Learning from Observations

Chapter 18, Sections 1–3

Outline

◇ Inductive learning

◇ Decision tree learning

◇ Measuring learning performance

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

Different kinds of learning:
– Supervised learning: we get correct answers for each training instance
– Reinforcement learning: we get occasional rewards
– Unsupervised learning: we don't know anything. . .

Inductive learning

Simplest form: learn a function from examples

f is the target function

An example is a pair x, $f(x)$, e.g., $\begin{array}{c|c|c} O & O & X \\ \hline & X & \\ \hline X & & \end{array}$, $+1$

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**

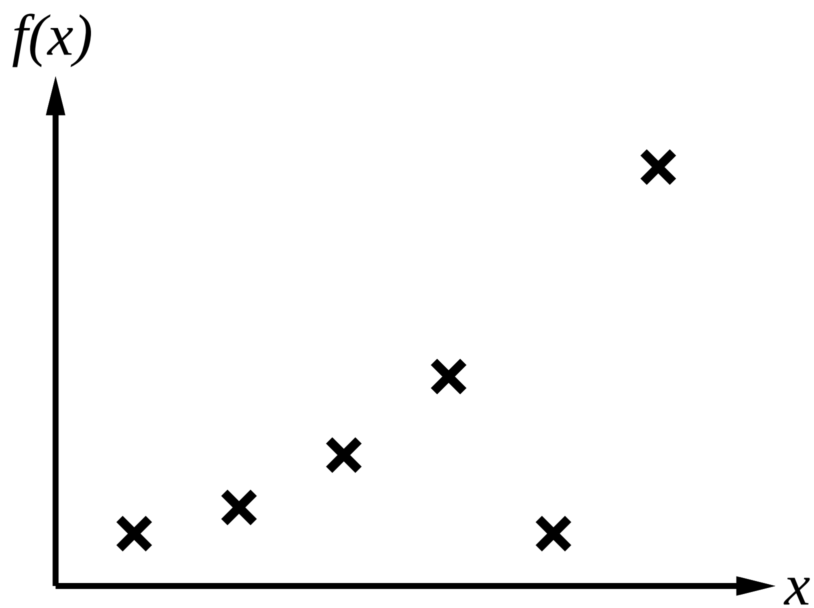
– **Assumes a deterministic, observable "environment"**

– **Assumes that the 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)

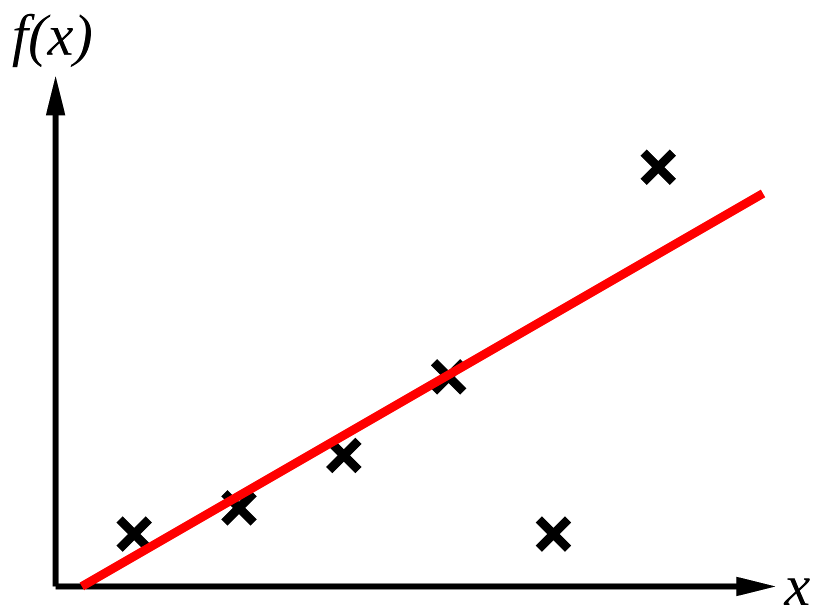
E.g., curve fitting:



