CSE 537 Fall 2015

LEARNING FROM EXAMPLES

AIMA CHAPTER 18 (1-3)

Instructor: Sael Lee

Slides are mostly made from AIMA resources,
Andrew W. Moore's tutorials: http://www.cs.cmu.edu/~awm/tutorials and
Bart Selman's Cornell CS4700 decision tree slides

LEARNING

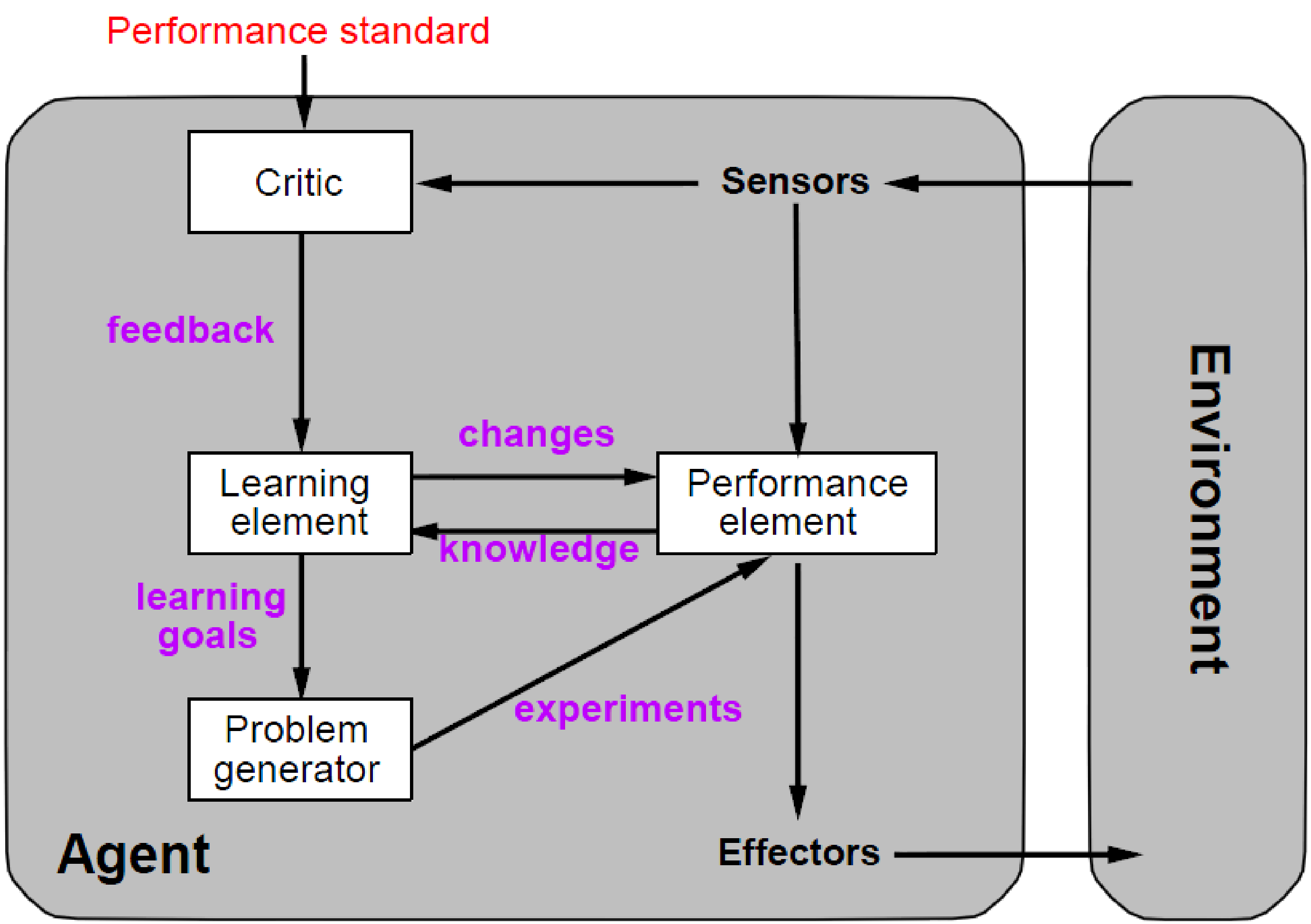
- An agent is **"learning"** if it improves its performance on future tasks after making observations about the world.
- Learning is essential for unknown environments,
  - i.e., when designer lacks omniscience
- Learning is useful as a system construction method,
  - i.e., expose the agent to reality rather than trying to write it all down
- Learning modifies the agent's decision mechanisms to improve performance
  - i.e., designer may not know how to solve a problem and leaves the agent to learn itself

