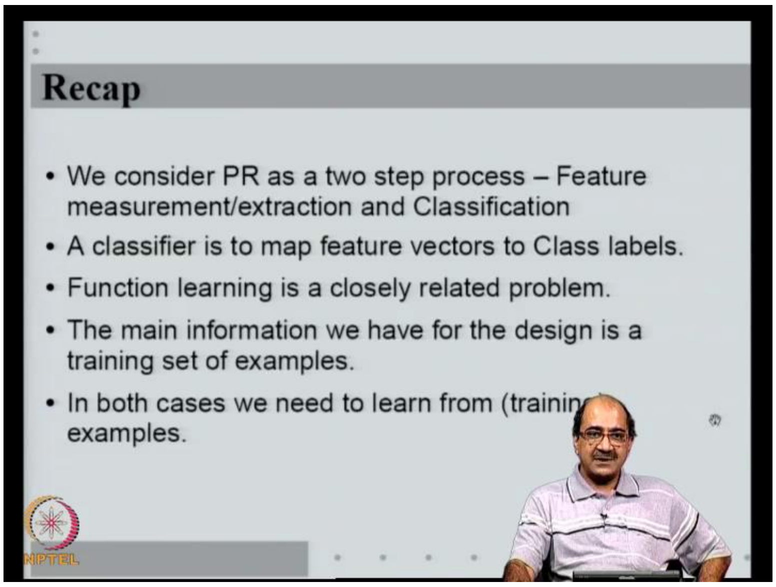
Pattern Recognition

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

Lecture - 2
Overview of Pattern Classifiers

Hello, welcome to this next lecture in Pattern Recognition. We will just briefly recall what has been done in the previous lecture, we have we as I said we consider pattern recognition as the two step process, featured measurement extraction, slash classification. First, there is a there are two steps; first, from the pattern you measure some features, so each pattern gets converted to feature vector, and then, there is the classification step, the classifier takes feature vectors and maps them to class labels. And as I just said, this course is about mostly designing classifiers, feature extraction is very much problem dependent and hence, there are not too many general techniques. But classifier design is what we would mostly be concentrating on and any classifier is something, that maps feature vectors to class labels.

(Refer Slide Time: 01:16)



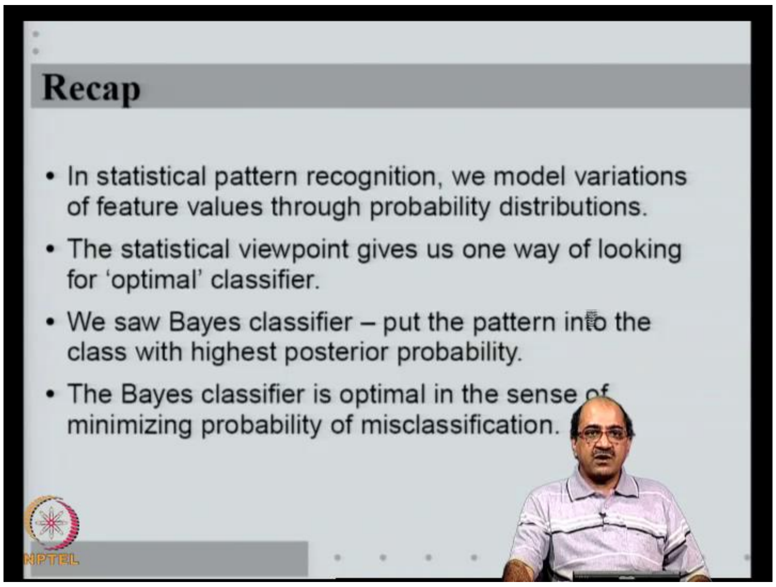
Recap

- We consider PR as a two step process – Feature measurement/extraction and Classification
- A classifier is to map feature vectors to Class labels.
- Function learning is a closely related problem.
- The main information we have for the design is a training set of examples.
- In both cases we need to learn from (trainin examples.

As I said, function learning or the regression problem is a closely related problem to pattern recognition and will be considering both of them together. In both cases, the main information we have, as we seen last time is a training set of examples where, given as set of examples, pattern belonging to class 0, pattern unpattern belonging to class 1. Such as, I have got many a’s and many b’s, and I need to learn a class for distinguished a and

b. So, the main information we have for the design, is a training set of examples and both for classified learning as well as regression or function learning, we need to learn from the training set of examples.

(Refer Slide Time: 01:51)



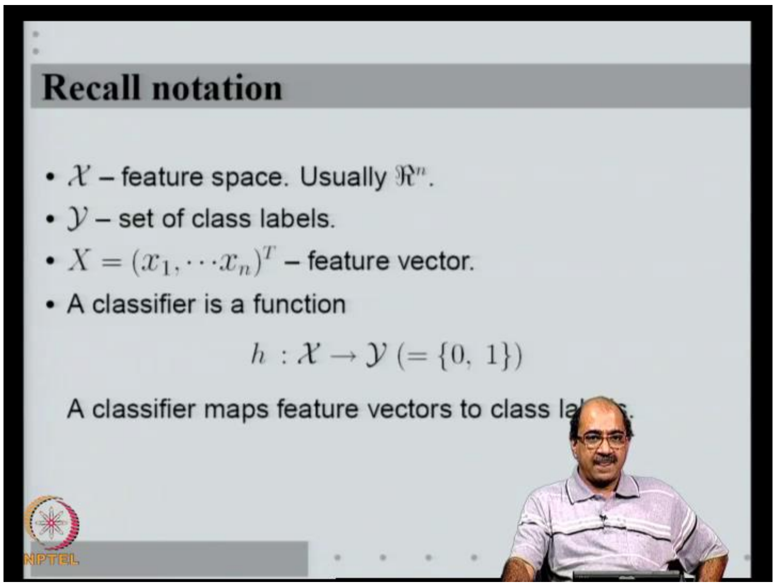
Recap

- In statistical pattern recognition, we model variations of feature values through probability distributions.
- The statistical viewpoint gives us one way of looking for 'optimal' classifier.
- We saw Bayes classifier – put the pattern into the class with highest posterior probability.
- The Bayes classifier is optimal in the sense of minimizing probability of misclassification.

In we have decided that, we will be following the statistical approached pattern recognition, so called statistical pattern recognition, the characterizing feature of it is that, we model the variations of feature values through probability distributions. So, a feature vector belonging to a class, now becomes a random variable or which takes different values with different probabilities. That is what, is meant by variations in feature values, is captured through probability distributions.

So, each of the feature vectors, so each of the features themselves will be contains random variables and the feature vector will be a vector of continuous random variables. As we seen, one immediate result of using the statistical viewpoint is that, we can define optimal right, we have looked at for example Bayes classifier, which says put the pattern in the in the class with highest posterior probability. Posterior probability, the probability that class is i, given the feature vector and we have mentioned that, the Bayes classifier is optimal in the sense that, it minimizes probability of misclassification. In this lecture we will we will show that the Bayes classified is indeed optimal.

(Refer Slide Time: 02:59)



Recall notation

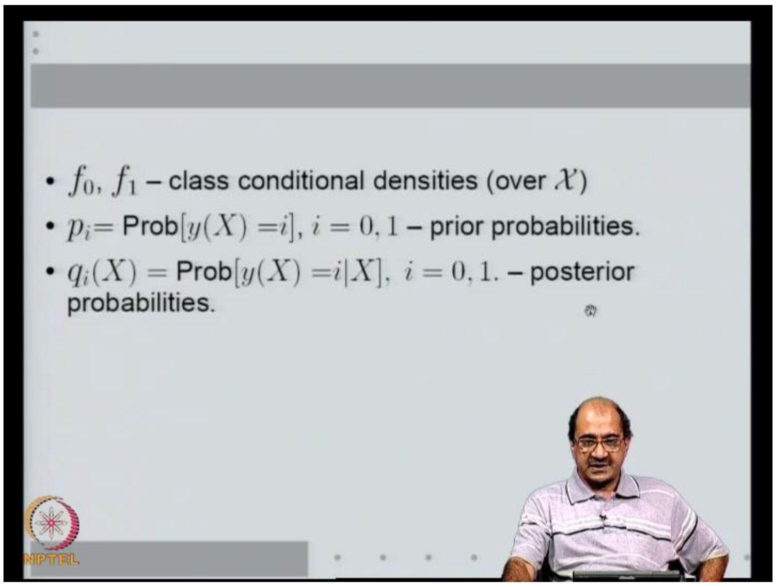
- $\mathcal{X}$ – feature space. Usually $\Re^n$.
- $\mathcal{Y}$ – set of class labels.
- $X = (x_1, \cdots x_n)^T$ – feature vector.
- A classifier is a function

$$h : \mathcal{X} \to \mathcal{Y} \; (= \{0, 1\})$$

A classifier maps feature vectors to class labels.

Let just briefly recall our notation, this script x denotes features space, the space of all possible feature vectors. Usually, if you have n dimensional feature vector that is, n features usually, the features space is R n, the undiminished real nucleance space, y is the set of class labels will be some finite set. From most of the course, we will be considering binary classification, in which case y is 0 1, X is the feature vector, as n components x 1 to x n. I have put a transpose on the expanded feature vector, because all vectors are column vectors. Hence, a classifier is a function that maps the feature space to class labels, this is for every feature vector, there is a class label, in in for for this class or class labels are only two is a binary classification problem, so class labels can it be either 0 or 1.

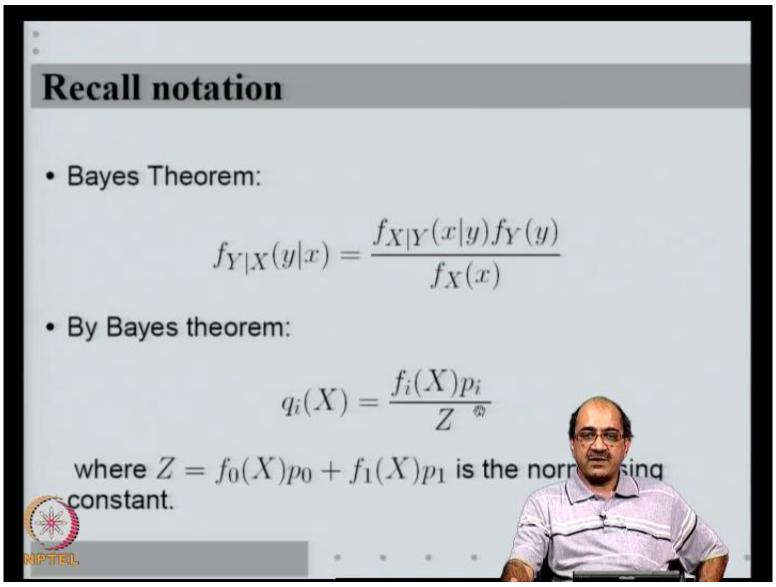
(Refer Slide Time: 03:48)



- f_0, f_1 – class conditional densities (over $\mathcal{X}$)
- p_i= Prob$[y(X) = i]$, $i = 0, 1$ – prior probabilities.
- $q_i(X)$ = Prob$[y(X) = i|X]$, $i = 0, 1$. – posterior probabilities.

We also introduce to the idea of class conditional densities and so on. So to recall f 0 and f 1 are the class conditional densities for class 0 and class 1, both are densities over the features space, p i is the prior probability, the probability that y of x equal to i, as I mentioned last time y of x denotes the random variable denoting the class of the feature vector X. Similarly, q i's are the posterior probability that, the probability that y of x is i conditioned on x, for a given x or the probability this class is i, those are the posterior probabilities.

(Refer Slide Time: 04:23)



Recall notation

- Bayes Theorem:

$$f_{Y|X}(y|x) = \frac{f_{X|Y}(x|y)f_Y(y)}{f_X(x)}$$

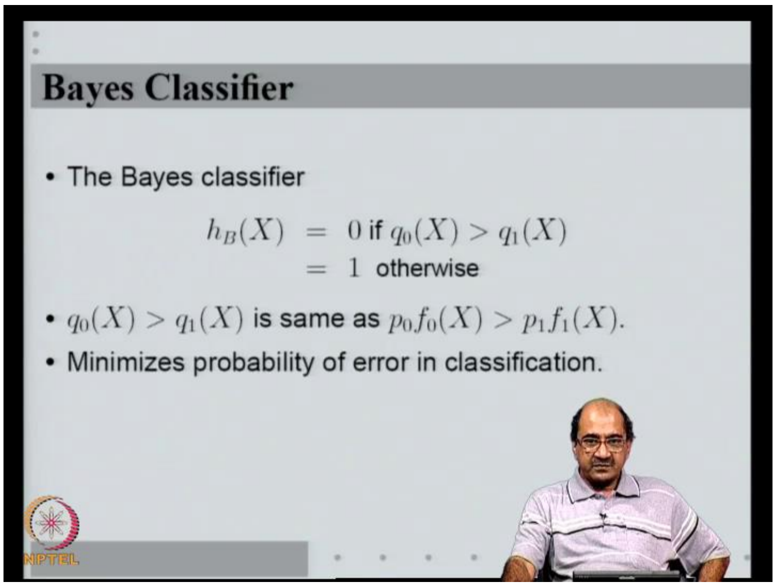
- By Bayes theorem:

$$q_i(X) = \frac{f_i(X)p_i}{Z}$$

where $Z = f_0(X)p_0 + f_1(X)p_1$ is the norr[illegible]sing constant.

Still recalling our notation, we as we have mentioned last time, Bayes theorem is given there we can get one conditional density in terms of other conditional density. Applying to our current case, it is a tells you that, q i x is f i x p i by z where, f i is class conditional and p i is prior probability, q i's are the posterior probabilities; where, Z is the normalizing constant, which simply f 0 p 0 plus f 1 p 1.

(Refer Slide Time: 04:54)



Bayes Classifier

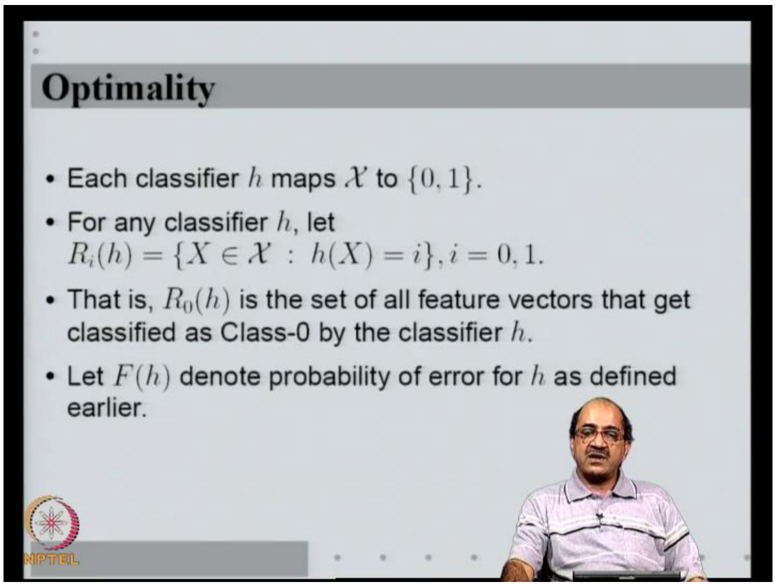
- The Bayes classifier

$$h_B(X) = 0 \text{ if } q_0(X) > q_1(X)$$
$$= 1 \text{ otherwise}$$

- $q_0(X) > q_1(X)$ is same as $p_0 f_0(X) > p_1 f_1(X)$.
- Minimizes probability of error in classification.

