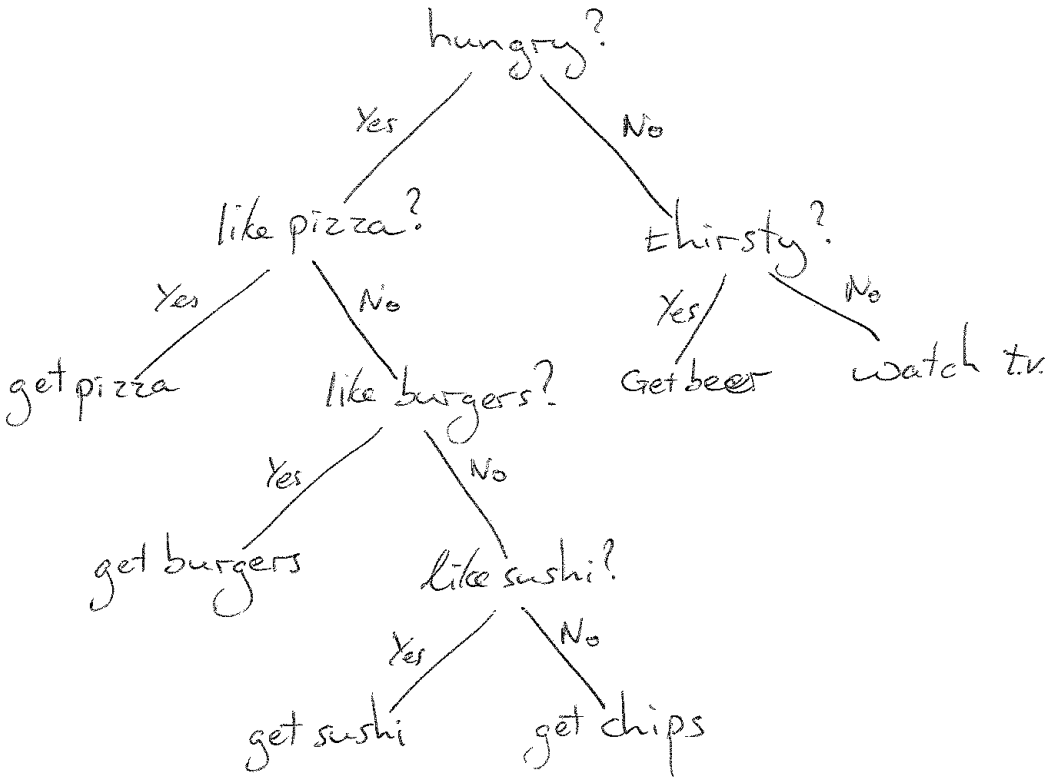
COMS 4030A

Adaptive Computation and Machine Learning

3. Decision Trees

A decision tree looks something like the following:



Every node in the tree is labelled by a question about an attribute from the given dataset. Start at the root node and consider the question at that node with respect to the given input. Follow the branch that corresponds to the answer. Continue to traverse the tree in this way until a leaf node is reached which gives us our decision.

We shall describe a way to construct a suitable decision tree for a dataset associated with a classification problem. That is, given a dataset in which every datapoint has an associated target classification, the objective is to construct a decision tree that predicts the correct class associated with every datapoint (or for an optimal number of datapoints).

The main issue when constructing a decision tree is choosing the attribute that should be used at a given node, that is, the 'question' to be asked at that node. Any attribute may be chosen at any node, however, we seek to minimise the number of questions required to obtain an outcome, which corresponds to finding a tree of minimum height.

For now, we assume that each attribute in the dataset uses discrete values (e.g., 0/1, T/F, Yes/No, A/B/C, or 0/1/2/3, etc.).

3.1. **Entropy.**

The method that we describe here for choosing an attribute is based on the idea of reducing the uncertainty in the dataset. The (Shannon) entropy of a probability distribution is used; we recall the definition here.

Let P denote a (discrete) probability distribution, i.e., $P = \{p_1, p_2, \ldots, p_k\}$ with $0 \leq p_i \leq 1$ for each i and $\sum_{i=1}^{k} p_i = 1$.

The **entropy** of P is:

$$H(P) = -\sum_{i=1}^{k} p_i \log_2 p_i$$

Examples:

If $P = \{0.5, 0.5\}$, then $H(P) = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$.

If $P = \{0.25, 0.75\}$, then $H(P) = -0.25 \log_2 0.25 - 0.75 \log_2 0.75 = 0.811$.

If $P = \{\frac{1}{3}, \frac{2}{3}\}$, then $H(P) = -\frac{1}{3} \log_2 \frac{1}{3} - \frac{2}{3} \log_2 \frac{2}{3} = 0.918$.

