Unit-3

Chapter-1

Tree models

Decision Trees are a non-parametric supervised **learning** method used for both classification and regression tasks. **Tree models** where the target variable can take a discrete set of values are called classification **trees.**

A decision tree is a tree-like graph with nodes representing the place where we pick an attribute and ask a question; edges represent the answers the to the question; and the leaves represent the actual output or class label. They are used in non-linear decision making with simple linear decision surface.

Let's illustrate this with help of an example. Let's assume we want to play badminton on a particular day — say Saturday — how will you decide whether to play or not. Let's say you go out and check if it's hot or cold, check the speed of the wind and humidity, how the weather is, i.e. is it sunny, cloudy, or rainy. You take all these factors into account to decide if you want to play or not.

So, you calculate all these factors for the last ten days and form a lookup table like the one below.

Day	Weather	Temperature	Humidity	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Cloudy	Hot	High	Weak	Yes
3	Sunny	Mild	Normal	Strong	Yes
4	Cloudy	Mild	High	Strong	Yes
5	Rainy	Mild	High	Strong	No
6	Rainy	Cool	Normal	Strong	No
7	Rainy	Mild	High	Weak	Yes
8	Sunny	Hot	High	Strong	No
9	Cloudy	Hot	Normal	Weak	Yes
10	Rainy	Mild	High	Strong	No

Table 1. Obeservations of the last ten days.

Now, you may use this table to decide whether to play or not. But, what if the weather pattern on Saturday does not match with any of rows in the table? This may be a problem. A decision tree would be a great way to represent data like this because it takes into account all the possible paths that can lead to the final decision by following a tree-like structure.

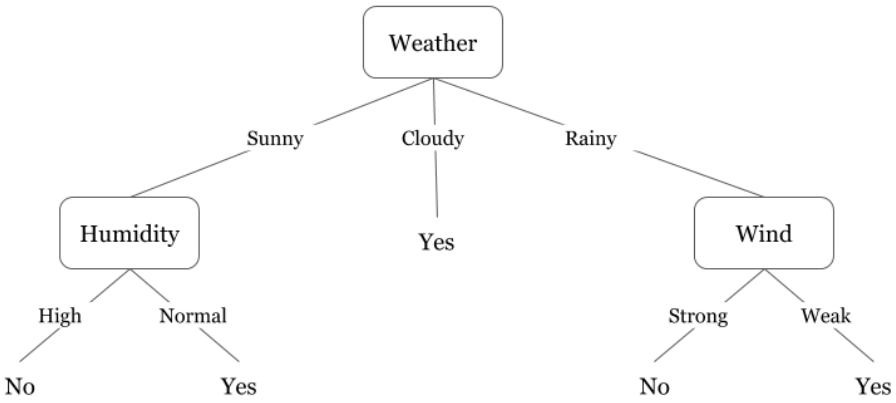


Fig 1. illustrates a learned decision tree. We can see that each node represents an attribute or feature and the branch from each node represents the outcome of that node. Finally, its the leaves of the tree where the final decision is made.

A general algorithm for a decision tree can be described as follows:

1. Pick the best attribute/feature. The best attribute is one which best splits or separates the data.
2. Ask the relevant question.
3. Follow the answer path.
4. Go to step 1 until you arrive to the answer.

The best split is one which separates two different labels into two sets.

Expressiveness of decision trees

Decision trees can represent any boolean function of the input attributes. Let's use decision trees to perform the function of three boolean gates AND, OR and XOR.

Boolean Function: AND

A	B	A AND B
F	F	F
F	T	F
T	F	F
T	T	T

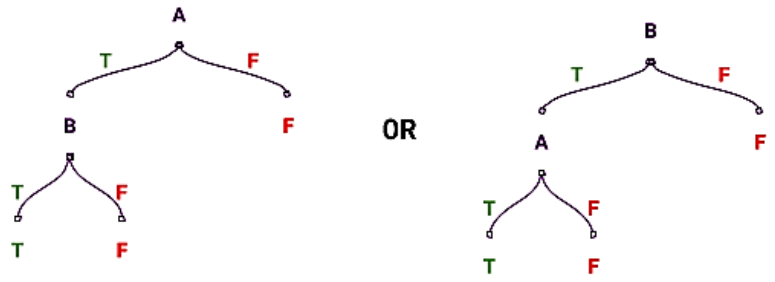


Fig 3. Decision tree for an AND operation.

In Fig 3., we can see that there are two candidate concepts for producing the decision tree that performs the AND operation.

