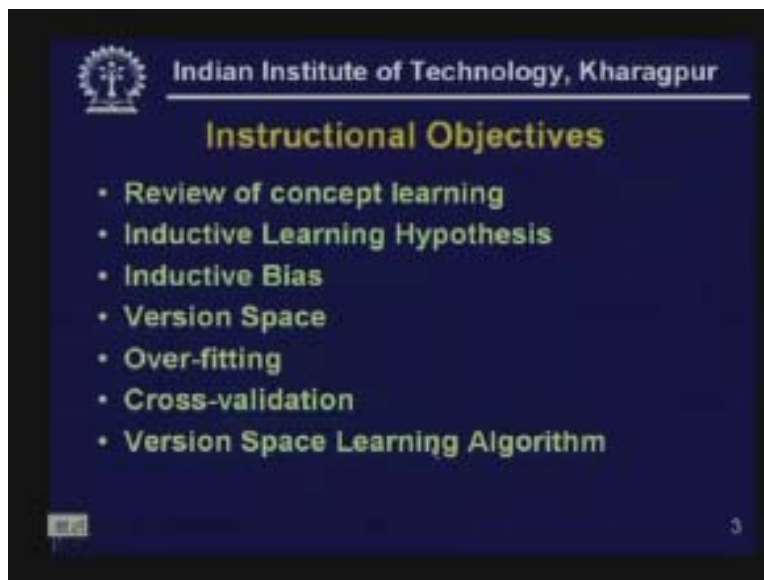


Artificial Intelligence
Prof. Sudeshna Sarkar
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture - 33
Introduction to Learning - II

Welcome, today we start with second lecture on learning. In the last class we had given introduction to machine learning. Today we will look further at the problem of concept learning. And in the subsequent classes we will look at some machine learning algorithms. The instructional objectives of today's class are as follows:

(Refer Slide Time: 1:32)



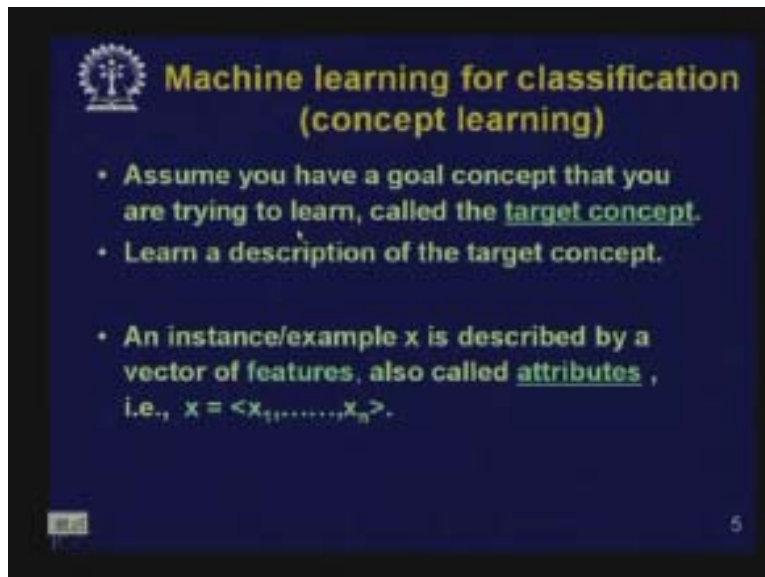
We will be reviewing concept learning or classifiers. We look more closely into the inductive learning hypothesis and explain the various types of inductive bias and also discuss why inductive bias is **regret** to select hypothesis. We will look at the definition of version space. We will discuss the concept of over-fitting and we will also look at the issue of Cross-validation in order to find out the accuracy of classifier. And we will look at some versions of the version space learning algorithm. **In the last class we discussed about concept learning or inductive learning for classification.**

(Refer Slide Time: 2:30)



According to this problem the assumption is that you have a goal concept that you are trying to learn and we call this a target concept. And as we discuss the target concept could be something like a description. You want to know whether the given object is an apple or not or if you are given an image you want to know whether this is an image of zero or not.

(Refer Slide Time: 2:45)



So your objective is to learn a description of the target concept. And what you are given is a set of examples which we call the training set. An example or an instance x is described by a vector of features also called attributes. For example, suppose x is an

instance x consists of n values $x_1 x_2 x_n$ then x_1 is the value for attribute 1 for this instance, x_n is the value of attribute n for this instance. Therefore, for the learning problem we have a number of attributes or features and each instance is described by its value for each of its attributes.

Our training set consists of a set of such training examples. And there is an underlying labeling function f that maps an instance to a class. Sometimes we will be interested in two class learning problems. In those cases we will say that some of those instances are positive example of class and others are negative examples of the class. So there can be two classes positive and negative and in general there can be a finite set of k classes and each instance is labeled to belong to one of these classes. So, f is a function which maps an instance to one of the classes from 0 to k minus 1.

(Refer Slide Time: 4:57)

Indian Institute of Technology, Kharagpur

Concept learning problem

- **Given:** A labeling function f that maps feature vectors into a discrete set of k classes. That is,
$$f: x \rightarrow \{0, 1, \dots, k-1\}.$$
- **Problem:** From a set of $(x, f(x))$ pairs, learn the target concept f .
- Your guesses of the target concept are called **hypotheses**.

Now our objective is, given a set of training examples that is a set of instances x and for each instance its label effect which is given to us we want to learn the target concept f . The function f is not known to us but we know f applied to the training instance. So we guess something which can possibly be f and these guesses are called the hypothesis that we form. So we talk about a hypothesis space.

We are going to search for f and where are we going to search for f ?

We define a hypothesis space. Once we have decided what are the attributes or features that we are using we will define a hypothesis space. And a hypothesis space is a collection of possible hypothesis. We will try to find good hypothesis out of those in the hypothesis space. So the hypothesis space h is a space of all possible hypothesis $h_1 h_2 h_n$ so $h_1 h_2 h_n$ are the different hypothesis in the hypothesis space. So, in the set of hypothesis suppose there are n hypothesis $h_1 h_2 h_n$ and we want to find one of these hypothesis which is possibly close to the target concept f that we are going to learn.

Now it may be the case or it may not be the case that the target concept f is exactly one of the hypotheses in this space. So, if you do not choose the hypothesis space well or if the hypothesis space is not expressive enough the target concept f may not belong to the hypothesis space. Nevertheless your objective is to find a hypothesis of the hypothesis space which is close enough to the target concept. So the objective of the learner is to find a hypothesis h which is a member of the hypothesis space H which fits the training data the best. Therefore in order to identify a good hypothesis or the best fit hypothesis from the hypothesis space the learning algorithm has to carryout some form of search through this hypothesis space that is through the space of possible hypothesis.

(Refer Slide Time: 8:04)



Now, this concept learning problem is schematically represented in this figure. So we decide a hypothesis representation which defines the hypothesis space h . the way we are going to represent the hypothesis defines the hypothesis space. Our objective is to find the desired hypothesis by searching from this hypothesis space. And the desired hypothesis is one which best fits the training example that we are given. The training examples are input to the learning algorithm. The learning algorithm searches through the hypothesis space to find one of the hypothesis which best fits the training example.

How do we get the hypothesis space?

We have got a set of attributes or features let us say $a_1 a_2 a_n$ are the n attributes. So the hypothesis space is defined in terms of these attributes. And the representation we choose to represent the hypothesis defines the hypothesis space. For example, if we consider that a hypothesis is a simple rule which can be expressed in predict calculus and the antecedent of the rule is a conjunction of literals so that could define an hypothesis space.

Suppose that we have three features $a_1 a_2 a_3$ then what is the possible hypothesis? The possible hypothesis could be a_1 which is a possible hypothesis, a_2 is a possible hypothesis, a_3 is a possible hypothesis, $a_1\bar{a}_2$ is a hypothesis, $a_2\bar{a}_3$ is a hypothesis, $a_3\bar{a}_1$ is a hypothesis, $a_1\bar{a}_2\bar{a}_3$ is a hypothesis, $a_1\bar{a}_2a_3$ is a hypothesis, $a_1a_2\bar{a}_3$ is a hypothesis, $a_1a_2a_3$ is a hypothesis.

is a hypothesis, a_1 and a_2 is a possible hypothesis, a_1 and a_2 bar is a hypothesis, a_1 bar and a_2 and a_3 bar is a hypothesis. So we can find out the number of such hypotheses. These are the possible hypothesis and we want to find one member of this hypothesis space that fits our training example.

