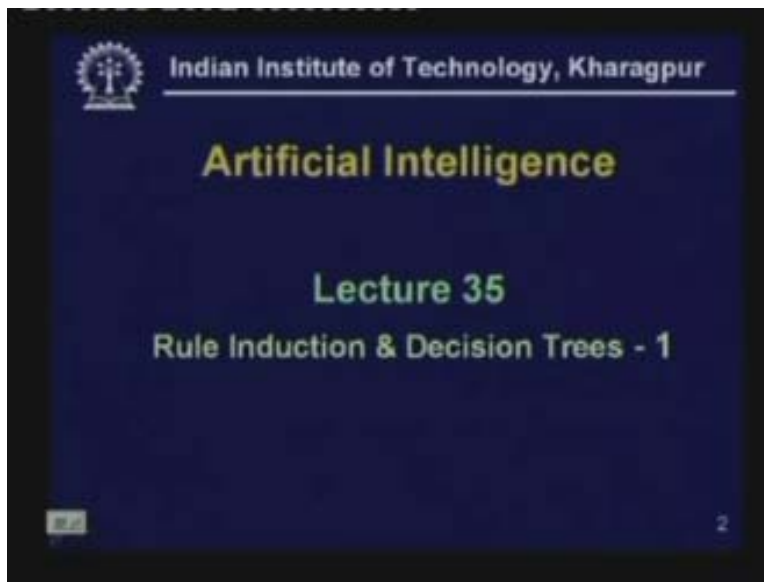


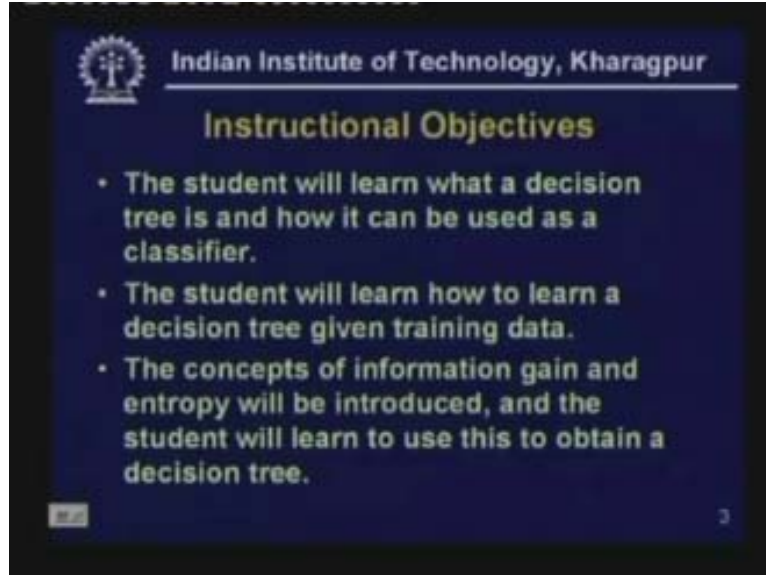
**Artificial Intelligence**  
**Prof. Sudeshna Sarkar**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**  
**Lecture - 34**  
**Rule Induction and Decision Trees - I**

Welcome, in the previous lectures we have looked at the general idea about what we mean by machine learning and we have also introduced you to the task of concept learning or classification. In today's class we are going to study algorithms for learning decision trees which can be used as classifiers for the concept learning task.

(Refer Slide Time: 01:14)



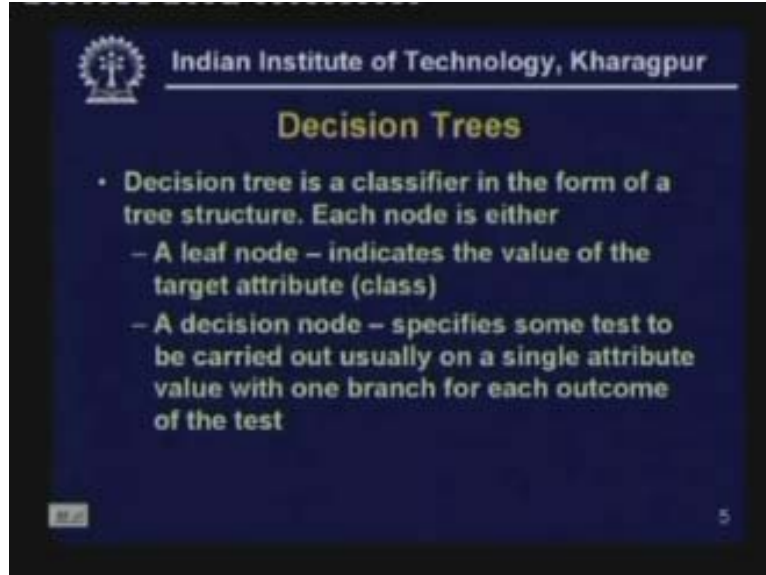
(Refer Slide Time: 01:27)



In today's lecture the objective is that the students will become familiar with the definition of a decision tree as what a decision tree is and how a decision tree can be used as a classifier. The student will also learn an algorithm to construct a decision tree from a given set of training examples, induction of decision trees. We will also introduce the student to the concept of information gain and entropy and how these are used to construct a decision tree.

Later we will review how learning algorithms are reviewed in terms of the training set error and test set error. We will also show how over-fitting occurs in decision trees and in a later class we will look at how over-fitting can be avoided in decision trees. So let us come to know what a decision tree is.

((Refer Slide Time: 02:43))



A decision tree is a classifier in the form of a tree structure. A decision tree is a tree through which we make decision. So the decision tree has a number of nodes. The leaf nodes indicate the value of the class or the value of the target attribute. So you have a set of training examples, the training examples consist of number of features or attributes.

One of the attributes is the target attribute which denotes the class that you are trying to learn. In the decision tree we will just look at examples of decision presently, the leaf nodes correspond to one of the classes the classifier is trying to learn. And the internal nodes are called decision nodes. The internal nodes usually specify some test to be carried out. And usually for most of the algorithms we will study the decision node usually tests the value of a single attribute. And corresponding to each value of the attribute you follow one of the branches that come out of this node. Let us now look at an example of a decision tree.

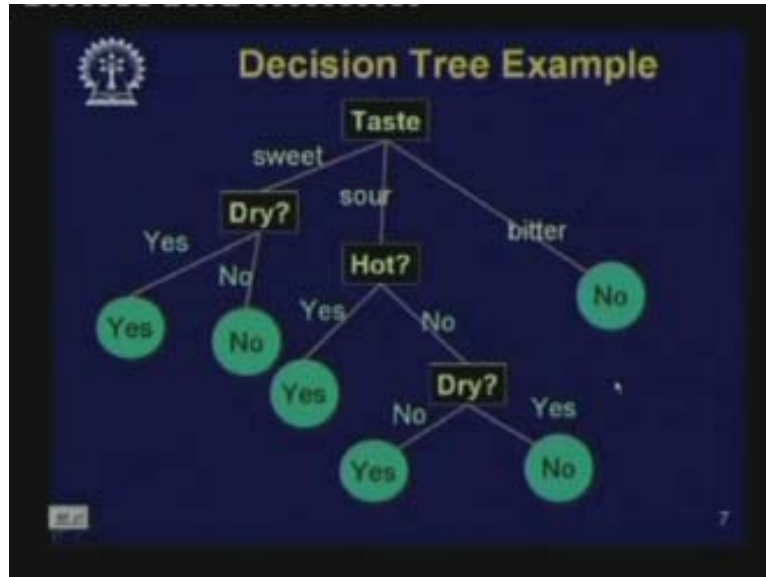
(Refer Slide Time: 04:01)



So this is a decision tree which has been constructed to decide whether for particular loan applicant whether the loan should be approved or rejected. So at the route the attribute employed is checked. So, employed in this case is a Boolean attribute. If employed is no if the person is not employed then the attribute credit score is checked. If the attribute credit score has value high then the loan is approved. If the attribute credit score has a value low then the loan is rejected.

On the other hand, if the person is employed then income is checked. If income is high the loan is approved if income is low the loan is rejected. So this is an example of a decision tree. As you can see there are four leaf nodes in this decision tree two of them correspond to approve and two of them correspond to reject. There are three internal nodes particular decision tree corresponding to the attribute employed credit score and income. And from every internal node there are number of branches corresponding to the different values that this attribute can take. employed can take the value of yes or no, credit score can take the value of high or low, income can take the values of high or low. So this is an example of a decision tree.

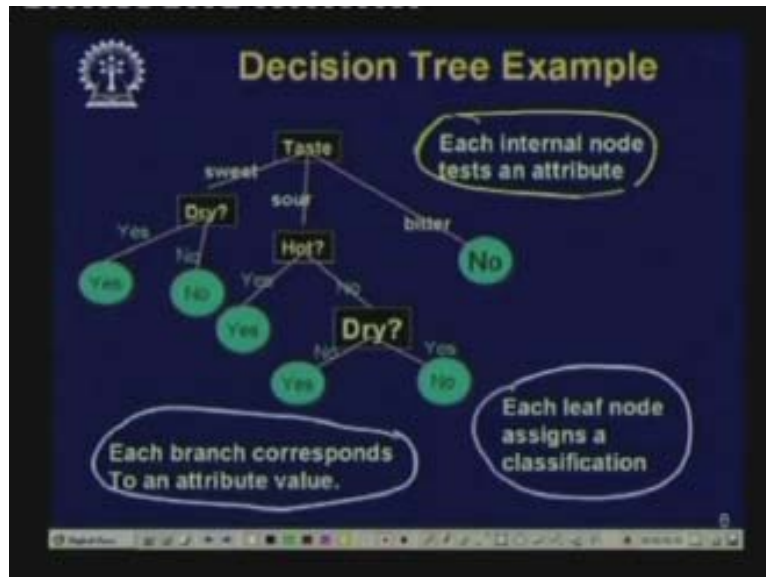
(Refer Slide Time: 5:49)



This is an example of another decision tree and this decision tree has been constructed to find out whether a person likes a particular type of food. There are several attributes that we have used; some of this is taste, dry, hot and so on. Test can be sweet, sour or bitter; dry can be yes or no; hot can be yes or no. Now look at this decision tree, the other root we check the attribute taste. So, if taste is sweet then you check the attribute dry, if dry is yes then this food is liked the answer is yes, if dry is no answer is no, if taste is our then the attribute hot is checked, if hot is yes then the food is liked, if hot is no then you check the attribute dry, if dry is no the food is liked, if dry is yes the food is not liked, if taste is bitter the food is not liked. Therefore this is an example of another decision tree.

