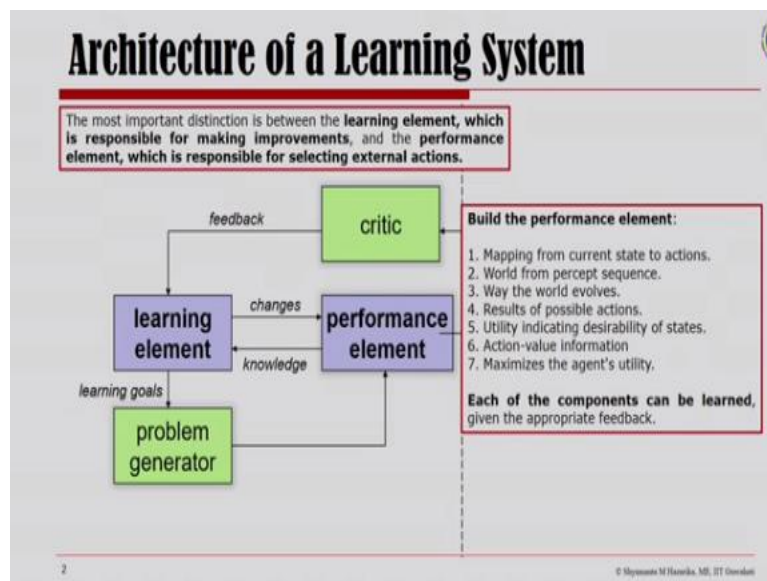


Fundamentals of Artificial Intelligence
Shyamanta M. Hazarika
Department of Mechanical Engineering
Indian Institute of Technology - Guwahati

Lecture – 28
Learning Decision Trees

Welcome to fundamentals of artificial intelligence, we continue our discussion on machine learning; machine learning is the ability of computational systems to improve its performance automatically based on experience. As seen in the last lecture, machine learning can be broadly categorized into 3 different types of learning; supervised learning where we have labelled input data to learn the model, unsupervised learning is about learning the structure of the data and then we have reinforcement learning, which is learning based on rewards. In this lecture, we would be looking at learning from observation particularly, learning decision trees. Before we move on to understanding what a decision trees and how one learns decision trees from observations, let us try to have a clear idea on how the elements of a learning system relate to what we are going to discuss today.

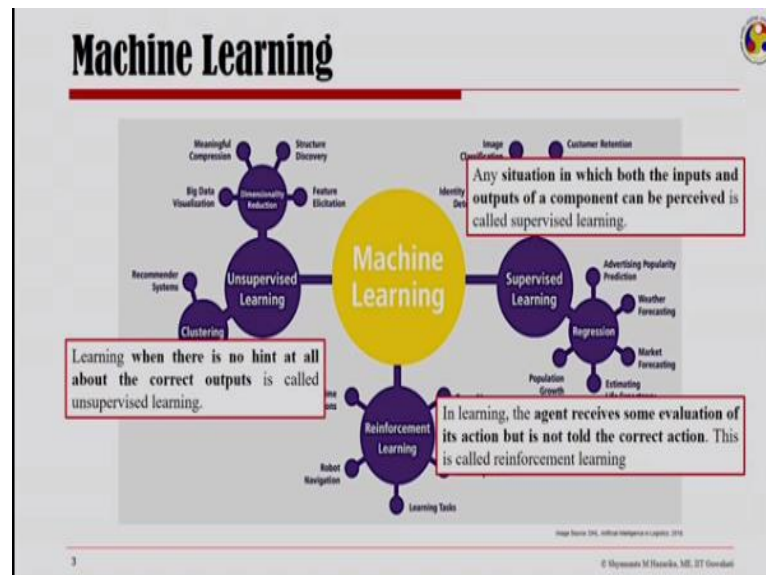
(Refer Slide Time: 02:12)



So, kindly recall the architecture of a learning system that we had discussed in the previous lecture, here the most important distinction is between the learning element which is responsible for making improvements and the performance element which is responsible for selecting external actions, one can build the performance element by mapping from current state to actions or creating the world from percept sequences. Look at the way the world evolves which is about result of possible actions, get the utility indicating desirability of states, look

at action value information and maximize the agent's utility. Now each of these components can be learned given the appropriate feedback and each component of the performance element can be described mathematically as a function, all learning can be seen as learning the representations of such a function.

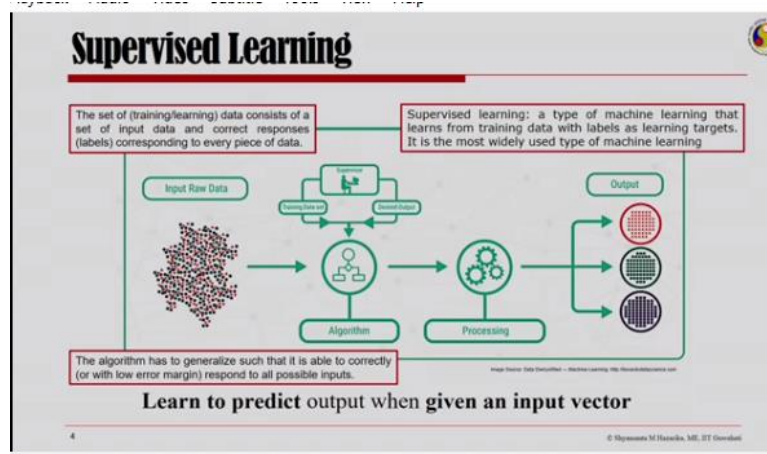
(Refer Slide Time: 03:28)



And recall what we had looked at in terms of the categories of machine learning. We had supervised learning now, this is any situation in which both the inputs and outputs of a component can be perceived and in unsupervised learning, there is no hint at all about the correct outputs whereas, reinforcement learning is about when the agent receives some evaluation of its action but is not told the correct action.

Now, if we look at learning from observation: learning from observation is under the category of supervised learning and in supervised learning, we learn to predict output when given an input vector.

(Refer Slide Time: 04:25)



Supervised learning is when the training data has labels as learning targets, supervised learning is the most widely used type of machine learning, the set of training data consists of set of input data and the correct responses are what are called labels corresponding to every piece of data. The algorithm in supervised learning has to generalize such that it is able to correctly or with very low error margin respond to all possible inputs.

