Lecture 15: Decision Trees & Random Forest

COMP90049
Knowledge Technologies

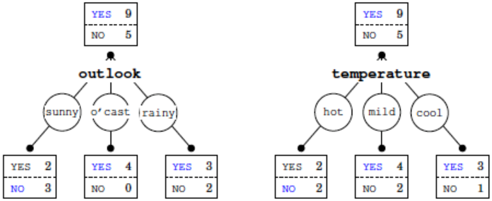
Hasti Samadi & Sarah Erfani & Karin Verspoor , CIS

Semester 2, 2019

THE UNIVERSITY OF MELBOURNE

From Decision Stumps to Decision Trees

- We have seen decision stumps in action in the context of 1-R

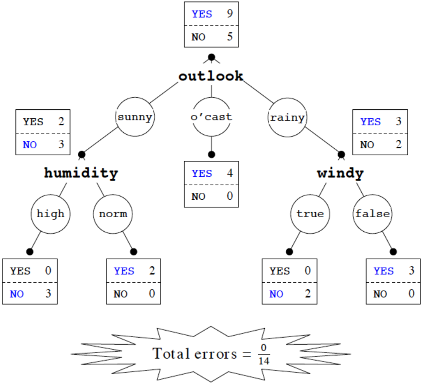


- Based the same concept we can build a more complex model with the ability to capture complex feature interaction

Decision Tree

A flow-chart-like tree structure

- Internal node denotes a test on an *attribute*
- Branch represents an *outcome* of the test (*attribute value*)
- Leaf nodes represent *class labels* or class distribution



Decision Tree

Advantages:

- Basic classification model
- Fast
- Scalable
- Interpretable

Disadvantage:

- Not highest accuracy (Random Forest to the rescue)

Training Decision Tree: Hunt's Algorithm

Let D_t be the set of training records that reach a node t

Training Decision Tree: Hunt's Algorithm

General Procedure:

- If D_t contains records that belong to more than one class, use an attribute to **split** the data into smaller subsets.

Training Decision Tree: Hunt's Algorithm

General Procedure:

- If D_t contains records that belong to more than one class, use an attribute to **split** the data into smaller subsets.

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

Training Decision Tree: Hunt's Algorithm

General Procedure:

- If D_t contains records that belong to more than one class, use an attribute to **split** the data into smaller subsets.

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

Training Decision Tree: Hunt's Algorithm

General Procedure:

- If D_t contains records that belong to more than one class, use an attribute to **split** the data into smaller subsets.

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

Training Decision Tree: Hunt's Algorithm

General Procedure:

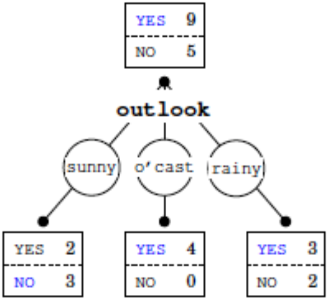
- If D_t contains records that belong to more than one class, use an attribute to **split** the data into smaller subsets.

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

Training Decision Tree: Hunt's Algorithm

General Procedure:

- If D_t contains records that belong to more than one class, use an attribute to **split** the data into smaller subsets.



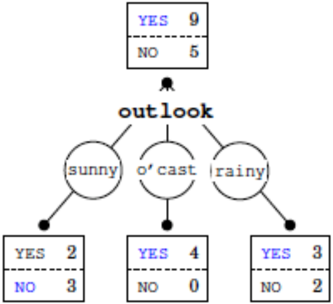
General Procedure:

- If D_t contains records that belong to more than one class, use an attribute to **split** the data into smaller subsets.
- Recursively apply the procedure to each subset.