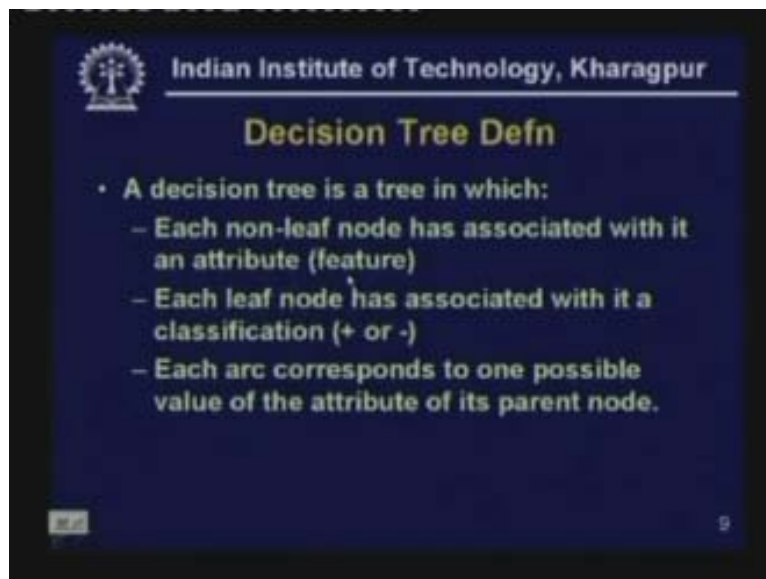
Here the depth of the different leaves vary. For example, the taste is bitter; this food is not liked irrespective of the other attributes. If the food is sour then if it is hot it is liked and if it is not hot but it is not dry also then it is liked otherwise it is not liked and so on. Now you note that each internal node in the decision tree tests an attribute, each leaf node assigns a classification and each branch corresponds to an attribute value. There are four internal nodes here

(Refer Slide Time: 08:05)



Taste, dry, hot and dry and each of them tests one attribute and each leaf node assigns a classification liked or not liked, each branch corresponds to an attribute value. So let us come to the formal definition of a decision tree. A decision tree is a tree in which each non leaf node has associated with it an attribute or a feature.

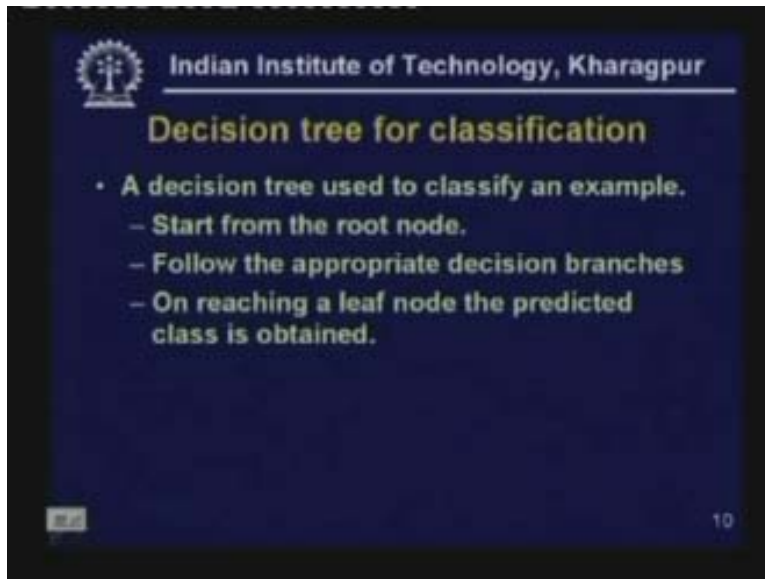
(Refer Slide Time: 08:33)



Each leaf node has associated with it a classification. In case the concept learning problem is a two class problem the classes can be plus or minus. In general there could be a finite number of classes. Each arc can be decision tree corresponds to one possible value of the attribute of its parent node. So, the decision tree is defined in terms of

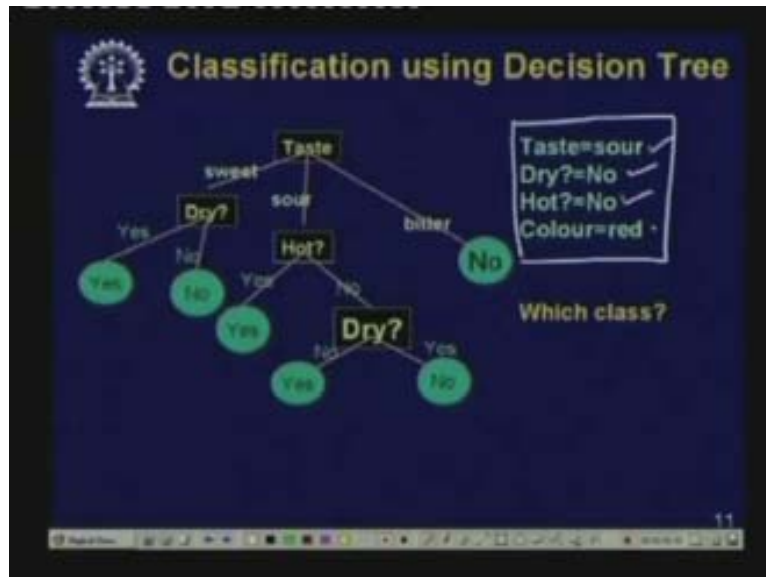
components, the internal nodes, the leaves and the branches. The leaves are the classes, the internal nodes are the decision nodes where an attribute value is checked, the branches coming out of the decision tree correspond to different values that the attribute can take. A decision tree can be used for classification. It is used to classify an example using a decision tree.

(Refer Slide Time: 09:36)



You start from the root node, you follow the appropriate decision branches corresponding to the attribute values of the example that you have got. On reaching a leaf node the predicted class is update. For example, let us look at this decision tree and let us look at this test data. We have test data whose taste is sour, dry is no, hot is no and color is red.

(Refer Slide Time: 10:13)



Now for this test data let us apply this to the decision tree. So initially we come to the root; taste is sour, so we will follow this branch. So we first check the value of taste, taste is sour to follow this branch we check the value of hot, hot is no so we follow this branch check the value of dry, dry is no so we come to this leaf and this leaf is yes the food is liked. You note that in the training set there could be other attributes for example color which has been not been used in the decision tree because the decision tree finds the correct class without consulting this attribute tree either along this path or may be in the entire tree. For example, color is not used in this entire tree.

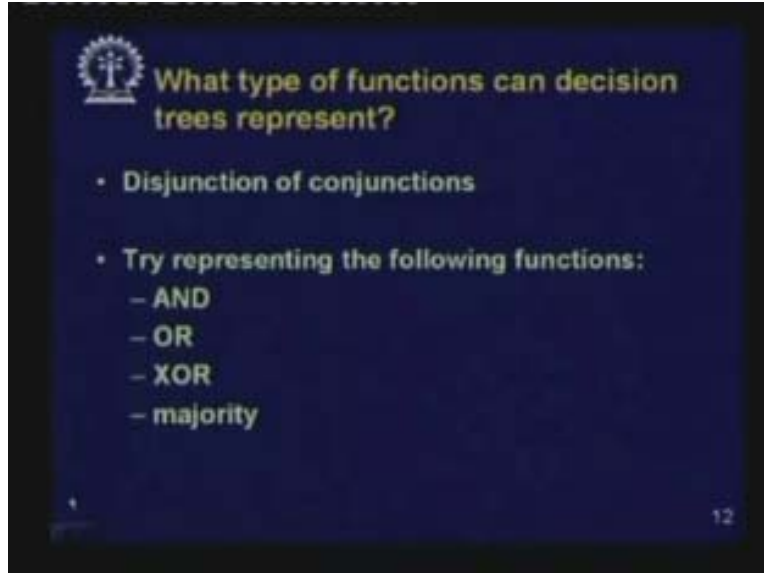
Suppose there is another training set whose taste is sweet let us say dry is yes, hot is yes, color equal to yellow. Now to find the class of this training example using the decision tree one would follow this path: taste is checked and taste is sweet then dry is checked dry is yes so the food is liked. The attributes hot or color are not checked. Now with this introduction I will ask you to see what a decision tree can represent.

What are the types of functions that can be represented by a decision tree?

Let us assume for simplicity that all the attributes are Boolean attributes and let us also assume that the classification problem is a Boolean classification problem. In a decision tree you follow the different branches and there are some leaves at the bottom, now some of the leaves are classified as yes, some of the leaves are classified as no. So, if there are two yes leaf the concept is true if you either reach this or that leaf. Therefore it is true if either of the leaves is reached. Corresponding to each leaf you follow a path and you reach that leaf if the decision you make from the root to that leaf then all the decisions follow a particular branch. So a particular leaf can be represented as conjunction of values and if you have more than one leaf which amongst is positive you can treat them as a disjunction of those conjunctions. So, you can easily see that a decision tree in general can represent a disjunction of conjunctions.