So, now, let us stay at Bayes classifier again, Bayes classifier is the function, which given an x assigns 0, if q 0 x is greater than q 1 x otherwise, assigns 1. As I said, Bayes classifier puts a feature vector X into the class, whose posterior probability is higher, q 0 is the posterior for class 0. So, if q 0 is higher than q 1, I put X in class 0, so that is the classifier, I put b a subscript and h to say, is a Bayes classifier. From our Bayes theorem, we know that checking q 0 greater than q 1 is same as checking p 0 f 0 greater than p 1 f 1. So, all we need here, class conditional densities and prior probabilities, and we are going to show, that the Bayes classifier minimizes probability of error in classification.

(Refer Slide Time: 05:37)

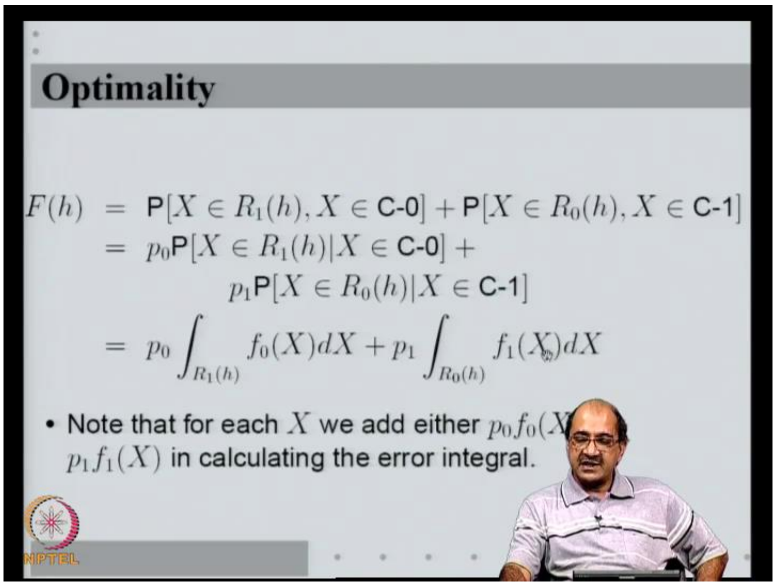


Optimality

- Each classifier h maps $\mathcal{X}$ to $\{0, 1\}$.
- For any classifier h, let $R_i(h) = \{X \in \mathcal{X} \ : \ h(X) = i\}, i = 0, 1.$
- That is, $R_0(h)$ is the set of all feature vectors that get classified as Class-0 by the classifier h.
- Let $F(h)$ denote probability of error for h as defined earlier.

As go through with these simple proof, as we know, each classifier h maps this the features space x to the set of class labels that is, 0 comma 1. So, for a given h, given a classifier, define R i of h at the set of all x, for which h x is equals to i that is, for example, R subscript 0 of h is the set of all feature vectors, that get mapped on to class 0 by classifier h. So, R 0 h is the set of feature vectors, which will be put in class 0 by a classifier h similarly, R 1 or the set of feature vector that are put in class 1 by a classifier h., as we defined last time, let F of h denote the probability of error for the classifier h.

(Refer Slide Time: 06:23)



Optimality

$$\begin{aligned} F(h) &= \text{P}[X \in R_1(h), X \in \text{C-0}] + \text{P}[X \in R_0(h), X \in \text{C-1}] \\ &= p_0 \text{P}[X \in R_1(h) | X \in \text{C-0}] + \\ &\qquad p_1 \text{P}[X \in R_0(h) | X \in \text{C-1}] \\ &= p_0 \int_{R_1(h)} f_0(X) dX + p_1 \int_{R_0(h)} f_1(X) dX \end{aligned}$$

- Note that for each X we add either $p_0 f_0(X$ $p_1 f_1(X)$ in calculating the error integral.

Now, we can derive an expression for F of h, when will we make an error if the classifier says class 1, while it is actually class 0 or vice versa. So, F of h is the probability, that X actually belongs to the region R one h where, the classifier would say class 1 and also X belongs to class 0 or disjointly X belongs to R 0 h and X belongs to C 1. Because, R 1 and R 0 disjoin sets, these two events are disjoined and the probability gets added. Now, we can always write probability a a and b as probability a given b into probability b.

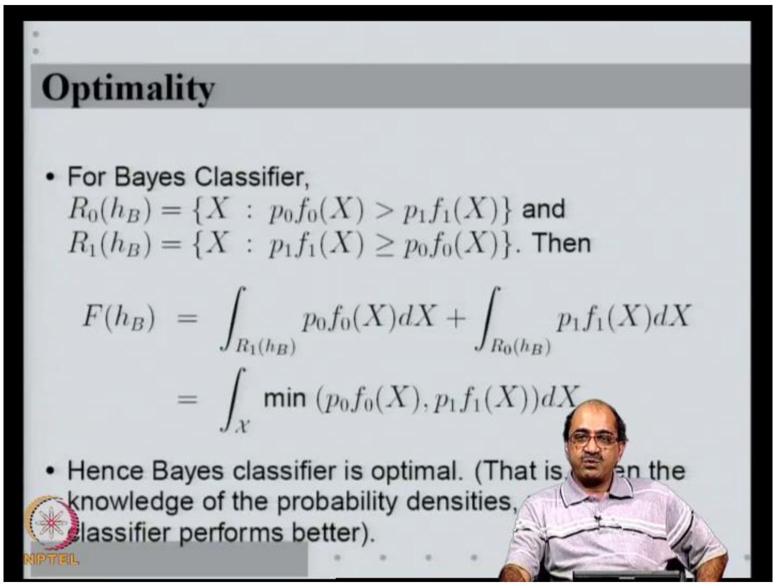
So, here I can write this as probability X belongs to R 1 h, conditioned on X belonging to C 0 multiplied by a probability X belongs to C 0 and probability x belongs to C 0 is the prior probability p 0. So, now, F of h becomes p 0 times probability X belongs to R 1 h condition X belongs to C 0 plus p 1 times probability X belongs to R 0 h, condition X belongs to C 1.

Now, we know that, condition on X belonging to C 0, the random variable X has density f 0 similarly, conditioned on X belonging to C 1, the random variable is density f 1. So, I can write both these probabilities as integration of the corresponding density functions or appropriate regions, so this becomes the value of F h. Now, in this expression there is something, that is very important to note R 0 h and R 1 h are mutually disjoint and they add up to the whole space.

Because, given a h, every X is either put in the space R 0 or put in the space R 1, for all X in R 1, the integral adds f 0 into p 0. And for every x, in R 0, the integral adds p 1 into f 1, which means for any classifier, the other integral is such that, at every X, we either add a p 0 f 0 or p 1 f 1. So, we actually integrate for the entire space either p 0 f 0 or p 1 f 1 at each X.

At which X we put p 0 f 0 in the integral and at which X we put p 1 f 1, depends on whether that X is in the region R 0 or R 1. And all that the classifier can do is to change these regions, it it can only decide, at which points p 0 f 0 has count in the integral and at which points, p 1 f 1 is count in the integral, the classifier can only change the regions R 0 or R 1.

(Refer Slide Time: 08:50)



Optimality

- For Bayes Classifier, $R_0(h_B) = \{X : p_0 f_0(X) > p_1 f_1(X)\}$ and $R_1(h_B) = \{X : p_1 f_1(X) \geq p_0 f_0(X)\}$. Then

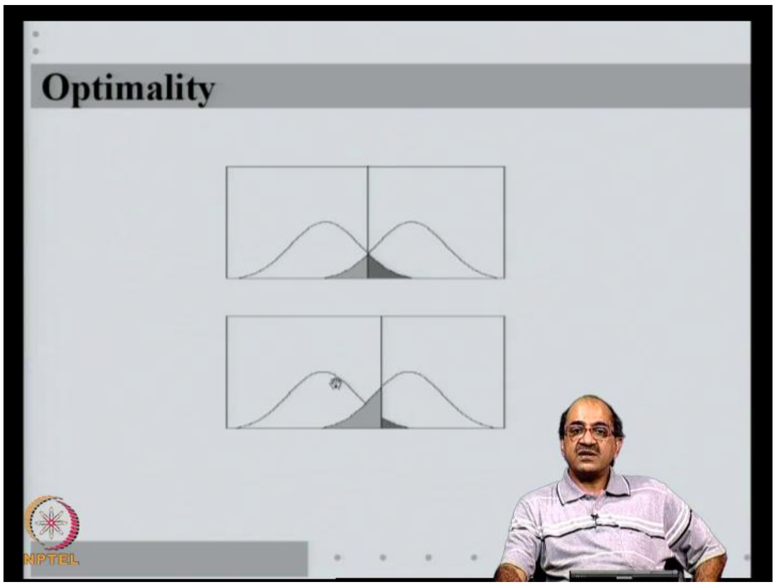
$$F(h_B) = \int_{R_1(h_B)} p_0 f_0(X) dX + \int_{R_0(h_B)} p_1 f_1(X) dX$$
$$= \int_{X} \min\left(p_0 f_0(X), p_1 f_1(X)\right) dX$$

- Hence Bayes classifier is optimal. (That is, [illegible]en the knowledge of the probability densities, [illegible] classifier performs better).

Now, let us look at, what the Bayes classifier regions are, when a Bayes classifier puts a pattern in R 0, if p 0 f 0 is greater than p 1 f 1 and puts it in R 1, if p 1 f 1 is greater than p 0 f 0. So, using the previous expression for the Bayes classifier, the probability of error of the Bayes classifier becomes integral of p 0 f 0 on R 1 plus integral of p 1 f 1 on R 0. But the R 1 and R 0 are such that in the region R 1, p 0 f 0 is less than p 1 f 1 and in the region R 0, p 1 f 1 is less than p 0 f 0, which means this integral is integral over the entire feature space of minimum between p 0 f 0 and p 1 f 1.

Since for every classifier h , F of h is integral where, at each X, we have to put either p 0 f 0 or p 1 f 1 and both of them are positive quantities, and Bayes classifier at each X, picks the minimum of them, no classifier can do better than this. So, Bayes classifier is optimal in the sense, is probability of error is less than that of any other classifier that is, given the knowledge of the probability densities, no other classifier can perform better. Let us say, very strong result the classifier is very intuitive, if we put it in the class, which has highest posterior probability that automatically minimizes probability of error.

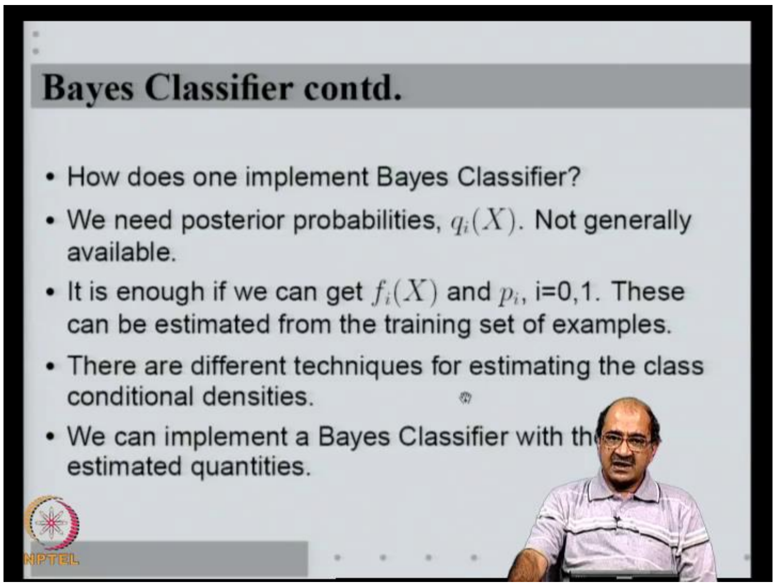
(Refer Slide Time: 10:13)


Optimality

So, one way of looking at this is suppose the left curve is p 0 f 0 and right curve is p 1 f 1, say essentially at any given X, this is a one dimensional features space, though the x axis is the feature space. So, given any X, that is the value of p 0 f 0 there and that is the value of p 1 f 1, whichever is higher, I put it in that class. So, that line denotes the optimal Bayes threshold, to the left of this line, the put on the on this class, right of that line, they have put on that class and the shaded regions are the error.

The error comes from a the area under this curve is the probability that feature vector of this class will have value outside this line and similarly, area on this side is that a feature vector of this class, will have it is value on this side right. So, as you can see, any classifier has to essentially put a threshold suppose, I put a threshold somewhere else one can see that, the error integral has increased that is, exactly the proof that you have done.

(Refer Slide Time: 11:20)



Bayes Classifier contd.

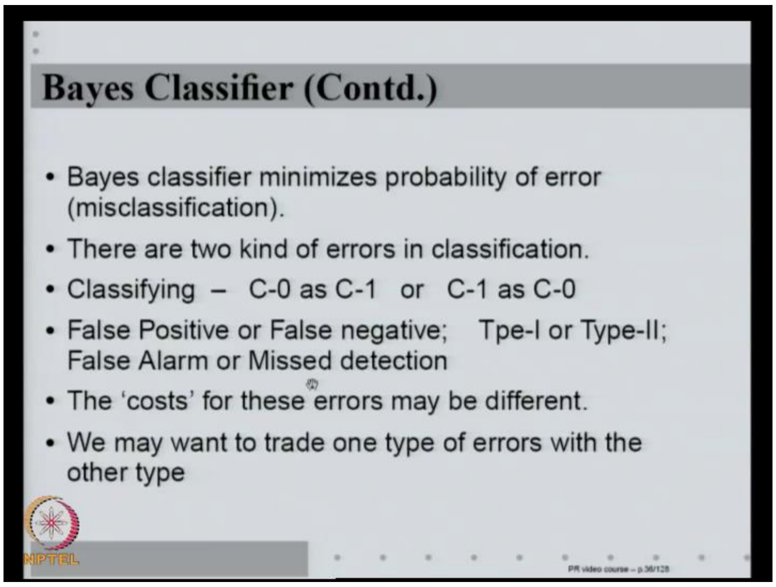
- How does one implement Bayes Classifier?
- We need posterior probabilities, $q_i(X)$. Not generally available.
- It is enough if we can get $f_i(X)$ and p_i, i=0,1. These can be estimated from the training set of examples.
- There are different techniques for estimating the class conditional densities.
- We can implement a Bayes Classifier with th estimated quantities.

So, Bayes classifier is of course, optimal to minimize the probability of error but, how does one implement the Bayes classifier, to implement the Bayes classifier, we need the posterior probabilities. That have those are not available to us, as I said the only information we have, is the training set of examples now, how do I get the posterior probabilities. As you already seen, we is enough, if we have the class conditional densities and the prior probabilities because, checking whether q 0 is greater than q 1 is same as checking where, the p 0 f 0 is greater than p 1 f 1.

Now, these class conditional densities and prior probabilities can be estimated from the training set of examples and there are different techniques of estimating this class conditional densities, which we will consider later on in the course. And then, one can implement a Bayes classifier with the estimated quantities, one small caveat is that, the Bayes classifier is optimal, if I exactly know the posterior probabilities.