Inductive learning method

Construct/adjust h to agree with f on training set
(h is consistent if it agrees with f on all examples)

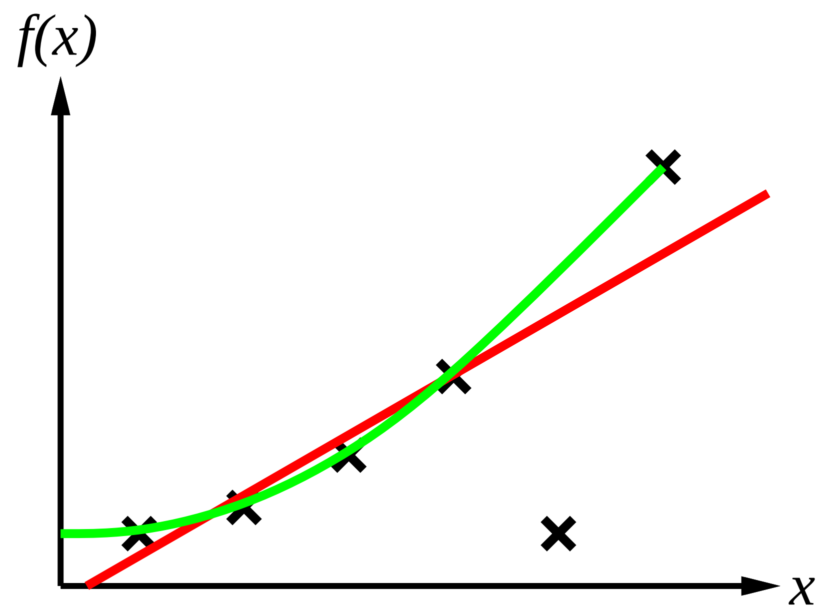
E.g., curve fitting:



Inductive learning method

Construct/adjust h to agree with f on training set
(h is consistent if it agrees with f on all examples)

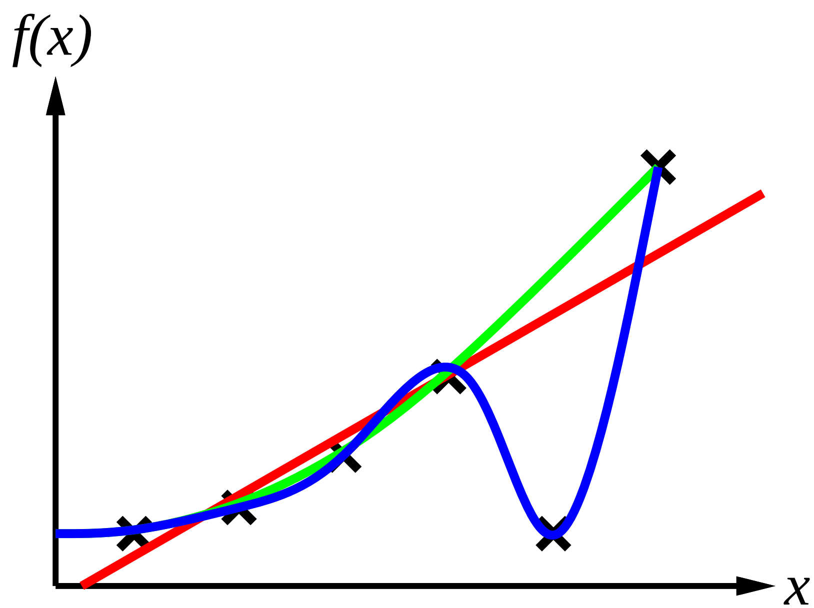
E.g., curve fitting:



Inductive learning method

Construct/adjust h to agree with f on training set
(h is consistent if it agrees with f on all examples)

