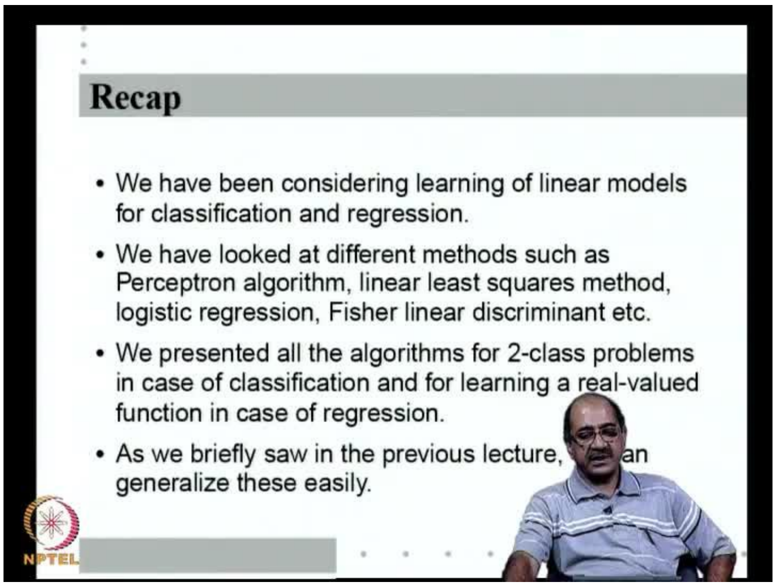
# Pattern Recognition

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

## Lecture - 19
## Linear Discriminant functions for Multi-Class Case; Multi-Class Logistic Regression

Hello and welcome to this next lecture on the course and pattern recognition. So, we looked at linear models for classification regression that is what we have currently been considering.

(Refer Slide Time: 00:42)



### Recap

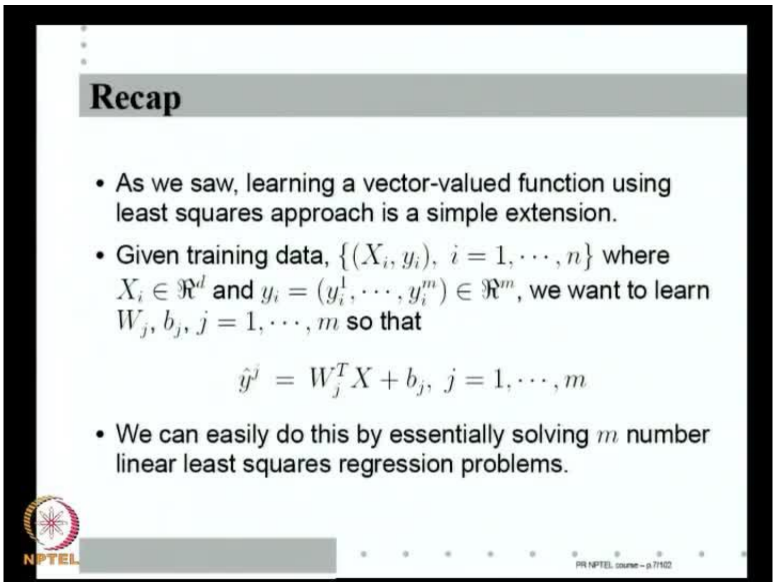
- We have been considering learning of linear models for classification and regression.
- We have looked at different methods such as Perceptron algorithm, linear least squares method, logistic regression, Fisher linear discriminant etc.
- We presented all the algorithms for 2-class problems in case of classification and for learning a real-valued function in case of regression.
- As we briefly saw in the previous lecture, [illegible]an generalize these easily.

We are looking at learning of linear classifiers and linear regression models and we looked at various methods we looked at perceptron, we looked at linear least squares various variations, there of linear least squares with regularization term, then logistic regression, fisher linear discriminant whole set of methods of learning linear classifiers and linear regression functions we have considered, and the...

So, far we have mostly concentrated on the two class problem; that is when we presented algorithms for classification, all the algorithms are only for two class classification problem. Similarly, when we discuss linear regression like least squares, we only considered the case when the target values or real value that means you are only learning a real-valued function over some arbitrary three dimensional space. So, we looked at only real valued functions in case of regression, and two class problems in case of classification.

Towards the end of last class we briefly saw how we can generalize this easily two multiple classes in the classification case, and vector-valued functions in the regression case. So, we will start with that again this class, we will briefly review how the generalization goes? Then we look at the generalization of logistic regression, which is somewhat little more complicated than simply generalizing least square methods for multiclass classification or vector-valued regression.

(Refer Slide Time: 02:11)



## Recap

- As we saw, learning a vector-valued function using least squares approach is a simple extension.
- Given training data, $\{(X_i, y_i),\ i = 1, \cdots, n\}$ where $X_i \in \Re^d$ and $y_i = (y_i^1, \cdots, y_i^m) \in \Re^m$, we want to learn $W_j, b_j, j = 1, \cdots, m$ so that

$$\hat{y}^j = W_j^T X + b_j,\ j = 1, \cdots, m$$

- We can easily do this by essentially solving $m$ number linear least squares regression problems.

PR NPTEL course – p.7/102

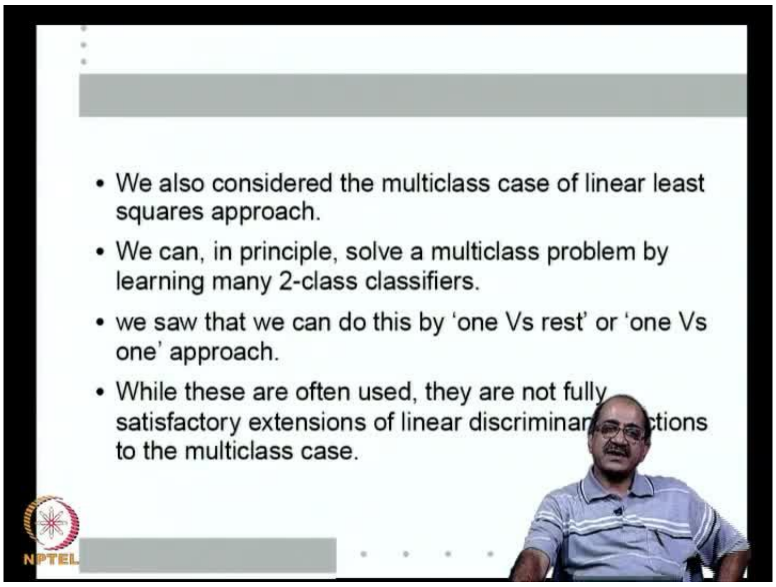
So, to start with, we saw in the last class that learning a vector-valued function using least squares approach is a very straight forward extension. What does learning a vector valued function means? That the the target values themselves are vectors. So, it means that we are given training data X i y i as usual n training data where X i, is in some R d, as usual, but now y i itself is a vector. Let us say is a m vector it is an R m, so they are m values.

So, let us say y i has components y i, 1 y i m and we want to learn linear models for each of them. So, what does that mean? We have each component of y will be predicted by linear model, which means we need to learn m vectors W j and m scalars b j such that the j th output or j th target y j, can be predicted as W j transpose X plus b j. So, in when viewed like this it is simply a a question of solving m number of linear least square regression problems separately.

Because the W j and b j that are involved in predicting the j th component of the output have nothing to do with the W's and b's in predicting any other component of the targets. So, even when the target is vector-valued essentially each component of the target value y has to be predicted using the linear model each of the linear models will have its own w and b. So, it is essentially simply solving m number of least square regression problems.

Of course, we can put the whole thing into a vector matrix notation, but except for notational complexity it does not give us any insight. So, the the substance of this which was seen already last class is that generalising linear regression using least squares approach to vector-valued functions is absolutely straight forward. You just simply learning instead of one W and b learning m number of W's and b's, right? Each will have their own targets.

(Refer Slide Time: 04:26)



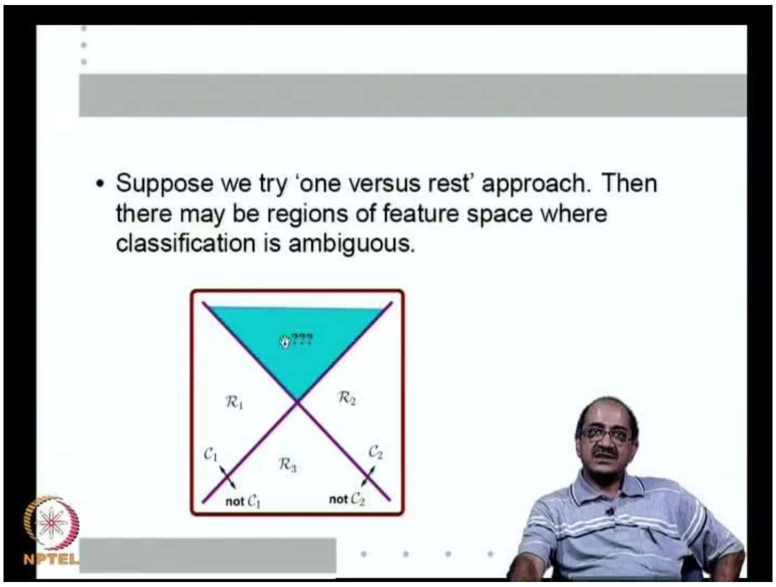
- We also considered the multiclass case of linear least squares approach.
- We can, in principle, solve a multiclass problem by learning many 2-class classifiers.
- we saw that we can do this by 'one Vs rest' or 'one Vs one' approach.
- While these are often used, they are not fully satisfactory extensions of linear discriminant functions to the multiclass case.

In a similar way we can extend linear least square approach to multiclass classification also. What does this mean? To go back to how we started last class for multiclass classification. In the beginning of the course we mentioned that essentially the two class problem is the fundamental problem, if we can solve the two class problem, we can easily solve multiclass problem at least in principle.

What is the in principle mean that we can have many two class classifiers to solve a multiclass problem. as as I mentioned there are essentially two approaches, which may be called one versus rest or one versus one. What is one versus rest means? Suppose, I got classes C 1 C 2 C K, so I learn a two class classifier which classifies C 1 or not C 1. Similarly, C 2 or not C 2, so on so I will have K number of two class classifiers I give the (( )) once I learn them. So, for each of them I know how to make the training samples given the original training samples and once I learn all the classifiers given a new pattern I will ask is it C 1 or not C 1 C 2 or not C 2.

Hopefully, if only one of them says yes then that is the class that can be given. In the one versus one approach I am learning two class problems as C 1 versus C 2 C 1 versus C 3, C 1 versus C 4. So, on the main reason for considering one versus one as opposed to one versus rest is the issues of sufficiency of training samples suppose you have ten classes each with 100 training samples. So, totally you have 1000 samples when you do one versus rest each of those classification problems will have 100 samples of one class and 900 samples of the other class because it is C 1 versus not C 1 and so on.

