COMP3411/9814: Artificial Intelligence

Learning and Decision Trees

Lecture Overview

- Learning agents
- Inductive learning
- Decision tree learning

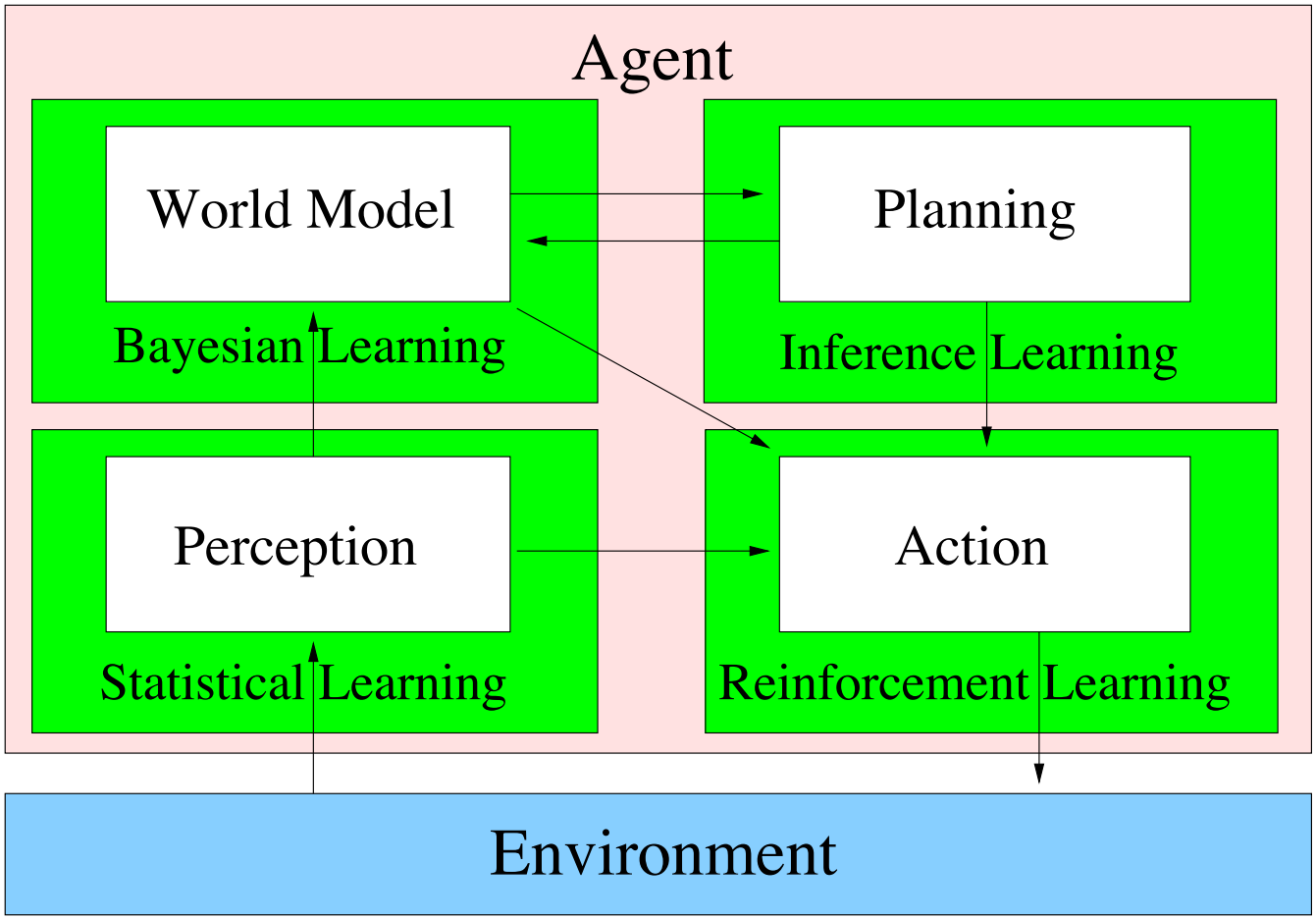
Motivation

- Reactive and Model-Based Agents choose their actions based only on what they currently perceive, or have perceived in the past.
- A Planning Agent can use Search techniques to plan several steps ahead in order to achieve its goal(s).
- Two classes of search strategies:
  - Uninformed search strategies can only distinguish goal states from non-goal states
  - Informed search strategies use heuristics to try to get "closer" to the goal

Learning

- Learning is essential for unknown environments,
  - i.e., when designer lacks omniscience
- Learning is useful as a system construction method,
  - i.e., expose the agent to reality rather than trying to write it down
- Learning modifies the agent's decision mechanisms to improve performance

Learning Agent



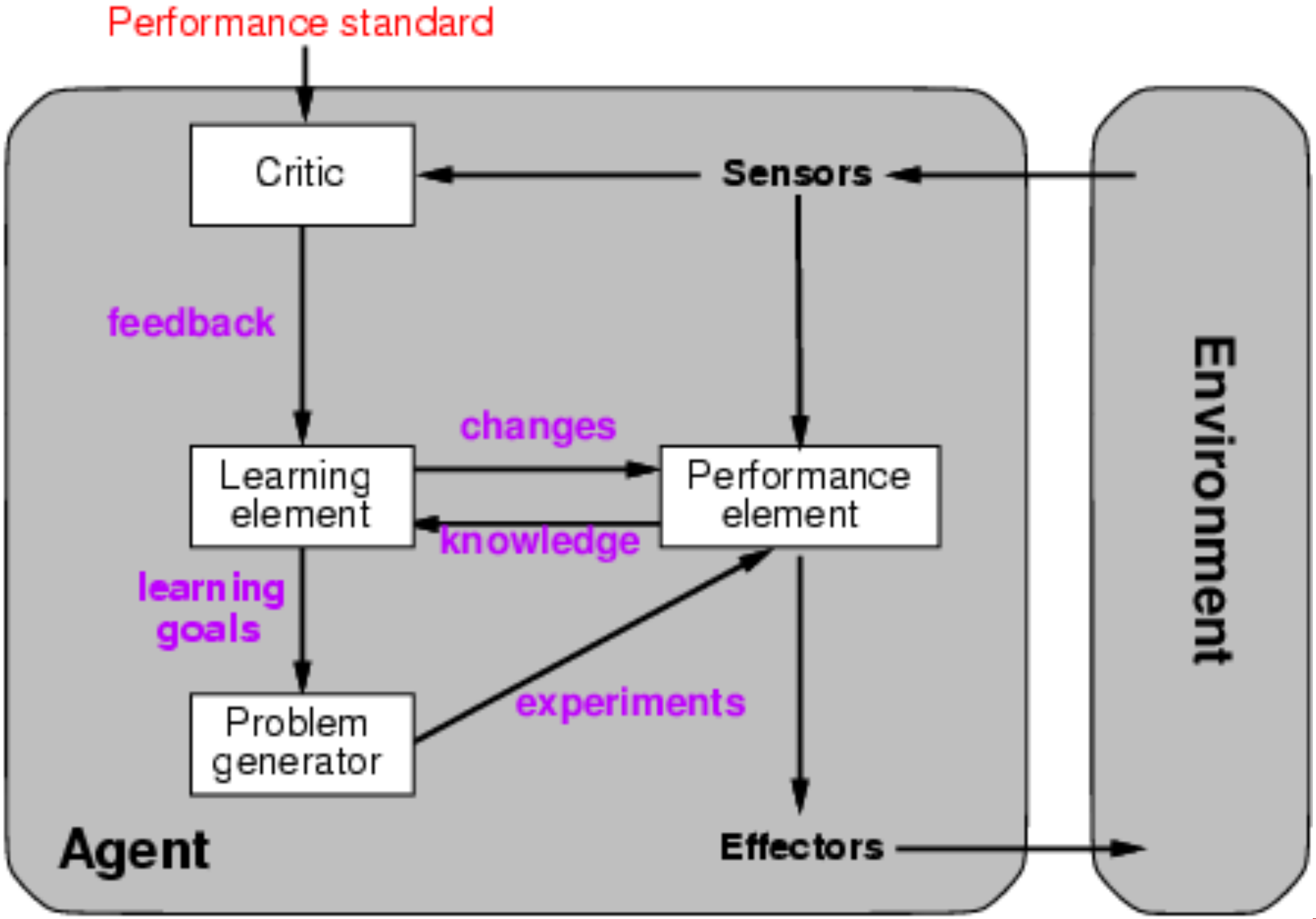
Learning agents

The Aim is to improve its performance on future tasks after making observations about the world.

