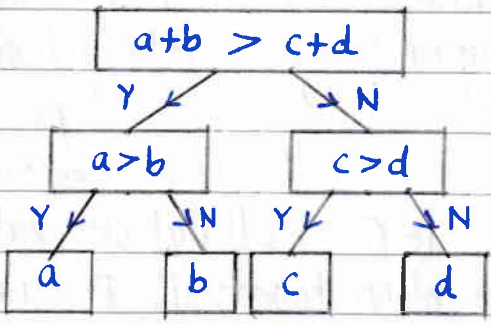
Decision Trees

It is a highly transparent pattern classification approach. A decision tree is a tree where each non-leaf or internal node is associated with a decision and the leaf nodes are generally associated with an outcome or a class label.

Decision trees are excellent tools for choosing between several courses of action. In case of a binary decision tree, each node gives the statement of the decision to be taken or the comparison to be made. There are two outgoing edges, one that represents a 'yes' or 'true' and other representing a 'no' or 'false'.

Example - There are 4 coins a, b, c, d out of which 3 coins are of equal weight and one coin is heavier. Find out the heavier coin.



The root node compares the weight of 'a+b' with 'c+d'. If 'a+b' is heavier than 'c+d', outcome is 'yes' & it takes the left branch. The node in the left branch compares the weight of 'a' with 'b'. If outcome is 'yes', then it chooses 'a' as the heavier coin. The same approach is used on the right side branch also.

This is the example of a simple decision tree, in the context of decision making. Some salient features are- i) There are 4 leaf nodes, corresponding to 4 possible outcomes. Each leaf node corresponds to one of the coins being heavier. ii) It requires two weighings to make the final decision, or to reach a leaf node. Each decision node (non-terminal or internal node) corresponds to a weighing operation. Further, there are two decision nodes from the root to any leaf. iii) Each path from the root to a leaf corresponds to a rule.

An internal node in the decision tree is associated with a test, based on the values of one or more features. The tests can be categorized as -

i) Axis-parallel test - This test is of the form $x > a_0$, where x is a feature and a_0 is a threshold value. e.g. height > 6 ft. Thus, this test is associated with a hyper-plane parallel to all the feature axes other than that of 'height', which splits the pattern space in 2 parts - patterns having height > 6 ft and another with height < 6 ft. This test involves only one feature. Such a split is called axis-parallel split.

(ii) Linear combination test - This test is of the form,

$$\sum_{i=1}^{d} a_i x_i > a_0$$, where x_i = i^{th} feature
a_i = weight associated with it.

This test involves a linear combination of feature values and the corresponding hyper-plane need not be parallel to any axis. e.g. 0.4 * height + 0.3 * weight > 38 can be a test. This test splits the space into two parts, based on an 'oblique' split. The 'axis-parallel test', can then be considered as a special case of the oblique split, where a_i is 0 for all but one value of i.

iii) Non-linear combination test - This is the most general form of test possible & is of the form, $f(x) > 0$, where $f(x)$ is any non-linear function of the components of x. It can be seen that axis-parallel test and oblique-splits can be considered as the special cases of this test. It is computationally the most expensive type of test.

Patterns can be classified using decision trees, where the nodes in the tree represent the status of the problem after making some decision. The leaf nodes give the class label of the classification rule, corresponding to the path from the root node to that leaf node.

Weaknesses of decision trees -

i) Design time could be large. For example, the number of oblique splits possible could be exponential in the number of training patterns. If there are 'n' training instances, each having 'd' attributes, there are atmost $2^d \binom{n}{d}$ distinct d-dimensional oblique splits. This problem is exponential in the number of training patterns.

ii) Simple (axis-parallel) decision trees do not represent non-rectangular regions well. Most decision tree algorithms only examine a single attribute at a time. In simple decision tree, all the decision boundaries are hyperplanes orthogonal to the axis of the feature being considered.

iii) For typical functions, such as parity function, which is of the form $x \in \{0,1\}^d$ and outputs 1 if the number of 1s in x is odd and 0 otherwise. Hence it is necessary to use all the 'd' bits to decide the output. This makes the decision tree very large.