If $P = \{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\}$, then $H(P) = -\frac{1}{3} \log_2 \frac{1}{3} - \frac{1}{3} \log_2 \frac{1}{3} - \frac{1}{3} \log_2 \frac{1}{3} = 1.585$.

If $P = \{0.25, 0.25, 0.5\}$, then $H(P) = -0.25 \log_2 0.25 - 0.25 \log 0.25 - 0.5 \log_2 0.5 = 1.5$.

Entropy is a measure of the randomness, or uncertainty, in the system.

For example, take the case $P = \{p_1, p_2\}$ with $p_1 + p_2 = 1$. We can think of this as a probability distribution over two events, where p_1 is the probability of event 1 occurring and p_2 is the probability of event 2 occurring. If $p_1 = 0.5$ and $p_2 = 0.5$, then the entropy (i.e., uncertainty) of the system is at a maximum since event 1 and event 2 are equally likely. If $p_1 = 0.33$ and $p_2 = 0.67$, then the entropy is lower since we know that event 2 is more likely than event 1. If

$p_1 = 0.25$ and $p_2 = 0.75$, then the entropy is even lower since event 2 is even more likely than event 1 in this case. If $p_1 = 0$ and $p_2 = 1$ then the entropy is 0 since there is a 100% chance that event 2 will occur.

For cases with three events, i.e., $P = \{p_1, p_2, p_3\}$, with $\sum_{i=1}^{3} p_i = 1$, the maximum entropy occurs when all three events are equally likely, i.e., $p_1 = p_2 = p_3 = \frac{1}{3}$. For any other combination of p_1, p_2 and p_3 the entropy will be less. Note that the entropy for $\{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\}$ is greater than the entropy for $\{\frac{1}{2}, \frac{1}{2}\}$. Why do you think this is so?

3.2. **Constructing a decision tree from a dataset.**

Consider a dataset S consisting of an array X of datapoints and an array T of corresponding target values. Let F_1, $F_2, \ldots, F_m$ be the **attributes** of the datapoints in X. Attributes are sometimes referred to as **features**.

We seek to construct a decision tree for the given dataset. At each node of the tree a question is asked about one attribute of the data. We must decide which attribute to use at each node. The first step is to choose an attribute for the **root node**, which induces the next layer of nodes and branches. The same method is repeated at every new node until the decision tree is complete.

For each attribute F define the $\text{Gain}(S, F)$ as follows:

$$\text{Gain}(S, F) = \text{Entropy}(S) - \frac{1}{|S|} \sum_{f \in \text{values of } F} |S_f| \, \text{Entropy}(S_f),$$

where $\text{Entropy}(S)$ denotes the entropy of the probability distribution on the target values of S, and S_f is the subset of S containing of all datapoints for which attribute F has value f.

To determine the attribute to choose at the root node, we calculate $\text{Gain}(S, F)$ for each attribute F and choose the attribute for which $\text{Gain}(S, F)$ is maximum.

Observe that for each attribute F, the terms $\text{Entropy}(S)$ and $\frac{1}{|S|}$ in the formula for $\text{Gain}(S, F)$ are unaffected by the choice of F, so it is only necessary to calculate

$$\sum_{f \in \text{values of } F} |S_f| \, \text{Entropy}(S_f)$$

for each attribute F. In fact, the attribute that gives the *minimum* value for the above sum will give the maximum Gain.

Example: Consider the following dataset on students doing COMS3, where we have combined the datapoints and targets into one table, for convenience:

$$S: \left[\begin{array}{c|c|c|c} \text{COMS2} & \text{doing labs?} & \text{doing tuts?} & \text{target} \\ \hline A & N & Y & Pass \\ C & Y & N & Fail \\ C & N & Y & Pass \\ B & Y & Y & Pass \\ B & N & N & Fail \\ C & Y & N & Pass \\ A & N & N & Fail \\ B & Y & N & Pass \end{array}\right]$$

In the above dataset, the attributes of the data are: COMS2, doing labs?, and doing tuts?.

Recall that Entropy(S) is the entropy of the probability distribution on the targets of the dataset. There are only 2 possible targets: $Pass$ and $Fail$. Of the 8 entries, there are 5 $Pass$'s and 3 $Fail$'s. Thus, the probability distribution on the targets is $P = \{p_1, p_2\}$, where
p_1 = probability of $Pass = \frac{5}{8}$,
p_2 = probability of $Fail = \frac{3}{8}$.
Thus, Entropy(S) $= H(P) = -\frac{5}{8}\log_2\frac{5}{8} - \frac{3}{8}\log_2\frac{3}{8} = 0.954$.

To decide which attribute to use at the root node, we calculate Gain(S, COMS2), Gain(S, doing labs?) and Gain(S, doing tuts?).