Now, if we consider a different class of hypothesis, suppose our hypothesis is a disjunction of exactly two literals then our hypothesis space will be different. What could be the possible hypothesis in this space? a_1 or a_2 is a hypothesis, a_1 bar or a_2 bar is a hypothesis, a_1 or a_3 bar is a hypothesis, a_2 and a_3 bar is a hypothesis and so on. So we will have a different set of hypothesis. So the attributes that we use and the way we represent a rule determines the hypothesis space. The hypothesis could also be describing other forms. For example, the hypothesis can be represented as a decision tree, the hypothesis can be represented as a neural network and so on.

(Refer Slide Time: 11:53)



Some more definitions to review:

A training set is a set of all training examples that are given to the learner. And on the basis of the training set the learner forms the hypothesis. Now after this hypothesis is formed it has to be evaluated. If you evaluate the hypothesis using the same examples on which you have trained you are likely not to get a good reflection because your hypothesis may try to fit the training data perfectly well but it may not work well for other the data that you have not seen. So it is good if you can use a different set of examples for testing. So the set of examples that you use for testing is called the testing set. This example in the testing set should ideally be not known to the algorithm before it has learned the hypothesis. So the testing set is the set of all examples given to the learner after it has learned the hypothesis.

(Refer Slide Time: 13:03)

Indian Institute of Technology, Kharagpur

Some Definitions

- **Training set** – The set of all training examples given to the learner.
- **Testing set** – The set of all examples given to the learner after it has learned a hypothesis. This set is used to test the accuracy of the learned hypothesis over unseen examples.

9

This set is used to test the accuracy of the learned hypothesis over the unseen example.

(Refer Slide Time: 13:18)

Indian Institute of Technology, Kharagpur

False positive and false negative

False negative

Instance space

True concept

hypothesis

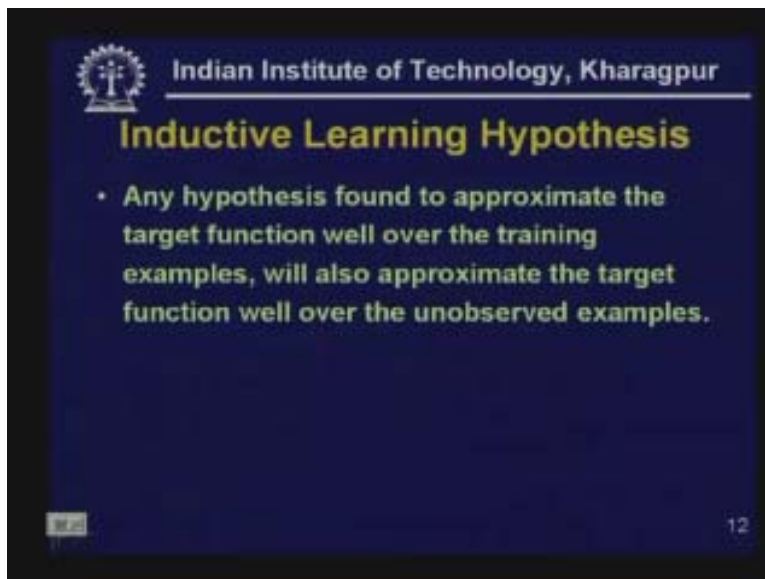
11

Here the oval represents the set of all possible instances and instances are described in terms of the values of the attributes. A concept is a subset of the instance space. So, only a subset of instance space belongs to the concept class and the other instances do not belong to the concept class. So the concept can be represented as a set which is a sub set of the instance space. And our objective is to learn this concept.

Now we find a hypothesis and this hypothesis in general may not be exactly the same as the concept. So when we have a hypothesis we see that there are some examples which are correctly classified by the hypothesis. For example, these examples are correctly classified by the hypothesis because they are positive according to the concept and they are also positive according to the hypothesis. Whereas if you look at these examples, for these instances they are actually members of the concept path but they are not members of the hypothesis. So these are called false negative instances.

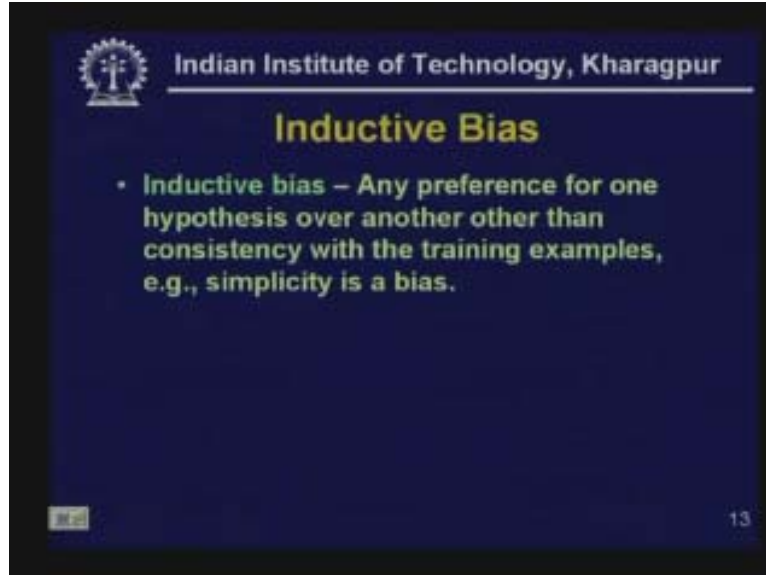
Similarly, these instances are classified as positive by the hypothesis but actually according to the concept they do not belong to the concept so they are called false positives. So this is the region of false negatives and this is the region of false positives and the middle region is labeled as positive by both the hypothesis as well as the concept and the rest of the instances are also labeled as negative by both the concept and the hypothesis. Therefore the false positives and false negatives are the two regions of error for the hypothesis. Now our objective as we said in inductive learning is given for a set of training examples we have to learn a description or we have to learn a hypothesis. And as we have noted that we cannot learn this deductively because we do not have access to information about all the instances we use the principle of inductive inference.

(Refer Slide Time: 16:26)



The inductive learning hypothesis states that any hypothesis which is found to approximate the target function well over the training examples will also approximate the target function well over the unobserved examples. Now let us see the assumption behind this. If the training set and the test set belong to the same distribution, that is, if there is some similarity between them then by looking at the training set if we can identify a pattern we can infer inductively that such a pattern actually exists and it will be true of all instances. Therefore this is the principle of inductive inference which helps us to make an inductive leap.

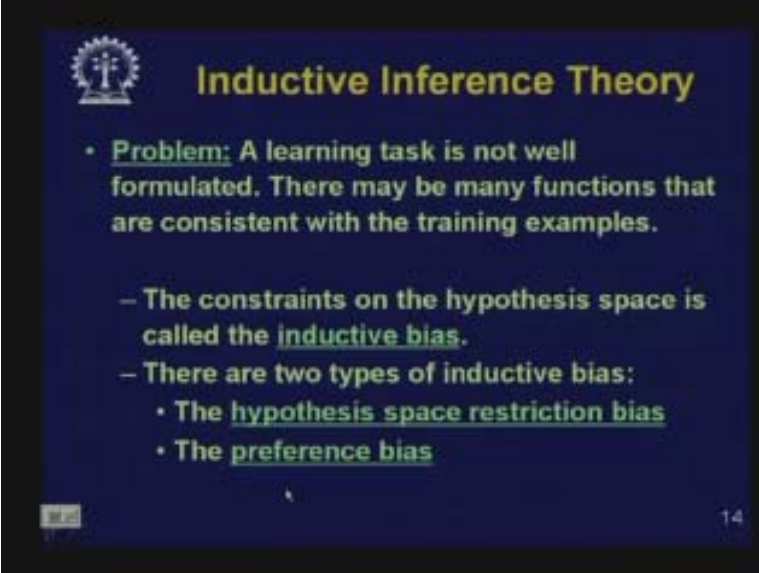
(Refer Slide Time: 17:21)



Now you are given a set of training examples. On the basis of set of hypothesis you want to learn a hypothesis. There could be several competing hypothesis. Now what is the basis on which you decide which of the hypothesis to choose? The hypothesis you must choose is the one which has the minimum error and the highest accuracy. But you cannot measure the true accuracy of the hypothesis because you do not have access to all the training examples. Therefore you must use some sort of bias which is called inductive bias. There are many biases which have been used. For example, simplicity of hypothesis is a possible bias. Therefore a learning task is not always well formulated so there could be many functions that are consistent with all the training examples. And the algorithm must have an inductive bias on the basis of which the algorithm will decide one of the hypotheses.

