Machine Learning

Chapter 3 Decision Tree Learning

3.1 Introduction

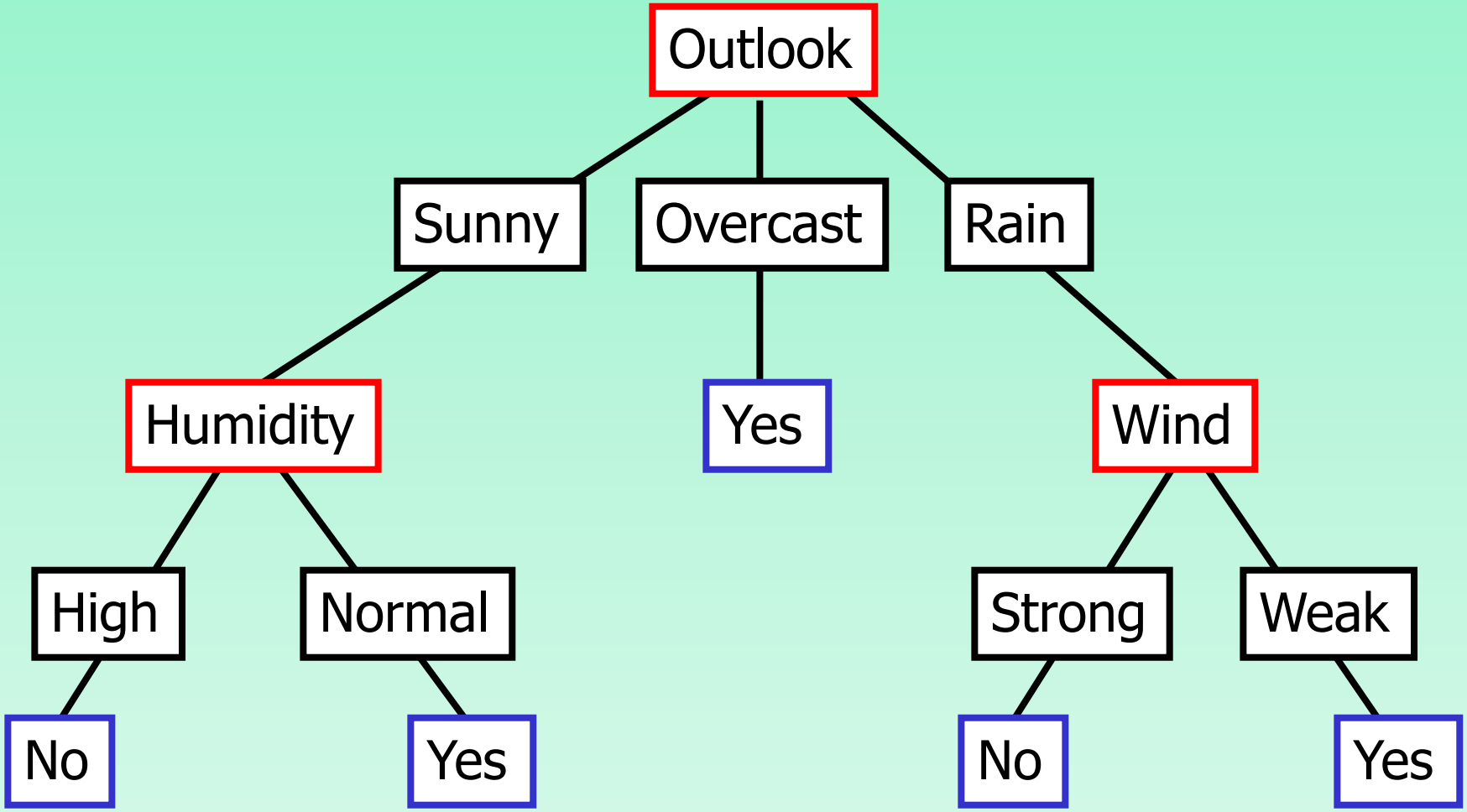
- Introduction of Decision Tree Learning
  - A method for approximating discrete-valued functions
  - Can be represented as if-then rules
  - Widely applied in many fields

3.2 Decision Tree Representation

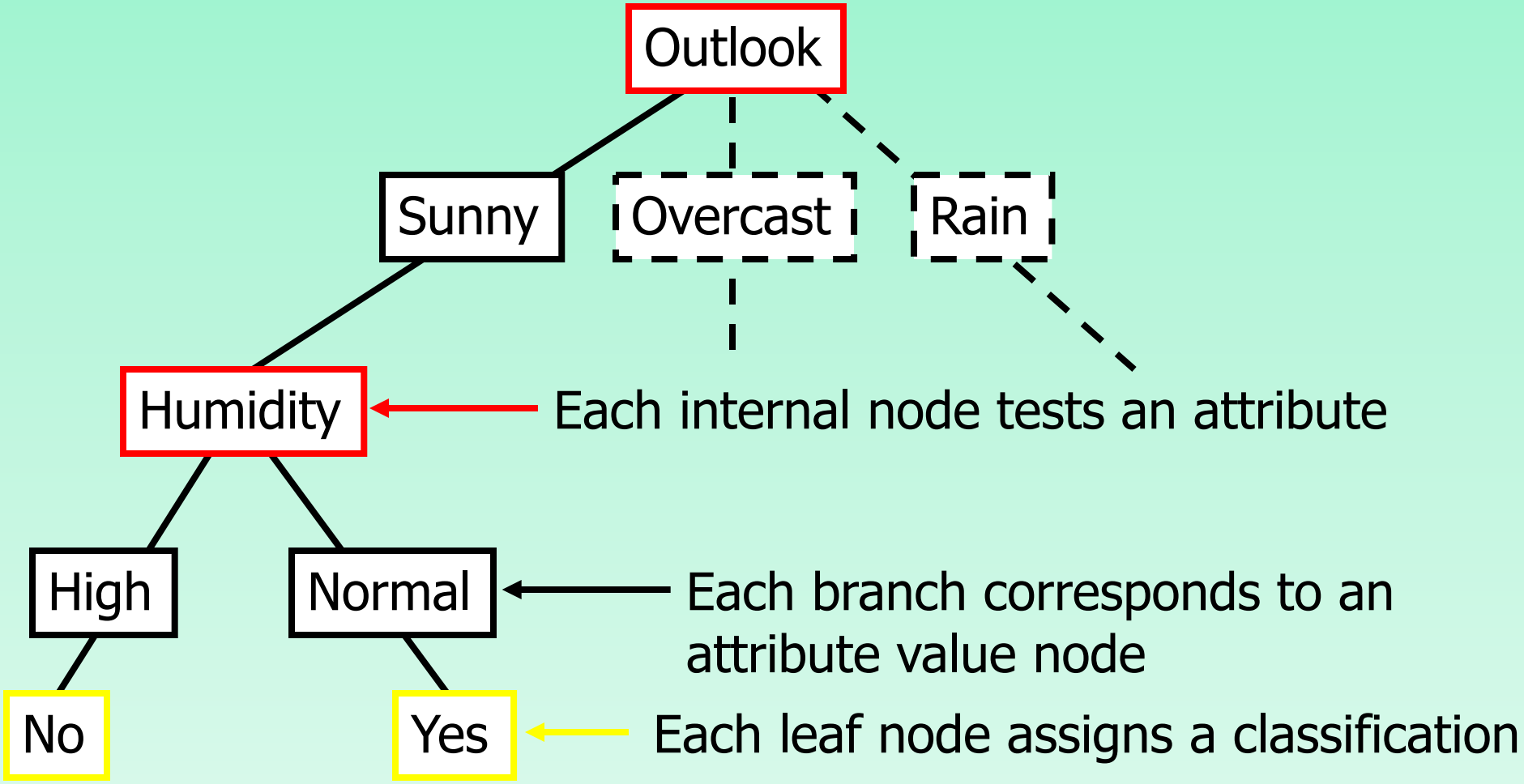
- Decision Tree
  - Classify instance by sorting them down the tree from the root to leaf node
  - Leaf node provides the classification
  - Each node specifies an attribute
  - Each branch corresponds to a value of an attribute

- Decision trees can represent a disjunction of conjunctions of constraints on the attribute values

