Module 12

Machine Learning

Lesson
36

Rule Induction and Decision Tree - II

Splitting Functions

What attribute is the best to split the data? Let us remember some definitions from information theory.

A measure of uncertainty or entropy that is associated to a random variable X is defined as

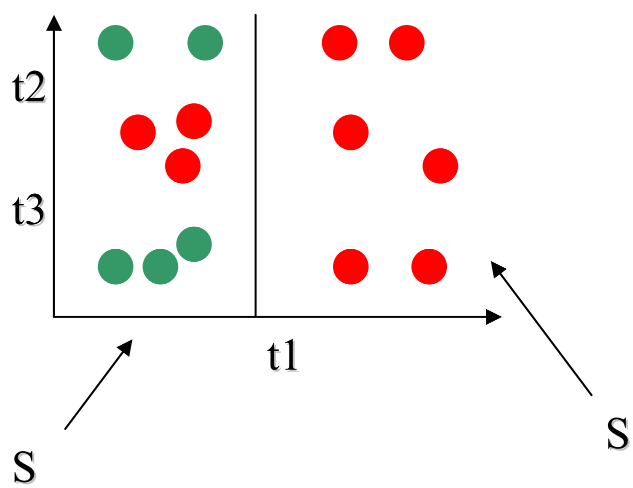
$$H(X) = - \Sigma \; pi \log pi$$

where the logarithm is in base 2.

This is the "average amount of information or entropy of a finite complete probability scheme".

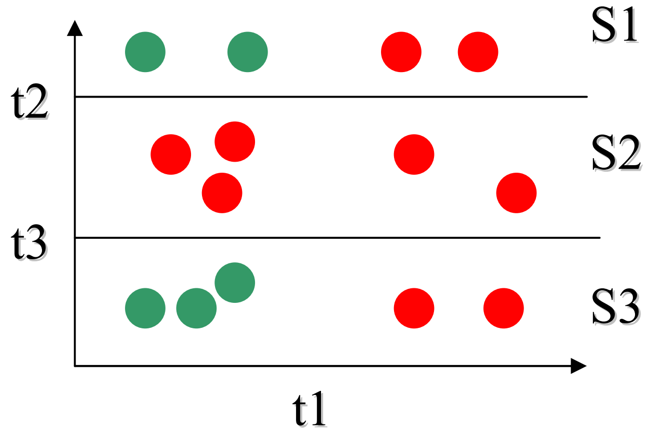
We will use a entropy based splitting function.

Consider the previous example:



Size divides the sample in two.
S1 = { 6P, 0NP}
S2 = { 3P, 5NP}

H(S1) = 0
H(S2) = -(3/8)log2(3/8)
-(5/8)log2(5/8)



humidity divides the sample in three.
S1 = { 2P, 2NP}
S2 = { 5P, 0NP}
S3 = { 2P, 3NP}

H(S1) = 1
H(S2) = 0
H(S3) = -(2/5)log2(2/5)
-(3/5)log2(3/5)

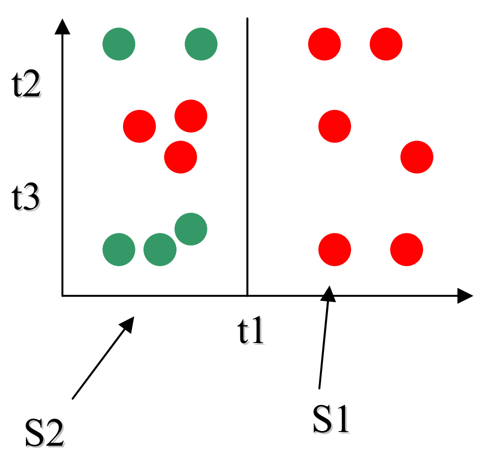
Let us define *information gain* as follows:

Information gain IG over attribute A: IG (A)

IG(A) = H(S) - Σv (Sv/S) H (Sv)

H(S) is the entropy of all examples. H(Sv) is the entropy of one subsample after partitioning S based on all possible values of attribute A.

Consider the previous example:



We have,

H(S1) = 0
H(S2) = -(3/8)log2(3/8)
-(5/8)log2(5/8)

H(S) = -(9/14)log2(9/14)
-(5/14)log2(5/14)

|S1|/|S| = 6/14
|S2|/|S| = 8/14

The principle for decision tree construction may be stated as follows:

Order the splits (attribute and value of the attribute) in decreasing order of information gain.

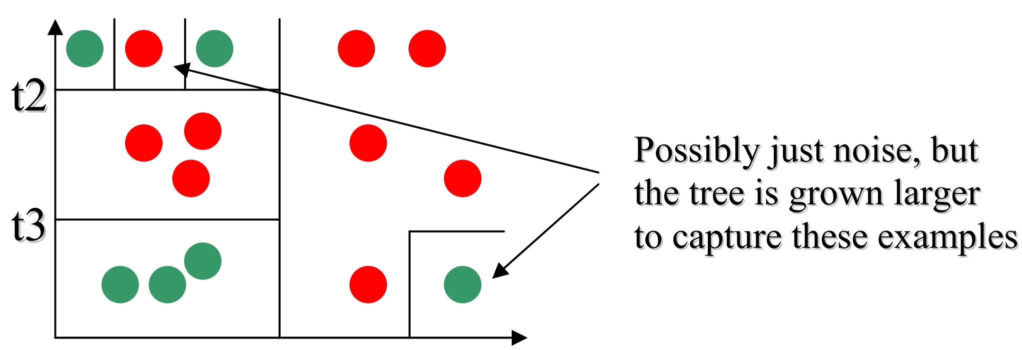
12.3.4 Decision Tree Pruning

Practical issues while building a decision tree can be enumerated as follows:

1) How deep should the tree be?
2) How do we handle continuous attributes?
3) What is a good splitting function?
4) What happens when attribute values are missing?
5) How do we improve the computational efficiency

The depth of the tree is related to the generalization capability of the tree. If not carefully chosen it may lead to overfitting.

A tree *overfits* the data if we let it grow deep enough so that it begins to capture "aberrations" in the data that harm the predictive power on unseen examples:



There are two main solutions to overfitting in a decision tree:

1) Stop the tree early before it begins to overfit the data.

+ In practice this solution is hard to implement because it is not clear what is a good stopping point.

2) Grow the tree until the algorithm stops even if the overfitting problem shows up. Then prune the tree as a post-processing step.

+ This method has found great popularity in the machine learning community.

A common *decision tree pruning algorithm* is described below.

1. Consider all internal nodes in the tree.
2. For each node check if removing it (along with the subtree below it) and assigning the most common class to it does not harm accuracy on the validation set.
3. Pick the node n* that yields the best performance and prune its subtree.
4. Go back to (2) until no more improvements are possible.

Decision trees are appropriate for problems where:

- Attributes are both numeric and nominal.
- Target function takes on a discrete number of values.
- A DNF representation is effective in representing the target concept.
- Data may have errors.