Plan:

Introduction to decision trees
- Why are they useful?
- Classification and Regression.

The big question:
How do we learn a decision tree from data?

One problem: There are too many trees that explain the same data. So how do we pick the best one?

Short interlude on information theory.
ID3 (Iterative Dichotomiser 3) for learning classification trees.

Algorithms for learning regression trees
Random forests, boosting and other methods for improving tree learning.
Remember: Regression predicts a value, classification predicts a class.

Your data consists of features or attributes after training you want to implement some function:

\[ f: \text{Features} \rightarrow \begin{cases} \text{Class} & - \text{Classification} \\ \text{Value} & - \text{Regression} \end{cases} \]

---

**Decision Tree**: The formal definition is not so helpful. It is a rooted directed acyclic graph. The leaf nodes represent classes or values. Since feature values you follow a path from the root to a leaf.
Decision trees for **Regression**

Leaf nodes represent values.

![Decision Tree Diagram]

Trying to predict how many hours a week the US president plays golf.

---

Advantages of decision trees:

- We can represent non-linear functions

- Small trees are easy to understand and hence give explainable machine learning models.

**Explainability:**

Why did the machine learning model refuse me a bank loan?
For the moment we will stick to classification.

Given some data, how do we learn a decision tree?

Issues:

How do we avoid overfitting?

How do we learn small trees?

We want the ordering of the nodes to

Somehow represent the importance of the features.
Even for the boolean case the order you consider the features or variables gives you different size/shape trees.

These trees (should unless I made a mistake) represent the same function.
**Facts (Complexity theory)**

For boolean functions you can get an exponential blow up in the size for different orders.

- It is NP-hard to find an ordering that gives the smallest tree.

**Implications for machine learning**

- We have to find heuristics that find small trees.
- Small trees give good, easy to understand explanations.
- Small trees also avoid overfitting.

We will look at one approach using information theory.
Information theory

Revision \(\log_2 x = y \iff 2^y = x\)

\[
\log_2 4 = 2 \quad (2^2 = 4) \quad \log_2 1 = 0 \quad 2^0 = 1.
\]

In computer science we often measure capacity on a logarithmic scale.

2 G is twice the size of 1 G

Even though the number bits grow slightly (possible 0-1 patterns grow exponentially)

\[
2^{1024} \quad 2^{2048} \quad \text{not twice the size.}
\]

Information theory is a measure of information that takes into account the probability of events.

Don't forget

\[
\log_b(x) = \frac{\log(x)}{\log(b)}
\]
If I flip an unfair coin with probability 0.9 Heads and 0.1 Tails, then knowing the coin came up heads is not so much information.

If the probabilities are 0.5 for Heads and 0.5 for Tails, then knowing that it is Heads is much more information.

\( H \) is a function that takes probability distribution as input and returns the amount of information.

We want certain properties, for example:

\[
H\left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right) \leq H\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right) \leq H\left(\frac{1}{2}, \frac{1}{2}\right)
\]

\[
H\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right) = H\left(\frac{1}{3}, \frac{2}{3}\right) + \frac{1}{3} H\left(\frac{1}{2}, \frac{1}{2}\right)
\]

Mathematically, the only sensible measure is:

\[
H\left(P_1, \ldots, P_n\right) = - \sum_{i=1}^{n} P_i \cdot \log P_i
\]

It does not matter which base, but conventionally we take \( \log \) to the base 2.
If we have a fair coin

\[ H \left( \frac{1}{2}, \frac{1}{2} \right) = -\frac{1}{2} \log(1/2) - \frac{1}{2} \log(1/2) = -\frac{1}{2} \left( \log 2 \right) = -(\log 1 - \log 2) = -0 = 0. \]

We can now calculate the entropy of sets

We observe

Suppose we have 47 people

\( \text{23 own car} \)
\( \text{24 don't own car} \)

\[ H(Y) = -\frac{23}{47} \log_2 \frac{23}{47} - \frac{24}{47} \log_2 \frac{24}{47} = 0.9997 \]

When we are building decision trees we are

Looking out what to split on,

Suppose our data contains the attribute

Gender = Male, Female

Education = University, Other

(12 own, 11 own, 20 own)
We need to build a tree, but which one?

We can think of the nodes as sets of observations, and we need to make a decision on what to split.

We keep splitting until we get sets that only contain car owners or non-car owners.

To decide how to split we use information gain.
\[ H(Y) \rightarrow \text{Gender} \]

\[
\begin{align*}
H(Y|\text{male}) & \quad H(Y|\text{female}) \\
Y_{\text{male}} (20) & \quad Y_{\text{female}} (27) \\
Y_{\text{male}} &= \{ y | \text{y is male} \}
\end{align*}
\]

In our example we have 20 males, 27 females with conditional entropy you need to scale.

So

\[
\text{Information - gain (Y, Gender)} = H(Y) - \left( \frac{1}{1Y} H(Y_{\text{male}}) + \frac{1}{1Y} H(Y_{\text{female}}) \right)
\]

For example

\[
H(Y_{\text{male}}) = -\left( \frac{12}{20} \log_2 \frac{12}{20} + \frac{8}{20} \log_2 \frac{8}{20} \right)
\]

owns a car \quad \text{does not own a car}

\[
= 0.9710
\]

18 for female

\[
H(Y_{\text{female}}) = -\left( \frac{20}{47} \cdot 0.9710 + \frac{27}{47} \cdot 0.9751 \right)
\]

\[
= 0.0256
\]
If instead we split on education,

\[ H(Y) = \left( \frac{1}{14} \cdot H(Y_n) + \frac{1}{4} \cdot H(Y_0) \right) \]

\[ H(Y_n) = -\frac{22}{22} \cdot \log_2 \left( \frac{22}{22} \right) - \frac{20}{22} \cdot \log_2 \left( \frac{20}{22} \right) \]

\[ H(Y_0) = -\frac{2}{24} \cdot \log_2 \left( \frac{2}{24} \right) - \frac{4}{24} \cdot \log_2 \left( \frac{4}{24} \right) \]

So the information gain on splitting on education would be 0.553.

So we split on education first.

To work out the leaves you would have to look at for example the gender of the 2 university education car owners.
So far all of our features have been Categorical.

If you have numeric values then you want to build trees:

\[ x > v \]

- Yes
- No

But how do we decide which value \( v \)?

We could try all the possible values. This is not so efficient. One useful observation is that the value \( v \) must come from the training set.

If \( x \) is say the income we could sort the data:

\[
\begin{array}{cccc}
0 & - & No & Car \\
1 & - & Car & \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\
\end{array}
\]

\[ \text{Income} \]

You then pick values \( v \) between the groups of Car & non-car owner.

For simply use the information gain or the new feature.
103 is only one learning algorithm. There are others & other information gain measures. See for CART with Gini impurity or CH5 with the gain ratio.

It is easy to overfit with decision trees. This is because they are very expressive. You can limit the depth to avoid this.

**Extensions to avoid overfitting:**

**Bagging:** Split your sample into smaller subsets $B_1, B_2, B_3, \ldots, B_n$.

Train a tree on each & then build a classifier that returns the average.
Random forests

Instead of dividing up the training set divide the features into random smaller sets and train on each subset of features. The intuition is that you are trying to learn small trees that work on a small number of features.