Decision Tree Learning

Berlin Chen 2004

References:

1. Machine Learning , Chapter 3
2. Data Mining: Concepts, Models, Methods and Algorithms, Chapter 7
3. Tom M. Mitchell's teaching materials

What is Decision Tree Learning ?

- Decision tree learning is a method for approximating discrete-valued target functions (classification results)
  - The learned function is represented by a decision tree
  - Decision trees also can be re-represented as sets of if-then rules to improve human readability

- Decision tree learning is a kind of inductive learning
  - Belongs to the logical model
    - No assumption of distributions of examples
  - Classification is done by applying Boolean and comparative operators to the feature values
  - A supervised learning method

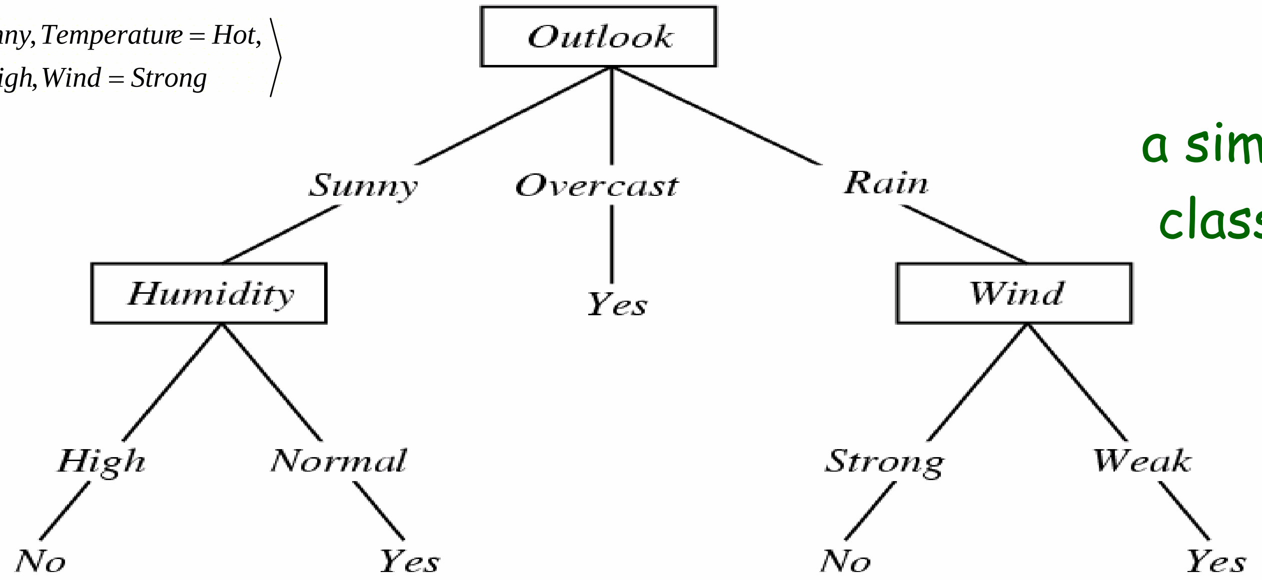
What is a Decision Tree ?

- Decision tree representation
  - Each internal node tests an attribute
    - Some test to be carried out
  - Each branch corresponds to attribute value
    - Outcome of the test on a given attribute
  - Each leaf node assigns a classification
    - Indication of a class
- Decision trees are usually generated in a top-down manner
  - Greedy search methods are employed
    - No-backtracking

What is a Decision Tree ?

- Decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances
  - Each path from the tree root to a leaf corresponds to a conjunction of attribute tests (a classification rule)

$$\left\langle \begin{array}{l} Outlook = Sunny, Temperature = Hot, \\ Humidity = High, Wind = Strong \end{array} \right\rangle$$



a simple Boolean classification

$$\begin{array}{l} (Outlook = Sunny \wedge Humidity = Normal) \\ \vee (Outlook = Overcast) \\ \vee (Outlook = Rain \wedge Wind = Weak) \end{array}$$

Graphical Representation of a Classification Problem

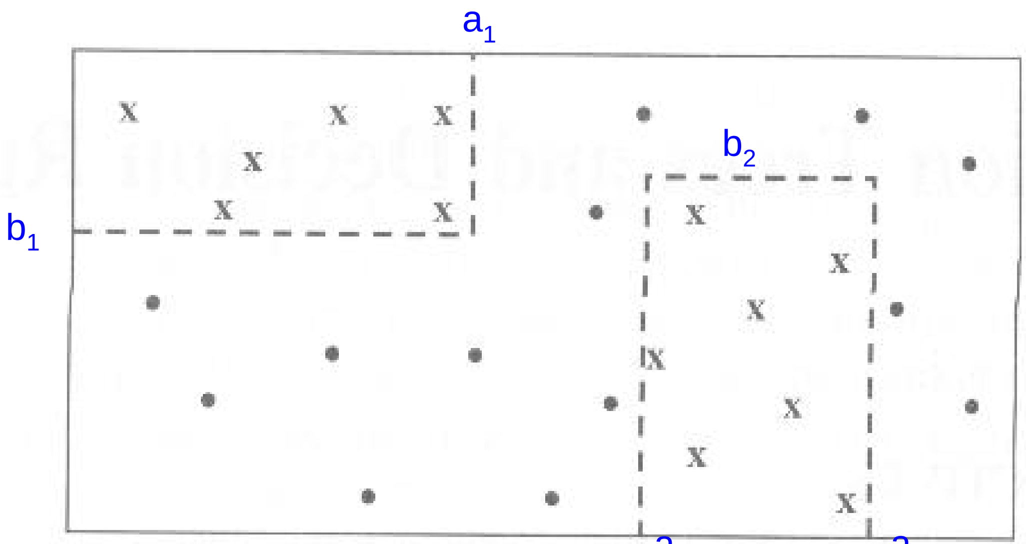


FIGURE 7.1 Classification of samples in a 2D space

– One or more hypercubes stand for a given class
  - OR-ed all the cubes to provide a complete classification for a class
  - Within a cube the conditions for each part are AND-ed

When to Consider Decision Trees

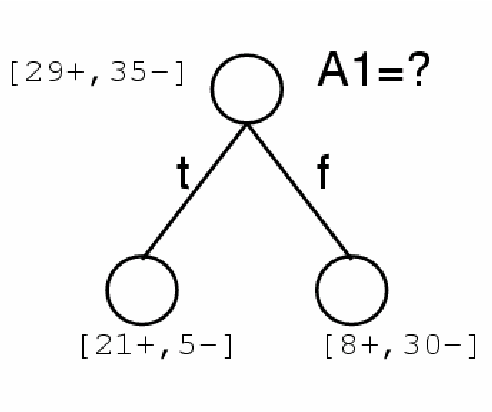
- Instances describable by attribute-value pairs
  - Symbolic or real-valued attribute
- Target function is discrete valued
- Disjunctive hypothesis may be required
- Possibly noisy training data
- Training data containing missing attribute values
- Examples
  - Equipment or medical diagnosis
  - Credit risk analysis
  - Modeling calendar scheduling preferences