(Refer Slide Time: 05:04)

Machine Learning

A **computer program** is said to **learn** from **experience E** with respect to some class of **tasks T** and **performance measure P**, if its **performance** at tasks in T, as measured by P, **improves with experience E**.

– Tom Mitchell

A **computer system** learns from **data**, which represent some **"past experiences"** of an **application domain**. Our focus: learn a target function that can be used to predict the values of a discrete class attribute.

© Deyanira M. Hristova, M.S., IT Consultant

Recall that the very basic definition of machine learning as put forward by Tom Mitchell is about learning a particular task T, where I am interested to improve its performance measure P based on certain experience E. Now, a computer system when I mean experiences, it is data of an application domain, so a computer system learns from data which represents some past experiences of an application domain. So, our focus here is to learn a target function that can be used to predict the values of a discrete class attribute.

(Refer Slide Time: 05:58)

Inductive Learning

- In supervised learning, the learning element is given the correct (or approximately correct) value of the function for particular inputs, and changes its representation of the function to try to match the information provided by the feedback.
 - An example is a pair $(x, f(x))$, where x is the input and $f(x)$ is the output of the function applied to x .

Pure Inductive Inference

Given a collection of examples of f , return a function h that approximates f . The function h is called a hypothesis.

So, in supervised learning, the learning element is given the correct or approximately correct value of the function for a particular input and changes its representation of the function to try and match the information provided by the feedback. So, the example would be a pair x and f of x , where x is the input and f of x is the output of the function applied to x . Given a collection of examples of such f , I am interested in a function h that approximates f .

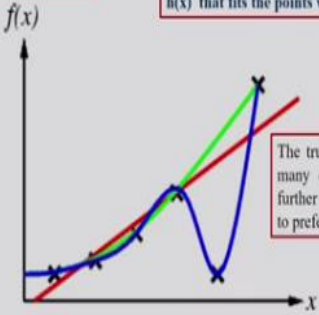
And the function h is called the hypothesis and this is pure inductive inference, so in pure inductive inference what I am interested in is getting to a hypothesis function h such that it approximates all such functions f , which somehow can explain the data in terms of the input x and the output of the function which is fx .

(Refer Slide Time: 07:28)

Inductive Learning

Three hypotheses for functions from which these examples could be drawn

From plane geometry: Examples are (x,y) points in the plane, where $y = f(x)$. The task is to find a function $h(x)$ that fits the points well.



The true f is unknown, so there are many choices for h , but without further knowledge, we have no way to prefer one over the other.

Any preference for one hypothesis over another, beyond mere consistency with the examples, is called a bias. There are always a large number of possible consistent hypotheses; learning algorithms exhibit bias.

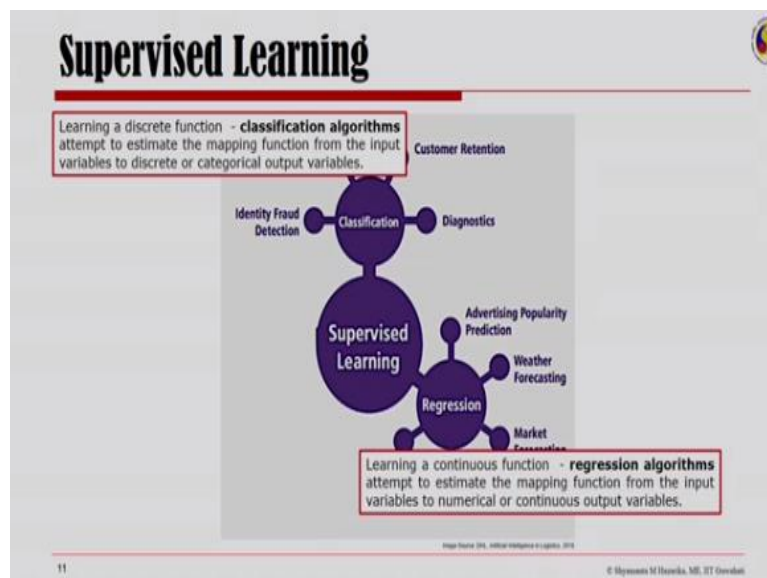
10 © Heymans M Hainke, MSc, ST Overholt

So, in terms of plain geometry if you look at it, so here are examples of xy points in the plane, where y is the output of the function f, so $y = f(x)$ and the task is to find a function h of x that fits the points well. The function h of x is called the hypothesis. Now, as you will realize, I could have a number of functions that could explain this group of points for example, I could go with these red lines shown on your screen which takes a couple of points and tries to match only between them.

I could have another function here which is trying to explain a more, bigger set of points, the first if you realize was a linear function and then I could have more complex functions describing all the points taken together. Now, the true function is unknown, so there are many choices for the hypothesis function h but without any further knowledge, we have no way to prefer one over the other.

Three hypotheses for functions from which these examples could be drawn is shown here and any preference for one hypothesis over another beyond mere consistency with the examples is called a bias and there are always a large number of possible consistent hypotheses, so we should realize that learning algorithms exhibit bias for we would prefer one over the other.

(Refer Slide Time: 09:30)



Now, in supervised learning we have 2 very interesting categories; one called classification which is about learning a discrete function. So classification algorithms in a way attempt to estimate the mapping function from the input variables to discrete or categorical output variables, on the other hand we have regression which is about learning a continuous function.

So, regression algorithms in a way attempt to estimate the mapping function from the input variables to numerical or continuous output variables. Today, we would look at a classification problem and we will look at regression in one of our future lectures.

(Refer Slide Time: 10:28)

Classification: Data and Goal

- **Data:** A **set of data records** (also called examples, instances or cases) described by
 - ✓ **attributes:** A_1, A_2, \dots, A_k .
 - ✓ **class:** Each example is labelled with a pre-defined class.
- **Goal:** To learn a **classification model** from the data that can be used to predict the classes of new (future, or test) cases/instances.

12 © Mageswarar V. Harshini, M.E., IIT Guwahati

So, classification here is a more graphical explanation of what classification is, we have label data and there is a classification algorithm which tries to draw boundaries between these group of label data and then when we have unlabelled data being sent, here if you note, I am sending a triangle and my model is able to recognize this as a triangle. So, in terms of classification, the data is a set of data records.