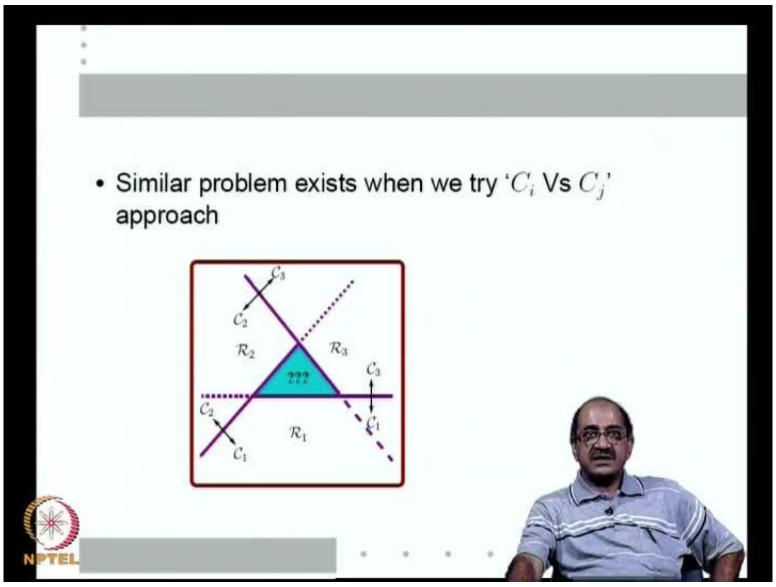
(Refer Slide Time: 06:56)



- Suppose we try 'one versus rest' approach. Then there may be regions of feature space where classification is ambiguous.

So, the sample sets are heavily unbalanced the training data which makes learning a little harder as opposed to that one versus one approach will always have balanced roughly balanced training data set. So, it is a little easier in terms of stability of the classifier learning algorithms these are very often used many standard packages of multiclass classification. Use one of the two approaches, but however in at least in the linear class classification case there are better methods than this. This is because neither of these approaches is wholly satisfactory to tackle a multiclass problem that also we have seen last class.

(Refer Slide Time: 08:05)



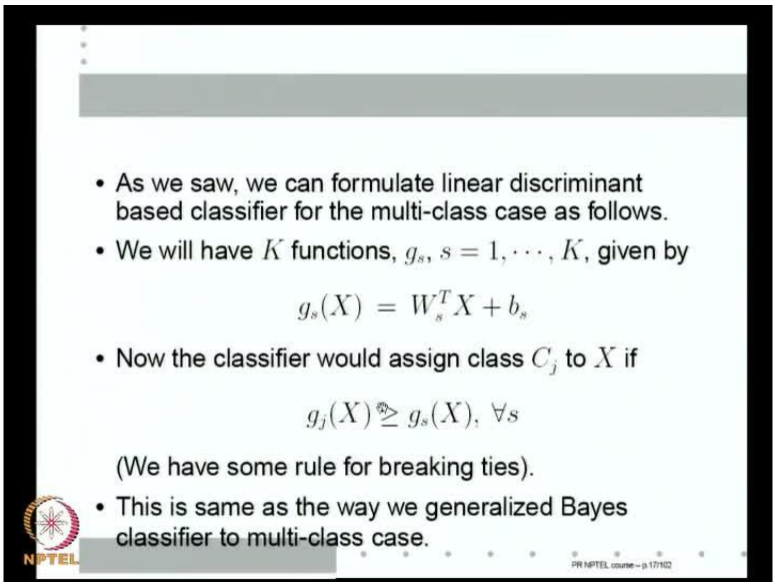
- Similar problem exists when we try '$C_i$ Vs $C_j$' approach

What happens is that, this kind of classifiers always has an undefined regions in the feature space intuitively. If I learn C 1 versus not C 1 and C 2 versus not 2, I give a new X to C 1 versus not C 1 classifier it is a C 1 and I give the same X to C 2 versus not C 2 classifier it says C 2.Then should I take C 1 or should I take C 2, that that is the issue. So, in in particular so if I have this hyper plane I land for C 1 versus not C 1 this hyper plane errant for C 2 versus not C 2. What happens here right, these points because these are on one side of this side of C 2 as well as this side of C 1. I think it is not clear.

So, there will always be regions in the feature space where the classification is not defined. Because ultimately the classifier has to be a function of the feature space this is not fully satisfactory. Same thing happens when we do C i versus C j. I have C classes here C one versus C 2, C 1 versus C 3 and C 1 versus C 2 depending on how this, how what are kind of hyper planes are learned is always possible.

To have region in the feature space, where C 1 versus C 2 classifier may say C 1 C 3 versus C 1 classifier may say C 3 and C 2 versus C 3 classifier may say C 2. So, I cannot even take majority vote, so they would be such undefined regions in the feature space, so in principle both one versus rest as well as one versus one are not completely satisfactory solutions to the multiclass problem.

(Refer Slide Time: 08:52)



- As we saw, we can formulate linear discriminant based classifier for the multi-class case as follows.
- We will have $K$ functions, $g_s$, $s = 1, \cdots, K$, given by

$$g_s(X) = W_s^T X + b_s$$

- Now the classifier would assign class $C_j$ to $X$ if

$$g_j(X) \geq g_s(X), \ \forall s$$

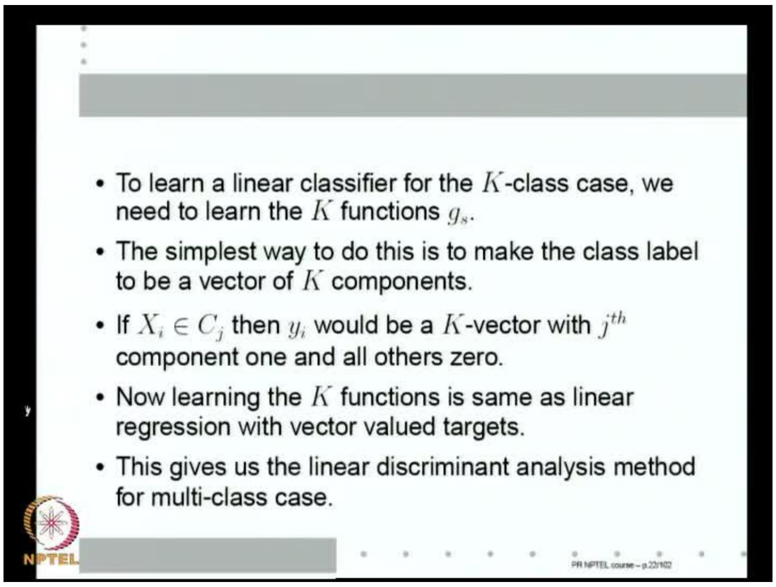
(We have some rule for breaking ties).
- This is same as the way we generalized Bayes classifier to multi-class case.

PR NPTEL course – p.17/102

We have seen last class that one way of generalising linear discriminant function is to have for a K class problem to have K discriminant functions. So, the ideas will have K functions call them g s, each of them is a linear function g s of X is W s transpose X plus b s, s goes from 1 to K. Then how do I use this K functions for classification? So given a new X I will ask which of the g's has the highest value. So, I will assign X to class C j, if g j of x is greater than g s of X for all s of course, is greater than equal to... So, they may be more than 1 j that satisfies this, right? So, may be g 1 X and g 2 X of the same value. They have they are better than all the other values then should I put it in 1 or 2.

Now, this is not as difficult a problem, because we can have some arbitrary way of breaking ties. For example, we will say that if more than 1 j g j satisfies this I put it in the class with least index. Now, while the complete correctness may be I am not at least this is a proper function, so as long as we have a simple rule for breaking ties, having K functions like this. We will completely specify a linear classifier in the multiclass case. You may recall that this is exactly how we generalize the Bayes classifier to multiclass case. It also had k such discriminant functions and you put X into that class whose function has the highest value in X.

(Refer Slide Time: 10:35)



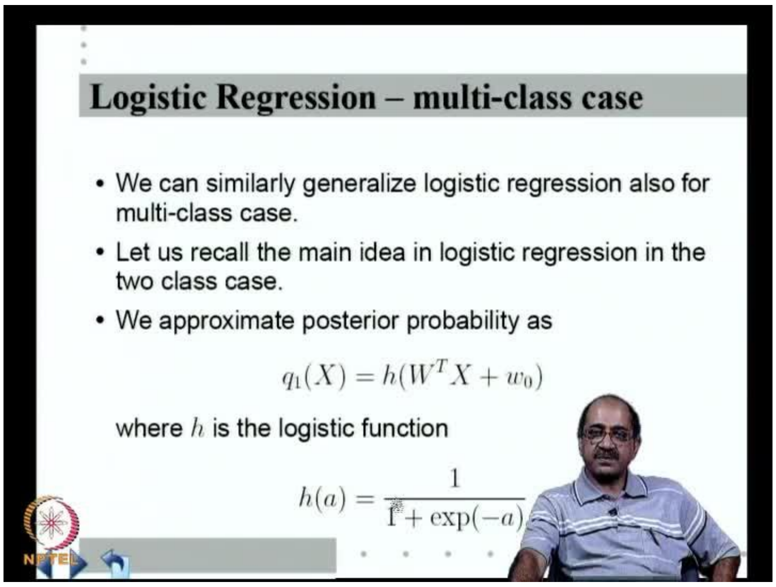
- To learn a linear classifier for the $K$-class case, we need to learn the $K$ functions $g_s$.
- The simplest way to do this is to make the class label to be a vector of $K$ components.
- If $X_i \in C_j$ then $y_i$ would be a $K$-vector with $j^{th}$ component one and all others zero.
- Now learning the $K$ functions is same as linear regression with vector valued targets.
- This gives us the linear discriminant analysis method for multi-class case.

PR NPTEL course – p.22/102

Now, how do I learn this, K functions as we seen briefly in the last class we can learn this K functions simply by doing a vector-valued regression. So, to learn the K class K K K class classifier we need to learn the K functions g s each is a linear function, so the way we do it is to make the class label itself as a vector. Instead of making class label take values 1 to K. We will make the class label take K dimensional vector values, so that class 1 will mean 1 0 0 0. Class 2 will mean 0 1 0 0 0 and so on.

We have already encountered this kind of representation while we were doing estimation of densities. So, a similar kind of representation we will use, so if X i, is in class C j then the corresponding class label y i would be a K vector with the j th component 1 and all others 0. Then the learning the K function is same as a linear regression with vector-valued targets right now using our earlier method of doing regression with vector valued targets. We can simply run least squares to learn the K class classifier. So, this is say this is one method of doing linear discriminant analysis for the multiclass case, so we learn this K function using least squares by essentially making our targets which are class labels into vectors.

(Refer Slide Time: 12:15)



## Logistic Regression – multi-class case

- We can similarly generalize logistic regression also for multi-class case.
- Let us recall the main idea in logistic regression in the two class case.
- We approximate posterior probability as

$$q_1(X) = h(W^T X + w_0)$$

where $h$ is the logistic function

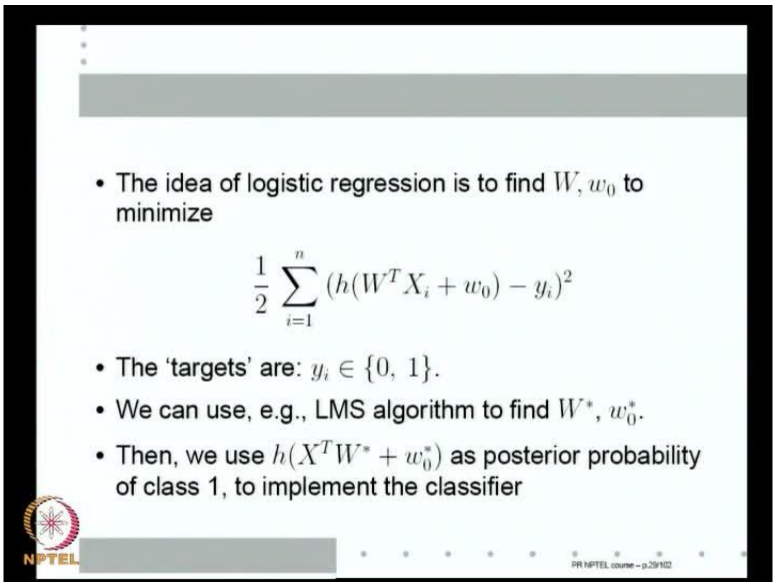
$$h(a) = \frac{1}{1 + \exp(-a)}$$

Now, let us move on the other algorithms that we can say, so this this what we have said so far generalizes linear least square method both have multiclass classification as well as vector-valued function regression. Now, let us look at the remaining ones, we consider logistic regression we considered fisher linear discriminant and so on. So, let us look at logistic regression how to generalize it to multiple cases. So, let us just recall what is the main idea of logistic regression?

