Fundamentals of Artificial Intelligence

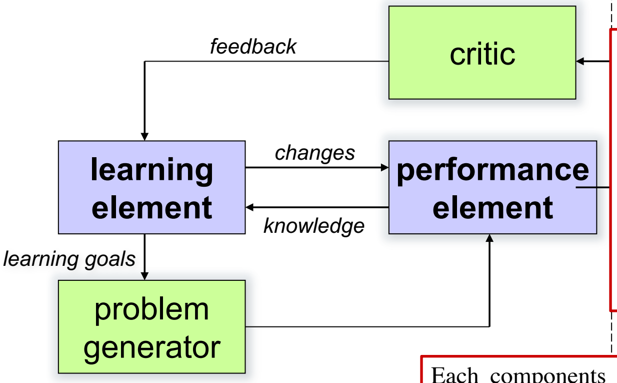
Learning Decision Trees

Shyamanta M Hazarika
Mechanical Engineering
Indian Institute of Technology Guwahati
s.m.hazarika@iitg.ac.in

http://www.iitg.ac.in/s.m.hazarika/

Architecture of a Learning System

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.**



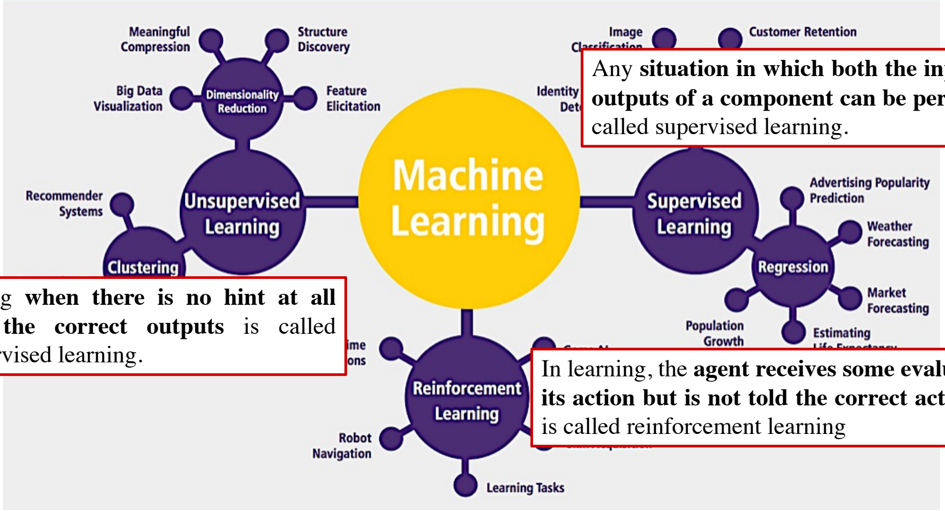
Build the performance element:

1. Mapping from current state to actions.
2. World from percept sequence.
3. Way the world evolves.
4. Results of possible actions.
5. Utility indicating desirability of states.
6. Action-value information
7. Maximizes the agent's utility.

Each of the components can be learned, given the appropriate feedback.

Each components of the performance element can be described mathematically as a function: **All learning can be seen as learning the representation of a function.**

Machine Learning



Any **situation in which both the inputs and outputs of a component can be perceived** is called supervised learning.

Learning **when there is no hint at all about the correct outputs** is called unsupervised learning.

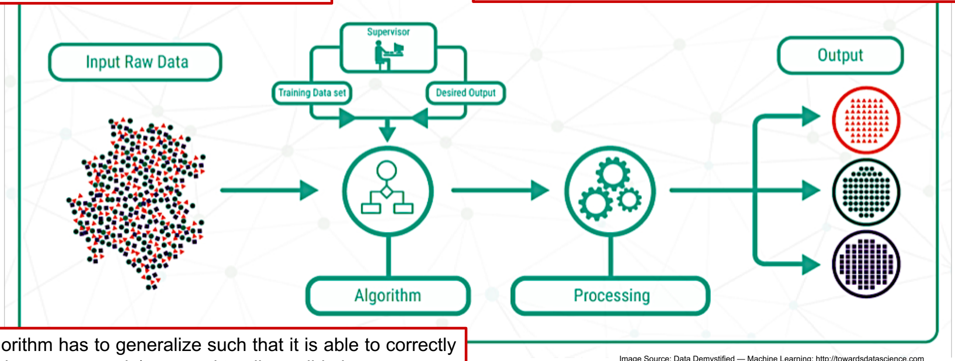
In learning, the **agent receives some evaluation of its action but is not told the correct action**. This is called reinforcement learning

