

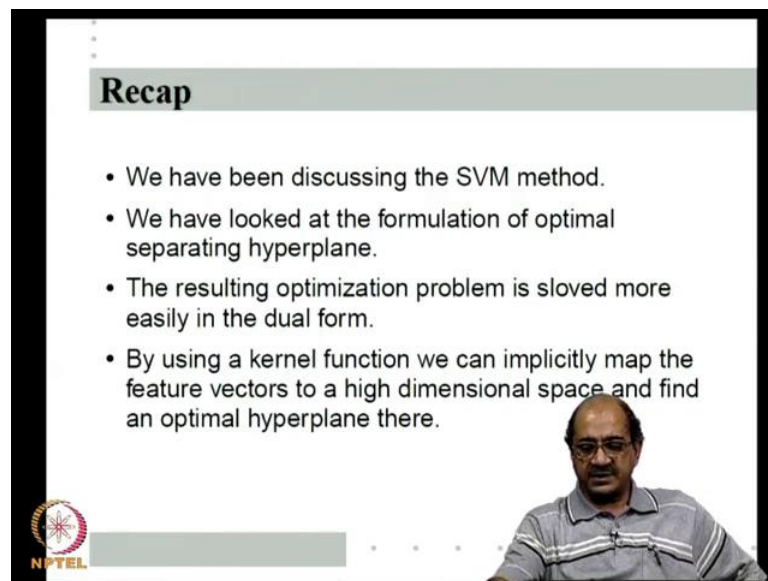
Pattern Recognition
Prof. P. S. Sastry
Department of Electronics and Communication Engineering
Indian Institute of Science, Bangalore

Lecture - 34

Kernel Functions for non-linear SVMs; Mercer and positive definite Kernels

Hello and welcome to the next lecture on this course on pattern recognition. We have been discussing the support vector method, we have looked at the basic support vector machine algorithm. So, this class we will we will just briefly review the support vector machine method. Then move on to study the kernel methods, the kernel functions which we only briefly touched upon we did not look at any kernels. So, this this class we will look at kernels in more detail, so to recap we have been discussing the support vector machine.

(Refer Slide Time: 00:51)



Recap

- We have been discussing the SVM method.
- We have looked at the formulation of optimal separating hyperplane.
- The resulting optimization problem is solved more easily in the dual form.
- By using a kernel function we can implicitly map the feature vectors to a high dimensional space and find an optimal hyperplane there.

NPTEL

Basically, it amounts to formulation of a optimal separating hyperplane problem. So, we seen how to define optimal separating hyperplane and generate an optimization problem whose solution is the optimal separating hyperplane. And we have seen that this problem is solved more easily in the dual form. By using a kernel function we can implicitly map the feature vectors to higher dimensional space and then find an optimal hyperplane there. This has been the overview of support optimization so far, so I will quickly recap this before going forward.

(Refer Slide Time: 01:29)

The optimization problem for SVM

- The optimal hyperplane is a solution of the following constrained optimization problem:
Find $W \in \mathbb{R}^m, b \in \mathbb{R}$ to

$$\begin{aligned} & \text{minimize} && \frac{1}{2}W^T W \\ & \text{subject to} && y_i(W^T X_i + b) \geq 1, \quad i = 1, \dots, n \end{aligned}$$

- Quadratic cost function and linear (inequality) constraints.
- Kuhn-Tucker conditions are necessary and sufficient. Every local minimum is global minimum.

NPTEL © NPTEL course - p 01158

So, the optimal hyperplane is a solution of the following constrained optimization problem, the optimization variables are W on b recall that our notation is that our feature vectors are in m dimensional space. So, W is also an R^m b in R , so you want to minimize half of $W^T W$ subject to $y_i(W^T X_i + b) \geq 1$. The X_i y_i are the examples they are n examples and for a given W b if $W^T X_i + b \geq 1$ y_i is greater than 1, which means if y_i is plus 1 $W^T X_i + b$ is greater than 1 y_i is minus one $W^T X_i + b$ less than minus 1.

So, these are the separability constraints, separability, so that if I want to take $W^T X_i + b$ is equal to 0 as my optimal hyperplane then on the the margin is given by the space between $W^T X_i + b$ is equal to 1 and $W^T X_i + b$ is equal to minus 1. So, if for all i is equal to 1 to n if these constraints are satisfied then this particular W b represent such a separating hyperplane and for that hyperplane this is the inverse of the margin. So, by minimizing this we maximize the margin and we minimize this over only that W b , which represent a separating hyperplanes.

So, that is how optimal separating optimal hyperplane is a solution of this problem where you are saying find the find the separating hyperplane. Those are my constraints such that it has the maximum margin because margin is inverse of this to maximize margin we are minimizing this. This is a very nice optimization problem we have got a quadratic

cost function, linear constraints it is almost a delight for any optimization theory algorithms you are going to solve such a problem.

Kuhn Tucker conditions are necessary and sufficient every local minimum is a global minimum. So, it is a very nice and simple optimization problem to solve. We showed that we can find the dual of this problem if you this is the primal problem, so you can find the dual of this problem dual will be on the Lagrangian multipliers. There are n inequality constraint there will be n Lagrangian multipliers called the mu 1 mu 2 mu n.

(Refer Slide Time: 03:50)

• The dual of this problem is:

$$\max_{\mu \in \mathbb{R}^n} q(\mu) = \sum_{i=1}^n \mu_i - \frac{1}{2} \sum_{i,j=1}^n \mu_i \mu_j y_i y_j X_i^T X_j$$

subject to $\mu_i \geq 0, \quad i = 1, \dots, n, \quad \sum_{i=1}^n y_i \mu_i = 0$

• Then the final solution is:

$$W^* = \sum \mu_i^* y_i X_i, \quad b^* = y_j - X_j^T W^*, \quad j \text{ such that } \mu_j > 0$$

NPTEL

Then the dual maximizes this function of mu. We derived the dual last class subject to these constraints I would not go into the derivation, but this we derived in last class. So, and the primal dual relationship tells me that if this has a solution, so has the dual and the optimal values are equal and more importantly for us from the Kuhn Tucker conditions for example, W star is given in terms of the optimal Lagrangian multipliers. So, if I know all the Lagrangian multipliers I I know immediately the W on b. That is the final solution final solution, W star is mu i star y i X i and b star is y j minus X j transpose W star for some j's such that mu j mu j star.

This should have been... I am sorry mu j star is greater than 0 this comes from your equating the gradient of your Lagrangian to 0 and this comes from the complimentary surplus condition as we derived last time. So, if I solve this dual I can obtain the mu I stars and once I get the mu i stars I got the my final solution W star and b star note that

the dual is also a quadratic cost function linear constraints. The the dimensional to the dual is n, which is equal to number of examples irrespective of the dimension of X_i and the input vectors X_i themselves come only as inner products here.

Now, unfortunately this problem has no solution if training data are not linearly separable this is the primal we started with if there is not even one w, b that satisfy these constraints and there is no physical solution and hence no optimal solution. So, the problem has no solution if training data are not linearly separable then like in any standard optimization problem we use slack variables.

(Refer Slide Time: 05:37)

- This problem has no solution if training data are not linearly separable.
- Hence, in general, we use slack variables.
- The optimization problem now is

$$\min_{W,b,\xi} \frac{1}{2}W^TW + C \sum_{i=1}^n \xi_i$$

subject to $y_i(W^T X_i + b) \ge 1 - \xi_i, \quad i = 1, \dots, n$

$$\xi_i \ge 0, \quad i = 1, \dots, n$$

NPTEL

PR NPTEL course - p 13158

So, the problem becomes y_i into $W^T X_i + b \geq 1 - \xi_i$ is what we wanted if we cannot get it greater than or equal to 1 minus ξ_i . Now, given any W on b for every X_i, y_i in my examples. So, that I can find a X_i is either this is satisfied, so every W, b is feasible now. There is no problem about any feasible solution, but now I cannot just minimise $W^T W$ because if I am not satisfying this constraint fully right ξ_i is my slack variable. So, this much slack is there in satisfying this constraint. So, I want to satisfy it with at least violation as possible, so also are some of X_i we are assuming $\xi_i \geq 0$ I am not assuming constraint $X_i, \xi_i \geq 0$.

So, each constraint if I cannot if I can correctly satisfy it if there is a separating hyperplane, then possibly ξ_i could be 0 if X_i are 0 then it will be like old old

solution. But even if I do not have a separating hyperplane then I can I would like to satisfy these constraints as much as possible. So, I use slack variable X_i and add the summation X_i to the cost function. So, I minimize half $W^T W$ also plus some constant times summation I is equal to $n X_i$ this tells me the total violation of my constraints C is as used as used as specified constant as we discussed last time essentially this will this this optimization problem will always have a solution.

If I take C larger and larger it essentially means that I am preferring only separable hyperplanes, we will not tolerate even little bit of violation of constraint if C is small I am allowing violation of constraint to improve margin. Unfortunately there is no simple way to say how C affects this. We will we will discuss this a little while later, but right now we will simply say that remember that C is as used as different constant. Using the slack variables allows us to ensure that even if the data is not linearly separable this this optimization problem will always have a solution. So, this is the optimization problem I want to solve.

(Refer Slide Time: 08:04)

• The dual problem now is:

$$\max_{\mu} \quad q(\mu) = \sum_{i=1}^n \mu_i - \frac{1}{2} \sum_{i,j=1}^n \mu_i \mu_j y_i y_j X_i^T X_j$$

subject to $0 \leq \mu_i \leq C, \quad i = 1, \dots, n, \quad \sum_{i=1}^n y_i \mu_i = 0$

• The only difference – upper bound on μ_i .

• We solve the dual and the final optimal hyperplane is

$$W^* = \sum \mu_i^* y_i X_i,$$

$$b^* = y_j - X_j^T W^*, \quad j \text{ such that } 0 < \mu_j < C.$$

NPTEL logo and footer text are visible at the bottom of the slide.

Very nicely the way we formulated it the niceness about the formulation is that its dual is very similar to the old dual. The cost function does not change this constraint does not change only change is that earlier I had only $0 \leq \mu_i$ I have $0 \leq \mu_i \leq C$. Essentially my primal has C here if a C tends to infinity this becomes same as the old problem. So, in the dual that is the only

thing that comes, which is very convenient for me. If I want to solve the primal, now I have increased the number of optimization variables all my data come in the constraints. But if I am solving the dual putting C or not putting C is simply a matter of while solving this problem whether or not I have an upper bound on μ_i .