Logistic regression essentially uses the linear least squares, but instead of using a simple linear model a W transpose X it uses h of X transpose X. So, the idea is that you approximate the posterior probability recall that we are using q's as the symbols of posterior probability. So, q and X is the posterior probability of class one, so we want to represent q 1 X as h of W transpose X plus W naught for some W and W naught that is what we learn. Ultimately, where h is the logistic function h of a, is 1 by 1 plus exponential minus a.

You see this logistic function of course we can have a small parameter that multiplies, say we can have 1 by 1 plus exponential minus eta a and eta determines the slope at 0, but essentially the logistic function as we see in last class if a is large positive is close to 1 a is large negative is close to 0. So, essentially it is a kind of a continuous approximation to the step function, so basically step function is what we want, and we using the continuous approximation.

(Refer Slide Time: 13:51)



- The idea of logistic regression is to find $W, w_0$ to minimize

$$\frac{1}{2}\sum_{i=1}^{n}(h(W^TX_i + w_0) - y_i)^2$$

- The 'targets' are: $y_i \in \{0,\ 1\}$.
- We can use, e.g., LMS algorithm to find $W^*,\ w_0^*$.
- Then, we use $h(X^TW^* + w_0^*)$ as posterior probability of class 1, to implement the classifier

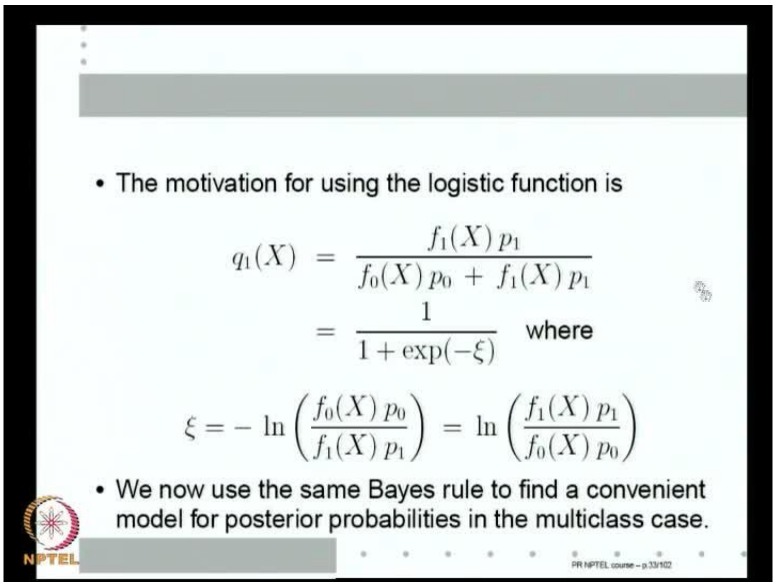
PR NPTEL course – p.29/102

So, in the logistic regression the idea for the two class case the idea is to find a W on W naught to minimize your usual criterion function of least squares where instead of taking the target as a linear function W transpose W naught. We are taking to be h of W transpose W naught, h is monotonous continuous differentiable. Hence, using l m s algorithm we can easily learn this. We take the targets to be 0 and 1 and as we see in a while while studying l m s algorithm the least square approach is easily generalizable to h of W transpose X i plus W naught as the model function as long as h is nice monotonous differentiable.

So, essentially on this function, we do a gradient descent to learn the optimal W 1 W naught having learn W 1 W naught we simply use h of X transpose, W star plus W naught as the posterior probability of class 1 for a given feature vector X. That is how we implement the classifier. So, this is the basic idea of logistic regression, so let us let us try to extract the essence of this idea.

So, what is it that you are using? What is it? What part of it is specific to 2 class. Let us see the way we look at that h of W transpose X f is w naught is not quite clear, how I can make this? How I can easily generalize this into multiple classes? So, how to get a field for, how I can generalize it to multiple classes? Let us ask exactly how am I using the two class restriction or two class fact in this formulation? So, the motivation, so what is that we are doing in the logistic regression; we are approximating the posterior probability by this logistic function. Now, why is it a good idea to why why do we think we can do it?

(Refer Slide Time: 15:53)



- The motivation for using the logistic function is

$$q_1(X) = \frac{f_1(X)\,p_1}{f_0(X)\,p_0 + f_1(X)\,p_1} = \frac{1}{1+\exp(-\xi)} \quad \text{where}$$

$$\xi = -\ln\left(\frac{f_0(X)\,p_0}{f_1(X)\,p_1}\right) = \ln\left(\frac{f_1(X)\,p_1}{f_0(X)\,p_0}\right)$$

- We now use the same Bayes rule to find a convenient model for posterior probabilities in the multiclass case.

PR NPTEL course – p.33/102

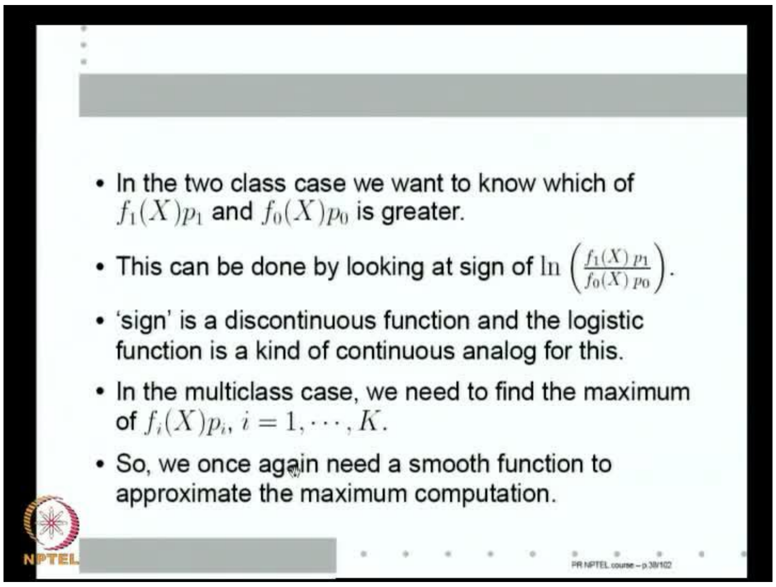
The reason we can think we can do it is that, I can always write the posterior probability in a two class case as f 1 X p 1 by f 0 x p 0 plus f 1 X p 1 this is simply by by Bayes rule q 1 X is probability class is 1 given X. So, I can use Bayes rule to say the density of X given class 1 into probability class 1 by the normalising factor. Now, this I can write like this essentially what I am saying is you you multiply, you divide both numerator and denominator by f 1 p 1, so it becomes 1 by 1 plus f 0 p 0 by f 1 p 1.

Now that I can write like this 1 by 1 plus exponential X i where X i is given by this, so I can always write any posterior probability two class case as 1 by 1 plus exponential minus X i because the only question is can I write X i to be X i is of course. A function of X function of the feature vector X, because X i is given by this is this function feature of vector X will it be linear. In the class of densities where a linear approximation to this function is nice, we can use logistic regression.

So, that is the basic motivation for using logistic regression in the two class case. So, for the multiclass case what we do is we will write the Bayes rule for the posterior probability for any class like this there will be more than two terms (( )) now. Then derive a appropriate function or the logistic function or some other function by the right way to approximate. So, before we go go there, let us while this motivation is clear from

Bayes rule. It may not still be quite clear where the two class fact is used. So, let us understand this.

(Refer Slide Time: 17:46)



- In the two class case we want to know which of $f_1(X)p_1$ and $f_0(X)p_0$ is greater.
- This can be done by looking at sign of $\ln\left(\frac{f_1(X)\,p_1}{f_0(X)\,p_0}\right)$.
- 'sign' is a discontinuous function and the logistic function is a kind of continuous analog for this.
- In the multiclass case, we need to find the maximum of $f_i(X)p_i,\ i = 1, \cdots, K$.
- So, we once again need a smooth function to approximate the maximum computation.

PR NPTEL course – p.38/102

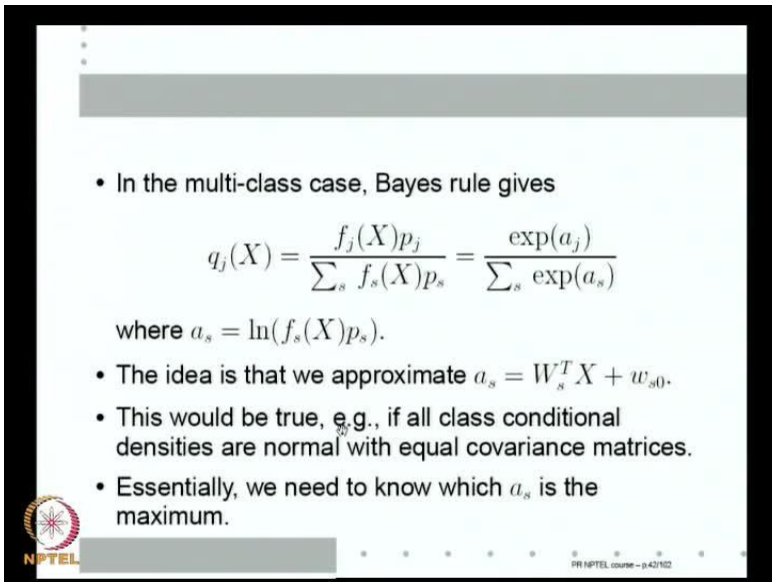
In the two class case essentially what what is that we want to know we want to know which of f 1 p 1 and f 2 p f 0 p 0 is greater. Given an X if you want to actually implement the ideal Bayes classifier, but for 0 1 loss function. The idea is that we calculate f 1 p f 1 p 1 and f 1 X p 0, this is a proportional to the corresponding posterior probabilities. Hence, whichever of these 2 is greater, we can put it in that class of course. In the in the in in the logistic regression, we have somehow approximating some proper function of these as a linear function, because we can only learn linear functions.

So, this is done by looking at... So, if I want to know which of these two is greater, I can take the ratio take the log, so whichever 1 is greater so if the numerator is greater the log will have positive sign if the denominator is greater the log will have negative sign. So, whichever is greater can be ascertained by taking sign of this. But sign of this is a discontinuous function we do not want to work with discontinuous functions. Hence we use the logistic function. Essentially we want sign of l n of this, we we are saying the l n of this can be approximated using a linear function. For example, if f f 1 and f 0 are Gaussian with the equal covariance matrices.

So, if I can approximate l n f 1 X p 0 X f 0 p 1 by f 0 p 0 by a linear function. Now I want sign of this now that that means I have to learn a discontinuous function because I cannot learn would not like to learn discontinuous function. I am essentially squishing it using the logistic function, so logistic function is a kind of continuous analogue for finding this for finding sign of this. So, now in the multiclass case what you have to learn there are now two of them, if there are only two of them, I can take the ratio and log because there are more than two of them.

