Supervised Learning

Learning paradigms

Reinforcement learning: Agent receives scalar punishment or reward

Supervised learning: Agent receives correct response/behavior from teacher

- e.g., learning a policy
- e.g., learning a conditional probability table

Unsupervised learning: Agent discovers regularities in environment

- e.g., cars and drivers

Varieties of supervised learning

Classification: response is one-of-n

e.g., animal, vegetable, or mineral

e.g., speech recognition in toy

Regression: response is continuous value or vector

e.g., gas price

e.g., reward model, $R(s)$

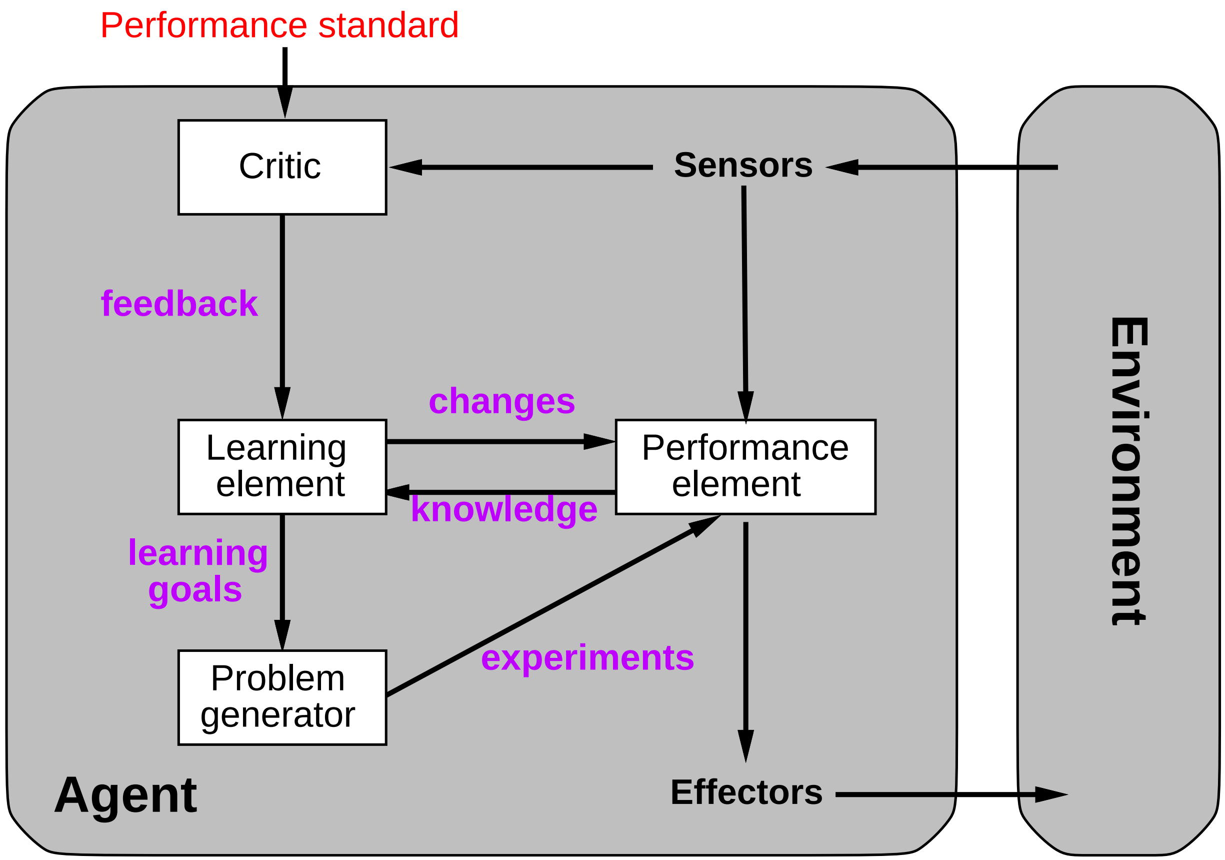
e.g., utility function, $U(s)$

Teaching signal can come from environment (self supervised learning)

e.g., environment model, $T(s, a, s') \equiv P(s'|s, a)$

e.g., stock prediction

Learning agents



Until now, "performance element" was agent

Supervised learning

Simplest form: learn a function from examples (**tabula rasa**, **inductive**)

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 (input, output pairs)

