Learning from Observations

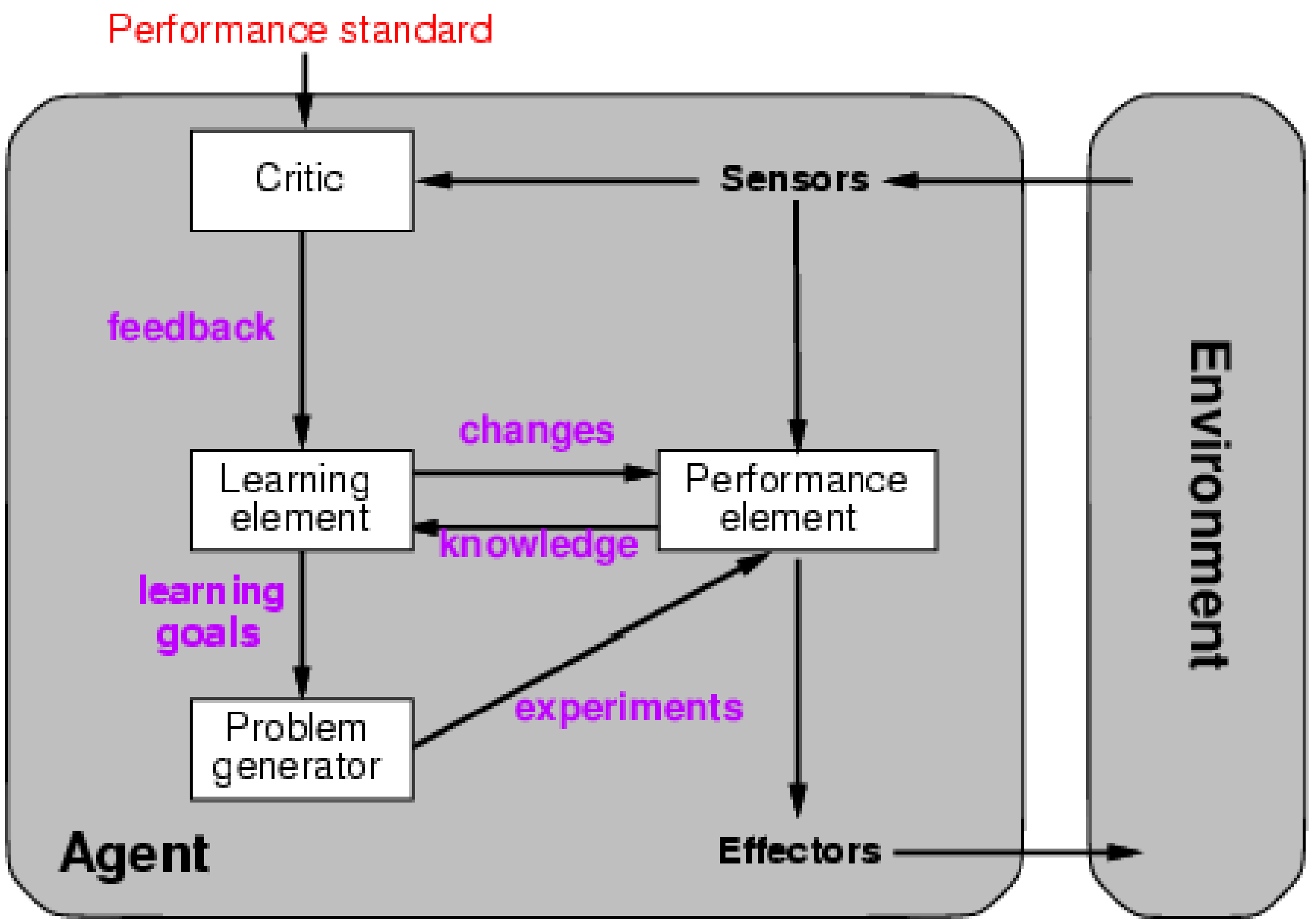
Outline

- Learning agents
- Inductive learning
- Decision tree learning

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

Learning agents



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

Types of Learning

- Supervised learning: correct answer for each example. Answer can be a numeric variable, categorical variable etc.