• Gain(S, COMS2): attribute COMS2 has 3 values, which are A, B and C. Thus:

$$\sum_{f \in \text{values of COMS2}} |S_f|\,\text{Entropy}(S_f) = |S_A|\,\text{Entropy}(S_A) + |S_B|\,\text{Entropy}(S_B) + |S_C|\,\text{Entropy}(S_C).$$

Entropy(S_A) is calculated on the subset of S of all datapoints for which COMS2 is A:

$$S_A = \left[\begin{array}{c|c|c|c} \text{COMS2} & \text{doing labs?} & \text{doing tuts?} & \text{target} \\ \hline A & N & Y & Pass \\ A & N & N & Fail \end{array}\right]$$

Note that $|S_A| = 2$. Entropy(S_A) is calculated on the target column.
The probability distribution on the targets is $P = \{p_1, p_2\}$, where
p_1 = probability of $Pass = \frac{1}{2}$,

p_2 = probability of $Fail = \frac{1}{2}$.
Thus, Entropy$(S_A) = H(P) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$.

Entropy(S_B) is calculated on the subset of S of all datapoints for which COMS2 is B:

$$S_B = \left[\begin{array}{c|c|c|c} \text{COMS2} & \text{doing labs?} & \text{doing tuts?} & \text{target} \\ \hline B & Y & Y & Pass \\ B & N & N & Fail \\ B & Y & N & Pass \end{array}\right]$$

Note that $|S_B| = 3$. The probability distribution on the targets is $P = \{p_1, p_2\}$, where
p_1 = probability of $Pass = \frac{2}{3}$,
p_2 = probability of $Fail = \frac{1}{3}$.
Thus, Entropy$(S_B) = H(P) = -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} = 0.918$.

Entropy(S_C) is calculated on the subset of S of all datapoints for which COMS2 is C:

$$S_C = \left[\begin{array}{c|c|c|c} \text{COMS2} & \text{doing labs?} & \text{doing tuts?} & \text{target} \\ \hline C & Y & N & Fail \\ C & N & Y & Pass \\ C & Y & N & Pass \end{array}\right]$$

Note that $|S_C| = 3$. The probability distribution on the targets is $P = \{p_1, p_2\}$, where
p_1 = probability of $Pass = \frac{2}{3}$,
p_2 = probability of $Fail = \frac{1}{3}$.
Thus, Entropy$(S_C) = H(P) = -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} = 0.918$.

From the above calculations,

$$|S_A|\,\text{Entropy}(S_A) + |S_B|\,\text{Entropy}(S_B) + |S_C|\,\text{Entropy}(S_C) = 2(1) + 3(0.918) + 3(0.918) = 7.508$$

hence $\text{Gain}(S, \text{COMS2}) = 0.954 - \frac{1}{8}(7.508) = 0.016$.

• Gain$(S, \text{doing labs?})$: attribute 'doing labs?' has 2 values, which are Y and N. Thus:

$$\sum_{f \in \text{values of 'doing labs?'}} |S_f|\,\text{Entropy}(S_f) = |S_Y|\,\text{Entropy}(S_Y) + |S_N|\,\text{Entropy}(S_N).$$

Entropy(S_Y) is calculated on the subset of S of all datapoints for which 'doing labs?' is Y:

$$S_Y = \left[\begin{array}{c|c|c|c} \text{COMS2} & \text{doing labs?} & \text{doing tuts?} & \text{target} \\ \hline C & Y & N & Fail \\ B & Y & Y & Pass \\ C & Y & N & Pass \\ B & Y & N & Pass \end{array}\right]$$

Note that $|S_Y| = 4$. The probability distribution on the targets is $P = \{p_1, p_2\}$, where
p_1 = probability of $Pass = \frac{3}{4}$,
p_2 = probability of $Fail = \frac{1}{4}$.
Thus, Entropy(S_Y) $= H(P) = -\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4} = 0.811$.

Entropy(S_N) is calculated on the subset of S of all datapoints for which 'doing labs?' is N:

$$S_N = \left[\begin{array}{c|c|c|c} \text{COMS2} & \text{doing labs?} & \text{doing tuts?} & \text{target} \\ \hline A & N & Y & Pass \\ C & N & Y & Pass \\ B & N & N & Fail \\ A & N & N & Fail \end{array}\right]$$

Note that $|S_N| = 4$. The probability distribution on the targets is $P = \{p_1, p_2\}$, where
p_1 = probability of $Pass = \frac{1}{2}$,
p_2 = probability of $Fail = \frac{1}{2}$.
Thus, Entropy(S_N) $= H(P) = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$.

From the above calculations,

$$|S_Y|\,\text{Entropy}(S_Y) + |S_N|\,\text{Entropy}(S_N) = 4(0.811) + 4(1) = 7.244$$

hence $\text{Gain}(S, \text{doing labs?}) = 0.954 - \frac{1}{8}(7.244) = 0.0485$.