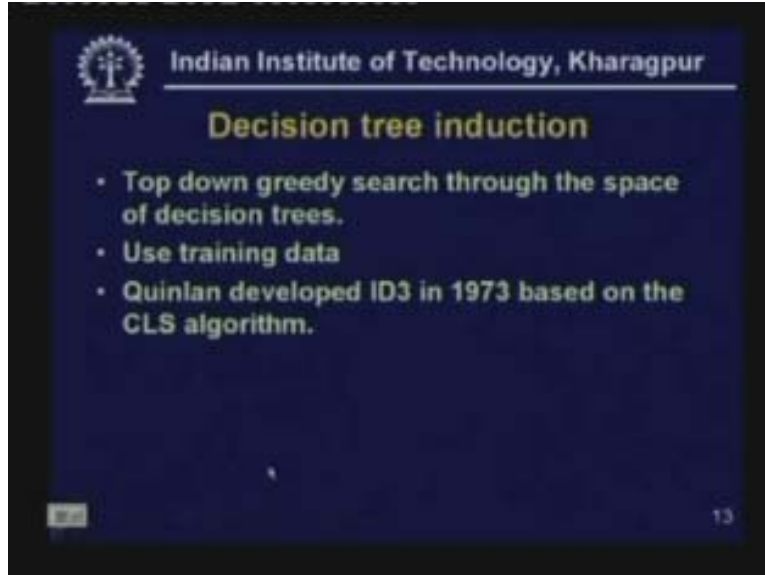
(Refer Slide Time: 13:45)



Any disjunction of conjunctions can be represented by a decision tree. As an exercise I will ask you to look at the following functions:

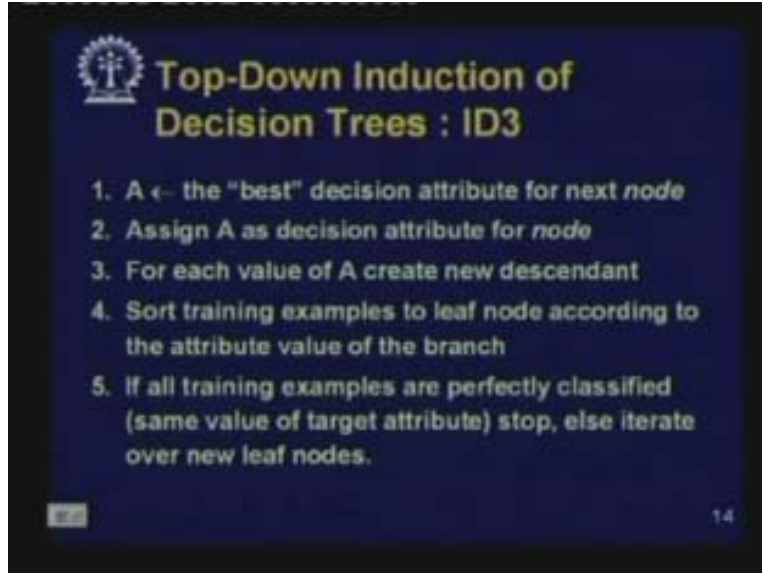
For the first three assume that you have two attributes  $a$  and  $b$ . So you are required to represent the AND function, the OR function and the XOR functions for the attributes  $a$  and  $b$ . In other words, you have to find a decision tree to represent  $a \text{ AND } b$ , another decision tree to represent  $a \text{ OR } b$ , another decision tree to represent  $a \text{ XOR } b$ . For the fourth case I will ask you find a decision tree to represent the majority function given that you have five attributes  $a$ ,  $b$ ,  $c$ ,  $d$  and  $e$ . So you want to find the majority of these five attributes. Please try this and let us see whether you can come up with the decision tree. Now let us go to the learning problem for the decision tree.

(Refer Slide Time: 15:19)



You have seen that, given a decision tree you can use a decision tree so that given a training example you can find the class of the training example. Now given the training set how can you get a decision tree? In the learning problem you will be given the set training examples, by processing the training examples you will come up with the decision tree and you will use this decision tree to classify new training examples. So the technique that we will employ is a greedy technique to find a decision tree given the training data. Now the decision tree algorithms have been arrived at by different people in different fields because of lot of applications. The basic algorithm that we will talk about today is based on the ID3 algorithm which was developed by Ross Quinlan based on the CLS algorithm.

(Refer Slide Time: 16:29)



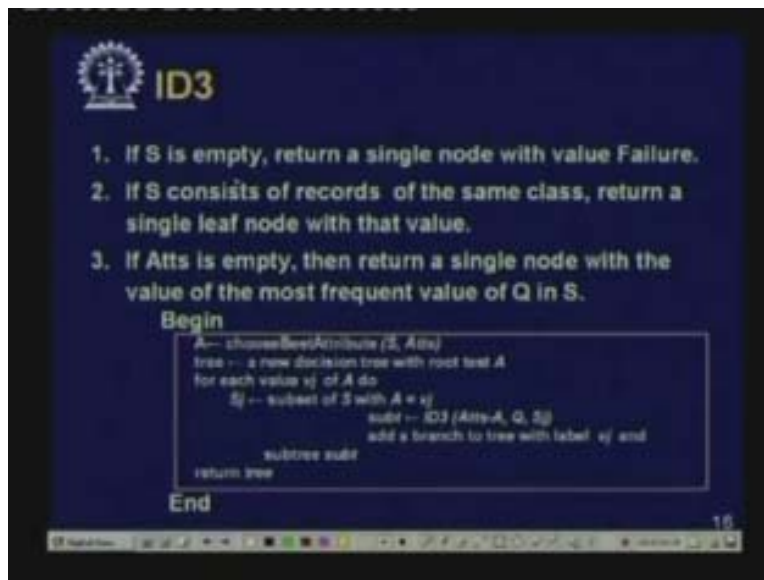
Now the algorithm that we will talk about is a greedy algorithm. You are given to training data and you want to come up with the decision tree. Now there could be different decision trees, a large number of decision trees that you could look for. So instead of searching the entire space of decision trees what we will do is use a simple greedy algorithm to induce decision tree. The algorithm is a top down greedy algorithm. It starts by deciding which node to make a root node. Once the root node is fixed it tries to find the children of that node then fixes each of these nodes and tries to find out those nodes. So this algorithm is based on top down induction of decision trees.

Now let us look at the basic steps in the algorithm. In the first step you try to find out the best decision attribute for the next node. So you have got with you a set of training examples let us say yes. Now based on yes you try to find that attribute A which is adjudged to be the best decision attribute for this node. Now you fix A to be the decision attribute of this node. So you make the root of the decision tree of the current node. Now you find out that all possible values that A can take suppose A can take the values true and false for each value of A you create a new descendant.

Now these new descendants can be a leaf node or an internal node. Now, if all the training examples for which A is true if all these training examples belong to the same class that is all of them are positive or all of them are negative make this a leaf node and label it by the value of the target attribute. If you see that the examples which come to this branch are a mixture of positive and negative examples then you try to make another tree to fit here so you do this recursively. So here you find the next attribute which is adjudged to be the best attribute for corresponding to the different values of the attribute you again have to construct a decision tree for its children. This is the sense of the decision tree learning algorithm.

Step 1) Let  $A$  be the best decision attributer for the next node. Assign  $A$  as decision attribute for node. For each value of  $A$  create a new descendant. Initially you had the set of training examples  $S$ , you find that subset of  $S$  for which  $A$  is true they will come here you find that a subset of  $A$  for which  $A$  is false and they will come here. So you sort the training examples to the appropriate leaf node according to the attribute value of the branch. If all training examples are perfectly classified stop otherwise iterate this algorithm over the new nodes. We will now describe the ID3 algorithm of Quinlan. This was one of the oldest algorithms devised by Quinlan. In the next class we will look at the newer versions of ID3 algorithm **c 4.5, c 5.0 carts etc**. So the ID3 algorithm has inputs, the set of attributes  $Atts$ ,  $q$  is the target attribute and  $s$  is the set of training example. So you are given a set of training examples, you are given a set of non target attributes  $Atts$  and the target attribute is  $q$ . and the ID3 algorithm will take these inputs, construct a decision tree and return the decision tree. **Now let us look at the details of this algorithm.**

(Refer Slide Time: 21:36)

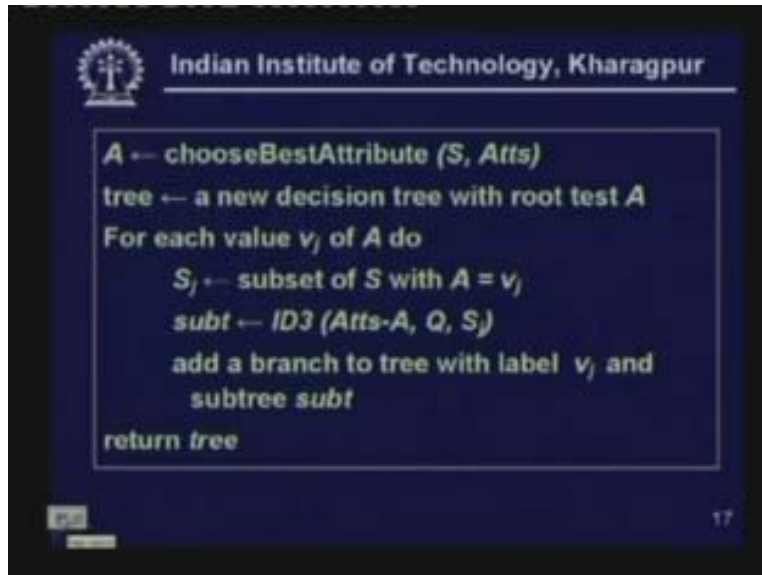


If  $S$  is empty, that is, there, no examples left, return a single node with value failure or return a single node with a default class.