These are also called examples instances or cases and they are described by k attributes and each example is labelled with a predefined class, so each example comes with a class and in terms of goal, the goal is to learn a classification model from the data which then can be used to predict classes of new instances. In our case here, we had looked at label data which was the rectangles, triangles and the circles.

We had a classification algorithm which could draw clear boundaries between these group of data records and then the goal was to really look for the class of the new unlabelled data that was coming to the algorithm that is the triangle and we saw that the triangle could be very nicely grouped to the group of triangles, so this is what classification is.

(Refer Slide Time: 12:36)

Classification:

- ✓ **Learning (training):** Learn a model using the **training data**.
 Model construction: describing a set of predetermined classes; Each tuple/sample is assumed to belong to a predefined class, as determined by the class label. The set of tuples used for model construction is the training set.
- Testing:** Test the model using **unseen test data** to assess the model accuracy.
 Model usage: for classifying future or unknown objects. If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known.

$$Accuracy = \frac{\text{Number of correct classifications}}{\text{Total number of test cases}}$$

13

© Shyamanta M Bhattacharya, MIT, IIT Guwahati

Classification is a 2 step process, so we have in its first step, learning where in the idea is to learn a model using the training data, so that is also referred to as model construction which is about describing a set of predetermined classes, each tuple or sample is assumed to belong to a predefined class as determined by the class label and the set of tuples used for the model construction is referred to as the training set.

So, you have a training set which is fed to the learning algorithm and we arrive at a model, so this is either referred to as training or learning or model construction and then we have the testing phase where we test the model using unseen test data to assess the model accuracy. So, accuracy here refers to the number of correct classifications divided by the number of test cases.

And testing is also referred to as model uses, which is about classifying future or unknown objects, if accuracy is acceptable we use the model to classify data tuples whose class levels are not known.

(Refer Slide Time: 14:12)

Fundamental Assumption

Assumption: The **distribution of training examples is identical to the distribution of test examples** (including future unseen examples).

- ❑ In practice, this assumption is often violated to certain degree.
- ❑ Strong violations will clearly result in poor classification accuracy.
- ❑ To achieve good accuracy on the test data, **training examples must be sufficiently representative of the test data.**

Now, the fundamental assumption underlying such a learning process is that the distribution of training examples is identical to the distribution of test examples including future unseen examples. In practice, this assumption is often violated to certain degree and very strong violations will clearly result in poor classification accuracy. To achieve good accuracy under test data, the training examples must be sufficiently representative of the test data, so this is very important.

So, now having introduced the idea of classification, one of the forms of supervised learning, let us focus on one example of learning from observation which is learning a decision tree.

(Refer Slide Time: 15:17)

Learning Decision Trees

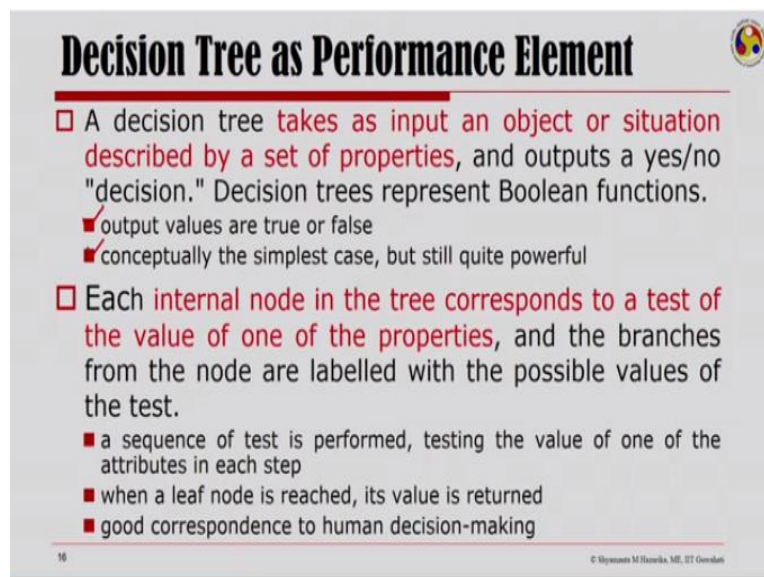
- ❑ Decision tree learning is **one of the most widely used techniques for classification.**
 - Its classification accuracy is competitive with other methods;
 - It is very efficient.
 - It serves as a good introduction to the area of inductive learning.

Decision tree learning uses a **decision tree (as a predictive model)** to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves)

So, learning decision trees is one of the most widely used technique for classification, its classification accuracy is competitive with other methods, decision trees is very efficient and

it serves as a good introduction to the area of inductive learning and this is precisely the reason why I have chosen here to talk of learning decision trees in this lecture today. So, decision tree learning uses a decision tree as a predictive model and we go from observations about an item which is represented in branches to conclusion about the items target value which is represented in trees. So basically a decision tree is a tree wherein every observation about an item is actually represented in the branch and the conclusion about the items target value is represented in the leaves.

(Refer Slide Time: 16:29)



Decision Tree as Performance Element

- A decision tree takes as input an object or situation described by a set of properties, and outputs a yes/no "decision." Decision trees represent Boolean functions.
 - ✓ output values are true or false
 - ✓ conceptually the simplest case, but still quite powerful
- Each internal node in the tree corresponds to a test of the value of one of the properties, and the branches from the node are labelled with the possible values of the test.
 - a sequence of test is performed, testing the value of one of the attributes in each step
 - when a leaf node is reached, its value is returned
 - good correspondence to human decision-making

16 © Sreyas M. Harshika, ME, IIT Guwahati

A decision tree can be looked at as a performance element for it takes as an input an object or a situation which is described by a set of properties and then it outputs a yes or no decision. Decision trees represent Boolean functions that is the output values are true or false and conceptually, they are one of the simplest cases but decision trees can be quite powerful.

So, each internal node in a decision tree actually corresponds to a test of the value of one of the properties and the branch, from that particular node are labelled with the possible values of the test. What we do is; we perform a sequence of tests and we test the value of one of the attributes in each step and when a leaf node is reach, its value is returned. Now, this is how we also take decisions given an option to us between 2 choices we would evaluate them and take one up and there after we look at what are the options available to us after we have arrived at that node and carry forward our decision.

