Machine Learning

Decision Tree Learning

Artificial Intelligence & Computer Vision Lab
School of Computer Science and Engineering
Seoul National University

Overview

- Introduction
- Decision Tree Representation
- Learning Algorithm
- Hypothesis Space Search
- Inductive Bias in Decision Tree Learning
- Issues in Decision Tree Learning
- Summary

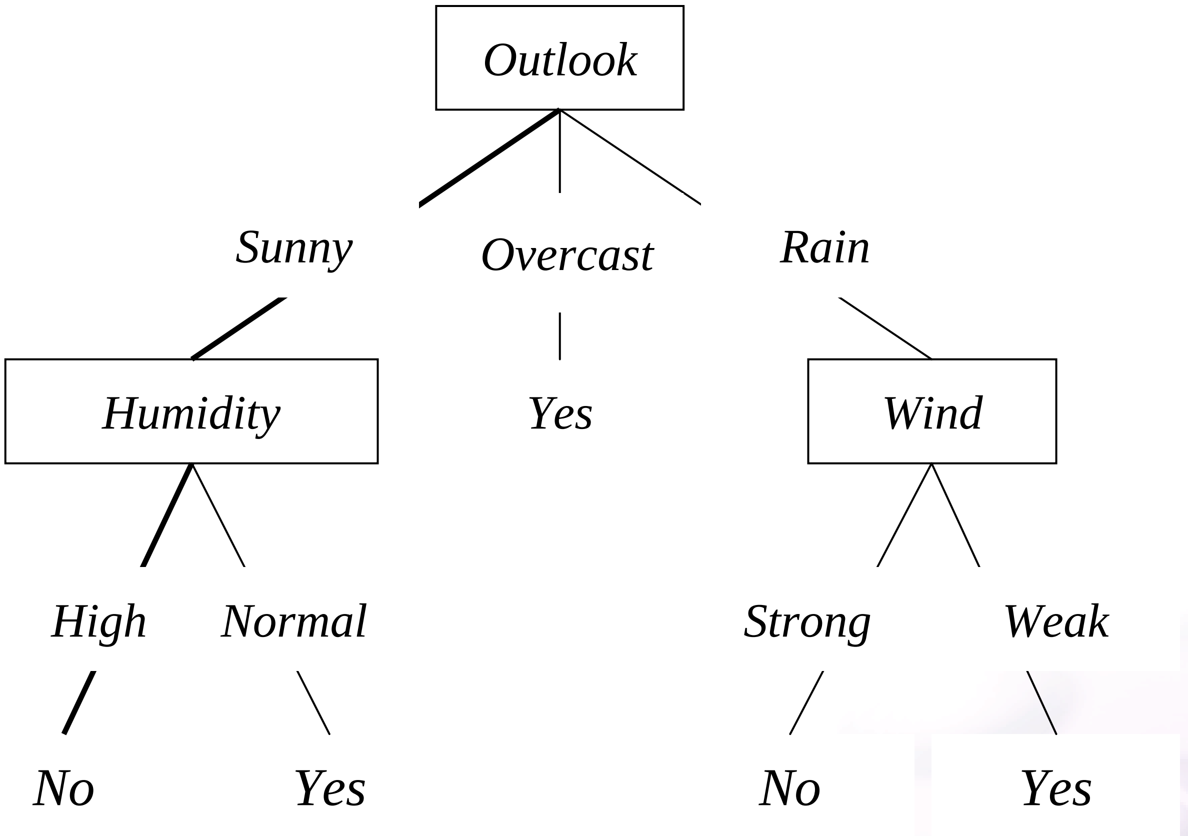
Introduction

- Decision tree learning is a method for approximating discrete-valued target function
- The learned function is represented by a decision tree
- Decision tree can also be re-represented as if-then rules to improve human readability

Decision Tree Representation

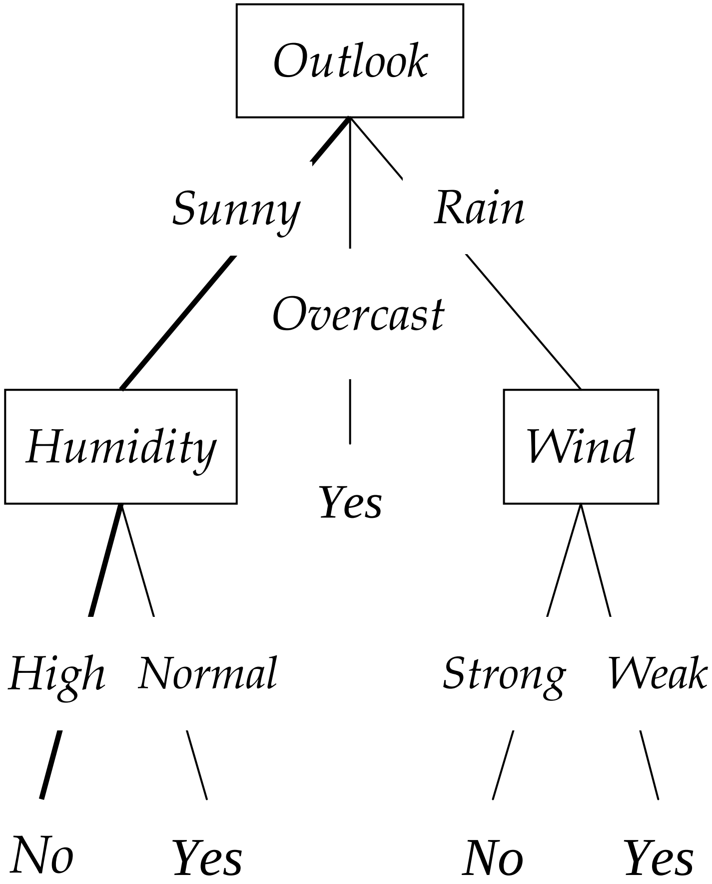
- Decision trees classify instances by sorting them down the tree from the root to some leaf node
- A node
  - Specifies some attribute of an instance to be tested
- A branch
  - Corresponds to one of the possible values for an attribute

