Machine Learning

Chapter 3

DECISION TREE LEARNING

Dr. T. ADILAKSHMI

Introduction

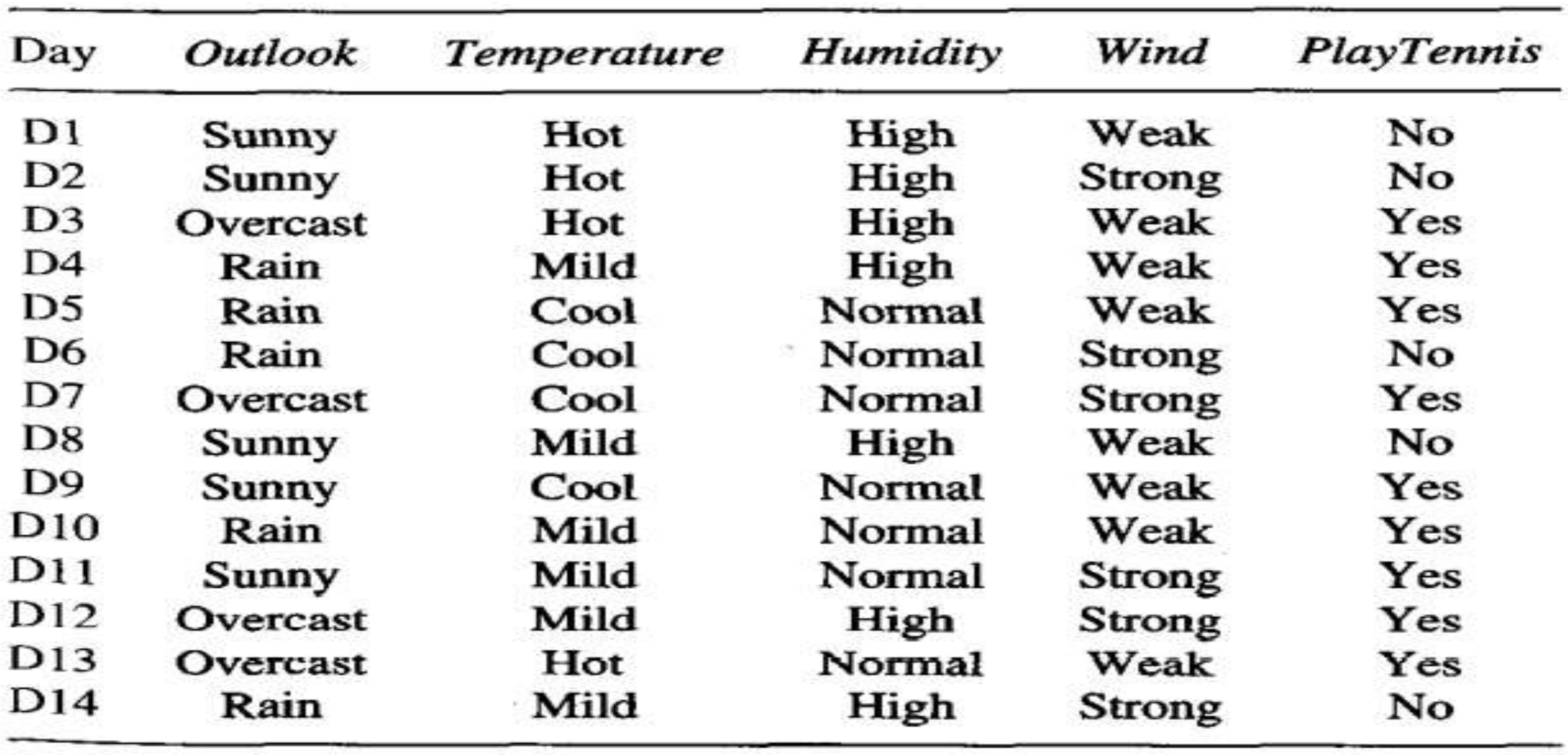
- Decision tree learning is a method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree.
- Learned trees can also be re-represented as sets of if-then rules to improve human readability

DECISION TREE REPRESENTATION

- Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance.
- Each node in the tree specifies a test of some attribute of the instance, and each branch descending from that node corresponds to one of the possible values for this attribute.
- An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute in the given example.
- This process is then repeated for the subtree rooted at the new node.

An Illustrative Example

- To illustrate the operation of ID3, consider the learning task represented by the training examples of Table 3.2. Here the target attribute PlayTennis, which can have values yes or no for different Saturday mornings, is to be predicted based on other attributes of the morning in question.



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

TABLE 3.2
Training examples for the target concept *PlayTennis*.

Figure 3.1 illustrates a typical learned decision tree.

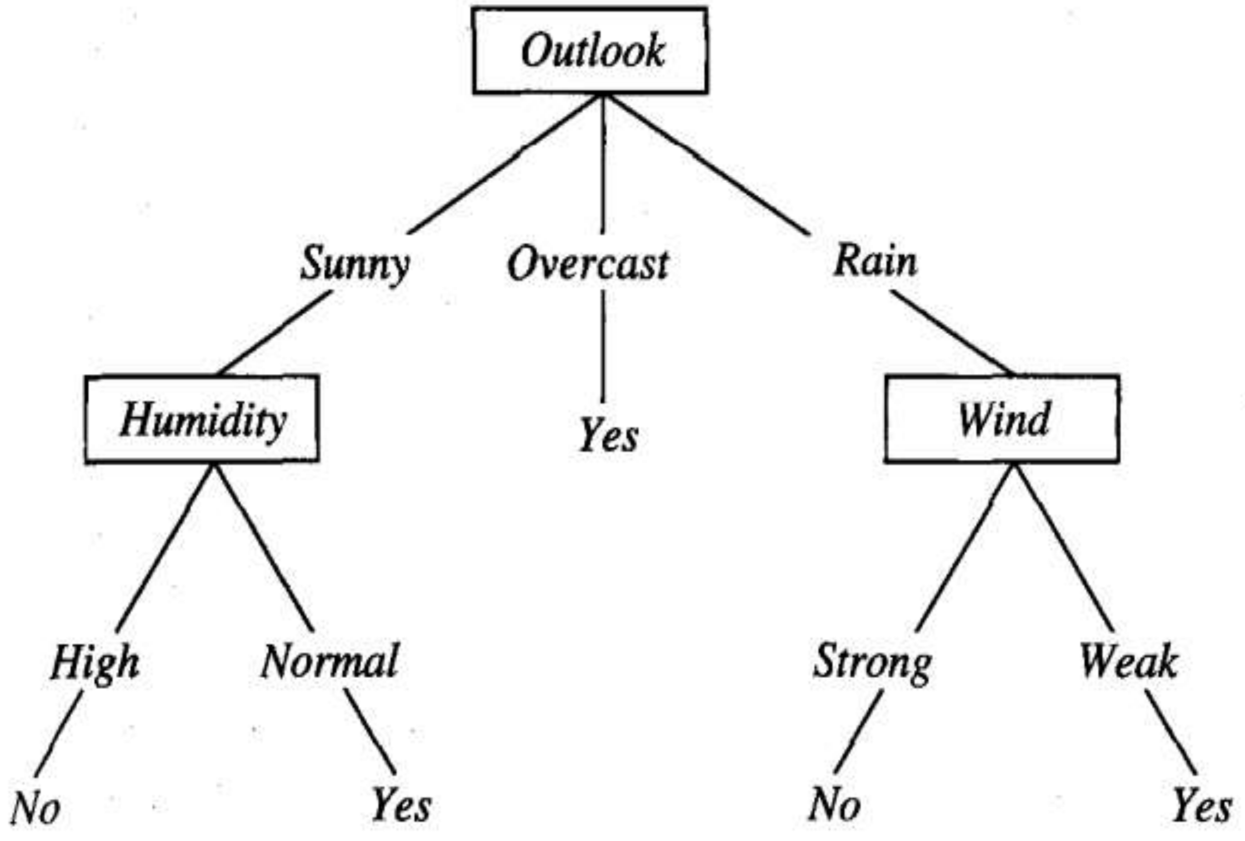


FIGURE 3.1
A decision tree for the concept *PlayTennis*. An example is classified by sorting it through the tree to the appropriate leaf node, then returning the classification associated with this leaf (in this case, *Yes* or *No*). This tree classifies Saturday mornings according to whether or not they are suitable for playing tennis.

- This decision tree classifies Saturday mornings according to whether they are suitable for playing tennis.
- For example, the instance

  (Outlook = Sunny, Temperature = Hot, Humidity = High, Wind = Strong)

  would be sorted down the leftmost branch of this decision tree and would therefore be classified as a negative instance (i.e., the tree predicts that PlayTennis = no)
- In general, 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, and the tree itself to a disjunction of these conjunctions.