There are several types of bias which could be used. There could be a bias while designing the hypothesis space. The set of hypothesis is defined by the hypothesis space and hypothesis space is defined by the data structure or representation that we used to describe the hypothesis. By restricting the language for the hypothesis space you have a restriction bias.

(Refer Slide Time: 18:23)



The slide features a dark blue background with a white logo in the top left corner. The title 'Inductive Inference Theory' is written in a yellow font. The content is presented as a bulleted list in white text. The first bullet point is a 'Problem' statement. The second is a general statement about constraints. The third is a list of two types of bias, each with a sub-bullet point.

- **Problem:** A learning task is not well formulated. There may be many functions that are consistent with the training examples.
- The constraints on the hypothesis space is called the **inductive bias**.
- There are two types of inductive bias:
 - The **hypothesis space restriction bias**
 - The **preference bias**

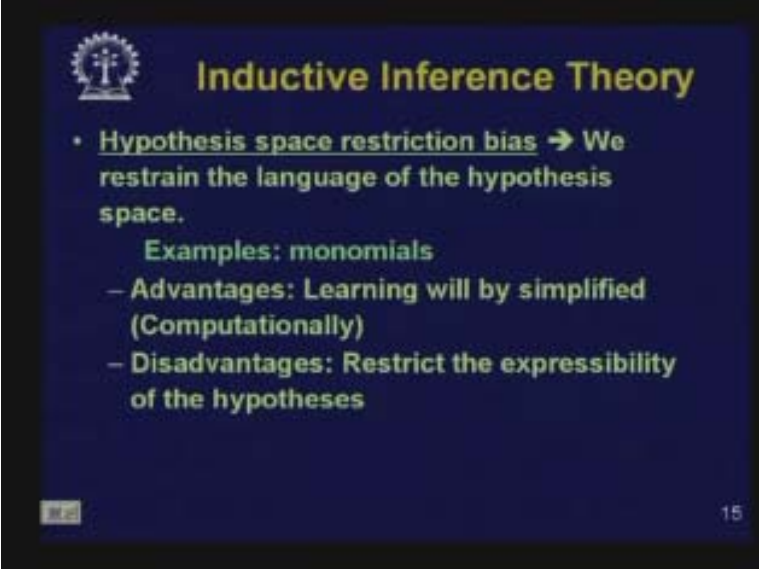
14

Also, given a hypothesis space even in this hypothesis space there could be several competing hypothesis and the one which you choose among them would depend on your preference bias. For example you might say that if I have rule whose size is 1 then I will prefer that rule to a rule with size 3. So I might say I prefer a simpler hypothesis. So we might also have a preference path. There could be two types of biases. One is the restriction bias which restricts the hypothesis space and preference bias which selects among the possible hypothesis in the same hypothesis space. So, in hypothesis space restriction bias we restrain the language of the hypothesis space. For example, we might consider that hypothesis of monomials.

What is a monomial?

Monomial is a conjunction of positive or negative literals. So monomials are a conjunction of literals. We might say that our hypothesis space must only be a monomial such as a_1 and a_2 , a_1 and \bar{a}_2 and a_3 , \bar{a}_1 and a_4 and a_3 and so on. These are examples of monomials. If we have a monomial we cannot have disjunctions. We cannot have hypothesis like a_1 or a_2 , we cannot have hypothesis like a_1 or a_2 and a_3 . So, if we decide that our hypothesis space is the space of monomials we are ruling out other types of hypothesis.

(Refer Slide Time: 20:03)



The slide features a dark blue background with a white logo in the top left corner. The title "Inductive Inference Theory" is written in yellow. The main content is in white text, including a bullet point about hypothesis space restriction bias, examples of monomials, and a list of advantages and disadvantages.

Inductive Inference Theory

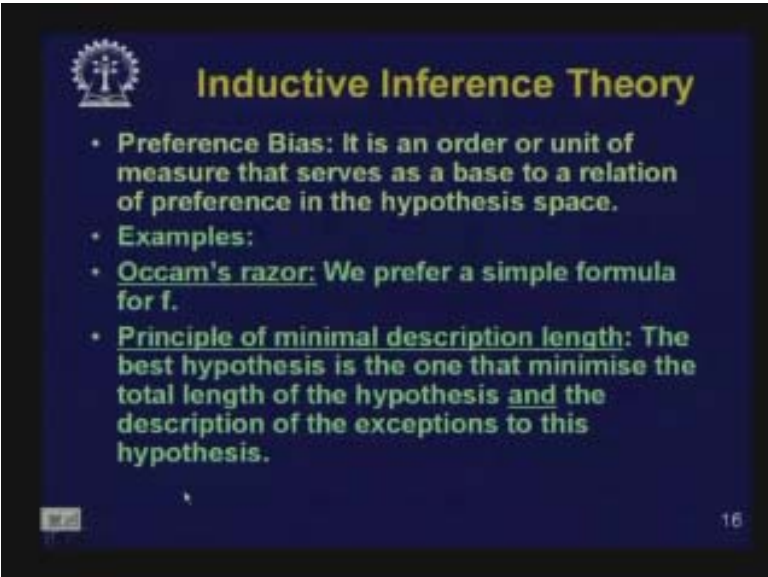
- **Hypothesis space restriction bias** → We restrain the language of the hypothesis space.
Examples: monomials
 - Advantages: Learning will be simplified (Computationally)
 - Disadvantages: Restrict the expressibility of the hypotheses

15

What is the advantage of choosing a restricted hypothesis space?

If we have a smaller hypothesis space the process of learning will be simpler. The size of the hypothesis space is smaller and the algorithm may be simpler. The disadvantage of choosing a small hypothesis space is that a small hypothesis space may not capture all possible hypotheses so it may not be able to express all types of hypothesis. If you take an unrestricted hypothesis space we can represent all type of functions but then there will be a lot of functions which are consistent with the training examples and we have to give strong preference bias to choose one of them. And trying to search from a large hypothesis space we will not get a good algorithm to do that.

(Refer Slide Time: 21:59)



The slide features a dark blue background with a white logo in the top left corner. The title "Inductive Inference Theory" is written in yellow. The main content is in white text, including a definition of preference bias, examples, and a principle of minimal description length.

Inductive Inference Theory

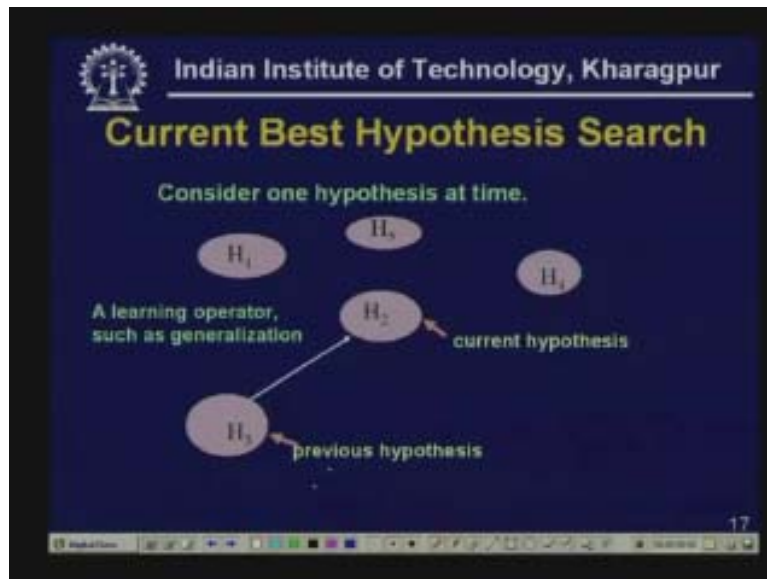
- **Preference Bias:** It is an order or unit of measure that serves as a base to a relation of preference in the hypothesis space.
- Examples:
 - **Occam's razor:** We prefer a simple formula for f .
 - **Principle of minimal description length:** The best hypothesis is the one that minimise the total length of the hypothesis and the description of the exceptions to this hypothesis.

16

In preference bias we try to use a bias to decide which hypothesis is more useful. There are several types of preference bias that have been considered. One very popular type of preference bias is called the occam's razor principle. So, occam's razor is a very old principle and it actually means that things should not be multiplied unless it is necessary. So what it translates to is that we prefer simpler formula for our hypothesis. So if you have two competing formula and one is simpler we will prefer them. And there is some intuitive explanation of occam's razor.