Training Decision Tree: Hunt's Algorithm

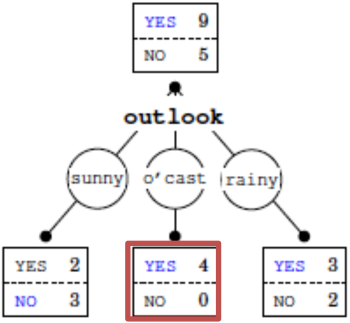
General Procedure:

- If D_t contains records that belong to more than one class, use an attribute to **split** the data into smaller subsets.
- Recursively apply the procedure to each subset.
  - If D_t contains records that **ALL** belong to the same class y_t, then t is a leaf node labelled as $\mathbf{y_t}$



General Procedure:

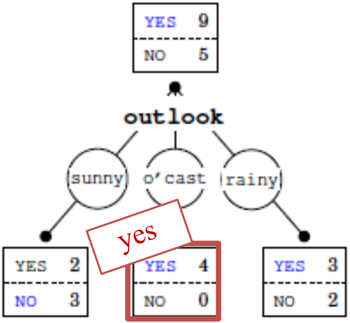
- If D_t contains records that belong to more than one class, use an attribute to **split** the data into smaller subsets.
- Recursively apply the procedure to each subset.
  - If D_t contains records that **ALL** belong to the same class y_t, then t is a leaf node labelled as $\mathbf{y_t}$



Training Decision Tree: Hunt's Algorithm

General Procedure:

- If D_t contains records that belong to more than one class, use an attribute to **split** the data into smaller subsets.
- Recursively apply the procedure to each subset.
  - If D_t contains records that ALL belong to the same class y_t, then t is a leaf node labelled as $\mathbf{y_t}$



Training Decision Tree: Hunt's Algorithm

Let D_t be the set of training records that reach a node t

General Procedure:

- If D_t contains records that belong to more than one class, use an attribute to **split** the data into smaller subsets.
- Recursively apply the procedure to each subset.
  - If D_t contains records that **ALL** belong to the same class y_t, then t is a leaf node labelled as $\mathbf{y_t}$
  - If D_t is an empty set, then t is a leaf node labelled by the **default class,** y_d (majority class in the data)

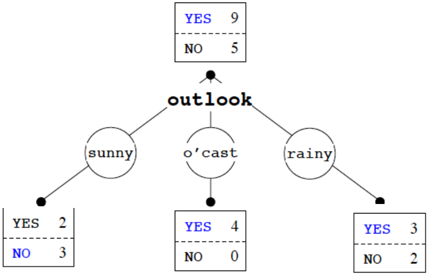
Training Decision Tree: Hunt's Algorithm

Let D_t be the set of training records that reach a node t

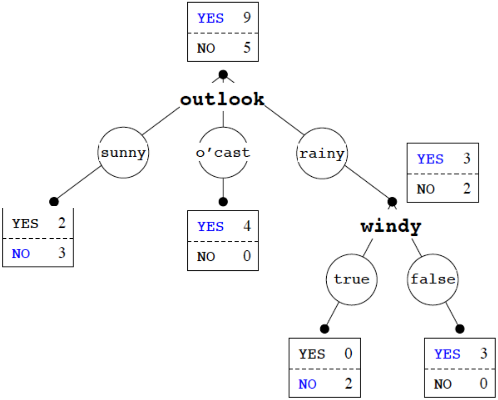
General Procedure:

- If D_t contains records that belong to more than one class, use an attribute to **split** the data into smaller subsets.
- Recursively apply the procedure to each subset.
  - If D_t contains records that **ALL** belong to the same class y_t, then t is a leaf node labelled as $\mathbf{y_t}$
  - If D_t is an empty set, then t is a leaf node labelled by the **default class,** y_d (majority class in the data)
  - If D_t contains records that belong to more than one class, use an attribute to **split** the data into smaller subsets.

