Artificial Intelligence

Prof. P. Dasgupta

Department of Computer Science & Engineering

Indian Institute of Technology, Kharagpur

Lecture No - 26
Learning: Decision Trees

Today, we will start the chapter of learning. So far, what we have seen in this course is, we started off by learning some methods of problem solving- automated problem solving- namely, through search.

(Refer Slide Time: 01:03)

Learning

Course: CS40022
Instructor: Dr. Pallab Dasgupta

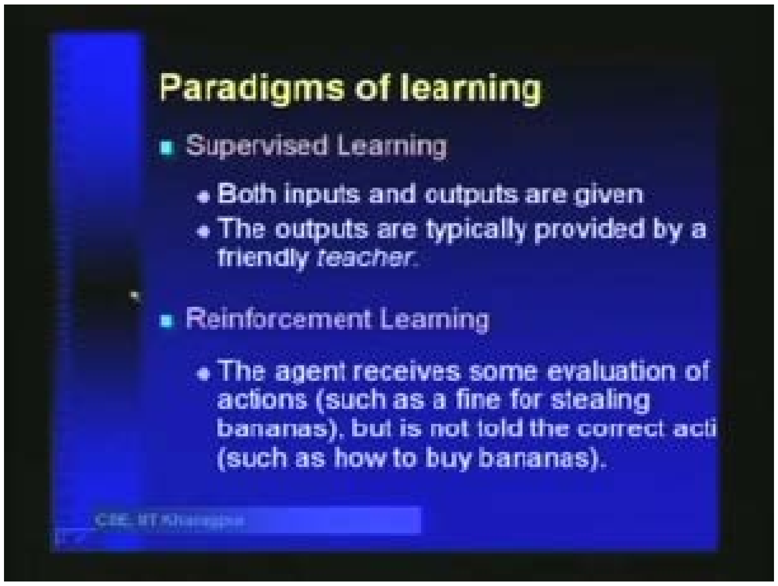
Department of Computer Science & Engineering
Indian Institute of Technology Kharagpur

And then, we saw the second most important thing in AI, that is, deduction. Then, we studied some applications of deductions, in the sense that we studied planning problems and also extended the deduction capabilities to cases where we do not know the exact rules, but we use probabilistic inferencing. Now, what remains in the computational part of AI? As I say, that the beginning of the course, that this course is mainly focused towards the computational aspects of AI or the main computational pillars around which AI has been built. 1 of the main things that we have left out so far is learning and the capability of a system to learn by itself from a given set of examples.

I will start by introducing some paradigms of learning. The main paradigms of learning and then, today, we will look at learning of decision, which is 1 of the ways of learning a function. The paradigms of learning are mainly of 3 types; the first type is called supervised learning. In supervised learning, both the inputs and outputs are given. What are we trying to learn? We are trying to learn some function; that function can be Boolean, it can be real valued, whatever. Our attempt is to learn some function. What is a function? It is a mapping from inputs to outputs, right?

In supervised learning, the training set that is given to us consist of valid input-output pair. Therefore, we have this output for this input; we have this output and our objective is to learn the function in a succinct way. 1 thing that we could do is, we could just store them in a table, and then, if we are asked to report the values, then, we just look up the table and return the values. But then, the problem will be that if we are given some value, some input, which does not belong to the table, then, we will not be able to answer.

(Refer Slide Time: 04:03)



Paradigms of learning

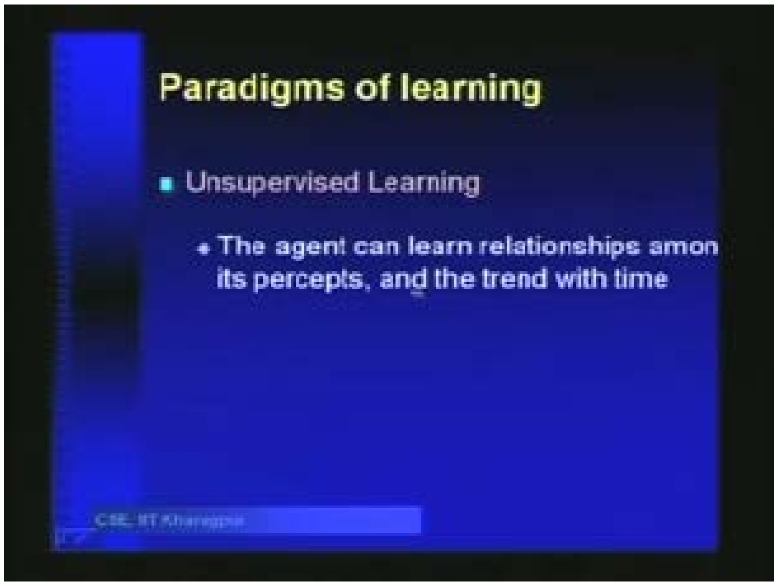
- Supervised Learning
  - Both inputs and outputs are given
  - The outputs are typically provided by a friendly *teacher*.
- Reinforcement Learning
  - The agent receives some evaluation of actions (such as a fine for stealing bananas), but is not told the correct acti (such as how to buy bananas).

CSE, IIT Kharagpur

Learning the function means that we should also not only be able to answer queries from the training set, but also be able to answer queries outside the training set. And these outputs are typically provided by a friendly teacher; that is why we call it supervised learning. We will see some methods of supervised learning in this chapter. The second paradigm of learning is called reinforcement learning. In reinforcement learning, the agent receives some evaluation of its actions, such as a fine for stealing bananas, but it is not told the correct action, such as how to buy banana.

We are not teaching the agent the correct actions, but there is some kind of cost criterion which the agent attempts to optimize and depending on the actions and the kind of cost feedback that it gets, it starts learning the function. For example, a person who does not know any traffic rules- it does not know how to drive- now starts learning driving on his own. And then, he gets a ticket for parking in the wrong area; he gets a ticket for speeding and that is the way in which he learns that yes, these are the rules of the game, so, that is called reinforcement learning. And the third paradigm of learning is purely unsupervised learning, where the agent can learn relationships among its percepts and the trend with time.