- Fig. 3.1

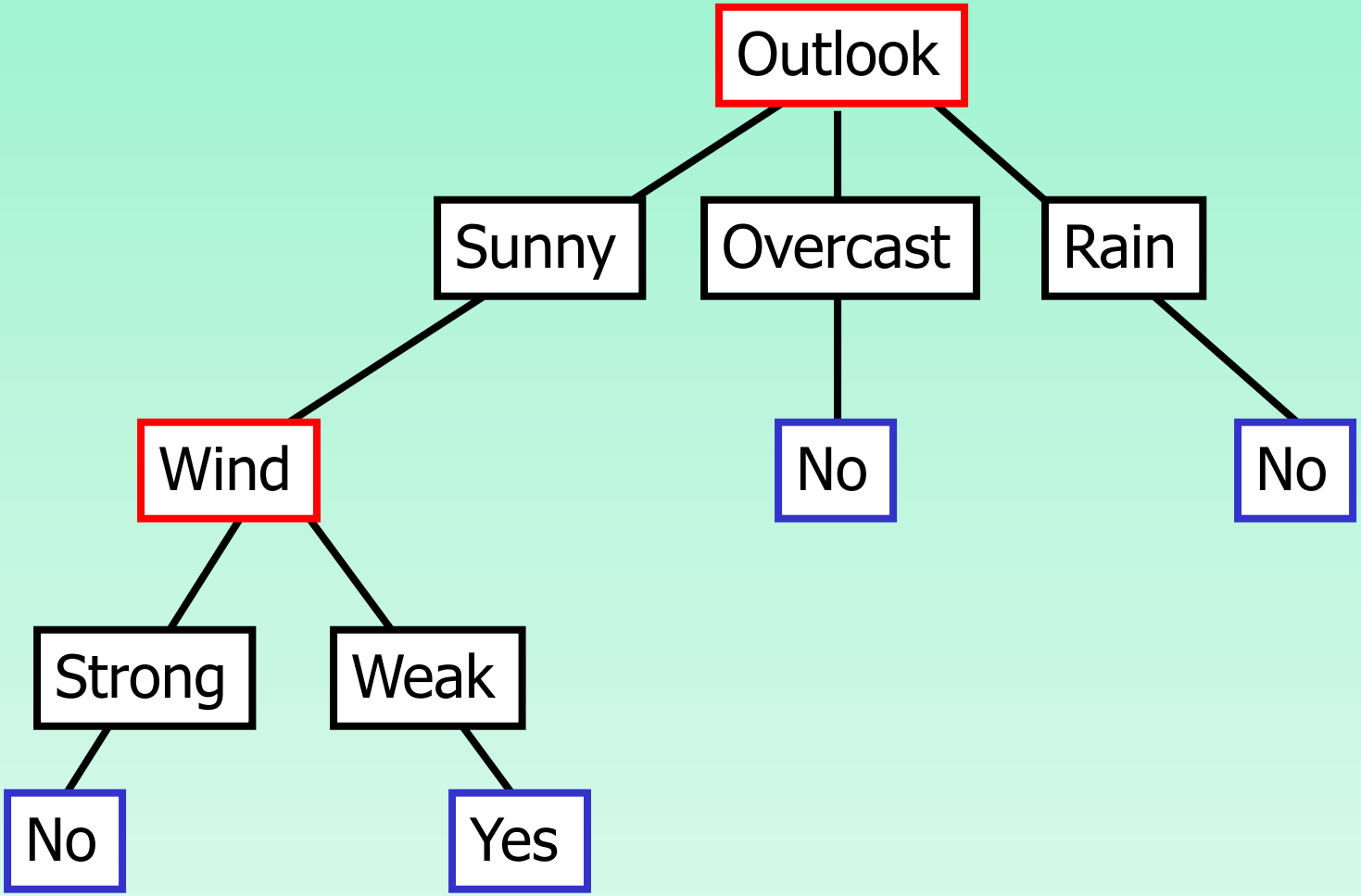

Outlook
Sunny
Overcast
Rain
Humidity
Yes
Wind
High
Normal
Strong
Weak
No
Yes
No
Yes

