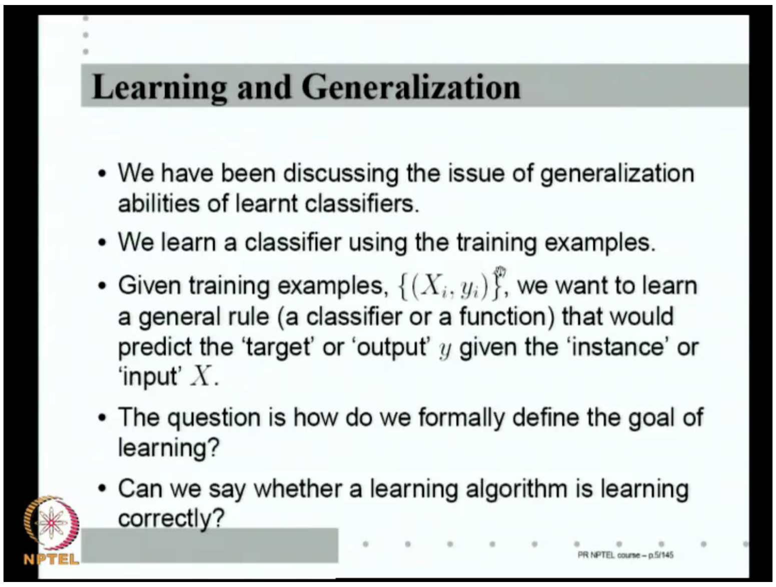
Pattern Recognition

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

Lecture - 20
Learning and Generalization; PAC learning framework

Welcome to this next lecture in the course on pattern recognition. Last class, we have been, we have just started discussing the theoretical issues of how does one address the generalization abilities of a learning algorithm. In general, how does one talk about learning from examples has been done correctly, is there a notion of correctness of a learning algorithm?

(Refer Slide Time: 01:01)



Learning and Generalization

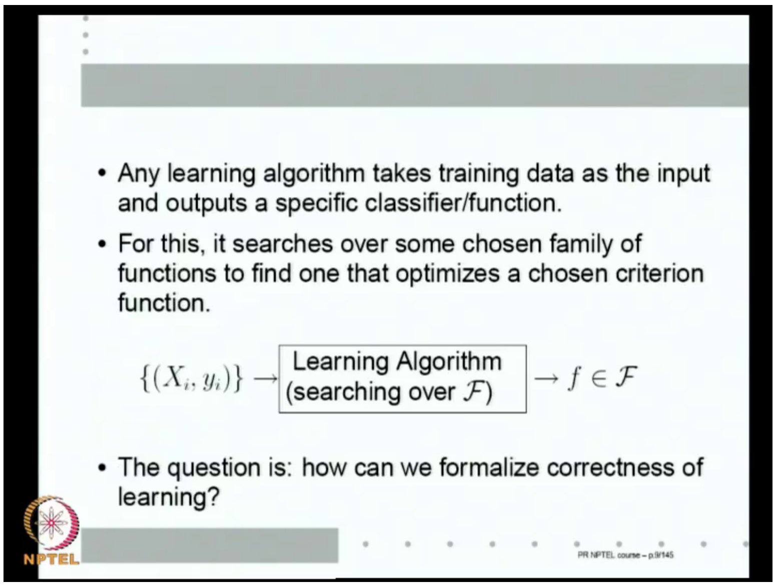
- We have been discussing the issue of generalization abilities of learnt classifiers.
- We learn a classifier using the training examples.
- Given training examples, $\{(X_i, y_i)\}_1^n$, we want to learn a general rule (a classifier or a function) that would predict the 'target' or 'output' y given the 'instance' or 'input' X.
- The question is how do we formally define the goal of learning?
- Can we say whether a learning algorithm is learning correctly?

PR NPTEL course – p.5/145

So, this class, we look at these issues in a formal sense and introduced first the concept of what is called probably approximately correct learning. So, let us start with what we said last time about learning and generalization. So, we have been discussing this issue of learning and generalization of classifiers. Of course, the, the issue is same for classifiers as well as learning regression functions. Basically, we learn a classifier or a function using training examples.

So, given training examples X i and y i, the idea is to learn a general rule call it a classifier, call it a function that would predict y which could be called the target or the output, given the input that is X, X can be called instance the input. So, given a new instance of the pattern or input for a function, you want to predict the target output of the class label. So, this is the general learning example, given the training examples I want to learn such a rule.

(Refer Slide Time: 02:08)



- Any learning algorithm takes training data as the input and outputs a specific classifier/function.
- For this, it searches over some chosen family of functions to find one that optimizes a chosen criterion function.

$$\{(X_i, y_i)\} \rightarrow \boxed{\begin{array}{l}\text{Learning Algorithm}\\ \text{(searching over } \mathcal{F}\text{)}\end{array}} \rightarrow f \in \mathcal{F}$$

- The question is: how can we formalize correctness of learning?

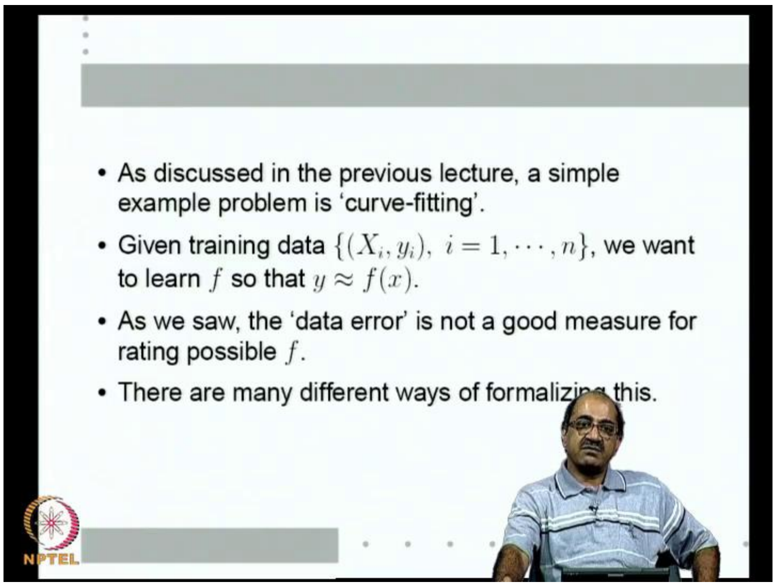
PR NPTEL course – p.9/145

So, the question we want to address in a next few classes is how does one formally define the goal of learning? Is there a notion that a learning algorithm is correctly learning or incorrectly learning? Any learning algorithm essentially takes training data as the input and outputs a classifier or a function. So, a learning algorithm can be thought of as a black box whose input is a set of training samples whose output is a function a classifier or a regression function, but essentially the output is a function.

As we have seen so far for this what is the algorithm do it searches over some chosen family of functions and finds one that optimizes some chosen criterion function. This is the algorithm chosen for example, all our linear classification learning algorithms for example, say linear least squares it searches over all functions of the form w transpose X plus b f X is equal to w transpose X plus b, over this class of functions it searches to find a function that minimizes the, so called sum of squared errors on the training examples.

So, essentially any learning algorithm searches over some chosen family of functions and outputs a function based on some criterion function that is optimized. So, the criterion function of course, is arbitrary and we want to know whether this is the way of defining. And if you chose a different criterion function will the will the correctness of learning change, but such as it is. So, we want to for the next few classes only look at learning algorithms like this. We will ultimately try that all up with risk minimization that we have considered earlier. But for now let us say a learning algorithm which searches over some family of functions script f, it takes is as input the training data and outputs some function.

(Refer Slide Time: 04:12)

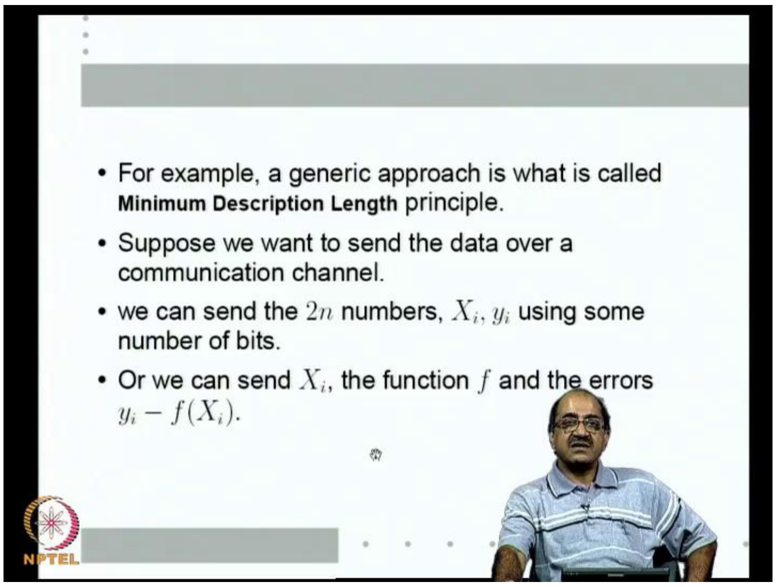


- As discussed in the previous lecture, a simple example problem is 'curve-fitting'.
- Given training data $\{(X_i, y_i),\ i = 1, \cdots, n\}$, we want to learn f so that $y \approx f(x)$.
- As we saw, the 'data error' is not a good measure for rating possible f.
- There are many different ways of formalizing this.

Now, we want to say how do i formally define correctness of learning that is the, that is the task for this class. And we are going to do it to start with in a in a simpler setting. Now, before we get into the actual formalism I like to first make you appreciate that there can be many different formalism simply only looking at one. So, we will just look at the problem in a bit generality to start with. So, as we discussed last time a simple example is curve fitting given points X i y i on real line, you want to fit a curve y is equal to f x. This is of course, least squares is the only algorithm we done so far. So, in that sense everybody knows least square curve fitting, so one simple function learning problem is curve fitting.

So, given training data X i y i we want to learn an f such that y can be approximated with f of x. Now, we already saw both while we were doing the least squares algorithm and hence regularized least squares. And also in the last class that the data error meaning the error between f of X i and y i that is if I take a particular function f the error on the training data which is f X i minus y i by that itself is not a good measure of how good a function f is we have seen. For example, if I got n points and if it a nth degree polynomial I can get 0 data error, but that does not mean that nth degree polynomial is the best fit.

(Refer Slide Time: 06:18)



- For example, a generic approach is what is called **Minimum Description Length** principle.
- Suppose we want to send the data over a communication channel.
- we can send the $2n$ numbers, X_i, y_i using some number of bits.
- Or we can send X_i, the function f and the errors $y_i - f(X_i)$.

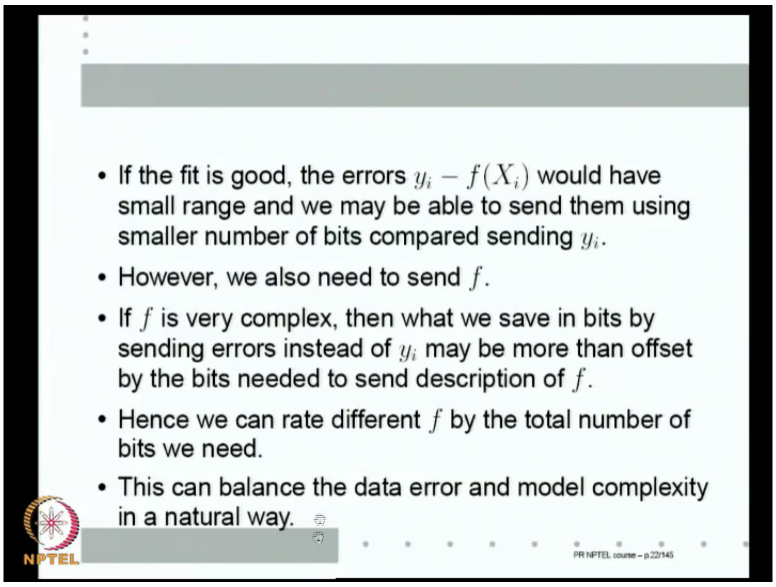
Now, this is the basic notion of how do we decide? Are we learning correctly? Do we have enough examples for the particular learning problem that we have? Is this learning problem more complex than some other learning problem and so on so forth? But the basic I all this comes because just minimizing data error is not good enough, because that is not, what we are interested in we are actually interested in how you perform on new unseen data. As I said there are many different ways of formalizing this. And what we will do is before we get to this specific formalism that we will do in detail. I will just look at one other thing very intuitively just to give you an idea that there are different ways of addressing this question.

So, one very interesting generic approach is, what is called minimum description length principle called MDL, what is minimum description length principle? Here we do not worry about whether we are generalizing well or not we simply say I have got training data X i y i I am fitting a function f so that I want so say y hat y i hat is f of X i I am only bother about the training data. Even then can I say whether the function is a good fit for the given data or not.

So for that what we do is we pretend that what we actually want to do is to send the X i y i over some communication channel. Now, we want to send X i y i over some communication channel, what can we do? We have to send 2 n numbers using some number of bits. We can, we can arbitrarily fix some precision in the numbers X i y i they want to send that will define the number of bits. Of course, it depends on the range of values y I you have, but X i and y i have, but such as it is for a given level of precision we can fix some number of bits that we need to send.

So, one way we can send these numbers for a channel is simply send the 2 n numbers X i and y i, because this is straight forward thing or what we can do is we can send all the X i's then we send a fitted function f and then instead of sending y i's we send y i minus f X i. So, we first send the function f and instead of sending X i and y i we send X i and y i minus f X i. For example, in the best case suppose function exactly fits then we do not have to send 0 we, we will we need only say one bit for learning 0.

(Refer Slide Time: 08:28)



- If the fit is good, the errors $y_i - f(X_i)$ would have small range and we may be able to send them using smaller number of bits compared sending y_i.
- However, we also need to send f.
- If f is very complex, then what we save in bits by sending errors instead of y_i may be more than offset by the bits needed to send description of f.
- Hence we can rate different f by the total number of bits we need.
- This can balance the data error and model complexity in a natural way.

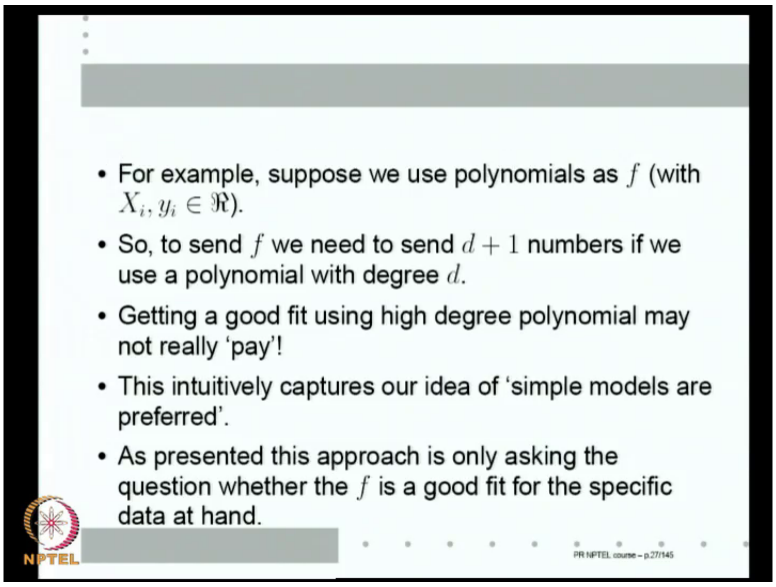
PR NPTEL course – p.22/145

So, we are sending only X i we are saving all the bits that we would have used for sending y i, but of course, we do not know how many bits we have we have already used up in sending f. So, one way we can send X i y i, otherwise we can send X i the function f and the errors y i minus f X i. Why would this be an interesting idea? If the fit is good y i minus f X i would have small range, because it was small range we will be able to send them using smaller number of bits compared to sending all y i, you know that that is straight forward.

