

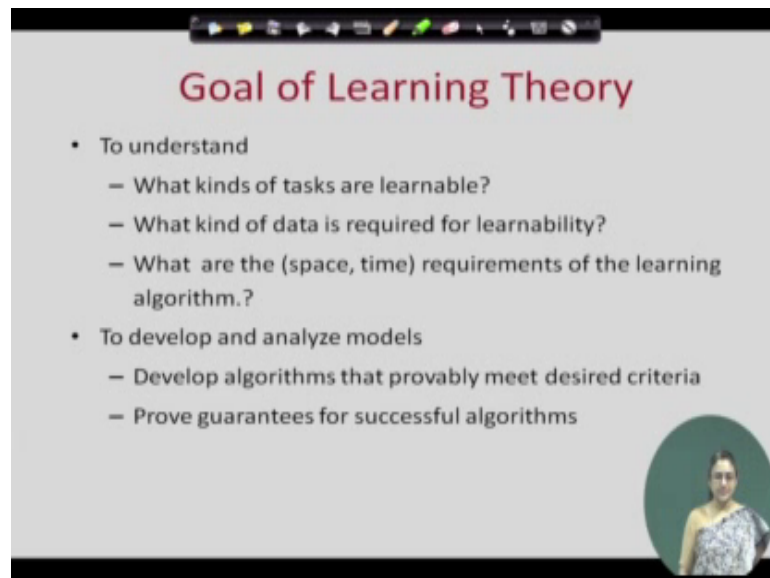
Introduction to Machine Learning
Prof. Sudeshna Sarkar
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Module – 7
Lecture - 32
Introduction to Computational Learning Theory

Good morning. Today, we start with module 7. In this module, we will talk about computational learning theory, and also Ensembles learning.

The first, we will start with computational learning theory. In the first part of the lecture, we will talk about the learning model that we will use, which is PAC model, and we will look at situations where we have a finite hypothesis space.

(Refer Slide Time: 00:53)

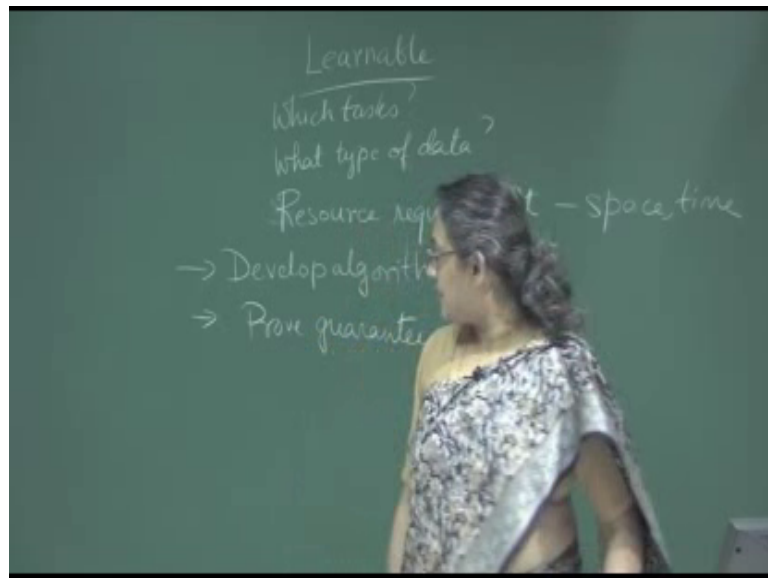


Goal of Learning Theory

- To understand
 - What kinds of tasks are learnable?
 - What kind of data is required for learnability?
 - What are the (space, time) requirements of the learning algorithm.?
- To develop and analyze models
 - Develop algorithms that provably meet desired criteria
 - Prove guarantees for successful algorithms

So, the goal of learning theory, so far in this course, we have looked at several learning algorithms. Today, we will try to look at in brief, in the basic underlying theory of computational machine learning. The goal of learning theory is to understand what is learnable.

(Refer Slide Time: 01:14)



So, we have looked at different tasks, different learning tasks. We want to understand what type of tasks can be learned; and so, which tasks; and in order to learn those tasks, what type of data is required. For example, we will talk about, how much data is required for a particular learning task; or what is the number of samples, size of the training set, which we call sample complexity.

So, there is a task, and there is a data; we want to understand which tasks are learnable, and in order to learn them, what size of data is required. We will also look at the resource requirement for an algorithm, if the task is learnable using some data, and that data is available to you. The algorithm has some resource requirement, that is, the space and time complexity of the algorithm that is required to solve the learning task, given the data.

So, this is our goal of learning theory. Also, we want to develop an analysis model. So, we want to develop these algorithms, and these algorithms must meet our desired criteria. So, we will define criteria, under which, we will say an algorithm is successful, and we will have to develop algorithms that meet those criteria. And we have to provide. So, algorithms that is provable. So, we have to provide, or prove guarantees. So, what sort of guarantees, or confidence, we have on that algorithm.