Decision Tree for PlayTennis



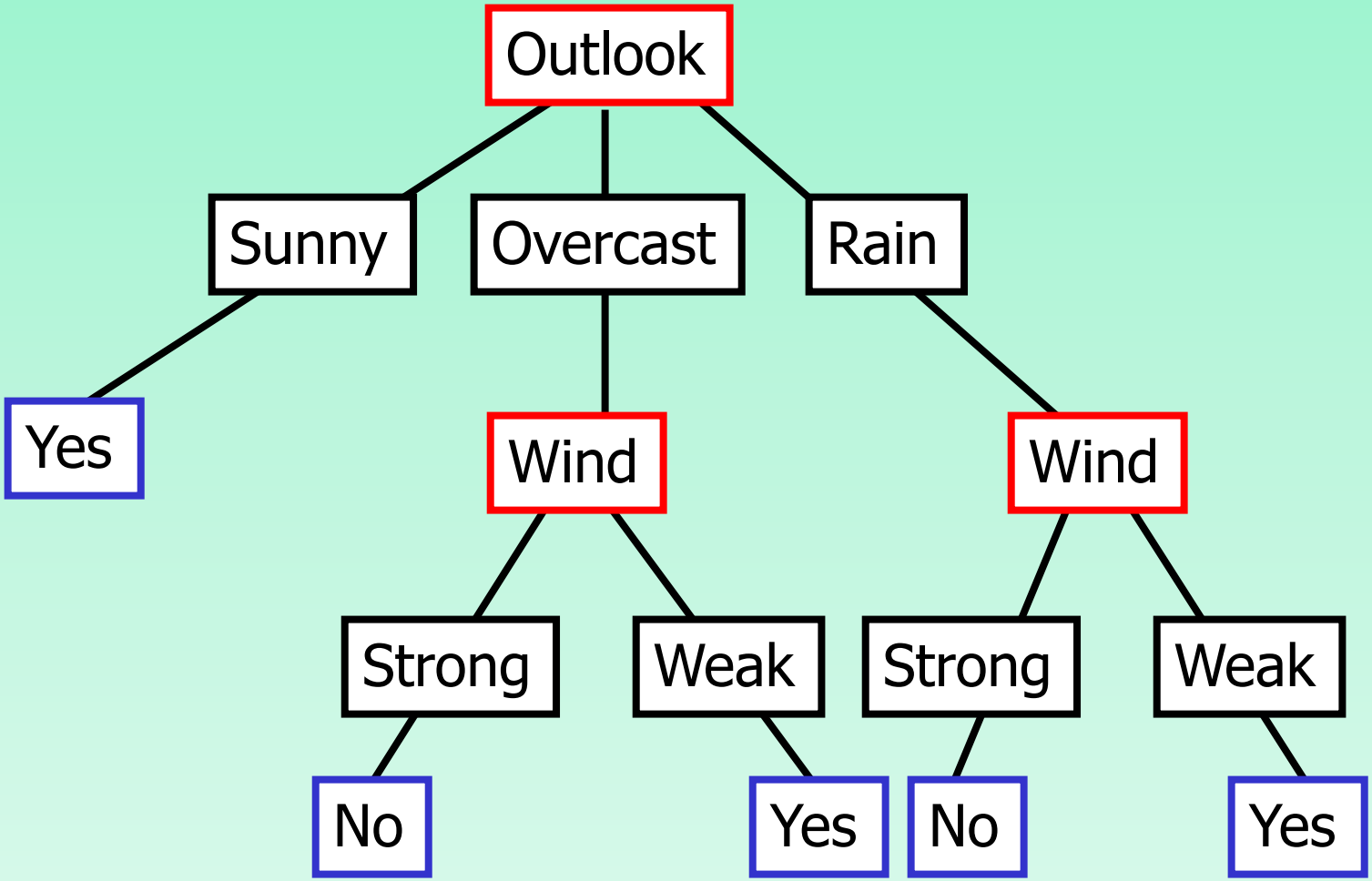
Decision Tree for Conjunction

Outlook=Sunny ∧ Wind=Weak



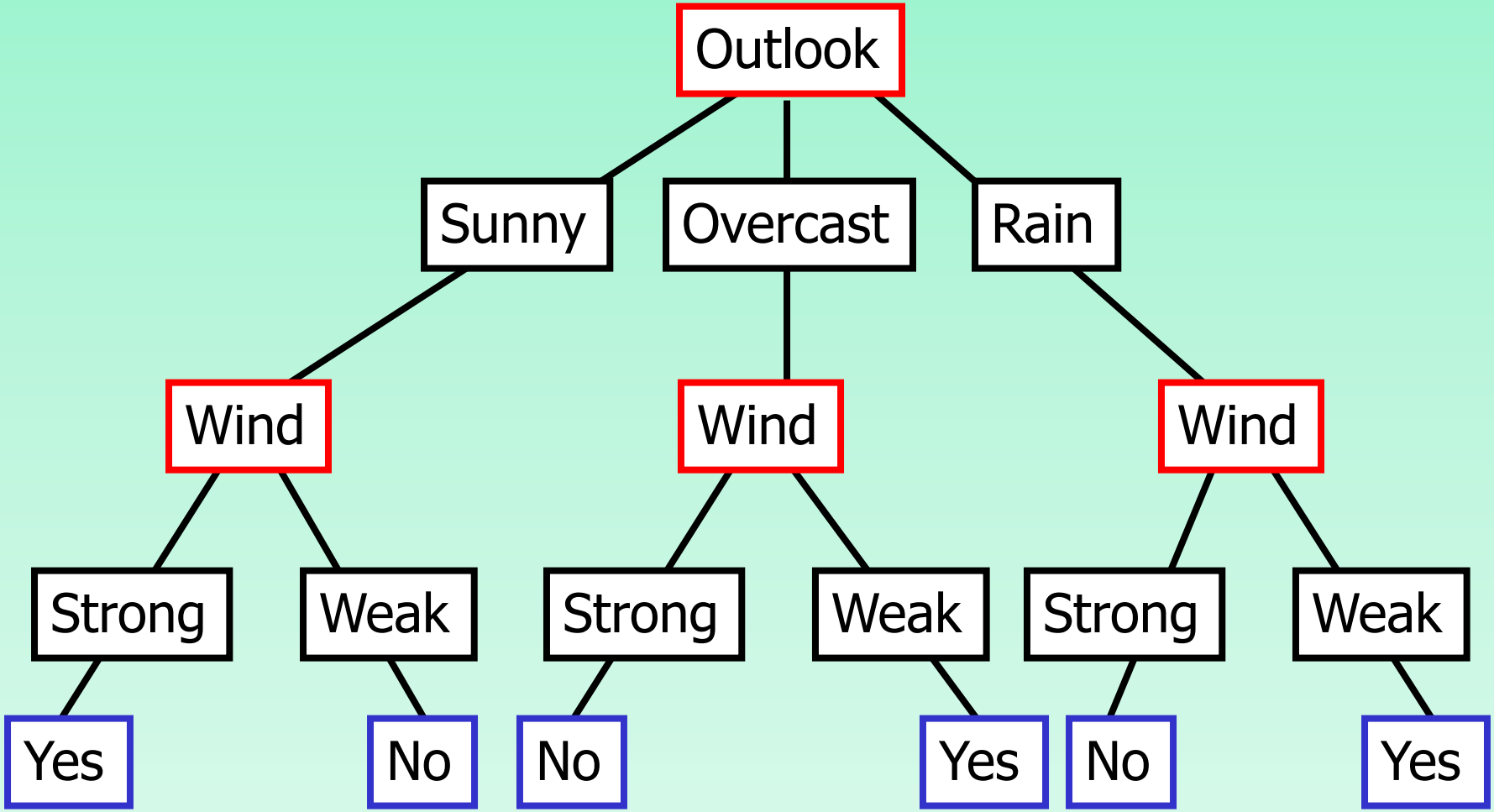
Decision Tree for Disjunction

Outlook=Sunny $\vee$ Wind=Weak



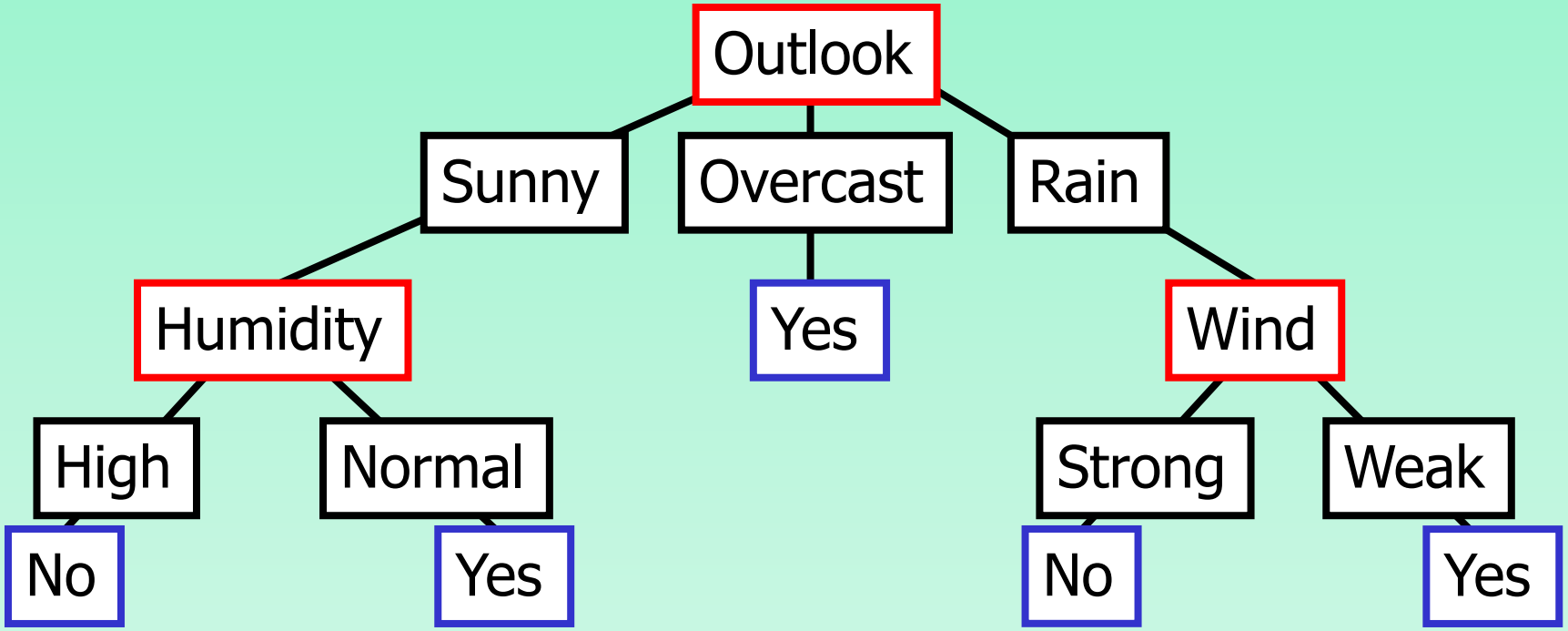
Decision Tree for XOR

Outlook=Sunny XOR Wind=Weak



Decision Tree

- decision trees represent disjunctions of conjunctions



(Outlook=Sunny ∧ Humidity=Normal)
∨ (Outlook=Overcast)
∨ (Outlook=Rain ∧ Wind=Weak)

3.3 Appropriate Problems for Decision Tree Learning

- problems with the following characteristics
  - Instance are represented by attribute-value pairs
  - The target function has discrete output values
  - Disjunctive descriptions may be required
  - The training data may contain errors
  - The training data may contain missing attribute values
- Examples:
  - Medical diagnosis
  - Credit risk analysis
  - Object classification for robot manipulator
- Classification problems
  - Classify examples into one of discrete set of possible categories

3.4 The Basic Decision Tree Learning Algorithm

- Most algorithms are variations on a core algorithm
- Employs a top-down, greedy search through the space of possible trees
- ID3(Quinlan 1986) C4.5(Quinlan 1993)

- ID3
  - Learn decision tree by constructing them top-down
  - Each attribute is evaluated using a statistical test to determine how well it alone classifies the training examples
  - The best attribute is selected as the root node of the tree
  - A branch is created for each possible value of this attribute
  - The training examples are sorted to the appropriate descendant node
  - Repeat above process

ID3 Algorithm

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 all attributes is empty, Return the single-node tree Root, with label=most common value of Target attribute in Examples
- Otherwise begin
  - A ← the Attribute from attributes that best classifies Examples
  - The decision attribute for Root ←A
  - For each possible value, vi, of A,
    - Add a new tree branch below Root, corresponding to the test A=vi
    - Let $\boldsymbol{Examples}_{vi}$ be the subset of Examples that have value vi for A
    - If $\boldsymbol{Examples}_{vi}$ 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（ $\boldsymbol{Examples}_{vi}$, Target_attribute, Attributes-{A}）
- End
- Return root

3.4.1 Which Attributes is the Best Classifier

- Information gain
  - Measures how well a given attribute separates the training examples
  - ID3 use it to select attributes at each step while growing the tree
- Entropy Measures homogeneity（同质，一致性） of examples
  - Characterizes the purity of an arbitrary collection of examples
  - The entropy of S relative to a boolean classification is

    $\text{Entropy}(S)=-(p+)\log_2(p+) - (p-)\log_2(p-)$
  - In general, if the target attribute can take on c different values

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

- One interpretation of entropy from information theory:
  - it specifies the minimum number of bits needed to encode the classification of an arbitrary member of S
  - Entropy(S)= expected number of bits needed to encode class of randomly drawn members of S (under the optimal, shortest length-code)
  - Why?
    - Information theory optimal length code assign

      $-\log_2 p$ bits to messages having probability p.