(Refer Slide Time: 17:57)

Learning decision trees

Decide whether to wait for a table at a restaurant.

Aim is to learn a definition for the goal predicate WillWait, where the definition is expressed as a decision tree.

Setting this up as a learning problem; we decide what properties or attributes are available to describe examples in the domain:

1. **Alternate:** is there an alternative restaurant nearby?
2. **Bar:** is there a comfortable bar area to wait in?
3. **Fri/Sat:** is today Friday or Saturday?
4. **Hungry:** are we hungry?
5. **Patrons:** number of people in the restaurant (None, Some, Full)
6. **Price:** price range (\$, \$\$, \$\$\$)
7. **Raining:** is it raining outside?
8. **Reservation:** have we made a reservation?
9. **Type:** kind of restaurant (French, Italian, Thai, Burger)
10. **WaitEstimate:** estimated waiting time (0-10, 10-30, 30-60, >60)

17

© Shyamantak Mishra, MIT 6.034

Now, let us try to understand the concept of the decision tree with an example which is about deciding whether to wait for a table at a restaurant, so we visit a restaurant and then we find it is all full, whether we should wait there or visit another one depends on a number of factors like 1. if you are hungry and you know there are other restaurants nearby, you would rather leave and go to another one. But then if you know that there are no nearby restaurants, you would prefer to wait, so how does one take such decisions, let us try to understand this with this following example. So, our aim is to learn here some form of a definition for the goal predicate which is will wait so, will wait is a predicate which if true means, that our decision is to wait and if it is false, then we prefer not to eat in that restaurant.

And now, we will see in our discussion here that the will wait predicate, the whole definition is expressed as a decision tree. Now, setting this whole problem up of learning the goal predicates definition and we need to decide what properties or attributes are available to describe examples in the domain because here we are learning from observations, so we need to be given a couple of examples from the domain and then we will take a decision on whether to wait or not to wait. Now these are the attributes that we will try to use, one is called alternative that is defining is there an alternative restaurant nearby, there is an option on whether there is a comfortable sitting area to wait, we also base our decisions on waiting at restaurants or not, based on whether it is a Friday or a Saturday or based on whether you are hungry.

One important factor when you make a decision on whether you would wait or not in a given restaurant is on how many people are there already waiting, what is the price range, is it

raining outside or have we made a reservation, what type of a restaurant would we prefer and what would be the estimated waiting time.

(Refer Slide Time: 20:52)

Attribute-based Representations

- Examples described by **attribute values**
 - Boolean, discrete, continuous
 - E.g., situations where I will/won't wait for a table:
- Classification of examples is **positive (T)** or **negative (F)**

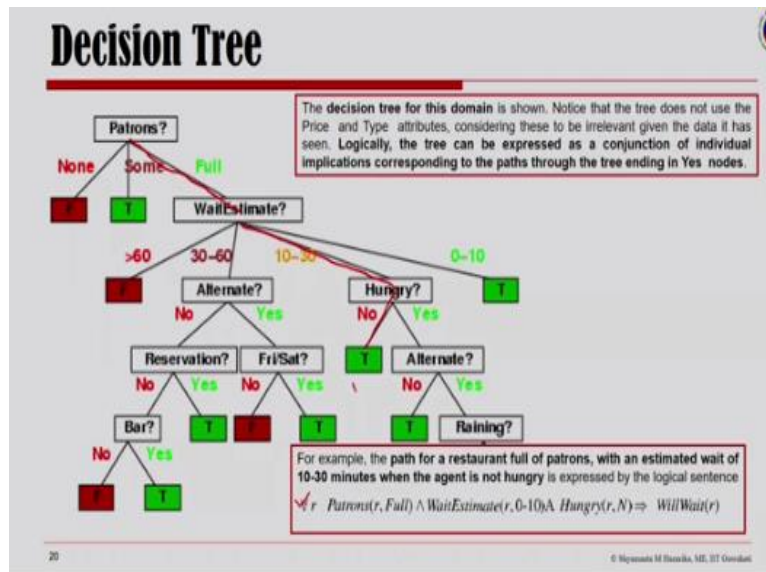
Example	Attributes										Target
	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	
X ₁	T	F	F	T	Some	\$\$\$	F	T	French	0-10	T
X ₂	T	F	F	T	Full	\$	F	F	Thai	30-60	F
X ₃	F	T	F	F	Some	\$	F	F	Burger	0-10	T
X ₄	T	F	T	T	Full	\$	F	F	Thai	10-30	T
X ₅	F	F	T	F	Full	\$\$\$	F	T	French	>60	F
X ₆	F	T	F	T	Some	\$\$	T	T	Italian	0-10	T
X ₇	F	T	F	F	None	\$	T	F	Burger	0-10	F
X ₈	F	F	F	T	Some	\$\$	T	T	Thai	0-10	T
X ₉	F	T	T	F	Full	\$	T	F	Burger	>60	F
X ₁₀	T	T	T	T	Full	\$\$\$	F	T	Italian	10-30	F
X ₁₁	F	F	F	F	None	\$	F	F	Thai	0-10	F
X ₁₂	T	T	T	T	Full	\$	F	F	Burger	30-60	T

Now, the examples are described by attribute values which could be Boolean, discrete or continuous even, like situations where I will or would not wait for a table could be just Boolean saying, we will wait is either positive or true or we will wait is not true. Classification of examples is there for either positive or negative, so we have 12 examples given to us.

Let me look at one example for you like, if you know there is an alternate restaurant nearby does not have very good seating space, it is not a Friday today, you are hungry and there are some patrons in this current restaurant and it is not raining outside and you know it is a good restaurant, you would love to wait, so you could have different outputs for the different examples.

And like let us focus on another one here, which is the fifth example where the estimated waiting time is somewhere less than an hour but you know the restaurant is full right now and you know there are alternate restaurants nearby, so you decide not to wait.

(Refer Slide Time: 22:31)



Now, given these examples to you which are observations we could create the decision tree. Now, even before creating the decision tree, I have shown you the decision tree for this domain here, so the decision tree for this domain shows that a couple of attributes are not used at all like, we have not used price and the type attribute considering this to be irrelevant given the data it has seen.