- For example, the decision tree shown in Figure 3.1 corresponds to the expression

  (Outlook = Sunny Λ Humidity = Normal)

  V (Outlook = Overcast)

  v (Outlook = Rain Λ Wind = Weak)

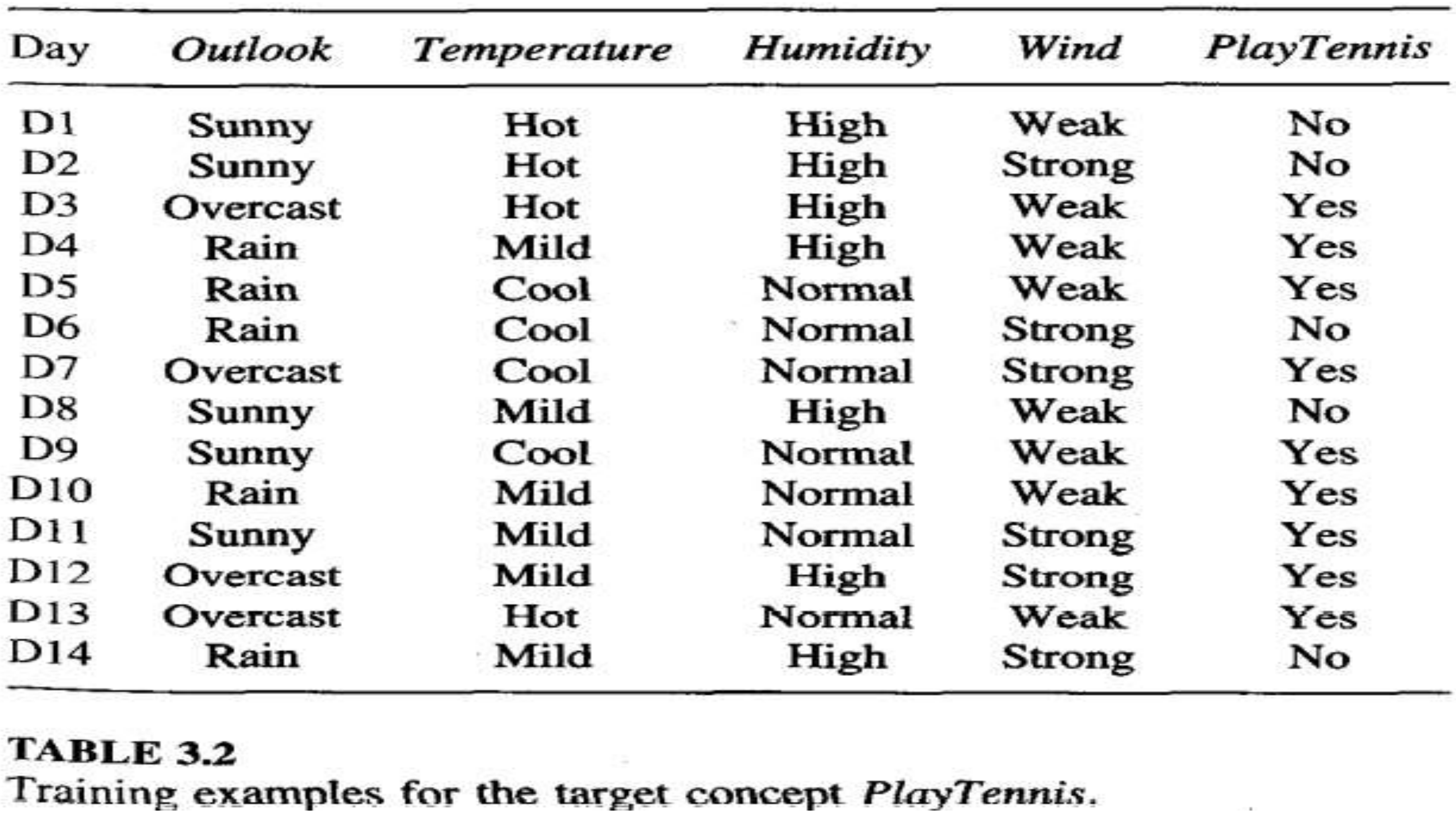
APPROPRIATE PROBLEMS FOR DECISION TREE LEARNING

Decision tree learning is generally best suited to problems with the following characteristics :

- **Instances are represented by attribute-value pairs.** Instances are described by a fixed set of attributes (e.g., Temperature) and their values (e.g., Hot). each attribute takes on a small number of disjoint possible values (e.g., **Hot, Mild, Cold)**
- **The target function has discrete output values.** The decision tree assigns a boolean classification (e.g., yes or no) to each example. Decision tree methods easily extend to learning functions with more than two possible output values.
- **Disjunctive descriptions may be required.** decision trees represent disjunctive expressions.
- **The training data may contain errors.** Decision tree learning methods are robust to errors, both errors in classifications of the training examples and errors in the attribute values that describe these examples
- **The training data may contain missing attribute values.** Decision tree methods can be used even when some training examples have unknown values (e.g., if the Humidity of the day is known for only some of the training examples).

An Illustrative Example

- To illustrate the operation of ID3, consider the learning task represented by the training examples of Table 3.2. Here the target attribute PlayTennis, which can have values yes or no for different Saturday mornings, is to be predicted based on other attributes of the morning in question.



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

TABLE 3.2
Training examples for the target concept *PlayTennis*.

Applications

- Many practical problems have been found to fit these characteristics.
- Decision tree learning has therefore been applied to problems such as learning to classify medical patients by their disease, equipment malfunctions by their cause, and loan applicants by their likelihood of defaulting on payments.
- Such problems, in which the task is to classify examples into one of a discrete set of possible categories, are often referred to as **classification problems**

THE BASIC DECISION TREE LEARNING ALGORITHM

- Our basic algorithm, ID3, learns decision trees by constructing them top-down, beginning with the question “which attribute should be tested at the root of the tree?” To answer this question, each instance attribute is evaluated using a statistical test to determine how well it alone classifies the training examples.
- The best attribute is selected and used as the test at the root node of the tree.
- A descendant of the root node is then created for each possible value of this attribute, and the training examples are sorted to the appropriate descendant node
- The entire process is then repeated using the training examples associated with each descendant node to select the best attribute to test at that point in the tree.

Which Attribute Is the Best Classifier?

- The central choice in the ID3 algorithm is selecting which attribute to test at each node in the tree.
- We would like to select the attribute that is most useful for classifying examples.
- What is a good quantitative measure of the worth of an attribute?
- We will define a statistical property, called information gain, that measures how well a given attribute separates the training examples according to their target classification

ENTROPY MEASURES HOMOGENEITY OF EXAMPLES

- Measure used in information theory, called entropy, that characterizes the (im)purity of an arbitrary collection of examples.
- Given a collection S, containing positive and negative examples of some target concept, the entropy of S relative to this boolean classification is

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

- where p+ is the proportion of positive examples in S and p- is the proportion of negative examples in S.
- In all calculations involving entropy we define 0log0 to be 0.
- To illustrate, suppose S is a collection of 14 examples of some boolean concept, including 9 positive and 5 negative examples (we adopt the notation [9+, 5-] to summarize such a sample of data).
- Then the entropy of S relative to this boolean classification is

$$\begin{aligned} Entropy([9+, 5-]) &= -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) \\ &= 0.940 \end{aligned}$$

- For example, if all members are positive (p+ = I), then p- is 0, and Entropy(S) = $-1 . \log_2(1) - 0 . \log_2 0 = -1 . 0 - 0 . \log_2 0 = 0$.
- entropy is 1 when the collection contains an equal number of positive and negative examples.
- If the collection contains unequal numbers of positive and negative examples, the entropy is between 0 and 1.

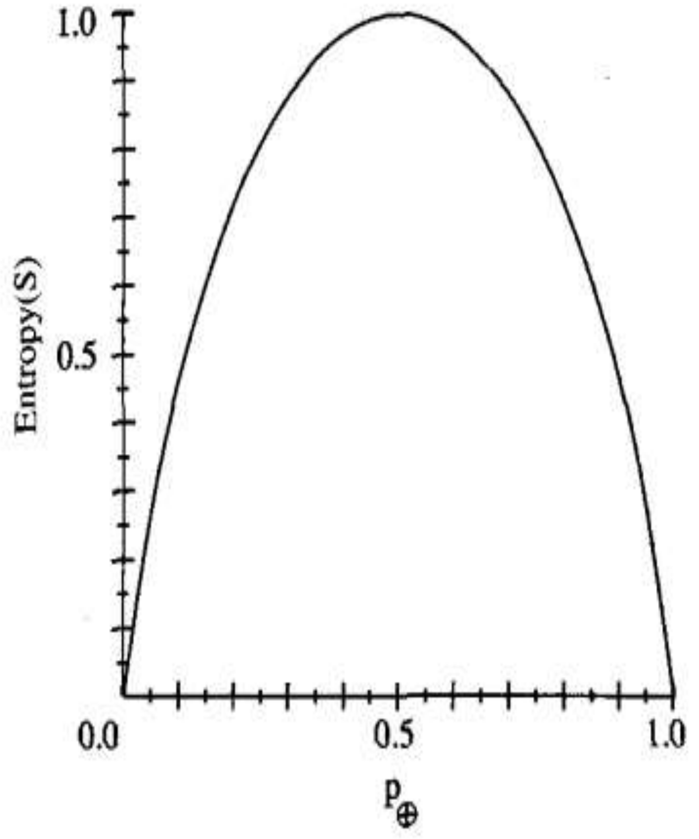


FIGURE 3.2
The entropy function relative to a boolean classification, as the proportion, $p_\oplus$, of positive examples varies between 0 and 1.