Image Source: DHL, Artificial Intelligence in Logistics, 2018.

Supervised Learning

The set of (training/learning) data consists of a set of input data and correct responses (labels) corresponding to every piece of data.

Supervised learning: a type of machine learning that learns from training data with labels as learning targets. It is the most widely used type of machine learning



The algorithm has to generalize such that it is able to correctly (or with low error margin) respond to all possible inputs.

Image Source: Data Demystified — Machine Learning; http://towardsdatascience.com

Learn to predict output when **given an input vector**

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.

Inductive Learning

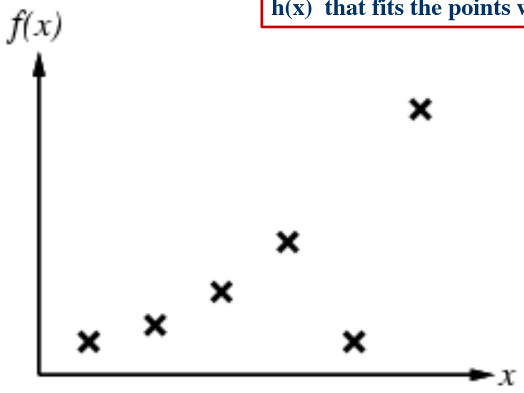
- □ In supervised learning, the learning element is given the correct (or approximately correct) value of the function for particular inputs, and changes its representation of the function to try to match the information provided by the feedback.
  - ■ An example is a pair (x, f(x)), where x is the input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

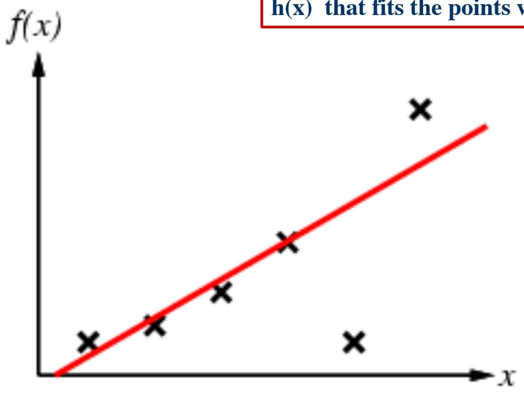
Inductive Learning

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.**



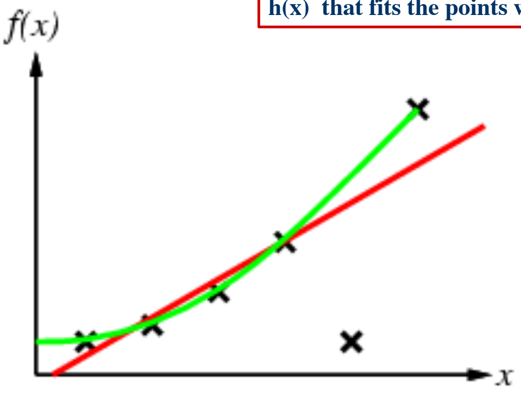
Inductive Learning

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.**



Inductive Learning

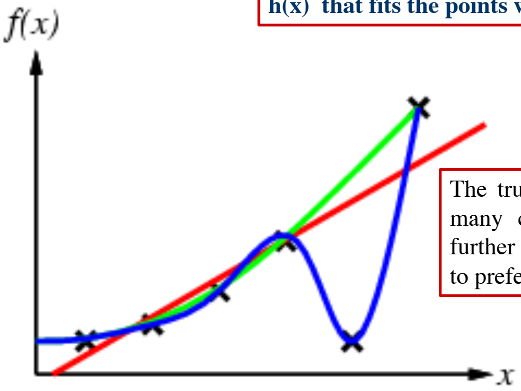
From plane geometry: Examples are (x,y) points in the plane, where y = f(x). **The task is to find a function h(x) that fits the points well.**



