

## **Part I Artificial Intelligence**

1 Introduction

2 Intelligent Agents

## **Part II Problem Solving**

3 Solving Problems by Searching

4 Informed Search and Exploration

5 Constraint Satisfaction Problems

6 Adversarial Search

## **Part III Knowledge and Reasoning**

7 Logical Agents

8 First-Order Logic

9 Inference in First-Order Logic

## **Part V Uncertain Knowledge and Reasoning**

13 Uncertainty

14 Probabilistic Reasoning

## **Part V Learning**

18 Learning from Observations

## **Part VII Communicating, Perceiving, and Acting**

22 Communication

# LEARNING FROM OBSERVATIONS

## CHAPTER 18, SECTIONS 1–3

## Outline

- ◇ Learning agents
- ◇ Inductive learning
- ◇ Decision tree learning
- ◇ Measuring learning performance

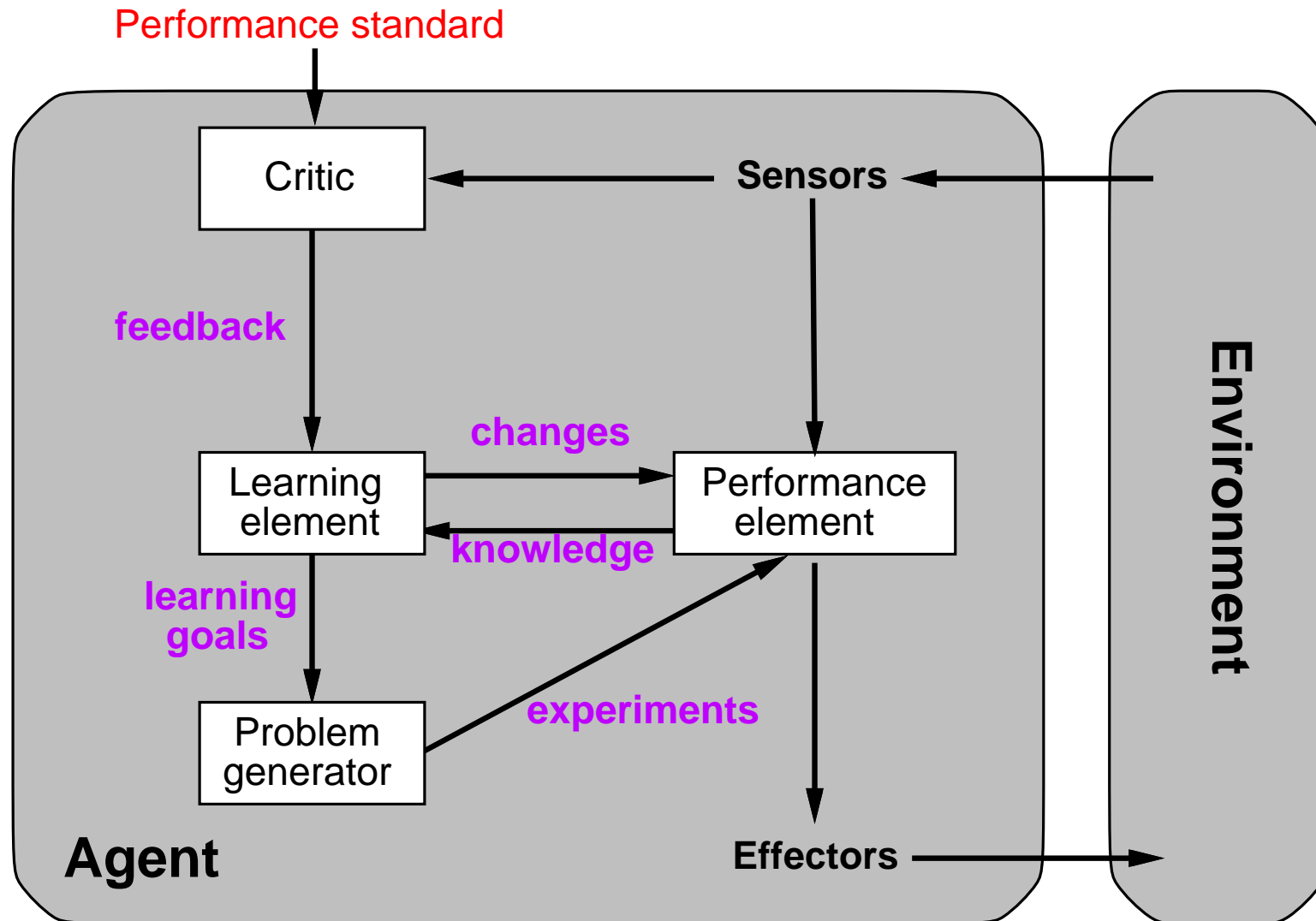
# Learning

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 agents



# Learning element

Design of learning element is dictated by

- ◇ what type of performance element is used
- ◇ which functional component is to be learned
- ◇ how that functional component is represented
- ◇ what kind of feedback is available

