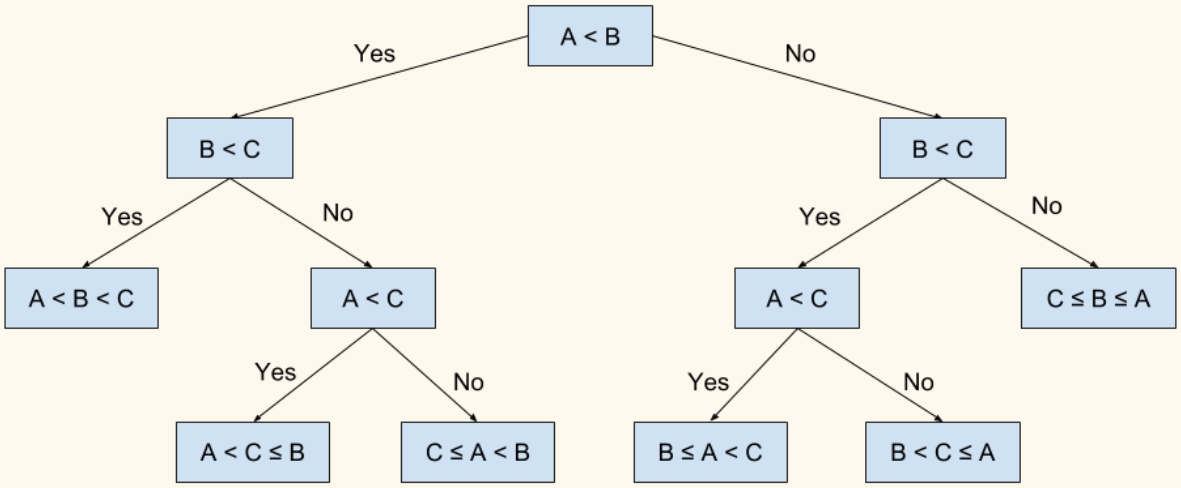
Decision Trees

Decision tree learning is a method for approximating discrete-valued target functions, in which the learning function is represented by a decision tree, where the tree is a set of if-then rules to improve readability. This method is very popular among inductive inference algorithms and has been successfully applied to a broad range of tasks.

Representation

Decision Trees classify instances by sorting them down the tree, to some leaf that provides the classification label. Each node of the tree tests some attribute of the instance, each branch descending from that node corresponds to one possible value of the attribute.



Decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances.

Fig: https://elf11.github.io/2018/07/01/python-decision-trees-acm.html

Characteristics of a Decision Tree Learning Problem

1. Instances are represented by attribute value pairs
2. The target function has discrete output values
3. Disjunctive descriptions may be required
4. The training data may contain errors
5. The training data may contain missing attribute values

Such problems are usually **classification problems.**

The basic ID3 algorithm learns decision trees by constructing them top-down, beginning with the attribute selection for the root. The descendant for the root is the set of values the attribute can take, and the algorithm is recursively applied.

To find the best attribute, we define **Information gain (G)** that measures how well a given attribute separates the training examples.

Entropy

Entropy characterizes the (im)purity of an arbitrary collection of samples. Given a collection S containing positive and negative examples of some target concept,

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

If the target concept can take more than two values, the entropy is just the summation of all these log-probability products for each value. The entropy distribution is given for a target value as

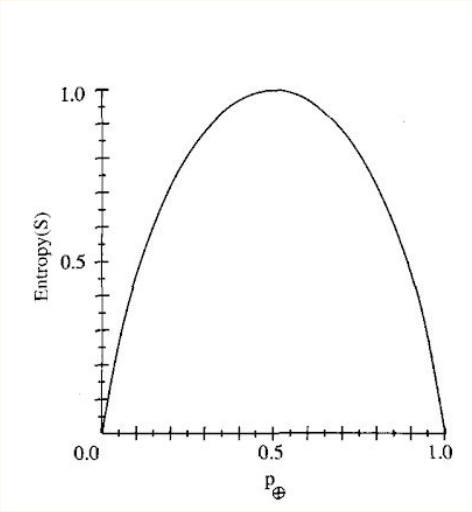


Fig : Tom Mitchell, Pg. 60

Information Gain

The information Gain measures the expected reduction in entropy. For an attribute A and a collection of attributes S, the Gain is given by

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

The second term represents the expected entropy of the set if it were classified through attribute A. The value of Gain is also the number of bits saved when encoding the target value of an arbitrary number of S, by knowing the value of attribute A.