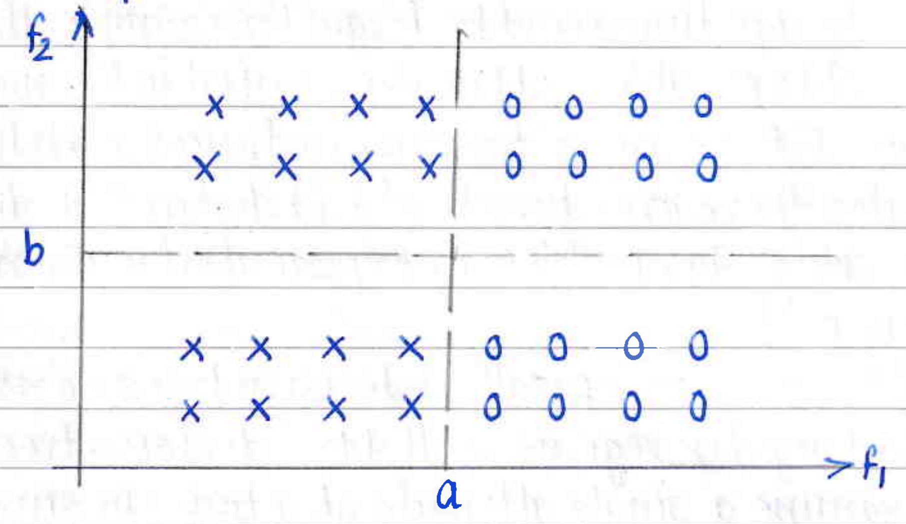
Construction of decision tree - A decision tree is induced from examples. The tree is constructed so that it agrees with the training set, which helps to describe a large number of cases in a concise way.

To construct a decision tree, one or more attributes have to be chosen at each node to make a decision. The most important attribute is used at the earliest level in the decision tree. This attribute is that which makes the most difference to the classification. At each node, the set of examples is split up and each outcome is a new decision tree learning problem itself.

A really good attribute divides the examples into sets that are distinct, such as all positive & all negative etc. i.e. many of the outcomes result in a definitive answer, then it is a good attribute to choose.

Once a correct attribute is chosen, the different outcomes represent new decision trees, and in each case, the most important attribute is chosen. This is done till all nodes give a final classification.

Consider the data point distribution as shown



Which decision is better ? i) $f_1 \geq a$ or ii) $f_2 \geq b$?

At each node, the query which makes data to the subsequent nodes as 'pure' as possible is chosen. Thus as an approach, instead of measuring how 'pure' the node is, the 'impurity' of the resulting nodes is minimized.

Measures of impurity -

1) Entropy impurity or information impurity -

The entropy impurity at a node N is $i(N)$ & is given by

$i(N) = -\sum_j P(\omega_j) \log_2 P(\omega_j)$, where $P(\omega_j)$ is the fraction of patterns at node N of category ω_j.

Consider the case, when the patterns are split equally into two subsets. Then $P(W_j) = 0.5$ and we get

$i(N) = -0.5 \log_2 (0.5) - 0.5 \log_2 (0.5) = 1$

When all patterns go along one branch and there are no patterns in the other branch, $i(N) = 0$.

Consider 10 patterns that are split in 3 classes, with 4, 5 & 1 samples respectively. Then $P(W_1) = 0.4$, $P(W_2) = 0.5$, $P(W_3) = 0.1$

$i(N) = -0.4 \log_2 (0.4) - 0.5 \log_2 (0.5) - 0.1 \log_2 (0.1) = 1.36$

2) Variance impurity -

$i(N) = P(W_1)\,P(W_2)$ in a 2 category case.

When $P(W_1) = P(W_2) = 0.5$, $i(N) = 0.25$

Generalising this to more categories,

$$i(N) = \sum_{i \neq j} P(W_i)\,P(W_j) = \left[1 - \sum_j P_i^2(W_j)\right]$$

This is called Gini impurity.

As in the earlier case, if $P(w_1) = 0.4$, $P(w_2) = 0.5$, $P(W_3) = 0.1$, then Gini impurity is

$$i(N) = \left[1 - 0.4^2 - 0.5^2 - 0.1^2\right] = 0.58$$

3) Misclassification impurity -

$i(N) = 1 - \max_j P(W_j)$. Hence in the previous example,

$i(N) = 1 - \max(0.4, 0.5, 0.1) = 0.5$.

This measures the minimum probability that a training pattern would be misclassified.

The attribute to be chosen should decrease the impurity as much as possible. The drop in impurity is defined as

$$\Delta i(N) = i(N) - P_L\, i(N_L) - (1 - P_L)\, i(N_R).$$

This is the case when there are only 2 branches at the node. Here P_L and N_L correspond to the left branch & N_R correspond to the right branch. If there are more than 2 branches, we get

$$\Delta i(N) = i(N) - \sum_j P_j \times i(N_j)$$

$\Delta i(N)$ is also called the gain in information at the node. The attribute which maximizes $\Delta i(N)$ is to be chosen.

Consider a case, where there are 100 examples, with 40 belonging to class C_1, 30 belonging to C_2 & remaining 30 belonging to C_3. Let some attribute x split these examples into 2 branches – left branch containing 60 examples & remaining 40 along the right branch. The left branch contains 40 examples of C_1, 10 examples of C_2 and 10 examples of C_3. The right branch contains 0 examples of C_1 & 20 each of C_2 and C_3 respectively. Calculate the gain in the information at the node.