Example scenarios:

Performance element	Component	Representation	Feedback
Alpha–beta search	Eval. fn.	Weighted linear function	Win/loss
Logical agent	Transition model	Successor–state axioms	Outcome
Utility–based agent	Transition model	Dynamic Bayes net	Outcome
Simple reflex agent	Percept–action fn	Neural net	Correct action

Supervised learning: correct answers for each instance

Reinforcement learning: occasional rewards

# Inductive learning (a.k.a. Science)

Simplest form: learn a function from examples (**tabula rasa**)

$f$  is the target function

An example is a pair  $x, f(x)$ , e.g.,  $\begin{array}{|c|c|c|} \hline O & O & X \\ \hline & X & \\ \hline X & & \\ \hline \end{array}, +1$

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

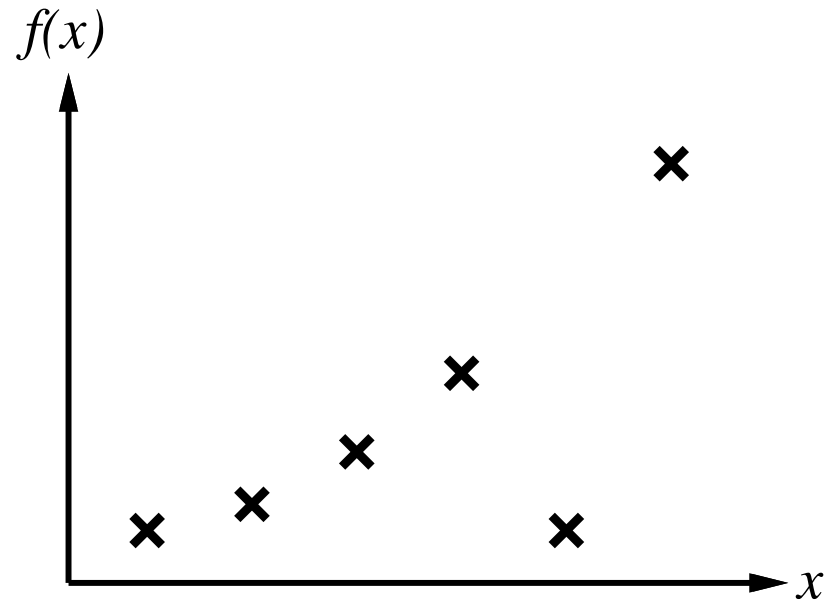
(This is a highly simplified model of real learning:

- Ignores prior knowledge
- Assumes a deterministic, observable “environment”
- Assumes examples are given
- Assumes that the agent wants to learn  $f$ —why?)

# 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:

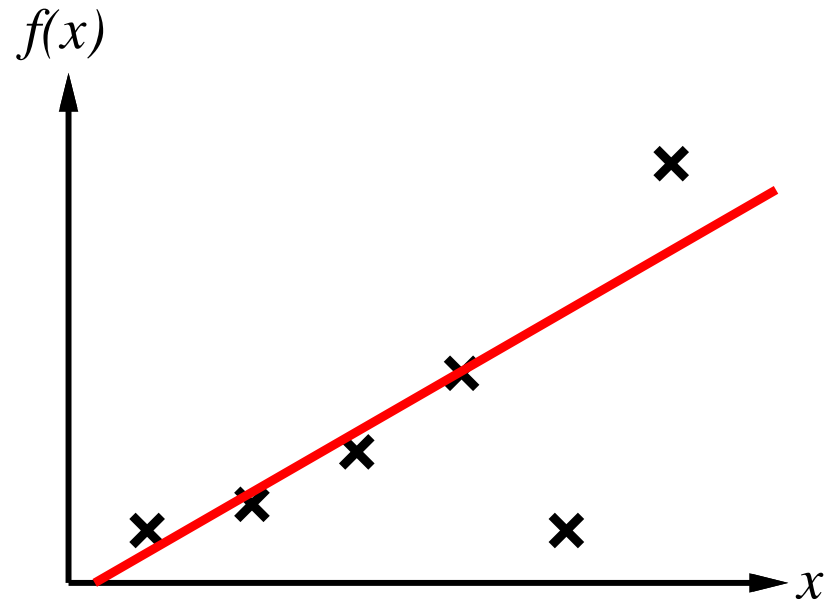




# Inductive learning method

Construct/adjust  $h$  to agree with  $f$  on training set  
( $h$  is consistent if it agrees with  $f$  on all examples)