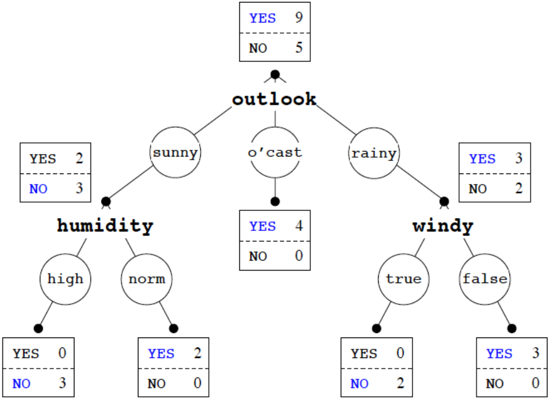
Hunt's Algorithm: Example



Hunt's Algorithm: Example



Hunt's Algorithm: Example

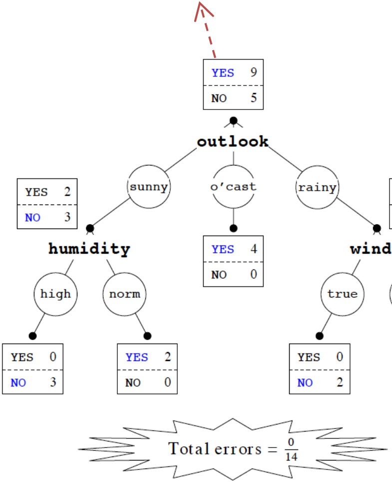


Predicting with Decision Tree

Outlook	Temperature	Humidity	Windy	Play
Rainy	High	Mild	true	?

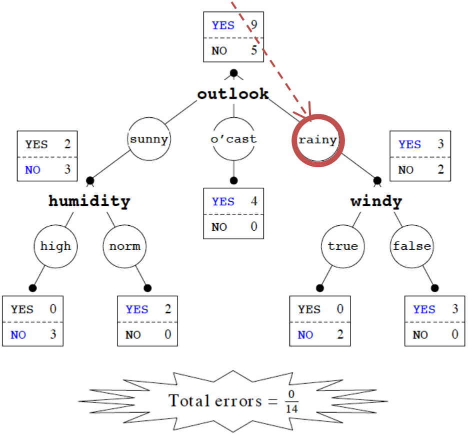
Predicting with Decision Tree

Outlook	Temperature	Humidity	Windy	Play
Rainy	High	Mild	true	?



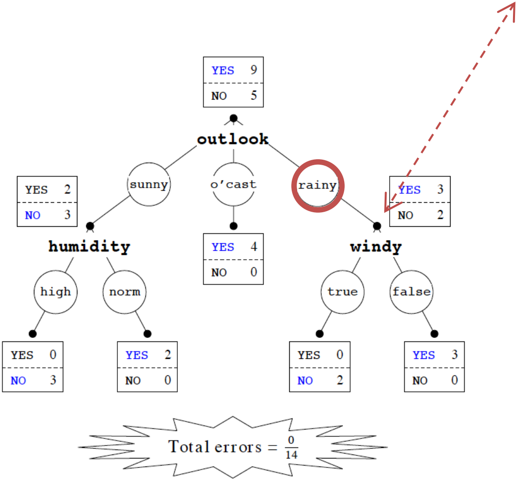
Predicting with Decision Tree

Outlook	Temperature	Humidity	Windy	Play
Rainy	High	Mild	true	?



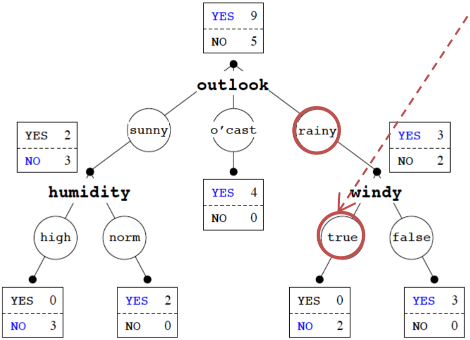
Predicting with Decision Tree

Outlook	Temperature	Humidity	Windy	Play
Rainy	High	Mild	true	?



Predicting with Decision Tree

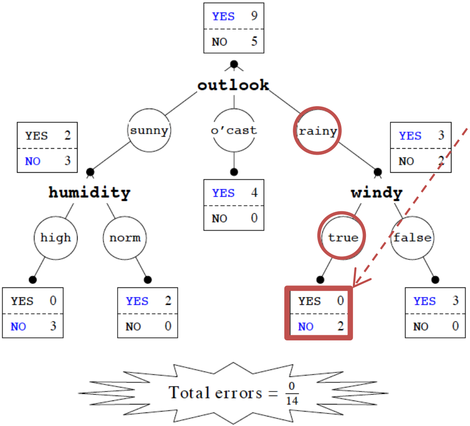
Outlook	Temperature	Humidity	Windy	Play
Rainy	High	Mild	true	?



