

3.4 THE BASIC DECISION TREE LEARNING ALGORITHM

Most algorithms that have been developed for learning decision trees are variations on a core algorithm that employs a top-down, greedy search through the space of possible decision trees. This approach is exemplified by the ID3 algorithm (Quinlan 1986) and its successor C4.5 (Quinlan 1993), which form the primary focus of our discussion here. In this section we present the basic algorithm for decision tree learning, corresponding approximately to the ID3 algorithm. In Section 3.7 we consider a number of extensions to this basic algorithm, including extensions incorporated into C4.5 and other more recent algorithms for decision tree learning.

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 (i.e., down the branch corresponding to the example’s value for this attribute). 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. A simplified version of the algorithm, specialized to learning boolean-valued functions (i.e., concept learning), is described in Table 3.1.

3.4.1 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. ID3 uses this information gain measure to select among the candidate attributes at each step while growing the tree.

3.4.1.1 ENTROPY MEASURES HOMOGENEITY OF EXAMPLES

In order to define information gain precisely, we begin by defining a measure commonly 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

$$\text{Entropy}(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus} \quad (3.1)$$

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

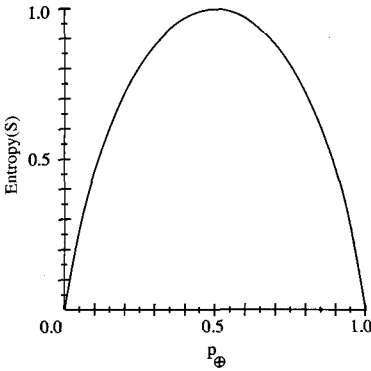
Summary of the ID3 algorithm specialized to learning boolean-valued functions. ID3 is a greedy algorithm that grows the tree top-down, at each node selecting the attribute that best classifies the local training examples. This process continues until the tree perfectly classifies the training examples,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

where p_{\oplus} is the proportion of positive examples in S and p_{\ominus} is the proportion of negative examples in S . In all calculations involving entropy we define $0 \log 0$ 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

$$\begin{aligned} Entropy([9+, 5-]) &= -(9/14) \log_2(9/14) - (5/14) \log_2(5/14) \\ &= 0.940 \end{aligned} \quad (3.2)$$

Notice that the entropy is 0 if all members of S belong to the same class. For example, if all members are positive ($p_{\oplus} = 1$), then p_{\ominus} is 0, and $Entropy(S) = -1 \cdot \log_2(1) - 0 \cdot \log_2 0 = -1 \cdot 0 - 0 \cdot \log_2 0 = 0$. Note 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

**FIGURE 3.2**

The entropy function relative to a boolean classification, as the proportion, p_+ , of positive examples varies between 0 and 1.

entropy is between 0 and 1. Figure 3.2 shows the form of the entropy function relative to a boolean classification, as p_+ 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 (i.e., a member of S drawn at random with uniform probability). 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.

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

$$\text{Entropy}(S) \equiv \sum_{i=1}^c -p_i \log_2 p_i \quad (3.3)$$

where p_i 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 $\log_2 c$.

3.4.1.2 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, $\text{Gain}(S, A)$ of an attribute A ,

relative to a collection of examples S , is defined as

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \quad (3.4)$$

where $Values(A)$ is the set of all possible values for attribute A , and S_v is the subset of S for which attribute A has value v (i.e., $S_v = \{s \in S | A(s) = v\}$). Note the first term in Equation (3.4) 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 described by this second term is simply the sum of the entropies of each subset S_v , weighted by the fraction of examples $\frac{|S_v|}{|S|}$ that belong to S_v . $Gain(S, A)$ is therefore the expected reduction in entropy caused by knowing the value of attribute A . Put another way, $Gain(S, A)$ is the information provided about the *target function value*, given the value of some other attribute A . The value of $Gain(S, A)$ is the number of bits saved when encoding the target value of an arbitrary member of S , by knowing the value of attribute A .

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

$$Values(Wind) = Weak, Strong$$

$$S = [9+, 5-]$$

$$S_{Weak} \leftarrow [6+, 2-]$$

$$S_{Strong} \leftarrow [3+, 3-]$$

$$\begin{aligned} Gain(S, Wind) &= Entropy(S) - \sum_{v \in \{Weak, Strong\}} \frac{|S_v|}{|S|} Entropy(S_v) \\ &= Entropy(S) - (8/14)Entropy(S_{Weak}) \\ &\quad - (6/14)Entropy(S_{Strong}) \\ &= 0.940 - (8/14)0.811 - (6/14)1.00 \\ &= 0.048 \end{aligned}$$

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.

Which attribute is the best classifier?

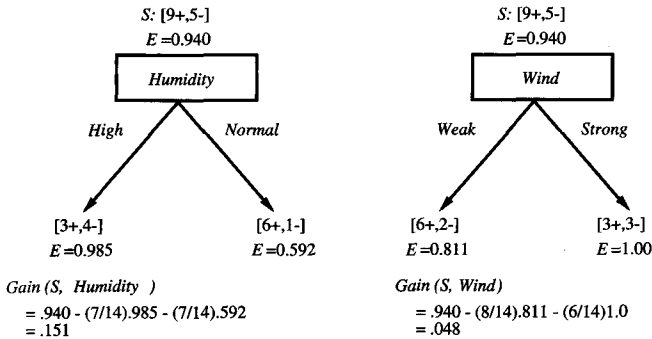


FIGURE 3.3

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*.

3.4.2 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. Consider the first step through

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*.

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

$$\text{Gain}(S, \text{Outlook}) = 0.246$$

$$\text{Gain}(S, \text{Humidity}) = 0.151$$

$$\text{Gain}(S, \text{Wind}) = 0.048$$

$$\text{Gain}(S, \text{Temperature}) = 0.029$$

where S denotes the collection of training examples from Table 3.2.

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 resulting partial decision tree is shown in Figure 3.4, along with the training examples sorted to each new descendant node. 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 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. 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). Figure 3.4 illustrates the computations of information gain for the next step in growing the decision tree. The final decision tree learned by ID3 from the 14 training examples of Table 3.2 is shown in Figure 3.1.

3.5 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 a simple-to-complex, 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