The ID3 algorithm can be summarized as

1. Create a root node for the tree
2. If all examples are positive, return a single node tree with label +
3. If all examples are negative, return a single node tree with label -
4. If Attribute set is empty, return a single node tree with the most common value of the target attribute
5. Otherwise, Begin
    a. A = Attribute with maximum gain
    b. The decision attribute for root = A
    c. For each possible value v of A
        i. Add a new branch corresponding to A = v
        ii. Let examples E be a subset of Examples that have value v for A
        iii. If E is empty
            1. Add a leaf node with label as most common value for target attribute
            2. Else make a subtree of the Attributes - {A}
6. End
7. Return root

Hypothesis space search in Decision Tree Learning

ID3 performs a hill-climbing search through the hypotheses space, where this space is the set of possible decision trees. Its capabilities are

1. The space of hypotheses is a complete space of finite discrete valued functions, relative to all available attributes. So the search is not through an incomplete hypothesis space (where the space may not contain the target function).
2. ID3 maintains only a single current hypothesis, does not determine the outcome of alternate hypotheses.
3. ID3 performs no backtracking in its search, which might risk a locally optimal solution.
4. This uses all training examples at each step to make statistically based decisions regarding how to refine the current hypothesis.

Inductive Bias in Decision Tree Learning

Inductive Bias is a set of assumptions that justify the classifications assigned by a learner to future instances. The ID3 algorithm search strategy introduces this bias as

1. It selects in favour of shorter trees over longer trees
2. It selects trees that place highest info-gain attributes closest to the root

BFS-ID3 searches all the possible decision trees and uses a greedy-heuristic search to find the shortest tree that models the given data fairly, by going level by level.

When an algorithm prefers a certain hypothesis over another, it is said to have a preference bias, as observed in ID3. Whereas, the algorithm that enforces a categorical restriction on the attributes, the bias is called a restriction bias, as observed in Candidate Elimination.

Typically, a preference bias is more desirable, as it allows the learner to work within a complete hypothesis space that is assured to contain the unknown target function.

Issues in Decision Tree Learning

I. Overfitting

Given a hypothesis space H, a hypothesis h in H is said to overfit if there exists some alternative hypothesis h' such that h has a smaller error than h' on the training data, but a larger error over the entire distribution.

Overfitting can be detected by the following method

1. Use of a training set and a validation set

2. Applying a statistical test to see if expanding a node is likely to produce improvement on the test set
3. Use an explicit measure for encoding the training examples and the decision tree, and halting growth when the encoding size is minimized.

Reduced Error Pruning

This approach considers each decision node to be candidates for pruning. It consists of removing the subtree at that node, making it a leaf and assigning it the most common class observed for that node. A node is pruned only if the resulting tree performs no worse on the validation set.

This is suitable when a large amount of data is available. When the data is not much, withholding part of it for the validation set reduces the number of examples for training, which is already less.

Rule Post Pruning

Rule post pruning involves the following steps

1. Infer the decision tree from the training set, growing the tree until the data is fit as well as possible, allowing overfitting.
2. Convert the learned tree into a set of equivalent rules, creating one rule for each path.
3. Generalise or prune each rule by removing any preconditions that increase the tree's accuracy.
4. Sort the pruned rules by their estimated accuracy, and consider them in sequence for classifying instances.

The tree is converted into rules because

1. Allows distinguishing between different contexts in which a node is used. Because each rule is distinct, pruning can be calculated differently for each path.
2. Removes distinction between attribute tests that occur near the route, reducing bookkeeping issues as to how to reorganise the tree.
3. Improves readability.

II. Integration of Continuous-Valued Attributes

This can be fixed by discretizing the continuous valued attributes, by dynamically defining new discrete values to fit these values.