

1

## Intelligent Autonomous Agents ICS606/EE606

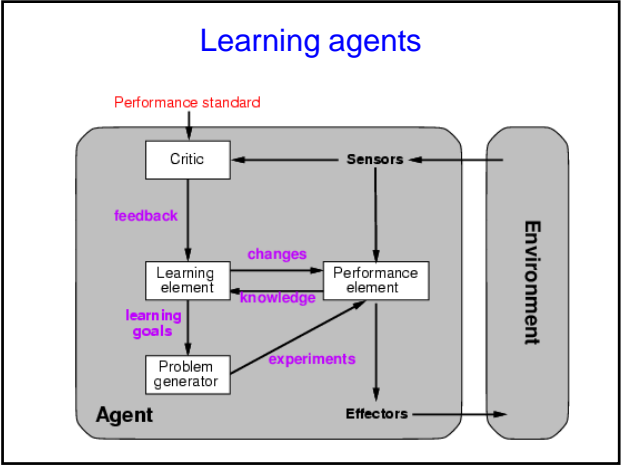
Nancy E. Reed  
nreed@hawaii.edu

## Learning Outline

- Learning agents
- Inductive learning
- Decision tree learning
- Reference
  - Chapter 18, AIMA

## 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 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
- Type of feedback:
  - **Supervised learning**: correct answers for each example
  - **Unsupervised learning**: correct answers not given
  - **Reinforcement learning**: occasional positive and/or negative rewards

## Example Scenarios

6

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

### Inductive Learning

- Simplest form: learn a function **from examples**
- $f$  is the **target function**
- An **example** is a pair  $(x, f(x))$

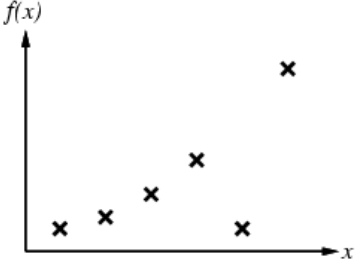
Problem: find a **hypothesis  $h$**  such that  $h \approx f$  given a **training set** of examples

This is a highly simplified model of real learning:

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

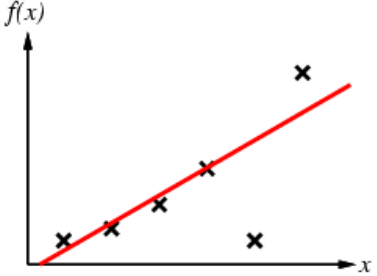
### Inductive Learning

- 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:



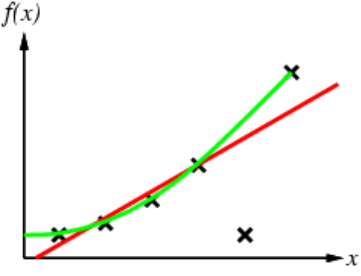
### Inductive Learning

- 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:



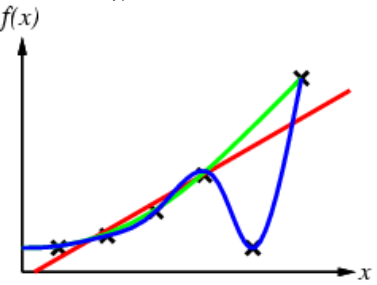
### Inductive Learning Method

- 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:



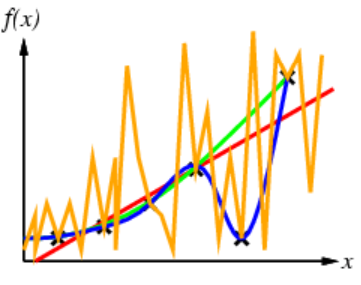
### Inductive learning method

- 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:



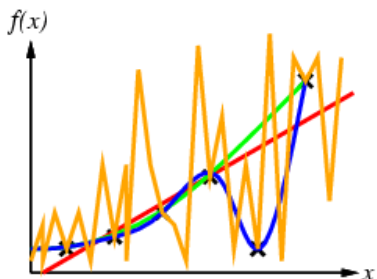
### Inductive Learning

- 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

- Ockham's razor: prefer the simplest hypothesis consistent with data



### Example: Learning Decision Trees

Question **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)