We want to find with the maximum of some K numbers f i X p i f f 0 X p i say f 1 X p 1 f 2 X p 2 and so on. Calculate all the K numbers and then ask which of these K numbers is large. So, we essentially if I have to find maximum of this K numbers to decide which class to put it essentially. So, which means we need a function maximum also is a non smooth function not differentiable. So, we essentially need some smooth function to approximate the maximum computation that is essentially what we need to generalize the logistic regression to multiclass case.

(Refer Slide Time: 20:39)



- In the multi-class case, Bayes rule gives

$$q_j(X) = \frac{f_j(X)p_j}{\sum_s f_s(X)p_s} = \frac{\exp(a_j)}{\sum_s \exp(a_s)}$$

where $a_s = \ln(f_s(X)p_s)$.

- The idea is that we approximate $a_s = W_s^T X + w_{s0}$.
- This would be true, e.g., if all class conditional densities are normal with equal covariance matrices.
- Essentially, we need to know which $a_s$ is the maximum.

PR NPTEL course – p.42/102

So, let us let us look at a a a suitable function for this, so once again for the multiclass case let us start with the Bayes rule. So, if I want the posterior probability of class j. I can write it as f j X p j by summation f s X p s summation over s as I said because multiple class there is a denominator has more terms as many terms as their classes. We will write this as exponential a j by summation over s exponential a s where a s is given by l n of f s X p s. So, essentially if a s is l n of this exponential l n will cancel each other. So, this is same, so the posterior probability.

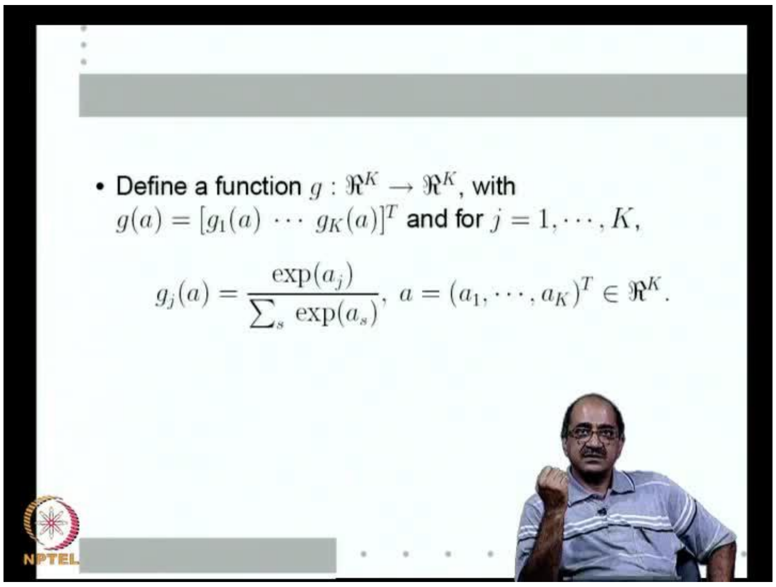
Now, has a form like this l n written as 1 by 1 plus exponential instead of that it has exponential a j by summation over s exponential a s and once again what we want to say is is, a linear approximation for a j's a j's of course are functions of X is a linear approximation for a j's is as far as the classification is concerned. So, the idea is once again we want to approximate each a s by W s transpose X plus W s naught it turns out to be... For example, if all class conditional densities are normal with equal covariance matrices.

What happens is when you take this l n if it is normal. Then I get a quadratic term and a linear term in X right the quadratic term will will all be X transpose sigma X that sigma is not class specific, so that will come out as a common factor in all these things. It will cancel, right? Hence, I can approximate the posterior probability with a function of this form where each a has a form like this.

So, this is one case where this approximation is exact now, many other cases also the approximation may be nice, so once again like in the two class case least square equation the idea is we approximate each of these a s's by f in function W s transpose X plus W s naught and you essentially use a method which is motivated by the fact that the posterior probability can be expressed as a function like this. So, let us go to the details now, so once again to to tie it up with what I said in the previous slide that we want to look at maximum.

If I somehow learnt all the W's right and given an X, I can calculate all the a s's the basic idea is which a s is maximum. Because I need to compare different q j's, because I need to compute different q j's and fit the fit the maximum of them. Because the denominator is same essentially given many such a s's, we need to find the maximum. That is why I said what we need is some smooth approximation to the maximum.

(Refer Slide Time: 23:33)

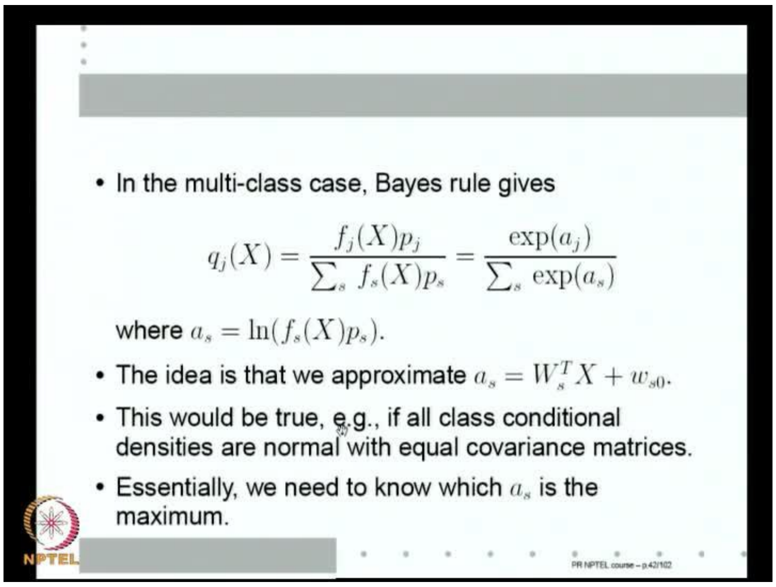


- Define a function $g : \Re^K \rightarrow \Re^K$, with $g(a) = [g_1(a) \cdots g_K(a)]^T$ and for $j = 1, \cdots, K$,

$$g_j(a) = \frac{\exp(a_j)}{\sum_s \exp(a_s)}, \quad a = (a_1, \cdots, a_K)^T \in \Re^K.$$

So, let us define this function, let us say this is a function g that maps R k to R k, K is the... This is the K dimensional real space to itself where K is the number of classes. So, g is a vector-valued function, so let us say components of g are g one g K, so if, is there any point in R K. Then we write g of K as g 1 a g 2 a g K a. They transpose because all our vectors are column vectors and for each j we define g j by this function exponential a j by summation of exponential a s.

(Refer Slide Time: 20:39)



- In the multi-class case, Bayes rule gives

$$q_j(X) = \frac{f_j(X)p_j}{\sum_s f_s(X)p_s} = \frac{\exp(a_j)}{\sum_s \exp(a_s)}$$

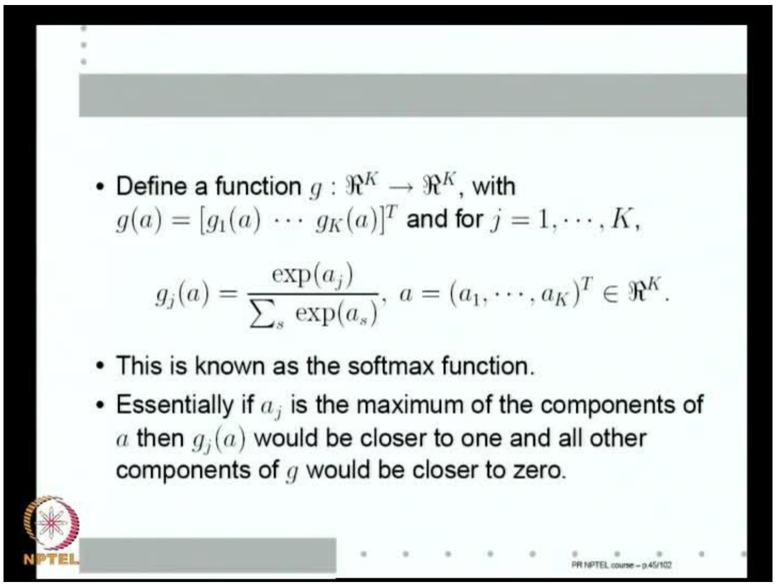
where $a_s = \ln(f_s(X)p_s)$.

- The idea is that we approximate $a_s = W_s^T X + w_{s0}$.
- This would be true, e.g., if all class conditional densities are normal with equal covariance matrices.
- Essentially, we need to know which $a_s$ is the maximum.

PR NPTEL course – p.42/102

So, a j's are components of a a is because g is domain is R K a is an R K, so a has components in a 1 to a K. So, this is the g function we define from what we said in the previous slide this g function is a is a good approximation for the posterior probability in multiclass case. The posterior probability has a similar structure.

(Refer Slide Time: 24:38)



- Define a function $g : \Re^K \to \Re^K$, with $g(a) = [g_1(a) \cdots g_K(a)]^T$ and for $j = 1, \cdots, K$,

$$g_j(a) = \frac{\exp(a_j)}{\sum_s \exp(a_s)}, \quad a = (a_1, \cdots, a_K)^T \in \Re^K.$$

- This is known as the softmax function.
- Essentially if $a_j$ is the maximum of the components of $a$ then $g_j(a)$ would be closer to one and all other components of $g$ would be closer to zero.

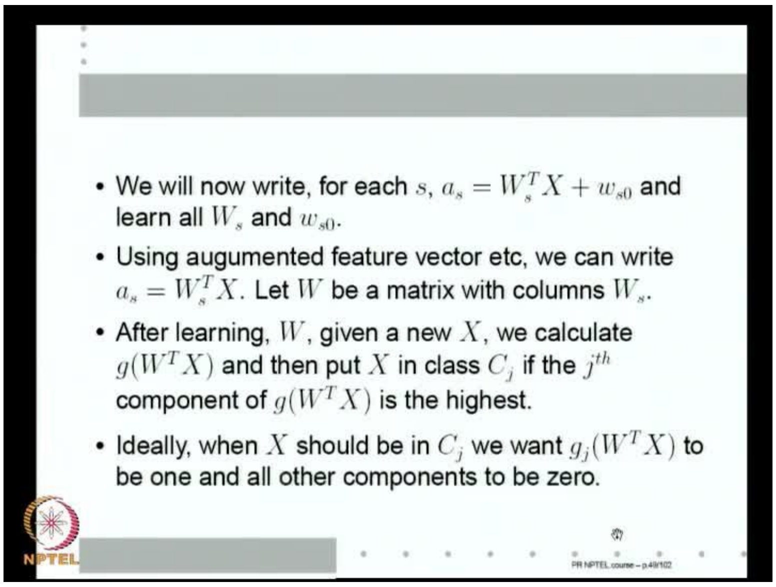
PR NPTEL course – p.45/102

So, this function this g function I am defining is a good approximation of the posterior probability function this g function is known as the softmax function. As I said, our idea is we need a smooth function, which essentially picks maximum of a given certain numbers idea is that if I, suppose a 1 is the maximum of all the a's. Then if I if I divide all the numerators and denominators in g j by exponentially 1.

So, for g 1 it will be 1 by 1 plus some terms all of them will be less than 0, because the exponential small numbers, so exponential negative numbers. Because it will be exponential say a 2 by a 1 there is a exponential a 2 minus a 1 a 1 is greater than a 2, so its exponential negative. So, we can assume that there will be closed to 0 so g 1 of such a a would be 1. All other g's will be closer to 0. So, the idea is that if a j is the maximum of the components of a then g g of a would be closer to 1. Then all the other j's and all the other components a j will be closer to 0 rather than 1, right?