So, logically the decision tree can be expressed as a conjunction of individual implications corresponding to the paths through the tree ending in yes nodes. So, for example the path for a restaurant full of patrons with an estimated wait time of 10 to 30 minutes and the agent is not hungry is expressed by the following logical formula for all our patrons you have full wait estimate is between 0 and 10.

And I am not hungry and then you would prefer to wait, so this path here, patrons full 10 to 30, okay and not hungry, so prefer to wait.

(Refer Slide Time: 24:08)

Expressiveness

- Decision Trees can be expressed as implication sentences
- In principle, they can **express propositional logic sentences**
 - Each row in the truth table of a sentence can be represented as a path in the tree
 - Often there are more efficient trees
- Some functions require exponentially large decision trees
 - Parity function, Majority function.

21

© Siphamon M. Dvorak, M.E., IT Overhaul

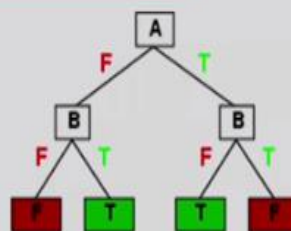
Now, you could see that decision trees can be expressed as implication sentences, in principle decision trees can express propositional logic sentences, each row in the truth table of a sentence can be represented as a path in the tree, often there are more efficient trees than this and some functions however, require exponentially large decision trees like parity functions and majority functions.

(Refer Slide Time: 24:47)

Expressiveness

Decision trees can express any function of the input attributes.
E.g., for Boolean functions, truth table row → path to leaf.

✓A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



Trivially, there is a consistent decision tree for any training set with one path to leaf for each example (unless f nondeterministic in x) but it probably won't generalize to new examples.

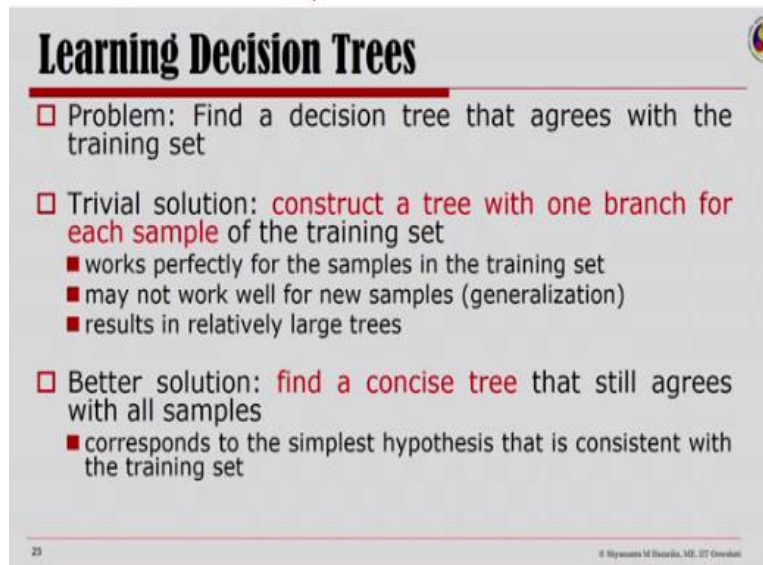
22

© Siphamon M. Dvorak, M.E., IT Overhaul

However, decision trees are quite expressive and decision trees can express any function of the input attributes, so here is an xor, A, B, A xor B, the truth table is shown here and the corresponding decision tree is shown on the right. So, if I have A false, I come here and B false, so I have A xor B as false which is shown here in the leaf of the decision tree or for that example again, I have A true, so I come this path and I have B false, I come this path and I have xor true, which is here shown as leaf of the decision tree, so trivially there is a consistent

decision tree for any training set with one path to leave for each example, unless of course F is non-deterministic in x but it probably would not generalize to new examples.

(Refer Slide Time: 25:59)



Learning Decision Trees

- Problem: Find a decision tree that agrees with the training set
- Trivial solution: **construct a tree with one branch for each sample** of the training set
 - works perfectly for the samples in the training set
 - may not work well for new samples (generalization)
 - results in relatively large trees
- Better solution: **find a concise tree** that still agrees with all samples
 - corresponds to the simplest hypothesis that is consistent with the training set

23 © Shyamantak Mishra, IIT Guwahati

So, now let us look at the problem of learning decision trees so, the problem is to find a decision tree that agrees with the training set. The trivial solution would be to construct a tree with one branch for each sample of the training set, now this works perfectly for the samples in the training set, may not work well for new samples that means, such a decision tree would not generalize and it will of course result in relatively large trees.

So, can we do better than this? The idea is to find a concise tree that still agrees with all samples and it corresponds to the simple hypothesis that is consistent with the training set. How do we go about getting to that concise tree?

(Refer Slide Time: 26:57)

Constructing Decision Trees

- In general, constructing the smallest possible decision tree is an intractable problem
- Algorithms exist for constructing reasonably small trees
- Basic idea: ✓ test the most important attribute first
 - Attribute that makes the most difference for the classification of an example.
 - ✓ Can be determined through information theory
 - Hopefully will yield the correct classification with few tests.

24

© Raymond M. Haralik, M.E., UT Graduate

So, constructing the smallest possible decision tree actually is an intractable problem. Algorithms exist for constructing reasonably small trees. The basic idea in constructing a decision tree is first to test the most important attribute, the most important attribute is the one that makes the most difference for the classification of an example and once that is done, we then go on to look at the next most important attribute and carry this forward.

Now, how does one get to the most important attribute; the most important attribute can be determined through information theory.

(Refer Slide Time: 27:51)

Decision Tree Learning

Aim: ✓ find a small tree consistent with the training examples
Idea: ✓ (recursively) choose "most significant" attribute as root of (sub)tree

```
function DTL(examples, attributes, default) returns a decision tree
  if examples is empty then return default
  else if all examples have the same classification then return the classification
  else if attributes is empty then return MODE(examples)
  else
    ✓ best ← CHOOSE-ATTRIBUTE(attributes, examples)
    ✓ tree ← a new decision tree with root test best
    for each value  $v_i$  of best do
      examplesi ← {elements of examples with best =  $v_i$ }
      subtree ← DTL(examplesi, attributes - best, MODE(examples))
      add a branch to tree with label  $v_i$  and subtree subtree
    return tree
```

25

© Raymond M. Haralik, M.E., UT Graduate

So, let us look at that and try to understand the decision tree learning algorithm. So the decision tree learning algorithm is about finding a small tree consistent with the training examples. Now as I was explaining, the idea is to choose the most significant attribute first as

the root of the tree thereafter, identify the next most significant attribute and carry forward this, so it is about recursively choosing the significant attribute as root of the tree.