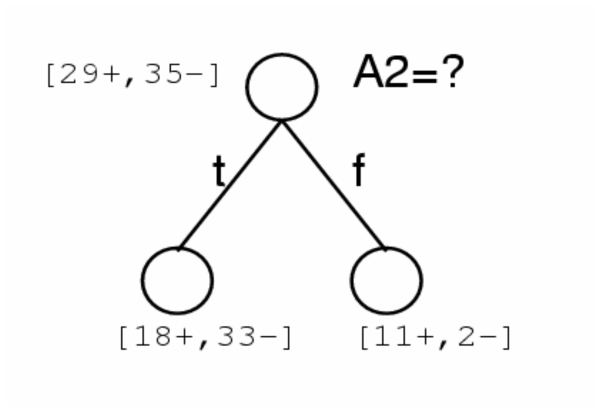
Key Requirements for Decision Trees

- Attribute-vale description
  - A fixed collection of properties or attributes
  - Attribute description must not vary from one case to another
- Predefined classes
  - Categorical assignments must be established beforehand
  - Again, DTL is supervised learning
  - A case can only belong to a particular class
- Sufficient data
  - Enough number of patterns can be distinguished from chance coincidences

Top-Down Induction of Decision Trees

- Main loop (of the ID3 algorithm)
  - A $\leftarrow$ the “best” decision attribute for next node
  - Assign A as decision attribute for node
  - For each value of A create new descendant of node
  - Sort training examples to (new) leaf nodes
  - If training examples perfectly classified Then STOP Else iterate over new leaf nodes
- Which attribute is best ?





ID3 Algorithm

ID3(*Examples*, *Target_attribute*, *Attributes*)

Examples are the training examples. Target_attribute is the attribute whose value is to be predicted by the tree. Attributes is a list of other attributes that may be tested by the learned decision tree. Returns a decision tree that correctly classifies the given Examples.

- 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
    - $A \leftarrow$ 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*

Review: Entropy

- Three interpretations for quantity of information
  1. The amount of **uncertainty** before seeing an event
  2. The amount of **surprise** when seeing an event
  3. The amount of **information** after seeing an event

- The definition of information: $define \quad 0\log_2 0 = 0$

$$I(x_i) = \log_2 \frac{1}{P(x_i)} = -\log_2 P(x_i)$$

  – $P(x_i)$ the probability of an event x_i

- Entropy: the average amount of information

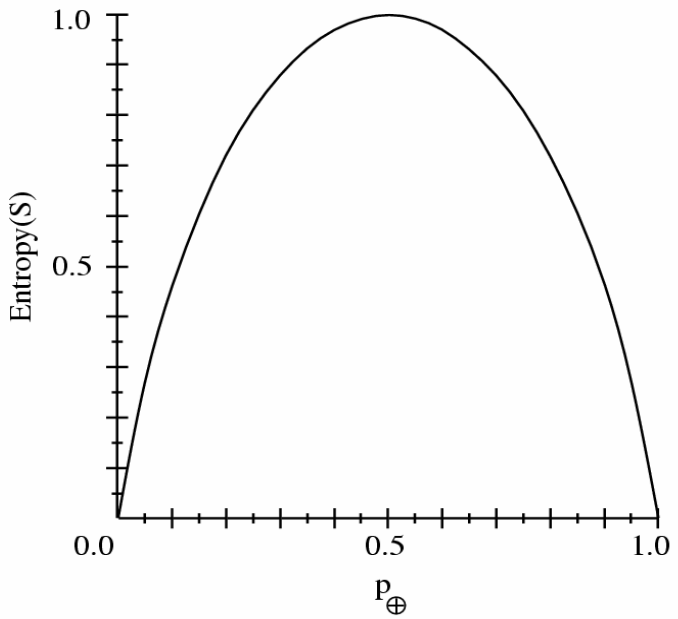
$$H(X) = E[I(X)]_X = E[-\log_2 P(x_i)]_X = \sum_{x_i} - P(x_i) \cdot \log_2 P(x_i)$$

where $X = \{x_1, x_2, \ldots, x_i, \ldots\}$

  – Have maximum value when the probability (mass) function is a uniform distribution

Review: Entropy

- For Boolean classification (0 or 1)



$$Entropy(X) = -p_1 \log_2 p_1 - p_2 \log_2 p_2$$

- Entropy can be expressed as the minimum number of bits of information needed to encode the classification of an arbitrary number of examples
  - If c classes are generated, the maximum of Entropy can be

