"Tree" Predictive Models

Dr Liew How Hui

Jan 2021

Revision

What classifiers have we learned?

kNN (class), wkNN (kknn), Logistic Regression (LR), Multinomial LR/Softmax Regression (nnet), ANN(?, nnet, neuralnet, ANN2), ElasticNet (Binomial, Multinomial)

What regressors have we learned?

kNN (FNN), wkNN (kknn), LM/Bayesian LM/GLM (Gaussian, Poisson, Gamma, ...), ElasticNet (Gaussian, Poisson, Gamma, ...), Local Regression

Outline

1. **Theory and Terminologies**

2. Impurity Measures

3. Implementations

4. Practical Considerations

Theory

A *decision tree* algorithm is a non-parametric method for classification (and regression). It can handle categorical features and extra the predicate logic from features.

The assumption of decision tree is "the data must be clean and logical". Based on this assumption, the algorithm uses **information theory** to compress the data table into a "decision tree".

Theory (cont)

- For a tree classifier, the prediction is

$$h(x) = \textit{ClassificationTree}(x),$$

$$\mathbb{P}(Y = j | X = x) = \frac{1}{k} \sum_{x_i \in N(x)} I(y_i = j).$$

- For a tree regressor, the prediction is

$$h(x) = \textit{RegressionTree}(x)$$

Theory / Terminologies (cont)

According to
`https://en.wikipedia.org/wiki/Decision_tree`, a
decision tree is a "flowchart"-like structure in which each
internal node (the decision node) represents a "test" on
a "predictor" or an "attribute", each branch represents
the outcome of the test, and each **leaf node** represents a
class label (decision taken after computing all attributes).

Theory / Terminologies (cont)
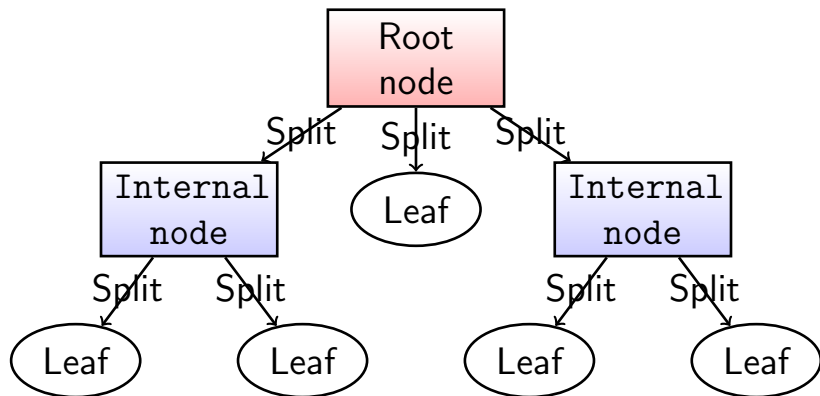
Theory / Terminologies (cont)

Decision trees have several nice advantages over kNN (as a non-parametric model):

1. once the tree is constructed, the training data does not need to be stored. Instead, we can simply store how many points of each label ended up in each leaf — typically these are pure so we just have to store the label of all points;

2. decision trees are very fast during test time, as test inputs simply need to traverse down the tree to a leaf (label);

3. decision trees require no metric because the splits are based on feature thresholds and not distances.

Theory / Terminologies (cont)

The goal in building a decision tree is to make sure it is "maximally compact" and "only has pure leaves".

A **consistent tree** is possible if no two input vectors have identical features but different labels.

Consider the data table with K classes:

$$S = \{(x_1, y_1), \cdots, (x_n, y_n)\}, y_i \in \{1, \ldots, K\}.$$

Example

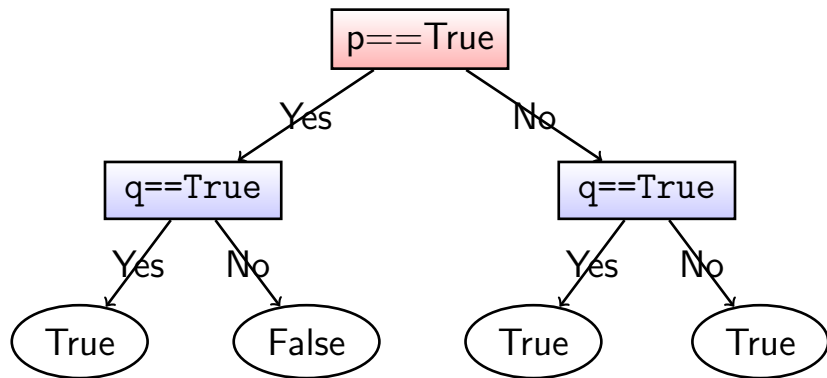Consider the truth table we learn in Discrete Maths:

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

where p are q are propositions.

The decision tree from this table will be **consistent** because no repeated inputs with different outputs.

Example (cont)

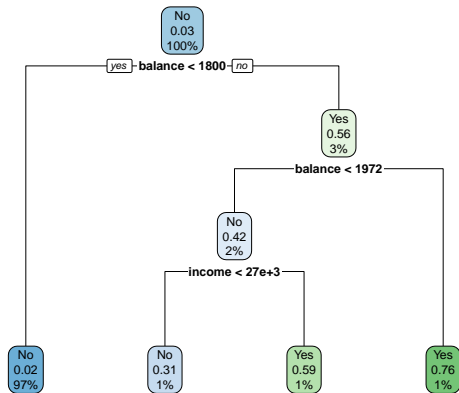Just encode all entries into a decision tree:

Problem with Brute-force: $p = 10$, each column 3 classes
$\rightarrow 3^{10} = 59049$ leaves!

Example (cont)

Example of Inconsistent Tree: ISLR's Default data set using all rows.

The recursive partitioning (rpart) tree (only 3% default=Yes):

Tree Construction

Brute-force tree construction do not give us "economical" decision tree!

