Machine Learning

Supervised Learning-I

Topics

- **Decision Tree Learning**
    - **Decision Tree representation**
    - **Appropriate problems for the decision tree**
    - **Basic decision tree algorithm**
    - **Basics of C4.5 and issues in decision tree learning.**
- **K-Nearest Neighbour**
    - **K-NN rule**
    - **Measuring Similarity**
    - **Irrelevant attributes**
    - **scaling problems.**

Decision Tree Learning

INTRODUCTION

- A decision tree is a type of supervised machine learning used to categorize or make predictions based on **how a previous set of questions were answered.**
- It builds a **flowchart-like tree structure** where each **internal node** denotes **a test on an attribute**, **each branch represents an outcome of the test**, and **each leaf node (terminal node) holds a class label.**
- It is a versatile supervised machine-learning algorithm, which is used for both classification and regression problems.

INTRODUCTION

- Decision tree learning is one of the most widely used and practical methods for inductive inference and these have been successfully applied to a broad range of tasks from learning to diagnose **medical cases to learning to assess credit risk of loan applicants**.
- it is also used in **Random Forest** to train on different subsets of training data, which makes random forest one of the most powerful algorithms in machine learning.

INTRODUCTION

- It is constructed by recursively **splitting the training data into subsets based on the values of the attributes** until a stopping criterion is met, such as the maximum depth of the tree or the minimum number of samples required to split a node.

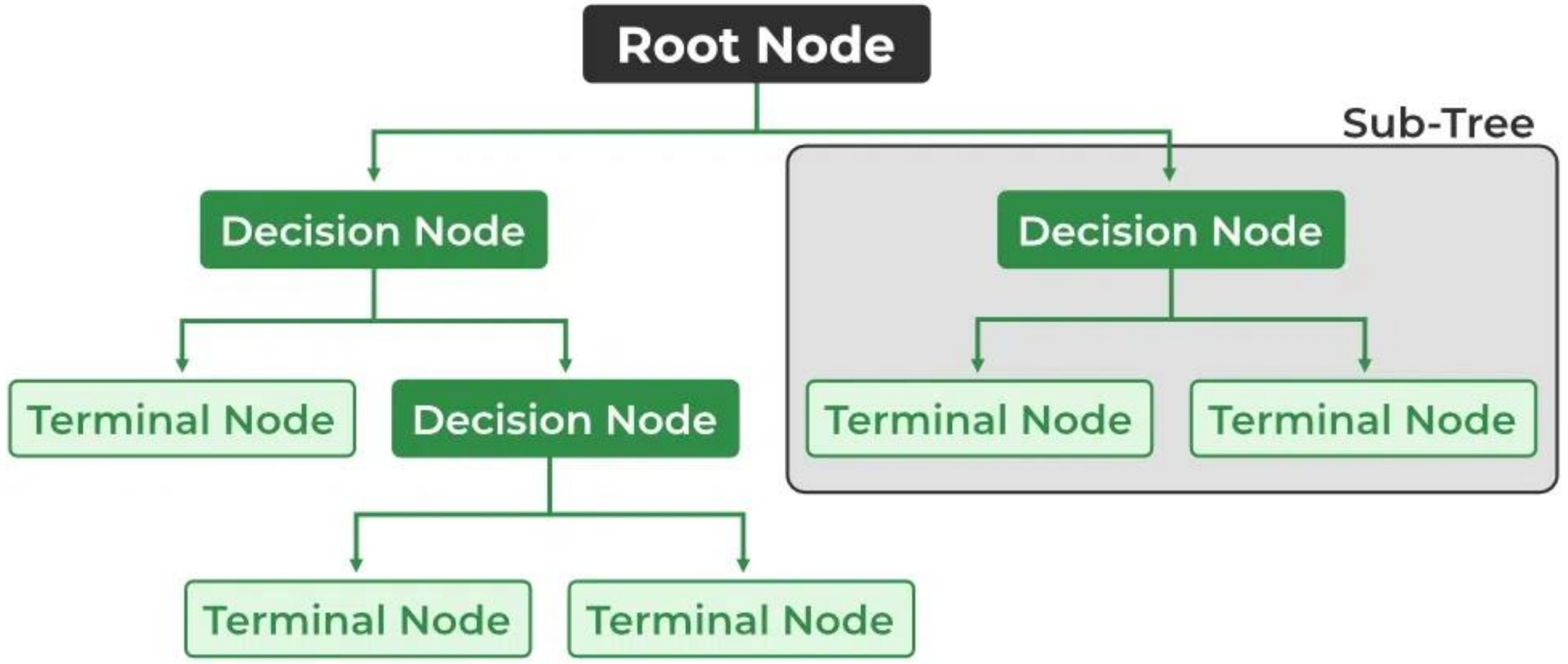
INTRODUCTION

Let’s define some key terms of a decision tree.

- **Root node:** The base of the decision tree.
- **Splitting:** The process of dividing a node into multiple sub-nodes.
- **Decision node:** When a sub-node is further split into additional sub-nodes.
- **Leaf node:** When a sub-node does not further split into additional sub-nodes; represents possible outcomes.
- **Pruning:** The process of removing sub-nodes of a decision tree.
- **Branch**: A subsection of the decision tree consisting of multiple nodes.

INTRODUCTION

Decision tree.



INTRODUCTION

How it works?

- 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.

INTRODUCTION

- Figure 3.1 illustrates a typical learned decision tree. This decision tree classifies **Saturday mornings according to whether they are suitable for playing tennis**.

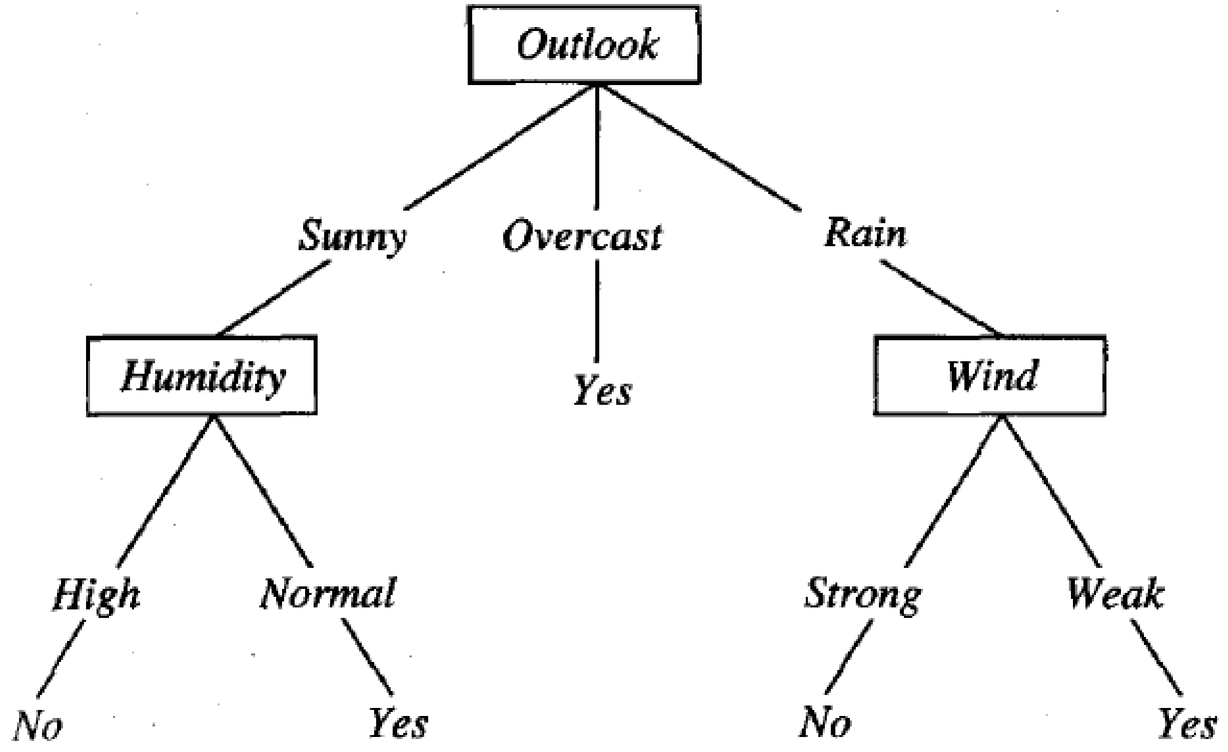


FIGURE 3.1
A decision tree for the concept *PlayTennis*. An example is classified by sorting it through the tree to the appropriate leaf node, then returning the classification associated with this leaf (in this case, *Yes* or *No*). This tree classifies Saturday mornings according to whether or not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

INTRODUCTION

- For example, the instance:

$$\langle Outlook = Sunny,\ Temperature = Hot,\ Humidity = High,\ Wind = Strong \rangle$$

- 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***).

INTRODUCTION

- 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

$$\begin{aligned} & (Outlook = Sunny \wedge Humidity = Normal) \\ \vee \quad & (Outlook = Overcast) \\ \vee \quad & (Outlook = Rain \wedge Wind = Weak) \end{aligned}$$

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.
- 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.

APPROPRIATE PROBLEMS FOR DECISION TREE LEARNING

- **Instances are represented by attribute-value pairs.**
  - Instances are described by a fixed set of attributes and their values (***Temperature, Hot***)
  - The easiest situation for decision tree learning is when each attribute takes on a **small number of disjoint possible values**

    (e.g., Hot, Mild, Cold)
  - **Some algs. allow handling real-valued attributes as well (e.g., representing Temperature values numerically).**

APPROPRIATE PROBLEMS FOR DECISION TREE LEARNING

- **The target function has discrete output values**
  - Apart boolean classification, Decision tree can allow learning functions with more than two possible output values.
  - A more substantial extension allows learning target functions with real-valued outputs

APPROPRIATE PROBLEMS FOR DECISION TREE LEARNING

- **Disjunctive descriptions may be required**
  - As noted above, decision tree naturally 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.

APPROPRIATE PROBLEMS FOR DECISION TREE LEARNING

- **The training data may contain missing attribute values**
  - Decision tree methods can be used even when some training examples have unknown values
  - *if the Humidity of the day is known for only some of the training examples*

APPROPRIATE PROBLEMS FOR DECISION TREE LEARNING

- **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.**

THE BASIC DECISION TREE LEARNING ALGORITHM - ID3 algorithm

- Most decision algorithms employ a **top-down, greedy search** through the space of possible decision trees.
- This approach is exemplified by the **ID3 algorithm** (Iterative Dichotomiser 3 Ross Quinlan 1986) and its successor **C4.5**
- **ID3 is a basic decision algorithm that** learns decision trees by constructing them top down, beginning with the question "**which attribute should be tested at the root of the tree?'**

THE BASIC DECISION TREE LEARNING ALGORITHM - ID3 algorithm

- For this, 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 BASIC DECISION TREE LEARNING ALGORITHM - ID3 algorithm

- 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.
- This forms a **greedy search** for an acceptable decision tree, in which the algorithm never backtracks to reconsider earlier choices

THE BASIC DECISION TREE LEARNING ALGORITHM - ID3 algorithm

Which Attribute Is the Best Classifier in ID3?

- How to Select 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.**
- ID3 uses this **information gain measure to select among the candidate attributes** at each step while growing the tree

THE BASIC DECISION TREE LEARNING ALGORITHM - ID3 algorithm

ENTROPY MEASURES HOMOGENEITY OF EXAMPLES

- We ***entropy*** first. It 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:

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

$p_{\oplus}$, is **the proportion of positive examples** in S
$p_{\ominus}$, is the proportion of **negative examples in S.**

THE BASIC DECISION TREE LEARNING ALGORITHM - ID3 algorithm

ENTROPY MEASURES HOMOGENEITY OF EXAMPLES

- To illustrate, suppose S is a collection of **14 examples** of some boolean concept, including **9 positive and 5 negative examples**

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

- **Notice that the entropy is 0 if all members of S belong to the same class.**

THE BASIC DECISION TREE LEARNING ALGORITHM - ID3 algorithm

ENTROPY MEASURES HOMOGENEITY OF EXAMPLES

- For example, if all members are positive ($p_\oplus$ = 1), then $p_\ominus$, is 0. Then entropy is;

$$Entropy(S) = -1 \cdot \log_2(1) - 0 \cdot \log_2 0 = -1 \cdot 0 - 0 \cdot \log_2 0 = 0.$$

- The 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**

THE BASIC DECISION TREE LEARNING ALGORITHM - ID3 algorithm

ENTROPY MEASURES HOMOGENEITY OF EXAMPLES

- Figure 3.2 shows the form of the entropy function relative to a boolean classification, as $p_\oplus$ varies between 0 and 1.

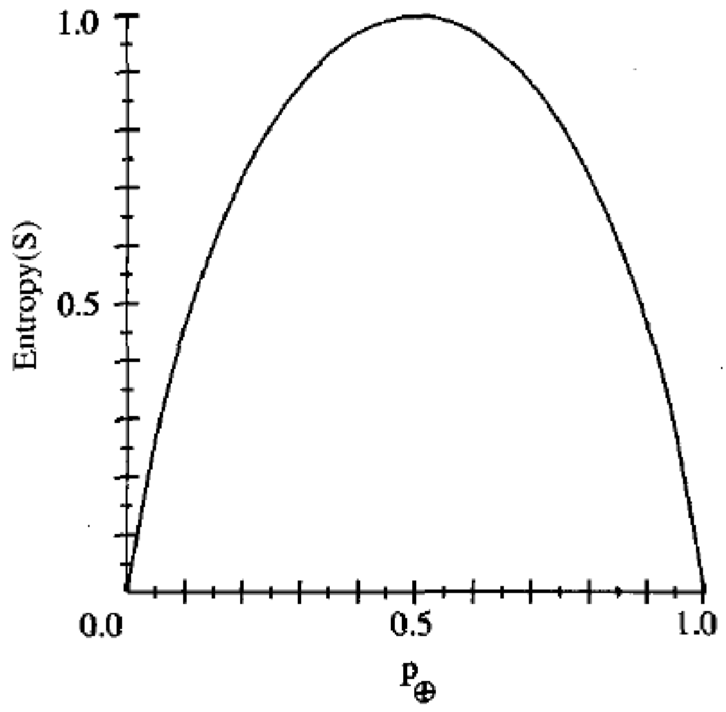


FIGURE 3.2
The entropy function relative to a boolean classification, as the proportion, $p_\oplus$, of positive examples varies between 0 and 1.

THE BASIC DECISION TREE LEARNING ALGORITHM - ID3 algorithm

ENTROPY MEASURES HOMOGENEITY OF EXAMPLES

- 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 $p_{\oplus}$, is 1, the receiver knows the drawn example will be positive, so no message need be sent.
- If $p_{\oplus}$ is 0.5, **one bit** is required to indicate whether the drawn example is positive or negative.
- If $p_{\oplus}$ 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.)

