BLG435E
Artificial Intelligence

Lecture 10: Learning

Outline

- Learning agents
- Inductive learning
- Decision tree learning

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

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

Successful Examples

- Learning to drive an autonomous vehicle
- Learning to classify objects
- Learning to play world-class chess
- Learning to recognize spoken words
- Learning to run for a humanoid robot
- Learning action outcomes

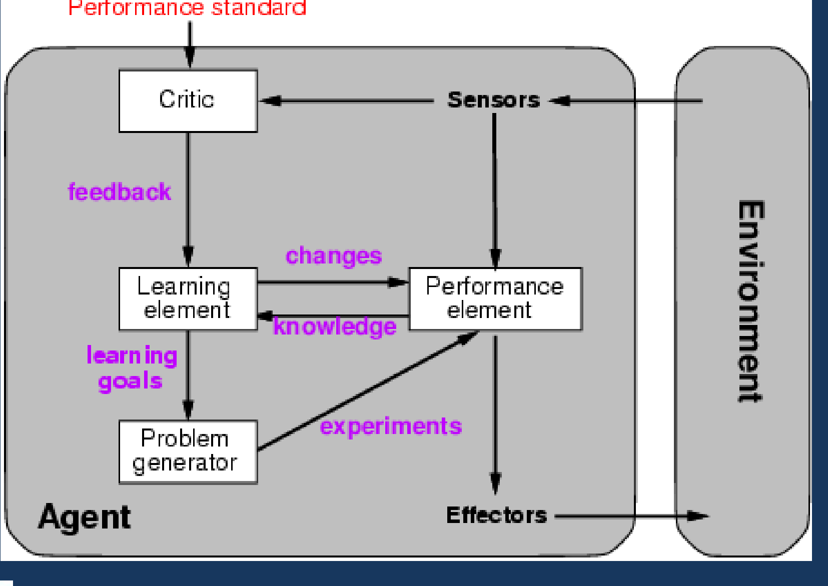
Why is learning important?

- Some tasks cannot be defined well, except by examples (e.g., recognizing people)
- Relationships and correlations can be hidden within large amounts of data
- Human designers often produce machines that do not work as well as desired in the environments in which they are used

Why is learning important?

- The amount of knowledge might be too large for explicit encoding by humans (e.g., medical diagnostic)
- Environments change over time
- New knowledge about tasks is constantly being discovered by humans

Learning agents



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 rewards

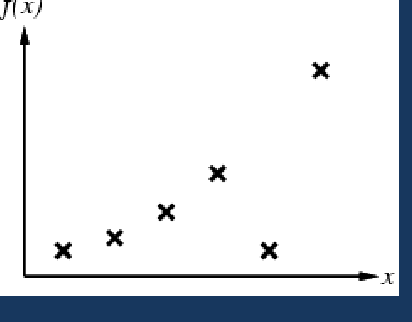
Inductive learning

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 ≈ f
  - given a training set of examples
- This is a highly simplified model of real learning:
  - Ignores prior knowledge
  - Assumes examples are given

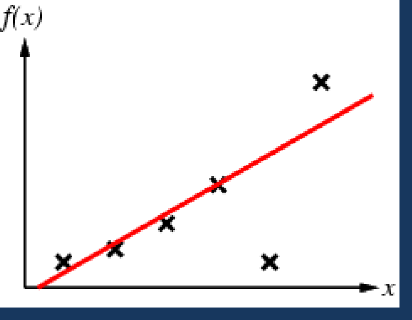
Inductive learning method

- 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:



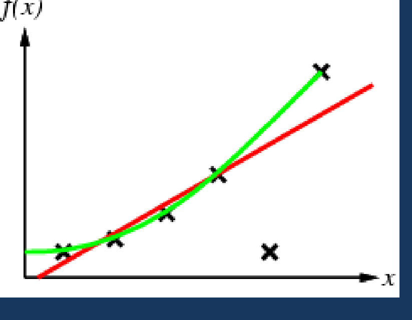
Inductive learning method

- 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:



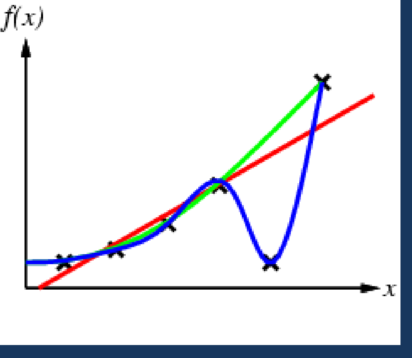
Inductive learning method

- 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:



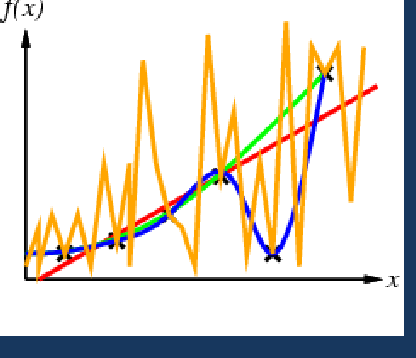
Inductive learning method

- 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: prefer the simplest hypothesis consistent with data