And once I get the best attribute, I then try to break this up based on the best attribute into 2 sub trees with the root as the best one and then continue this forward.

(Refer Slide Time: 28:50)

Choosing an attribute

□ Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"

✓ Splitting the examples by testing on attributes. Patrons is a good attribute to test first!

Type is a poor attribute, because it leaves us with four possible outcomes, each of which has the same number of positive and negative answers.

□ Patrons? is a better choice

Patrons is a fairly important attribute, because if the value is None or Some, then we are left with example sets for which we can answer definitively (No and Yes, respectively). If the value is Full, we will need additional tests.

28 © Sreyas M Shrivastava, MIT, ET Gordon

So, the idea of getting a good attribute that splits the examples into subsets that are ideally all positive or all negative is what would be the best idea to look for the first attribute on which I would split the decision tree. So, splitting the examples by testing on the attributes in the given table that we have patron is a good attribute to test first now, why is patron a good attribute to test first?

This is because we could see that it has 2 negatives, 4 positives and it has some confusion with regards to full, type on the other hand is a poor attribute because it leaves us with 4 possible outcomes and these have almost the same number of positive and negative answers. Now, as already explained patron is a fairly important attribute a good one for us because if the value is none or some, then we are left with example sets for which we can answer definitely yes or no.

(Refer Slide Time: 30:24)

Using Information Theory

□ Implement Choose-Attribute in the DTL algorithm

We need a formal measure of "fairly good" and "really useless". The measure should have its maximum value when the attribute is perfect and its minimum value when the attribute is of no use at all. One suitable measure is the expected amount of information provided by the attribute, where we use the term in the mathematical sense from information theory.

□ Information Content?

To understand the notion of information, think about it as providing the answer to a question, for example, whether a coin will come up heads.

Information theory provides a mathematical basis for measuring the information content.

Let E be an event that occurs with probability $P(E)$.

If we are told that E has occurred, then we received

$$I(E) = \log_2 \frac{1}{P(E)}$$

Result of a fair coin flip provides 1 bit of information

bits of information.

27

© Raymond S. Sturges, M.E., ET Consultant

If the value is full, then possibly we will need some additional tests to be done. Now, this very idea that we can choose attributes in the decision tree learning algorithm by finding out a good attribute first, requires that we need a formal measure of what we mean by fairly good or useless for that matter attributes. Now, the measure should have its maximum value when the attribute is perfect and its minimum value when the attribute is of no use at all.

One measure that we use is the expected amount of information provided by the attribute and we use the term in its mathematical sense from information theory. So we look at what is called the information content of that particular attribute. Let us try to understand information content vis-à-vis information theory, to understand the notion of information think about it has providing the answer to a question for example, if I am talking of flipping a coin, whether a coin will come up head. Now, information theory provides a mathematical basis for measuring the information content of an answer. If I have E as an even and the probability of E occurring is PE and then if I am told that E has occurred, then we have received \log_2 of 1 over PE amount of information, that is if I was talking of flipping a coin and I was thinking of whether I will get heads, I know the probability of heads is 1 over 2 . And if I take here that probability and I take \log to the base 2 , I would know that the result of a fair coin flip would provide one bit of information. So any attribute will provide me with some form of information and that is what I will try to exploit in choosing the best attribute in the decision tree learning algorithm.

(Refer Slide Time: 32:57)

Entropy



- Interested in the information content of a source; rather than the information of any particular message. Information content is the average information per message.

Information content is also called entropy.

- Suppose we have an information source S which emits symbols from an alphabet $\{s_1, \dots, s_k\}$ with probabilities $\{p_1, \dots, p_k\}$ and each emission is independent of the others.
- **Entropy is the average amount of information** we get when we observe a symbol emitted by S .

Entropy depends only on the probability distribution and not on the alphabet used by S .

$$H(S) = -\sum_i p_i \log \frac{1}{p_i}$$

28

© Shyamantak Mishra, M.E., ET Oosthuizen

So, we look at the concept called entropy which is the information content of a particular message. Now, if you are interested in the information content of a source rather than just the message, then the information content is the average information that is coming, suppose we have a source S which emits symbols from an alphabet S_1 to S_k with probabilities p_1, p_2, \dots on and so forth up to p_k . Now, each emission being independent of the other, then entropy is defined as the average amount of information we get, when we observe a symbol emitted by S . So the information that I get from the source S , which is the average of the information this is what is entropy. Now, if you closely look at this, we will see that entropy depends on the probability distribution and not on the alphabet used by us because S could emit any alphabet but the average amount of information that is coming to me is based on the probability distribution.

(Refer Slide Time: 34:40)

Entropy



- The entropy is the average amount of information for a set of examples D

$$entropy(D) = -\sum_{j=1}^{|C|} \Pr(c_j) \log_2 \Pr(c_j)$$

$$\sum_{j=1}^{|C|} \Pr(c_j) = 1$$

- ✓ $\Pr(c_j)$ is the probability of class c_j in data set D ; We use entropy as a **measure of impurity or disorder** of data set D . (Or, a measure of information in a tree)

30

© Shyamantak Mishra, M.E., ET Oosthuizen

Now, let us try to understand this by going back to the example of flipping the coin, now if I am tossing a fair coin, then I know that the probability of a head is 1 over 2 and the probability of a tail is 1 over 2, so if I am looking at the information content or the entropy of the head and the tail flips, then all I have is one bit of information. On the other hand, if the coin is loaded and let us assume, that it is loaded in such a way that 99% time it is going to be head. Then I know that the probability of head is 99 over 100, probability of tail is 1 over 100 and if I am looking at now, the average information content or the entropy I know it is going to be 0.08 bit now, this is interesting to note. What is interesting to note is that as the probability of head goes to 1, the information of the actual answer goes to 0 that means, when I know for sure then the information content goes down. And therefore, when I am taking decision, I would rather take decisions on attributes where the information content is high, so the entropy which is the average amount of information for a set of examples D is given by the average of the information content, here the probability of the class C in data D is given as Pr C of j, so we use entropy as a measure of impurity or disorder of the data set.

