Data Mining
Lecture 7: Decision Trees

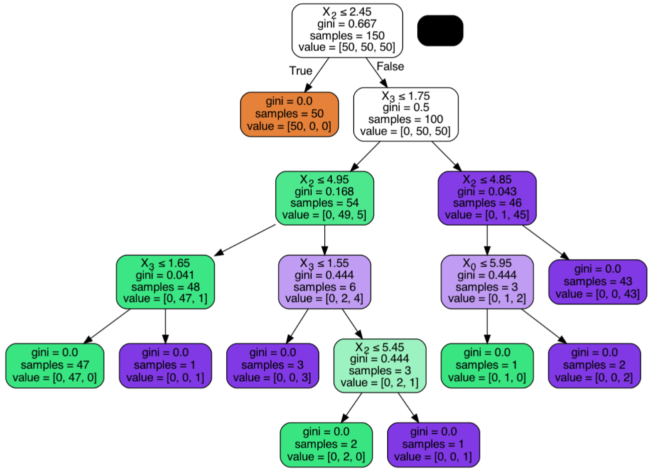
Jo Grundy

ECS Southampton

14^{th} March 2022

Decision Trees - Introduction

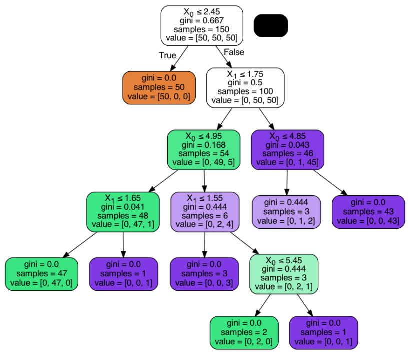
A decision tree is like a flow chart.
For example, the iris dataset:

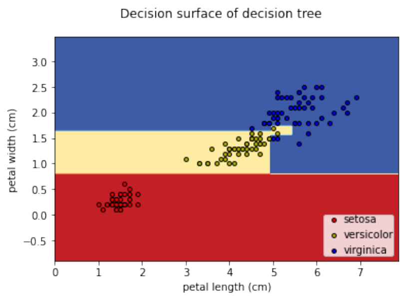


has four features, only three are used here, and one is only used once.

Decision Trees - Introduction

2D iris dataset:





Decision Trees - Introduction

Decision Trees can be 'hand crafted' by experts
They can also be built up using machine learning techniques

They are **interpretable**, it is easy to see how they made a certain decision

They are used in a wide range of contexts, for example:

- Medicine
- Financial Analysis
- Astronomy

Especially in medicine, the explicit reasoning in decision trees means experts can understand why the algorithm has made its decision.

Decision Trees - Introduction

Each node is a test, each branch is an outcome of the test
Can be used for tabulated data with a range of data types, e.g. numerical, categorical
ipynb demo

Decision Trees - Tree Growing Algorithms

There are a good number of algorithms to build decision trees

- CART - Classification And Regression Trees
- ID3 - Iterative Dichotomiser 3
- C4.5 - improved ID3
- C5.0 - improved C4.5
- CHAID - Chi - squared Automatic Interaction Detector

Decision Trees - CART

The CART algorithm was published by Breiman *et al* in 1984

- Find best split for each feature - minimises impurity measure
- Find feature that minimises impurity the most
- Use the best split on that feature to split the node
- Do the same for each of the leaf nodes

The CART algorithm depends on an impurity measure. It uses Gini impurity

Gini impurity measures how often a randomly chosen element from a set would be incorrectly labelled if it was randomly labelled according to the distribution of labels in the set. The probabilities for each label are summed up.

Decision Trees - CART

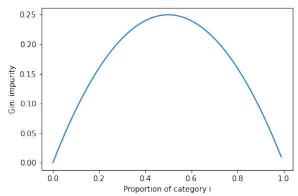
Gini Impurity (I_G) sums up probability of a mistake for each label:

$$mistake \quad probability = \sum_{k \neq i} p_k = 1 - p_i$$

For J classes:

$$Gini(p) = \sum_{i=1}^{J} p_i \sum_{k \neq i} p_k = \sum_{i=1}^{J} p_i(1 - p_i)$$

It reaches its minimum when all cases in the node fall into a single category



Decision Trees - CART

Maximum improvement in impurity found using the equation:

$$Gini(root) - (Gini(Left)\frac{n_L}{n} + Gini(Right)\frac{n_R}{n})$$

Where $Gini(root)$ is the impurity of the node to be split, $Gini(Left)$ and $Gini(Right)$ is the impurity of the left and right branches, n_L and n_R are the numbers in left and right branches.