(Refer Slide Time: 05:17)



Paradigms of learning

- Unsupervised Learning
  - The agent can learn relationships amon its percepts, and the trend with time

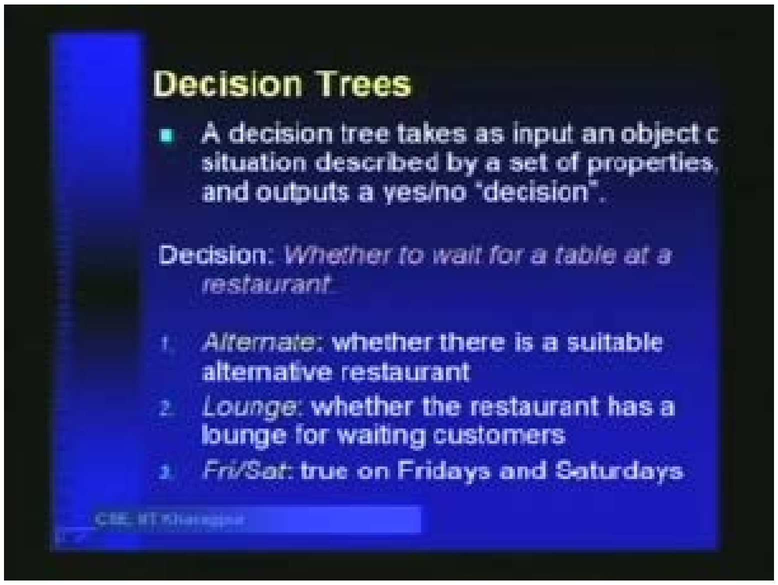
CSE, IIT Kharagpur

So, if you do not have any supervision nor any feedback, then, what do we learn? We just learn some relationships among the inputs or the percepts. We see that whenever this happens, then, that also happens. That is the kind of learning which is unsupervised. So, we will mainly study in this chapter, supervised learning and reinforcement learning. (Student speaking). In reinforcement learning, you get a feedback corresponding to your actions, so, there is a cost associated with it and if you do something wrong, you get a larger penalty; if you do something right, you might get a reward.

So, it is a sort of reward based learning, but you are never told what is right, what is wrong. The feedback is in terms of the cost, so, that is reinforcement learning. An unsupervised learning does not even have that. It is just from observations; whatever relationships that you learn about the universe. Things like data mining can come under unsupervised learning. You have lot of data and you try to just discover patterns in them. You are not given any information about what is the domain of the data. You do not have any information about what are the rules that play in that domain.

So, you do not know anything of that sort, but you just try to attempt to learn some relationships between them. That is unsupervised learning. So, in this, we are not going to focus much on unsupervised learning. The first kind of learning that we are going to study is the learning of decision trees. Let me first introduce what is the meaning of a decision tree. A decision tree takes as input an object or situation described by a set of properties, and outputs a yes-no decision. Let us take an example. Suppose we want to decide whether to wait for a table at a restaurant, and if you go to Dreamland or Sahara on Friday or Friday evening, or, you know, Saturday, then, you will see what we mean by whether we want to wait or not. There can be real rush.

(Refer Slide Time: 08:48)



Decision Trees

- A decision tree takes as input an object c situation described by a set of properties, and outputs a yes/no "decision".

Decision: *Whether to wait for a table at a restaurant.*

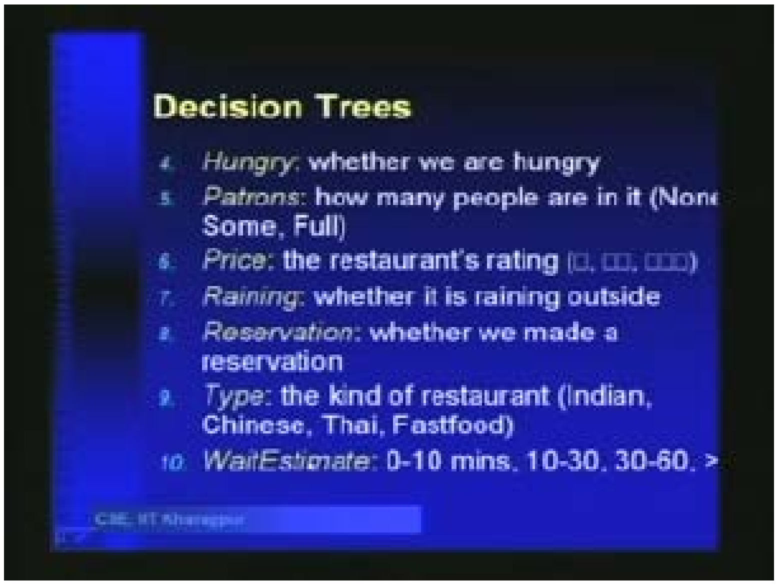
1. *Alternate*: whether there is a suitable alternative restaurant
2. *Lounge*: whether the restaurant has a lounge for waiting customers
3. *Fri/Sat*: true on Fridays and Saturdays

CSE, IIT Kharagpur

Let us see what are the features or the properties that can affect the decision; that is very important. We must first decide that what are the variables which will affect our decision. Variable number 1 is alternate, which signifies whether there is a suitable alternative restaurant. In our case, well, there are quite a few alternatives, but you might find all of them equally packed. Whether a restaurant has a lounge for waiting customers does not apply to the campus, but in Calcutta, you might have, whether it is a Friday or a Saturday. Those are the rush days, right?

Then, whether you are hungry. That can influence a decision. How many people are in it? None, some and full. Well, if you are in a city and in a busy area in office time and you find that the restaurant is empty, then, you may actually decide not to go in there. On the other hand, if you find- there are some people that might be good, if it is full, then, you might feel that it is going to take a lot of time.