THE BASIC DECISION TREE LEARNING ALGORITHM - ID3 algorithm

ENTROPY MEASURES HOMOGENEITY OF EXAMPLES

- Thus far we have discussed entropy in the special case where the **target classification is boolean.** More generally, if the target attribute can take on c different values, then the entropy of S relative to this c-wise classification is defined as :

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

- where p_i is the proportion of S belonging to class i. Note also that if the target attribute can take on c possible values, the entropy can be as large as log, c

THE BASIC DECISION TREE LEARNING ALGORITHM - ID3 algorithm

INFORMATION GAIN MEASURES

- 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, called **information gain.**
- **Information gain** is simply the **expected reduction in entropy** caused by **partitioning the examples** according to this attribute.

THE BASIC DECISION TREE LEARNING ALGORITHM - ID3 algorithm

INFORMATION GAIN MEASURES

- The information gain, ***Gain(S, A)*** of an attribute A, relative to a collection of examples S, is defined as

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

Values(A) →the set of all possible values for attribute A

S → the subset of S for which attribute A has value v

$$S_v = \{s \in S | A(s) = v\}$$

THE BASIC DECISION TREE LEARNING ALGORITHM - ID3 algorithm

INFORMATION GAIN MEASURES

- First term in Equation is just the **entropy of the original collection S,** and the second term is the **expected value of the entropy after S is partitioned using attribute A**.
- The **expected entropy** is the sum of the entropies of each subset S_v, weighted by the fraction of examples that belong to $\frac{|S_v|}{S}$ that belong to S_v
-

THE BASIC DECISION TREE LEARNING ALGORITHM - ID3 algorithm

INFORMATION GAIN MEASURES

- ***Gain(S, A)*** is therefore
  - The expected reduction in entropy caused by knowing the value of attribute A.
  - Information provided about the target &action value, given the value of some other attribute A.
  - The number of bits saved when encoding the target value of an arbitrary member of S, by knowing the value of attribute A.

THE BASIC DECISION TREE LEARNING ALGORITHM - ID3 algorithm

INFORMATION GAIN MEASURES

- *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**. Calculate the information gain due to sorting the **original 14 examples** by the attribute **Wind** :*

THE BASIC DECISION TREE LEARNING ALGORITHM - ID3 algorithm

INFORMATION GAIN MEASURES Type equation here.

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

THE BASIC DECISION TREE LEARNING ALGORITHM - ID3 algorithm

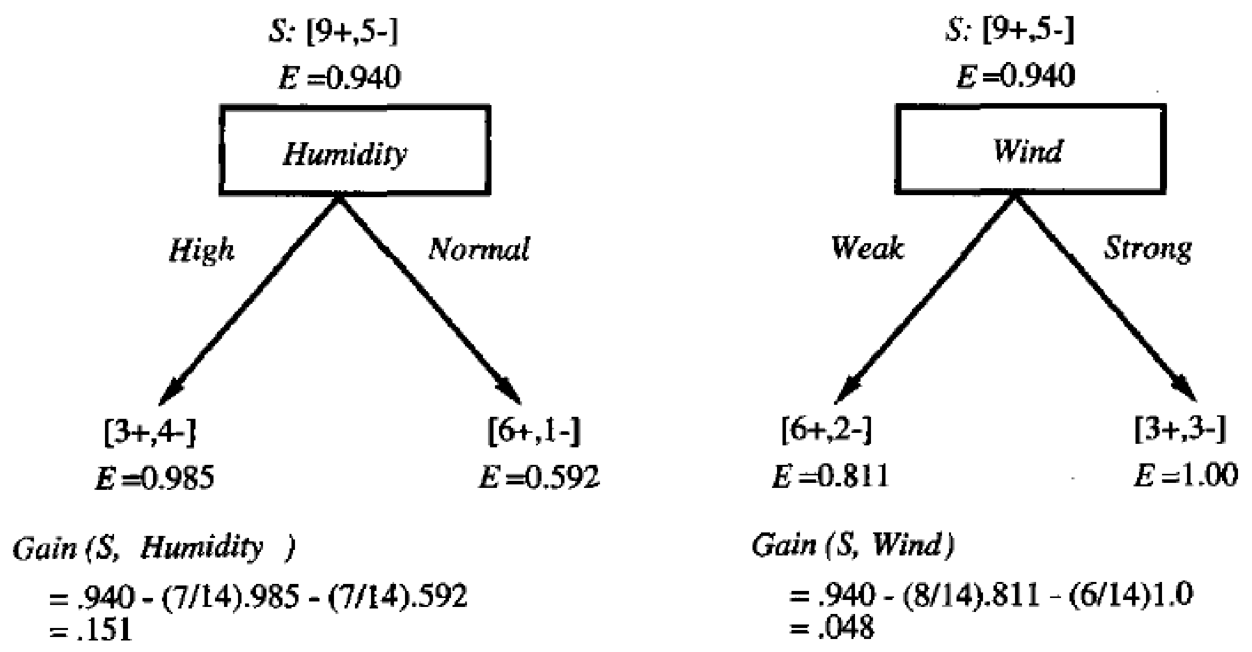
INFORMATION GAIN MEASURES

- **Information gain is precisely the measure** used by ID3 to select the **best attribute** at each step in growing the tree. The use of information gain to evaluate the relevance of attributes is summarized in Figure 3.3.
- In this figure the information gain of two different attributes, Humidity and Wind, is computed in order to determine which is the better attribute for classifying the training examples shown in Table 3.2.

THE BASIC DECISION TREE LEARNING ALGORITHM - ID3 algorithm

INFORMATION GAIN MEASURES

Which attribute is the best classifier?



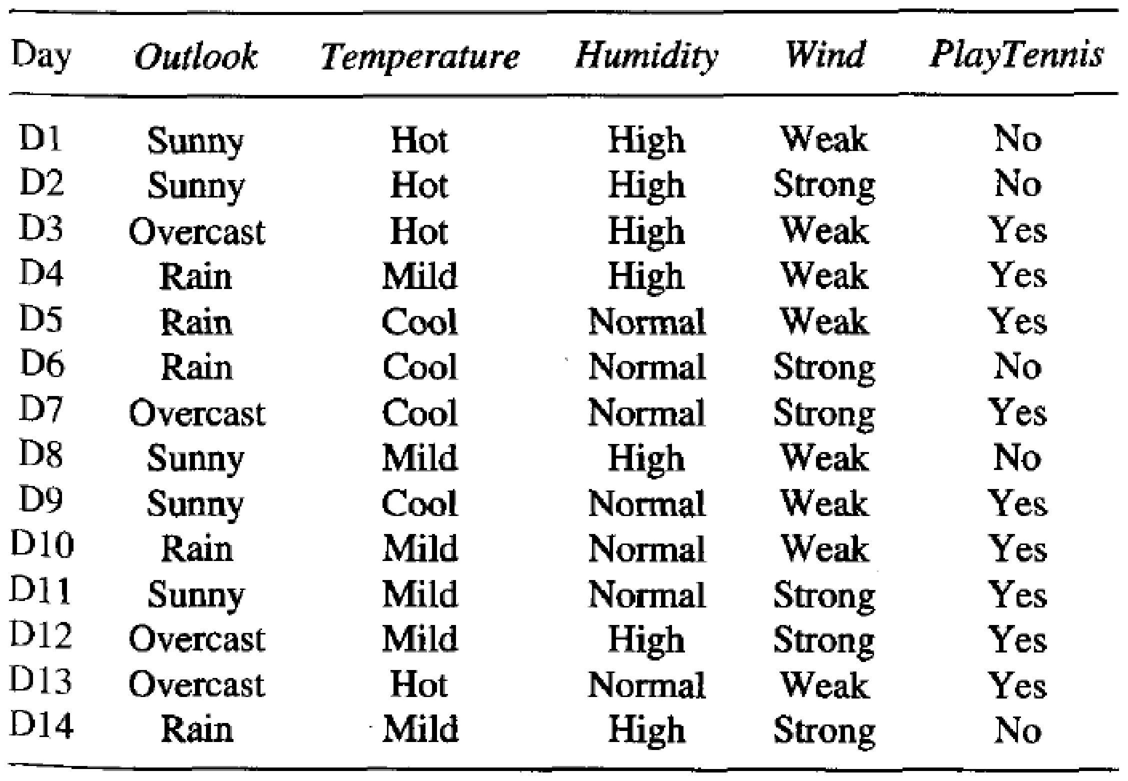
Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool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

***Humidity* provides greater information gain than *Wind*, relative to the target classification**

THE BASIC DECISION TREE LEARNING ALGORITHM - ID3 algorithm

An Illustrative Example

To illustrate the operation of ID3, consider the learning task represented by the training examples of Table 3.2



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

Here the target attribute ***PlayTennis,*** which can have values ***yes*** or ***no*** for different Saturday mornings, is to be predicted basedon other attributes of the morning in question

THE BASIC DECISION TREE LEARNING ALGORITHM - ID3 algorithm

An Illustrative Example

- First step is to find which attribute should be tested first in the tree?
- To do this, 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 BASIC DECISION TREE LEARNING ALGORITHM - ID3 algorithm

An Illustrative Example

- The information gain values for all four attributes are

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

- *S* denotes the collection of training examples from Table 3.2.

THE BASIC DECISION TREE LEARNING ALGORITHM - ID3 algorithm

An Illustrative Example

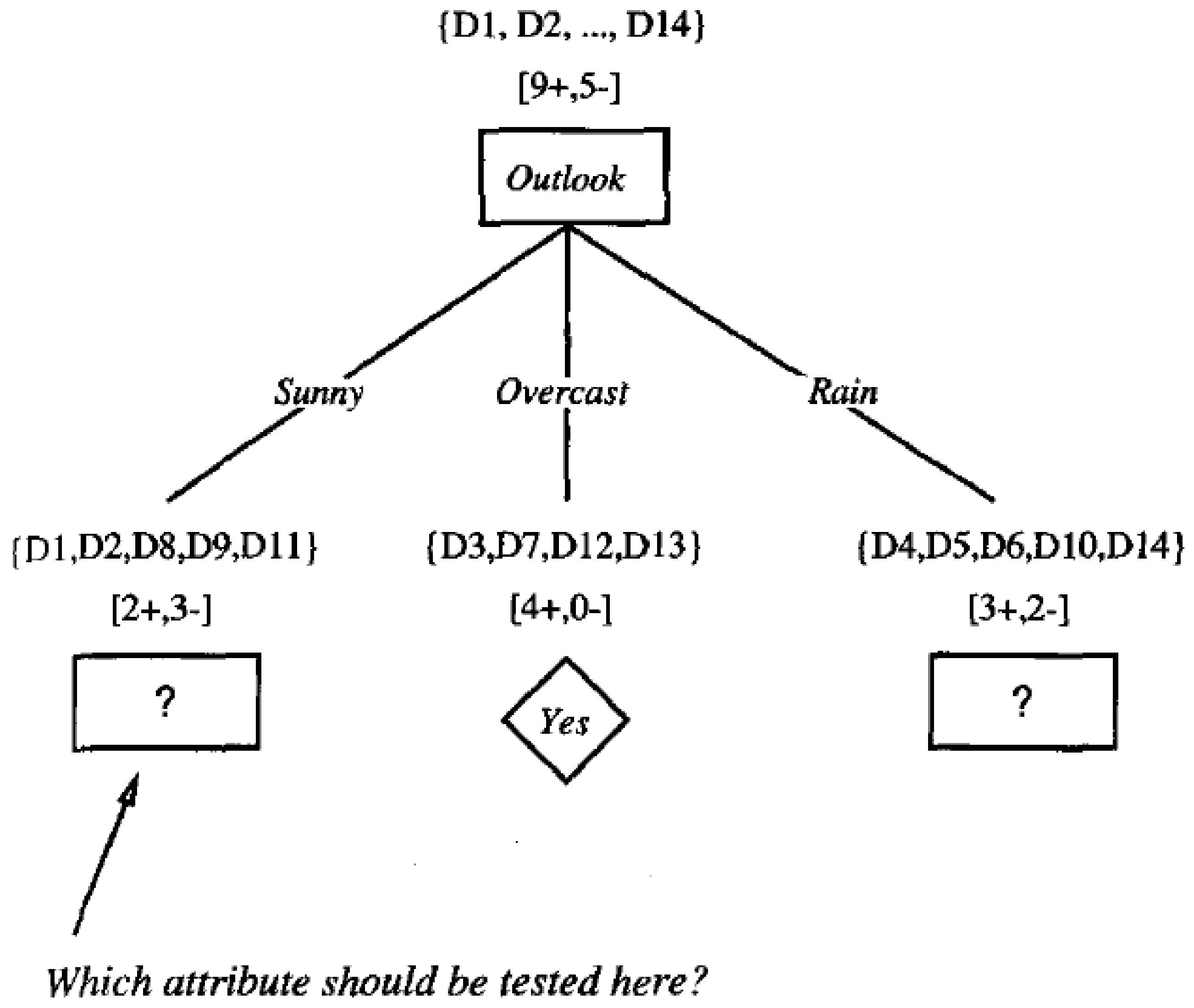
- According to the **information gain measure**, **Outlook** is selected as the decision attribute for the root node, and branches are created below the root for each of its possible values (i.e., **Sunny, Overcast, and Rain**).

- The resulting partial decision tree is shown in **Figure 3.4,** along with the **training examples sorted** to each new descendant node.

THE BASIC DECISION TREE LEARNING ALGORITHM - ID3 algorithm

An Illustrative Example

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



THE BASIC DECISION TREE LEARNING ALGORITHM - ID3 algorithm

An Illustrative Example

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

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

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

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

THE BASIC DECISION TREE LEARNING ALGORITHM - ID3 algorithm

An Illustrative Example

- Note that every example for which Outlook = Overcast is also a positive example of PlayTennis.
- Therefore, this node of the tree becomes a leaf node with the classification **PlayTennis = Yes.**
- In contrast, the descendants corresponding to **Outlook = Sunny** and **Outlook = Rain** still have nonzero entropy, and the decision tree will be further elaborated below these nodes

THE BASIC DECISION TREE LEARNING ALGORITHM - ID3 algorithm

An Illustrative Example

- The process of selecting a new attribute and partitioning the training examples is now repeated for each nonterminal descendant node, this time using only the training examples associated with that node.
- Attributes that have been incorporated higher in the tree are excluded, so that any given attribute can appear at most once along any path through the tree.