Of course, we can put a constant multiplied to all the a j's to accentuate this differences. But essentially this function roughly picks maximum of a 1 a 2 a K. Because if a j is the maximum of them the corresponding g j of a will be closer to 1 or g s of a for all other s will be closer to 0. So, this is a kind of a nice smooth approximation to the maximum function. That is why it is called the softmax function. So, the idea is now to use the softmax function for learning the classifier.

(Refer Slide Time: 26:44)



- We will now write, for each $s$, $a_s = W_s^T X + w_{s0}$ and learn all $W_s$ and $w_{s0}$.
- Using augumented feature vector etc, we can write $a_s = W_s^T X$. Let $W$ be a matrix with columns $W_s$.
- After learning, $W$, given a new $X$, we calculate $g(W^T X)$ and then put $X$ in class $C_j$ if the $j^{th}$ component of $g(W^T X)$ is the highest.
- Ideally, when $X$ should be in $C_j$ we want $g_j(W^T X)$ to be one and all other components to be zero.

PR NPTEL course – p.49/102

So, now we will write for each s a s as W s transpose X plus W s naught and we learn all the W s’s and W s naughts. See earlier I have W transpose X plus W naught, but now I have more than one such factor, so each vector has a subscript. So, W s is a vector and now my scalar the Bayes term has to 2 surfaces. Because earlier we are calling it W naught, So I called it now W s naught as we already seen enough number of times all such f n functions can be once again written simply as W transpose X. We use augmented feature vector. So, essentially we will call it W tilde and X tilde and write W X tilde transpose X tilde.

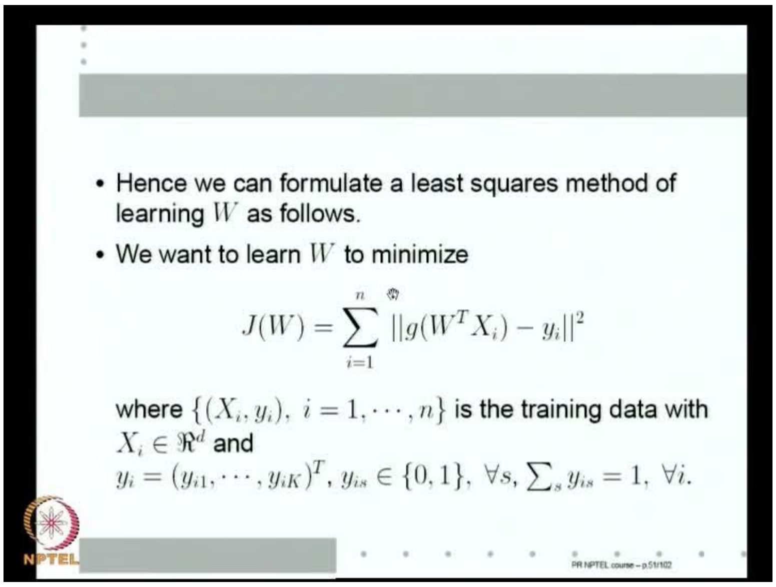
But as we have been doing so far in this course, we we mentally assume that we are done augmentation and use the same symbols. So, might as well write a s h W s transpose X. So, each W s is a vector. Now augmented with an extra complement of W s 0 now, let us capital W be a matrix whose columns are W s, right? Now, let us say we learned this W. We will we will we in a couple of minutes we will see how to learn this W, but suppose we learned the w, so what is its use for us now if we give me nu X I calculate g of W transpose X what is W transpose X W is a matrix.

Now, whose columns are W s? So, W transpose X will be a vector whose first component will be w one transpose X second component will be W 2 transpose X so on. So, we are essentially we we are getting the a s's, so this will be the, that we seen earlier. So, W transpose X will be the whose components are a s's. So, g of a will be what will be a k vector again. and it essentially picks the maximum of all the W transpose W s transpose X's. So, we calculate j of W transpose X. Then put X in the class c j if the j th component of j W transpose X is the highest, right? Ideally if X is in c j.

We want g j of W transpose X to be 1 and all other components to be 0. That is how we want g 2 be, but in in practice we will put X in c j if g j of W transpose X is greater than g s of W transpose X for all other s, but the fact that this g is softmax. Hence, this what we want from our W tells us how we can formulate our objective function. Essentially the W that is good for us is that if if X is a class 1 pattern then g 1 of W transpose X should be 1 and all other should be 0 if X is a class j pattern. Then g j of W transpose X should be 1 and all other should be 0.

So, this is simple for us now I can think of it as a vector-valued regression. So, give me a training pattern X i and let us say its corresponding class label is y i. Now, I represent y I as a vector as usual 1 in its class, so if I give that as the target essentially I vector-valued function where if X is in class j. Then the j th component of this vector-valued function should be 1 all other should be 0 right that is my time data.

(Refer Slide Time: 30:15)



- Hence we can formulate a least squares method of learning $W$ as follows.
- We want to learn $W$ to minimize

$$J(W) = \sum_{i=1}^{n} ||g(W^T X_i) - y_i||^2$$

where $\{(X_i, y_i),\ i = 1, \cdots, n\}$ is the training data with $X_i \in \Re^d$ and $y_i = (y_{i1}, \cdots, y_{iK})^T$, $y_{is} \in \{0, 1\},\ \forall s,\ \sum_s y_{is} = 1,\ \forall i.$

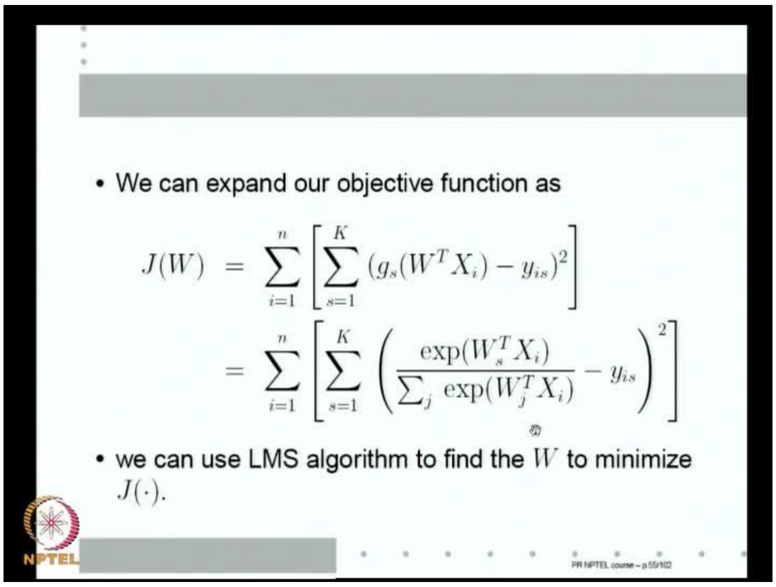
PR NPTEL course – p.51/102

So, this is now how we can formulate a least squares approach for learning W, we will learn w to minimize g of W transpose X i minus y i whole square, right? where X i and y i are our training data. So, X i is an R power d as earlier and y i is now a K vector whose components are y i 1 y i k each of the components is in 0 or 1. They sum to 1, right? So, that means only one component is 1 all others are 0. So, the comprehend that is 1 tells me what is the class label of X. Now g is also a vector and essentially if X i is the j-th class sample then in this y i the j th component will be 1 all others will be 0.

So, I am essentially want I essentially want g of W transpose X i to be close to the vector 0 0 0 1 in the j th position 0 0 0. The that is why this is a good thing to minimize I did not put half we can put half we do not have to put half I have been putting half, so that when the differentiate the 2 will go away, but really that makes no difference. Another thing I wanted to notice is earlier we just simply writing you know g of W transpose X minus y whole square. Now we cannot do that because they are vector-valued functions, so g of W transpose X i is a vector y i is a vector that is why I put a norm on X z. So, is the norm square so this is what we want to minimize (( )).

So, let us look at more closely how we are going to minimize this function. So, this is our function right we want to minimize over the training data. So, given an X i my my model will say j of W transpose X i as its vector-valued class label its true vector-valued class label should be y i. So, I am take the difference between these 2 because they are vectors the difference is the norms norm.

(Refer Slide Time: 32:29)



- We can expand our objective function as

$$J(W) = \sum_{i=1}^{n} \left[ \sum_{s=1}^{K} (g_s(W^T X_i) - y_{is})^2 \right]$$
$$= \sum_{i=1}^{n} \left[ \sum_{s=1}^{K} \left( \frac{\exp(W_s^T X_i)}{\sum_j \exp(W_j^T X_i)} - y_{is} \right)^2 \right]$$

- we can use LMS algorithm to find the $W$ to minimize $J(\cdot)$.

NPTEL
PR NPTEL course – p.55/102

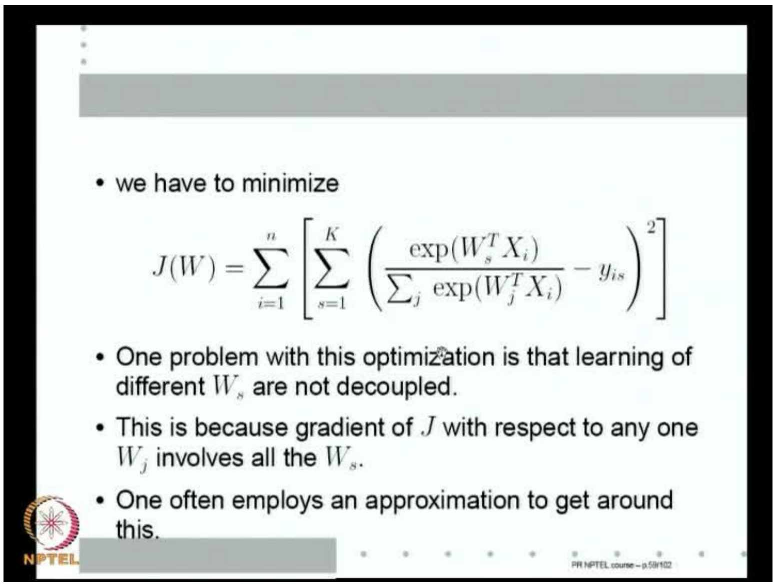
So, because I want to minimize the squares of the errors I am taking the squares of the norms. Now, we will take the usual Euclidean norm by by expanding this norm we can simplify our j of W as follows. Outer summation still remains i is equal to 1 to n the inner 1 this norm or recall that both g of W transpose X i and y i are K vectors i. So, I can write the inner...

I can write the norm as in inner summation s equal to 1 to K g s of W transpose X i minus y i s whole square. Now, I know what g s of W transpose X i is, so let us expand that so g s of W transpose X i is exponential W transpose X i plus summation over J exponential W J transpose X i minus y i whole square. So, J of course, I wrote J of W J is a function of all the W s s. That is why I wrote it as J of W, so this is the explicit function now we want to minimize this, it does not quite look like the earlier logistic.

Very nice form some h of W transpose X plus W naught minus y whole square, but it still I have I have to learn K vectors. This is the relationship I can certainly differentiate J of W with respect to each of these W s s and then find the W to minimize this. So, for example, we can use l m s algorithm to find W that minimizes J l m s algorithm as you know is nothing, but a variant descent on this J. So, I need to find derivatives with respect to the component of W. For example, I can find gradient with respect to each of the W s s.

Then move with the direction of the decreasing gradients. So, in principle this is the multiclass logistic regression algorithm the just 1 of course, this can be done now. There is nothing left except that the expression is little complicated. The the gradient descent optimization might be a little complicated, but this completely solves your logistic relation problem for multiple classes.