So, these are the things which we need to study, when we talk about machine learning theory. Now, we have seen earlier that there are 2 core aspects of machine learning. So, one aspect is designing algorithm, right. So, in machine learning, we design algorithms.

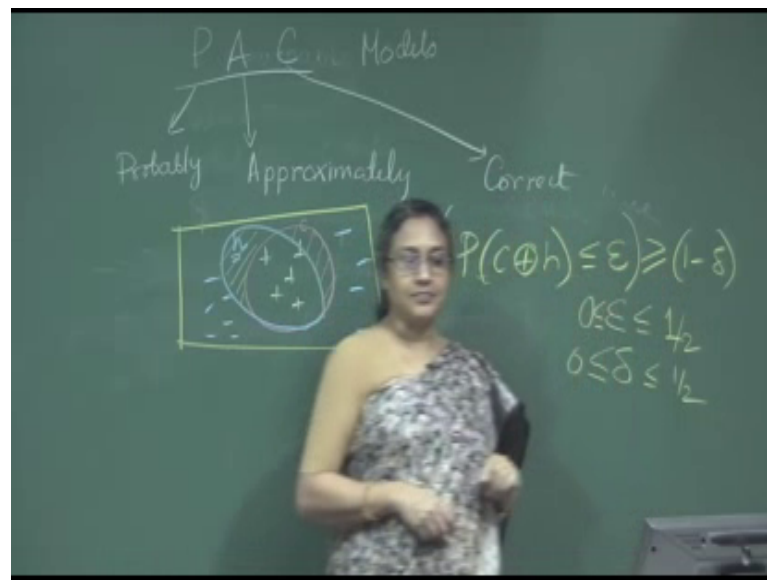
(Refer Slide Time: 03:55)



And we usually design algorithms that optimize certain criteria. So, we are given some data; we assume a hypothesis space; and that algorithm finds a hypothesis, from the hypothesis space that is optimizing certain criteria. So, this is the most machine learning algorithms are of this type. So, one is to design algorithms; second is, we have to find out what is the confidence on this algorithm. So, in the sense that we want to know how well this algorithm will work on future data instances; that is what the generalization ability of the algorithm? So, given that we train the algorithm on the training data, how well it will generalize and perform on future data?

Now, in order to formally talk about this task, we will require some setting, or learning model. So, the second that we will talk about this, in this class, is the PAC setting. PAC stands for PAC. P stands for probably; A stands for approximately; and C stands for correct. So, and we will study PAC models. So, scenarios where we get an, get a model, that is probably approximately correct; and we will explain what we mean by that. So, very simply, let us denote that this is our universe, or hypothesis.

(Refer Slide Time: 06:05)



This is our universe, which contains the different instances.

Suppose, c is the target function which we wish to learn, so this is the instance space, and the instances which are within c are members of the class. And we want to learn c ; c is the ideal class that we wish to learn. And in order to learn c , we have got some training examples, which are labeled as positive, or negative. So, we have given this label. We have got positive and negative examples. So, we want to come up with the hypothesis h . So, c is the original, actual hypothesis, which comprises, the positive examples are within c ; the negative examples are outside c . We have a sample from the instance space, as part of our training data and we want to come up with the hypothesis h .

Now, h may not be exactly equal to c . So, what is the error region? The error region is given by $c \text{ XOR } h$; this is an error region. c says that these instances are within the concept; but h does not include them. This region is also error, c says, outside the concept, but h says, it is inside the concept. So, this region and this region are the error regions. This region and the outer region, they are the correct region. So, this $c \text{ XOR } h$ is the error region, and we want the probability of this error region to be small. So, we want, we want probability of $c \text{ XOR } h$ to be less than equal to epsilon.

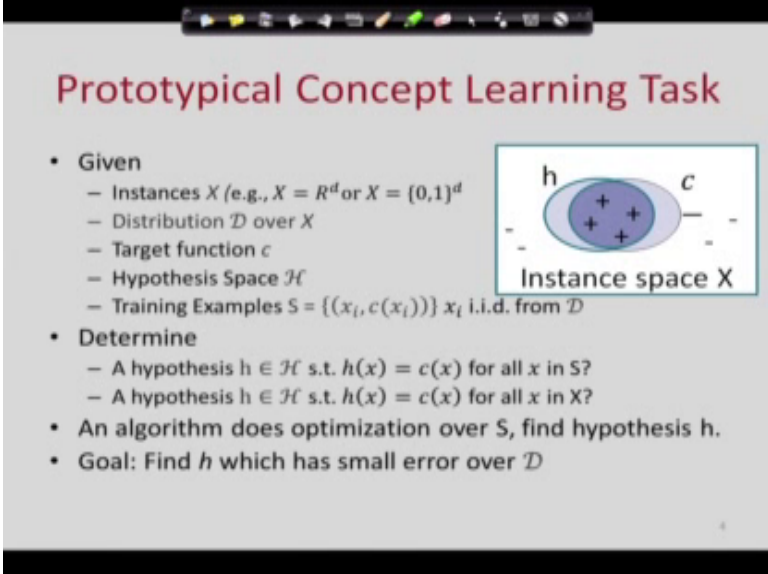
Ideally, we want c to be, h to be equal to c , but sometimes; it will not be possible to get c equal to h . So, we want to bound the probability that an instance will occur in this region. So, a hypothesis is said to be approximately correct, if the error is less than equal to