The decision tree learning algorithm

The basic algorithm used in decision trees is known as the ID3 (by Quinlan) algorithm. The ID3 algorithm builds decision trees using a top-down, greedy approach. Briefly, the steps to the algorithm are: - Select the best attribute → A - Assign A as the decision attribute (test case) for the **NODE**. - For each value of A, create a new descendant of the **NODE**. - Sort the training examples to the appropriate descendant node leaf. - If examples are perfectly classified, then STOP else iterate over the new leaf nodes.

Pseudocode: 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.

The pseudocode assumes that the attributes are discrete and that the classification is binary. Examples are the training example. *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. Finally, it returns a decision tree that correctly classifies the given *Examples*.

Tree learning as variance reduction
Reduction in Variance

Reduction in variance is an algorithm used for continuous target variables (regression problems). This algorithm uses the standard formula of variance to choose the best split. The split with lower variance is selected as the criteria to split the population:

$$\text{Variance} = \frac{\Sigma(X - \overline{X})^2}{n}$$

Above X-bar is mean of the values, X is actual and n is number of values.

Steps to calculate Variance:

1. Calculate variance for each node.
2. Calculate variance for each split as weighted average of each node variance.

Rule models

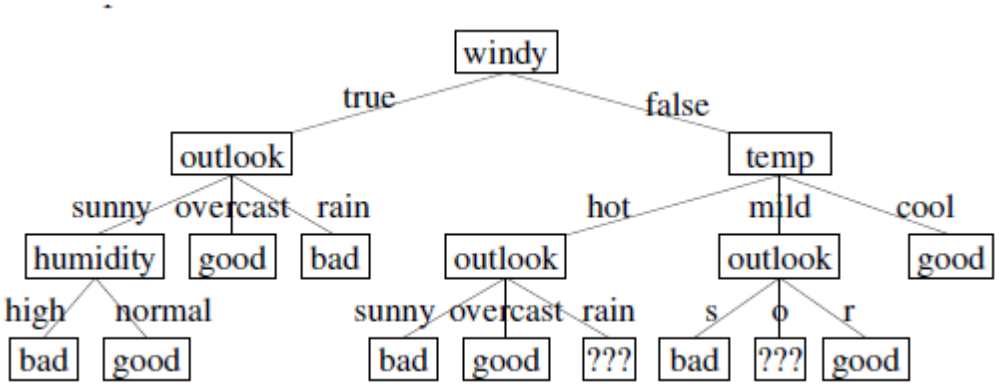
A rule-based system is used to store and manipulate knowledge to interpret information in a useful way. It is often used in artificial intelligence applications and research.

Learning ordered rule lists

Learning Rules
In learning rules, we are interested in learning rules of the form:
if A1 ^ A2 ^ . . . then C
where A1, A2, . . . are the preconditions/constraints/body/ antecedents of the rule and C is the postcondition/head/ consequent of the rule.
A first-order rule can contain variables, as in:
if Parent(x, z) ^ Ancestor(z, y) then Ancestor(x, y)
A Horn clause is a rule with no negations.

Learning Rules from Decision Trees

Each path in a decision tree from root to a leaf:



can be converted to a rule, e.g.:
if ¬windy ^ hot ^ sunny then bad
Why? Rules are considered more readable/understandable.
Different pruning decisions can be made for different rules.
Rule Post-Pruning
Rule post-pruning removes conditions to improve error.
if windy ^ sunny ^ high then bad
if windy ^ sunny ^ normal then good
if windy ^ overcast then good

if windy ^ rain then bad
if ¬ windy ^ hot ^ sunny then bad
if ¬windy ^ hot ^ overcast then good
if ¬windy ^ mild ^ sunny then bad
if ¬windy ^ mild ^ rain then good
if ¬windy ^ cool then good

Rule Ordering

Order rules by accuracy and coverage.
4 if overcast then good
3 if ¬windy ^ rain then good
3 if sunny ^ high then bad
2 if sunny ^ normal then good
2 if ¬windy ^ cool then good
2 if windy ^ rain then bad
2 if hot ^ sunny then bad
1 if ¬windy ^ mild ^ sunny then bad

Sequential Covering Algorithms

Basic Algorithmic Idea:
1. Learn one good rule.
2. Remove the examples covered by the rule.
3. Repeat until no examples are left.

Learning First-Order Rules

Suppose we are interested in learning concepts involving relationships between objects.
• When one person is an ancestor of another number.
• When one number is less than another number.
• When one node can reach another node in a graph.
• When an element is in a set.
The concept involves intermediate objects and relations.

Learning Unordered rule sets