- Unsupervised learning: correct answers not given – just examples (e.g. – the same figures as above , without the labels)

- Reinforcement learning: occasional rewards

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 $h \approx f$

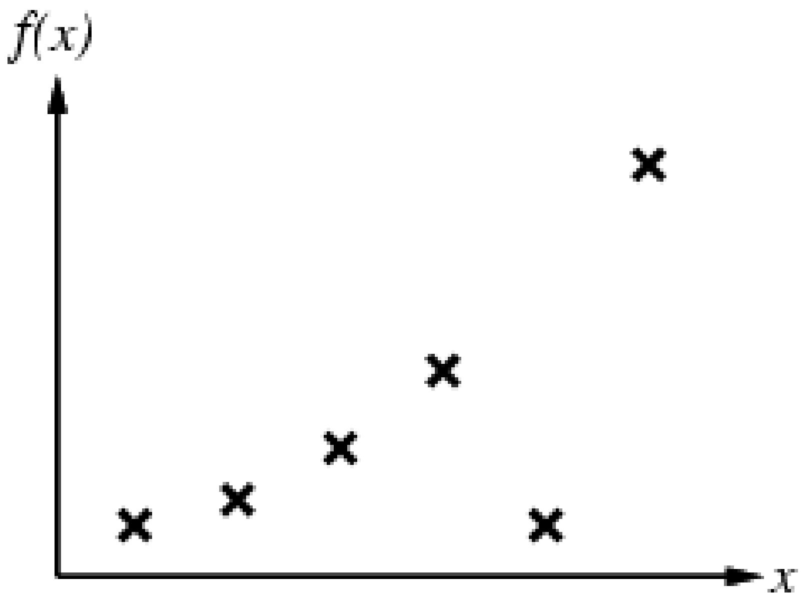
given a training set of examples

(This is a highly simplified model of real learning:

- Ignores prior knowledge
- Assumes examples are given)
-

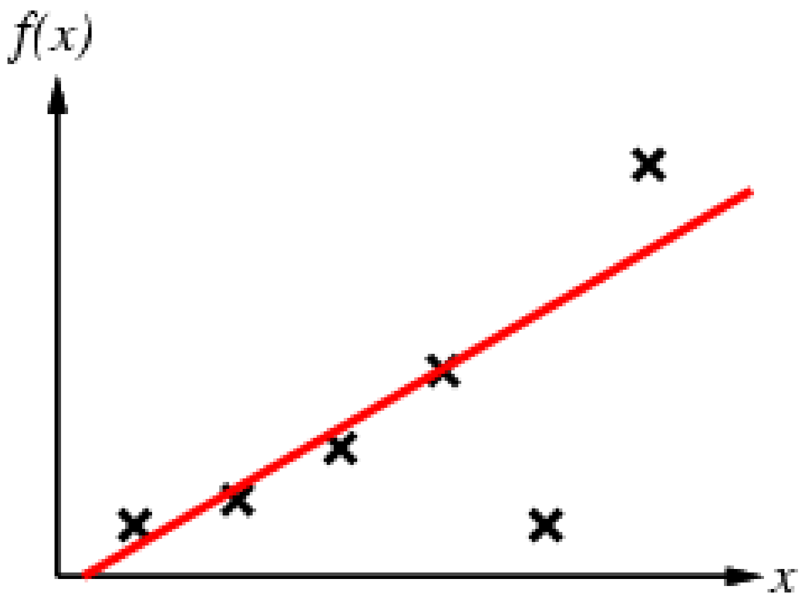
Inductive learning method

- 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:
-



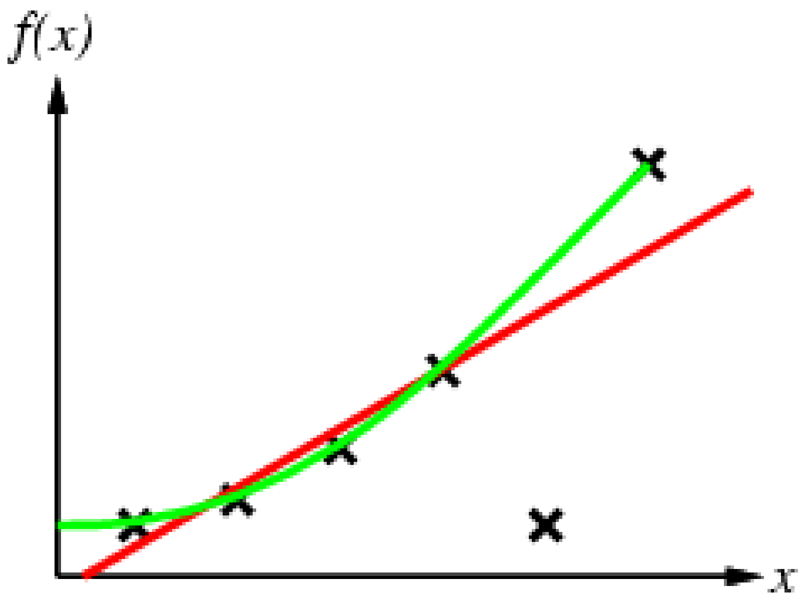
Inductive learning method

- 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:
-



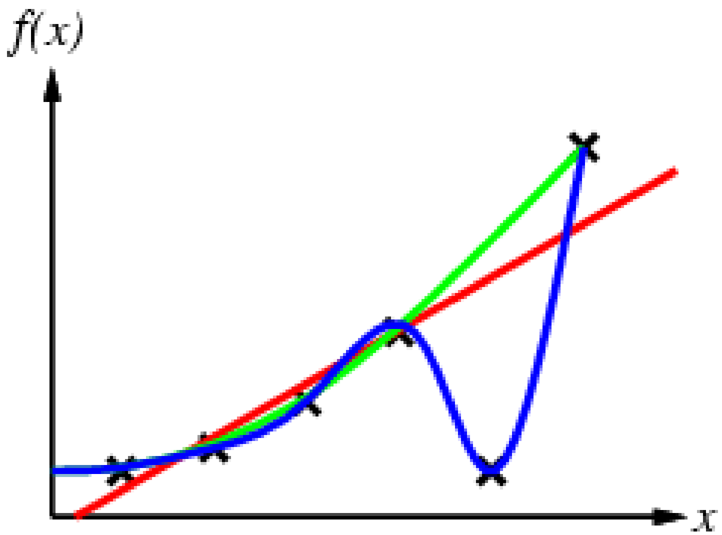
Inductive learning method

- 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:
-



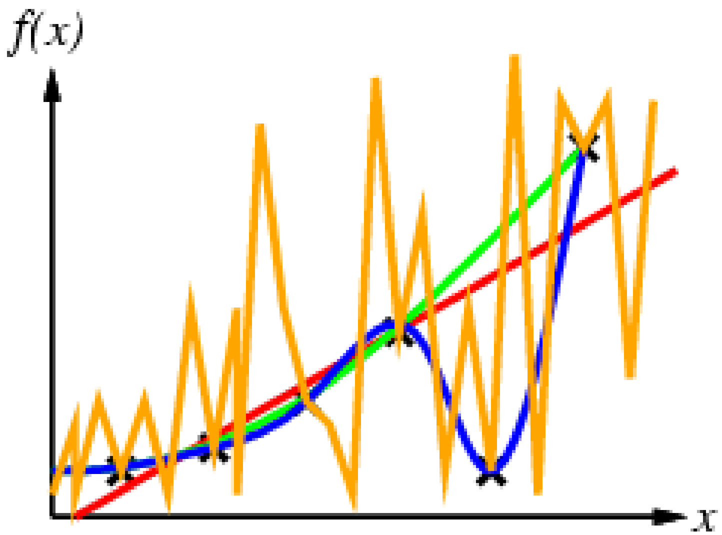
Inductive learning method

- 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:
-