Step 2) If  $S$  is not empty but consists of records of the same class. So, if all the examples in  $S$  belong to the same class you make this node a leaf node with the value of that class and label it with the value of that class. Otherwise if  $Atts$  is empty that is there are no more attributes left then return a single node and label this node with the majority classification of the examples. So  $S$  is the set of examples that you have, you find a majority class that these example belong to and since there are no more attributes left to make a decision node you make a leaf there with the majority classification. Otherwise that is, if  $S$  is non empty but  $S$  is not homogeneous that is  $S$  consists of both positive as well as negative examples and  $Atts$  is not empty then what you do is you choose the best attribute out of  $Atts$ . So  $Atts$  is the attribute you have left choose the best attribute out of

Atts and let that be A and how do you choose the best attribute? You do it by processing the training set S. Remember, this is a greedy algorithm.

(Refer Slide Time: 23:16)



Now once you have chosen this decision attribute a you will try to construct a decision tree which will have A as its root. So you will construct a tree and initially let tree be the new decision tree whose root is A. Now for the attribute a you find all the value  $V_j$  that a can take.

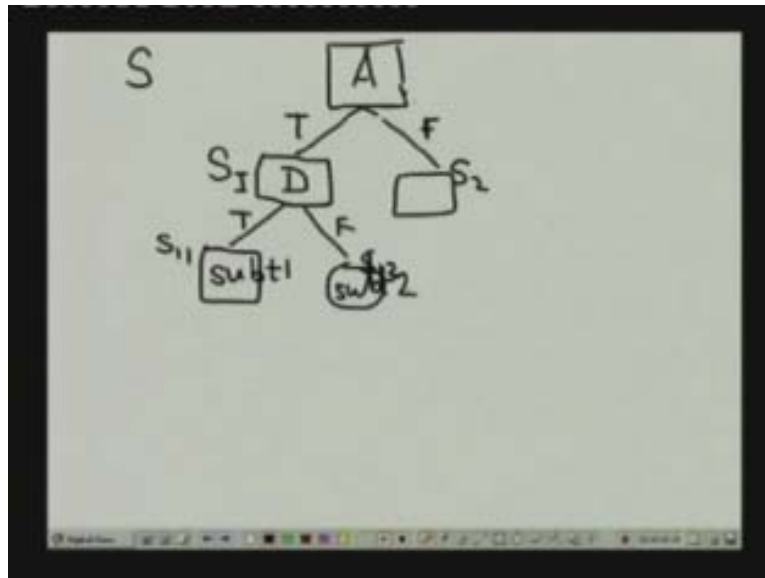
Let  $S_j$  be the subset of S with A is equal to  $V_j$ . So you have all the training examples S you find all the values that this attribute can take and you make so many branches. In each branch you push those examples for which the attribute of A has the value of  $V_j$ . So the training set that you have at the root will be partitioned into several subsets. So the subset  $S_j$  will contain those training examples for which A is equal to  $V_j$ . Now at this branch we will make a subtree which you denote by sub t and you get it by recursively calling ID3 with the remaining attributes Atts minus A. the target attribute Q and the subset of training examples are this node  $S_j$ .

Why Atts minus A?


At the current node you have used up the attribute A so you have come to this branch for A is equal to  $V_j$ . Now this attribute will no longer be available for further deciding the class. So you remove A from your attribute set and you call the decision tree algorithm recursively on the remaining examples  $S_j$  that come to this branch. Now you will add a branch to the tree with label  $V_j$  and subtree sub t and you will return this tree. Therefore you start with a training examples S. You find that attribute A is the best attribute for this training example. Once you fix this attribute you look at different values this attribute can take.

Suppose that you can take true and false what you will do is that corresponding to this you find out whether all the examples that come here so all the examples that come here let us say they are called  $S_1$  and  $S_2$ . Now if  $S_1$  is only positive and only negative this will be a leaf node otherwise we select another attribute which is adjudged the best attribute with this node. And corresponding to that we will again sort the training example  $S_1$  into  $S_{11}$  and  $S_{12}$ ,  $S_{11}$  for  $D$  is equal to true and  $S_{12}$  for  $D$  is equal to false and then you will get this new node and corresponding to this new node you will construct a subtree and you will put this subtree on the left side here and corresponding to this you will construct another subtree which you will put here. This is how the decision tree learning proceeds.

(Refer Slide Time: 27:48)



(Refer Slide Time: 27:56)

 Indian Institute of Technology, Kharagpur

### Greedy selection

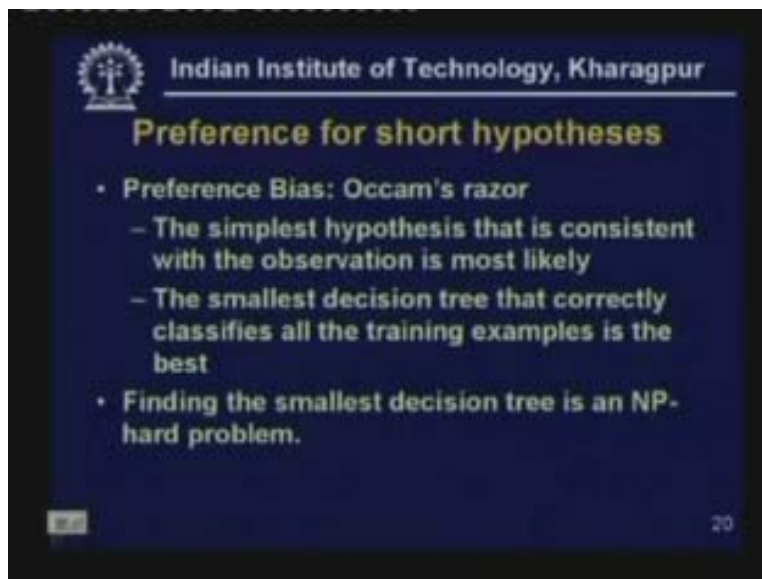
- ID3 tries to find the attribute that best separates the training examples.
- The algorithm uses a greedy search, and never backtracks.

19

Thus this algorithm is a greedy algorithm. It does greedy selection. ID3 tries to find the attribute that best separates the training examples. You have some training examples and you try to find an attribute. Which attribute will you adjudge the best? The ID3 stops when you get to a leaf where all the examples have the same class. So, if  $S$  is a mixture of positive and negative examples you try to use an attribute which will ultimately put the positive and negative examples separately. So you try to select that attribute which can achieve a separation from positive and negative examples in the best possible manner. For this the algorithm uses a greedy search and the algorithm never backtracks.

Now which attribute you want to select? There could be several strategies that you can employ. You can select the attribute completely randomly; you can use the heuristic list values. That is, you use the attribute with the smallest number of possible values. Another heuristic is you can choose max values. That is, choose the attributes which has the largest number of possible values. Or you can choose the heuristic max gain which corresponds to choosing the attribute with the largest expected information gain. So this max gain is the heuristic we will study and it has been found to be a very effective heuristic for learning decision trees.

(Refer Slide Time: 31:15)



What is the motivation behind this heuristic?

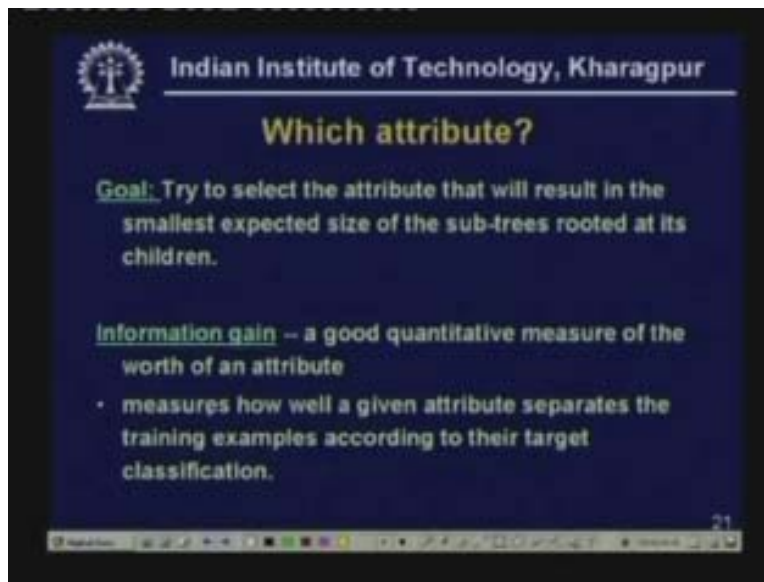
In any inductive learning algorithm we use a bias which tells us whether to prefer one particular classifier over the other. The sort of bias that we use in this case is a preference for short classifier or a preference for short trees. So we will just recall a principle which has been widely known as the occam's razor. Occam's razor decrease that entities should not be multiplied unnecessarily. It means that the simplest hypothesis should be chosen over more complex ones if they have the same performance over the data.

Suppose you have some training data and there are a number of decision trees possible which are all consist with the training data then you should select the one which is



shortest which is simplest. In other words the smallest decision tree that correctly classifies all the training examples should be selected. Unfortunately trying to find the smallest decision tree that is consistent with examples is known to be an NP-hard problem. It is a very difficult problem and it is not easy to find the smallest decision tree. So what we do is we use a heuristic instead and this heuristic tries to find a smallest decision tree. And we try to use this idea use this heuristic in order to decide which attribute to select. So our goal is, we try to select the attribute that will result in the smallest expected size of the subtree rooted at the children.

(Refer Slide Time: 32:00)



Indian Institute of Technology, Kharagpur

## Which attribute?

**Goal:** Try to select the attribute that will result in the smallest expected size of the sub-trees rooted at its children.