Total errors $= \frac{0}{14}$

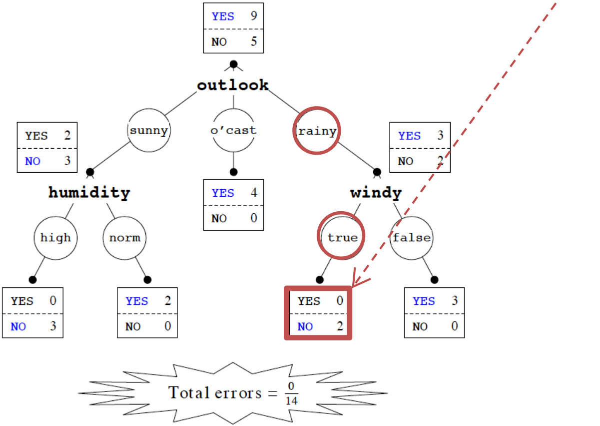
Predicting with Decision Tree

Outlook	Temperature	Humidity	Windy	Play
Rainy	High	Mild	true	?



Predicting with Decision Tree

Outlook	Temperature	Humidity	Windy	Play
Rainy	High	Mild	true	NO



Criterion for Attribute Selection

- There could be more than one tree that fits the same data!

Criterion for Attribute Selection

- There could be more than one tree that fits the same data!
- How do we choose the attribute to partition the instances at a given node?

Criterion for Attribute Selection

- There could be more than one tree that fits the same data!
- How do we choose the attribute to partition the instances at a given node?
- We want to get the **smallest** tree

Criterion for Attribute Selection

- There could be more than one tree that fits the same data!
- How do we choose the attribute to partition the instances at a given node?
- We want to get the **smallest** tree
- Prefer attributes that create more **homogeneous (pure)** nodes

Criterion for Attribute Selection

- There could be more than one tree that fits the same data!
- How do we choose the attribute to partition the instances at a given node?
- We want to get the **smallest** tree
- Prefer attributes that create more **homogeneous (pure)** nodes
- We need to measure node **purity**

Criterion for Attribute Selection

- There could be more than one tree that fits the same data!
- How do we choose the attribute to partition the instances at a given node?
- We want to get the **smallest** tree
- Prefer attributes that create more **homogeneous (pure)** nodes
- We need to measure node **purity**
- One Measure is **Entropy**

Criterion for Attribute Selection

- There could be more than one tree that fits the same data!
- How do we choose the attribute to partition the instances at a given node?
- We want to get the **smallest** tree
- Prefer attributes that create more **homogeneous (pure)** nodes
- We need to measure node **purity**
- One Measure is **Entropy** (measure of **predictability**)

Entropy (Recap)

The entropy of a discrete random event x with possible states 1,..,n is:

$$H(x) = -\sum_{i=1}^{n} P(i) \log_2 P(i)$$

(where $0 \log_2(0)$ is 0)

Entropy (Recap)

When **Entropy** is low

- The event is predictable (more homogeneous)

When **Entropy** is high

- The event is unpredictable (less homogeneous)

Entropy (Recap)

When **Entropy** is low

- ➢ The event is predictable (more homogeneous)

When **Entropy** is high

- ➢ The event is unpredictable (less homogeneous)

In developing a Decision Tree:

✓ We prefer **attributes** (for splitting the dataset) that **reduce the entropy** more.

Information Gain

- The expected reduction in entropy caused by knowing the value of an attribute.
- Compare:
  - the entropy before splitting the tree using the attribute's values
  - the weighted average of the entropy over the children after the split (**Mean Information**)
- If the entropy decreases, then we have a better tree (more predictable)

Information Gain

- The expected reduction in entropy caused by knowing the value of an attribute.
- Compare:
    - the entropy before splitting the tree using the attribute's values
    - the weighted average of the entropy over the children after the split (**Mean Information**)
- If the entropy decreases, then we have a better tree (more predictable)
- Information Gain is the difference between the *entropy at the root* (before splitting) and the *Mean Information*

Information Gain

- We calculate the **Mean Information** for a tree stump with *m* attribute values as:

$$\text{Mean Info}(x_1, .., x_m) = \sum_{i=1}^{m} P(x_i)H(x_i)$$

where $H(x_i)$ is the entropy of the class distribution for the instances at node x_i