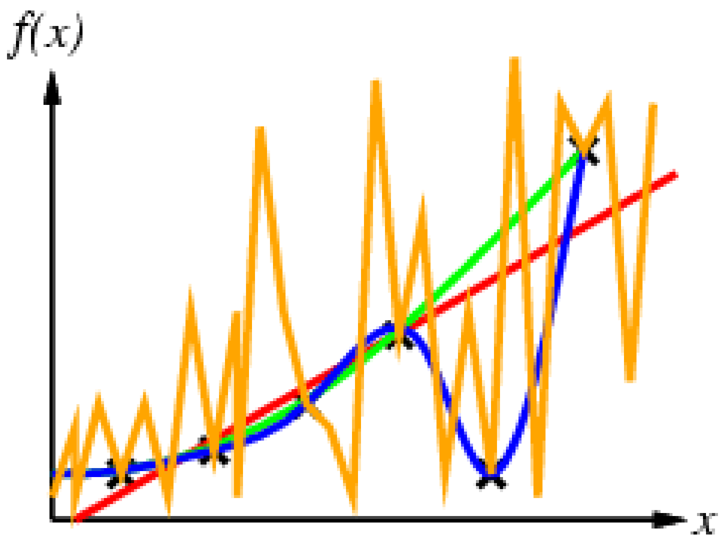
Inductive learning method

- 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: prefer the simplest hypothesis consistent with data

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

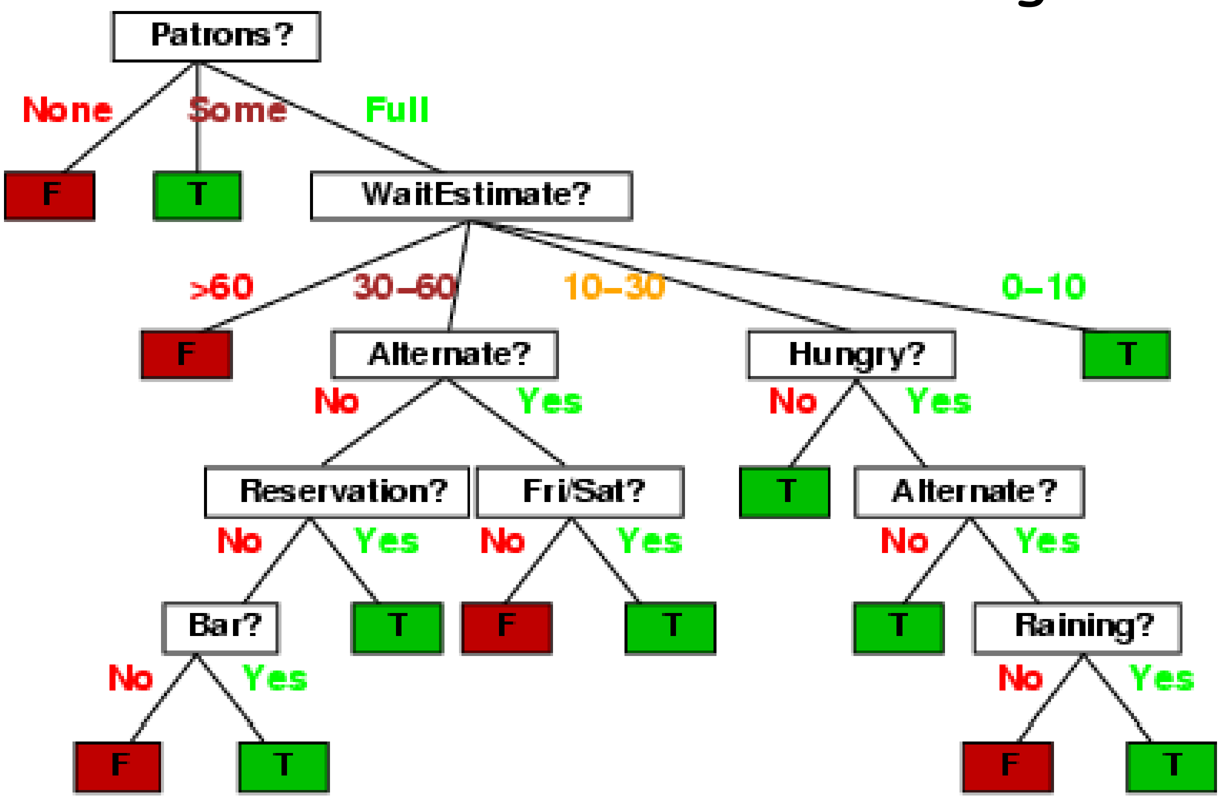
- Examples described by attribute values (Boolean, discrete, continuous)
- E.g., situations where I will/won't wait for a table:

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
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)
-
- The set of examples used for learning is called training set.