- One interpretation of entropy from information theory is that it specifies the minimum number of bits of information needed to encode the classification of an arbitrary member of **S**
- For example, if **p+** is 1, the receiver knows the drawn example will be positive, so no message need be sent, and the entropy is zero.
- On the other hand, if **p+** is 0.5, one bit is required to indicate whether the drawn example is positive or negative.
- If **p+** is 0.8, then a collection of messages can be encoded using on average less than 1 bit per message by assigning shorter codes to collections of positive examples and longer codes to less likely negative examples.

```
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
```

* The best attribute is the one with highest *information gain*, as defined in Equation (3.4).

TABLE 3.1
Summary of the ID3 algorithm specialized to learning boolean-valued functions. ID3 is a greedy algorithm that grows the tree top-down, at each node selecting the attribute that best classifies the local training examples. This process continues until the tree perfectly classifies the training examples, or until all attributes have been used.

- More generally, if the target attribute can take on c different values, then the entropy of ***S*** relative to this c-wise classification

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

- where ***pi*** is the proportion of ***S*** belonging to class ***i.***
- Note the logarithm is still base 2 because entropy is a measure of the expected encoding length measured in ***bits.***
- Note also that if the target attribute can take on c possible values, the entropy can be as large as $\log_2$ ***c.***

INFORMATION GAIN MEASURES THE EXPECTED REDUCTION IN ENTROPY

- Given entropy as a measure of the impurity in a collection of training examples, we can now define a measure of the effectiveness of an attribute in classifying the training data.
- The measure we will use, called **information gain, is simply the expected reduction in entropy caused by partitioning the examples according to this attribute.**
- More precisely, the information gain, Gain(S, A) of an attribute A, relative to a collection of examples S, is defined as

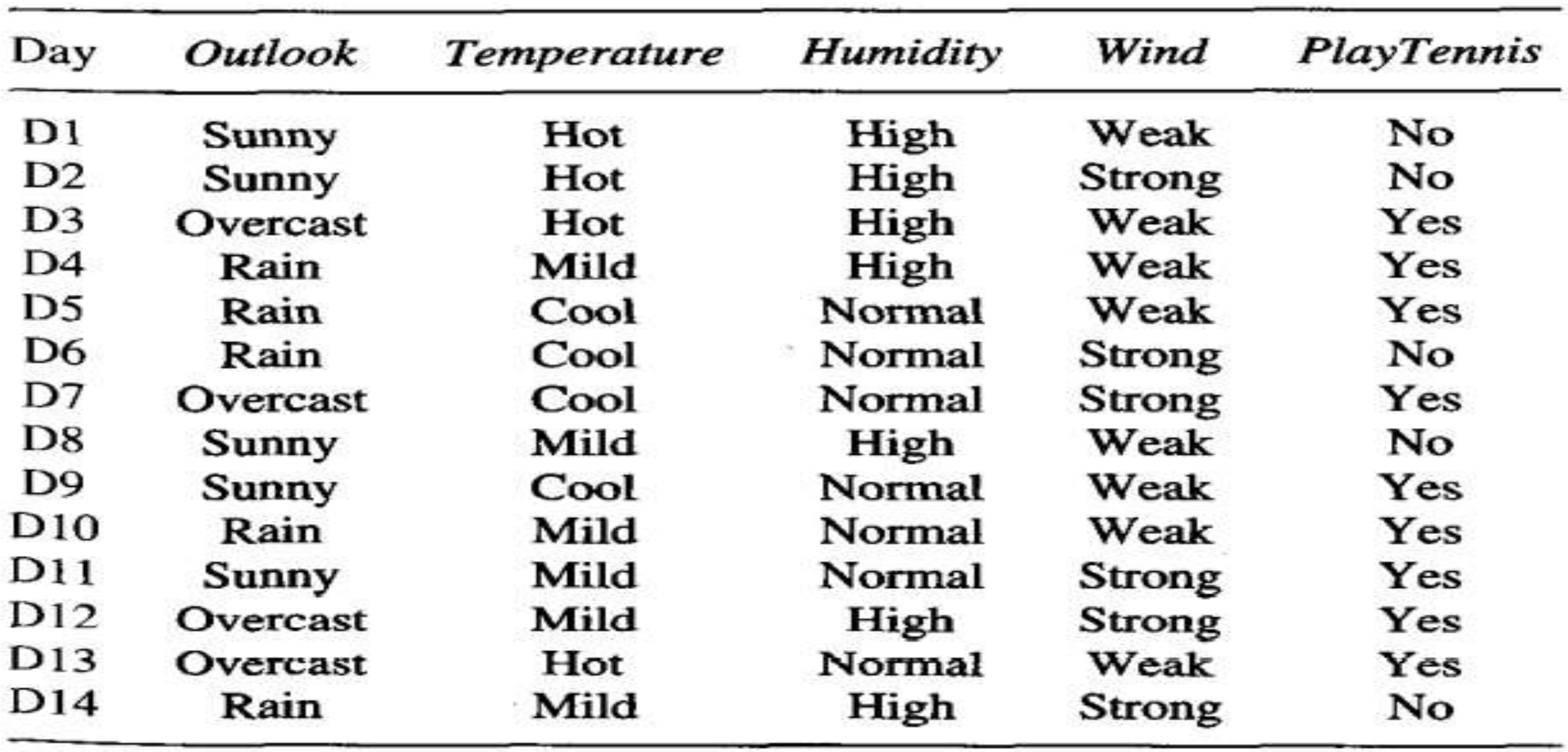
$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

- For example, suppose S is a collection of training-example days described by attributes including Wind, which can have the values Weak or Strong.
- As before, assume S is a collection containing 14 examples, [9+, 5-]. Of these 14 examples, suppose 6 of the positive and 2 of the negative examples have Wind = Weak, and the remainder have Wind = Strong.
- The information gain due to sorting the original 14 examples by the attribute Wind may then be calculated as

$$\begin{aligned}
Values(Wind) &= Weak, Strong \\
S &= [9+, 5-] \\
S_{Weak} &\leftarrow [6+, 2-] \\
S_{Strong} &\leftarrow [3+, 3-] \\
Gain(S, Wind) &= Entropy(S) - \sum_{v \in \{Weak, Strong\}} \frac{|S_v|}{|S|} Entropy(S_v) \\
&= Entropy(S) - (8/14) Entropy(S_{Weak}) \\
&\quad - (6/14) Entropy(S_{Strong}) \\
&= 0.940 - (8/14)0.811 - (6/14)1.00 \\
&= 0.048
\end{aligned}$$

An Illustrative Example

- To illustrate the operation of ID3, consider the learning task represented by the training examples of Table 3.2. Here the target attribute PlayTennis, which can have values yes or no for different Saturday mornings, is to be predicted based on other attributes of the morning in question.



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

TABLE 3.2
Training examples for the target concept *PlayTennis*.