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.**

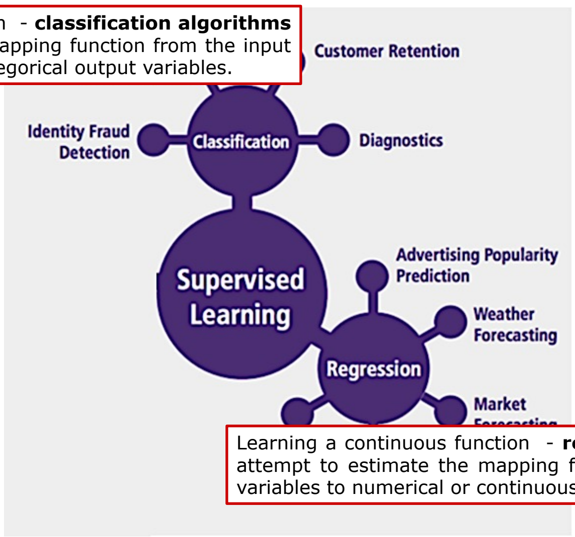


The true *f* is unknown, so there are many choices for *h*, but without further knowledge, we have no way to prefer one over the other.

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.

Supervised Learning

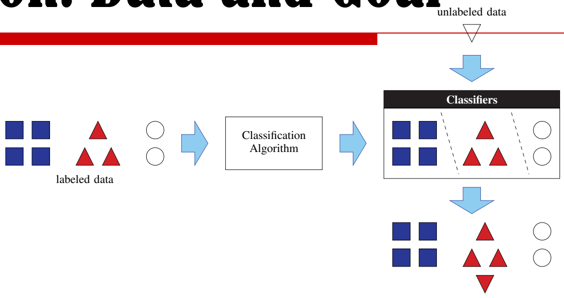
Learning a discrete function - **classification algorithms** attempt to estimate the mapping function from the input variables to discrete or categorical output variables.



Learning a continuous function - **regression algorithms** attempt to estimate the mapping function from the input variables to numerical or continuous output variables.

Image Source: DHL, Artificial Intelligence in Logistics, 2018.

Classification: Data and Goal



- □ Data: A **set of data records** (also called examples, instances or cases) described by
  - ■ *k* attributes: A_1, A_2, … A_k.
  - ■ a 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.

Classification:

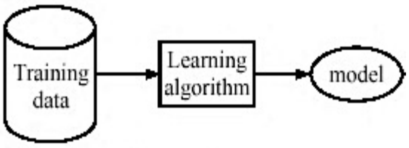
1. Learning (training): Learn a model using the training data.

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.

2. 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

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



Step 1: Training

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.

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)

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

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:

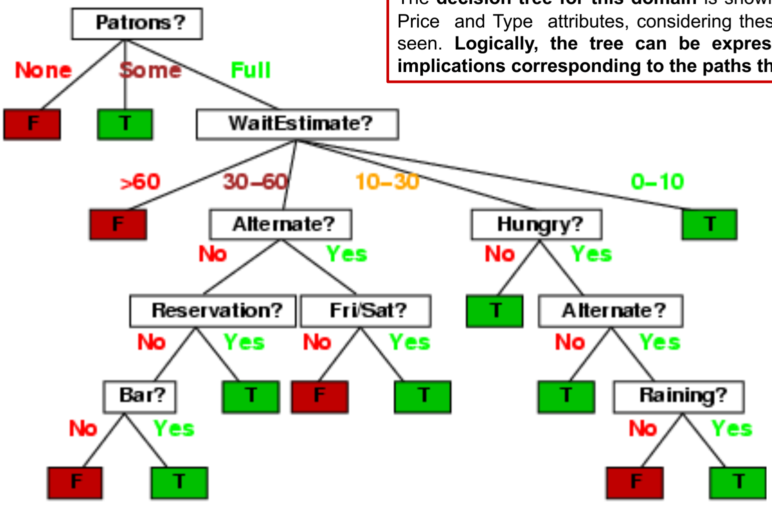
1. **Alternate**: is there an alternative restaurant nearby?
2. **Bar**: is there a comfortable bar area to wait in?
3. **Fri/Sat**: is today Friday or Saturday?
4. **Hungry**: are we hungry?
5. **Patrons**: number of people in the restaurant (None, Some, Full)
6. **Price**: price range ($, $$, $$$)
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)