Smart "splitting" of the data S should be used! The most popular way of splitting a data is to "split" based on a selected predictor (depending on whether it is categorical or numerical). The **splitting criteria** differ as follows:

predictors	qualitative (categorical)	quantitative (numerical)
splits	disjoint subsets of the predictor categories	disjoint ranges of the predictors value

Tree Construction (cont)

For the qualitative predictors, there are two types of splitting criteria:

1. *Binary split*: Divides values into two subsets.
2. *Multi-way split* / *L-way split*: Use as many partitions as distinct values.

For the splitting, we know what we don't want to have a uniform distribution: $p_1 = p_2 = \cdots = p_K = \frac{1}{K}$. This is the worst case since each leaf is equally likely because the prediction is going to be a random guessing.

Outline

1. Theory and Terminologies

2. **Impurity Measures**

3. Implementations

4. Practical Considerations

"Impurity" Measures

"Impurity" measures are used to choose the "best" predictor to perform "splitting" for the "decision-tree construction algorithm". The "impurity" measures generally measure the homogeneity of the target variable within the subsets. We introduce them following `https://en.wikipedia.org/wiki/Decision_tree_learning`:

- Gini Impurity Index (default in 'rpart')
- Entropy and Information Gain
- Variance Reduction (?)
- Deviance (using in 'tree')
- Gain Ratio

Gini Impurity (Index)

Gini impurity is used by the *CART (classification and regression tree)* algorithm for classification trees. It is defined as a measure of total variance across the K classes:

$$G(S) = \sum_{j=1}^{K} p_j(1 - p_j) = 1 - \sum_{j=1}^{K} p_j^2, \quad p_j = \frac{|S_j|}{|S|} \quad (1)$$

where p_j is the proportion of observations in S that belong to class j and $S_j = \{(x, y) \in S : y = j\} \subseteq S$ (all inputs with labels j) such that $S_i \cap S_k = \emptyset$ when $i \neq k$ and $S = S_1 \cup \cdots \cup S_K$.

Gini Impurity (cont)

Gini impurity will be maximised when observations are heterogeneous, i.e. **impure**. For example, when all classes are **uniformly distributed**, i.e. $p_1 = \cdots = p_K = \frac{1}{K}$, then

$$G(S) = 1 - \sum_{k=1}^{K} \frac{1}{K^2} = 1 - \frac{1}{K}.$$

Gini impurity will be minimised when observations are homogenous, i.e. **pure**. For example, when one class is having all the observations or is empty, then

$$G(S) = 1 - 1 = 0.$$

Gini Impurity (cont)

The *Gini impurity of a tree* is defined as

$$G^T(\underbrace{\{S_1, \cdots, S_L\}}_{S^A}) = \sum_{i=1}^{L} \frac{|S_i|}{|S|} G^T(S_i). \qquad (2)$$

In the splitting of a decision tree, we want a branch with more "pure" nodes, so the predictor with the **lowest** Gini impurity is selected as the splitting predictor and the tree algorithm is repeated for the subsequent nodes.

Gini Impurity (cont)

Example 5.2.1 (Tennis / Golf Example) [Mit97, §3.4.2].

Day	Outlook	Temperature	Humidity	Wind	Play
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Compute the **Gini impurity** of each feature (excluding Day) with respect to the output "Play".

Gini Impurity (cont)

Example 5.2.1 (cont)

Let S_{Weak} and S_{Strong} be the split of Wind is "Weak" and "Strong" respectively, $S^{Wind} = \{S_{Weak}, S_{Strong}\}$, $S_{Weak,Yes} = S_{Weak} \cap \{Play = Yes\}$, ..., and $S_{Strong,No} = S_{Strong} \cap \{Play = No\}$, etc.

$$\mathbb{P}(S_{Weak}) = \frac{8}{14}; \quad \mathbb{P}(S_{Strong}) = \frac{6}{14}$$

$$\mathbb{P}(S_{Weak,Yes}) = \frac{6}{8}; \quad \mathbb{P}(S_{Weak,No}) = \frac{2}{8} \Rightarrow G(S_{Weak}) = 1 - (\frac{6}{8})^2 - (\frac{2}{8})^2 = 0.375$$

$$\mathbb{P}(S_{Strong,Yes}) = \frac{3}{6}; \quad \mathbb{P}(S_{Strong,No}) = \frac{3}{6} \Rightarrow G(S_{Strong}) = 1 - (\frac{3}{6})^2 - (\frac{3}{6})^2 = 0.5$$

$$G(S^{Wind}) = \frac{8}{14}(0.375) + \frac{6}{14}(0.5) = 0.4286$$

Gini Impurity (cont)

Example 5.2.1 (cont): Gini impurity of Humidity:

- $\mathbb{P}(S_{High}) = \dfrac{7}{14}; \quad \mathbb{P}(S_{Normal}) = \dfrac{7}{14}$

- $\mathbb{P}(S_{High,Yes}) = \dfrac{3}{7}; \quad \mathbb{P}(S_{High,No}) = \dfrac{4}{7} \Rightarrow$

  $G(S_{High}) = 1 - (\dfrac{3}{7})^2 - (\dfrac{4}{7})^2 = 0.4898$

- $\mathbb{P}(S_{Normal,Yes}) = \dfrac{6}{7}; \quad \mathbb{P}(S_{Normal,No}) = \dfrac{1}{7} \Rightarrow$

  $G(S_{Normal}) = 1 - (\dfrac{6}{7})^2 - (\dfrac{1}{7})^2 = 0.2449$

- $G(S^{Humidity}) = \dfrac{7}{14}(0.4898) + \dfrac{7}{14}(0.2449) = 0.3674$

Gini Impurity (cont)

Example 5.2.1 (cont)

Try and see if you can find

$G(S^{Temperature}) = 0.4405$

$G(S^{Outlook}) = 0.3429$

Entropy and Information Gain

Information gain is used by the ID3, C4.5 and C5.0 tree-generation algorithms. Information gain is based on the concept of Entropy and information content from information theory [CT06]:

$$H(S) = -\sum_{i=1}^{K} p_i \log_2 p_i \tag{3}$$

where p_i is the proportion of observations in S that belong to class $i \in \{1, \cdots, K\}$ and S a collection with K number of classes.