- We will focus on specific type of learning problem that given a <u>collection of input-output pairs</u>, <u>learn a function the predicts the output fro new input (supervised learning)</u>

FORMS OF LEARNING

- Any component of an agent can be improved by learning.
- The improvement and the techniques to use to improve depends on four factors:
  - Which **components** to improve
  - What **prior knowledge** the agent already has.
  - What **representation** is used for data and component.
  - What **feedback** is available to learn from

LEARNING AGENT



COMPONENTS TO LEARN

- Mapping conditions to action
- Infer relevant information from the percept
- Utility information (desirability of state)
- Action-value information (desirability of action)
- Goals that describe states that has the maximum utility

REPRESENTATION AND PRIOR KNOWLEDGE

- Examples
  - Logical sentences
  - Bayesian networks

- For the following methods we will be looking at
  - Input: Factored representations (A vector of attribute values)
  - Output: continouse numerical value or a discrete value

TYPES OF LEARNING

Classification by representation

- Inductive learning
  - Learning a general function or rule from specific input-output pair
- Deductive (analytical) learning
  - Going from a known general rule to a new rule that is logically entailed but is useful because it allows more efficient processing.

Classification by types of feedback

- **Unsupervised learning**
  - Learns patterns in the input even though not explicit feedback (output) is supplied.
- Reinforcement learning
  - Learns from a series of reinforcements – rewards or punishments
- **Supervised learning**
  - Given example input-output pairs learns a function the maps input to output
- **Semi-supervised learning**
  - Given a few labeled samples and some unlabeled examples and learns a function the maps input to output

VOCABULARIES OF LEARNING

- What is being learned?
  - Parameters, structures (ex> Bayes net), hidden concepts
- What for?
  - Prediction, diagnosis, summarization
- How?
  - Passive vs Active,
  - Online vs Offline
- Output?
  - Classification/ Regression/ Clusters
- Other details
  - Generative model vs discriminative model

SUPERVISED LEARNING

The task of supervised learning:

Given a **Training set** of N example input-output pairs,

(x1, y1), ... (xN, yN)

where each *yj* was generated by an unknown function *y = f(x),*

discover a function *h* (***hypothesis***) that approximates the **true function** *f*.

Supervised learning problem is :

- **Classification** problem if y is discrete and finite
- **Regression** problem if y is continuous number

Measure accuracy of hypothesis with test set.

Hypothesis generalizes well if it correctly predicts the value of y for novel examples.

AIMA Chapter 18 (3)

DECISION TREES

LEARNING DECISION TREES

Task:

- Given: collection of examples (x, f(x))
- Return: a function *h* (*hypothesis*) that approximates *f*
- *h is a decision tree*

Input: an object or situation described by a set of attributes (or features)

Output: a “decision” – the predicts output value for the input.

The input attributes and the outputs can be discrete or continuous.