Decision Trees - CART

Example:

Car Make	Type	Colour	Price	Mileage	Bought?
VW	Polo	Grey	£2000	82000	Yes
Ford	Fiesta	Purple	£1795	95000	Yes
Ford	Fiesta	Grey	£1990	90000	No
VW	Golf	Red	£1800	120000	Yes
VW	Polo	Grey	£900	150000	No
Ford	Ka	Yellow	£1400	100000	Yes

Can go through and calculate best split for each feature.

Decision Trees - CART

Calculate root node impurity:

$$Gini(root) = \frac{1}{3}(1-\frac{1}{3}) + \frac{2}{3}(1-\frac{2}{3}) = \frac{2}{9} + \frac{2}{9} = \frac{4}{9}$$

Impurity decrease (or Information gain if using Entropy) is thus:

$$I_G = Gini(root) - (Gini(Left)\frac{n_L}{n} + Gini(Right)\frac{n_R}{n})$$

Decision Trees - CART

Car Make	VW	Ford	4/9 - (2/9 + 2/9)
	Y, Y, N	Y, Y, N	= 0
Type	Golf	Not Golf	4/9 - (0 + 0.4)
	Y	Y, Y, Y, N, N	= 0.044
Colour	Grey	Not Grey	4/9 - (2/9 + 0)
	Y, N, N	Y, Y, Y	= 0.222
Price	> 1000	< 1000	4/9 - (0.267+ 0)
	Y, N, Y, Y, Y	N	= 0.178

This gives the first split. The same process is repeated for each impure node until all nodes are pure.

Decision Trees - CART

The second split has the following items:

Car Make	Type	Colour	Price	Mileage	Bought?
VW	Polo	Grey	£2000	82000	Yes
Ford	Fiesta	Grey	£1990	90000	No
VW	Polo	Grey	£900	150000	No

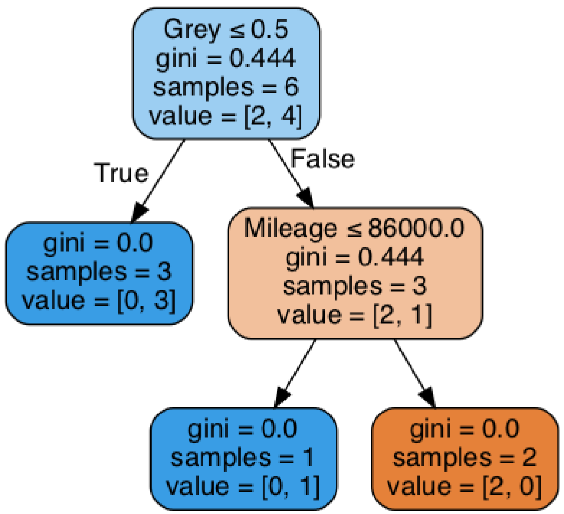
Decision Trees - CART

The second Split:

Car Make	VW	Ford	4/9 - (1/3 + 0)
	Y, N	N	= 0.111
Type	Polo	Not Polo	4/9 - (1/3 + 0)
	Y, N	N	= 0.111
Mileage	above 85,000	below 85,000	4/9 - (0 + 0)
	N, N	Y	= 0.444
Price	> 1000	< 1000	4/9 - (0+ 0)
	N,N	Y	= 0.444

The best splits both remove all impurity so we are done:

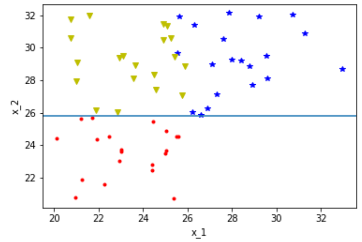
Decision Trees - CART

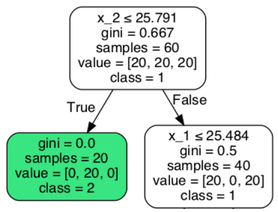


ipynb demo

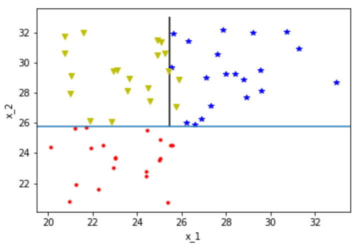
Decision Trees - CART

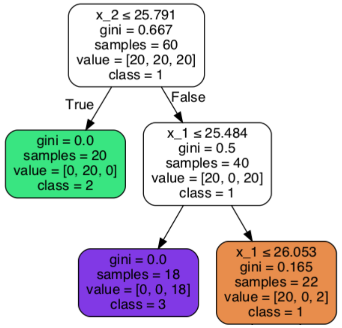
With three classes:



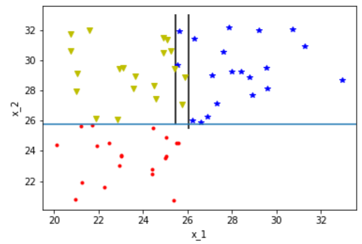


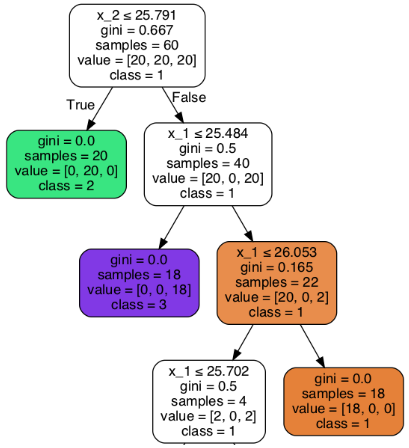
Decision Trees - CART



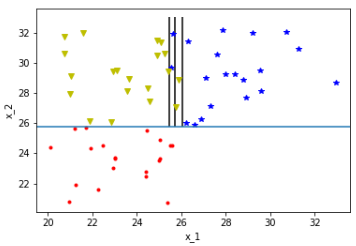


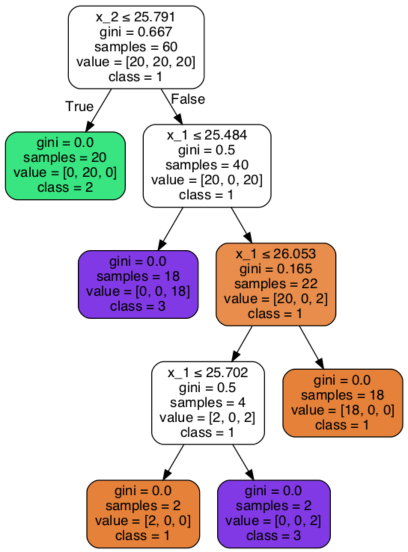
Decision Trees - CART





Decision Trees - CART





Overfitting..

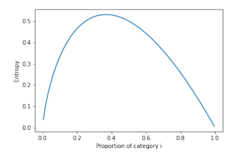
Decision Trees - ID3

Similar to CART, Iterative Dichotomy 3 (ID3) minimises entropy instead of Gini impurity

Entropy:

$$H(S) = \sum_{x \in X} -p(x) \log_2 p(x)$$

Where S is the data set, X is the set of classes in S, $p(x)$ is the proportion of the number of elements in class x to the number of elements in set S



Decision Trees - ID3

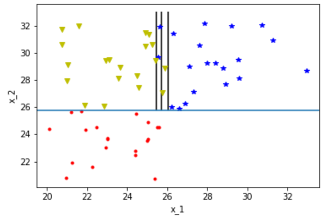
Information Gain is measured for a split along each possible attribute A

$$I_G(S, A) = H(S) - \sum_{x \in X} \frac{|S_A|}{|S|} H(S_A)$$

ID3 is very similar to CART, though doesn't technically support numerical values

Decision Trees - Pruning

Overfitting is a serious problem with Decision Trees



.. are four splits really required here?

The trees it creates are too complex.

One solution is **pruning**

This can be done in a variety of ways, including:

- Reduced Error Pruning
- Entropy Based Merging

Decision Trees - Reduced Error Pruning

Growing a decision tree fully, then removing branches without reducing predictive accuracy, measured using a *validation set*.