Decision trees

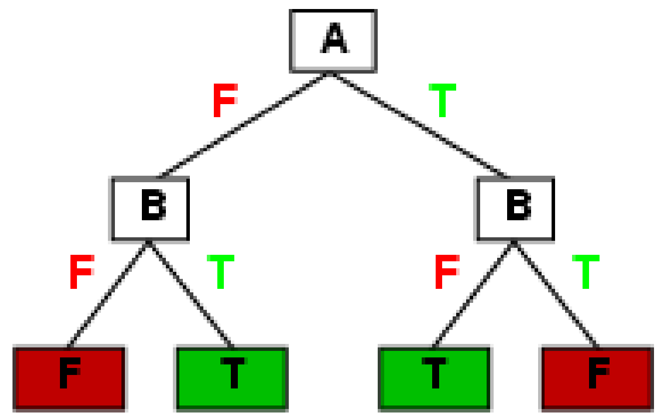
- One possible representation for hypotheses
- E.g., here is the “true” tree for deciding whether to wait:



Expressiveness

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

Finding 'compact' decision trees

- Motivated by Ockham's razor.
- However, finding the smallest decision tree is an unsolved problem (NPc).
- There are heuristics that find reasonable decision trees in most practical cases.

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

How many purely conjunctive hypotheses (e.g., $Hungry \wedge \neg Rain$)?