But if I am implementing the Bayes classifier with estimated posterior probabilities obviously, the optimality no longer holds because, there may be others in estimation. And hence, my classifier is not truly the Bayes classifier but however, it is, that is the best we can do. So, we will estimate the class conditional densities and prior probabilities from the training set of examples and implement the Bayes classifier. We will of course, look at the techniques for estimating and also many other details about the Bayes classifier, as the course progresses.

(Refer Slide Time: 12:44)


Bayes Classifier (Contd.)

- Bayes classifier minimizes probability of error (misclassification).
- There are two kind of errors in classification.
- Classifying – C-0 as C-1 or C-1 as C-0
- False Positive or False negative; Tpe-I or Type-II; False Alarm or Missed detection
- The 'costs' for these errors may be different.
- We may want to trade one type of errors with the other type

PR video course – p.36/128

But Bayes classifier minimizes probability of error that is what we said, of two flows that it giving a class label which is wrong, is minimum amount all possible classifiers. But, there are two kinds of errors in classification, we can classify a class 0 pattern as a class 1 pattern or a class 1 pattern as a class 0 pattern. These are variously called as false positives or false negatives, type one error or type two error, false alarm, missed detection and so on.

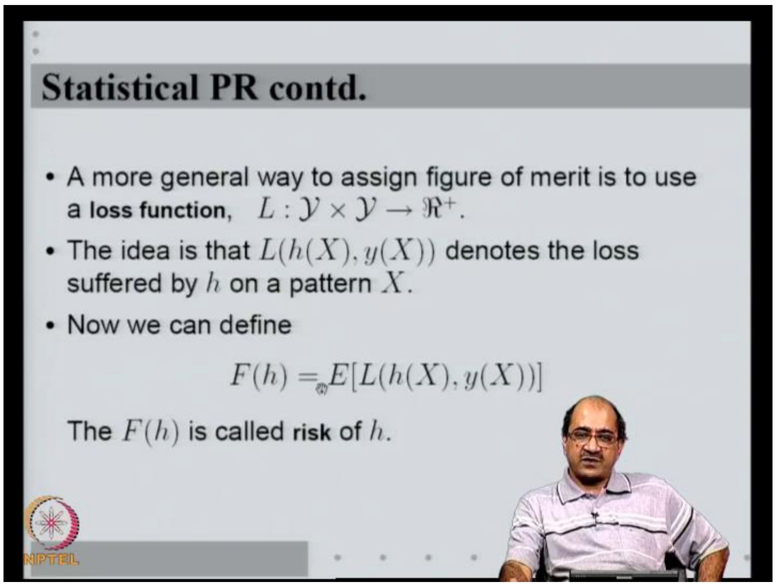
Essentially, some of this terminology comes from second world war, during the second world war time is, when this Bayes generation theory was actually developed. And the important classification problem to be solved is, they are these radars to detect fighter air craft enemy, air craft. So, based on the radar signal reflection that we got, somebody has to make a decision, whether what we have is a enemy air craft coming into bomb or there is no enemy air craft.

So, if those are class 0 and class 1, if I make an error, there can two kinds of errors, one is when I there is an enemy air craft, I actually think there is no enemy air craft or when there is no enemy air craft where, just a flying bird or something, I may think that is an enemy air craft. There are two kinds of errors and; obviously, the cause of these errors are very different right, a false alarm that is, thinking that something is an enemy air craft is only so much of inconvenience for people.

Because, I may sound an alarm or everybody goes into bomb shelters and other hand, I miss detection when there is actually an enemy air craft. But, I do not think that, there is one can be a very, very fact like that, many people may die or similar things happen in biometric recognitions. Suppose, I have I have an identity authentication system, you give your finger print to authenticate a customer. If he is genuine customer but, the system says he is not genuine, there is only a phone call from an angry customer.

On the other hand, a non-genuine customer gives his finger print and the system wrongly takes him to be genuine customer, that can be real loss to the system. So, in general, the cause of errors may be different and we may want to trade one type of error with other. We may not be bothered about minimizing probability of error but may we may want to minimize the cost of the error well, that can also be done.

(Refer Slide Time: 15:11)



Statistical PR contd.

- A more general way to assign figure of merit is to use a **loss function**, $L : \mathcal{Y} \times \mathcal{Y} \rightarrow \Re^+$.
- The idea is that $L(h(X), y(X))$ denotes the loss suffered by h on a pattern X.
- Now we can define

$$F(h) = E[L(h(X), y(X))]$$

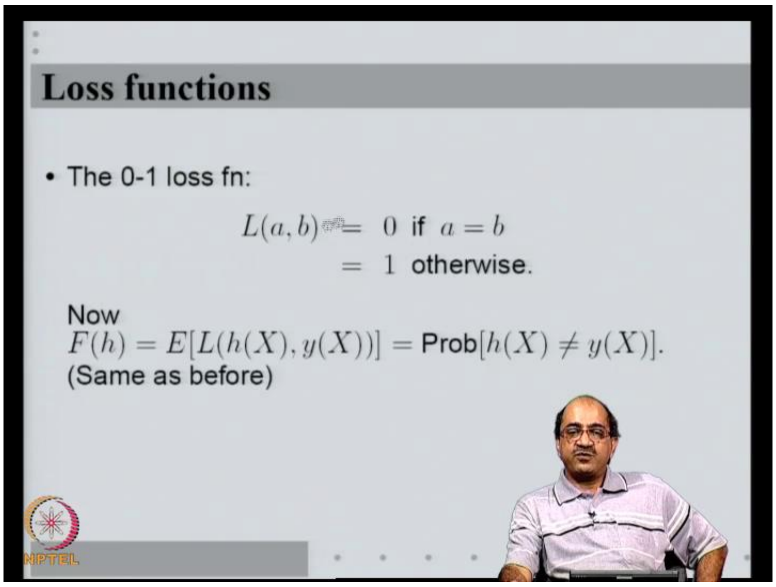
The $F(h)$ is called **risk** of h.

It is done by, what is called loss functions. So, earlier we have been rating classifiers based on their probability of error but, a more general method to assign a rating or a figure of merit to a classifier is, what is called a loss function. A loss function is a function, whose domain is y cross y, y as you you may remember, is the set of class labels. That is, given any pair of class labels, it assigns a positive real number, l is a function from y cross y to R plus, R plus is the positive real line.

So, given any pair of class labels, it assigns a positive number to it, the idea is that, on a given X, y X the random variable denotes the true class of X, h of X denotes the class labels that a classifier h will give. So, l of h X comma y X denotes the loss suffered by a classifier h and pattern X, the classifier would call over h of X whereas, it should have called out y of X. So, the loss function measures the discrepancy between h X and y X, the cost of the discrepancy between h X and y X.

So, the idea is that, I choose a loss function so that, l of h X comma y X denotes, in in in some units, the actual loss I suffered by h in a pattern x in the sense that, h would have called h of X whereas, the it should have called y of X. Now, we can define our figure of merit F, which assigns a number to each classifier h, as expected value of l of h X comma y X. This expectation is of course, with respect to the underlying distributions of X and certain expectation of the loss function is called the risk function. So, many classification algorithms actually choose a convenient loss function and then, try to minimize their risk, risk is the expectation of loss.

(Refer Slide Time: 16:56)



Loss functions

- The 0-1 loss fn:

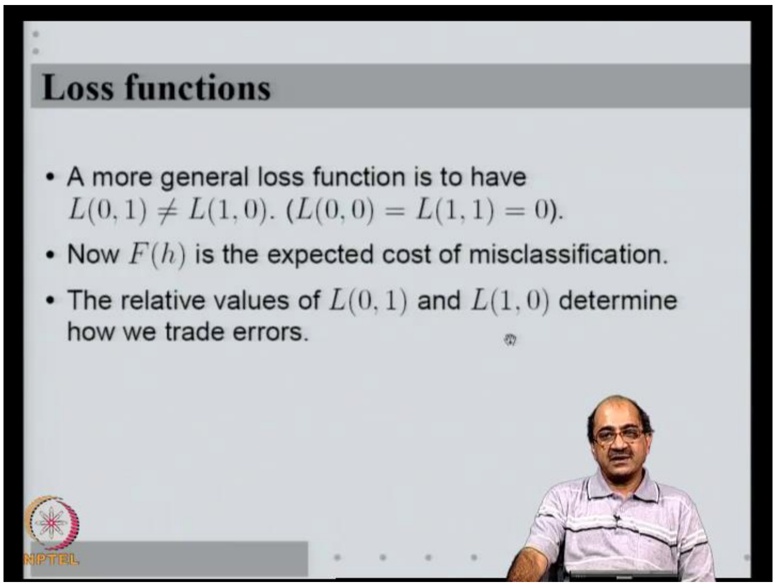
$$L(a,b) = 0 \text{ if } a = b$$
$$= 1 \text{ otherwise.}$$

Now
$F(h) = E[L(h(X), y(X))] = \text{Prob}[h(X) \neq y(X)]$.
(Same as before)

So, let us take some examples something called the zero one loss function, defined by l of a b is 0, if a equal to b 1, otherwise. That is, l has two arguments, one is about the classifier sets, other is about the two classes, if those are same, there is no cost otherwise, the cost is 1, irrespective of each error is made, so is like earlier thing. So, what will be the expectation of l l is a zero one random variable.

So, it is expectation is same as probability, that takes value 1, as probability h x is not equal to y X, which is same as the probability of error we defined earlier. So, what we did earlier can also be done through loss functions if we think of, what is called a zero one loss function.

(Refer Slide Time: 17:38)

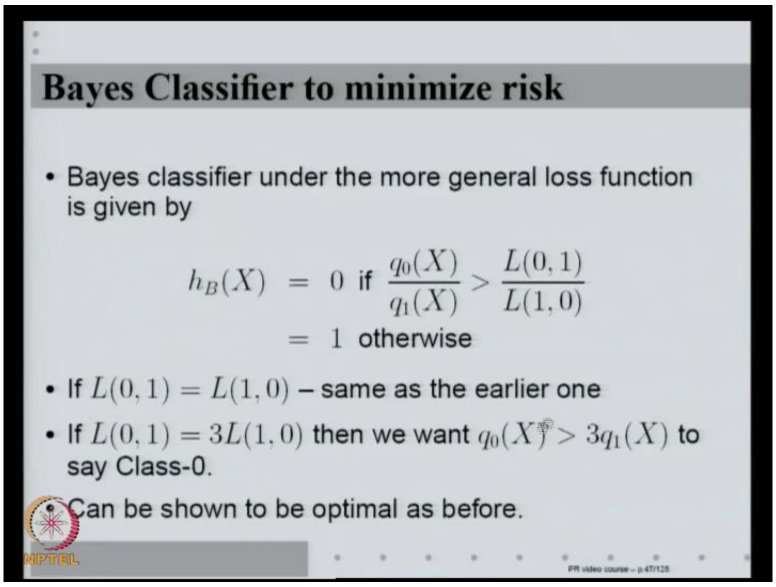


Loss functions

- A more general loss function is to have $L(0,1) \neq L(1,0)$. ($L(0,0) = L(1,1) = 0$).
- Now $F(h)$ is the expected cost of misclassification.
- The relative values of $L(0,1)$ and $L(1,0)$ determine how we trade errors.

But loss function can be more general that is to say, even though, loss and correct decision is 0, loss and incorrect decision need not have to be equal. In the zero one loss function, the loss and the incorrect decision is same, whenever a is not equal to b, the loss is same. But we can in general, have L of 0 1 not equal to L of 1 0, L of 0 1 is, when I say 0 and two class is 1 versus when I say 1 whereas, two class is 0. Now, the F h, which is the expectation of loss, which is the risk, is the expected cost of misclassification right. Essential, the relative values of L 0 1 and L 1 0 determine, how we can trade one kind of error with another kind of error.

(Refer Slide Time: 18:17)



Bayes Classifier to minimize risk

- Bayes classifier under the more general loss function is given by

$$h_B(X) = 0 \text{ if } \frac{q_0(X)}{q_1(X)} > \frac{L(0,1)}{L(1,0)}$$
$$= 1 \text{ otherwise}$$

- If $L(0,1) = L(1,0)$ – same as the earlier one
- If $L(0,1) = 3L(1,0)$ then we want $q_0(X) > 3q_1(X)$ to say Class-0.
- Can be shown to be optimal as before.

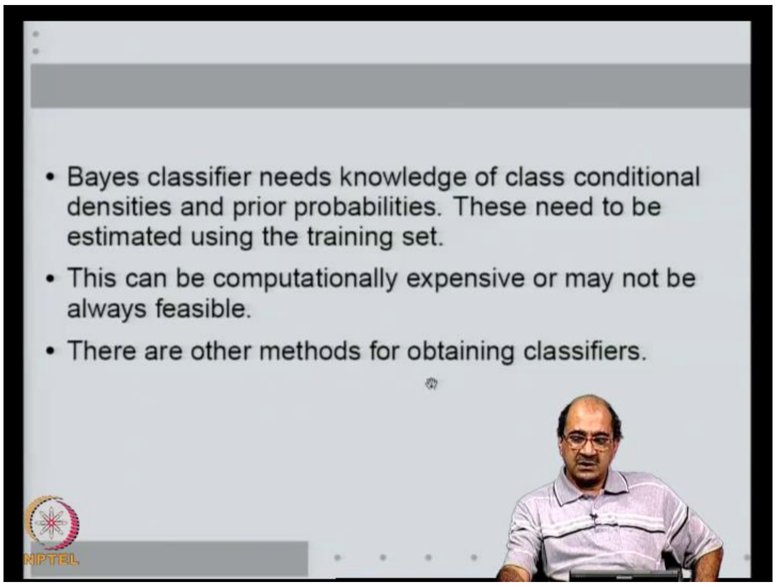
PR video course – p.47/128

As it turns out, we will drive this later in the course, as it turns out the Bayes classifier in this case will be, it it is once again based on the ratio of the posterior probabilities. Earlier, if the ratio is greater than 1, we are putting in class 0 but, now the ratio has to be greater than some threshold, which is given by L 0 1 by L 1 0. Just to intuitively get a field for it, if L 0 1 is equal to L 1 0, that the two kinds of errors of the same cost then, it is same as the old Bayes classifier.

On the other hand suppose, L 0 1 is 3 times L 1 0 what does that mean, L 0 1 is, I say 0 whereas, the two class is 1, so wrongly classifying something into 0 is thrice as costly as wrongly classifying something into 1. That means, before I classify something into 0, I better be very, very careful because, wrongly calling 0 is 3 times as expensive and hence, what does the Bayes classifier say, if that is the case, is not enough if q 0 is greater than q 1.

The posterior probability as 0 is greater than posterior probability of 1, that is not enough, posterior probability of 0 should be 3 times more than, 3 times the posterior probability 1, for me to take the risk and call class 0. So, that is essentially, what the Bayes classifier represent, we will show later on formally that, this transfer to the Bayes classifier that minimizes risk. And once again for the criteria of minimizing risk, Bayes classifier is optimal, that also we will show later on.

