

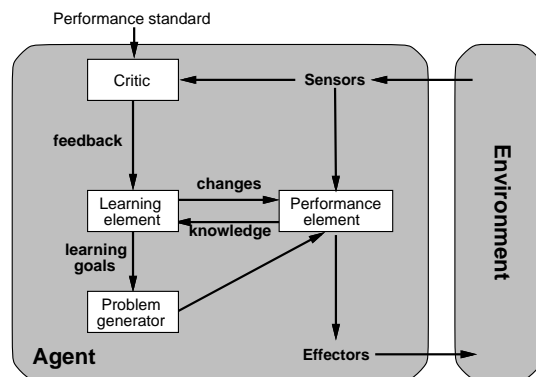
LEARNING

CHAPTER 18

Outline

- ◇ Inductive Learning
- ◇ Learning Decision Trees
- ◇ Version Spaces

General model of Learning agents



- learning element
- performance element (previously the entire agent)
- example: automated taxi

Classification based on feedback

How an autonomous agent that senses and acts in its environment can learn to choose optimal actions to achieve its goals?

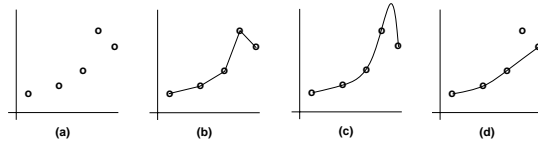
- *supervised learning*: each time the agent performs an action in its environment, a trainer provides the correct action.
- *reinforcement learning*: trainer provides *only* a reward or penalty to indicate desirability of resulting state.
- *unsupervised learning*: no hint about the correct action.

How does learning to play tennis fit into the general learning model??

Inductive learning

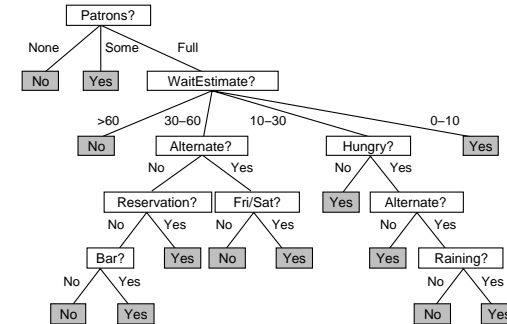
inductive inference: given a collection of examples $(x, f(x))$
return a function h (hypothesis) that approximates f

bias: any preference for one hypothesis over another



Decision trees

input: object or situation described by a set of properties
output: "yes/no" decision (e.g., *WillWait?*) or a category
– first identify attributes to describe examples in the domain



$$\forall r \text{ Patrons}(r, \text{Full}) \wedge \text{WaitEstimate}(r, 0-10) \wedge \text{Hungry}(r, N) \Rightarrow \text{WillWait}(r)$$

Expressiveness of decision trees

- a decision tree corresponds to a set of implication sentences
- Can decision trees represent any set??
 - no, since they talk about a single object (e.g., restaurant)
 - each attribute test is a proposition (no $Nearby(r_2, r)$)
- Decision trees: fully expressive within the class of propositional languages
 - any Boolean function can be written as a decision tree
 - but not all functions can be represented efficiently

Consider the set of all functions on n attributes:

- there are 2^n functions (truth table with 2^n rows)
 - the "answer" column is a 2^n bit number that defines function
 - most of the functions require 2^n bits to represent
- $\Rightarrow 2^{2^n}$ different functions on n attributes (scary!!!)

how to find consistent hypotheses in such a large space??