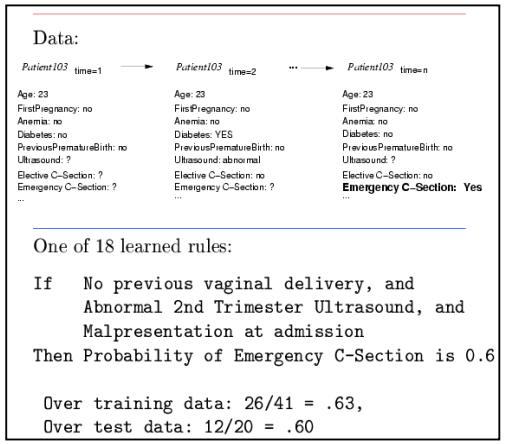
Learning element

- Design of a learning element is affected by
  - Which components of the performance element are to be learned
  - What feedback is available to learn these components
  - What representation is used for the components

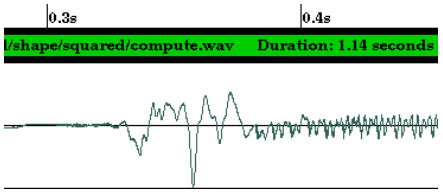
Learning agents



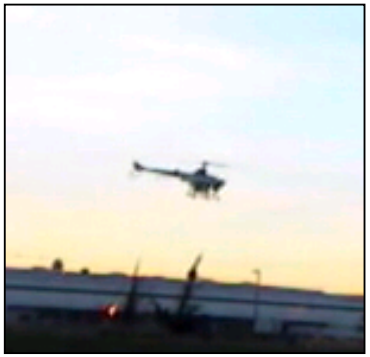
Machine Learning – Practice



Mining Databases



Speech Recognition



Control learning

Object recognition

Text analysis

Peter H. van Oppen , Chairman of the Board & Chief Executive Officer
Mr. van Oppen has served as chairman of the board and chief executive officer of ADIC since its acquisition by Interpoint in 1994 and a director of ADIC since 1986. Until its acquisition by Crane Co. in October 1996, Mr. van Oppen served as chairman of the board of directors, president and chief executive officer of Interpoint . Prior to 1985, Mr. van Oppen worked as a consulting manager at Price Waterhouse LLP and at Bain & Company in Boston and London. He has additional experience in medical electronics and venture capital. Mr. van Oppen also serves as a director of Seattle FilmWorks Inc. and Spacelabs Medical, Inc.. He holds a B.A. from Whitman College and an M.B.A. from Harvard Business School, where he was a Baker Scholar.

- Supervised learning
- Bayesian networks
- Hidden Markov models
- Unsupervised clustering
- Reinforcement learning

Types of Learning

- Supervised Learning
  - Agent is presented with examples of inputs and their target outputs, and must learn a function from inputs to outputs that agrees with the training examples and generalizes to new examples
- Reinforcement Learning
  - Agent is not presented with target outputs for each input, but is periodically given a reward, and must learn to maximize (expected) rewards over time
- Unsupervised Learning
  - Agent is only presented with a series of inputs, and must find and aims to find structure in these inputs

Supervised Learning

- We have a training set and a test set, each consisting of a set of items; for each item, a number of input attributes and a target value are specified.
- The aim is to predict the target value, based on the input attributes.
- Agent Is presented with the input and target output for each item in the training set; it must then predict the output for each item in the test set
- Various learning paradigms are available:
  - Decision Tree
  - Neural Network
  - Support Vector Machine, etc.

Supervised Learning

- Given a training set and a test set, each consisting of a set of items for each item in the training set, a set of features and a target output
- Learner must learn a model that can predict the target output for any given item (characterized by its set of features)
- Learner is given the input features and target output for each item in the training set
  - Items may be presented all at once (batch) or in sequence (online)
  - Items may be presented at random or in time order (stream)
  - Learner **cannot** use the test set **at all** in defining the model
- Model is evaluated by its performance on predicting the output for each item in the test set

Methods vs Models

- Various learning methods can be used to generate models
  - Decision Trees
  - Support Vector Machines
  - Neural Networks/Deep Learning
- Evaluate methods by evaluating models on a variety of datasets
  - Problem with availability of standard benchmark datasets
  - Models depend on problem formulation and on parameters
  - End users may only care about a model, not a general method
  - Most machine learning research evaluates methods, not models

Supervised Learning – Methodology

- Feature "engineering" – select relevant features
- Choose representation of input features and outputs
- Pre-processing method to extract features from raw data
- Choose learning method(s) to evaluate
- Choose training regime (including parameters)
- Evaluation
  - Choose realistic baseline for comparison
  - Choose type of internal validation, e.g. cross-validation
  - Sanity check results with human expertise, other benchmark

Supervised Learning – Issues

- Framework (decision tree, neural network, SVM, etc.)
  - representation (of inputs and outputs)
  - pre-processing / post-processing
  - training method (perceptron learning, backpropagation, etc.)
  - generalization (avoid over-fitting)
  - evaluation (separate training and testing sets)

Inductive learning

Simplest form: learn a function from examples

f is the target function
An example is a pair (*x*, *f(x)*)

Problem: find a hypothesis *h*
such that $h \approx f$
given a training set of examples

Inductive learning

Simplest form: learn a function from examples

f is the target function
An example is a pair (*x*, *f(x)*)

Problem: find a hypothesis *h*
such that $h \approx f$
given a training set of examples

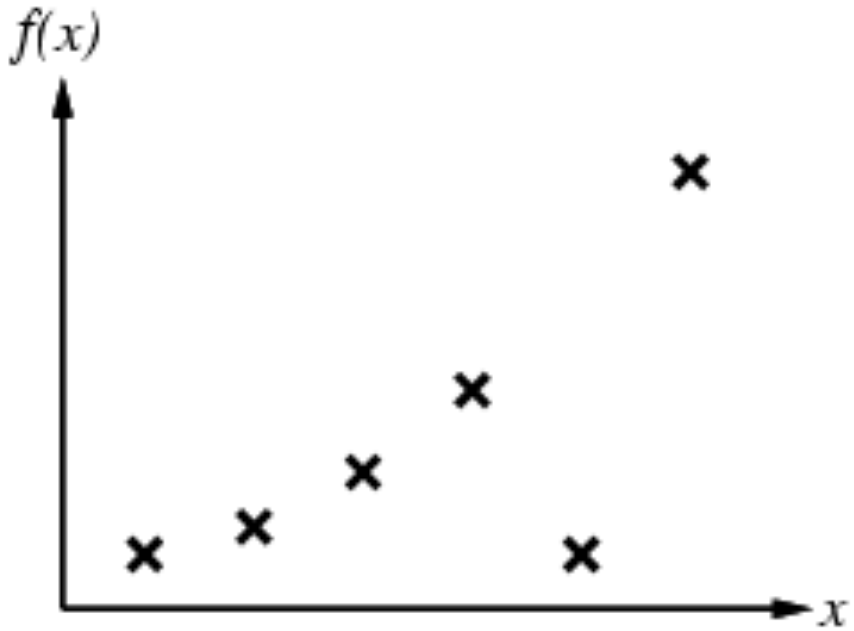
(This is a highly simplified model of real learning:
- Ignores prior knowledge
- Assumes examples are given)

Inductive learning method

Construct/adjust *h* to agree with *f* on training set

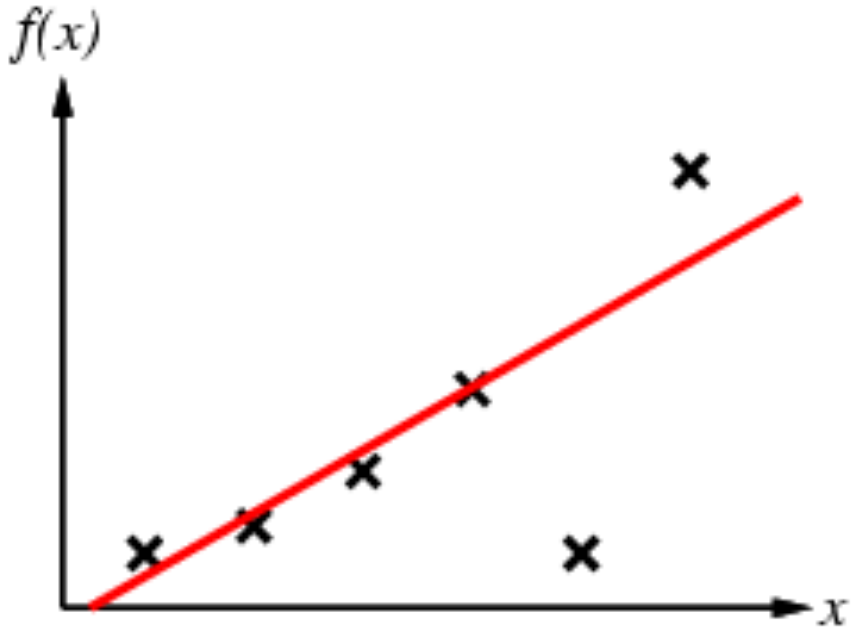
(*h* is consistent if it agrees with *f* on all examples)

- E.g., curve fitting:



Inductive learning method – Curve Fitting

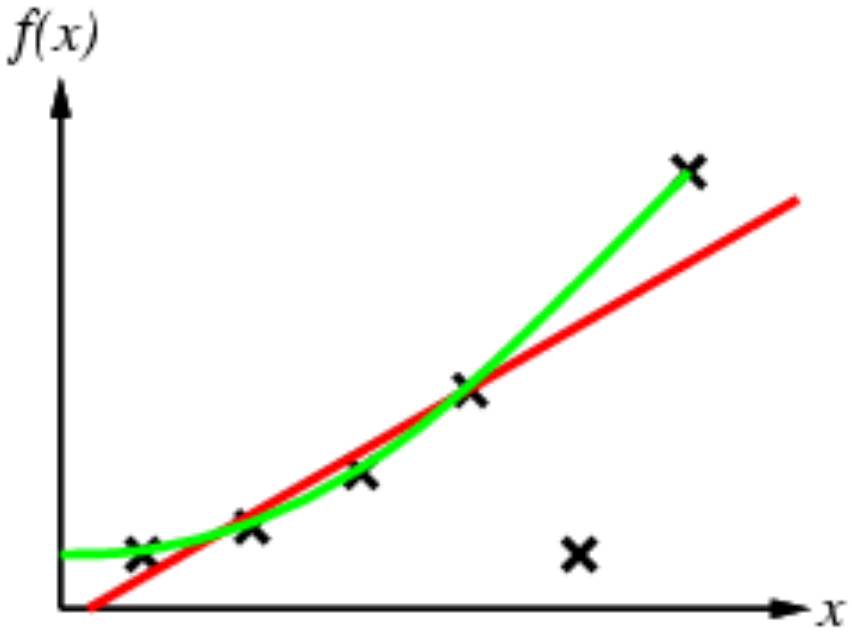
Which curve gives the “best fit” to these data?



straight line?

Curve Fitting

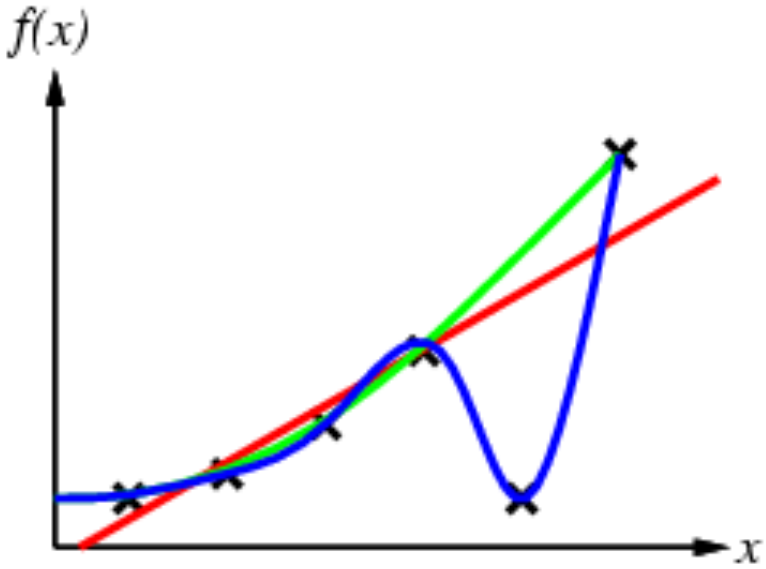
Which curve gives the “best fit” to these data?



parabola?

Inductive learning method

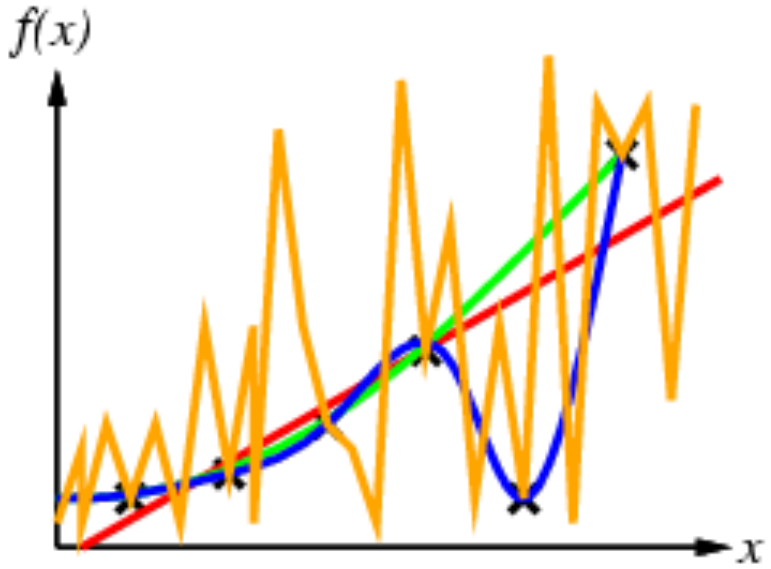
Which curve gives the “best fit” to these data?



4th order polynomial?

Inductive learning method

Which curve gives the “best fit” to these data?



Something else?

Ockham's razor

"The most likely hypothesis is the simplest one consistent with the data."

Ockham's razor

"The most likely hypothesis is the simplest one consistent with the data."

- Occam's razor (sometimes spelled Ockham's razor) is a principle attributed to the 14th- century English logician and Franciscan friar William of Ockham.

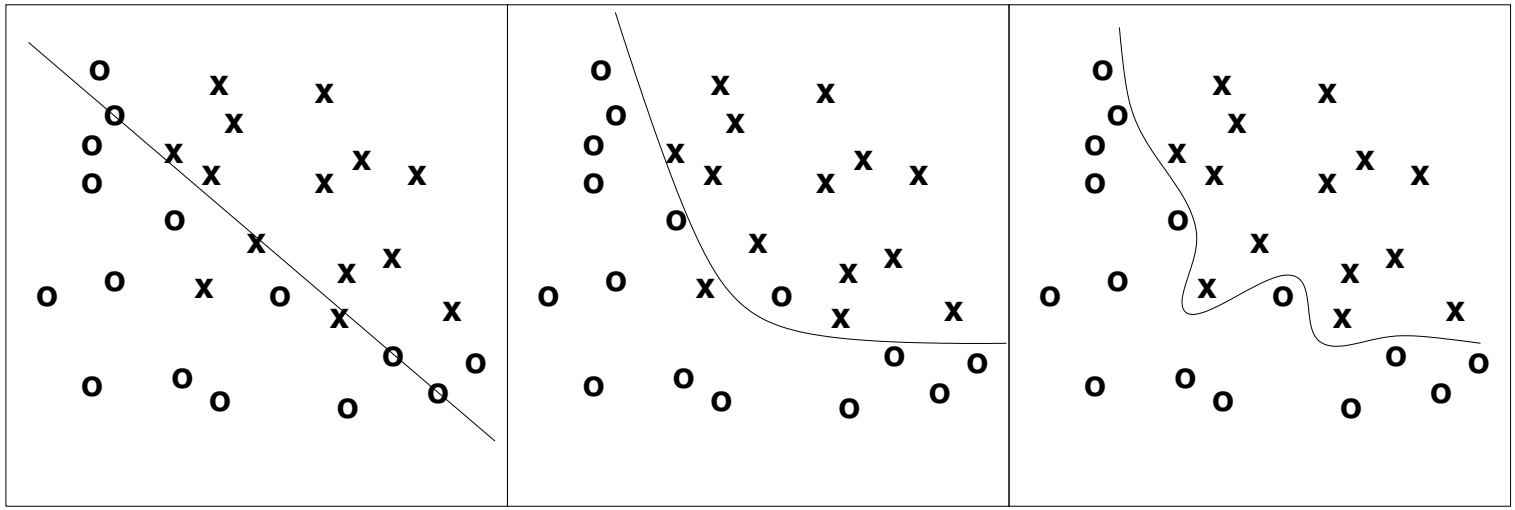
The principle states that the explanation of any phenomenon should make as few assumptions as possible, eliminating those that make no difference in the observable predictions of the explanatory hypothesis or theory.

This is often paraphrased as "All other things being equal, the simplest solution is the best."