Entropy and Information Gain (cont)

The entropy over tree ($Values(A) = \{1, ..., L\}$) is

$$H^T(\underbrace{\{S_1, \cdots, S_L\}}_{S^A}) = \sum_{i \in Values(A)} \frac{|S_i|}{|S|} H(S_i). \qquad (4)$$

The *information gain* measures how well *a given predictor separates the training data S according to their response variable classification*:

$$IG(S^A) = H(S) - H^T(S^A) \qquad (5)$$

where S_i is the subset of S for which predictor A is belonging to class i.

Features that perfectly partition should give **maximal information gain** or **minimal entropy**.

Entropy and Information Gain (cont)

Example 5.2.2 (tennis example)

Compute the **information gain** of all features (note: Day is not a feature).

The **overall entropy (of the target output) before splitting**:

$$H(S) = -(\frac{9}{14} \log_2 \frac{9}{14} + \frac{5}{14} \log_2 \frac{5}{14}) = \underline{0.9403}$$

Entropy and Information Gain (cont)

Example 5.2.2 (cont)

The entropy and information gain **after splitting on Wind** are

- $\mathbb{P}(S_{Weak,Yes}) = \frac{6}{8}; \quad \mathbb{P}(S_{Weak,No}) = \frac{2}{8} \Rightarrow$
  $H(S_{Weak}) = -(\frac{6}{8}\log_2\frac{6}{8} + \frac{2}{8}\log_2\frac{2}{8}) = 0.8113$

- $\mathbb{P}(S_{Strong,Yes}) = \frac{3}{6}; \quad \mathbb{P}(S_{Strong,No}) = \frac{3}{6} \Rightarrow$
  $H(S_{Strong}) = -(\frac{3}{6}\log_2\frac{3}{6} + \frac{3}{6}\log_2\frac{3}{6}) = 1$

- $H(S^{Wind}) = \frac{8}{14}H(S_{Weak}) + \frac{6}{14}H(S_{Strong}) = 0.8922$
  $IG(S^{Wind}) = H(S) - H(S^{Wind}) = 0.9403 - 0.8922 = 0.0481$

Entropy and Information Gain (cont)

Example 5.2.2 (cont)

The information gain **after splitting on** Humidity:

- $\mathbb{P}(S_{High,Yes}) = \dfrac{3}{7}, \quad \mathbb{P}(S_{High,No}) = \dfrac{4}{7}$
  $\Rightarrow H(S_{High}) = 0.9852$

- $\mathbb{P}(S_{Normal,Yes}) = \dfrac{6}{7}, \quad \mathbb{P}(S_{Normal,No}) = \dfrac{1}{7}$
  $\Rightarrow H(S_{Normal}) = 0.5917$

- $H(S^{Humidity}) = \frac{7}{14}H(S_{High}) + \frac{7}{14}H(S_{Normal}) = 0.7885$
  $IG(S^{Humidity}) = 0.9403 - 0.7885 \approx 0.1518$

Gini Impurity (cont)

Example 5.2.2 (cont)

Try and see if you can find

$H(S^{Temperature}) = 0.9111$

$IG(S^{Temperature}) = 0.0292$

$H(S^{Outlook}) = 0.6936$

$IG(S^{Outlook}) = 0.2467$

Entropy and Information Gain (cont)

A simple R implementation is listed below.

```
entropy = function(S) {
    counts = table(S)
    p = counts/sum(counts)
    return(-sum(ifelse(p==0,0,p*log2(p))))
}

# A = split_attribute_name, R = target / response name
info_gain = function(d.f,A,R="target") {
    HS = entropy(d.f[R])
    counts = table(unlist(d.f[A]))
    p = counts/sum(counts)
    HSi = unlist(lapply(split(d.f, d.f[A]), function(s) {
                    entropy(s[R]) }))
    return(HS - sum(p*HSi))
}

d.f = read.csv('playtennis.csv')
print(entropy(d.f$playtennis))
print(info_gain(d.f, 'wind', 'playtennis'))
```

Entropy and Information Gain (cont)

A simple Python implementation is listed below.

```
import numpy as np, pandas as pd

def entropy(S):
    _,counts = np.unique(S, return_counts=True)
    p = counts/np.sum(counts)
    return -sum(p[i]*np.log2(p[i]) for i in range(len(p))
                                   if p[i]!=0.0)

# A = split_attribute_name, T = target / response name
def info_gain(df,A,T="target"):
    HS = entropy(df[T])
    valsA,counts = np.unique(df[A].tolist(), return_counts=True)
    p = counts/np.sum(counts)
    HSi = [entropy(df.where(df[A]==i).dropna()[T]) for i in valsA]
    return HS - sum(p*np.array(HSi))

df = pd.read_csv('playtennis.csv')
print(entropy(df['playtennis']))
print(info_gain(df, 'wind', 'playtennis'))
```

Variance Reduction

Variance reduction [BFSO84] is often employed in cases where the target variable is continuous (regression tree): The variance reduction of a node N is defined as the total reduction of the variance of the target variable x due to the split at this node:

$$I_V(N) = \frac{1}{|S|^2} \sum_{i \in S} \sum_{j \in S} \frac{1}{2}(x_i - x_j)^2 - \left(\frac{1}{|S_t|^2} \sum_{i \in S_t} \sum_{j \in S_t} \frac{1}{2}(x_i - x_j)^2 + \frac{1}{|S_f|^2} \sum_{i \in S_f} \sum_{j \in S_f} \frac{1}{2}(x_i - x_j)^2 \right)$$

where S, S_t, and S_f are the set of presplit sample indices, set of sample indices for which the split test is true, and set of sample indices for which the split test is false, respectively.