That is the only only difference, so when I am solving the dual I can just throw in a C solve it. If I want I can increase the C constraint and de solve it to C how my μ_i change that tells me as I am finding a good separating hyperplane or not and so on. So, when I am solving in the dual domain whether I put that extra constraint or not it is simply a matter of putting one extra upper bound on all my Lagrange multipliers the solution is still the same. Once, I solve this $W^* = \mu_i^* y_i X_i + \sum_j y_j \text{transpose } W^*$ for some j. Now, the computed slackness means that this is for some j such that μ_j is strictly between both 0 and C.

(Refer Slide Time: 09:38)

Non-linear discriminant functions

- The idea is that we map the feature vectors into a high dimensional space and find a linear classifier there.
- In general, we can use a mapping, $\phi : \mathcal{X}^m \rightarrow \mathcal{X}^{m'}$.
- In $\mathcal{X}^{m'}$, the training set is

$$\{(Z_i, y_i), i = 1, \dots, n\}, Z_i = \phi(X_i)$$
- We can find optimal hyperplane by solving the dual.
(Replacing $X_i^T X_j$ with $Z_i^T Z_j$).

NPTEL

Earlier it is, we were only saying μ_j is strictly greater than 0. Now, given this then we moved on to ask how I can use this method to solve non-linear classifiers to obtain non-linear classifiers. The idea is that is the basic SVM idea is that we have mapped the feature vectors into higher dimensional space and then find a linear classifier there what does that mean I originally I am in the m dimensional space. So, I may use a transformation phi that maps \mathbb{R}^m to $\mathbb{R}^{m'}$ phi even though is non bold here, it is a vector valued function remember that. So, in in the $\mathbb{R}^{m'}$ space the new space what

will be the training set the training set will be $Z_i y_i$ where Z_i is equal to ϕ of X_i earlier the training set is $X_i y_i$.

Now, it is in the m dimensional space, now in the m prime dimensional space the training set will be $Z_i y_i$ where Z_i is ϕ of X_i . Now, with respect to Z_i , I want to find a linear classifier that is same as we can find an optimal hyperplane by solving the dual. What does that mean? My dual is this the only way X 's come is here. So, I am the these are the feature vectors, so if I am finding a linear classifier in the Z domain this simply X_i transpose X_j becomes Z_i transpose Z_j . So, we can solve the dual by replacing X_i transpose X_j with Z_i transpose Z_j that means that this is the dual I will solve, so it is very simple.

(Refer Slide Time: 10:56)

• So, we now solve the dual:

$$\max_{\mu} \quad q(\mu) = \sum_{i=1}^n \mu_i - \frac{1}{2} \sum_{i,j=1}^n \mu_i \mu_j y_i y_j Z_i^T Z_j$$

subject to $0 \leq \mu_i \leq C, \quad i = 1, \dots, n, \quad \sum_{i=1}^n y_i \mu_i = 0$

• The dual is optimization over \mathbb{R}^n .

• We solve the dual and the final optimal hyperplane is

$$W^* = \sum \mu_i^* y_i Z_i,$$

$$b^* = y_j - Z_j^T W^*, \quad j \text{ such that } 0 < \mu_j < C.$$

NPTEL

PR NPTEL course - p.24158

Basically if I calculate all the Z_i 's Z_j , I say is equal to ϕ X_i 's from my given training set I just have to replace X_i transpose X_j by Z_i transpose Z_j . Once I solve this I can find my optimal W in \mathbb{R}^m prime W^* is $\mu_i^* y_i Z_i$ b^* is this, the nice thing is that my original feature vector is m dimensional space may be 100 m dimensional space. My transformer feature vector may be in terms of the n dimensional space, but the dimensionality of the dual does not change.

The dual is an optimization of \mathbb{R}^n , where n is the number of examples it does not depend on what is the dimension of X is or what is the dimension of Z is. So, that is a very nice thing, another advantage of solving in the dual domain and we can solve this

and find this the only problem. Of course, is to do this I have the examples for each X_i I have to calculate Z_i that may be a costly calculation. Then, of course I do not have to do it in, I can only once compute $Z_i^T Z_j$ even that is costly computation and then I solve this dual then I have to calculate W^* , which is a in the high dimensional space. From, now on every time you give me a new vector X , I have to find Z is equal to $\phi(X)$ and then I have to do $Z^T W^*$.

(Refer Slide Time: 12:35)

Kernel function

- Suppose we have a function, $K : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$, such that

$$K(X_i, X_j) = \phi(X_i)^T \phi(X_j)$$
 Called Kernel function.
- Suppose computation of $K(X_i, X_j)$ is about as expensive as that of $X_i^T X_j$.
- Replacing $Z_i^T Z_j$ by $K(X_i, X_j)$, we can solve dual without ever computing any $\phi(X_i)$. Efficient for obtaining optimal hyperplane.

NPTEL PR NPTEL course - p.27158

So, there is lot of computation to be done even though I can automatically find a optimal hyperplane in the high dimensional space by solving the dual, but even this can be avoided. This is what we said last time suppose there is a function K that takes 2 m dimensional vectors and gives my real number. So, that $K(X_i, X_j)$ is $\phi(X_i)^T \phi(X_j)$ there is $Z_i^T Z_j$ we call say a function kernel function. What do you mean there is such a function? Of course, there is such a function if you give me ϕ I can define $K(X_i, X_j)$ to be $\phi(X_i)^T \phi(X_j)$.

What we mean is that ϕ might be very expensive function $\phi(X_i)$ might be 10,000 dimensional vector, whereas X_i is only 100 dimensional vector in that sense $\phi(X_i)^T \phi(X_j)$ might be 100 times more expensive than $X_i^T X_j$. Because $X_i^T X_j$ involves only 100 multiplications $\phi(X_i)^T \phi(X_j)$ may involve 10,000 multiplications by saying that there exists a kernel function. We mean that this function is such that computation of $K(X_i, X_j)$ is roughly about as expensive as

computation of $X_i^T X_j$ for such a function then I can replace $Z_i^T Z_j$ by $K(X_i, X_j)$ and can solve which means in my dual.

I do not have to actually calculate all the Z_i 's this $Z_i^T Z_j$ is replaced by $K(X_i, X_j)$ which is what as costly to compute as $X_i^T X_j$. So, it is like I am just finding a linear classifier individual space because I replace $X_i^T X_j$ what was there in the original linear classifier by $K(X_i, X_j)$, which is about roughly same computation. So, at least for solving the dual the kernel function is useful, now what happens after I solve the dual? After I solve the dual I have obtained W^* . So, even though in the dual I can replace $Z_i^T Z_j$ by $K(X_i, X_j)$ ultimately W^* lives in the m dimensional space. So, does it mean that I have to always take every vector into the m dimensional space and then use W^* to classify it? No.

(Refer Slide Time: 14:35)

Kernel function based classifier


- Let μ_i^* be soln of Dual. Then $W^* = \sum \mu_i^* y_i \phi(X_i)$.
- Then we have (for j s.t. $0 < \mu_j < C$)

$$b^* = y_j - \phi(X_j)^T W^* = (y_j - \sum_i \mu_i^* y_i \phi(X_i)^T \phi(X_j))$$
- Given a new pattern X we only need to compute

$$f(X) = Z^T W^* + b^* = \phi(X)^T W^* + b^*$$

$$= \sum_i \mu_i^* y_i \phi(X_i)^T \phi(X) + b^*$$

$$= \sum_i \mu_i^* y_i K(X_i, X) + (y_j - \sum_i \mu_i^* y_i K(X_i, X_j))$$


© NPTEL course - p. 32/158

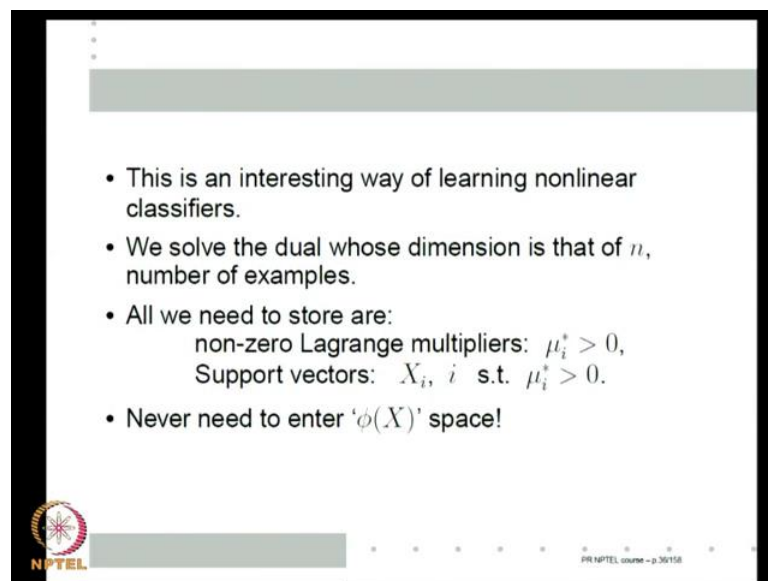
As it turns out the kernel also helps me to find the classification, very new patterns also very efficiently let us say μ_i^* is the solution of the dual. Then we know W^* is $\mu_i^* y_i \phi(X_i)$. Because it is a linear classifier of the Z domain, so is $\mu_i^* y_i z_i$, which is same as $\mu_i^* y_i \phi(X_i)$. Now, first let us see how I can get b^* I know b^* is $y_j - Z_j^T W^*$ for an appropriate j . Now, Z_j is $\phi(X_j)$, now $\phi(X_j)^T W^*$ can be written as $W^*^T \phi(X_j)$ W^* is this.

So, it will be $y_j - \mu_i^* y_i \phi(X_i)^T \phi(X_j)$, now $\phi(X_i)^T \phi(X_j)$ is $K(X_i, X_j)$. So, I do not have to actually calculate any $\phi(X_i)$'s or $\phi(X_j)$'s for calculating

b^* I can do it only in terms of $K(X_i, X_j)$. Now, is that good enough yes what do I have to do given a new pattern X_i effectively need to compute $Z^* = \phi(X_i)^T W^* + b^*$. Given an X_i have to first transform into the Z that is $\phi(X_i)$, then there is use the optimal hyperplane. So, I have to calculate $\phi(X_i)^T W^* + b^*$. Now, W^* is given by this, so this term is $W^* \phi(X_i)$. So, this is $\phi(X_i)^T W^*$, so that is $\sum y_i \phi(X_i)^T \phi(X_i) + b^*$.