(Refer Slide Time: 19:43)



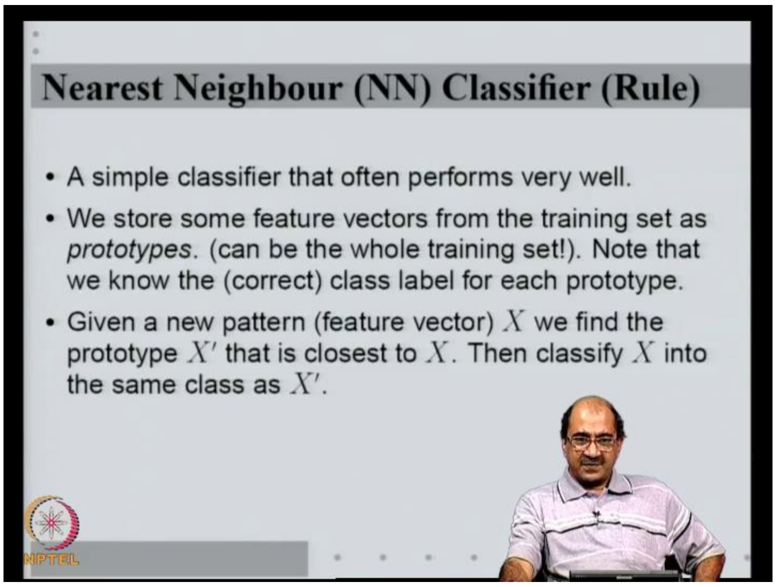
- Bayes classifier needs knowledge of class conditional densities and prior probabilities. These need to be estimated using the training set.
- This can be computationally expensive or may not be always feasible.
- There are other methods for obtaining classifiers.

So, Bayes classifier is indeed a very good classifier, it can minimize risk for any kind of loss function and that is that is that is, the best we can ask for but, the it is real problem is that, it needs knowledge of class conditional densities and prior probabilities. If I know them of course, I can do nothing better but, if I do not know them, they have to be estimated, the estimation has it is problems.

Firstly, it may be computationally very expensive, as we shall see or it may be not feasible sometimes because, the underlying densities are not very nice or no, there might be other reasons, why I may not want to estimate the class conditional densities. Of course, then, I cant implement, also if I implement the Bayes classifier with estimated densities because, address in estimation it may, may no longer be optimal.

So, considering all this, implementation wise I will go for Bayes classifier, only if I have sufficient faith in the estimated class conditional densities. Otherwise, what do I do, there are many other methods of obtaining classifiers. So, for the rest of this class, we will at least quickly review a few of them, many of them we will we will look at more details later on in the course.

(Refer Slide Time: 20:57)



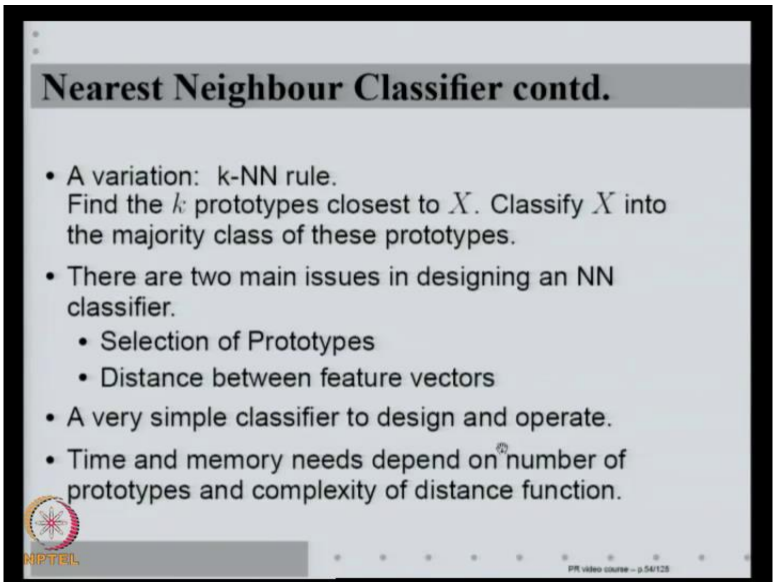
Nearest Neighbour (NN) Classifier (Rule)

- A simple classifier that often performs very well.
- We store some feature vectors from the training set as *prototypes*. (can be the whole training set!). Note that we know the (correct) class label for each prototype.
- Given a new pattern (feature vector) X we find the prototype X' that is closest to X. Then classify X into the same class as X'.

So, one other method, which is very simple and often performs well is, what is called a nearest neighbor classifier. What is the nearest neighbor classifier? Which store the training example, the feature vectors from the training set, as prototypes. We can choose some of them or we can choose the whole training set but, anyway we take the training examples, store them and call them prototypes. The thing to remember it that because, the training set for each each prototype, we know the actual class label because, we have the training set.

Now, we are ready to classify, given a new pattern, we find the prototype X prime that is closest to the new pattern X and then, classify X into the same class as X prime. That is, now you give me a new pattern X to classify within my prototypes as h for X prime that is closest to X. Now, because x prime is a prototype, I know the class label of X prime, I give the same class label to X, later I have sufficiently many prototypes, anyone lying close to a particular prototype should have the same class label as the prototype.

(Refer Slide Time: 21:58)



Nearest Neighbour Classifier contd.

- A variation: k-NN rule.
  Find the k prototypes closest to X. Classify X into the majority class of these prototypes.
- There are two main issues in designing an NN classifier.
  - Selection of Prototypes
  - Distance between feature vectors
- A very simple classifier to design and operate.
- Time and memory needs depend on number of prototypes and complexity of distance function.

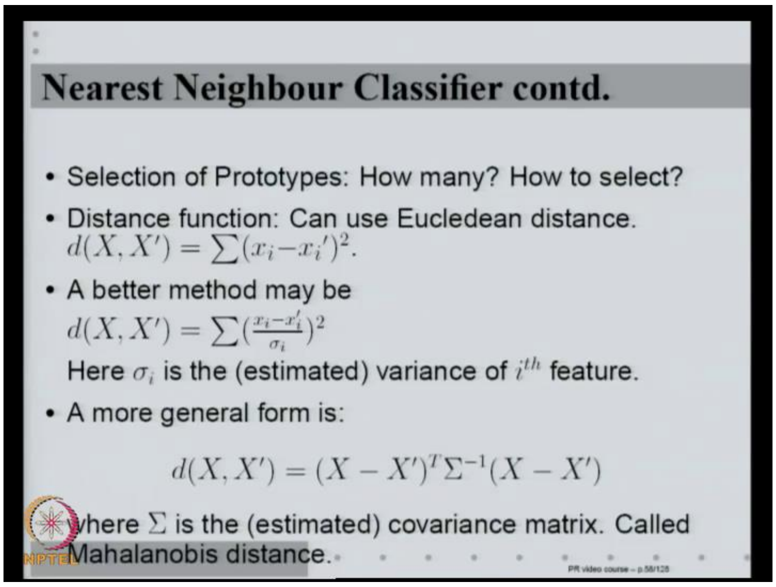
PR video course – p.54/128

A slight variation is, what is called a k nearest neighbor rule, so instead of finding 1 closest prototype, I may find 5 closest prototypes and then, take a majority rule. In a two class case, if I find 5 closest prototypes, 3 or more of them are in one class, so I will chose that class for X. It is it is kind of, if a patterns happens to be near the class boundary instead of, taking one nearest prototype, taking a few more and taking the majority decision generally, gives you less errors, this is called a k nearest neighbor rule.

Generally you take k to be the k to be an odd integer, small odd integer 3 or 5 or 7, the two main issues in designing a nearest neighbor classifier. We have to decide, what we want to chose at prototypes, do you want to choose the whole of the training set as prototypes or if you want to choose a subset, which subset should we chose and what do you mean by closest right, what distance is the most appropriate distance.

But given that, we can solve these two issues, this is a very simple classifier to design and operate it, really there is nothing I need to , except to store the prototypes. And the classification rule is also very simple, I just need to find the find a prototype that is closest to the new pattern and classify it into the same class. The actual performance, how much of time and memory you need, depends on the number of prototypes, how complex is the distance function and so on.

(Refer Slide Time: 23:30)



Nearest Neighbour Classifier contd.

- Selection of Prototypes: How many? How to select?
- Distance function: Can use Eucledean distance. $d(X, X') = \sum (x_i - x_i')^2$.
- A better method may be $d(X, X') = \sum (\frac{x_i - x_i'}{\sigma_i})^2$ Here σ_i is the (estimated) variance of i^{th} feature.
- A more general form is:

$$d(X, X') = (X - X')^T \Sigma^{-1} (X - X')$$

where Σ is the (estimated) covariance matrix. Called Mahalanobis distance.

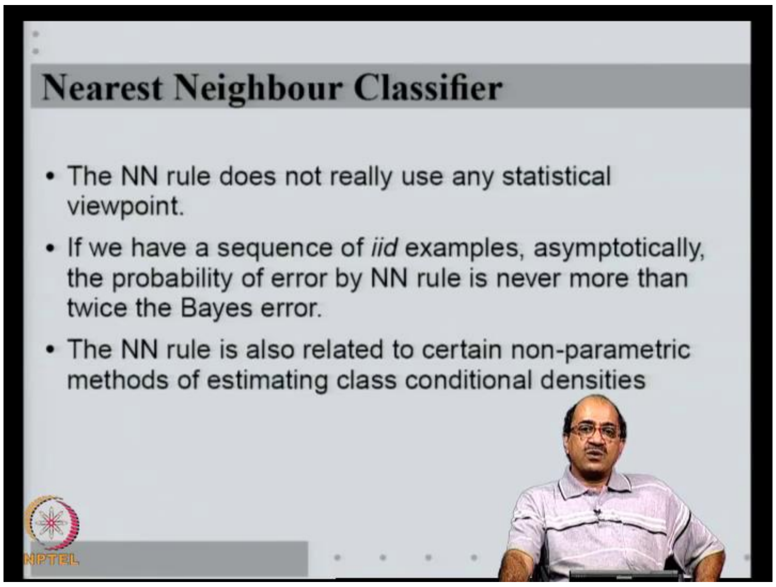
PR video course – p.58/128

Selection of prototypes, how do I select, how many to select, there are the few techniques we may come back and discuss it, a little while later. Essentially, we will like to make sure that, there are enough prototypes at all all parts of the feature space. So, wherever I get a new pattern, I have some prototype that is closest and of the same class to the new new pattern. What about the distance function, one can of course, choose the Eucledean distance because, all our feature vectors are real real vectors.

So, I can use Eucledean distance as the distance, the only problem with the Eucledean distance is, it is very sensitive to the actual numerical range. If different features take values in different ranges, the ones, which are numerically higher will dominate. Suppose, there are two features, one feature take values between 100 and 200, another feature takes values between 0.1 and 0.2, no matter how much a difference the 0.1 0.2 feature, the first feature always dominates.

So, we can always normalize with respect to variation of individual feature values, so I can choose additional functions where, the i th term in Eucledean distance is weighted by 1 by sigma i where, sigma i is the estimated variance of the i th feature. Given all the examples, I can find the estimated variance of the i th feature and I can use that so that, you know, I understand a difference between X i and X i prime, relative to it is standard deviation. A slightly better thing is to actually make this distance dependent not only just variances of individual features but, the entire covariance matrix. So, I can make the distance function as x minus X transpose X prime transpose sigma inverse X minus X prime where, sigma is the estimated covariance matrix and this is called the Mahalanobis distance, often used in economics.

(Refer Slide Time: 25:27)



Nearest Neighbour Classifier

- The NN rule does not really use any statistical viewpoint.
- If we have a sequence of *iid* examples, asymptotically, the probability of error by NN rule is never more than twice the Bayes error.
- The NN rule is also related to certain non-parametric methods of estimating class conditional densities

We will come back to this later on, what can we say about the nearest neighbor classifier, how does it perform, it of course, uses no statistical information. It is a very simple intuitive classifier but, one can show that, asymptotically that is, add the number of examples that the number of prototypes approaches infinity. If we assume that the prototypes are independently drawn then, the error made by nearest neighbor rule is never more than twice that of the Bayes error.

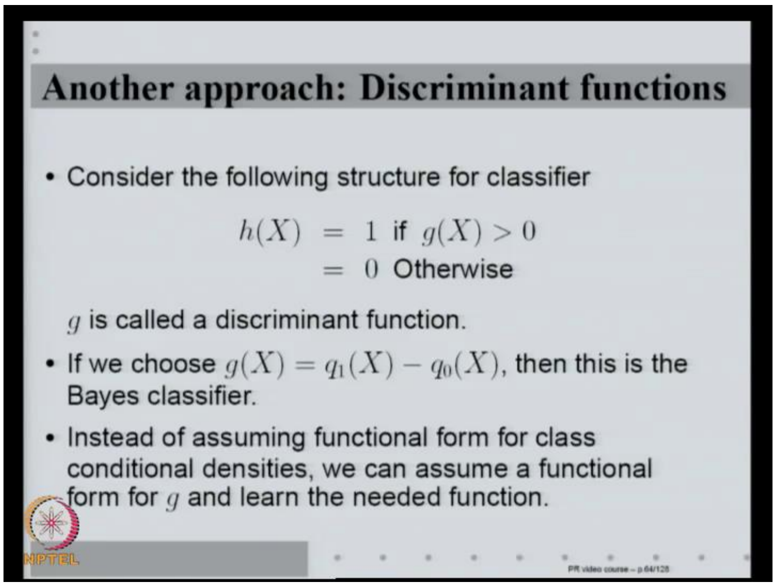
Where the probability of error by Bayes error is 0.1 then, nearest neighbor rule never makes more than 0.2 probability of error that is, if Bayes rule can be 90 percent correct, nearest neighbor rule is guaranteed to be at least 80 percent correct. So, this is quite interesting because, instead of, doing all the costly estimation implementing the Bayes classifier, I can simply implement the nearest neighbor classifier.

If with the given feature vectors, my Bayes rule can give me very good performance let us say, error by the Bayes classifier is only 5 percent then, at worst my nearest neighbor might give me 10 percent the error, which might be tolerable. But, nearest suppose, the pattern recognition problem is very complicated and Bayes rule can give you only 80 percent accuracy then, unfortunately nearest neighbor may give you at less than 60 percent accuracy.

But, that apart, this is very interesting issue that even though, we do not chose by statistic information, we can still bound the error of the nearest neighbor by twice the Bayes error. It is also related to certain, what are called non-parametric methods of estimating class conditional densities, we will discuss that later. By this point, I will simply mention that because of, it is simplicity and because of, it is some guarantees and performance, nearest neighbor classifier is often the the benchmark for any classifier design.

That is, I I should not go for a complicated classifier, unless I am at least getting sufficient improvement on nearest neighbor classifier. So, given any problem on the chosen feature vector, I should first implement the nearest neighbor classifier, that tells me the minimum performance I can get without doing any work. So, if I am designing a complicated involved classifier, there should be some reason so, nearest neighbor performance by the nearest neighbor classifier is often uses the bench mark to say, whether you know my my specially designed classifier is doing well or not, well Bayes and nearest neighbor classifiers are not the only classifiers.

(Refer Slide Time: 28:00)



Another approach: Discriminant functions

- Consider the following structure for classifier

$$\begin{aligned} h(X) &= 1 \text{ if } g(X) > 0 \\ &= 0 \text{ Otherwise} \end{aligned}$$

g is called a discriminant function.