There are fewer simpler hypothesis simpler patterns than more complex patterns and if a simpler pattern can fit our training data it has a better chance of fitting unseen data. And the second type of preference bias which is also very similar is the principle of minimum description length mdl which states that the best hypothesis is the one that minimizes the total length of the hypothesis and the description of the exceptions to the hypothesis. These are the some of biases that people have used.

(Refer Slide Time: 23:38)




When we have a hypothesis space which consists of a set of hypothesis how do we search for the hypothesis that we want. The search could be carried out in many ways. We will just mention two simple possibilities. One possibility is, you consider one hypothesis at a time which you call your current hypothesis. If the current hypothesis is not good enough you choose another hypothesis which is your next hypothesis. So this is the previous hypothesis and you try to go to a neighboring hypothesis using some operators. So we could use certain operators and we can try to make the hypothesis general or more specific.

Therefore our current best hypothesis search looks at one hypothesis at a time and if it goes to a point where there are no good alternatives then it might backtrack to the previous job. Therefore several algorithms have been suggested to do current best hypothesis search. But the thing to note is that if the hypothesis space is very large this

search will take a long time. The other type of search which has been advocated for theoretical learning algorithms is Least Commitment Search.

Least Commitment Search considers multiple search at a time so it maintains a set of possible of hypothesis and as it gets more training examples it tries to restrict the set of hypothesis or adds some more.

(Refer Slide Time: 25:36)



Indian Institute of Technology, Kharagpur

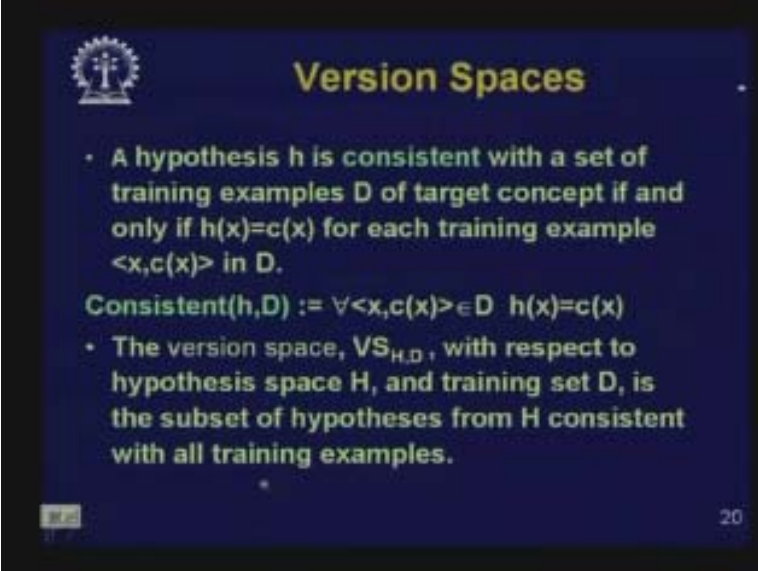
Version Space

- Original hypotheses space:
 $H = H_1 \vee H_2 \vee \dots \vee H_n$
- The set of consistent hypotheses remaining is called the version space.
- Version space learning algorithm

19

So, one type of such algorithm uses the notion of what we call a version space. Suppose our original hypothesis space is H and H consists of n hypothesis H_1, H_2, H_n , some of these hypotheses is consistent with the example and some of them are not. If we remove these hypotheses that are not consistent with the examples we get the remaining hypothesis consistent with the current example. The hypothesis which is consistent with all current examples constitutes the version space. As we look at more examples the version spaces is further restricted. So our algorithm might maintain the current version of space and as it sees more examples the version space can get slowly restricted. This is called version space learning.

(Refer Slide Time: 26:42)



Version Spaces

- A hypothesis h is consistent with a set of training examples D of target concept if and only if $h(x)=c(x)$ for each training example $\langle x, c(x) \rangle$ in D .

$\text{Consistent}(h, D) := \forall \langle x, c(x) \rangle \in D \quad h(x)=c(x)$

- The version space, $VS_{H,D}$, with respect to hypothesis space H , and training set D , is the subset of hypotheses from H consistent with all training examples.

20


Formally a hypothesis h is consistent with a set of training examples D if and only if $h(x)$ and $c(x)$ where $c(x)$ or $f(x)$ is the target concept that we trying to learn, c is the target concept. So, if a hypothesis h agrees with the target classification c for every training example that we have seen then that hypothesis is consistent with the training example. In other words h is said to be consistent with the training set d if for all instances x $c(x)$ belonging to the training set D $h(x)$ is equal to $c(x)$.

What is the version space?

The version space is denoted by $VS_{H,D}$ so the version space with respect to the hypothesis space H and the training set D is the subset of hypothesis from the hypothesis space that are consistent with all the training examples. This is called the version space.

In other words, version space $VS_{H,D}$ is a set of all hypothesis belonging to the hypothesis space such that it is consistent with the training example. With this definition we can outline a theoretically simple algorithm called a list then eliminate algorithm. In list then eliminate algorithm when we have not seen any examples when d is null we say that a version space is the entire hypothesis space. So we list the entire hypothesis in the hypothesis space. Then as we take one example at a time we throw out all the hypotheses that are inconsistent in the training example and then we continue. Now this algorithm is very simple to describe but actually it is unrealistic because usually for any reasonable complex hypothesis the hypothesis space is really large and it is not practical to store the entire hypothesis in the hypothesis space.

(Refer Slide Time: 28:04)



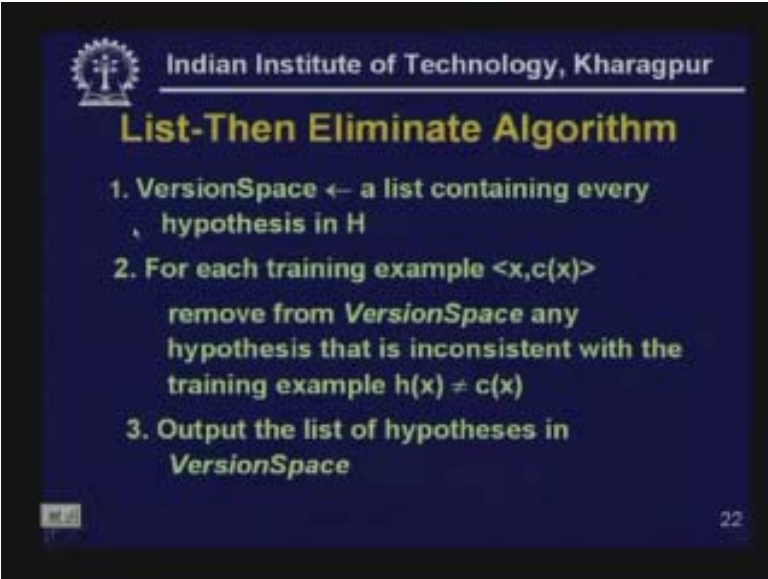
Indian Institute of Technology, Kharagpur

Version Space

- Version Space
 $VS_{H,D} = \{h \in H \mid \text{Consistent}(h,D)\}$
- List-Then-Eliminate Algorithm
 - obvious way to represent the Version Space
 - exhaustively, unrealistic

21

(Refer Slide Time: 29:20)



Indian Institute of Technology, Kharagpur

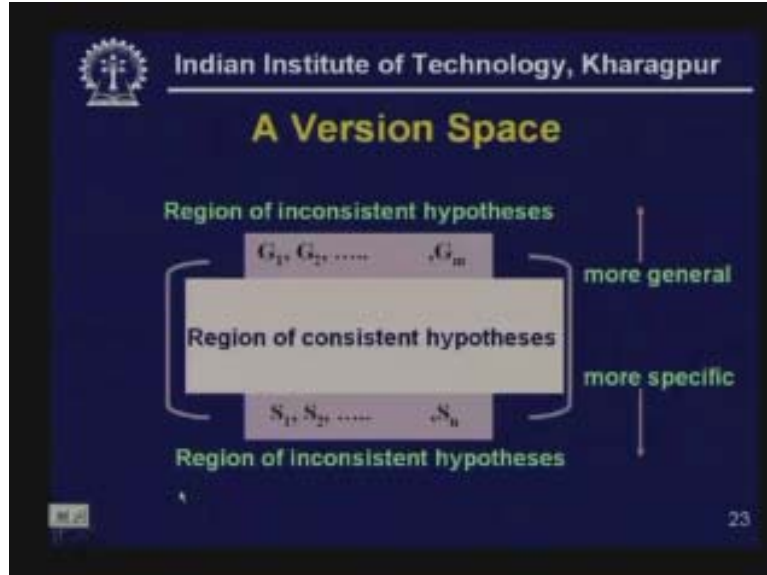
List-Then Eliminate Algorithm

1. VersionSpace \leftarrow a list containing every hypothesis in H
2. For each training example $\langle x, c(x) \rangle$
remove from VersionSpace any hypothesis that is inconsistent with the training example $h(x) \neq c(x)$
3. Output the list of hypotheses in VersionSpace

22

Nevertheless the list then eliminate algorithm can be theoretically described as follows: We maintain a area of structure called the version space which is a list containing every hypothesis in the hypothesis space initially. Then we process each training example one at a time. We remove from the version space any hypothesis with the current training example. And at the end we output the list of the hypothesis is remaining in the version space. So this is the basic version space algorithm.