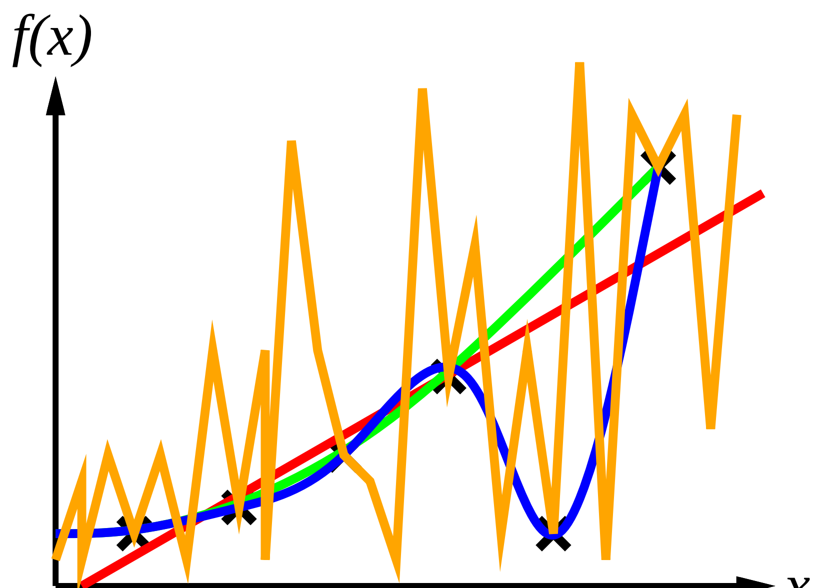
E.g., curve fitting:



Inductive learning method

Construct/adjust h to agree with f on training set
(h is consistent if it agrees with f on all examples)

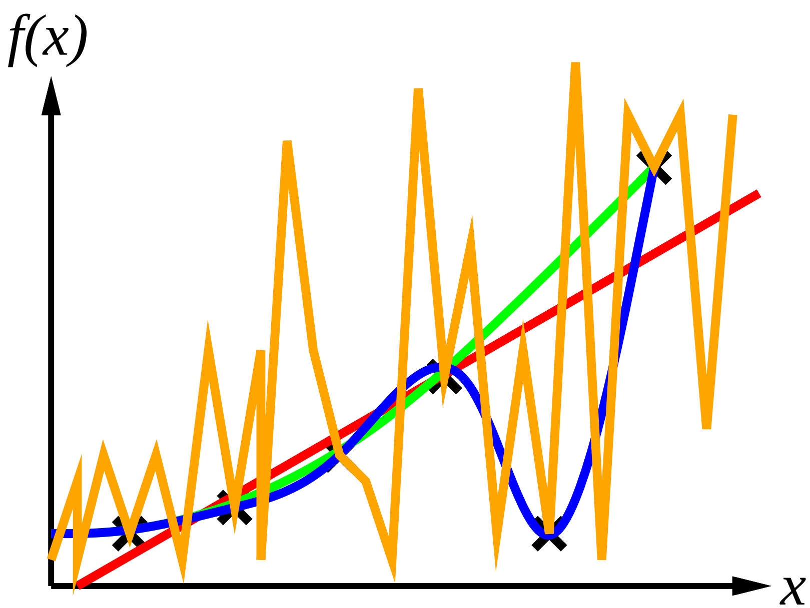
E.g., curve fitting:



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: maximize a combination of consistency and simplicity