Attribute : X

Class	X=a=60 Left Branch	X=b=40 Right Branch	Total Examples
C_1	40	0	40
C_2	10	20	30
C_3	10	20	30
	60	40	100

1) Using entropy impurity –

The entropy impurity at the split is –

$$i(N) = -0.4 \log_2 0.4 - 0.3 \log_2 0.3 - 0.3 \log_2 0.3 = 1.57$$

The entropy of the left branch $i(N_L)$ is –

$$i(N_L) = -\frac{40}{60} \log_2 \frac{40}{60} - \frac{10}{60} \log_2 \frac{10}{60} - \frac{10}{60} \log_2 \frac{10}{60} = 1.25$$

The entropy of the right branch $i(N_R)$ is –

$$i(N_R) = -\frac{20}{40} \log_2 \frac{20}{40} - \frac{20}{40} \log_2 \frac{20}{40} = 1$$

The drop in impurity i.e. gain is

$$\Delta i(N) = 1.57 - \left(\frac{60}{100} \times 1.25\right) - \left(\frac{40}{100} \times 1\right) = 0.42$$

2) Using Gini impurity –

The impurity at the node is

$$i(N) = (1 - 0.4^2 - 0.3^2 - 0.3^2) = 0.66$$

The impurity at the left node is-
$i(N_L) = (1-0.6667^2-0.1667^2-0.1667^2) = 0.50$

The impurity at the right node is -
$i(N_R) = (1-0.5^2-0.5^2) = 0.50$

The drop in impurity i.e. gain is
$\Delta i(N) = 0.66 - \left(\frac{60}{100} \times 0.50\right) - \left(\frac{40}{100} \times 0.50\right) = 0.16$

3) Using misclassification impurity-
The impurity at the node is
$i(N) = 1 - \max(0.4, 0.3, 0.3) = 0.6$

The impurity at the left node is
$i(N_L) = 1 - \max\left(\frac{4}{6}, \frac{1}{6}, \frac{1}{6}\right) = 0.333$

The impurity at the right node is
$i(N_R) = 1 - \max\left(\frac{2}{4}, \frac{2}{4}\right) = 0.5$

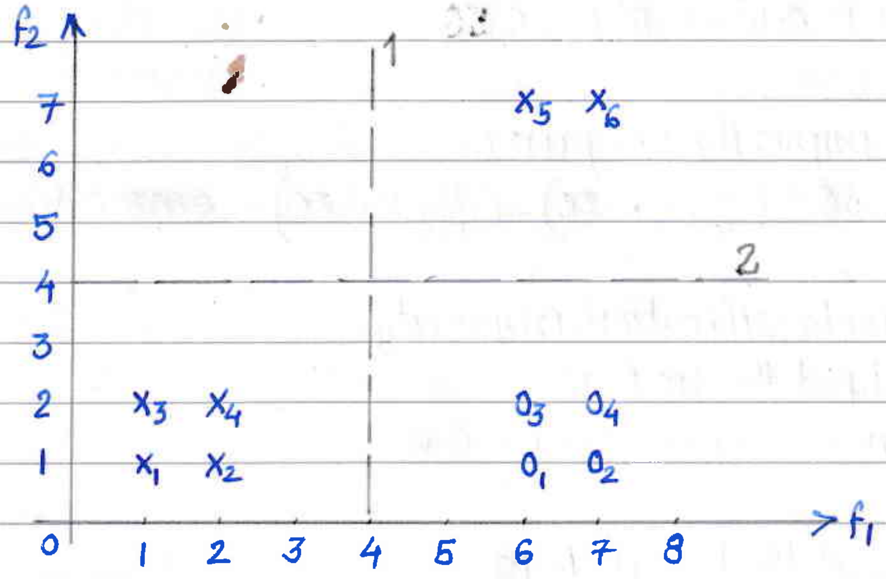
The drop in impurity i.e. gain is
$\Delta i(N) = 0.6 - \left(\frac{60}{100} \times 0.333\right) - \left(\frac{40}{100} \times 0.5\right) = 0.202$

Splitting at the nodes - Each decision outcome at a node is called a split. The root node splits the full training data. Each successive decision splits the data at the node into proper sub-sets. The decision rule at each non-leaf node is of the form $f_i(x) > a_0$.
Depending on this split rule, there can be 2 approaches-
i) Axis parallel split - Pertaining to splitting, depending on the value of a single attribute and is of the form $x_j > a_0$.
ii) Oblique split - Here, the general form of the equation is
$a*f_1 + b*f_2 + c > 0$.