Of course, this saving in bits is not coming free, we can send y i minus f X i only if we already send f, because at the receiver given X i if I want to create y i. I have to first calculate f X i and add it to y i minus f X I, so that I can get y I. So, while we may save lot of bits in sending y i minus f X i we have to also consider the cost of sending f. So, if f is very complex then what we save in bits by sending the errors instead of y I, it may not offset, it is more it may be it may be offset by the bits that need to send the description f.

So if I can fit a simple function which is a good fit then sending f would not cost too much in terms of bits And I am saving lot of bits in sending y i minus f X instead of y I, on the other hand if f is complex then even if the fit was good and I am saving bits by sending y i minus f X i I am losing bits in sending f. So, one way I can rate different f is to say what is the total number of bits that I need to send the data using different f? The f that allows me to send the data with the least number of bits could be called the best f that is what is that is why it is called the minimum description length principle we want to describe the data using least number of bits.

(Refer Slide Time: 10:42)



- For example, suppose we use polynomials as f (with $X_i, y_i \in \Re$).
- So, to send f we need to send $d+1$ numbers if we use a polynomial with degree d.
- Getting a good fit using high degree polynomial may not really 'pay'!
- This intuitively captures our idea of 'simple models are preferred'.
- As presented this approach is only asking the question whether the f is a good fit for the specific data at hand.

PR NPTEL course – p.27/145

So, either we can describe by simply giving X i and y i or describe the function f and then X i and y i minus f X i. So, this kind of balances the data error and model complexity in natural way as you seen with the complex model, you may get low data error, but the, because of the model complexity that may not be the choice. We looked at the regularized least squares as an example for this, so MDL, the minimum description principle is another way to balance the data on model complexity. To be a little more specific, suppose we are fitting polynomial functions X i and y i and on real line. So, we are fitting y as a equal to f X is a polynomial function. So, if I am fitting a dth degree polynomial it has d plus 1 coefficients.

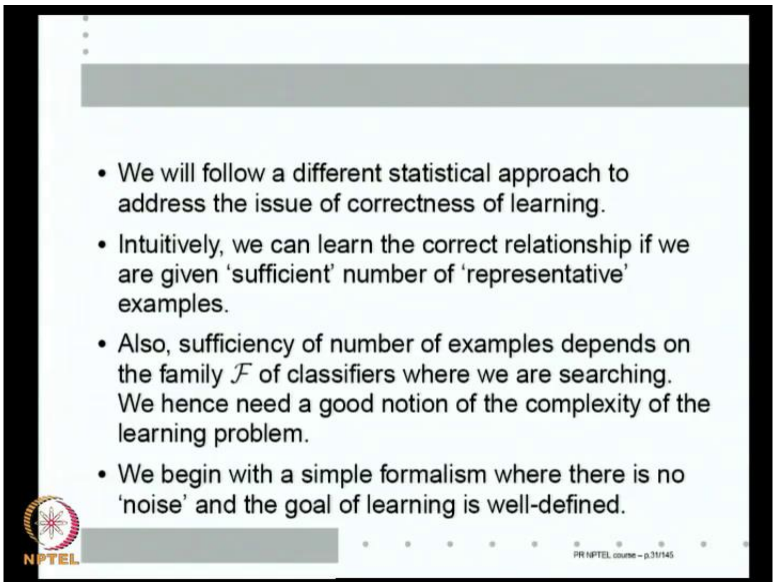
So, sending f means sending another d plus 1 numbers, so what does that means? I can either send 2 n numbers X i y i or I can send n numbers X i that is X i is common. So, let us not worry about X I, so either I can send n numbers y i or i send d plus 1 numbers of course, these have to be send with good precision. And then the send the n numbers y i minus f X I, if the fit is good i save lot of bits in y i minus f X i. So, even if I am sending d plus 1 numbers for f overall I may get, but this will also immediately tell me that getting a good fit using high degree polynomial will not pay.

For example, under MDL if I used a dth degree polynomial to get 0 errors I, I do not use any bits in sending y i minus f X i, but instead of sending n numbers y i I am sending n plus 1 coefficients for the nth degree polynomial. So, getting a good fit using a high degree polynomial will not pay and the disc minimum description principle naturally takes care of, naturally takes care of balancing the data error on the model complexity. Essentially, this is an idea of you know the basic idea that everybody would like to say simple models are preferred there is something called Occam’s razor that some of you might have heard of which is generally stated by the pitty saying that things should be simple but not simpler.

So, essentially given lot of data you want a model to explain it a simple model that explains it is always better than a complex model that explains it, so essentially want to put simple model. So, minimum description principle kind of captures is over comes rather, but in this kind of description length we are only looking at the training data. We are asking whether f is a good fit for this specific data at hand, we are not asking at all about whether it will fit new data there is no statistics here.

Of course, there can be statistical model build around minimum description principle, we would not do it, we will we will consider slightly different statistical model which is also related to this. But we consider MDL for two reasons; one is this is another interesting way of looking at data error verses model complexity trade off 2 is that this issue of what is correct? What is the best fit? The definition of what is a best fit is, is not is not something unique.

(Refer Slide Time: 13:53)



- We will follow a different statistical approach to address the issue of correctness of learning.
- Intuitively, we can learn the correct relationship if we are given 'sufficient' number of 'representative' examples.
- Also, sufficiency of number of examples depends on the family $\mathcal{F}$ of classifiers where we are searching. We hence need a good notion of the complexity of the learning problem.
- We begin with a simple formalism where there is no 'noise' and the goal of learning is well-defined.

PR NPTEL course – p.31/145

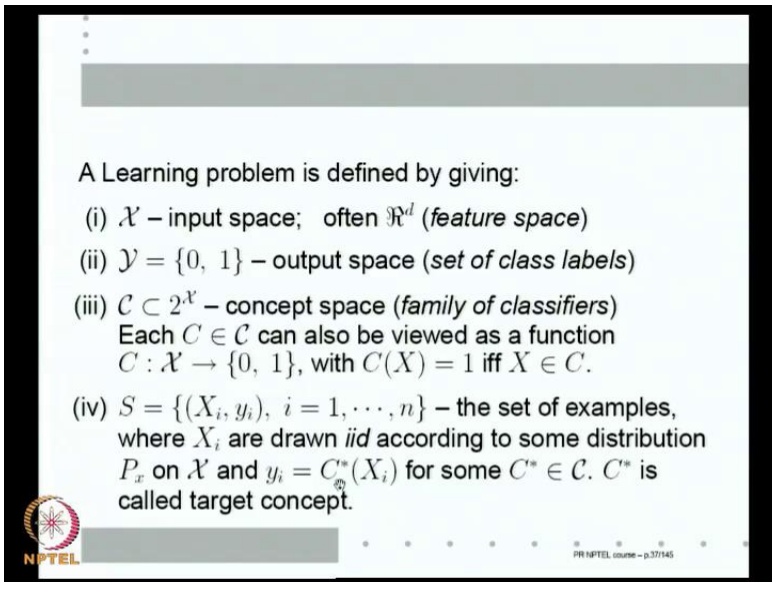
Here is a another very nice interesting way of asking which is a good function fit we would not consider MDL anymore it is just being intuitively described just to let you know that there are other ways of doing it, What we are going to do is we will concentrate on a particular and a different statistical approach, once again to address the issue of correctness of learning where this is statistical in the sense we are explicitly going to ask whether the learned function generalizes well.

So, we are bothered about how it does over all the patterns not just the once in the training set. Now, intuitively again one can always learn the correct relationship between X i and y i, if you are given sufficient number of representative examples I have put both sufficient and representative in codes its intuitive saying if I have given large number of representative examples. So I should be able to learn correctly, if I have solved a large number of good representative integration problems in class 12 then I should be able to crack any integration problem that comes in the exam. So, our learning certainly depends both on having enough number of examples and enough number of good examples that is what I mean by representative. So, if I all the examples are that I solve are very simple then maybe I have I do not learn the concept well. The second thing that we should not remember is that how many examples is sufficient of course, depends on the family of classifiers from which over which you are searching. For example, if I am even when I am searching say linear classifiers if I am searching for linear classifier over R 2 I need to only learn 3 parameters a hyper plane in R 2 is specified by 3 numbers or we can even parametrize the 2 numbers.

But if I am learning it in a 100 dimensional space I need to learn of the order of 100 parameters. So, if I have to learn more parameters intuitively I would need more examples. Simply if I am learning non-linear classifiers I may need many more examples than learning linear classifiers. So intuitively once again we would suspect that how many examples is sufficient depends on how complex is the class of classifiers over which you are trying to learn.

So, the formalism we are going to present has to capture all these issues So, going to start with of course, we will expand we will extended it later, but to get out ideas clear. We will start with a simple formalism where in the training example there is no noise and where the goal of learning is very well defined. Historically this is the first formalism that attempted to study complexity of learning from a computer science point too, but from this statistical learning theory point of view there are other formalism that were predating, today, both these kinds of things are have come together. So, we will we will first start with this formalism.

(Refer Slide Time: 17:02)



A Learning problem is defined by giving:

(i) $\mathcal{X}$ – input space; often $\Re^d$ (*feature space*)

(ii) $\mathcal{Y} = \{0, 1\}$ – output space (*set of class labels*)

(iii) $\mathcal{C} \subset 2^{\mathcal{X}}$ – concept space (*family of classifiers*)
Each $C \in \mathcal{C}$ can also be viewed as a function $C : \mathcal{X} \rightarrow \{0, 1\}$, with $C(X) = 1$ iff $X \in C$.

(iv) $S = \{(X_i, y_i),\ i = 1, \cdots, n\}$ – the set of examples, where X_i are drawn *iid* according to some distribution P_x on $\mathcal{X}$ and $y_i = C^*(X_i)$ for some $C^* \in \mathcal{C}$. C^* is called target concept.

PR NPTEL course – p.37/145

So, in formalism we will say a learning problem is defined by giving the following, you given some set I will call it script X which we called the input space. For most problems X will be some d-dimensional real Euclidian space, because you looking at all real valued features are functions defined over real Euclidian spaces. So, X is essentially a feature space for us, but anyway if the learning problem we will say the input space is the script X. Then we have an output space for this formalism we will consider only the classification problem. So, the output space is 0 1 of course, output space could be R, but now we will concentrate only on the classification problem.

So, output space is denoted by script y and for us is the set containing only 0 1. So, this is the set of class labels then we say were given a concept space; concept space denoted by script C is a subset of 2 power X. If you if you recall one of the earlier classes, we considered this notation 2 power script X is the power set of script X that is 2 power X is the set of all possible subsets of X.

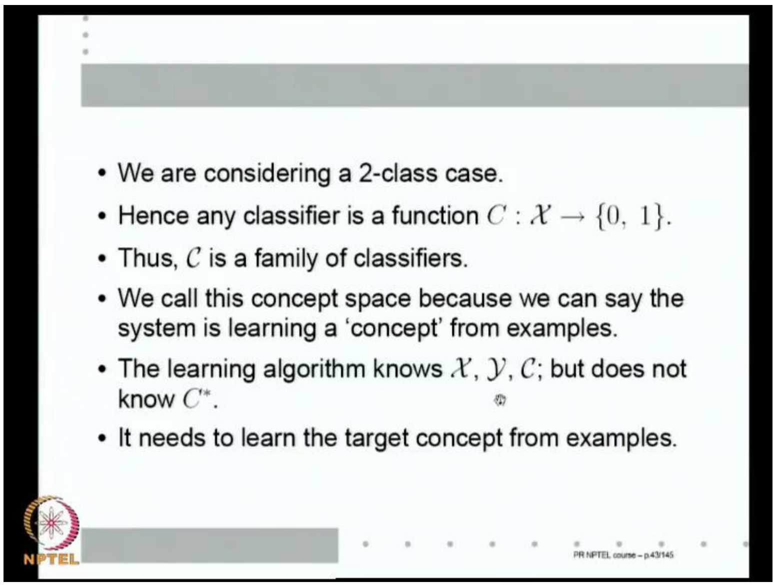
So, essentially we are given some particular class of subsets of X which we will call the concept space, some particular class of subsets of X which we will call the concept space. Why, because that denotes a family of classifiers, why does they denote a family of classifiers? If I take any C belonging to script C that will be a subset of X, because it is a subset of X i can also think of it as a function that maps X to 0 1 where the function is C of X is 1, if and only if that X is in the subset C. A, a caution about notation whenever you are considering any subset of any set we can always represent that subset also as a function like this, this is called the characteristic function of the subset.

So we are going to use the same symbol C to denote the function as well as a subset that is mixed in this part of the sentence see we are saying C of X here So, C is a function instead of saying X belongs to C here C is a set. So, essentially any C there is a subset of X can be represented by its characteristic function which we also denote by the a same symbol C where on any given X from script X C of X takes value 1, if that X is in the subset C. So, in that sense any subset any class of subsets of X can be thought of as a particular family of classifiers. For example, if you think of linear classifiers they are define by hyperplanes, a hyperplane can also be viewed as giving you a subset on which on the positive side of the hyperplane, so those are the half spaces.

So I can think of half spaces as the class of subsets of X that I am working on or I can think of hyperplanes as the class of classifiers I am working on. So, any case C denotes the family of classifiers that we will be working with then; obviously, the, the, the problem is a learning problem. So, we have a set of examples, so as usual we will represent the set of examples S as X i y i there are n examples. We assume X i's are drawn i i d according to some distribution P x on X the distribution can be arbitrary, we may not even know it.

So, the examples are drawn like this and each y i is obtained as C star X i So, there is some particular element C star belonging to this class C which we call the target concept. And all my examples are classified exactly according to C star so C star is like a god given classifier. So, there is noise in the example given an X i there is a god given classification for X i which is given by C star X i. So, we are given this is i i d examples X i along with their class label which is obtained from a target concept C star.

(Refer Slide Time: 21:27)



- We are considering a 2-class case.
- Hence any classifier is a function $C : \mathcal{X} \rightarrow \{0, 1\}$.
- Thus, $\mathcal{C}$ is a family of classifiers.
- We call this concept space because we can say the system is learning a 'concept' from examples.
- The learning algorithm knows $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{C}$; but does not know C^*.
- It needs to learn the target concept from examples.

PR NPTEL course – p.43/145

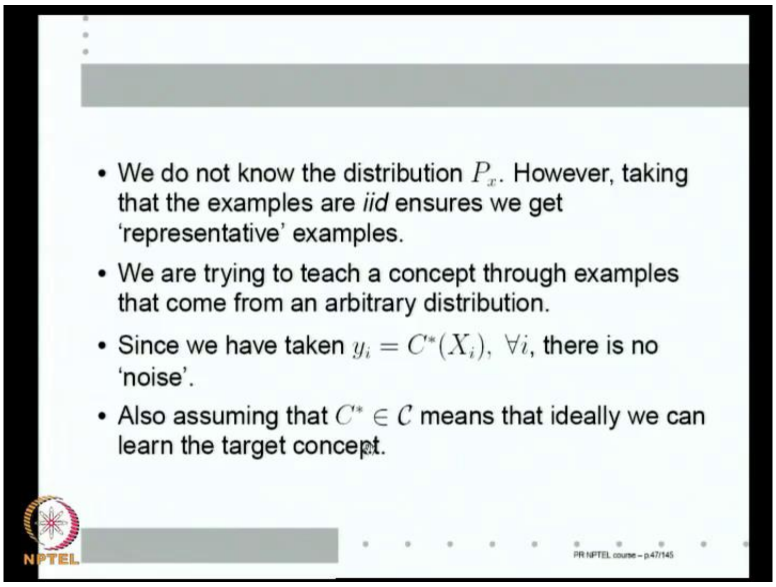
So, my task now is to given these examples searching over C can I learn some classifier which is a good approximation to C star. So, a few comments on the formalism we are considering a 2 class case that is clear, because we are taking Y is equal to 0 1 and that is why we are able to represent any classifier also as a subset of X. So, every classifier is the function from X to 0 1 and hence it is also a subset of X.