Now, $\phi(X_i)^T \phi(X_i)$ is $K(X_i, X_i)$ b^* is this this also can be written as $K(X_i, X_j)$. So, everything can be written in terms of the kernel function. Now, so if I solve the dual and store all my μ_i^* and the corresponding X_i 's then given any new pattern I have to just calculate this to decide its classification. I calculate this and based on this sign I will say class plus 1 or class minus 1 and to calculate this I never need to calculate any $\phi(X)$'s. Because you already agreed that $K(X_i, X_j)$ is about as complicated as $X_i^T X_j$ and more importantly I never need to calculate $\phi(X_i)$ or $\phi(X_j)$ if you give me the kernel function.

(Refer Slide Time: 16:47)



So, this is really an interesting way of learning a non-linear classifier we have solved the dual whose dimension is that of n number of examples irrespective of what is the high dimensional space in which you are finding the classifier. All we need to store after we solve the dual are the non zero Lagrange multipliers and the corresponding training data X_i , which are called the support vectors once I have that I know how to calculate

classification of any new pattern. This is all need to calculate the classification of any new pattern, which means I need to enter the $\phi(X)$ space at all. I never to calculate $\phi(X)$ minus (()) I do not even know where in some sense if you tell me the K the kernel function. I do not even need to know what ϕ is, I do not even need to know what the range space of $\phi(X)$ is right.

(Refer Slide Time: 17:44)

Support Vector Machine

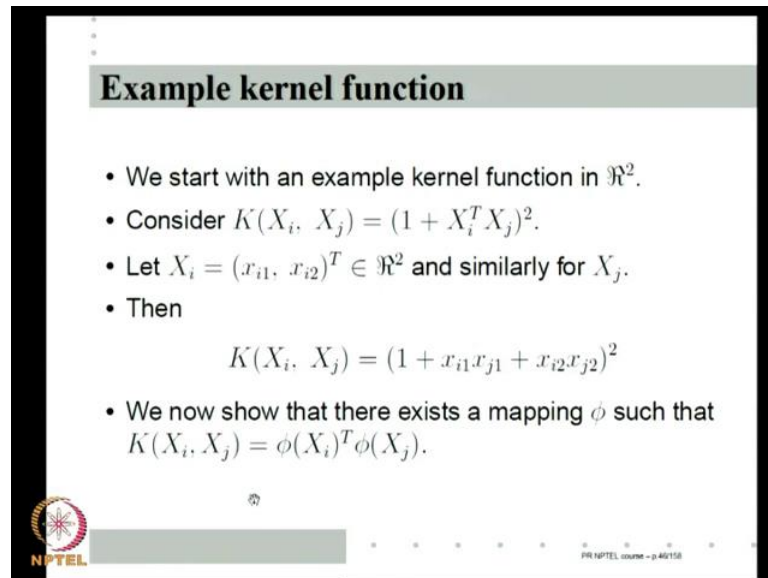
- Obtain μ_i^* by solving the Dual with $Z_i^T Z_j$ replaced by $K(X_i, X_j)$. (Choose a suitable Kernel function. Use 'penalty const', C if needed).
- Store non-zero μ_i^* and the corresponding support vectors.
- Classify any new pattern X by sign of
$$f(X) = \sum \mu_i^* y_i K(X_i, X) + (y_j - \sum_i \mu_i^* y_i K(X_i, X_j))$$
- If we have a suitable Kernel function, we never need to compute $\phi(X)$.
- The range space of ϕ can even be infinite dimensional!

NPTEL © NPTEL course - p.41/150

So, this is what support vector machine is what does support vector machine do I essentially obtain μ_i^* by solving the dual with $Z_i^T Z_j$ replaced by $K(X_i, X_j)$. All it means is I choose a suitable kernel function, now I can use a penalty constant C if needed I can choose a C and as we said using a penalty constant C is simply a matter of putting an upper bound on μ . So, in the dual solution it is a very simple thing to incorporate once I got my μ_i^* I store non 0 μ_i^* and the corresponding support vectors.

Now, I am ready give me any new pattern this is what I compute and classify. So, if we have a suitable kernel function we never need to compute $\phi(X)$ we never need to know what $\phi(X)$ is as a matter of fact in a matter of seeing the range space of ϕ can even be infinite dimensional for all I care. Because I would never need to compute a single $\phi(X)$, as long as I have a kernel function.

(Refer Slide Time: 18:41)



Example kernel function

- We start with an example kernel function in \mathbb{R}^2 .
- Consider $K(X_i, X_j) = (1 + X_i^T X_j)^2$.
- Let $X_i = (x_{i1}, x_{i2})^T \in \mathbb{R}^2$ and similarly for X_j .
- Then

$$K(X_i, X_j) = (1 + x_{i1}x_{j1} + x_{i2}x_{j2})^2$$

- We now show that there exists a mapping ϕ such that $K(X_i, X_j) = \phi(X_i)^T \phi(X_j)$.

NPTEL PRE NPTEL course - p.46/158

So, now let us ask where do kernel functions come from, so let us start with a simple example in two dimensional feature vectors. Let us say we are working in \mathbb{R}^2 , so take a kernel function $K(X_i, X_j) = 1 + X_i^T X_j$ is 1 plus $X_i^T X_j$ whole square. As I point out this is about as expensive as translating $X_i^T X_j$ suppose X_i is here, of course it is two dimensions. But in addition to $X_i^T X_j$, once I calculate $X_i^T X_j$ all I have is one addition and 1 multiplication to get this square.

So, it is just one multiplication more than $X_i^T X_j$ we will use the same kernel function in any dimensions. Later on as we see, so if X_i is a 100 dimensional space $X_i^T X_j$ involves hundred multiplications. Whereas, $K(X_i, X_j)$ involves 101 multiplication, so $K(X_i, X_j)$ is about as expensive as $X_i^T X_j$. So, if I represent the two components of X_i , I am looking at two dimensional vectors here by X_{i1} and X_{i2} .

Similarly, X_j as X_{j1} and X_{j2} then $K(X_i, X_j) = 1 + X_{i1}X_{j1} + X_{i2}X_{j2}$ whole square this is nothing but $X_i^T X_j$ because they are two dimensional (()). This is what my kernel function is we are, now going to show that there exists a map ϕ . So, that $K(X_i, X_j) = \phi(X_i)^T \phi(X_j)$ and we are also going to show that this map is such that a linear classifier in the $\phi(X)$ domain is actually a quadratic classifier in the X domain.

(Refer Slide Time: 20:12)

- Consider $\phi : \mathbb{R}^2 \rightarrow \mathbb{R}^6$ given by

$$Z = \phi(X) = [1 \quad \sqrt{2}x_1 \quad \sqrt{2}x_2 \quad x_1^2 \quad x_2^2 \quad \sqrt{2}x_1x_2]$$
 (Here, $X = (x_1 \ x_2) \in \mathbb{R}^2$).
- It is easy to see that a linear discriminant function in terms of Z (i.e., in \mathbb{R}^6) would be a quadratic discriminant function in terms of X (i.e., in \mathbb{R}^2).
- Now we show that

$$K(X_i, X_j) = (1 + X_i^T X_j)^2 = Z_i^T Z_j = \phi(X_i)^T \phi(X_j)$$

So, as earlier let us choose $\phi: \mathbb{R}^2 \rightarrow \mathbb{R}^6$ given by $1 \ \sqrt{2}x_1 \ \sqrt{2}x_2 \ x_1^2 \ x_2^2 \ \sqrt{2}x_1x_2$. This $\mathbb{R}^2 \rightarrow \mathbb{R}^6$ is very easy to see that a linear discriminant function in terms of Z . So a linear function in terms of Z will have terms constant terms of X_1 terms of X_2 terms of X_1^2 terms of X_2^2 and $X_1 X_2$. So, a linear classifier need a discriminant function in terms of Z would be a quadratic discriminant function in terms of X .

Now, that is the whole idea of a SVM, now we are going to show that with this transformation $K(X_i, X_j)$. That is $1 + X_i^T X_j$ whole square is actually $\phi(X_i)^T \phi(X_j)$, which means if I use this kernel function I am effectively finding a quadratic discriminant function all right using just a linear technique. Because if I wanted a linear thing it would have been $X_i^T X_j$ I want a quadratic thing it is simply $1 + X_i^T X_j$ whole square.

I am still solving the same dual, so let us show this first. So, what will be $\phi(X_i)^T \phi(X_j)$? Now, $\phi(X_i)$ will be or Z_i will be it will be $X_i + 1 \ \sqrt{2}x_{i1} \ \sqrt{2}x_{i2} \ x_{i1}^2 \ x_{i2}^2 \ \sqrt{2}x_{i1}x_{i2}$ similarly, for j . So, when I take $\phi(X_i)^T \phi(X_j)$ I will get one here $\sqrt{2} \sqrt{2}$ will give me $2 \ x_{i1}x_{j1} \ 2 \ x_{i2}x_{j2}$ right. $2 \ x_{i1}x_{j1} \ 2 \ x_{i2}x_{j2}$ similarly I get $x_{i1}^2 \ x_{j1}^2 \ x_{i2}^2 \ x_{j2}^2$.

(Refer Slide Time: 21:52)

• We have

$$\begin{aligned} Z_i^T Z_j &= 1 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} + x_{i1}^2x_{j1}^2 + x_{i2}^2x_{j2}^2 \\ &\quad + 2x_{i1}x_{i2}x_{j1}x_{j2} \\ &= (1 + x_{i1}x_{j1} + x_{i2}x_{j2})^2 \\ &= K(X_i, X_j) \end{aligned}$$

• Easy to see it works for $X \in \mathbb{R}^n$ in general.