(Refer Slide Time: 10:35)


Decision Trees

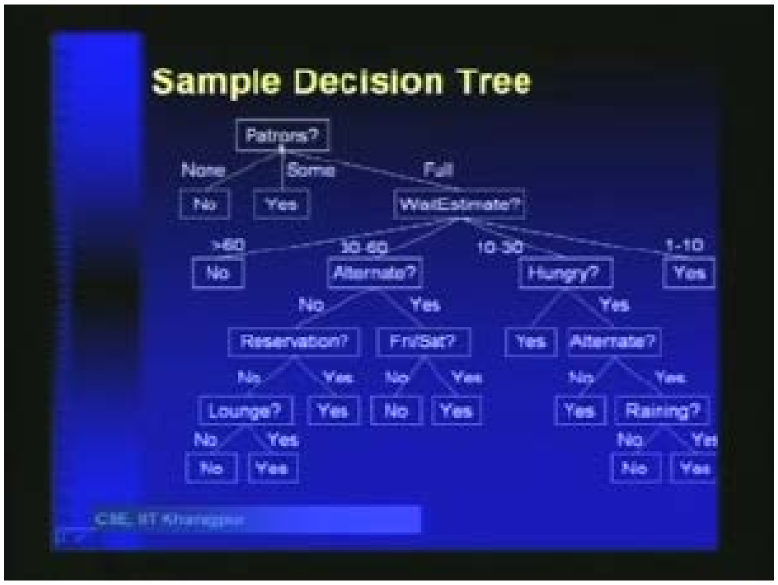
4. *Hungry*: whether we are hungry
5. *Patrons*: how many people are in it (None, Some, Full)
6. *Price*: the restaurant's rating (□, □□, □□□)
7. *Raining*: whether it is raining outside
8. *Reservation*: whether we made a reservation
9. *Type*: the kind of restaurant (Indian, Chinese, Thai, Fastfood)
10. *WaitEstimate*: 0-10 mins, 10-30, 30-60, >

CSE, IIT Kharagpur

Price, the restaurants' rating, right. Now, I think there is something wrong with the version of power point that I am using here, and the 1 where I developed these slides. Actually, this is supposed to come as star, but it as come as boxes. This is 1 star, 2 star and 3 star. That is the rating- whether it is raining outside. If it is raining, then, you may not want to go and look for another restaurant, whether you have a reservation and the kind of restaurant- Indian, Chinese, Thai, fast food, etc., right? And finally, the wait estimate as given by the waiter, that whether it is 0 to 10 minutes, 10 to 30, 30 to 60, greater than 60. So, you see, this variable out here is a is a real valued variable, but for the purpose of taking a decision, we have broken it up into intervals of 0 to 10, 10 to 30, 30 to 60 and greater than 60.

You could alternatively use some fuzzy measure we have. A decision tree will look something like this- at every level, you have 1 of the variables and this branching is for the different kinds of decisions that you may have. For example, for patrons, you may have 3 possible choices- that is, none, some and full. In this case, let us say that if there are no patrons, then, your decision immediately becomes no. If there are some patrons, which means you have space in the restaurant, then, it is always yes; otherwise, if it is full, then, you ask for the wait estimate. If the wait estimate is greater than 60 minutes, more than an hour, then, the decision becomes no. If it is 30 to 60, then, you look for an alternate.

(Refer Slide Time: 11:31)



If it is between 10 to 30, then, depends on whether you are hungry. If it is 1 to 10 minutes, then, you are always willing to wait. Again, if your wait estimate is between 30 to 60, look for an alternate, and if there is an alternate, then, you look for whether it is Friday or Saturday, because if it is Friday or Saturday, then, you may have the same situation in the other restaurants as well. So, the decision will be to stay here. Otherwise, if it is not Friday or Saturday, then, you are more likely to get a place in some other restaurant, so you just walk out. And on the other hand, if the alternate is no, you do not have an alternate, then, you check whether you have a reservation.

If yes, then, you stay. If no, then, you look whether it has a good lounge. If it does not have a lounge, then, you go away. If it has a lounge, then, you stay. This is how a decision tree will look like. Now, we could have made this decision tree in many other ways. For example, we could have first looked for wait estimate and under that, we could have looked for patrons and hungry, etc. So, the order in which we check for these variables can vary. The point is that how do we arrive at proper decision tree? And first of all, what do we mean by a proper decision tree?

Let us see what is it that we have to do. We would like ideally, that this decision tree to be as small as possible. It should not be the case that for every person, we have to examine all variables. Like for example, here, if there are no patrons, then, you immediately arrive at the decision. You do not have to examine any other variables, right? Now, the larger your decision tree, the more problem you will have in representing it. Why? Because if I have n variables and I have to examine every variable on every path, then, I will have a tree of size exponential. So, if every decision is yes-no, every variable is a Boolean variable, then, I will have 2 to the power of n leaf nodes, and the tree will have a size of the order of 2 to the power of n.

If the branching is more than that. For example, if you have a real valued variable or a variable like patrons, which has actually 3 possible values, or you have something like wait estimate, which has 4 different choices, then, the branching is even larger and your tree is even larger. Note that if you are allowed to create a tree which is of arbitrarily large size, then, we can fit in any pattern into it. But that is not the objective of the exercise. The objective of the exercise is to create a decision tree which is small and elegant and helps us in arriving at the decision. How do we learn such a decision tree? Let us first see what is the input. The input will be a set of cases, a set of samples.

For example, we can have an input. Let us say I will write the example here. Let us say example X1 and then, I have all the variables here, so, it is like a truth table. I have this- sorry, this is actually lounge, then, Friday, then, whether we are hungry, then, whether there are patrons, and similarly, the other attributes and then, here, we have the yes-no decision. Suppose I have yes here, no here. Then, here, I check what is the decision. For example, for the wait estimate, you can have some value here that says its 0 to 10. This is 1 example that somebody has observed.

