Learning

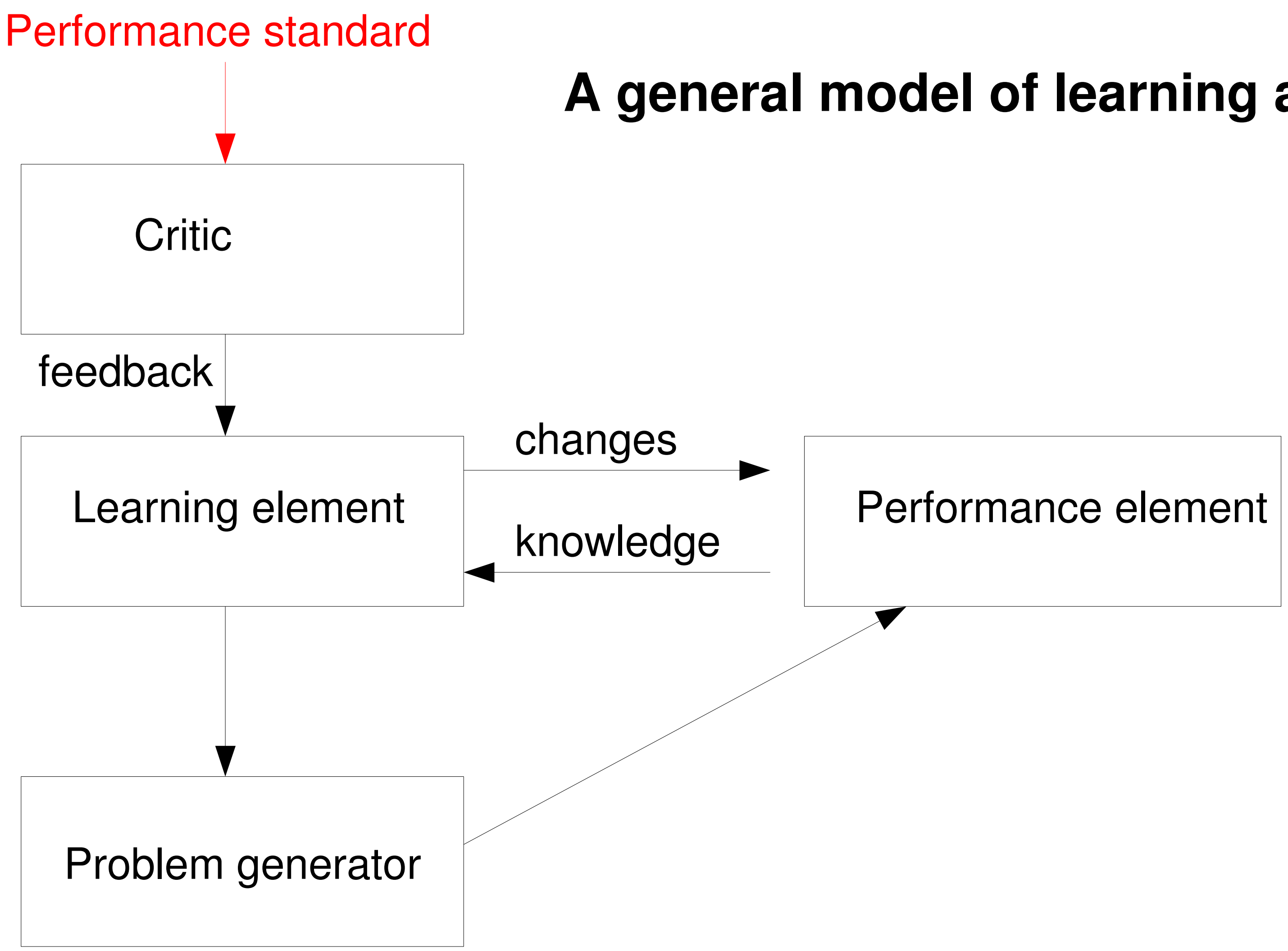
Chapters 11 & 12

Outline

- Introduction
- Decision tree
- Minimum size decision tree

Introduction

A general model of learning agents



Introduction

- Types of learning
  - Supervised learning
    - correct answers for each example
  - Unsupervised learning
    - correct answers not given
  - Reinforcement learning
    - occasional rewards

- Scope of this lecture: supervised learning

Introduction

- Type of learning (cont'd)
  - Inductive -- Use specific examples to reach general conclusions
  - Analogical -- Determine correspondence between two different representations
  - Etc.
- Focus of this lecture: inductive learning

