Lecture 20

Learning

Lusi Li
Department of Computer Science
ODU

Reading for This Class:
Chapter 18, Russell and Norvig

Review

- Last Class
  - First-order Logic
- This Class
  - Learning
  - Supervised Learning
- Next Class
  - Unsupervised Learning

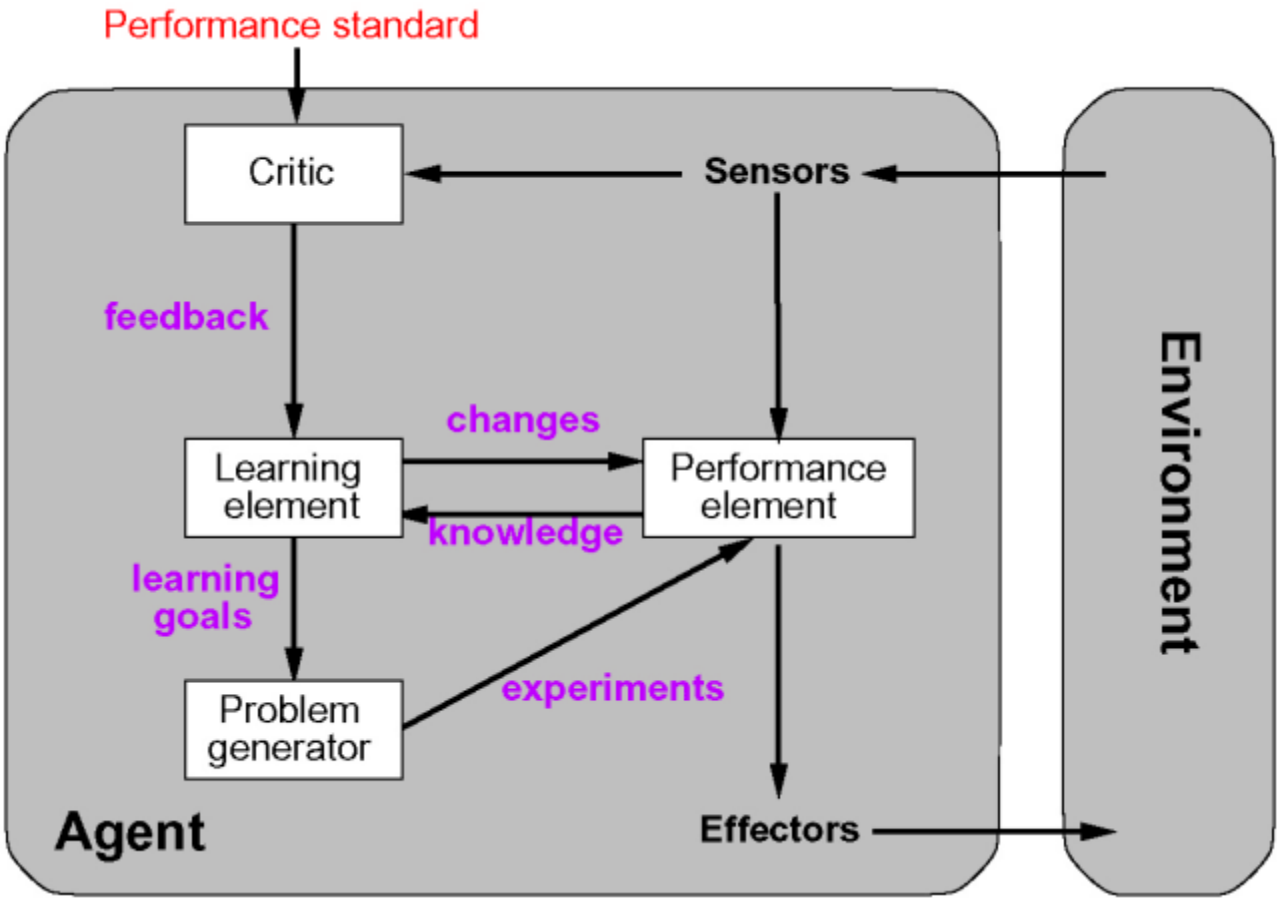
- The hard deadline for Homework 4: April 26
- Final Exam: April 25, 26

Machine Learning

- **Concepts and definitions given by Tom M. Mitchell**
  - **Machine learning is the study of computer algorithms that improve automatically through experience.**
  - **Machine learning is an inherently interdisciplinary field, built on concepts from artificial intelligence, probability and statistics, information theory, philosophy, control theory, psychology, neurobiology, and other fields.**
  - **Learning:**

    Study of algorithms that
    - **improve their performance P**
    - **at some task T**
    - **with experience E**

Learning Agent Model



Introduction to Learning Agents

- **Any component of an agent can be improved by machine learning**
  - **The improvements, and the used techniques depend on**
    - **Which component is to be learned**
    - **What prior knowledge the agent has**
    - **What representation is used for the data and the component**
    - **What feedback is available to learn from**

- **Components to be learned**
    - **mapping from conditions to actions**
    - **methods of inferring world properties from percept sequences**
    - **changes in the world**
    - **exploration of possible actions**
    - **utility information about the desirability of world states**
    - **goals to achieve high utility values**

Representation and Prior Knowledge

- **Many possible representation schemes**
  - **weighted polynomials (e.g. in utility functions for games)**
  - **propositional logic**
  - **first order logic**
  - **probabilistic methods (e.g. belief networks)**