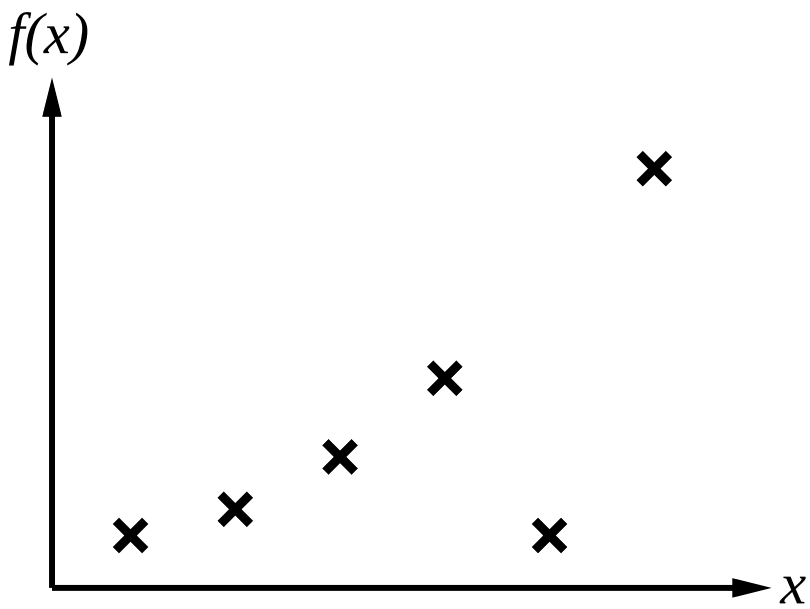
This is a highly simplified model:

- **Ignores prior knowledge**
- **Assumes a deterministic, observable "environment"**
- **Assumes examples are provided**
- **Assumes that the agent wants to learn f—why?**

Inductive learning method

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

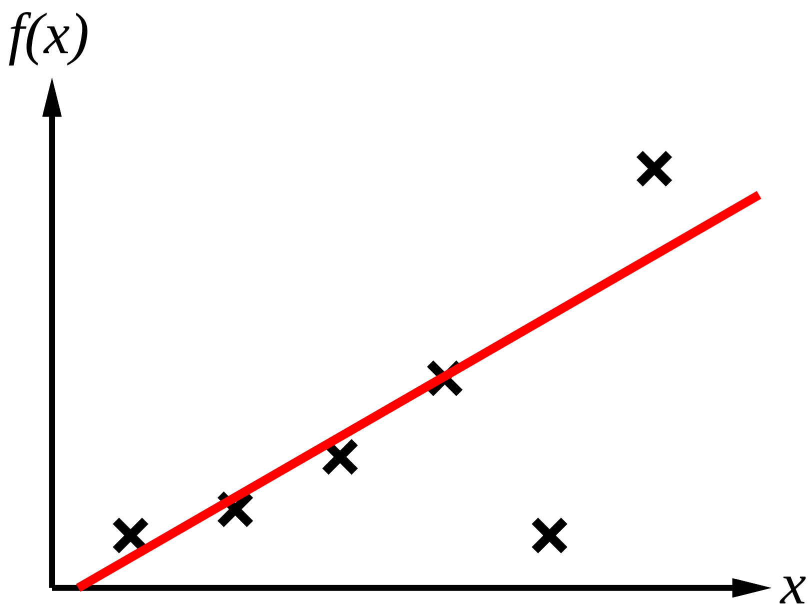
E.g., curve fitting:



Inductive learning method

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

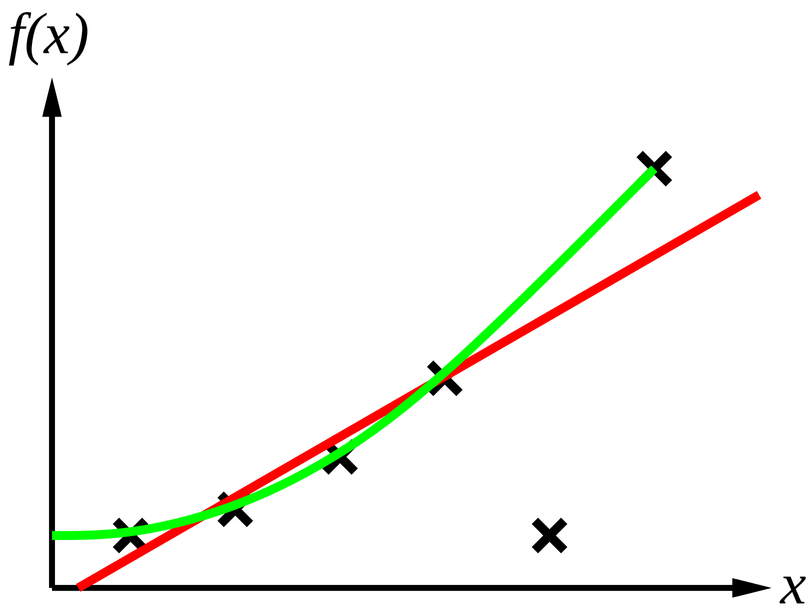
E.g., curve fitting:



Inductive learning method

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

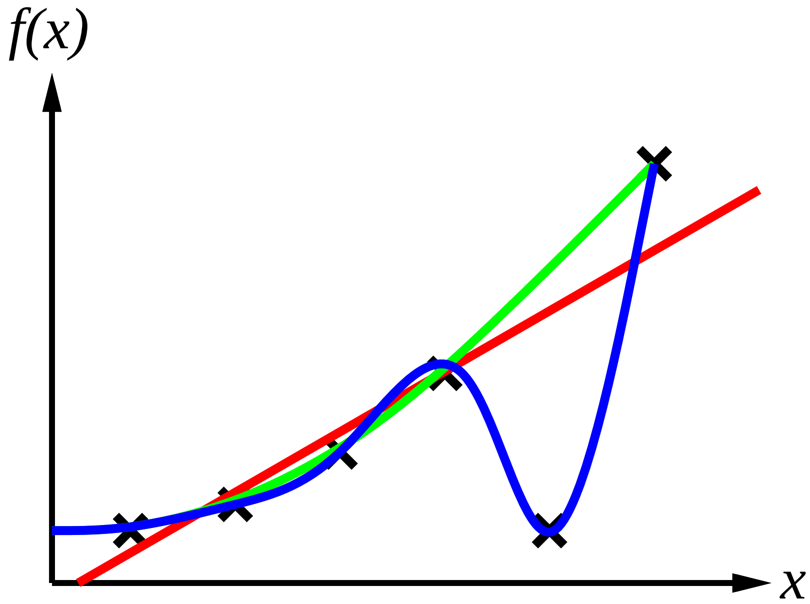
E.g., curve fitting:



Inductive learning method

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

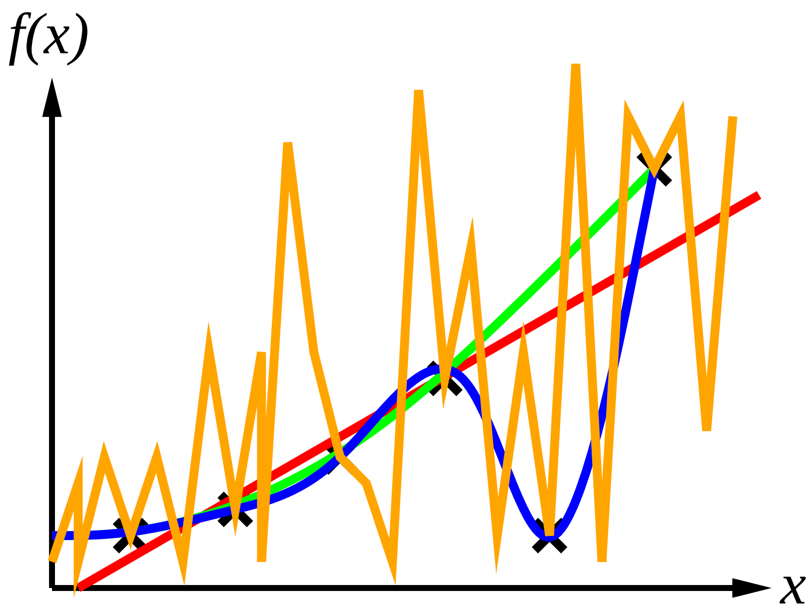
E.g., curve fitting:



Inductive learning method

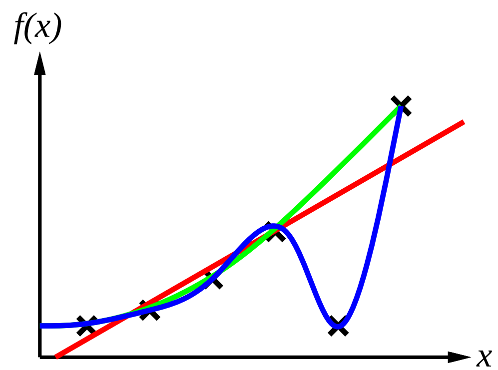
Construct/adjust h to agree with f on training set
(h is consistent if it agrees with f on all examples)

E.g., curve fitting:



Which is the best model?

Hypothesis complexity



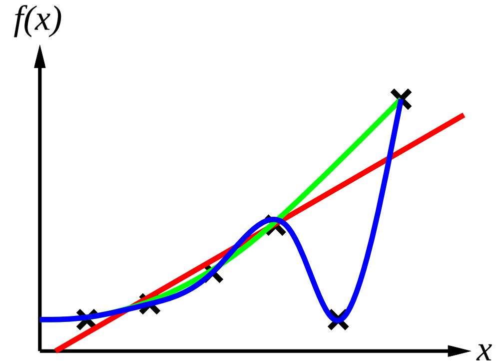
Which hypothesis h you prefer depends on whether and how much nondeterminism (noise) there is in training set.

If h is too simple, it will fail to capture structure in data.

If h is too complex, it will capture noise in the data (a.k.a. overfitting).

Challenge of inductive learning is to find model of appropriate complexity.

Hypothesis complexity (contd.)