For the given dataset, develope a decision tree using - i) Axis-parallel split and ii) Oblique split.

$X_1 = (1,1); X_2 = (2,1); X_3 = (1,2); X_4 = (2,2); X_5 = (6,7); X_6 = (7,7)$

$O_1 = (6,1); O_2 = (7,1); O_3 = (6,2); O_4 = (7,2)$



i) Axis-parallel split - The rule at the first node in the axis-parallel split would be $f_1 > 4$ and the rule at the next level node would be $f_2 > 4$. The decision tree would be as shown-

$$\begin{tikzcd}
 & \boxed{f_1 > 4} \arrow[dl, "Y"'] \arrow[dr, "N"] & \\
\boxed{f_2 > 4} \arrow[d, "Y"'] \arrow[dr, "N"] & & \boxed{X} \\
\boxed{X} & \boxed{O} &
\end{tikzcd}$$

X_1 to X_4

X_5, X_6 O_1 to O_4

ii) Oblique split - In oblique split, using the general form, $a * f_1 + b * f_2 + c > 0$, we get

$2a + b + c > 0$

$7a + 7b + c > 0$

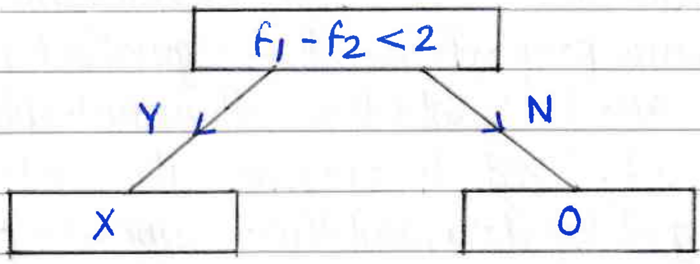
$6a + 2b + c < 0$

Solving the inequalities, we get $a = -1, b = 1$ & $c = 2$.

The rule of the oblique split therefore is $-f_1 + f_2 + 2 > 0$

i.e. $f_1 - f_2 < 2$

The decision tree would be as shown -



Rules for stopping splitting -

1) Reduction in impurity - splitting can be stopped, when the reduction in impurity is very small. i.e. $\max i(s) \leq \beta$, where β is a predecided small value.

2) Depending on the global criteria function - If the criteria function is of the form $\alpha * size + \sum_{leaf\ nodes} i(N)$, where α is a positive constant & size = no. of nodes

The strategy would be to stop splitting when this global criteria reaches a predecided minimum value.

Overfitting & pruning -

If every pattern in the training set is associated with a unique path through the decision tree, the tree becomes too detailed and specific. If the height of the decision tree keeps on increasing and the tree becomes very large because of splitting small size training sets at nodes, it is called over fitting. Decision tree pruning is used to prevent overfitting. Pruning removes splitting on attributes which are not wanted.

One method to carry out pruning is to use cross validation. Cross-validation is carried out by keeping a part of the training data aside for validation. This is done repeatedly, keeping different subsets for validation and recording the performance found in each case. The decision tree giving the best result during validation is considered.

Another method of pruning is to find the irrelevent attributes and prune them. A split which divides the data into the subsets in the same proportion as the original set, results in zero information gain. Attributes which result in such splits can be found and pruned. The lack of reduction in impurity can be used to stop splitting. Looking at the data, and the number of positive and negative examples, we can calculate the extent to which it deviates from a perfect absense of pattern. If the degree of deviation is statistically unlikely, then this is considered as a good evidence for the presence of a significant pattern in the data. The deviation in terms of number of positive & negative examples p_i and n_i and the expected numbers $\hat{p}_i$ and $\hat{n}_i$ will be

$$D = \sum_{i=1}^{v} \frac{(p_i - \hat{p}_i)^2}{\hat{p}_i} + \frac{(n_i - \hat{n}_i)^2}{\hat{n}_i}$$, where v is the sample size.

D is distributed according to χ^2 (Chi-squared) distribution. The probability that the attribute is really irrelevant can be calculated with the help of the standard χ^2 table. This is called χ^2 pruning.

Among the many decision tree learning algorithms, notable ones are –

1) ID-3 - Iterative dichotomiser 3.
2) CART - Classification and Regression Tree
3) CHAID - Chi-squared Automatic interaction detector
4) MARS - Multivariate adaptive regression spline
5) QUEST - Quick unbiased efficient statistical tree.

Advantages of decision trees –

1) Simple to understand and interpret 2) Requires little data preparation 3) Ability to handle both numerical and categorical data 4) Uses a transparent i.e. 'white box' model. 5) Possible to validate a model using statistical tests. 6) Robust performance even if it's assumptions are somewhat violated from the true model from which the data were generated. 7) Performs well with large datasets.