Decision Tree Representation (cont.)



A Decision Tree for the concept *PlayTennis*

Decision Tree Representation (cont.)



- Each path corresponds to a conjunction of attribute tests. For example, if the instance is (*Outlook=sunny, Temperature=Hot, Humidity=high, Wind=Strong*) then the path of (*Outlook=Sunny* $\wedge$ *Humidity=High*) is matched so that the target value would be NO as shown in the tree.
- A decision tree represents a disjunction of conjunction of constraints on the attribute values of instances. For example, three positive instances can be represented as *(Outlook=Sunny* $\wedge$ *Humidity=normal)* $\vee$ *(Outlook=Overcast)* $\vee$ *(Outlook=Rain* $\wedge$ *Wind=Weak)* as shown in the tree.

What is the merit of tree representation?

Decision Tree Representation (cont.)

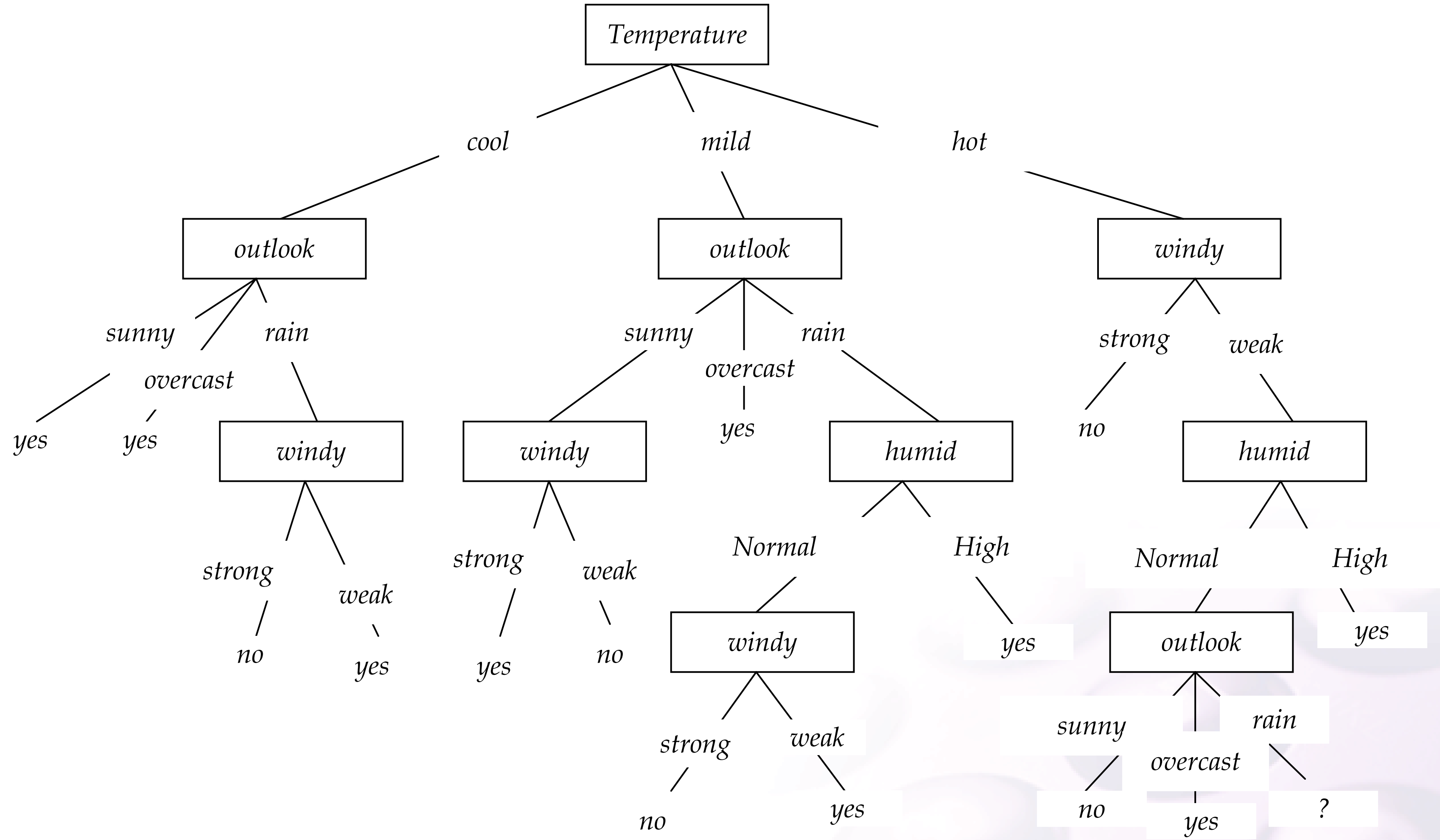
- Appropriate Problems for Decision Tree Learning
  - Instances are represented by attribute-value pairs
  - The target function has discrete output values
  - Disjunctive descriptions may be required
  - The training data may contain errors
    - Both errors in classification of the training examples and errors in the attribute values
  - The training data may contain missing attribute values
  - Suitable for classification