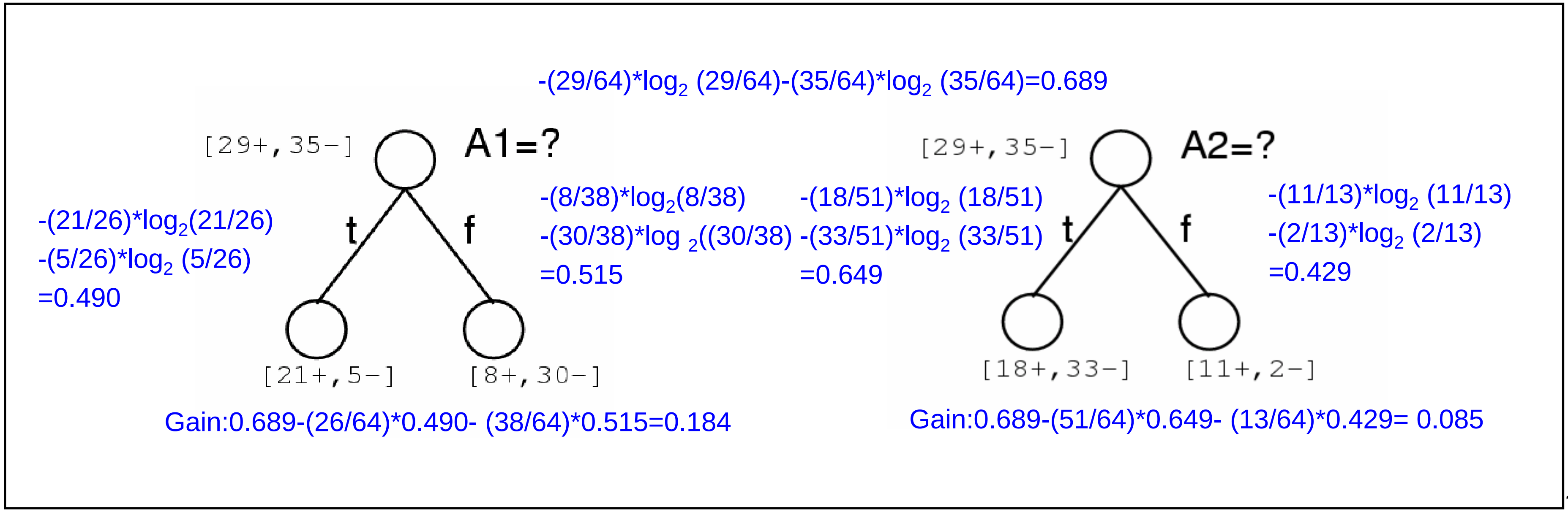
$$Entropy(X) = \log_2 c$$

Information Gain

- Gain(S, A)=expected reduction in entropy due to sorting/partitioning on A

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

weighted sum of entropies over the subsets



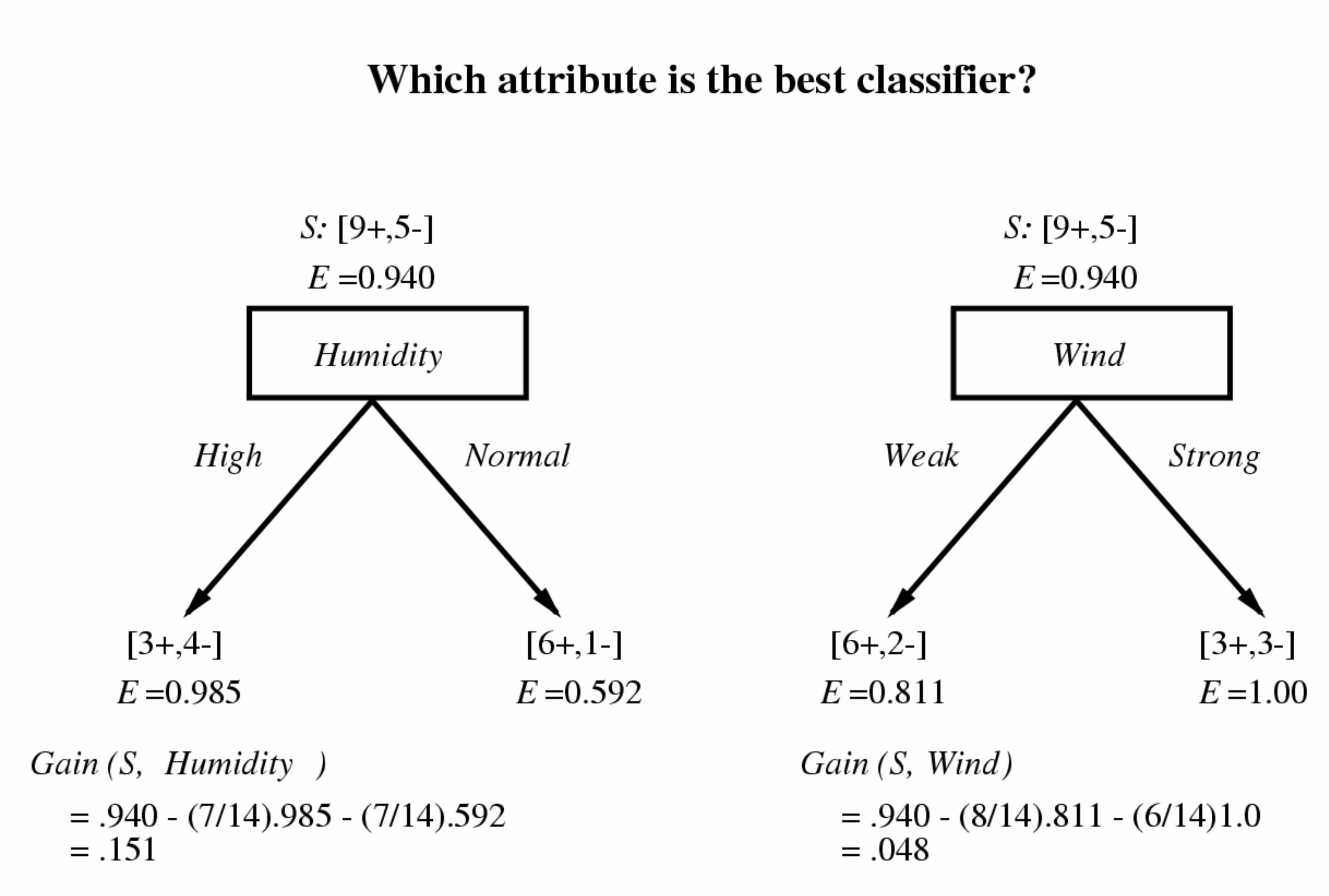
An Illustrative Example

- Target Attribute: *PlayTennis*

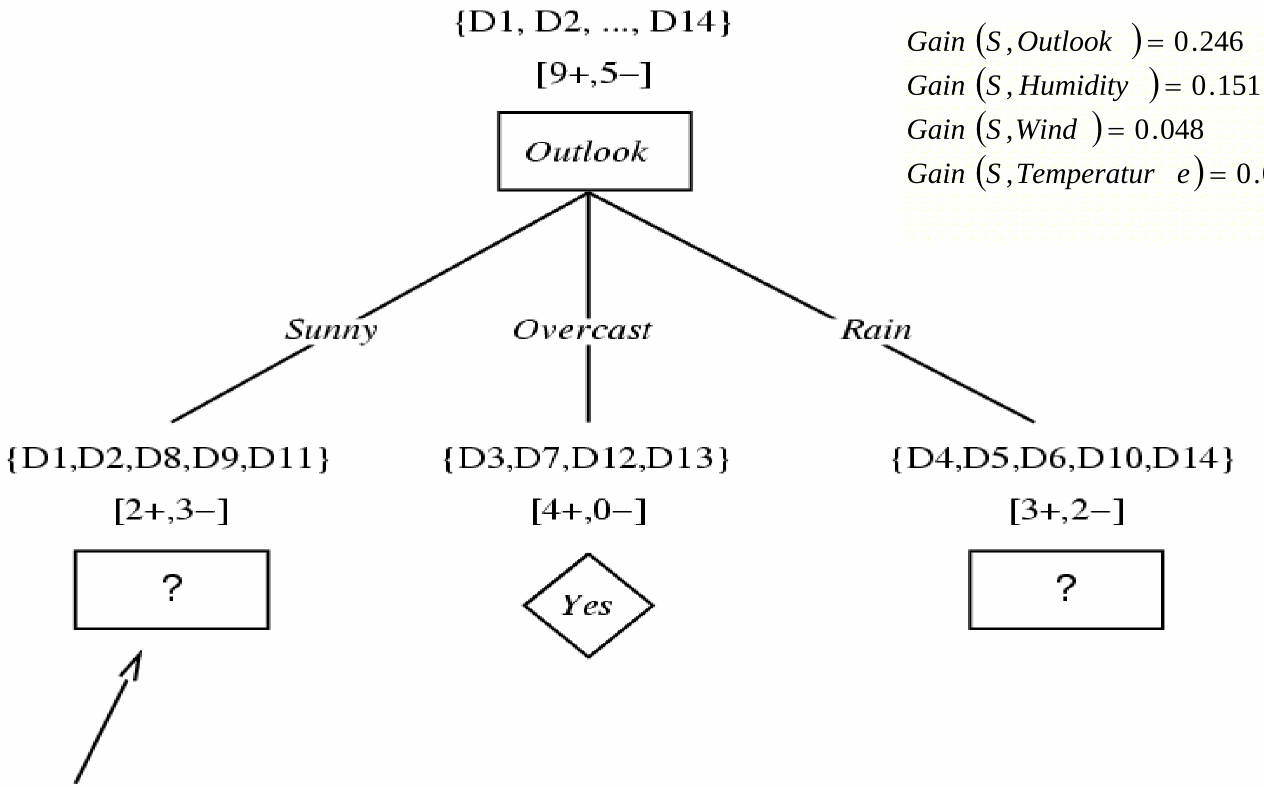
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	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

An Illustrative Example

- Select the Next Features
  - For example, two different attributes are considered



An Illustrative Example



$Gain(S, Outlook) = 0.246$ ✓

$Gain(S, Humidity) = 0.151$

$Gain(S, Wind) = 0.048$

$Gain(S, Temperature) = 0.029$

Which attribute should be tested here?