epsilon, where epsilon is a parameter; epsilon is between 0 and 1. Or, we can say, epsilon is between zero and half. So, epsilon is the error parameter, and if for a given epsilon, the probability of this error region is less than equal to epsilon then we say, the hypothesis is approximately correct.

Now, there would be certain cases where we may not be, you know, some situations, a learning algorithm will mostly expected to give us approximately correct hypothesis, but there may be certain situations where the learning algorithm may not give us the correct hypothesis, which is epsilon approximate. So, the probability, we want to bound the probability that the learning algorithm gives us an approximately correct, correct hypothesis. This probability must be greater than 1 minus delta, right.

So, probability that we get an approximately correct hypothesis is greater than 1 minus delta. So, delta is called the confidence parameter. Again, delta can be between zero and half. So, this, the probability with which we get an approximately correct hypothesis, this is called the confidence. This is controlled by the delta parameter, right. We can set delta small, let us say, 0.01, then, we can say with 99 percent, wait, for 0.99 probabilities we get an approximately correct hypothesis. And what is approximately correct will be, will depend on epsilon.

(Refer Slide Time: 10:55)



Prototypical Concept Learning Task

- **Given**
 - Instances X (e.g., $X = \mathbb{R}^d$ or $X = \{0,1\}^d$)
 - Distribution \mathcal{D} over X
 - Target function c
 - Hypothesis Space \mathcal{H}
 - Training Examples $S = \{(x_i, c(x_i))\}$ x_i i.i.d. from \mathcal{D}
- **Determine**
 - A hypothesis $h \in \mathcal{H}$ s.t. $h(x) = c(x)$ for all x in S ?
 - A hypothesis $h \in \mathcal{H}$ s.t. $h(x) = c(x)$ for all x in X ?
- An algorithm does optimization over S , find hypothesis h .
- Goal: Find h which has small error over \mathcal{D}

The diagram shows the Instance space X as a large rectangle. Inside, a blue oval represents the hypothesis h , and a red oval represents the target function c . The hypothesis h contains several '+' signs, while the target function c contains several '-' signs.

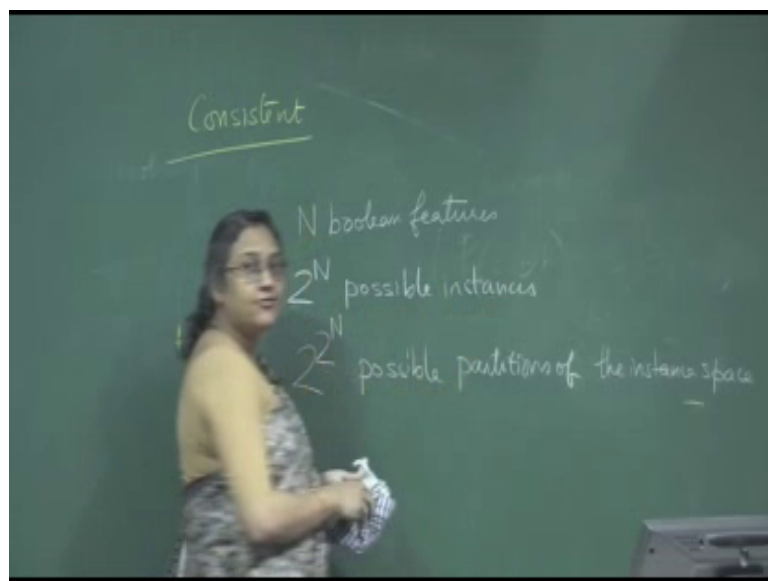
Now, in a prototypical concept learning task, we have talked about concept learning, in all the earlier classes. In a concept learning task, we have instances x , and x for example,

can come from, if your features are real numbers, x can come from \mathbb{R} to the power d , or if the features are Boolean, x can come from zero 1 to the power d , where d is the number of features, or the dimensions of the instance space. So, X is our instance space, and we have a distribution d over the instance space. So, X is our instance space, and there is a distribution d over the instance space; and let us see with the target function, and capital H is the hypothesis space.