**Information gain** – a good quantitative measure of the worth of an attribute

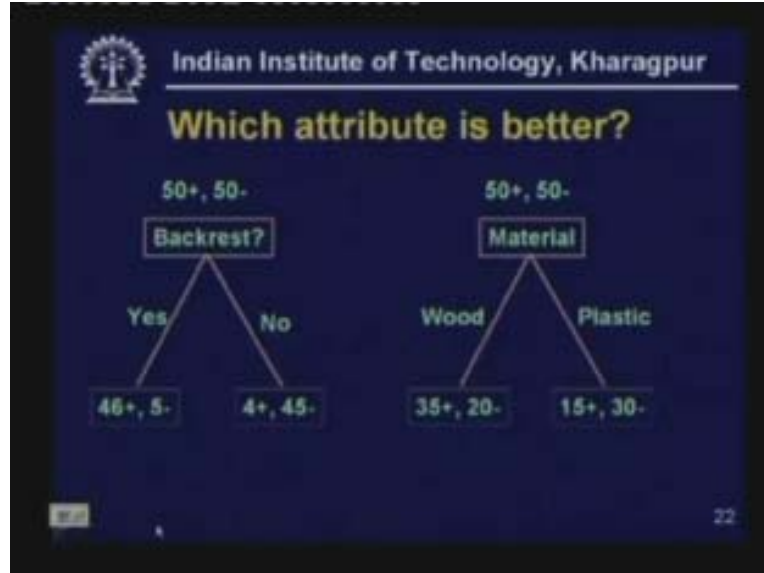
- measures how well a given attribute separates the training examples according to their target classification.

21

So we have a choice of attributes and we try to select that attribute for which the subtrees rooted that the children will be small. And a heuristic that we will use is the information gain heuristic. The information gain heuristic is a good quantitative measure of the worth of an attribute. It measures how well a given attribute separates the training examples according to their target classification. Your objective is to achieve a separation from the positive and from the negative examples. So you try to select that attribute which achieves this separation maximally. So this is the heuristic that you use and information gain can do this quite effectively.



(Refer Slide Time: 33:05)



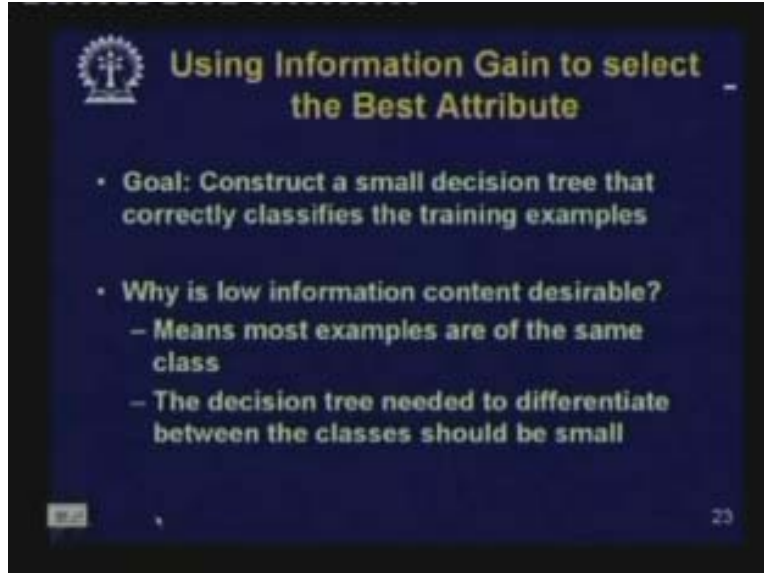
Now let us look at some examples. Suppose you have some objects and you want to classify them as being a chair or not a chair. Suppose initially you have 50 positive examples and 50 negative examples and you have a choice of many attributes. Let us say two of the attributes are backrest and material. Suppose the attribute backrest has values yes and no. among objects of backrest yes there are 51 such objects 46 of them are chairs and 5 are non chairs. There are 49 objects with no backrest, 4 of them are chairs, 45 are non chairs. On the other hand, if you look at the attribute material the value of the material can be wood or plastic.

For material equal to wood there are 55 objects there are 35 positive and 20 negative and for material equal to plastic there are 45 objects in which 15 is positive and 30 negative. Now which of these attributes is better?

Initially, the examples are a mixture; it is a uniform mixture of 50 positive and 50 negative examples. So, each one being a chair or a non chair is equally likely. For backrest is equal to yes we see that a large majority are chairs or few small ones are non chairs. For backrest equal to no a large majority are non chairs and only a few are chairs. For material equal to wood the majority is chair 35 and 20 are non chairs but here the percentage of positive examples is not very high.

Similarly, here 15 plus and 30 minus more negative but this sort of distribution is less homogeneous than the distribution you get when you use backrest. So, if you really want to achieve a separation from the positive and negative examples may be this is likely to be a better option. A backrest is likely to be a better option to distinguish between the positive examples from negative examples. This however does not guarantee that inconjunction with another feature the material may prove to be a very good option. So we are really not looking ahead but only making a greedy choice based on the current information. So we will use information gain to select the best attribute.

(Refer Slide Time: 36:09)



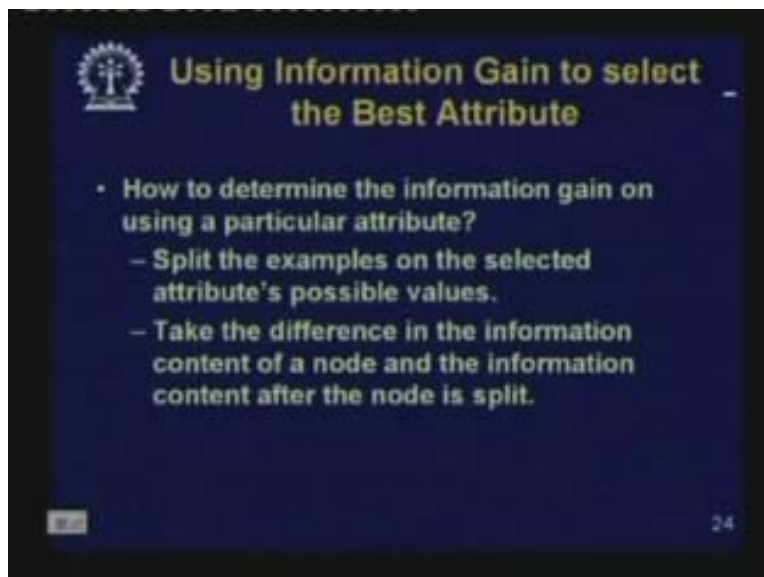
Using Information Gain to select the Best Attribute

- Goal: Construct a small decision tree that correctly classifies the training examples
- Why is low information content desirable?
  - Means most examples are of the same class
  - The decision tree needed to differentiate between the classes should be small

23

Our goal is to construct a small decision tree that correctly classifies the training example. And we will look for the attribute for which we get examples with low information content. That is a more homogeneous example set. That is, most of the examples belong to the same class. Our heuristic is that if the examples we have mostly belong to the same class then the decision tree that will be needed to classify those examples would be smaller. So this is the idea based on which we use this sort of heuristic.

(Refer Slide Time: 37:05)



Using Information Gain to select the Best Attribute

- How to determine the information gain on using a particular attribute?
  - Split the examples on the selected attribute's possible values.
  - Take the difference in the information content of a node and the information content after the node is split.

24

Now how will we determine the information gain on using a particular attribute?

First of all we have to have a measure for the information content of the set of examples. We will presently see that we use the concept of entropy to find the information content or a measure of purity and a purity of the set of examples. Once we have the measure of information content we look at different situations that we will get by using different attributes at the root. So, when we use an attribute A at the root we split the examples into different classes corresponding to the different values of A.

Now we find the resulting sets and find the information content of each of the resulting sets. Take the weighted average and compare it with the original information content. If the information content becomes lower this is a reasonable attribute to take. So we find out that attribute for which the information content becomes the lowest after splitting on that attribute and we will select that attribute. So we are trying to determine the information gain on using a particular attribute and we will split the examples on the selected attributes possible values and we find the difference in the information content of a node and the information content after the node is split.

(Refer Slide Time: 38:58)

Indian Institute of Technology, Kharagpur

## Entropy

- S is a sample of training examples
- p is the proportion of positive examples
- n is the proportion of negative examples
- Entropy measures the impurity of S

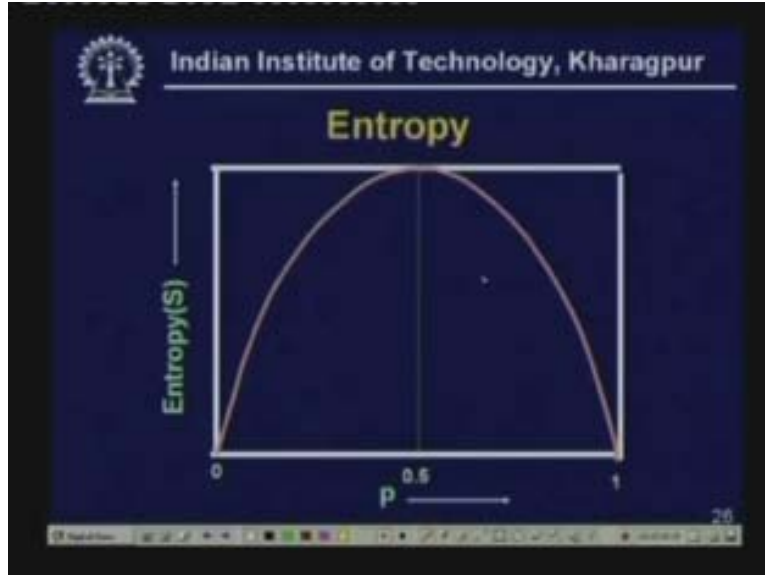
$$\text{Entropy}(S) = -p \log_2 p - n \log_2 n$$
$$\text{Entropy}(S) = -\sum_c p_i \log_2 p_i$$

25

To measure the information content of a set of examples S we will use the concept of entropy. Suppose S is a sample of training examples that we have and let us say in this sample S p is the proportion of positive examples and n is the proportion of negative examples, so entropy measures the impurity of S, it is defined to be minus p log b minus n log n where p is the proportion of positive examples, n is the proportion of negative examples and the algorithm is base two. So this definition is if we have only two classes.