• Gain(S, doing tuts?): attribute 'doing tuts?' has 2 values, which are Y and N. Thus:

$$\sum_{f \in \text{values of 'doing tuts?'}} |S_f|\,\text{Entropy}(S_f) = |S_Y|\,\text{Entropy}(S_Y) + |S_N|\,\text{Entropy}(S_N).$$

Entropy(S_Y) is calculated on the subset of S of all datapoints for which 'doing tuts?' is Y:

$$S_Y = \left[\begin{array}{c|c|c|c} \text{COMS2} & \text{doing labs?} & \text{doing tuts?} & \text{target} \\ \hline A & N & Y & Pass \\ C & N & Y & Pass \\ B & Y & Y & Pass \end{array}\right]$$

Note that $|S_Y| = 3$. The probability distribution on the targets is $P = \{p_1, p_2\}$, where
p_1 = probability of $Pass = 1$,
p_2 = probability of $Fail = 0$.
Thus, Entropy$(S_Y) = H(P) = -\log_2 1 - 0\log_2 0 = 0$. (Note that we always take $0\log_2 0 = 0$.)

Entropy(S_N) is calculated on the subset of S of all datapoints for which 'doing labs?' is N:

$$S_N = \left[\begin{array}{c|c|c|c} \text{COMS2} & \text{doing labs?} & \text{doing tuts?} & \text{target} \\ \hline C & Y & N & Fail \\ B & N & N & Fail \\ C & Y & N & Pass \\ A & N & N & Fail \\ B & Y & N & Pass \end{array}\right]$$

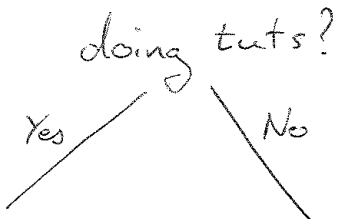
Note that $|S_N| = 5$. The probability distribution on the targets is $P = \{p_1, p_2\}$, where
p_1 = probability of $Pass = \frac{2}{5}$,
p_2 = probability of $Fail = \frac{3}{5}$.
Thus, Entropy$(S_N) = H(P) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.971$.

From the above calculations,

$$|S_Y|\,\text{Entropy}(S_Y) + |S_N|\,\text{Entropy}(S_N) = 3(0) + 5(0.971) = 4.855$$

hence $\text{Gain}(S, \text{doing tuts?}) = 0.954 - \frac{1}{8}(4.855) = 0.347$.

Thus, the maximum gain is obtained from using the attribute 'doing tuts?' at the root node, so the decision tree starts as follows:



Next, the process is repeated on the Y branch of the tree and also on the N branch of the tree. On the Y branch, the dataset is restricted to the datapoints in S for which 'doing tuts?' has value Y, i.e.,

COMS2	doing labs?	doing tuts?	target
A	N	Y	$Pass$
C	N	Y	$Pass$
B	Y	Y	$Pass$

On the above dataset, the target is always $Pass$ hence we can make this node a leaf node labeled with the decision $Pass$.

On the N branch, the dataset is restricted to the datapoints for which 'doing tuts?' has value N, i.e.,

COMS2	doing labs?	doing tuts?	target
C	Y	N	$Fail$
B	N	N	$Fail$
C	Y	N	$Pass$
A	N	N	$Fail$
B	Y	N	$Pass$

On the above dataset, there are both $Pass$ and $Fail$ targets, so another attribute is required at this node. That is, a question can be asked about either: COMS2 or doing labs?. Both $\text{Gain}(S, \text{COMS2})$ and $\text{Gain}(S, \text{doing labs?})$ are calculated for this dataset and whichever attribute gives maximum Gain is used as the attribute for this node. This process continues until all branches of the tree end in a leaf node, in which case the tree is complete.

A leaf node is created in two ways. Firstly, if all the datapoints in the current dataset have the same target, then a leaf node is created and labeled with that target. Secondly, if the current dataset has no more attributes left to test, then a leaf node is created. In this case, it may happen that the targets are not all the same. Then the label chosen for the leaf node is the most common target in the current dataset. (It may happen that there is a tie for the most common, and then the label must be chosen some other way, or the label can indicate a split decision. Another possibility is that the dataset is empty at this node, which can also be indicated by some special label.)

3.3. **ID3 Algorithm.**

The method for constructing a decision tree from a dataset using the entropy measure described above is called the ID3 algorithm. An outline of the algorithm is given below.