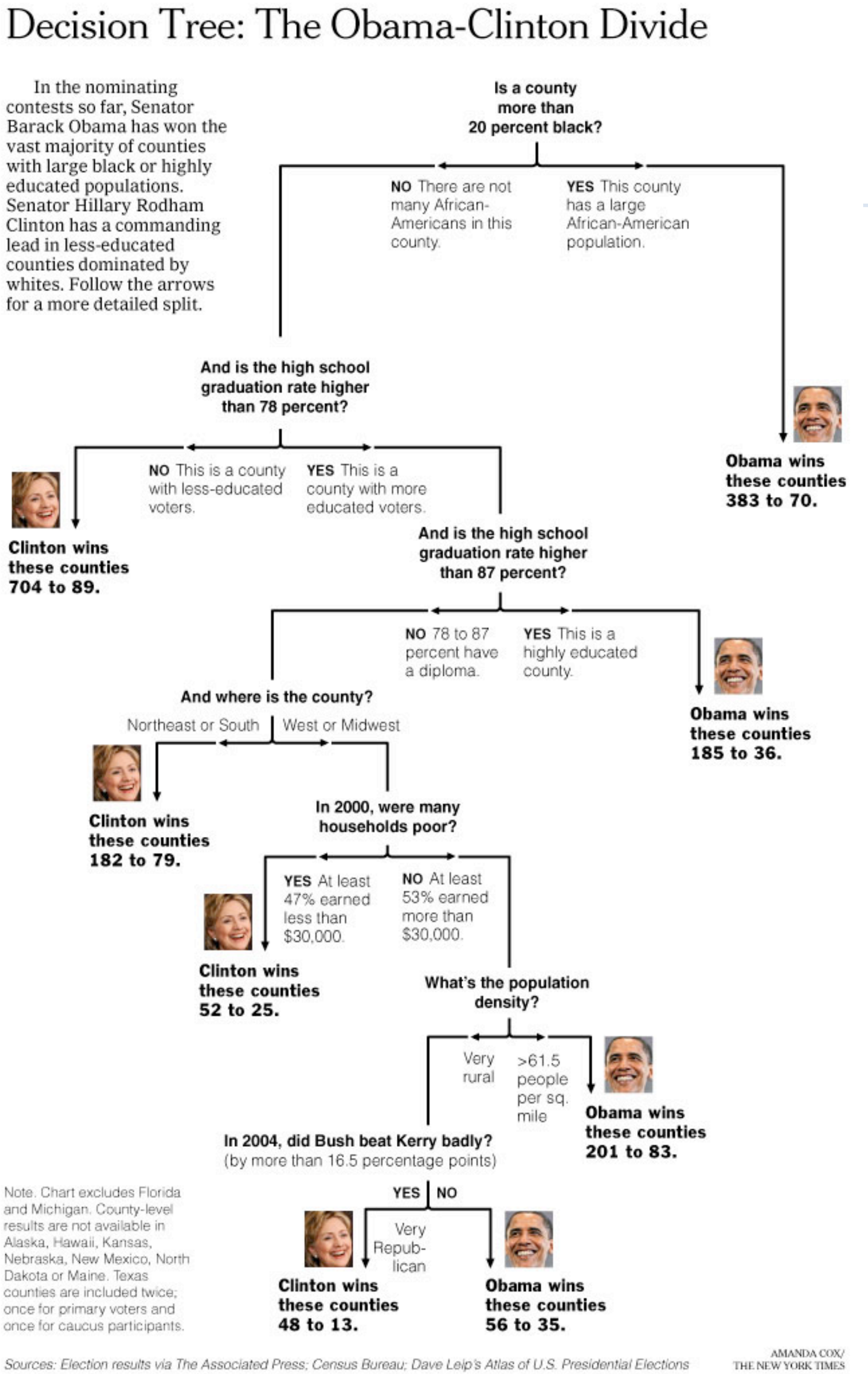
We will focus on decision trees for Boolean classification:

each example is classified as positive or negative.

DECISION TREE

- What is a decision tree?
- A tree with two types of nodes:
  - Decision nodes: Specifies a choice or test of some attribute with 2 or more alternatives; → every decision node is part of a path to a leaf node
  - Leaf node: Indicates classification of an example

New York Times
April 16, 2008



DECISION THREE REPRESENTATION

Problem: decide whether to wait for a table at a restaurant. What attributes would you use?

Attributes used by in the book

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)

What about restaurant name?

It could be great for generating a small tree but …

It doesn't generalize!

ATTRIBUTE-BASED REPRESENTATIONS

Examples described by attribute values (Boolean, discrete, continuous)

E.g.