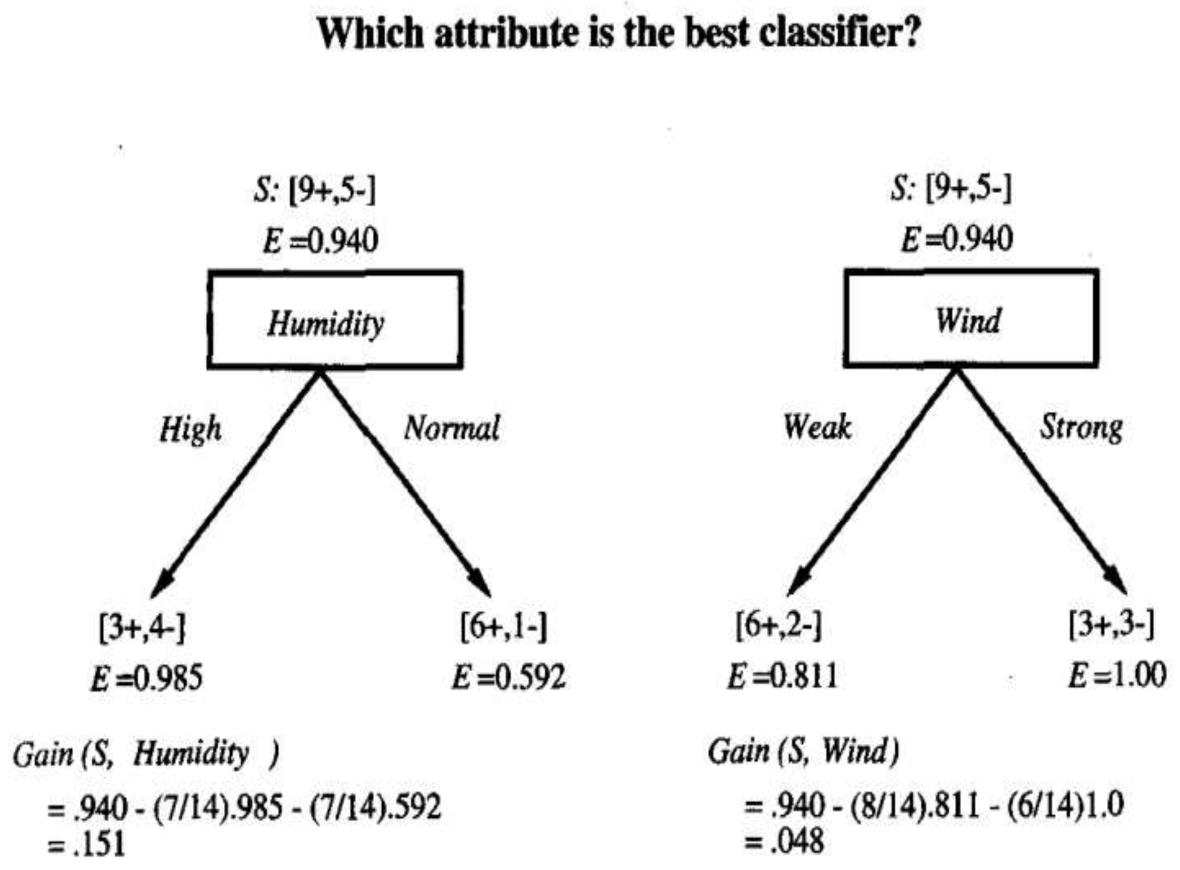
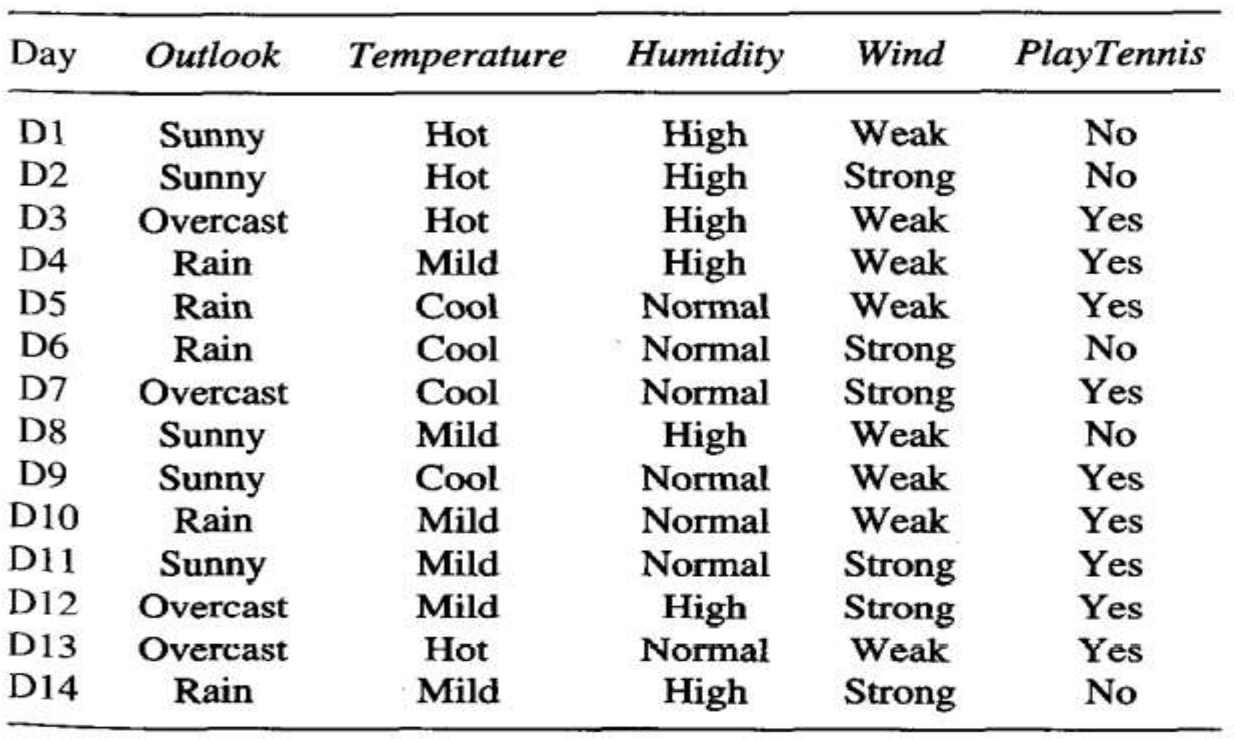


FIGURE 3.3
Humidity provides greater information gain than *Wind*, relative to the target classification. Here, E stands for entropy and S for the original collection of examples. Given an initial collection S of 9 positive and 5 negative examples, $[9+, 5-]$, sorting these by their *Humidity* produces collections of $[3+, 4-]$ ($Humidity = High$) and $[6+, 1-]$ ($Humidity = Normal$). The information gained by this partitioning is .151, compared to a gain of only .048 for the attribute *Wind*.

An Illustrative Example

- To illustrate the operation of ID3, consider the learning task represented by the training examples of Table 3.2. Here the target attribute PlayTennis, which can have values yes or no for different Saturday mornings, is to be predicted based on other attributes of the morning in question.



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

TABLE 3.2
Training examples for the target concept *PlayTennis*.

- Consider the first step through the algorithm, in which the topmost node of the decision tree is created. Which attribute should be tested first in the tree? ID3 determines the information gain for each candidate attribute (i.e., Outlook, Temperature, Humidity, and Wind), then selects the one with highest information gain. The computation of information gain for two of these attributes is shown in Figure 3.3. The information gain values for all four attributes are

$$\begin{aligned} Gain(S, Outlook) &= 0.246 \\ Gain(S, Humidity) &= 0.151 \\ Gain(S, Wind) &= 0.048 \\ Gain(S, Temperature) &= 0.029 \end{aligned}$$

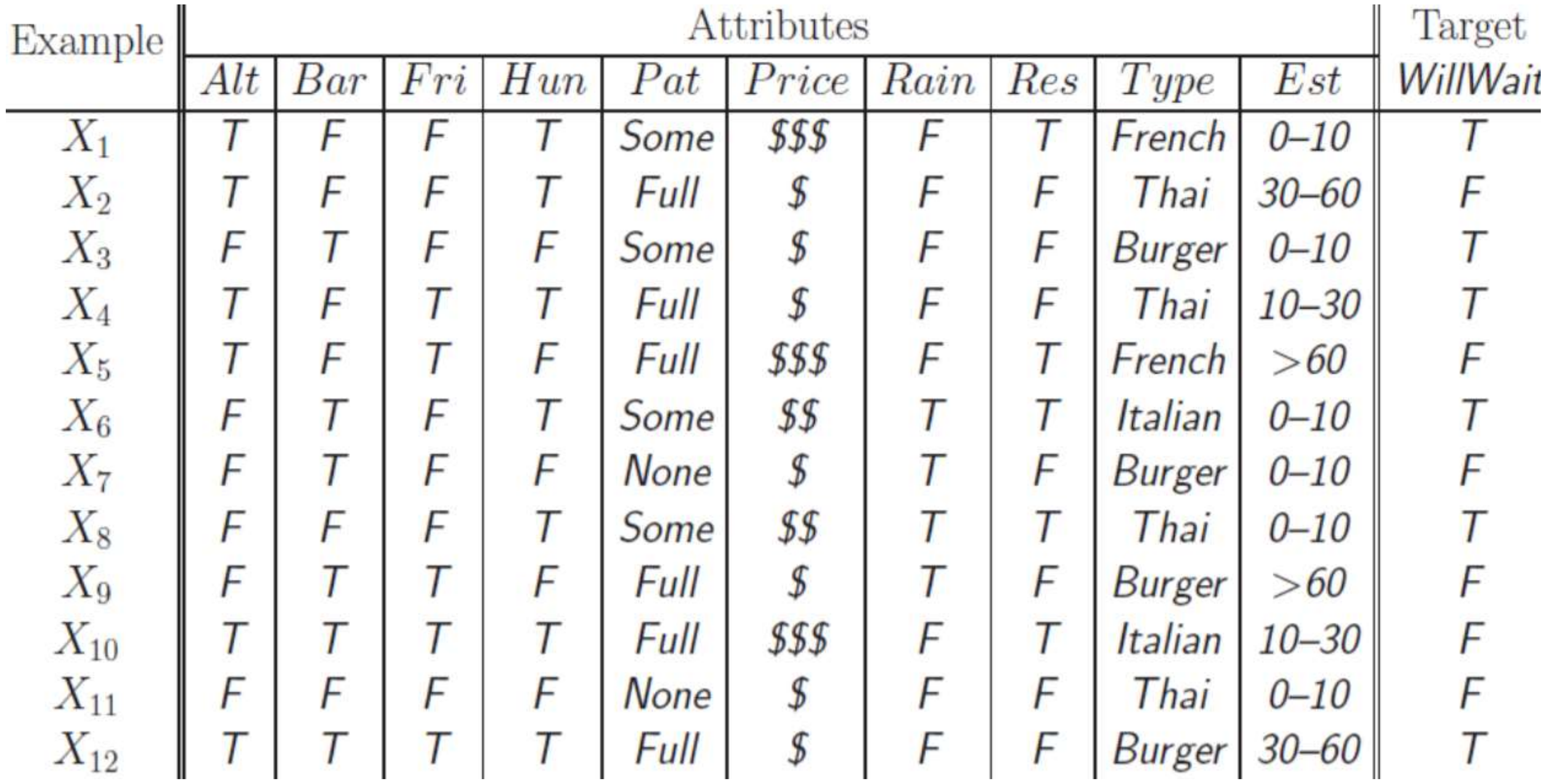
- According to the information gain measure, the Outlook attribute provides the best prediction of the target attribute, PlayTennis, over the training examples. Therefore, Outlook is selected as the decision attribute for the root node, and branches are created below the root for each of its possible values (i.e., Sunny, Overcast, and Rain).