(Refer Slide Time: 37:02)

Entropy

- An estimate of the probabilities of the possible answers before any of the attributes have been tested is given by the proportions of positive and negative examples in the training set.
- Suppose the training set contains p positive examples and n negative examples. Then an estimate of the information contained in a correct answer is

$$I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = -\frac{p}{p+n} \log_2 \frac{p}{p+n} - \frac{n}{p+n} \log_2 \frac{n}{p+n}$$

Or it can be seen as a measure of information in a tree. Now, an estimate of the probabilities of the possible answer before any of the attributes have been tested is given by the proportion of positive and negative examples in the training set. So, I would try to understand the probabilities or possible answers of the attributes, now how do I get to the probabilities of possible answers?

I get it by the proportion of positive and negative examples in the training set that is suppose, the training set contains P positive examples and N negative examples, then an estimation can

be made of the information content of a correct answer and that could be given by what we have been looking at the information of the probability of the positives and the negatives.

(Refer Slide Time: 38:10)

Information Gain

- How much information we still need after the attribute test?
- A chosen attribute A divides the training set E into subsets E_1, \dots, E_v according to their values for A , where A has v distinct values.

A random example has the i th value for the attribute with probability $(p_i + n_i) / (p + n)$.

$$\text{remainder}(A) = \sum_{i=1}^v \frac{p_i + n_i}{p + n} I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right)$$

- **Information Gain** (IG) or reduction in entropy from the attribute test:

$$\text{IG}(A) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - \text{remainder}(A)$$

Now, question is how much information did we gain in this process of looking at an attribute and trying to understand its entropy. This will depend on how much information we still need after the attribute test that is let us say, I have chosen an attribute A and the attribute divides the training set into subsets; E_1, E_2, \dots, E_v according to their values for A , where A has v distinct values.

Then, the amount of information that I would still need after I have taken away this is actually, the amount of information that would be still covered by these v subsets because I have this attribute A which is dividing the training set into some v subsets, according to their values and each of them have v distinct values, then that would be the information we will still need after the attribute test is done.

And given that information which I refer to as the remainder of A , the information gain or the reduction in entropy from the attribute test is actually the information gained because of that attribute minus the remainder of the information that is still required. Given this information gain, I can now see which is the attribute that has the highest information content.

(Refer Slide Time: 40:04)

Information gain

For the training set,

$$I\left(\frac{6}{12}, \frac{6}{12}\right) = 1$$

Consider the attributes *Patrons* and *Type*; and others too.

$$IG(Patrons) = 1 - \left[\frac{2}{12} I\left(\frac{0}{2}, \frac{2}{2}\right) + \frac{4}{12} I\left(\frac{1}{4}, \frac{3}{4}\right) + \frac{6}{12} I\left(\frac{2}{6}, \frac{4}{6}\right) \right] = .0541 \text{ bits}$$

$$IG(Type) = 1 - \left[\frac{2}{12} I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{2}{12} I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{4}{12} I\left(\frac{2}{4}, \frac{2}{4}\right) + \frac{4}{12} I\left(\frac{2}{4}, \frac{2}{4}\right) \right] = 0 \text{ bits}$$

Patrons has the highest IG of all attributes and so chosen by the DTL algorithm as the root.

33 © Raymond M. Stalla, M.S., ET Consultant.

So, for the training set that is given to me, if you recall I have 6 positive and 6 negative examples of the total 12 examples that I have if you see I have been given 6 positive and 6 negative examples to decide on my training, so for the training set I have 6 positive and 6 negative examples and the information content is equal to 1 bit. Consider the attributes now, let us take 2 attributes; one the patron and the other the type.

We can take the others as well one after another but let us consider these 2 first. So if I was looking at the information gained for patrons, I could see that the total information is 1 bit but then the remainder is that I have 2 which are negatives, 4 which are positives and 6 out of the 12, out of which 2 are positive and 4 are negatives. So this is important for me to realize that if I am using patron as an attribute, it breaks this data set into 3 subsets; non, some and full. And non has all negatives, so the probability of an yes is 0, all of it is no, some has all of them to be yes, nothing to be no and the third categorization here which have 6 elements in it, 2 of them are positive and 4 of them are negative. So I can find out the information content of patron which comes out as 0.0541 bits similarly, I can look at the information gain of type. So, I have type breaking up as French, Italian, Thai and burger, so 4 subgroups and 2 of them is in French which is 1 positive, 1 negative, Italian; again 2 out of 12, 1 positive, 1 negative, Thai; 4 out of 12, 2 positives, 2 negatives. Burger; 4 out of 12, 2 positives, 2 negatives, so if I try to find out the information gain of type, I get the information gained as 0. From this analysis, one thing is clear that patron has the highest information gain of all attributes.

And the information gained based on type is nil, so when I am using the choose attribute function for my decision tree learning algorithm, patron is chosen as the root.

(Refer Slide Time: 43:59)

The Decision Tree Induced

- Decision tree learned from the 12 example-training set.

- Substantially simpler than "true" tree - a more complex hypothesis isn't justified by small amount of data.

34 © Sreyas M Shrivastava, MS, ET Graduate

And the decision tree learned from the 12 example training set that I have is shown here, so patron has the highest information gain, so it is the root of the tree. The next attribute is hungry thereafter, we have type and this is if you notice is a substantially simpler tree than the true tree that I have shown previously, now a more complex hypothesis is not justified by a small amount of data.