Gain Ratio

There is a natural bias in information gain measure that **favours** predictors with large number of values. For example, use Day as a predictor to calculate the information gain of "tennis" data table.

- For $i = 1, 2, 6, 8, 14$, $\mathbb{P}(S_{D_i, Yes}) = 0$, $\mathbb{P}(S_{D_i, No}) = 1$
  $\Rightarrow H(S_{D_i}) = -(0 \times \log_2 0 + 1 \times \log_2 1) = 0$
- For $i = 2, 3, 45, 7, 9, 10, 11, 12, 13$, $\mathbb{P}(S_{D_i, Yes}) = 1$,
  $\mathbb{P}(S_{D_i, No}) = 0$
  $\Rightarrow H(S_{D_i}) = -(1 \times \log_2 1 + 0 \times \log_2 0) = 0$
- $IG(S^{Day}) = 0.9403 - \dfrac{1}{14} \times 0 - \cdots - \dfrac{1}{14} \times 0 = 0.9403$.

Gain Ratio (cont)

If "Day" is an attribute, it will win out as the root but it is useless in prediction just like a person's id number is useless in identifying the performance of a person.

Gain ratio is a modification of information gain that **reduces its bias** on highly branching predictors.

The *intrinsic information* of A, is the entropy of the "splitting factor":

$$H(A) = -\sum_{i=1}^{K} \frac{|A_i|}{|A|} \log_2 \frac{|A_i|}{|A|}. \tag{6}$$

Gain Ratio (cont)

The gain ratio, $R(S^A)$ is defined as

$$R(S^A) = \frac{IG(S^A)}{H(A)} \tag{7}$$

where $IG(S^A)$ is the information gain (5).

The predictor with the maximum gain ratio is selected as the splitting predictor and the algorithm is similar to the information gain.

Gain Ratio (cont)

Example Final Exam Jan 2019, Q2(b)

Gain ratio is a modification of information gain. Discuss on why and how the modification is done on information gain. (2 marks)

Solution: Gain ratio is a modification of information gain that **reduces its bias on highly branching predictors**. It takes into account the number and size of branches when choosing a predictor. This is done by normalising information gain by the intrinsic information of a split, which is defined as the information need to determine the branch to which an observation belongs.

Gain Ratio (cont)

Example 5.2.5: Compute the **gain ratio** of the output "Play" against other features (excluding Day).

Solution: The gain ratio of Wind:

- $H(S) = 0.9403$
- $IG(S^{Wind}) = 0.0481$
- $H(Wind) = -(\frac{8}{14} \log_2 \frac{8}{14} + \frac{6}{14} \log_2 \frac{6}{14}) = 0.9852$
- $R(S^{Wind}) = \frac{IG(S^{Wind})}{H(Wind)} = 0.0488$

Gain Ratio (cont)

The calculation of gain ratio of Humidity:

- $IG(S^{Humidity}) =$
  $0.9403 - \dfrac{7}{14} \times 0.9852 - \dfrac{7}{14} \times 0.5917 = 0.1519$
- $H(Humidity) = -(\dfrac{7}{14} \log_2 \dfrac{7}{14} + \dfrac{7}{14} \log_2 \dfrac{7}{14}) = 1$
- $R(S^{Humidity}) = 0.1519$

Try to use the steps above to check that

$R(S^{Temperature}) = 0.0188$

$R(S^{Outlook}) = 0.1564.$

Gain Ratio (cont)

Final Exam Jan 2019, Q2(a)

State definition for the following terms:

- (i)   Entropy   (1 mark)
- (ii)   Information gain   (1 mark)
- (iii)   Intrinsic information   (1 mark)
- (iv)   Gain ratio   (1 mark)
- (v)   Gini impurity   (1 mark)

Gain Ratio (cont)

Solution:

(i) Entropy characterizes the impurity in a collection of data.

(ii) Information gain measures how well a given split separates the data according to their response variable classification.

(iii) Intrinsic information represents the potential information generated by splitting the data into different partitions.

(iv) Gain ratio is the normalisation of information gain by intrinsic information.

(v) Gini impurity measures total variance across the classes.

Gain Ratio (cont)

Example (SRM-02-18, Q9)

A classification tree is being constructed to predict if an insurance policy will lapse. A random sample of 100 policies contains 30 that lapsed. Consider two splits:

Split 1: One node has 20 observations with 12 lapses and one node has 80 observations with 18 lapses.

Split 2: One node has 10 observations with 8 lapses and one node has 90 observations with 22 lapses.

Gain Ratio (cont)

Example (SRM-02-18, Q9) cont.

The total Gini impurity after a split is the weighted average of the Gini impurity at each node, with the weights proportional to the number of observations in each node.

The total entropy after a split is the weighted average of the entropy at each node, with the weights proportional to the number of observations in each node.

Gain Ratio (cont)

Example (SRM-02-18, Q9) cont.

Determine which of the following statements is/are true?

I. Split 1 is preferred based on the total Gini impurity.

II. Split 1 is preferred based on the total entropy.

III. Split 1 is preferred based on having fewer classification errors.

(A) I only

(B) II only

(C) III only

(D) I, II, and III

(E) The correct answer is not given by (A), (B), (C), or (D).