• Thus $K(X_i, X_j) = (1 + X_i^T X_j)^2$ results in a quadratic discriminant function. ☺

NPTEL

PRE NPTEL course - p-54158

And once again this root two will give me $X_{i1} X_{i2}$ into $X_{j1} X_{j2}$ with a 2 in front, this is it what $Z_i^T Z_j$ is. Now, I can easily see that this is same as $X_{i1} X_{j1} X_{i2} X_{j2}$ whole square I will get $1 X_{i1}^2 X_{j1}^2 X_{i2}^2 X_{j2}^2$ and then $2 X_{i1} X_{j1} 2 X_{i2} X_{j2} 2 X_{i1} X_{i2} X_{j1} X_{j2}$. So, $K(X_i, X_j) = Z_i^T Z_j = \phi(X_i)^T \phi(X_j)$, so easy to see it works for n in general. The only reason we took 2 as you can see, now is because it becomes cumbersome to write this expression.

If I have $X_1 X_2 \dots X_n$ $X_1^2 X_2^2 \dots X_n^2$ and all pairs then accordingly get all of that. Accordingly I get the square root, so this works in general for \mathbb{R}^n , which means if I use this this results in a learning a quadratic discriminant function. So, in the dual if I wanted to learn a linear discriminant function I have to use $X_i^T X_j$ instead of that I use $1 + X_i^T X_j$ whole square, which is a 1 time computation dual. Then I can use this same function in my final classification I get a quadratic discriminant function.

(Refer Slide Time: 23:32)

- From this example, it is also easy to see that for a given Kernel function, the mapping ϕ (or the dimension of its range space) is not unique.
- Consider the same Kernel fn $K(X_i, X_j) = (1 + X_i^T X_j)^2$.
- Consider the mapping $\phi: \mathbb{R}^2 \rightarrow \mathbb{R}^7$ given by $Z = \phi(X) = [1 \quad \sqrt{2}x_1 \quad \sqrt{2}x_2 \quad x_1^2 \quad x_2^2 \quad x_1x_2 \quad x_1x_2]$
- It is easy to see that this mapping also works.

The other thing that we want to mention is that for a given kernel function neither the mapping phi nor its range space is unique. So, we chosen this particular phi, but you know for the same kernel function many other phi's work. What other phi's work? Very very easy to see what other phi's work, for the same kernel function instead of taking the last component as root 2 into $X_1 X_2$.

Suppose, I taken it to be naught \mathbb{R}^2 to \mathbb{R}^6 , but \mathbb{R}^2 to \mathbb{R}^7 and the last two components of Z are $X_1 X_2 X_1 X_2$ as you can see if I do $Z^T Z$ I get $X_i X_i + X_j X_j + 2 X_i X_j$ that is what earlier a root 2 $X_1 X_2$ term was given here. So, basically very trivially I can exhibit another phi another dimension for the range space and which works with the same kernel function. So, for a given kernel function the mapping phi is not necessarily needed, there can be many mappings.

(Refer Slide Time: 24:51)

Kernel functions

- How do we obtain Kernel functions in general?
- What kind of symmetric functions capture the inner product in an appropriate space?
- We look at two important characterizations for Kernel functions.

NPTEL PRE NPTEL course - p-61118

So, how do you in general obtain kernel functions we know the kernel function has to be symmetric. Because $K(X_i, X_j) = \phi(X_i)^T \phi(X_j)$ the inner product by this transpose is symmetric. So, it has to be symmetric, so what kind of symmetric functions can be, will be represented as inner products in some appropriate space. We currently defined two characterizations for this one of them we may expand later on.

(Refer Slide Time: 25:28)

Mercer Kernels

- **Mercer Theorem:** Given a symmetric function, $K : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$, there exists an inner product space \mathcal{H} and a mapping $\phi : \mathbb{R}^m \rightarrow \mathcal{H}$ so that $K(X_1, X_2) = \phi(X_1)^T \phi(X_2)$ if for all square-integrable functions g ,

$$\int K(X_1, X_2)g(X_1)g(X_2)dX_1 dX_2 \geq 0.$$

NPTEL

But right now let us define two ways of saying when a function will satisfy what we want first is called what is called Mercer's Theorem kernel function that satisfy these are

called Mercer kernels. Mercer Theorem says that given a symmetric function, which maps \mathbb{R}^m class \mathbb{R}^m to \mathbb{R}^3 exists a space H . I will call it an inner product space meaning in this space H there is an inner product defined inner product is like your dot product $X^T y$.

So, there exists a space H technically this is called a Hilbert space meaning is a vector space. A vector space in which there is a norm of vector defined inner product between two vectors defined and with respect to this norm this space is closed meaning every Cauchy sequence is convergent there. If you do not understand what I am saying it does not really matter, that is not at present important. So, basically given any symmetric function there exists a space H in which an inner product is defined.

The mapping $\phi: \mathbb{R}^m \rightarrow H$, so that $K(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle$ for all $x_1, x_2 \in \mathbb{R}^m$ this is what we want. Such a thing is true if the K satisfies the following for all squared integrable functions $g: \mathbb{R}^m \rightarrow \mathbb{R}$ $\int \int K(x_1, x_2) g(x_1) g(x_2) dx_1 dx_2$ this integral should be positive by this integral. This is actually because $x_1, x_2 \in \mathbb{R}^m$ are m dimensional vectors this is say integral over \mathbb{R}^m or if you want it is a double integral over $\mathbb{R}^m \times \mathbb{R}^m$.

So, because there is a $\int \int K(x_1, x_2) dx_1 dx_2$ there is a another integral over \mathbb{R}^m $\int g(x)^2 dx$ is a squared integral function meaning $\int g(x)^2 dx < \infty$ $\int \int K(x_1, x_2) g(x_1) g(x_2) dx_1 dx_2$ is less than infinity. So, given any such $g: \mathbb{R}^m \rightarrow \mathbb{R}$ K should satisfy this if K satisfies this then such a K will be a kernel function. Because there will be an appropriate ϕ satisfying $K(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle$. So, any kernel that satisfies this theorem is called a Mercer kernel you do not prove the theorem, but will I will show you how to use this theorem to show that the previous kernel that we considered is really a kernel function.

(Refer Slide Time: 27:43)

Positive definite kernels

- Let \bar{K} be a $n \times n$ matrix with $\bar{K}_{i,j} = K(X_i, X_j)$. A **positive definite kernel** is the function K such that \bar{K} is positive definite for all n and all data sets $\{X_1, \dots, X_n\}$.
- That is, given any n , and any feature vectors, X_1, \dots, X_n , we have, for all scalars c_1, \dots, c_n ,

$$\sum_{i,j=1}^n c_i c_j K(X_i, X_j) \geq 0$$

- If input space is compact, both these notions are same.

NPTEL PRE NPTEL course - p.05/158

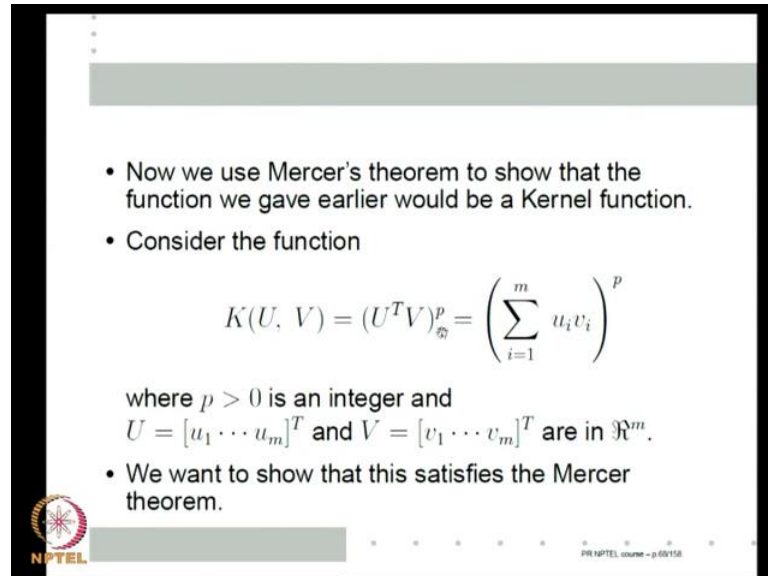
A second characterization of kernels as what are called positive definite kernels see given a kernel function and some data X_1 to X_n . Let us make matrix an n by n matrix \bar{K} whose i, j th element $\bar{K}_{i,j}$ is $K(X_i, X_j)$ this will be a symmetric matrix because this kernel function is symmetric. Then we say the kernel function is positive definite if the matrix \bar{K} is positive semi definite for every n . All data sets what does a matrix being positive semi definite mean the quadratic form of the matrix should be greater than or equal to 0. So, you can actually write it what it means is given any positive integer n and any n feature vectors X_1 to X_n from \mathbb{R}^m .

Then given any n and any X_1 to X_n then for every set of n scalars c_1 to c_n n real numbers summation over i, j $c_i c_j K(X_i, X_j)$ this is the quadratic form of matrix is always greater or equal to 0. So, a function a symmetric function K is said to be a positive definite kernel if for every positive integer n and any feature vector gets X_1 to X_n in \mathbb{R}^m . Given these we have summation over i, j 's $c_i c_j K(X_i, X_j)$ greater than or equal to 0 for every set of n scalars.

Now, as it turns out if the space you are working in compact for example, \mathbb{R}^m is compact any any real Euclidian space under this standard Euclidian norm is compact. So, if the space in which you are working is compact then a mercer kernel and positive definite kernel are the same one and the same these are somewhat subtle concepts, which

are not concerned as in this particular course, but we can just remember this. So, these are the two ways in which I can form kernels.

(Refer Slide Time: 29:50)



• Now we use Mercer's theorem to show that the function we gave earlier would be a Kernel function.

• Consider the function

$$K(U, V) = (U^T V)^p = \left(\sum_{i=1}^m u_i v_i \right)^p$$

where $p > 0$ is an integer and $U = [u_1 \cdots u_m]^T$ and $V = [v_1 \cdots v_m]^T$ are in \mathbb{R}^m .

• We want to show that this satisfies the Mercer theorem.

NPTEL

PR NPTEL course - p.65/158

Now, let us use the mercer's theorem to show that the function we gave earlier is a kernel. And we we use that proof to motivate a few more kernels, but we do not start with that function we start with slightly different function. Let us say I have function K of U comma V, I need two vectors U V is u transpose V to the power p earlier we used 1 plus U transpose V to some integer namely square there.