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

Alternative: prefer hypothesis that maximizes a combination of consistency and simplicity (consider the trade off)

Ultimate goal is to achieve good generalization to inputs not in training set (a.k.a. out of sample performance)

Attribute-based representations

How do we represent training examples?

attribute-value representation

e.g., dog

number-of-legs=4

tail=true

height=34.2

ears=floppy

hair=short

Attributes can have Boolean, discrete, continuous values

Can represent an example as an attribute vector

e.g., [4 true 34.2 floppy short]

Attribute-based representations (contd.)

Continuous values can be represented discretely
  e.g., (< 20, $20 - 30$, $30 - 40$, $40+$)

Discrete values can be represented with Booleans
  e.g., < 20 can be represented as (1, 0, 0, 0)
  e.g., $30 - 40$ can be represented as (0, 0, 1, 0)

In designing a learning system, representation is often key.

Sample supervised learning problem

Should I wait for table at restaurant?

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)

Sample supervised learning problem (contd.)

Alt = is there a suitable alternative restaurant nearby?

Bar = is there a comfortable bar area to wait in?

Fri = is it Friday or Saturday night?

Hun = are we hungry?

Pat = how many patrons are in the restauarant?

Price = restaurant price range

Rain = is it raining outside?

Res = do we have a reservation?

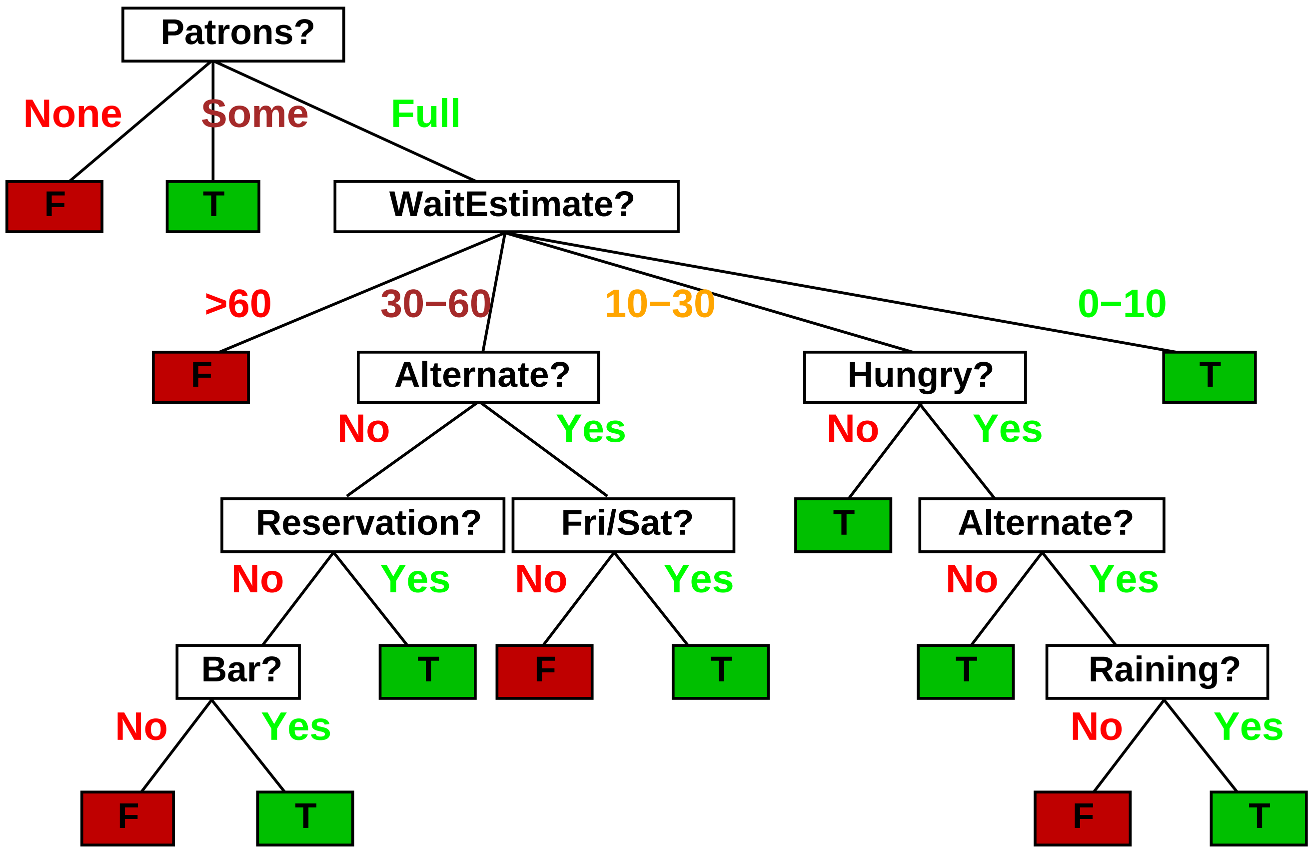
Type = type of food served in restauarnt