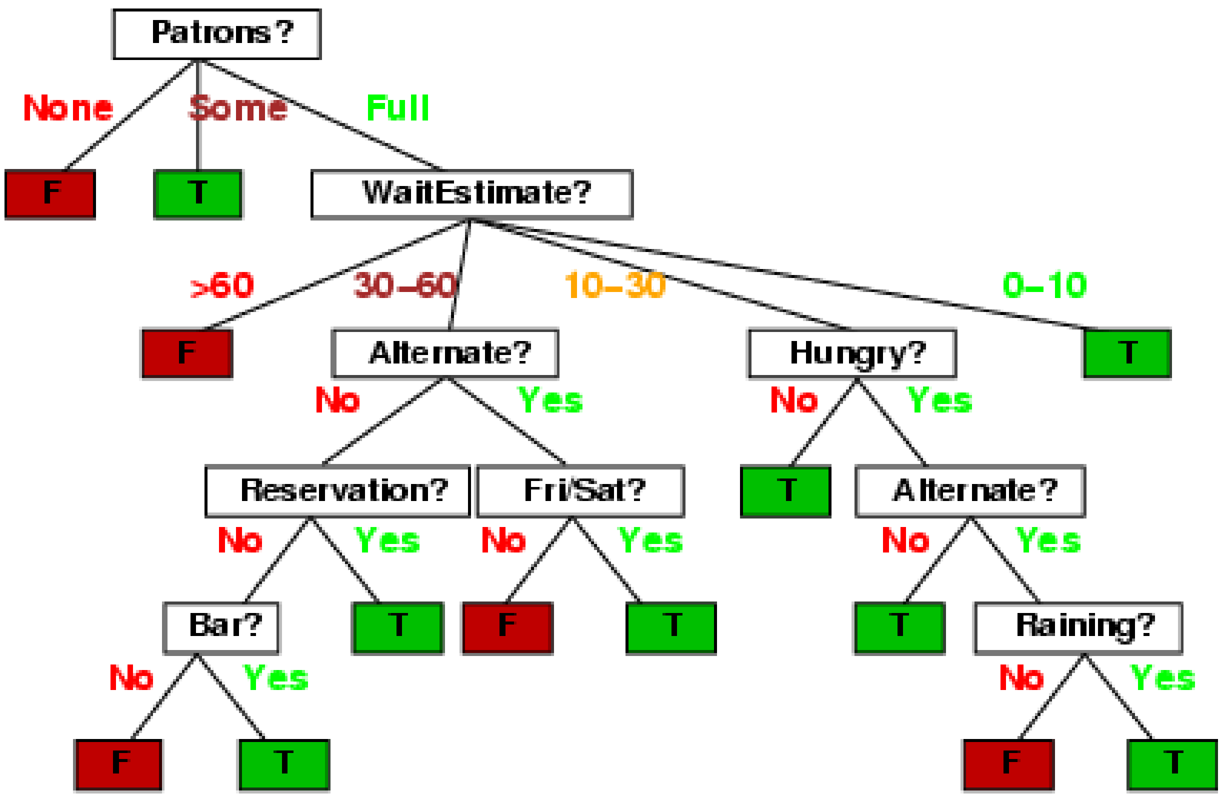
Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	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)

REPRESENTATION FOR HYPOTHESES

One possible representation for hypotheses

E.g., here is a tree for deciding whether to wait:



EXPRESSIVENESS OF DECISION TREES

Any particular decision tree hypothesis for WillWait goal predicate can be seen as a disjunction of a conjunction of tests, i.e., an assertion of the form:

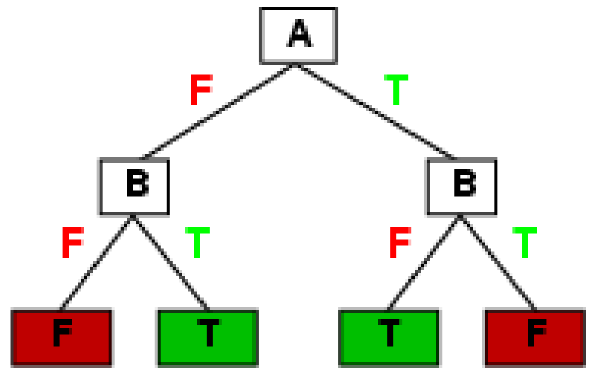
$$\forall s \;\; \mathit{WillWait}(s) \leftrightarrow (P1(s) \vee P2(s) \vee \ldots \vee Pn(s))$$

Where each condition Pi(s) is a conjunction of tests corresponding to the path from the root of the tree to a leaf with a positive outcome.