- The process of selecting a new attribute and partitioning the training examples is now repeated for each non-terminal descendant node, this time using only the training examples associated with that node.
- This process continues for each new leaf node until either of two conditions is met:

(1) every attribute has already been included along this path through the tree, or

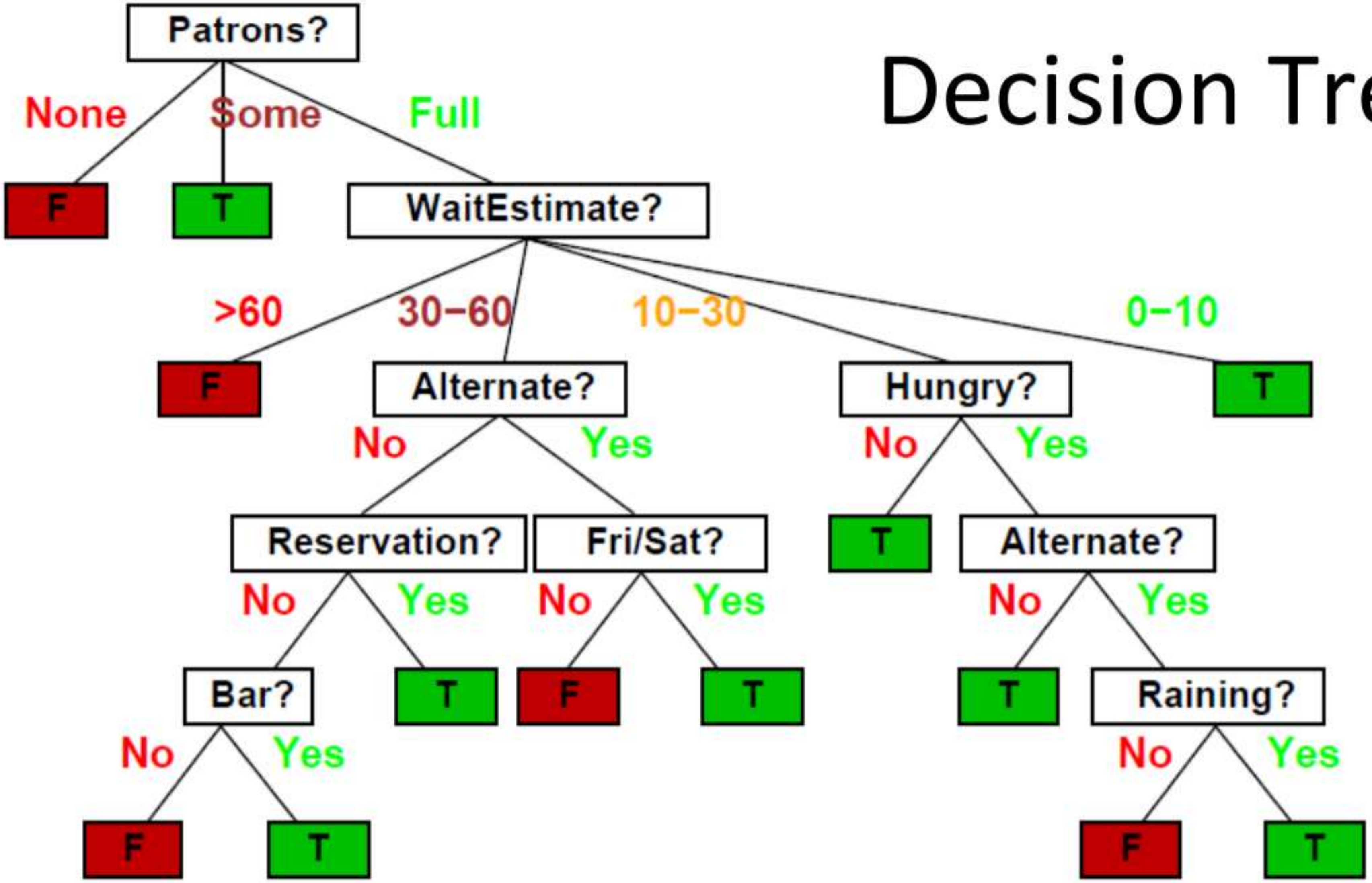
(2) the training examples associated with this leaf node all have the same target attribute value (i.e., their entropy is zero).

Exapmple 1



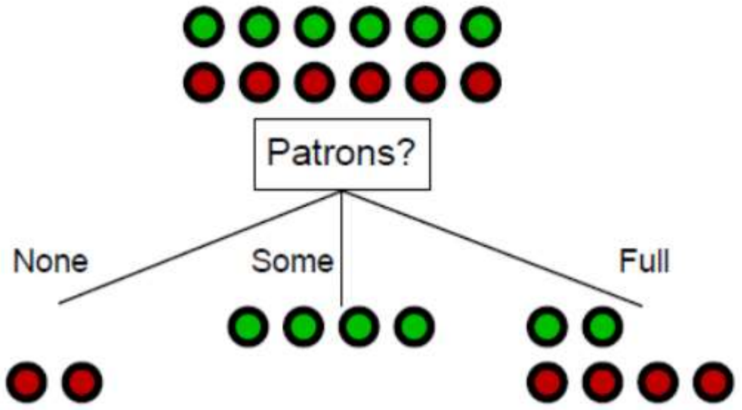
Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*T*	*F*	*F*	*T*	*Some*	*$$$*	*F*	*T*	*French*	*0–10*	*T*
X_2	*T*	*F*	*F*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*30–60*	*F*
X_3	*F*	*T*	*F*	*F*	*Some*	*$*	*F*	*F*	*Burger*	*0–10*	*T*
X_4	*T*	*F*	*T*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

Decision Trees



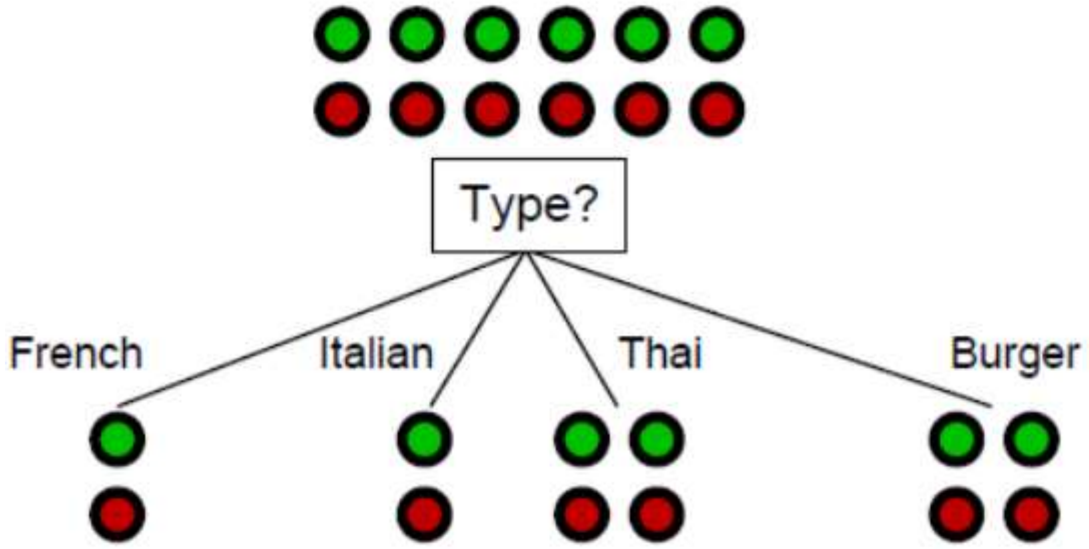
What is the output of the decision tree on this pattern?

Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*
F	T	F	F	Full	$$	T	F	Thai	10-30



- What is the information gain in this example?
  - H(E) = 1, since classes are evenly split at the top.
  - $H(E_1) = 0$, since we only have **will not wait** cases on the left child.
  - $H(E_2) = 0$, since we only have **will wait** cases on the middle child.
  - $H(E_3) = -\frac{2}{6}\log_2\left(\frac{2}{6}\right) - \frac{4}{6}\log_2\left(\frac{4}{6}\right) = 0.9183.$

$$I(E, \text{Patrons}) = H(E) - \frac{2}{12} * H(E_1) - \frac{4}{12} * H(E_2) - \frac{6}{12} * H(E_3)$$
$$= 1 - \frac{2}{12} * 0 - \frac{4}{12} * 0 - \frac{6}{12} * 0.9183 = 0.5409$$



- What is the information gain in this example?
  - H(E) = 1, since classes are evenly split.
  - $H(E_1) = H(E_2) = H(E_3) = H(E_4) = 1$, classes at all children are also evenly split.

