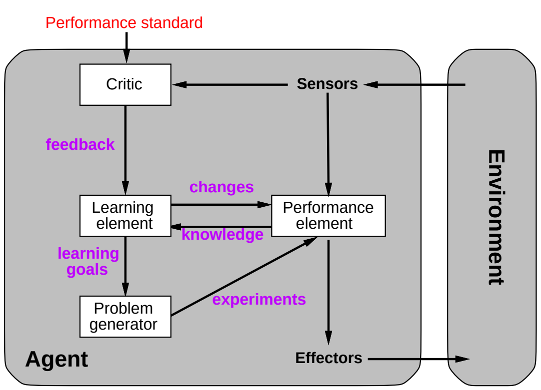
MACHINE LEARNING

Overview

- Most of the time we can't program our agents to do everything right to begin with.
- We don't have enough information about the environment.
- So we get them to *learn* what to do.
- Different forms of learning:
  - Inductive learning; and
  - Reinforcement learning.

Learning agents



- 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
- *Supervised learning*: correct answers for each instance.
- *Reinforcement learning*: occasional rewards.

- 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

- Today we'll look at *inductive learning* of decision trees and *reinforcement learning*.

Inductive learning

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

$$\begin{array}{c|c|c} O & O & X \\ \hline & X & \\ \hline X & & \end{array} \quad , \quad +1$$

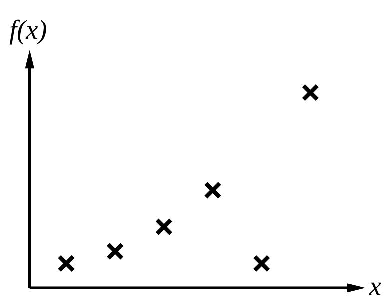
- Problem: find a(n) *hypothesis* h such that

$$h \approx f$$

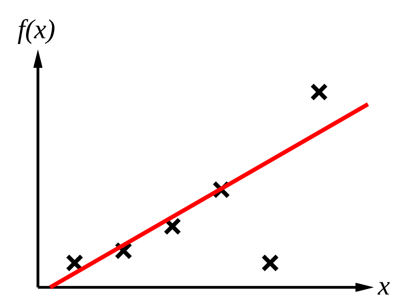
given a *training set* of examples

Inductive learning method

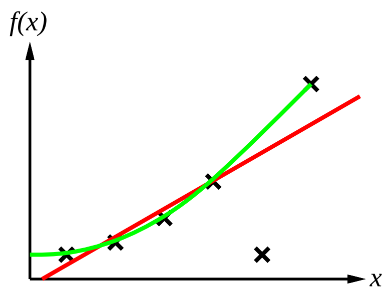
- Construct/adjust h to agree with f on training set
(*h* is *consistent* if it agrees with f on all examples)
E.g., curve fitting:



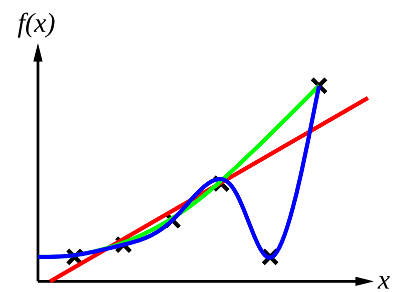
- Construct/adjust h to agree with f on training set
(*h* is *consistent* if it agrees with f on all examples)
E.g., curve fitting:



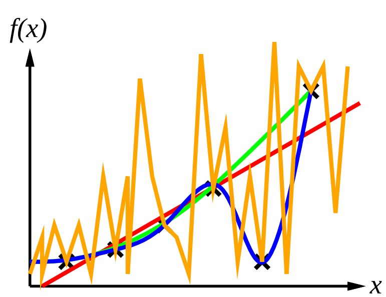
- Construct/adjust h to agree with f on training set
  (h is *consistent* if it agrees with f on all examples)
  E.g., curve fitting:



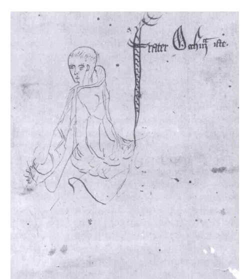
- Construct/adjust h to agree with f on training set
  (h is *consistent* if it agrees with f on all examples)
  E.g., curve fitting:



- 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



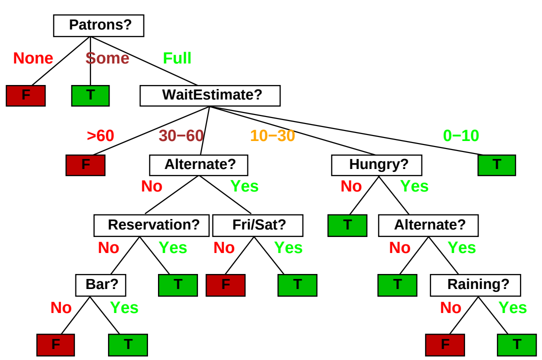
Attribute-based representations

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*

- Examples described by *attribute values* (Boolean, discrete, continuous, etc.)
- *Classification* of examples is *positive* (T) or *negative* (F)

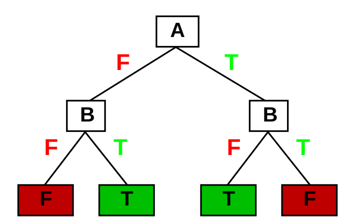
Decision trees

- Here is the "true" tree for deciding whether to wait:



- Decision trees can express any function of the input attributes.
- 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, $\exists$ a consistent decision tree for any training set with one path to leaf for each example.
  - unless f nondeterministic in x
- This trivial tree 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?

- How many distinct decision trees with n Boolean attributes?

  = number of Boolean functions

- How many distinct decision trees with n Boolean attributes?

  = number of Boolean functions

  = number of distinct truth tables with 2^n rows

- 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}

- 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}

  6 Boolean attributes means 18,446,744,073,709,551,616 trees

- 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}

  6 Boolean attributes means 18,446,744,073,709,551,616 trees
- How many purely conjunctive hypotheses ($Hungry \wedge \neg Rain$)?

- 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}

  6 Boolean attributes means 18,446,744,073,709,551,616 trees
- How many purely conjunctive hypotheses ($Hungry \wedge \neg Rain$)?
- 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 with training set $\Rightarrow$ may get worse predictions

Decision tree learning

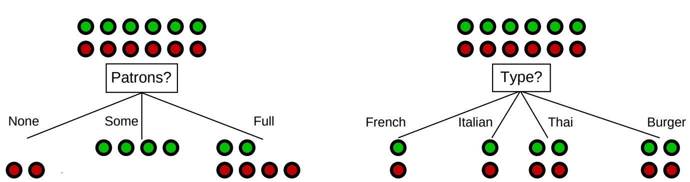
- Aim: find a small tree consistent with the training examples.
- Idea: (recursively) choose "most significant" attribute as root of (sub)tree.

Decision tree learning

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".



- *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 $\langle P_1, \ldots, P_n \rangle$ is

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

(also called *entropy* of the prior)

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

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

  bits needed to classify a new example.
- 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.

  $$H(\langle p_i/(p_i+n_i), n_i/(p_i+n_i)\rangle)$$

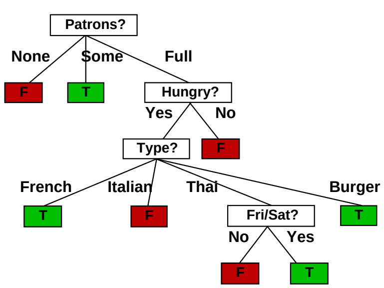
  bits needed to classify a new example
- *Expected* number of bits per example over all branches is

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

- For *Patrons*?, this is 0.459 bits.
- For *Type* this is (still) 1 bit
- Choose the attribute that minimizes the remaining information needed

Back to the example

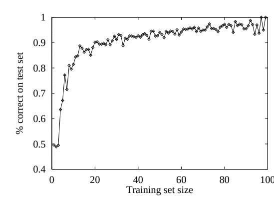
- Decision tree learned from the 12 examples:



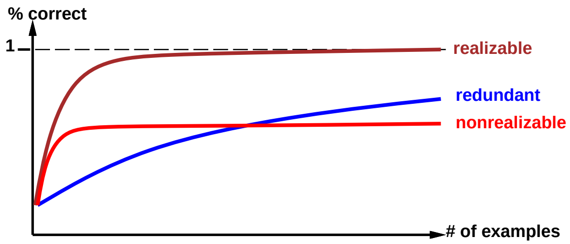
- Substantially simpler than "true" tree—a more complex hypothesis isn't justified by small amount of data

Performance measurement

- How do we know that $h \approx f$?
  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



- Learning curve depends on
  - *realizable* (can express target function) vs. *non-realizable* non-realizability can be due to missing attributes or restricted hypothesis class
  - 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 approximately consistent with training examples
- Decision tree learning using information gain
- Learning performance = prediction accuracy measured on test set