Learning Algorithm

- Main question
  - Which attribute should be tested at the root of the (sub)tree?
    - Greedy search using some statistical measure

- Information gain
  - A quantitative measure of the worth of an attribute
  - How well a given attribute separates the training example according to their target classification
  - Information gain measures the expected reduction in entropy

Learning Algorithm (cont.)

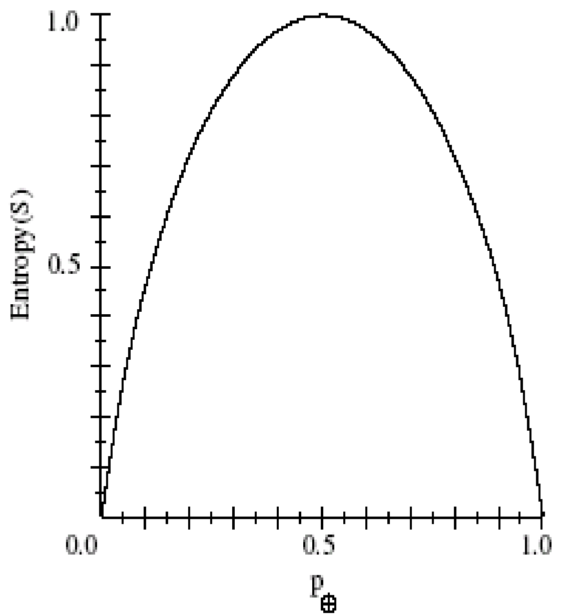


Learning Algorithm (cont.)

- Entropy
  - characterizes the (im)purity of an arbitrary of examples

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

- S is a sample of training examples
- $p_{\oplus}$ is the proportion of positive examples in S
- $p_{\ominus}$ is the proportion of negative examples in S



- Entropy specifies the minimum # of bits of information needed to encode the classification of an arbitrary member of *S*
- For example
  - The information required for classification of Table 3.2 $=-(9/14)\log_2(9/14)-(5/14)\log_2(5/14)=0.940$

Learning Algorithm (cont.)

- According to information theory
  - Optimal length code assigns $-\log_2 p$ bits to message having probability p
- General form of entropy

$$Entropy(s) \equiv \sum_{i=1}^{c} - p_i \log_2 p_i$$

c : Number of values.

p_i : The proportion of S belonging to class i

Learning Algorithm (cont.)

- Information gain and entropy

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

✓ *Values (A)*: the set of all possible values for attribute *A*

✓ S_v : the subset of *S* for which attribute *A* has value *v*

  – First term: the entropy of the original collection
  – Second term: the expected value of the entropy after *S* is partitioned using attribute *A*

- *Gain (S ,A)*
  – The expected reduction in entropy caused by knowing the value of attribute *A*
  – The information provided about the target function value, given the value of some other attribute *A*

Learning Algorithm (cont.)

- ID3 (*Examples*, *Target_attribute*, *Attributes*)
  - Create a *Root* node for the tree
  - If all *Examples* are positive, return the single node tree *Root*, with label= +
  - If all *Examples* are negative, return the single node tree *Root*, with label= −
  - If *Attributes* is empty, return the single-node tree *Root*, with label = most common value of *Target_attribute* in *Examples*
  - Otherwise begin

Continued to Next Slide →

Learning Algorithm (cont.)

- Otherwise begin
  - $A \leftarrow$ the attribute from *Attributes* that best classifies *Examples*
  - The decision attribute for *Root* $\leftarrow A$
  - For each possible value, v_i ,of A,
    - Add a new tree branch below *Root*, corresponding to the test $A = v_i$
    - Let $Examples_{v_i}$ be the subset of *Examples* that have value v_i for A
    - If $Examples_{v_i}$ is empty
      - » Then below this new branch add a leaf node with label = most common value of *Target_attribute* in *Examples*
      - » Else below this new branch add the subtree ID3($Examples_{v_i}$, *Target_attribute*, *Attributes* – {A})
- end
- Return *Root*

Learning Algorithm (cont.)