Introduction

- Simplest form of inductive learning: 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 $h \approx f$

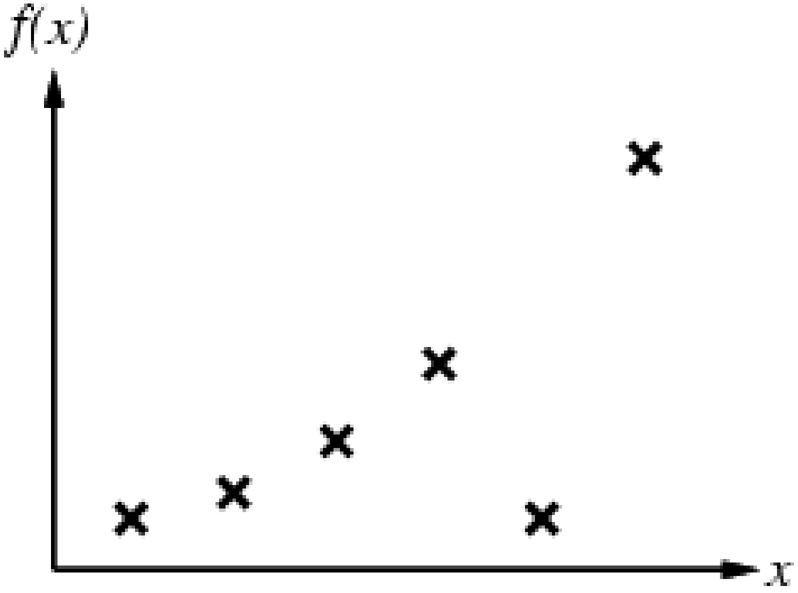
given a training set of examples

(This is a highly simplified model of real learning:

- Ignores prior knowledge
- Assumes examples are given)

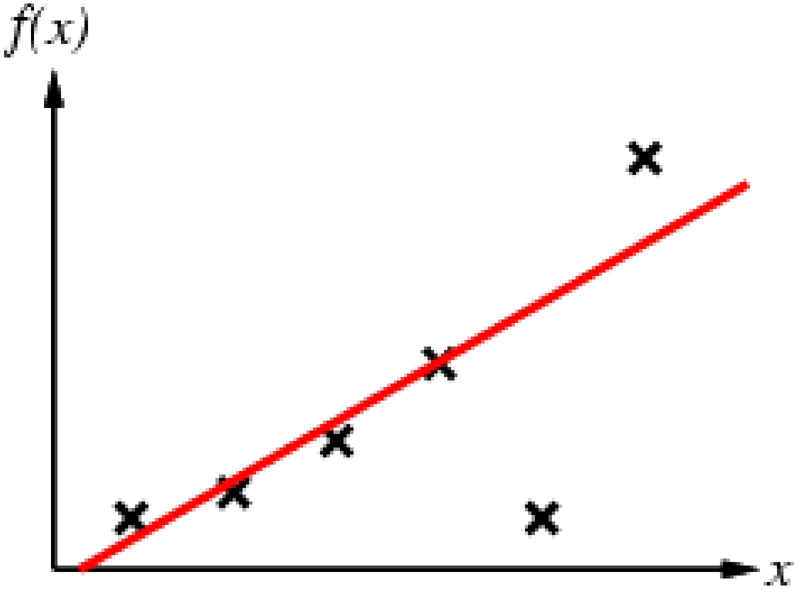
Introduction

- 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:



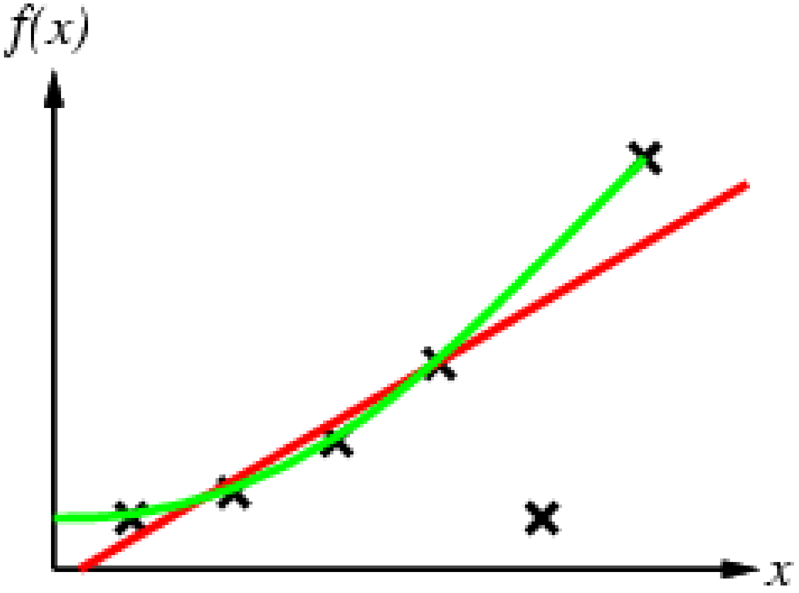
Introduction

- 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:



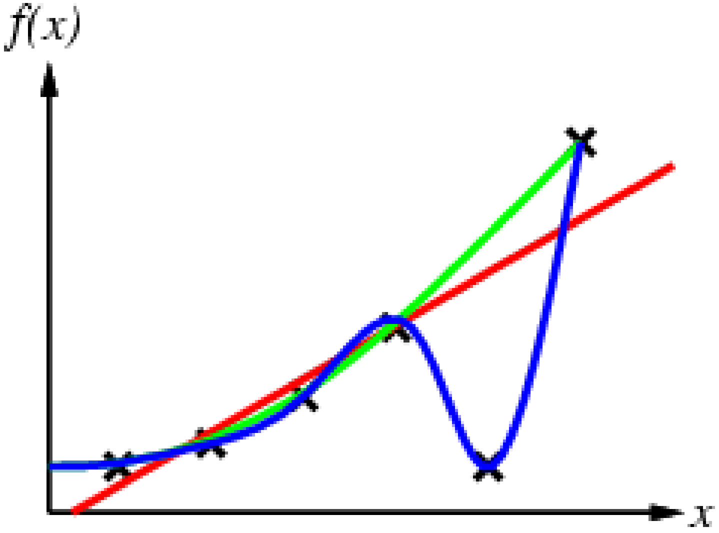
Introduction

- 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:



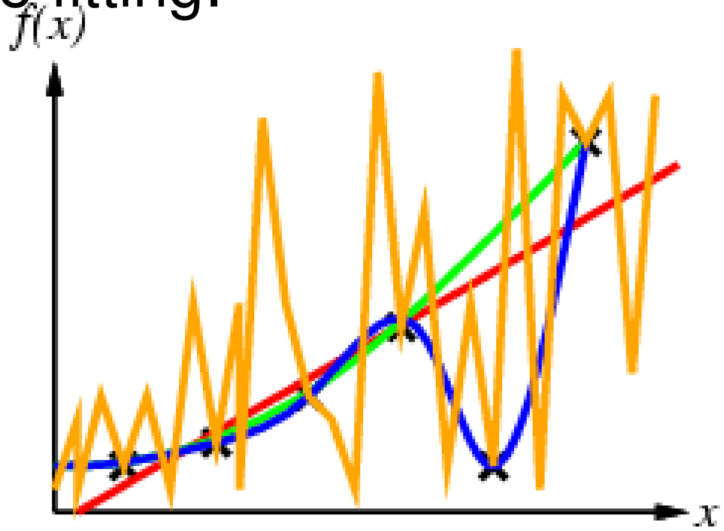
Introduction

- 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:



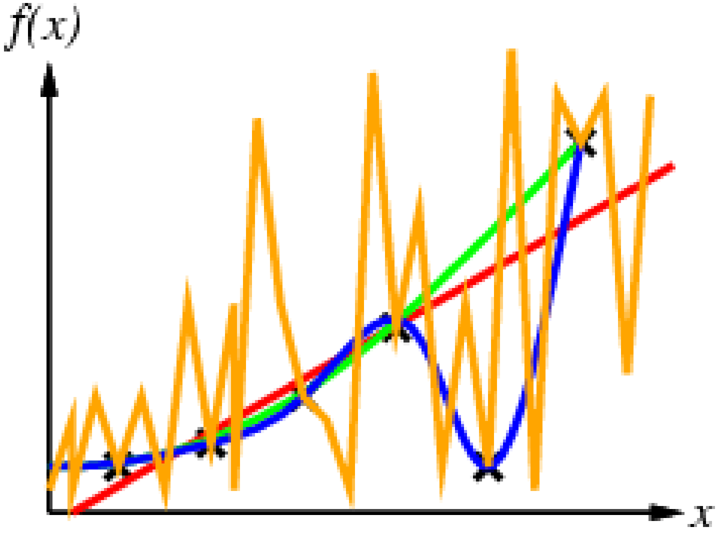
Introduction

- 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:



Introduction

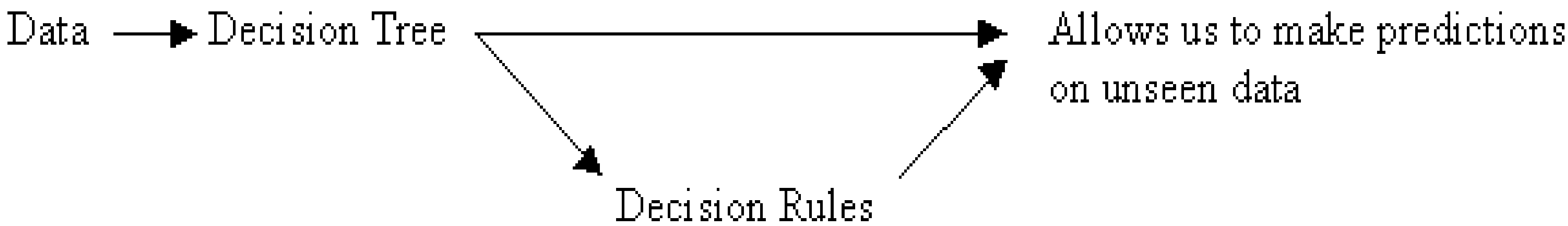
- Construct/adjust *h* to agree with *f* on training set
- (*h* is consistent if it agrees with *f* on all examples)
- E.g., curve fitting:



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

Decision tree (1)

Construed by looking for regularities in data



- Input: a training set of positive and negative examples of a concept
    - Each example has a set of features/attributes
- Output: a tree-graph description, and eventually rules
- Useful for classifying whether future examples are positive or negative.

Decision tree (2)

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)

– WaitEstimate: estimated waiting time (0-10, 10-30, 30-60, >60)

University of Irvine machine learning repository

http://mlr.cs.umass.edu/ml/

- Data sets repository
  - Received from different sources (health care database, census bureau center, etc)
  - Used to test the learning algorithms
- "Census Income" dataset
  - Prediction task is to determine whether a person makes over 50K a year.
  - Attributes: education; marital-status; occupation; race; sex; native-country; etc.

Decision tree (3)

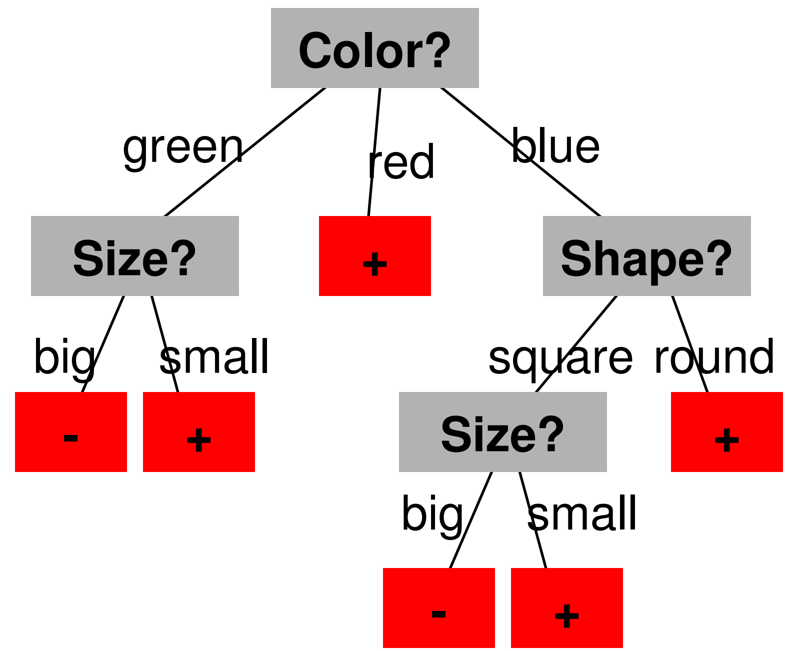
- Examples described by attribute values (e.g. Boolean, discrete)
- E.g., situations where I will/won't wait for a table:

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)