      So the expected number of bits to encode

      (+ or -) of random member of S:

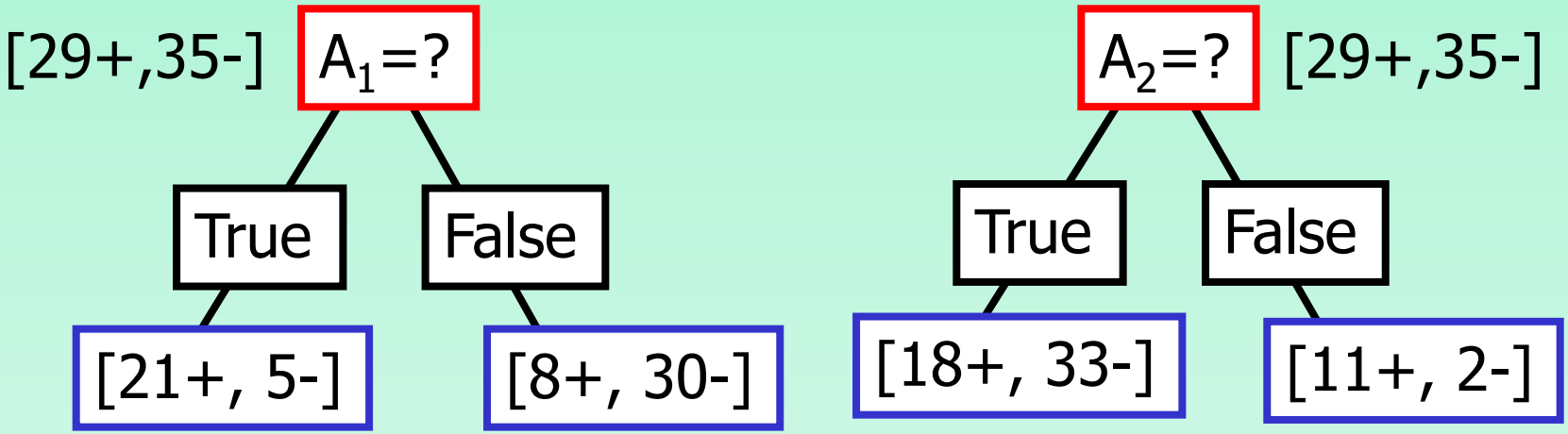
      $-p_+ \log_2 p_+ - p_- \log_2 p_-$

- Information Gain is the expected reduction in entropy caused by partitioning the examples according to this attribute

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

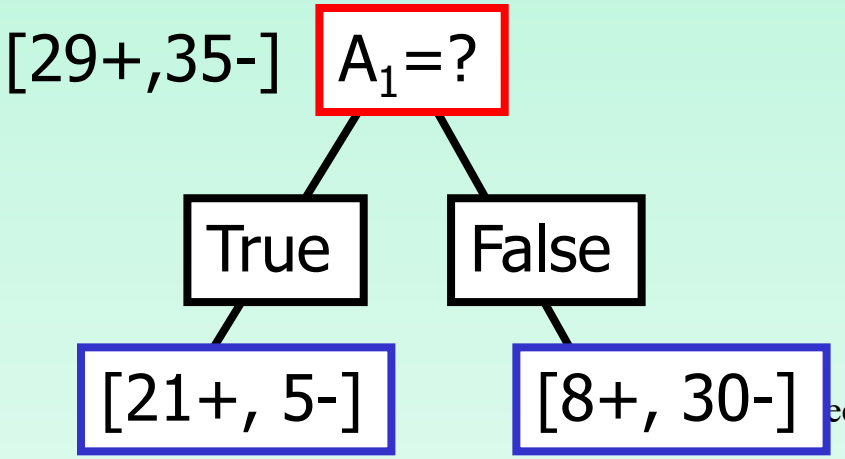
  - Gain(S,A) is the expected reduction in entropy caused by knowing the value of attribute A

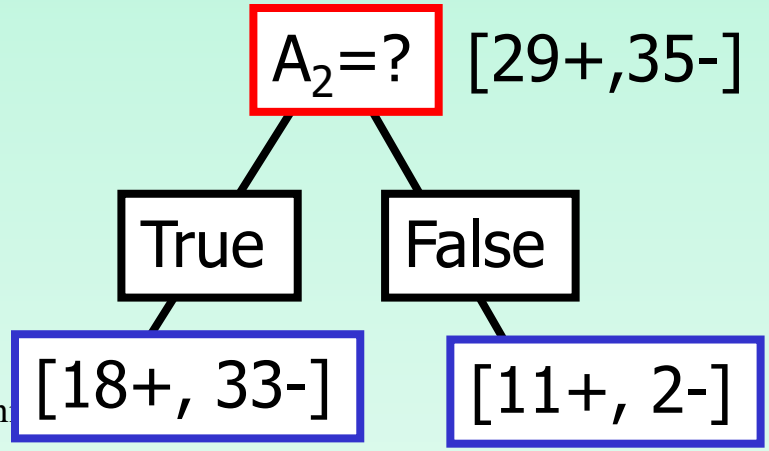
Entropy([29+,35-]) = -29/64 $\log_2$ 29/64 – 35/64 $\log_2$ 35/64
= 0.99



- Entropy([21+,5-]) = 0.71
- Entropy([8+,30-]) = 0.74
- Gain(S,A1)=Entropy(S)
- -26/64*Entropy([21+,5-])
- -38/64*Entropy([8+,30-])
- =0.27

Entropy([18+,33-]) = 0.94
Entropy([11+,2-]) = 0.62
Gain(S,A_2)=Entropy(S)
-51/64*Entropy([18+,33-])
-13/64*Entropy([11+,2-])
=0.12