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)

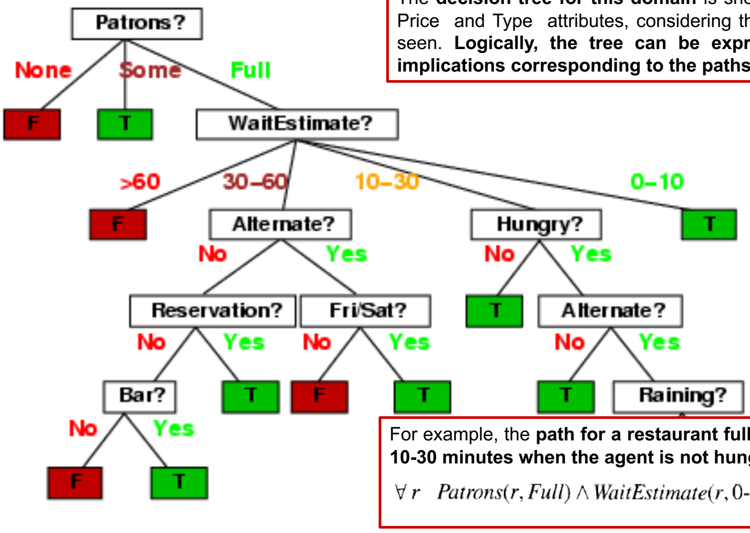
Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	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

Decision Tree



The **decision tree for this domain** is shown. Notice that the tree does not use the Price and Type attributes, considering these to be irrelevant given the data it has seen. **Logically, the tree can be expressed as a conjunction of individual implications corresponding to the paths through the tree ending in Yes nodes**.

Decision Tree



The **decision tree for this domain** is shown. Notice that the tree does not use the Price and Type attributes, considering these to be irrelevant given the data it has seen. **Logically, the tree can be expressed as a conjunction of individual implications corresponding to the paths through the tree ending in Yes nodes**.

For example, the **path for a restaurant full of patrons, with an estimated wait of 10-30 minutes when the agent is not hungry** is expressed by the logical sentence

$$\forall r \quad Patrons(r, Full) \wedge WaitEstimate(r, 0\text{-}10) \wedge \ Hungry(r, N) \Rightarrow \ \ WillWait(r)$$

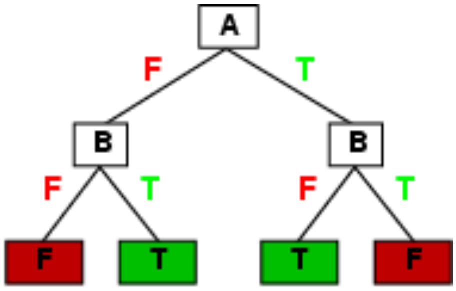
Expressiveness

- 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.