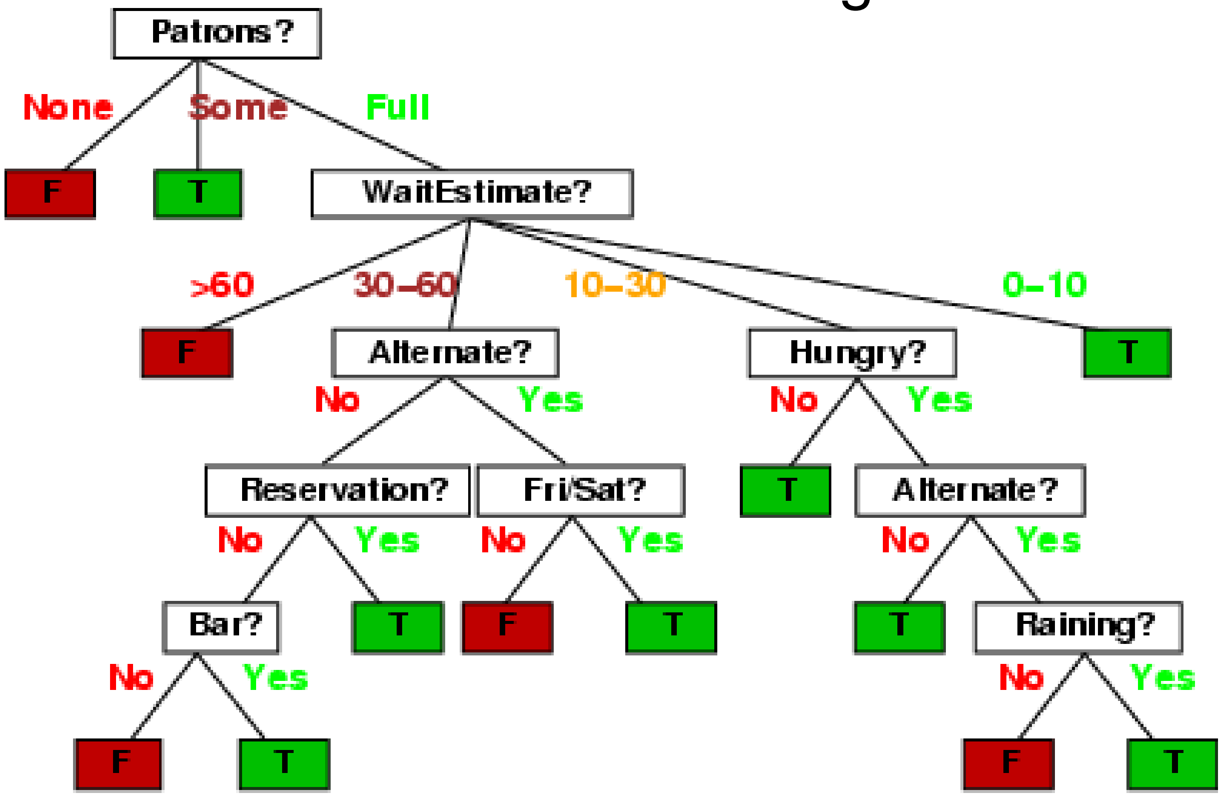
Decision tree (4)

- A **decision tree** is a tree where
  - each **non-leaf** node is associated with an attribute (feature)
  - each **leaf** node is associated with a classification (T or F, + or -, POS or NEG)
  - each **arc** is associated with one of the possible values of the attribute at the node where the arc is directed from



Decision tree (5)

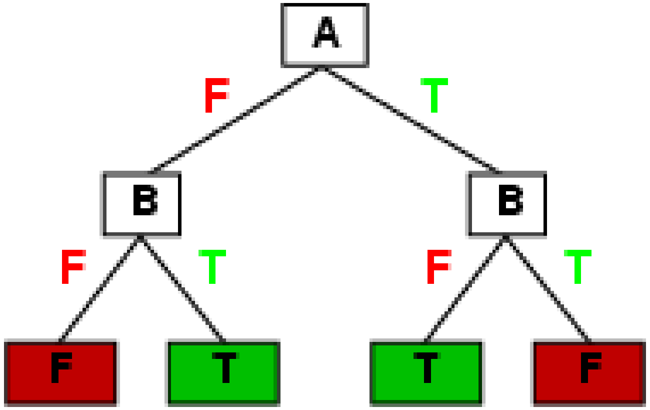
- One possible representation for hypotheses
- E.g., here is the full tree for deciding whether to wait:



Decision tree (6)

- 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



Exercise

- Give decision trees to represent the following boolean functions:

  $A \wedge \neg B$

  $A \vee [B \wedge C]$

  $[A \wedge B] \vee [C \wedge D]$

  $A \vee \neg A$

  A -> (B ->A)

  Any comments?