So that is why C is a family of classifiers, we essentially calling this a concept space, because we can think of this as learning some concept from example learning the concept of when integration by a parts is useful. So, I will give you some examples where it is useful some examples where it is not useful can you learn general rules of where it is useful or may be more simple and more colorful example let us say I have a son 4 5 years old I, I every day I take him on the terrace of my house and as cars passed by on the road I tell him that is a Maruti car, this is a Ford car whatever let us say error is a Maruti car and this is not a Maruti car.

Now, can he learn to recognize of course, kids of that age have remarkably nice memory for shapes. So, generally within a few days they learn the car names easily so I may be classifying the examples where reading what model it is on the car. But the kid may be learning it using his own shape descriptors, but any case that is the kind of thing. So, here my examples comes randomly I do not know what distribution I just stand on my terrace and whatever car passes by that is my example I show to the child and from these examples we are learning.

So, you can think of it as that is why you we often think of this as a concept learning from examples the learning algorithm essentially is. So, coming back to the formalism now forget about the examples. Coming back to the formalism, the learning algorithm knows X; obviously, knows Y knows the class of C, because that is over which it is set it does not know C star, it is only given examples I had learn C star. So, the environment may through a, some example one learning problem is the environment pick some C star from C. And those examples are the learning algorithm and the learning algorithm has to learn that particular C star. Today may be teaching my son to recognize makes of cars, tomorrow may be teaching makes of bikes.

(Refer Slide Time: 24:09)



- We do not know the distribution P_x. However, taking that the examples are *iid* ensures we get 'representative' examples.
- We are trying to teach a concept through examples that come from an arbitrary distribution.
- Since we have taken $y_i = C^*(X_i),\ \forall i$, there is no 'noise'.
- Also assuming that $C^* \in \mathcal{C}$ means that ideally we can learn the target concept.

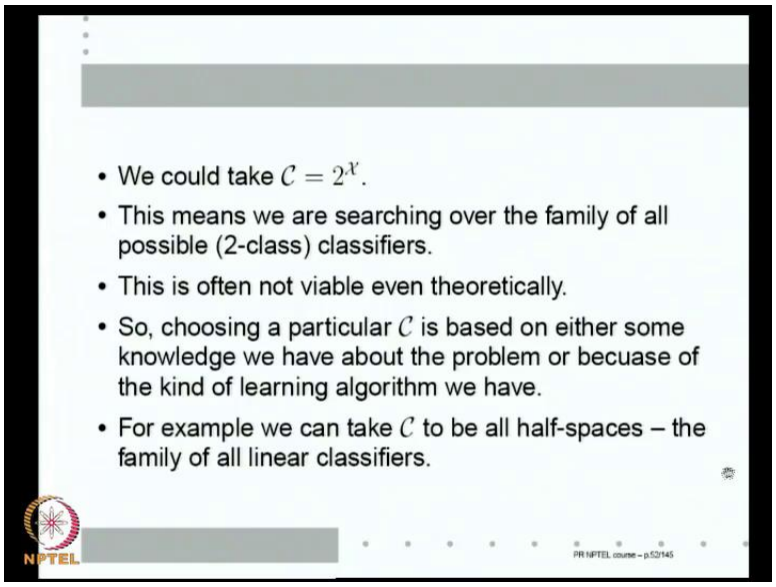
PR NPTEL course – p.47/145

So, no matter what it is once I identify a class C the environment throws examples at me which are classifier acting to some C star picked up from C and I have to learn that. So I essentially have to learn the target concept from examples. So, let us continue with the comments we do not know the distribution P x, we are not assuming any knowledge of P X, P x can be arbitrary we, we may have no knowledge of P x. And no matter what is the distribution is we should still learn I cannot order the distribution of how Maruti and non Maruti cars comes on my road whichever way the nature orders them that is how I have to learn.

However, we are making examples i i d ensures that we get representative samples from whatever distribution we are expected to learn, we assumed i i d so that examples are independent. So, for example, if I if we think of a teacher teaching you math's concept of how to integrate by parts, we do not allow the teacher to cheat you by solving simple examples in class and giving difficult examples in the exam.

So, examples are randomly drawn from all the problems in the text book in a independent manner. So, they will be simple one, they will be complicated one, so everything comes. So, by making examples i i d we can kind of ensure representative examples are same. We are trying to teach a concept through examples that come from arbitrary distribution since we are taken y i to be C star X i in this problem there is no scope for noise in this formalism and assuming C star belongs to C means that ideally we can learn the target concept. Because searching over C saying that I should I should properly approximate C star is reasonable, because the target concept exist in my bag.

(Refer Slide Time: 25:48)



- We could take $\mathcal{C} = 2^{\mathcal{X}}$.
- This means we are searching over the family of all possible (2-class) classifiers.
- This is often not viable even theoretically.
- So, choosing a particular $\mathcal{C}$ is based on either some knowledge we have about the problem or becuase of the kind of learning algorithm we have.
- For example we can take $\mathcal{C}$ to be all half-spaces – the family of all linear classifiers.

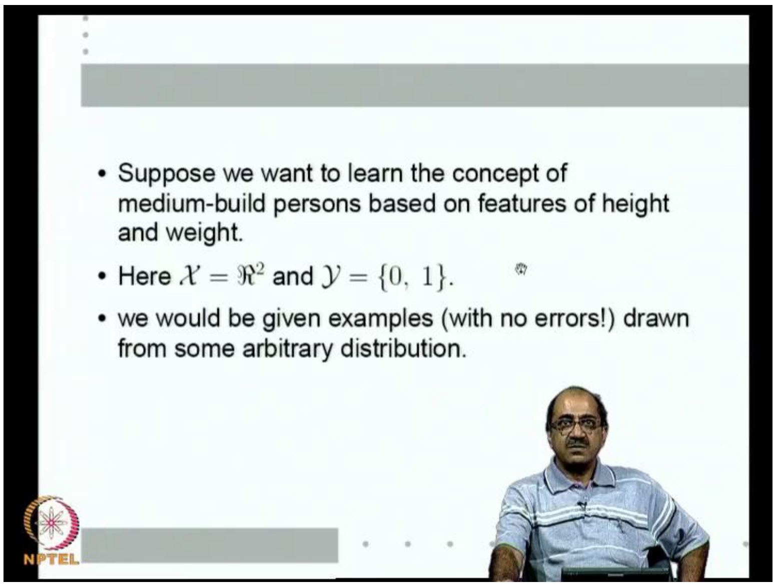
PR NPTEL course – p.52/145

Now, before we go further let us ask why do I need this particular family of classifiers I can for examples take C to be 2 power X, what does that mean? It means that I am searching over every possible classifier 2 class classifier. If I take C to be 2 power X it simply means that I am searching over all possible 2 class classifiers. So, no matter what the target concept is I would always be able to learn there is no restrictions.

Obviously, this looks too ambitious, as we shall see later on even theoretically it is not viable if you take C is equal to 2 power X except in the special case where X is a finite set. You will never be able to learn well, but any way we can take C is equal to 2 power X where the idea is that we choose a particular C based on some knowledge we have about the problem not because of the kind of learning algorithm, we have maybe I have an algorithm that can only work with certain C.

So, the best I can do is to set over that C and hope that the target concept is in that or I have some extra knowledge about the problem, because of which I can take a particular C even though I do not know C star. For example, the second guess occurs when we choose linear classifiers we know that linear separabilities very hard, we know that there many situation where even the best linear classifier may not be good enough. But still because the algorithms are simple and you know they have nice structure, we may still want to learn linear classifiers.

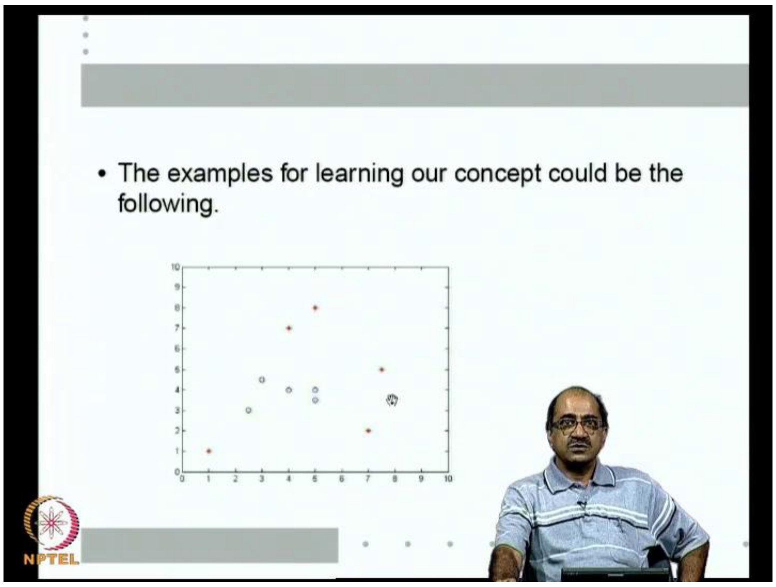
(Refer Slide Time: 27:31)



- Suppose we want to learn the concept of medium-build persons based on features of height and weight.
- Here $\mathcal{X} = \Re^2$ and $\mathcal{Y} = \{0, 1\}$.
- we would be given examples (with no errors!) drawn from some arbitrary distribution.

So, one example is instead of taking 2 power X we take set of 4 half spaces and learn the family of linear classifiers this is because we have a good learning algorithm. Here is another example where we may choose C based on some knowledge we have once again to make it a little colorful. We will continue this example for all the theoretical development, we have in this class. So, let us say we want to learn the concept of medium build persons based on 2 features height and weight, because it is a simple R 2 classification problem, the ideal medium build is just to give it you know some flavor.

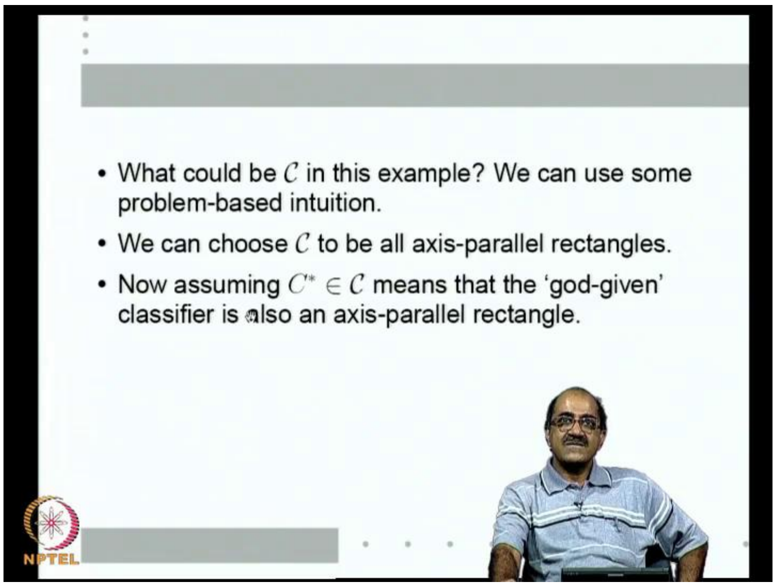
(Refer Slide Time: 28:12)



- The examples for learning our concept could be the following.

So, X is R 2 y 0 1 is a 2 class problem and we would be given some examples of course, examples have no errors drawn from some arbitrary distribution. So, for example, my examples could be like this there are there are 2 axes there are some point which are all positive examples, one class examples. All those red stars are the other class examples, because it is medium build essentially in the middle of the height weight range is what we will think of medium built any extremes are not medium built.

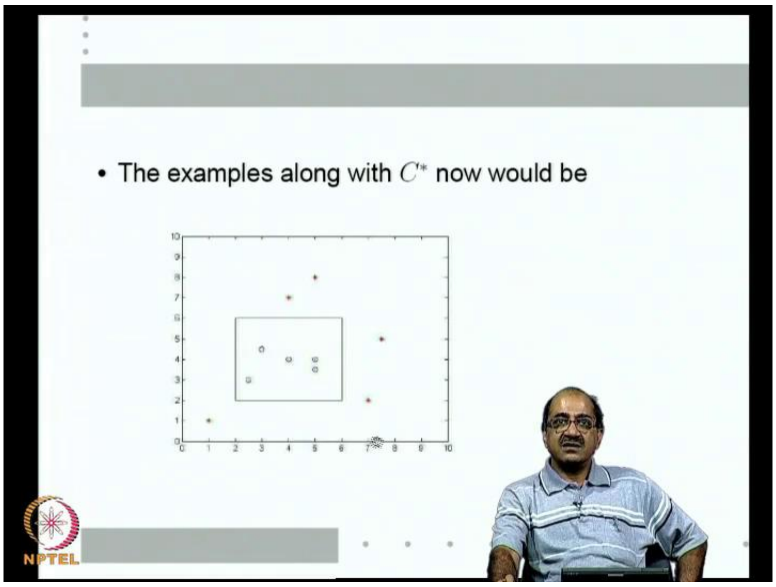
(Refer Slide Time: 28:40)



- What could be $\mathcal{C}$ in this example? We can use some problem-based intuition.
- We can choose $\mathcal{C}$ to be all axis-parallel rectangles.
- Now assuming $C^* \in \mathcal{C}$ means that the 'god-given' classifier is also an axis-parallel rectangle.

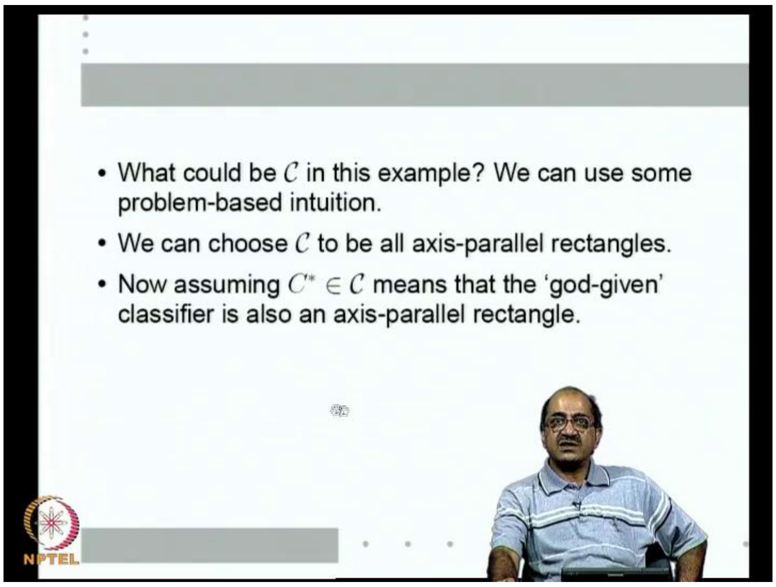
So, for example, these could be one set of examples for our concept. We have given this what can we say about C, for example, we will say that if somebody is medium built at a particular height and weight under the same weight. And another height also we will call somebody medium built then at that weight in between all the heights also he should be medium built. So, one way in which we can approximate medium built is to say that the class C should be axis parallel rectangles.