- If we choose $g(X) = q_1(X) - q_0(X)$, then this is the Bayes classifier.
- Instead of assuming functional form for class conditional densities, we can assume a functional form for g and learn the needed function.

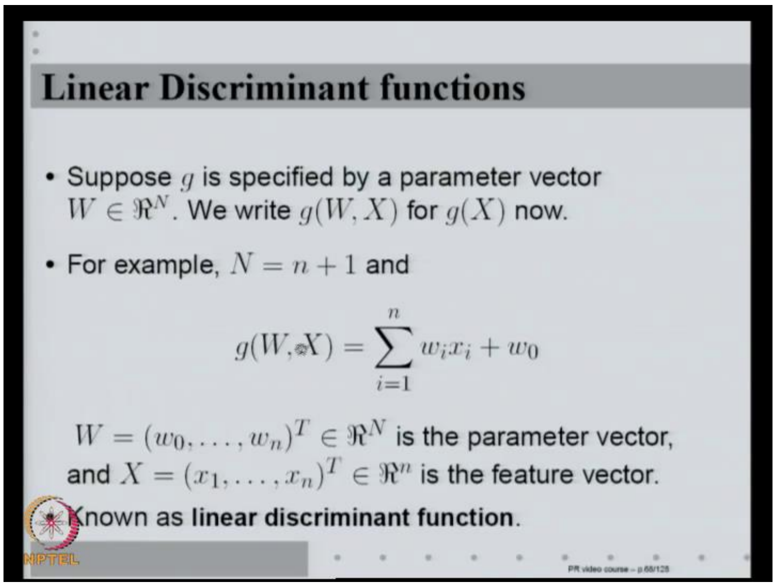
PR video course – p.64/128

There are many other approaches to classifier design, we will consider another approach in this introductory class, it is called the discriminant function based approach. So, the structure of this classifier is the following, as you know a classifier is a function, that maps every feature vector to 1 or 0, we are only considering two class case here. So, I choose a classifier structure as, h of X is 1, if some other function g of x is greater than 0, 0 otherwise.

That function g is called a discriminant function of course, I have to design the discriminant function and a classifier that uses this structure is called a discriminant function based classifier. Because, if I choose g of x to be q 1 x minus q 0 x then, this is the Bayes classifier, if q 1 x minus q 0 x is greater than 0 then, I put in class 1 otherwise, I put in class 0.

So, that is same as the Bayes classifier so, Bayes classifier is also a discriminant function Bayes classifier but, the discriminant function is obtained using the posterior probability. But, the interesting thing about the discriminant function classifiers is, I can choose any other discriminant function too right. The idea is, I will try to tweak a proper discriminant function instead of, worrying about actually estimating the class conditional densities. So, the idea is instead of, assuming some functional form for the class conditional densities and estimating them, we can assume the functional form for g and directly learn the needed classifier.

(Refer Slide Time: 29:14)



Linear Discriminant functions

- Suppose g is specified by a parameter vector $W \in \Re^N$. We write $g(W, X)$ for $g(X)$ now.
- For example, $N = n + 1$ and

$$g(W, X) = \sum_{i=1}^{n} w_i x_i + w_0$$

$W = (w_0, \ldots, w_n)^T \in \Re^N$ is the parameter vector, and $X = (x_1, \ldots, x_n)^T \in \Re^n$ is the feature vector.

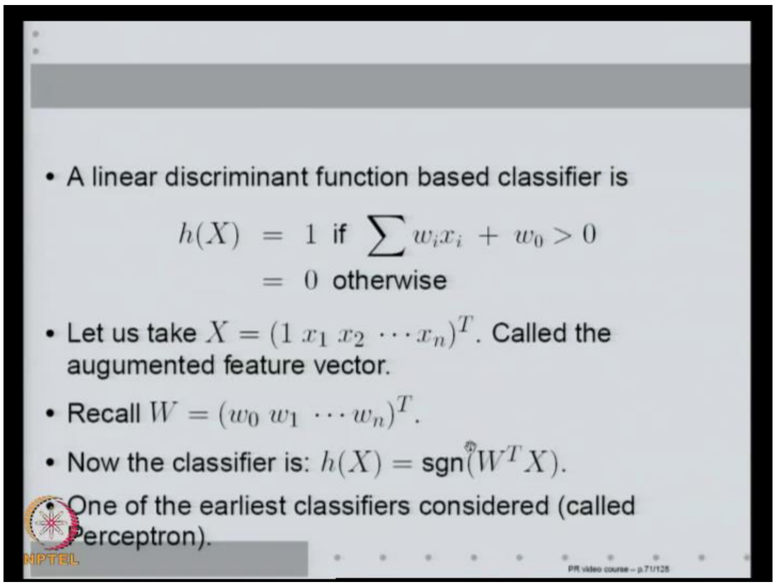
Known as **linear discriminant function**.

PR video course – p.68/128

Generally, a discriminant function would have some or would be specified by a some parameter so that, I can tweak the discriminant function. So, let us say, g W comma X instead of, g of X is the discriminant function where, W is the parameter vector and let us say, that capital N, number of parameters. So, here is one example discriminant function, I can take the number of parameters or the discriminant function to be 1 more than the diameter of the feature vector and the discriminant function is simply w i summation i is equal to 1 to n, w i x i plus W naught.

There is a weigh each feature component with some weight w i and then, add a threshold and if this sum is greater than 0, I have put in one class, less than 0, I have put in the other class. Where, w 0 to w n are the parameters, x 1 to x n are the features right such a such a classifier is called a linear discriminant function based classifier. This is linear because, it is linear in the components of the feature vector or in parameters, linear discriminant functions are another important class of classifiers.

(Refer Slide Time: 30:21)



- A linear discriminant function based classifier is

$$h(X) = 1 \text{ if } \sum w_i x_i + w_0 > 0$$
$$= 0 \text{ otherwise}$$

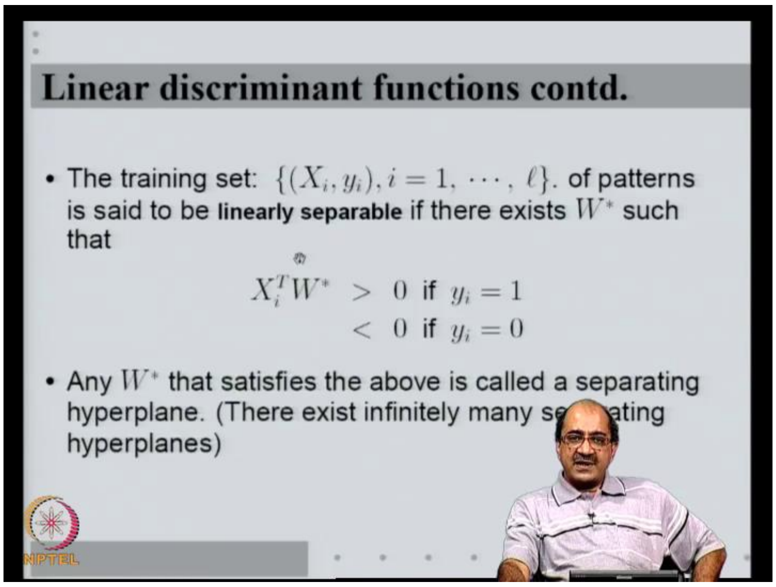
- Let us take $X = (1\ x_1\ x_2\ \cdots x_n)^T$. Called the augumented feature vector.
- Recall $W = (w_0\ w_1\ \cdots w_n)^T$.
- Now the classifier is: $h(X) = \text{sgn}(W^T X)$.
- One of the earliest classifiers considered (called Perceptron).

PR video course – p.71/128

Let just go a little bit deeper into it, a linear discriminant function based classifier, as we just seen in the previous slide is given by this. Very often, we may not want to see this, extra constant term may look, as if this is not really a linear function but, we can easily get rid of the extra constant term. We take a new feature vector, which is same as the old feature vector x 1, x 2, x n but, it has one extra component, which is permanently set to 1 right.

Take this as the new feature vector, is often called the augmented feature vector, if w 0 to w n is the parameter vector then, my classifier is nothing but, I take then, I can write this w 0 is w 0 into 1, which is same as w 0 into x 0. So, the entire thing is now, simply a dot product between two vectors so, I find w transpose x and the sign of the w transpose x is what, determines the classification. This kind of a classifier is one of the earliest classifiers design in 1950’s is called a Perceptron. We will come back to Perceptron later on and we will also find out, how one fixes w or how one learns w from the training examples.

(Refer Slide Time: 31:29)



Linear discriminant functions contd.

- The training set: $\{(X_i, y_i), i = 1, \cdots, \ell\}$. of patterns is said to be **linearly separable** if there exists W^* such that

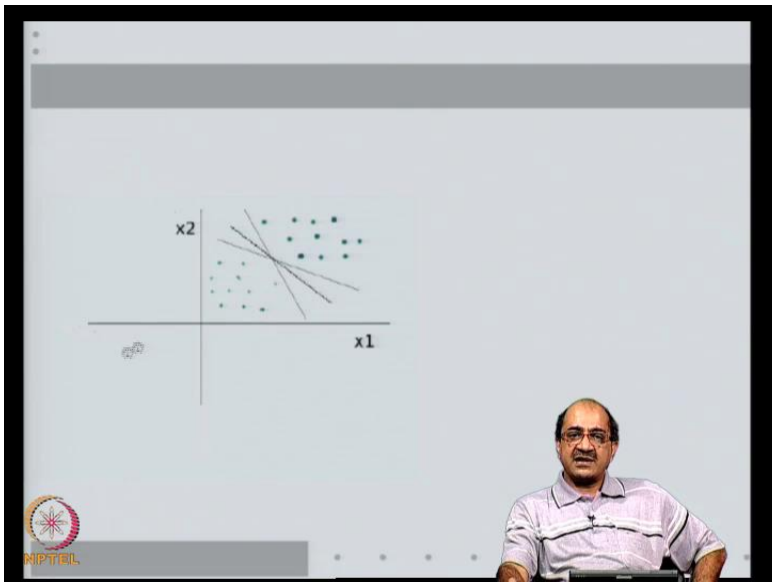
$$\begin{aligned} X_i^T W^* &> 0 \text{ if } y_i = 1 \\ &< 0 \text{ if } y_i = 0 \end{aligned}$$

- Any W^* that satisfies the above is called a separating hyperplane. (There exist infinitely many separating hyperplanes)

In considering linear discriminant function, there is one concept that is very important given a training set X i y i. So, that is set of pattern x i along with class labels y i we say, this training set is linearly separable, if there exists some w star. So that, x i transpose w star is greater than 0, if y i is 1 that is, X i comes from class 1, less than 0 if X i comes from class 0. Of course, the labels 0 and 1 are arbitrary, which I call class 0, and which I call 1 is arbitrary.

So, essentially, if there exists a w such that, X i transpose w star greater than zero means one class, less than 0 means another class. If such a w star exists then, we call the the set, a linearly separable set, any w star that satisfies this is called a separating hyperplane and it is easy to see, if there exists one separating hyperplane, there exists infinitely many, what does this mean.

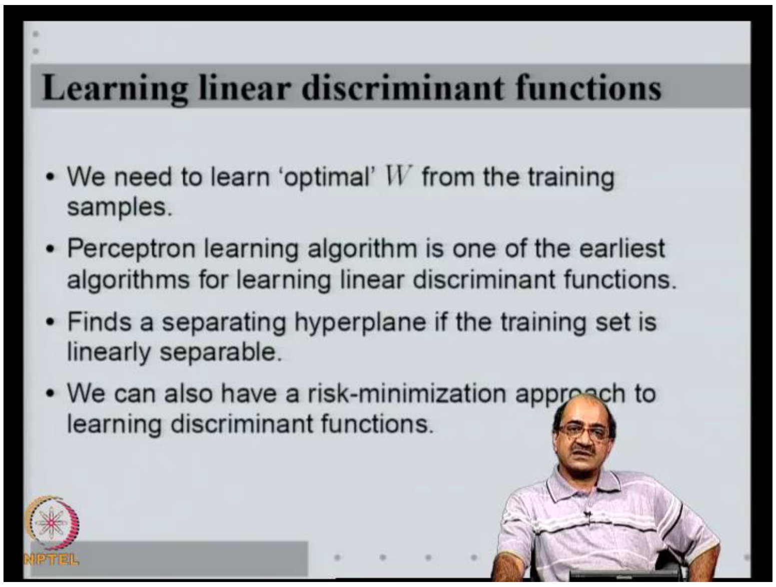
(Refer Slide Time: 32:35)



In a geometrically, a w can be, let us look at a figure so, basically what we have, is a line often the line passes through origin, even other than it is all right. So, w transpose X i plus a threshold greater than 0, one class less than 0 another class simply means, that in the features space between the two classes, I can put a line. Such that, all patterns of one class are on one side of the line, all patterns of the other class are on the other side of the line right.

Thus what this means, recall that, X is an augmented feature vector right so, essentially X transpose w is given by w i X i plus w naught so, there is the line and as you can see, there is one line, that can separate the two types of patterns of the two classes. Then, the I can draw many other lines, there will be actually as a matter of fact, a whole cone of lines and any line passing through this cone would be a separating hyperplane right. So, whenever a training set is linearly separable, there exists infinitely many separating hyperplanes so, a linear discriminant function will certainly work, if the training set or let us say, the pattern classes themselves are linearly separable.

(Refer Slide Time: 33:58)



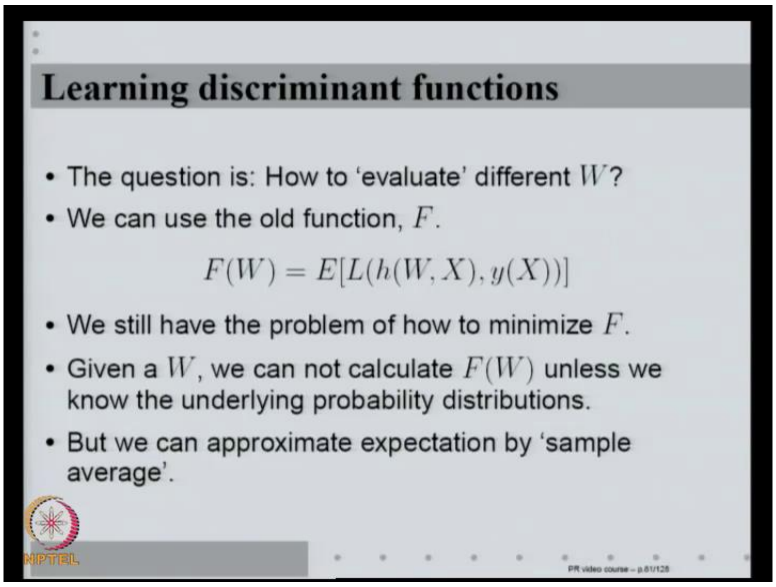
Learning linear discriminant functions

- We need to learn 'optimal' W from the training samples.
- Perceptron learning algorithm is one of the earliest algorithms for learning linear discriminant functions.
- Finds a separating hyperplane if the training set is linearly separable.
- We can also have a risk-minimization approach to learning discriminant functions.