Attribute-based representations

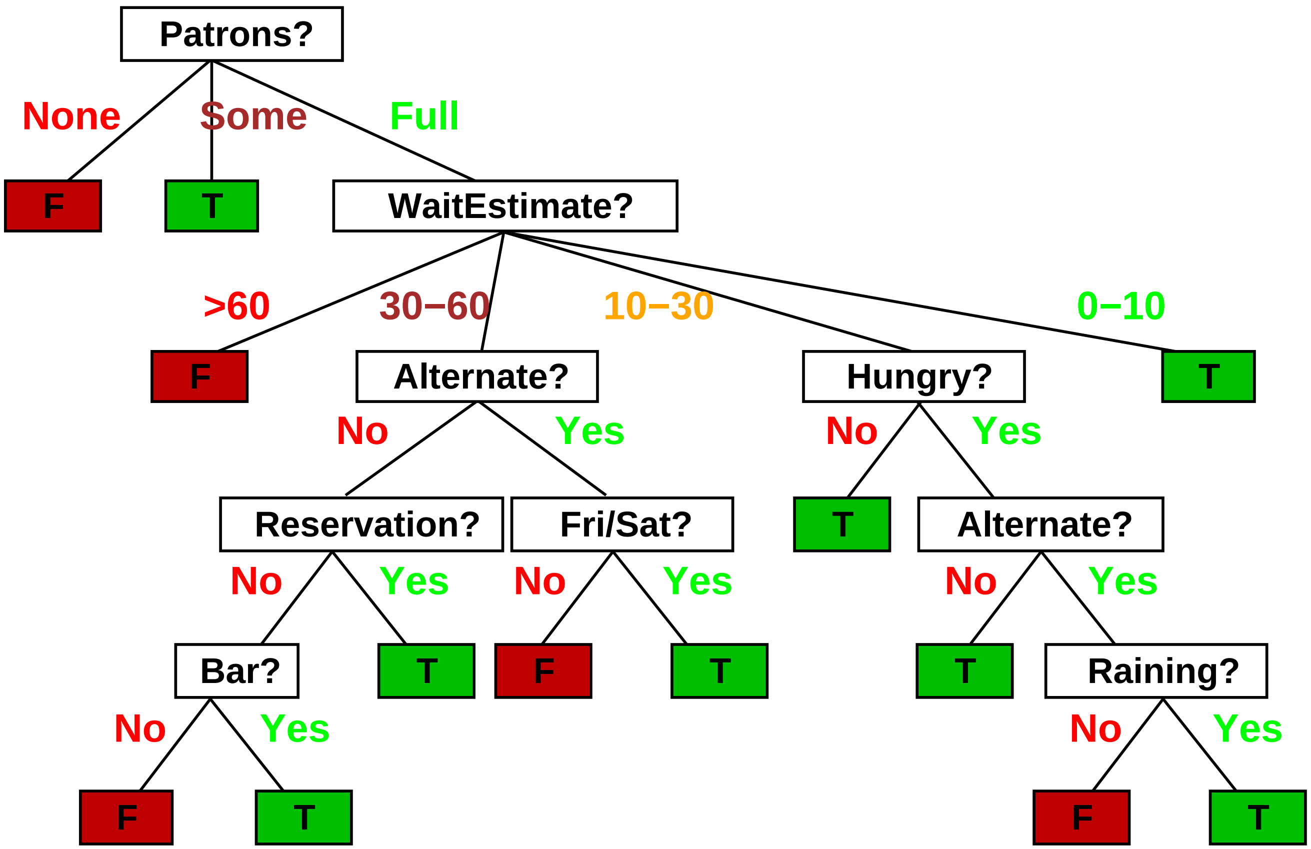
Examples described by attribute values (Boolean, discrete, continuous, etc.)
E.g., situations where I will/won't wait for a table:

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*

$^*Alt(ernate), Fri(day), Hun(gry), Pat(rons), Res(ervation), Est(imated\ waiting\ time)$

Decision trees

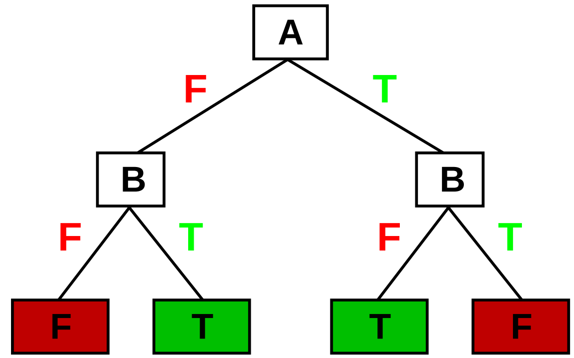
Decision trees are one possible representation for hypotheses, e.g.:



Expressiveness

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

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



Trivially, there is a consistent decision tree for any training set with one path to a leaf for each example
– but it does probably not generalize to new examples

We prefer to find more **compact** decision trees

Hypothesis spaces

How many distinct decision trees are there with n Boolean attributes??