(Refer Slide Time: 44:46)

Performance of Decision Tree Learning

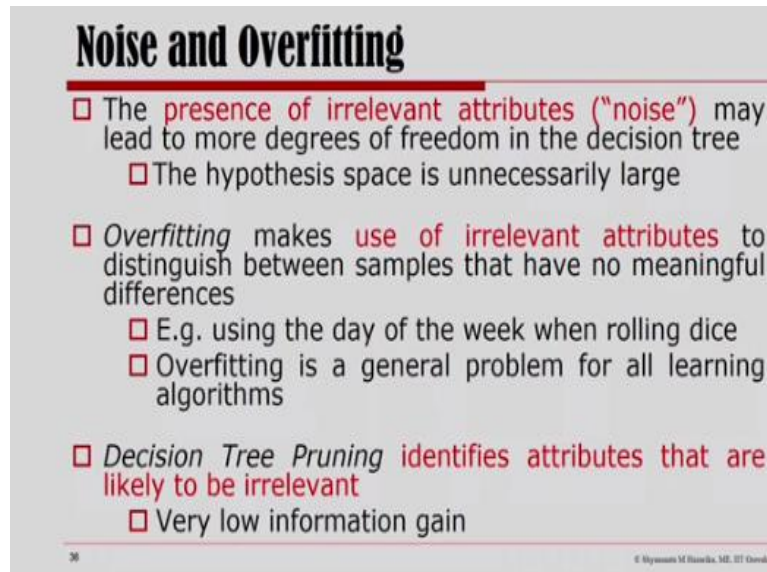
- Quality of Predictions
 - Predictions for the classification of unknown examples that agree with the correct result are obviously better.
 - Can be measured easily.
 - It can be assessed in advance by splitting the available examples into a training set and a test set
 - learn the training set, and assess the performance via the test set

35 © Sreyas M Shrivastava, MS, ET Graduate

And this tree is good enough for the 12 example training set that we have. Now, how do you decide on the quality or the performance of the decision tree learning? The performance of the decision tree learning is based on the quality of predictions, so predictions for classification of unknown examples that agree with the correct result are obviously, better and they can be measured easily.

Now, one can access this in advance by splitting the available examples into a training set and a test set and one can learn the training set and access the performance via the test set. The size of the tree is another performance criteria of the decision tree learning, a smaller tree specially depth wise is a more concise representation and is a better result for the decision tree learning algorithm.

(Refer Slide Time: 45:47)



Noise and Overfitting

- The presence of irrelevant attributes ("noise") may lead to more degrees of freedom in the decision tree
 - The hypothesis space is unnecessarily large
- Overfitting makes use of irrelevant attributes to distinguish between samples that have no meaningful differences
 - E.g. using the day of the week when rolling dice
 - Overfitting is a general problem for all learning algorithms
- Decision Tree Pruning identifies attributes that are likely to be irrelevant
 - Very low information gain

36 © Mounir M. Hachem, M.S., IT Specialist

The presence of irrelevant attributes which is referred to as noise may lead to more degrees of freedom in the decision tree and under such cases, the hypothesis space is unnecessarily large and one other problem is the problem of over fitting which is actually making use of irrelevant attributes to distinguish between samples that have no meaningful differences. Example; if I was talking of rolling of a dice and then using the attribute of the day of the week to decide on the result, then that is an irrelevant attribute.

And somehow leading to the related problem of over fitting. Now over fitting is a general problem for all learning algorithms and then one related concept is the concept of decision tree pruning which is about identifying those attributes that are likely to be irrelevant and then we can leave these attributes out from the decision tree, these attributes have very low information gain.

(Refer Slide Time: 47:09)

Avoid Overfitting

- **Overfitting:** A tree may overfit the training data
 - Good accuracy on training data but poor on test data
 - Symptoms: tree too deep and too many branches, some may reflect anomalies due to noise or outliers
- Two approaches to avoid overfitting
 - **Pre-pruning:** Halt tree construction early
 - Difficult to decide because we do not know what may happen subsequently if we keep growing the tree.
 - **Post-pruning:** Remove branches or sub-trees from a “fully grown” tree.
 - This method is commonly used. C4.5 uses a statistical method to estimate the errors at each node for pruning.
 - A validation set may be used for pruning as well.

27

© Sigamita M. Hartzel, ME, IT, Oostland

Over fitting or a tree that would over fit the training data could have good accuracy on training data but would have very poor accuracy on test data, so trees that are too deep and have too many branches where some may reflect anomalies due to noise or outliers could be symptoms of over fitting a decision tree. There are as already hinted at 2 approaches to avoid over fitting.

One we can think of post pruning and other is about taking actions pre pruning. So we could halt tree construction early but it is very difficult to decide because we do not know what may happen subsequently, if we keep growing the tree. Post pruning; we remove branches or sub trees from a full grown tree, now, this method is commonly used. C4.5, which uses a statistical method to estimate the error at each node for pruning is a very popular algorithm.

(Refer Slide Time: 48:26)

Broadening the Applicability

- **Extend decision tree induction to a wider variety of problems;** a number of issues must be addressed:
 1. Missing data
In many domains, not all the attribute values will be known for every example. The values may not have been recorded, or they may be too expensive to obtain.
 2. Multivalued Attributes
When an attribute has a large number of possible values, the information gain measure gives an inappropriate indication of the attribute's usefulness.
 3. Continuous-valued Attributes
Certain attributes (such as Height; Weight) have a large or infinite set of possible values. They are therefore not well-suited for decision-tree learning in raw form.

A decision-tree learning system for real-world applications must be able to handle all of these problems. **Handling continuous-valued variables is especially important** - physical and financial processes provide numerical data. Several commercial packages have been built that meet these criteria, and they have been used to develop several hundred fielded systems.

28

© Sigamita M. Hartzel, ME, IT, Oostland

A validation set may be used for pruning as well now, you can extend decision tree induction to a wider variety of problems and this can happen if we could address a number of issues. One; we should be able to talk of missing data, in many domains not all attribute values will be known for every example, the values may not have been recorded or they may be too expensive to obtain.

So, decision tree learning algorithms should be able to deal with missing data, it should be also able to handle multivalued attributes that is when an attribute has a large number of possible values, then the information gained measure would only give an inappropriate indication of the attributes usefulness, so we should be able to do better than just the information gained.

Certain attributes such as height, weight or even the price that we have been talking of in this given example have a large or infinite set of possible values, they are therefore not well-suited for decision tree learning in its raw form, so a decision tree learning algorithm for real world application must be able to handle all of these problems, handling continuous valued variables is especially important, Physical processes, financial processes all of these in real world include numerical data. Now, several commercial packages have been built that meet these criteria and they have been used to develop several hundred fielded systems. We have looked at a very simple example from 12 observations to arriving at a decision tree and we have looked at how information content or entropy is exploited to make the choice of the attribute to arrive at a decision tree.

We will look at the second form of supervised learning which is regression in the next lecture, thank you.