Prefer the simplest hypothesis that fits the data

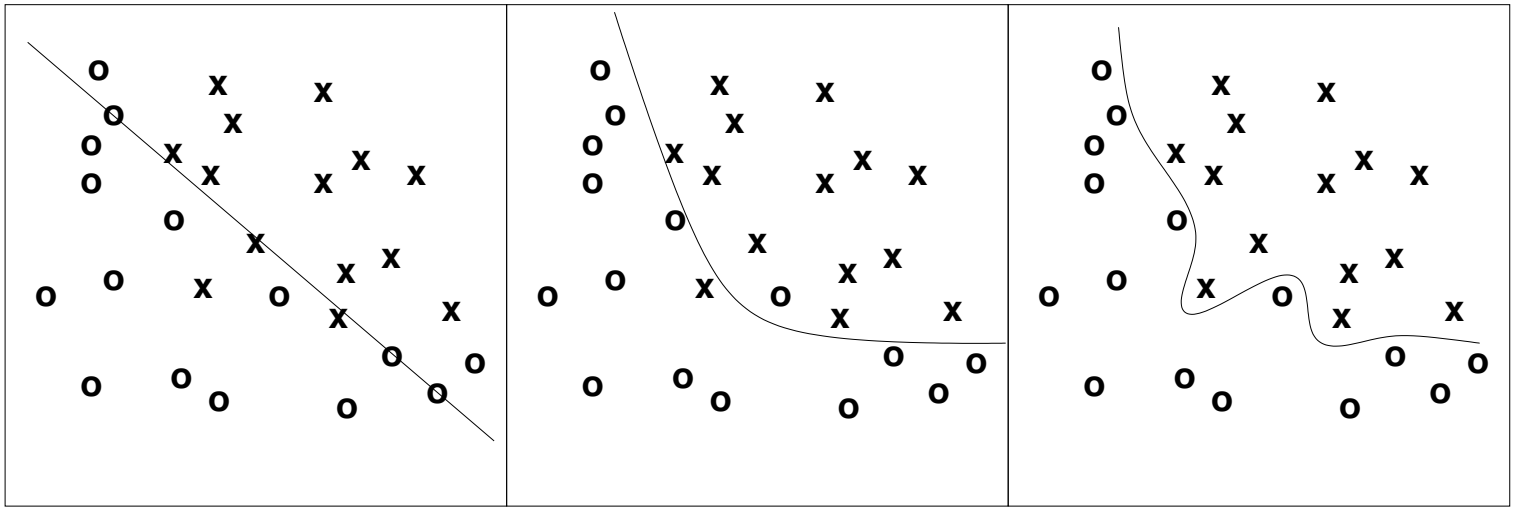
Ockham's razor

"The most likely hypothesis is the simplest one consistent with the data."



Ockham's razor

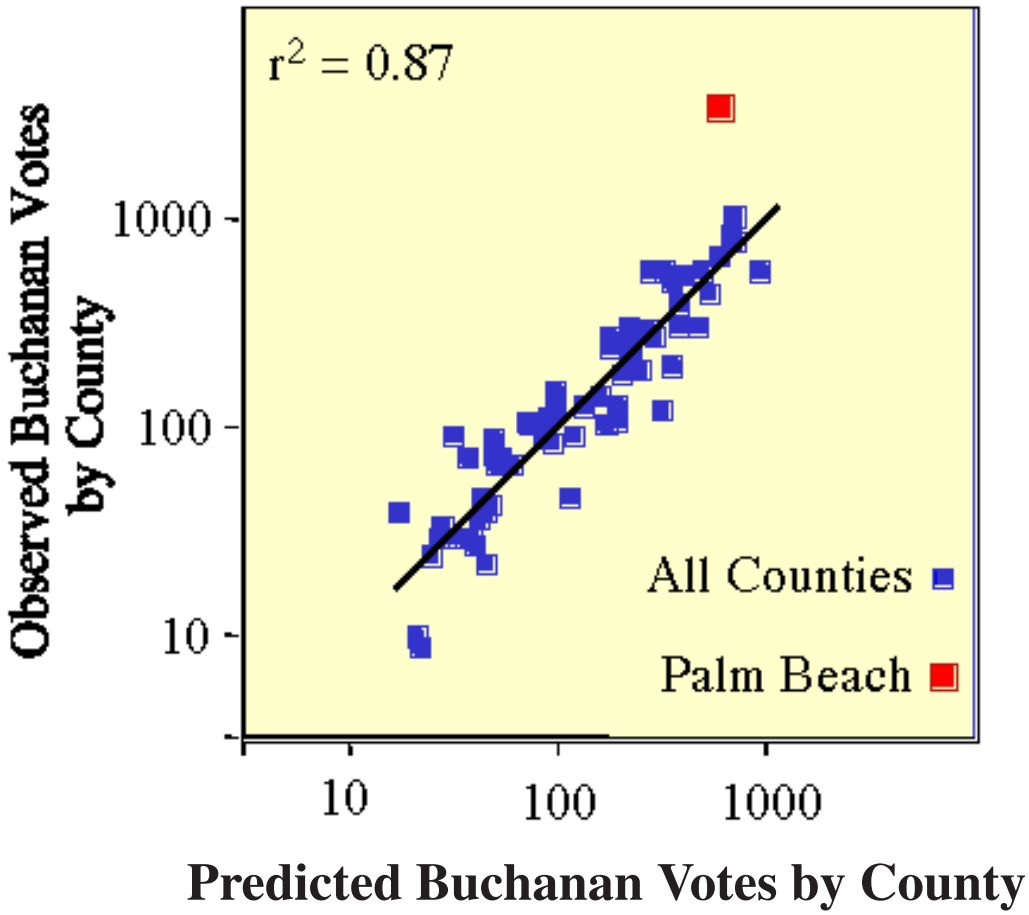
"The most likely hypothesis is the simplest one consistent with the data."



inadequate good compromise over-fitting

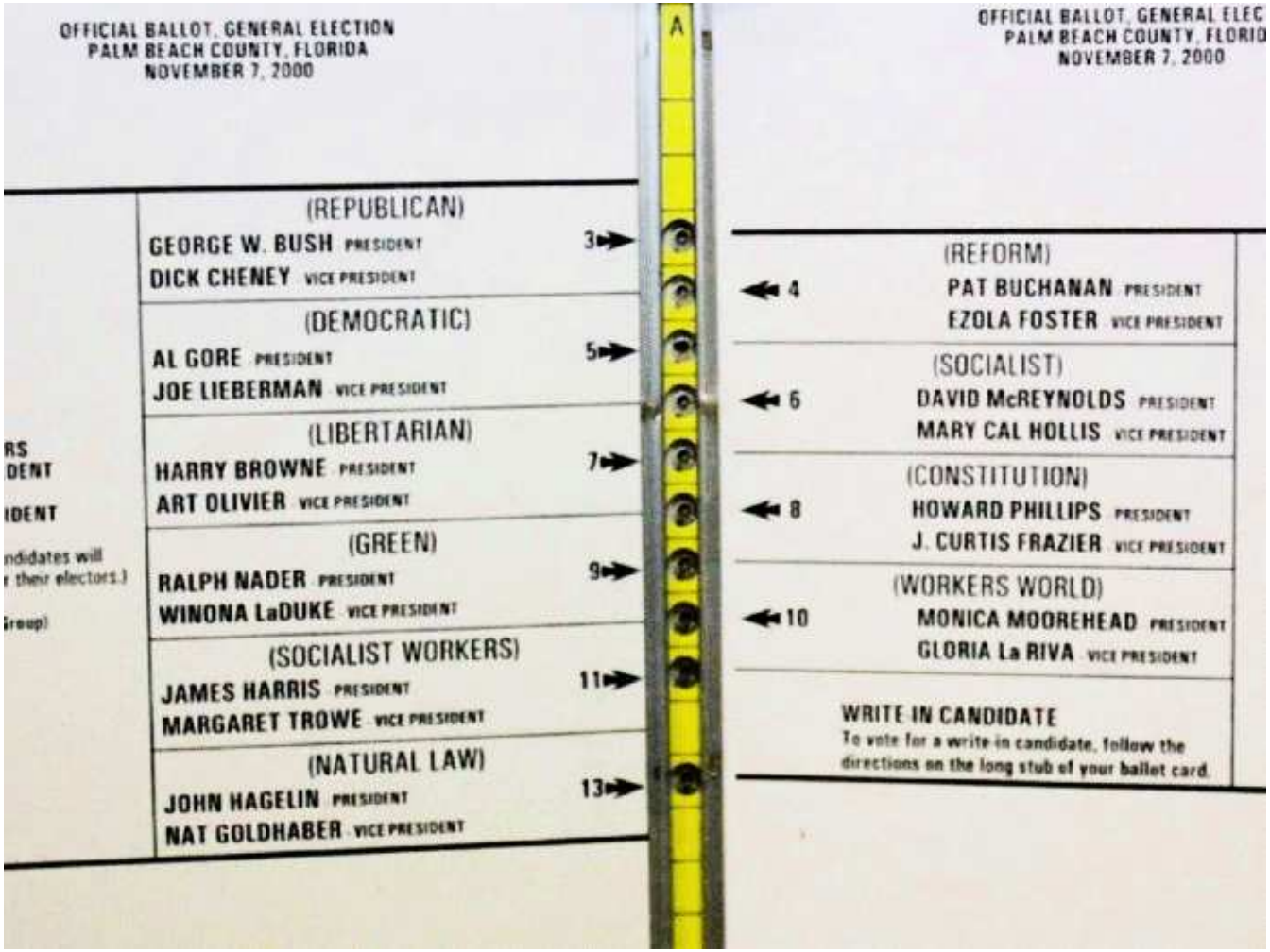
Since there can be noise in the measurements, in practice need to make a tradeoff between simplicity of the hypothesis and how well it fits the data.