(Refer Slide Time: 29:54)

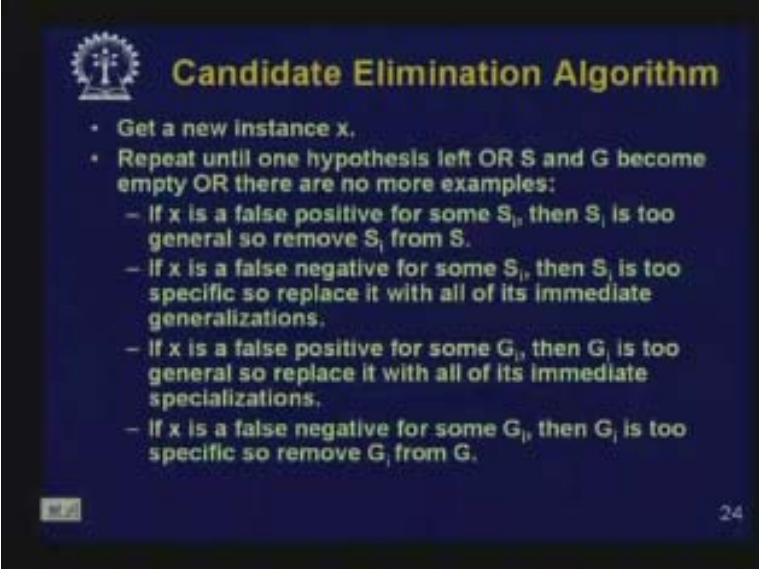


Obviously this algorithm is not practical because the number of such hypothesis is very large. Now we will just briefly discuss a variation of the algorithm which instead of keeping the entire list of consistent hypothesis tries to represent them in a succinct manner.

A version space can be represented by keeping a list of all consistent hypotheses. Additionally a version space can be represented by keeping the set of general hypothesis or a set of hypothesis. There are some hypotheses in the hypothesis space and there are inconsistent hypothesis. This inconsistent hypothesis is either more general than one of the hypothesis in the hypothesis space or more specific than one of the hypothesis in the hypothesis space.

In many cases the version space can be succinctly represented by a set of most general hypothesis in the version space and by a set of more specific hypothesis in the version space. Any hypothesis will belong to the version space. The specific set is called the s set and the general set is called the g set. If a hypothesis is less general than one element of g set and most general element of s set then the hypothesis belongs to the version space. With this idea an algorithm called the version space algorithm or candidate elimination algorithm has been formed which maintains the g set and s set.

(Refer Slide Time: 31:52)



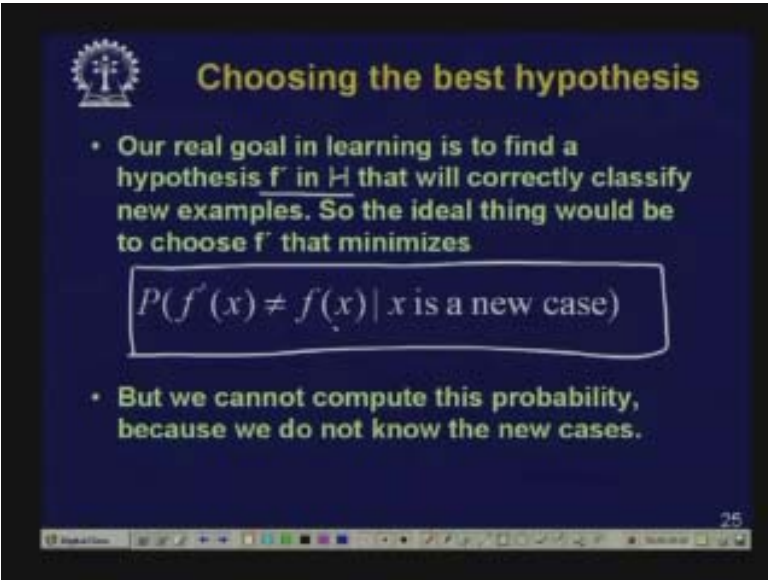
Candidate Elimination Algorithm

- Get a new instance x .
- Repeat until one hypothesis left OR S and G become empty OR there are no more examples:
 - If x is a false positive for some S_i , then S_i is too general so remove S_i from S .
 - If x is a false negative for some S_i , then S_i is too specific so replace it with all of its immediate generalizations.
 - If x is a false positive for some G_i , then G_i is too general so replace it with all of its immediate specializations.
 - If x is a false negative for some G_i , then G_i is too specific so remove G_i from G .

24

Now let us see what we mean by choosing the best hypothesis. We are given a set of training examples but our real goal in learning is to find a hypothesis f' belonging in h that will correctly classify all examples. So the ideal thing would be to choose that hypothesis f' that minimizes this probability that f' misclassifies the concept.

(Refer Slide Time: 32:31)



Choosing the best hypothesis

- Our real goal in learning is to find a hypothesis f' in H that will correctly classify new examples. So the ideal thing would be to choose f' that minimizes

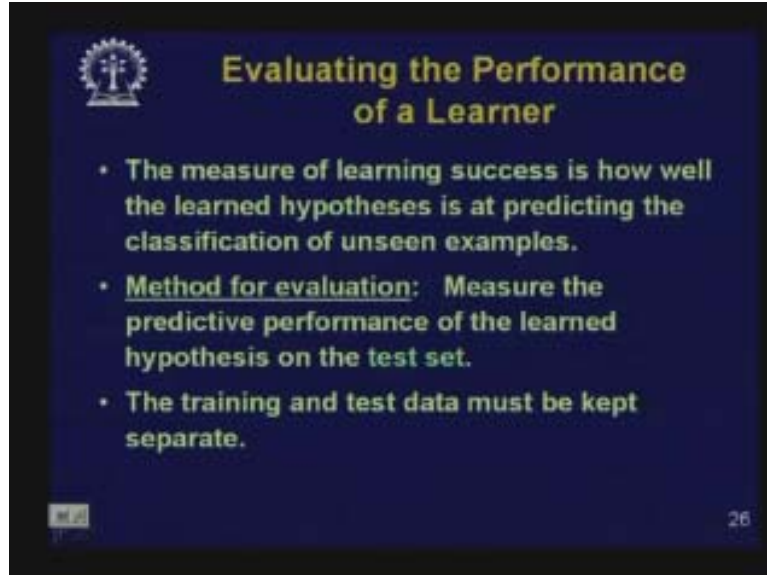
$$P(f'(x) \neq f(x) | x \text{ is a new case})$$

- But we cannot compute this probability, because we do not know the new cases.

25

So f is what we are trying to learn, f' is what we have so we want f and f' to be very close. But we cannot compute this probability because we do not have all the guesses with us.

(Refer Slide Time: 32:53)



How do we measure the success of a learning algorithm?

We have to measure how well the learned hypothesis predicts the classification of unseen examples.

How do we therefore evaluate a classification?

We have to measure the predictive performance of the hypothesis. If we learn on the basis of a training set and we test the performance on the same set we are cheating in some sense because our hypothesis may very well classify the training set but may not do so well on the test set because already we have seen the training set and we can be biased towards the training set. Therefore, in order to make a better judgment of the accuracy of the classifier we must measure it using unseen example and this set of unseen examples is called the test set. And do the extent possible we must keep the training set and the test set disjoint.

(Refer Slide Time: 34:01)



When we test the accuracy of the classifier we will get some sort of curve like this. So what we expect is as the size of the training set increases the accuracy of the classifier may increase. So we plot on the y axis the accuracy of the classifier and we measure the accuracy of classifier on the test set which is disjoint from the training set. And as we include more and more training examples we plot the accuracy of the classifier so we may typically get some curve like this and the accuracy may increase or decrease as the number of training set increases. So this sort of curve is called a learning curve.

