Suppose we have a domain with the following positive and negative examples:

- Positive example 1: \( \text{Length} = 3 \wedge \text{Gills} = \text{no} \wedge \text{Beak} = \text{yes} \wedge \text{Teeth} = \text{many} \)
- Positive example 2: \( \text{Length} = 4 \wedge \text{Gills} = \text{no} \wedge \text{Beak} = \text{yes} \wedge \text{Teeth} = \text{many} \)
- Positive example 3: \( \text{Length} = 5 \wedge \text{Gills} = \text{no} \wedge \text{Beak} = \text{yes} \wedge \text{Teeth} = \text{few} \)
- Positive example 4: \( \text{Length} = 5 \wedge \text{Gills} = \text{no} \wedge \text{Beak} = \text{yes} \wedge \text{Teeth} = \text{many} \)
- Positive example 5: \( \text{Length} = 5 \wedge \text{Gills} = \text{no} \wedge \text{Beak} = \text{yes} \wedge \text{Teeth} = \text{few} \)

Each internal node is labeled with a feature, and each edge is labeled with a value. Each leaf node is labeled with a class variable.

Decision tree: an example

For now assume categorical data.
Growing a feature tree


Algorithm GrowTree(D, F)

\begin{itemize}
  \item \textbf{Input}: data D; set of features F.
  \item \textbf{Output}: feature tree T with labelled leaves.
\end{itemize}

1. if Homogeneous(D) then return Label(D) \quad \text{// Homogeneous, Label; see text}
2. $S \leftarrow$ BestSplit(D, F) \quad \text{// e.g., BestSplit-Class (Algorithm 5.2)}
3. split D into subsets $D_i$ according to the literals in S;
4. for each $i$ do
5. \hspace{1em} if $D_i \neq \emptyset$ then $T_i \leftarrow$ GrowTree($D_i$, F) \text{ else } $T_i$ is a leaf labelled with Label($D_i$);
6. end
7. return a tree whose root is labelled with S and whose children are $T_i$.

Homogeneous(D) returns true if the instances in D are homogeneous enough to be labelled with a single label, and false otherwise;

Label(D) returns the most appropriate label for a set of instances D;

BestSplit(D, F) returns the best set of literals to be put at the root of the tree.

Learning decision trees

Finding optimal decision tree is NP-hard.

Rough idea: top down approach - at each step select the best attribute to perform a split on.

The best split method

Algorithm BestSplit-Class(D, F)

\begin{itemize}
  \item \textbf{Input}: data D; set of features F.
  \item \textbf{Output}: feature f to split on.
\end{itemize}

1. $I_{\text{min}} \leftarrow -1$;
2. for each $f \in F$ do
3. \hspace{1em} split D into subsets $D_1, \ldots, D_l$ according to the values $v_j$ of $f$;
4. \hspace{1em} if $\text{Imp}(D_1, \ldots, D_l) < I_{\text{min}}$ then
5. \hspace{2em} $I_{\text{min}} \leftarrow \text{Imp}(D_1, \ldots, D_l)$;
6. \hspace{2em} $f_{\text{best}} \leftarrow f$;
7. end
8. end
9. return $f_{\text{best}}$
Node impurity

Given a feature and one of its values, a corresponding split divides the data D into subsets, \( D_1, \ldots, D_i \), where \( i \) is the number of the feature.

- Ideal situation: \( D_1 \) contains all the positive or negative examples from \( D \). This is a pure split.
- Notation: Pos/Neg number of pos/neg examples in \( D_1 \)
- We will define impurity in terms of \( p = \text{Pos} / (\text{Pos} + \text{Neg}) \)

Desired properties from an impurity function:
- Maximal when \( p = \frac{1}{2} \)
- Minimal when \( p = 0 \) or \( p = 1 \)
- Symmetric in \( p, 1-p \)

Digression: information theory

I am thinking of an integer between 0 and 1,023. You want to guess it using the fewest number of questions.

Most of us would ask "is it between 0 and 512?"

This is a good strategy because it provides the most information about the unknown number.

Initially you need to obtain \( \log_2(1024) = 10 \) bits of information. After the first question you only need \( \log_2(512) = 9 \) bits.

Information and Entropy

By halving the search space we obtained one bit.

In general, the information associated with a probabilistic outcome:

\[
I(p) = -\log p
\]

Why the logarithm?