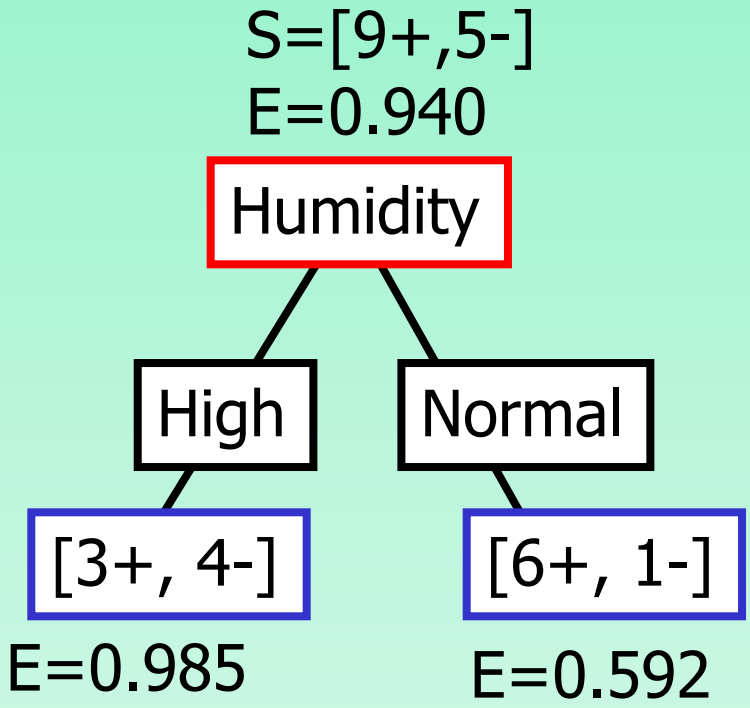


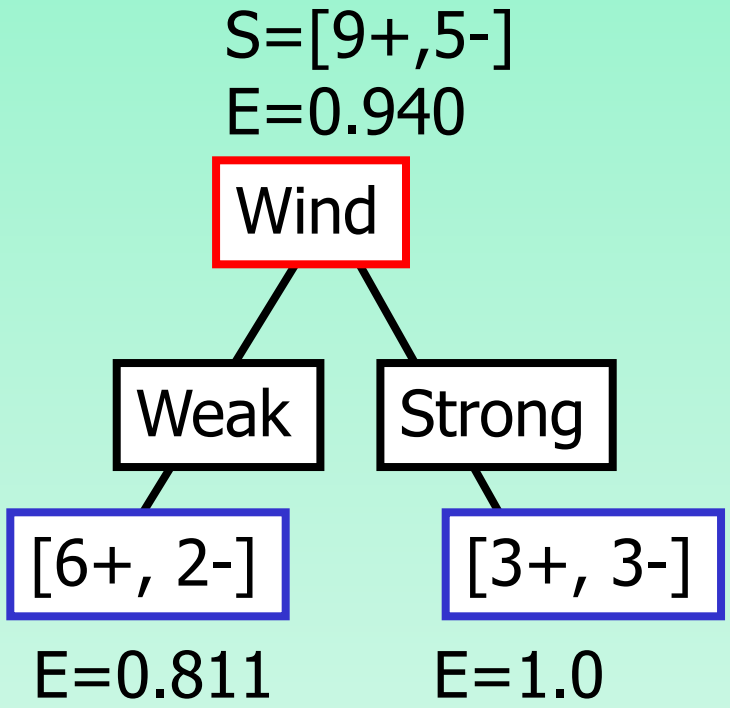


Training Examples

Day	Outlook	Temp.	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	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	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

Selecting the Next Attribute

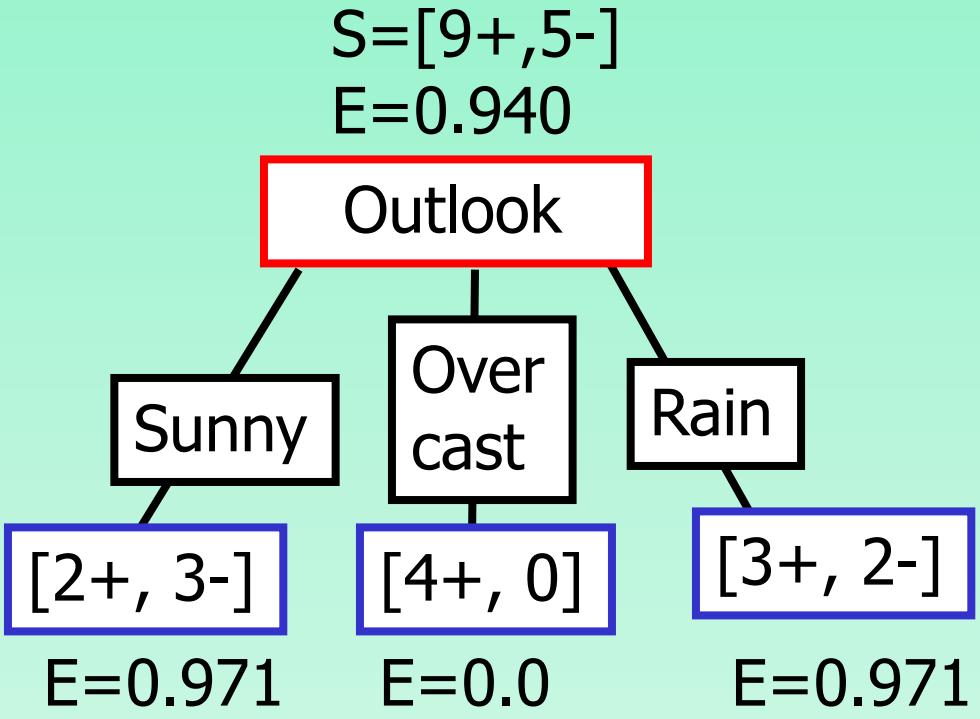




Gain(S,Humidity)
=0.940-(7/14)*0.985
– (7/14)*0.592
=0.151

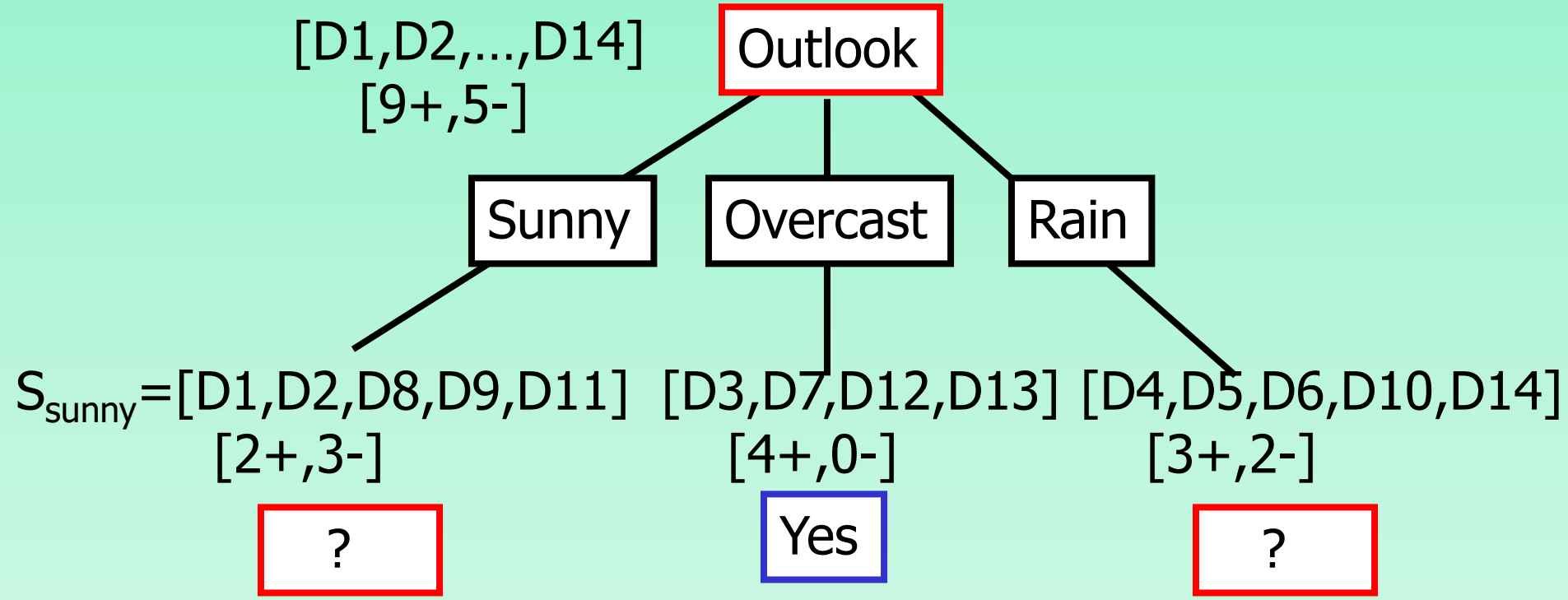
Gain(S,Wind)
=0.940-(8/14)*0.811
– (6/14)*1.0
=0.048