Outliers



[faculty.washington.edu/mtbrett]

Butterfly Ballot



[faculty.washington.edu/mtbrett]

Learning decision trees

Problem: decide whether to wait for a table at a restaurant, based on the following attributes:

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)

Restaurant Training Data

	Alt	Bar	F/S	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

Attribute-based representations

- Examples described by attribute values (Boolean, discrete, continuous)
  - E.g., situations where I will/won't wait for a table:

	Alt	Bar	F/S	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

Attribute-based representations

- Examples described by attribute values (Boolean, discrete, continuous)
  - E.g., situations where I will/won't wait for a table:

	Alt	Bar	F/S	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

Attribute-based representations

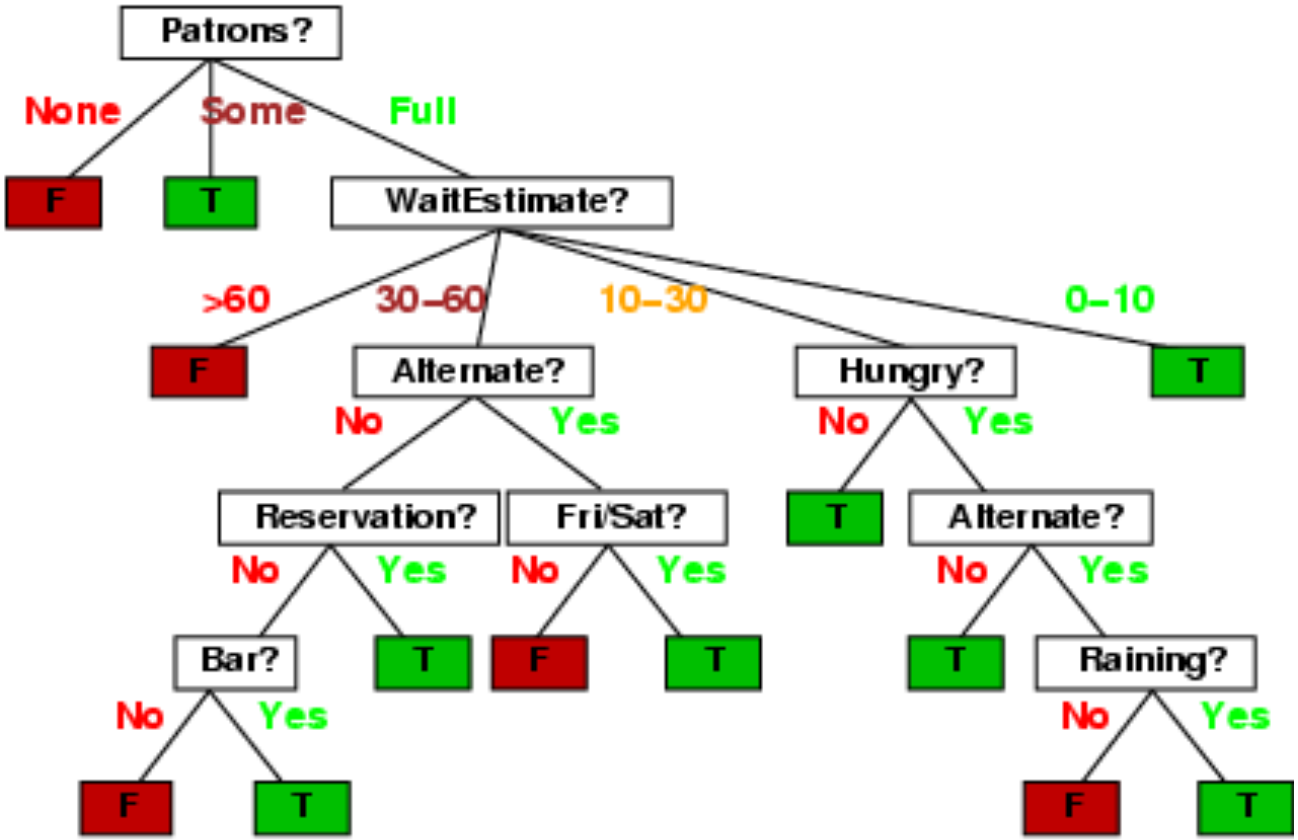
- Examples described by attribute values (Boolean, discrete, continuous)
  - E.g., situations where I will/won't wait for a table:

	Alt	Bar	F/S	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

- Classification of examples is positive (T) or negative (F)

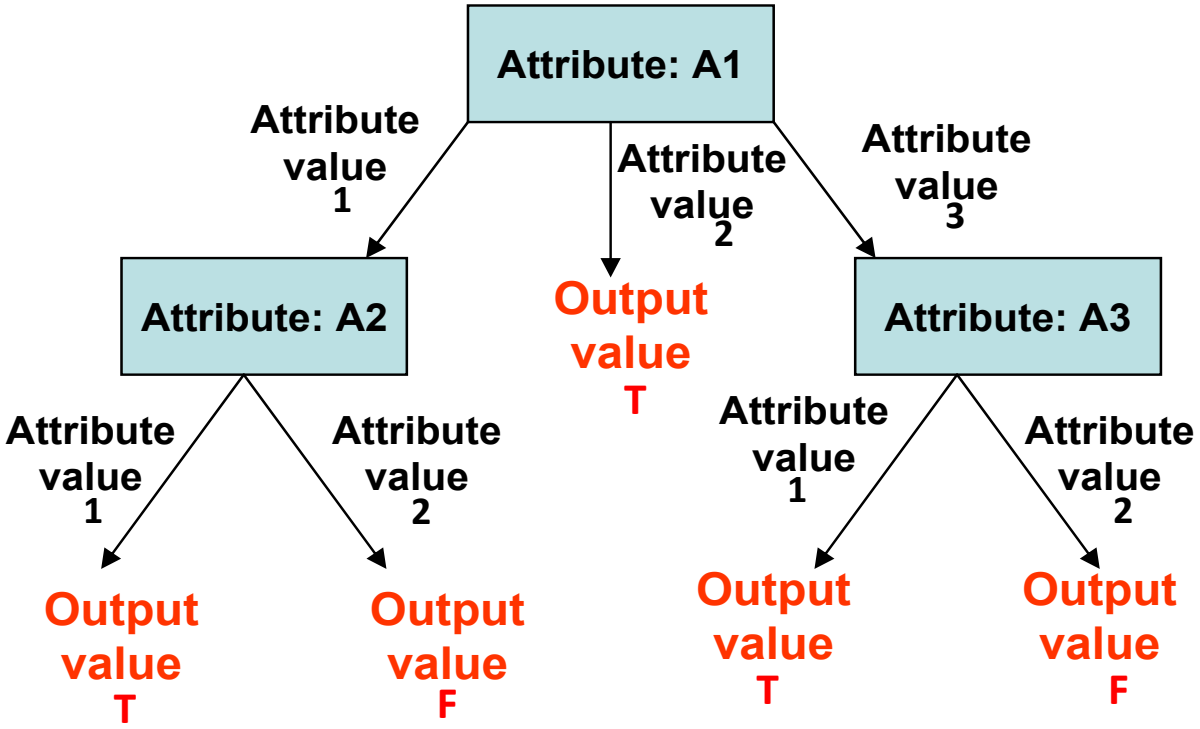
Decision trees

- One possible representation for hypotheses
  - E.g., here is the "true" tree for deciding whether to wait:



Decision trees - general

- One possible representation for hypotheses



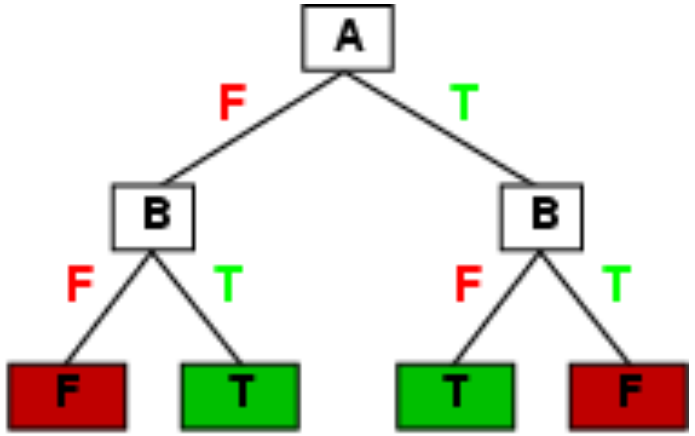
Generalization

- Provided the training data are not inconsistent, we can split the attributes in any order and still produce a tree that correctly classifies all examples in the training set.
- However, we really want a tree which is likely to generalize to correctly classify the (unseen) examples in the test set.
- In view of Ochkam's Razor, we prefer a simpler hypothesis, i.e. a smaller tree.
- But how can we choose attributes in order to produce a small tree?

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

- Prefer to find more compact decision trees

Hypothesis spaces

How many distinct decision trees with *n* Boolean attributes?