Est = estimated wait for a table

Decision trees

One possible representation for hypotheses

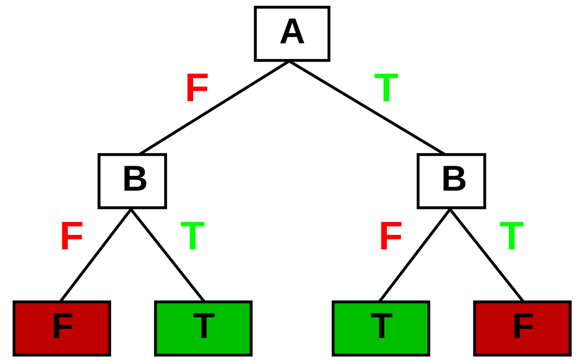
E.g., here is the "true" tree for deciding whether to wait:



Expressiveness

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

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



Trivial to construct a consistent decision tree for any training set with one path to leaf for each example (if $f(x)$ is deterministic).

But will the decision tree generalize well to new examples?
E.g., consider the function $f(A, B) = \neg B$

Goal: construct decision tree that captures regularities of the data.
This tree will tend to be more compact than the full truth table.

Hypothesis spaces

How many distinct decision trees with n Boolean attributes??

Hypothesis spaces

How many distinct decision trees with n Boolean attributes??

$=$ number of Boolean functions

Hypothesis spaces

How many distinct decision trees with n Boolean attributes??

$=$ number of Boolean functions
$=$ number of distinct truth tables with 2^n rows

Hypothesis spaces

How many distinct decision trees with n Boolean attributes??

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

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

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

Expressive (complex) hypothesis space
- increases chance that target function can be expressed
- increases number of hypotheses consistent with training set
  - $\Rightarrow$ may get worse predictions

Hypothesis spaces

Contrast decision trees with conjunctive hypotheses.

How many distinct decision trees with n Boolean attributes??

$=$ 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
$\Rightarrow \quad 3^n$ distinct conjunctive hypotheses

Decision tree learning

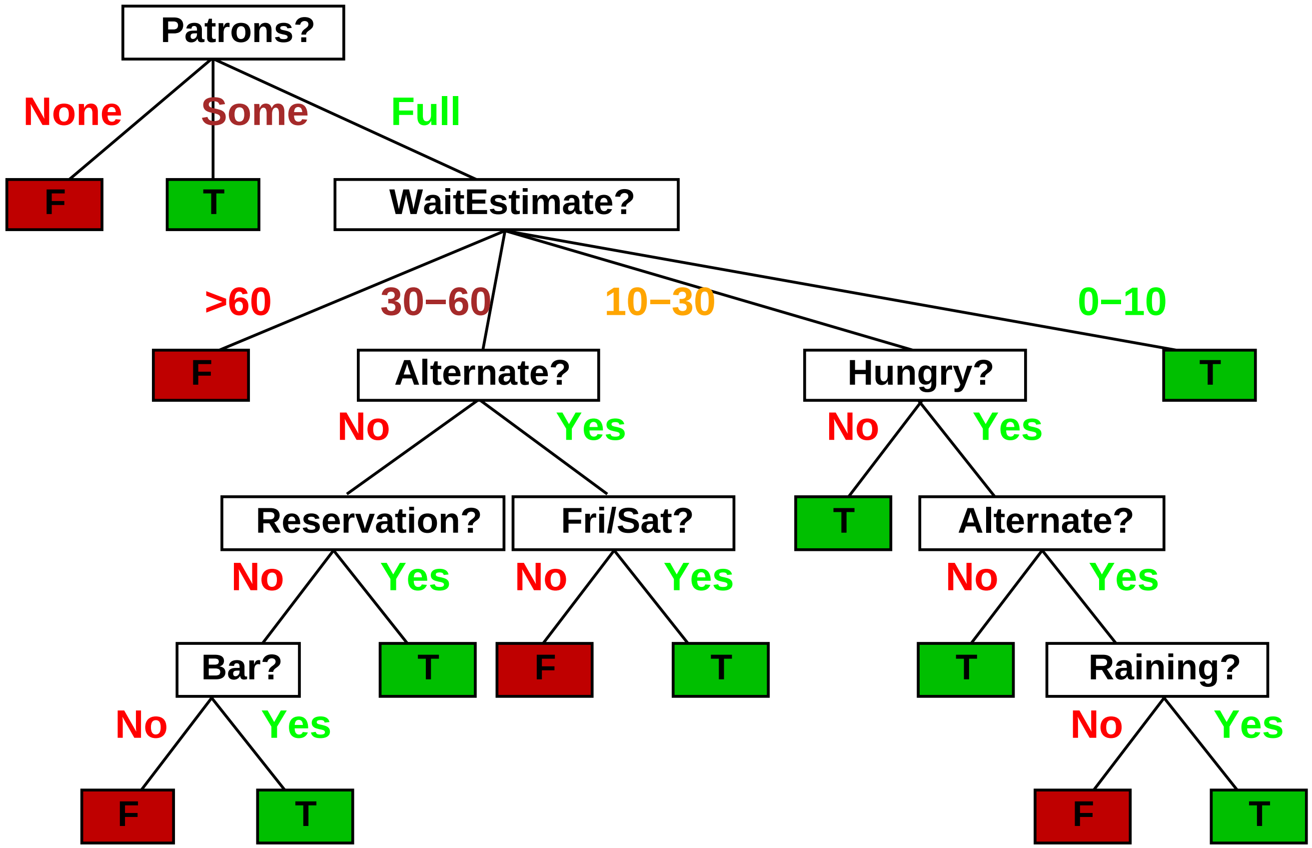
Aim: find a compact tree consistent with the training examples