- 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

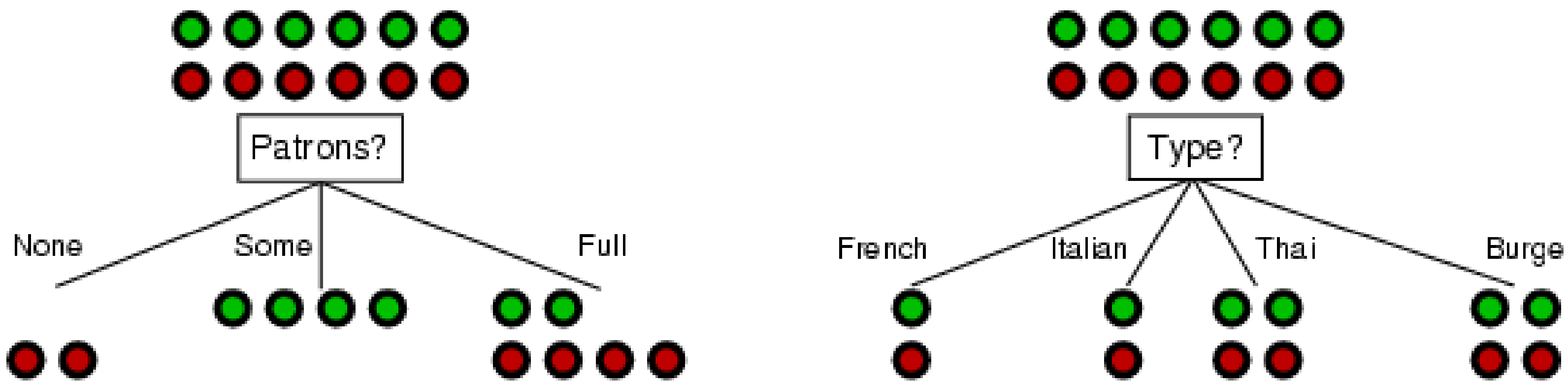
Decision tree learning

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

```
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

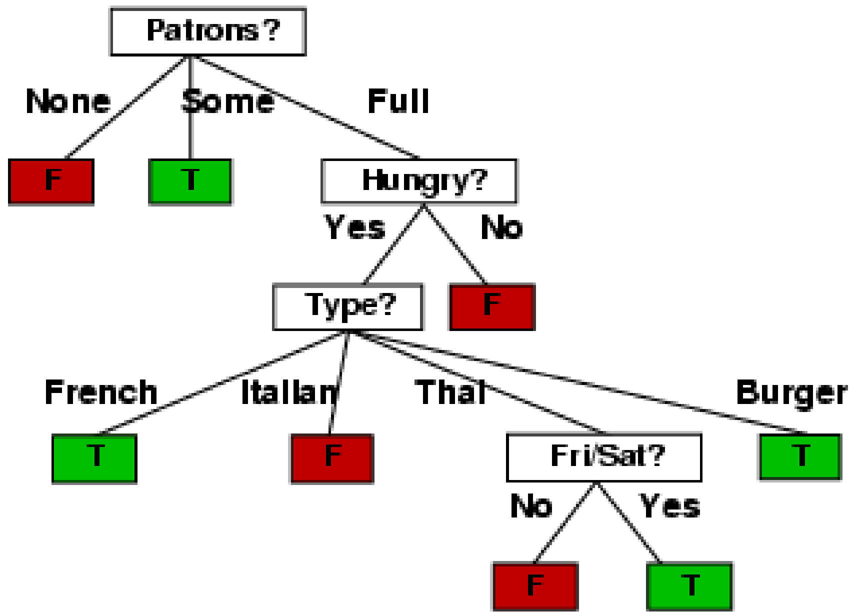
- Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"



- *Patrons?* is a better choice

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

Using information theory

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

  $I(P(v_1), \ldots, P(v_n)) = \Sigma_{i=1} -P(v_i) \log_2 P(v_i)$
- 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

- A chosen attribute *A* divides the training set *E* into subsets $E_1, \ldots, E_v$ according to their values for *A*, where *A* has *v* distinct values.

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

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

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

- Choose the attribute with the largest IG

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):

$$IG(Patrons) = 1 - [\frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I(\frac{2}{6}, \frac{4}{6})] = .541 \text{ bits}$$

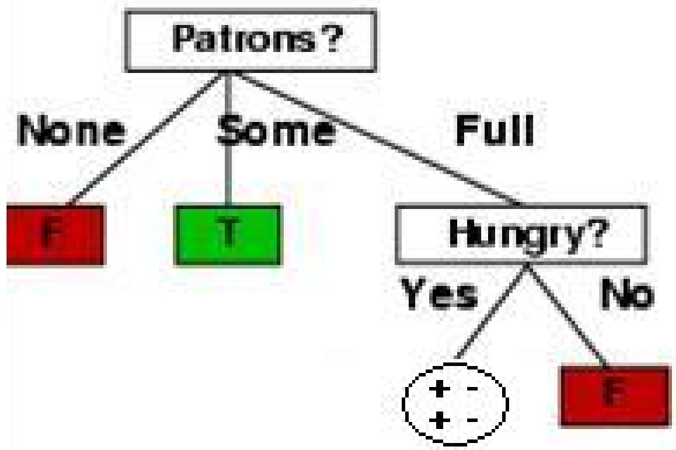
$$IG(Type) = 1 - [\frac{2}{12} I(\frac{1}{2}, \frac{1}{2}) + \frac{2}{12} I(\frac{1}{2}, \frac{1}{2}) + \frac{4}{12} I(\frac{2}{4}, \frac{2}{4}) + \frac{4}{12} I(\frac{2}{4}, \frac{2}{4})] = 0 \text{ bits}$$

Patrons has the highest IG of all attributes and so is chosen by the DTL algorithm as the root