EXPRESSIVENESS CONT.

Decision trees can express any Boolean 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



HYPOTHESIS SPACES

How many distinct decision trees with *n* Boolean attributes?

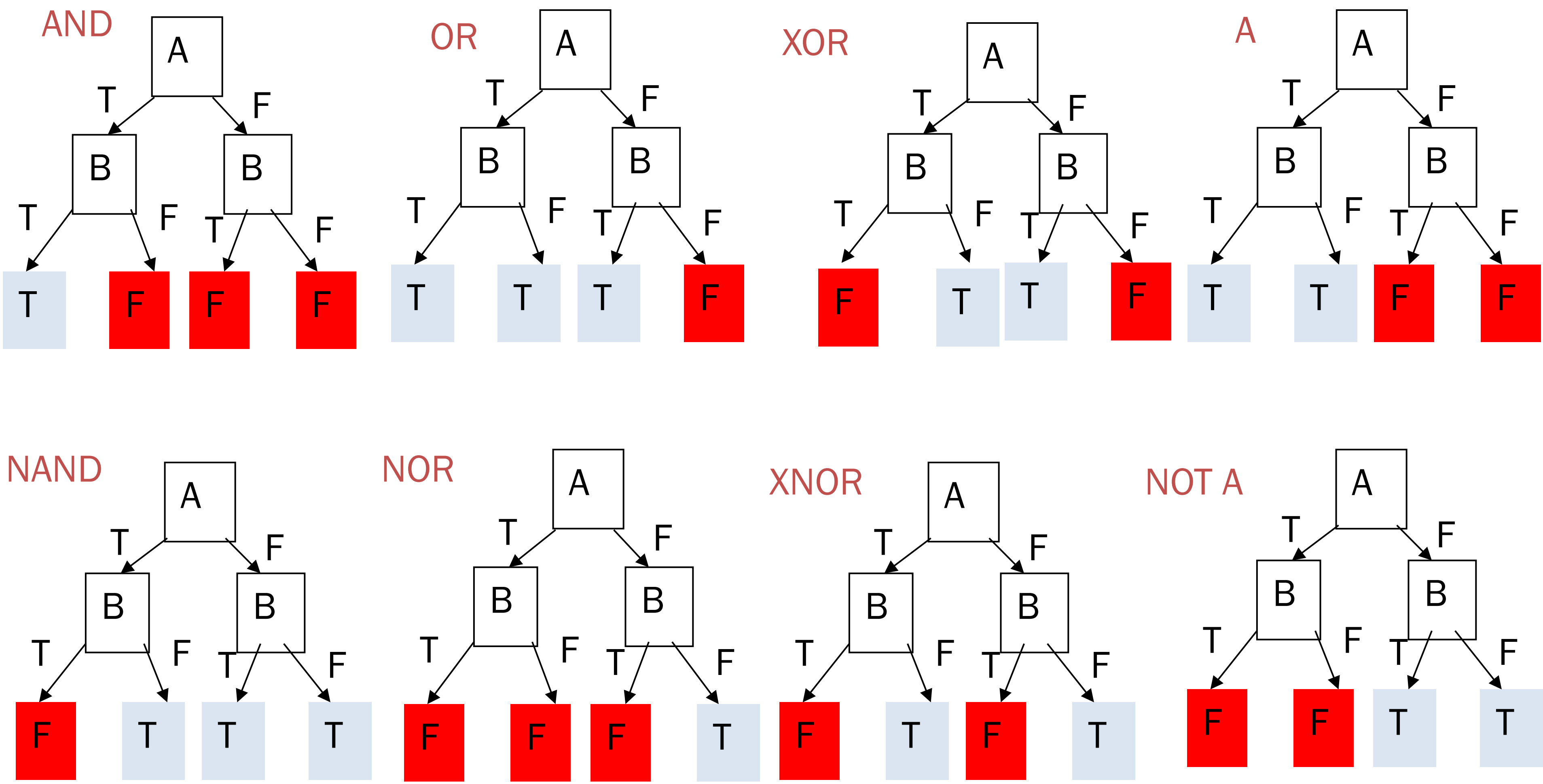