- **Prior Knowledge**
  - **background knowledge available before a task is tackled**
  - **can increase performance or decrease learning time considerably**
  - **many learning schemes assume that no prior knowledge is available**

Feedback

- Provides information about the actual outcome of actions

- Three types of feedback determine the three main types of learning:
- supervised learning
  - learns a function that maps from input to output
  - agent observes some example input-output pairs
- unsupervised learning
  - learns patterns from the input
  - no specific output values are supplied
- reinforcement learning
  - learns from a series of reinforcements: rewards and punishments
  - it is up to the agent to decide which of the actions prior to the reinforcement were most responsible for it, and to alter its actions for more rewards in the future

Supervised Learning

- **Problem Setting**
  - **Input:** a training set of N example input-output pairs (X, Y)

$$(x_1, y_1),\ (x_2, y_2),\ \ldots,\ (x_N, y_N),$$

where each pair was generated by an unknown function

$$y = f(x) \quad or \quad f: X \rightarrow Y$$

Goal: a hypothesis function $h \in \mathcal{H}$ that approximates the true function f

Output: output the predicated labels Y_test, given a new test set X_test, by

Y_test = h(X_test)

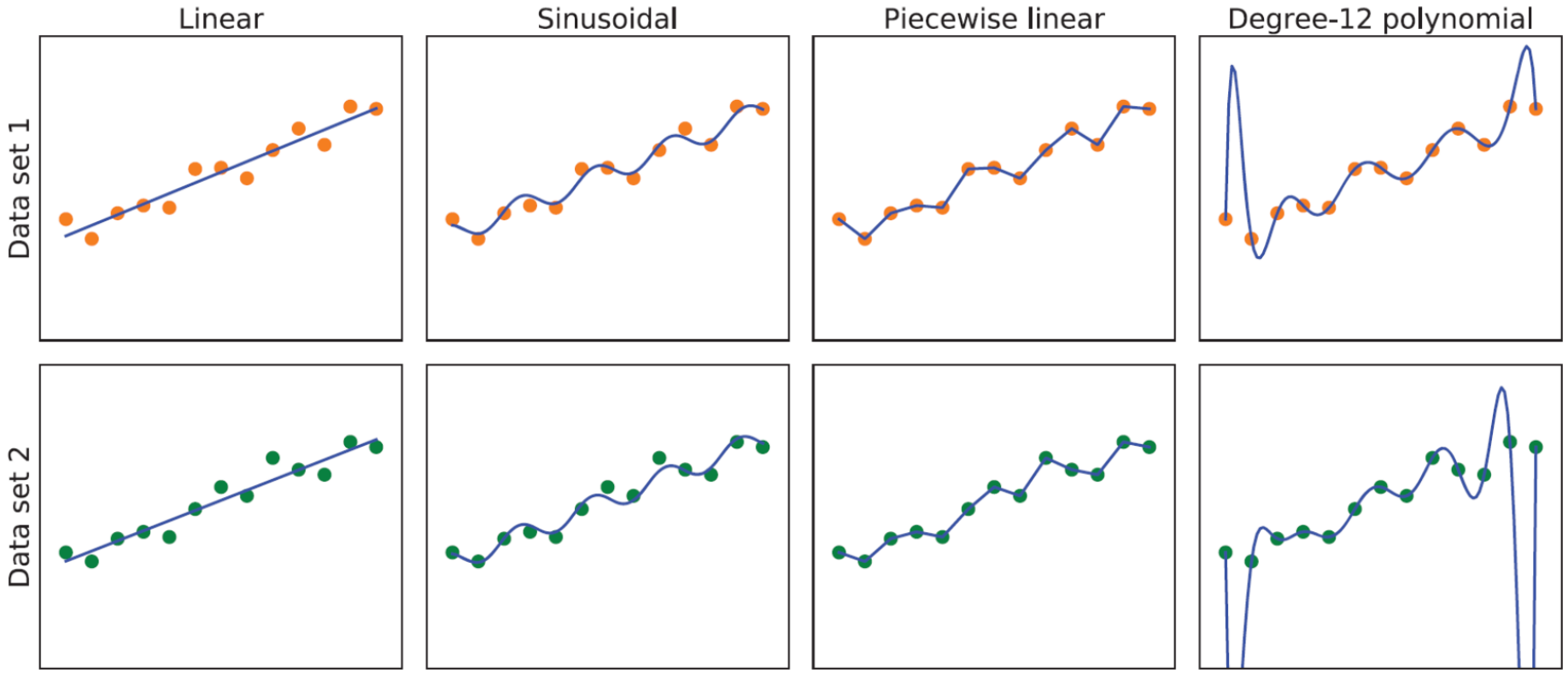
  - Function h is called a hypothesis
    - It is drawn from a hypothesis space $\mathcal{H}$ of possible functions.

$$\mathcal{H} = \{h |\ h: X \rightarrow Y\}$$

      - E.g., the set of linear functions, set of polynomials of degree 3
    - Good hypothesis generalize well – predict unseen examples correctly

Finding Hypotheses to Fit Data

$h(x) = w_1 x + w_0$ $h(x) = w_1 x + \sin(w_0 x)$ $h(x) = \sum_{i=0}^{12} w_i x_i$