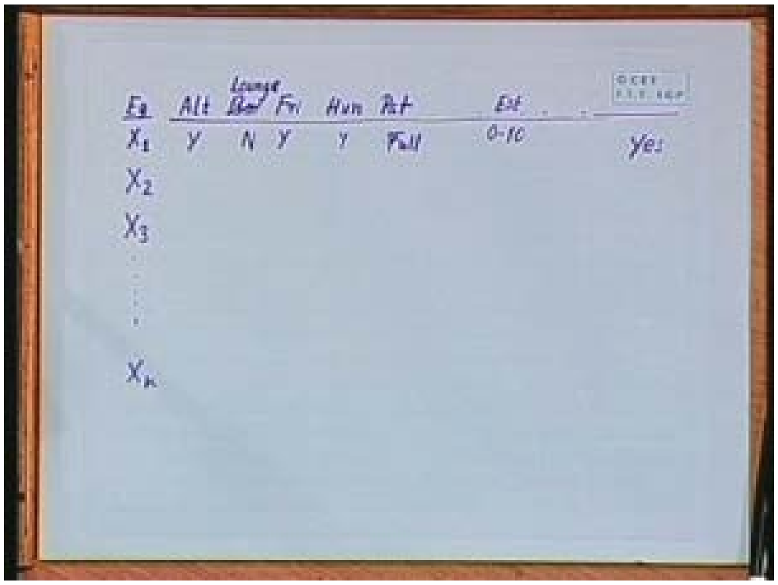
Previously, somebody had seen that well, in a particular case, where an alternative was available, there was no lounge, it was a Friday, the person was hungry, there were patrons. It will not be yes or no; it will be something like, let us say, full. If it was full, the wait estimate was 10, then, that person decided to wait. So, this is 1 case which we know from previous experience. Then, similarly, I have case 2, I have case 3 and I have some k cases like this. These are k examples that I have; k samples that I have. For each of these samples, I know what was the value of these variables, and I also know what was the decision that was taken.

This is our training set; this is our training set and obviously, if this training set was of the order of 2 to the power of n, then, our function is almost totally specified. If this is actually a truth table, in the sense that for all possible combinations of values of these variables, we are given the decision here, then, this is indeed a truth table and then, the function is completely specified. If the function is completely specified, then, there is nothing to learn actually and you can actually represent that Boolean, that function Boolean or whatever in a succinct form and we know that there are logic optimization techniques which can help us in doing that.

But the learning part comes here, because this training set is incomplete- it does not give you the decision for all possible combinations of values of the variables. So, then, what are we attempting to do? We are attempting to take this set and create a decision tree. A decision tree which will be small and which will be succinct. And why do we expect the decision to be tree to be small? Because it is not the case that every decision is affected by all of these variables; some of these variables can be overriding, if you know that, say that, the restaurant is empty, then, you do not care about any of the other variables and you will just simply go away.

What it means is that many of the variables can become do not cares in many of the decision situations and we want to appropriately utilize this do not care situations to reduce the size of the decision tree. So, it is also a kind of logic optimization kind of problem, except that we are not given the complete data. (Student speaking). No, what we need to do, therefore, is to be able to create a decision tree which can classify all of these data correctly.

(Refer Slide Time: 21:31)



Eg	Alt	Lounge Bar	Fri	Hun	Pat	Est	. . .	
X_1	Y	N	Y	Y	Full	0-10		yes
X_2								
X_3								
.								
.								
.								
X_k								

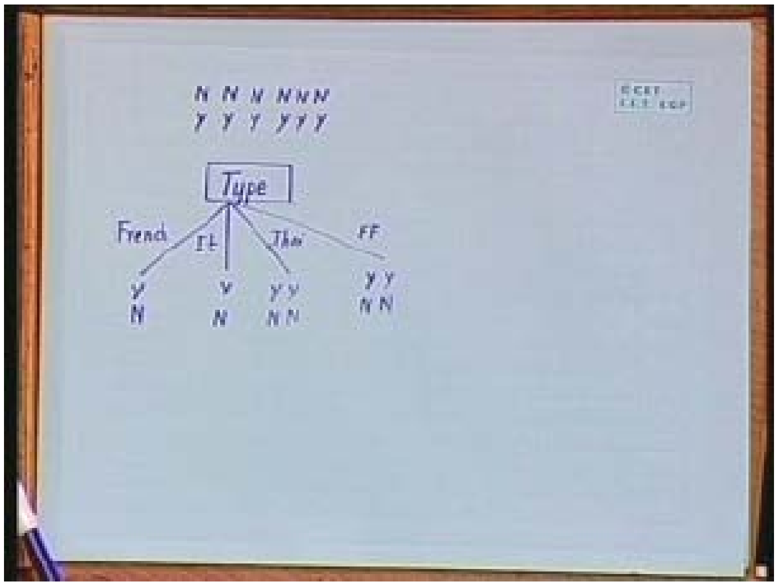
See, the problem which samples is that these samples need not always be consistent; there are always eccentric people who, when finding everything to their choice, will still walk away. And there are also eccentric people, who, finding all the odds against them, will still wait in the restaurant. Therefore, the samples can be noisy, the samples set can be incomplete and it is under this kind of scenario that we have to find a decision tree which best fits the data and is also small within that domain. This is the objective that we want to achieve. You can never be sure that what you have learnt is actually the correct 1, because the training data is incomplete, so, you can never be sure about that.

But what we should be able to do is, that decision tree should be able to classify the set of examples with some margin, because of the noise of the data. And it should also be small, because then, it indicates our stand that there will be lot of variables which will overwrite others, which is typically the scenario in most of the real world cases, so, that is the objective of learning decision trees. Now, let us see how we go about doing this. The idea is to be able to choose the features 1 by 1, so that they are able to appropriately classify the yes from the no.

When we say that I have a good classifier, then, if I know the value of that variable, I will be able to distinguish from some of the yes cases from the some of the no cases. If I am able to do that- if I am able to do that for all cases, then, that variable alone can do the classification. I do not need anything else, if there is some variable which, if you take this branch, you get all the yes cases; if you get this branch, you get all the no cases, then, that variable alone is your decision tree. You do not need anything else. But we may not be so lucky.