Inducing decision trees from examples

example: (attribute values & goal predicate values)
example classification: positive/negative

Example	Attributes										Goal
	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	WillWait
X_1	Yes	No	No	Yes	Some	\$\$\$	No	Yes	French	0-10	Yes
X_2	Yes	No	No	Yes	Full	\$	No	No	Thai	30-60	No
X_3	No	Yes	No	No	Some	\$	No	No	Burger	0-10	Yes
X_4	Yes	No	Yes	Yes	Full	\$	No	No	Thai	10-30	Yes
X_5	Yes	No	Yes	No	Full	\$\$\$	No	Yes	French	>60	No
X_6	No	Yes	No	Yes	Some	\$\$	Yes	Yes	Italian	0-10	Yes
X_7	No	Yes	No	No	None	\$	Yes	No	Burger	0-10	No
X_8	No	No	No	Yes	Some	\$\$	Yes	Yes	Thai	0-10	Yes
X_9	No	Yes	Yes	No	Full	\$	Yes	No	Burger	>60	No
X_{10}	Yes	Yes	Yes	Yes	Full	\$\$\$	No	Yes	Italian	10-30	No
X_{11}	No	No	No	No	None	\$	No	No	Thai	0-10	No
X_{12}	Yes	Yes	Yes	Yes	Full	\$	No	No	Burger	30-60	Yes

How to build a decision tree for a set of examples??

Inducing decision trees cont.

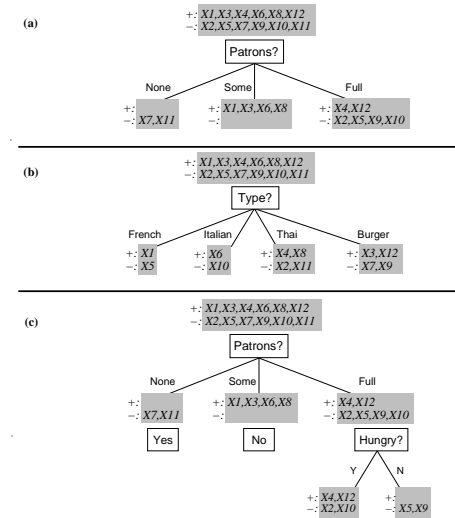
Ockam's razor: most likely hypothesis is the simplest one that is consistent with all observations.

– heuristic: test the most important attribute first

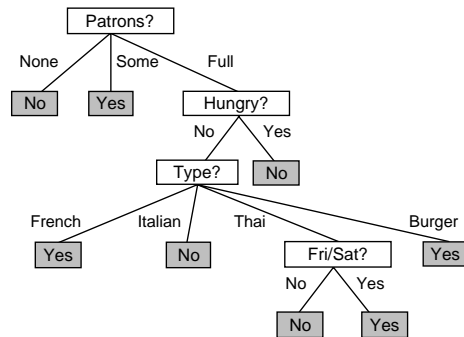
Algorithm

- classify examples in +/- sets.
- choose "best" attribute for test
- apply algorithm recursively for each test outcome
 - if + & - examples, choose best attribute to split them
 - if all + or - done
 - no examples left \Rightarrow no such examples observed
 - no attributes left \Rightarrow same description, diff. classification (noise)

Splitting Examples



Induced decision tree



examples generated by agent using original tree
 but induced tree different form original tree

Decision trees learning algorithm

function DECISION-TREE-LEARNING(*examples*, *attributes*, *default*) **returns** a decision tree

inputs: *examples*, set of examples
attributes, set of attributes
default, default value for the goal predicate

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** MAJORITY-VALUE(*examples*)

else

best \leftarrow CHOOSE-ATTRIBUTE(*attributes*, *examples*)

tree \leftarrow a new decision tree with root test *best*

for each value v_i of *best* **do**

examples_i \leftarrow {elements of *examples* with *best* = v_i }

subtree \leftarrow DECISION-TREE-LEARNING(*examples_i*, *attributes* – *best*, MAJORITY-VALUE(*examples_i*))

add a branch to *tree* with label v_i and subtree *subtree*

end

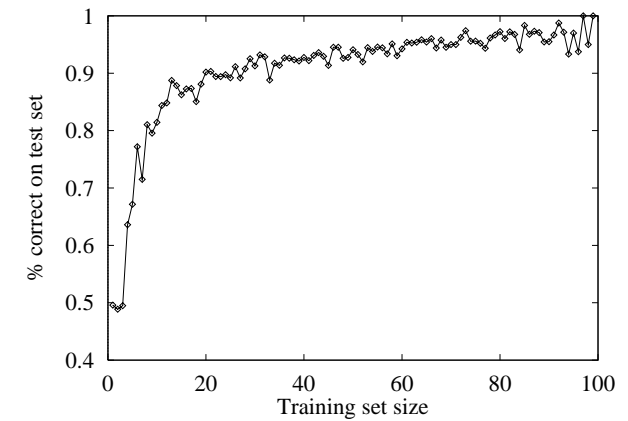
return *tree*

Assessing performance of learning algorithm

Methodology:

1. Collect a large set of examples
2. Divide it into two disjoint sets: training and test set
3. Use training set to generate hypothesis H
4. Find how many examples in test set are correctly classified by H
5. Repeat 1-4 for different sizes (and) order of training sets.

Learning curve



prediction quality increases when training set grows

Learning general logical descriptions

inductive learning: search for a good hypothesis in a large *hypothesis space*

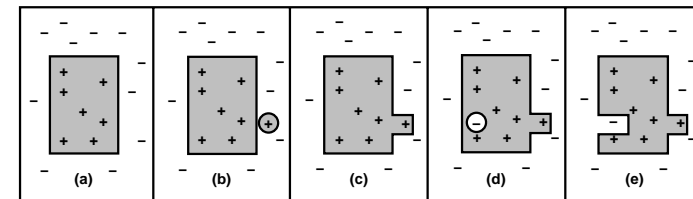
Hypotheses:

- start with a goal unary predicate Q (e.g., *WillWait*)
- find an equivalent logical expression (candidate definition C_i)
- each hypothesis H_i is a sentence: $\forall x Q(x) \Leftrightarrow C_i(x)$
e.g., $\forall r \text{ WillWait}(r) \Leftrightarrow \text{Patrons}(r, \text{Some}) \vee$
 $\text{Patrons}(r, \text{Full}) \wedge \neg \text{Hungry}(r) \wedge \text{Type}(r, \text{French}) \vee \dots$
- *hypothesis space*: set of all hypotheses $\{H_1, H_2, \dots, H_n\}$
- algorithm believes one is correct: $H_1 \vee H_2 \vee \dots \vee H_n$
- each H_i predicts that examples satisfying C_i will be examples of Q

Examples:

- objects to which the goal concept may or may not apply
- false negative for H_i
- false positive for H_i

Current-best-hypothesis search



- a) consistent hypothesis
- b) false negative
- c) generalization (drop conditions)
- d) false positive
- e) specialization (add extra conditions)

Current-best-hypothesis search algorithm

function CURRENT-BEST-LEARNING(*examples*) **returns** a hypothesis

$H \leftarrow$ any hypothesis consistent with the first example in *examples*

for each remaining example in *examples* **do**

if e is false positive for H **then**

$H \leftarrow$ **choose** a specialization of H consistent with *examples*

else if e is false negative for H **then**

$H \leftarrow$ **choose** a generalization of H consistent with *examples*

if no consistent specialization/generalization can be found **then fail**

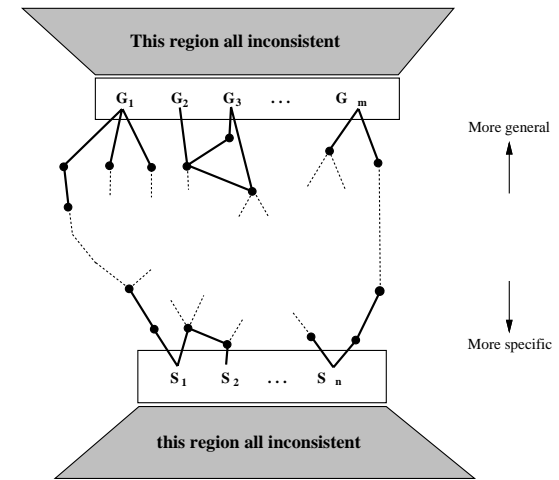
end

return H

- searches for a consistent hypothesis
- backtracks when no consistent specialization/generalization can be chosen.

What if we keep all the hypotheses consistent with all data so far??

Least commitment search



version space

Why learning works

Computational learning theory