

Reference: [Tom M. Mitchell 1997] Machine Learning

2.3 Decision Tree Learning

Decision tree learning is one of the most widely used and practical methods for inductive inference. It is a method for approximating discrete-valued functions that is robust to noisy data and capable of learning disjunctive expressions. This chapter describes a family of decision tree learning algorithms that includes widely used algorithms such as **ID3**, **Assistant**, and **C4.5**. These decision tree learning methods search a completely expressive hypothesis space and thus avoid the difficulties of restricted hypothesis spaces. Their inductive bias is a preference for small trees over large trees.

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. These learning methods are among the most popular of inductive inference algorithms and 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.

2.3.1 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.

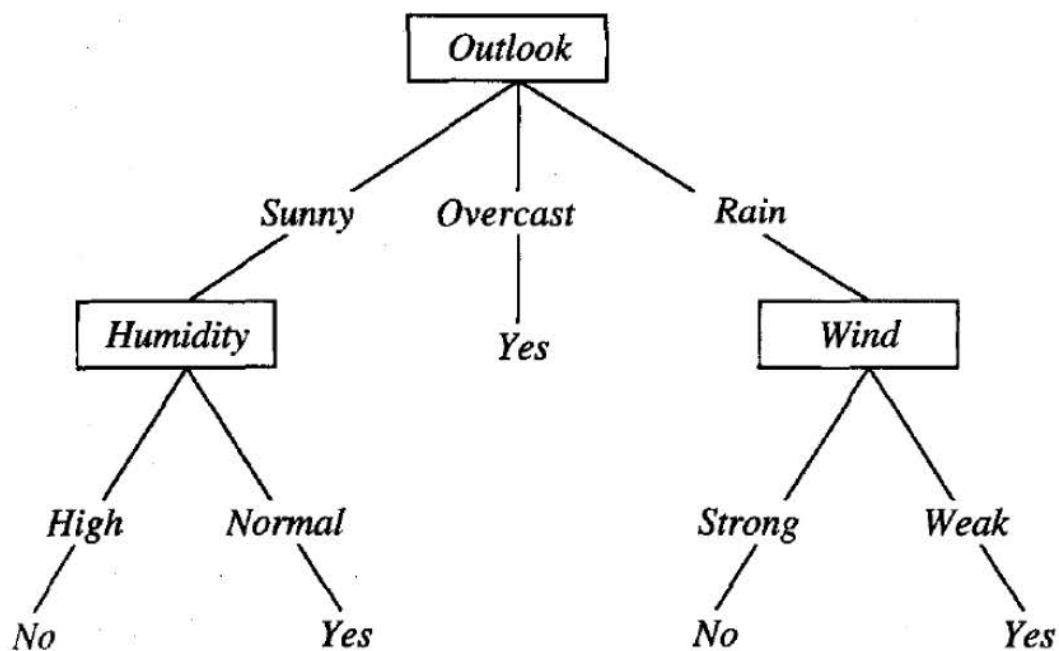


Figure (2.1) A decision tree for the concept *Play Tennis*.

Figure (2.1) illustrates a typical learned decision tree. 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 ^ Humidity = Normal)*
- ∨ *(Outlook = Overcast)*
- ∨ *(Outlook = Rain ^ Wind = Weak)*

2.3.2 Appropriate Problems for Decision Tree Learning:

Although a variety of decision tree learning methods have been developed with somewhat differing capabilities and requirements, 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*). 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*). However, extensions to the basic algorithm allow handling real-valued attributes as well (e.g., representing *Temperature* numerically).
- + ***The target function has discrete output values.*** The decision tree in Figure 3.1 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. A more substantial extension allows learning target functions with real-valued outputs, though the application of decision trees in this setting is less common.
- + ***Disjunctive descriptions may be required.*** As noted above, decision trees 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.
- + ***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).

Decision tree learning has 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*.

2.3.3 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). The basic algorithm for decision tree learning, corresponding approximately to the ID3 algorithm presented in figure (2.2).

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.

2.3.4 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.

Decision Tree Algorithm ID3

Inputs: 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.

Output: 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_i**, be the subset of **Examples** that have value v_i for **A**
 - **If** **Examples_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_i, Target_attribute, Attributes - {A})
- ✚ End
- ✚ Return **Root**

* The best attribute is the one with highest **information gain**.

Figure (2.2) Summary of the *ID3* algorithm specialized to learning boolean-valued functions. 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.