An Illustrative Example

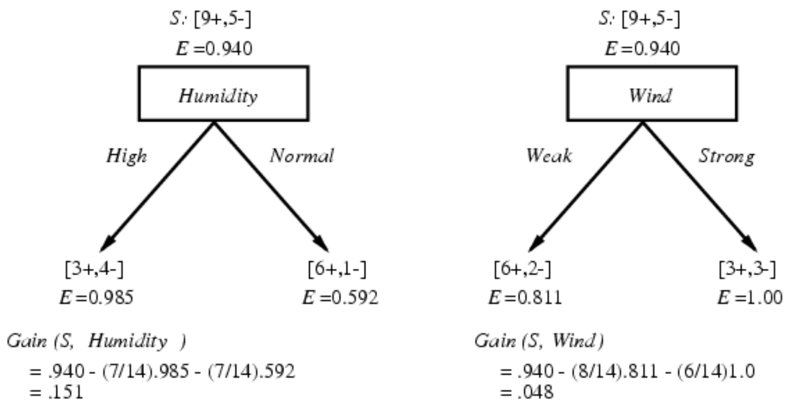
Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*Play Tennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Training examples for the target concept *PlayTennis*

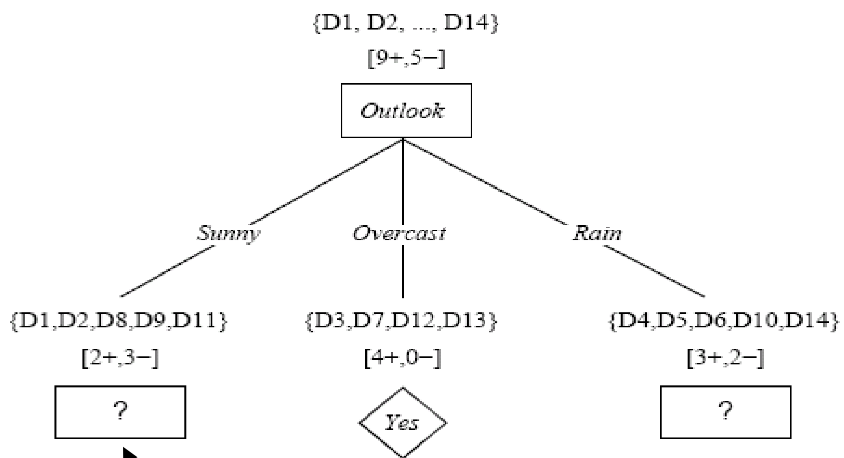
Learning Algorithm (cont.)

An Illustrative Example

- Selecting the root node
  - The information gain values for all four attributes
    - *Gain(S, Outlook)*= 0.246 🡸 selected as root attribute
    - *Gain(S, Humidity)*= 0.151
    - *Gain(S, Wind)*= 0.048
    - *Gain(S, Temperature)*= 0.029



- Adding a subtree



Which attribute should be tested here?

S_{sunny} = {D1,D2,D8,D9,D11}

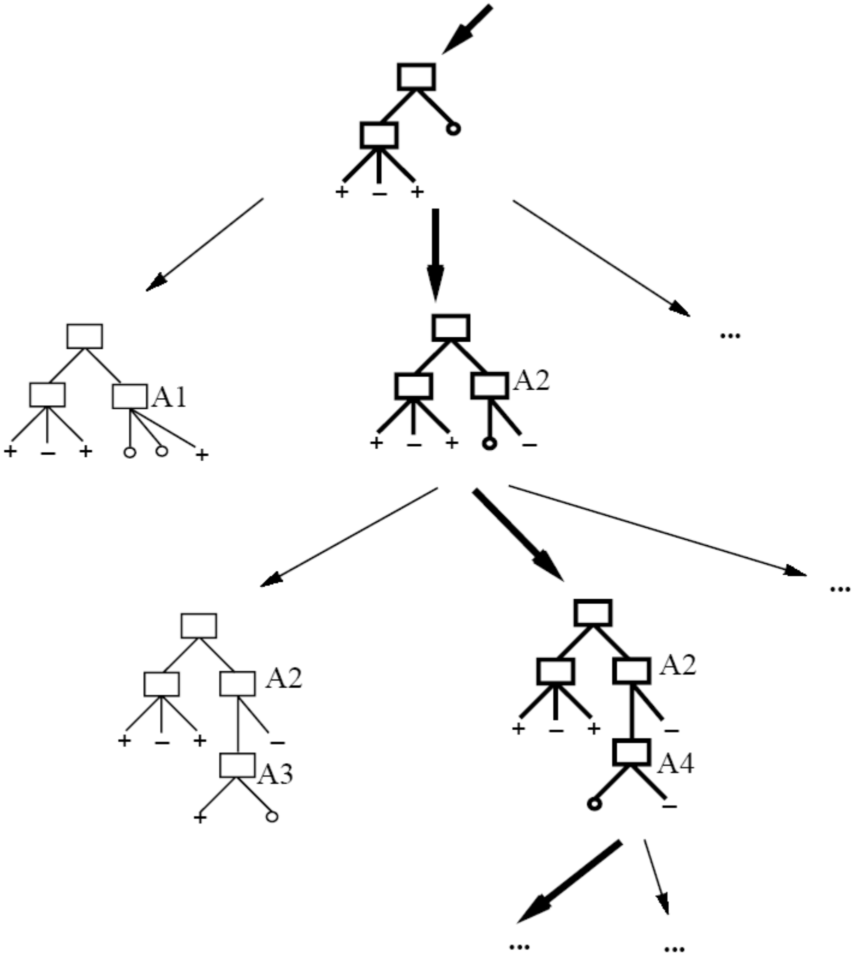
$Gain(S_{sunny}, Humidity) = .970 - (3/5)\, 0.0 - (2/5)\, 0.0 = .970$