We try to choose a variable which will be able to at least separate out some of the yes cases from the no cases. For example, let us say- suppose we see that if we split it on type, on the restaurant type, and let us say that originally I had that 6 yes cases- 6 no cases and 6 yes cases. This is our training data set. In the training data set out here, I had 6 yeses and 6 no. 6 instances where the decision was yes and 6 instances where the decision was no. Now, I am trying to see that if I split the cases based on type, then, what is the scenario that I have?

(Refer Slide Time: 26:10)

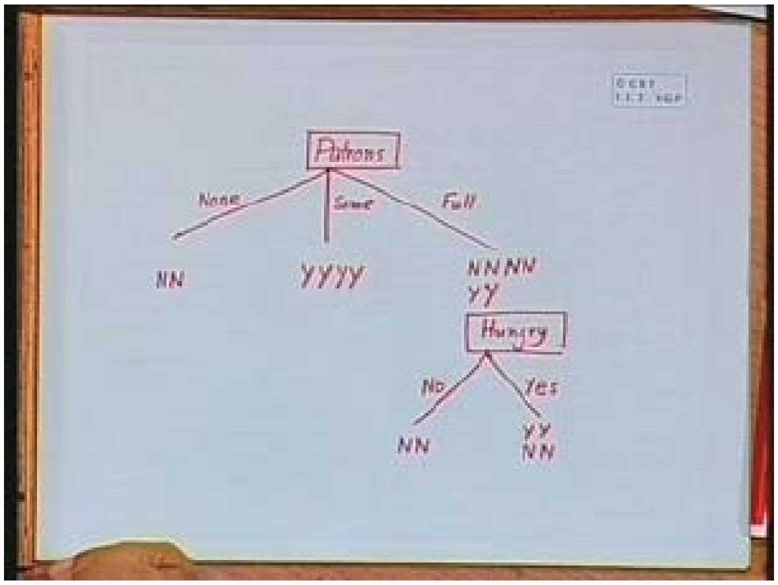


Let us say that if I go for French, then, I have 1 yes and 1 no. If I go for Italian, then, I have again 1 yes, 1 no. If I go for Thai, then, I have 2 yes and 2 no. If I go for fast food, I get 2 yes and 2 no. If this is the scenario, then, is this a good classifier or bad? This is bad because this does not help us in distinguishing between the yes cases and the no cases. In each of these cases, half of the people said yes, half of the people said no. So, actually, it leads us to believe that the type was not a factor in deciding between yes and no. Type was not an important factor in deciding between yes and no, because in each of these cases, half of the people said yes, half of the people said no.

So, for every positive case, we got a negative case. On the other hand, suppose we look at patrons; suppose we look at patrons and we find that for none; so, patrons can take 3 values- none, some and full. For none, I have 2 nos; for some, I have 4 yes, and for full, I have 2 nos. And no 4 nos and 2 yes. So, that accounts for these 6 nos and 6 yeses. Suppose it is this classification that patrons has given as- so, now, this is a good classifier. Why? Because we know that if there are no patrons, then, that immediately takes us to the decision no.

If there are some patrons, then, it immediately takes us to the decision yes. There is no conflict between the people who find this to be some, but if it is full, then, we still have some people saying yes and some people saying no, which means that this now becomes a sub-problem and we again have to classify this sub-problem and let us say, then, we check for at this state. Let us say that we check for hungry and then, let us say hungry- if it is no, then, I have 2 nos; if it is yes, then, I have 2 yes and 2 nos.

(Refer Slide Time: 29:47)



This is again a good classifier, because if it is no, then, you immediately arrive at a decision no. If the restaurant is full and you are not hungry, then, your decision is going to be no. On the other hand, if you are hungry, then, half of the people are willing to wait and half of the people are going to go away. Again, here, you use another classify, right? Now, in this case, we were somewhat lucky because each of these were able to segregate out some of the yes and no cases. We may not be so lucky always, so, it is a difficult task to choose between the variables.

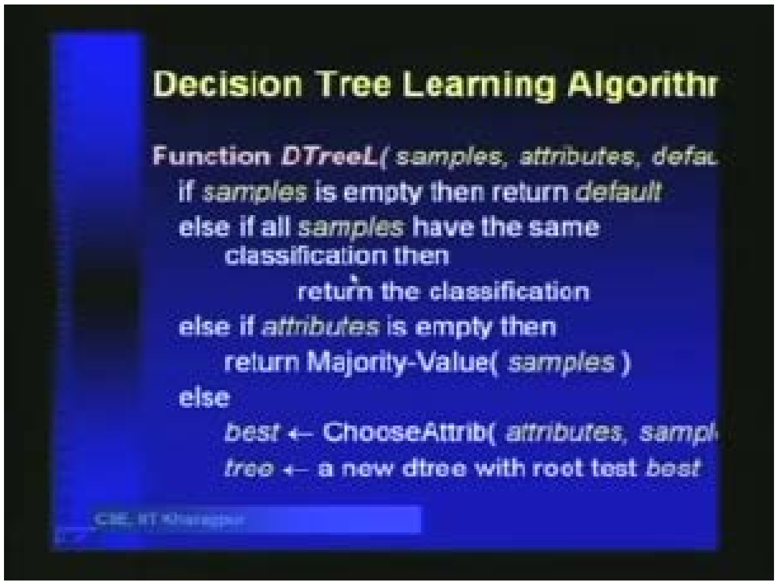
But once you choose a variable, then, how do we proceed after choosing the variables? Then, corresponding to each of the decisions, we will simply classify the examples and then, on the reduced sample set for each of them, we will recursively run the decision tree generation algorithm again. So, is it intuitively clear how we create the decision tree? Now, to arrive at the decision, to arrive at the choice of the variable, there is some information theoretic measures for choosing this.