(Refer Slide Time: 34:43)



- we have to minimize

$$J(W) = \sum_{i=1}^{n} \left[ \sum_{s=1}^{K} \left( \frac{\exp(W_s^T X_i)}{\sum_j \exp(W_j^T X_i)} - y_{is} \right)^2 \right]$$

- One problem with this optimization is that learning of different $W_s$ are not decoupled.
- This is because gradient of $J$ with respect to any one $W_j$ involves all the $W_s$.
- One often employs an approximation to get around this.

PR NPTEL course – p.59/102

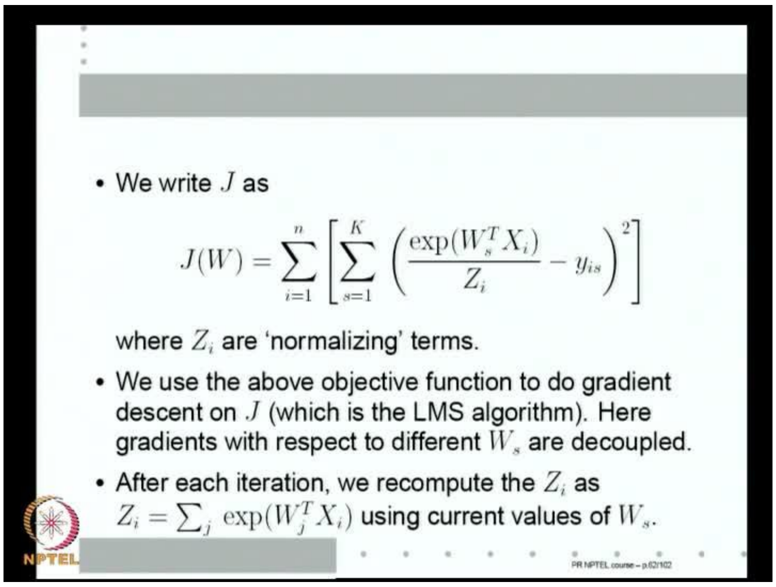
But having said that this is often computationally a little complicated, why is this a little complicated? This is what we want to minimize. The the main problem with this optimization is that learning of different W s s are not decoupled. See earlier even though say in the least squares case of learning vector-valued functions even though I am learning many vectors W s, each has its own targets and the learning different W s s are completely decoupled. So, it is just a question of running say K or m number of different simple linear least square regressions, but now it is not that as this function has all the W s s mixed inside.

So, specifically what does that mean? If I want to take variant of J with respect to a particular W J it involves all the other W s s, because of this normalising term because of this term if I want gain with respect to W, say W 1. Then the s is equal to 1 1 term is what comes here, but this denominator will be there not only that all the other terms in this summation will have W 1 in the denominator. So, you have to take derivative with respect to that, so it will be horrendously complicated expression for the derivative.

So, unlike the the case of vector value regression using simple linear least squares learning of the different W s s. In the case of multiclass logistic regression are not decoupled this can make it computationally more difficult, because it can become computationally difficult one. Sometimes does some approximations of course, when you do approximations you are not you are no longer exactly minimizing the function that you should be minimizing.

But often these approximations are good enough the approximation is based on the observation that basically this problem the problem that gradient of J with respect to any 1 W J involves all the other W's or other W vectors comes because of only this denominator term. In the original posterior probability model the denominator term is essentially a normalising constant. So, the idea is that instead of using all the other w's here can I use a a simple normalising constant here, so that is the basic idea of the approximation.

(Refer Slide Time: 37:11)



- We write $J$ as

$$J(W) = \sum_{i=1}^{n} \left[ \sum_{s=1}^{K} \left( \frac{\exp(W_s^T X_i)}{Z_i} - y_{is} \right)^2 \right]$$

where $Z_i$ are 'normalizing' terms.

- We use the above objective function to do gradient descent on $J$ (which is the LMS algorithm). Here gradients with respect to different $W_s$ are decoupled.
- After each iteration, we recompute the $Z_i$ as $Z_i = \sum_j \exp(W_j^T X_i)$ using current values of $W_s$.

PR NPTEL course – p.62/102

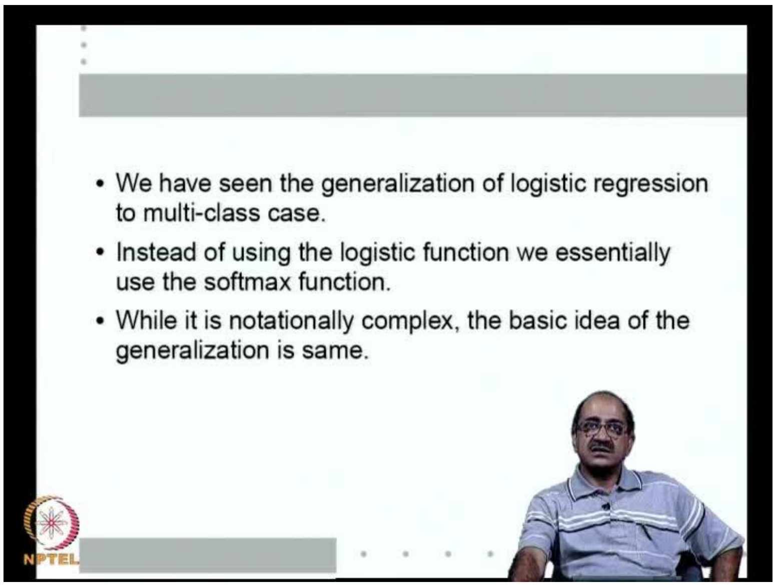
So, how do I run the approximation? So, I rewrite g f as this. So, this of course depends on all the other W s i instead of explicitly saying it depends on all the other W s i simply say is some term that depends on the index i. Because this corresponds to the i th data sample I call it some Z i, thus how I wrote it now? I I pretend that Z i's are constants. They are dependent on i, but otherwise there are not dependent W's, then if I pretend like that. Then if I take the gradient of this with respect to W 1 only that s is equal to 1 term in this in this summation continues.

Similarly, if I want gradient with respect to W 2 only X is equal to 2 term in this inner summation (( )). So, once again all my W's will get decoupled to use the above objective function as I said to do gradient descent on J, what does that mean? If I use l m s algorithm on this then gradient with respect to different W s s are completely decoupled. So, you can now run this as if I am doing different m number of different individual function learning problems. But there is not quite true because this Z, I have to be proper normalising constants.

So, what 1 does is the following you you have currently some values of W's use that to calculate the current Z i. Once you calculate Z i you pretend that Z i does not depend on W anymore. Then calculate gradients now gradients become very simple terms and use the each of the gradients of W s s for a particular s use the gradient of J with respect to W s to find that new value of W s new gradient descent individual. Each of the W s and after each iteration you recompute Z i using the new current values of W s. This way is not completely decoupled because iterations have to run in in step. So, after each after doing one gradient descent on for each of the W's I use all their values to recompute the Z i's.

But still the gradient computation is much much simpler compared to earlier one, where each of the terms in this inner summation contribution gradient. Now only 1 time contributes the gradient. So, computation of gradient becomes simple and recomputing Z i after every iteration is a small extra computation cost way. So, often this approximation gives you lot of computational advantages. So, when one uses logistic regression with multiple classes. We either take this and do the brute force gradient descent on this using the full gradient or we use this approximation and keep the computing Z i is often this is computationally simpler.

(Refer Slide Time: 40:20)

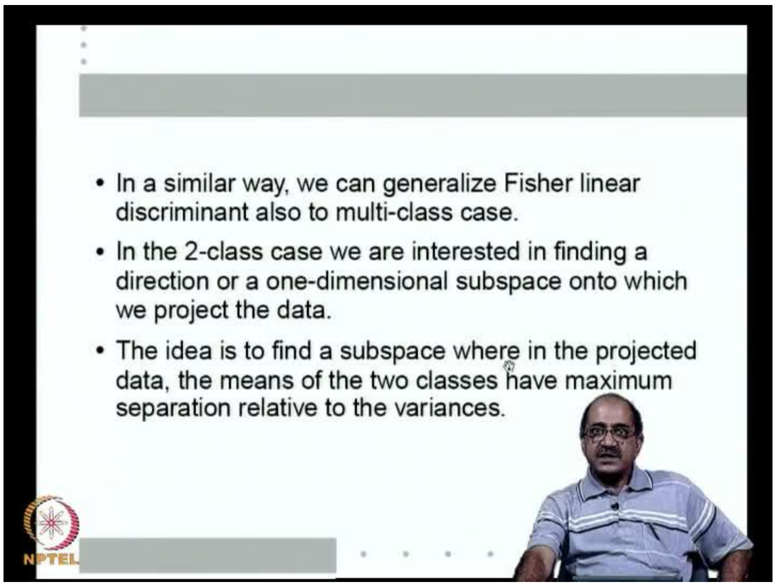


- We have seen the generalization of logistic regression to multi-class case.
- Instead of using the logistic function we essentially use the softmax function.
- While it is notationally complex, the basic idea of the generalization is same.

So, we can sum up. So, we have seen the generalization of logistic regression multiple classes it is a little more complicated than generalizing least square regression (( )). Classification is simple least squares regression classification, but the basic idea as we have is instead of using logistic function in the posterior probability modelling, we are using a softmax function, which is also quite straight forward essentially a logistic function is used as the as a continuous approximation to the function of sign of l n posterior probability of a of class 1 by posterior probability of class 0.

In the multiclass case I need to find the maximum of all the posterior probabilities, so I have to use some continuous some smooth analog of the max function that is the softmax function. So, once we see this connection essentially is notationally complex, but the basic idea of generalization is the same because of this complexity of course, is not completely decoupled. But as we seen using that approximation we can partially decouple it, so this gives us the multiclass logistic regression.

(Refer Slide Time: 41:36)

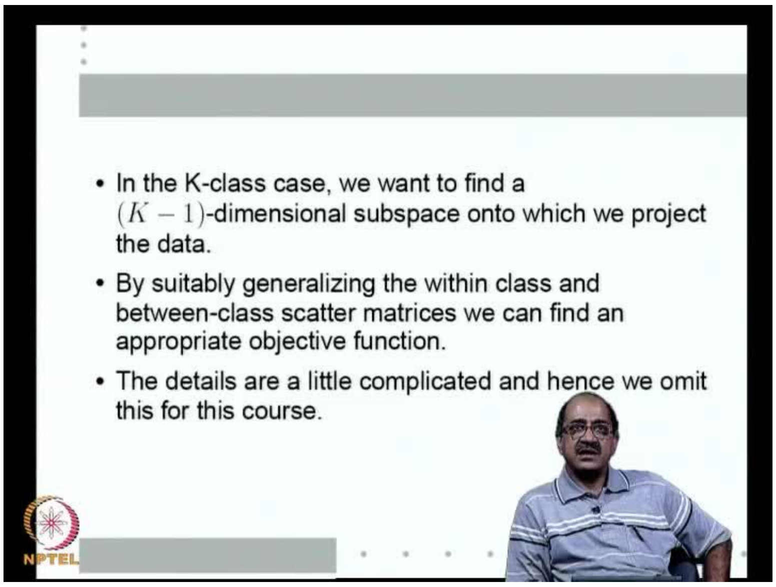


- In a similar way, we can generalize Fisher linear discriminant also to multi-class case.
- In the 2-class case we are interested in finding a direction or a one-dimensional subspace onto which we project the data.
- The idea is to find a subspace where in the projected data, the means of the two classes have maximum separation relative to the variances.

Now, what about fisher linear discriminant that also can be generalized to multiple classes, so once again, let us ask what is the basic idea? In the 2 class fisher linear discriminant the 2 class case what we are doing we are interested in finding a direction a particular vector W a single direction can be thought of as a one dimensional subspace. So, saying it a two class case, we want to find a one dimensional subspace onto which we want to project the data which is the one dimensional subspace.