- Start at leaf nodes
- Look up branches at last decision split
- replace with a leaf node predicting the majority class
- If validation set classification accuracy is not affected, then keep the change

This is a simple and fast algorithm that can simplify over complex decision trees

Decision Trees - Entropy Based Pruning

Grow a decision tree fully, then

- Chose a pair of leaf nodes with the same parent
- What is the entropy gain from merging them?
- If lower than a threshold, merge nodes

This doesn't require additional data

Decision Trees - Missing Data

ID3 ignores missing data, CART generally puts them to the node that has the largest number of the same category

- You can assign a branch specifically to an unknown value
- You can assign it to the branch with the most of the same target value
- You can weight each branch using the known factors and put it in the most similar branch

Decision Trees - Regression

CART - Classification and Regression Trees How do we use decision trees for regression? i.e. to give numerical values rather than a classification.

Could use classification, but using each numerical value as a class..

Problems?

- How would you generalise?
- Loses all meaning of ordering, or similarity

Solution?

Use Variance instead of Gini or Entropy

Decision Trees - Regression

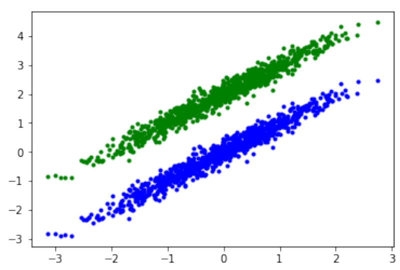
Maximise Variance Gain:

- Split on the feature values that give maximum gain in variance
- Should make similar numbers group together
- I. e. lower numbers on one side, higher on the other

Decision Trees - In General

There are problems with Decision Trees:

- Finding an optimal Tree is NP-complete
- They overfit, so don't generalise well - Hence need to prune
- Information Gain is biased to features with more categories
- Splits are axis aligned..



Decision Trees - Ensemble Methods

Bagging: Bootstrap aggregating
Uniformly sample initial dataset with replacement in to m subsets
For example:
if data set has 5 samples, $(s_1, s_2, s_3, s_4, s_5)$
make a whole bunch of similar data sets:

- $(s_5, s_2, s_2, s_1, s_5)$
- $(s_4, s_2, s_1, s_3, s_3)$
- $(s_2, s_5, s_3, s_1, s_1)$
- $(s_3, s_1, s_2, s_4, s_4)$

Train a different decision tree on each set
To classify, apply each classifier and chose the correct one by majority vote
If doing regression, take the mean of the values
This improves generalisation, as decreases variance without increasing bias

Decision Trees - Ensemble Methods

Boosting - Kearns and Valiant (1988):
"Can a set of weak learners create a single strong learner?"

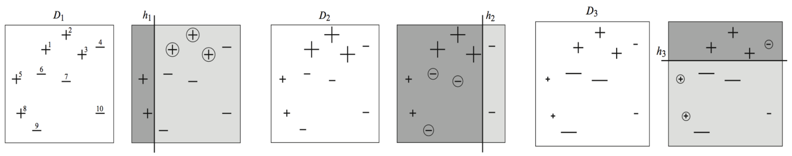
We make a weighted sum of very weak learners
- so long as they all learn different things then it works!
AdaBoost:

- Train a weak learner on one feature
- See what it does well on
- Weight the remaining data more
- repeat

This makes a series of weak learners that have learned how to use different features to discriminate between classes.

Decision Trees - Ensemble Methods

AdaBoost:



From Schapire and Freund, 2012)
Shortcomings are identified by high weight data points

Decision Trees - Ensemble Methods

Gradient boosting with trees - Friedman (1999):
Generalise Adaboost to Gradient boosting to handle any loss function
Shortcomings are where the residuals are larger
So fit a tree to the residuals:

- $x_1, y_q - f(x_1)$
- $x_2, y_q - f(x_1)$
- $x_3, y_q - f(x_1)$
- $\vdots$
- $x_n, y_q - f(x_n)$

more detail available at
`http://www.chengli.io/tutorials/gradient_boosting.pdf`

Decision Trees - Ensemble Methods

Random Forests

Apply bagging
but when learning the tree for each subset, chose the split by searching over a random sample of the features

Reduces overfitting

Decision Trees - Summary

Advantages:

- Interpretability
- Ability to work with numerical and categorical features
- Good with mixed tabular data

Disadvantages:

- Might not scale effectively for lots of classes
- Features that interact are problematic