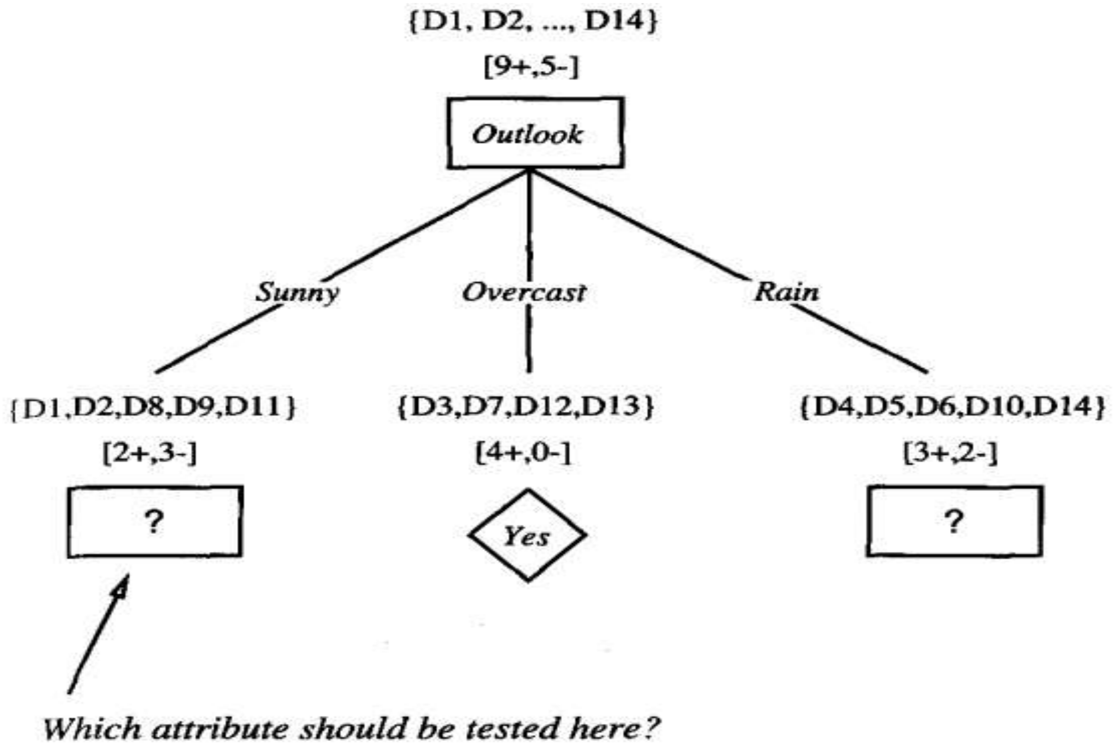
$$I(E, \text{Type}) =$$

$$= H(E) - \frac{2}{12} * H(E_1) - \frac{2}{12} * H(E_2) - \frac{4}{12} * H(E_3) - \frac{4}{12} * H(E4)$$

$$= 1 - \frac{2}{12} * 1 - \frac{2}{12} * 1 - \frac{4}{12} * 1 - \frac{4}{12} * 1 = 0$$

HYPOTHESIS SPACE SEARCH IN DECISION TREE LEARNING

- As with other inductive learning methods, ID3 can be characterized as searching a space of hypotheses for one that fits the training examples.
- The hypothesis space searched by ID3 is the set of possible decision trees.
- ID3 performs hill-climbing search through this hypothesis space, beginning with the empty tree, then considering progressively more elaborate hypotheses in search of a decision tree that correctly classifies the training data.
- The evaluation function that guides this hill-climbing search is the information gain measure. This search is depicted in Figure 3.5.



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

FIGURE 3.4
The partially learned decision tree resulting from the first step of ID3. The training examples are sorted to the corresponding descendant nodes. The *Overcast* descendant has only positive examples and therefore becomes a leaf node with classification *Yes*. The other two nodes will be further expanded, by selecting the attribute with highest information gain relative to the new subsets of examples.

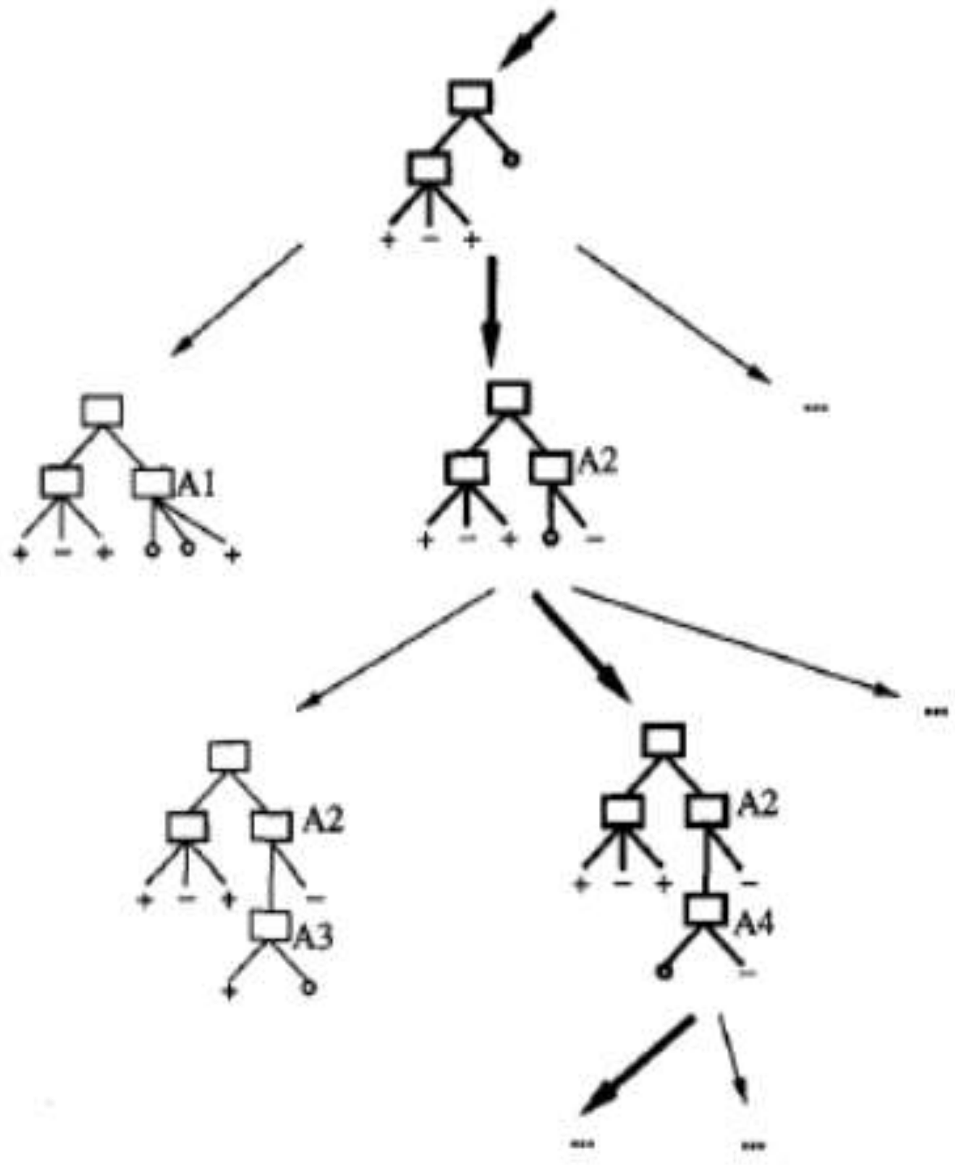


FIGURE 3.5
Hypothesis space search by ID3. ID3 searches through the space of possible decision trees from simplest to increasingly complex, guided by the information gain heuristic.

Capabilities and Limitations of ID3

- ID3's hypothesis space of all decision trees is a complete space of finite discrete-valued functions, relative to the available attributes.
- ID3 maintains only a single current hypothesis as it searches through the space of decision trees. This contrasts, with the earlier version space candidate Elimination method, which maintains the set of all hypotheses consistent with the available training examples.
- By determining only a single hypothesis, ID3 loses the capabilities that follow from explicitly representing all consistent hypotheses. It does not have the ability to determine how many alternative decision trees are consistent with the available training data.

- ID3 in its pure form performs no backtracking in its search. Once it, selects an attribute to test at a particular level in the tree, it never backtracks to reconsider this choice. Therefore, it is susceptible to the usual risks of hill-climbing search without backtracking, converging to locally optimal solutions that are not globally optimal.
- ID3 uses all training examples at each step in the search to make statistically based decisions regarding how to refine its current hypothesis. This contrasts with methods that make decisions incrementally, based on individual training examples (e.g., FIND-S or CANDIDATE-ELIMINATOION).
- One advantage of using statistical properties of all the examples (e.g., information gain) is that the resulting search is much less sensitive to errors in individual training examples.
- ID3 can be easily extended to handle noisy training data by modifying its termination criterion to accept hypotheses that imperfectly fit the training data

ISSUES IN DECISION TREE LEARNING

Practical issues in learning decision trees include determining how deeply to grow the decision tree, handling continuous attributes, choosing an appropriate attribute selection measure, and handling training data with missing attribute values, handling attributes with differing costs, and improving computational efficiency.