(Refer Slide Time: 35:01)

The figure is a slide titled "Fitting hypotheses to the training data" with a logo in the top left. It contains a list of conditions for a learning problem. The slide number "26" is in the bottom right corner.

- Given
 - Training examples $\langle x, f(x) \rangle$ for some unknown function f .
 - A space of possible hypotheses H
 - An error function $\text{err}(f')$ that measures the error rate of any hypothesized function f' in H
- Find
 - The function f^* in H that minimizes the error, $\text{err}(f^*)$

So we want to fit the hypothesis to the training data. So we have some training examples, we have a space of possible hypothesis and we have an error function and we try to find a hypothesis which minimizes the error function.

(Refer Slide Time: 35:01)

Indian Institute of Technology, Kharagpur

An example of an error function

- Squared error:

$$J(\hat{f}) = \sum_{x \in S} [f(x) - \hat{f}(x)]^2$$

where S is the set of training examples.

26

There could be several error functions that we could use. For example our error function could be the number of examples which have been misclassified or our error function could be the sum of squared errors between $f(x)$ and $\hat{f}(x)$. If you test the accuracy on your training data you will usually note that as you train more and more the accuracy on the training data increases.

(Refer Slide Time: 35:51)

The problem of overfitting

- The more you train, the better you fit the training data. Ultimately all the examples will be memorized. If there is any noise in the training data, the algorithm will fit the noise. This is called overfitting.

accuracy

100

0

1 10 100 1000

Epochs of training

Training data

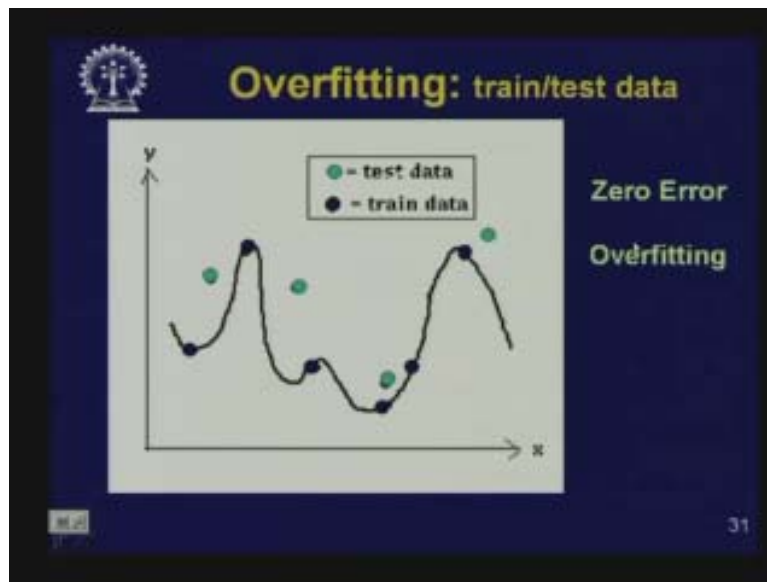
Test data

30

However, if you try to go on increasing the accuracy on the set of training data what might happen is that the accuracy on unseen data increases up to a point and then reduces. This is a common phenomenon you can observe that as your algorithm is tuned more and more the accuracy on the training set may go on increasing but after a point the accuracy on the test set might decrease. This is a typical behavior you might notice with many learning algorithms.

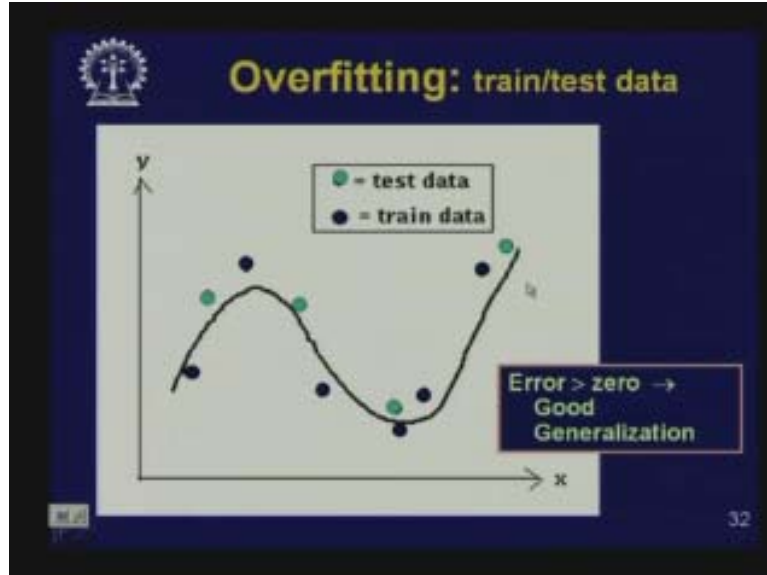
What is happening usually is that your algorithm is over fitting the data. It is fitting itself to all the idiosyncrasies in the training data which makes it incapable of generalizing over unseen data. This phenomenon is called over fitting because more you train the better you fit the training data. Ultimately all the examples might be memorized. The data that you get may not be pure data there may be noise in the data. For example, suppose the color of apple is green it might appear as yellow. So, if you try to fit in the noise in the training data the algorithm will over fit the training data and work very well on the training data but it will not work well on the test data. This is an example to illustrate over fitting.

(Refer Slide Time: 37:51)



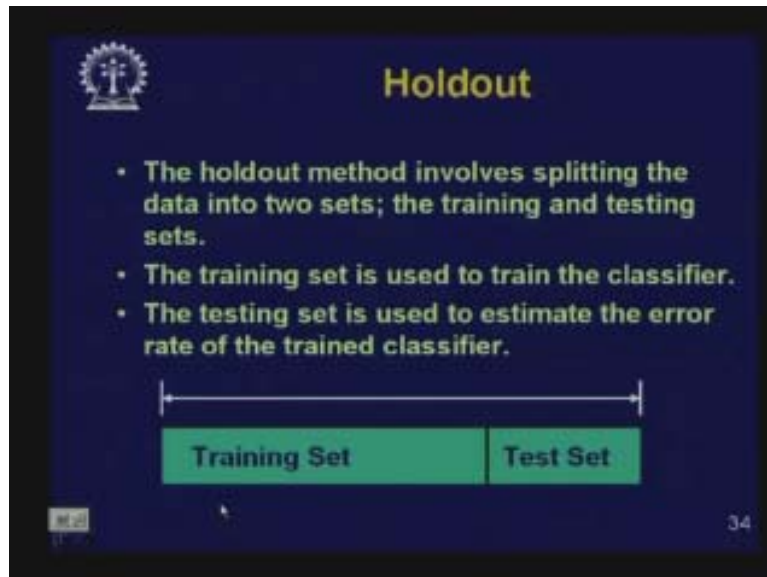
These black things here are the training examples and the curve you have got fits all the training examples. The green balls are the test data and this curve does not fit the test data very well, it has zero error on the training data but it has some error on the test data.

(Refer Slide Time: 38:23)



For the same examples we have another curve which has **from** finite error on the training data but it fits the test data in a better fashion.

(Refer Slide Time: 38:40)

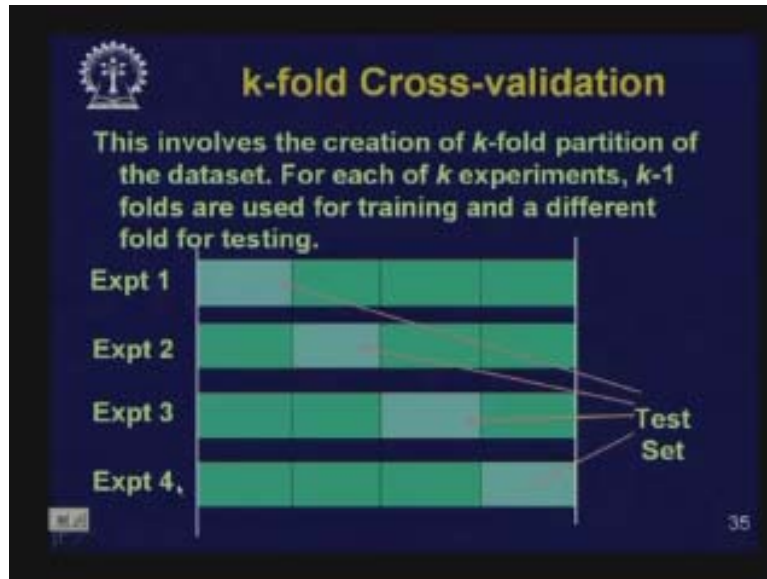


How are you going to properly test the accuracy of your classifier?