ID3 Algorithm

Given a dataset with attributes $F_1, \ldots, F_m$, which are all discrete (i.e., each attribute has a finite set of possible values) and discrete classification values for targets, a decision tree is built as follows:

Initially, **assign** S to be the whole dataset.

if all datapoints in S have the same target value, create a leaf node and label it with that target value,
else if there are no attributes left to test, create a leaf node and label it with the target value that is the most common in S
else find the feature $\hat{F}$ that maximises the information gain
for each value f of $\hat{F}$
add a new branch and node and calculate S_f by selecting only those datapoints with $\hat{F}$-value equal to f
remove attribute $\hat{F}$
recursively call the algorithm on dataset S_f.

EXERCISES

(1) Complete the decision tree in the above example.

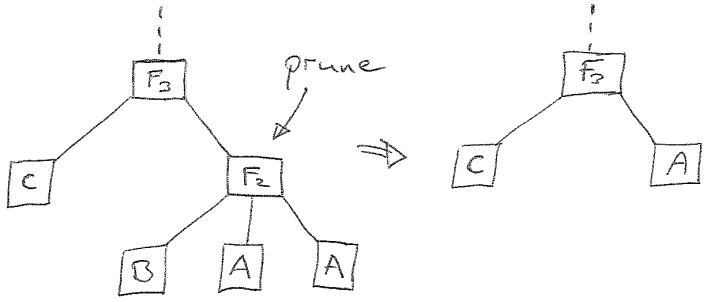
(2) Construct the decision tree for the following dataset using the entropy method described above:

$S:$

F_1	F_2	F_3	target
T	0	B	Yes
F	2	A	No
F	1	C	Yes
T	2	B	Yes
F	0	A	No
F	2	C	No
F	0	A	Yes
T	1	B	No
T	0	B	Yes
F	1	C	Yes
T	2	A	No
T	2	C	No

3.4. **Pruning a decision tree.**

It is very likely that overfitting will occur when constructing a decision tree using the ID3 algorithm. To avoid overfitting, the method of **pruning** a tree can be used. This means replacing any subtree with a leaf node and giving the node a label that is the most common target of all the datapoints that would be decided in this subtree. That is, consider every datapoint in the dataset whose prediction would be obtained by following a path that ends up in a leaf within this subtree. Of those datapoints choose the most common target.



To decide where to prune a tree, a validation set is used (as in the case of neural networks). The initial dataset is split into three subsets: training, validation and test datasets. (The usual splitting rules apply: approximately: $50\%, 25\%, 25\%$ or $60\%, 20\%, 20\%$).

The initial tree, say T, is built using the ID3 ALGORITHM on the training dataset. Using tree T, calculate the $\mathbf{error}_V$ on the validation set as follows: Each datapoint from the validation dataset is input into the existing tree and a class prediction is obtained. If the prediction differs from the target, this is a misclassification. The validation set error on T is then:

$$\text{error}_V(T) = \frac{\text{number of misclassifications}}{\text{number of datapoints}}$$

Choose any subtree that you are considering pruning and let T' be the pruned tree. As above, calculate the validation set error on the pruned tree, $\text{error}_V(T')$. If $\text{error}_V(T') < \text{error}_V(T)$ then replace the tree T by the pruned tree T'; otherwise keep the original tree T.

The above process can be repeated until there are no ways left to prune the tree that reduce the validation set error (or until the height of the tree is at a specified level). After pruning is complete, the test dataset is used to evaluate the tree.

3.5. **Gini Impurity.**

An alternative to using Entropy for constructing a decision tree for a given dataset is the Gini Impurity. For a probability distribution $P = \{p_1, p_2, \ldots, p_k\}$, the **Gini Impurity** of P is defined as:

$$\text{Gini}(P) = 1 - \sum_{i=1}^{k} p_i^2.$$

For example, if $P = \{\frac{1}{4}, \frac{3}{4}\}$, then $\text{Gini}(P) = 1 - ((\frac{1}{4})^2 + \frac{3}{4})^2) = 0.375$.

The minimum Gini Impurity occurs if the distribution is of the form $P = \{0, \ldots, 0, 1, 0, \ldots, 0\}$, in which case $\text{Gini}(P) = 0$. This occurs if all datapoints in the set have the same target. Such a set is called **pure**. As the target values become more mixed, the impurity increases.

The maximum Gini Impurity occurs if the distribution is of the form $P = \{\frac{1}{k}, \frac{1}{k}, \ldots, \frac{1}{k}\}$, in which case $\text{Gini}(P) = 1 - \sum_{i=1}^{k}(\frac{1}{k})^2 = 1 - \frac{k}{k^2} = \frac{k-1}{k}$. In this case, as k increases, the Gini Impurity tends to 1.