Selecting the Next Attribute



Gain(S,Outlook)
=0.940-(5/14)*0.971
-(4/14)*0.0 – (5/14)*0.0971
=0.247

ID3 Algorithm

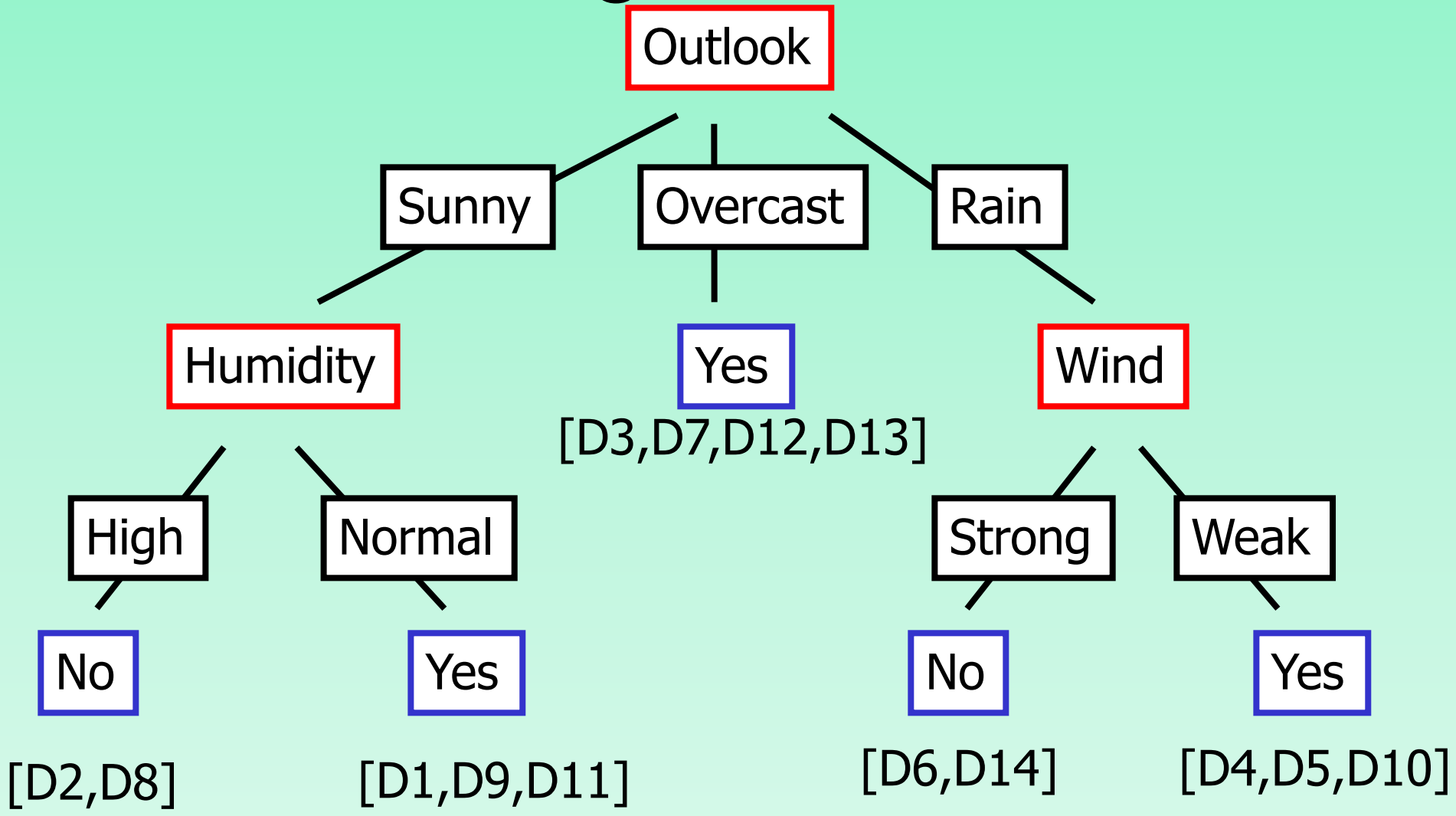


Gain(S_{sunny} , Humidity)=0.970-(3/5)0.0 – 2/5(0.0) = 0.970

Gain(S_{sunny} , Temp.)=0.970-(2/5)0.0 –2/5(1.0)-(1/5)0.0 = 0.570

Gain(S_{sunny} , Wind)=0.970 -(2/5)1.0 – 3/5(0.918) = 0.019

ID3 Algorithm



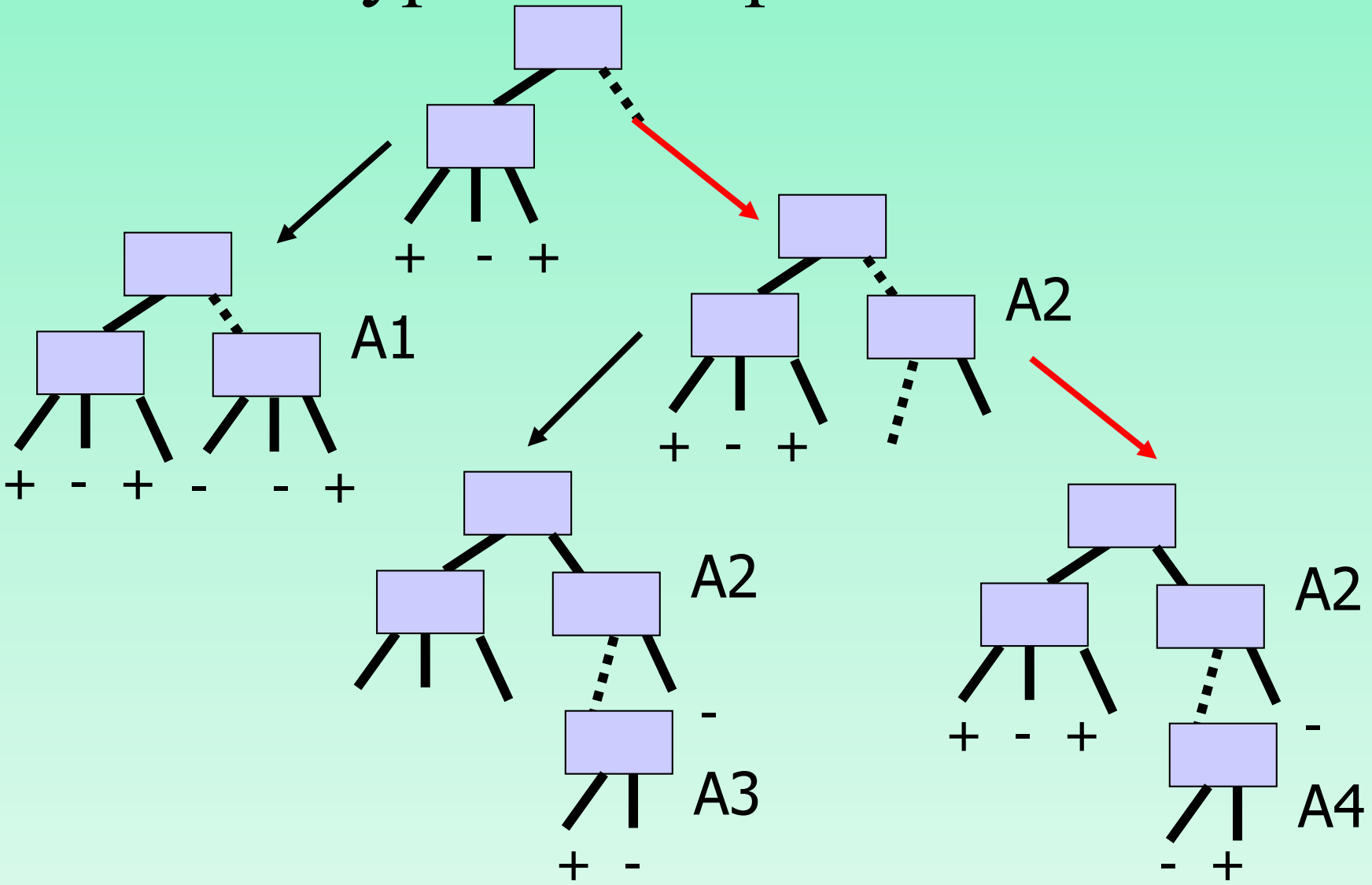
ID3 Algorithm

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 all attributes is empty, Return the single-node tree Root, with label=most common value of Target attribute in Examples
- Otherwise begin
  - A ← the Attribute from attributes that best classifies Examples
  - The decision attribute for Root ←A
  - For each possible value, vi, of A,
    - Add a new tree branch below Root, corresponding to the test A=vi
    - Let $\boldsymbol{Examples}_{vi}$ be the subset of Examples that have value vi for A
    - If $\boldsymbol{Examples}_{vi}$ 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（ $\boldsymbol{Examples}_{vi}$, Target_attribute, Attributes-{A}）
- End
- Return root

3.5 Hypothesis Space Search ID3



- ID3's capability and limitations
  - Its H is a complete space of finite discrete-valued functions
  - IDS3 maintains only a single current h
  - No backtracking in its search.
    - Converge to locally optimal solution
  - Use all training examples at each step to make statistically based decisions.
    - much less sensitive to errors in individual training examples.

3.6 Inductive Bias in Decision Tree Learning

- ID3 search strategy
  - Selects in favor of shorter trees over longer ones
  - Selects trees that place the attributes with highest information gain closest to the root

- Approximate inductive bias of ID3
  - Shorter trees are preferred over larger trees
  - BFS-ID3 finds a shortest decision tree
- A closer approximation to the inductive bias of ID3
  - Shorter trees are preferred over larger trees.
  - Trees that place high information gain attributes close to the root are preferred over those that do not.