You are given a set of examples. Now what you should do is that instead of using all the examples for training you should keep aside some of the examples for testing. So you hold out some of the examples. You divide the total set of examples you have into two sets as the training set and the test set. On the basis of the training set you train your classifier and then you measure accuracy with the test set. So the hold out method

advocates splitting the available data into two sets. The training set is used to train the classifier and after the classifier is trained testing set is used to estimate the error of the classifier.

(Refer Slide Time: 39:35)



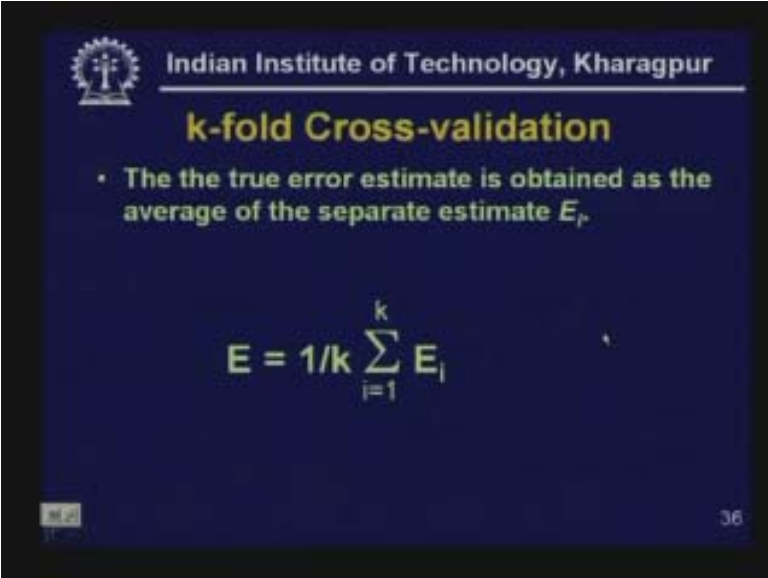
Of course this hold out method requires that enough number of examples are available so that even after keeping aside some test set you have enough data remaining use for training. Usually for most algorithms, the more data you use for training the better the accuracy of the classifier will be which you can expect. So this sort of hold out will only work well if there is sufficient amount of data. But some times what happens is that lot of data is not available. In certain fields it is not very easy to get data.

In those cases you have to use some intuition to optimally use the data that you have. So, one of the techniques is called the k-fold Cross-validation. The k-fold Cross-validation is useful when you do not have a large amount of training data. So what you do is you partition the available data into k classes. For example, this is the case of four fold Cross-validation.

We have divided the data into four classes **and then we ran** four experiments. In experiment one we keep aside the first class for testing and the remaining three we used for training. In the second experiment we keep the second data aside for testing and used the remaining three for training and so on. So we have four experiments and for each experiment we get accuracy and we get the estimate the accuracy of classifier an average of this accuracy. So this is called k-fold Cross-validation. It could be used to conserve training data and make the best use of the training data. And one extreme case of k-fold Cross-validation is Leave One Out Cross-validation where if you have m data you divide the data into m sets of one example each and you run m experiments, in experiment i you leave aside one of the data i and use the rest for training and on the trained data you test

one. So this sort of approach is used when you do not have a large amount of examples available to you.

(Refer Slide Time: 42:10)



Indian Institute of Technology, Kharagpur

k-fold Cross-validation

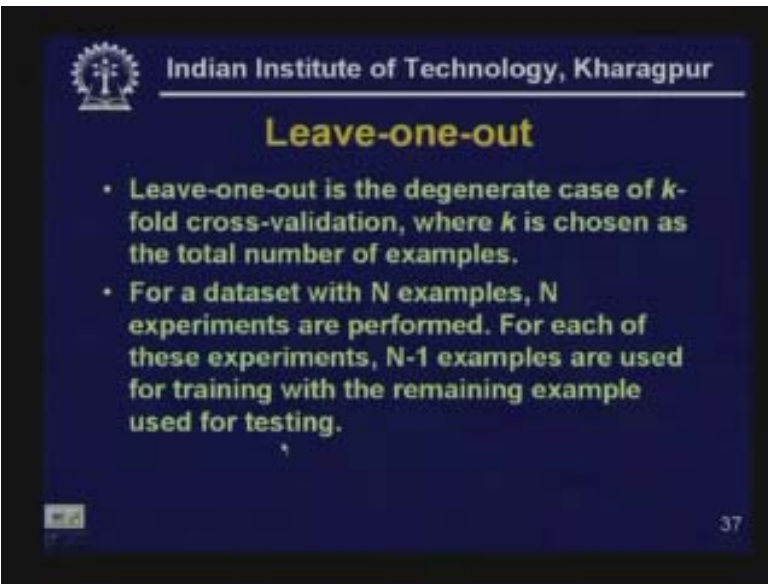
- The true error estimate is obtained as the average of the separate estimate E_i .

$$E = 1/k \sum_{i=1}^k E_i$$

36

Therefore in k-fold Cross-validation the true error estimate is obtained as the average. You have k estimates of error and you take the average to obtain the average estimate of the error.

(Refer Slide Time: 42:10)



Indian Institute of Technology, Kharagpur

Leave-one-out

- Leave-one-out is the degenerate case of k-fold cross-validation, where k is chosen as the total number of examples.
- For a dataset with N examples, N experiments are performed. For each of these experiments, N-1 examples are used for training with the remaining example used for testing.

37

Leave-one-out Cross-validation will often be referred to as LOOCV that stands for Leave-one-out Cross-validation. It is a case of k-fold Cross-validation where k is chosen

as the total number of examples. So, if you have a data set of N examples N experiments are to be performed. For each of these experiments one example is kept aside for testing and $N-1$ examples are used for training.

(Refer Slide Time: 42:10)

Indian Institute of Technology, Kharagpur

Questions

1. Consider a 2-class learning problem having 4 boolean attributes: (A_1, \dots, A_4) . The following examples are available for training. Assuming a bias in favour of simpler hypotheses, propose a hypothesis for the concept Q .

A1	A2	A3	A4	Q
T	T	F	F	-
F	T	F	T	-
F	T	T	T	+
T	T	T	F	+
F	T	T	T	+
F	T	F	F	-
T	T	F	T	-

36

Here are some questions:

1) Suppose you have a 2-class learning problem so the classes are plus and minus, the classes are referred to by q so q is the target concept that you are trying to learn. And you have got seven different examples and there are four attributes to this learning problem a_1 one a_2 two a_3 three and a_4 four and these attributes are Boolean attributes. And you have got these seven examples so you assume that there is a bias in favor of simpler hypothesis. So your objective would be to find a good hypothesis for this particular concept q . So the training data r first example is, true true false false it is negative, the second example is, A_1 one is false A_2 two true then false, true its class is negative.

Third example: false true true true class positive.

Example 4: true true true false class is positive.

Example 5: false true true true class is positive.

Example 6: false true false false class is negative.

Example 7: true true false true class is negative.

So, given these examples you try to find a simple hypothesis for q . And you can also try to find several hypotheses which are consistent with these training examples and find the simplest of them. **Use your own measure of simplicity.**

2) Again you are given another training set. Now this training set is slightly more complicated. We have again four attributes $A_1 A_2 A_3 A_4$ and we have a target concept q . So, again for these training examples you must try to figure out different hypothesis and

propose one of the hypothesis which is a good hypothesis which is consistent with all the training examples. Here again we have a ten training examples.

Example 1: A1 is high A2 is red A3 is in A4 is NV and classification is positive.

Example 2: A1 is high A2 is green A3 is out A4 is V and classification is positive.

Third example: A1 low A2 green A3 in A4 is V and classification is negative.

Example 4: A1 is high A2 is red A3 is in A4 is NV and Q is positive.

Example 5: A1 is low A2 is green A3 is out A4 is NV Q is negative.

Example 6: A1 is low A2 is red A3 is in A4 is V and Q is negative.

Example 7: A1 is high A2 is red A3 is out A4 is NV and Q is negative.