For a dataset S, the Gini Impurity of S, denoted Gini(S), is defined on its set of targets (as in the Entropy case) by finding the probabilities $p_1, \ldots, p_k$ for each of the possible target classes and obtaining the Gini Impurity for this distribution. For example, consider a dataset that looks like:

$$S: \left[\begin{array}{c|c|c|c|c} F_1 & F_2 & \ldots & F_m & \text{target} \\ \hline \vdots & \vdots & \vdots & \vdots & A \\ \vdots & \vdots & \vdots & \vdots & B \\ \vdots & \vdots & \vdots & \vdots & A \\ \vdots & \vdots & \vdots & \vdots & C \\ \vdots & \vdots & \vdots & \vdots & B \\ \vdots & \vdots & \vdots & \vdots & A \\ \vdots & \vdots & \vdots & \vdots & C \end{array}\right]$$

The probability distribution on the targets of S is $P = \{p_1, p_2, p_3\}$, where $p_1 = P(A) = \frac{3}{7}$, $p_2 = P(B) = \frac{2}{7}$, $p_3 = P(C) = \frac{2}{7}$. Then $\text{Gini}(S) = 1 - ((\frac{3}{7})^2 + (\frac{2}{7})^2 + (\frac{2}{7})^2) = 0.653$.

Constructing a decision tree using Gini Impurity is very similar to using Entropy. To choose an attribute for a given node, calculate the following Gain for each attribute F:

$$\text{Gain}(S, F) = \text{Gini}(S) - \frac{1}{|S|} \sum_{f \in \text{values of } F} |S_f| \, \text{Gini}(S_f).$$

Then, as in the ID3 algorithm, a branch is created for each value f of F and the algorithm recurses on each of the sets S_f until leaf nodes are created.

3.6. **Dealing with continuous variables.**

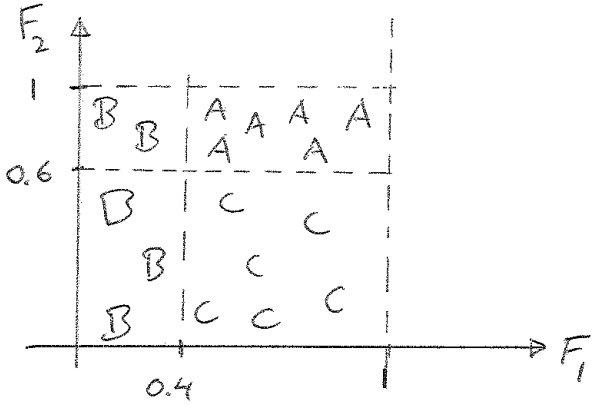
Suppose that a dataset has attributes that are not discrete, but continuous, i.e., they can take any real value in some range, as in the following example:

$$S: \left[\begin{array}{c|c|c} F_1 & F_2 & \text{target} \\ \hline 0.75 & 0.92 & A \\ 0.51 & 0.43 & B \\ 0.89 & 0.14 & C \\ 0.72 & 0.25 & A \\ \vdots & \vdots & \vdots \end{array}\right]$$

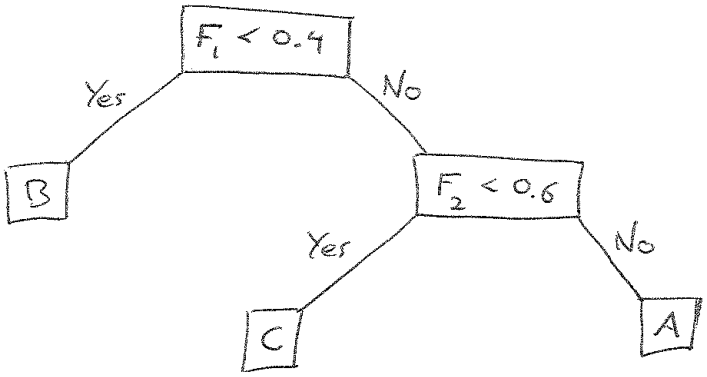
To build a decision tree for this dataset, the types of questions that are asked about attributes are of the form: $F_1 < 0.6$, or $F_2 < 0.3$, for example. The values 0.6 and 0.3 must be chosen somehow - they are called **split-points**.

There are different ways of choosing a split-point. One way is to consider a number of possible split-points at regular intervals, say at 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9. For each of these possible split-points, calculate the Gain (using the Entropy method) for that feature for each split point. Do this for each continuous attribute and also calculate the Gain for any discrete-valued attributes. Then choose the attribute/split point that gives the maximum Gain.

For example, consider a dataset that has two continuous attributes F_1 and F_2 in the range $[0, 1]$ and targets in the set $\{A, B, C\}$ that is illustrated in the following diagram:



A good decision tree for the above dataset would be:



When dealing with continuous-valued attributes, an attribute may be used more than once as a node question. For example, we may ask '$F_1 < 0.4$' at the root and then later on '$F_1 < 0.2$'. (In the discrete case, asking about the same attribute a second time is not useful since the data has been split into subsets where that attribute has same value.)