3.6.1 Restriction Biases and Preference Biases

- ID3 vs. Candidate-Elimination Algorithm
  - ID3 searches a complete hypothesis space, but incompletely through this space.
  - C-E searches an incomplete H, but it searches this space completely
  - ID3's inductive bias is a consequence of the ordering of hypotheses by its search strategy
  - C-E's inductive bias is the consequence of the expressive power of its hypothesis representation

- Preference Bias
  - The inductive bias of ID3 is a preference for certain hypotheses over others, with no hard restriction on the hypotheses
- Restriction Bias
  - The bias of C-E is in the form of a categorical restriction on the set of hypotheses considered
- Combination of preference bias and restriction bias
  - Checker sample in chapter 1
- Typically, a preference bias is more desirable than a restriction bias

3.6.2 Why Prefer Short Hypotheses

- Philosophy Base for the ID3's inductive bias
  - Occam's razor:
    - Prefer the simplest hypothesis that fits the data
- Argument in favor
  - Prefer simple explanations for the motions of the planets over more complex explanations
  - Fewer short hypotheses than long hypotheses
    - A short hypothesis that fits the data is unlikely to be a coincidence
    - A long hypothesis that fits the data might be a coincidence
  - Complex h often fail to generalize correctly to subsequent data

3.6.2 Why Prefer Short Hypotheses

- A major difficulty with the Occam's razor
  - By the same reasoning: some special small trees are more better(17 leaf nodes, 11 nonleaf).
    - Many small sets,but most of them arcane
    - Why small sets of h consisting of decision trees with short descriptions are more relevant than others
  - Size of a h is determined by the representation.
    - Two learners using different internal representations get different h, both justifying their contradictory conclusions by Occcam's razor
- Considering Occam's razor from a process of evolution and natural selection
  - Inductive bias decides internal representations

3.7 Issue In Decision Tree Learning

- Practical issues in DT learning
  - How deeply to grow the decision tree
  - Handle continuous attributes
  - Choose an appropriate attribute selection measure
  - Handling training data with missing attribute values
  - Handling attributes with differing costs
  - Improving computational efficiency
- To address those problems ID3 is extended to C4.5

3.7.1 Avoiding Overfitting the data

- Consider error of hypothesis h over
  - Training data: $error_{train}(h)$
  - Entire distribution D of data: $error_D(h)$

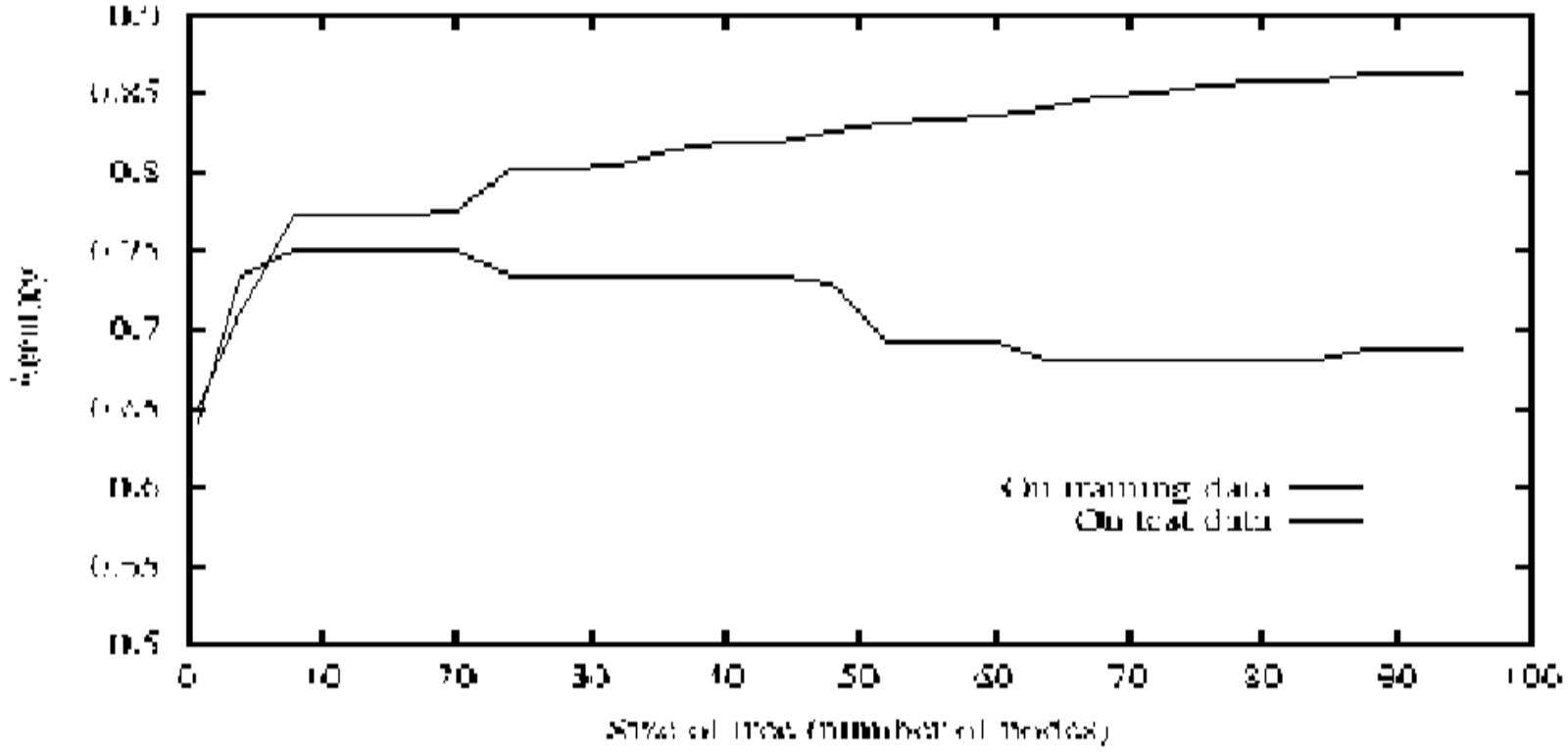
Hypothesis $h \in H$ *overfits* training data if there is an alternative hypothesis $h' \in H$ such that

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

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

- Sample Figure 3.6

Overfitting in Decision Tree Learning



- What lead to overfitting
  - Random noise in the training example
  - Small number of examples are associated with leaf nodes
    - occur coincidental（巧合的） regularities（规则性）, (some attributes happens to partition the examples very well, despite being unrelated to the actual target function)

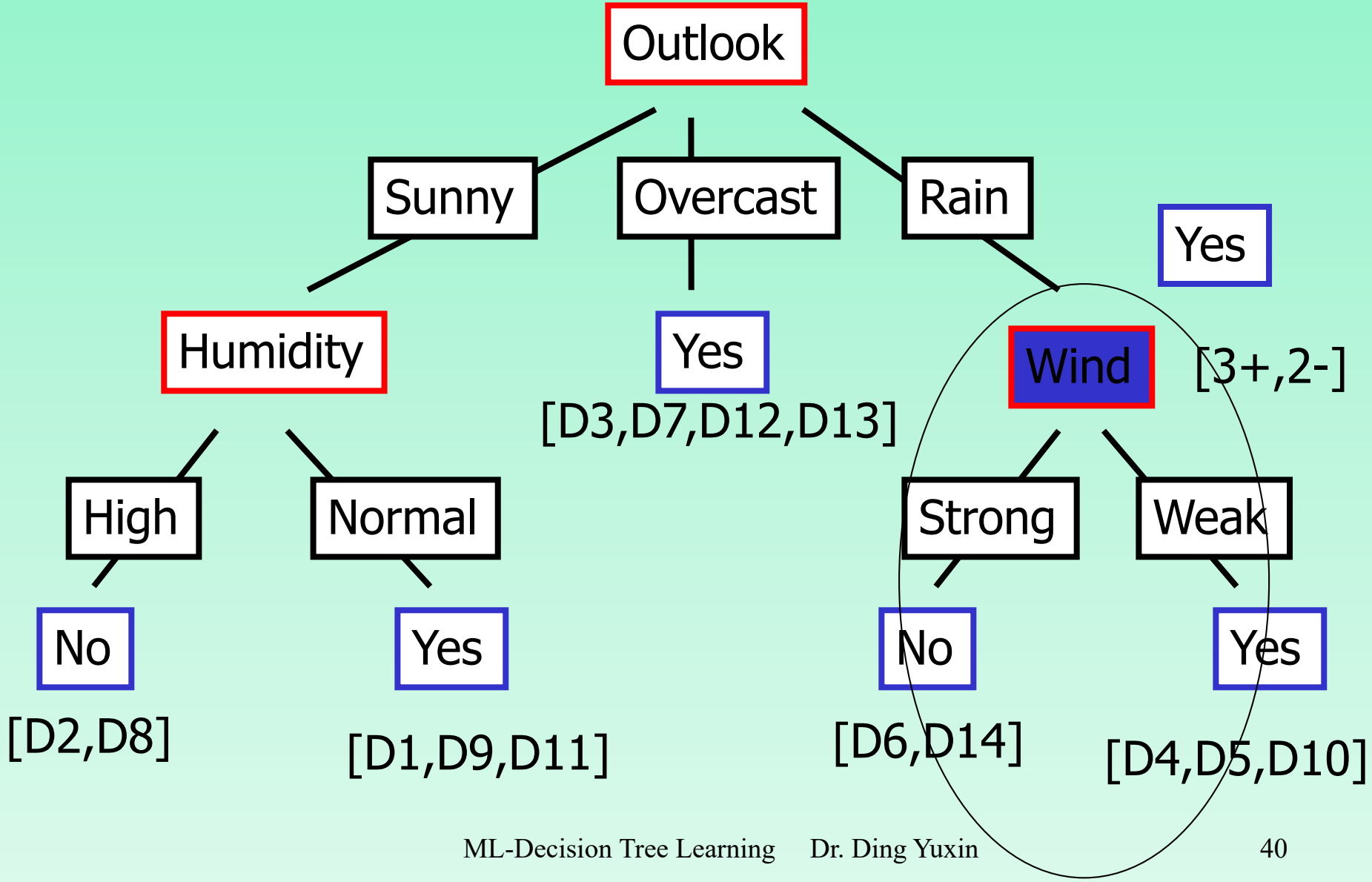
- How can we avoid overfitting?
  - Stop growing the tree before perfectly classifying the training data
  - Grow full tree then post-prune the tree
- remark
  - The first might seem more direct, estimating when to stop growing the tree is difficult
  - The second is more successful in practice