Now, how do I learn linear discriminant functions, we need to learn the optimal w from the training samples, perceptron learning algorithm is one of the earliest algorithms for learning linear discriminant functions. It finds a separating hyperplane, if the training set is linearly separable but, the sad thing about this algorithm is that, if the given set is not linearly separable, the algorithm cannot come out and says is not linearly separable, it is just going to an infinite loop.

So, if I know, the if I suspect I have good confidence that a training set is linearly separable then, perceptron is say very good algorithm to use. We will we will study Perceptron algorithm, it is convergence proof and you know, many other interesting aspects of Perceptron later on. But, in general, when we we may or may not know, whether training set is really separable, we can also use the risk minimization approach to learning linear discriminant functions, how do I. What do you mean by you this risk minimization approach, just like what we did in the risk minimizing Bayes classifier, you choose a convenient loss function and then, define expectation as a loss function as risk and find a linear discriminant function that minimizes the risk right.

(Refer Slide Time: 35:07)



Learning discriminant functions

- The question is: How to 'evaluate' different W?
- We can use the old function, F.

$$F(W) = E[L(h(W, X), y(X))]$$

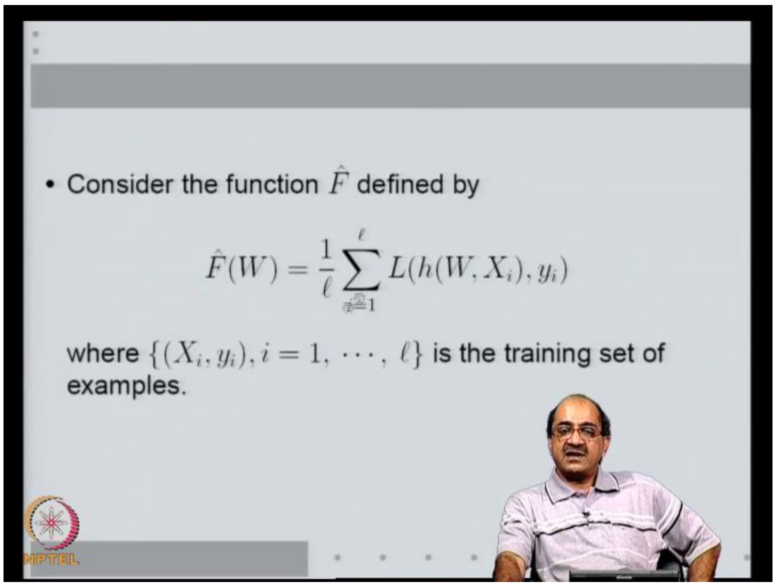
- We still have the problem of how to minimize F.
- Given a W, we can not calculate $F(W)$ unless we know the underlying probability distributions.
- But we can approximate expectation by 'sample average'.

PR video course – p.81/128

So, let us look a a little more into this so, the idea is, I want to minimize some criterion function so, to minimize what is the criterion function, the criterion function should evaluate different choices of discriminant functions and each choice of discriminant function is choice of a particular parameter vector W. So, ultimately, I am asking, can I write a criterion function, that can evaluate different W. We can always use the old function, if am using a linear discriminant function with w as the prime parameter vector let us say, h of w comma x as we given a layer is the classifier.

So, L of h of W comma X comma y x is the loss suffered on h on a pattern X, if you choose this parameter vector W so, take it is expectation and that is the risk for W. So, now, what you want to do is, to find a W that minimizes this expectation now, how do I minimize the F. Given a particular W, if I want to calculate F of W, I need to evaluate this expectation and to evaluate this expectation, we need the underlying probability distributions well. If have the underlying probability distributions of course, I can use the Bayes classifier so, what do I do, how do I evaluate how do I evaluate this. The idea is that, here F is expectation of some function and as you , if you want to find mean of some random variable, a good approximation is the sample mean. So, if I have training examples, I can find the sample mean of this same function on the training examples and that, I can use as an approximation to W approximation to this F W.

(Refer Slide Time: 36:43)



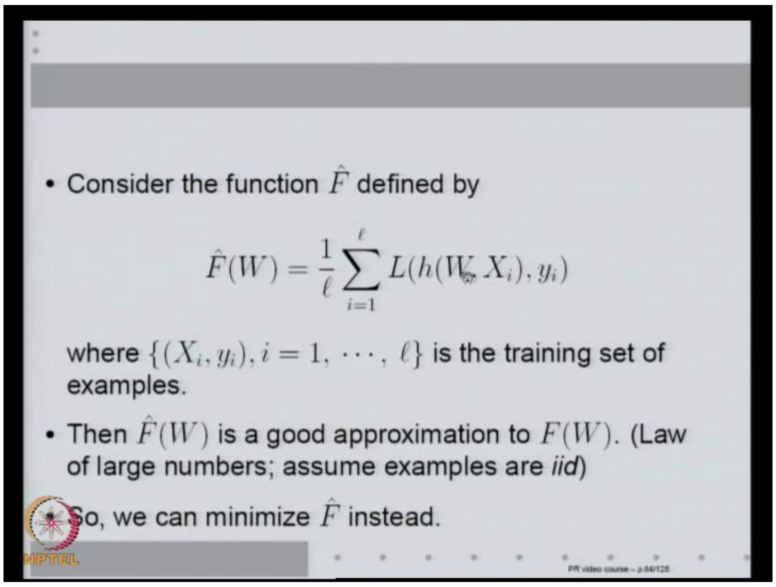
- Consider the function $\hat{F}$ defined by

$$\hat{F}(W) = \frac{1}{\ell}\sum_{i=1}^{\ell} L(h(W, X_i), y_i)$$

where $\{(X_i, y_i), i = 1, \cdots, \ell\}$ is the training set of examples.

So, let us call such a function, F hat of W so, F hat of W is L of h W comma X i comma y i summed over 1 to l and divided by 1 by l, as you can see F is expectation of this l, l of x so, to say. So, if I want sample mean, it is l of X i summed over i divided by the number of samples. That is what, F hat is right, I sum the loss over the samples X i y i and divide out the numbers of samples so, this is the sample mean estimated for F hat.

(Refer Slide Time: 37:22)



- Consider the function $\hat{F}$ defined by

$$\hat{F}(W) = \frac{1}{\ell}\sum_{i=1}^{\ell} L(h(W, X_i), y_i)$$

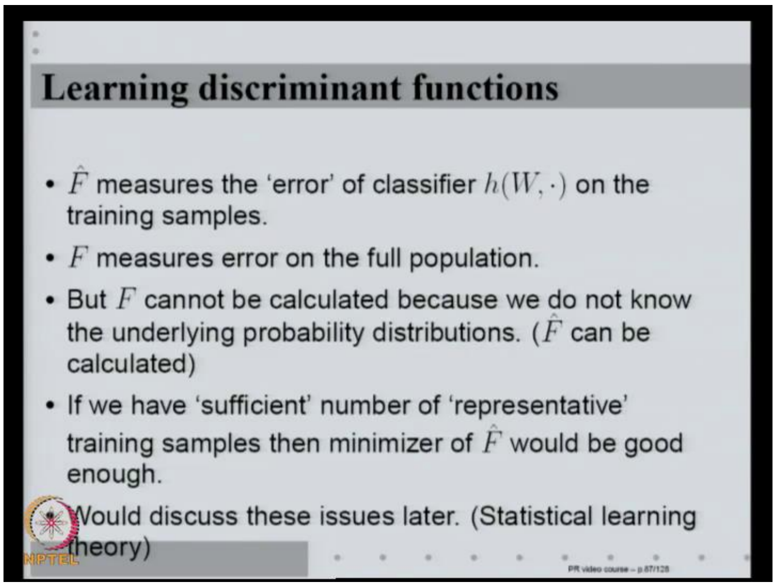
where $\{(X_i, y_i), i = 1, \cdots, \ell\}$ is the training set of examples.

- Then $\hat{F}(W)$ is a good approximation to $F(W)$. (Law of large numbers; assume examples are *iid*)
- So, we can minimize $\hat{F}$ instead.

PR video course – p.84/128

So, we would expect that , F hat is a good approximation to F, why a law of large numbers right, we know, if there is sufficiently many samples and samples are independent then, sample mean is a good approximation to the population. Sample mean as a matter of fact, convergence to the true expected value that is, the law of large numbers so, F hat is a good approximation to F. So, we can minimize F hat instead of, minimizing F because, F hat can be calculated given the training set, I can calculate F hat.

(Refer Slide Time: 37:50)



Learning discriminant functions

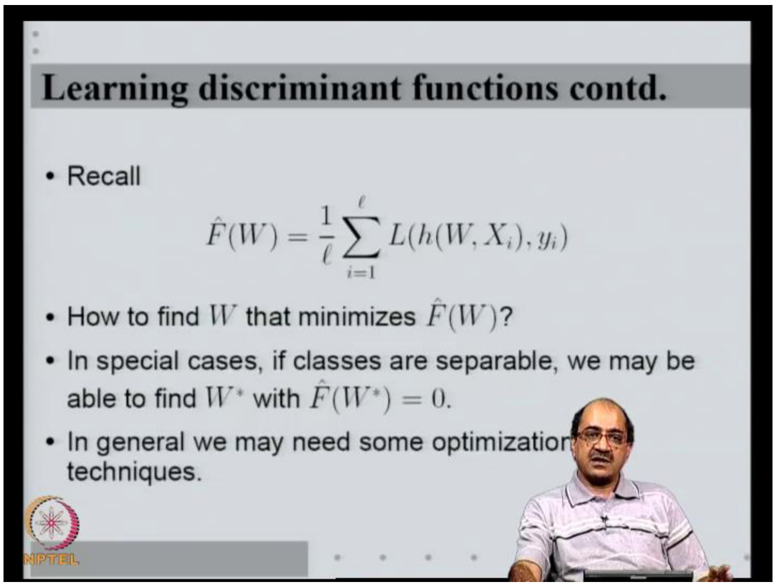
- $\hat{F}$ measures the 'error' of classifier $h(W,\cdot)$ on the training samples.
- F measures error on the full population.
- But F cannot be calculated because we do not know the underlying probability distributions. ($\hat{F}$ can be calculated)
- If we have 'sufficient' number of 'representative' training samples then minimizer of $\hat{F}$ would be good enough.
- Would discuss these issues later. (Statistical learning theory)

PR video course – p.67/128

So, let us let us understand this again, F hat measures the error of classifier on the training samples only because, it is a obtained as a sample mean. F of course, is obtained as a expectation so, it measures error on the full population so, what we are saying is that, why do we use F hat. We can calculate F because, we may not know the underlying probability distributions so, we are calculating F hat, which can be calculated.

So, our our gut feeling is that, if there are sufficiently many representative training samples, if the sample is large and they are independent then, the sample mean should be a good approximation to population mean. So, F hat would be a good approximation to F and hence, I am minimizing F hat should be good enough. Because, there are few deeper issues involved here, issues that concerned, what is called statistical learning theory, we will we will deal with that later on.

(Refer Slide Time: 38:41)



Learning discriminant functions contd.

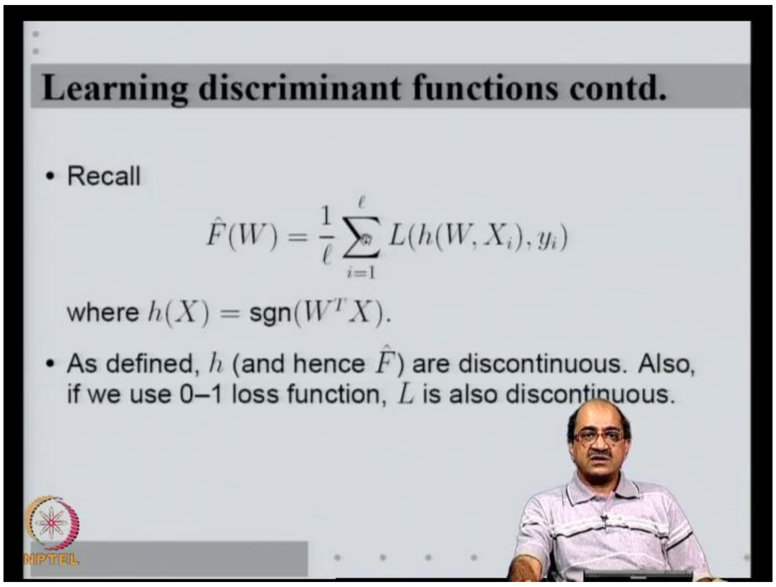
- Recall

$$\hat{F}(W) = \frac{1}{\ell}\sum_{i=1}^{\ell} L(h(W, X_i), y_i)$$

- How to find W that minimizes $\hat{F}(W)$?
- In special cases, if classes are separable, we may be able to find W^* with $\hat{F}(W^*) = 0$.
- In general we may need some optimization techniques.

So, this is what, we want to minimize to learn my linear discriminant function, how do I minimize this, there is still a problem. We can ask special algorithms such as Perceptron algorithm, which can find a W that such that, F hat W, 0 that will of course, be minimum. Because, we know loss function values are all positive so, for any given W, F hat W is always greater than or equal to 0. So, if I can find one W, for which F hat W is 0 then, that is the minimum so, there can be some special algorithms, which try to find that. But, in general, we have to use a standard optimization techniques, standard or otherwise some optimization technique now, we can always use an optimization technique.

(Refer Slide Time: 39:25)



Learning discriminant functions contd.

- Recall

$$\hat{F}(W) = \frac{1}{\ell}\sum_{i=1}^{\ell} L(h(W, X_i), y_i)$$

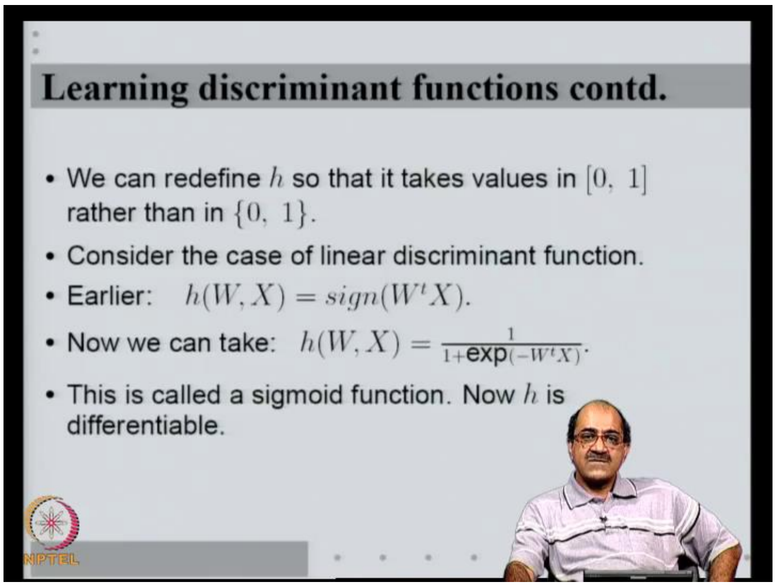
where $h(X) = \text{sgn}(W^T X)$.

- As defined, h (and hence $\hat{F}$) are discontinuous. Also, if we use 0–1 loss function, L is also discontinuous.