THE BASIC DECISION TREE LEARNING ALGORITHM - ID3 algorithm

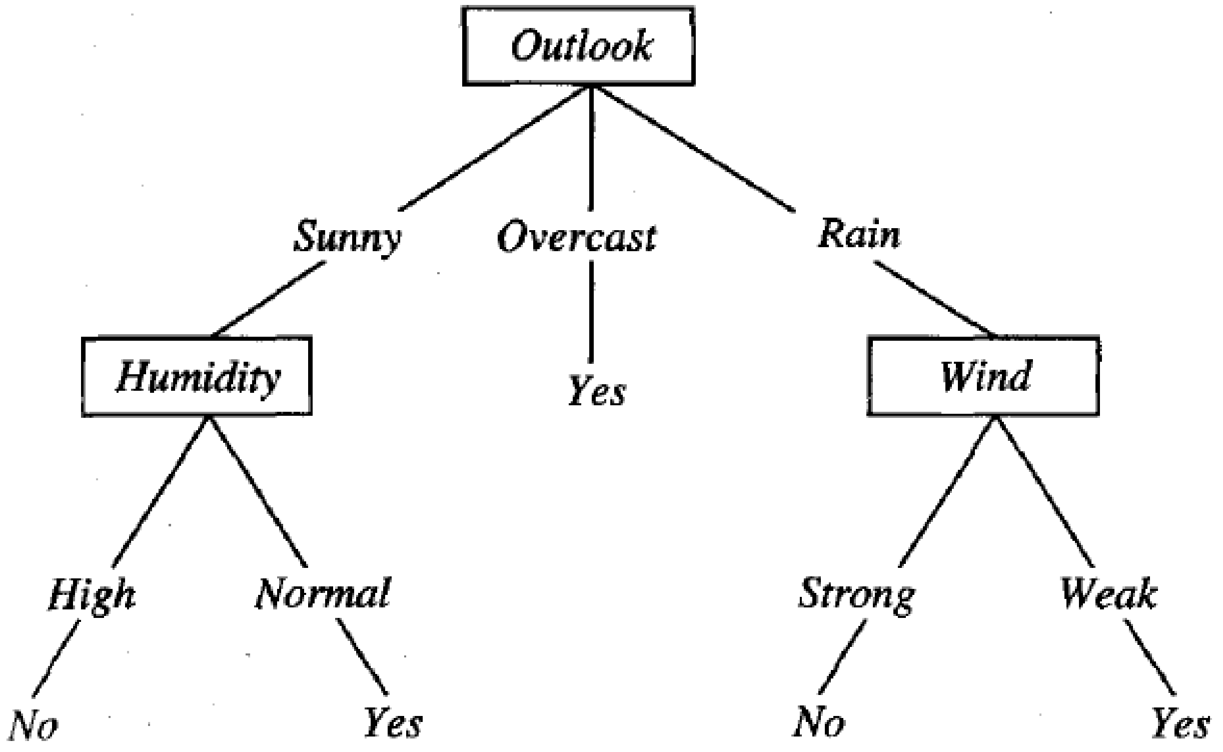
An Illustrative Example

- This process continues for each new leaf node until either of two conditions is met:
  - Every attribute has already been included along this path through the tree, or
  - The training examples associated with this leaf node all have the same target attribute value (i.e., their entropy is zero).

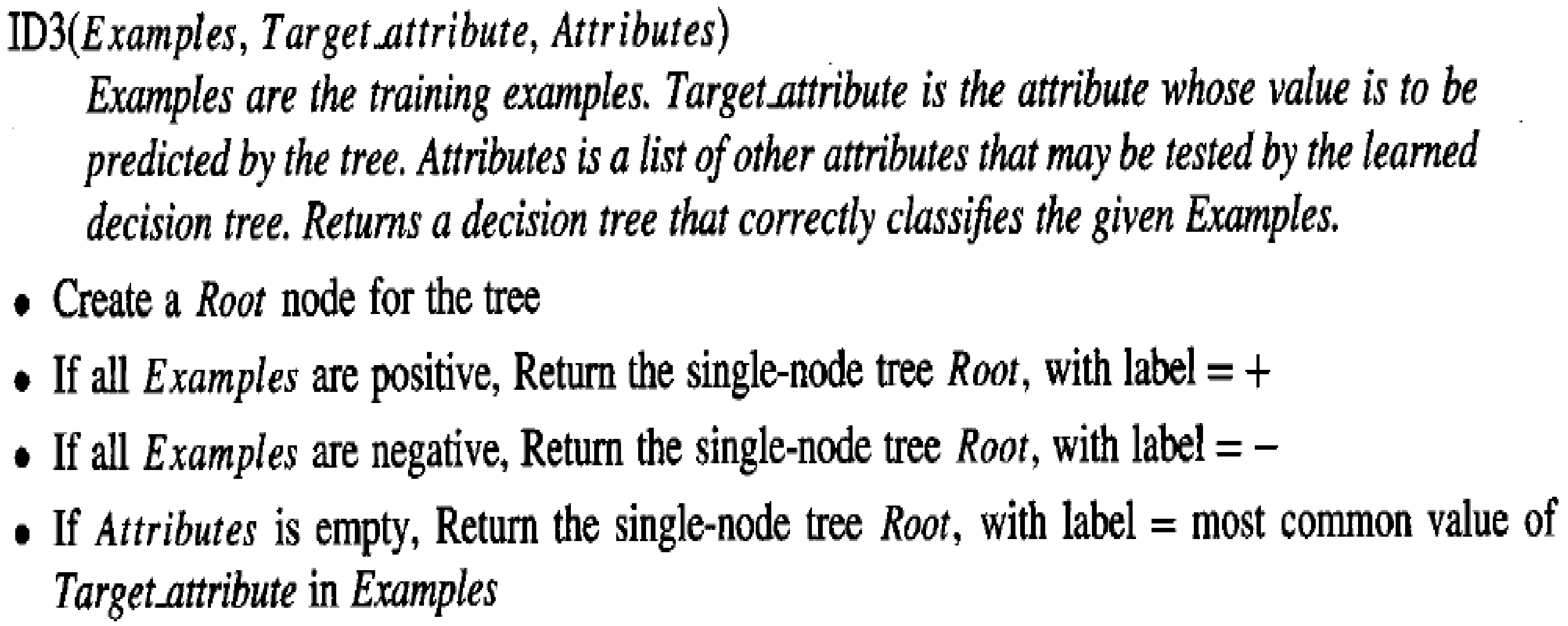
THE BASIC DECISION TREE LEARNING ALGORITHM - ID3 algorithm

Final Decision Tree

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



THE BASIC DECISION TREE LEARNING ALGORITHM - ID3 algorithm

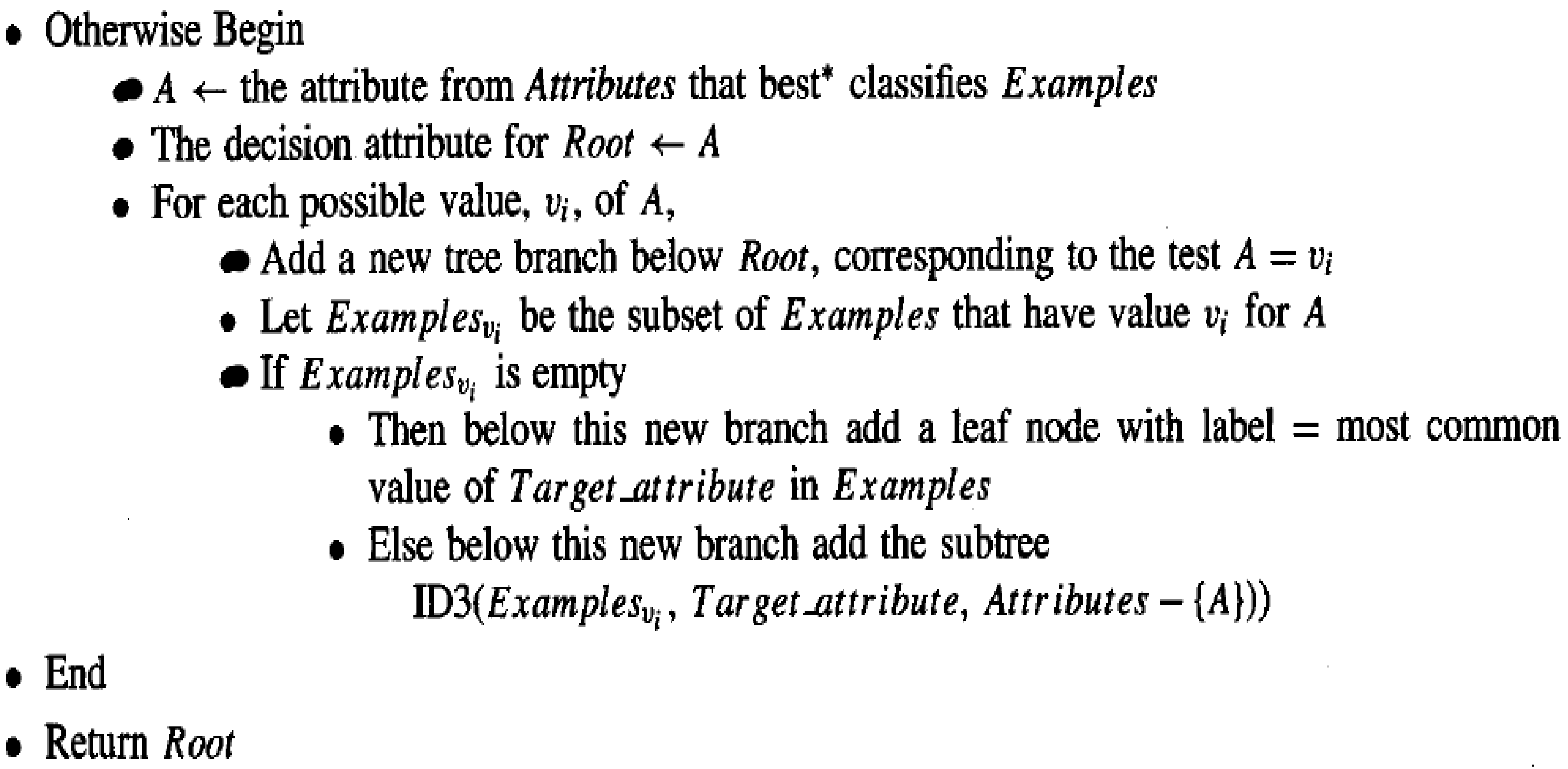


ID3(*Examples*, *Target_attribute*, *Attributes*)

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*

THE BASIC DECISION TREE LEARNING ALGORITHM - ID3 algorithm



```
• 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 $Root$
```

HYPOTHESIS SPACE SEARCH IN DECISION TREE LEARNING

- 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 begins with **the empty tree,** then considers progressively more **elaborate hypotheses** in search of a decision tree that correctly classifies the training data.

- The evaluation functions that guides this simple-to-complex search is **the information gain measure**.

HYPOTHESIS SPACE SEARCH IN DECISION TREE LEARNING

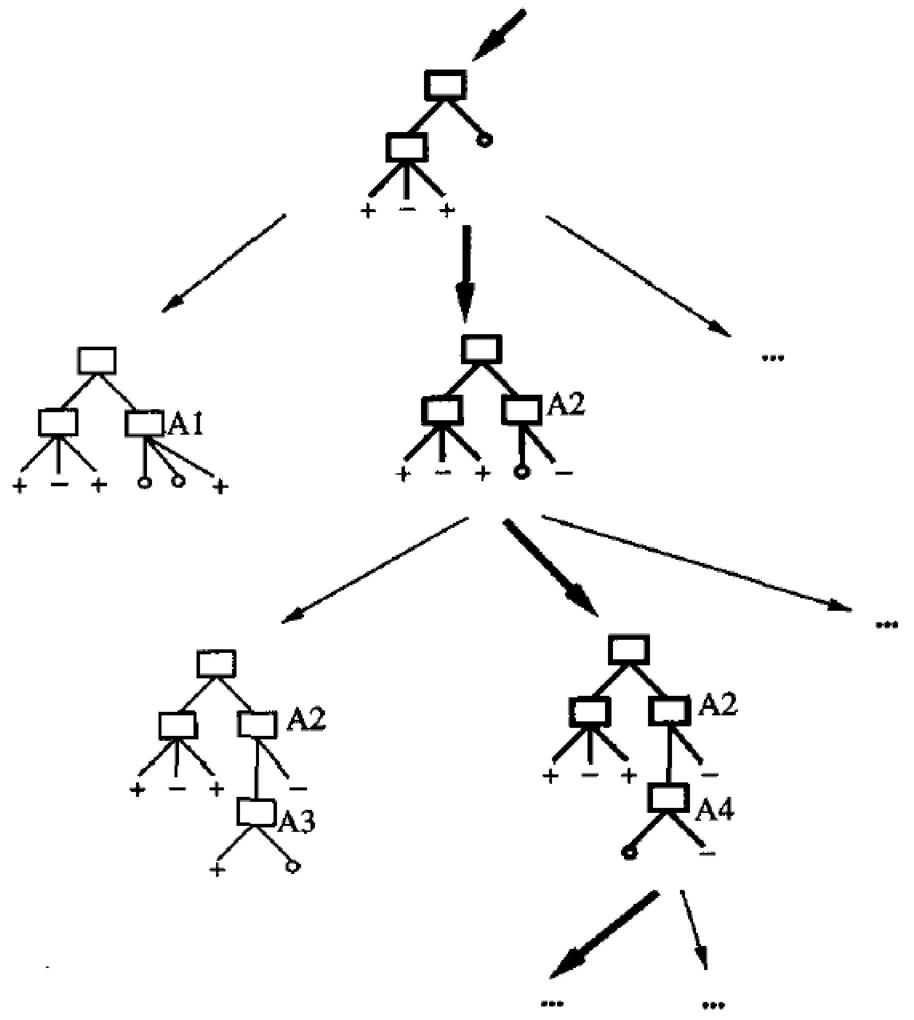


FIGURE 3.5
Hypothesis space search by ID3. ID3 searches through the space of possible decision trees from simplest to increasingly complex, guided by the information gain heuristic.

HYPOTHESIS SPACE SEARCH IN DECISION TREE LEARNING

By viewing ID3 in terms of its search space and search strategy, we can get some insight into its capabilities and limitations

1. ID3's hypothesis space of all decision trees **is a complete space of finite discrete-valued functions**, relative to the available attributes. So no major risks of methods that search incomplete hypothesis spaces
2. ID3 maintains **only a single current hypothesis** as it searches through the space of decision trees. This means it does not maintain the **set of all hypotheses consistent** with the available training examples.

HYPOTHESIS SPACE SEARCH IN DECISION TREE LEARNING

3. 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.** Thus, it has the risk of **converging to locally optimal solutions** that are not globally optimal.
4. ID3 uses all training examples at each step in the search to make statistically based decisions regarding how to refine its current hypothesis. Therefore, resulting search is **much less sensitive to errors in individual training examples.**

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
- Handling training data with missing attribute values
- Handling attributes with differing costs
- Improving computational efficiency.