= number of Boolean functions
= number of distinct truth tables with 2^n rows
= 2^{2^n} distinct decision trees

E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 trees

Decision tree learning

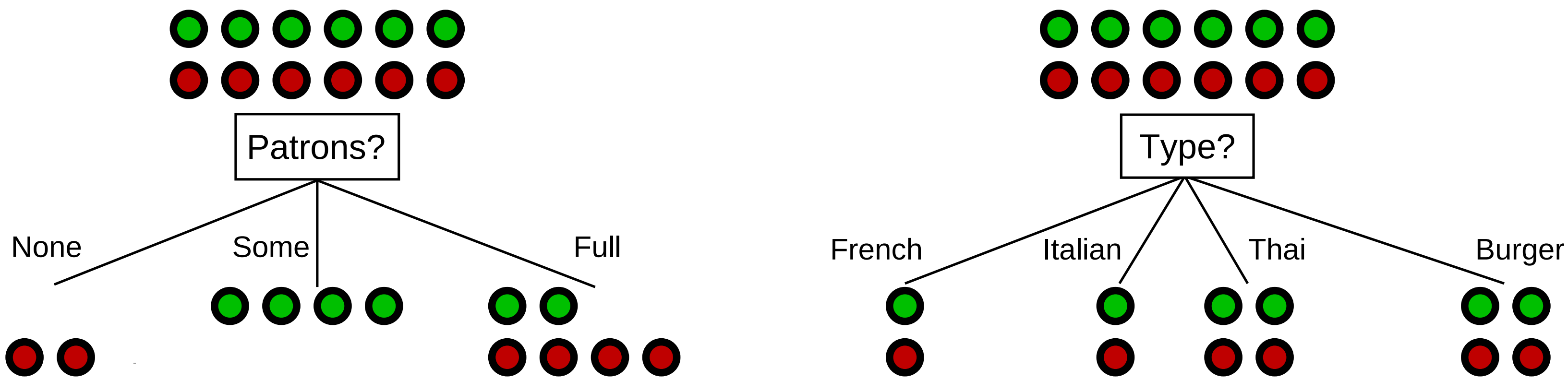
Aim: find a small tree consistent with the training examples