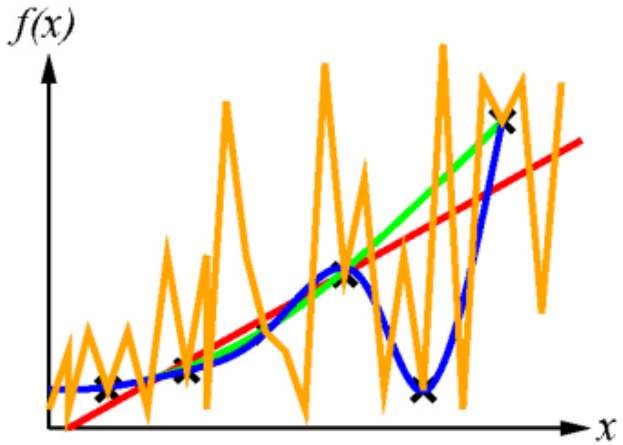


Finding hypotheses to fit data. Top row: four plots of best-fit functions from four different hypothesis spaces trained on data set 1. Bottom row: the same four functions, but trained on a slightly different data set (sampled from the same $f(x)$ function).

Supervised Learning

- Construct/adjust *h* to agree with *f* on training set
- *h* is consistent if each x_i in the training set has h(x_i) = y_i

E.g., curve fitting:



- How do we choose from among multiple consistent hypotheses?
- Ockham's razor: prefer the simplest hypothesis consistent with the data

Decision Tree Learning

- Regression learning – learning continuous function
- Classification learning – learning discrete-valued function
  - Boolean classification
    classification of examples is positive (T) or negative (F)

- A decision tree
  - simplest yet useful supervised learning algorithm
  - takes as input an example described by a set of attributes and returns a "decision" – the predicted output value for the input
  - learns a tree from the training examples
    - it starts with the root node and expands further branches and constructs a tree-like structure.

Decision Tree Structure

- Each decision (internal) node: test one attribute value
- Each branch from a node: selects one value of the chosen attribute
  - Each branch can be considered as a sub-tree
- Each leaf node represents the outcome

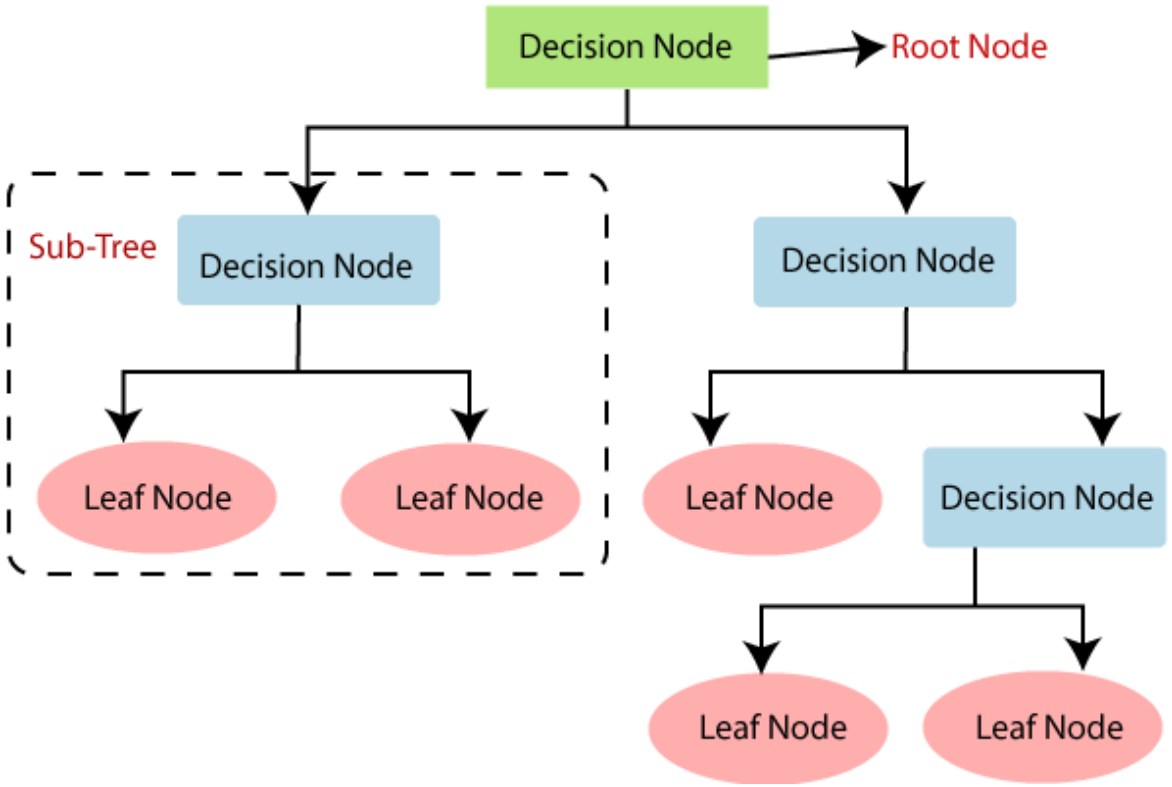


Image source: javatpoint.com

Sample Dataset

Given:

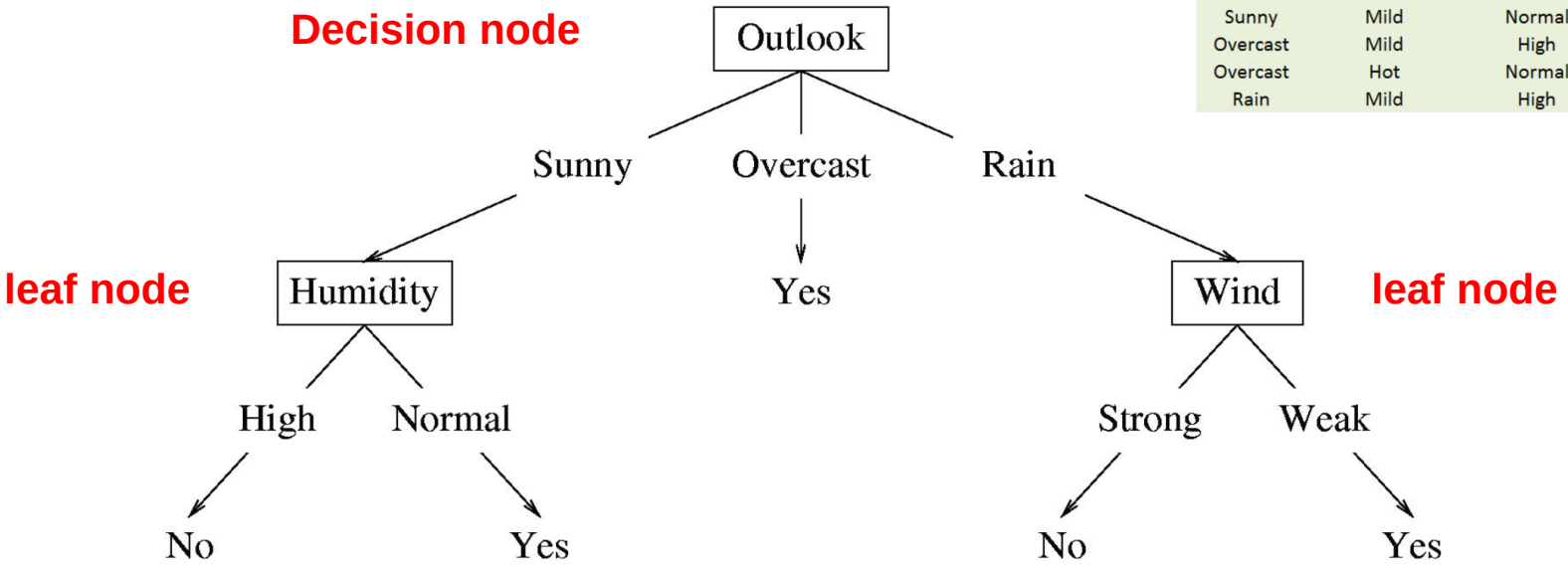
- **Columns denote attributes (or features)** A_i **(**$i = 1,2,3,4$**)**
- **Rows denote labeled instances** $(x_j, y_j)(j = 1, 2 \ldots, 14)$
- **Class label denotes whether a tennis game was played (Yes/No)**

	Predictors				Response
	Outlook	**Temperature**	**Humidity**	**Wind**	**Class**
	Sunny	Hot	High	Weak	No
	Sunny	Hot	High	Strong	No
	Overcast	Hot	High	Weak	Yes
	Rain	Mild	High	Weak	Yes
	Rain	Cool	Normal	Weak	Yes
(x_6, y_6)	Rain	Cool	Normal	Strong	No
	Overcast	Cool	Normal	Strong	Yes
	Sunny	Mild	High	Weak	No
	Sunny	Cool	Normal	Weak	Yes
	Rain	Mild	Normal	Weak	Yes
	Sunny	Mild	Normal	Strong	Yes
	Overcast	Mild	High	Strong	Yes
	Overcast	Hot	Normal	Weak	Yes
	Rain	Mild	High	Strong	No

Decision Tree

- **A possible decision tree**
- **Each decision node: test one attribute A_i**
- **Each branch from a node: selects one value for A_i**
- **Each leaf node: predict Y**

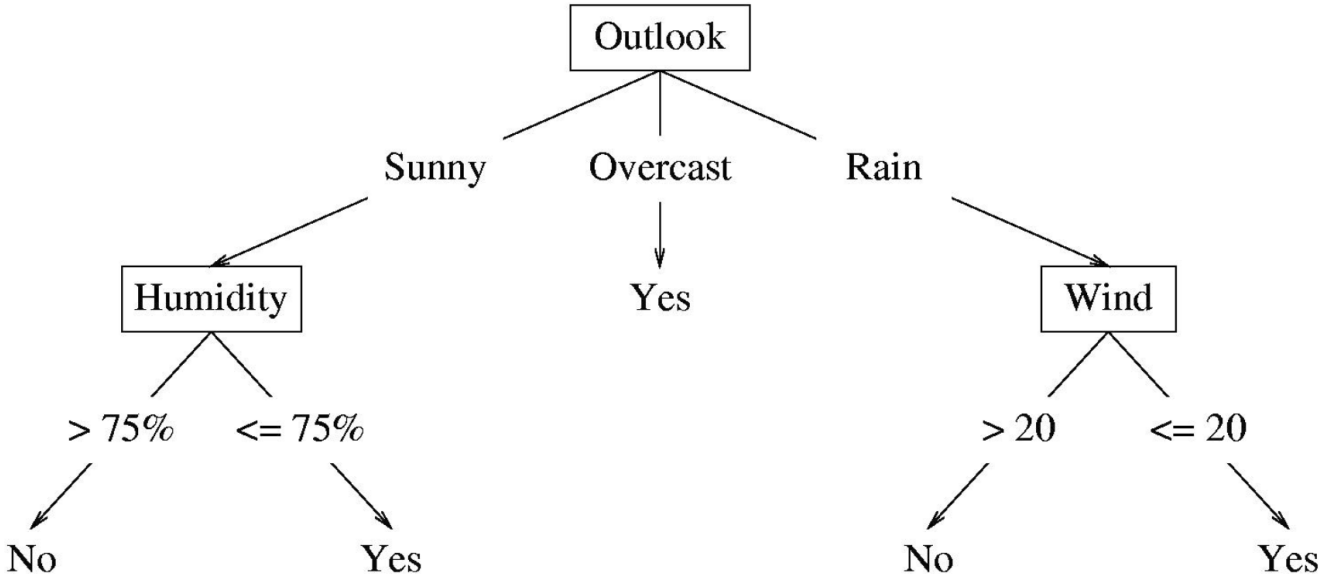
Predictors				Response
Outlook	**Temperature**	**Humidity**	**Wind**	**Class**
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No