Assume we have two independent events \( x \) and \( y \). We would like the information they carry to be additive. Let's check:

\[
I(x, y) = -\log P(x, y) = -\log P(x)P(y)
\]

\[
= -\log P(x) - \log P(y) = I(x) + I(y)
\]

The Entropy, or information associated with a random variable \( X \):

\[
\text{Entropy}(X) = - \sum_x P(X = x) \log P(X = x)
\]
Entropy

For a Bernoulli random variable:

\[
Entropy(p) = -p \log p - (1 - p) \log(1 - p)
\]

Maximal when \( p = 1/2 \).
A split is most informative when \( p = 1 \) or \( p = 0 \)

Example

Consider again the data in Example 4.4. We want to find the best feature to put at the root of the decision tree. The four features available result in the following splits:

- **Length** = [3, 4, 5]
  - [2+, 0−] [1+, 3−] [2+, 2−]
- **Gills** = [yes, no]
  - [0+, 4−] [5+, 1−]
- **Beak** = [yes, no]
  - [5+, 3−] [0+, 2−]
- **Teeth** = [many, few]
  - [3+, 4−] [2+, 1−]

Let’s calculate the impurity of the first split. We have three segments: the first one is pure and so has entropy 0; the second one has entropy \(- (1/4) \log(1/4) - (3/4) \log(3/4) = 0.5 + 0.31 = 0.81\); the third one has entropy 1. The total entropy is then the weighted average of these, which is \( 2/10 \cdot 0 + 4/10 \cdot 0.81 + 4/10 \cdot 1 = 0.72 \).

Example

Similar calculations for the other three features give the following entropies:

- **Gills**: \( 4/10 \cdot 0 + 6/10 \cdot (-5/6) \log(5/6) = 0.39 \);
- **Beak**: \( 8/10 \cdot (-5/8) \log(5/8) + 2/10 \cdot 0 = 0.76 \);
- **Teeth**: \( 7/10 \cdot (-3/7) \log(3/7) + 4/10 \cdot (4/7) \log(4/7) + 3/10 \cdot (-2/3) \log(2/3) - (1/3) \log(1/3) = 0.97 \).

We thus clearly see that ‘Gills’ is an excellent feature to split on; ‘Teeth’ is poor; and the other two are somewhere in between.
Impurity of a split

When scoring the impurity of a split it is common to use the purity gain:

\[ \text{Imp}(D) - \text{Imp}\left(\{D_1, \ldots, D_i\}\right) \]

When using the entropy as an impurity function, this is called the information gain.

Issue with information gain

Favors attributes with many values
Such attribute splits D to many subsets, and if these are small, they will tend to be pure anyway
One way to fix this is the information gain ratio:

\[ \text{GainRatio}(D) = \frac{\text{IG}(D)}{\text{Entropy}(D)} \]

There are lots of different impurity functions and splitting criteria that are used.

When to Stop?

If we keep splitting until perfect classification we might over-fit.
Heuristics:
- Stop when splitting criterion is below some threshold
- Stop when number of examples in each leaf node is below some threshold

Alternative: prune the tree; potentially better than stopped splitting, since split may be useful at a later point.

Pruning

To avoid overfitting a tree is often pruned by considering the effect of removal of each subtree (computed on a separate pruning set)

Algorithm \text{PruneTree}(T, D)
\begin{algorithm}
\textbf{Input}: decision tree $T$; labelled data $D$.
\textbf{Output}: pruned tree $T'$.
\begin{algorithmic}
\State for every internal node $N$ of $T$, starting from the bottom do
\State $T_N$ --- subtree of $T$ rooted at $N$;
\State $D_N$ --- \{ $x \in D | x$ is covered by $N$\};
\State if accuracy of $T_N$ over $D_N$ is worse than majority class in $D_N$ then
\State replace $T_N$ in $T$ by a leaf labelled with the majority class in $D_N$;
\State end
\EndFor
\State return pruned version of $T$
\end{algorithmic}
\end{algorithm}
Cost sensitivity of splitting criteria

Suppose you have 10 positives and 10 negatives, and you need to choose between the two splits \([8+, 2-][2+, 8-]\) and \([10+, 6-][0+, 4-]\).

* You duly calculate the weighted average entropy of both splits and conclude that the first split is the better one.
* Just to be sure, you also calculate the average Gini index, and again the first split wins.
* You then remember somebody telling you that the square root of the Gini index was a better impurity measure, so you decide to check that one out as well. Lo and behold, it favours the second split...! What to do?

You then remember that mistakes on the positives are about ten times as costly as mistakes on the negatives.

* You're not quite sure how to work out the maths, and so you decide to simply have ten copies of every positive: the splits are now \([80+, 20-][20+, 80-]\) and \([100+, 60-][0+, 40-]\).
* You recalculate the three splitting criteria and now all three favour the second split.
* Even though you're slightly bemused by all this, you settle for the second split since all three splitting criteria are now unanimous in their recommendation.

Turns out that \(\sqrt{\text{Gini}}\) is insensitive to class imbalance.

Multi-class classification

Decision trees work for multi-class classification. All the impurity function we considered have a multi-class version. For example:

\[
\text{Entropy}(X) = - \sum_x P(X = x) \log P(X = x)
\]

Continuous variables

If variables are continuous we can bin them. Alternative: learn a simple classifier on a single dimension, e.g. find decision point, classifying all data to the left in one class and all data to the right in the other.