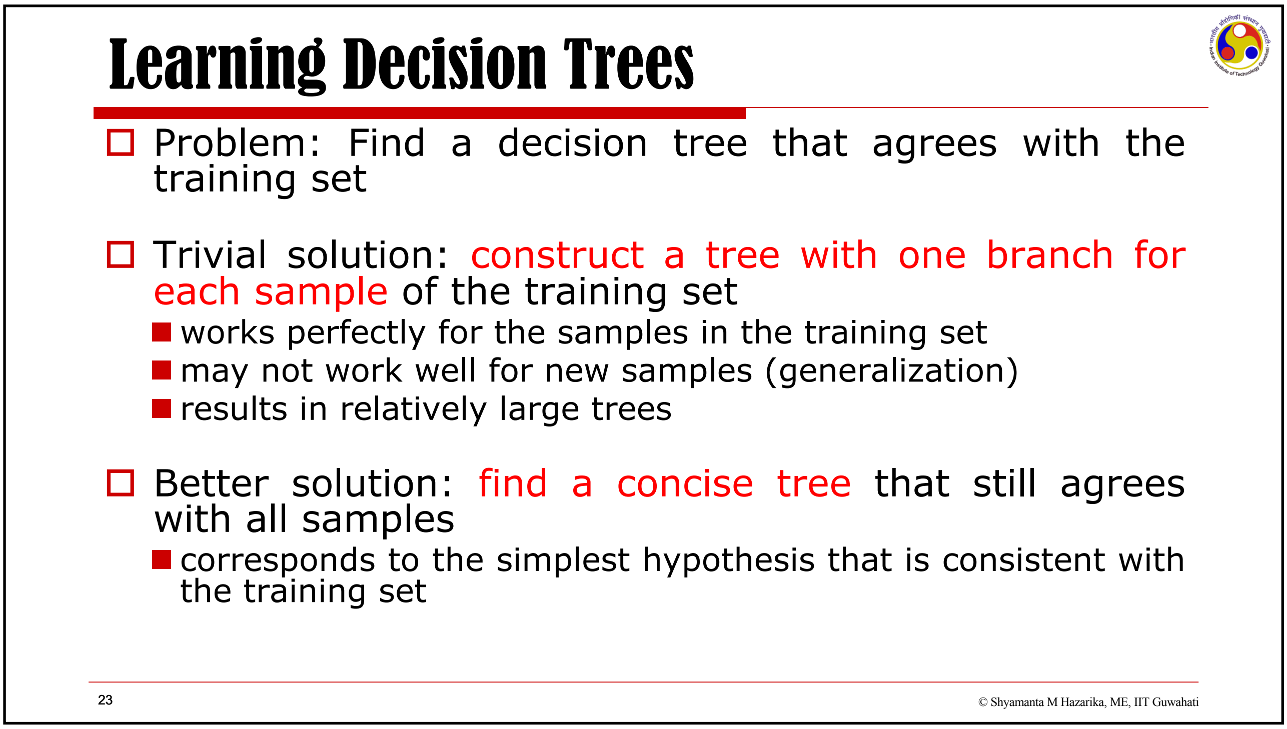
Expressiveness

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



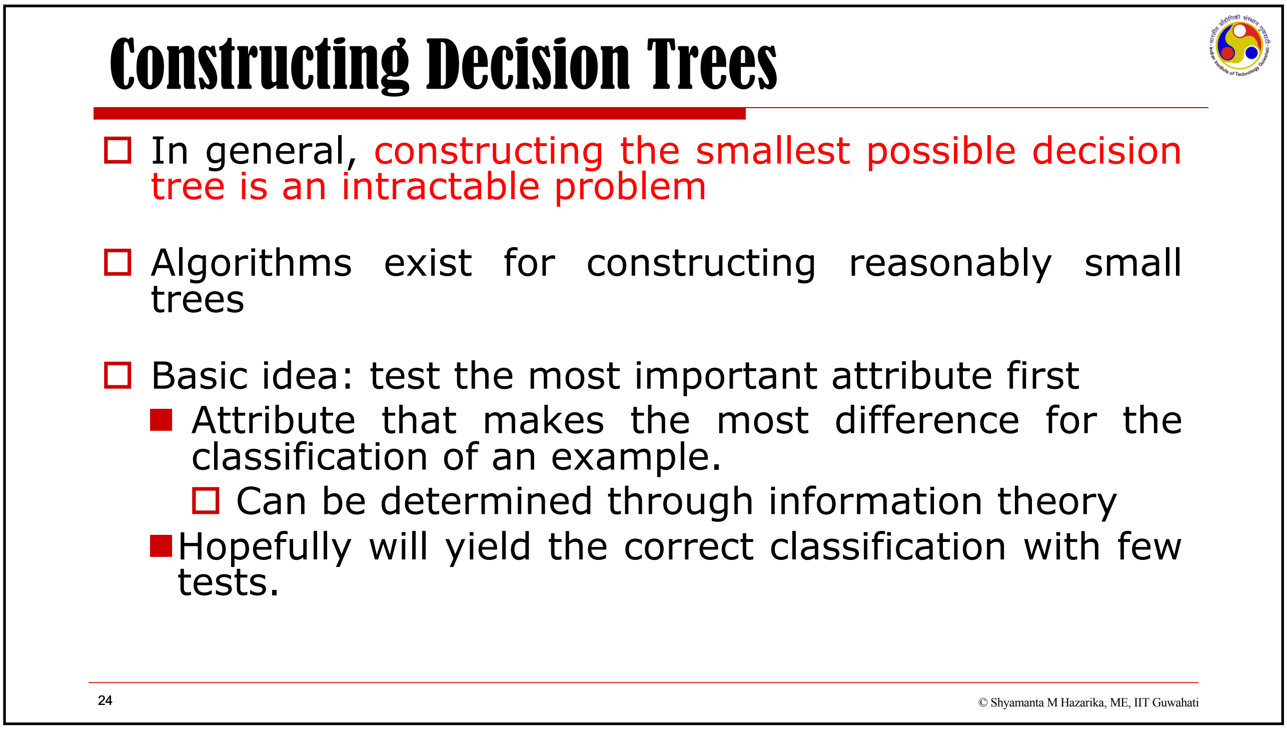
Trivially, there is a consistent decision tree for any training set with one path to leaf for each example (unless *f* nondeterministic in *x*) but it probably won't generalize to new examples.