And we have training examples x_i, y_i , and because c is the actual concept, the y_i is equal to $c(x_i)$. So, the training example, S is the set of training examples, which comprises tuples $x_i, c(x_i)$; and these training examples are drawn from the distribution d , and they are identically, independently drawn. So, they are i.i.d from distribution d . So, we have a instance space, which consists of all the instances. This is the instance space, and we have a target concept c , which we want to learn, and we have a hypothesis space capital H . We want to find a hypothesis small h which belongs to capital H , given the training examples S . The training examples are given by plus and minus. They are labeled as either plus or minus.

The learning task is to determine a hypothesis small h belonging to the space, such that you know, we will first look at the scenario, where given the training examples, you find a hypothesis, which agrees with the training example. That is, for the training examples, it produces the correct labeling. Such hypothesis are called consistent hypothesis.

(Refer Slide Time: 13:29)



We will also see that in certain cases, we cannot look for consistent hypothesis, and we will allow inconsistent hypothesis. So, we can try to find out a hypothesis which agrees with the labeling; the labeling that the hypothesis produces, agrees with the labeling in the training set.

Ideally, we really want a hypothesis which produces the correct output for all instances. But, if we do not have the labeling of all instances, we can never be sure that the hypothesis that we come up with will actually label all instance correctly. But, we can definitely try to find a hypothesis, if one hypothesis exists, which labels the training samples correctly. So, we are given the training sample S , and the algorithm that we will design will optimize over S , and come up with the hypothesis small h .

Our real goal is to find a hypothesis which is highly accurate with respect to the distribution d over the instance space. So, we ideally want a hypothesis which has small error over the distribution, over the instances, but in reality we do not have all the instances; we have a sample; and based on that we have to come up with the hypothesis.

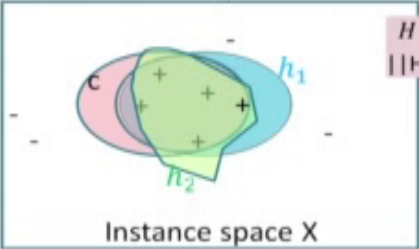
So, we do not know whether the hypothesis will actually label the, all the instances correctly. But, we have to have some idea about how well the hypothesis will generalize. If we really have to absolutely guarantee that the hypothesis is the correct hypothesis, it must label all instances correctly, and for that you have to see all the examples. There is no shortcut to this. This is often called no free lunch, but we have seen that when we are doing inductive inference, we often use some types of bias, so that we can come up with a hypothesis. For example, we can put a bias on the form of the hypothesis class, so that we only allow hypothesis of certain restricted type. This is called restriction bias, which we have discussed earlier.

So, in inductive inference, where we have to generalize beyond the training data, we will need a bias to decide which hypothesis to come up with. That bias can be a restriction bias, which restricts the hypothesis space to simple hypothesis, or it could be a preference bias, which says, within a hypothesis space, certain hypothesis are preferable over other hypothesis. For example, when we say, we prefer small decision trees over large decision a tree that is a preference bias.

(Refer Slide Time: 16:44)

Function Approximation

- How many labeled examples in order to determine which of the 2^{2^N} hypothesis is the correct one?
- All 2^N instances in X must be labeled!
- Inductive inference: generalizing beyond the training data is impossible unless we add more assumptions (e.g., bias)

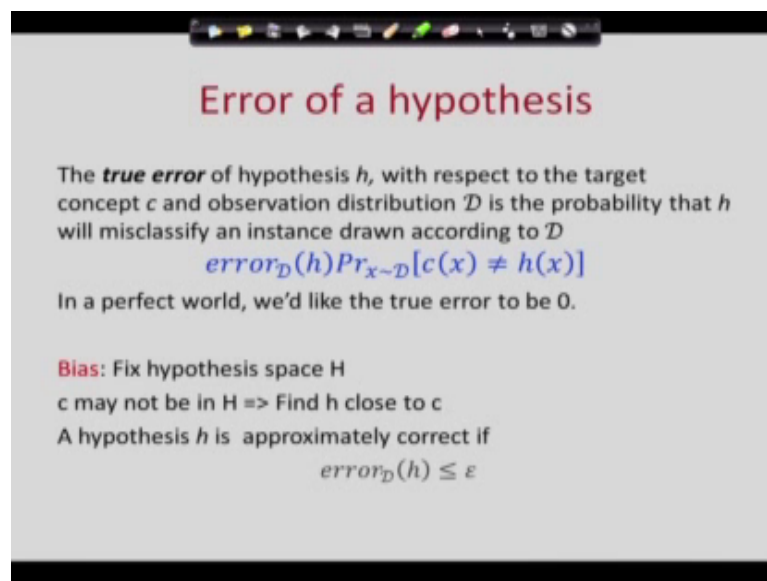


The diagram shows an instance space X containing several regions. A target region c is highlighted in pink. Two hypothesis regions, h_1 (blue) and h_2 (green), are shown. h_1 is a large region containing c , while h_2 is a smaller region that overlaps with c . The regions are partitioned into positive (+) and negative (-) areas. A pink box on the right contains the text: $H = \{h: X \rightarrow Y\}$ and $|H| = 2^{|X|} = 2^{2^N}$.