There is still a problem, this is F hat and let us recall that, the classifier h is the sign of W transpose X, if W transpose X is greater than 0 is one class, less than 0 another class which means, h is a discontinuous function right. And we still have not decided, what loss function to use but, if we use zero one loss function then, loss function is also discontinuous.

Now, irrespective of that loss function discontinuous or not, if h is discontinuous then, F hat is discontinuous, if L is discontinuous F hat is discontinuous which means, we cannot use any standard of optimizational algorithms for minimizing this. Because, we cannot even differentiate F hat well, we can cater on to this problem by changing both our h and L so that, they become differentiable.

(Refer Slide Time: 40:11)

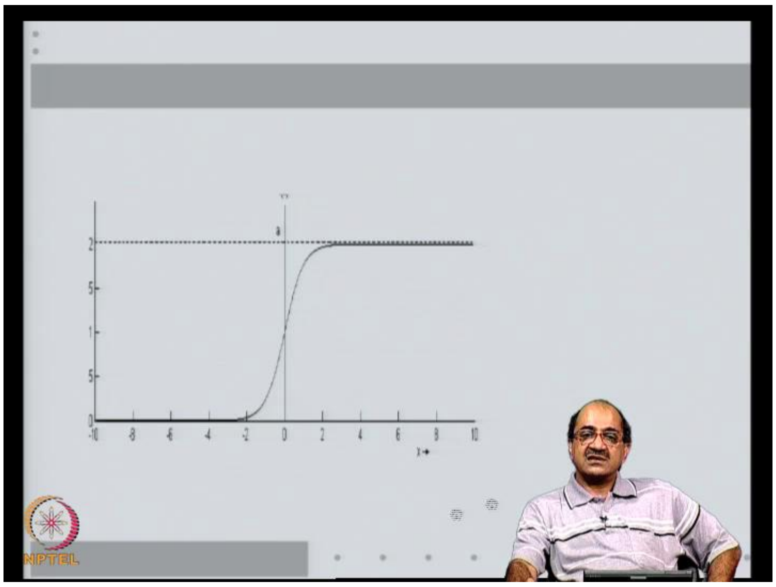


Learning discriminant functions contd.

- We can redefine h so that it takes values in $[0,\ 1]$ rather than in $\{0,\ 1\}$.
- Consider the case of linear discriminant function.
- Earlier: $h(W, X) = sign(W^t X)$.
- Now we can take: $h(W, X) = \frac{1}{1+\exp(-W^t X)}$.
- This is called a sigmoid function. Now h is differentiable.

We redefine h so that, instead of, taking value 0 on one only, we see h takes values in the interval 0 1 so, for example, earlier we said h of W X is the sign of W transpose X. Now, it is saying, sign of W transpose X we say, h W X is 1 by 1 plus exponential minus W transpose X.

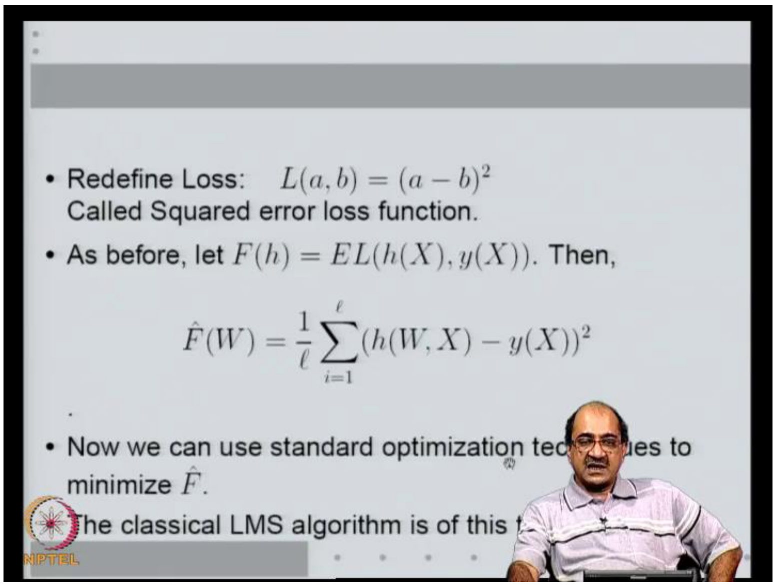
(Refer Slide Time: 40:37)



How does this function look, if I plot W transpose X on the x axis, 1 by 1 plus exponential minus W transpose X has this kind of form. As you can see, W transpose X is a little positive, it is close to 1, if there little negative is close to 0. But, the nice thing is, it is nicely differentiable as a matter of fact, in the definition of the function, we can add a small parameter inside the exponent and based on the value of that parameter, I can make this function as cheap as I want.

So, this is kind of a continuous approximation, with this tough function that the sign function. Now, when I use this h, if h is if the output of h is greater than 0.5, I can put in this class, less than 0.5, I can put in this class. So, by changing h, from sign of W transpose X to this, we have made h differentiable. Such a h is called a sigmoid function so, we we solved one problem namely non differentiability of h.

(Refer Slide Time: 41:44)



- Redefine Loss: $L(a, b) = (a - b)^2$
  Called Squared error loss function.
- As before, let $F(h) = EL(h(X), y(X))$. Then,

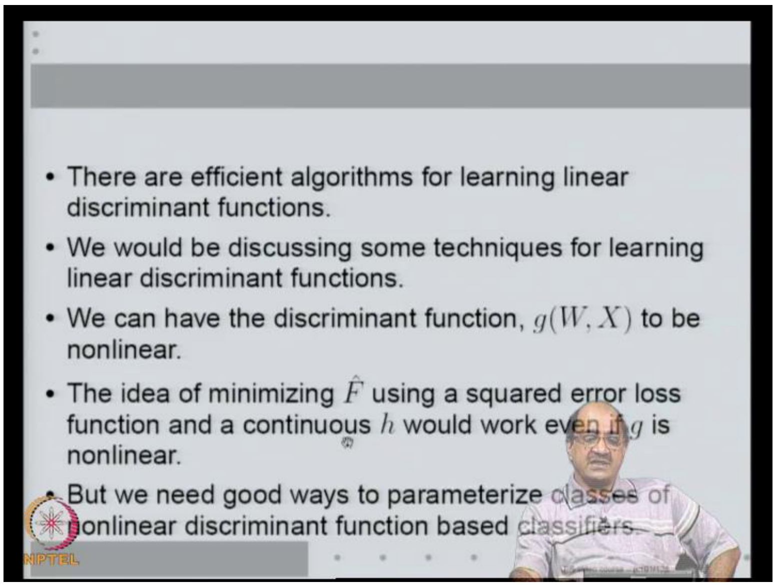
$$\hat{F}(W) = \frac{1}{\ell}\sum_{i=1}^{\ell}(h(W, X) - y(X))^2$$

.

- Now we can use standard optimization tec ies to minimize $\hat{F}$.
- The classical LMS algorithm is of this t

We can similarly, solve the non differentiability of L where, is to taking a zero one loss function, we can take a squared error loss function. So, we take L a b to be a minus b, which is called the square error loss function, which is also very popular in pattern recognition. Now, F hat W will be 1 by l summation h of W W comma X i minus y of X i whole square. So, this now because, h is differentiable and this square function is differentiable, one can differentiate so, for example, one can use gradient decent to optimize this right and such an algorithm is called the LMS algorithm, we will we will look at LMS algorithm later on.

(Refer Slide Time: 42:24)

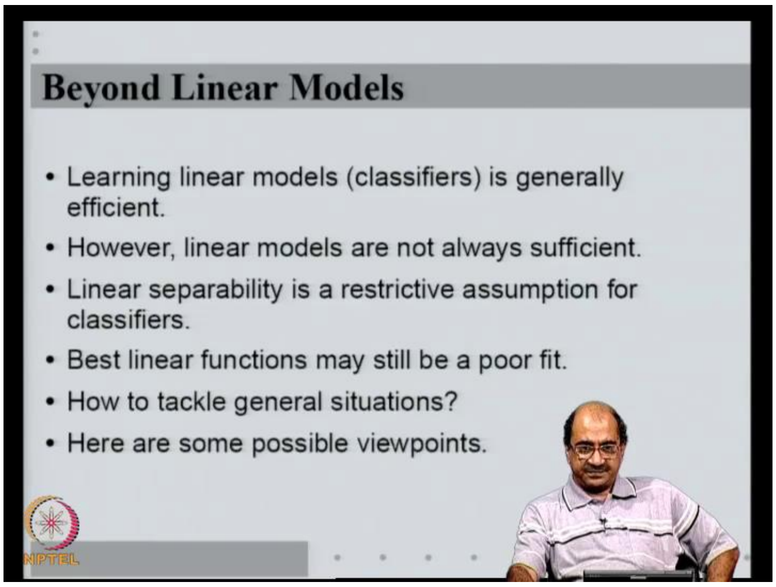


- There are efficient algorithms for learning linear discriminant functions.
- We would be discussing some techniques for learning linear discriminant functions.
- We can have the discriminant function, $g(W, X)$ to be nonlinear.
- The idea of minimizing $\hat{F}$ using a squared error loss function and a continuous h would work even if g is nonlinear.
- But we need good ways to parameterize classes of nonlinear discriminant function based classifiers.

So, there is a efficient algorithm for learning linear discriminant functions, we would be discussing some of this techniques for learning linear discriminant functions later in the course. So, essentially we have considered three approaches so far, the Bayes classifier, the nearest neighbor classifier, the linear discriminant function Bayes classifier well. There are few more but, let us just look at a summary of Bayes, the linear discriminant function based classifier.

Classifiers can take us, there are efficient algorithms for learning linear discriminant functions and essentially, a discriminant function based classifier is more general all right. We just said, it a depends on discriminant function so, you can also use non-linear discriminant functions, the idea of minimizing F hat using a squared error loss function, works even for non-linear discriminant functions.

(Refer Slide Time: 43:15)



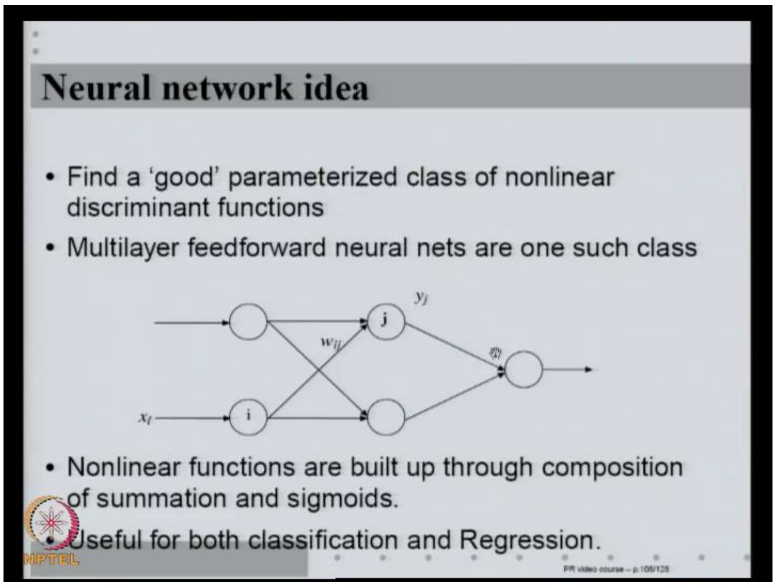
Beyond Linear Models

- Learning linear models (classifiers) is generally efficient.
- However, linear models are not always sufficient.
- Linear separability is a restrictive assumption for classifiers.
- Best linear functions may still be a poor fit.
- How to tackle general situations?
- Here are some possible viewpoints.

Now, but, non-linear is not so straight forward, the the method we got of minimizing F hat W to find linear discriminant function is a generally efficient method. So, in general, learning linear models is efficient but, a linear discriminant function may not always be sufficient. Linear separability may be restrictive condition even, if I minimize the risk, the risk of the best linear classifier may still be a poor fit right.

Now, if I want to learn non-linear discriminant functions, how do I go about it, in this course or even in the fields, there are there are there the few general approaches to learning non-linear classifiers. So, I will I will briefly tell you three approaches for learning non-linear classifiers, each of them to be viewed as a stylize read, this this is still an interacting lecture so, we are we we are just surveying the field. So, at this level of detail I will I will just give you three possible viewpoints, on how one tackles non-linear functions, non-linear discriminant functions.

(Refer Slide Time: 44:33)



Neural network idea

- Find a 'good' parameterized class of nonlinear discriminant functions
- Multilayer feedforward neural nets are one such class

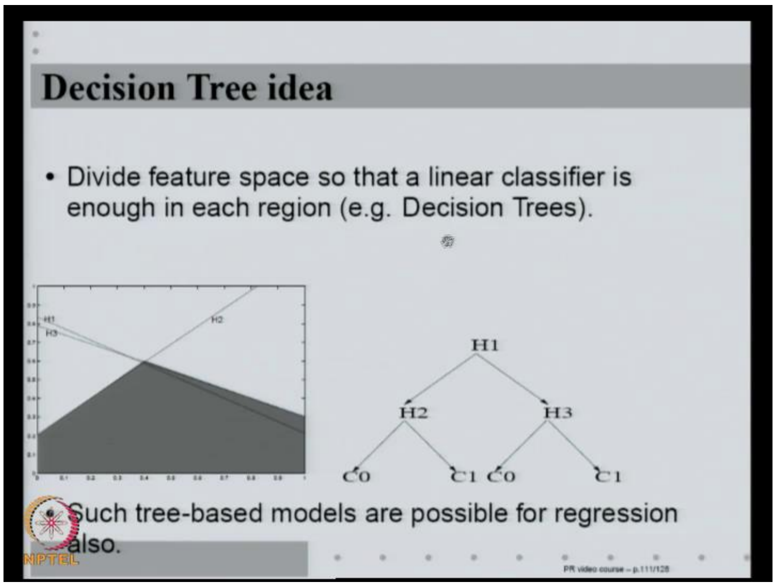
- Nonlinear functions are built up through composition of summation and sigmoids.
- Useful for both classification and Regression.

PR video course – p.108/128

One is what, can be calling neural network idea see, a linear discriminant function I know how to parameterize, W transpose X plus W naught. So, in a similar way, I have to find a good parameterized class of non-linear discriminant functions, one good class of such non-linear discriminant functions is, what goes in the name multilayer feed forward neural networks.

The basic idea is the following, each of these circles here, are essentially linear discriminant functions so, I give my give the pattern to many linear discriminant functions and I keep taking the outputs of linear discriminant functions and combine them to through linear discriminant functions. So, it is a kind of function composition approach to finding non-linear functions using non-linear functions, are built up through composition of simple linear functions on those sigmoid functions, that we talked about. Multilayer feed forward networks are a particular class like that and it is useful for learning non-linear classifiers and in this course, we will be considering this as one approach to non-linear classification.

(Refer Slide Time: 45:39)



Decision Tree idea

- Divide feature space so that a linear classifier is enough in each region (e.g. Decision Trees).