I am not going to describe the computation of that in this lecture. But it is there in Russell and Norvig's book, and I recommend that you take a look at that. There is a information theoretic measure for the information content that you are able to increase by applying a decision on 1 of the variables. So, you do that for each of the variables individually; check which 1 of them adds the maximum amount of information according to that formula and choose that variable. So, intuitively, that formula is going to capture the variable which is able to act as the best classifier for the set of samples that you have. Let us now look at the decision tree learning algorithm. This is a pseudo-code for the decision tree learning algorithm.

If samples is empty, then, we return default. What is default? Default is the default decision that we are going to take. We will always have a default decision. Later on, we will see that in case where we do not have any information, then, the default decision will be the majority among the samples that we have. For example, here, when- text please- when we had this decision, suppose we did not have any other attributes to classify this. Then, here the majority was no. So, we would say that if there is no other classifiers except patrons- suppose we only have patrons as the variable other; we are not given any other variables, then, we use patrons to do this classification and for the case where it is full, our decision will be no, because of the majority.

Now, coming back to the code, if samples is empty, then, we return default. If you do not have any more samples to classify, else, if all samples have the sample classification like we had for the patrons' case, then, return that decision. So, if you have all yes, then, return yes; if you have all nos, then, return no. Otherwise, if attributes is empty, if there is no other attributes to split the set of samples, then, return majority value among the samples. So, if we have 5 nos and 3 yeses, then, return no, otherwise, this is the main part. We choose an attribute from this remaining set of attributes and the remaining set of samples.

(Refer Slide Time: 33:01)



Decision Tree Learning Algorithr

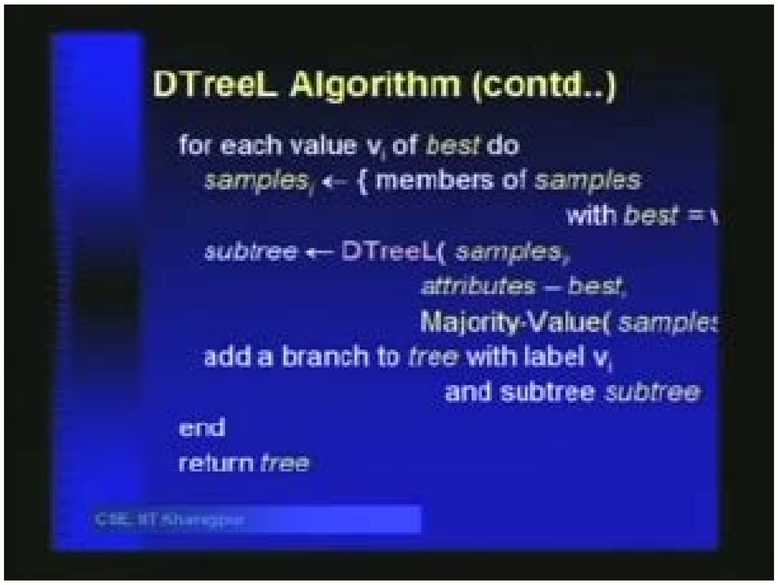
```
Function DTreeL( samples, attributes, defau
  if samples is empty then return default
  else if all samples have the same
      classification then
          return the classification
  else if attributes is empty then
      return Majority-Value( samples )
  else
      best ← ChooseAttrib( attributes, sampl
      tree ← a new dtree with root test best
```

CSE, IIT Kharagpur

This is the step which will use that formula to choose the proper attribute and then, we initialize tree to a new decision tree with root test as best. This is recursively going down from the top node to the successor nodes. So, at every level, we have a set of samples; we have a remaining set of attributes and our objective is to split these samples based on a chosen attribute. That chosen attribute is best, so, we create a sub-tree with the root test best. In the root of the sub-tree, we will examine the value of the variable best.

Then, for each value vi of best- now, see, best may not be a Boolean; it may be a multi-valued variable like we had patrons, like we had wait estimate; it can be a multi-valued variable. So, for each value vi of best, we do we split the samples based on the value of variable vi. Is it clear? So, suppose we look at the variable best and it can have 3 possible options. So, for each of those options, we split the sample sets into the options that fit into that.

(Refer Slide Time: 35:31)



DTreeL Algorithm (contd..)

```
for each value $v_i$ of best do
  samples_i ← { members of samples
                          with best = v
  subtree ← DTreeL( samples,
                 attributes – best,
                 Majority-Value( sample
  add a branch to tree with label $v_i$
                    and subtree subtree
end
return tree
```

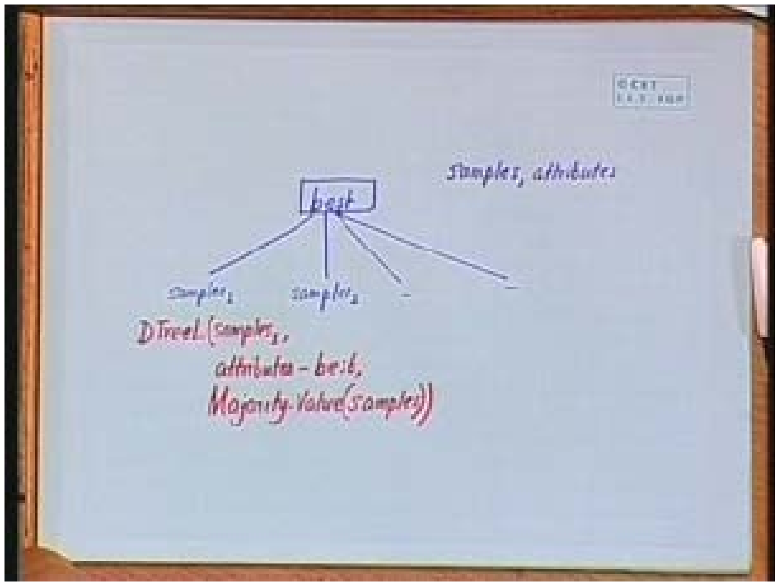
CSE, IIT Kharagpur

Suppose I have none, some and full as the values of patrons. Those examples which have none will be pushed into the none set, those which have some which will be pushed into the some and those which are full will be pushed into the full. These are the samples. Then, we create a sub-tree, so, these are the children of the best. Best is the variable around which we are splitting and the children will be simply recursively generated by the recursive call to decision tree learning algorithm with the reduced set of samples which is samples psi.