Now, so, basically, the problem in learning theory is about function approximation. Suppose, we have N features, N Boolean features and we have a hypothesis. So, we have N Boolean features. If you have N Boolean features, how many possible instances are there? So, each of these features can be marked as true or false. So, that is 2 to the power N possible instances. A particular, take any hypothesis h_i . So, what h_i will do is that it will say, a subset of these instances are positive, the rest are negative. So, a hypothesis corresponds to a partition of the instance space into positive and negative.

How many ways can we find subsets of the instance space, of the size 2 to the power N ? There are 2 to the power 2 to the power N possible subsets of 2 to the power instances. That is, there are 2 to the power 2 to the power N possible partitions of the instance space. And we have to choose among them. Now, if we really want to make sure that given the training set, only one of these is correct, we have to see all the 2 to the power N instances. Without seeing all the instances and their labels, we will not be accurately able to pin point exactly one hypothesis from here. So, the set of the hypothesis space is 2 to the power 2 to the power N , and ideally, in order to fix one hypothesis, exactly one hypothesis, which matches our c , we have to look at all the instances.

(Refer Slide Time: 19:20)



Error of a hypothesis

The **true error** of hypothesis h , with respect to the target concept c and observation distribution \mathcal{D} is the probability that h will misclassify an instance drawn according to \mathcal{D}

$$\text{error}_{\mathcal{D}}(h) = \Pr_{x \sim \mathcal{D}}[c(x) \neq h(x)]$$

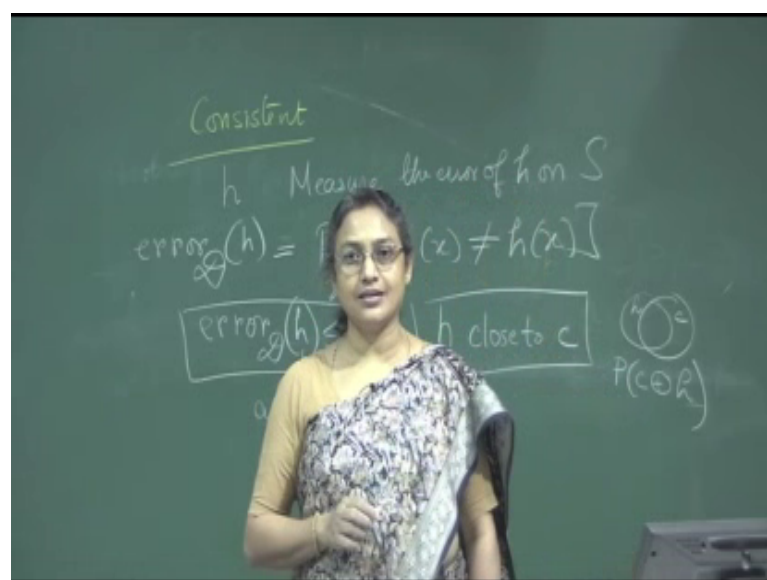
In a perfect world, we'd like the true error to be 0.

Bias: Fix hypothesis space H
 c may not be in $H \Rightarrow$ Find h close to c
A hypothesis h is approximately correct if

$$\text{error}_{\mathcal{D}}(h) \leq \epsilon$$

But, which it is not always possible to do that because; this requires exponential number of examples. We cannot do it in polynomial time. So, it is not practical to do it. So, we will come up with the hypothesis by looking at fewer instances. So, we will only see the labeling of a sample of the instance, and based on that labeling, we will come up with the hypothesis.

(Refer Slide Time: 19:55)



Consistent

Let h be a hypothesis. Measure the error of h on S

$$\text{error}_S(h) = \Pr_{x \in S}[c(x) \neq h(x)]$$

$\text{error}_S(h) \leq \epsilon \Rightarrow h$ close to c

$P(c \in H)$

So, we come up with the hypothesis h , and we can measure the error of h on S . S is our training data. On S , we can measure error on h . But, we really want to know, what is the