Gain Ratio (cont)
Solution:

$$
\begin{aligned}
\text{Gini(Split 1)} &= \frac{20}{100}\left[1-(\frac{12}{20})^2-(\frac{8}{20})^2\right] + \frac{80}{100}\left[1-(\frac{18}{80})^2-(\frac{62}{80})^2\right] \\
&= 0.2 \times 0.48 + 0.8 \times 0.34875 = 0.375 \\
\text{Gini(Split 2)} &= \frac{10}{100}\left[1-(\frac{8}{10})^2-(\frac{2}{10})^2\right] + \frac{90}{100}\left[1-(\frac{22}{90})^2-(\frac{68}{90})^2\right] \\
&= 0.1 \times 0.32 + 0.9 \times 0.3693827 = 0.3644
\end{aligned}
$$

$\Rightarrow$ **Lower Gini impurity** is better. $\therefore$ Split 2 is preferred. I is wrong.

$$
\begin{aligned}
\text{H(Split 1)} &= -\frac{20}{100}\left[\frac{12}{20}\log_2\frac{12}{20}+\frac{8}{20}\log_2\frac{8}{20}\right] - \frac{80}{100}\left[\frac{18}{80}\log_2\frac{18}{80}+\frac{62}{80}\log_2\frac{62}{80}\right] \\
&= -0.2 \times (-0.9710) - 0.8 \times (-0.7692) = 0.8096 \\
\text{H(Split 2)} &= -\frac{10}{100}\left[\frac{8}{10}\log_2\frac{2}{10}+\frac{2}{10}\log_2\frac{2}{10}\right] - \frac{90}{100}\left[\frac{22}{90}\log_2\frac{22}{90}+\frac{68}{90}\log_2\frac{68}{90}\right] \\
&= -0.1 \times (-0.7219) - 0.9 \times (-0.8024) = 0.7944
\end{aligned}
$$

$\Rightarrow$ **Smaller entropy** (vs larger information gain) is better. $\therefore$ Split 2 is preferred. II is wrong.
Remark: SRM Answer is different because they use natural log.

For Split 1, there are 8+18=26 errors while for Split 2 there are 2+22 = 24 errors. With fewer errors, Split 2 is preferred.

Answer: (E)

Gain Ratio (cont)

Final Exam Jan 2019, Q2(d)

A classification tree is being constructed to predict if an insurance policy will lapse. A random sample of 1000 policies contains 320 that lapsed. You are considering two splits:

Split 1: One node has 260 observations with 140 lapses and one node has 740 observations with 180 lapses.

Split 2: One node has 120 observations with 70 lapses and one node has 880 observations with 250 lapses.

Gain Ratio (cont)

Final Exam Jan 2019, Q2(d) cont.

Determine which split is preferred based on

   (i)   Gini impurity   (4 marks)

  (ii)   Information gain   (4 marks)

 (iii)   Gain ratio   (3 marks)

 (iv)   Classification errors   (3 marks)

Gain Ratio (cont)

Final Exam Jan 2019, Q2(d) cont.

Can you get the results below?

(i) $G(S_1) = 0.4017$; $G(S_2) = 0.4163$

(ii) $IG(S_1) = 0.0532$; $IG(S_2) = 0.0291$

(iii) $R(S_1) = 0.0644$; $R(S_2) = 0.0550$

(iv) Answer given in lecture notes.

Outline

Decision Tree Implementations

The evolution of various classification (and regression[*])
tree algorithms is shown below:

1st generation: node splitting
- THAID [MM72]
- CHAID [Kas80]

2nd generation: tree-prunning, CV, gain ratio.
- CART (Breiman et al., 1984)
- ID3 (Quinlan, 1986)
- M5 (Quinlan, 1992)[*]
- C4.5 (Quinlan, 1993)
- FACT (Loh and Vanichsetakul, 1988)

Decision Tree Implementations (cont)

3rd generation: more enhanced split rules.

- QUEST (Loh and Shih, 1997)
- CRUISE (Kim and Loh, 2001, 2003)
- Bayesian CART (Chipman et al., 1998; Denison et al., 1998)[*]

4th generation: ensemble methods

- GUIDE (Loh, 2002, 2009; Loh and Zheng, 2013; Loh et al., 2015)
- CTREE (Hothorn et al., 2006)
- MOB (Zeileis et al., 2008)[*]
- Random forest [Bre01]
- TARGET (Fan and Gray, 2005)
- BART (Chipman et al., 2010)

Decision Tree Implementations (cont)

Aspect	Original ID3	C4.5	CART
Features	Discrete features only; cannot handle numeric features	continuous and discrete features	continuous and discrete features
missing values	Handles missing attributes by ignoring them in information gain computation		?
splitting criteria	**Multi-category** splitting which maximises information gain / min. entropy	maximises gain ratio	uses Gini impurity to perform **binary split**
shape	short and wide tree	short and wide tree	taller tree
pruning	no pruning, prone to overfitting	performs post-pruning (bottom-up pruning)	performs cost-complexity pruning

ID3 Algorithm

ID3 (Iterative Dichotomiser 3) was developed in 1986 by Ross Quinlan. The training observations are sorted to the corresponding internal nodes. If the nodes only contain observations from a single class, then the node will become a leaf node. Otherwise, it will be further expanded, by selecting the predictor with highest information gain relative to the new subsets of observations.

ID3 Algorithm (cont)

Final Exam May 2019, Q5(b)

A decision tree model is built to detect whether an online transaction is a fraud transaction (Yes/No), based on three variables – credit card association, transaction amount and country. Table Q5(b) – i to Q5(b) – iii shows the number of observations for different variables with different levels, for fraud and not fraud cases respectively.

Credit Card Associations	Fraud = Yes	Fraud = No
Amex	11	20
Master	34	47
Visa	25	33

ID3 Algorithm (cont)

Final Exam May 2019, Q5(b) cont.

Transaction Amount	Fraud = Yes	Fraud = No
<1000	52	21
≥1000	18	79

Table Q5(b) – ii

Country	Fraud = Yes	Fraud = No
A	25	32
B	17	46
C	28	22

Table Q5(b) – iii

ID3 Algorithm (cont)

Final Exam May 2019, Q5(b) cont.

Construct a decision tree with leaves purity of at least 70% by using gain ratio. State the prediction and respective purity for each terminal node.   (15 marks)

Solution: Let S be the whole table of Fraud, Fraud=Yes be positive $(+)$ class and Fraud=No be negative (-) class.