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

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

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

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

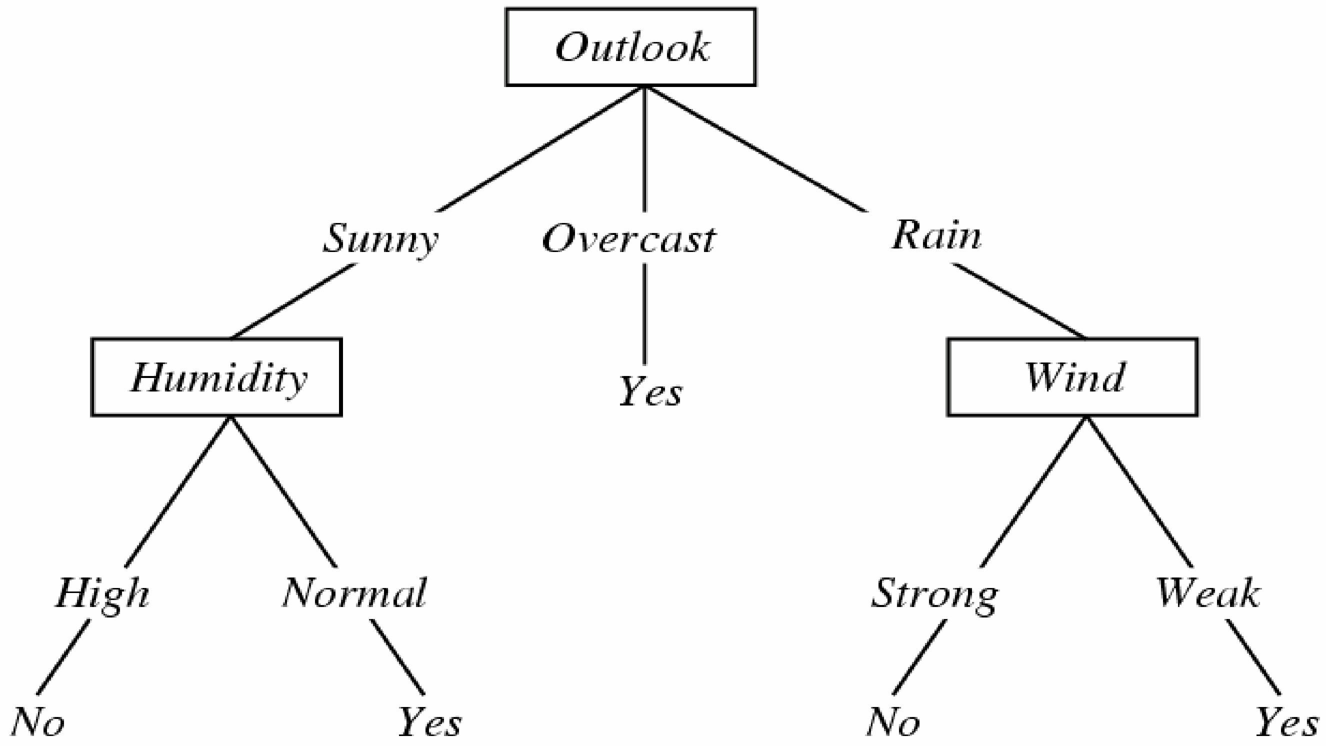
An Illustrative Example

- The process of selecting a new attribute and partitioning the training examples is repeated for each nonterminal descendant node
  - Use the training samples associated with that node
  - Use the attributes that haven't been used along the path through the tree

- The process terminates when either the following two conditions is met for each new leaf node
  - Every attribute has already been included along the path through the tree
  - The training examples associated with this leaf node have the same target attribute value (entropy is zero)

An Illustrative Example

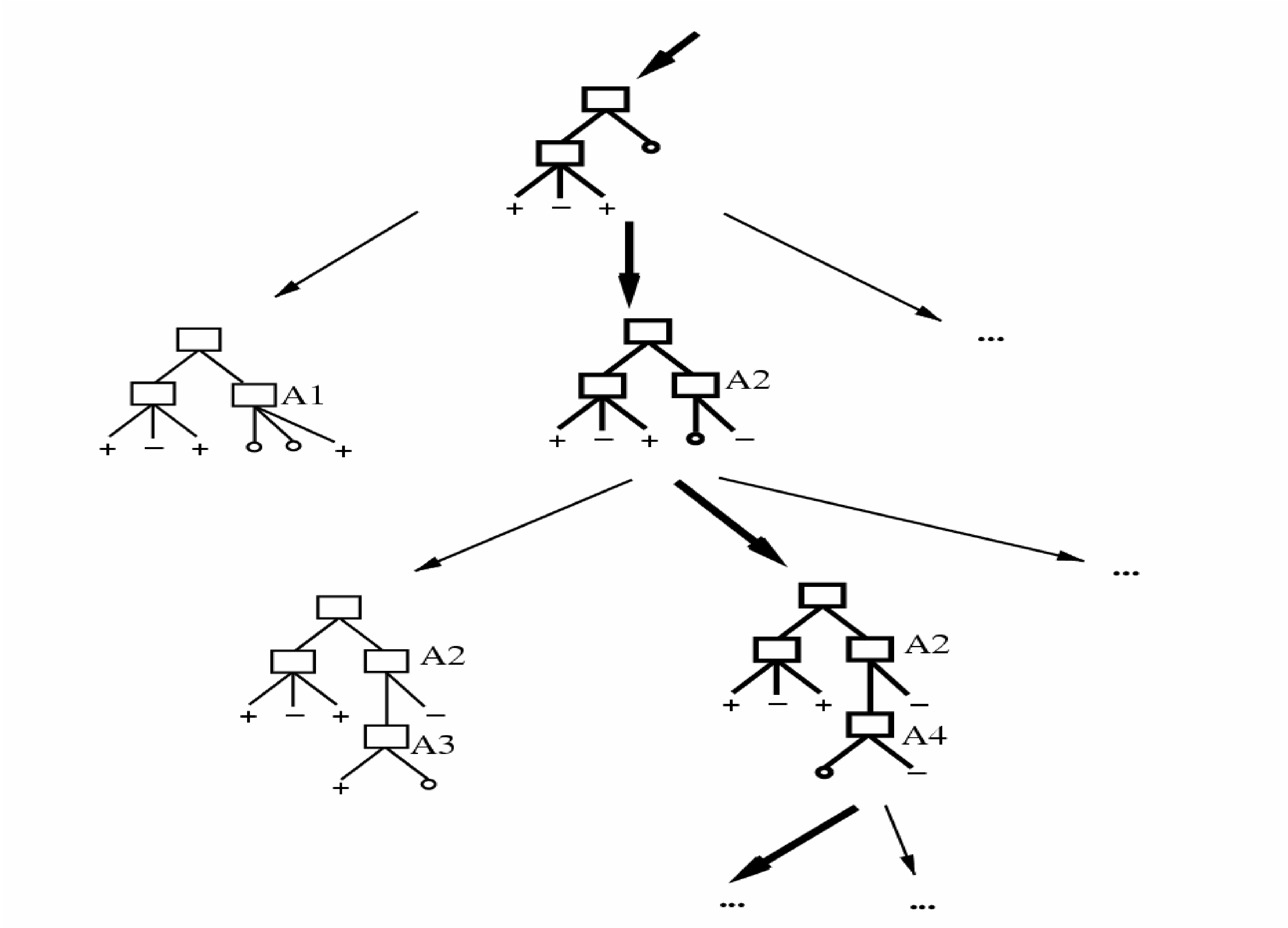
- The final decision tree



Hypothesis Space Search by ID3

- Hypothesis space is complete
  – Target function surely in there
- Outputs a single hypothesis (which one?)
  – Can not explicit represent all consistent hypotheses
- No backtracking
  – Output a locally optimal solution (not globally optimal)
- Statistically based search choices
  – Robust to noisy data
  – Use the statistical properties of all samples, do not make decisions incrementally based on individual training examples
- Inductive bias
  – Implicitly select in favor of short trees over longer ones