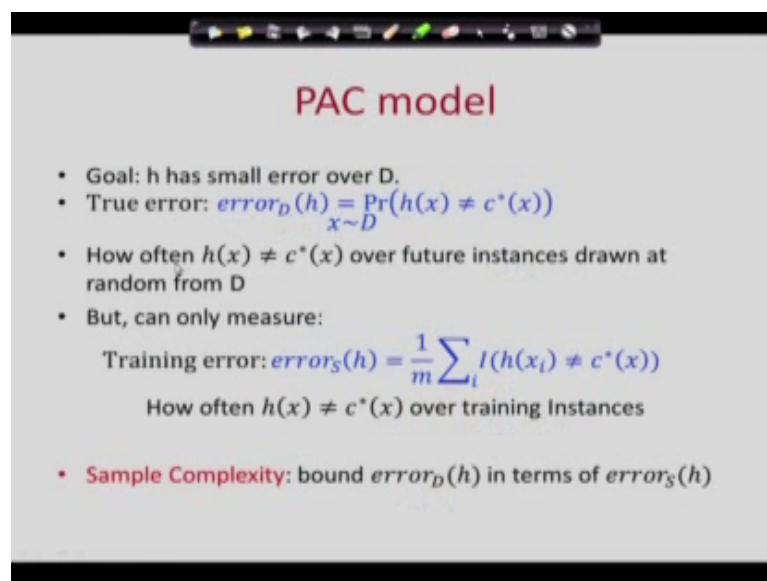
true error of the h , and we want to really, the learning problem is to minimize the true error of the h . But, we can only find out the sample error of h . So, the true error of the h is denoted by error of h , according to the distribution D . And this we can define as the probability that $c(x)$ and $h(x)$ do not agree, where x is drawn according to the distribution D , right. So, this is the true error of h .

And we cannot really find the true error; we can find the sample error, for the training examples, or for another sample, for which we have the labeled values. So, in fact, if h is the hypothesis space, we are trying to find small h belonging to capital H . But, it is possible that c possible, that c may not be in H ; c may not be in H is possible. But, what we really want is that we want at least h to be close to c . So, h must be close to c , this is what we want; and this is what we depicted by that diagram, saying that if this is c , this is h , we look at the error region of c and h , and we want this to be small.

And so, c has zero error, zero true error; what we want is an h , so that that error of h , with respect to the distribution, is less than equal to epsilon. So, we want $\text{error}_D(h)$ less than equal to epsilon, and as we have said, we call a hypothesis, if under this condition; this hypothesis is called approximately correct.

Now, the goal is that h has small error over the distribution.

(Refer Slide Time: 23:02)



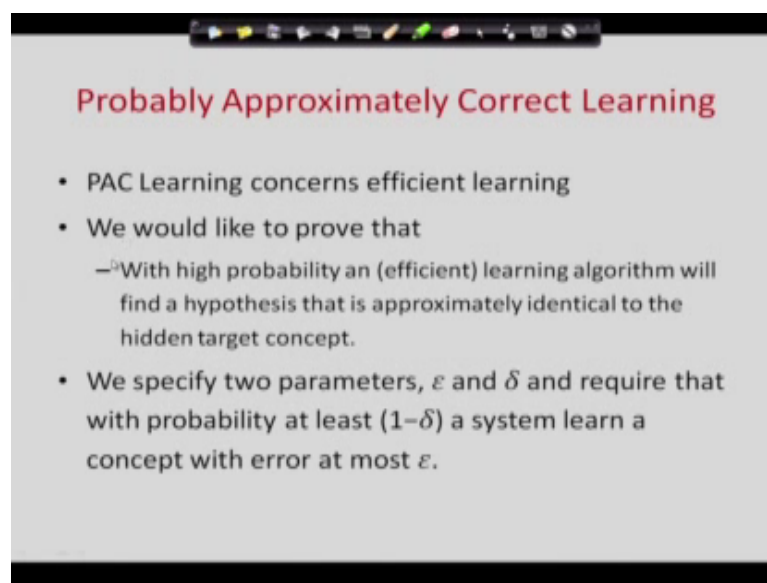
PAC model

- Goal: h has small error over D .
- True error: $\text{error}_D(h) = \Pr_{x \sim D}(h(x) \neq c^*(x))$
- How often $h(x) \neq c^*(x)$ over future instances drawn at random from D
- But, can only measure:

Training error: $\text{error}_S(h) = \frac{1}{m} \sum_i I(h(x_i) \neq c^*(x_i))$
 How often $h(x) \neq c^*(x)$ over training instances
- **Sample Complexity:** bound $\text{error}_D(h)$ in terms of $\text{error}_S(h)$

Now, we want to know, how often $h(x)$ and $c(x)$ differ over future instances. The sample error, this is the true error, the sample error is defined over the training set. Let us say, training set has size m , and over all the training set, we can find out how many of them do not, the label of h and c do not match; and $1/m$ is the average error. So, this is the training error. And we want to; we can find the training error. So, what we now will look at, how to bound the true error in terms of the training error. So, if I get a particular training error, how can we estimate how much the true error can be, or what is the bound of the true error; that is what we will like to look at.