To answer this question, we need to recall the Gain ratio, Equation (7) on Page 35:

$$R(S^A) = IG(S^A)/H(A).$$

ID3 Algorithm (cont)

Final Exam May 2019, Q5(b) cont.

Total, $S = [70+, 100-] \Rightarrow H(S) = -\left[\frac{70}{170} \log_2 \frac{70}{170} + \frac{100}{170} \log_2 \frac{100}{170}\right] = 0.9774$

For $X_1 = $ Credit card association [Amex=A; Master=M; Visa=V]:

$A = [11+, 20-] \qquad \Rightarrow H(Fraud|X_1 = A)$
$\qquad\qquad\qquad\qquad\quad = -\left[\frac{11}{31} \log_2 \frac{11}{31} + \frac{20}{31} \log_2 \frac{20}{31}\right] = 0.9383$

$M = [34+, 47-] \qquad \Rightarrow H(S_M) = 0.9813$

$V = [25+, 33-] \qquad \Rightarrow H(S_V) = 0.9862$

$IG(S^{X_1}) = 0.9774 - \left[\frac{31}{170} \times 0.9383 + \frac{81}{170} \times 0.9813 + \frac{58}{170} \times 0.9862\right] = 0.0023$

$H(X_1) = -\left[\frac{31}{170} \log_2 \frac{31}{170} + \frac{81}{170} \log_2 \frac{81}{170} + \frac{58}{170} \log_2 \frac{58}{170}\right] = 1.4866$

$R(S^{X_1}) = \dfrac{0.0023}{1.4866} = 0.0015$

ID3 Algorithm (cont)

Final Exam May 2019, Q5(b) cont.

For $X_2 =$ Transaction amount $[< 1000 = L; \geq 1000 = H]$:

$L = [52+, 21-] \qquad \Rightarrow H(S_L) = 0.8657$

$H = [18+, 79-] \qquad \Rightarrow H(S_H) = 0.6921$

$IG(S^{X_2}) = 0.9774 - \left[\frac{73}{170} \times 0.8657 + \frac{97}{170} 0.6921\right] = 0.2108$

$H(X_2) = -\left[\frac{73}{170} \log_2 \frac{73}{170} + \frac{97}{170} \log_2 \frac{97}{170}\right] = 0.9856$

$R(S^{X_2}) = 0.2139$

For $X_3 =$ Country $[A; B; C]$:

$A = [25+, 32-] \qquad \Rightarrow H(S_A) = 0.9891$

$B = [17+, 46-] \qquad \Rightarrow H(S_B) = 0.8412$

$C = [28+, 22-] \qquad \Rightarrow H(S_C) = 0.9896$

$IG(S^{X_3}) = 0.0430$

$H(X_3) = -\left[\frac{57}{170} \log_2 \frac{57}{170} + \frac{63}{170} \log_2 \frac{63}{170} + \frac{50}{170} \log_2 \frac{50}{170}\right] = 1.5786$

$R(S^{X_3}) = 0.0272$

ID3 Algorithm (cont)

Final Exam May 2019, Q5(b) cont.

The transaction amount, X_2, has the highest gain ratio, hence it should be chosen as the root node.

For $X_2 = L(< 1000)$: prediction=Yes; purity $= \frac{52}{52+21} = 0.7123$ (the "purity" is more than 70%)

For $X_2 = H(\geq 1000)$: prediction=No; purity $= \frac{79}{18+79} = 0.8144$ (the "purity" is more than 70%)

ID3 Algorithm (cont)

Final Exam May 2019, Q5(b) cont.

Therefore, we have the classification tree below.
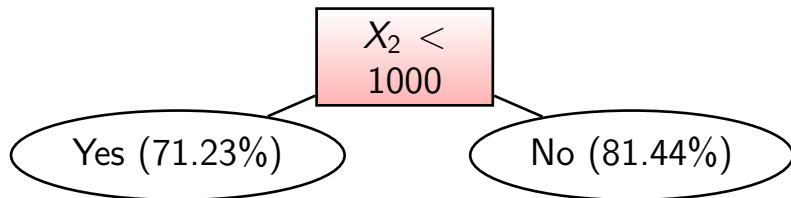
ID3 Algorithm (cont)

The ID3 algorithm is restricted in dealing with categorical data only. However, [Mit97, §3.7.2] mentioned it is possible to introduce a threshold to turn the numeric feature to a binary categorical data. [Mit97, §3.7.3] also pointed out that ID3 can be extended to use gain ratio.

Note that decision trees are myopic in general and ID3-trees are prone to overfitting as the tree depth increases. Therefore, we need more sophisticated tree algorithms.

ID3 Algorithm (cont)

Since ID3 cannot handle numeric data, C4.5 is a successor to ID3 which removed the restriction that features must be categorical by dynamically defining a discrete attribute (based on numerical variables) that partitions the continuous attribute value into a discrete set of intervals. The C4.5 algorithm (https://en.wikipedia.org/wiki/C4.5_algorithm) constructs a decision tree based on the gain ratio.

C4.5 is superseded in 1997 by a commercial system See5/C5.0 according to https://rulequest.com/see5-comparison.html) and CART.

ID3 Algorithm (cont)

C5.0 is Quinlan's latest version release under a proprietary license. It uses less memory and builds smaller rulesets than C4.5 while being more accurate.

The `C50` library implements C5.0 decision trees and rule-based models for pattern recognition extend the work of [Qui86]. The company `https://www.rulequest.com` has the pattern of C5.0 and provides `Cubist` (Rule- And Instance-Based Regression Modelling) and `outliertree` (Explainable outlier detection through decision tree conditioning based on `GritBot`) for statistical inference with tree models.

ID3 Algorithm (cont)

Example 5.3.6: Use C5.0 to construct a decision tree for the tennis data.