Minimum size decision tree (1)

- A decision tree helps identify an hypothese consistent with the data
- But we said earlier that inductive learning looks for the simplest hypothesis
- In the present context, this means building the minimum size decision tree
- As you'll see, here things become tricky!

Minimum size decision tree (2)

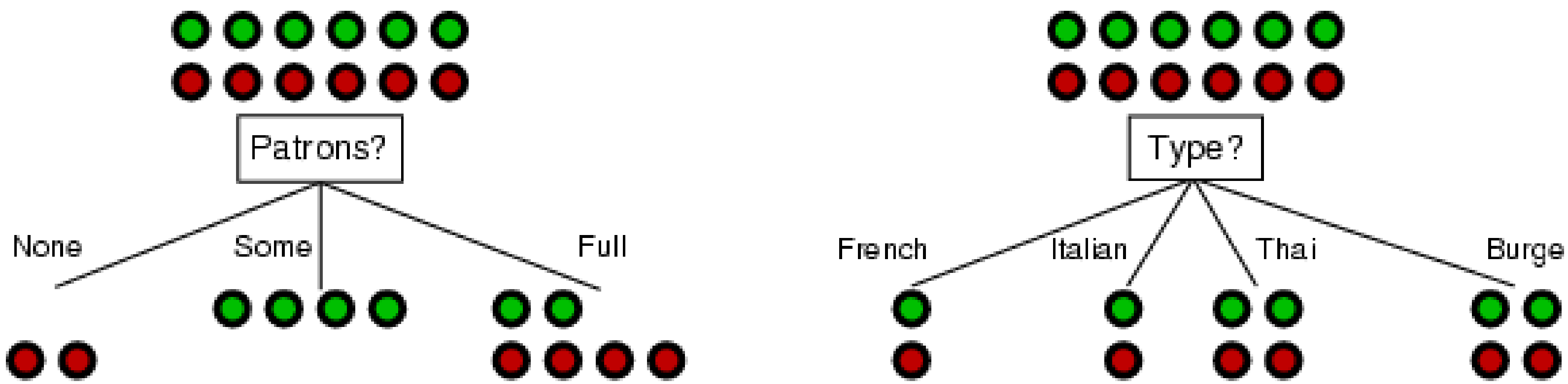
- The key problem is choosing which attribute to split a given set of examples.
- Some possibilities are:
  - **Random:** Select any attribute at random
  - **Least-Values:** Choose the attribute with the smallest number of possible values (fewer branches)
  - **Most-Values:** Choose the attribute with the largest number of possible values (smaller subsets)
  - **Max-Gain:** Choose the attribute that has the largest expected information gain, i.e. select attribute that will result in the smallest expected size of the subtrees rooted at its children.
- The ID3 algorithm uses the Max-Gain method of selecting the best attribute.

Minimum size decision tree (3)

- ID3: A greedy algorithm for Decision Tree Construction developed by Ross Quinlan, 1987
- Consider a smaller tree a better tree
- Top-down construction of the decision tree by recursively selecting the "best attribute" to use at the current node in the tree, based on the examples belonging to this node.
  - Once the attribute is selected for the current node, generate children nodes, one for each possible value of the selected attribute.
  - Partition the examples of this node using the possible values of this attribute, and assign these subsets of the examples to the appropriate child node.
  - Repeat for each child node until all examples associated with a node are either all positive or all negative.

Minimum size decision tree (4)
Choosing a good attribute

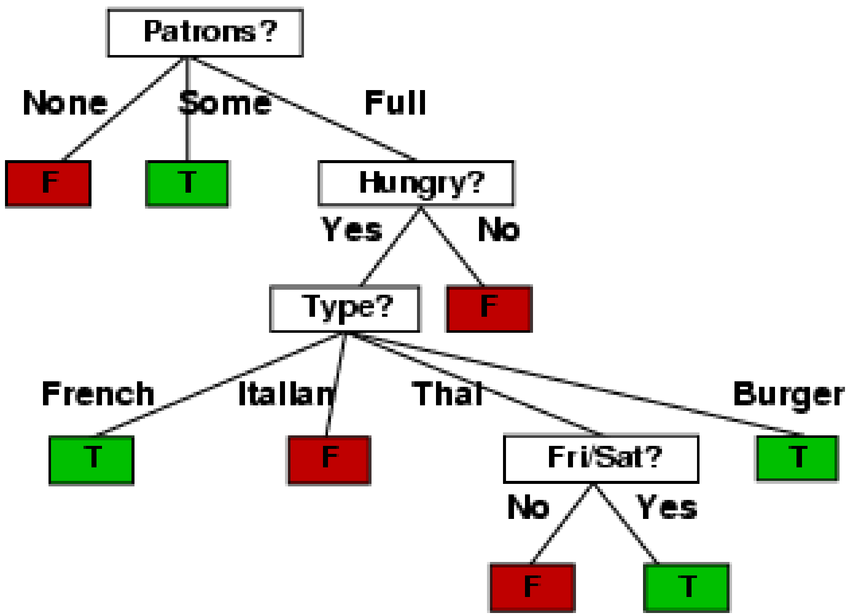
- Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"