= number of Boolean functions

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

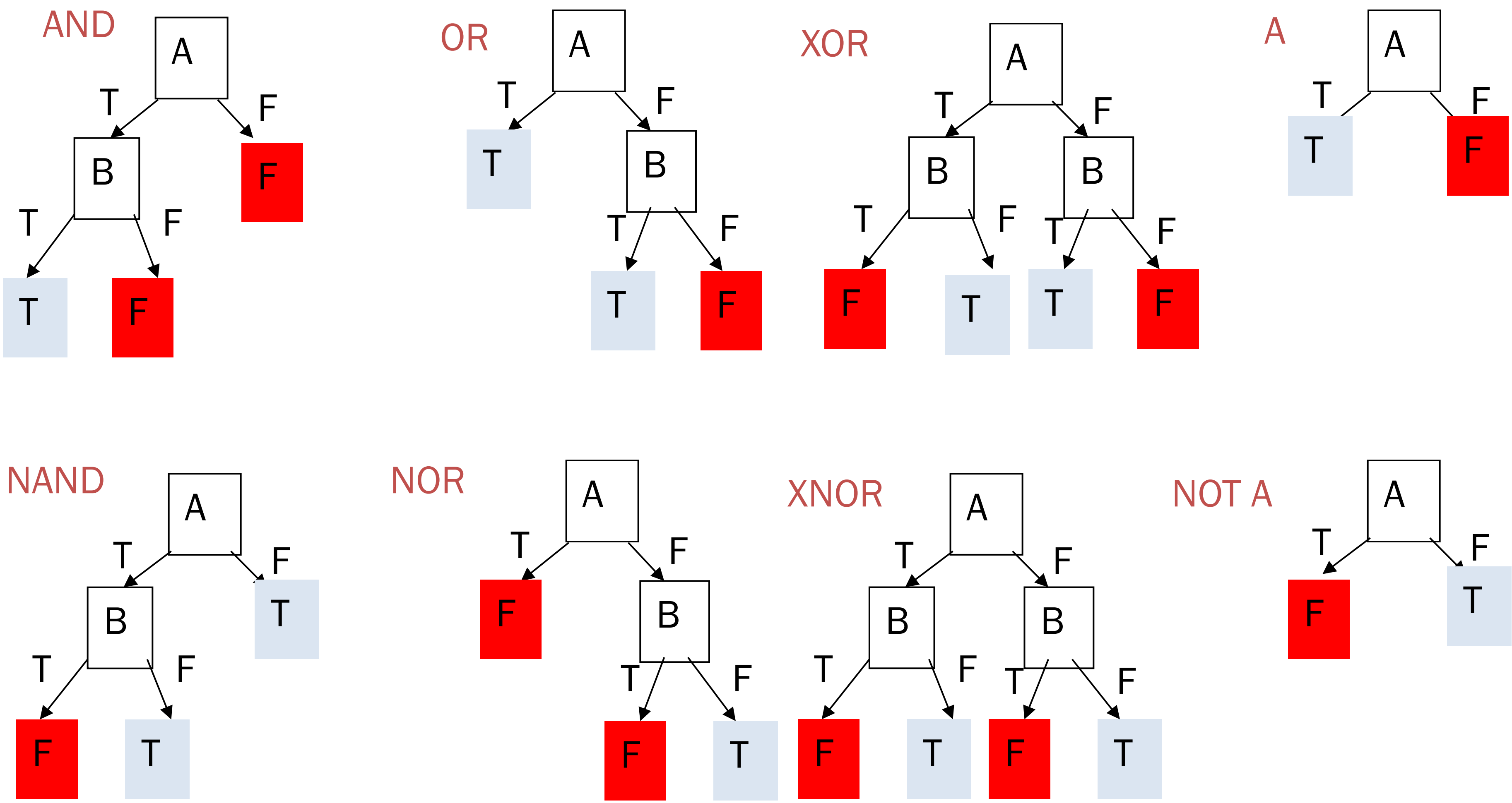
With 6 Boolean attributes, there are 18,446,744,073,709,551,616 possible trees!