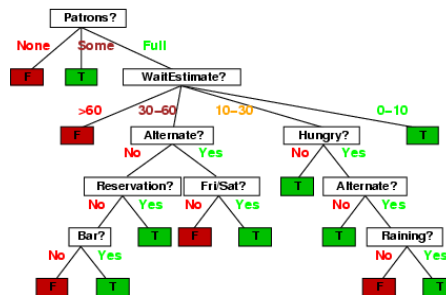
### Attribute-Based Representations

- 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 <sub>1</sub>	T	F	F	T	Some	\$\$\$	F	T	French	0-10	T
X <sub>2</sub>	T	F	F	T	Full	\$	F	F	Thai	30-60	F
X <sub>3</sub>	F	T	F	F	Some	\$	F	F	Burger	0-10	T
X <sub>4</sub>	T	F	T	T	Full	\$	F	F	Thai	10-30	T
X <sub>5</sub>	T	F	T	F	Full	\$\$\$	F	T	French	>60	F
X <sub>6</sub>	F	T	F	T	Some	\$\$	T	T	Italian	0-10	T
X <sub>7</sub>	F	T	F	F	None	\$	T	F	Burger	0-10	F
X <sub>8</sub>	F	F	F	T	Some	\$\$	T	T	Thai	0-10	T
X <sub>9</sub>	F	T	T	F	Full	\$	T	F	Burger	>60	F
X <sub>10</sub>	T	T	T	T	Full	\$\$\$	F	T	Italian	10-30	F
X <sub>11</sub>	F	F	F	F	None	\$	F	F	Thai	0-10	F
X <sub>12</sub>	T	T	T	T	Full	\$	F	F	Burger	30-60	T

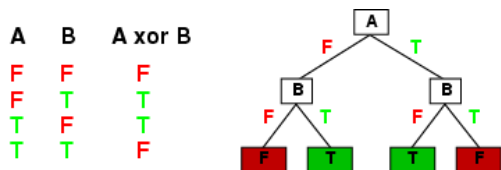
### Decision trees

- 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 → path to leaf.
- 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?

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

- Each attribute can be
  1. in (positive),
  2. in (negative), or
  3. out
 ⇒  $3^n$  distinct conjunctive hypotheses

## Hypothesis Spaces

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

- Each attribute can be in (positive), in (negative), or out
  - ⇒  $3^n$  distinct conjunctive hypotheses
- More expressive hypothesis space
  - increases chance that target function can be expressed
  - increases number of hypotheses consistent with training set
  - ⇒ may get worse predictions

## Decision Tree Learning

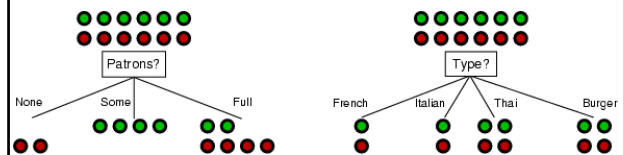
- Aim: find a **small tree** consistent with the training examples
- Idea: (recursively) choose "most significant" attribute as root of (sub)tree

```

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
       $examples_i$  ← {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



### Information 25

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, cont 26

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

### Using Information Theory

- To implement Choose-Attribute in the DTL algorithm
- Information Content (Entropy):  
 $I(\langle P(v_1), \dots, P(v_n) \rangle) = \sum_{i=1}^n -P(v_i) \log_2 P(v_i)$
- For a training set containing  $p$  positive examples and  $n$  negative examples:

$$I(\langle \frac{p}{p+n}, \frac{n}{p+n} \rangle) = -\frac{p}{p+n} \log_2 \frac{p}{p+n} - \frac{n}{p+n} \log_2 \frac{n}{p+n}$$

### Information Gain

- 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.

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

- Information Gain (IG) or reduction in entropy from the attribute test:  
 $IG(A) = I(\langle \frac{p}{p+n}, \frac{n}{p+n} \rangle) - remainder(A)$
- Choose the attribute with the largest IG

### Information Gain

For the training set,  $p = n = 6$ ,  $I(6/12, 6/12) = 1$  bit

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 is chosen by the DTL algorithm as the root

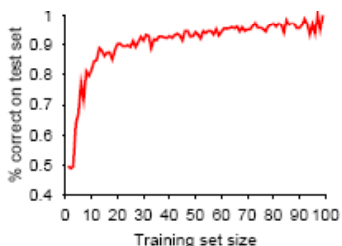
### Example cont.

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

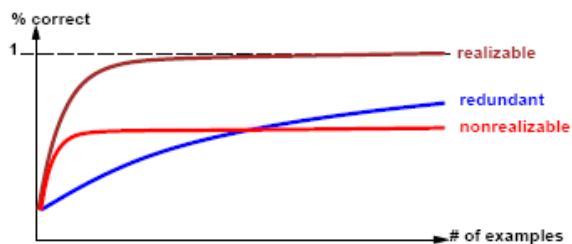
**Learning curve** = % correct on test set as a function of training set size



## Performance Measurement, cont

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
- For supervised learning, the aim is to find a simple hypothesis approximately consistent with training examples
- Decision tree learning using information gain
- Learning performance = prediction accuracy measured on test set