say that ((5:25)), right. So, the kernel corresponding to function h, right when you give it as input x and x square, which is going to compute the inner product of h of x and h of x square, right.

(Refer Slide Time: 05:47)



So, one of the things that we require for a function to be a kernel function, right is there it should be symmetric. So, k should be symmetric, positive-definite. So, if we do not really understand that, so k should be symmetric in the sense that if I give it the set of x and x prime x prime, so the kernel functions for x and x prime should be the same as x prime, x. The positive-definite essentially means that if I take any vector x transpose k, x, that should always be positive. So, there are technical reasons for why this condition should be satisfied. For one last way to think about it to say that this essentially you would want this to be whole thing to be positive, so that your optimization problem will work as you wanted to.

So, that is rough intuition behind why you need this condition. Some of the popular choices for the kernel functions are the polynomial kernel. This is essentially 1 plus. I got the parameter d is something that we choose. The other one is Gaussian or the radial basis function. The other one is sometimes called the neural network kernel or the sigma del radial kernel. So, what do these kernels buy you, right? So, as I was mentioning earlier, you have a data there is given to in the original dimensions, right. The data might be badly mixed up in that original dimension, it might not be easily separable at all in the original dimension, but then when you look at the transformed dimension, then the data becomes linearly separable, right.

Let us take the example of the polynomial kernel and see what happens. So, I am going to look at the polynomial kernel of dimension two, right. It is essentially 1 plus ((9:50). So, I am assuming that vector x consists of x 1, x 2 and x prime consists of ((10:10)), right. So, these are like two-dimensional vectors and on which I am defining two-dimensional polynomial kernel. So, this is dimensionality of the kernel doesn't necessarily have to match the dimensionality of the underlined space, but in this case I am assuming this has. So, if we take the square of this, so I essentially end up with the expression that has six components to it, right.

So, if you think about it, this is somewhat like taking the inner product in a very higher dimensional space than the original space. Originally x and x prime were residing in a two-dimensional space, but if you look at what is happening now, it is essentially something like this. So, h of x is 1, h 1 of x is 1, h2 of x root 2 of x, the first coordinate; h3 of x is root of x 2, second coordinate; h4 of x with the x1 square, h5 of x2 square, h 6 of x root 2 into x1. So, if you imagine that I transformed my original two-dimensional representation into a six-dimensional representation is essentially taking all the second order terms along with original terms.

Now, if I take the inner product of h1 of x and h2 x prime, I will exactly end up with this expression, right. Inner product of h of x and h of x prime, right. So, the entire six-dimensional vector if I take the inner product, we are basically going to end up with this expression. So, what we have done here, we took the inner product in two-dimensional space, right and performed the squaring operation. So, this is going to give me number which is equal to the number I will get by first transforming the data point from two-dimensions is six-dimension and then, taking the inner product in the six-dimensional

space. So, essentially this allow to work in much higher dimensional space than originally intended, but by only looking at inner product computation in the original space, right.

So, why this is a useful property to have in the case of support vector machine, is that all over operation here operate only within inner product whether we are finally trying to predict the output f of x, or when you are trying to solve the dual problem ld. So, all we really need to do is know what the inner product is, right. Now, I am able to take the inner product in a higher dimensional space, but then do the computational only in the lower dimensional space. This allows us to have a much greater advantage than simply operating with the original dimension.

(Refer Slide Time: 13:50)

So, to see how this polynomial transmission really helps us, let us look at a very simple example, right. I am going to assume that the single dimension, right and then I have data points, let us assume this is 0. I have data point that look like this, right. So, this is the dimension x1. So, obviously there is no single line that I can draw to separate these into two classes neatly, right. So, I can assume my slack variables and try to draw the line here that says ok I am not making too many errors bla bla, so on and so forth, but still that is not the satisfactory solution, but let us looks at what happens if I try to block this data in two-dimensional plane, where I have x1 as one of my axis and x1 square as my other axis. So, two 0's will probably get mapped to somewhere here, right. So, that will

be x1 that is corresponding to this here and then looking at the square of that (Refer Time: 15:00), but then looking at these data points, so this is going to go here, this will probably go here, right.

Now, it is very clear that I can draw straight line, right. I can draw if can find the linear decision boundary that separates these two classes once I have done the appropriate transformation. So, this essentially allows us to solve larger class of problems. You see the kinds of bases transformation, then we could do just by operating in the original space and trying to solve the linear optimal hyper-plane problem. So, this is essentially what all your choices of kernel functions or all about, right. So, if you look at any SVM tool, they will tell you to pick one kernel function which is either the polynomial kernel. So, in which case you have to pick in the appropriate d or you have to pick radial basis function or Gaussian kernel, in which case you have to pick in appropriate gamma that tells you how fast the Gaussian is going to decay, or we can pick sigmoid or artificial neural network kernel where you have to pick kappa 1, kappa 2 which are the parameters that define how quickly this sigmoid function rises. We will see more about that when you look at neural networks.

So apart from this, you still have one further parameter that we will have to worry about which is this constant c, right. So, this tells you how much slack that you are willing to tolerate, right. So, if you think about it, if c is very large, if c is infinite, then we have to be in the completely separable case because even a very small value of zeta will cost this objective function to become very large, right. Even for a small value of zeta, right so you will find that this thing is actually very bad, right. C has to be if c is very large, then zeta has to be very small, right and if c is small, then zeta can be large. So, what this is going to tell you is that in the higher dimensional space, where you are assuming that the higher the dimensionality that you are projecting into, more likely is the data will be separated, right because if c is large, right because you are going to try in fit more complex surface, right. C will look, the surface will look very vigil, right and if c is small, then you will really get much smoother surface, but the possibility of you making errors is also higher. So, that's trade of that you have to figure out empirically by looking at how you are doing the, how you are performing in the actual data.

This brings us to the end of the module on Support Vector Machines and these are very powerful classifiers and quite often they are the first classifiers of choice for people

when they are trying to solve new problem which they really don't know much. So, one thing I should point out that people have come up with different kinds of kernel functions. So, I have given you three choices of kernels here. These are essentially the most popularly used kernel choices, especially if you are operating with text data, you would like to use linear kernel d's 1 and in most other forms, you will be looking at rbf kernels, but then for special forms of data like graphs and strings and so on and so forth, people have defined their own kernels and as long as they are satisfying your properties of the kernel function, you can define your own kernels and then have them help you solve the problem, right. That is topics for another day or perhaps for another course. So, we will stop here. Just let me reframe that SVM are one of the most popular and powerful classifier that are currently being used widely.

Thank you.