- **What prediction would we make for**
  <outlook=sunny, temperature=hot, humidity=high, wind=weak> ? **No**

Decision Tree

- **If attribute values are continuous, the nodes can test the value of an attribute against a threshold**

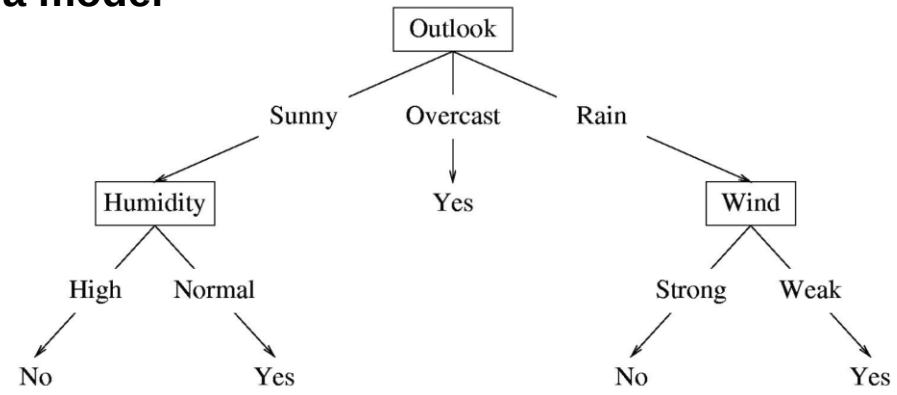


- **What prediction would we make for**
  <outlook=sunny, temperature=hot, humidity=60%, wind=10> ? **Yes**

Decision Tree Learning

- **Problem Setting:**
- **Set of possible instances (or examples) X**
  - **Each instance x in X is a vector of attribute values**
  - **E.g., <outlook=sunny, temperature=hot, humidity=high, wind=weak>**
- **Unknown target function $f: X \rightarrow Y$**
  - **Y is discrete valued**
- **Set of function hypotheses $\mathcal{H} = \{h \mid h: X \rightarrow Y\}$**
  - **Each hypothesis h is a decision tree or a model**
  - **Tree sorts x to a leaf, which assigns y**
- **Apply the model h to new data X_test**

  $Y\_test \leftarrow h(X\_test)$



Sample Dataset

- **An example for Boolean decision tree consists of a vector of input attributes, *X*, and a single Boolean output value *y*.**
  - **e.g., positive examples: *WillWait* is *Yes***
  - **Training set – the complete set of examples for finding the hypothesis (tree)**

Example	Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	*Yes*
X_2	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	*No*
X_3	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	*Yes*
X_4	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	*Yes*
X_5	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	*No*
X_6	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	*Yes*
X_7	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	*No*
X_8	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	*Yes*
X_9	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	*No*
X_{10}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	*No*
X_{11}	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	*No*
X_{12}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	*Yes*

Examples for the restaurant domain.