H1
H3
H2
H1
H2
H3
C0
C1
C0
C1

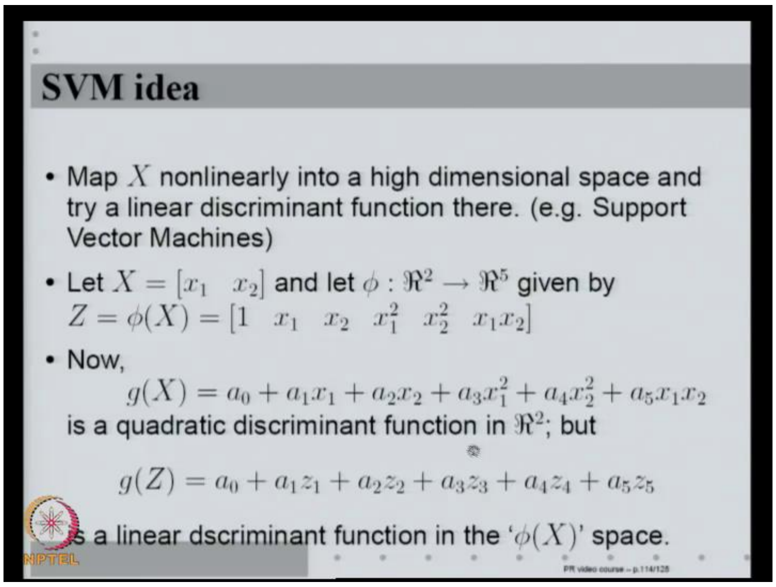
- Such tree-based models are possible for regression also.

PR video course – p.111/128

And another idea is, if the linear discriminant function does not work in the entire space, I can ask, can I cut the feature space in a different regions so that, within each region a linear linear classifier would work, here is an example right. So, the the shaded region is one class, the other region is another class so, a single line cannot separate one class from other but, if I can put two lines so that, for example, in one half of the feature space I can use one line, in the other half of the feature space I can use the other line.

So, one way of doing this is that, I first cut the feature space with respect to one line let us say, I cut it with respect to a line like that, that is what I call H 1 then, I am asking that the feature vector fall in this side or that side. Depending on which side it falls, I can once again say, I I can use two different linear discriminant functions to classify. So, instead of classifying the pattern in one step, I will use multiple linear discriminant functions right. This is like cutting the feature space into different regions and within the each region, using a linear discriminant function. Such tree based models are possible for both classification as well as function learning right, time permitting to us end of this course, we will look at a few base of learning such tree base classifiers.

(Refer Slide Time: 47:08)



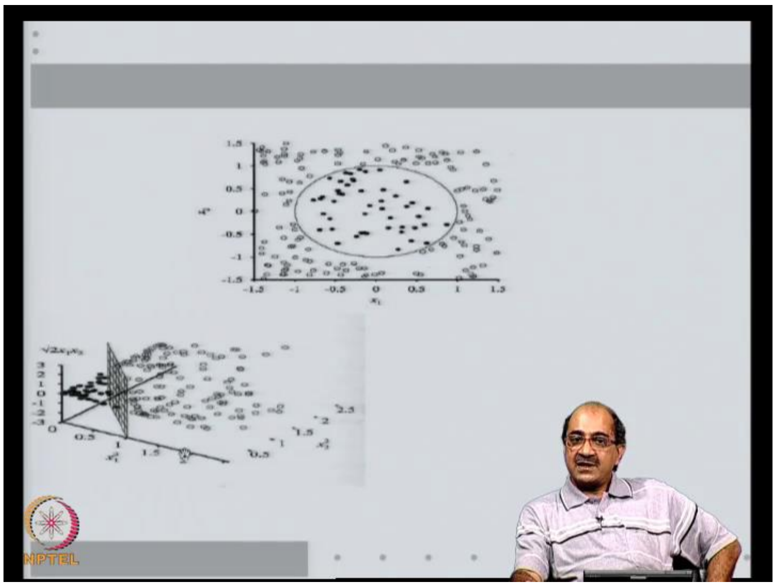
SVM idea

- Map X nonlinearly into a high dimensional space and try a linear discriminant function there. (e.g. Support Vector Machines)
- Let $X = [x_1 \quad x_2]$ and let $\phi : \Re^2 \rightarrow \Re^5$ given by $Z = \phi(X) = [1 \quad x_1 \quad x_2 \quad x_1^2 \quad x_2^2 \quad x_1x_2]$
- Now,
$$g(X) = a_0 + a_1x_1 + a_2x_2 + a_3x_1^2 + a_4x_2^2 + a_5x_1x_2$$
is a quadratic discriminant function in $\Re^2$; but
$$g(Z) = a_0 + a_1z_1 + a_2z_2 + a_3z_3 + a_4z_4 + a_5z_5$$
is a linear dscriminant function in the '$\phi(X)$' space.

PR video course – p.114/128

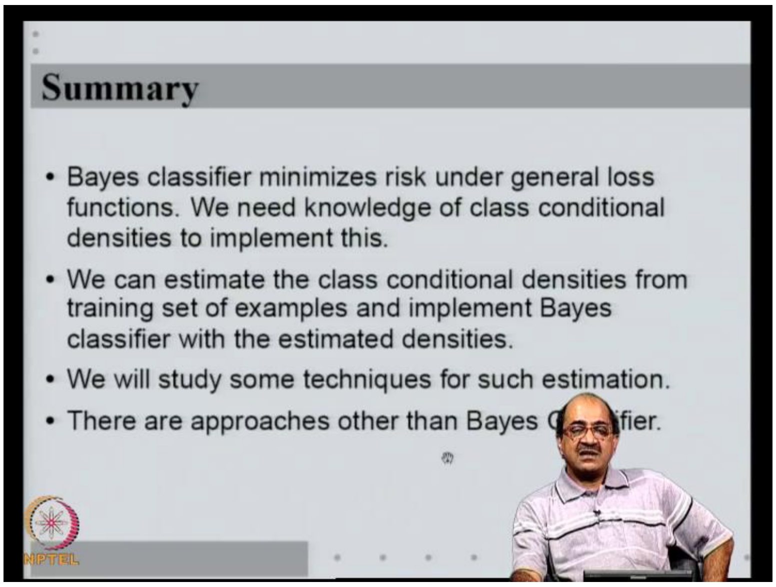
What is the third approach? A third approach is what may be called an SVM approach, SVM stands for support vector machines, this is today possibly the best way to learn non-linear classifiers. The idea is very simple, you map X, the feature vector non-linearly into high dimensional space and then, learn a linear discriminant function there. What does that mean suppose, the feature vectors are two components x 1, x 2 suppose, there is a function phi, that maps R 2 to R 5 given by Z, is phi of X is a is now a, I am sorry phi should have been six, it is a six component vector given by 1, x 1, x 2, x 1 square, x 2 square, x 1 x 2. Now, if I think of a linear discriminant function a 0 plus a 1 x 1 plus, a think of a non-linear discriminant function a 0 plus a 1 x 1 plus a 2 x 2 plus a 3 x 1 square plus a 4 x 2 square plus a 5 x 1 x 2. That is, the linear discriminant function in the Z space right, in the in terms of components of Z, this is linear right.

(Refer Slide Time: 48:13)



What does that mean say, for example, in the original feature space x y space, everything inside the circle is one class, outside the circle is another class. Now, this essentially non-linear because, no linear function in x can separate the two classes but, if I change my axis so that, I have x 1 square, x 2 square, x 1 x 2 as my axis. Then, in the new space, all the points inside the circle will come on one side, all points outside the circle will come on the other side and a hyperplane can separate them right. So, this is the basic idea of SVM you you you map X non-linearly into a high dimensional space and try a linear discriminant function there right. That is the basic of SVM idea right so, there is a these are the three ideas, that we use to learn non-linear classifiers right.

(Refer Slide Time: 49:11)

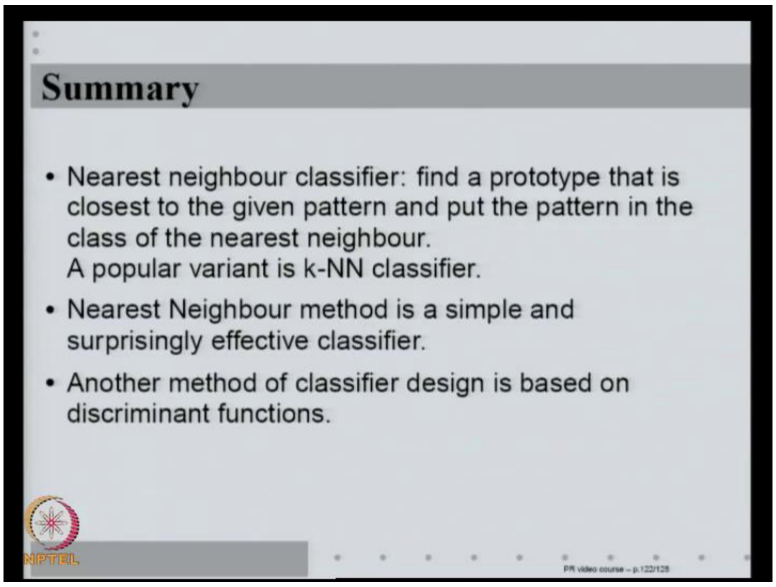


Summary

- Bayes classifier minimizes risk under general loss functions. We need knowledge of class conditional densities to implement this.
- We can estimate the class conditional densities from training set of examples and implement Bayes classifier with the estimated densities.
- We will study some techniques for such estimation.
- There are approaches other than Bayes C[illegible]fier.

So, let us summarize our discussion so far, the Bayes classifier minimizes risk under any general loss function, given the knowledge of class conditional densities no other classifier can do better than this. We have seen the optimality of Bayes classifier in the simple case of minimizing probability of error, we will also prove it under minimizing general disc function but any case, Bayes classifier minimize risk under any general loss function. It is main problem is that, the it it needs the knowledge of class conditional densities, how do I get class conditional densities to implement Bayes classifier, I can estimate the class conditional densities from the training set of examples.

(Refer Slide Time: 50:21)



Summary

- Nearest neighbour classifier: find a prototype that is closest to the given pattern and put the pattern in the class of the nearest neighbour.
  A popular variant is k-NN classifier.
- Nearest Neighbour method is a simple and surprisingly effective classifier.
- Another method of classifier design is based on discriminant functions.

PR video course – p.122/128

There are various estimation techniques, we will look at each of them in this course and see see,, how we can estimate the class conditional densities, what are the errors in estimation and so on, that is one one kind of approaches to classifier result. But, as you said, there approaches other than the Bayes classifier, nearest neighbor classifier is another approach to classifier design, nearest neighbor classifier is very simple.

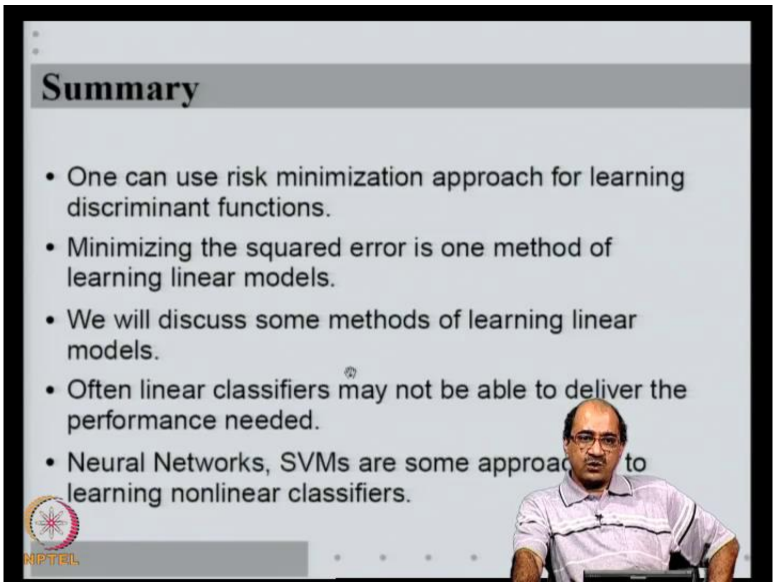
It simply says, find a prototype that is closest to the given pattern and put the pattern in the same class at the nearest neighbor, a variant is k-NN instead of finding one closest neighbor, you find k closest neighbours and put the classifier in the in the in the class of the majority class. So, either nearest neighbor k-NN is the very simple classifier to implement and as we have seen, it say it has surprisingly effective classifier.

In in many situations, nearest neighbor classifier gives to sufficient performance and you do not need to go for any more advanced classifiers and in any case, even on a many complicated problem a nearest neighbor classifier is always implemented. Because, that gives you some base line performance that you can always rely on right so, nearest neighbor classifiers is one other approach. And the issues there are how do I chose the prototypes, how do I decide on the distance metric and so on.

But, given certain choices for these issues, a nearest neighbor method is the simple and very effective classifier. When we consider certain techniques density estimation, we will come back to the nearest neighbor classifier, explore it is relationship with Bayes classifier and and revisit the issue of, how to bound the probability of the nearest neighbor classifier by the Bayes classifier error.

Another method for classifier design is based on, what are called discriminant functions, what are the discriminant function based classifier do, it uses a function g and the classifier simply say, h of x is 1 if the g of x is greater than 0, is 0 otherwise. And such a g called a discriminant function and g may have a parameter vector and now, learning a classifier, is essentially learning the parameter vector of the discriminant function. We have seen one example namely, linear discriminant functions, as one example of discriminant functions.

(Refer Slide Time: 52:37)



Summary

- One can use risk minimization approach for learning discriminant functions.
- Minimizing the squared error is one method of learning linear models.
- We will discuss some methods of learning linear models.
- Often linear classifiers may not be able to deliver the performance needed.
- Neural Networks, SVMs are some approaches to learning nonlinear classifiers.

I have also said that, we can use this same risk minimization approach, with which we have shown the optimality of Bayes classifier, for learning linear discriminant functions also. So, even though to to use risk minimization approach, we may not be able to use zero one loss function, we have to use squared error loss function. Learning discriminant functions using squared error loss function and minimizing the risk, is a very very standard approach to learning linear models that is why, many linear model learning linear models go under linear least squares learning approaches.

So, both for learning classifiers as well as regression functions, we often use a linear model, use a squared error loss function and minimize risk. Of course, we cannot calculate risk as I said because, we do not have the underlying probability distributions. But, we can always approximate the expectation by the sample mean and hence, obtain the affract function, which is often called the empirical risk function, which is nothing but, the sample mean of the expectation. And because, this sample mean can be evaluated on the on the samples, we can always minimize the sample mean.

So, minimize the squared error is one method of learning linear models, we will we will discuss some methods of learning linear models. Specifically, we look at Perceptron algorithm, we will look at LMS algorithm, we will look at what is called logistic regression and so on. So, we will discuss some methods for learning linear models, what is also good to know at this point is that, learning many of the non-linear models can also utilize the same approach.

Risk minimization approach that is, we chose a suitable loss function, very often squared error loss function and minimize the the affract function that is the empirical risk to find the best parameter vector. So, both for learning linear models and also for learning some of the non-linear models, we will often use minimizing risk, using a suitable risk function. As we said, the linear classifiers may not be able to deliver the performance needed and we have seen at least three approaches.

We will certainly explore both the neural network and SVM approach for learning non-linear classifiers, both of them have some specialized algorithms as well as risk minimization algorithms so, we look at them also as risk minimization methods. So, this kind of completes the overview of the course, from next next lecture, we will start again at the Bayes classifier and go into the details of design of Bayes classifier.

Thank you.