(Refer Slide Time: 29:11)



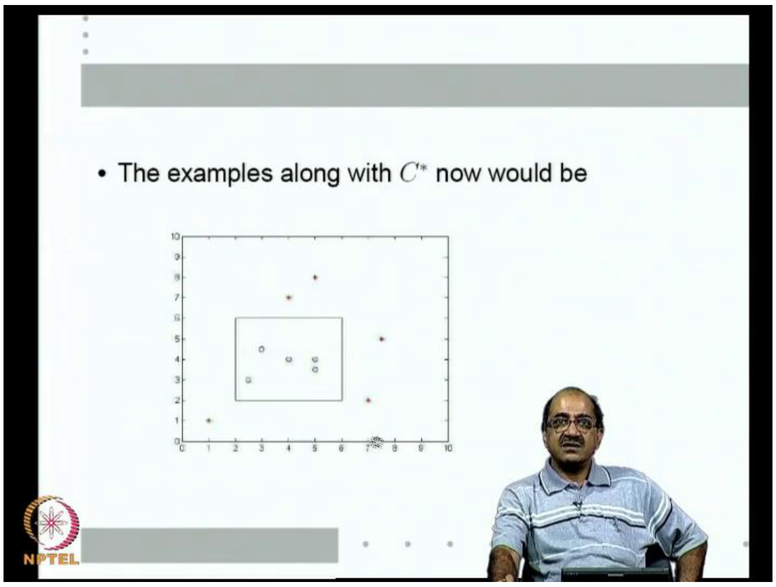
- The examples along with C^* now would be

(Refer Slide Time: 28:40)



- What could be $\mathcal{C}$ in this example? We can use some problem-based intuition.
- We can choose $\mathcal{C}$ to be all axis-parallel rectangles.
- Now assuming $C^* \in \mathcal{C}$ means that the 'god-given' classifier is also an axis-parallel rectangle.

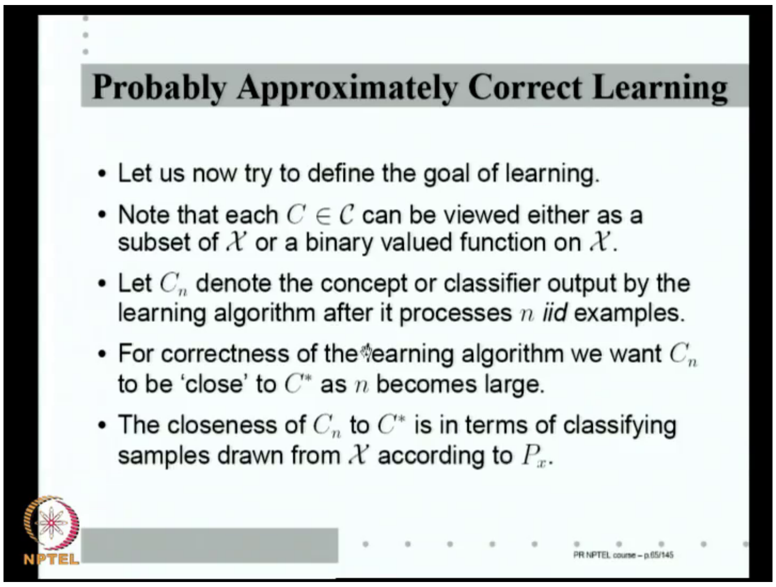
(Refer Slide Time: 29:11)



- The examples along with C^* now would be

So, my god given concept can be that they, they are some weight height ranges if the height weight falls inside that Then we will call it medium built outside we call it not medium built. So, we can think that the unknown C star would be some rectangle whose sides are parallel to the coordinate axis. So, we will call it axis parallel rectangle, so we can choose C to be set of all axis parallel rectangles, once we choose C to be set of all axis parallel rectangles assuming that C star belongs to C means that the god given classifier is also an axis parallel rectangles. So, we are assuming that all our examples come classifier with respect to one particular axis parallel rectangle. We do not know which particular axis parallel rectangle that is what we have to learn.

(Refer Slide Time: 30:00)



Probably Approximately Correct Learning

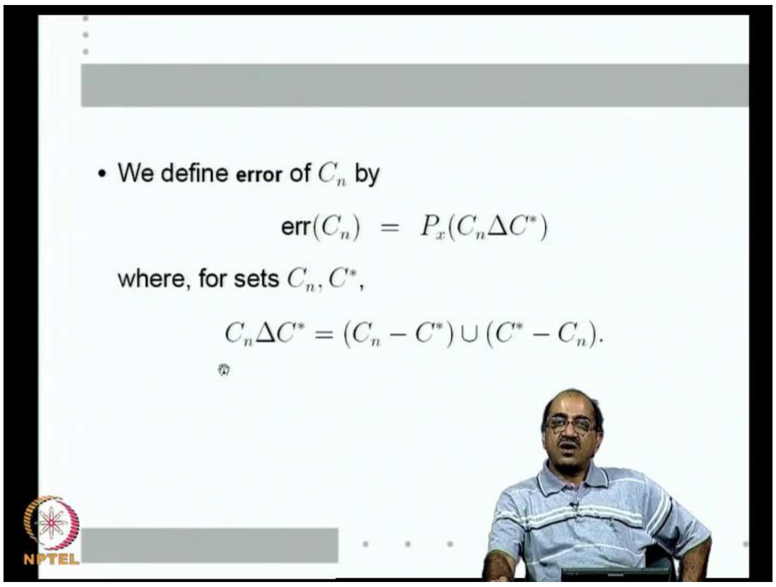
- Let us now try to define the goal of learning.
- Note that each $C \in \mathcal{C}$ can be viewed either as a subset of $\mathcal{X}$ or a binary valued function on $\mathcal{X}$.
- Let C_n denote the concept or classifier output by the learning algorithm after it processes n *iid* examples.
- For correctness of the learning algorithm we want C_n to be 'close' to C^* as n becomes large.
- The closeness of C_n to C^* is in terms of classifying samples drawn from $\mathcal{X}$ according to P_x.

PR NPTEL course – p.65/145

Now, we will keep the example aside, we will first develop the theory of defining goal of learning and then we will come back to the example. So, the particular kind of formalism we are considering is called probably approximately correct learning PAC learning for short PAC, we will we will we will define it shortly not that each C in C can be viewed either as a subset of X or a binary valued function as I said. And we will simultaneously be using both notions of a element of the concept class.

Let us say C subscript n denotes the concept or the classifier output by the learning algorithm after it processes n i i d example we have seen a learning algorithm with a black box who takes samples and outputs a function from its class of function, so script C is your class of functions or classifiers. So, after seeing n i i d examples what it outputs? Let us denote it by C of n C subscript of n for correctness of the learning algorithm, we want C n to become close to C star as n becomes large, because now we know examples, all examples have classified according to C star we have a very good notion of what is correctness. So, we want C n to become close to C star as n becomes large by close to C star, we do not mean the subset C n should become same as the subset C star. The closeness is in terms of classifying samples drawn from X according to the distribution.

(Refer Slide Time: 31:56)



- We define error of C_n by

$$\text{err}(C_n) = P_x(C_n \Delta C^*)$$

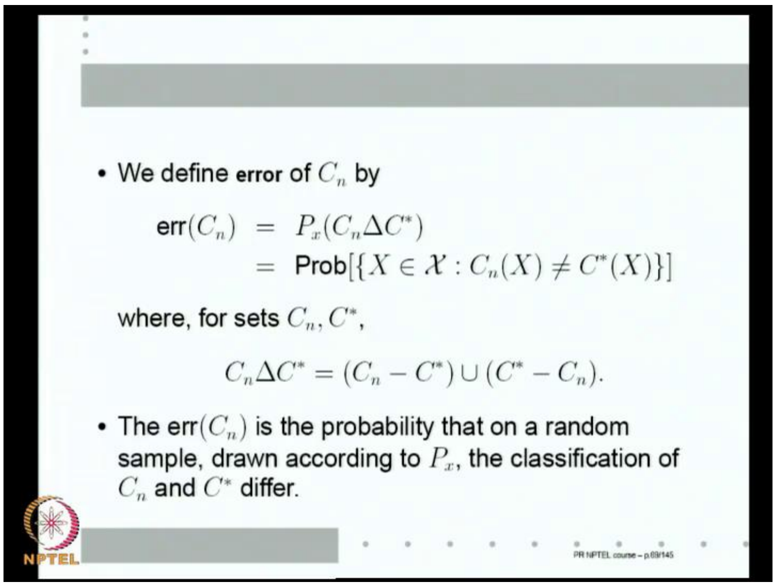
where, for sets C_n, C^*,

$$C_n \Delta C^* = (C_n - C^*) \cup (C^* - C_n).$$

So, C n and C star are closed if both of them as classifiers are equally good, actually a sets where the identical or not is not of particular interest to us So, let us use all this intuition to define whole of learning now. So, given that I have output C n let us first define what we call by error of C n, how badly does C n do? So we will define error C n at the P x probability P x a distribution remember, so P x a distribution over the space input space script X which means for every subset of X. It assigns some probability. So, so error of a classifier C n a classifier C n is a subset of X error of this concept or classifier C n. It is defined as the probability assign by the P x distribution to the subset of X which is C n delta C star where delta represents the so called symmetric difference between 2 sets.

So, C n delta C star is C n minus C star union C star minus C n C n minus C star is set of points in C n, but not in C star. And this is set of points in C star but not in C n So, C n delta C star consist of the set of all points in X which are in C n and not in C star or in C star not in C n. Now C n and C star are subsets are also classifiers, so if a point is inside the set it, it is it is given as a positive class outside we have given it as a negative class. So, this essentially those are in C n, but not in C star then C n would say one for their class where C star would say 0. Similarly, those in C star, but not in C n C star would say 1 for that class C n would say 0 for that class. So, the symmetric difference C n delta C star essentially gives us the set of all points on which the classification of C n and C star will differ .

(Refer Slide Time: 33:42)



- We define error of C_n by

$$\begin{aligned} \text{err}(C_n) &= P_x(C_n \Delta C^*) \\ &= \text{Prob}[\{X \in \mathcal{X} : C_n(X) \neq C^*(X)\}] \end{aligned}$$

where, for sets C_n, C^*,

$$C_n \Delta C^* = (C_n - C^*) \cup (C^* - C_n).$$

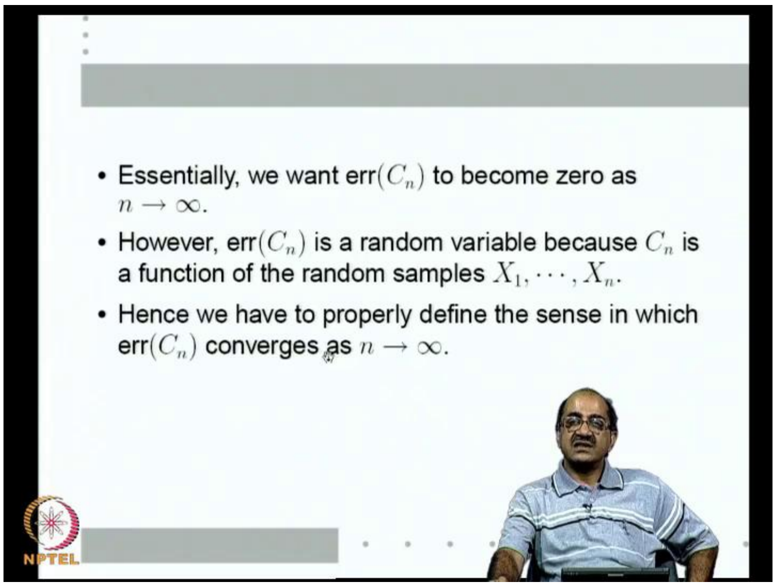
- The err(C_n) is the probability that on a random sample, drawn according to P_x, the classification of C_n and C^* differ.

PR NPTEL course – p.69/145

So, this P x probability of C n delta C star is same as a probability of the set of all X on which C n X is not equal to C star X, this is essentially what you want. If this is 0 then we for, for all our purposes C n and C star are same, now even if the subset C n and C stars are different this probability could be 0, because the distribution P x is arbitrary. We know nothing of the distribution P x they might be subsets of X for which P x value 0.

So, if C n and C star differ by a subset of X whose P x probability 0 then the error would also be 0. So, error C n is essentially is P x probability of C n delta C star which is same as the probability that C n and C star would differ on classifier. So, this is actually the generalization error we define it, because there is the notion of the correct concept. So, this is the generalization error of the classifier C n the probability that a randomly drawn X would be wrongly classified I can say wrongly classified because C star X is god given correct classification for it.

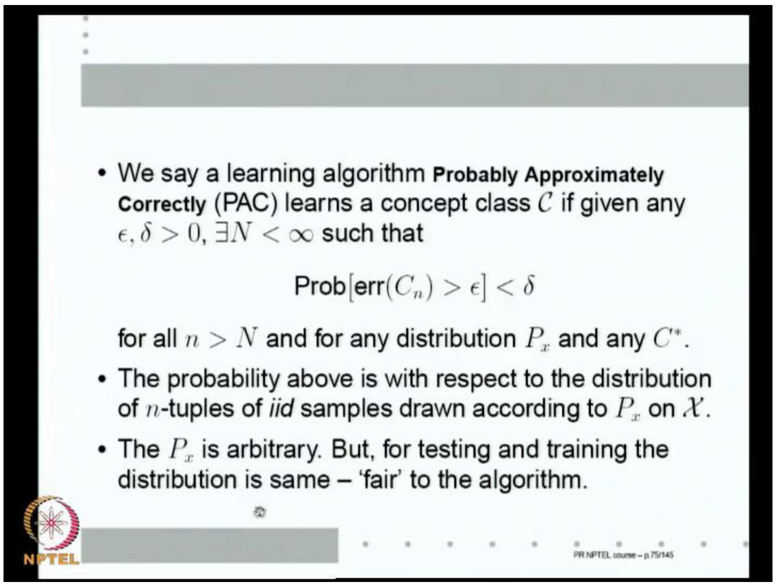
(Refer Slide Time: 35:01)



- Essentially, we want err(C_n) to become zero as $n \to \infty$.
- However, err(C_n) is a random variable because C_n is a function of the random samples $X_1, \cdots, X_n$.
- Hence we have to properly define the sense in which err(C_n) converges as $n \to \infty$.

So that is what we define error of C n to be. So, error of C n is the probability that on a random sample drawn according to P x the classification of C n and C star differ. So, what does, what we want out of the algorithm as we see more and more examples we want error of C n to become 0 then C n becomes essentially same as C star and that tells us that our learning is fine. So, we should define our goal of learning as, as error C n going to 0 as n tends to infinity. Before we can define this, we should remember that error C n even though it is defined as a probability is itself a random variable, why because C n is a random variable C n is a function of X 1 X 2 X n. So, because C n is a random variable which is a function of the random variables X 1 to X n error C n will be a random variable.

(Refer Slide Time: 35:58)



- We say a learning algorithm **Probably Approximately Correctly** (PAC) learns a concept class $\mathcal{C}$ if given any $\epsilon, \delta > 0$, $\exists N < \infty$ such that

$$\text{Prob}[\text{err}(C_n) > \epsilon] < \delta$$

for all $n > N$ and for any distribution P_x and any C^*.
- The probability above is with respect to the distribution of n-tuples of *iid* samples drawn according to P_x on $\mathcal{X}$.
- The P_x is arbitrary. But, for testing and training the distribution is same – 'fair' to the algorithm.