Solution:

```
1 library(C50)
2 tennis.data = read.csv("playtennis.csv")
3 tennis.data$day = NULL
4 model = C5.0(playtennis ~., data=tennis.data)
5 plot(model)
```

ID3 Algorithm (cont)

Example 5.3.6 (cont)
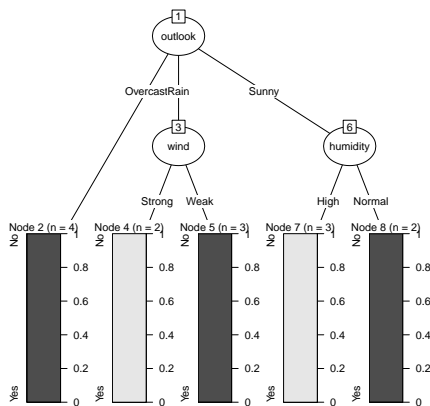
Same as ID3 tree for tennis data (Example 5.3.3).

CART (cont)

Classification And Regression Tree (CART) is a binary tree generation algorithm using Gini index as its splitting criteria which can handle both categorical and numeric predictors. CART handles missing values by surrogating tests to approximate outcomes.

A detail account of the algorithm is given in [BFSO84].

CART (cont)

In R, CART is supported by `tree`, `rpart` and `party`. The `tree` library provides a limited functions of constructing, plotting and pruning trees. The `rpart`, which stands for "recursive partitioning" for classification, regression and survival trees, implements most of the functionality of [BFSO84]. The `rpart.plot` or `partykit::plot.party` can be used to draw nice diagrams for `rpart`.

The `party` library is designed based on continuous input, however, it supports nominal, ordinal, numeric, censored as well as multivariate response variables. The core of the package is `ctree()`, an implementation of conditional inference trees which embed tree-structured regression models into a well defined theory of conditional inference procedures.

CART (cont)

In Python, CART for classification is implemented as
`sklearn.tree.DecisionTreeClassifier`, which is
capable of both binary classification with labels $\{-1, 1\}$
and multiclass classification with labels $\{0, \cdots, K-1\}$.

CART (cont)

Example 5.3.8: Use CART to construct a decision tree for the tennis data in Example 5.2.1

Solution: We can either use `tree` library:

```
1   library(tree)
2   tennis.data = read.csv("playtennis.csv")
3   tennis.data = tennis.data[,2:ncol(tennis.data)]  # Remove 'day'
4   t.m = tree(playtennis~., data=tennis.data, control=tree.control(
5           nobs=nrow(tennis.data),minsize=1))
6   print(t.m); plot(t.m); text(t.m, cex=0.8)
```

or the `rpart` library:

```
1   library(rpart)
2   tennis.data = read.csv("playtennis.csv")
3   tennis.data$day = NULL
4   dt = rpart(playtennis ~ ., data=tennis.data,
5       control=rpart.control(minsplit=2, minbucket=1, cp=0.001))
6   print(dt); plot(dt, margin=0.05); text(dt, use.n=TRUE, cex=0.8)
```

CART (cont)

Example 5.3.8 (cont)
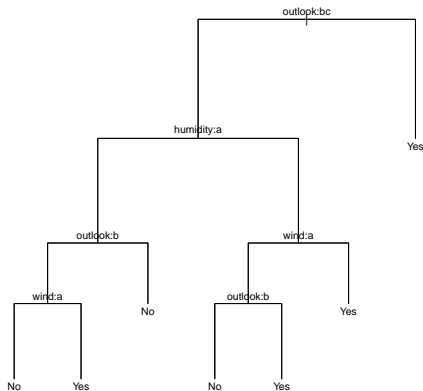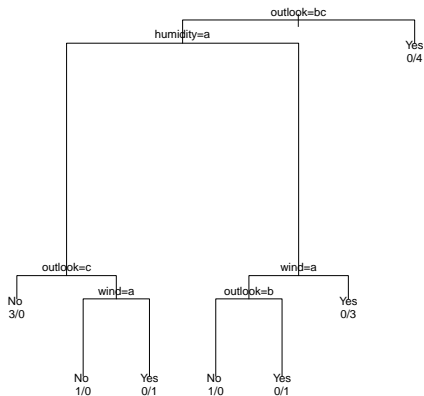
(a) tree   (b) rpart

CART (cont)

The decision tree of the "flame" data can be analysed using `rpart`'s CART tree and produces the following "partitions":
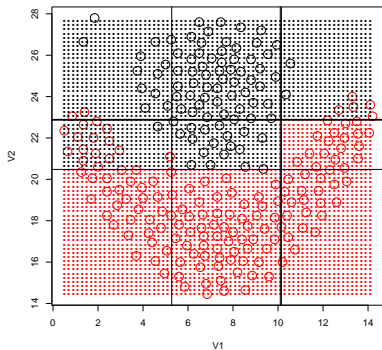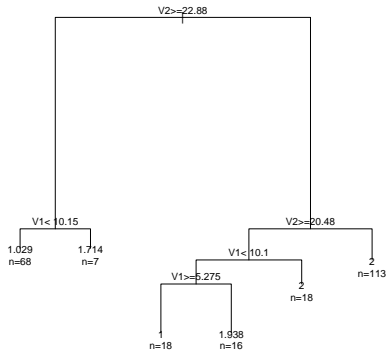
CART (cont)

CART is very similar to C4.5, but it differs in that it supports numerical target variables, i.e. it supports regression. CART constructs binary trees using the feature and threshold that yield the largest information gain at each node.

CART (in `rpart`) and CTree (conditionally inference tree, in the `party` and the `partykit`) are two available regression trees in R.

The scikit-learn uses an optimised version of the CART algorithm (`sklearn.tree.DecisionTreeRegressor`); however, the implementation does not support categorical variables for now (if the input is categorical, it will convert to number?).