- A key question to avoid overfitting
  - Define a criterion to evaluate the correct final tree size
- Solutions:
  - Use a separate set of examples to evaluate the utility of post-pruning nodes from the tree
  - Use all available data for training, but apply a statistical test to estimate whether expanding a particular node is likely to produce an improvement
  - Use an explicit measure of the complexity for encoding the training examples and the decision tree, halting growth when this encoding size is minimized

- Remark
  - The first is the most common (training and validation set approach)
    - A training set，learning hypothesis
    - A validation set，evaluate the accuracy of hypothesis
  - motivation：
    - the learner may be misled by random errors within the training set, but the validation set is unlikely to exhibit the same random fluctuations
  - Validation set should be large enough to provide a statistically significant sample of the instances
    - Usually one-third of the available examples for the validation set

3.7.1.1 Reduced Error Pruning

- Pruning Steps
  - Pruning
    - removing the subtree rooted at that node, making it a leaf node
  - Assigning node's classification
    - the most common classification of the training examples
  - Removing a node
    - if the resulting pruned tree performs no worse than the original over the validation set
  - Repeat until further pruning is harmful
- The available data is split into 3 subsets
  - Training examples
  - Validation examples used for pruning the tree
  - Test examples used for estimating the real accuracy
  - This approach need a large amount of data


Outlook
Sunny
Overcast
Rain
Yes
Humidity
Yes
[D3,D7,D12,D13]
Wind
[3+,2-]
High
Normal
Strong
Weak
No
Yes
No
Yes
[D2,D8]
[D1,D9,D11]
[D6,D14]
[D4,D5,D10]

3.7.1.2 Rule Post-Pruning

- Steps:
  - (1) Infer a full tree from training
    - Fitting the training data as well as possible and allowing overfitting to occur.
  - (2)Convert the tree into an equivalent set of rules
    - creating one rule for each path
  - (3)Prune each rule
    - removing any preconditions that result in improving its estimated accuracy
  - (4)Sort the pruned rules
    - Sorting by their estimated accuracy
    - classifying subsequent instances in this sequence

- Sample: decision tree in Fig. 3.1
  - The leftmost path of the tree
    - if (outlook=sunny)∧(Humidity=High) then PlayTennis=No
  - Consider removing (outlook=sunny) and (Humidity=High)
  - Select the pruning steps produced the greatest improvement on the rule accuracy

- Estimate rule accuracy
  - Use a validation set
  - Based on the training set itself
    - using a pessimistic estimate
      - make up for the fact that the training data gives an estimated biased in favor of the rules
    - Procedure($Error_S$, $Error_D$,binomial distribution)
      - Calculate the rule accuracy
      - Calculating the standard deviation assuming a binomial distribution
      - For a given confidence level, the lower-bound estimate is taken as the measure of rule performance
    - Remark
      - For large data sets, the pessimistic estimate is very close to the observed accuracy
      - Is not statistically valid, but useful in practice

- Why convert DT to rules before pruning
  - Distinguishing among the different contexts in which a decision node is used
    - Making different pruning decision for each node
  - Removes the distinction between attribute tests that occur near the root of the tree and those that occur near leaves
    - Avoid messy bookkeeping
  - Improve readability

3.7.2 Incorporating Continuous-Valued Attributes

- Restriction of ID3 on attribute values
  - Target attribute is discrete value
  - Attributes tested in the decision nodes are discrete values

- The second restriction can easily be removed
  - dynamically defining new discrete-valued attributes

- Example:
  - Temperature = 24.50C
  - (Temperature > 20.00C) = {true, false}
  - Where to set the threshold?

Temperature	40^0C	48^0C	60^0C	72^0C	80^0C	90^0C
PlayTennis	No	No	Yes	Yes	Yes	No

- Defining the threshold-based boolean attribute
  - Pick a threshold ,c, that produce the greatest information gain
    - Sorting the examples according to A
    - Identify adjacent examples that differ in their target classification
    - Cerate a set of candidate thresholds midway between the corresponding values of A
      - It can be shown that c that maximizes information gain lies at such boundary(Fayyad1991)
    - computing the information gain of each threshold

- Extension to this approach
  - Splits the continuous attribute into multiple intervals

3.7.3 Alternative Measures for Selecting Attributes

- Bias in Information gain
  - favors attributes with many values
- The gain ratio measure
  - penalizing attributes by incorporating a term, called split information
  - split information is sensitive to how broadly and uniformly an attribute splits the data

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

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

- Splitinformation term discourage the selection of attributes with many uniformly distributed values

- Issue:
  - the denominator can be zero or very small when $S_i \approx S$.
    - applying the GainRatio test only considering those attributes with above average Gain

- A distance-based measure
  - Defining a distance metric between partitions of the data
  - Attribute is evaluated based on the distance between the data partition it creates and the perfect partition
  - Choose the attribute whose partition is closest to the perfect partition
  - Lopez de Mantaras (1991) proved that it is not biased toward attributes with large numbers of values

- Mingers (1989a) experiments
  - the choice of attribute selection measure appears to have a smaller impact on final accuracy than does the extent and method of post-pruning

3.7.4 Handling Training Examples with Missing Attribute Values

- Handle missing attribute example
  - How to estimate the missing attribute value based on other examples for which this attribute has a known value
  - How to calculate Gain(S,A) if <x, c(x)> is a training sample in S and that the value A(x) is unknown.

- Dealing with the missing attribute value
  - Assign it the value that is most common among training examples at node n
  - Assign it the most common value among examples at node n that have the classification c(x)
  - Assign a probability to each of the possible values of A

3.7.5 Handling Attributes with Differing Costs

- Instance attributes may have associated costs
  - Prefer decision trees that use low-cost attributes where possible
    - relying on high-cost attributes only when needed to produce reliable classifications
- Consider attribute costs by introducing a cost term into the attribute selection measure

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

  - Tan and Schlimmer(1990) and Tan(1993), Nunez(1988)(1991)

Summary and Further Reading

- Decision Tree learning provides a practical method for concept learning and for learning other discrete-valued functions
- ID3
  - Greedy search
  - Growing from root downward
  - Search a complete H
  - Inductive bias, a preference for smaller trees
- Overfitting problem

- C4.5 is a software extension of the basic ID3 algorithm designed :
  - Avoiding overfitting the data
  - Determining how deeply to grow a decision tree.
  - Reduced error pruning.
  - Rule post-pruning.
  - Handling continuous attributes.
  - Choosing an appropriate attribute selection measure.
  - Handling training data with missing attribute values.
  - Handling attributes with differing costs.

- Homework: 3.3,3.2

- Hunt
- Quinlan
- Mingers