PR NPTEL course – p.75/145

So, when we say error C n converges to 0 we should properly define the sense in which error C n converges as some of you may know when we are considering sequence as random variables there are multiple ways of defining convergence. So, keeping this in mind; this is how we will we are going to define correctness. We say learning algorithm probably, approximately, correctly learns a concept class C. If given any epsilon delta which are strictly positive there exists a number n less than infinity such that probability that error C n is greater than epsilon is less then delta for all n greater than n. So, this part simply means that probability error C n greater than epsilon goes to 0 as n tends to infinity.

So, for those of you, who are familiar with modes of convergence of random variables this is convergence in probability that we are saying error C n converges to 0 in probability. It really does not matter if you do not know what convergence and probability means we, we do not need that in this course otherwise, so that is why I have explicitly put it here. So, this definition is good enough this definition is actually definition of convergence and probability. So, given any epsilon delta greater than 0 there exist a n such that probability error C n greater than epsilon is less then delta.

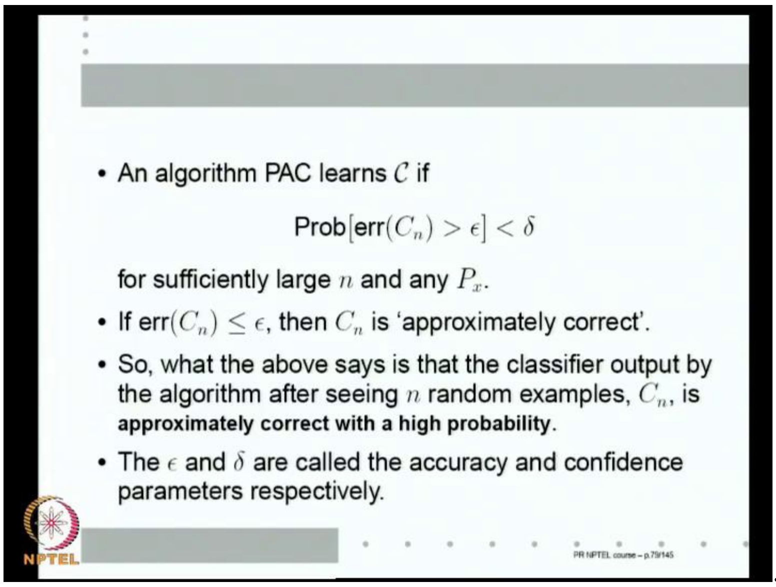
So, no matter how small delta you give me and how small epsilon you give me I can always find a N such that the probability that the concept I output after seeing an examples has error more than epsilon is less than delta. This should happen no matter what P x is no matter what C star is I am only know class C m I am I am there to learn any, any C star belonging to C as long as you give me examples. So, no matter what P x is no matter what C star is the algorithm should ensure this that is when we say the algorithm can probably approximately correctly learn the concept class C i i in shortly I will explain these adjectives in the definition. First what is this probability with respect to probability respect to what distribution this probability? What is the random variable inside here C n? C n is a function of X 1 to X n.

So, this probability is with respect to the distribution of n tuples of i i d samples, because C n is a function on n i i d samples. So, this probability is essentially the N fold product probability of P x, because these, these examples are drawn i i d. So, this probability with respect to the distribution of N tuples of i i d samples drawn according to P x on X P x is arbitrary, but the testing and training distributions are the same. We are giving samples with respect to P x and we are assessing you on error C n error C n can be 0 as you already said even if C n is not equal to C star, as long as under P x C n and C star as C n delta C star is 0 as 0 probability mass.

So, while P x is arbitrary the fairness in the in the in the formalism comes, because I am giving you training examples drawn i i d according to P x. And I am defining correctness also with respect to the P x probability of where you make errors. So, for example, if there is a space in X where P x probability 0 subset of X i where P x probability is 0. Then not being able to learn how to classify those x's, you are not penalized for because you will never ever see those examples. So, you are not expected to learn or more colorfully let us say in the town in which I live or let us say I am trying to teach my son to distinguish between Maruti cars and ambassador cars. And let us say in the town I I live all Maruti cars are non white and all ambassadors are white.

So you have seen many examples and if my son learn thought that color is what determines Maruti versus Ambassador I cannot blame him that is essentially what this means, because errors C n is respect to the P x distribution. So, I am giving you training samples of P x distribution and I am rating you once again based on how you do on a random sample drawn with respect to the same P x.

(Refer Slide Time: 40:32)



- An algorithm PAC learns $\mathcal{C}$ if

$$\text{Prob}\left[\text{err}(C_n) > \epsilon\right] < \delta$$

for sufficiently large n and any P_x.

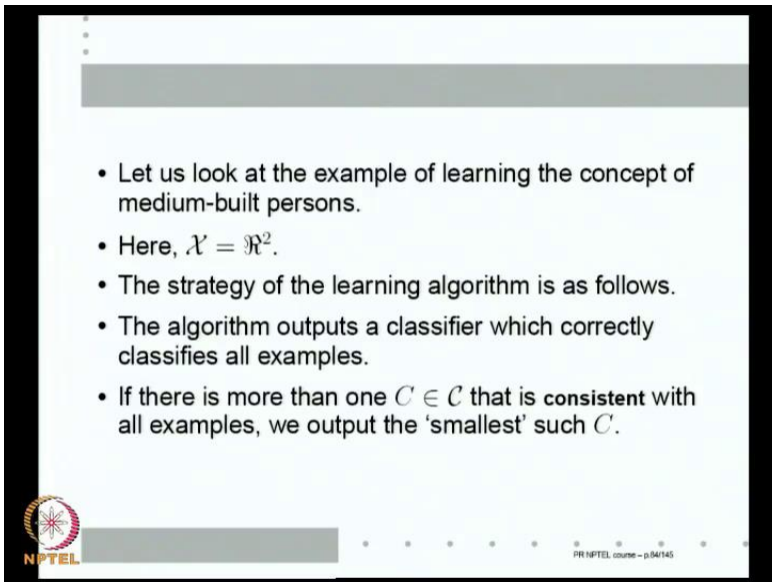
- If $\text{err}(C_n) \leq \epsilon$, then C_n is 'approximately correct'.
- So, what the above says is that the classifier output by the algorithm after seeing n random examples, C_n, is **approximately correct with a high probability**.
- The ϵ and δ are called the accuracy and confidence parameters respectively.

PR NPTEL course – p.79/145

So, for example, if I learnt what a medium build persons in India are that that notion may not carry over to medium built persons. Let us say in northern Europe, because there generally people are much more hefty, but the, the algorithm cannot be penalize that is what is mean by even though I am saying that this should hold for any distribution. It is still fair to the algorithm, because both training and testing are done on the same distribution. So, essentially an algorithm PAC learns if probability error C n greater than epsilon is less than delta for sufficiently large n and P x and of course, any C star. Why does the name error C n less than epsilon means C n is an approximately correct classifier, because it does not make more than epsilon errors.

So, what this says is the classifier output by the algorithm after seeing n random examples is approximately correct with a large probability, it is probability of being not approximately correct is small or it is probability of being approximately correct is. So, I can write the same thing as probability error C n less than equal to 1 minus delta is greater than 1 minus epsilon greater than equal 1 minus delta.

(Refer Slide Time: 41:49)



- Let us look at the example of learning the concept of medium-built persons.
- Here, $\mathcal{X} = \Re^2$.
- The strategy of the learning algorithm is as follows.
- The algorithm outputs a classifier which correctly classifies all examples.
- If there is more than one $C \in \mathcal{C}$ that is **consistent** with all examples, we output the 'smallest' such C.

PR NPTEL course – p.84/145

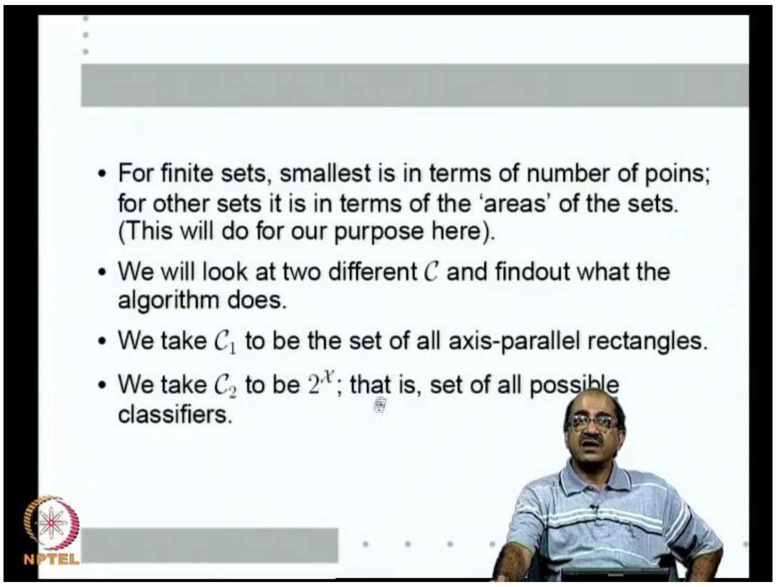
So, essentially what you are saying is that I learn an approximately correct classifier with a high probability that is where that is why it is called probably, approximately correct learning, the epsilon delta are often called the accuracy and confidence parameters. So, I am learning to an accuracy of epsilon and with a confidence delta. The idea is that no matter what accuracy and confidence you give me I should be able to tell you an n, so that if you give me that many examples I can learn to that accuracy and confidence. Let us go back to the example of learning that medium built persons and think of a simple algorithm strategy and ask is that a PAC learning algorithm.

So, X is R 2 now I have I have to have learning algorithm this, this is 3 d. So, I do not have to worry about how I am implementing the algorithm where there is a real computer algorithm or not I am just looking at a strategy for the algorithm, it is it knows the class C and it has to stretch over C. So, this is what the algorithm does after seeing n examples the algorithm outputs a classifier that correctly classifies all the example, it changes into it, it is back C looks for some C which correctly classifies all the examples.

Now, because C star is in C and C star classifies all examples correctly, at any given time no matter how many examples you have seen they will always be at least one classifier in my bag which correctly classifies all the examples. So, I should I should always be able to find a classifier which correctly classifies all examples though C star correctly classifies all the examples, there is no guarantee that given any finite subset of examples that is all I can see in a learning algorithm they may be other classifiers within C which also classify this finite examples correctly.

So, there can be more than one C in script C that is consistent with all by consistent we mean it classifies all the examples correctly. So, more than one C what should my algorithm do it is still should still have a simple rule of what to do this is an algorithm all this say. So, what it does is it outputs the smallest such set C, because each classifier is a subset of X i can have a notional smallest set C.

(Refer Slide Time: 43:34)



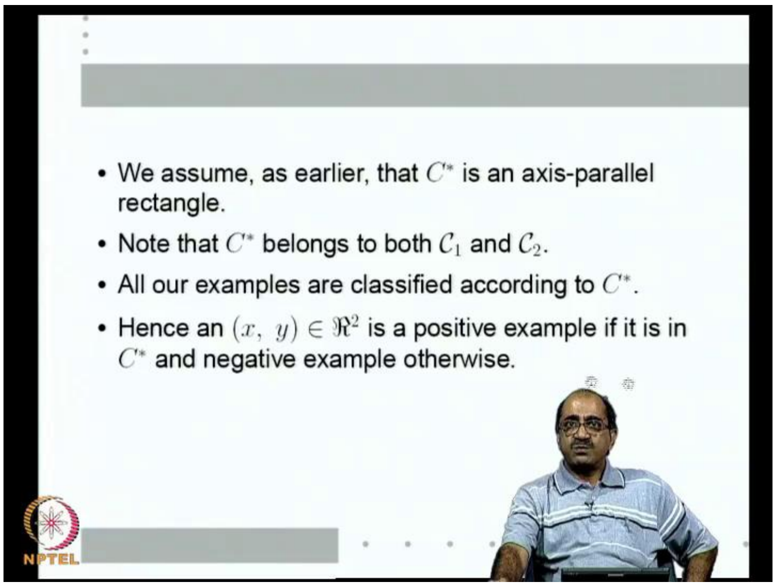
- For finite sets, smallest is in terms of number of poins; for other sets it is in terms of the 'areas' of the sets. (This will do for our purpose here).
- We will look at two different $\mathcal{C}$ and findout what the algorithm does.
- We take $\mathcal{C}_1$ to be the set of all axis-parallel rectangles.
- We take $\mathcal{C}_2$ to be $2^{\mathcal{X}}$; that is, set of all possible classifiers.

So, if there is more than one C that is consistent with examples I output the smallest subset how do I define small, if there is a finite subset of C in my, so finite subset of X in my bag and if that finite subset is consistent then I am comparing finite subsets. So, smallest is in terms of number of points for finite sets. So, if I am looking at an infinite set and a finite set finite set is smaller than the infinite set, if I am looking a 2 finite sets; the one which has small number of points is on the other hand if I am comparing 2 infinite sets they are subsets of R 2.

So, we will say there is in terms of areas, because there is problems of if suppose one set is aligned what is its area, it does not matter for us this is simple example we will never encounter that. So, essentially the algorithms strategy is very simple output, any element of your C find an element of yours of your concept class that correctly classifies all the examples. If more than once such elements exists, find the smallest, smallest is in terms of number of points. If there are finite sets in terms of area, if there are infinite subsets of R 2 we look at 2 different concept classes and both use the same strategy and find out what the algorithm does. So, for C 1 we take the set of all axis parallel rectangles, we have as already seen that might be a nice thing, because we know this C star is an axis parallel rectangle.

Of course, this C 1 is still uncountably infinite that number of elements in the C 1 is uncountably infinite, because given any 2 real real numbers one above the other in R 2 I can draw an axis parallel rectangle with those 2 as the as 2 vertices. So, the number of axis parallel rectangle is still uncountably infinite, but C 1 is contains only set of axis parallel rectangle, no other subset of R 2 is involve. We will take C 2 to be 2 power X that is C 2 is the set of all possible classifiers. The idea is we are going to show that if I learn with this strategy, this strategy looks very simple the strategy is same I am just trying to look at the simplest element in my concept class that is consisting with all examples looks a very nice strategy. Now, not I am not bother about how I can implement for various concept classes, but forget that.

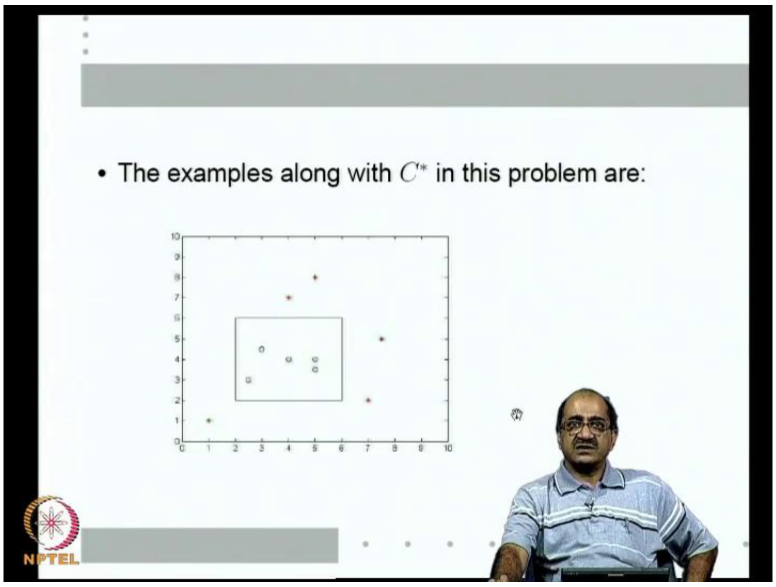
(Refer Slide Time: 46:18)



- We assume, as earlier, that C^* is an axis-parallel rectangle.
- Note that C^* belongs to both $\mathcal{C}_1$ and $\mathcal{C}_2$.
- All our examples are classified according to C^*.
- Hence an $(x,\ y) \in \Re^2$ is a positive example if it is in C^* and negative example otherwise.