$Gain(S_{sunny}, Temperature) = .970 - (2/5)\, 0.0 - (2/5)\, 1.0 - (1/5)\, 0.0 = .570$

$Gain(S_{sunny}, Wind) = .970 - (2/5)\, 1.0 - (3/5)\, .918 = .019$

Hypothesis Space Search

- Hypothesis space
  - The set of possible decision trees
  - Simple to complex, hill-climbing search



Hypothesis Space Search (cont.)

- Capability
  - Hypothesis space of all decision trees is a complete space of finite discrete-valued functions
  - ID3 maintains only a single current hypothesis
    - Can not determine how many alternative decision trees are consistent with the available training data
    - Can not pose new instance queries that optimally resolve among competing hypothesis
  - No backtracking in its search
    - Converging to local minima
  - ID3 uses all training example at each step to make statistically based decisions regarding how to refine its current hypothesis
    - The resulting search is much less sensitive to errors in individual training examples

Inductive Bias in Decision Tree Learning

- Note H is the power set of instances X
- Inductive Bias in ID3
  - Approximate inductive bias of ID3
    - Shorter trees are preferred over larger tress
    - BFS-ID3
  - A closer approximation to the inductive bias of ID3
    - Shorter trees are preferred over longer trees. Trees that place high information gain attributes close to the root are preferred over those that do not.

Inductive Bias in Decision Tree Learning (cont.)

- Difference between ID3 & C-E

ID3

- Searches a *complete* hypothesis space *incompletely*
- Inductive bias is solely a consequence of the ordering of hypotheses by its search strategy

Candidate-Elimination

- Searches an *incomplete* hypothesis space *completely*
- Inductive bias is solely a consequence of the expressive power of its hypothesis representation

Inductive Bias in Decision Tree Learning (cont.)

- Restriction bias and Preference bias

Preference bias
- ID3
- Preference for certain hypotheses over others
- Work within a complete hypothesis space

Restriction bias
- Candidate-Elimination
- Categorical restriction on the set of hypotheses considered
- Possibility of excluding the unknown target function

Inductive Bias in Decision Tree Learning (cont.)

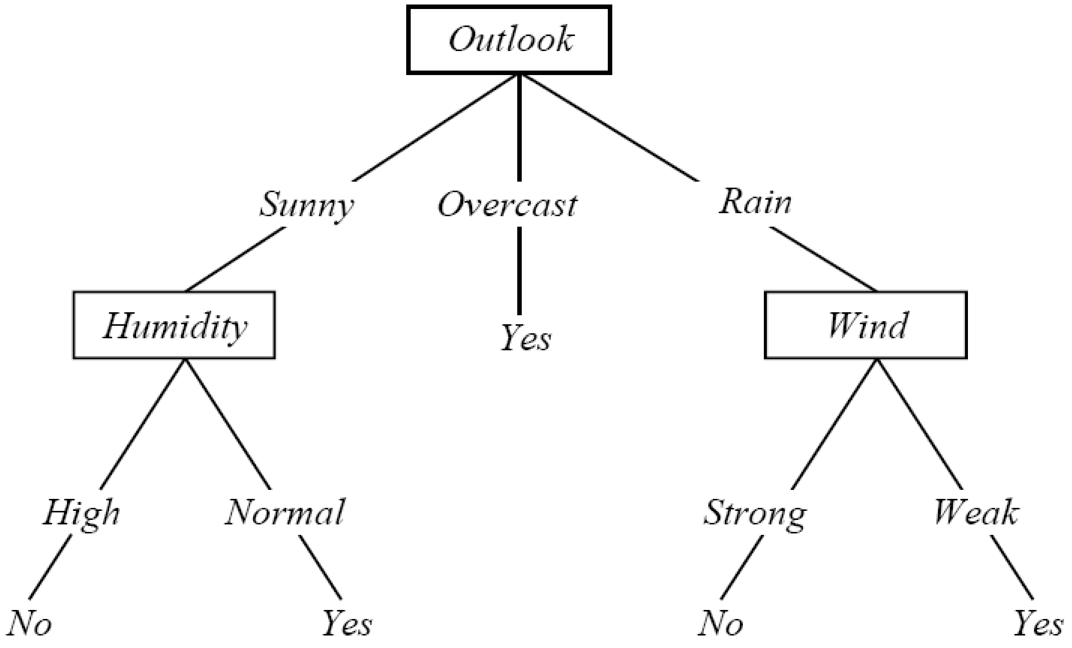
- Occam's razor
  - Prefer the simplest hypothesis that fits the data
  - Argument in favor
    - Fewer short hypotheses than long hypotheses
  - Argument opposed
    - There are many ways to define small sets of hypotheses
    - What's so special about small sets based on size of hypothesis?