- Splitting on *Patrons?* is a better choice
- It is likely this will result in a less complex graph. Can you see why?

Minimum size decision tree (4)

- Decision tree obtained using the ID3 algorithm



- Substantially simpler than “true” tree---a more complex hypothesis isn’t justified by small amount of data

Entropy

- Given an arbitrary categorization, C into categories c1, ..., cn, and a set of examples, S, for which the proportion of examples in ci is pi, then the entropy of S is:

$$Entropy(S) = \sum_{i=1}^{n} -p_i \, log_2(p_i)$$

- Boolean classification (+ and -)

$$Entropy(S) = -p_+ \, log_2(p_+) - p_- \, log_2(p_-)$$

Intuitively, this calculates the « disorder » in the data

Information gain

- The information gain for an attribute is the expected reduction of entropy if the examples were to be partitioned according to that attribute
- This is calculated using the following equation

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

- Given a node, you choose to split on the attribute with the greatest information gain
- No need to memorize these equations. Just be aware they exist!

Exercise

We want to predict the outcome of the next tennis match between the two top-ranked players: Federera and his main rival Nadale. From the official website of Federera, we collect the following (assumedly representative) dataset.

Time	Match type	Court surface	Outcome
Morning	Master	Grass	F
Afternoon	Grand slam	Clay	F
Night	Friendly	Hard	F
Afternoon	Friendly	Mixed	N
Afternoon	Master	Clay	N
Afternoon	Grand slam	Grass	F
Afternoon	Grand slam	Hard	F
Afternoon	Grand slam	Hard	F
Morning	Master	Grass	F
Afternoon	Grand slam	Clay	N
Night	Friendly	Hard	F
Night	Master	Mixed	N
Afternoon	Master	Clay	N
Afternoon	Master	Grass	F
Afternoon	Grand slam	Hard	F
Afternoon	Grand slam	Clay	F