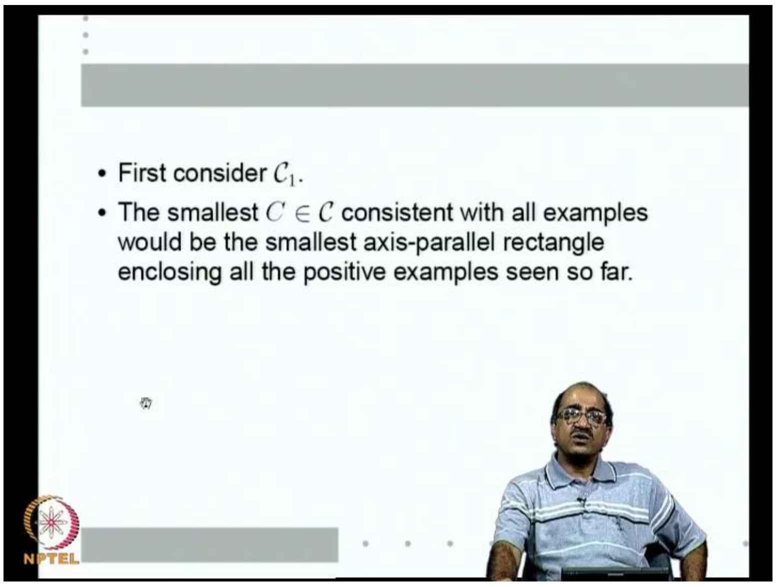
It looks a very nice strategy and what more can I do I am I I am and I was told there is no noise, so might as I look for a classifier that classifies all examples correctly. If there is more than one such thing I can find in my bag I am giving you the simplest one, but the same strategy when the bag is different can make a difference between being able to learn well to not being able to learn well that is what we are going to show. Of course, we assume as earlier that C star, the god given classifier is an axis parallel rectangle and C star belongs to both C 1 and C 2, because C 1 is the set of all axis parallel rectangles C star will be in C 1 C 2 is set of all possible subsets of R 2. So, once again C star will be in C 2, all our examples are classified according to C star. So, hence given any X y it is positives if it is inside the rectangle C star negative if it is outside.

(Refer Slide Time: 46:47)



- The examples along with C^* in this problem are:

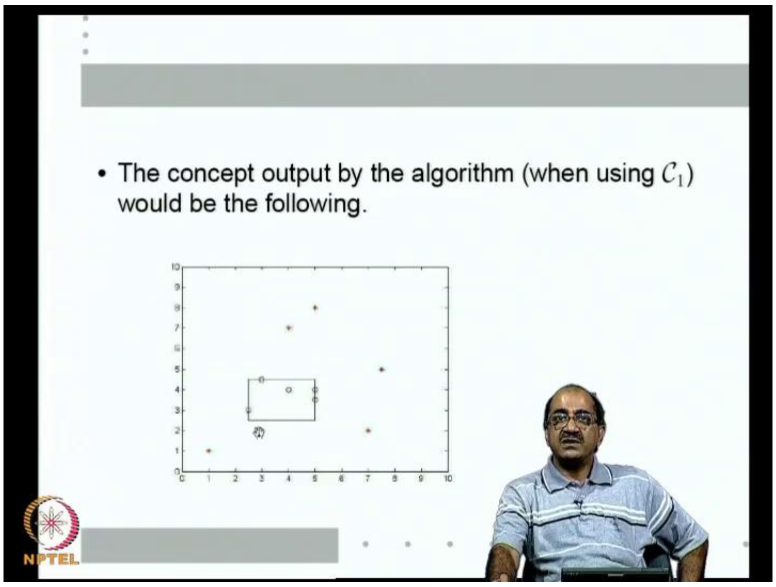
(Refer Slide Time: 47:08)



- First consider $\mathcal{C}_1$.
- The smallest $C \in \mathcal{C}$ consistent with all examples would be the smallest axis-parallel rectangle enclosing all the positive examples seen so far.

So, let us recall this is our sample this is the C star which we do not know. So, when I get examples I get points randomly sampled according to P x, if a point falls inside this it will come with a positive of one label, if it falls outside it will comes to a negative or 0 label. So, first consider the, our learning strategy with the concept class C 1, so what does it do after seeing n examples? It is looking for a a smallest element in C that is consistent with all examples. Now, all elements in C at infinite sets, so it is looking for the axis parallel rectangle with smallest area that is consistent with all examples.

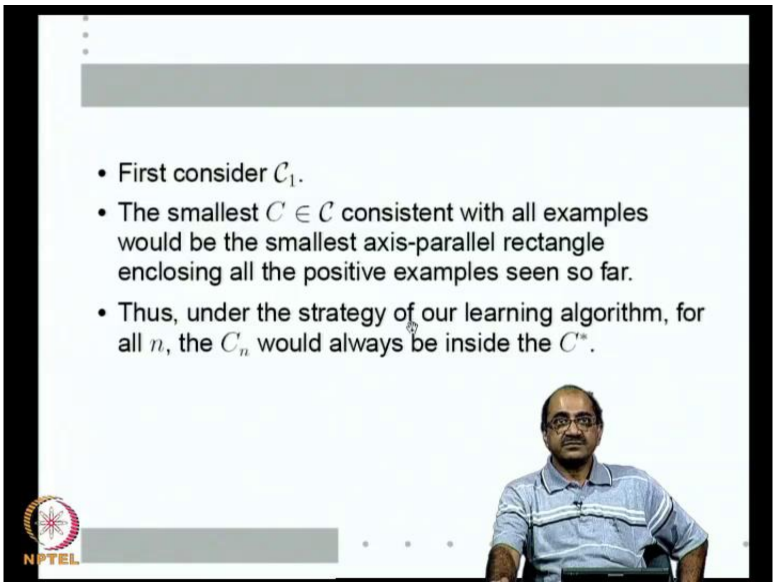
(Refer Slide Time: 47:37)



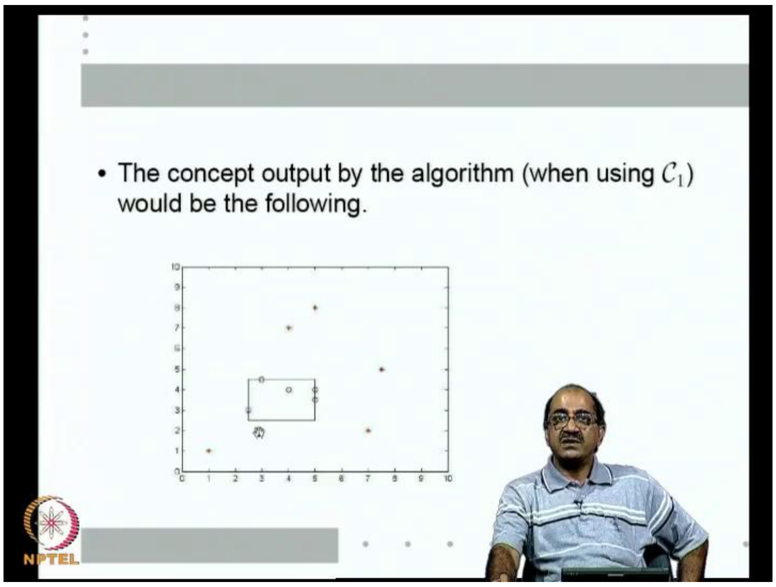
- The concept output by the algorithm (when using $\mathcal{C}_1$) would be the following.

So, what will that be? It will be axis parallel rectangle that encloses all positive examples seen so far see I have seen this positive examples. So, all of them have to be inside the rectangle by inside, we will define on the rectangle also.

(Refer Slide Time: 48:14).

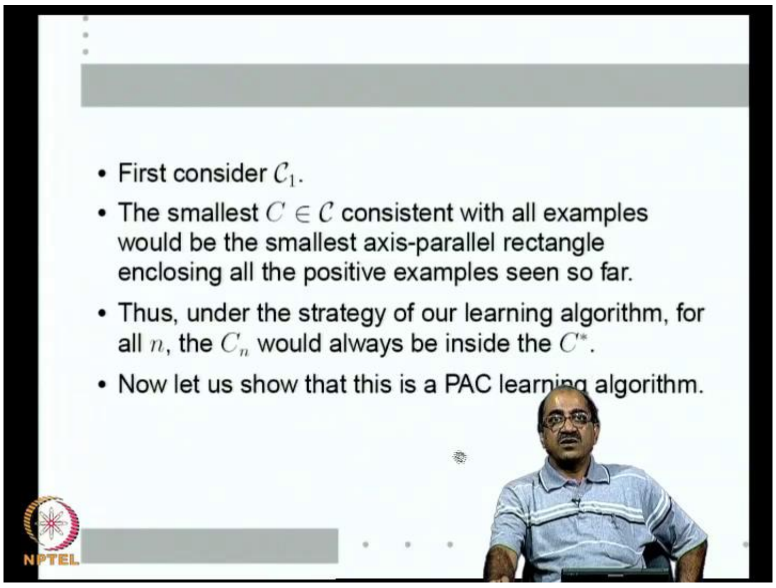

- First consider $\mathcal{C}_1$.
- The smallest $C \in \mathcal{C}$ consistent with all examples would be the smallest axis-parallel rectangle enclosing all the positive examples seen so far.
- Thus, under the strategy of our learning algorithm, for all n, the C_n would always be inside the C^*.

(Refer Slide Time: 47:37).


- The concept output by the algorithm (when using $\mathcal{C}_1$) would be the following.

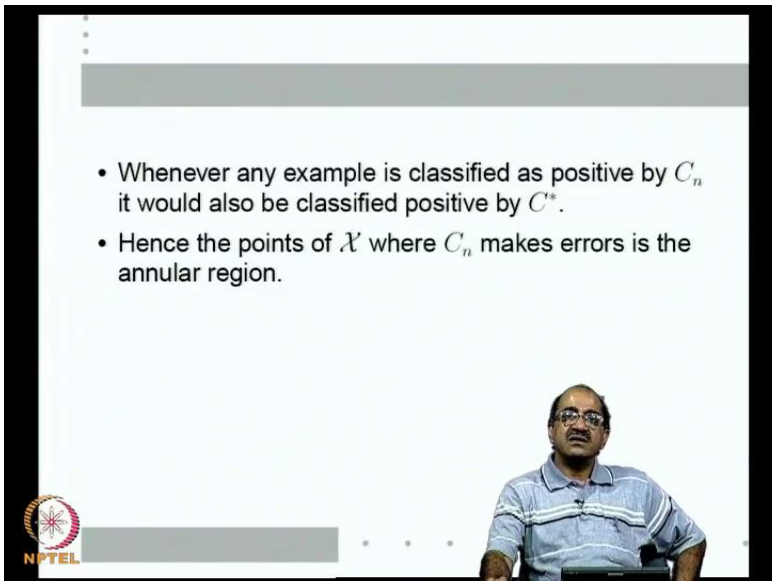
So I am looking at all possible axis parallel rectangles which contain these points and among them I am looking at the smallest one namely the smallest area, so that will be the smallest axis parallel rectangle that encloses the positive examples consider output of algorithm would be this. So, the smallest C consistent with all examples would be the smallest axis parallel rectangle enclosing all the positive examples seen so far. That means under the strategy of the learning algorithm for all n C n would always be inside C star. Because all the examples have to be also inside C star, because C star is the correct thing and C n is the smallest such enclosing rectangle. So, C n would be always inside C star.

(Refer Slide Time: 48:39)



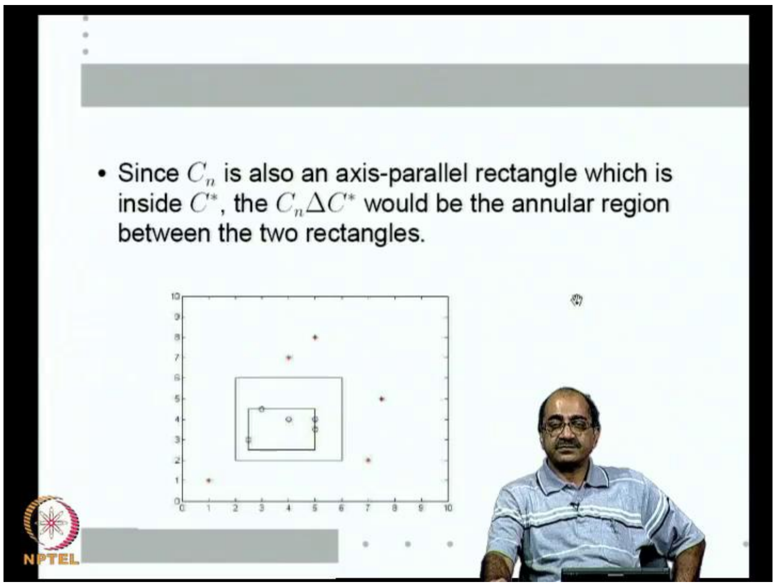
- First consider $\mathcal{C}_1$.
- The smallest $C \in \mathcal{C}$ consistent with all examples would be the smallest axis-parallel rectangle enclosing all the positive examples seen so far.
- Thus, under the strategy of our learning algorithm, for all n, the C_n would always be inside the C^*.
- Now let us show that this is a PAC learning algorithm.

(Refer Slide Time: 48:41)



- Whenever any example is classified as positive by C_n it would also be classified positive by C^*.
- Hence the points of $\mathcal{X}$ where C_n makes errors is the annular region.

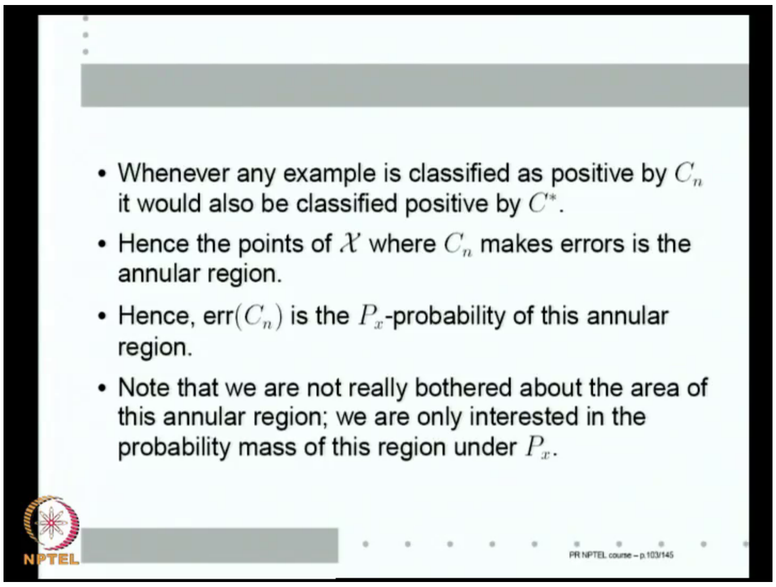
(Refer Slide Time: 49:01)



- Since C_n is also an axis-parallel rectangle which is inside C^*, the $C_n \Delta C^*$ would be the annular region between the two rectangles.