We want to take the idea is to take that subspace where after projecting the data the means of the 2 classes have maximum separation relative to their variances that is the basic idea. So, you want to find at particular one dimensional subspace this is the 2 class problem. I want to find a particular one dimensional subspace where there is a proper separation. This proper separation is formula is as you as you may recall using that within class covariance within class scatter matrix between class scatter matrix.

(Refer Slide Time: 42:38)



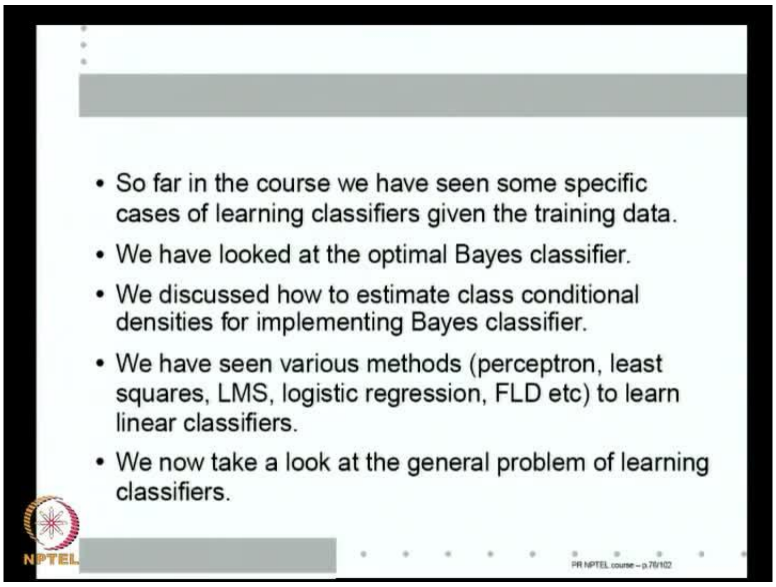
- In the K-class case, we want to find a $(K-1)$-dimensional subspace onto which we project the data.
- By suitably generalizing the within class and between-class scatter matrices we can find an appropriate objective function.
- The details are a little complicated and hence we omit this for this course.

So, in principle generalizing fisher linear discriminant to multiple classes would mean what for a K class case we want to find a K minuses in the 2 class case we are finding a 1 dimensional subspace. So, in the K class case we want to find a K minus one dimensional subspace onto which you want to project the data. So, that is fair enough so its once again essentially I am thinking the fisher linear discriminant think of y as W transpose X plus plus some v that that W gives me the one dimensional subspace. Now, this y will be a K minus 1 dimensional subspace the the issue is now in the 1 dimensional space finding the separation is quite straight forward.

How to rate differential 1 dimensional subspaces in terms of where the data of the 2 class are well separated? So, we have to do a little more work on suitably generalizing the, so called within class and between class scatter matrices. So, that we can find once again appropriate objective function the details are complicated and at least for this course we will omit them. It is there in the reference textbooks that were given in the beginning of the course, but the basic idea is just this.

We we find a appropriate K minus 1 dimensional subspace on which we project the data. So, this kind of completes our discussion of all linear classifiers, we have considered various methods of learning linear classifiers. Linear regression models and we have considered all the algorithms in great detail. Also seen how they can be generalized to the vector case in the regression and the multiclass case for classification.

(Refer Slide Time: 44:19)



- So far in the course we have seen some specific cases of learning classifiers given the training data.
- We have looked at the optimal Bayes classifier.
- We discussed how to estimate class conditional densities for implementing Bayes classifier.
- We have seen various methods (perceptron, least squares, LMS, logistic regression, FLD etc) to learn linear classifiers.
- We now take a look at the general problem of learning classifiers.

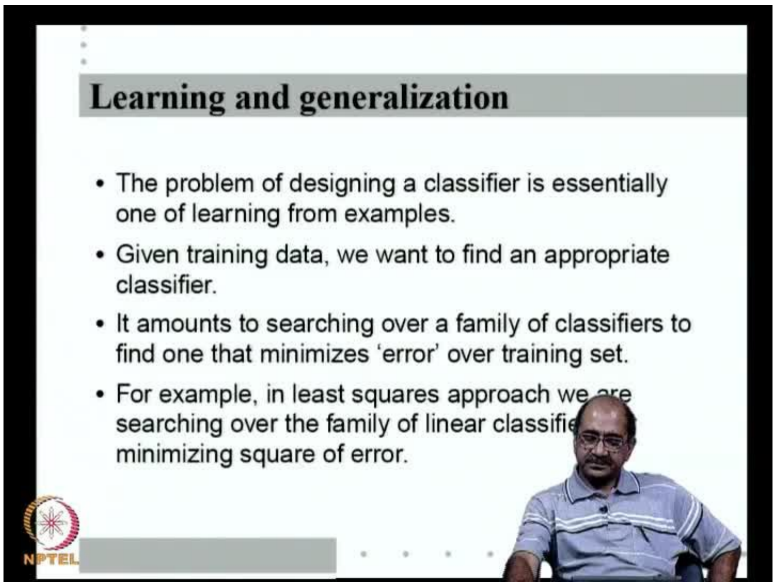
PR NPTEL course – p.76/102

So, what have we done, so far we have we looked at some specific cases of linear learning classifiers given the training data with we looked at 2 major things, so far one is the optimal Bayes classifier and how to estimate class conditional densities for implementing Bayes classifier? That is one way of, but of course Bayes classifier can also be viewed as a discriminant function as you already seen. So, we can think of that also as a discriminant function.

The only difference is it may not be linear, but if we can estimate class conditional densities, we can implement the Bayes classifier standard discriminant function. The other major chunk we looked at is various methods to learn linear classifiers. We looked at perceptron least squares l m s logistic regression fisher linear discriminant which I called f l d here. So, all these once again you know various ways of learning linear models for classification regression.

Now, now what we will do is we take a look at we take a more generalized look at the problem of learning classifiers. Now, that we at least got some algorithms we seen how the learning goes in practice what what the algorithms do on? So, on you can actually step back and ask what is common to these algorithms. Then look at some issues of whether we are learning the right classifier? Is there such a notion at the right classifier? How do we know what we learnt is correct? Issues like that.

(Refer Slide Time: 45:48)



## Learning and generalization

- The problem of designing a classifier is essentially one of learning from examples.
- Given training data, we want to find an appropriate classifier.
- It amounts to searching over a family of classifiers to find one that minimizes 'error' over training set.
- For example, in least squares approach we are searching over the family of linear classifie minimizing square of error.

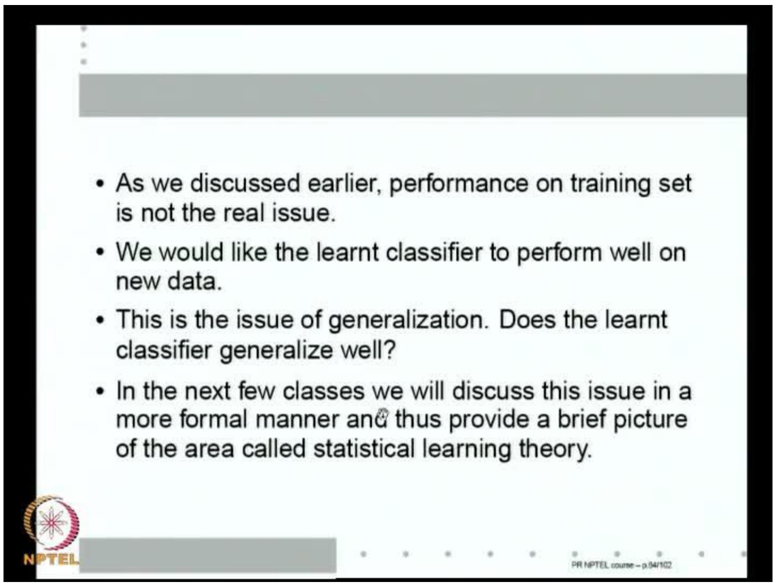
In the in the very beginning of the course we talked about generalization. We we we looked at the problem of designing classifiers as essentially one of learning from examples say we started with learning from examples at the basic issue. So, given training data, so we are given say in the classification 2 class classification case we are given feature vectors of class 1 feature vectors of class 2. We are essentially be asked to say what is common to all the feature vectors of class 1 and distinct from feature vector of class 2. There are various ways of representing this distinction.

Say for example, linear discriminant functions is one one way of saying is what is common to all feature vectors of class 1 is that if I use this W and W naught, then W times further expression W naught is greater than 0. If X is in class in 1 class less than 0 of X in other class. So, essentially given the examples, we want to learn some general principles by looking at the examples, so that is what we have been doing, given training data we want to find an appropriate classifier. We have some algorithms for finding classifiers of certain structures.

Now, what do all the classifiers amount to amount to searching over a family. Some family of functions to find one that minimizes error for, example our linear classification learning algorithms all they looking at a particular restricted class of classifiers they have restricted family of classifier namely linear classifiers. So, among all linear classifier that means among all W comma W naught are all all vectors W lest say augmented all vectors W. So, each W represents a classifier.

So, all vectors W is a family of classifiers we are asking among all vectors W, which gives me some minimum error over the training set. Error is how we define so far we only looked at squared error, so to say, but does not matter we are we are we are looking for one classifier over a family of classifiers that minimizes the error over training set. For example, in linear least squares we are searching over the family of linear classifiers for minimizing square of the error.

(Refer Slide Time: 48:00)



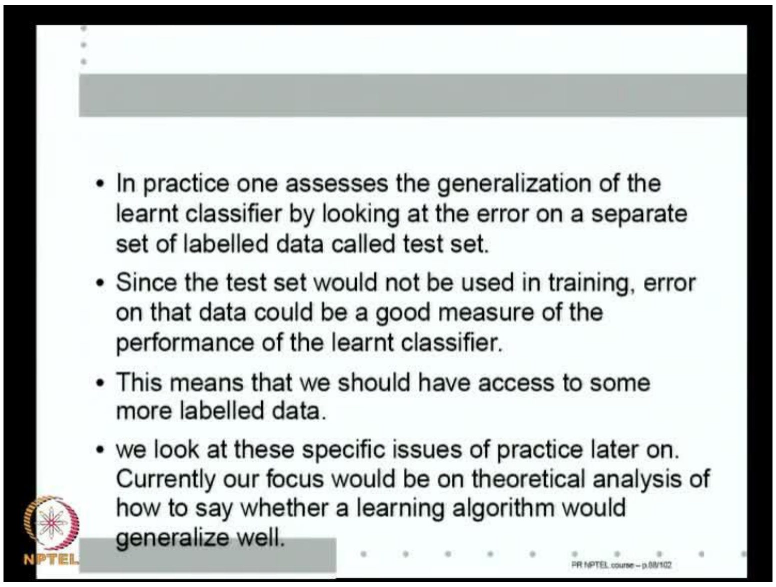
- As we discussed earlier, performance on training set is not the real issue.
- We would like the learnt classifier to perform well on new data.
- This is the issue of generalization. Does the learnt classifier generalize well?
- In the next few classes we will discuss this issue in a more formal manner and thus provide a brief picture of the area called statistical learning theory.

PR NPTEL course – p.84/102

As, we also discussed earlier performance on training set is not the real issue after for training set. We already know the class labels, so predicting the correct class labels and training set is not the issue is like solving an example problems in your maths course just. So, that you are testing yourself as you understand or not and one hope said by solving all the example problems well you you would do well in the test. So, in the same way, what the learnt classifier should do is to perform well on new data.