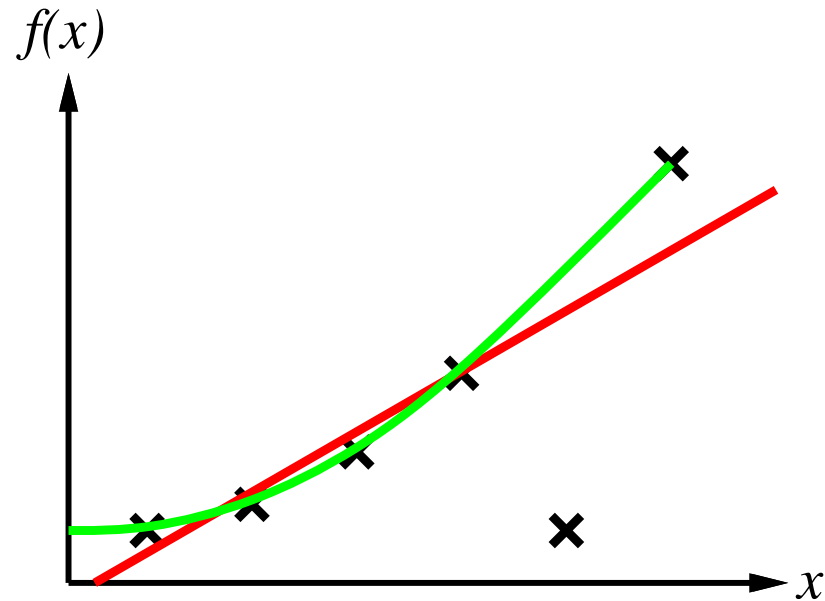
E.g., curve fitting:



# Inductive learning method

Construct/adjust  $h$  to agree with  $f$  on training set  
( $h$  is consistent if it agrees with  $f$  on all examples)

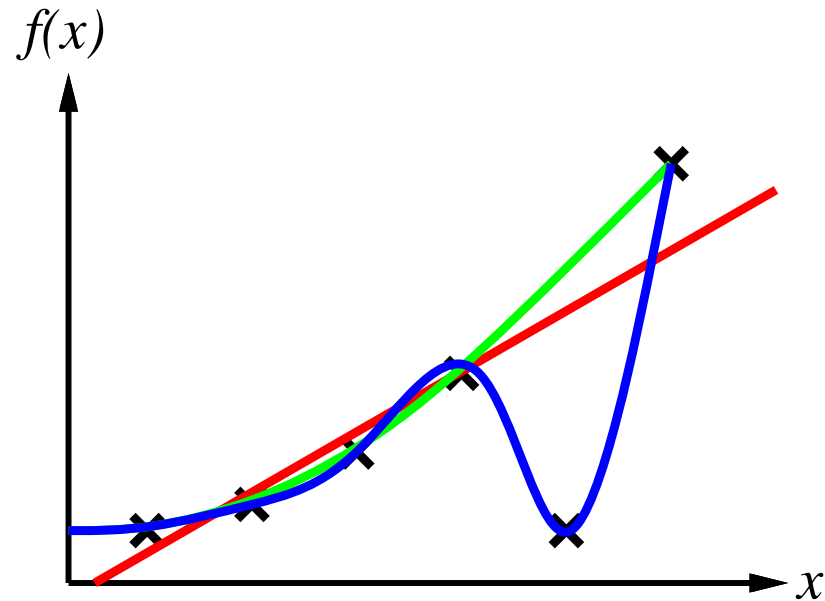
E.g., curve fitting:



# Inductive learning method

Construct/adjust  $h$  to agree with  $f$  on training set  
( $h$  is consistent if it agrees with  $f$  on all examples)