Now, let us show that this is a PAC learning algorithm whenever an example is classified as positive by C n. It would also be classified positive by C star as we just now seen C n will always be inside C star, hence the points of input space X where C n makes errors would be the annular region between the 2 rectangles. So, this is what my algorithm will learn the smallest axis parallel rectangle, this is the god given rectangle. So, only for points that falling in the annular region the classificational anything falling inside C n is also inside C star. So, the 2 classifiers 2 classification likely anything that is outside C star, once again both C n and C star will classify as negative.

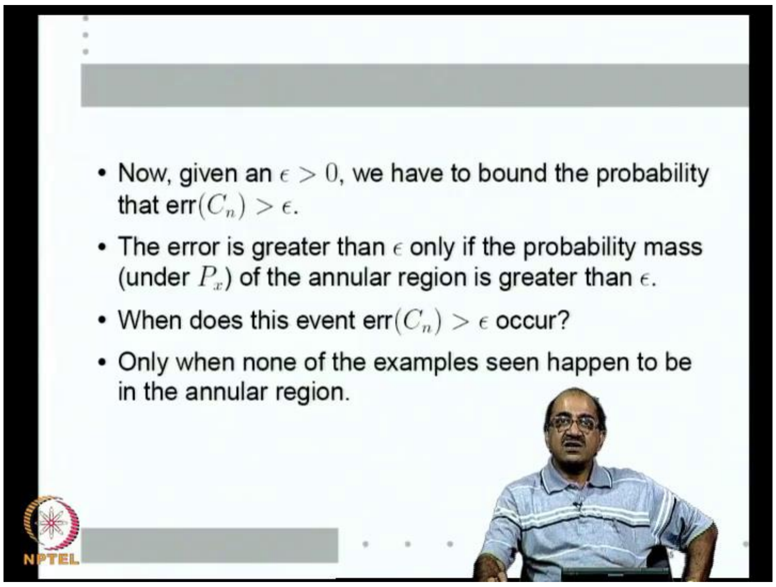
(Refer Slide Time: 49:44)



- Whenever any example is classified as positive by C_n it would also be classified positive by C^*.
- Hence the points of $\mathcal{X}$ where C_n makes errors is the annular region.
- Hence, err(C_n) is the P_x-probability of this annular region.
- Note that we are not really bothered about the area of this annular region; we are only interested in the probability mass of this region under P_x.

PR NPTEL course – p.103/145

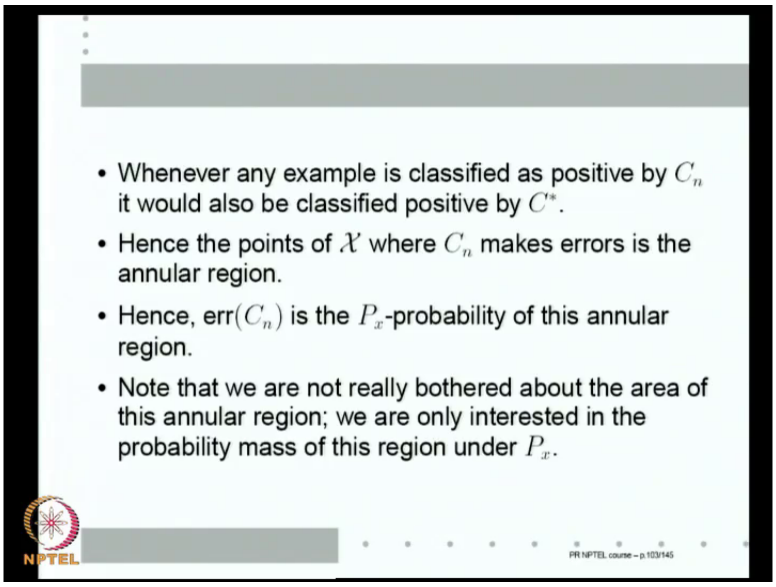
(Refer Slide Time: 50:06)



- Now, given an $\epsilon > 0$, we have to bound the probability that err$(C_n) > \epsilon$.
- The error is greater than ϵ only if the probability mass (under P_x) of the annular region is greater than ϵ.
- When does this event err$(C_n) > \epsilon$ occur?
- Only when none of the examples seen happen to be in the annular region.

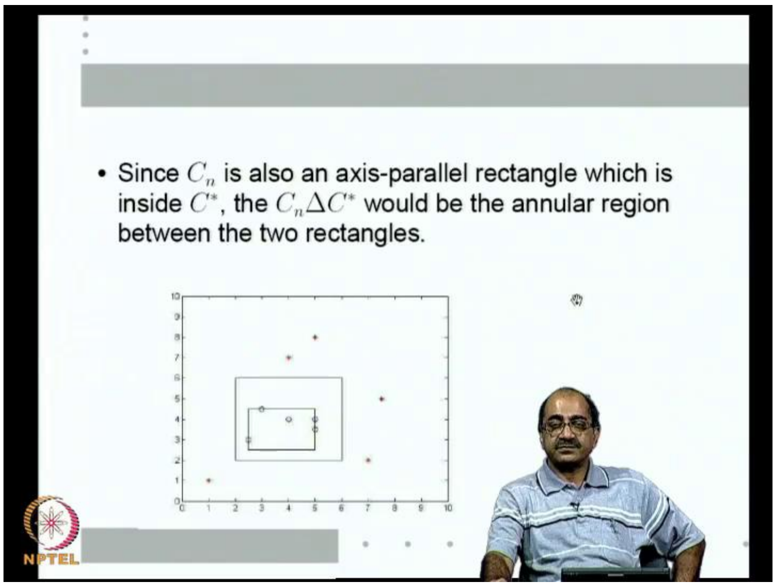
So, the only place where C n and C star will differ is this annular region between the 2 rectangles. So, since C n is also an axis parallel rectangle which is inside C star the C n delta C star, because one of them is subset of others. So, C n delta C star will be only C star minus C n that is this annular region. So, error C n is the P x probability of this annular region. Note that we are not really bothered about the area of this annular region. We are only interested in the probability mass of this region under P x that is what error C n is the idea can be anything. Given any epsilon, what we have to show do is the bound the probability that error C n is greater than epsilon that is what we have to show for probability approximately correct learning.

(Refer Slide Time: 49:44)


- Whenever any example is classified as positive by C_n it would also be classified positive by C^*.
- Hence the points of $\mathcal{X}$ where C_n makes errors is the annular region.
- Hence, err(C_n) is the P_x-probability of this annular region.
- Note that we are not really bothered about the area of this annular region; we are only interested in the probability mass of this region under P_x.

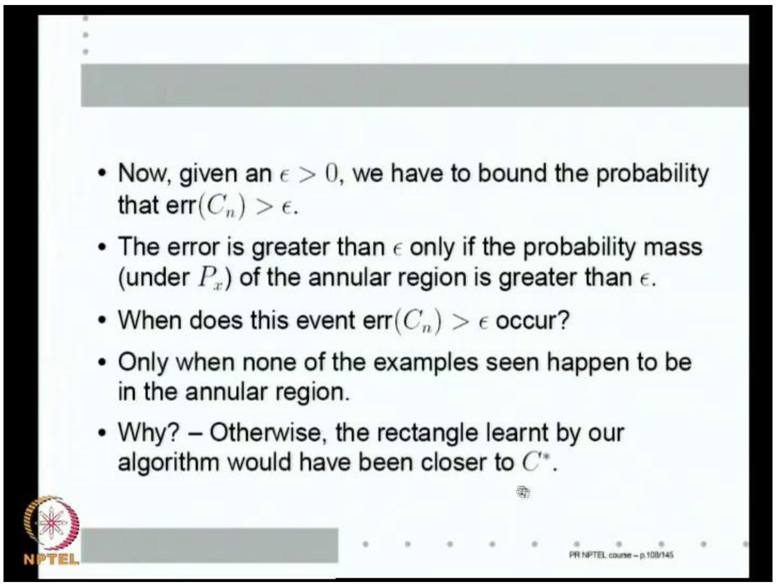
PR NPTEL course – p.103/145

(Refer Slide Time: 49:01)


- Since C_n is also an axis-parallel rectangle which is inside C^*, the $C_n \Delta C^*$ would be the annular region between the two rectangles.

The error will be greater than epsilon only with the probability mass of the annular region is greater than epsilon. So, when does this event of error C n greater than epsilon occur for a, the, the probability mass of this annular region is greater than epsilon means when none of the examples seen happen to be in the annular region. Why would that much annular region come in the first place? Because all the examples that I have seen happen to come either inside C n or outside C star, no examples fell in the annular region. Suppose, an example fell in the annular region by my strategy of finding the smallest if is fell in the annular region, it will be the positive example by my strategy of smallest rectangle enclosing all the positive examples my C n would be slightly bigger and the annular region will shrink.

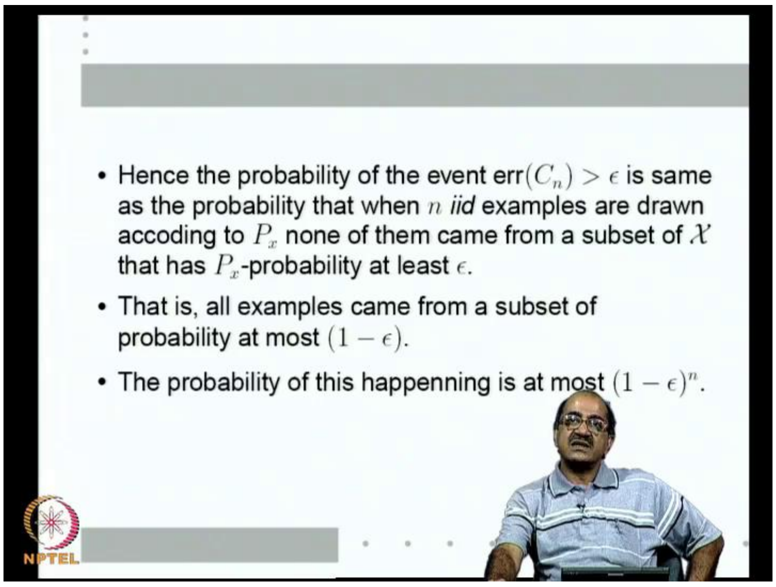
(Refer Slide Time: 51:16)



- Now, given an $\epsilon > 0$, we have to bound the probability that err$(C_n) > \epsilon$.
- The error is greater than ϵ only if the probability mass (under P_x) of the annular region is greater than ϵ.
- When does this event err$(C_n) > \epsilon$ occur?
- Only when none of the examples seen happen to be in the annular region.
- Why? – Otherwise, the rectangle learnt by our algorithm would have been closer to C^*.

PR NPTEL course – p.108/145

(Refer Slide Time: 51:31)

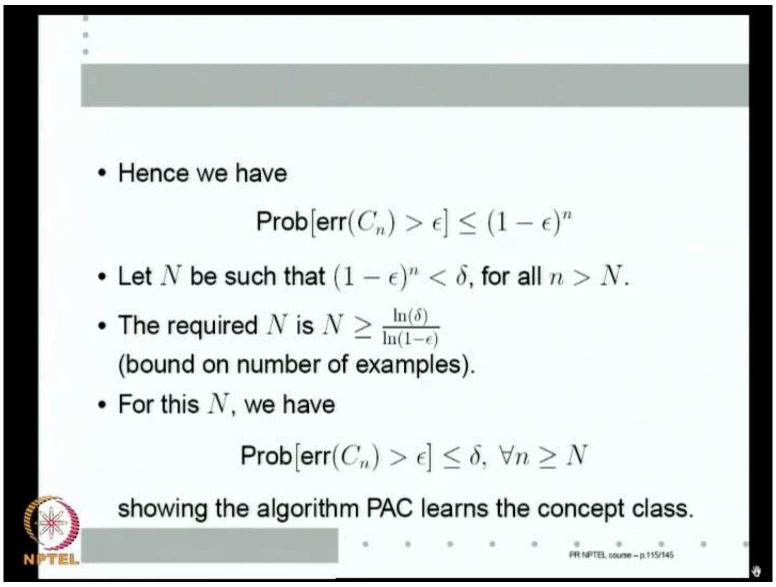


- Hence the probability of the event err$(C_n) > \epsilon$ is same as the probability that when n *iid* examples are drawn acoding to P_x none of them came from a subset of $\mathcal{X}$ that has P_x-probability at least ϵ.
- That is, all examples came from a subset of probability at most $(1 - \epsilon)$.
- The probability of this happenning is at most $(1 - \epsilon)^n$.

So, the even that I make some error is linked to none of the examples seen happens to be in the annular region why otherwise the rectangle learnt by our algorithm would have been closer to C star. So, hence the probability of the event error C n greater than epsilon error C n greater than epsilon is event, because C n is a random variable. So, the probability of the event error C n greater than epsilon is same as the probability that when n i i d examples are drawn according to P x, none of them happened to come from a subset of X that has probability of at least epsilon.

So I have seen n examples, but none of them fell in the annular region and we are saying annular region is at least epsilon probability. So, the probability that the annular region will have at least epsilon probability mass is same as when I drawn n i i d examples according to P x not even one of them came from a particular subset of X that has a probability mass of at least epsilon. That is all examples came from a subset of probability at most 1 minus epsilon. What is the probability of this happening if I have drawn n examples? All of them come from a subset with probability mass 1 minus epsilon is 1 minus epsilon to the power n.

(Refer Slide Time: 52:36)



- Hence we have

$$\text{Prob}[\text{err}(C_n) > \epsilon] \leq (1-\epsilon)^n$$

- Let N be such that $(1-\epsilon)^n < \delta$, for all $n > N$.
- The required N is $N \geq \frac{\ln(\delta)}{\ln(1-\epsilon)}$ (bound on number of examples).
- For this N, we have

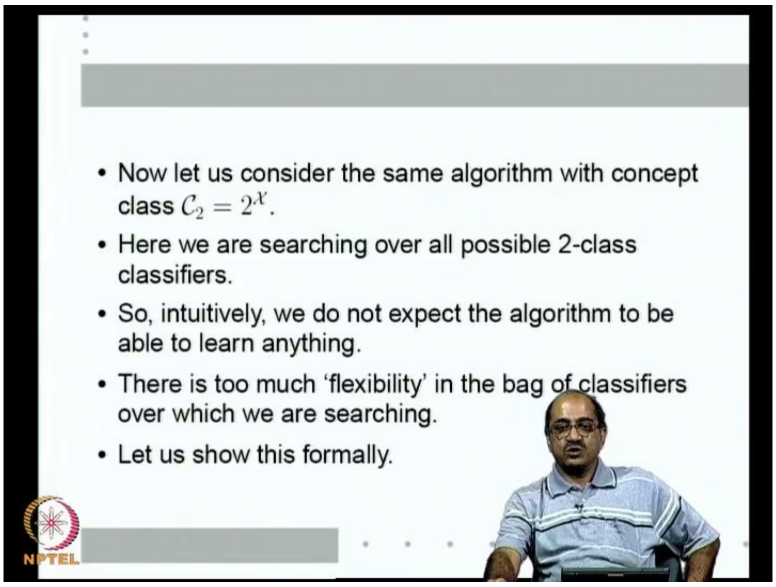
$$\text{Prob}[\text{err}(C_n) > \epsilon] \leq \delta, \ \forall n \geq N$$

showing the algorithm PAC learns the concept class.