Given *Patrons* as root node, the next attribute chosen is *Hungry?*, with IG(Hungry?) = I(1/3, 2/3) – (2/3*1 + 1/3*0) = 0.252

Next step

Given *Patrons* as root node, the next attribute chosen is *Hungry?*, with IG(Hungry?) = I(1/3, 2/3) – (2/3*1 + 1/3*0) = 0.252



Computational Learning Theory – why learning works

- PAC learning(Probably Approximately Correct)
- This has been a breakthrough in the theory of machine learning.
- Basic idea: A really bad hypothesis will be easy to identify, With high probability it will err on one of the training examples.
- A **consistent** hypothesis will be probably approximately correct.
- Notice that if there are more training example, then the probability of “approximately correct” becomes higher!

Braodening the applicability of Decision Trees

- Missing data
- Multivalued attributes
- Continuous or integer values attributes
- Continuous output attributes

Computational Learning Theory – how many examples are needed

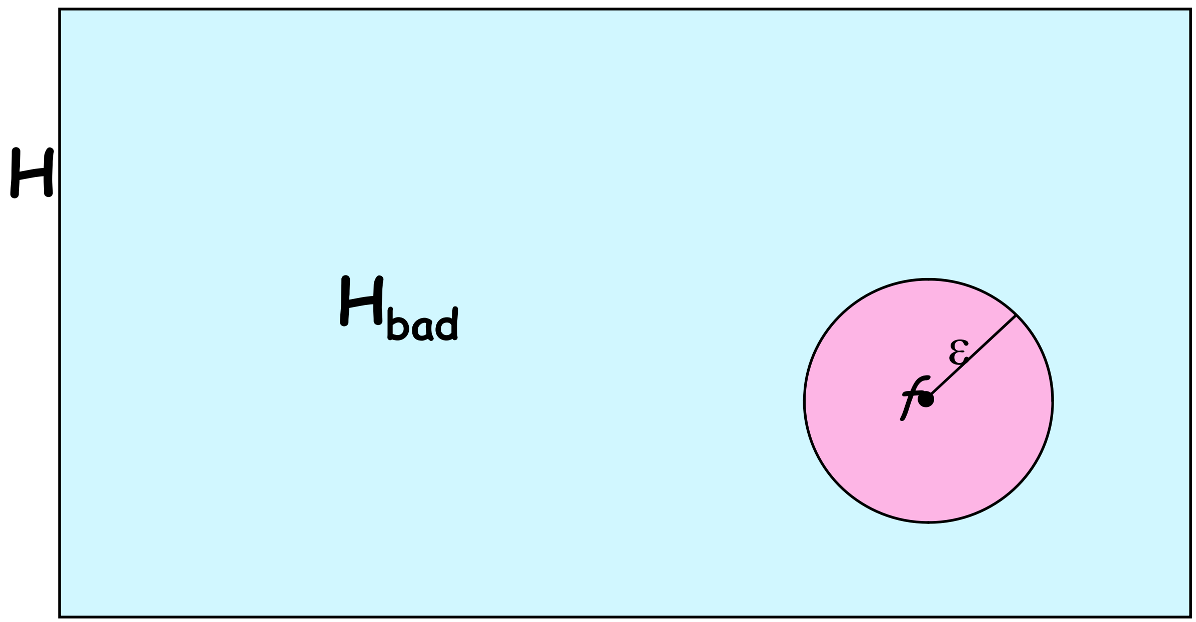
Let:

- X the set of all possible examples
- *D* the distribution from which samples are drawn
- **H** the set of possible hypothesis
- m the number of examples in the training set

And assume that f, the true function is in H.

error

- Error(h) = P(h(x) ≠ f(x) | x drawn from *D*)
- An **approximately correct** hypothesis h, is a hypothesis that satisfies Error(h) < ε