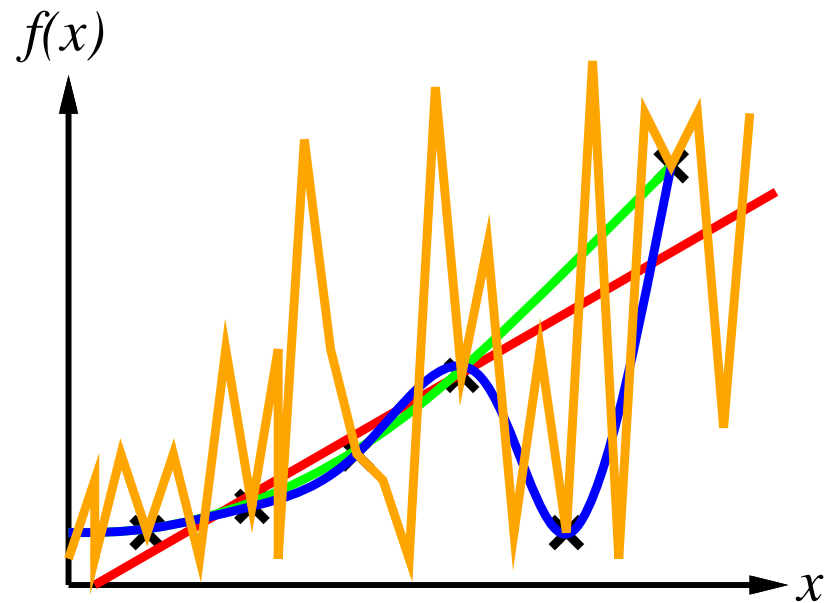
E.g., curve fitting:



# Inductive learning method

Construct/adjust  $h$  to agree with  $f$  on training set  
( $h$  is consistent if it agrees with  $f$  on all examples)

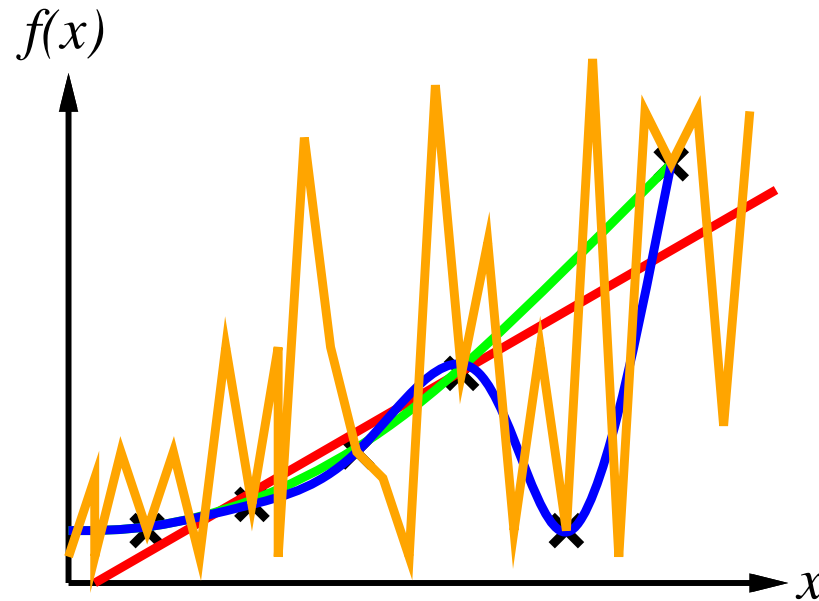
E.g., curve fitting:



# Inductive learning method

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:



Ockham's razor: maximize a combination of consistency and simplicity

## Attribute-based representations

Examples described by **attribute values** (Boolean, discrete, continuous, etc.)

E.g., situations where I will/won't wait for a table:

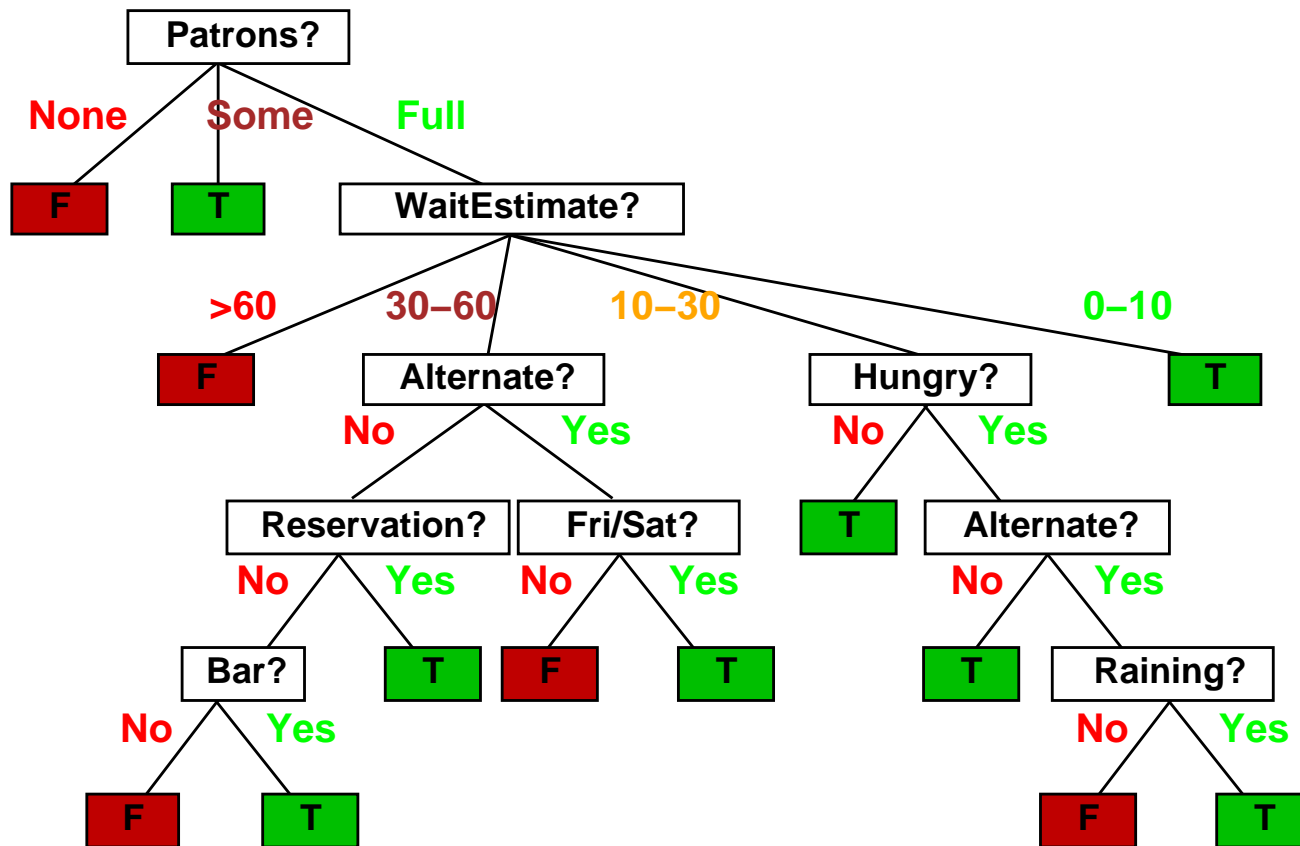
Example	Attributes										Target
	<i>Alt</i>	<i>Bar</i>	<i>Fri</i>	<i>Hun</i>	<i>Pat</i>	<i>Price</i>	<i>Rain</i>	<i>Res</i>	<i>Type</i>	<i>Est</i>	<i>WillWait</i>
$X_1$	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>Some</i>	<i>\$\$\$</i>	<i>F</i>	<i>T</i>	<i>French</i>	<i>0-10</i>	<i>T</i>
$X_2$	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>Full</i>	<i>\$</i>	<i>F</i>	<i>F</i>	<i>Thai</i>	<i>30-60</i>	<i>F</i>
$X_3$	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>Some</i>	<i>\$</i>	<i>F</i>	<i>F</i>	<i>Burger</i>	<i>0-10</i>	<i>T</i>
$X_4$	<i>T</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>Full</i>	<i>\$</i>	<i>F</i>	<i>F</i>	<i>Thai</i>	<i>10-30</i>	<i>T</i>
$X_5$	<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>Full</i>	<i>\$\$\$</i>	<i>F</i>	<i>T</i>	<i>French</i>	<i>&gt;60</i>	<i>F</i>
$X_6$	<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>Some</i>	<i>\$\$</i>	<i>T</i>	<i>T</i>	<i>Italian</i>	<i>0-10</i>	<i>T</i>
$X_7$	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>None</i>	<i>\$</i>	<i>T</i>	<i>F</i>	<i>Burger</i>	<i>0-10</i>	<i>F</i>
$X_8$	<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>Some</i>	<i>\$\$</i>	<i>T</i>	<i>T</i>	<i>Thai</i>	<i>0-10</i>	<i>T</i>
$X_9$	<i>F</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>Full</i>	<i>\$</i>	<i>T</i>	<i>F</i>	<i>Burger</i>	<i>&gt;60</i>	<i>F</i>
$X_{10}$	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>Full</i>	<i>\$\$\$</i>	<i>F</i>	<i>T</i>	<i>Italian</i>	<i>10-30</i>	<i>F</i>
$X_{11}$	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>None</i>	<i>\$</i>	<i>F</i>	<i>F</i>	<i>Thai</i>	<i>0-10</i>	<i>F</i>
$X_{12}$	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>Full</i>	<i>\$</i>	<i>F</i>	<i>F</i>	<i>Burger</i>	<i>30-60</i>	<i>T</i>