For the above reason, and also because of the need to search for good split points, it is often better to use random forests for datasets whose attributes are mainly continuous-valued (see below).

3.7. **Regression trees.**

A decision tree can also be constructed for a regression problem, that is, a dataset in which the targets are real values (as opposed to a classification problem where the targets are discrete values). An example of such a dataset is the following:

$$S = \left[\begin{array}{c|c|c|c|c} F_1 & F_2 & \dots & F_m & \text{target} \\ \hline \vdots & \vdots & \vdots & \vdots & 2.7 \\ \vdots & \vdots & \vdots & \vdots & 2.35 \\ \vdots & \vdots & \vdots & \vdots & 1.98 \\ \vdots & \vdots & \vdots & \vdots & -0.33 \\ \vdots & \vdots & \vdots & \vdots & -1,05 \\ \vdots & \vdots & \vdots & \vdots & 0.77 \\ \vdots & \vdots & \vdots & \vdots & 0.5 \end{array}\right]$$

In the case of a regression problem, the Entropy and Gini Impurity measures are not appropriate. Instead, the mean square error is used as a measure.

If the column of target values for dataset S is $\begin{bmatrix} t_1 \\ \vdots \\ t_N \end{bmatrix}$ then the **mean square error** of S, denoted $MSE(S)$, is obtained by first calculating the mean of the targets:

$$\text{mean of } S = \mu_S = \tfrac{1}{N}\sum_{i=1}^{N} t_i$$

and then computing

$$MSE(S) = \tfrac{1}{N}\sum_{i=1}^{N}(t_i - \mu_S)^2.$$

For example, for the above dataset, first calculate the mean:

$$\mu_S = \frac{1}{7}(2.7 + 2.35 + 1.98 - 0.33 - 1.05 + 0.77 + 0.5) = 0.989.$$

Then

$$\begin{aligned} MSE(S) &= \tfrac{1}{7}\left((2.7-\mu_S)^2+(2.35-\mu_S)^2+(1.98-\mu_S)^2+(-0.33-\mu_S)^2\right. \\ &\qquad \left.+(-1.05-\mu_S)^2+(0.77-\mu_S)^2+(0.5-\mu_S)^2\right) \\ &= 1.707. \end{aligned}$$

If all the target values in the dataset are close together, then MSE is small, and the more spread out the target values are, the greater MSE becomes. If the values are all the same then MSE is 0.

The algorithm for building the decision tree is the same as in the earlier cases, except that MSE is used instead of Entropy or Gini Impurity. For each attribute F, calculate the Gain as follows:

$$\text{Gain}(S,F) = MSE(S) - \frac{1}{|S|}\sum_{f\in\text{values of } F} |S_f|\, MSE(S_f).$$

Using the observation that $MSE(S_f) = \frac{1}{|S_f|}\sum_{i\in S_f}(t_i-\mu_{S_f})^2$, the above simplifies to:

$$\text{Gain}(S,F) = MSE(S) - \frac{1}{|S|}\sum_{f\in\text{values of } F}\ \sum_{i\in S_f}(t_i-\mu_{S_f})^2.$$

Next, select the attribute F for which the Gain is maximum.

Then, as in the ID3 algorithm, a branch is created for each value f of F and the algorithm recurses on each of the sets S_f until leaf nodes are created. When a leaf node is created, the label it is given is the average of all the values in the current dataset.

The above algorithms for using trees for classification and regression are often referred to as CART algorithms (Classification and Regression Trees).

Decision trees are applicable to various machine learning tasks; they are invariant under scaling and other transformations of feature values, are robust to inclusion of irrelevant features, and produce models that are explainable. However, decision trees are prone to overfit their training sets and learn irregular patterns, especially with deeper trees - they have low bias, and very high variance.

EXERCISES

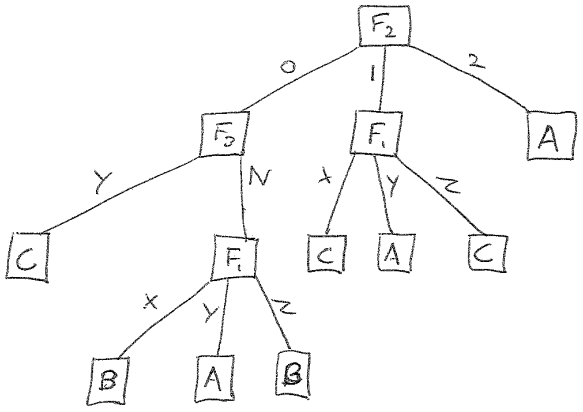
(1) Construct the decision tree for the following dataset using
   (a) the Entropy method and
   (b) the Gini Impurity method.