Hypothesis Space Search by ID3



Inductive Bias in ID3

- Inductive bias
  - The set of assumptions that, together with the training data, deductively justify the classifications assigned by the learner to further instances
- Inductive bias exhibited by ID3
  - As mentioned, select in favor of short trees over longer ones
  - Select trees that place the attributes with highest information gain closest to the root
- Again, ID3 can be characterized as follows
  - A greedy search using the information gain heuristic
  - Does not always find the shortest consistent tree
  - No backtracking

Restriction Biases and Preference Biases

- Version Space Candidate-Elimination Algorithm
  - An incomplete hypothesis space (only a subset of hypotheses is expressed) introduces a hard restriction bias (or a language bias)
  - A complete search strategy introduces no bias
- ID3
  - A complete hypothesis space introduces no bias
  - An incomplete search strategy introduces a preference bias (or a search bias)
- Learning the numerical evaluation for Checkers
  - A linear combination of a fixed set of board features $\rightarrow$ a restriction bias
  - LMS algorithm $\rightarrow$ a preference bias
- A preference bias is more desirable than a restriction bias

Occam's Razor

- Why prefer short hypotheses ?

- Argument in favor
  - Fewer short hypotheses than long hypotheses
    - A short hypothesis that fits data unlikely to be coincidence
    - A long hypothesis that fits data might be coincidence

Issues in Decision Tree Learning

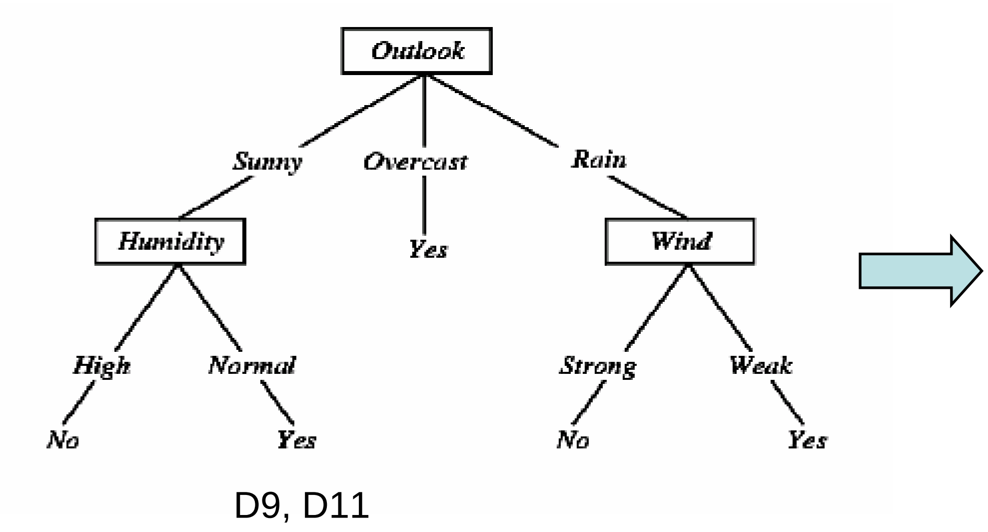
- Avoiding Overfitting the Data
- Incorporating Continuous-Valued Attributes
- Alternative Measures for Selecting Attributes
- Handling Training Examples with Missing Attribute Values
- Handling Attributes with Differing Costs

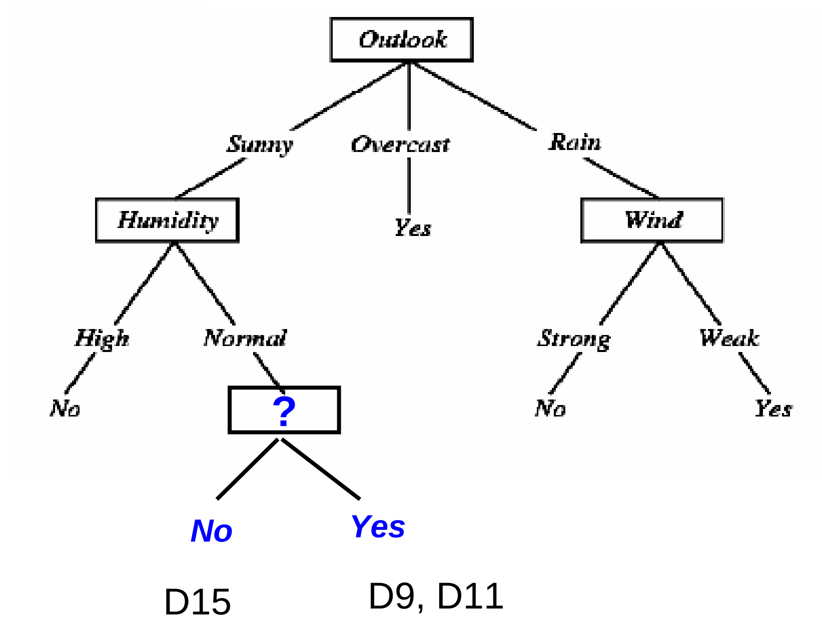
Overfitting in Decision Trees

- Consider adding a noisy training example, D15
  - *Sunny*, *Hot*, *Normal*, *Strong*, *PlayTennis=No*

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
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

- What effect on earlier tree ?





  - The random noise introduced in the training examples can lead to overfitting

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 hyopthesis $h' \in H$ such that