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



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

Decision Tree Learning

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

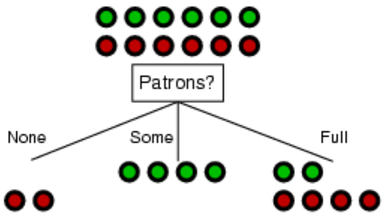
```
function DTL($examples$, $attributes$, $default$) returns a decision tree
    if $examples$ is empty then return $default$
    else if all $examples$ have the same classification then return the classification
    else if $attributes$ is empty then return MODE($examples$)
    else
        $best \leftarrow$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

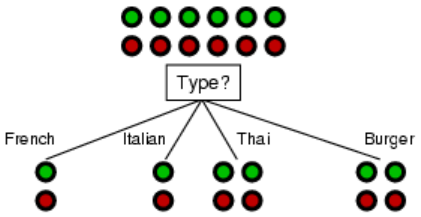
Choosing an attribute

- ☐ 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.

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.

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 $\{s_1, …., s_k\}$ with probabilities $\{p_1,…, p_k\}$ 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.

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

Entropy

- □ For tossing of a *fair* coin the average information content is given as

$$I\left(\frac{1}{2},\frac{1}{2}\right) = -\ \frac{1}{2}\ \log_2\frac{1}{2} - \ \frac{1}{2}\ \log_2\frac{1}{2}\ \ =\ 1 \text{ bit.}$$

If the coin is loaded to give 99% heads we get

$$I\left(\frac{1}{100},\frac{99}{100}\right) = -\ \frac{1}{100}\ \log_2\frac{1}{100} - \ \frac{99}{100}\ \log_2\frac{99}{100}\ \ =\ 0.08 \text{ bit.}$$

- □ As the probability of heads goes to 1, the information of the actual answer goes to 0.

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)

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

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