(Refer Slide Time: 24:01)



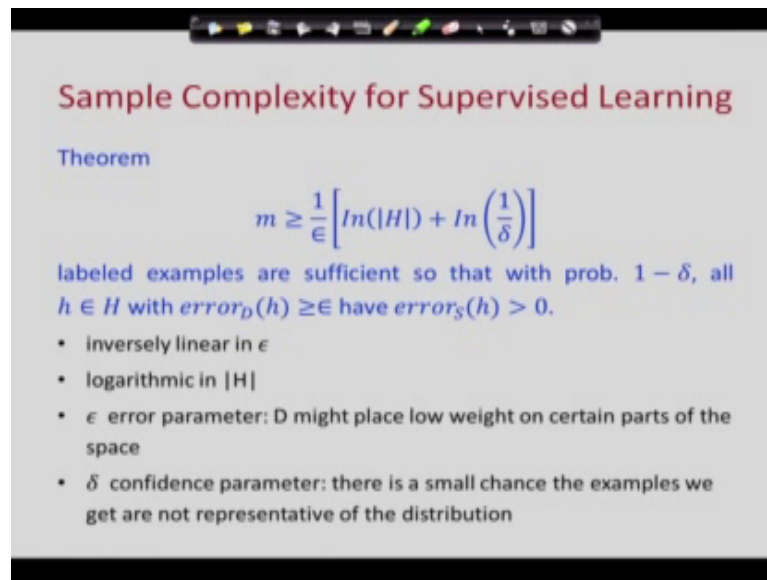
Probably Approximately Correct Learning

- PAC Learning concerns efficient learning
- We would like to prove that
 - With high probability an (efficient) learning algorithm will find a hypothesis that is approximately identical to the hidden target concept.
- We specify two parameters, ϵ and δ and require that with probability at least $(1-\delta)$ a system learn a concept with error at most ϵ .

Now, PAC learning, as I said, Probably Approximately Correct learning, that concerns efficient learning. We will like to prove that with high probability, that is, with probability greater than equal to $1 - \delta$, an efficient learning algorithm, we will tell what does efficient mean, an efficient learning algorithm will find a hypothesis that is approximately identical to the target concept. Approximately identical means, the error is less than equal to ϵ . And we want an efficient algorithm, that is, an algorithm whose running time and space requirement is polynomial in certain things; polynomial in certain parameters. What those parameters are, we will look at.

So, for this Probably Approximately Correct learning, we require 2 parameters ϵ and δ . And as we said, we require that with probability at least $1 - \delta$, a system learns a concept with error at most ϵ .

(Refer Slide Time: 25:15)



Sample Complexity for Supervised Learning

Theorem

$$m \geq \frac{1}{\epsilon} \left[\ln(|H|) + \ln\left(\frac{1}{\delta}\right) \right]$$

labeled examples are sufficient so that with prob. $1 - \delta$, all $h \in H$ with $error_D(h) \geq \epsilon$ have $error_S(h) > 0$.

- inversely linear in ϵ
- logarithmic in $|H|$
- ϵ error parameter: D might place low weight on certain parts of the space
- δ confidence parameter: there is a small chance the examples we get are not representative of the distribution

So, this is the theorem which says that if the size of the training sample m is greater than equal to $\frac{1}{\epsilon} \log$ natural logarithm of the size of the hypothesis space, plus $\log \frac{1}{\delta}$, then, if we have that many number of training examples, it is sufficient to be able to PAC learn a hypothesis. We can PAC learn a hypothesis, if we are given so many examples. If you are given examples larger than this, and we come up with the hypothesis that is consistent with the training examples, that hypothesis will be probably approximately correct with the parameters δ and ϵ .

So, m , as we see is, inverse linear in ϵ , logarithmic in the size of the hypothesis space, inverse, linear in $\log \frac{1}{\delta}$. So, this is the sample complexity for learning. You require this number of exam; any algorithm will require at least these numbers of examples, to guarantee that the task can be learned. Now, we will look at the proof of this theorem. So, suppose, we have a target concept, and this hypothesis space, the hypothesis space is the set of hypotheses; and this hypothesis space consists of a number of hypotheses.

(Refer Slide Time: 27:19)



And suppose, k of those hypothesis have error greater than epsilon, so this hypothesis contains many hypotheses, some of which may have error less than epsilon; those are acceptable to us. And it can contain k bad hypothesis, whose error are greater than epsilon. Now, we want that after we have seen m samples, any hypotheses which are consistent with those m samples must have error less than epsilon. That is, none of these bad hypotheses should become consistent.

So, if I take a fixed hypothesis from the hypothesis space, the probability that that hypothesis h_i is consistent with one training example, is less than $1 - \epsilon$. Why is it so? h_i belong to H_{bad} . So, the error of h_i is greater than epsilon. So, probability that h_i is consistent with one training example is less than equal to $1 - \epsilon$. Now, we look at m , all the m training examples. The probability that a bad hypothesis h_i is consistent with all the m training examples is given by $1 - \epsilon$ to the power m . The probability that h_i is consistent with the entire training set is less than equal to $1 - \epsilon$ to the power m .

Now, we have k such bad hypothesis. Probability that any one of them is consistent with the training examples is less than equal to k into $1 - \epsilon$ (Refer Time: 29:23). Now, k , you know, there are some bad hypothesis. How many bad hypotheses are there? It could be any number. It could be 0 or 1, or 2; maximum value of k is the size of the hypothesis space, where all the hypotheses are bad. So, k into $1 - \epsilon$ to the

power m is less than equal to size of the hypothesis space into 1 minus epsilon to the power m .