The reduced set of attributes, which means attributes minus best, because we have already taken a decision on best. And also, majority value of samples means that why do we need this? I will come back to this. What does this mean? It means that at the root level, I have, say, k samples, out of which, whichever is majority- yes may be majority or no may be majority- whatever is that; that is the value that we put here and the reason we put here: I am going to come back in a moment; then, add a branch to tree with label vi and sub-tree. Do you follow?

Are you with me about what we did in this algorithm? Okay, we started at every level with a set of samples which we are calling samples and a set of attributes. Then, from this attribute set, we choose 1 which is called best. And then, we took this sample set and split it up, depending on the values that this best can take. So, here, I will have samples 1, samples 2, samples 3 and samples 4. Then, I will recursively call the D tree decision, tree learning algorithm here and pass this samples 1 here- this attributes, this set, minus this attribute, namely, best and I am passing another thing- majority values of samples. Why?

(Refer Slide Time: 39:06)



(Student speaking). We may not have any more sub-tree below this and if you look at the previous slide- (Student speaking). No, it is not sample set, because you have the samples I set already passed as the parameter, so, you can compute the value from that also. The problem that you will have is, if you do not have any attributes left, then, what you need to do is that you will just take sample psi and take its majority and return. And the other scenario that can happen is that you run out of samples, then, we return this default. See, this is the third parameter.

So, if the sample set is empty, then, we return that value. Now, let us understand what we mean by this. Suppose we are splitting on this value variable best, and for some value of best, we do not have any samples for that scenario. Suppose you have that weight estimate. It is 0 to 10, ten to 30, 30 to 60 and greater than 60. Let us see that we never had a sample for greater than 60. None of the samples have the wait estimate greater than 60. In that case, what will be report as the decision when the time is greater than 60? Based on the the samples that are given to us, we cannot.

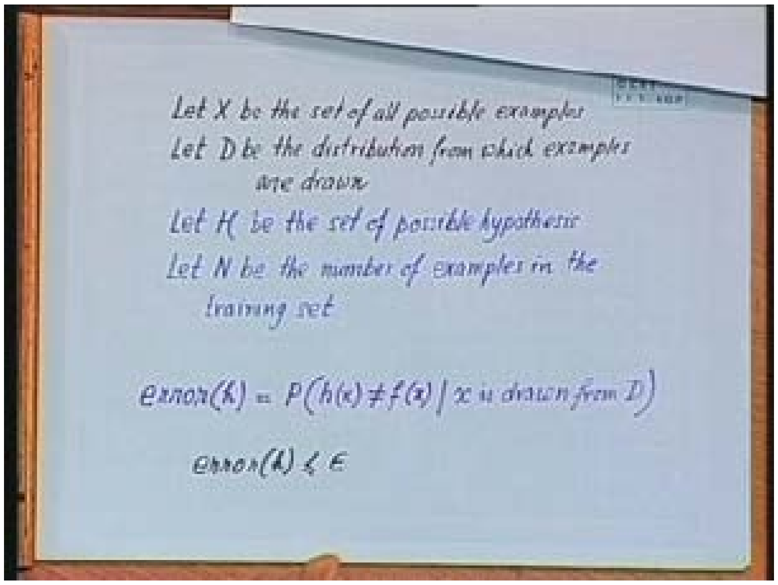
For those cases, what we do is, we look at the parent sample set and whatever is the majority value there, that is the value that we used here. It is just that we decide to use that; it may not be correct, but then, we do not have any sample which covers that case, so, we will never know for sure. Is it clear? This is the decision tree learning algorithm. Then, after this, I will just quickly like to add on a little bit of analysis on very important question. The question that we want to answer here is, for the decision tree to be properly learned, how many examples do we need?

There has to be some minimum number of examples which will enable us to learn the proper decision tree. Now, suppose there is a proper decision tree for a given domain, which we do not know. That is the right decision tree is there, we may not know that, but it is there, and that decision tree governs the set of decisions for that domain. Now, we are just given a set of samples, so, what we are given is like this: let us say that this is the actual decision tree function that we want to learn. Now, you appreciate that this is a function; this is a function which is Boolean, because it has a yes no answer. So, let us look at an error epsilon around the function.

That will encompass the set of functions which we have here. What are these functions in this region? They are all functions which are within an error of epsilon of the original function. Now, I have not yet defined what is that error. Let me first try- let me define the function first. Let us first give some definitions. Let X be the set of all possible examples, let D be the distribution from which examples are drawn. I will come back to this later. Let H be the set of possible hypothesis. So, this is the set of possible hypothesis, means that this is the set of all possible decision trees that you can construct out of the set of examples.

These are all the possible set of hypothesis that you can fit the set of the samples into it. And let N be the number of examples in the training set, namely X. Now, we define error H as probability that HX not equal to FX, given that X is drawn from the distribution D. Are you with me about what we are doing? This HX denotes the set of- this is a particular hypothesis that we have been able to create out of the set of samples.

(Refer Slide Time: 48:54)



Let X be the set of all possible examples
Let D be the distribution from which examples are drawn
Let H be the set of possible hypothesis
Let N be the number of examples in the training set