But in general, let us say we start with U transpose V to the power p U transpose V is nothing but i is equal to 1 to m U i V i whole to the power p. So, a function this symmetric K U V U transpose V to the power of p where p is a positive integer. U and V are vectors in R m with these components consider this function we are first going to show that this is a kernel function. To show that let us first look at this expression.


(Refer Slide Time: 30:41)

• By expanding the $(U^T V)^p$ we get an expression

$$\left(\sum_{i=1}^m u_i v_i \right)^p = \sum_{r_1, \dots, r_m} \frac{p!}{r_1! r_2! \dots r_m!} \prod_{i=1}^m (u_i v_i)^{r_i}$$

where the summation is over all non-negative integers, r_1, \dots, r_m such that

$$r_1 + r_2 + \dots + r_m = p$$

 © NPTEL, course - p 09158

This is $U_1 V_1$ plus $U_2 V_2$ plus $U_m V_m$ to the power p . So, I can use a multinomial expansion if I use multinomial expansion what is the kind of terms I will get, I get a sum of products. Each product is $U_1 V_1$ to the power R_1 into $U_2 V_2$ to the power R_2 into $U_3 V_3$ to the power R_3 so on, where $R_1 R_2$ or such that we we form the that sum should be p . For example, the binomial case I will get 0 on $p-1$ on $p-1$. So, you get $U_1 V_1$ to the power 0 $U_2 V_2$ to the power p $U_1 V_1$ to the power 1 $U_2 V_2$ to the power $p-1$.

$U_1 V_1$ to the power 2 $U_2 V_2$ to the power $p-2$ and so on. But in a multinomial case there will be $U_i V_i$ to the power R_i , where R_i satisfy $R_1 + R_2 + R_m$ is equal to p . The coefficients turn out to be this kind of $\binom{p}{r_1, r_2, \dots, r_m}$ coefficients the summation is over all non negative integers R_1 to R_m that satisfy this. So, any something like you know sum of n terms raised to the power p can be expand as sum of products like this using a multinomial expansion.

(Refer Slide Time: 32:01)

• We need to show

$$\int_{\mathbb{R}^m} \int_{\mathbb{R}^m} \left(\sum_{i=1}^m u_i v_i \right)^p g(U) g(V) dU dV > 0.$$

• This becomes a sum of integrals by expanding $(\sum u_i v_i)^p$. A typical term here is

$$\frac{p!}{r_1! r_2! \dots r_m!} \int \int (u_1 v_1)^{r_1} (u_2 v_2)^{r_2} \dots (u_m v_m)^{r_m} g(U) g(V) dU dV$$

$$= \frac{p!}{r_1! r_2! \dots r_m!} \left(\int u_1^{r_1} u_2^{r_2} \dots u_m^{r_m} g(U) dU \right)^2 \geq 0$$

NPTEL

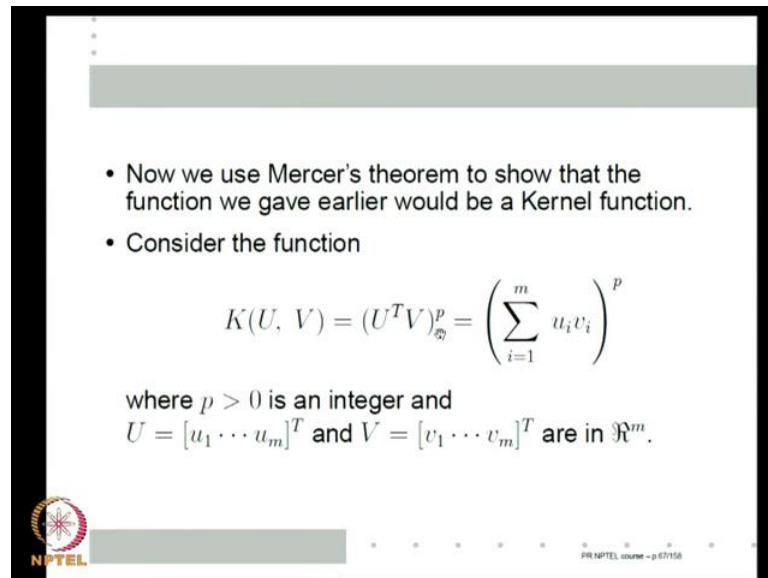
PRE NPTEL course - p 73158

So, keeping that in mind what is that we have to show we have to show that $\int \int (\sum u_i v_i)^p g(U) g(V) dU dV$ is greater than or equal to 0. I am sorry it is not greater than 0 greater than equal to 0, now because this $dU dV$ is actually written it as two integrals just to make it clear to you dU and dV both are in represent integrals for \mathbb{R}^m and this is my UV . Now, if I use the previous expansion this thing to the power p can be written as sum of this, because they are finite sum the inside, integral is a finite sum.

So, I can pull this sum outside, so by expanding it this becomes a sum of integrals. What will each integral be? Each integral will contain one of these terms into $g(U) g(V) dU dV$. Because this is sum of this if the sum is pulled out each integral will be this, which means using that it becomes a sum of integrals, where each one each integral. We have a term that p factor by r_1 factor $r_2 \dots r_m$ factorial $U_1 V_1$ to the power r_1 $U_2 V_2$ to the power r_2 $U_m V_m$ to the power r_m $g(U) g(V) dU dV$.

Now, you see what do I have here U_1 to the power r_1 U_2 to the power r_2 U_m to the power r_m $g(U)$ can come out as the V integral. What is the left is V_1 to the power r_1 V_2 to the power r_2 V_m to the power r_m here the same integral, because U and V are, of course dummy variables of integration. So, this is nothing but U_1 to the power r_1 U_2 to the power r_2 U_m to the power of r_m $g(U) dU$ whole Square. Hence it is always greater than or equal to 0.

(Refer Slide Time: 33:49)



• Now we use Mercer's theorem to show that the function we gave earlier would be a Kernel function.

• Consider the function

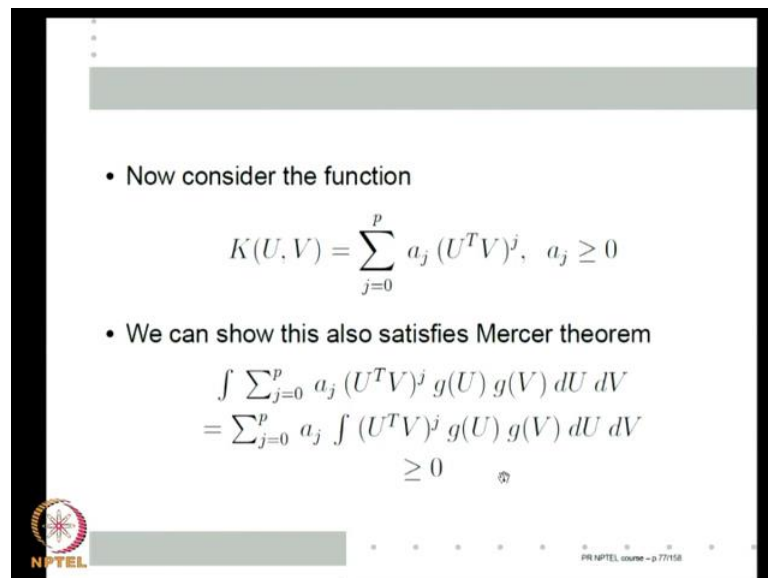
$$K(U, V) = (U^T V)^p = \left(\sum_{i=1}^m u_i v_i \right)^p$$

where $p > 0$ is an integer and $U = [u_1 \cdots u_m]^T$ and $V = [v_1 \cdots v_m]^T$ are in \mathbb{R}^m .

NPTEL logo and footer text are visible at the bottom of the slide.

So, this shows that this function $U U^T V$ to the power p is a kernel is a Mercer kernel for every positive integer V starting from here.

(Refer Slide Time: 34:04)



• Now consider the function

$$K(U, V) = \sum_{j=0}^p a_j (U^T V)^j, \quad a_j \geq 0$$

• We can show this also satisfies Mercer theorem

$$\int \sum_{j=0}^p a_j (U^T V)^j g(U) g(V) dU dV = \sum_{j=0}^p a_j \int (U^T V)^j g(U) g(V) dU dV \geq 0$$

NPTEL logo and footer text are visible at the bottom of the slide.

Now, we can show that, now I suppose I have another function $K U V$ is not just U transpose V to the power someone power, but some a_j times u transpose V to the power j where j 's are integers going from j is 0 to p . So, it is a 0 plus a 1 U transpose V plus a 2 U transpose V square a 3 U transpose V cube and so on. All the way up to a $p U$

transpose V to the power p where all the coefficients are non negative, this also a kernel why this is what I have to show.

This is $K = U^T V$ this summation j is equal to 0 to p a_j is a $U^T V$ to the power j $U^T V$ $U^T V$ this is what I have to show to be positive, now this sum can come out of the integral. Now, this integral I know is positive because I already shown that $U^T V$ to the power j is a kernel.

(Refer Slide Time: 35:01)

• Hence functions of the form

$$K(X_1, X_2) = \sum_{j=0}^p a_j (X_1^T X_2)^j, \quad a_j \geq 0$$

are kernels (satisfying Mercer's theorem).

• A special case is

$$K(X_1, X_2) = (1 + X_1^T X_2)^p$$

which is an example we considered earlier.

• This is called a polynomial kernel.


NPTEL

PR: NPTEL course - p.00158

So, this is because a_j 's are positive it is true, so what this means is that functions of the form $K(X_1, X_2)$, which is given as j is equal to 0 to p $a_j X_1^T X_2$ to the power j are kernels. What is a special case $1 + X_1^T X_2$ power p if I expand this in binomial theorem I will get 1 plus some constant into $X_1^T X_2$ plus some other constant $X_1^T X_2$ square and so on. So, in general these kind of functions are kernels and hence this is a kernel.

So, this is the kernel that we used earlier this is called a polynomial kernel it is called a polynomial kernel because we know as we have seen if you put p is equal to 2 this kernel effectively finds a quadratic discriminant function in the original space. So, in general if I put p it is giving me a discriminant function, which is polynomial of degree p .

(Refer Slide Time: 36:01)



• Now consider the functions of the type

$$K(U, V) = \sum_{j=0}^{\infty} a_j (U^T V)^j, \quad a_j \geq 0$$

• Our proof only involved interchanging integration and summation.