In general if you have a number of classes then entropy of the set of examples is defined to be minus sigma of pi log pi where pi denotes the fraction of examples which belong to class c<sub>i</sub> and we submit over all c<sub>i</sub>. Therefore this is the general definition of entropy when you have arbitrary number of classes. The entropy function can be plotted for the two class problem as shown here. This is an approximate shape of the entropy curve.

(Refer Slide Time: 40:38)



We have plotted entropy on the y axis with the value of p on the x axis. We see that when p is 0 entropy is 0 when p is 1 entropy is 1 and the curve has a shape like this so we see that entropy is maximum when p is 1 by 2. When p is 1 by 2 entropy is 1 when p is 0 and 1 entropy is smallest. So the impurity of the data is maximum when p is 1 by 2 that is we have a exactly equal number of positive and negative examples then the impurity is maximum. When impurity is smallest then either p is 1 and all examples are positive and p is 0 where all examples are negative. If p is 1 by 2 we have a mixture so entropy is the measure of the impurity.

(Refer Slide Time: 40:38)

**Entropy**

Entropy(S)= expected number of bits needed to encode class (+ or -) of randomly drawn members of S.

Why?

- Information theory optimal length code assign  $-\log_2 p$  bits to messages having probability p.
- So the expected number of bits to encode (+ or -) a random member of S:  
 $-p \log_2 p - n \log_2 n$

27

In information theory entropy is actually interpreted as the expected number of bits to encode a class. Suppose you have two classes plus or minus and you get randomly drawn examples of  $S$  entropy denotes the expected number of bits to encode a class. Suppose all the class is positive then you require 0 bits to indicate whether the objective belongs to positive or negative class because you know all classes are positive. If all classes are negative also the same thing will happen. If classes can be either positive or negative with equal probability you can always use one bit to encode the classes as 0 or 1.

Suppose positive classes are more frequent than negative classes what you can do is that you can choose an encoding so that positive class gives smaller number of bits than a negative class. So patterns involving positive classes will be more likely and to encode them you can use smaller number of bits. In information theory we find optimal length code and it is found that if a message has probability  $p$  then you require an optimally  $-\log_2 p$  bits to encode that message. So the expected number of bits to encode a random member of  $S$  by information theory is given by  $-\sum p \log_2 p$  which is how entropy is defined. We will use this concept in deciding the information content of a set of examples that we get in a decision tree. So, entropy characterizes the impurity of a collection. Just to give an example suppose  $S$  is a collection of 6 positive and 4 negative examples entropy of  $S$  can be computed as  $-\frac{6}{10} \log_2 \frac{6}{10} - \frac{4}{10} \log_2 \frac{4}{10}$  which can be evaluated to be 0.97.

(Refer Slide Time: 44:11)

Indian Institute of Technology, Kharagpur

## Entropy

- Entropy characterizes the impurity of a collection.
- Suppose  $S$  is a collection of 6 positive and 4 negative examples.
- Entropy( $S$ ) =  
$$-\frac{6}{10} \log_2 \left(\frac{6}{10}\right) - \frac{4}{10} \log_2 \left(\frac{4}{10}\right)$$
$$= 0.97$$

28

(Refer Slide Time: 44:18)

Indian Institute of Technology, Kharagpur

## Entropy

- If the target attribute takes  $m$  different values then the entropy of  $S$  with respect to the  $m$ -wise classification is defined as
- Entropy( $S$ ) =  $-\sum_{i=1}^m p_i \log_2 p_i$

$p_i$  is the proportion of  $S$  belonging to class  $c_i$

29

If the target attribute takes  $m$  different values that is there are  $m$  classes then the entropy of  $S$  with respect to the  $m$  wise classification is defined as minus sigma  $p_i \log p_i$  and this  $p_i$  is the proportion of  $S$  belonging to class  $c_i$ . Now how would we use information gain to select the best attribute. So we introduce the concept of remaindering.

(Refer Slide Time: 44:46)

Indian Institute of Technology, Kharagpur

## Using Information Gain to select the best attribute

- Define *Remainder (A)* as the weighted sum of the information content of each subset of the examples partitioned by the possible values of the attribute.
  - Measures the total disorder or inhomogeneity of the children nodes.

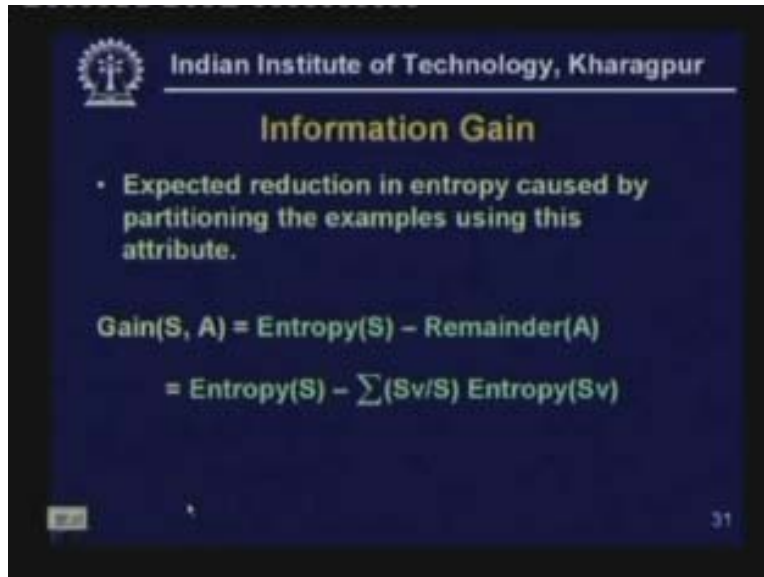
30

Suppose we have a set of training examples  $S$  we can find its entropy. When we classify using  $A$  we split  $S$  into different subset. We define remainder  $A$  as the weighted sum of the information content of each of these subsets. So  $S$  is partitioned into subsets by possible values of the attribute, remainder  $A$  is the weighted sum of the entropy of the



subsets. So remainder A can be taken to measure the total disorder or in-homogeneity of the children nodes when you classify using the attribute A. So the expected reduction in entropy caused by partitioning the examples using the attribute A is given by gain (S, A).

(Refer Slide Time: 45:36)



Indian Institute of Technology, Kharagpur

### Information Gain

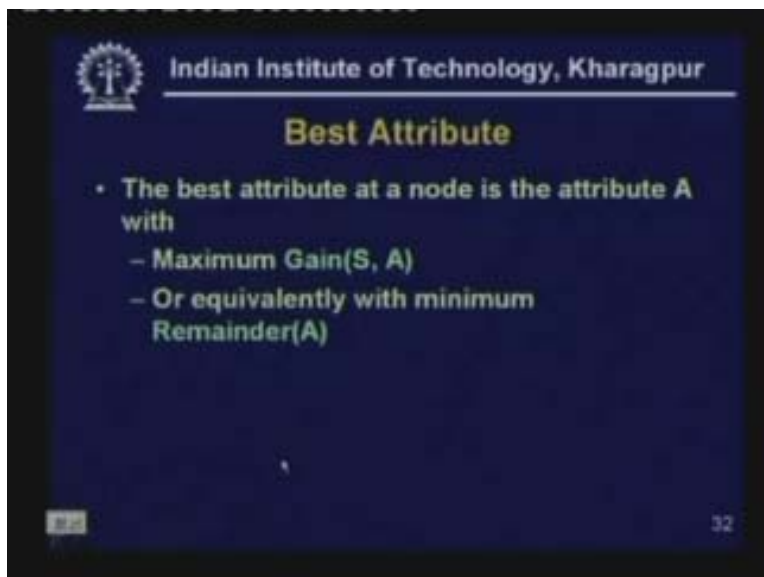
- Expected reduction in entropy caused by partitioning the examples using this attribute.

$$\text{Gain}(S, A) = \text{Entropy}(S) - \text{Remainder}(A)$$
$$= \text{Entropy}(S) - \sum (S_v/S) \text{Entropy}(S_v)$$

31

So the gain (S, A) entropy S minus remainder A and remainder A is the weighted sum of entropy (S<sub>v</sub>). (S<sub>v</sub>) is the subset of S for which the value is V and we sum it over all attribute values that A can take.

(Refer Slide Time: 46:24)



Indian Institute of Technology, Kharagpur

### Best Attribute

- The best attribute at a node is the attribute A with
  - Maximum Gain(S, A)
  - Or equivalently with minimum Remainder(A)

32

So we define the best attribute to be the attribute for which gain (S, A) is maximum or equivalently for which remainder A is minimum. Now let us look at an example of decision tree. We have some training data; the training data has three attributes state, season and barometer and the attribute which we are trying to learn is weather.

(Refer Slide Time: 46:44)

Indian Institute of Technology, Kharagpur

### Decision Tree Example

State	Season	Barometer	Weather
AK	Winter	Down	Snow
HI	Winter	Down	Sun
HI	Summer	Up	Sun
CA	Summer	Up	Rain
AK	Winter	Up	Snow
CA	Winter	Down	Sun
AK	Summer	Down	Sun
CA	Winter	Up	Rain
HI	Summer	Down	Sun

Predicting the weather  
Target attribute = Weather  
Attributes = State, Season, Barometer

33

State can be AK, HI, CA and season can be winter or summer, barometer can be down or up. And we are trying to predict whether the weather is snow or sun or rain. So the target attribute is weather and the other attributes are state, season and barometer.