Information Gain

- We calculate the **Mean Information** for a tree stump with *m* attribute values as:

$$\text{Mean Info}(x_1, .., x_m) = \sum_{i=1}^{m} P(x_i)H(x_i)$$

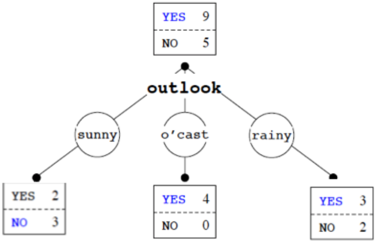
where $H(x_i)$ is the entropy of the class distribution for the instances at node x_i

- **Information Gain**

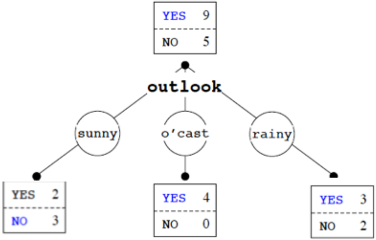
$$IG(R_A|R) = H(R) - \sum_{i=1}^{m} P(x_i)H(x_i)$$

$$IG(R_A|R) = H(R) - \text{Mean Info}(x_1,..,x_m)$$

Calculating Information Gain



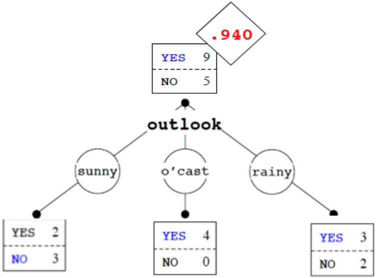
Calculating Information Gain



entropy before splitting the tree (entropy at root)

$$H(R) =$$
$$-((9/14)\log_2(9/14) + (5/14)\log_2(5/14)) =$$
$$-(-.4098 - 0.5305) = 0.940$$

Calculating Information Gain



entropy before splitting the tree (entropy at root)

$$H(R) =$$
$$-((9/14)\log_2(9/14) + (5/14)\log_2(5/14)) =$$
$$-(-.4098 - 0.5305) = 0.940$$

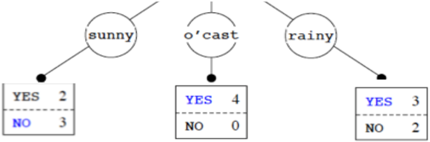
Calculating Information Gain

entropies <u>after</u> splitting the tree (outlook)

$$H(rainy) =$$
$$-(3/5)\log_2(3/5) + (2/5\log_2(2/5))) =$$
$$-(-0.4422 - 0.5288) = 0.971$$

$$H(overcast) =$$
$$-((4/4)\log_2(4/4) + (0/4\log_2(0/4))) = 0$$

$$H(sunny) =$$
$$-((2/5)\log_2(2/5) + (3/5\log_2(3/5))) = 0.971$$



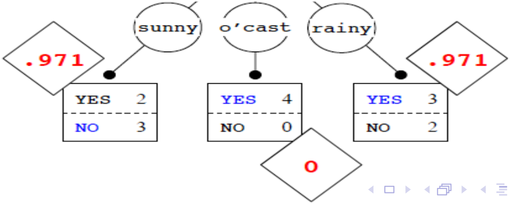
Calculating Information Gain

entropies after splitting the tree (outlook)

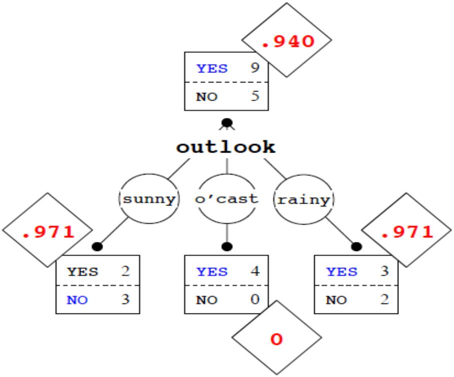
$$H(rainy) =$$
$$-(3/5)\log_2(3/5) + (2/5\log_2(2/5))) =$$
$$-(-0.4422 - 0.5288) = 0.971$$

$$H(overcast) =$$
$$-((4/4)\log_2(4/4) + (0/4\log_2(0/4))) = 0$$

$$H(sunny) =$$
$$-((2/5)\log_2(2/5) + (3/5\log_2(3/5))) = 0.971$$



Calculating Information Gain



Mean Information for Outlook

$$Mean\_info(outlook) = P(rainy)H(rainy) + P(overcast)H(overcast) + P(sunny)H(sunny) = 5/14 * 0.971 + 0 + 5/14 * 0.971 = 0.693$$