= number of Boolean functions

= number of distinct truth tables with 2^n rows = 2^{2^n}

- E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 trees

Decision tree learning

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

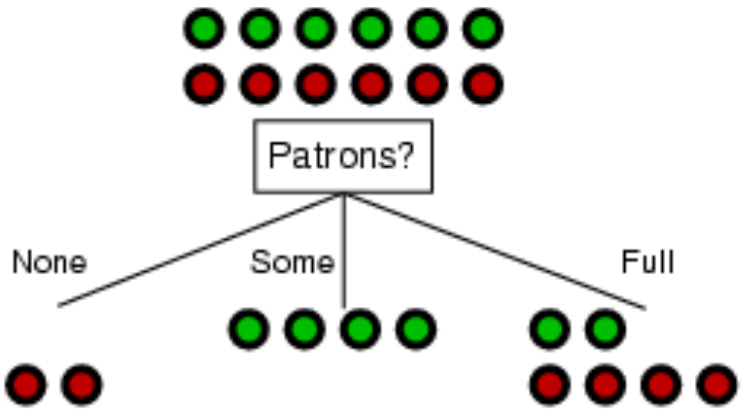
```
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes,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

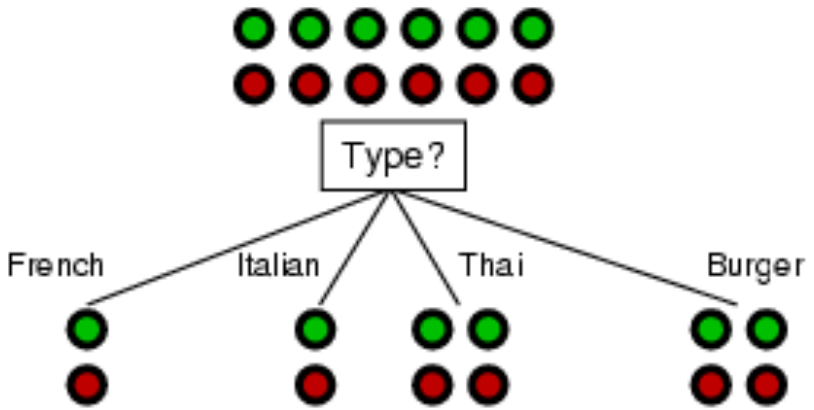
Choosing an attribute

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

Choosing an attribute

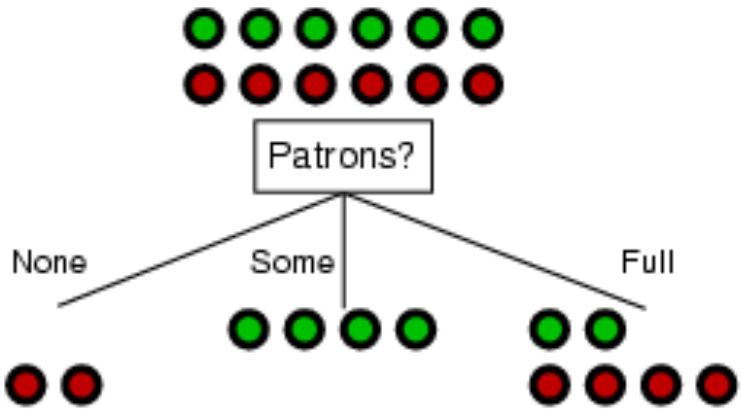
- Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"

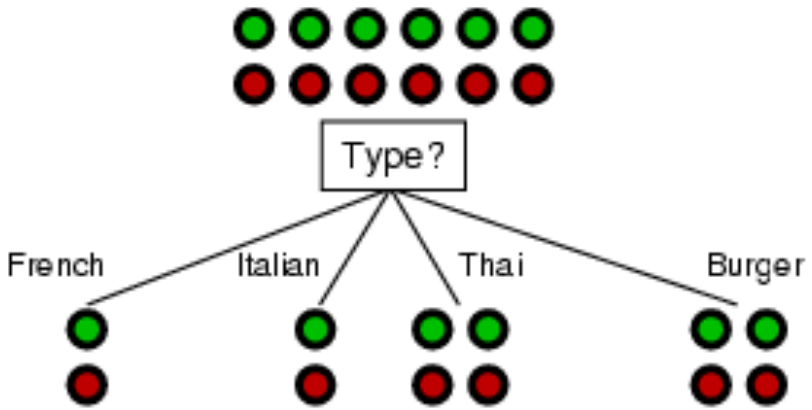




Choosing an attribute

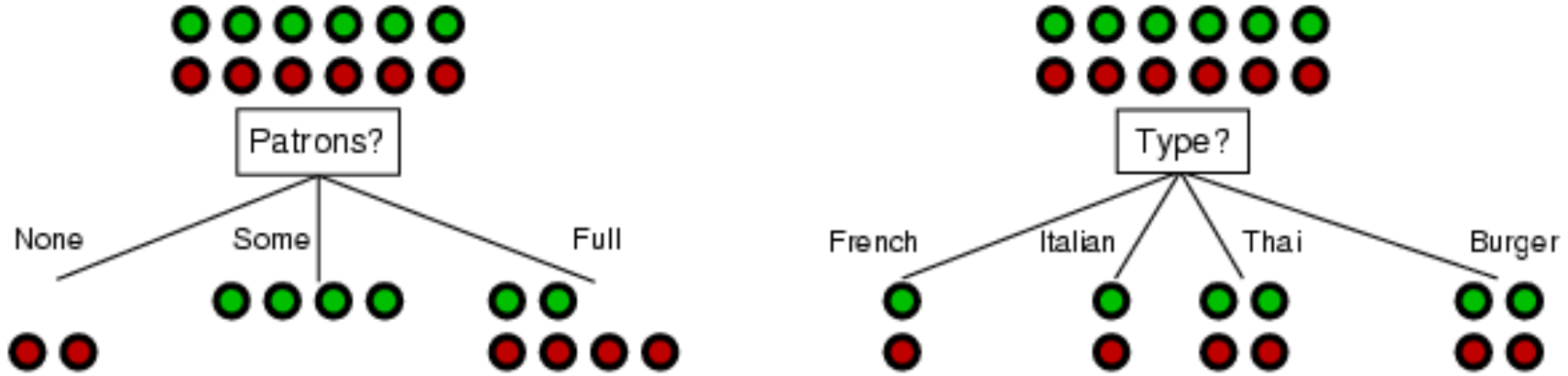
- Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"





- *Patrons?* is a better choice

Choosing an attribute



- *Patrons?* is a better choice
- Patrons is a "more informative" attribute than Type, because it splits the examples more nearly into sets that are "all positive" or "all negative".
- This notion of "informativeness" can be quantified using the mathematical concept of "entropy".
- A parsimonious tree can be built by minimizing the entropy at each step

Entropy

Entropy is a measure of how much information we gain when the target attribute is revealed to us. In other words, it is not a measure of how much we know, but of how much we don't know.

If the prior probabilities of the *n* target attribute values are p_1 , . . . , p_n then the entropy is

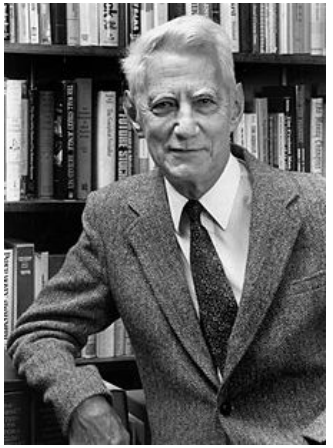
$$H(\langle p_1, \ldots, p_n \rangle) = \sum_{i=1} -p_i \log_2 p_i$$

Information Gain

Information gain is based on information theory concept called ***Entropy***

In information theory, the **Shannon entropy or information entropy** is a measure of the **uncertainty** associated with a random variable.

- It quantifies the information contained in a message, usually in bits or bits/symbol.
- It is the minimum message length necessary to communicate information.



Claude Elwood Shannon (April 30, 1916 – February 24, 2001), an American electrical engineer and mathematician, has been called "the father of information theory"

Entropy and Huffmann Coding

Entropy is the number of bits per symbol achieved by a (block) Huffman Coding scheme.