• For finite sum it is always possible.

• For infinite sum, a sufficient condition is that the above sum is uniformly convergent

• Then the above would also satisfy Mercer's theorem.

NPTEL course - p.05/158

Now, let us say instead of looking at a finite sums like this suppose I have infinite sums like this $\sum_j a_j U^T V$ to the power j a j 's are still positive. But no infinite sums where will be the problem earlier when we tried to show that this thing the sum will be inside the integral and I pulled out this sum out of the integral. Now, this can always be true for finite sums, but if it is infinite then the question is can I pull the summation can I interchange the summation and the integral I can do.

So, if the summation is uniformly convergent. So, if I have considered the infinite sums a proof only involved interchanging integration and summation finite sums is always possible for infinite sums a sufficient condition is that the above sum is uniformly convergent. So, if you have series like this which are uniformly convergent then they also form kernel functions. Now, using this we can find another interesting kernel law.

(Refer Slide Time: 36:56)

• Consider the function

$$K(X_1, X_2) = e^{-\frac{(X_1 - X_2)^T (X_1 - X_2)}{2\sigma^2}}$$

• We can show it satisfies the theorem by noting

$$e^{-(X_1 - X_2)^T (X_1 - X_2)} = e^{-X_1^T X_1} e^{-X_2^T X_2} e^{2X_1^T X_2},$$

and

$$e^{2X_1^T X_2} = \sum_{p=0}^{\infty} \frac{(2X_1^T X_2)^p}{p!}$$

NPTEL PRE NPTEL course - p.00158

Let us say I consider a function $e^{-\frac{(X_1 - X_2)^T (X_1 - X_2)}{2\sigma^2}}$. We call it $2\sigma^2$ because it looks like a Gaussian essentially for minus norm of $X_1 - X_2$ whole square by $2\sigma^2$. This is a kernel because $e^{-\frac{(X_1 - X_2)^T (X_1 - X_2)}{2\sigma^2}}$. If I expand this I get an $e^{-X_1^T X_1}$ term $e^{-X_2^T X_2}$ term and I get $e^{2X_1^T X_2}$ term as we have already seen what we have to do in the Mercer Theorem.

We have got $K(X_1, X_2) = e^{-X_1^T X_1} e^{-X_2^T X_2} e^{2X_1^T X_2}$, so if $K(X_1, X_2)$ can be perfectly factorized into a term involving only X_1 a term involving only X_2 . Then I am done because one goes with X_1 other goes with X_2 , ultimately I want to show that to be a square of an integral. So, essentially $e^{-X_1^T X_1} e^{-X_2^T X_2} e^{2X_1^T X_2}$, of course no problem only this cross term we have to know what to do, but the cross term can be written as an infinite sum like that in the Taylor expansion. Now, because any exponential to this is Taylor of an exponential function this is uniformly convergent. Hence, using what we have shown earlier this becomes a kernel, so let us put all this together to look at the kernels.

(Refer Slide Time: 38:21)

Some Popular Kernel functions

- Polynomial kernel:
$$K_p(X_1, X_2) = (1 + X_1^T X_2)^p$$
- Gaussian kernel
$$K_G(X_1, X_2) = e^{-\frac{\|X_1 - X_2\|^2}{\sigma^2}}$$
- Sigmoidal kernel
$$K_S(X_1, X_2) = \tanh(a X_1^T X_2 + \theta)$$

NPTEL PRE NPTEL course - p 9/11/18

So, these are some of the very popular kernels used with SVM's. The first is called the polynomial kernel. The polynomial kernel $K_p(X_1, X_2)$ is defined to be $(1 + X_1^T X_2)^p$. As we have seen this essentially gives you a polynomial of degree up to p . As p goes to infinity, this is called the Gaussian kernel. What you have just considered is $X_1 - X_2$ to the power minus norm $X_1 - X_2$ whole square by some constant normally written as σ^2 or $2\sigma^2$.

Essentially this can give rise to an effective ϕ whose range space is infinite dimensional because as we ultimately saw it involves this $X_1^T X_2$ to the power p kind of terms where p goes all the way up to infinity. That interesting kernel is what is called a Sigmoidal kernel. $K_S(X_1, X_2)$ is $\tanh(a X_1^T X_2 + \theta)$. Of course, instead of tan hyperbolic I can also use sigmoid that is I can write this as $\frac{1}{1 + e^{-a X_1^T X_2 + \theta}}$. I just put tan hyperbolic for because we are considering ultimately function that goes both positive and negative. Anyway these are some of the popular kernels used these two are already shown to be a kernel this I have not, but similarly it can be shown to be a kernel function.

(Refer Slide Time: 39:53)

SVM with Gaussian kernel

- Suppose we use $K_G(X_1, X_2) = e^{-\frac{\|X_1 - X_2\|^2}{\sigma^2}}$
- Classification of a new pattern X is determined by

$$f(X) = \sum_{i \in S} \mu_i^* y_i K(X_i, X) + b^*$$
$$= \sum_{i \in S} \mu_i^* y_i e^{-\frac{\|X_i - X\|^2}{2\sigma^2}} + b^*$$

- Same as a Gaussian RBF (neural) network. Centers of hidden nodes are the support vectors.

NPTEL

PR NPTEL course - p.94158

Let us just take a look at this, what sum of the kernels mean. Suppose we use the Gaussian kernel, if I use the Gaussian kernel what will be my final classification of a new pattern after learning $f(X)$ is $\mu_i^* y_i K(X_i, X) + b^*$ that is what I will calculate right. As we have already seen, now $K(X_i, X)$ is nothing but e to the power minus $\|X_i - X\|^2$ by $2\sigma^2$ this is what I calculate what does this look like you think of $\mu_i^* y_i$ as some constant b^* as some other constant.

Then this looks exactly like what a Gaussian RBF network. So, if I have a standard Gaussian RBF network X is the input then the output of the hidden node output of the i th hidden node will be e to the power minus norm of X , minus this centre of the i th hidden node whole square. Then you multiply the output of the i th hidden node with the weight from the hidden node to the output node and this is the bias of the output node So, this is exactly a RBF network Gaussian RBF network and the centres of the hidden nodes are the support vectors.

As we have seen when we did RBF network we said we actually want some relevant very important vectors to become the centres. So, here the the support vectors become the centres as we have already mentioned the support vectors are a very important by product of the SVM (()) see the support vectors are the one closest to the separating boundary. As we have seen those are the only vectors at which the inequality constraints satisfied by equality constrains which means the solution of the optimization problem is

not going to change if I leave the support vectors in and take away all the other training data. If I remove all the training data give only the support vectors.

Now, I ask you to find the optimal hyperplane you would still find the same one. So, in that sense support vectors are essentially the most important from the point of a classification, because they are closest to this classification boundary. If I know how to separate them I would separate the rest rest of the patterns anyway. So, in some sense support vectors are the most important of your examples as far as the classification problem is concerned.

So, in that sense support vectors are a very important by product we will see some examples next class. But here as you can see if I use a Gaussian kernel an SVM with a Gaussian kernel is equivalent to the Gaussian RBF neural network, where I do not have to worry about how to find the centres and so on. As a matter of fact they have been many people use algorithms whereby you solve an SVM you get this support vectors use them as centres of your RBF, but may not use these (()) multipliers. You relearn the output while its using a linear regression, but any case this expression shows that SVM is the Gaussian kernel is equivalent to a Gaussian RBF neural network.

(Refer Slide Time: 43:01)

SVM with Sigmoidal Kernel

- Classification of X is determined by sign of

$$f(X) = \sum_{i \in S} \mu_i^* y_i K(X_i, X) + b^*$$
$$= \sum_{i \in S} \mu_i^* y_i \tanh(aX^T X_i + \theta) + b^*$$

- Same as the output of a three layer feedforward network with tanh activation function.
- There as many hidden nodes as support vectors and weights into the hidden node are given by the support vectors.

NPTEL logo and footer text: PRE NPTEL course - p 09/158

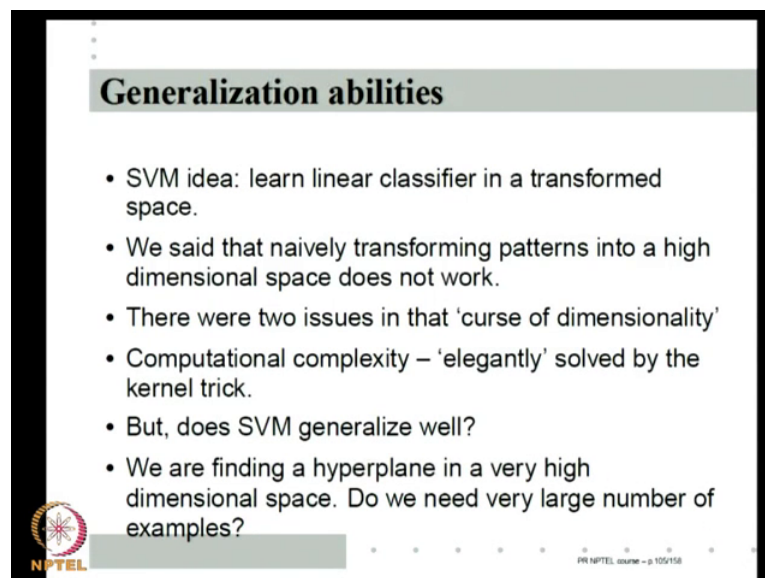
Similarly SVM with a sigmoid kernel, this is equivalent to the other neural network. Classification of X is determined by this, so substitute the function K . So, what is this X_i , I can think of as the vector of weights mapping the input layer to the i th hidden node

in the hidden layer where there is only one hidden node. So, then the net input net input into that hidden the i th hidden node will be a $X^T X_i$ if a X_i is the vector of all the weights that are coming into the i th hidden node θ_i could be the θ_i is the bias of the i th hidden node.

This is the activation function of the i th hidden node, then these are the weights from the i th hidden node to the output node and this is the bias of the output node. So, this is same as the output of three layer feed forward network with one output node and tan hyperbolic activation function for all the hidden nodes whereas linear activation function for the output node. Essentially there are as many hidden nodes as there are support vectors and weights into the hidden node are given by the support vector.