Idea: (recursively) choose "most significant" attribute as root of (sub)tree

```
function DTL(examples, attributes, parent-exs) returns a decision tree
    if examples is empty then return PLURALITY-VALUE(parent-exs)
    else if all examples have the same classification then return the classification
    else if attributes is empty then return PLURALITY-VALUE(examples)
    else
        A ← argmax_{a ∈ attributes} IMPORTANCE(a, examples)
        tree ← a new decision tree with root test A
        for each value v_i of A do
            exs ← {e ∈ examples such that e[A] = v_i}
            subtree ← DTL(exs, attributes−A, examples)
            add a branch to tree with label (A = 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—it 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 a Boolean question with prior $\langle 0.5, 0.5 \rangle$

The information in an answer when prior is $V = \langle P_1, \ldots, P_n \rangle$ is

$$\begin{aligned} H(V) &= \quad \textstyle\sum_{k=1}^{n} P_k \log_2 \frac{1}{P_k} \\ &= -\textstyle\sum_{i=1}^{n} P_k \log_2 P_k \end{aligned}$$

(this is called the entropy of V)

Information contd.

Suppose we have p positive and n negative examples at the root
$\Rightarrow$ we need $H(\langle p/(p+n),\ n/(p+n)\rangle)$ bits to classify a new example
E.g., for our example with 12 restaurants, $p = n = 6$ 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