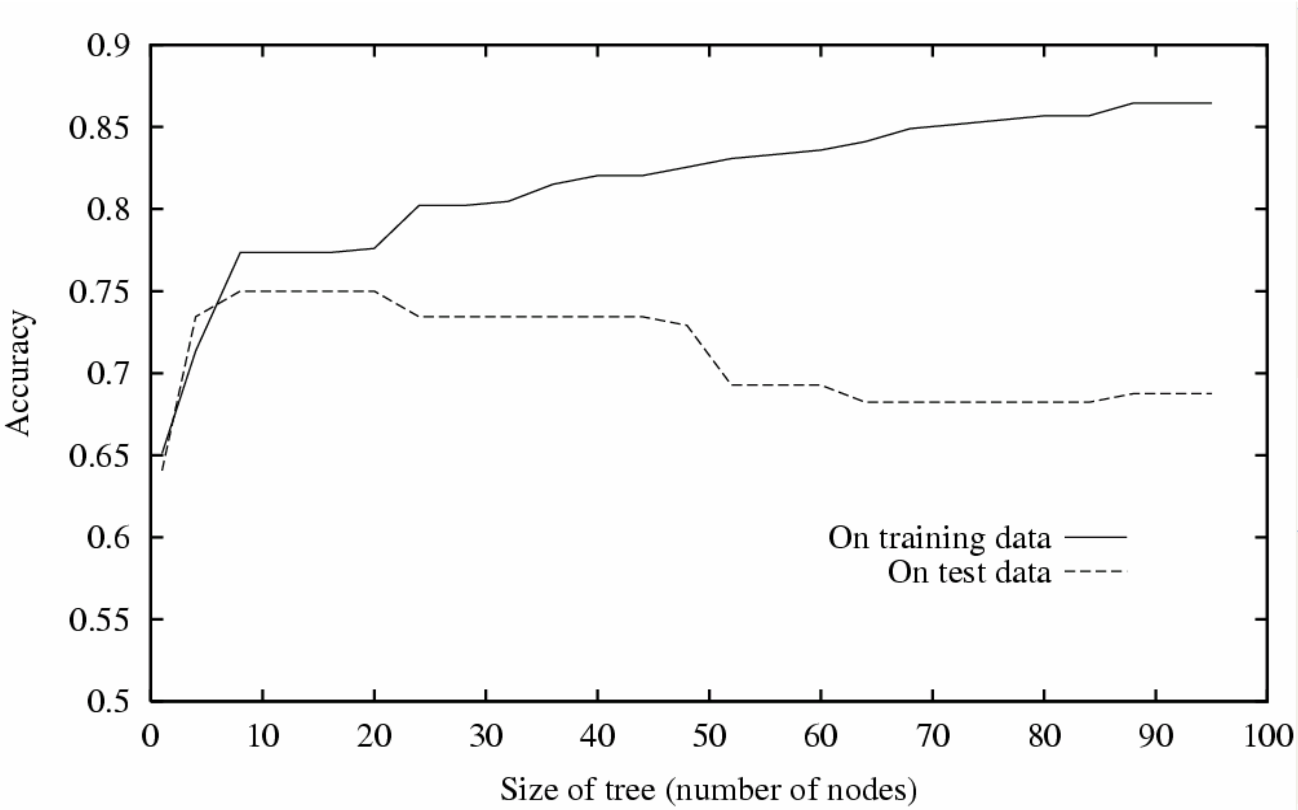
$$error_{train}(h) < error_{train}(h')$$

and

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

Overfitting in Decision Tree Learning

- Example: Prediction of Diabetes



  - Accuracy measured over training example increases monotonically
  - Accuracy measured over independent test example first increases then decreases

Pruning Decision Trees

- Remove parts of the decision tree (subtrees) that do not contribute to the classification accuracy of unseen testing samples (mainly because of overfitting)
  - Produce a less complex and more comprehensible tree

- Two ways
  - Stop growing when data split not statistically significant (earlier stop before perfection classification of training data)
    - Prepruning
    - Hard to estimate precisely
  - Grow full tree, then post-prune the tree
    - Much more promising

Avoiding Overfitting

- How to select best tree (correct final tree size)?
  - Measure performance over separate validation data set (training and validation set approach)
  - Measure performance over training data
    - Statistical tests, e.g., if there are no significant different in classification accuracy before and after splitting, then represent a current node as a leaf (called prepruning)
  - MDL (Minimum Description Length) minimize ?
    - *size*(*tree*)+*size*(*misclassifications*(*tree*))

Reduced-Error Pruning

- Split data into *training* (2/3) and *validation* (1/3) set, and do until further pruning is harmful:

    1. Evaluate impact on validation set of pruning each possible node (plus those below it)

    2. Greedily remove the one that most improves validation set accuracy

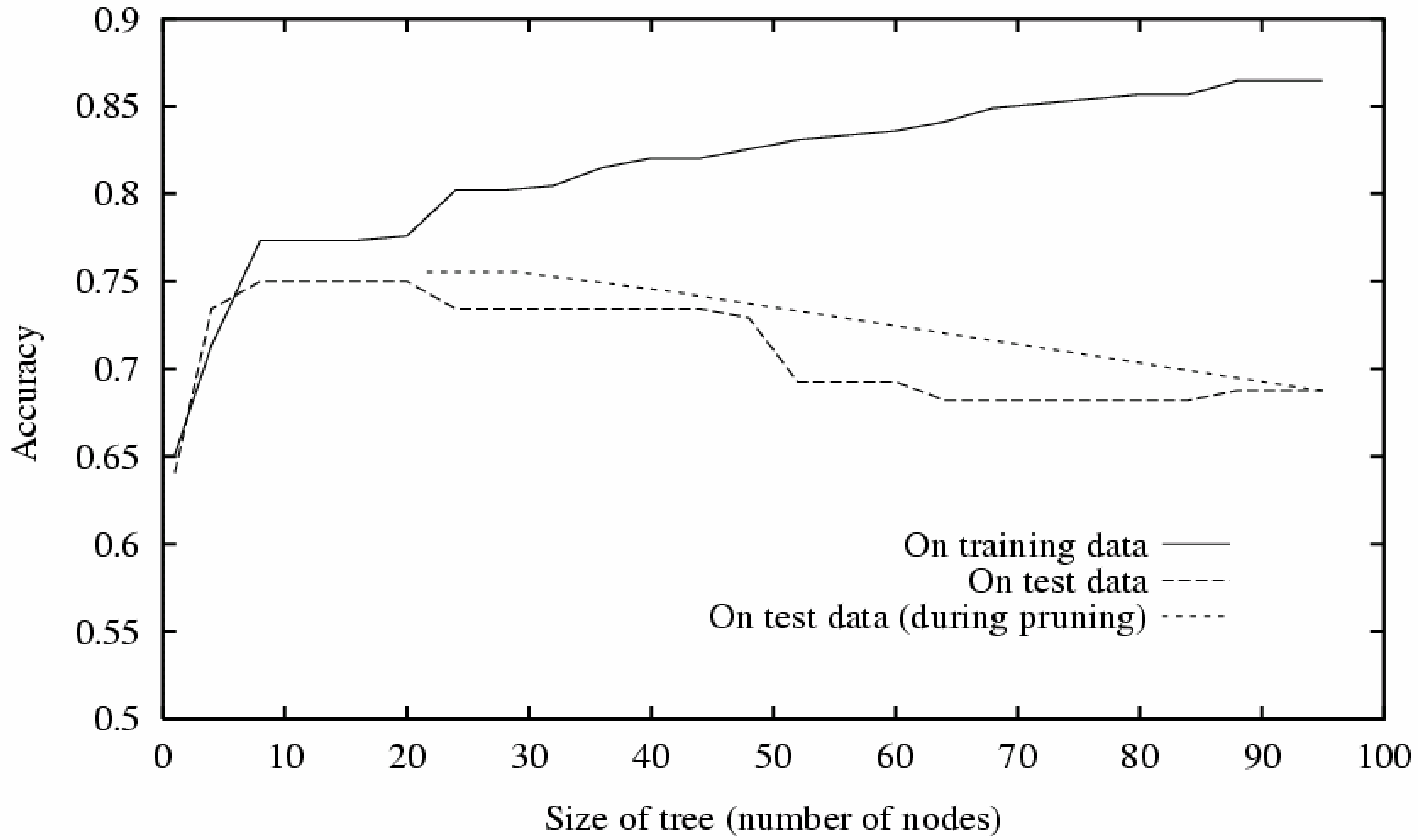
        – Prune leaf nodes added due to coincidental regularities in the training set

Produces smallest version of most accurate subtree

What if data is limited ?