There are even more decision trees!

EXPRESSIVENESS:
BOOLEAN FUNCTION WITH 2 ATTRIBUTES → 2^{2^2} DTS



EXPRESSIVENESS: 2 ATTRIBUTE → 2^{2^2} DTS



DECISION TREE LEARNING ALGORITHM

- Decision trees can express any Boolean function.
- Goal: Finding a decision tree that agrees with training set.
- We could construct a decision tree that has one path to a leaf for each example, where the path tests sets each attribute value to the value of the e xample.

What is the problem with this from a learning point of view?

Problem: This approach would just memorize example.
How to deal with new examples? It doesn't generalize!

(But sometimes hard to avoid --- e.g. parity function, 1, if an even number of inputs, or majority function, 1, if more than half of the inputs are 1).

We want a compact/smallest tree.
But finding the smallest tree consistent with the examples is NP-hard!

- Overall Goal: get a good classification with a small number of tests.

DATA (INPUT-OUTPUT)

Examples described by attribute values (Boolean, discrete, continuous)

E.g., situations where I will/won't wait for a table:

input- output

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)

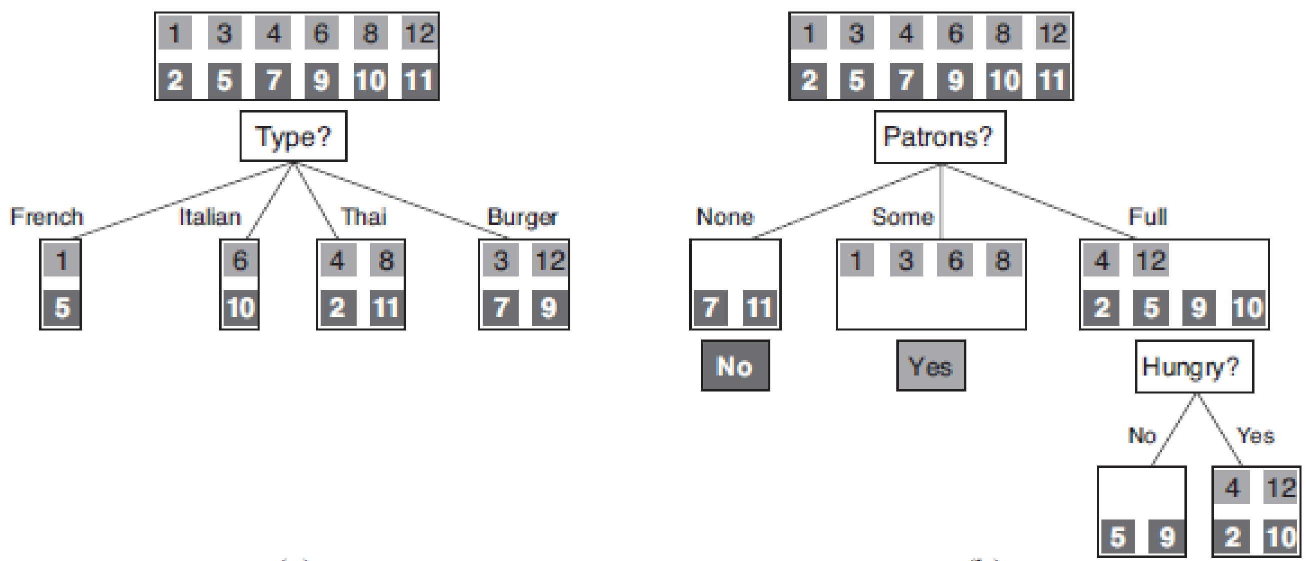
DECISION TREE LEARNING

- Goal:
  - find a *small* tree consistent with the training examples

- Idea:
  1. (recursively) choose "most significant" attribute as root of (sub)tree;
  2. Use a divide-and-conquer greedy search through the space of possible decision trees.
  3. Greedy because there is no backtracking. It picks highest values first.

- Divide-and-conquer greedy construction
  - Which attribute should be tested?
    - Heuristics and Statistical testing with current data
  - Repeat for descendants

- “most significant attribute”:
  - One that makes the most difference to the classification of an example such that we may get to the correct classification with a small number of tests (= shallow tree)
- Ex> Patrons is better attribute than types.