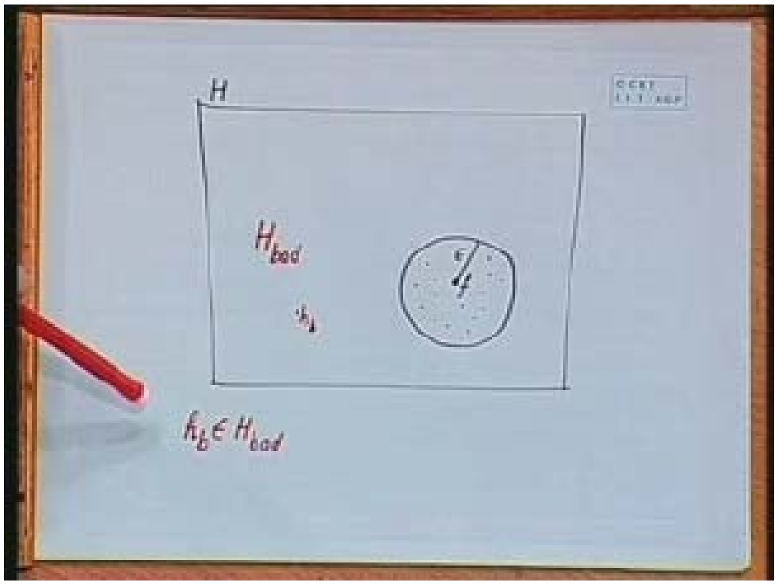
$error(h) = P(h(x) \neq f(x) \mid x \text{ is drawn from } D)$

$error(h) \leq \epsilon$

Now, the probability that this is not the actual decision tree, is the error H that we are talking about. And then, we will say that this particular decision- the set of examples, is sufficient to keep this error bounded by epsilon. If we have error H less than equal to epsilon, where epsilon is some small constant within which we want to keep this; now, if you increase the size of the examples and give all possible- say, if you produce the entire truth table- then, we can learn h, learn the function exactly, so, our error will be 0. Now, if you make n very small, then, we can have many different hypothesis which will fit the same data and so, our error will be more. Is it clear?

The probability that the hypothesis that we choose is not the actual 1 will be more. So, now, if you look at this diagram diagrammatically, then, suppose this H is the enter set of hypothesis and then, we create a circle of radius epsilon around F. So, all the hypotheses that we have in this region- they are all acceptable. But once that we have here, we will call them as HBad; these are the hypotheses which we do not want to arrive at. And our objective is to decide that what should be the size of the training set, so that we are always learning hypothesis which are in this set and not outside this. Now, what we do is, we first calculate the probability that a seriously wrong hypothesis which we call HB, which belongs to HBad, are the bad cases. So, we first compute the probability that some HB here is consistent with the N examples that I have given to us.

(Refer Slide Time: 51:10)

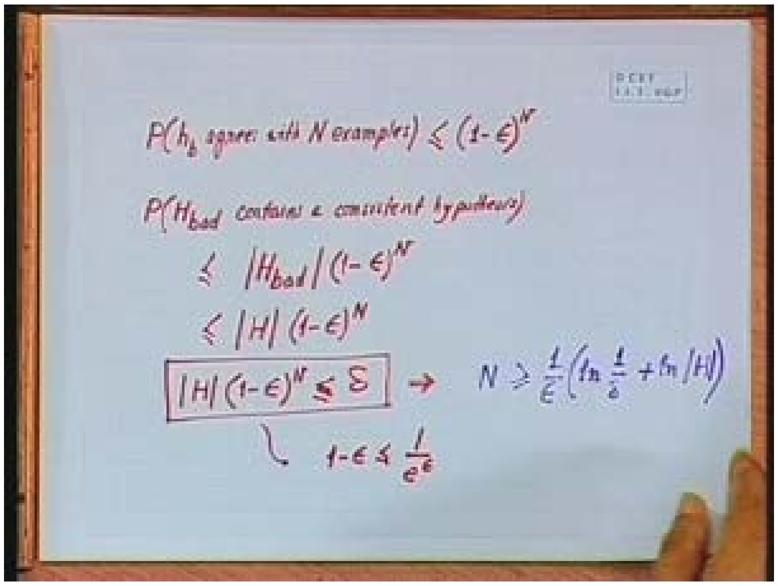


If this is consistent with the n examples given to us, then, we can always end up learning HB, which is going to be bad for us. So- (Student speaking). No, see, we are given only n examples, but there can be many other examples which we are not given. In those examples, in the other ones, this hypothesis will produce incorrect value, so, that is the error that we are talking about. It will produce vastly incorrect values. We do not want that to happen. Let us see how we go about this analysis. Probability that HB agrees with N examples is less than or equal to 1 minus epsilon to the power of N.

Why? Probability that it agrees with 1 example is 1 minus epsilon and the probability and it agrees with all N examples is 1 minus epsilon to the power of N. Do we have it? Then, probability that HBad contains a consistent hypothesis is the sum of these probabilities. So, it is less than or equal to HBad. Cardinality of the HBad set times 1 minus epsilon to the power of N, because if 1 hypothesis has this probability, then, if we take the total set, then, it is the cardinality of that set times this.

This, we can write, is less than or equal to cardinality of H times 1 minus epsilon to the power of N and this is less than or equal to this, because recall that HBad is actually a subset of each, right. We can write this and what we want here is, we want this probability that HBad contains the consistent hypothesis to be within some limit delta. Now, given this, we can compute that if we are given this, then, from this, we can derive the expression for N which is greater than or equal to 1 by epsilon ln, natural log of 1, by delta plus.

(Refer Slide Time: 56:03)



$P(h_b \text{ agrees with } N \text{ examples}) \leq (1-\epsilon)^N$

$P(H_{bad} \text{ contains a consistent hypothesis})$

$\leq |H_{bad}| (1-\epsilon)^N$

$\leq |H| (1-\epsilon)^N$

$\boxed{|H|(1-\epsilon)^N \leq \delta} \rightarrow N \geq \frac{1}{\epsilon}\left(\ln \frac{1}{\delta} + \ln |H|\right)$

$1-\epsilon \leq \frac{1}{e^{\epsilon}}$

Now, we can get this by applying that this; from this 1 minus epsilon, we can have 1 minus epsilon is less than or equal to 1 by e to the power of epsilon that is being substituted here, so, I will have 1 by e to the power of N epsilon, and then, I take the log to get N down and that gives us this expression. What we have been able to do is that by applying this formula, we will get the value of N, which is going to give us the minimum number of examples that are needed, so that we can keep the probability that HBad contains a consistent hypothesis within a limit of delta. So, this is used to compute the number of sample sets that are required for a given domain when we know the function.