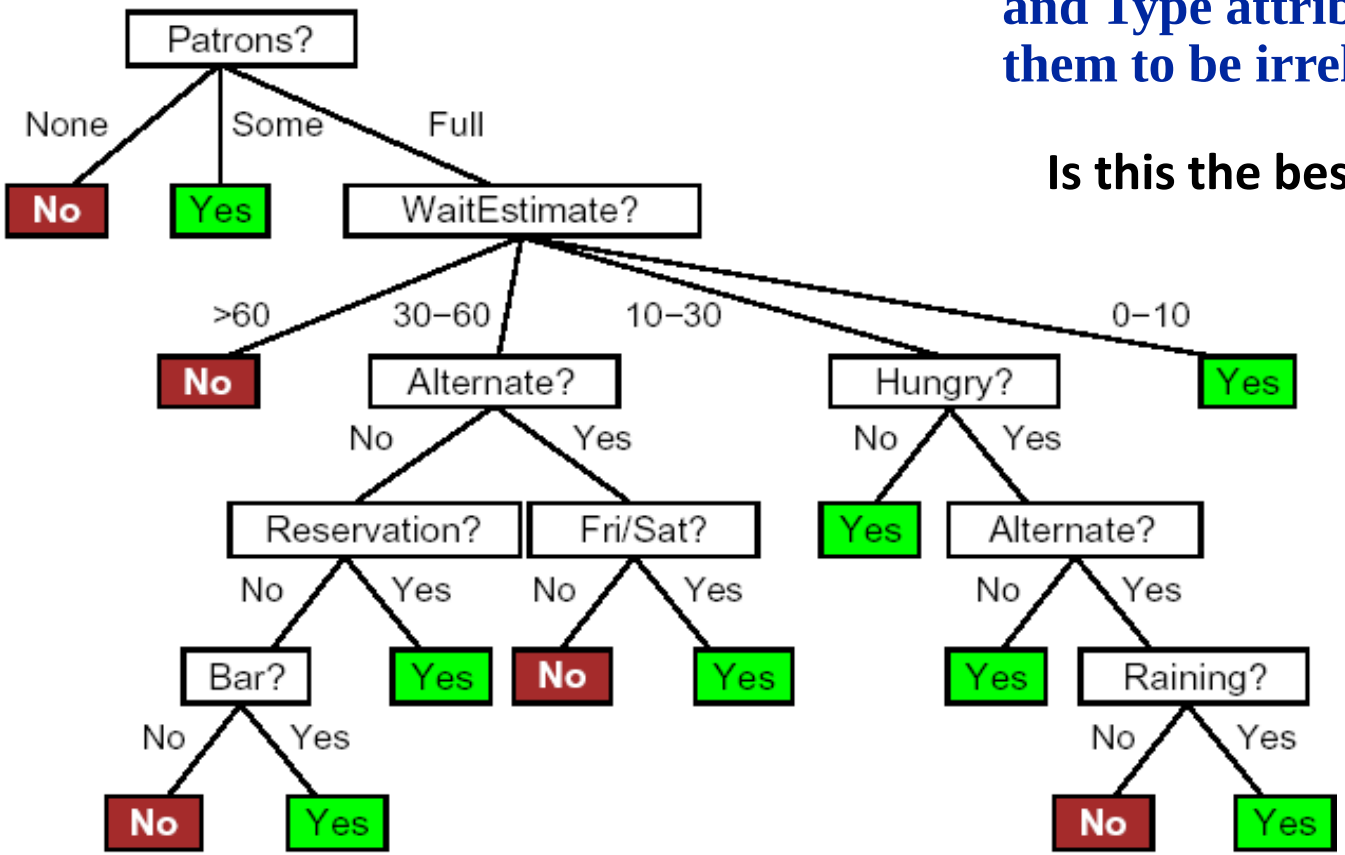
Attributes for Restaurant Problem

Model a patron's decision of whether to wait for a table at a restaurant.

1. Alternate: **whether there is a suitable alternative restaurant nearby**
2. Bar: **whether the restaurant has a comfortable bar area to wait in**
3. Fri/Sat: **true on Fridays and Saturdays**
4. Hungry: **whether we are hungry**
5. Patrons: **how many people are in the restaurant (values are None, Some, and Full)**
6. Price: **the restaurant's price range ($, $$, $$$)**
7. Raining: **whether it is raining outside**
8. Reservation: **whether we made a reservation**
9. Type: **the kind of restaurant (French, Italian, Thai, or burger)**
10. WaitEstimate: **the wait estimated by the host (0 – 10, 10 – 30, 30 – 60, >60 minutes)**

Decision trees

- **Making decision by performing a sequence of tests**



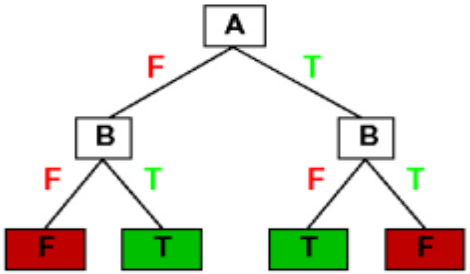
A decision does not use the Price and Type attributes --considering them to be irrelevant

Is this the best decision tree?

Expressiveness of decision trees

- Decision trees can express any Boolean function of the input attributes
  - truth table row → path to leaf