Decision Trees

- Simplest form of an inductive learning algorithm
- Decision tree
  - takes as input an object or situation described by a set of attributes
  - returns a decision - predicted output value
- If the output is from
  - a discrete function: Classification
  - a continuous function: Regression
- How to manuals

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. Wait Estimate: estimated waiting time (0-10, 10-30, 30-60, >60)

Attribute-based representations

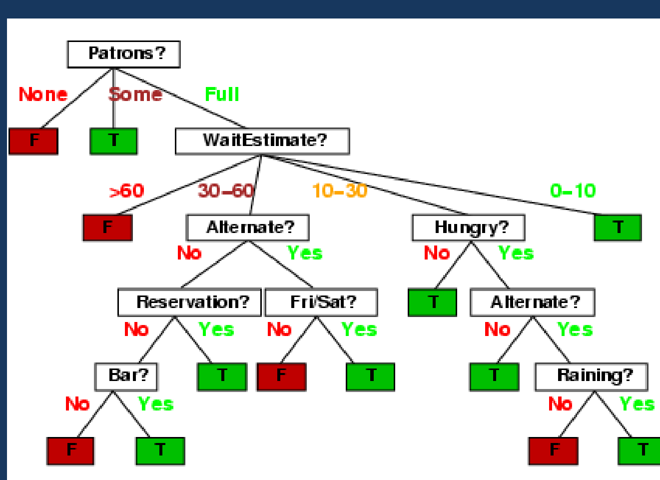
- Examples described by attribute values (Boolean, discrete, continuous)
  - e.g., situations where I will/won't wait for a table:

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
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

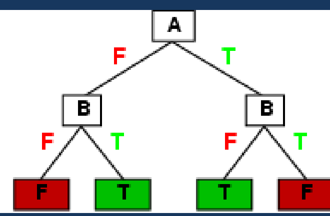
- One possible representation for hypotheses
  - e.g., here is the "true" tree for deciding whether to wait:



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



- for any training set with one path to leaf for each example → 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
- More expressive hypothesis space
  - increases chance that target function can be expressed
  - increases number of hypotheses consistent with training set
    - may get worse predictions

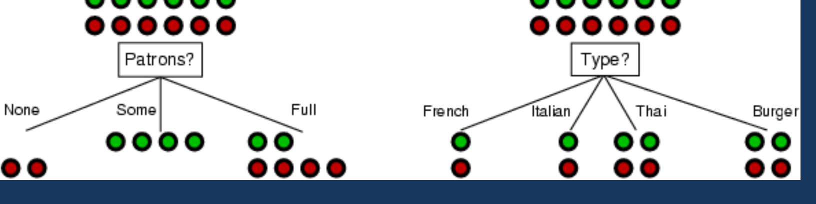
Decision tree learning

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



- *Which* is a better choice?

Using information theory

- To implement `Choose-Attribute` in the DTL algorithm
- Information Content (Entropy, Shannon and Weaver(1945)):

$$I(P(v_1), \ldots, P(v_n)) = \Sigma_{i=1} -P(v_i) \log_2 P(v_i)$$

- For a training set containing *p* positive examples and *n* negative examples:

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

Information gain

- A chosen attribute *A* divides the training set *E* into subsets $E_1, \ldots, E_v$ according to their values for *A*, where *A* has v distinct values.

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

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

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

- Choose the attribute with the largest IG

Information gain

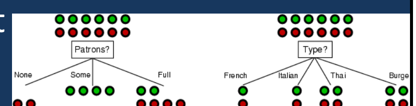
For the training set, $p = n = 6$, $I(6/12, 6/12) = 1$ bit

Consider the attributes *Patrons* and *Type* (and others too):

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

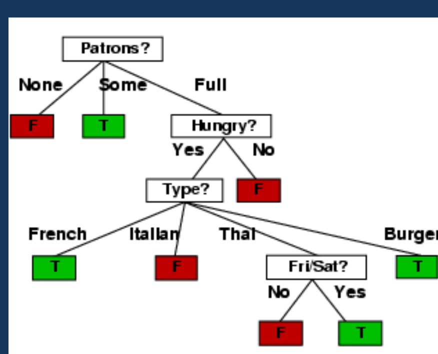
$$IG(Type) = 1 - [\frac{2}{12} I(\frac{1}{2}, \frac{1}{2}) + \frac{2}{12} I(\frac{1}{2}, \frac{1}{2}) + \frac{4}{12} I(\frac{2}{4}, \frac{2}{4}) + \frac{4}{12} I(\frac{2}{4}, \frac{2}{4})] = 0 \text{ bits}$$

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



Example 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 $h \approx f$?
  - 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