Avoiding Overfitting the Data

- The algorithm grows each branch of the tree just deeply enough to perfectly classify the training examples. In fact it can lead to difficulties when there is noise in the data, or when the number of training examples is too small to produce a representative sample of the true target function. In either of these cases, this simple algorithm can produce trees that ***overfit the training examples.***
- We say that a hypothesis overfits the training examples **if some other hypothesis that fits the training examples less well actually performs better over the entire distribution of instances** (i.e., including instances beyond the training set).

impact of overfitting in a typical application of decision tree learning. the ID3 algorithm is applied to the task of learning which medical patients have a form of diabetes.

The horizontal axis of this plot indicates the total number of nodes in the decision tree, as the tree is being constructed. The vertical axis indicates the accuracy of predictions made by the tree. The solid line shows the accuracy of the decision tree over the training examples, whereas the broken line shows accuracy measured over an independent set of test examples that are not included in the training set.

The accuracy of the tree over the training examples increases monotonically as the tree is grown. However, the accuracy measured over the independent test examples first increases, then decreases. As can be seen, once the tree size exceeds approximately *25 nodes,* further elaboration of the tree decreases its accuracy over the test examples despite increasing its accuracy on the training examples.

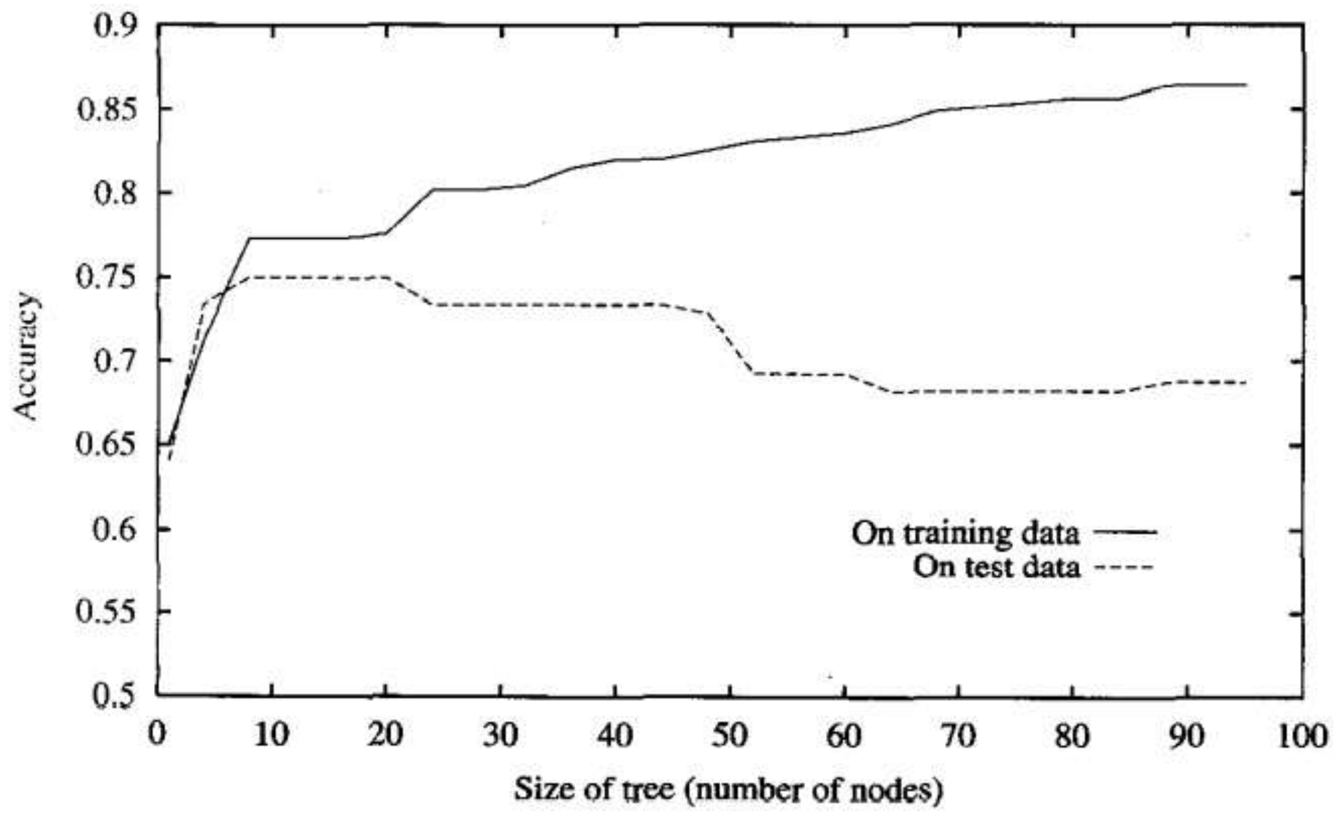


FIGURE 3.6
Overfitting in decision tree learning. As ID3 adds new nodes to grow the decision tree, the accuracy of the tree measured over the training examples increases monotonically. However, when measured over a set of test examples independent of the training examples, accuracy first increases, then decreases. Software and data for experimenting with variations on this plot are available on the World Wide Web at http://www.cs.cmu.edu/~tom/mlbook.html.

- How can it be possible for tree h to fit the training examples better than h', but for it to perform more poorly over subsequent examples?
- One way this can occur is when the training examples contain random errors or noise.

There are several approaches to avoiding overfitting in decision tree learning. These can be grouped into two classes:

- ➢ approaches that stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data
- ➢ approaches that allow the tree to overfit the data, and then post-prune the tree.

- Although the first of these approaches might seem more direct, the second approach of post-pruning overfit trees has been found to be more successful in practice.

Criterion to be used to determine the correct final tree size.

- Use a separate set of examples, distinct from the training examples, to evaluate the utility of post-pruning nodes from the tree
- Use all the available data for training, but apply a statistical test to estimate whether expanding (or pruning) a particular node is likely to produce an improvement beyond the training set. A chi-square test is used to estimate whether further expanding a node is likely to improve performance over the entire instance distribution, or only on the current sample of training data.
- Use an explicit measure of the complexity for encoding the training examples and the decision tree, halting growth of the tree when this encoding size is minimized. This approach, based on a heuristic called the Minimum Description Length principle

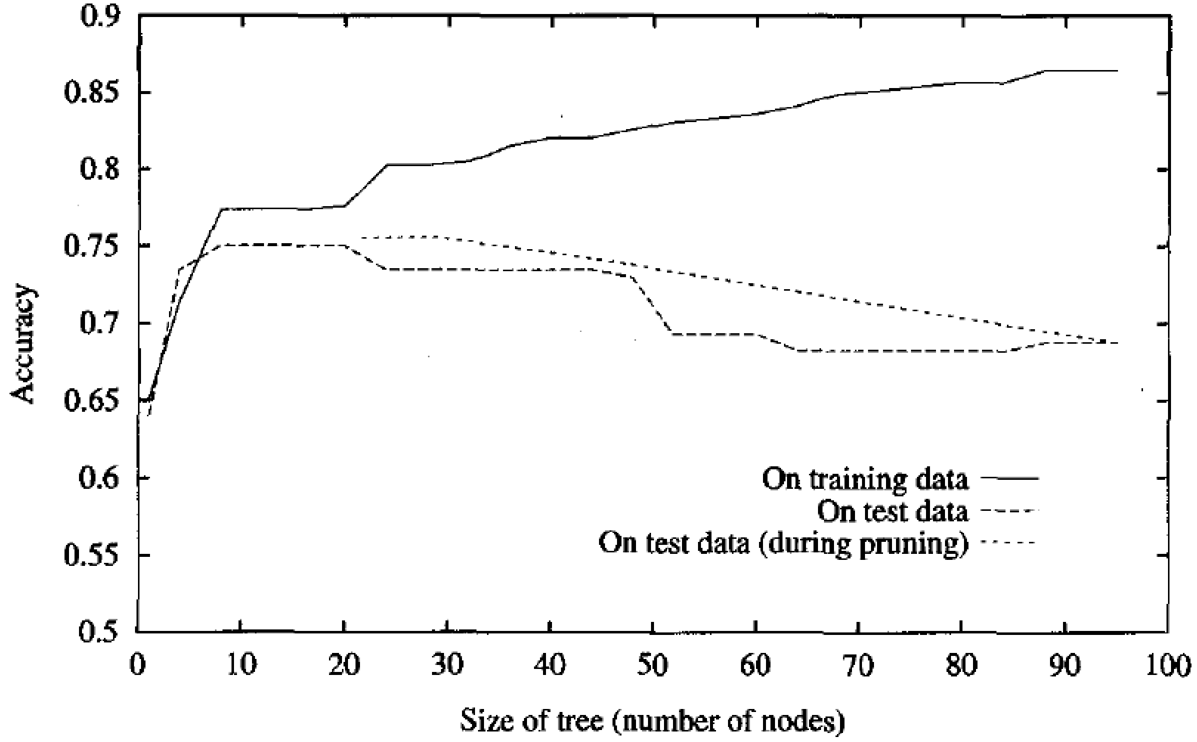
validation

- The available data are separated into two sets of examples: a training set, which is used to form the learned hypothesis, and a separate validation set, which is used to evaluate the accuracy of this hypothesis over subsequent data and, in particular, to evaluate the impact of pruning this hypothesis.
- One common heuristic is to withhold one-third of the available examples for the validation set, using the other two-thirds for training.