PR NPTEL course – p.115/145

So, the probability of this happening is at most 1 minus epsilon to the power n hence what can we say probability error C n greater then epsilon is bounded above by 1 minus epsilon to the power n. Now, the rest is done, we know as n tends to infinity 1 minus epsilon to the power n goes to 0 for any epsilon greater than 0. So, I can always find a n capital N such that 1 minus epsilon to the power n is less then delta for all n greater than capital N. Specifically if I want n I can put n here and solve for it that means n should be greater than or equal to l n delta by l n 1 minus epsilon. So, these many examples so I can get a bound on examples by the time I seen these many examples, my probability of making an error more than epsilon is less then delta.

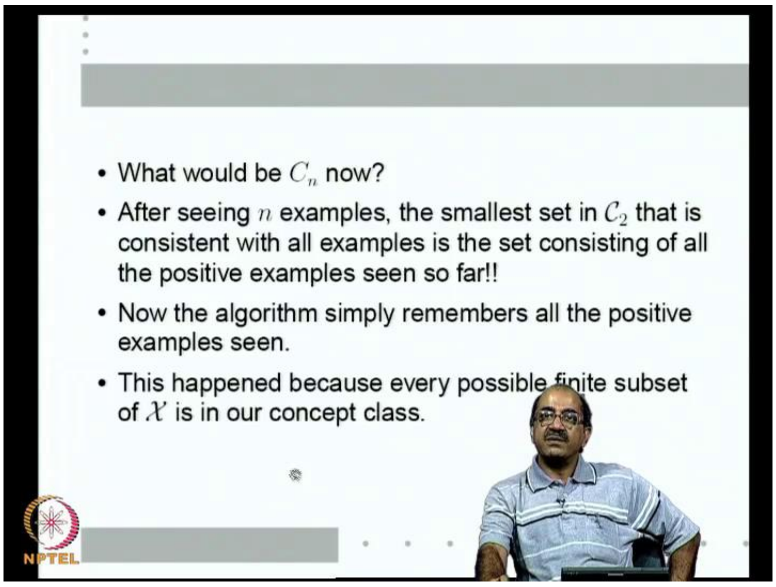
(Refer Slide Time: 53:37)



- Now let us consider the same algorithm with concept class $\mathcal{C}_2 = 2^{\mathcal{X}}$.
- Here we are searching over all possible 2-class classifiers.
- So, intuitively, we do not expect the algorithm to be able to learn anything.
- There is too much 'flexibility' in the bag of classifiers over which we are searching.
- Let us show this formally.

So, for this n we have probability error C n greater than epsilon is less then delta for all n greater than n no matter what C star is and that shows that the algorithm PAC learns the concept class. Now, let us go back to the other one, let us say we work with C 2 is equal to 2 power X here we are searching over all possible 2 class classifiers. So, obviously we, we do not expect the algorithm to learn anything, we are learning over every possible thing. It is very difficult to learn that is because there is too much flexibility in the bag of classifiers over which you are searching, let us show this formally in this class now.

(Refer Slide Time: 54:03)

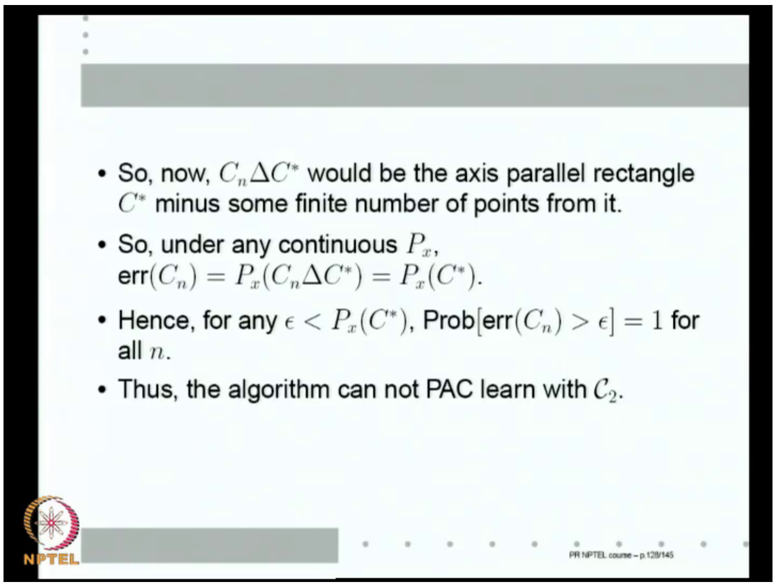


- What would be C_n now?
- After seeing n examples, the smallest set in $\mathcal{C}_2$ that is consistent with all examples is the set consisting of all the positive examples seen so far!!
- Now the algorithm simply remembers all the positive examples seen.
- This happened because every possible finite subset of $\mathcal{X}$ is in our concept class.

So, for the second one when I am taking my concept class to be 2 power X, what will be C n? After seeing n examples, what is the smallest subset that is consistent with all examples? Because every possible subset of X is in my bag after seeing n examples the smallest C element of C 2 that is consistent with all examples is simply the set finite set consisting of all the positive examples seen so far that is the smallest set if I say any of this. If X is any of these points is positive otherwise it is negative that is consistent with all examples.

So, this algorithm simply remembers all positive example, because I am taking this smallest set in my bag C 2 that is consistent through all examples the smallest set is nothing. But the set of all positive examples seen so far this happen here only, because every possible finite set is also in our concept class in the earlier, it could not happen because our concept class contained only axis parallel rectangles. So, I could not give this as the smallest set, but here, because every possible subset is there I will never learn anything more than remembering the set of positive examples.

(Refer Slide Time: 55:13)



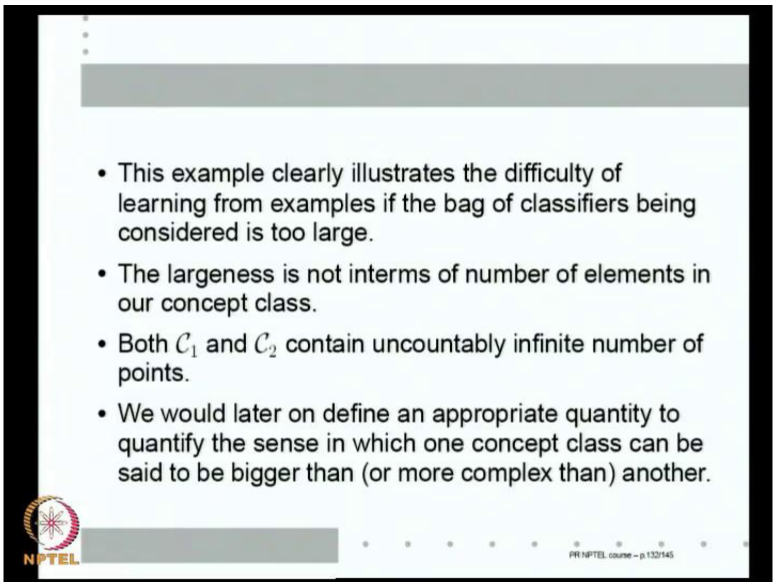
- So, now, $C_n \Delta C^*$ would be the axis parallel rectangle C^* minus some finite number of points from it.
- So, under any continuous P_x, err$(C_n) = P_x(C_n \Delta C^*) = P_x(C^*)$.
- Hence, for any $\epsilon < P_x(C^*)$, Prob[err$(C_n) > \epsilon$] $= 1$ for all n.
- Thus, the algorithm can not PAC learn with $\mathcal{C}_2$.

PR NPTEL course – p.128/145

So, this; obviously, does not PAC learn, so C n delta C star would be what once again C n is a subset of C star, because C n is the positive set of positive examples. So, it is only C star minus C n that is C star minus some finite number of points from it. So, if I take some finite number point makes no difference. For example, P x is a continuous distribution error C n is P x C n delta C star which is same as P x C star, because C star minus some finite number of points. So, which is still be P x C star which means if I take any epsilon that is less than P x C star probability error C n is greater than epsilon is one for all n no matter how large n is.

Hence this algorithm does not PAC learn I hope you can appreciate the example my learning strategy is same I am just looking in my bag to find a classifier that is consistent with all examples and it is the smallest among all such consistent classifier. But with the same strategy depending on whether I am searching on a nice bag or a too flexible a bag at one end I I I I can PAC learn other end I cannot.

(Refer Slide Time: 56:26)

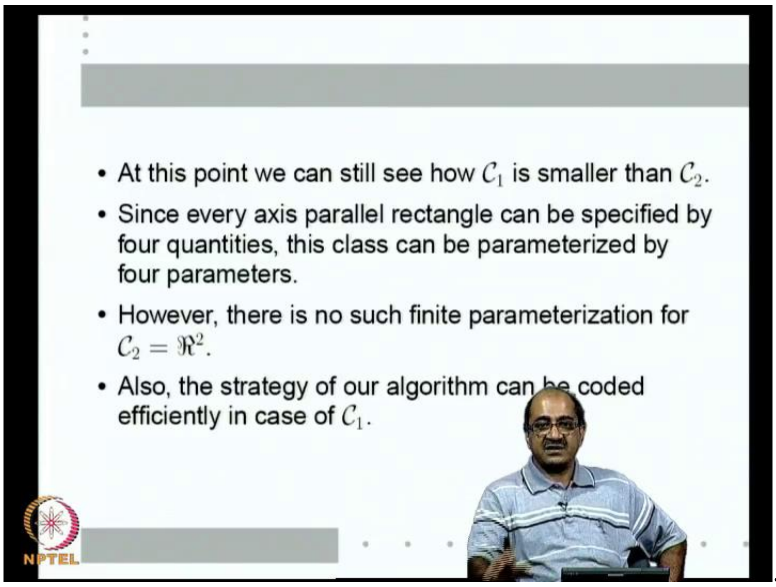


- This example clearly illustrates the difficulty of learning from examples if the bag of classifiers being considered is too large.
- The largeness is not interms of number of elements in our concept class.
- Both $\mathcal{C}_1$ and $\mathcal{C}_2$ contain uncountably infinite number of points.
- We would later on define an appropriate quantity to quantify the sense in which one concept class can be said to be bigger than (or more complex than) another.

PR NPTEL course – p.132/145

So, example clearly illustrate the difficulty of learning from examples the bag of classifiers consist too large. So, normally all of us know that we need sufficient number of examples representative examples and so on. But we do not we are not so conscious that it also depends on the bag in which I search it as a matter of fact in with this 2 power X as it is easy to see from our proof. No matter how many examples you give me I still cannot learn the concept, so is not just having sufficient number of examples I should also be searching over a bag that is not too flexible that is not true fuse. Of course, this largeness between C 1 and C 2 is not in the terms of in terms of number of elements in these 2 concept classes after all both C 1 and C 2 contain uncountably infinite number of points.

(Refer Slide Time: 57:18)

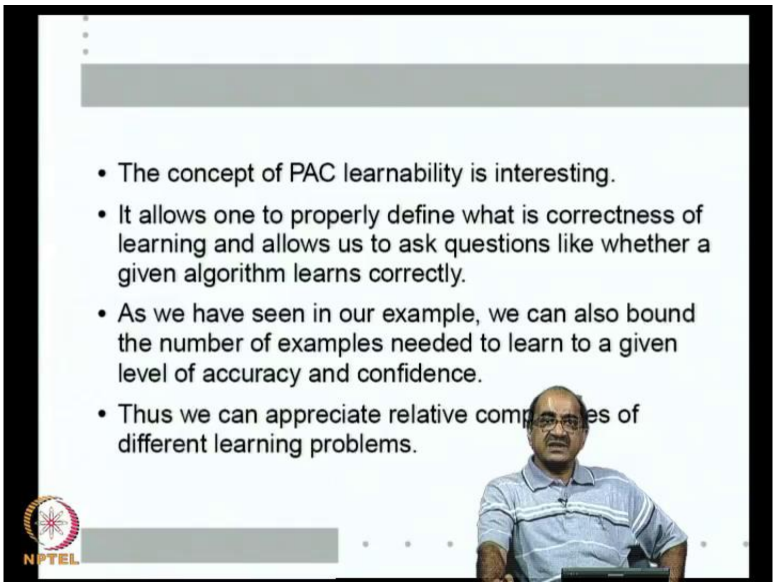


- At this point we can still see how $\mathcal{C}_1$ is smaller than $\mathcal{C}_2$.
- Since every axis parallel rectangle can be specified by four quantities, this class can be parameterized by four parameters.
- However, there is no such finite parameterization for $\mathcal{C}_2 = \Re^2$.
- Also, the strategy of our algorithm can be coded efficiently in case of $\mathcal{C}_1$.

.

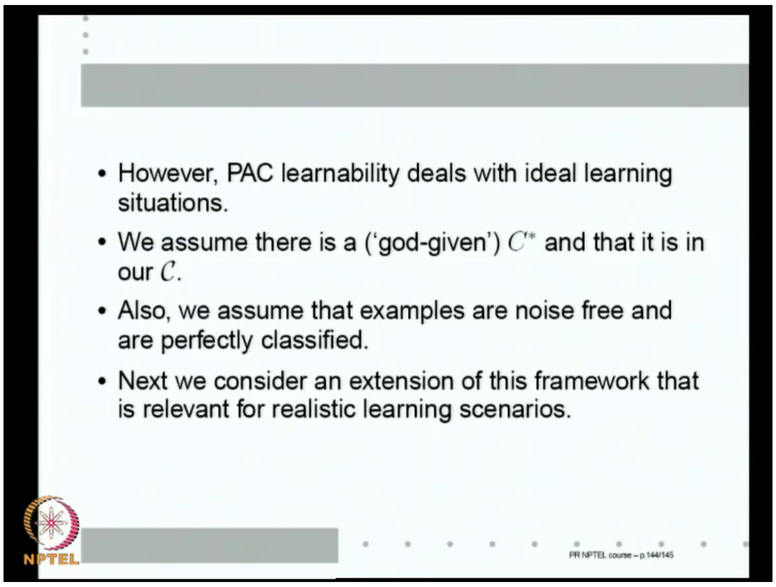
We would later on define a proper quantitative measure to say when we can say one concept class is more complex than the other. But now, we can say this following every axis parallel rectangle can be specified by four quantities. If you give me the left bottom vertex and the top vertex, those are four real numbers a axis parallel rectangle is uniquely defined which means it can be parameterized by four parameters, whereas I cannot finitely parameterize the set of all possible outset of R 2. In that sense the axis parallel rectangle say much simpler concept class than the other also of course, given that I only need these 4 numbers by finding mean next max X mean by max y among all positive examples I can very easily calculate the, the smallest axis parallel rectangle and it is very efficient.

(Refer Slide Time: 58:05)



- The concept of PAC learnability is interesting.
- It allows one to properly define what is correctness of learning and allows us to ask questions like whether a given algorithm learns correctly.
- As we have seen in our example, we can also bound the number of examples needed to learn to a given level of accuracy and confidence.
- Thus we can appreciate relative comp[illegible]es of different learning problems.

(Refer Slide Time: 58:21)



- However, PAC learnability deals with ideal learning situations.
- We assume there is a ('god-given') C^* and that it is in our $\mathcal{C}$.
- Also, we assume that examples are noise free and are perfectly classified.
- Next we consider an extension of this framework that is relevant for realistic learning scenarios.

PR NPTEL course – p.144/145

So, the concept of PAC learning is interesting, because you know it allows us to define properly what the correctness of learning algorithm. It allows us to bound the number of examples tells you the relative complexities of different learning problems. However it deals with very ideal learning situations, we have a god given classifier then perfectly classifier noise free. So, this is then, so that we get our teeth into the problem. So, next class, we will try to extend this formalism, so that we can look at more, more realistic learning situations. Thank you.