Issues in Decision Tree Learning

- Determine how deeply to grow the decision tree
- Handling continuous attributes
- Choosing an appropriate attribute selection measure
- Handling training data with missing attribute values
- Handling attributes with differing costs
- Improving computational efficiency

Issues in Decision Tree Learning (cont.)

- Overfitting in decision trees
  - Consider adding noisy training example
    - *<Sunny, Hot, Normal, Strong, PlayTennis = No>*
    - What effect on earlier tree?



Issues in Decision Tree Learning (cont.)

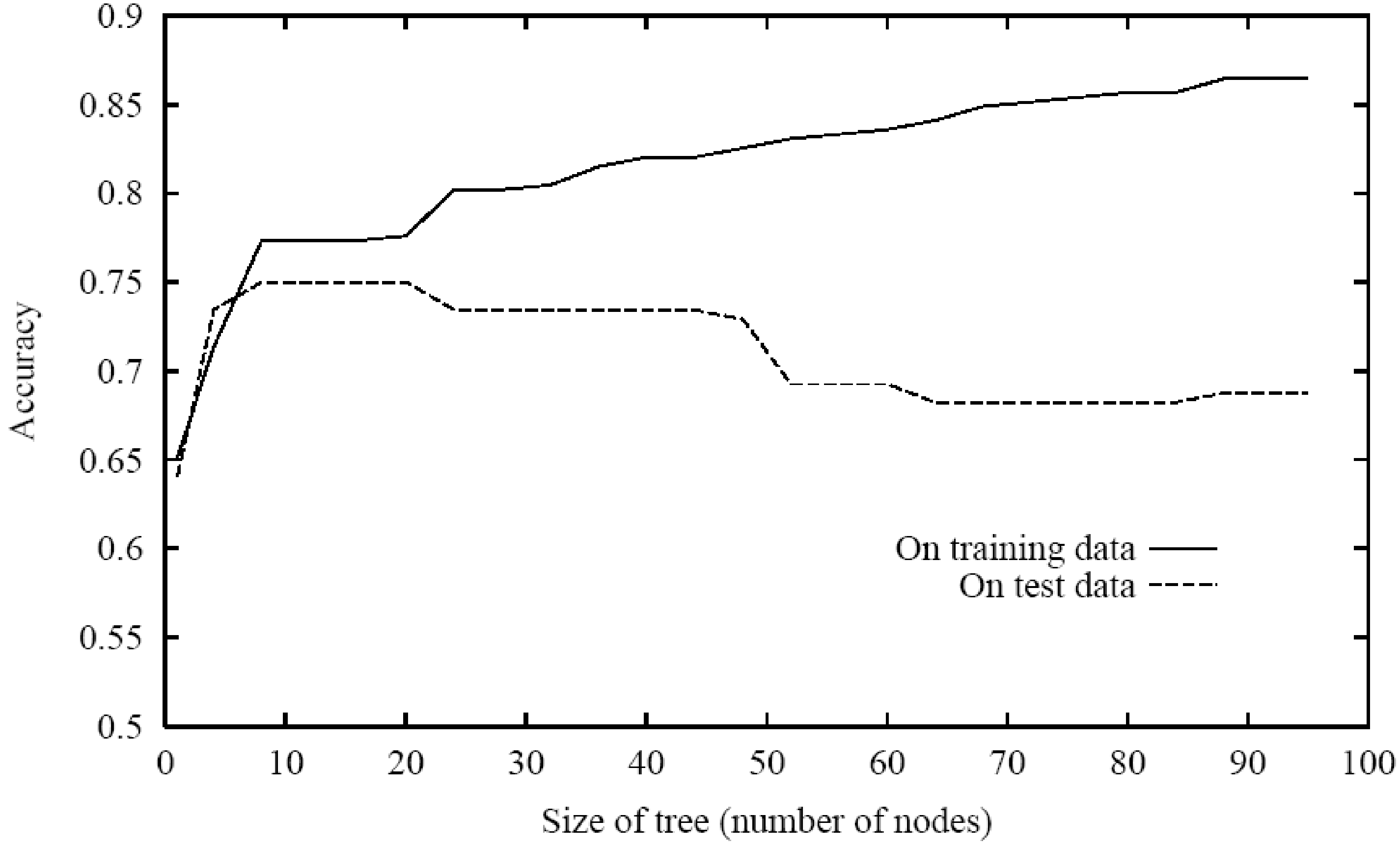
- Overfitting
  - Consider error of hypothesis h over
    - Training data: $error_{train}(h)$
    - Entire distribution D of data: $error_D(h)$
  - Hypothesis $h \in H$ overfits training data if there is an alternative hypothesis $h' \in H$ such that

$$error_{train}(h) < error_{train}(h')$$

and

$$error_D(h) > error_D(h')$$

Issues in Decision Tree Learning (cont.)