(a) (b)

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

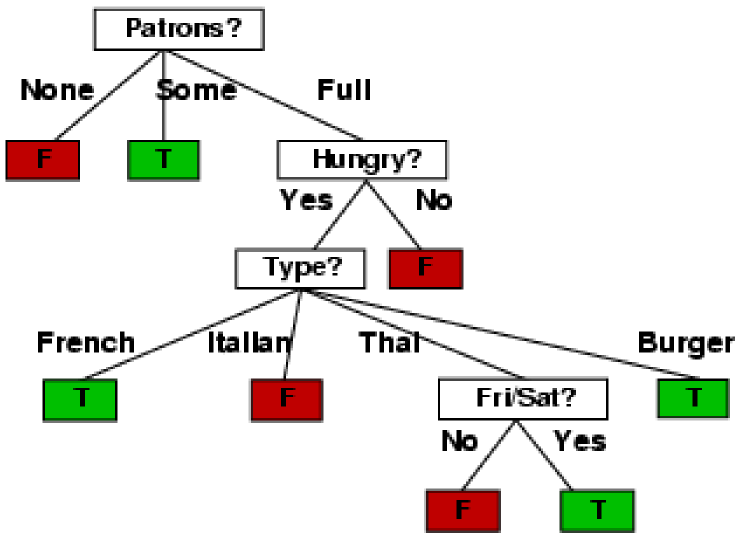
if *examples* is empty **then return** PLURALITY-VALUE(*parent_examples*)
else if all *examples* have the same classification **then return** the classification
else if *attributes* is empty **then return** PLURALITY-VALUE(*examples*)
else
  $A \leftarrow \operatorname{argmax}_{a \in attributes}$ IMPORTANCE(a, *examples*)
  tree ← a new decision tree with root test A
  for each value v_k of A **do**
    $exs \leftarrow \{e : e \in examples$ **and** $e.A = v_k\}$
    subtree ← DECISION-TREE-LEARNING(*exs*, *attributes* − A, *examples*)
    add a branch to *tree* with label $(A = v_k)$ and subtree *subtree*
  return *tree*

Figure 18.4 The decision-tree learning algorithm. The function IMPORTANCE is described in Section ??. The function PLURALITY-VALUE selects the most common output value among a set of examples, breaking ties randomly.

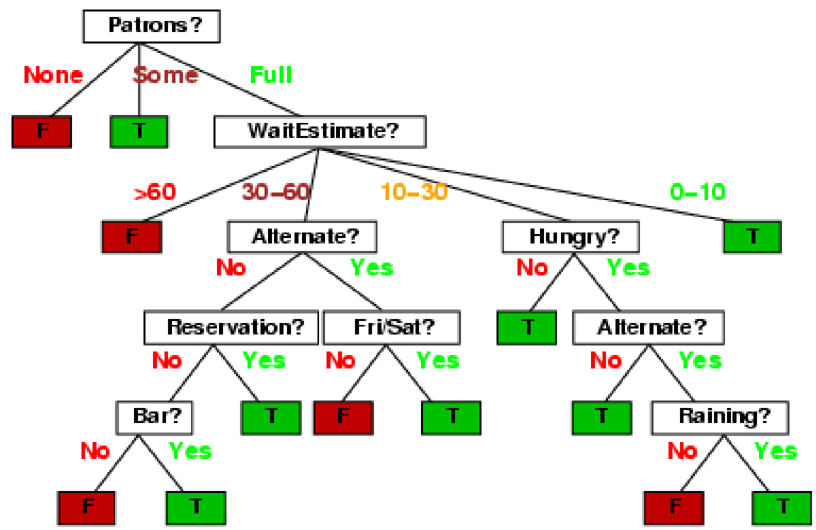
EXAMPLE CONTD.

- Decision tree learned from the 12 examples:

Learned Three



Original Tree



Substantially simpler than "true" tree ---
but a more complex hypothesis isn't justified
from just the data.

EVALUATIONS OF ACCURACY OF THE LEARNING

- One way is to look at a learning curve
- Decide how many examples we need as well