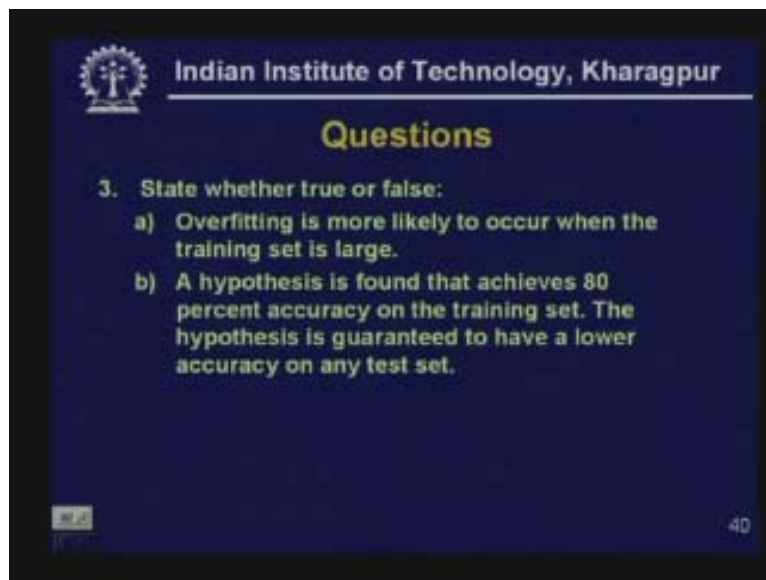
Example 8: A1 is low A2 is green A3 is in A4 is NV and Q is positive.

Example 9: A1 is high A2 is red A3 is in A4 is V and Q is negative.

Example 10: A1 is low A2 is blue A3 is in A4 is NV and Q is positive.

Therefore your exercise would be given these training examples, find some competing hypothesis and choose one hypothesis.

(Refer Slide Time: 47:40)



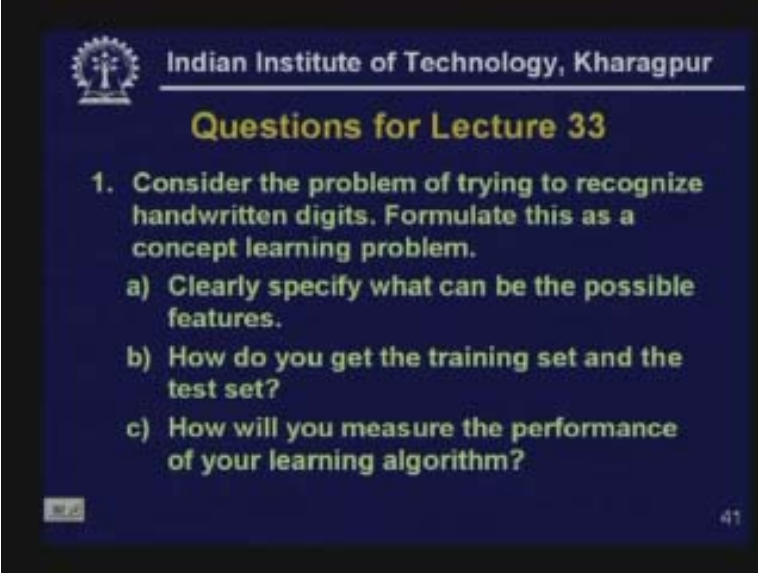
Then the last question:

3) You state whether true or false:

a) Overfitting is more likely to occur when the training set is large.

B) A hypothesis is found that achieves 80% accuracy on the training set. The hypothesis is guaranteed to have a lower accuracy on any test set. State whether true or false, whether the accuracy on any test set will always be lower than the accuracy on the test set.

(Refer Slide Time: 49:39)



Indian Institute of Technology, Kharagpur

Questions for Lecture 33

1. Consider the problem of trying to recognize handwritten digits. Formulate this as a concept learning problem.
 - a) Clearly specify what can be the possible features.
 - b) How do you get the training set and the test set?
 - c) How will you measure the performance of your learning algorithm?

41

Some questions from the previous lecture:

Consider the problem of trying to recognize handwritten digits. Formulate this as a concept learning problem.

- a) Clearly specify what can be the possible features.
- b) How do you get the training set and the test set?
- c) How will you measure the performance of your learning algorithm?

In order to recognize handwritten digit what is the input that you get? The input is an image. Suppose you write this digit 1 or write this digit 7 the input is an image containing this digit. And in this input usually when you are trying to recognize digits this image will be only in black and white in two colors, the background color and the pixel color of the digit. So, from this image your objective would be to find out whether this is 1 or 7 or 2 or 3 and so on. Therefore your raw input is an image like jpeg or bitmap etc. It is true from what you can get is the pixel value. Therefore you get the pixel value and you can do some sort of image processing so that you ultimately get the black and white image. So you binaries it and you get a black and white image where some pixels are black and some are white and from this image you want to know the identity of this digit. So the raw input is the set of pixels. But the number of pixels will depend on the size of the window. Suppose this window is 20 pixels by 20 pixels so you have 400 pixels where each pixel is either white or black, this is the raw image.

Now you can try to get a hypothesis using these raw features. Therefore if you use these raw features your algorithm may not be able to get a good hypothesis unless all the possible ones are written exactly in the same way. Some people might write 1 like this, some people write 1 like this, some people write 1 like this some people write like this so all these have different pixel representations. Now if you want to do a very good job of recognizing these digits may be it will help if you can extract some more sophisticated

features from this raw image that is available to you. For example, you could try to find whether there is a vertical line.

You can try to look at the image and you can try to detect all vertical lines in the image or all possible lines in the image. You can try to get different transforms and you can find the moment of the image. So you can get set of more sophisticated features which can be used by a learning algorithm. So the raw input to this learning problem is a set of pixels whether it is white or black.

Initially you put with an image which consists of colored and pixels after you do some preprocessing or binarisation you get some raw pixels and you can further make more pixels indicated by getting more sophisticated features. For example, the value of the first moment, the result of half transform, the different lines that are present in the image, you can use different masks. So those are the inputs to your learning problem.

Now how would you get the training set and the test set? If handwritten digit recognition is the problem that you wish to address you should get several of your friends or to write down the different digits in their own handwriting. The style of writing the digits differs from person to person. Different digits may be expressed in different ways. So if you get different types of digits written by different people or friends then you can try to use these as your training set or the test set. Or you could take certain places where you have access to handwritten digits, for example in the post office you could scan the pin code on the letters written by various and try to recognize those digits. So you collect a set of such samples. And once you have collected sufficient number of samples you use some for training and the rest for testing.

To measure the performance of learning algorithm you use a training set on the basis of which you train your classifier and evaluate the accuracy on an unknown set.

(Refer Slide Time: 55:07)

Indian Institute of Technology, Kharagpur

Questions for Lecture 33

2. Consider the problem of trying to play a game of ludo. Formulate this as a learning problem.

- a) Clearly specify what your system will try to learn.
- b) How can you get the training examples for this system?

42

2) Consider the problem of trying to play a game of Ludo, formulate this as a learning problem. Clearly specify what your system will try to learn. How can you get the training examples for this system?

Now, in order to learn to play a game of Ludo there are the two things you could do. Either you could work it out in the supervised learning framework or you could work it out in the reinforcement learning framework. In supervised learning framework you might get a set database of boards, actually in Ludo you have at least two or more players and suppose there are two players player one and player two and there is a board and different pieces are in different positions and when it is your turn to play you roll a die and depending on the score of the die you have to play your move. So basically what you are trying to learn is, given the board and given the die **what should be your next move** is what you need to know.

Now, in order to learn this if you have a set of training examples where for different board positions and for different rolls of die you have the recommended move to make given by an expert then you could try to learn the move from the board. In Ludo the number of possible board configurations is very large and for each configuration you have several possibilities of what the die throw turns out so the number of possible situation is very large and this is not a very easy learning problem unless you are able to generalize in some sense. So using these raw features is not really very practical. Therefore you have to generalize the board in some manner.

The second way by which you can learn is you can play automated games against each other by random moves and you can generate a huge number of database of such games and you can use reinforcement learning to learn. But even in the first case what you really need to do know is in order to make this learning program practical the raw board configuration is not very easy to deal with because there are so many raw board configurations. So, to handle games like Chess, Ludo, Backgammon where there are so many configurations what you try to do is you try to get certain features from analyzing the raw board. For example the features could be the number pieces of the opponent which are in his home base, number pieces of opponent which are in unprotected situations, whether your die roll is able to capture an opponent. Therefore like this you can set up some features.

For example, if you have twenty or thirty features then for your current board position you can evaluate the values of these features and then from the examples you have you can abstract them in the features phase. For each instance you can find the values of these features and then you can see what the possible moves are and from this you can learn.