Overfitting in Decision Tree Learning

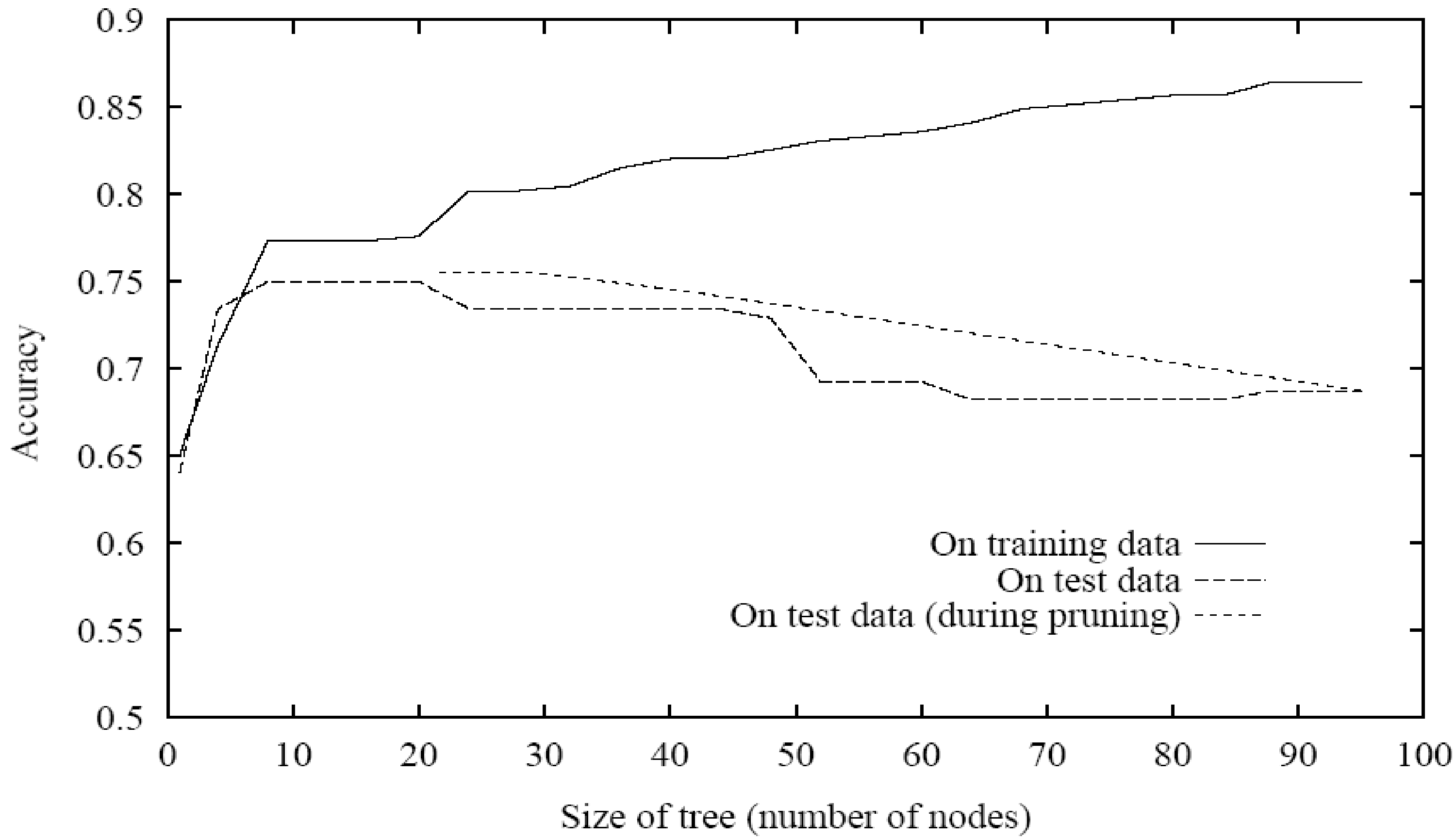
Issues in Decision Tree Learning (cont.)

- Avoiding overfitting
  - How can we avoid overfitting?
    - Stop growing before it reaches the point where it perfectly classifies the training data
    - Grow full tree, then post-prune
  - How to select best tree?
    - Measure performance statistically over training data
    - Measure performance over separate validation data set
    - MDL: minimize the complexity for encoding the training examples and the decision tress

Issues in Decision Tree Learning (cont.)

- Reduced-error pruning
  - Split data into training set, validation set used for pruning, and test set for measuring accuracy over future unseen examples.
  - Do until further pruning is harmful:
    1. Evaluate impact on *validation* set of pruning each possible node (plus those below it), starting at its maximum size and lowest accuracy over test set.
    2. Greedily remove the one that most improves *validation* set accuracy
  - Produces smallest version of most accurate subtree
  - What if data is limited?