Example 1: $H(\langle 0.5, 0.5 \rangle) = 1$ bit.

Suppose we want to encode, in zeroes and ones, a long message composed of the two letters A and B, which occur with equal frequency. This can be done efficiently by assigning A=0, B=1. In other words, one bit is needed to encode each letter.

Entropy and Huffmann Coding

Example 2: $H(\langle 0.5, 0.25, 0.25 \rangle) = 1.5$ bits.

Suppose we need to encode a message consisting of the letters A, B and C, and that B and C occur equally often but A occurs twice as often as the other two letters. In this case, the most efficient code would be A=0, B=10, C=11. The average number of bits needed to encode each letter is 1.5 .

If the letters occur in some other proportion, we would need to "block" them together in order to encode them efficiently. But, the average number of bits required by the most efficient coding scheme is given by

$$H(\langle p_1, \ldots, p_n \rangle) = \sum_{i=1}^{n} -p_i \log_2 p_i$$

Entropy

- Suppose we have *p* positive and *n* negative examples at a node.
  $\rightarrow H(\langle p/(p+n), n/(p+n)\rangle)$ bits needed to classify a new example.
  - e.g. for 12 restaurant examples, *p* = *n* = 6 so we need 1 bit.

- An attribute splits the examples *E* into subsets E_i, each of which (we hope) needs less information to complete the classification.

  Let E_i have p_i positive and n_i negative examples
  $\rightarrow H(\langle p_i/(p_i+n_i), n_i/(p_i+n_i)\rangle)$ s needed to classify a new example

  $\rightarrow$ expected number of bits per example over all branches is

$$\sum_i \frac{p_i+n_i}{p+n} H(\langle \frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}\rangle)$$

For Patrons, this is 0.459 bits, for Type this is (still) 1 bit.

Entropy

- Entropy is a measure of "randomness" (lack of uniformity)
  - Related to prior distribution of some random variable
  - Higher entropy means more randomness
  - "Information" (about distribution) reduces entropy
- Idea: Split based on information gain
  - Loss of entropy based on "communicating" value of attribute
  - Related to Shannon's information theory
  - Measure information gain in bits

Definition. If the prior probabilities of *n* attribute values are *p*1,⋯,*pn*, then the entropy of the distribution is

$$H(\langle p_1, \ldots, p_n \rangle) = \sum_{i=1}^{n} -p_i \log_2 p_i$$