Idea: choose "most significant" attribute as root of tree, and recurse for each subtree

function DTL(*examples*, *attributes*, *default*) **returns** a decision tree

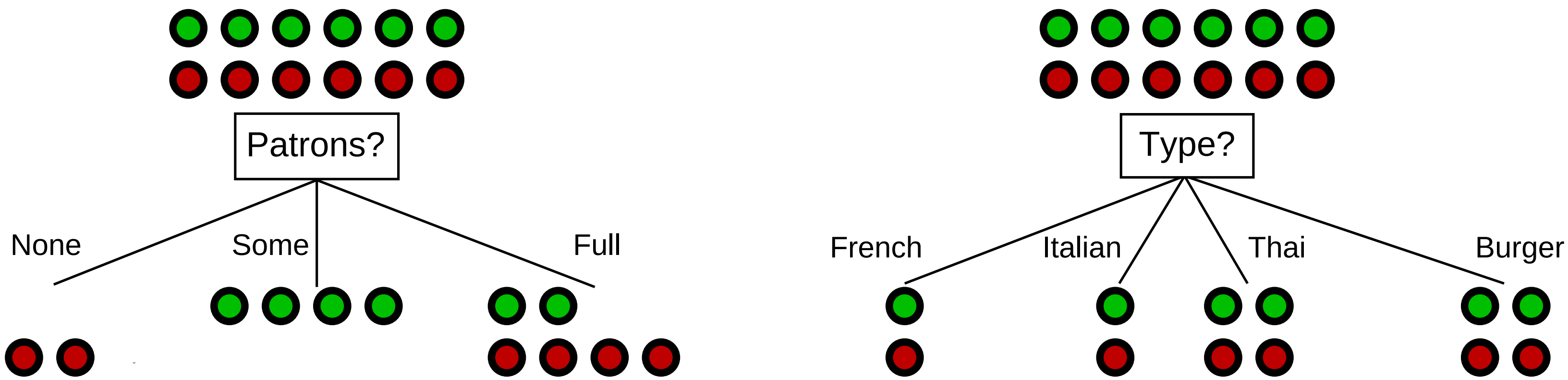
    if *examples* is empty **then return** *default*
    else if all *examples* have the same classification **then return** the classification
    else if *attributes* is empty **then return** MODE(*examples*)
    else
        best ← CHOOSE-ATTRIBUTE(*attributes*, *examples*)
        tree ← a new decision 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*

A sample decision tree



Choosing an attribute

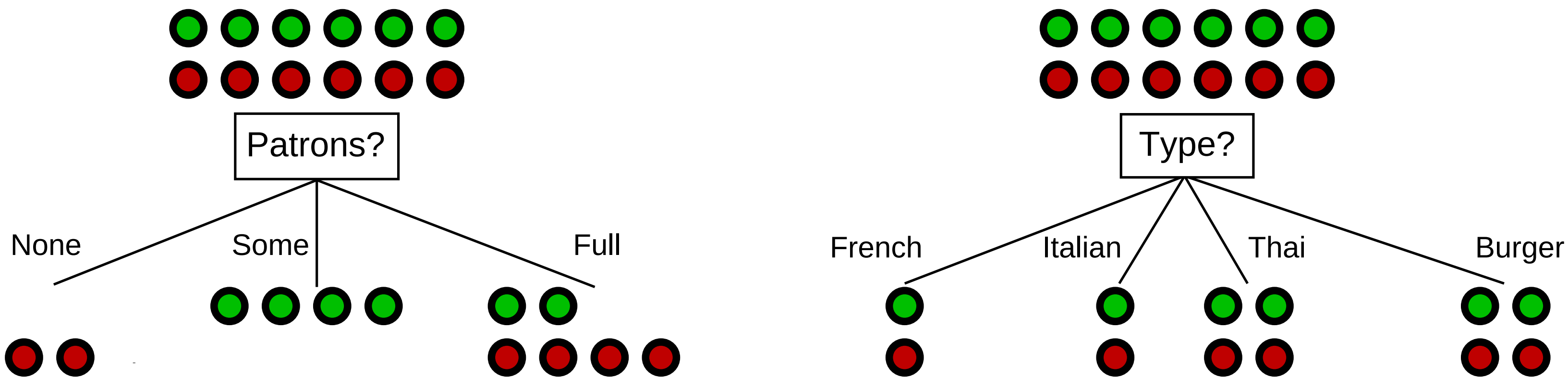
Two possible attributes that could be used to split data:



Which is better?