$\Rightarrow$ we need $H(\langle p_i/(p_i+n_i),\ n_i/(p_i+n_i)\rangle)$ bits to classify a new example

The **expected** number of bits per example over all branches is

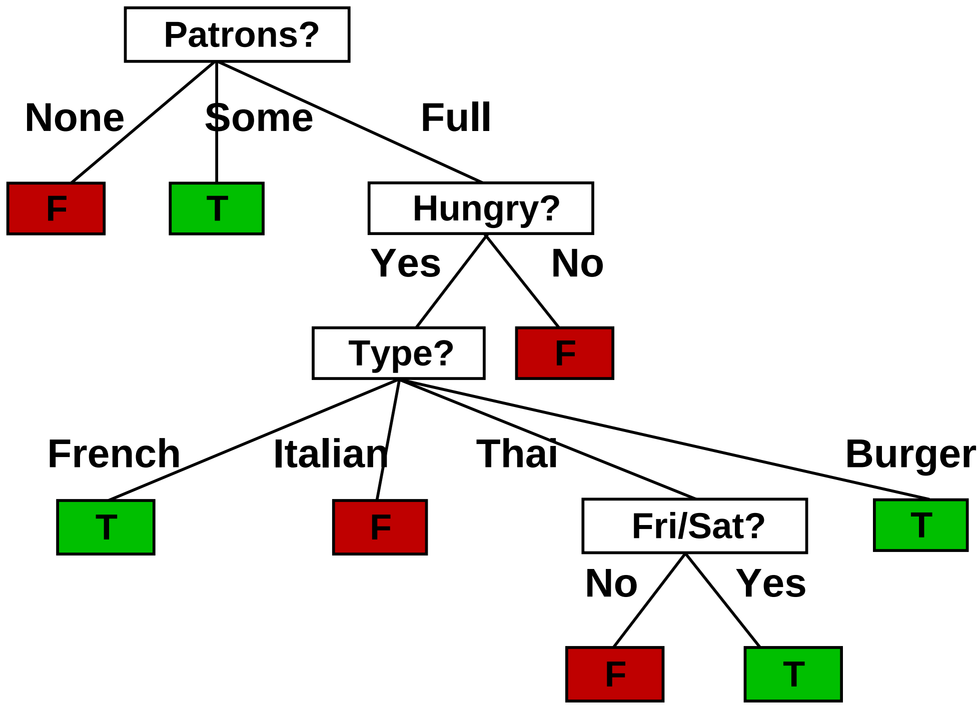
$$\Sigma_i \ \frac{p_i + n_i}{p + n} H(\langle p_i/(p_i+n_i),\ n_i/(p_i+n_i)\rangle)$$

For $Patrons?$, this is 0.459 bits, for $Type$ this is (still) 1 bit

$\Rightarrow$ choose the attribute that minimizes the remaining information needed

Example contd.

Decision tree learned from the 12 examples:



Substantially simpler than the "true" tree
– a more complex hypothesis isn't justified by that small amount of data

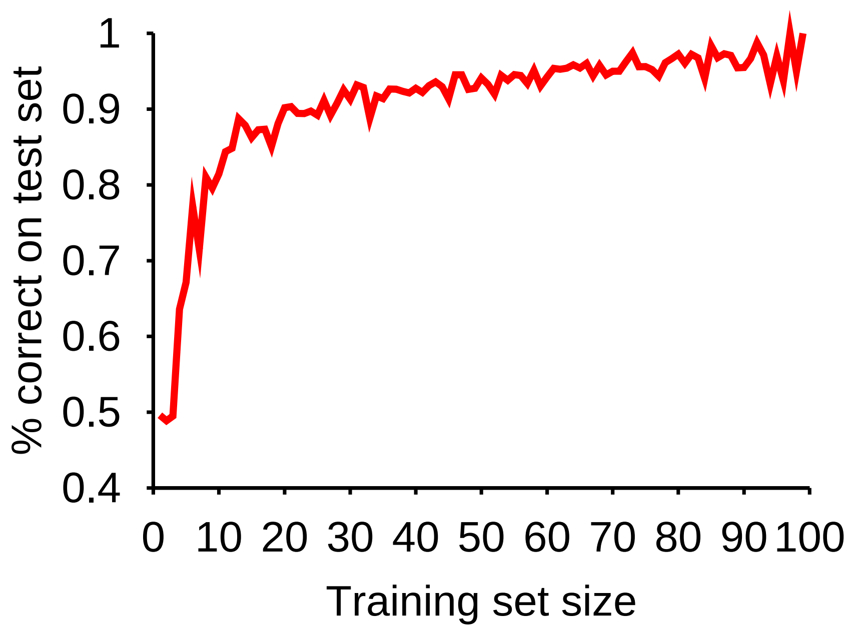
Performance measurement

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)

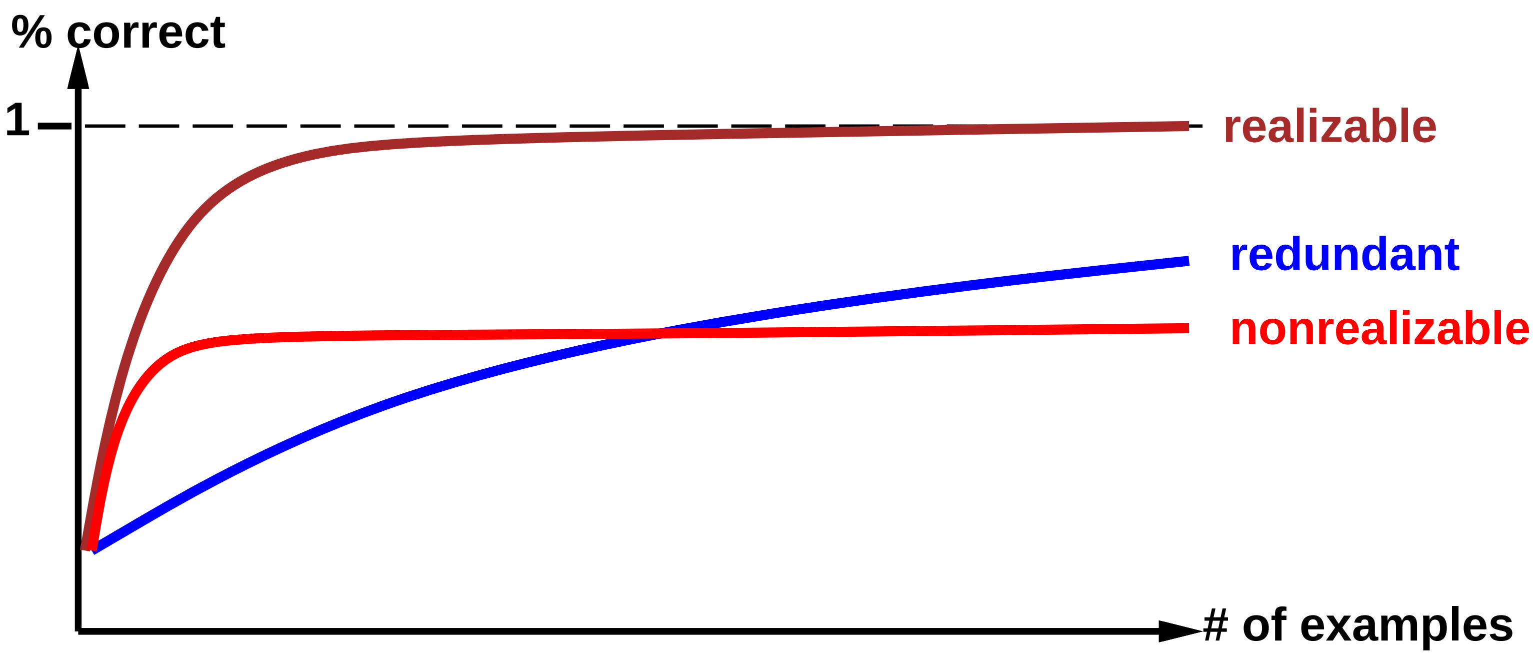
Learning curve $=$ % correct on **test** set as a function of **training** set size



Performance measurement contd.

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 is needed for unknown environments, or for 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 that is approximately consistent with training examples

Decision tree learning is using information gain, or entropy

Learning performance = prediction accuracy measured on test set
– the test set should contain new examples, but with the same distribution