(Refer Slide Time: 47:10)

Indian Institute of Technology, Kharagpur

### Decision Tree Example

State	Season	Barometer	Weather
AK	Winter	Down	Snow
HI	Winter	Down	Sun
HI	Summer	Up	Sun
CA	Summer	Up	Rain
AK	Winter	Up	Snow
CA	Winter	Down	Sun
AK	Summer	Down	Sun
CA	Winter	Up	Rain
HI	Summer	Down	Sun

State:  
AK: 2 Snow, 1 Sun → 0.92  
HI: 3 Sun → 0.00  
CA: 2 Rain, 1 Sun → 0.92  
Entropy = 0.82

Season:  
Winter: 2 Snow, 2 Sun, 1 Rain → 1.52  
Summer: 3 Sun, 1 Rain → 0.81  
CA: 2 Rain, 1 Sun → 0.92  
Entropy = 1.20

Barometer:  
Down: 1 Snow, 4 Sun → 0.72  
Up: 1 Snow, 1 Sun, 2 Rain → 1.50  
Entropy = 1.07

34



Now let us look at each of these attributes. The attribute state can have three values AK, HI and CA and in AK we find three examples 2 snow and 1 sun, in HI there are 3 sun, in CA there are 2 rain and 1 sun so entropy of AK is 0.92 of HI is 0 of CA is 0.92 so the average reduction in information content is given by 0.62 or average information content is given by 0.62. For the attribute season for the value winter there are 2 snow, 2 sun and 1 rain.

Summer there are 3 sun and 1 rain, for CA there are 2 sun and 1 rain and the average entropy is 1.2. For barometer corresponding to down we have 1 snow 4 sun and corresponding to up we have 1 snow 1 sun 2 rain so the average entropy is 1.07. So we get the minimum entropy if we use state as the attribute and the maximum entropy if we use season as the attribute. Hence, you can use state to split the root.

(Refer Slide Time: 48:41)

**Decision Tree Example**

State	Season	Barometer	Weather
AK	Winter	Down	Snow
AK	Winter	Up	Snow
AK	Summer	Down	Sun
HI	Winter	Down	Sun
HI	Summer	Up	Sun
HI	Summer	Down	Sun
CA	Summer	Up	Rain
CA	Winter	Down	Sun
CA	Winter	Up	Rain

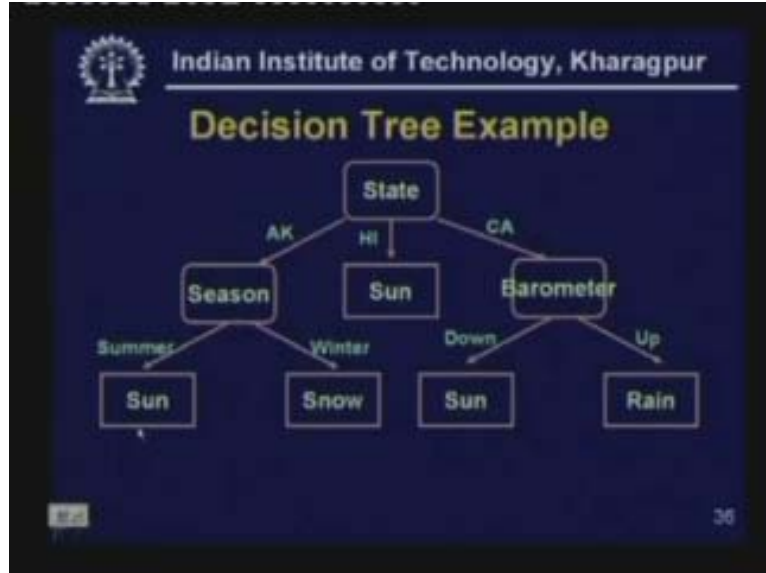
**State = AK:**  
Split on Season  
Winter = Snow  
Summer = Sun

**State = HI:**  
Leaf node = Sun

**State = CA:**  
Split on Barometer  
Up = Rain  
Down = Sun

For state is equal to AK we can split on season, For state is equal to HI we get a leaf node, for state is equal to CA we can split in barometer and we can finally get this decision tree, for state is equal to AK we split on season, if it is summer then sun, for winter it is snow, for state is equal to HI it is always sun, for state is equal to CA the barometer is down, for sun the barometer is up it is rain.

(Refer Slide Time: 48:51)



This is the decision tree that you get from the table by using the information gain heuristic the details you have to work out.

(Refer Slide Time: 49:22)

The true error of hypothesis  $h$  with respect to target function  $f$  and distribution  $D$  is the probability that  $h$  will misclassify an instance drawn at random according to  $D$ .

$$\text{error}_D(h) \equiv \mathbb{P}_{x \sim D}[f(x) \neq h(x)]$$

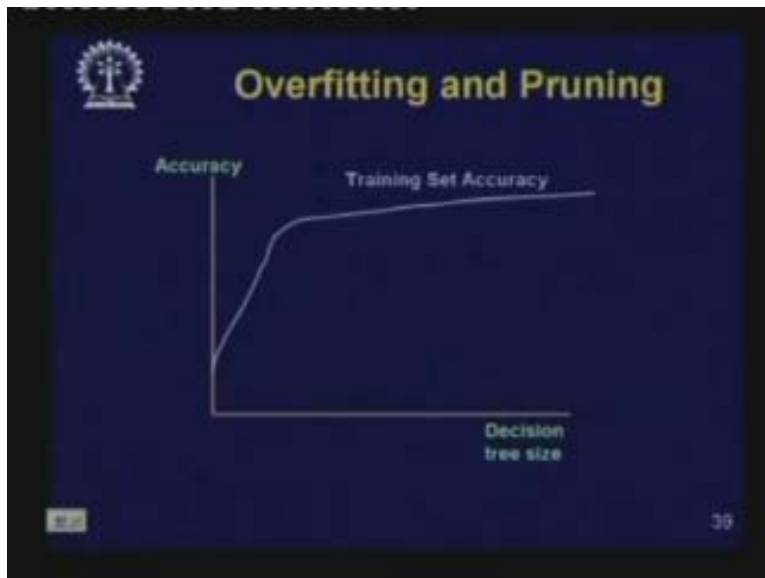
Now we will briefly review how we will evaluate a decision tree. Once we have a constructed a decision tree we want to know how good the decision tree is. When you get a classifier you evaluate the classifier by computing its accuracy or equivalently by finding the error. Now how would you use the classifier?

You would use the classifier to predict the classification of a new training example.

So, given a distribution of possible examples we want a classifier for which the true error given the distribution of examples is smallest. But what you have with you is a set of training examples. You can evaluate the training error with respect to the training examples. This is what you can evaluate but you really want to find the true error. So, to review the true error of a hypothesis  $h$  with respect to target function  $f$  and a distribution  $D$  is the probability that  $h$  will misclassify an instance drawn randomly according to  $D$ . So the true error of the hypothesis  $h$  is the probability that  $h(x)$  does not agree with  $f(x)$ .

The sample error is what you get from the training example. The sample error of  $h$  with respect to target function  $f$  and data sample  $S$  is the proportion of examples that  $h$  misclassifies. We do not have the true error with us but we will use a sample error as an indicator of the true error. But if we use the training error to learn our hypothesis and use the same training set to evaluate the hypothesis we are actually cheating because we have the scope to fit the classifier with the training data. In order to get a better idea of the true error what we do is we keep aside some of the data that we had as the test set which we do not use to train the classifier.

(Refer Slide Time: 52:01)

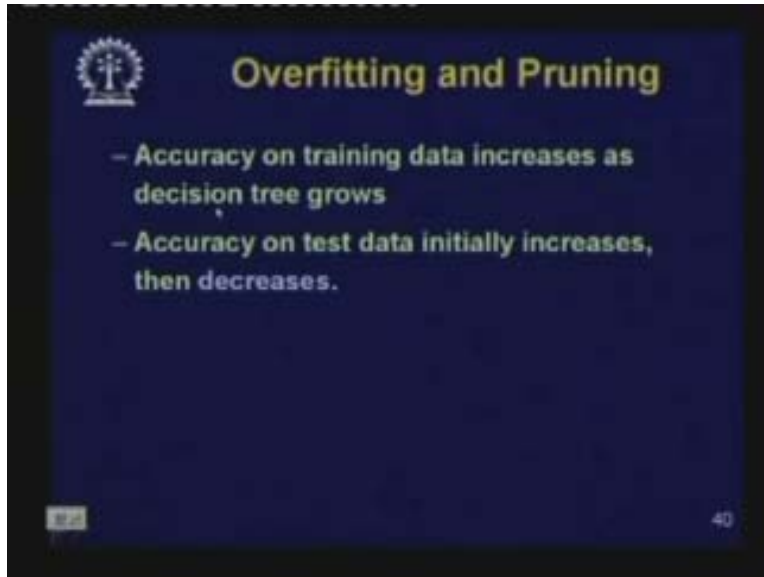


Typically this is the sort of behavior you get. If you are using the decision tree and as the decision tree size increases typically the accuracy on the training set increases. Initially you have a small accuracy but as you grow the decision tree the accuracy on the training set increases. However, the accuracy on the test set initially increases after which it may go down. So this is a typical behavior that takes place and this is due to over-fitting. Later we will see how decision trees can be pruned to avoid over-fitting.