Choosing an attribute

Two possible attributes that could be used to split data:



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.

Scale: 1 bit $=$ answer to Boolean question with prior $\langle 0.5, 0.5 \rangle$

What if there is a question with 4 possible answers with priors $< .25, .25, .25, .25 >$?

What if there is a question with 4 possible answers with priors $< .5, .5, 0, 0 >$?

What if there is a question with 2 possible answers with priors $< 0, 1 >$? $< 1, 0 >$?

What is function relating information to p if priors are $< p, 1 - p >$?

Information

The information value of an answer given prior $\langle p_1, \ldots, p_n \rangle$ is

$$H(\langle p_1, \ldots, p_n \rangle) = \sum_{i=1}^{n} - p_i \log_2 p_i$$

Information is also called entropy or uncertainty.

Communication metaphor:

I want to pass some information to you over phone line.
We share knowledge of priors.
How many bits of data do I need to send to communicate information?

Information contd.

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

$\Rightarrow$ $H(\langle p/(p+n), n/(p+n)\rangle)$ bits needed to classify a new example

E.g., 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

$\Rightarrow$ $H(\langle p_i/(p_i+n_i), n_i/(p_i+n_i)\rangle)$ bits needed to classify a new example

$\Rightarrow$ **expected** number of bits per example over all branches is

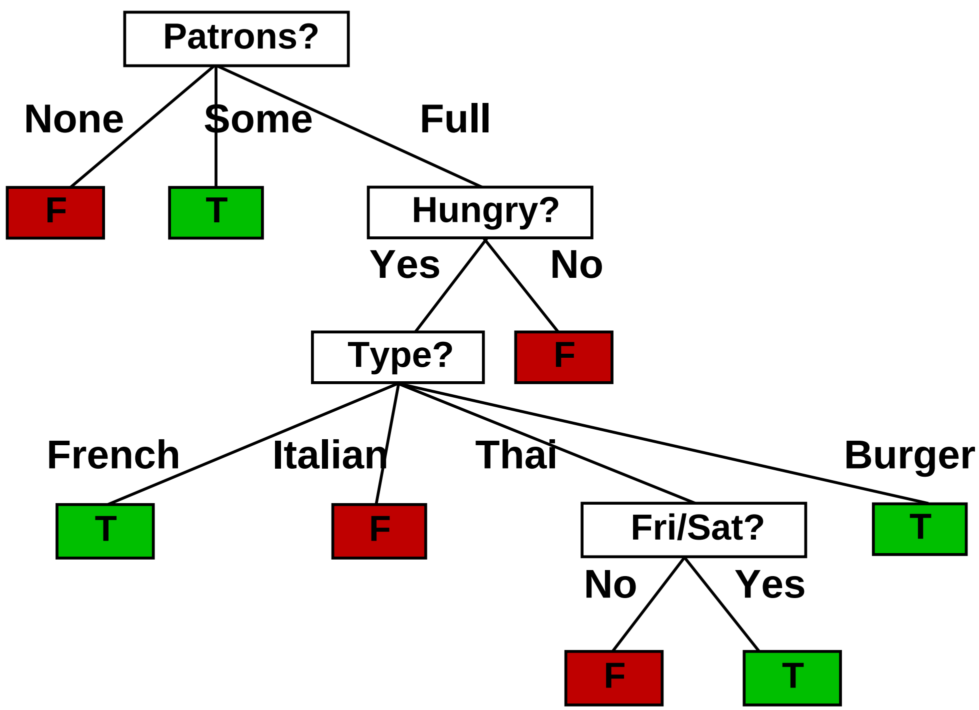
$$\Sigma_i \; \frac{p_i + n_i}{p + n} \, H(\langle p_i/(p_i + n_i), n_i/(p_i + n_i)\rangle)$$