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, … , E_v according to their values for *A*, where *A* has *v* distinct values.

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

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

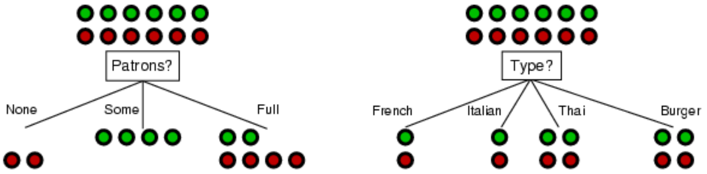
- □ **Information Gain** (IG) or reduction in entropy from the attribute test:

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

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.

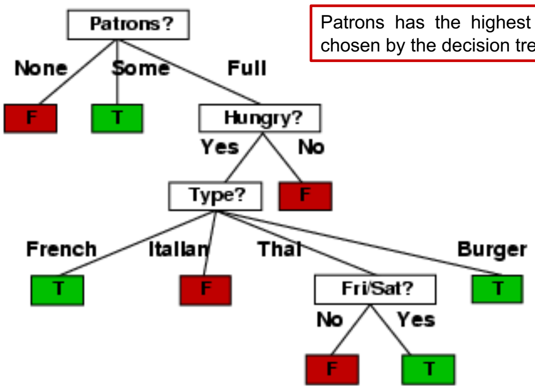
$$IG(Patrons) = 1 - [\frac{2}{12}I(0,1) + \frac{4}{12}I(1,0) + \frac{6}{12}I(\frac{2}{6},\frac{4}{6})] = .0541 \text{ bits}$$

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

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

The Decision Tree Induced

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



Patrons has the highest gain of any of the attributes and chosen by the decision tree learning algorithm as the root.

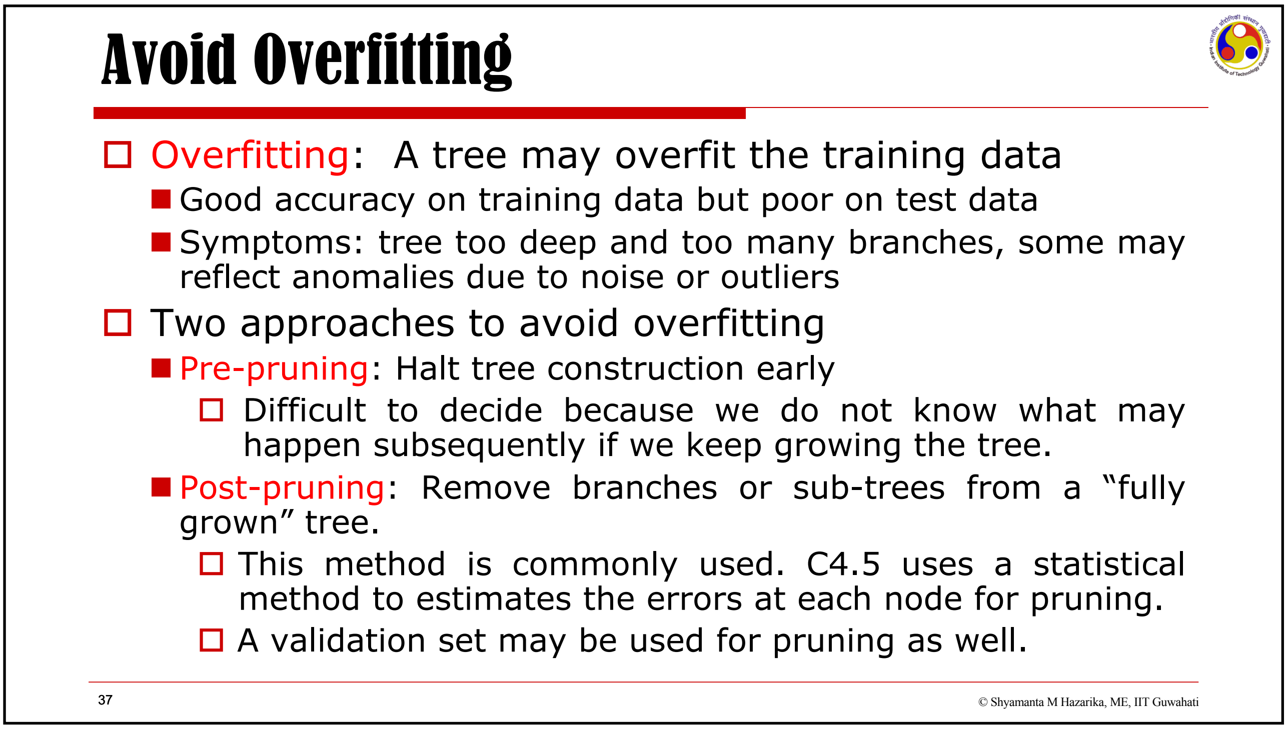
□ Substantially simpler than "true" tree - a more complex hypothesis isn't justified by small amount of data.

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
- Size of the tree
  - A smaller tree (especially depth-wise) is a more concise representation

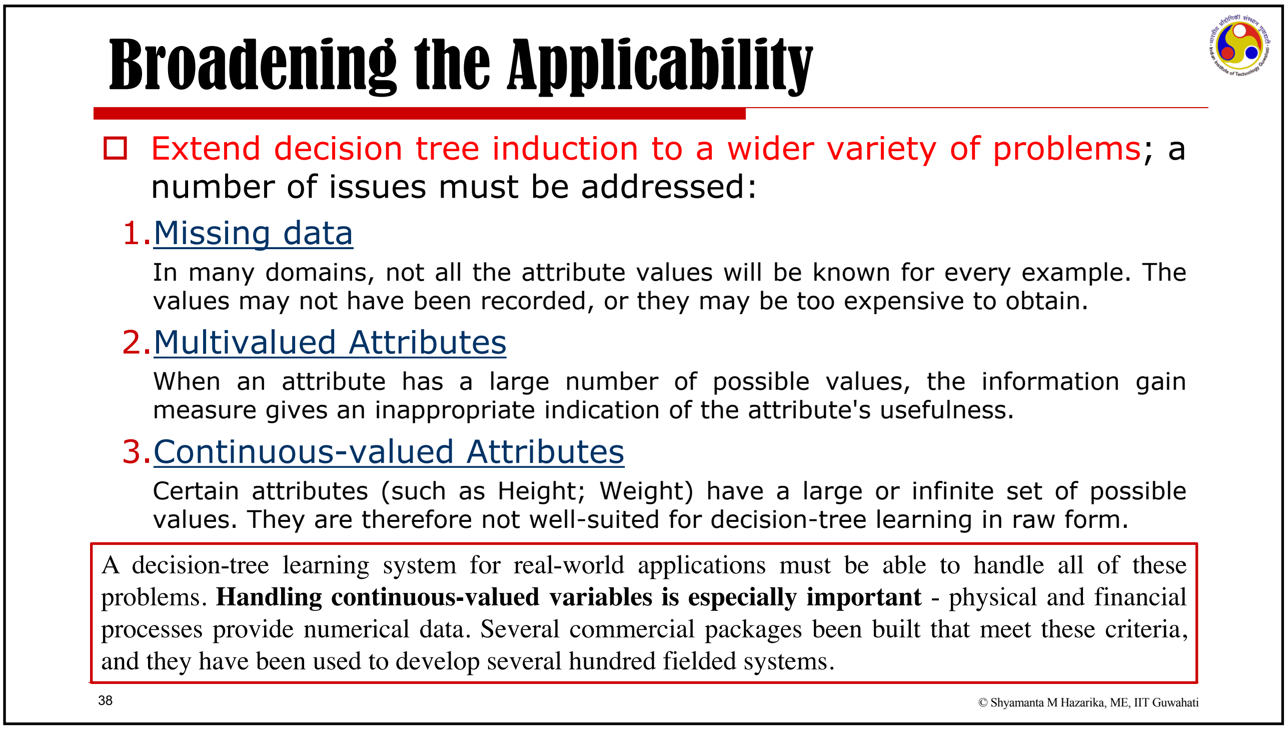
Noise and Overfitting

- 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



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 estimates the errors at each node for pruning.
    - □ A validation set may be used for pruning as well.



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 been built that meet these criteria, and they have been used to develop several hundred fielded systems.