So, we choose a feature, a sign \("+/\)" and a threshold. This determines a half space to be +1 and the other half -1.
Regression trees

A leaf node of a regression tree will output a fixed value for the label $Y$.

Therefore, the objective of splitting is to obtain subsets that have low variance.

Assume a split partitions a set of target values $Y$ into sets $Y_1, \ldots, Y_l$.

The weighted variance is:

$$\text{Var}(Y) = \frac{1}{|Y|} \sum_{y \in Y} (y - \bar{y})^2$$

$$\bar{y} = \frac{1}{|Y|} \sum_{y \in Y} y$$

The weighted variance is:

$$\text{Var}(\{Y_1, \ldots, Y_l\}) = \sum_{j=1}^{l} \text{Var}(Y_j)$$

Example

Imagine you are a collector of vintage Hammond tonewheel organs. You have been monitoring an online auction site, from which you collected some data about interesting transactions:

<table>
<thead>
<tr>
<th>#</th>
<th>Model</th>
<th>Condition</th>
<th>Leslie</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>B3</td>
<td>excellent</td>
<td>no</td>
<td>4513</td>
</tr>
<tr>
<td>2.</td>
<td>T202</td>
<td>fair</td>
<td>yes</td>
<td>625</td>
</tr>
<tr>
<td>3.</td>
<td>A100</td>
<td>good</td>
<td>no</td>
<td>1051</td>
</tr>
<tr>
<td>4.</td>
<td>T202</td>
<td>good</td>
<td>no</td>
<td>270</td>
</tr>
<tr>
<td>5.</td>
<td>M102</td>
<td>good</td>
<td>yes</td>
<td>870</td>
</tr>
<tr>
<td>6.</td>
<td>A100</td>
<td>excellent</td>
<td>no</td>
<td>1770</td>
</tr>
<tr>
<td>7.</td>
<td>T202</td>
<td>fair</td>
<td>no</td>
<td>99</td>
</tr>
<tr>
<td>8.</td>
<td>A100</td>
<td>good</td>
<td>yes</td>
<td>1900</td>
</tr>
<tr>
<td>9.</td>
<td>E112</td>
<td>fair</td>
<td>no</td>
<td>77</td>
</tr>
</tbody>
</table>

Regression trees

Algorithm GrowTree($D, F$)

Input : data $D$; set of features $F$.
Output : feature tree $T$ with labelled leaves.

1. if Homogeneous($D$) then return Label($D$); /* Homogeneous, Label: see text */
2. $S \leftarrow \text{BestSplit}(D, F)$; /* e.g., \text{BestSplit-Class} (Algorithm 5.2) */
3. split $D$ into subsets $D_i$ according to the literals in $S$;
4. for each $i$ do
5. if $D_i \neq \emptyset$ then $T_i \leftarrow \text{GrowTree}(D_i, F)$ else $T_i$ is a leaf labelled with Label($D$);
6. end
7. return a tree whose root is labelled with $S$ and whose children are $T_i$.

Homogeneous($D$): determines if the variance in a node is sufficiently low
Label($D$): returns the average label
BestSplit($D, F$): uses variance as the impurity measure

Example

From this data, you want to construct a regression tree that will help you determine a reasonable price for your next purchase. There are three features, hence three possible splits:

Model = [A100, B3, E112, M102, T202]

[1051, 1770, 1900] [4513] [77] [870] [99, 270, 625]

Condition = [excellent, good, fair]

[1770, 4513] [270, 870, 1051, 1900] [77, 99, 625]

Leslie = [yes, no]

[625, 870, 1900] [77, 99, 270, 1051, 1770, 4513]

The means of the first split are 1574, 4513, 77, 870 and 331, and the weighted average of squared means is $3.21 \cdot 10^5$. The means of the second split are 3142, 1023 and 267, with weighted average of squared means $2.68 \cdot 10^6$; for the third split the means are 1132 and 1297, with weighted average of squared means $1.55 \cdot 10^7$. We therefore branch on Model at the top level. This gives us three single-instance leaves, as well as three A100s and three T202s.
Example

For the A100s we obtain the following splits:
Condition = [excellent, good, fair] [1770][1051, 1900]
Leslie = [yes, no] [1900][1051, 1770]
Without going through the calculations we can see that the second split results in less variance (to handle the empty child, it is customary to set its variance equal to that of the parent). For the T202s the splits are as follows:
Condition = [excellent, good, fair] [270][99, 625]
Leslie = [yes, no] [625][99, 270]

Comments on decision trees

- Fast training
- Established software (CART, C4.5)
- Users like them because they produce rules which can supposedly be interpreted (but decision trees are very sensitive with respect to training data)
- Able to handle missing data (how?)

When are decision trees useful?

Limited accuracy for problems with many features. Why?

Solution: Train multiple trees on subsets of the data. This approach is called random forests