But based on this curve this is the optimal tree size for which the test set accuracy is highest. So, one can think of stop growing the decision tree at this point and you can find it out you can keep aside test set and when you have the decision tree at various stages you check its accuracy on the test set and you select the tree for which the test set

accuracy is highest so this is one method and the other method is to grow the entire decision tree and then prune the decision tree.

(Refer Slide Time: 53:48)



So accuracy on the training data increases as the decision tree grows and accuracy on the test data initially increases but then decreases.

(Refer Slide Time: 52:40)

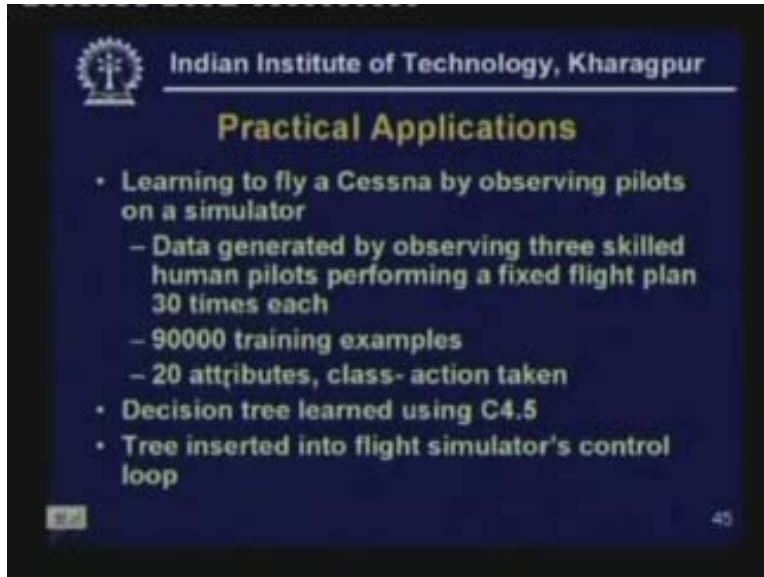


Some practical applications of decision trees:

Decision trees have been used for classifying patients by disease, they have been used for diagnosis for breast cancer. For a particular data set it was found that humans are correct,

human doctors are correct 65% of the time and a decision tree classifier was found to have an accuracy of 75% higher than that of human experts. Decision trees have been used for classifying equipment malfunctions, for classifying loan applications by level of risk. BP designed a decision tree for separating gas and oil for offshore oil platforms.

(Refer Slide Time: 53:40)



Indian Institute of Technology, Kharagpur

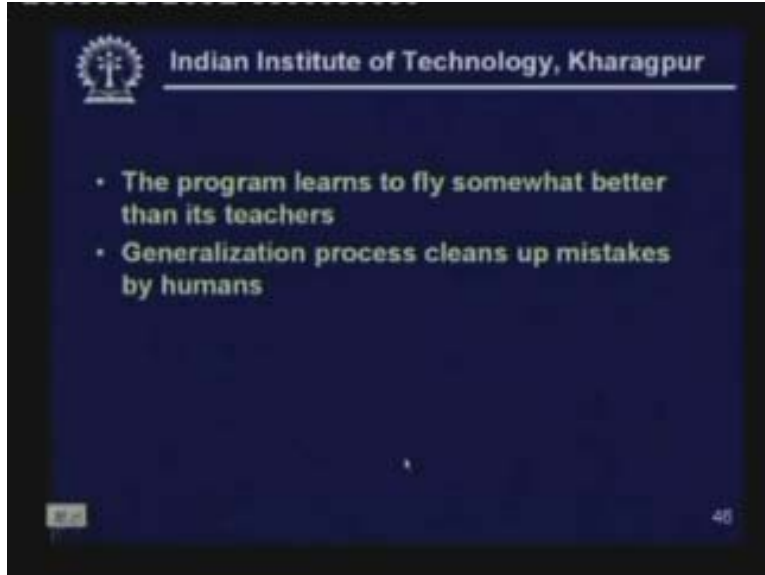
### Practical Applications

- Learning to fly a Cessna by observing pilots on a simulator
  - Data generated by observing three skilled human pilots performing a fixed flight plan 30 times each
  - 90000 training examples
  - 20 attributes, class- action taken
- Decision tree learned using C4.5
- Tree inserted into flight simulator's control loop

43

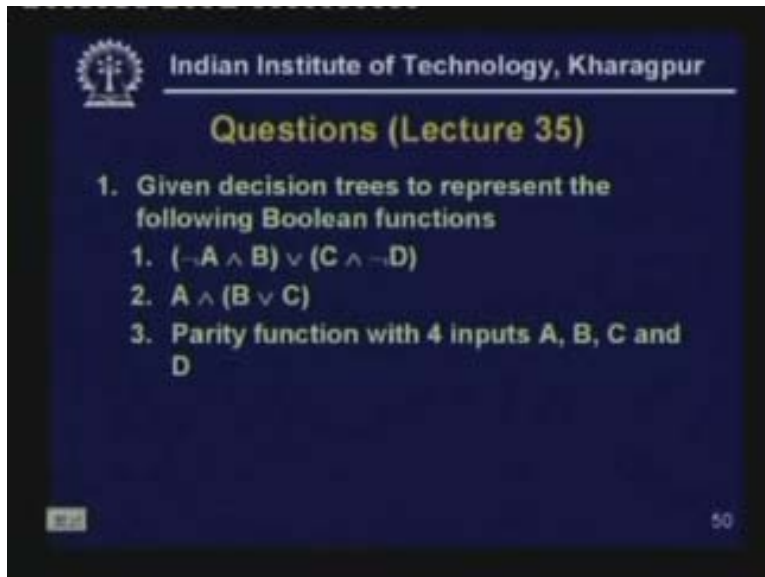
Another application was learning to fly a Cessna by observing pilots on a simulator. Data was generated by observing three skilled human pilots performing a fixed flight plan 30 times each and 90,000 training examples were obtained, 20 attributes, the class was the action taken like thrust, flap etc. The algorithm used was the learning algorithm c 4.5 and this tree was inserted into the flight simulators control loop and tested the program learn to fly somewhat better than its teachers.

(Refer Slide Time: 55:21)



The generalization process cleans up mistakes by humans. This was a quite successful exercise.

(Refer Slide Time: 55:40)



Some questions:

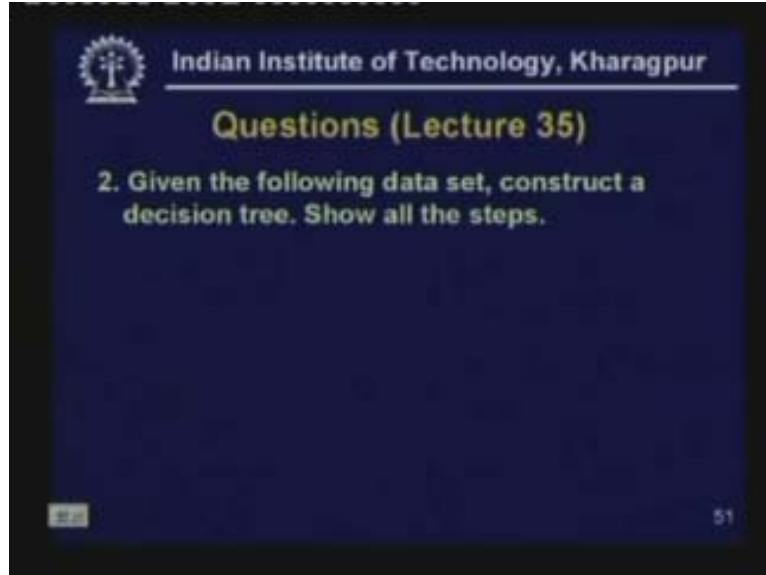
1) Give decision trees to represent the following Boolean functions:

a) NOT A and B or C and NOT D

b) A AND B OR C

3) Parity functions with four inputs A, B, C and D.

(Refer Slide Time: 56:16)



2) Given the following data set construct a decision tree show all the steps.

(Refer Slide Time: 56:23)

District	House Type	Income	Prev. Cust.	Outcome
Suburban	Detached	High	No	Nothing
Suburban	Detached	High	Yes	Nothing
Rural	Detached	High	No	Responded
Urban	Semi-detached	High	No	Responded
Urban	Semi-detached	Low	No	Responded
Urban	Semi-detached	Low	Yes	Nothing
Rural	Semi-detached	Low	Yes	Responded
Suburban	Terrace	High	No	Nothing
Suburban	Semi-detached	Low	No	Responded
Urban	Terrace	Low	No	Responded
Suburban	Terrace	Low	Yes	Responded
Rural	Terrace	High	Yes	Responded
Rural	Detached	Low	No	Responded
Urban	Terrace	High	Yes	Nothing

2

In this data set there are four attributes; district, house type, income, previous customer and the result is the outcome and this is what you are trying to learn. The outcome can be nothing or responded, district is suburban or rural or urban, house type is detached or semi detached or terrace, income is high or low, previous customer yes or no.

There are fourteen examples:

Example:

- 1) District is suburban, house type detached, income high, previous customer no, outcome nothing;
- 2) District suburban, detached, high, yes, outcome nothing;
- 3) Rural, detached, high, no, responded;
- 4) Urban, semi detached, high, no, responded;
- 5) Urban, semi detached, low, no, responded;
- 6) Urban, semi detached, low, yes, nothing;
- 7) Rural, semi detached, low, yes, responded;
- 8) Suburban, terrace, high, no, nothing;
- 9) Suburban, semi detached, low, no, responded;
- 10) Urban, terrace, low, no, responded;
- 11) Suburban, terrace, low, yes, responded;
- 12) Rural, terrace, high, yes, responded;
- 13) Rural, detached, low, no, responded;
- 14) Urban, terrace, high, yes, nothing.