ID3 has itself been extended to address most of these issues, with the resulting system renamed C4.5

ISSUES IN DECISION TREE LEARNING

Here we are looking at the following issues

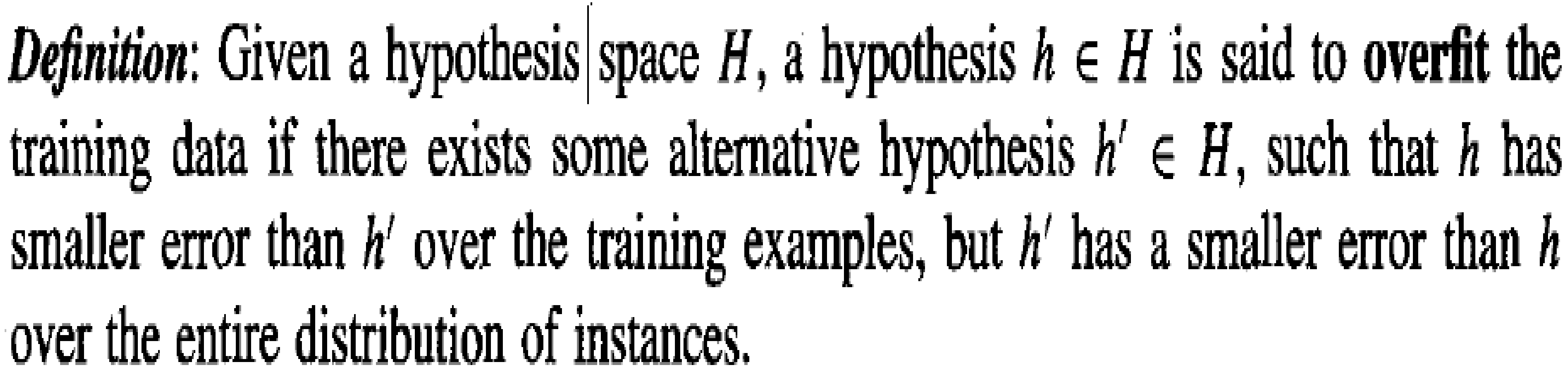
Issues in learning decision trees include

1. Avoiding Overfitting the Data
   - Reduced error pruning
   - Rule post-pruning
2. Incorporating Continuous-Valued Attributes
3. Alternative Measures for Selecting Attributes
4. Handling Training Examples with Missing Attribute Values
5. Handling Attributes with Differing Costs

ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data

- ID3 grows each branch just deeply enough to perfectly classify the training examples.
- In fact this strategy 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.

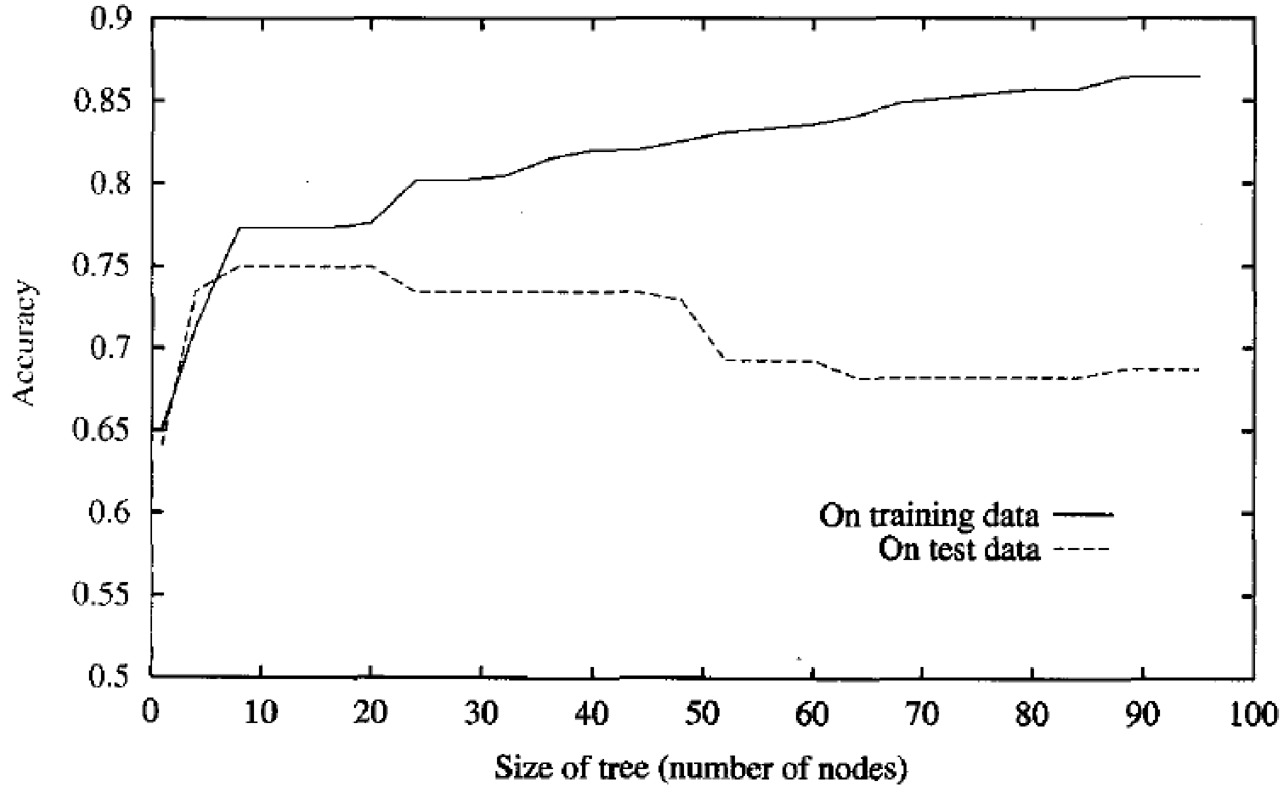
ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data



Definition: Given a hypothesis space H, a hypothesis $h \in H$ is said to **overfit** the training data if there exists some alternative hypothesis $h' \in H$, such that h has smaller error than h' over the training examples, but h' has a smaller error than h over the entire distribution of instances.

ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data

- **Figure 3.6** illustrates the impact of overfitting in a typical application of decision tree learning. In this case, the ID3 algorithm is applied to the task of learning which medical patients have a form of diabetes.



ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data

- 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 (not included in the training set).

ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data

- Predictably, 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

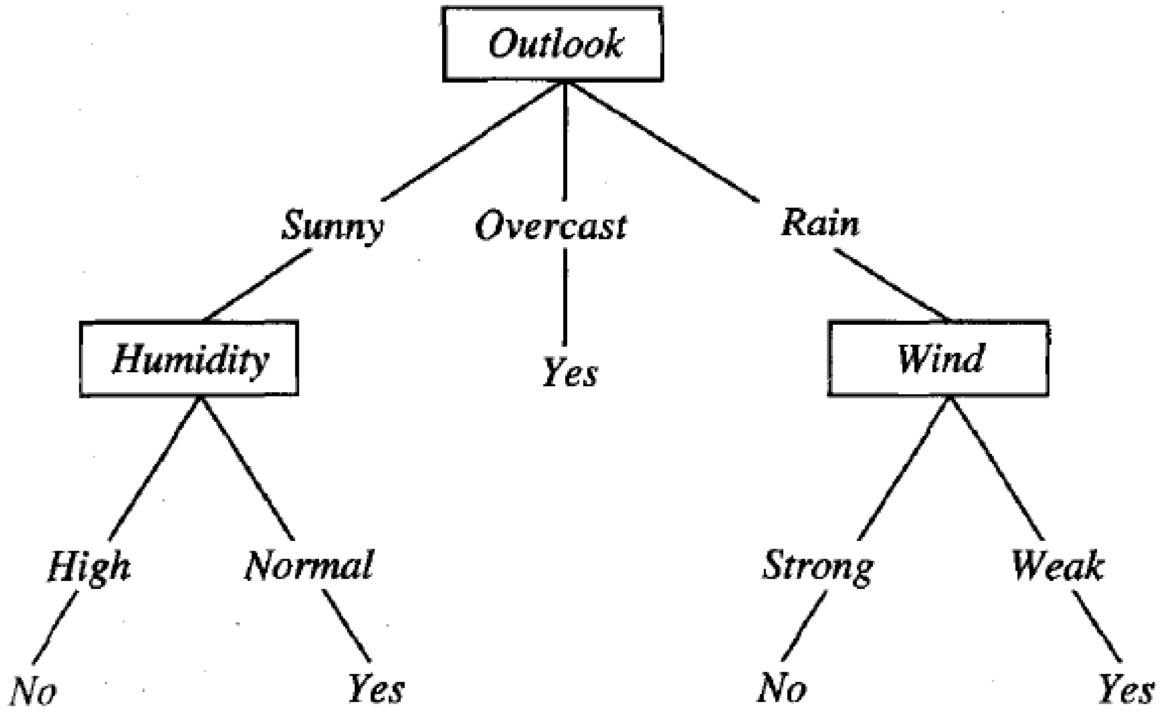
ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data

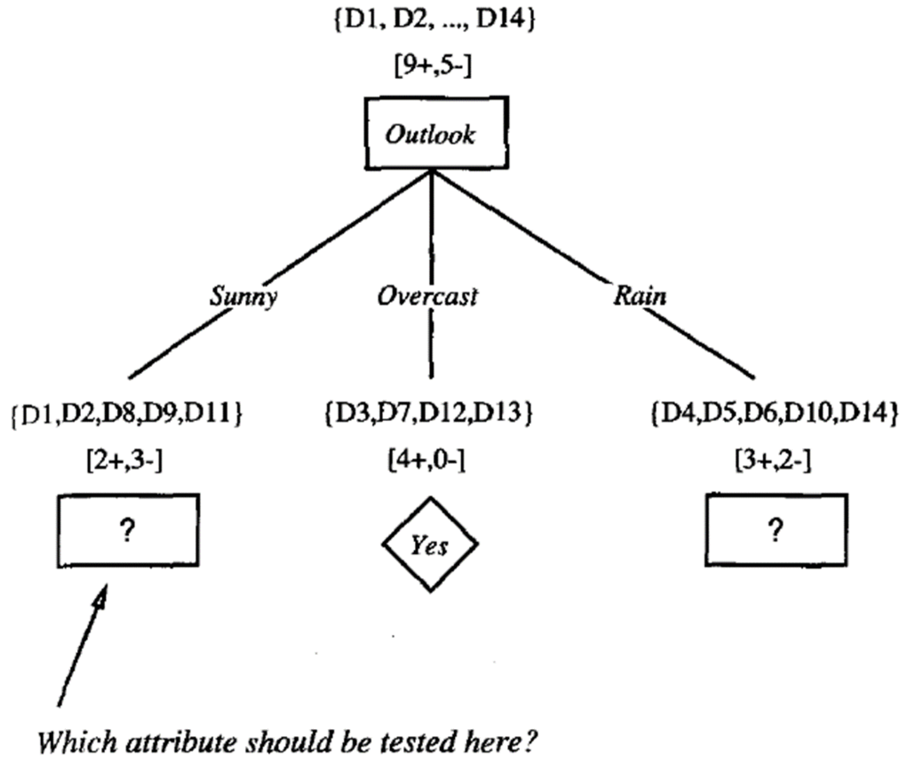
- One way for tree ***h*** to fit the training examples better than **h',** but for it to perform more poorly over subsequent examples is **random errors or noise in the data**.
- Consider the effect of adding the following positive training example, **incorrectly labeled as negative**, to the (**otherwise correct)** examples (D1 – D14)

$$\langle Outlook = Sunny, Temperature = Hot, Humidity = Normal,$$
$$Wind = Strong, PlayTennis = No\rangle$$

ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data

- If this row was added without error-free (***PlayTennis=Yes***), we would have obtained the tree shown below (as in **fig 3.1**)





ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data

- If this row is added with error (***PlayTennis=No***), the new example will be sorted into the second leaf node(**under Humidity**) from the left in the tree along with the previous positive examples **D9 and D11.**
- Because the new example is labeled as a **negative example**, ID3 will search for further refinements to the tree below this node (as there will 2+ and 1- examples).
- The result is that ID3 will output a **decision tree (h)** that is more complex than the original tree from **Figure 3.1 (h').**

ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data

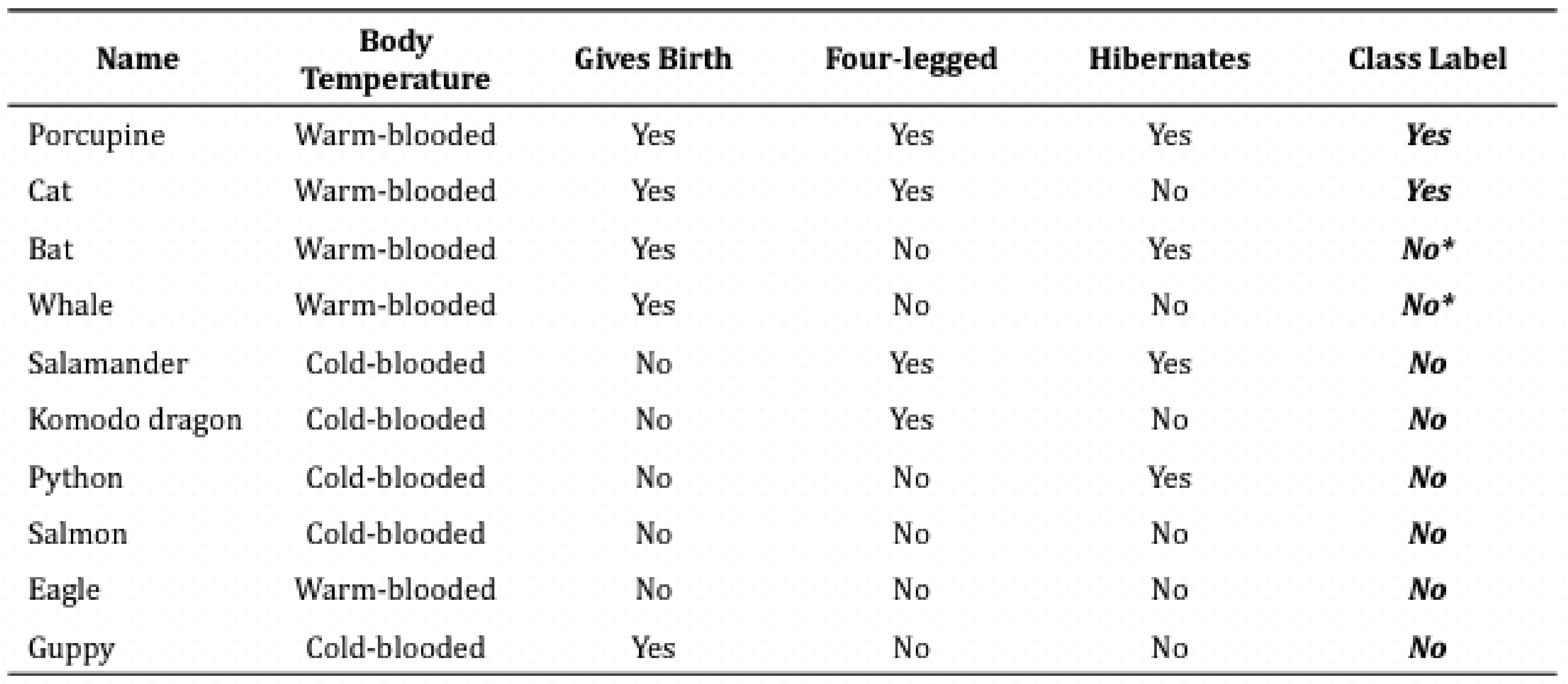
- Of course ***h will fit*** the collection of training examples perfectly, whereas the simpler ***h' will not***.
- Thus **we expect h** to **outperform h'** over subsequent data drawn from the same instance distribution.
- In fact, overfitting is possible even **when the training data are noise-free,** especially when small numbers of examples are associated with leaf nodes.

ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data

Overfitting Due to Noise: An Example

training set for classifying mammals.

An example training set for classifying mammals. Asterisks denote mislabelings.

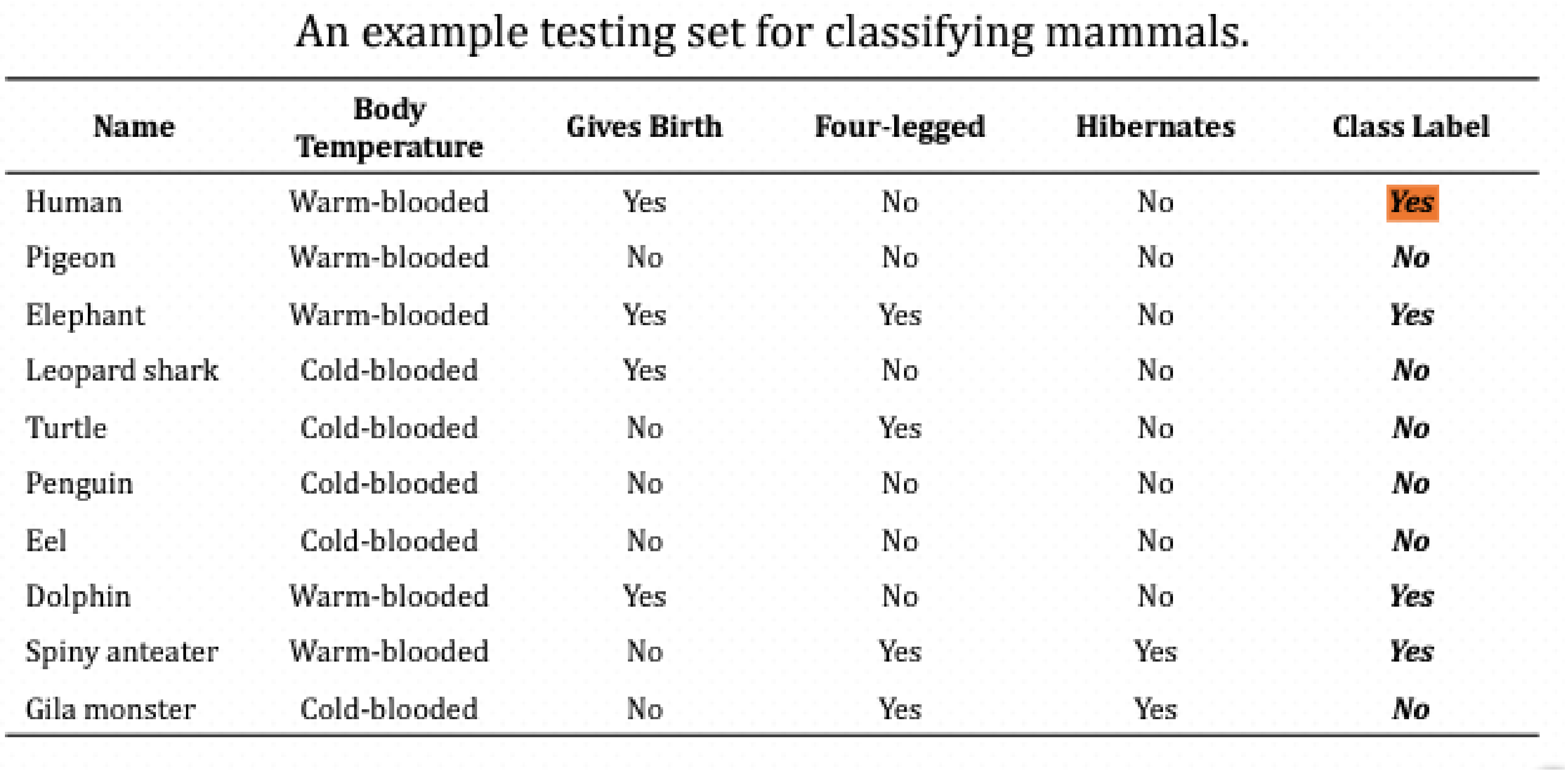


Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Class Label
Porcupine	Warm-blooded	Yes	Yes	Yes	**_Yes_**
Cat	Warm-blooded	Yes	Yes	No	**_Yes_**
Bat	Warm-blooded	Yes	No	Yes	**_No*_**
Whale	Warm-blooded	Yes	No	No	**_No*_**
Salamander	Cold-blooded	No	Yes	Yes	**_No_**
Komodo dragon	Cold-blooded	No	Yes	No	**_No_**
Python	Cold-blooded	No	No	Yes	**_No_**
Salmon	Cold-blooded	No	No	No	**_No_**
Eagle	Warm-blooded	No	No	No	**_No_**
Guppy	Cold-blooded	Yes	No	No	**_No_**

ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data

Overfitting Due to Noise: An Example

testing set for classifying mammals.

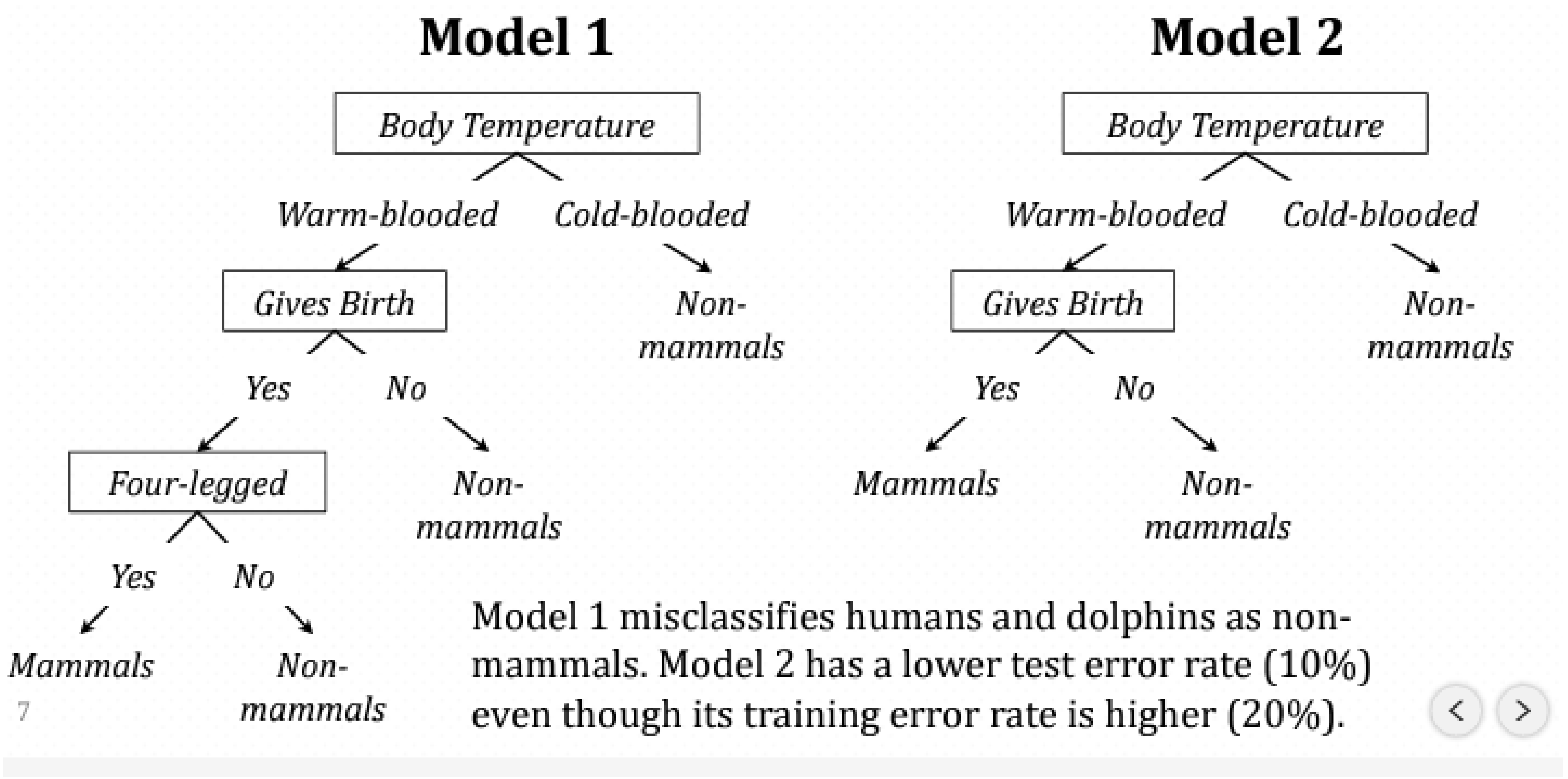


An example testing set for classifying mammals.

Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Class Label
Human	Warm-blooded	Yes	No	No	***Yes***
Pigeon	Warm-blooded	No	No	No	***No***
Elephant	Warm-blooded	Yes	Yes	No	***Yes***
Leopard shark	Cold-blooded	Yes	No	No	***No***
Turtle	Cold-blooded	No	Yes	No	***No***
Penguin	Cold-blooded	No	No	No	***No***
Eel	Cold-blooded	No	No	No	***No***
Dolphin	Warm-blooded	Yes	No	No	***Yes***
Spiny anteater	Warm-blooded	No	Yes	Yes	***Yes***
Gila monster	Cold-blooded	No	Yes	Yes	***No***

ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data

Overfitting Due to Noise: An Example



Model 1 misclassifies humans and dolphins as non-mammals. Model 2 has a lower test error rate (10%) even though its training error rate is higher (20%).

ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data

Overfitting Due to Noise: An Example

- Model 1 (tree with overfitting) classifies all training samples correctly. However, it misclassifies test samples such as human and dolphin.
- Model2 represents pruned tree., which correctly classifies test samples.

ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data

- Overfitting is a **significant practical difficulty** for decision tree learning and many other learning methods.

- There are several approaches to avoiding overfitting in decision tree learning. **These can be grouped into two classes:**
  1. **Approaches that stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data,**
  2. **Approaches that allow the tree to overfit the data, and then post-prune the tree.**

ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data

- The difficulty in the first approach is estimating precisely when to stop growing the tree.
- Thus, post-pruning overfit trees has been found to be more successful.
- Whether it is first or second approach, a key question is what criterion is to be used to determine the correct final tree size.

ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data

- Approaches include:

1. Use a **separate set of examples**, distinct from the training examples, to evaluate the utility of post-pruning nodes from the tree.
2. 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.
3. 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**

ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data

- The first approach is the most common and is often referred to as a training and validation set approach.
- In this approach, 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 (to evaluate the impact of pruning this hypothesis.).
- The **validation set** can be expected to provide **a safety check against overfitting the spurious characteristics** of the training set.

ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data

- Of course, it is important that **the validation set be large enough** to itself provide a statistically significant sample of the instances.

- One common heuristic is to withhold **one-third of the available examples for the validation set**, using the other two-thirds for training..

ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data

REDUCED ERROR PRUNING:

- To prevent overfitting, this approach 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.

ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data

REDUCED ERROR PRUNING:

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.

ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data

REDUCED ERROR PRUNING:

- The impact of reduced-error pruning on the accuracy of the decision tree is illustrated in Figure 3.7.

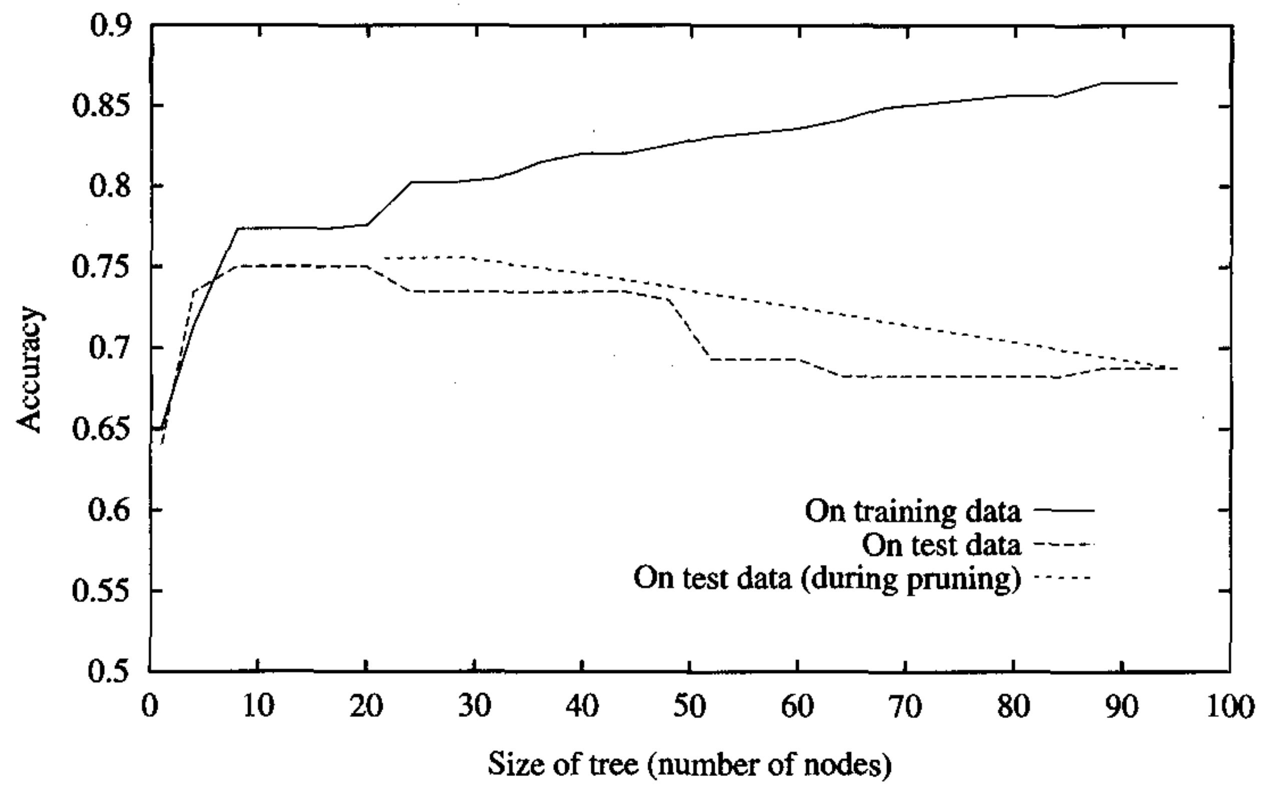


FIGURE 3.7
Effect of reduced-error pruning in decision tree learning. This plot shows the same curves of training and test set accuracy as in Figure 3.6. In addition, it shows the impact of reduced error pruning of the tree produced by ID3. Notice the increase in accuracy over the test set as nodes are pruned from the tree. Here, the validation set used for pruning is distinct from both the training and test sets.

ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data

REDUCED ERROR PRUNING:

- 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
    - A set of test examples used to provide an unbiased estimate of accuracy over future unseen examples.

ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data

REDUCED ERROR PRUNING:

- Using a separate set of data to guide pruning is an effective approach provided **a large amount of data is available.**
- The **major drawback o**f this approach is that when data is limited, withholding part of it for the validation set reduces even further the number of examples available for training.

ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data

RULE POST-PRUNING

- Rule post-pruning is one quite successful method for finding high accuracy hypotheses.
- 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 much 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.**

ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data

RULE POST-PRUNING

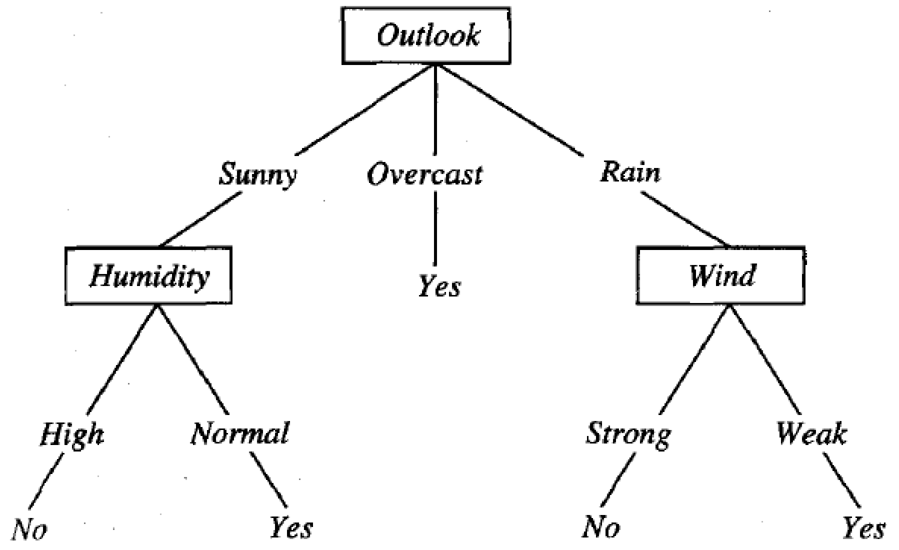
- Rule post-pruning involves the following steps:
  3. Prune (generalize) each rule by removing any preconditions that result in improving its estimated accuracy.
  4. Sort the pruned rules by their estimated accuracy, and consider them in this sequence when classifying subsequent instances

ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data

RULE POST-PRUNING – Illustration

- consider again the decision tree in Figure 3.1

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).**



ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data

RULE POST-PRUNING – Illustration

- For example, the leftmost path of the tree in Figure 3.1 is translated into the rule.

$$\begin{array}{ll} \text{IF} & (Outlook = Sunny) \wedge (Humidity = High) \\ \text{THEN} & PlayTennis = No \end{array}$$

- Next, each such rule is pruned by removing any antecedent, or precondition, whose removal does not worsen its estimated accuracy.

ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data

RULE POST-PRUNING – Illustration

- It would select whichever of these .runing steps produced the greatest improvement in estimated rule accuracy, then consider pruning the second precondition as a further pruning step.
- No pruning step is performed if it reduces the estimated rule accuracy.
- As noted above, one method to estimate rule accuracy is to use a validation set of examples disjoint from the training set

ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data

RULE POST-PRUNING – Illustration

- As noted above, one method to estimate rule accuracy is to use a validation set of examples disjoint from the training set.
- Another method, used by C4.5, is to evaluate performance based on the training set itself, using **a pessimistic estimate** to make up for the fact that the training data gives an estimate biased in favor of the rules

ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data

RULE POST-PRUNING – Why convert the decision tree to rules before pruning?

There are three main advantages.

- Converting to rules allows distinguishing among the different contexts in which a decision node is used. Because each distinct path through the decision tree node produces a distinct rule, the pruning decision regarding that attribute test can be made differently for each path.

ISSUES IN DECISION TREE LEARNING - Avoiding Overfitting the Data

RULE POST-PRUNING – Why convert the decision tree to rules before pruning?

There are three main advantages.

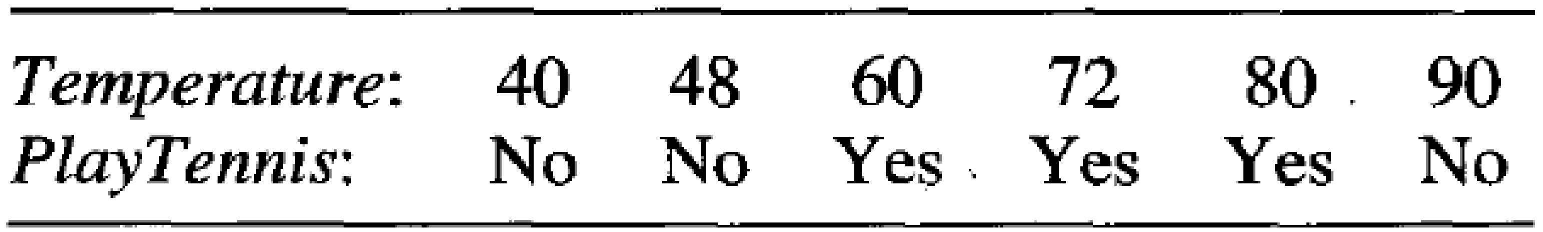
- Converting to rules removes the distinction between attribute tests that occur near the root of the tree and those that occur near the leaves. Thus, we avoid messy bookkeeping issues such as how to reorganize the tree if the root node is pruned while retaining part of the subtree below this test.
- Converting to rules improves readability. Rules are often easier for to understand.

ISSUES IN DECISION TREE LEARNING - Incorporating Continuous-Valued Attributes

- Our initial definition of ID3 is restricted to attributes that take on a discrete set of values. Here the attributes tested in the decision nodes of the tree are discrete valued
- We incorporate continuous-valued decision attributes into the learned tree.
- 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.

ISSUES IN DECISION TREE LEARNING - Incorporating Continuous-Valued Attributes

- As an example, suppose we wish to include the continuous-valued attribute ***Temperature*** in describing the training example days.
- Suppose further that the training examples associated with a particular node in the decision tree have the following values for Temperature and the target attribute ***PlayTennis.***



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

ISSUES IN DECISION TREE LEARNING - Incorporating Continuous-Valued Attributes

- Clearly, we would like to **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.

ISSUES IN DECISION TREE LEARNING - Incorporating Continuous-Valued Attributes

- 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** , then the best can be selected (Temperature > 54)
- This is dynamically created boolean attribute can then compete with the other discrete-valued candidate attributes available for growing the decision tree.

ISSUES IN DECISION TREE LEARNING - 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**.
- For example, **Date attribute**, which has a very large number of possible values (e.g., March 4, 1979), if added to the data in Table 3.2, it would have the **highest information gain of any of the attributes.**
- Thus, it would be selected as the decision attribute for the root node of the tree and lead to a tree of depth one, which perfectly classifies the training data.

ISSUES IN DECISION TREE LEARNING - Alternative Measures for Selecting Attributes

- However, it would be a very poor predictor of the target function over unseen instances (Overfitting).
- 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:

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

ISSUES IN DECISION TREE LEARNING - Alternative Measures for Selecting Attributes

- S_1 through S_c , are the c subsets of examples resulting from partitioning S by the c-valued attribute A.
- Note that Split information 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**

ISSUES IN DECISION TREE LEARNING - Alternative Measures for Selecting Attributes

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

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

- Notice that the **Splitinformation** term discourages the selection of attributes with many uniformly distributed values.

ISSUES IN DECISION TREE LEARNING - Alternative Measures for Selecting Attributes

- For example, consider a collection of n examples that are completely separated by attribute A (e.g., Date).
- In this case, the S**plitinformation** value will be $\log_2 n$. In contrast, a boolean attribute B that splits the same n examples exactly in half will have **SplitInformation** of 1.
- If attributes A and B produce the same information gain, **then clearly B will score higher according to the Gain Ratio measure**.

ISSUES IN DECISION TREE LEARNING - Alternative Measures for Selecting Attributes

- One practical issue that arises in using **GainRatio the denominator can be zero or very small.**
- This either makes the GainRatio undefined or very large for attributes that happen to have the same value for nearly all members of S.
- 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.

ISSUES IN DECISION TREE LEARNING - Alternative Measures for Selecting Attributes

- An alternative to the GainRatio, **is a distance-based measure introduced** by Lopez de Mantaras (1991).
- This measure is based on defining **a distance metric between partitions of the data**.
- **Each attribute** is evaluated based on the distance between the **data partition it creates and the perfect partition** (i.e., the partition that perfectly classifies the training data).
- The attribute whose partition is closest to the perfect partition is chosen.

ISSUES IN DECISION TREE LEARNING –

Handling Training Examples with Missing Attribute Values

- 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).
- 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).

ISSUES IN DECISION TREE LEARNING –

Handling Training Examples with Missing Attribute Values

- 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.

ISSUES IN DECISION TREE LEARNING –

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 in 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

ISSUES IN DECISION TREE LEARNING –

Handling Attributes with Differing Costs

- ID3 can be modified to take into account attribute costs by introducing **a cost term into the attribute selection measure**.
- For example, we might 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

ISSUES IN DECISION TREE LEARNING –

Handling Attributes with Differing Costs

- **Tan and Schlimmer (1990) and Tan (1993)** describe one such approach in which Attribute cost is measured by **the number of seconds required to obtain the attribute value**
- Then replacing the information gain attribute selection measure by the following measure

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

ISSUES IN DECISION TREE LEARNING –

Handling Attributes with Differing Costs

- Nunez (1988) describes a related approach and its application to **learning medical diagnosis rules**.
- Here the attributes are different symptoms and laboratory tests with differing costs.
- His system uses a somewhat different attribute selection measure

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

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

K-Nearest Neighbour

Topics

- K-NN rule
- Measuring Similarity
- Irrelevant attributes
- scaling problems

K-NN rule

- K-Nearest Neighbour approach to classification to determine the class of object x, find the training example most similar to it. Then label x with this example's class.
- To establish that an object is more similar to x than to y, Too many arbitrary and subjective factors are involved in answering them.

K-NN rule

- The first row in Table 3.1 gives the attribute values of object x. What is its class?

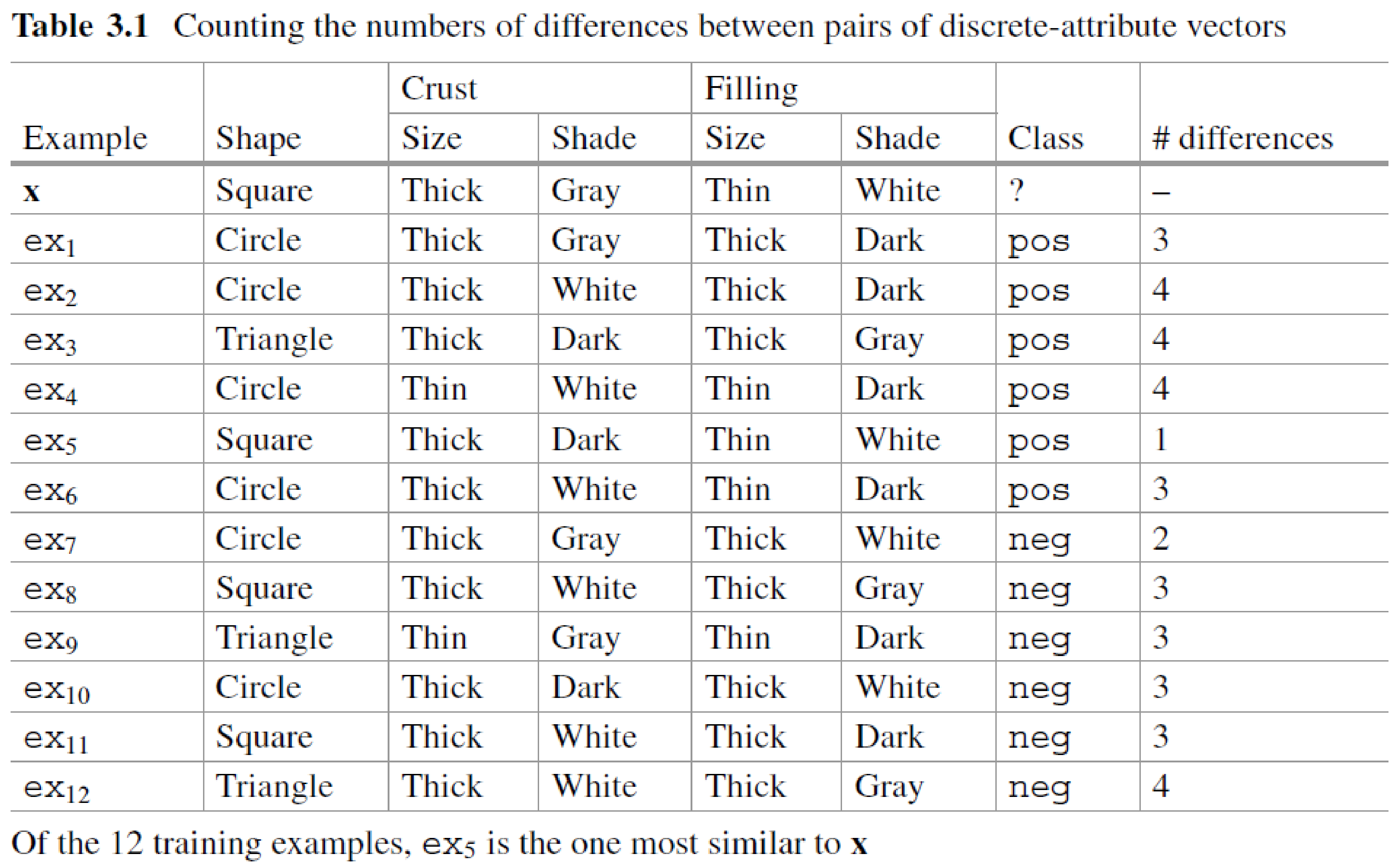


Table 3.1 Counting the numbers of differences between pairs of discrete-attribute vectors

		Crust		Filling			
Example	Shape	Size	Shade	Size	Shade	Class	# differences
x	Square	Thick	Gray	Thin	White	?	–
ex_1	Circle	Thick	Gray	Thick	Dark	pos	3
ex_2	Circle	Thick	White	Thick	Dark	pos	4
ex_3	Triangle	Thick	Dark	Thick	Gray	pos	4
ex_4	Circle	Thin	White	Thin	Dark	pos	4
ex_5	Square	Thick	Dark	Thin	White	pos	1
ex_6	Circle	Thick	White	Thin	Dark	pos	3
ex_7	Circle	Thick	Gray	Thick	White	neg	2
ex_8	Square	Thick	White	Thick	Gray	neg	3
ex_9	Triangle	Thin	Gray	Thin	Dark	neg	3
ex_{10}	Circle	Thick	Dark	Thick	White	neg	3
ex_{11}	Square	Thick	White	Thick	Dark	neg	3
ex_{12}	Triangle	Thick	White	Thick	Gray	neg	4

Of the 12 training examples, ex_5 is the one most similar to **x**

Of the 12 training examples, ex_5 is the one most similar to **x.**
This suggests that we should label x with pos, the class of ex5.