Issues in Decision Tree Learning (cont.)



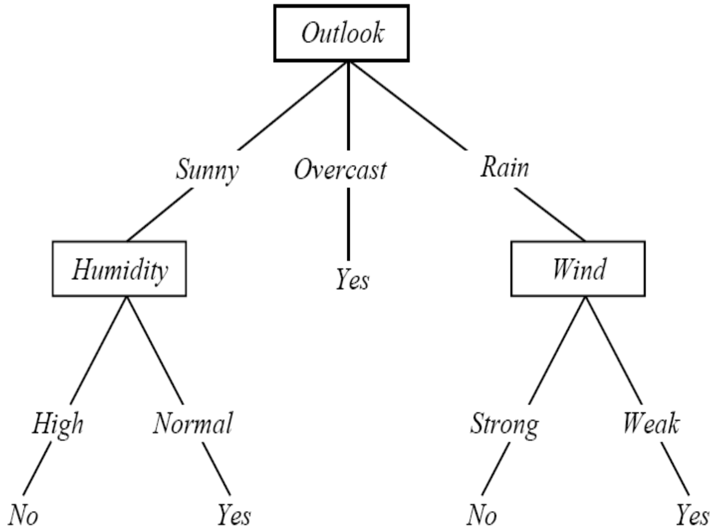
Effect of Reduced-Error Pruning in Decision Tree Learning

Issues in Decision Tree Learning (cont.)

- Rule post-pruning
  - Most frequently used method (e.g., ch.4.5)
    1. Convert tree to equivalent set of rules
    2. Prune each rule independently of others
    3. Sort final rules into desired sequence for use
  - In C4.5, evaluation of performance is based on the training set itself, using pessimistic estimate :
    - Calculate the rule accuracy over the training set
    - Calculate the standard deviation in this estimated accuracy assuming on a binomial distribution.
    - The lower-bound estimate is taken as the measure of rule performance for a given confidence level.

Issues in Decision Tree Learning (cont.)

- Converting a tree to rules



IF *(Outlook = Sunny)* $\wedge$ *(Humidity = High)*

THEN *PlayTennis = No*

IF *(Outlook = Sunny)* $\wedge$ *(Humidity = Normal)*

THEN *PlayTennis = Yes*

…

Advantages of rule post-pruning over reduced-error Pruning

- Allows distinguishing among the different contexts in which a decision node is used.
- Removes the distinction between attributes near the root and those near the leaves.
- Improves readability.

Issues in Decision Tree Learning (cont.)

- Continuous valued attributes
  - Define new discrete valued attributes that partition the continuous attribute value into a discrete set of intervals

Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

  - Find a set of thresholds midway Between different target values of the attribute : $Temperature_{>54}$ and $Temperature_{>85}$
  - Pick a threshold, *c*, that produces the greatest information gain : $temperature_{>54}$

Issues in Decision Tree Learning (cont.)

- Attributes with many values
  - Problem
    - If attribute has many values, *Gain* will select it
    - Imagine using *Date = Oct_13_2004* as attribute
  - One approach: use *GainRatio* instead

$$GainRatio(S,A) \equiv \frac{Gain(S,A)}{SplitInformation(S,A)}$$

$$SplitInformation(S,A) \equiv -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

where S_i is subset of S for which A has value v_i

(What if $|S_i|$ is much closer to $|S|$? SplitInformation(S,A) becomes very small so that Attribute A would be selected with large value of GainRatio(S,A) even when Gain(S,A) is small.)

Issues in Decision Tree Learning (cont.)

- Unknown attribute values
  - What if some examples missing values of *A*?

  Use training examples, sort through tree:

  - If node *n* tests *A*, assign most common value of *A* among other examples sorted to node *n*
  - Assign most common value of *A* among other examples with same target value
  - Assign probability p_i to each possible value v_i of *A* and classify new examples in same fashion

Issues in Decision Tree Learning (cont.)

- Attributes with costs
  - Use low-cost attributes where possible, relying on high-cost attributes only when needed to produce reliable classficiations
  - Tan and Schlimmer (1990)

$$\frac{Gain^2(S,A)}{Cost(A)}$$

  - Nunez (1988)

$$\frac{2^{Gain(S,A)}-1}{(Cost(A)+1)^w}$$

where $w \in [0, 1]$ determines importance of cost

Issues in Decision Tree Learning (cont.)

- Enhancements in C4.5
  - Allows for attributes that have a whole range of discrete or continuous values
  - Post-pruning after induction of trees, e.g. based on test sets, in order to increase accuracy
  - Uses gain ratio as the information gain measure to replace the old biased method
  - Handles training data with missing attribute values by replacing them with the most common or the most probable value

Summary

- Practical method using greedy search for concept learning and for learning other discrete-valued functions
- ID3 searches a complete hypothesis space
- Preference for smaller trees
- Overfitting the training data
- Large variety of extensions to the basic ID3