$S:$

F_1	F_2	F_3	target
a	0	Y	A
b	0	N	C
c	1	Y	B
b	1	Y	B
c	0	N	A
a	0	N	C
c	1	N	B
a	1	Y	A
b	0	Y	B
a	1	N	C

   (c) Find the predictions from the trees obtained in (a) and (b) for the following inputs:
   (i) $(b, 1, N)$
   (ii) $(c, 1, Y)$.

(2) Suppose that the decision tree below has been constructed using some training dataset.



Let the following dataset be the validation set.

Validation set :

F_1	F_2	F_3	target
X	2	Y	A
Y	0	N	B
Z	1	Y	C
Z	1	N	B
Y	2	N	A
Z	0	Y	C
X	2	N	A
Y	1	Y	A
Z	2	Y	C
X	0	N	C
Y	0	Y	B
X	0	N	B

(a) Compute the validation set error for the above decision tree.
(b) For each (non-leaf) node in the tree, prune the node and see if the validation set error decreases on the pruned tree.

(3) Suppose a decision tree has been constructed from some dataset. Then it can be used to make predictions for new datapoints that are not part of the original dataset. Suppose a datapoint has one of its attribute values missing. Describe how a prediction could still be made in this case. For example, for the tree in Question 2, suppose the input is: $[Z, ?, Y]$. What prediction could be obtained from the tree?

(4) A regression tree is to be constructed for the dataset below using MSE on the targets. The attributes F_1 and F_2 can take any real value between 0 and 1.
Determine what question should be asked at the root node of the tree.
When choosing a split-point value for attributes F_1 and F_2, just consider the values 0.25, 0.5 and 0.75.

$$S = \left[\begin{array}{c|c|c|c} F_1 & F_2 & F_3 & \text{target} \\ \hline 0.8 & 0.2 & a & 2.7 \\ 0.7 & 0.1 & b & 2.3 \\ 0.2 & 0.9 & c & 1.7 \\ 0.3 & 0.9 & b & 2.1 \\ 0.3 & 0.6 & a & 2.0 \\ 0.1 & 0.4 & c & 1.3 \\ 0.4 & 0.8 & b & 2.9 \\ 0.1 & 0.3 & c & 1.4 \end{array}\right]$$

3.8. **Random Forests.**

If one tree is not good enough for a given dataset, then perhaps a forest of trees is required. Random forests are a way of averaging multiple decision trees, trained on different parts of the same training set, with the goal of reducing the variance. This comes at the expense of a small increase in the bias and some loss of interpretability, but generally greatly boosts the performance in the final model.

Creating a random forest

Given a dataset (with targets), a random forest is created in the following way.

First, obtain a random sample of the data by randomly selecting a subset of datapoints.
Next, randomly select a subset of the features of the dataset.
Then construct a decision tree as in the above sections on the sample of data using only the selected features.
Repeat the above steps until the required number of decision trees have been created.

Notes on the above process:
The random sample of data must be selected using replacement of datapoints so that datapoints can be used in the creation a number of different trees. This is an example of **bootstrap sampling**. The process of training multiple models (trees in this case) on different bootstrap samples is known as **bootstrap aggregating**, or **bagging**.

By also choosing a random sample of features, it is expected that each tree in the forest is unique; this is intended to add diversity the forest. This diversity helps in reducing overfitting and improving the model's generalization capability.

The following hyperparameters are required in the creation of a random forest:
number of trees (or estimators) in the forest,
number of datapoints in a random sample,
number of features in a random subset of features,
criterion for spliting a node in each tree (Entropy/Gini impurity/MSE/random),
in regression case, how split points are chosen.

In addition, further hyperparameters may be included, such as:
maximum depth of each tree,
maximum number of leaf nodes in each tree,
minimum number of leaves required to split an internal node.

Applying a random forest

Once a random forest has been created for a given dataset, it can be used to make predictions on any given input. The predictions are made as follows.

For the given input, generate the corresponding output from each tree in the random forest. Then the final output is obtained by **majority voting** of the outputs in the case of classification and by **averaging** of the outputs in the case of regression.

Out-of-bag error

Random forests have an internal mechanism for estimating the model error without the need for a separate validation set. This is done using **out-of-bag samples**, or OOBs.

The out-of-bag error is calculated as follows.
Select a subset of datapoints from the dataset, called the **out-of-bag** dataset, or **OOB** dataset.
For each datapoint in the OOB dataset, find all trees in the random forest that were **not** created using that particular datapoint. Use that datapoint as input to each of these trees and obtain all the corresponding outputs.
Take the majority vote of these outputs and compare that with the true value of the datapoint, to obtain the OOB error for this datapoint.
The average of the OOB errors for all the datapoints gives the overall OOB error.