REDUCED ERROR PRUNING

- One approach, called reduced-error pruning, is to consider each of the decision nodes in the tree to be candidates for pruning.
- Pruning a decision node consists of removing the subtree rooted at that node, making it a leaf node, and assigning it the most common classification of the training examples affiliated with that node. Nodes are removed only if the resulting pruned tree performs no worse than-the original over the validation set.
- Nodes are pruned iteratively, always choosing the node whose removal most increases the decision tree accuracy over the validation set. Pruning of nodes continues until further pruning is harmful .

When pruning begins, the tree is at its maximum size and lowest accuracy over the test set. As pruning proceeds, the number of nodes is reduced and accuracy over the test set increases. Here, the available data has been split into three subsets: the training examples, the validation examples used for pruning the tree, and a set of test examples used to provide an unbiased estimate of accuracy over future unseen examples.



RULE POST PRUNING

Rule post-pruning involves the following steps:

1. Infer the decision tree from the training set, growing the tree until the training data is fit as well as possible and allowing overfitting to occur.
2. Convert the learned tree into an equivalent set of rules by creating one rule for each path from the root node to a leaf node.
3. Prune (generalize) each rule by removing any preconditions that result in improving its estimated accuracy.
4. Sort the pruned rules by their estimated accuracy, and consider them in this sequence when classifying subsequent instances.

In rule postpruning, one rule is generated for each leaf node in the tree. Each attribute test along the path from the root to the leaf becomes a rule antecedent (precondition) and the classification at the leaf node becomes the rule consequent (postcondition).

Figure 3.1 is translated into the rule

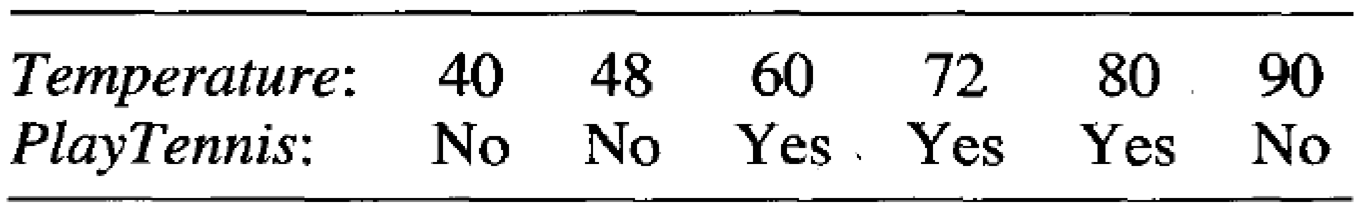
IF (Outlook = Sunny) Λ (Humidity = High)

THEN PlayTennis = No

- No pruning step is performed if it reduces the estimated rule accuracy. Tennis = No

Incorporating Continuous-Valued Attributes

- Our initial definition of ID3 is restricted to attributes that take on a discrete set of values.
- First, the target attribute whose value is predicted by the learned tree must be discrete valued.
- Second, the attributes tested in the decision nodes of the tree must also be discrete valued.
- This second restriction can easily be removed so that continuous-valued decision attributes can be incorporated into the learned tree.
- This can be accomplished by dynamically defining new discrete valued attributes that partition the continuous attribute value into a discrete set of intervals. In particular, for an attribute A that is continuous-valued, the algorithm can dynamically create a new boolean attribute A, that is true if $A < c$ and false otherwise. The only question is how to select the best value for the threshold c.



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

- pick a threshold, ***c, that produces the greatest*** information gain. By sorting the examples according to the continuous attribute **A, then identifying adjacent examples that differ in their target classification, we** can generate a set of candidate thresholds midway between the corresponding values of ***A.*** These candidate thresholds can then be evaluated by computing the information gain associated with each.

- In the current example, there are two candidate thresholds, corresponding to the values of Temperature at which the value of PlayTennis changes: (48 + 60)/2, and (80 + 90)/2.
- The information gain can then be computed for each of the candidate attributes, $Temperature_{>54}$,and $Temperature_{>85}$, and the best can be selected (Temperature>54).
- This dynamically created boolean attribute can then compete with the other discrete-valued candidate attributes available for growing the decision tree

Alternative Measures for Selecting Attributes:

- There is a natural bias in the information gain measure that favors attributes with many values over those with few values.
- As an extreme example, consider the attribute Date, which has a very large number of possible values (e.g., March 4, 1979). If we were to add this attribute to the data in Table 3.2, it would have the highest information gain of any of the attributes.
- This is because Date alone perfectly predicts the target attribute over the training data. Thus, it would be selected as the decision attribute for the root node of the tree and lead to a (quite broad) tree of depth one, which perfectly classifies the training data.
- Of course, this decision tree would fare poorly on subsequent examples, because it is not a useful predictor despite the fact that it perfectly separates the training data.

- One alternative measure that has been used successfully is the gain ratio (Quinlan 1986).
- The gain ratio measure penalizes attributes such as Date by incorporating a term, called split information, that is sensitive to how broadly and uniformly the attribute splits the data:

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

where S1 through S, are the ***c subsets of examples resulting from partitioning S*** by the c-valued attribute A. Note that SplitInfomzation is actually the entropy of S with respect to the values of attribute A. This is in contrast to our previous uses of entropy, in which we considered only the entropy of S with respect to the target attribute whose value is to be predicted by the learned tree.

- The Gain Ratio measure is defined in terms of the earlier Gain measure, as well as this Splitlnformation, as follows

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

- the Splitlnfomzation term discourages the selection of attributes with many uniformly distributed values. For example, consider a collection of n examples that are completely separated by attribute A (e.g., Date). In this case, the Splitlnfomzation value will be log, n. In contrast, a boolean attribute B that splits the same n examples exactly in half will have Splitlnfomzation of 1. If attributes A and B produce the same information gain, then clearly B will score higher according to the Gain Ratio measure.

- One practical issue that arises in using GainRatio in place of Gain to select attributes is that the denominator can be zero or very small when $|S_i| \approx |S|$

for one of the Si. This either makes the GainRatio undefined or very large for attributes that happen to have the same value for nearly all members of S. To avoid selecting attributes purely on this basis, we can adopt some heuristic such as first calculating the Gain of each attribute, then applying the GainRatio test only considering those attributes with above average Gain

Handling Training Examples with Missing Attribute Values

- In certain cases, the available data may be missing values for some attributes.
- For example, in a medical domain in which we wish to predict patient outcome based on various laboratory tests, it may be that the lab test Blood-Test-Result is available only for a subset of the patients.
- In such cases, it is common to estimate the missing attribute value based on other examples for which this attribute has a known value. One strategy for dealing with the missing attribute value is to assign it the value that is most common among training examples at node n.

- Alternatively, we might assign it the most common value among examples at node n that have the classification c(x).
- The elaborated training example using this estimated value for A(x) can then be used directly by the existing decision tree learning algorithm.

- A second, more complex procedure is to assign a probability to each of the possible values of *A rather than simply assigning the most common value to A(x).*
- These probabilities can be estimated again based on the observed frequencies of the various values for *A among the examples at node n.*
- *For example, given a* boolean attribute *A, if node n contains six known examples with A = 1 and four* with *A = 0, then we would say the probability that A(x) = 1 is 0.6, and the* probability that *A(x) = 0 is 0.4. A fractional 0.6 of instance x is now distributed* down the branch for *A = 1, and a fractional 0.4 of x down the other tree branch.*

Handling Attributes with Differing Costs

- In some learning tasks the instance attributes may have associated costs. For example, in learning to classify medical diseases we might describe patients I terms of attributes such **as Temperature, BiopsyResult, Pulse, BloodTestResults,** etc. These attributes vary significantly in their costs, both in terms of monetary cost and cost to patient comfort. In such tasks, we would prefer decision trees that use low-cost attributes where possible, relying on high-cost attributes only when needed to produce reliable classifications.

- ID3 can be modified to take into account attribute costs by introducing a cost term into the attribute selection measure.
- divide the ***gain*** by the cost of the attribute, so that lower-cost attributes would be preferred. While such cost-sensitive measures do not guarantee finding an optimal cost-sensitive decision tree, they do bias the search in favor of low-cost attributes.
- Tan and Schlimmer (1990) and Tan (1993) used

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

Nunez (1988) used

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

where ***w E [0, 1] is a constant that determines the relative importance of cost*** versus information gain.