So, once again support vectors see in in our Sigmoidal neural network one thing that we did not know how to choose is the number of hidden nodes. So, once again if I use a support vector machine with a Sigmoidal kernel what I get is essentially like a three layer feed forward network to Sigmoidal activation functions and where these support vectors determine the hidden nodes. So, in that sense this kernel functions are very nice SVM with kernel functions gives rise to non-linear classifiers that we have already seen earlier.

(Refer Slide Time: 44:59)



Generalization abilities

- SVM idea: learn linear classifier in a transformed space.
- We said that naively transforming patterns into a high dimensional space does not work.
- There were two issues in that 'curse of dimensionality'
- Computational complexity – 'elegantly' solved by the kernel trick.
- But, does SVM generalize well?
- We are finding a hyperplane in a very high dimensional space. Do we need very large number of examples?

NPTEL

PR NPTEL course - p 105/158

Now, let us let us go back to the beginning we said that the basic idea of SVM is learn a linear classifier in a transformed space you know. So, the idea is from the original feature space you transform all the feature vectors into high dimensional space and there learn a

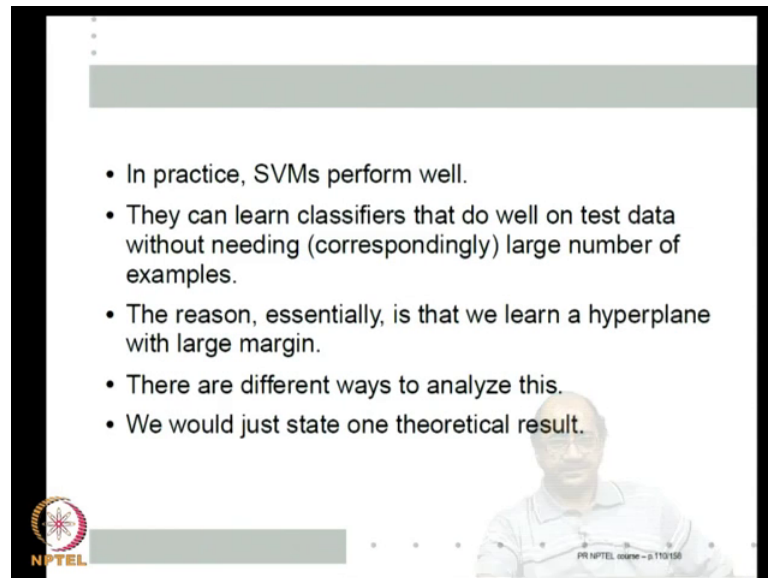
linear classifier. As we also said that the naively doing this will not work right you just transform it and do linear classifier learning there will not work. What are the two problems? We said this is curse of dimensionality there are two issues if I naively transform if I want a quadratic classifier as I said if I have 100 dimensional feature vector to start with it become 10,000 dimensional feature vector.

If I want a cubic classifier 100 dimensional feature vector becomes a million dimensional feature vector, so naively transforming will get us caught in this so called curse of dimensionality. In this curse of dimensionality there are two issues 1 is computational complexity 1 is as I said you know we have to first transform all of them into very high dimensional space do inner products in the high dimensional space to learn the classifier. Even after learning the classifier you give me any new pattern I have to, now once again transform it to high dimensional space and keep doing inner products in the high dimensional space.

So, computationally very complex this is elegantly solved by the kernel trick by using the kernel function and solving the optimization problem in the dual by choosing a optimal hyperplane solving the optimization problem in the domain. Using the kernel we never need to actually compute the transformation we never need to compute the inner products in the high dimensional space. So, the computational complexity issue is elegantly solved by the kernel trick here is the second issue in the curse of dimensionality does it generalize well for all my cleverness.

I am still learning a hyperplane classifier in terms of n dimensional space, so it should have a VC dimension of 10,000. Hence how can I get around? How can I learn with anything less than 10,000 into 10 times 10 into 10,000 examples? So, to say, so the idea is because you are finding a hyperplane in a very high dimensional space do we need correspondingly larger number of examples to get good generalization.

(Refer Slide Time: 47:23)



- In practice, SVMs perform well.
- They can learn classifiers that do well on test data without needing (correspondingly) large number of examples.
- The reason, essentially, is that we learn a hyperplane with large margin.
- There are different ways to analyze this.
- We would just state one theoretical result.

As it turns out in practice SVM's perform very well that is they learn classifiers that do very well on test data without needing correspondingly large number of examples. We need as many examples as you would have needed if you are learning a linear classifier in the original domain. Why does this happen? The reason essentially is that we learn a hyperplane with a large margin we are not searching for any hyperplane classifier, but hyperplane with a large margin different ways to analyze.

For example we can ask V V c dimension of hyperplane classifiers is dimension plus 1, but suppose we restricted ourselves to all classifiers with a minimum margin of δ right. Then whether the V c dimension will change obviously right V c then it will not be same as this these set of classifiers are set of just hyperplane classifiers. So, we can for example, ask what will be V c dimension of large margin classifiers there is a way of looking at it. Currently we are not looking at this what I will do is we will just state one theoretical result in this lecture much later I may come back to this.

(Refer Slide Time: 48:35)

Some theoretical results

- Let P_{err}^n be the error rate on a test set for an SVM trained with n random examples.
- Then we can show (for SVM with no slack variables)

$$EP_{\text{err}}^n \leq \min \left(\frac{s}{n}, \frac{[R^2 \|W\|^2]}{n}, \frac{m}{n} \right)$$

where s is number of support vectors, R is the radius of smallest sphere enclosing all examples, $\|W\|^{-2}$ is the margin of the maximum margin hyperplane (in the feature space of dimension m).

NPTEL

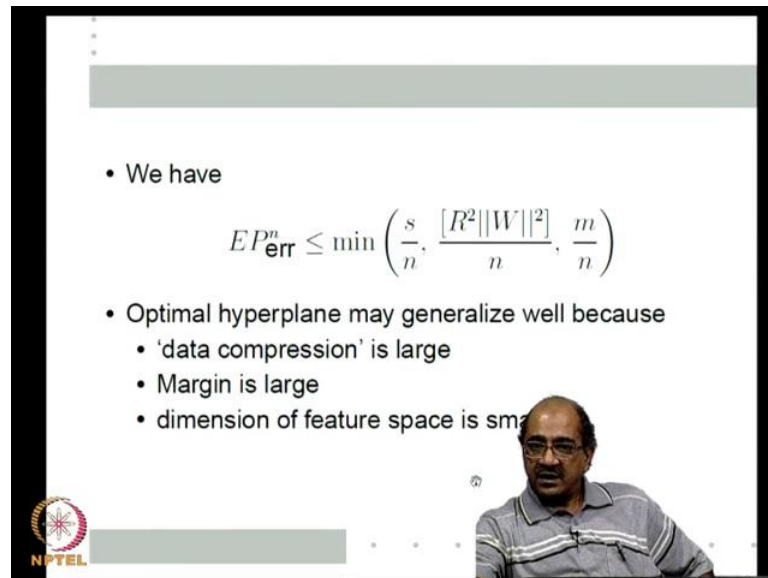
PR NPTEL course - p 112158

So, to say result is the following there is a P_{err}^n error be the error on test set for an SVM trained with n random variable meaning for a given n . I choose n random examples trained and SVM. Now, use this SVM to find error in a test set the probability of error on a test set is given by this. Obviously this is a random variable because it depends on n random examples. If I use different n random examples I may get a different SVM and hence it will have a different test error rate.

But if I ask what is the expected error rate when I learn with n dynamic samples one can bound it above by a minimum of these three numbers. What three numbers s by n , where s is the number of support vectors. So, s by n is the fraction of support vectors out of n examples how many happen to be support vectors or R squared norm W square by n where R is the radius of the smaller sphere enclosing all the examples. 1 by norm W square is the margin of the maximum margin hyperplane.

So, essentially this keeps decreasing as you increasing the margin third is the usual thing your dimension by the number of examples. So, far we only know this from our V c theory. Because for linear classifiers V c dimension is essentially proportional to the dimension of the feature space, so this is dimension of the feature space and n is the total number of examples. This is for SVM with no slack variables you learning a linear classifier in a m dimensional space.

(Refer Slide Time: 50:20)



- We have
$$E P_{\text{err}}^n \leq \min \left(\frac{s}{n}, \frac{[R^2 \|W\|^2]}{n}, \frac{m}{n} \right)$$
- Optimal hyperplane may generalize well because
 - 'data compression' is large
 - Margin is large
 - dimension of feature space is small

Now, let us try and make sense of this this is what we have expected error rate is been more in these three. So, we can essentially say that optimal hyperplane as I said everything is happening because we are learning optimal hyperplane optimal hyperplane may generalize well. Because of many reasons one is that the data compression is large what does that mean s is the number of support vectors n is the total number of examples I have if s by n is small then I know expected error rate is small.

So, if only 1 percent of my data turn out to be support vectors then I have a good chance that I have got a very good classifier. Because expected error rate is bounded above by 0.01. So, essentially we we seen that support vectors are the most important subset of my training examples, if you keep the support vectors and throw away everything else I will still get the same optimal hyperplane. So, in some sense support vectors represent the rest of the examples as far as the classification problem is concerned.

So if I got a very good n site into the classification problem that is what data compression means I can compress all the examples to a few support vectors that may be the key to my good performance. As a matter of fact this is very nice because when you use a support vector machine at the end I have some way of guessing how good would be the classifier I have learnt not using the test set simply by looking at the fraction support vector.


If I got 20 percent of my data support vectors I do not know I may have got a good classifier may not have got a good classifier. Because, of course this is only an upper bound, but if I finally, landed up with only 1 percent of my examples being support vectors. Then I have a very good confidence that I have learnt a good classifier. So, one reason optimal hyperplane may generalize well because we are not learning any hyperplane.

We are learning optimal hyperplane that is when the support vectors come into play is because data compression is large the 2 is, of course the margin is large that also will help. So, if I have large margin that also can result in error rate being small third is the usual thing that we know if I have got many more examples than the dimension which I am learning the linear classifier then also. So, what it means is even if m is very large let us say m is 100 times n . I have got actually less number of examples than I have 1000 examples in a million dimensional space m is 1000 times n , but if one of these two terms is small then also I can learn a good a low error classifier.