K-NN rule

Dealing with continuous attributes is just as simple

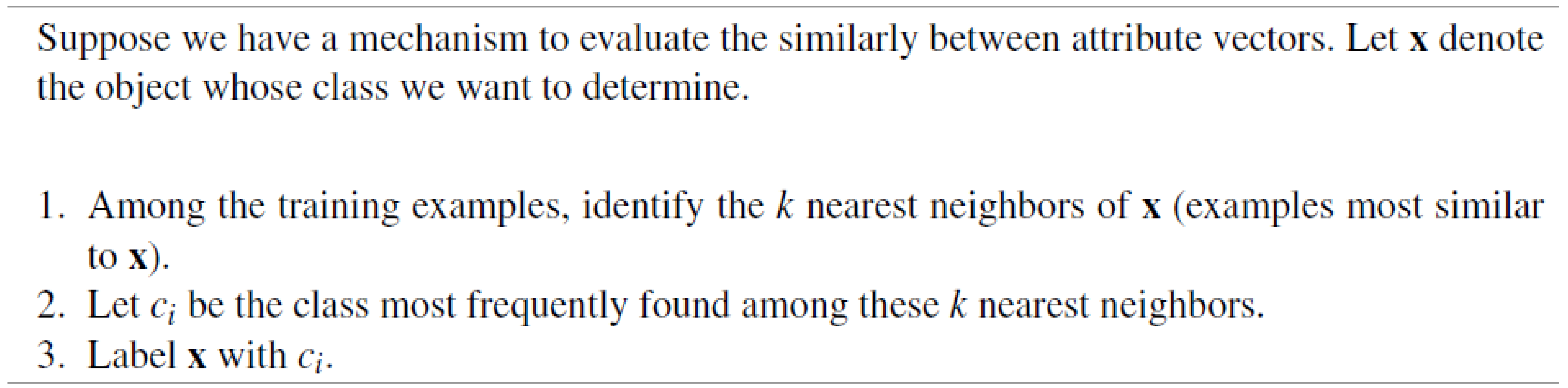
- The fact that each example can be represented by a point in an n-dimensional space makes it possible to calculate the **geometric distance between any pair of examples**, for instance, by the **Euclidean distance**
- The training example with the smallest distance from x in the instance space is, geometrically speaking, **x's nearest neighbor**.

K-NN rule

From a Single Neighbor to k Neighbors

- k-NN classifier is a more robust approach identifies not one, but several nearest neighbors, and then lets them vote.
- Here k is the number of the voting neighbors.

Table 3.2 The simplest version of the *k*-NN classifier



Suppose we have a mechanism to evaluate the similarly between attribute vectors. Let **x** denote the object whose class we want to determine.

1. Among the training examples, identify the *k* nearest neighbors of **x** (examples most similar to **x**).
2. Let c_i be the class most frequently found among these *k* nearest neighbors.
3. Label **x** with c_i.

K-NN rule

What should be the value of k?

- Note that, in **a two-class domain**, k should be an odd number so as to prevent ties.
- For instance, a 4-NN classifier might face a situation where the **number of positive neighbors is the same as the number of negative neighbors**.
- This will not happen to a 5-NN classifier.

K-NN rule

What should be the value of k?

- As for domains that have more than two classes, using an odd number of nearest neighbors does not prevent ties.
- For instance, the 7-NN classifier can realize that three neighbors belong to class C1, three neighbors belong to class C2, and one neighbor belongs to class C3.
- The engineer designing the classifier then needs to define a mechanism to choose between C1 and C2.

K-NN rule

An Illustration

- Figure 3.1 shows several positive and negative training examples, and also some objects (the big black dots) whose classes the k-NN classifier is to determine.

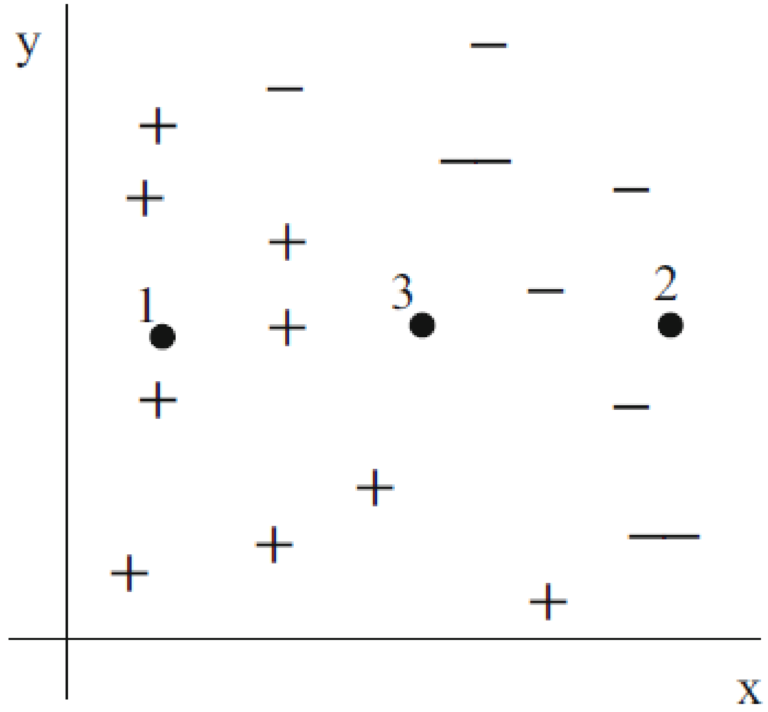


Fig. 3.1 Object **3**, finding itself in the borderline region, is hard to classify.

K-NN rule

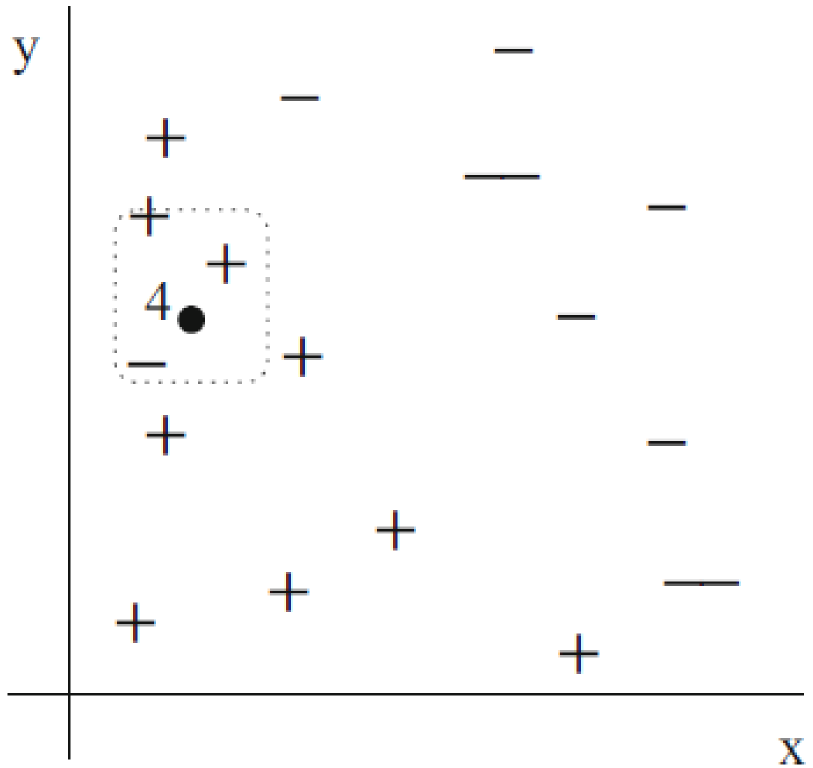
An Illustration

- Objects 1 and 2 are surrounded by examples from the same class, and their classification is therefore straightforward.
- On the other hand, object 3 is located in the “no man’s land” between the positive and negative regions. So a small amount of attribute noise can send it to either side

K-NN rule

An Illustration

- In this figure object 4 finds itself deep in the positive region, but class noise has mislabeled its nearest neighbor in the training set as negative.
- Whereas the 1-NN classifier will go wrong, here, the 3-NN classifier will give the correct answer because the other two neighbors, which are positive, will outvote the single negative neighbor.



The 1-NN classifier will misclassify object 4, but the mistake is corrected if the 3-NN classifier is used

Measuring Similarity

- A natural way to find the nearest neighbors of object x is to compare the geometrical distances of the individual training examples from x.

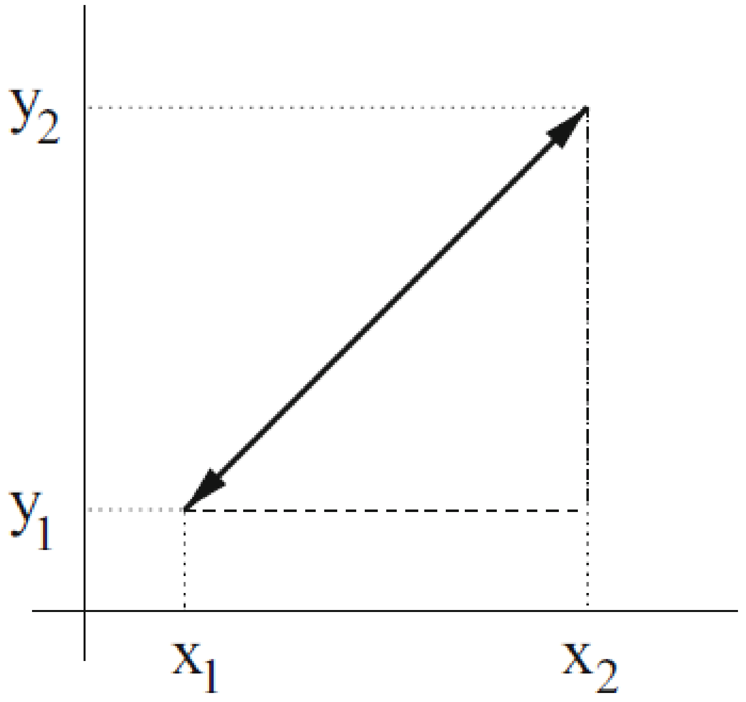


Fig. 3.2 The Euclidean distance between two points in a two-dimensional space is equal to the length of the triangle's hypotenuse

Measuring Similarity

- **Euclidean Distance** In a plane, the geometric distance between two points, $\mathbf{x} = (x_1, x_2)$ **and** $\mathbf{y} = (y_1, y_2)$ is obtained with the help of the pythagorean theorem as indicated in Fig. 3.2:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}.$$

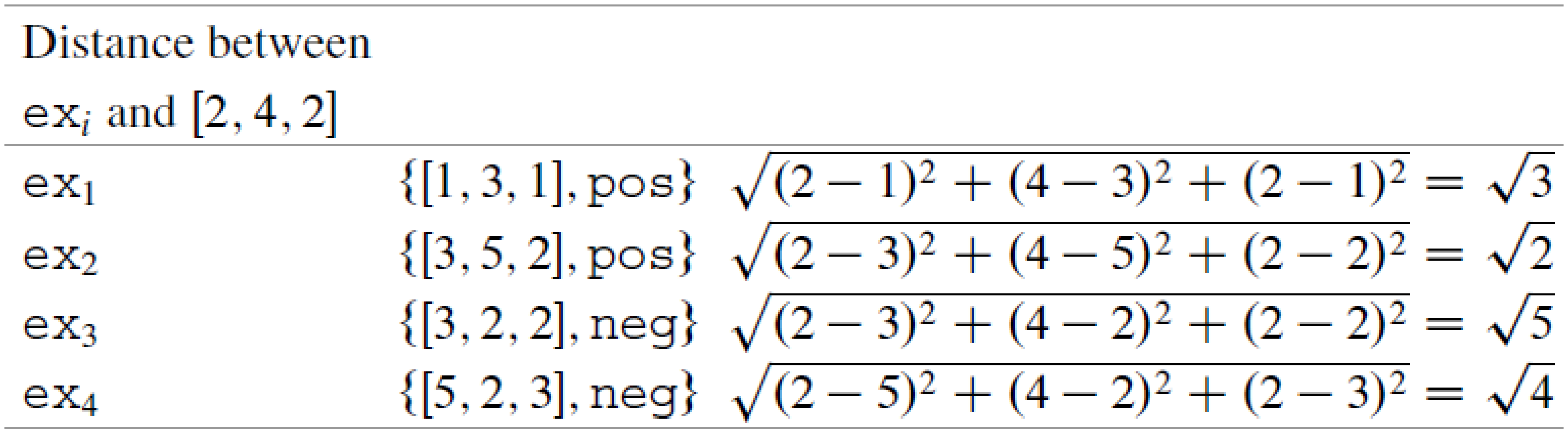
Measuring Similarity

- This formula is easy to generalize to a domain with n continuous attributes where the Euclidean distance between $(x_1, x_2, \dots, x_n)$ and $(y_1, y_2, \dots, y_n)$ is defined as follows:

$$d_E(\mathbf{x}, \mathbf{y}) = \sqrt{\Sigma_{i=1}^{n}(x_i - y_i)^2}$$

Measuring Similarity

- The way this metric is used in the context of k-NN classifiers is illustrated in Table 3.3



Distance between ex_i and $[2, 4, 2]$		
ex_1	$\{[1,3,1], \text{pos}\}$	$\sqrt{(2-1)^2+(4-3)^2+(2-1)^2} = \sqrt{3}$
ex_2	$\{[3,5,2], \text{pos}\}$	$\sqrt{(2-3)^2+(4-5)^2+(2-2)^2} = \sqrt{2}$
ex_3	$\{[3,2,2], \text{neg}\}$	$\sqrt{(2-3)^2+(4-2)^2+(2-2)^2} = \sqrt{5}$
ex_4	$\{[5,2,3], \text{neg}\}$	$\sqrt{(2-5)^2+(4-2)^2+(2-3)^2} = \sqrt{4}$

Table 3.3 Using the nearest-neighbor principle in a 3-dimensional Euclidean space

Using the following training set of four examples described by three numeric attributes, determine the class of object **x = [2, 4,** 2]

Measuring Similarity

- Note that Calculating the Euclidean distances between x and the training examples, we realize that x’s nearest neighbor is ex2. Its label being pos, the 1-NN classifier returns the positive label.
- The same result is obtained by the 3-NN classifier because two of x’s three nearest neighbors (ex1 and ex2) are positive, and only one (ex4), is negative.