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

# Decision trees

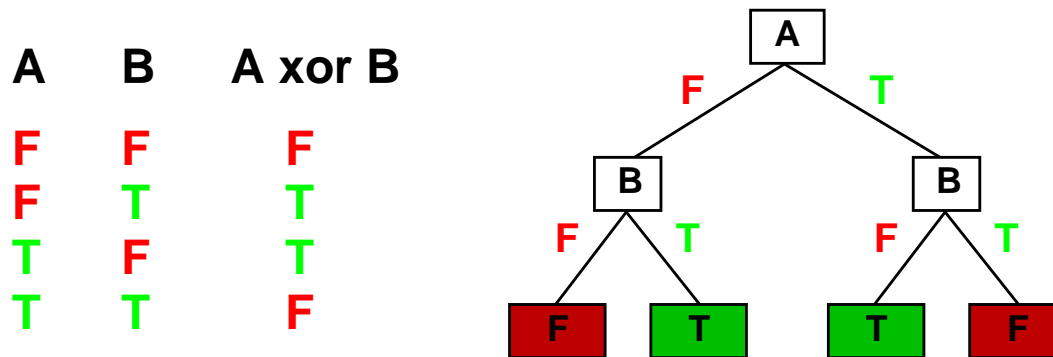
One possible representation for hypotheses

E.g., here is the “true” tree for deciding whether to wait:



# Expressiveness

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



Trivially, there is a consistent decision tree for any training set  
w/ 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??

# Hypothesis spaces

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

= number of Boolean functions

# Hypothesis spaces

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

= number of Boolean functions

= number of distinct truth tables with  $2^n$  rows

# Hypothesis spaces

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

= number of Boolean functions

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

# 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

# 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

How many purely conjunctive hypotheses (e.g.,  $Hungry \wedge \neg Rain$ )??

# 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

How many purely conjunctive hypotheses (e.g.,  $Hungry \wedge \neg Rain$ )??

Each attribute can be in (positive), in (negative), or out

$\Rightarrow 3^n$  distinct conjunctive hypotheses

More expressive hypothesis space

– increases chance that target function can be expressed 😊

– increases number of hypotheses consistent w/ training set

$\Rightarrow$  may get worse predictions 😞

# Decision tree learning

Aim: find a small tree consistent with the training examples

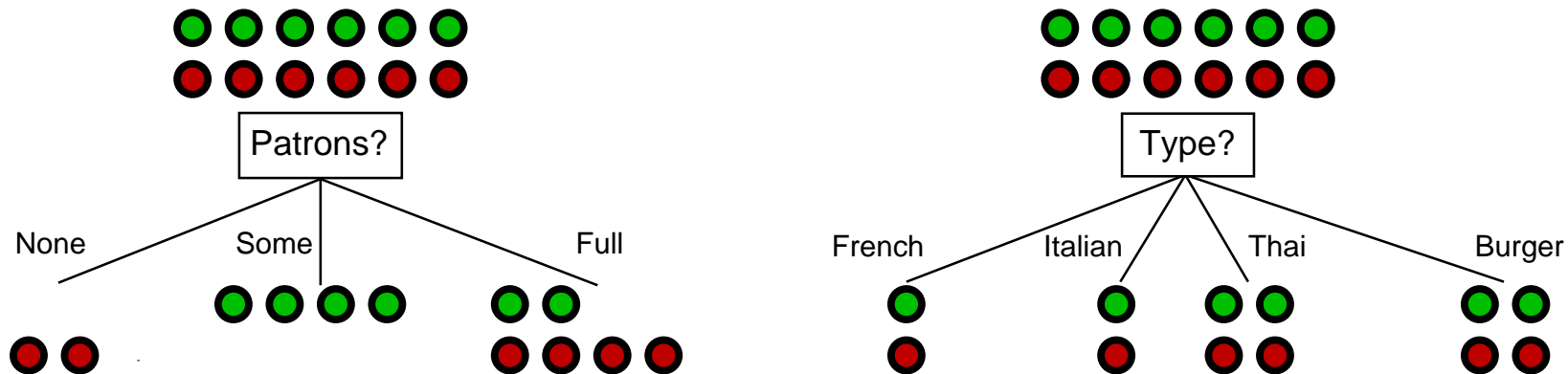
Idea: (recursively) choose “most significant” attribute as root of (sub)tree

```
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
```



# 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—gives **information** about the classification

# Information

Information answers questions

The more clueless I am about the answer initially, the more information is contained in the answer

Scale: 1 bit = answer to Boolean question with prior  $\langle 0.5, 0.5 \rangle$

Information in an answer when prior is  $\langle P_1, \dots, P_n \rangle$  is

$$H(\langle P_1, \dots, P_n \rangle) = \sum_{i=1}^n -P_i \log_2 P_i$$

(also called **entropy** of the prior)

## Information contd.

Suppose we have  $p$  positive and  $n$  negative examples at the root

$\Rightarrow H(\langle p/(p+n), n/(p+n) \rangle)$  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