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

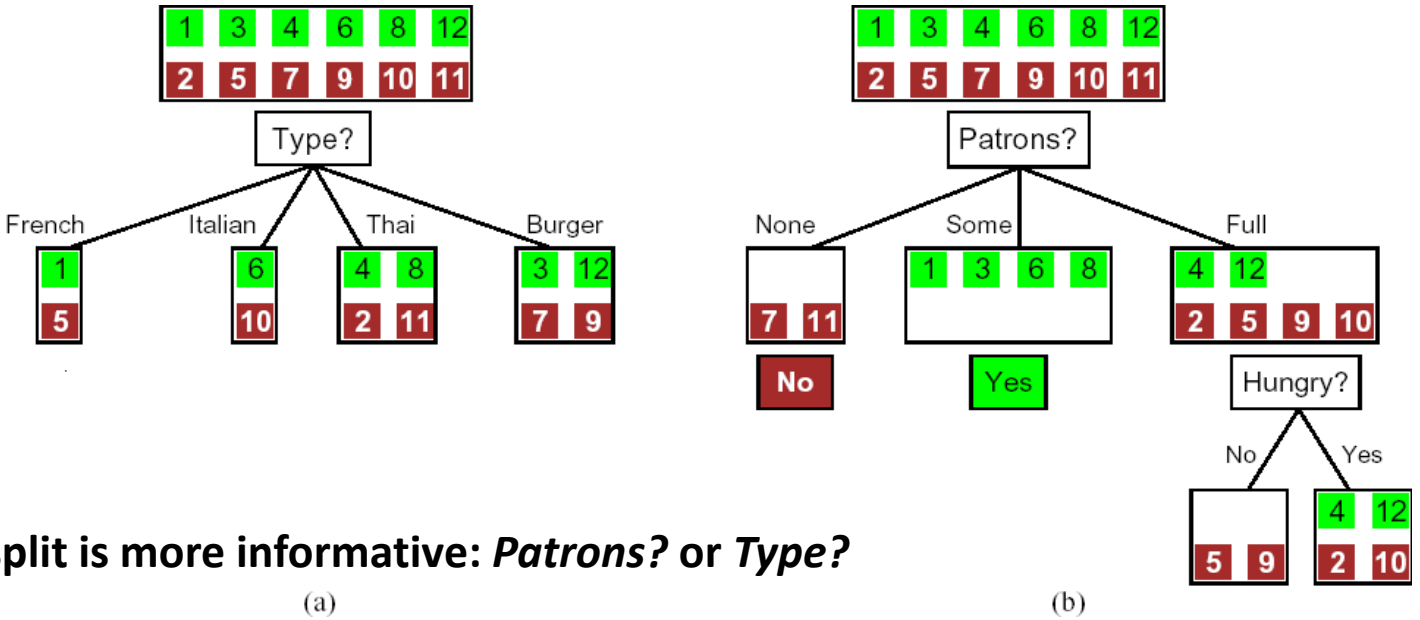


- In the worst case, the tree will require exponentially many nodes.
- We can build a trivial decision tree for any training set with one path to leaf for each example
  - just memorizes the observations, i.e., does not extract any pattern (rule), so it cannot extrapolate (generalize) to new (it has not seen) examples

- We should find the smallest decision tree that is consistent with the examples
  - finding the smallest is intractable ☹
  - how to find a "smallish" (compact) one?
    - to test the most important (efficient) attribute first

Choosing an attribute

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



Which split is more informative: *Patrons?* or *Type?*

- ***Patron?* is a better choice because it leaves us with example sets for which we can answer definitively (Yes/No)**
- **After the first attribute test splits up the examples, each outcome is a new decision tree learning problem itself, with fewer examples and one fewer attributes.**

Basic Algorithm of Decision Tree

- node = root of decision tree
- Main loop:
- 1. A ← the "best" decision attribute for the next node.
- 2. Assign A as decision attribute for node.
- 3. For each value of A, create a new child of node.
- 4. Sort training examples to leaf nodes.
- 5. If training examples are perfectly classified, stop.
  Else, recurse over new leaf nodes.

Basic Algorithm of Decision Tree

Key problem: choosing which attribute to split a given set of examples

How do we choose which attribute is best?

Some possibilities are:

- **Random:** Select any attribute at random
- **Least-Values:** Choose the attribute with the smallest number of possible values
- **Most-Values:** Choose the attribute with the largest number of possible values
- **Max-Gain:** Choose the attribute that has the largest expected *information gain* i.e., attribute that results in smallest expected size of subtrees rooted at its children

Recursive decision trees

- **Four cases for each recursive problem:**

1) some positive and some negative examples

– choose the best attribute to split them

2) all positive (or negative) examples

– done (Yes / No)

3) no examples left (no such example has been observed)

– return a default value calculated from the majority classification at the node's parent

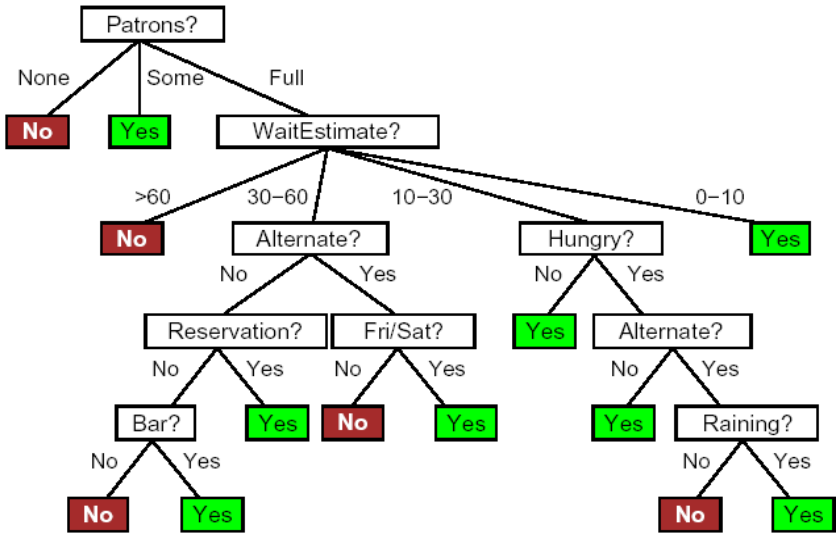
4) no attributes left but both positive and negative examples (same attributes values but different classifications)