So, this is the probability with which a consistent hypothesis has error greater than epsilon. So, our algorithm is trying to find a consistent hypothesis. The probability that a consistent hypothesis has error greater than epsilon is given by this. And we want this probability to be less than equal to delta. Now, if we use the fact that $1 - x$ is less than equal to e to the power minus x , we will see that if we can set H into e to the power minus epsilon m less than equal to delta, then, this condition will be satisfied. And we have this worked out, but let me just state it, and we will also look at this proof.

And from this, from this, by manipulation, we get this; that m is greater than, if m is greater than $\frac{1}{\epsilon} \ln H + \ln \frac{1}{\delta}$, then, so many labeled examples are sufficient, in the sense that if a learner finds a hypothesis which is consistent with m training examples, then, that hypothesis is a PAC hypothesis.

(Refer Slide Time: 31:05)

Sample Complexity: Finite Hypothesis Spaces Realizable Case

PAC: How many examples suffice to guarantee small error whp.
Theorem

$$m \geq \frac{1}{\epsilon} \left[\ln(|H|) + \ln\left(\frac{1}{\delta}\right) \right]$$

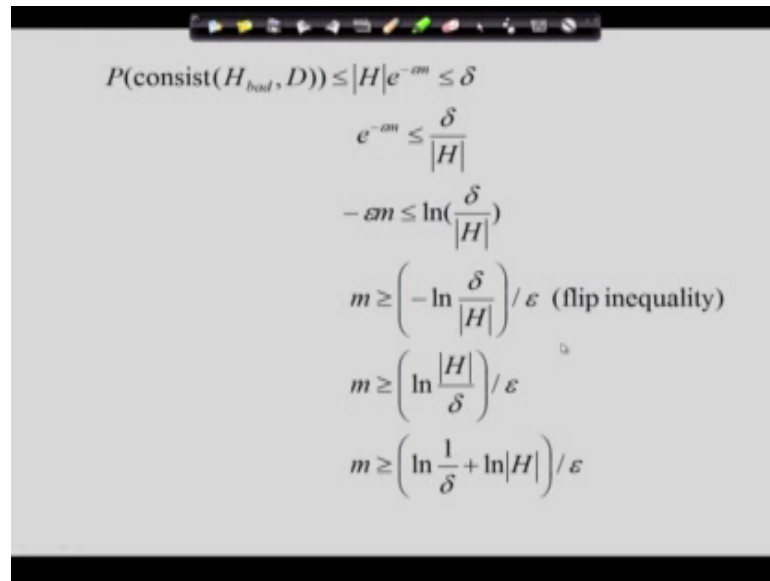
labeled examples are sufficient so that with prob. $1 - \delta$, all $h \in H$ with $err_D(h) \geq \epsilon$ have $err_S(h) > 0$.

Statistical Learning Way:
With probability at least $1 - \delta$, all $h \in H$ s.t. $err_S(h) = 0$ we have

$$err_D(h) \leq \frac{1}{m} \left[\ln(|H|) + \ln\left(\frac{1}{\delta}\right) \right]$$

The other way of looking at this is that if you are using m examples, then, and outputting a consistent hypothesis, then, with probability at least 1 minus delta, that hypothesis will have, the error of that hypothesis will be less than equal to this. So, with probability at least 1 minus delta, a hypothesis which looks at m examples, its true error will be less than equal to this; you, can bound the true error of the hypothesis.

(Refer Slide Time: 31:45)


$$\begin{aligned}P(\text{consist}(H_{\text{bad}}, D)) &\leq |H|e^{-\epsilon m} \leq \delta \\e^{-\epsilon m} &\leq \frac{\delta}{|H|} \\-\epsilon m &\leq \ln\left(\frac{\delta}{|H|}\right) \\m &\geq \left(-\ln\frac{\delta}{|H|}\right)/\epsilon \quad (\text{flip inequality}) \\m &\geq \left(\ln\frac{|H|}{\delta}\right)/\epsilon \\m &\geq \left(\ln\frac{1}{\delta} + \ln|H|\right)/\epsilon\end{aligned}$$

So, this is, you know, just working out of the particular thing, that if H is to the power minus m less than equal to δ , how we get the value of m . This is what we worked out and put in the previous slide.

Now, we have looked at the case where the learner looks at m training examples, outputs are consistent hypothesis; and we have looked at how many examples will be needed to be seen, in order to guarantee that that consistent hypothesis has error less than ϵ . And we want to do this with confidence greater than 1 minus δ , for this we have seen. But, there are scenarios where we cannot get consistent hypothesis, but, the hypothesis that the learner outputs is inconsistent; that is, it has error on the training set.

In the next class, we will look at how to handle inconsistent hypothesis, and what is the, what are the relation in that case.

Thank you.