$\Rightarrow H(\langle p_i/(p_i+n_i), n_i/(p_i+n_i) \rangle)$  bits needed to classify a new example

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

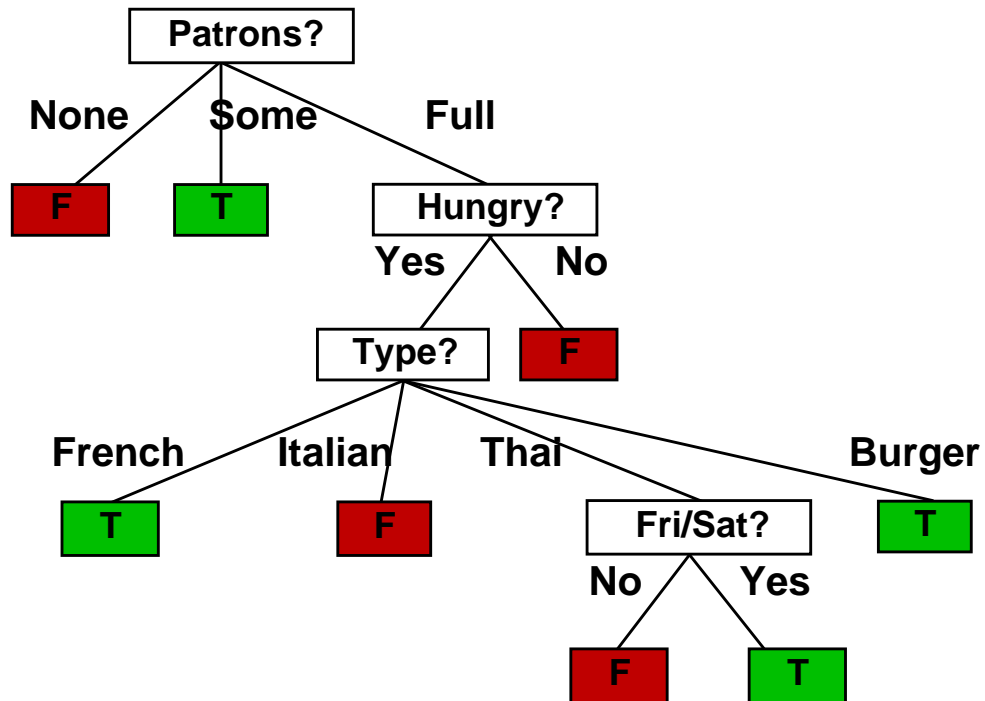
$$\sum_i \frac{p_i + n_i}{p + n} H(\langle p_i/(p_i + n_i), n_i/(p_i + n_i) \rangle)$$

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

$\Rightarrow$  choose the attribute that minimizes the remaining information needed

## Example contd.

Decision tree learned from the 12 examples:



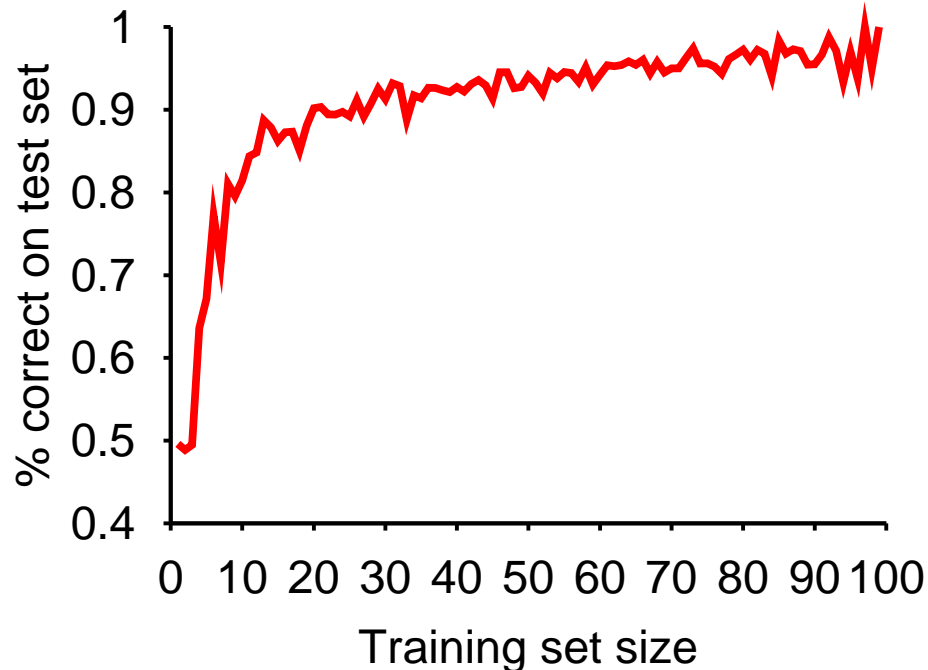
Substantially simpler than “true” tree—a more complex hypothesis isn’t justified by small amount of data