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

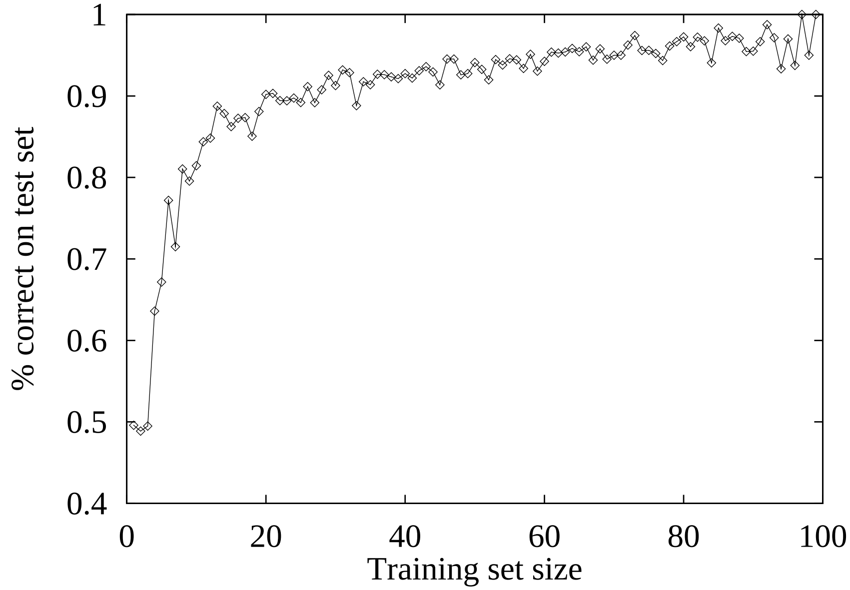
Performance measurement

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

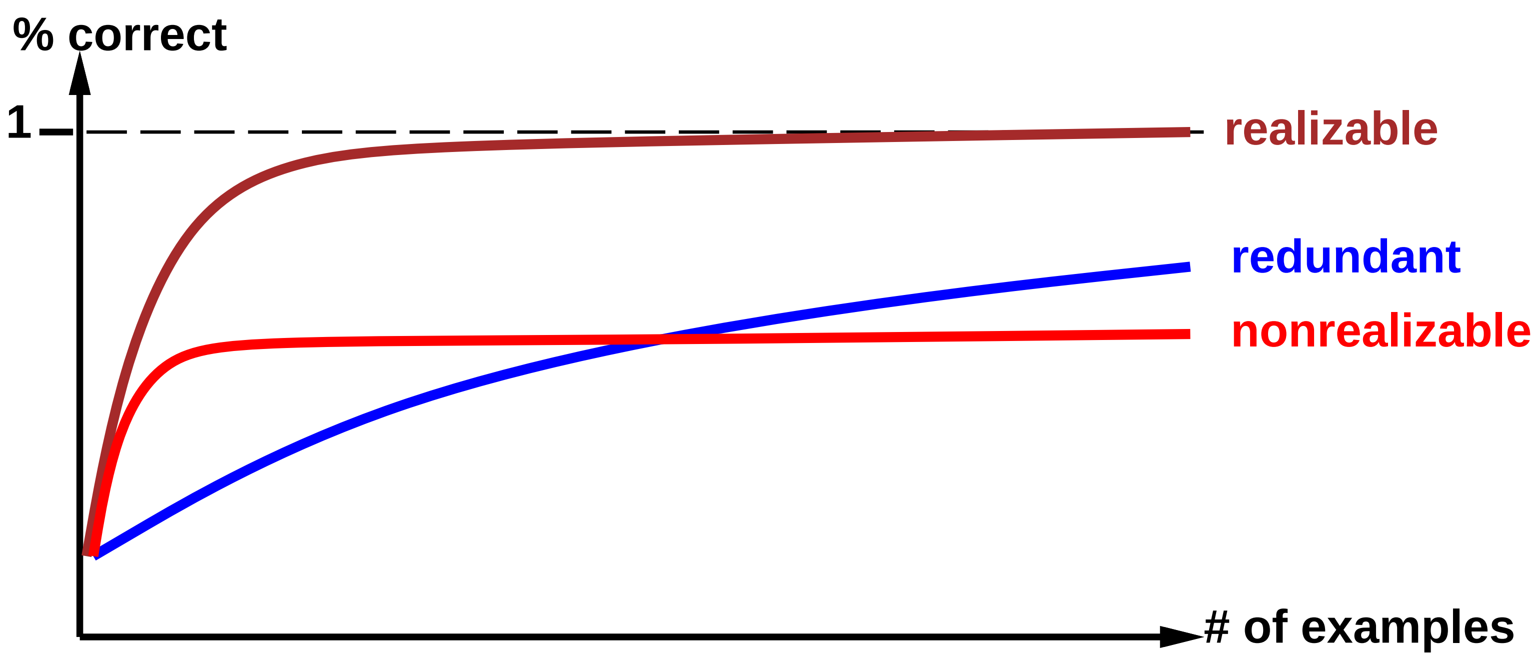


Performance measurement contd.

Learning curve depends on

– realizable (can express target function) vs. non-realizable
  non-realizability can be due to missing attributes
  or restricted hypothesis class (e.g., thresholded linear function)

– redundant expressiveness (e.g., loads of irrelevant attributes)



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