Learning Decision Tree Classifiers

J. R. QUINLAN

University of Sydney, Sydney, Australia ⟨*quinlan@m.12.cs.su.02.au*⟩

Inductive inference is the process of moving from concrete examples to general models. In one form, the goal is to learn how to classify objects or situations by analyzing a set of *instances* whose classes are known. *Classes* here are mutually exclusive labels such as medical diagnoses, qualitative economic projections, image categories, or failure modes. Instances are typically represented as *attribute-value vectors* that give the numerical or nominal values of a fixed collection of properties. Learning input consists of a set of such vectors, each belonging to a known class, and the output consists of a mapping from attribute values to classes. This mapping should accurately classify both the given instances and other unseen instances.

A *decision tree* is a formalism for expressing such mappings. A tree is either a *leaf* node labeled with a class or a structure consisting of a *test* node linked to two or more subtrees. A test node computes some *outcome* based on the attribute values of an instance, where each possible outcome is associated with one of the subtrees. An instance is classified by starting at the root node of the tree. If this node is a test, the outcome for the instance is determined and the process continues using the appropriate subtree. When a leaf is eventually encountered, its label gives the predicted class of the instance.

A decision tree can be constructed from a set of instances by a *divide-and-conquer* strategy. If all the instances belong to the same class, the tree is a leaf with that class as label. Otherwise, a test is chosen that has different outcomes for at least two of the instances, which are partitioned according to this outcome. The tree has as its root a node specifying the test and, for each outcome in turn, the corresponding subtree is obtained by applying the same procedure to the subset of instances with that outcome.

From a *geometric perspective*, a set of x attributes defines an x-dimensional *description space* in which each instance is a point. Partitioning a set of instances according to test outcome corresponds to inserting decision surfaces in this space. In many systems, each test is constrained to reference the value of a single attribute A, so that a test outcome might be $A = c$ for some value c of a nominal attribute, or $A < t$ in which a numeric attribute is compared to a threshold t. Surfaces produced by single-attribute tests are *hyperplanes* orthogonal to the tested attribute A; any decision tree using only such tests will partition the description space into hyperrectangles, each associated with a class.

More *complex tests* can be used to reduce the number of times that the instances are subdivided, thus avoiding the *data fragmentation* problem. For multivalued nominal attributes, the simplest generalization is to subset tests $A \in \{c_1, c_2, \ldots\}$. Other sophisticated tests use more than one attribute, for example, logical combinations, sets of conditions, or linear combinations of numeric attributes (this latter permitting decision surfaces that are arbitrary hyperplanes).

Unless there are identical attribute-value vectors labeled with different classes, any *choice of tests* leads to a tree that is consistent with the instances. Smaller trees are preferred, however, because they are easier to understand and may also have higher predictive accuracy. Potential tests are assessed on the class distributions of the subsets associated with test outcomes, using heuristics such as the *Gini index of diversity* or *information gain ratio*.

Real data are commonly affected by *noise* arising from misclassification or incorrect measurement or recording of attribute values. These cause the divide-and-conquer algorithm to generate elaborate trees that attempt to model the discrepancies. Such *overfitting* is usually addressed by *pruning* the initial tree—identifying subtrees that contribute little to predictive accuracy and replacing each by a leaf. Common pruning techniques are based on cost-complexity models, pessimistic accuracy estimates, or minimizing the length of a message describing the tree and the data.

In tasks with *more than two classes*, an alternative to growing a single tree is to construct a tree for each class that distinguishes it from all others. The idea can be taken further, encoding classes as bit strings with error correction and producing a separate tree for each bit.

Higher predictive accuracy can usually be obtained by generating *multiple trees* from the data, all of which are used in classifying a new instance. More than one test can be used to partition the instances at each stage, giving families of superimposed trees, or multiple training sets can be samples from the data. The predictions from several trees can be combined by simple voting or by more sophisticated techniques such as *stacking*.

Even though the divide-and-conquer algorithm is fast, *efficiency* can become important in tasks with hundreds of thousands of instances or where many trees are to be produced. The most time-consuming facet is sorting the instances on a numeric attribute to find the best threshold t. This can be expedited if possible thresholds for a numeric attribute are determined just once, effectively converting the attribute to discrete intervals, or if the threshold is determined from a subset of the instances. Parallel algorithms for finding t have also been developed. Finally, an *incremental* approach that updates the tree as new data come to hand can be faster than repeatedly regrowing a complete tree.

No single reference covers all these issues, although Quinlan [1996] contains pointers to the relevant research literature. Breiman et al. [1984] and Quinlan [1996] describe complete systems in considerable detail. Michie et al. [1994] compare alternative classifier-learning methodologies (including several based on decision trees) on applications in industry and commerce. Some of the earliest work is reported in Hunt et al. [1966], and Russell and Norvig [1995] and Winston [1992] give excellent tutorial overviews of inductive learning.

REFERENCES

BREIMAN, L., FRIEDMAN, J. H., OLSHEN, R. A., AND STONE, C. J. 1984. *Classification and Regression Trees*. Wadsworth, Belmont, CA.

HUNT, E. B., MARIN, J., AND STONE, P. J. 1966. *Experiments in Induction*. Academic Press, New York.

MICHIE, D., SPIEGELHALTER, D. J., AND TAYLOR, C. C., EDS. 1994. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, Hertfordshire, UK.

QUINLAN, J. R. 1996. Decision trees and instance-based classifiers. In *CRC Handbook of Computer Science and Engineering*. A. B. Tucker, Ed., CRC Press, Boca Raton, FL.

QUINLAN, J. R. 1993. *C4.5: Programs for Machine Learning*. Morgan-Kaufmann, San Francisco.

RUSSELL, S., AND NORVIG, P. 1995. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs, NJ.

WINSTON, P. H. 1992. *Artificial Intelligence*, 3rd edition. Addison-Wesley, Reading, MA.