- m ≥ 1/ ε (ln|**H**| - ln δ) , where δ is the probability that H_{bad} contains a hypothesis consistent with all examples

Examples

- Learning a Boolean function;
- Learning a conjunction of n literals
- Learning decision lists

Boolean function

- A general boolean function on *n* attributes can be represented by its truth table
- Size of truth table 2^n

A	B	C	F(A,B,C)
T	T	T	F
T	T	F	T
T	F	T	T
T	F	F	F
F	T	T	F
F	T	F	F
F	F	T	F
F	F	F	T

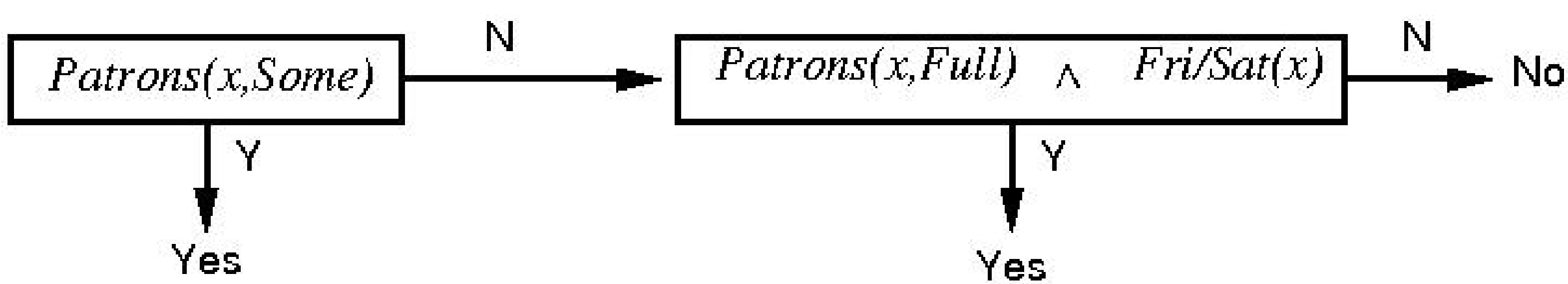
Arbitrary boolean function on 3 attributes

Conjuntion of literals

- A literal is a variable or its negation
- $A \wedge \neg B \wedge C$ is an example of conjunction of literals

Learning Decision Lists

- A decision list consists of a series of tests, each of which is a conjunction of literals. If the tests succeeds, the decision list specifies the value to be returned. Otherwise, the processing continues with the next test in the list.
- Decision lists can represent any boolean function hence are not learnable.



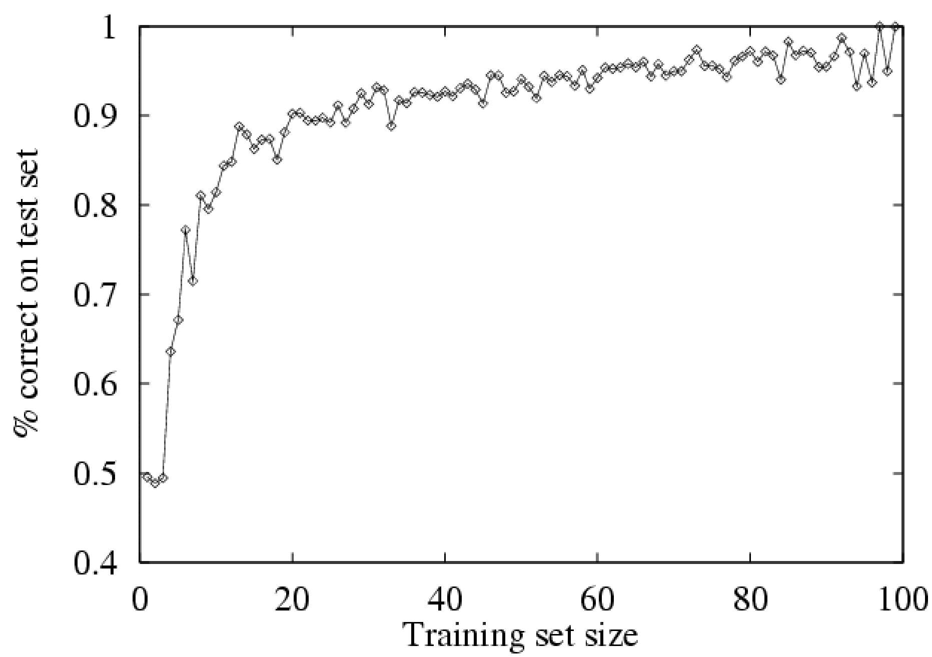
K- DL

- A k-DL is a decision list where each test is restricted to at most k literals.
- K- Dl is learnable!

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 for learning decision trees – restaurant problem

Regression and Classification with Linear Models

- Univariate linear regression
- Multivariate linear regression
- Linear classifiers with a hard threshold
- Linear classifiers with a logistic regression

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