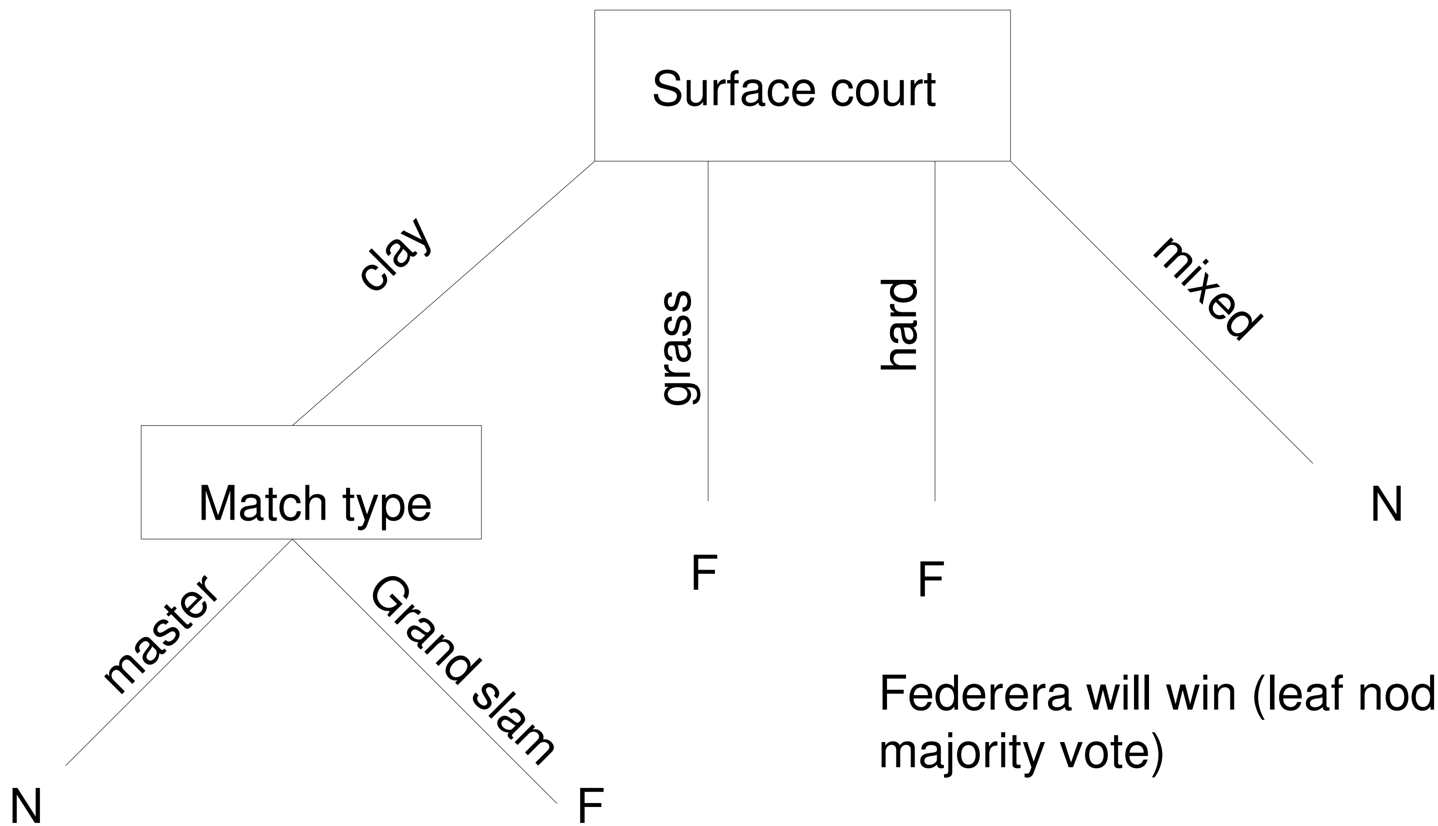
Construct a decision tree based on these data (choose attributes at random)

The next match is a Grand slam's match on clay court surface, and takes place in the afternoon. Predict the outcome of the match using the above decision tree.

Solution

Decision tree (using ID3, entropy)

The outcome is F if Federera wins, and N otherwise



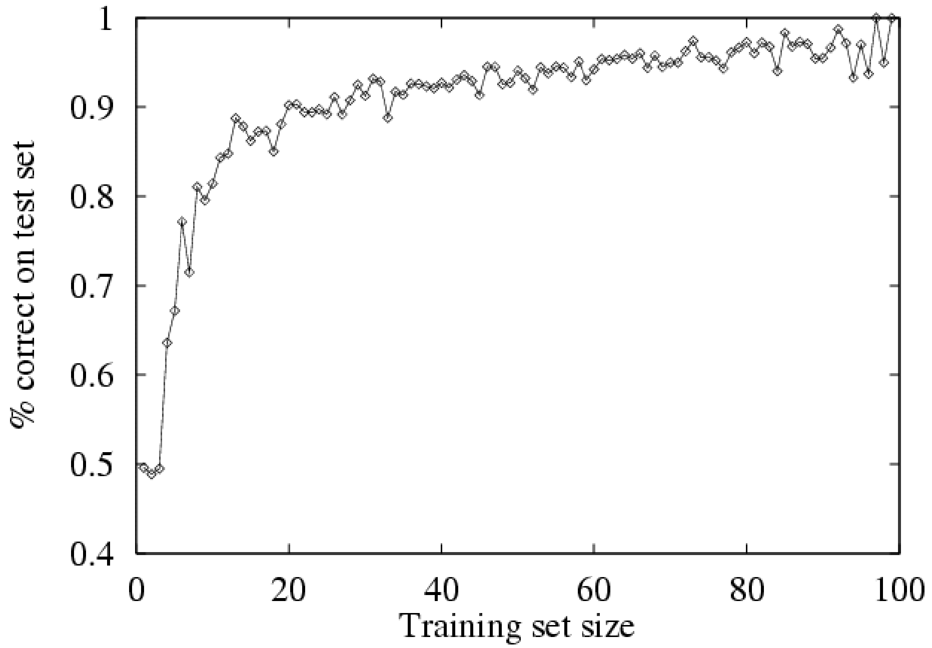
Federera will win (leaf node classified by majority vote)

Performance measurement

- How do we know that $h \approx f$?
  - Use theorems of computational/statistical learning theory
  - 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



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