This is basically what learning the optimal hyperplane is buying us I want to emphasize again that this is for a learning a linear SVM in m dimensional space without the slack variables. This is when you actually learn an optimal sub training hyperplane once we put slack variables we cannot actually use the theorem. But you know roughly if C is sufficiently large if I think that I am getting a separating hyperplane the theorem would kind of hand waving would be useful, but for a separating hyperplane even if m is large meaning. If I replace m by m' keeping n same m' by n might be much larger than n .

But s by n can still be small or the margin can still be large, so that my expected error would be much smaller even though I do not have sufficient number of examples. This is basically how learning optimal hyperplane is helping. So, both in terms of computational cost and in terms of generalization abilities posing the problem as one of learning optimal hyperplane solving it in the dual and kernelizing the inner product you know gives a very, very elegant solution.

(Refer Slide Time: 54:16)



Kernel Trick

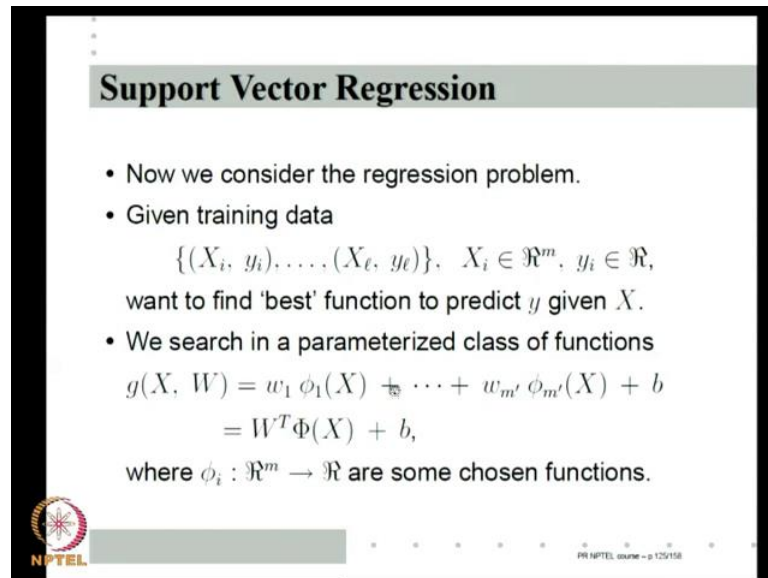
- We use $\phi : \mathcal{R}^n \rightarrow \mathcal{H}$ to map pattern vectors into appropriate high dimensional space.
- Kernel fn allows us to compute innerproducts in \mathcal{H} implicitly without using (or even knowing) ϕ .
- Through kernel functions, we learn nonlinear classifiers using 'linear techniques'.
- We can elegantly construct non-linear versions of linear techniques.
- Algorithms that use only innerproducts can be implicitly executed in a high dimensional \mathcal{H} , e.g., Fisher discriminant, regression etc.

PR NPTEL course - p 122158

So, let us sum up kernel trick we use a function phi, so map the original space into some high dimensional space kernel function allows us to compute inner products in H implicitly without using or even knowing the transformation phi through kernel functions. We learn non-linear classifiers using linear techniques we can elegantly construct non-linear versions of many linear techniques. Essentially between using between learning a linear classifier and non-linear classifier I simply solve the same dual problem where $X^i \text{ transpose } X^j$ is simply replaced by $K X^i \text{ transpose } X^j$.

So, using a kernel I can take a linear technique and convert it into a non-linear technique. We can use this trick in many other algorithms right, we can similarly, design fisher linear discriminant regression wherever the actual training data enter into a solution only as inner products. Only in terms of $X^i \text{ transpose } X^j$ by kernelizing $X^i \text{ transpose } X^j$ we can take a linear function a linear technique and turn it to non-linear technique. For example, I can use kernels and turn fisher linear discriminant into a non-linear fisher discriminant. So, let us look at the same thing in the regression context for the just to see how the kernel trick can be used.

(Refer Slide Time: 55:43)



Support Vector Regression

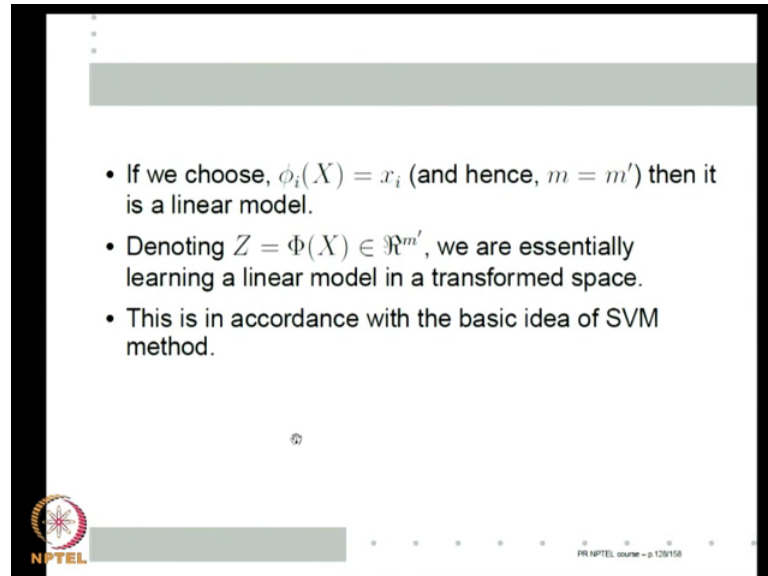
- Now we consider the regression problem.
- Given training data
 $\{(X_i, y_i), \dots, (X_\ell, y_\ell)\}, X_i \in \mathbb{R}^m, y_i \in \mathbb{R}$,
want to find 'best' function to predict y given X .
- We search in a parameterized class of functions
$$g(X, W) = w_1 \phi_1(X) + \dots + w_{m'} \phi_{m'}(X) + b$$
$$= W^T \Phi(X) + b,$$
where $\phi_i : \mathbb{R}^m \rightarrow \mathbb{R}$ are some chosen functions.

NPTEL PR NPTEL course - p 125/158

So, what is the regression problem unlike classification problem I am given data X_i, Y_i . So, $X_1, Y_1, X_2, Y_2, \dots, X_\ell, Y_\ell$ we want to find the best function to predict Y given X . X_i 's are in m dimensional space I put ℓ here sometimes we will use ℓ as the number of examples sometimes use n as the number of examples. So, in general we search in a parameterized class of functions. So we are learning a function $g(X, W)$, which is written as $W_1 \phi_1(X) + W_2 \phi_2(X) + \dots + W_{m'} \phi_{m'}(X) + b$, which I write as $W^T \Phi(X) + b$ where each ϕ_i is a \mathbb{R}^m to \mathbb{R} function.

So, this is what the kind of linear least squares method we have seen where ϕ_i 's are pre chosen functions, but then of course I have to solve this m' dimensional problem. Now, basically there are two ways of looking at it. Of course, if we choose $\phi_i(X)$ equal to X_i then it is a linear model m' is equal to m and $\phi_i(X)$ is equal to X_i , the i th component then it simply your $W^T X + b$ your linear model otherwise as we said it is like using some fixed functions.

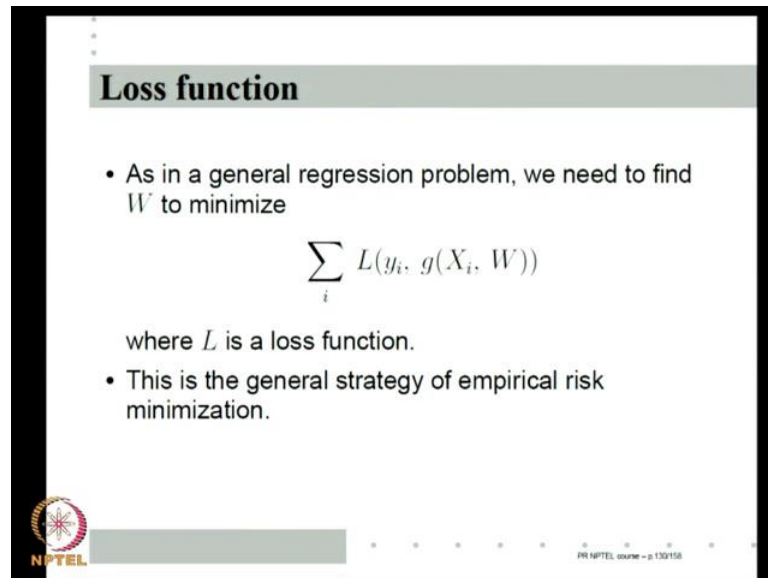
(Refer Slide Time: 57:01)



- If we choose, $\phi_i(X) = x_i$ (and hence, $m = m'$) then it is a linear model.
- Denoting $Z = \Phi(X) \in \mathbb{R}^{m'}$, we are essentially learning a linear model in a transformed space.
- This is in accordance with the basic idea of SVM method.

But what you are going to think of it as actually I think of Z is capital phi of X , so transforming m dimensional X to m prime dimensional Z . Then we are essentially learning a linear model in the transformed space this is the basic idea of SVM. So, the question is can we use the kernels see we could have done it by using linear least squares. But if you want to do it using linear least squares then I have to actually solve the m prime dimensional space m to m dimensional optimization problem. Every time you give me an X i have to actually calculate phi i of X , then do W transpose phi X plus b to do my prediction. So, the idea is that can I do it using kernel trick as it turns out we can do this using kernel trick i.

(Refer Slide Time: 57:48)



Loss function

- As in a general regression problem, we need to find W to minimize

$$\sum_i L(y_i, g(X_i, W))$$

where L is a loss function.

- This is the general strategy of empirical risk minimization.

NPTEL

PR NPTEL course - p 132158

I just give you a a a basic idea of how we are going to do this like in all regression problem we essentially minimize a loss function. This is the strategy of empirical risk minimization we choose some loss function, so given a particular X_i I am saying $g(X_i, W)$ for a. If W is my model I am saying $g(X_i, W)$ whereas, I should have said y_i . So, $|y_i - g(X_i, W)|$ is the loss sum over all I that gives me the empirical loss and that is what I minimize. So, essentially what we are going to show is that we will have a special loss function using which by doing empirical risk minimization. We can use kernel trick while learning linear regression, this is what we will see in the next class.

Thank you.