# Performance measurement

How do we know that  $h \approx f$ ? (Hume's **Problem of Induction**)

- 1) Use theorems of computational/statistical learning theory
- 2) Try  $h$  on a new **test set** of examples  
(use **same distribution over example space** as training set)

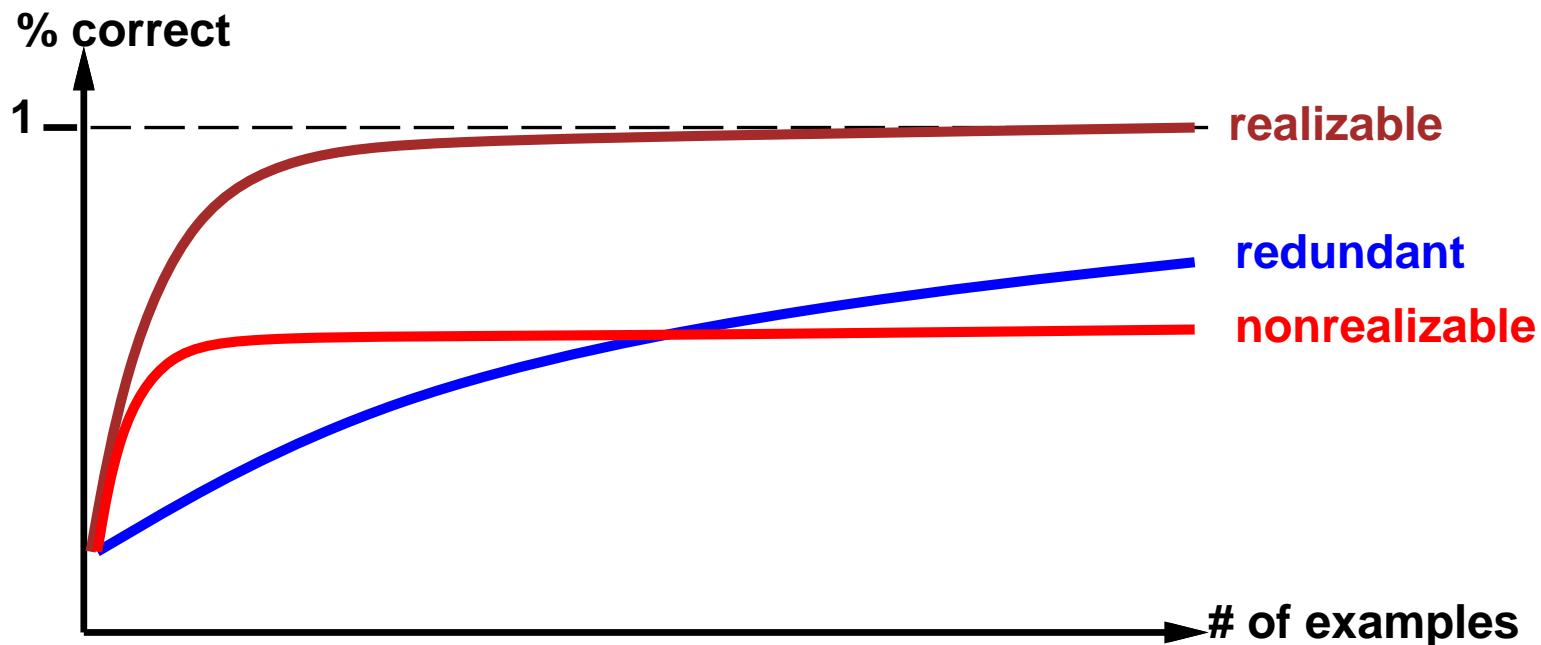
**Learning curve** = % correct on test set as a function of training set size



## Performance measurement contd.

Learning curve depends on

- **realizable** (can express target function) vs. **non-realizable**  
non-realizability can be due to missing attributes  
or restricted hypothesis class (e.g., thresholded linear function)
- redundant expressiveness (e.g., loads of irrelevant attributes)



## Summary

Learning needed for unknown environments, lazy designers

Learning agent = performance element + learning element

Learning method depends on type of performance element, available feedback, type of component to be improved, and its representation

For supervised learning, the aim is to find a simple hypothesis that is approximately consistent with training examples

Decision tree learning using information gain

Learning performance = prediction accuracy measured on test set