CART (cont)

Example SRM-02-18, Q33

The regression tree shown below was produced from a dataset of auto claim payments. Age Category (agecat: 1, 2, 3, 4, 5, 6) and Vehicle Age (veh_age: 1, 2, 3, 4) are both predictor variables, and log of claim amount (LCA) is the dependent variable.

Consider three autos I, II, III:

1. An Auto in Age Category 1 and Vehicle Age 4
2. An Auto in Age Category 5 and Vehicle Age 5
3. An Auto in Age Category 5 and Vehicle Age 3

CART (cont)

Example SRM-02-18, Q33 (cont)
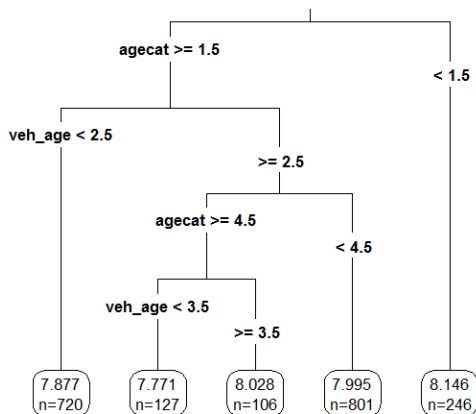
CART (cont)

Example SRM-02-18, Q33 (cont)

Rank the estimated LCA of Autos I, II, and III.

- (A) $LCA(I) < LCA(II) < LCA(III)$
- (B) $LCA(I) < LCA(III) < LCA(II)$
- (C) $LCA(II) < LCA(I) < LCA(III)$
- (D) $LCA(II) < LCA(III) < LCA(I)$
- (E) $LCA(III) < LCA(II) < LCA(I)$

Answer: (E) Do you know why?

Outline

Practical Considerations

- overfitting
- tree pruning
- complexity

Example SRM-02-18, Q29: Determine which of the following considerations may make decision trees preferable to other statistical learning methods.

1. Decision trees are easily interpretable.

2. Decision trees can be displayed graphically.

3. Decision trees are easier to explain than linear regression methods.

Practical Considerations (cont)

(A) None

(B) I and II only

(C) I and III only

(D) II and III only

(E) The correct answer is not given by (A), (B), (C), or (D).

Solution: All three statements are true. See [JWHT13, Section 8.1]. The statement that trees are easier to explain than linear regression methods may not be obvious. For those familiar with regression but just learning about trees, the reverse may be the case. However, for those not familiar with regression, relating the dependent variable to the independent variables, especially if the dependent variable has been transformed, can be difficult to explain.

Answer: (E)

Practical Considerations (cont)

Tree Pruning:

A smaller tree with fewer splits may lead to a lower variance, better interpretation and slightly more "generalisation". There are two approaches in "pruning" a complex decision tree:

- **Pre-pruning**: This type of algorithms will stop growing the tree earlier, before it perfectly classifies the training set.
- **Post-pruning**: This type of algorithms will allow the tree to perfectly classify training set, and post prune the tree.

Practical Considerations (cont)

Complexity:

Let n be the number of samples and p be the number of features (which are used throughout the lecture). According to https://scikit-learn.org/stable/modules/tree.html, the run time cost to construct a balanced binary tree is $O(p \times n \log(n))$ and query time $O(\log(n))$. Although the tree construction algorithm attempts to generate balanced trees, they will not always be balanced.

Regarding a Tutorial Question

Information Gain / Gini Impurity both give
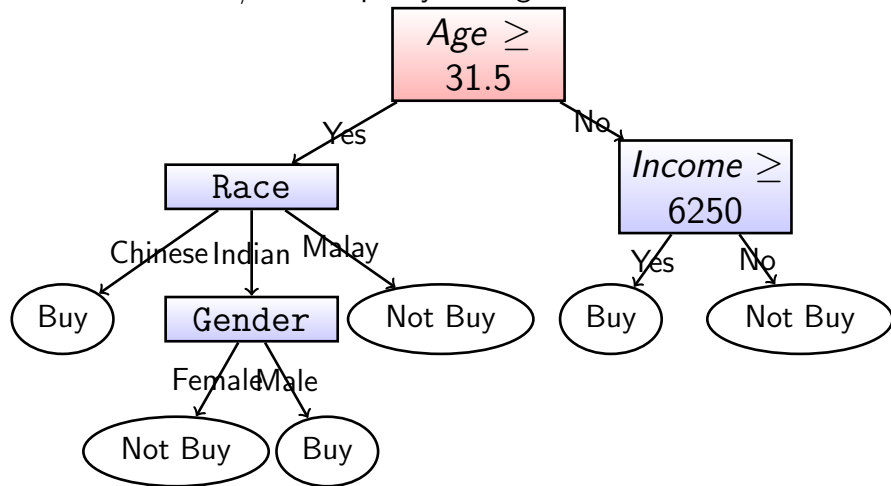
References I

Leo Breiman, Jerome Friedman, Charles J. Stone, and R. A. Olshen, *Classification and regression trees*, Chapman and Hall/CRC, 1984, ISBN 9780412048418.

Leo Breiman, *Random forests*, Mach. Learn. **45** (2001), no. 1, 5–32.

Thomas M. Cover and Joy A. Thomas, *Elements of information theory*, 2nd ed., John Wiley & Sons, Inc., 2006.

References II

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani, *An introduction to statistical learning: with applications in r*, Springer Texts in Statistics, Springer Science+Business Media New York, 2013.

G. V. Kass, *An exploratory technique for investigating large quantities of categorical data*, Applied Statistics **29** (1980), no. 2, 119–127.

Tom M. Mitchell, *Machine learning*, MIT Press, 1997.

References III

R. Messenger and L. Mandell, *A modal search technique for predictive nominal scale multivariate analysis*, Journal of the American Statistical Association **67** (1972), no. 340, 768–772.

J. R. Quinlan, *Induction of decision trees*, Machine Learning **1** (1986), 81–106, https://doi.org/10.1007/BF00116251.