Effect of Reduced-Error Pruning

- Split data into three subsets
  - Training, Validation, Test



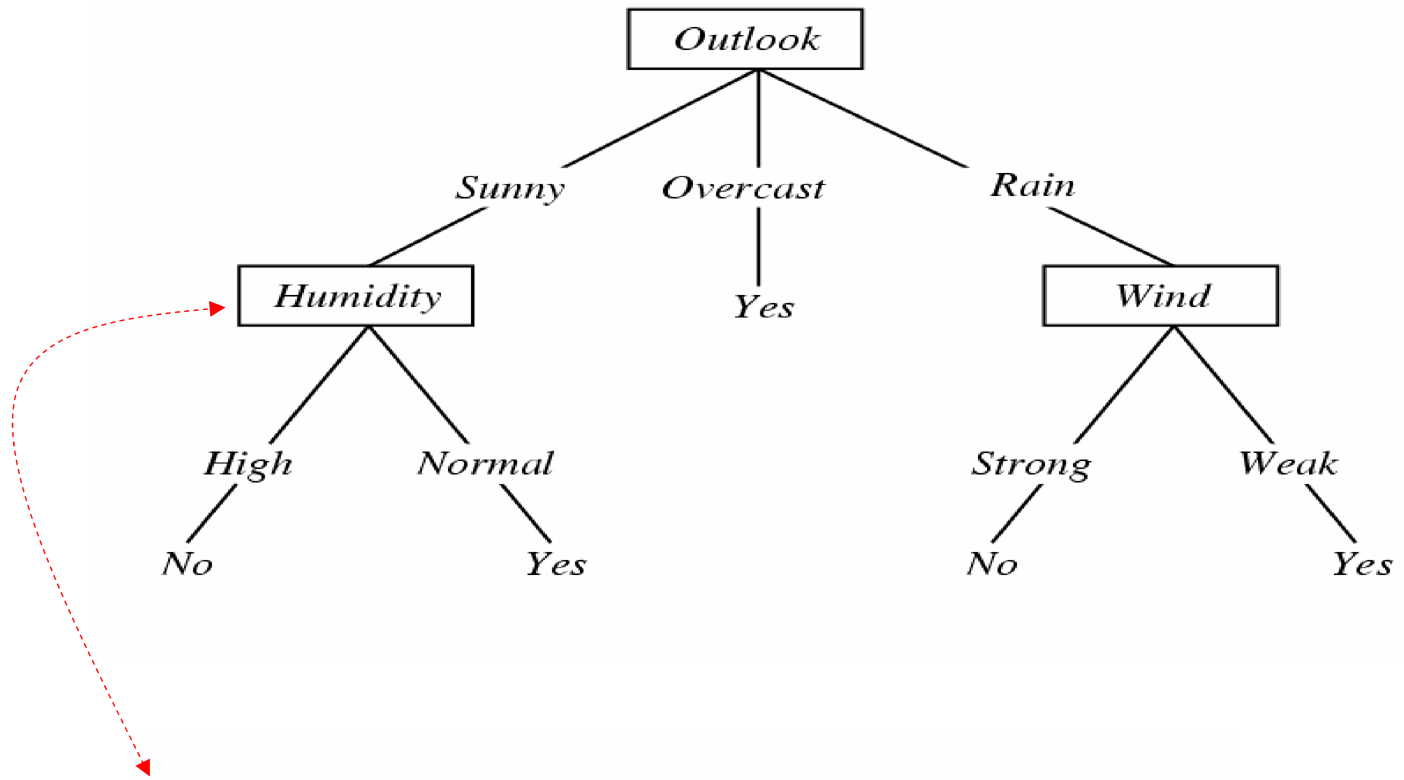
Rule Post-Pruning

1. Convert tree to equivalent set of rules
2. Prune (generalize) each rule independently of others
3. Sort final rules into desired sequence for use

Perhaps most frequently used method (e.g., C4.5)

IF $(Outlook = Sunny) \wedge (Humidity = High)$
THEN $PlayTennis = No$

Converting A Tree to Rules



IF $(Outlook = Sunny) \wedge (Humidity = High)$
THEN $PlayTennis = No$

IF $(Outlook = Sunny) \wedge (Humidity = Normal)$
THEN $PlayTennis = Yes$

Incorporating Continuous-Valued Attributes

- Create a discrete attribute to test continuous
  - *Temperature* = 82.5
  - (*Temperature* > 72.3) = *t*, *f*

- Split into two intervals

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

(48+60)/2
=54

(80+90)/2
=85

  - Candidate thresholds evaluated by computing the information gain associated with each

- Split into multiple intervals

Attributes with Many Values

- Problem:
  - If attribute has many values, *Gain* will select it
  - Imagine using Date= Jun_3_1996 as attribute
    - Training set separated into very small subsets
    - Have highest information gain

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
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

- One approach: use *GainRatio* instead

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

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

Entropy of S with respect to the values of attribute A

  - Where S_i is subset of S for which A has value v_i
  - *SplitInformation* discourages the selection of attributes with many uniformly distributed values

Attributes with Costs

- Instance attributes may have associated costs
- How to learn a consistent tree with low expected cost ?
- One approach: replace gain by
  - 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

introduce a cost term into the attribute selection measure
- Low-cost attributes preferred
- No guarantee to find optimal DTL

Unknown Attribute Values

- What if some examples missing values of *A* ?
  - In a data set, some attribute values for some examples can be missing, for example, because that
    - The value is not relevant to a particular examples
    - The value is not recorded when the data was collected
    - An error was made when entering data into a database

- Two choices to solve this problem
  - Discard all examples in a database with missing data
    - What if large amounts of missing values exists ?

  - Define a new algorithm or modify an existing algorithm that will work with missing data

Unknown Attribute Values

- One approach: Use training example anyway sort through tree
  - Fill a missing value with most probable value
    - 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 sorted to node *n* with same target value
  - Fill a missing value based on the probability distribution of all values for the given attribute
    - Assign probability p_i to each possible value v_i of *A* at node *n*
      - Assign fraction p_i of example to each descendant in tree
- Also, the unseen test data with missing attribute values can be classified in similar fashion

Unknown Attribute Values

- Example

TABLE 7.2. A simple flat database of examples with one missing value

Database T:

Attribute1	Attribute2	Attribute3	Class
A	70	True	CLASS1
A	90	True	CLASS2
A	85	False	CLASS2
A	95	False	CLASS2
A	70	False	CLASS1
?	90	True	CLASS1
B	78	False	CLASS1
B	65	True	CLASS1
B	75	False	CLASS1
C	80	True	CLASS2
C	70	True	CLASS2
C	80	False	CLASS1
C	80	False	CLASS1
C	96	False	CLASS1

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

F : no. of examples with a known value for a given attribute divided by total no. of examples

$$Entropy(S) = -8/13 \log_2(8/13) - 5/13 \log_2(5/13) = 0.961$$

$$\sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$
$$= 5/13(-2/5 \log_2(2/5) - 3/5 \log_2(3/5)) + 3/13(-3/3 \log_2(3/3) - 0/3 \log_2(0/0)) + 5/13(-3/5 \log_2(3/5) - 2/5 \log_2(2/5)) = 0.747$$

$$Gain(S,A) = 13/14(0.961 - 0.747) = 0.199$$

$$SplitInformation(S,A) = -(5/14 \log 5/14 + 3/14 \log 3/14 + 5/14 \log 5/14 + 1/14 \log 1/14) = 1.876$$

$$GainRatio(S,A) = \frac{0.199}{1.876}$$

Treat the example with missing value as a specific group

Unknown Attribute Values

- Example (cont.)
  - If node *n* tests *A,* assign most common value of *A* among other training examples sorted to node *n*

T_1: (*Attribute1 = A*)