Measuring Similarity

A More General Formulation :

- In domains where the examples are described by a mixture of discrete and continuous attributes

$$d_M(\mathbf{x}, \mathbf{y}) = \sqrt{\Sigma_{i=1}^{n} d(x_i, y_i)}$$

- For instance, we can use $d(x_i, y_i) = (x_i - y_i)^2$ for continuous attributes, whereas for discrete attributes, we put $d(x_i, y_i) = 0$ if $x_i = y_i$ and $d(x_i, y_i) = 1$ if $x_i \neq y_i$.

Measuring Similarity

A More General Formulation :

- Note that if all attributes are continuous, the formula is identical to Euclidean distance;
- If the attributes are all discrete, the formula simply specifies the number of attributes in which the two vectors differ.

Measuring Similarity

A More General Formulation :

- In purely Boolean domains, where for any attribute only the values true or false are permitted, this case is called **Hamming distance, $\boldsymbol{d_H}$.**
- For instance, the Hamming distance between the vectors

  $x = (t, t, f, f)$and $y = (t, f, t, f)$ is $d_H = 2$.

Measuring Similarity

Attribute-to-Attribute Distances Can Be Misleading:

- We must be careful not to apply the general formula mechanically, ignoring the specific aspects of the given domain.
- Suppose our examples are described by three attributes, **size, price, and season.**
- Of these, the first two are obviously **continuous, and the last, discrete.**

Measuring Similarity

Attribute-to-Attribute Distances Can Be Misleading:

- If $x = (2,1.5, summer)$ and $y = (1, 0.5, winter)$; then Equation gives the following distance between the two:

$$d_M(\mathbf{x}, \mathbf{y}) = \sqrt{(2-1)^2 + (1.5-0.5)^2 + 1} = \sqrt{3}$$

- We see that $d(summer, winter) = 1$, however in reality summer and winter are not neighboring seasons. Also in reality , **spring and fall** are more similar to each other than **summer and winter. We can see that the two values, 0 and 1, will clearly not suffice, here.**

Measuring Similarity

Attribute-to-Attribute Distances Can Be Misleading:

- Mixing continuous and discrete attributes can be risky in another way.
- For example , the difference between two sizes say **size1 = 1 and size2= 12, which means that d(size1, size2) = 121,** which can totally dominate the **difference between two seasons.**
- This observation is closely related to the problem of **scaling**

Measuring Similarity

Distances in General

- There are quite a few other formulas for determining similarities. Suffice it so say that any distance metric has to satisfy the following requirements:

1. the distance must never be negative;
2. the distance between two identical vectors, **x** and **y**, is zero;
3. the distance from **x** to **y** is the same as the distance from **y** to **x**;
4. the metric must satisfy the triangular inequality: $d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}) \geq d(\mathbf{x}, \mathbf{z})$.

Irrelevant Attributes and Scaling Problems

- In machine learning some attributes are irrelevant in the sense that their values have nothing to do with the given example’s class.
- But they do affect the geometric distance between vectors.

Irrelevant Attributes and Scaling Problems

- In the training set from Fig. 3.3, the examples are characterized by two numeric attributes: body-temperature (the horizontal axis) and shoe-size (the vertical axis). The black dot stands for the object that the k-NN classifier is expected to label as healthy (pos) or sick (neg).

Fig. 3.3 The "vertical" attribute is irrelevant for classification, and yet it affects the geometric distances between examples. Object **1** is positive, even though its nearest neighbor in the 2-dimensional space is negative

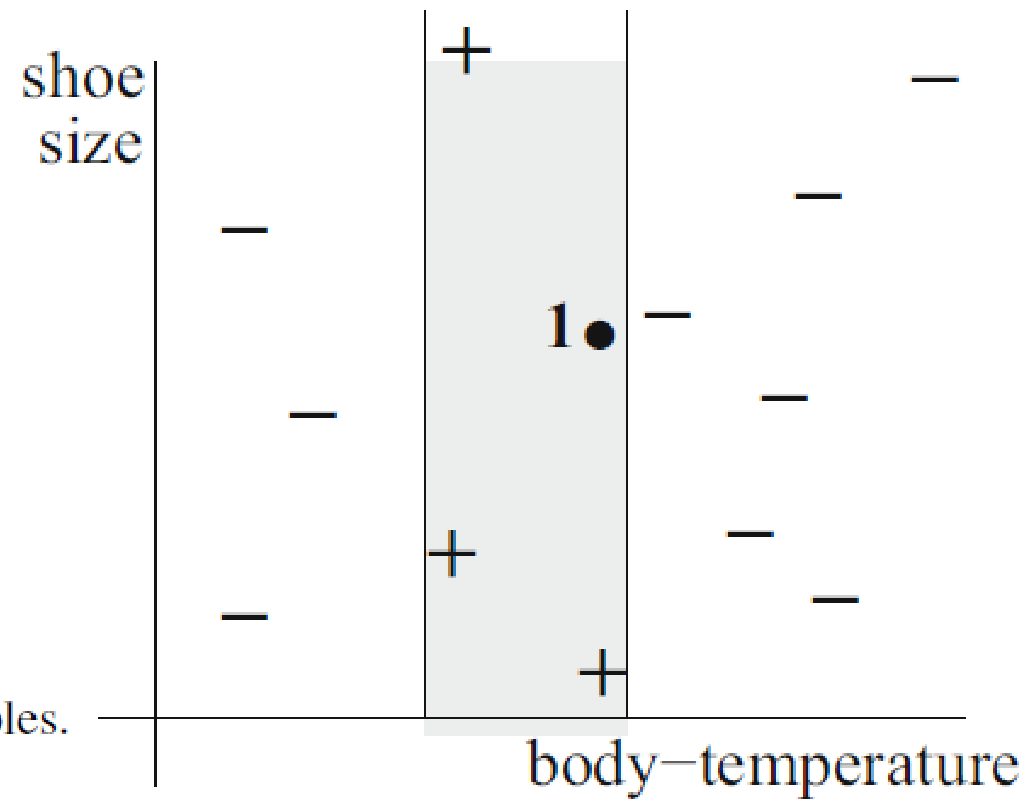


Fig. 3.3

Irrelevant Attributes and Scaling Problems

- As you can see, all positive examples find themselves in the shaded area delimited by two critical points along the “horizontal” attribute: temperatures exceeding the maximum indicate fever; those below the minimum, hypothermia.
- Show-size is being unable to betray anything about a person’s health.
- Common sense requires that it should be labeled as positive—despite the fact that its nearest neighbor happens to be negative.

Irrelevant Attributes and Scaling Problems

- Thus if only the first attribute is used, the Euclidean distance between the two examples is

$$d_E(x, y) = \sqrt{(x_1 - y_1)^2} = |x_1 - y_1|.$$

- If both attributes are used, the Euclidean distance will be

$$d_E(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}.$$

- Even though second attribute is irrelevant, yet it affects, adversely, k-NN's notion of similarity

Irrelevant Attributes and Scaling Problems

- How much damage is caused by irrelevant attributes depends on how many of them are used to describe the examples.
- There is no need to panic, if there is one out of 100 examples
- If the vast majority of the attributes have nothing to do with the class we want to recognize, then the geometric distance will become almost meaningless, and the classifier’s performance will be dismal.

Irrelevant Attributes and Scaling Problems

The Scales of Attribute Values:

- Suppose we want to evaluate the similarity of two examples,

  $x = (t, 0.2, 254)$ and $y = (f\ , 0.1, 194)$,

- the first is **Boolean**, the second is continuous with values from interval **[0,1],** and the third is continuous with values from interval **[0,1000]**
- Using Eq. (3.2), calculate the distance between x and y, obtaining the following:

$$d_M(\mathbf{x}, \mathbf{y}) = \sqrt{(1-0)^2 + (0.2-0.1)^2 + (254-194)^2}$$

Irrelevant Attributes and Scaling Problems

The Scales of Attribute Values:

- We notice that the third attribute completely dominates, reducing the other two to virtual insignificance.
- The distance, $d_M(x, y)$ will hardly change no matter how we modify their values within their ranges.
- If we divide, all values of the third attribute by 1000, thus “squeezing” its range to [0,1], the impacts of the attributes will become more balanced.

Irrelevant Attributes and Scaling Problems

- We can see that the scales of the attribute values can radically affect the k-NN classifier’s behavior

Irrelevant Attributes and Scaling Problems

Another Aspect of Attribute Scaling:

- Consider the following two training examples, **ex1 and ex2**, and the object x whose class we want to determine.

$$\begin{aligned}\mathrm{ex}_1 &= [(10, 10), \mathrm{pos})]\\ \mathrm{ex}_2 &= [(20, 0), \mathrm{neg})]\\ \mathbf{x} &= (32, 20)\end{aligned}$$

- The distances are

$$d_M(\mathbf{x}, \mathrm{ex}_1) = \sqrt{584} \text{ and } d_M(\mathbf{x}, \mathrm{ex}_2) = \sqrt{544}.$$

- The latter being smaller, the 1-NN classifier will label x as neg.

Irrelevant Attributes and Scaling Problems

Another Aspect of Attribute Scaling:

- Suppose, however, that the second attribute expresses temperature, and does so in centigrades. If we decide to use Fahrenheits instead, the three vectors will change as follows

$$\begin{aligned} \text{ex}_1 &= [(10, 50), \text{pos})] \\ \text{ex}_2 &= [(20, 32), \text{neg})] \\ \mathbf{x} &= (32, 68) \end{aligned}$$

- Recalculating the distances,

$$d_M(\mathbf{x}, \text{ex}_1) = \sqrt{808} \text{ and } d_M(\mathbf{x}, \text{ex}_2) = \sqrt{1440}.$$

- It is the first distance that is smaller, and 1-NN will therefore classify x as positive.

Irrelevant Attributes and Scaling Problems

This seems a bit silly. The examples are still the same, except that we chose different units for temperature; and yet the classifier's verdict has changed.

Irrelevant Attributes and Scaling Problems

Normalizing Attribute Scales:

- One way out of this trouble is to re-scale them in a way that makes all values fall into the same interval, [0,1]
- Simplest is the one that first identifies, for the given attribute, its maximum (MAX) and minimum (MIN), and then replaces each value, x, of this attribute using the following formula:

$$x = \frac{x - MIN}{MAX - MIN}$$

Irrelevant Attributes and Scaling Problems

Normalizing Attribute Scales:

- Suppose that, in the training set consisting of five examples, a given attribute acquires the following values, respectively:

$$[7, 4, 25, -5, 10]$$

- We see that MIN = -5 and MAX = 25. Subtracting MIN from each of the values, we obtain the following:

$$[12, 9, 30, 0, 15]$$

Irrelevant Attributes and Scaling Problems

Normalizing Attribute Scales:

- The “new minimum” is 0, and the “new maximum” is

  MAX - MIN = 25 – (-5) = 30.

- Dividing the obtained values by MAX MIN, we obtain a situation where all the values fall into [0,1].

$$[0.4, 0.3, 1, 0, 0.5]$$

Weighted Nearest Neighbors (for general classification)

- The weighted k-NN classifier first finds the weight of each of the nearest neighbors.
- Computed weights are proportional to its distance from the object x(to be classified). The closer the neighbor, the greater its impact.
- The weighted k-NN classifier then sums up the weights of for each class and attaches to x the **class whose weight is maximum.**

Weighted Nearest Neighbors (for general classification)

- Suppose the k neighbors are ordered according to their distances, $d_1, d_2, \ldots, d_k$ from x so that d_1 is the smallest distance and d_k is the greatest distance.
- The weight of the i-th closest neighbor is calculated as follows:

$$w_i = \begin{cases} \frac{d_k - d_i}{d_k - d_1}, & d_k \neq d_1 \\ 1 & d_k = d_1 \end{cases}$$

Weighted Nearest Neighbors (for general classification)

- Obviously, the weights thus obtained will range from **0 for the most distant neighbor** to **1 for the closest one**.
- This means that the approach actually considers only k-1 neighbors (because w_k = 0).

Weighted Nearest Neighbors (for general classification)

- The task is to use the weighted 5-NN classifier to determine the class of object x (C1/C2/C3) .
- Let the distances between x and the five nearest neighbors be

$$d_1 = 1\ (C2)$$
$$d_2 = 3\ (C1)$$
$$d_3 = 4\ (C2)$$
$$d_4 = 5\ (C1)$$
$$d_5 = 8\ (C3)$$

Weighted Nearest Neighbors (for general classification)

- Since the minimum is $d_1 = 1$ and the maximum is $d_5 = 8$, the individual weights are calculated as follows:

$$w_i = \frac{d_5 - d_i}{d_5 - d_1} = \frac{8 - d_i}{8 - 1} = \frac{8 - d_i}{7}$$

$$w_1 = \frac{8-1}{7} = 1, w_2 = \frac{8-3}{7} = \frac{5}{7}, w_3 = \frac{8-4}{7} = \frac{4}{7}, w_4 = \frac{8-5}{7} = \frac{3}{7}, w_5 = \frac{8-8}{7} = 0.$$

$$\sum_{c1} = \frac{5}{7} + \frac{3}{7} = \frac{8}{7}, \qquad \sum_{c2} = 1 + \frac{4}{7} = \frac{11}{7}, \qquad \sum_{c3} = 0,$$

so x is classified as c2

Weighted Nearest Neighbors (for general classification)

Use the examples from Table 3.7 to classify object

y = [3,3,] with the 5-NN classifier.

x_1	1	1	1	2	3	3	3	4	5
x_2	1	2	4	3	0	2	5	4	3
class	+	−	−	+	+	+	−	−	−

Table 3.7

- **Will weighted 5-NN classifier change anything?**