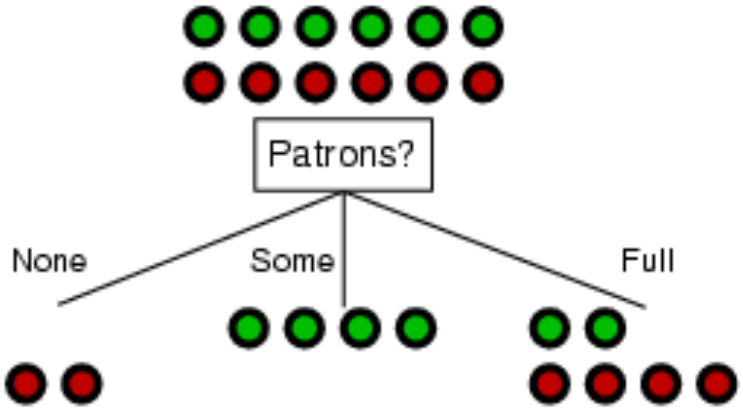
Entropy and Huffmann Coding W

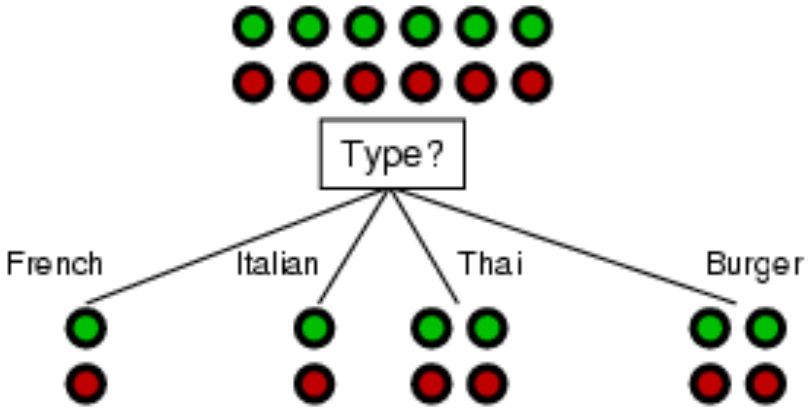
- Entropy is the number of bits per symbol achieved by a (block) Huffman Coding scheme

- **Example 1:**
  $H(\langle 0.5,0.5 \rangle) = 1$ bit

- To encode in 0s and 1s, a long message composed of the two letters A and
- B which occur with equal frequency, assign A=0 and B=1
- One bit (binary digit) is needed to encode each letter

Choosing and Attribute



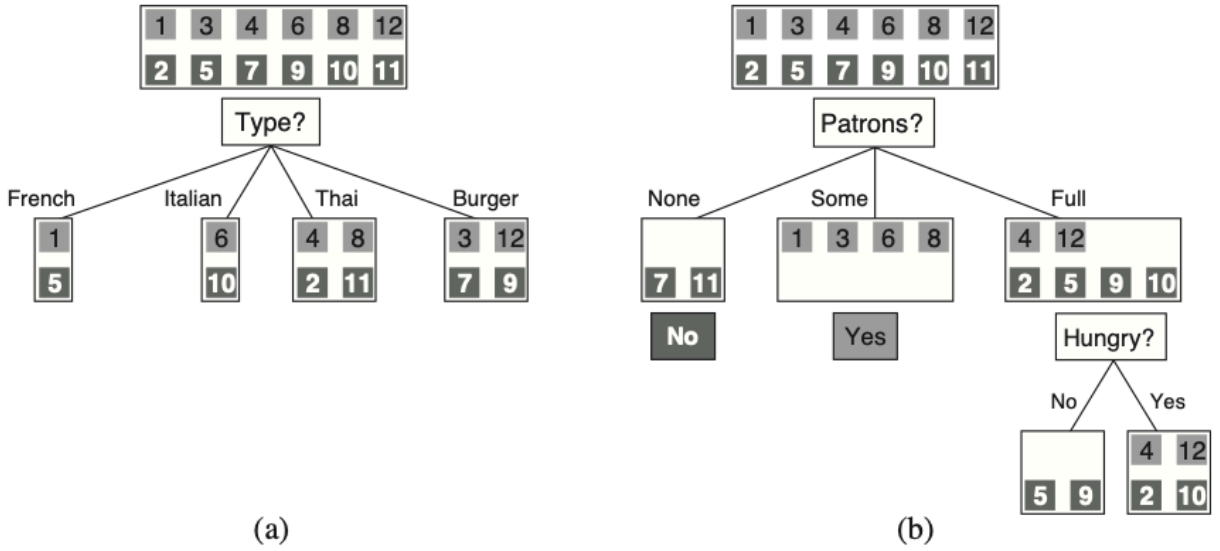


$$\text{For Patrons, Entropy} = \frac{1}{6}(0)+\frac{1}{3}(0)+\frac{1}{2}\left[-\frac{1}{3}\log(\frac{1}{3})-\frac{2}{3}\log(\frac{2}{3})\right]$$

$$= 0+0+\frac{1}{2}\left[\frac{1}{3}(1.585)+\frac{2}{3}(0.585)\right] = 0.459$$

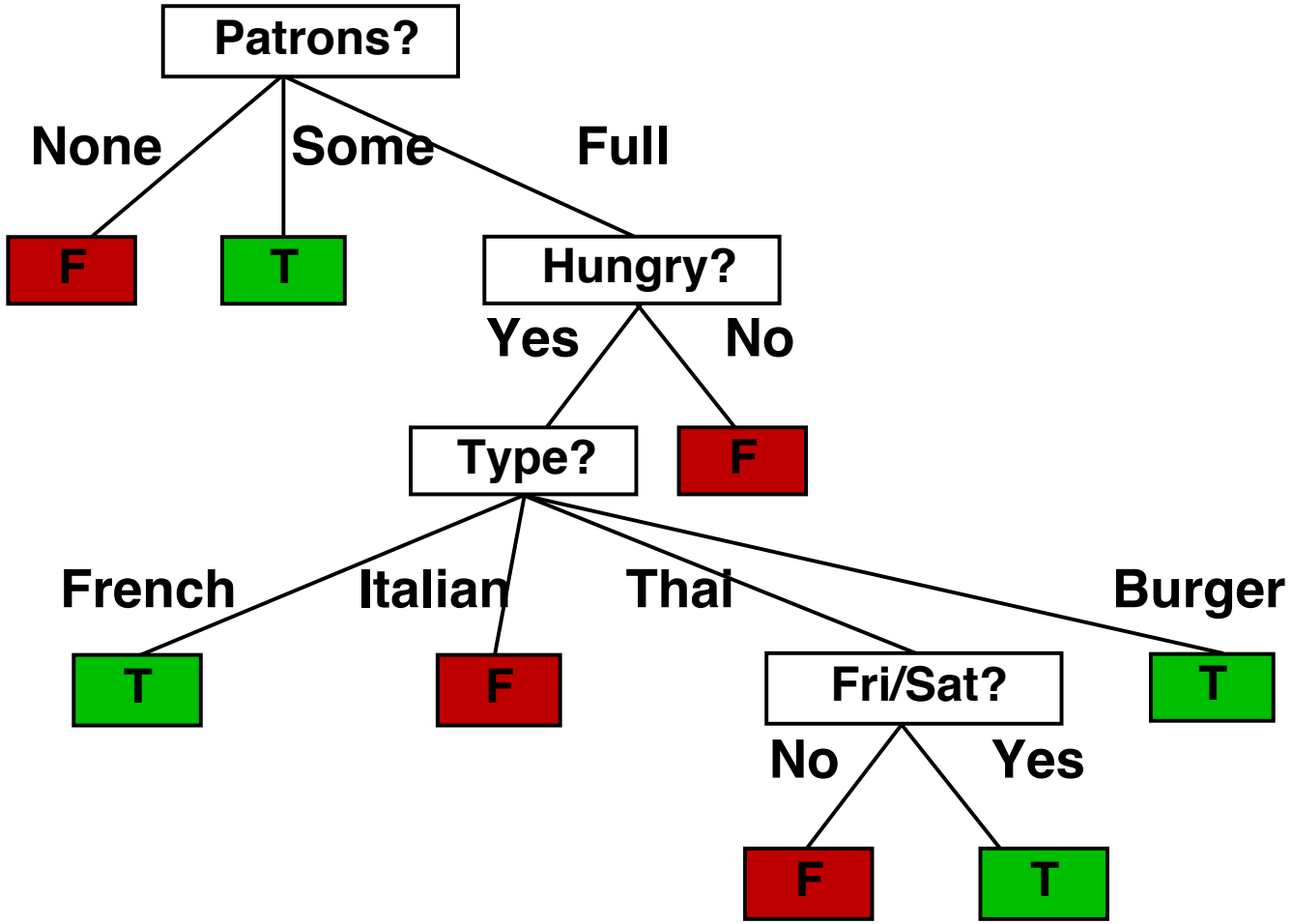
$$\text{For Type, Entropy} = \frac{1}{6}(1)+\frac{1}{6}(1)+\frac{1}{3}(1)+\frac{1}{3}(1) = 1$$

Choosing Next Attribute

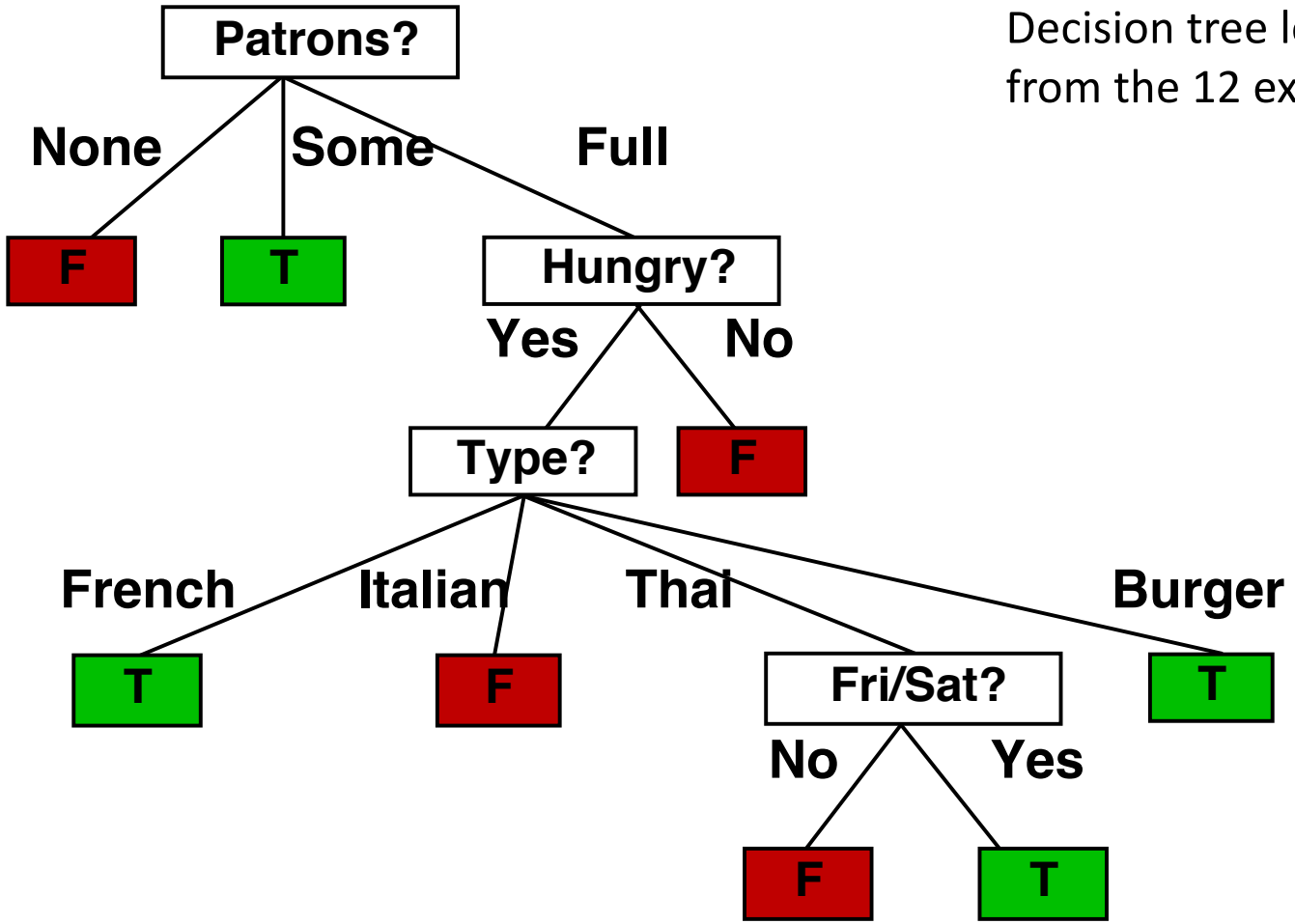


After splitting on Patrons, split on Hungry

Induced Tree



Induced Tree



Decision tree learned from the 12 examples.

Decision Trees

Decision tree learning is a method for approximating discrete value target functions, in which the learned function is represented by a decision tree.

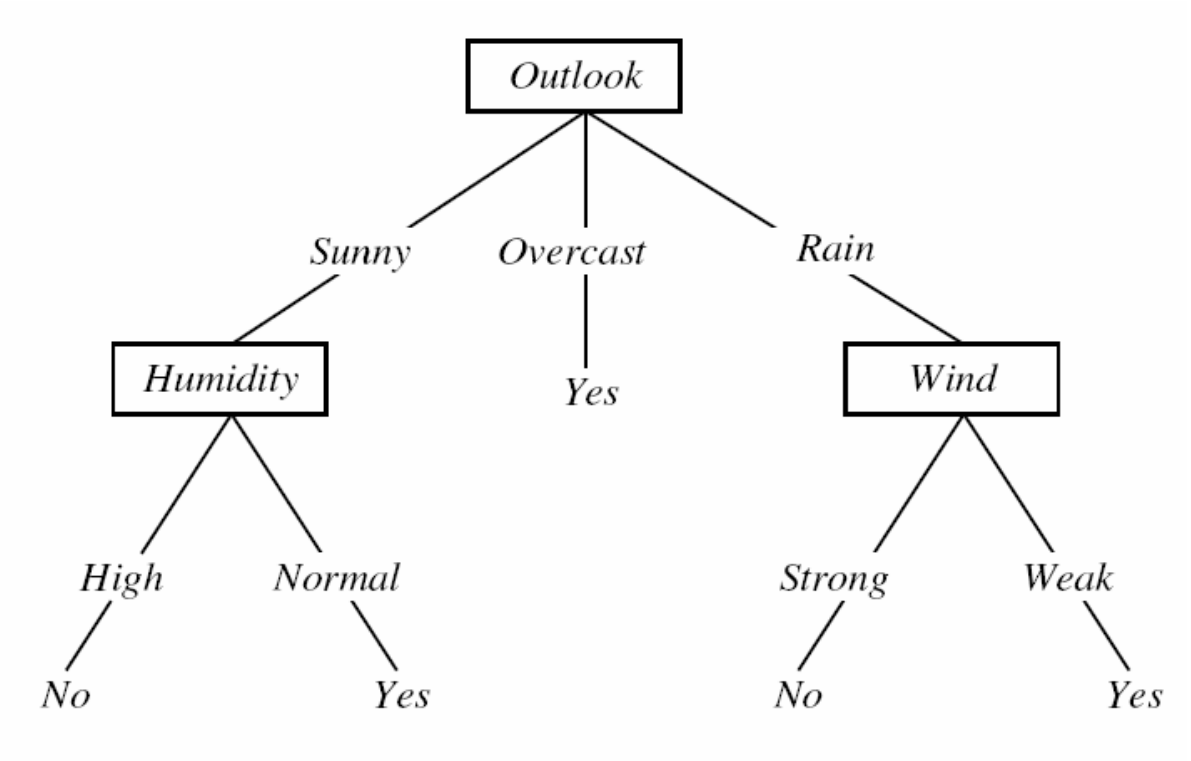
Decision trees can also be represented by if-then-else rule.

Decision tree learning is one of the most widely used approach for inductive inference .

Training Examples

Day	Outlook	Temperature	Humidity	Wind	PlayTenis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Decision tree



Using the tree

An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute in the given example. This process is then repeated for the subtree rooted at the new node.

Basic Decision Tree Learning Algorithm

- ID3 Algorithm (Quinlan 1986) and it's successors C4.5 and C5.0
- ***Employs a top-down induction***

[ID3, C4.5, Quinlan]

node = Root

Main loop:

1. $A \leftarrow$ the "best" decision attribute for next *node*
2. Assign A as decision attribute for *node*
3. For each value of A, create new descendant of *node*
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes

- Greedy search the space of possible decision trees.
- The algorithm never backtracks to reconsider earlier choices.

http://www.rulequest.com/Personal/