How do we know the classifier performs? Well on new data this as we considered earlier is the issue of generalization does the learnt classifier generalize. Well you have seen lot of examples by looking at the examples have a generalized well, so that I learnt the general principle, so that now I can classify any new pattern. So, what we are going to do next is look at this in a more formal sense, which provides us you know some way of asking some formal questions about generalization and in the process give you some idea of what is called statistical learning theory. Our our treatment on statistical learning theory has to be (( )) simplistic because we are not assuming too much mathematical background for this course, but we will look at the the the formal statistical learning theory.

(Refer Slide Time: 49:26)



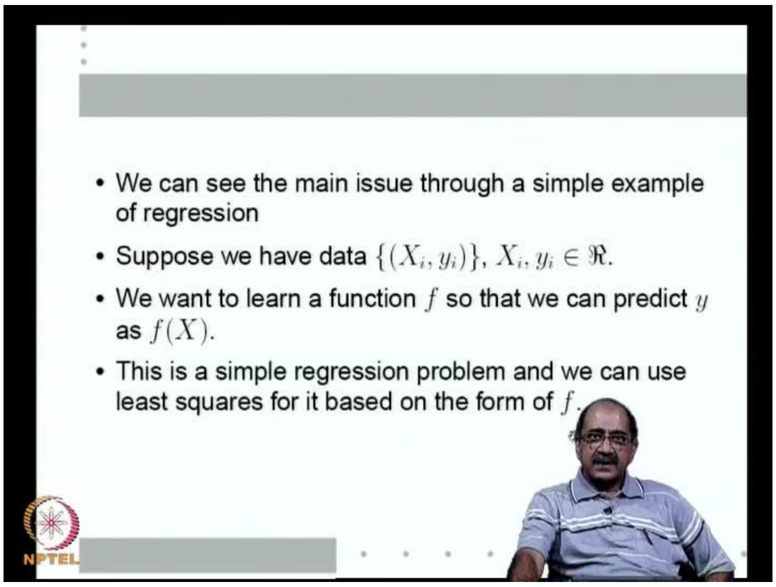
- In practice one assesses the generalization of the learnt classifier by looking at the error on a separate set of labelled data called test set.
- Since the test set would not be used in training, error on that data could be a good measure of the performance of the learnt classifier.
- This means that we should have access to some more labelled data.
- we look at these specific issues of practice later on. Currently our focus would be on theoretical analysis of how to say whether a learning algorithm would generalize well.

Let us quickly look before we go there the formalism will start next class, but before we go there let us just try to get some intuitive idea of what we are looking at in practice of course. We can always assess the generalization by looking at errors on a separate set just like the way we look at whether students learned well or not by giving a test. After the classifiers learnt we can give extra data, which is called the test set. That data also has labels, so I can calculate errors, so I just run the classifier and the test set and ask whether it does well.

Since, the test set is not used for training error on the test data could be a good measure of the performance of the learnt classifier this means that of course, we should have more data. We cannot use all the data we have for training and you know if we let away some data may be we do not have enough data for training. We will come to all these issues later on we will consider these also later in the course. So, the specific issues of practice we will consider later how one actually assesses in practice given classifier?

But right now our focus will be now more theoretical analysis of how to say whether a learning algorithm would generalize well or not given any data. Can I say this algorithm would generalize well and that algorithm may not generalize well and so on so forth.

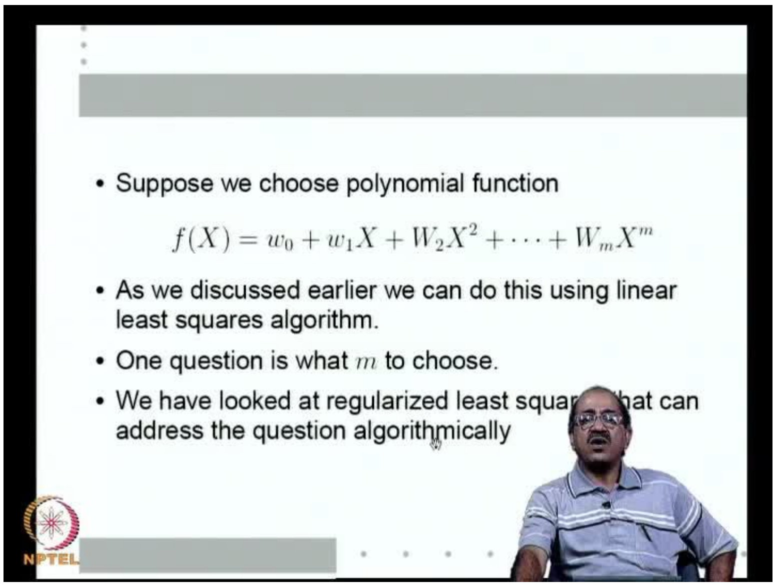
(Refer Slide Time: 50:40)



- We can see the main issue through a simple example of regression
- Suppose we have data $\{(X_i, y_i)\}$, $X_i, y_i \in \Re$.
- We want to learn a function $f$ so that we can predict $y$ as $f(X)$.
- This is a simple regression problem and we can use least squares for it based on the form of $f$.

So, to ask what this means let us let us take the every simple example of regression. Let us say we have 1 dimensional data X i y i both x X and y are in on real line. This is the standard least square curve fitting problem, which all of us have solved in class 12. We are giving points X and y and you want to find a good function relationship y is equal to f of X. We want to learn a function, so that we can predict y as f of X. Simplest regression problem we can use least squares for this. Basically of course, we have to have some particular form for f.

(Refer Slide Time: 51:19)



- Suppose we choose polynomial function

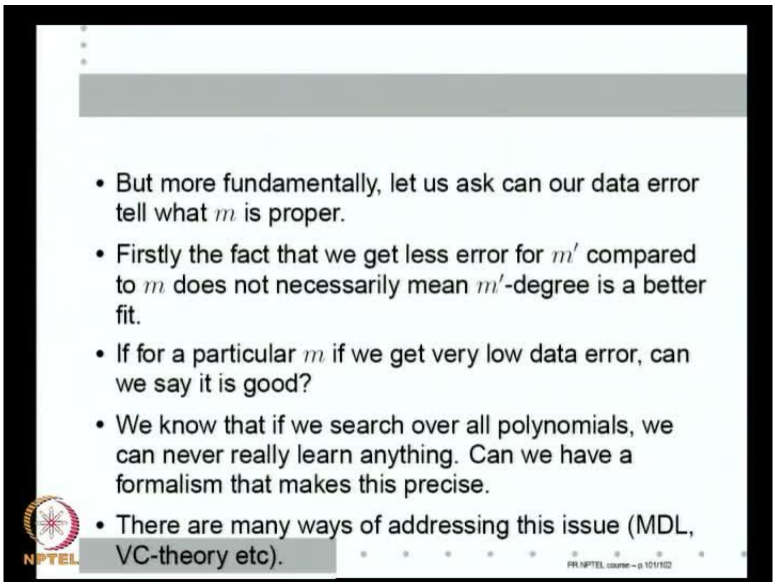
$$f(X) = w_0 + w_1 X + W_2 X^2 + \cdots + W_m X^m$$

- As we discussed earlier we can do this using linear least squares algorithm.
- One question is what $m$ to choose.
- We have looked at regularized least squar hat can address the question algorithmically

So, let us say we choose a polynomial function we do not know what m, but we choose a polynomial function w 0 w 1 X W 2 X square W m X m. As we have seen earlier, this is this easily fits into linear least squares. Linear least squares does not (( )) mean that the function is linear is only linear in parameters. So, already seen when we did least squares regression did this exactly fits into linear least square problems, so the question is what m. Should we choose we were already asked this question at an algorithmic level. We said that obviously we do not know what m to choose and all just, because because if we choose a very large m we may artificially think we are getting low data error.

So, we will introduce the regularization, which can be viewed as a Bayesian approach to estimation, but that is an algorithmic thing. We said we take both data error and some regularization term and there is a regularization constant. So these algorithmic issues are important and ultimately that is how we are going to control generalization. But first our focus is theoretically understanding, why we need such a regularization? So, essentially one way of thinking of it is what m should I choose.

(Refer Slide Time: 52:34)



- But more fundamentally, let us ask can our data error tell what $m$ is proper.
- Firstly the fact that we get less error for $m'$ compared to $m$ does not necessarily mean $m'$-degree is a better fit.
- If for a particular $m$ if we get very low data error, can we say it is good?
- We know that if we search over all polynomials, we can never really learn anything. Can we have a formalism that makes this precise.
- There are many ways of addressing this issue (MDL, VC-theory etc).

PR NPTEL course – p.101/102

Can our data error tell us what m is proper of course, we already know it cannot, but let us let us say what it means? Suppose I have 2 m's m 1 m prime that I get less error for m prime compared to m does not necessarily mean m prime degree polynomial is better than is a better fit. Suppose, you have given me the data I fit a straight line to the data. Let us say at for the best fit straight line I get my squared error as some 2.16. Some number, then let us say I fit a quadratic now obviously quadratic possibly may fit better. So, let us say I got 2.09 as the error for the best fit quadratic, does it mean that the quadratic is a better fit than linear?

If you think 2.09 is too close to 2.16, let us say for quadratic I get 1.98 is it better is 1.94 better. How do I decide? Can I say that looking at the errors is quadratic better than. Now in in its simplicity of course, this cannot be answered because we know if I have got n points and I fitted a n th degree polynomial. I will get 0 error. But obviously nobody would think of fitted saying the n th degree for polynomial is best fit for n points is like saying given 2 points I I can conclude that the relationship is linear or we cannot?

So, how do we address this issue more theoretically if at a particular m. We get very low data error can we say it is good obviously not as we just now seen if I fit a n th degree polynomial I get 0 error no nothing can be lowered than that. But that is not good, I can ask the other way how many examples I should have for me to believe that for low data error are? Means they will be low test error. For example, if I am fitting a straight line to

100 points and if I got point naught naught 1 error I can be I can I have fairly good confident that straight is the right thing. But if I got the same thing with only 2 or 4 points I do not know whether I got the right straight line.

So, one way of asking this is if I am fitting a particular kind of function how many examples should I need before I can believe my data error, so that is that is certainly 1 important question to ask. We know that say for example, if I want to fit 100 degree polynomial I intuitive no I need more points. Then I want to put third degree polynomial, but can I formalize this for a particular class of models that I want to fit? How many examples do would I need for me to have confidence in my data error? We also know that suppose I say set your all possible polynomial.

Then no matter how much data I have? I can never get anywhere if I do not fix m, but I say you said of course, now we do not know how to do that we only know least square algorithm. So, we can know it only if m is fixed, but let us say I have a learning algorithm which can search over whatever I want then if I say search over all possible polynomials. Then obviously no matter how much data I have I would never get a good fit. Because I always over fit I will always if I have n points I will always fit the n th degree polynomial because my bag of classifiers has polynomials of all degrees.

So, we know intuitively that we will never learn anything. If we do not restrict the class of models over which if can we formalize this can we make this notion more precise. When something learnable is, when is something not learnable, but there are many ways of handling this there are things called minimum description, length principle Vapnik Chervonenkis theory. So, on so what I am going to do is starting next class.

We will have a brief review of this in a in a in a in a intuitive form first and then we will only look at the, so called Vapnik Chervonenkis theory of statistical learning. That is the only formalism we will consider, which at least let us understand what model complexity mean? Why model complexity m comes where the regularization issues come up? It will also give you some idea of how many examples are sufficient for us to believe our data error. So, that is going to be our plan for the next few lectures.

Thank you.