

Learning from Observations

Chapter 18
Section 1 – 3

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

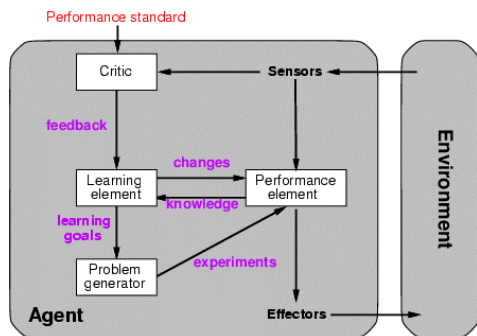
Learning

- 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

Inductive learning

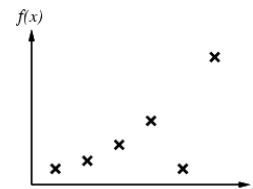
- Simplest form: learn a function from examples
- f is the **target function**
- An **example** is a pair $(x, f(x))$
- Problem: find a **hypothesis** h
such that $h \approx f$
given a **training set** of examples
- (This is a highly simplified model of real learning:
 - Ignores prior knowledge
 - Assumes a deterministic, observable "environment"
 - Assumes examples are given)

Learning agents



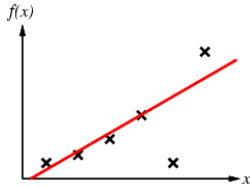
Inductive learning method

- 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:



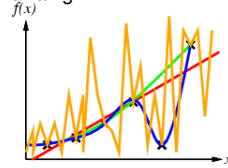
Inductive learning method

- 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:



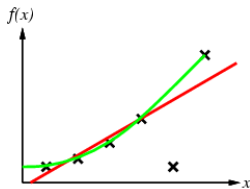
Inductive learning method

- 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:



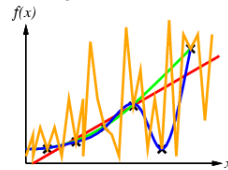
Inductive learning method

- 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

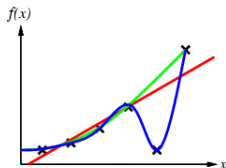
- Construct/adjust h to agree with f on training set
- (h is **consistent** if it agrees with f on all examples)
- E.g., curve fitting:



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

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:



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)

Attribute-based representations

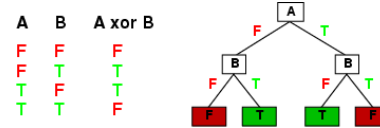
- Examples described by **attribute values** (Boolean, discrete, continuous)
- E.g., situations where I will/won't wait for a table:

Example	Attributes										Target
	All	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	
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	F
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	F	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)

Expressiveness

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

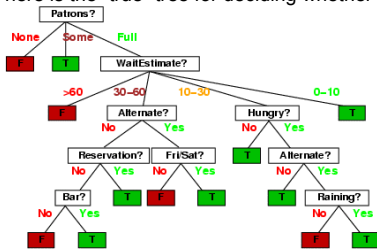


- Trivially, there is a consistent decision tree for any training set with one path to leaf for each example (unless *f* nondeterministic in *x*) but it probably won't generalize to new examples

- Prefer to find more **compact** decision trees

Decision trees

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



Hypothesis spaces

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

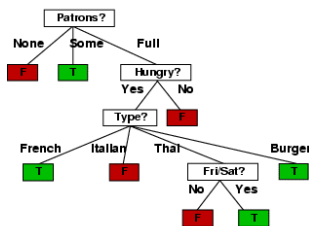
= number of Boolean functions

= 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 trees

- Another possible representation for hypotheses



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 ⇒ 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 vi of best do
    examplesi ← {elements of examples with best = vi}
    subtree ← DTL(examplesi, attributes - best, MODE(examplesi))
    add a branch to tree with label vi and subtree subtree
  return tree
    
```

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.

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

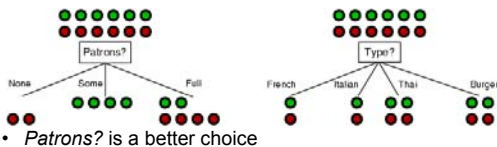
- Information Gain (IG) or reduction in entropy from the attribute test:

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

- Choose the attribute with the largest IG

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

Computing Entropy and IG

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

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

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

Example	Attributes								Target		
	Att	Bar	Fri	Han	Pat	Price	Rain	Res			
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

Using information theory

- To implement `Choose-Attribute` in the DTL algorithm
- Information Content (Entropy):

$$I(P(v_1), \dots, P(v_n)) = \sum_{i=1}^n -P(v_i) \log_2 P(v_i)$$
- For a training set containing p positive examples and n negative examples:

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

Information gain

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

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

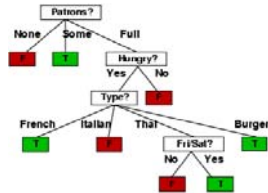
$$IG(\text{Patrons}) = 1 - \left[\frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I\left(\frac{2}{6}, \frac{4}{6}\right) \right] = .0541 \text{ bits}$$

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

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 the first tree---a more complex hypothesis isn't justified by small amount of data

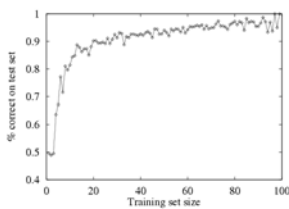
Summary

- 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

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



Why Learning Works

- There is a theoretic foundation: Computational Learning Theory.
- The underlying principle: Any hypothesis that is consistent with a sufficiently large set of training examples is unlikely to be seriously wrong: it must be **Probably Approximately Correct (PAC)**.
- The **Stationarity** Assumption: The training and test sets are drawn randomly from the same population of examples using the same probability distribution.