Att.2	Att.3	Class	w
70	True	CLASS1	1
90	True	CLASS2	1
85	False	CLASS2	1
95	False	CLASS2	1
70	False	CLASS1	1
90	*True*	*CLASS1*	*5/13*

T_2: (*Attribute1 = B*)

Att.2	Att.3	Class	w
90	*True*	*CLASS1*	*3/13*
78	False	CLASS1	1
65	True	CLASS1	1
75	False	CLASS1	1

T_3: (*Attribute1 = C*)

Att.2	Att.3	Class	w
80	True	CLASS2	1
70	True	CLASS2	1
80	False	CLASS1	1
80	False	CLASS1	1
96	False	CLASS1	1
90	*True*	*CLASS1*	*5/13*

FIGURE 7.7 Results of test x_1 are subsets T_i (initial set T is with missing value).

```
If      Attribute1 = A
          Then
              If      Attribute2 <= 70
                      Then
                              Classification = CLASS1    (2.0 / 0);
                      Else
                              Classification = CLASS2    (3.4 / 0.4);
Elseif  Attribute1 = B
          Then
                              Classification = CLASS1    (3.2 / 0);
Elseif  Attribute1 = C
          Then
              If      Attribute3 = True
                      Then
                              Classification = CLASS2    (2.4 / 0);
                      Else
                              Classification = CLASS1    (3.0 / 0).
```

FIGURE 7.8 Decision tree for the database T with missing values

Generating Decision Rules

- In a decision tree, a path to each leaf can be transformed into an IF-THEN production rule
  - The IF part consists of all tests on a path
  - The ELSE part is a final classification
- The IF parts of the rules are mutual exclusive and exhaustive

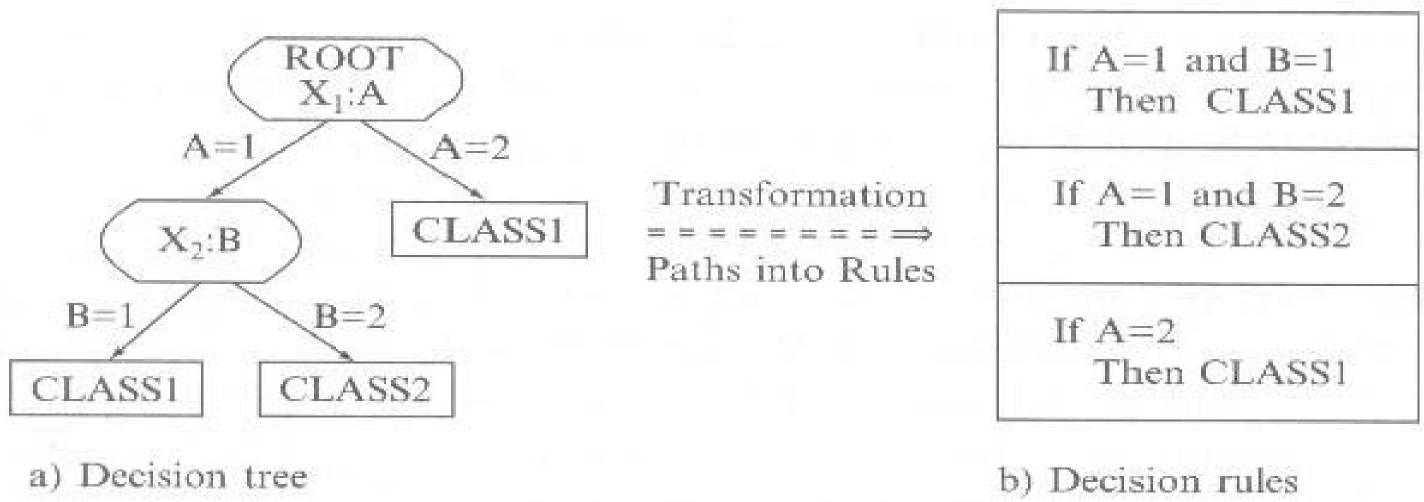


FIGURE 7.10 Transformation of a decision tree into decision rules

Reducing Complexity of Decision Trees

- One possible approach is to reduce the number of attribute values (i.e. branch number of a node)
  - A large number of values causes a large space of data

- Group the attributes values

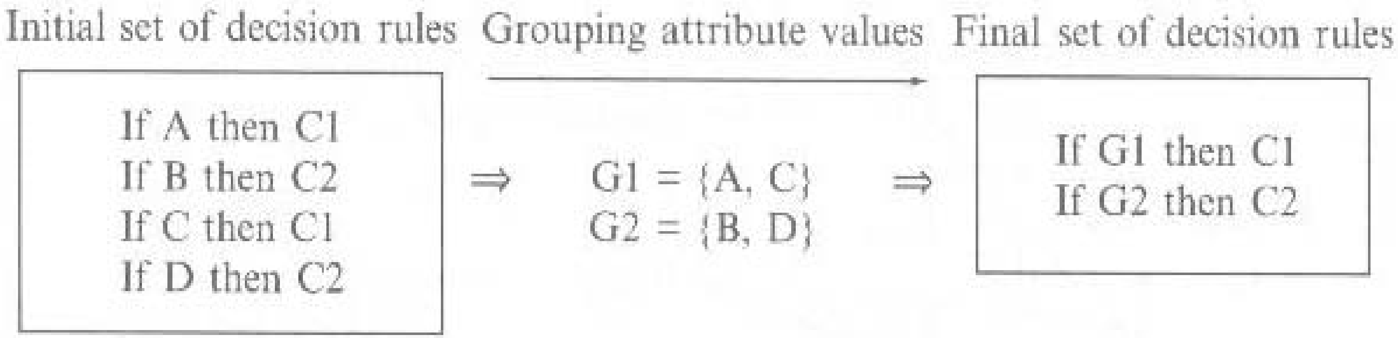


FIGURE 7.11 Grouping attribute values can reduce decision-rules set

Pro and Con for DTL

- Pro
  - Relatively simple, readable, and fast
  - Do not depend on underlying assumptions about distribution of attribute values or independence of attributes
- Con
  - Complex classifications require a large number of training sample to obtain a successful classification
  - Orthogonality of attributes is assumed during classification

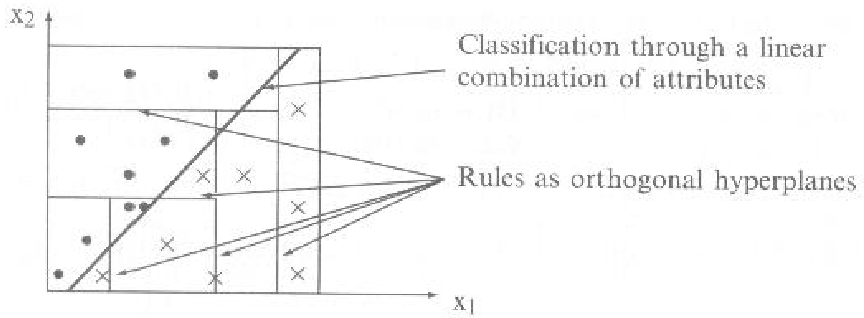


FIGURE 7.12 Approximation of nonorthogonal classification with hyperrectangles

What if a class is defined through a linear combination of attributes

Summary

- DTL provides a practical method for concept learning and for learning other discrete-valued functions
- ID3 searches a complete hypothesis space but employs an incomplete search strategy
- Overfitting the training data is an important issue in DTL
- A large variety of extensions to the basic ID3 algorithm has been developed