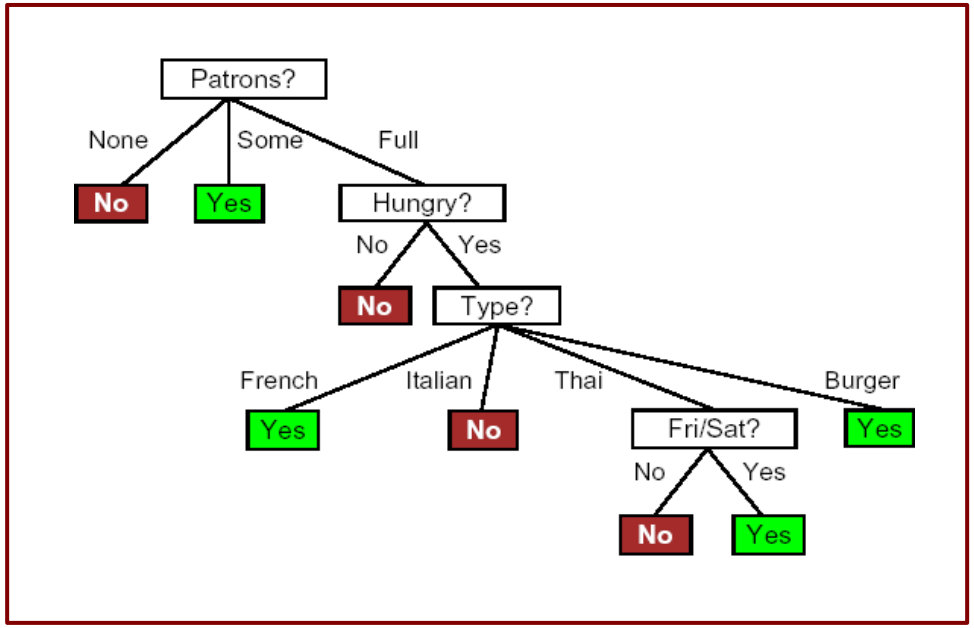
– something is wrong! some of the data are incorrect (noise)

– insufficient attributes or domain is nondeterministic (uncertainty, probability)

Decision tree learned (induced)

- **Decision tree learned from the 12 examples with Max-Gain**





- **The tree not only agrees with all the examples, but is much simpler than the original tree**
- **Learning algorithm does not include the test for *Raining* and *Reservation*?**

Because it can classify all the examples without them!

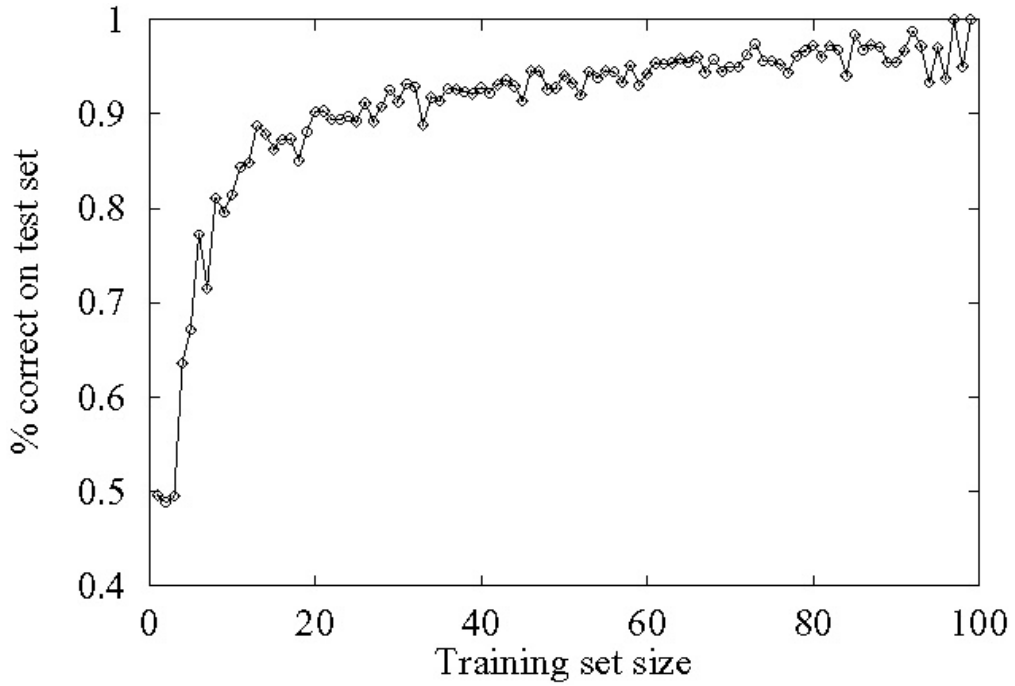
Performance measurement

- Good learning algorithm – produces hypotheses (functions) that correctly predicts the classifications of unseen examples
- Assessing prediction quality of a hypothesis
  - Checking its prediction against the correct classification on a set of examples
    – test set

  1) Collect a large set of examples
  2) Divide it into two disjoint sets: the training set and the test set
  3) Apply the learning algorithm to the training set, generating a hypothesis *h*
  4) Measure the percentage of examples in the test set that are correctly classified by *h*
  5) Repeat steps 2 to 4 for different size of training sets and different randomly selected training sets of each size
  - Plot average prediction quality as a function of the size of the training set
    – learning curve

Performance measurement

- **Learning curve (happy graphs)**
  - **% correct on test set as a function of training set size**
  - **the prediction quality increases as the training set grows**
  - **try *h* on a new test set of examples (use same distribution over example space as training set)**



Summary

- Machine Learning
- Learning Agents
  - Elements
- Learning Methods
  - Three different types of learning
- Supervised Learning
- Learning Decision Tree
  - Generating learning decision tree
- Performance Measure

What I want you to do

- Review Chapter 18
- Work on your assignment