Calculating Information Gain

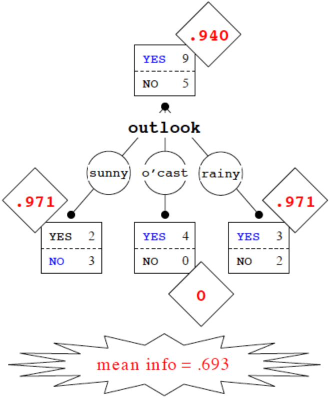
Information Gain

$$IG(R_A | R) = H(R) - \text{Mean Info}\ (x_1, .., x_m)$$

Information Gain

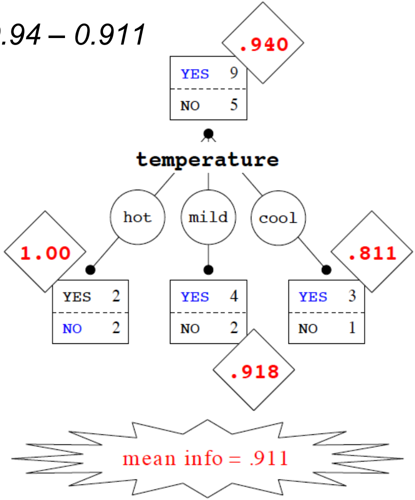
$$IG(R_A | R) = H(R) - Mean\ Info\ (x_1, .., x_m)$$

$$IG\ (\ outlook\ |\ R\ ) = 0.94 - 0.693 = 0.247$$



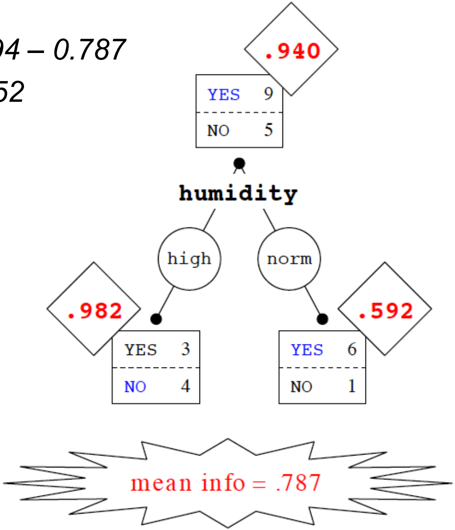
Building a tree by using the Information Gain

IG (temperature | R) = 0.94 – 0.911
= 0.029



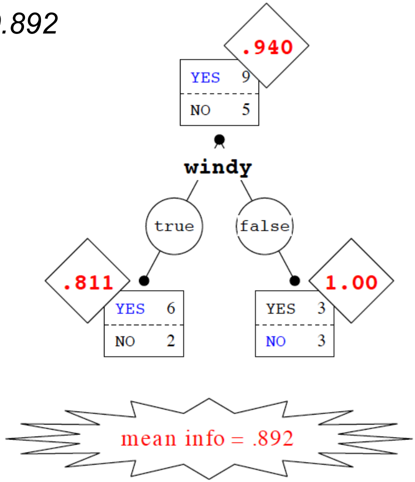
Building a tree by using the Information Gain

$$IG\ (\ humidity\ |\ R\ ) = 0.94 - 0.787$$
$$= 0.152$$



Building a tree by using the Information Gain

$$IG\,(\,windy\,|\,R\,) = 0.94 - 0.892$$
$$= 0.048$$



Building a tree by using the Information Gain

$$IG(\mathit{outlook} \mid R) = 0.247$$

$$IG(\mathit{temperature} \mid R) = 0.029$$

$$IG(\mathit{humidity} \mid R) = 0.152$$

$$IG(\mathit{windy} \mid R) = 0.048$$

Building a tree by using the Information Gain

IG (outlook | R) = 0.247 *higher Information Gain*

IG (temperature | R) = 0.029

IG (humidity | R) = 0.152

IG (windy | R) = 0.048

Shortcomings of Information Gain

ID	Outlook	Temperature	Humidity	Windy	Play
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes

Shortcomings of Information Gain

Decision Tree & Random Forest

COMP90049 Knowledge Technologies

Decision Tree
Definition
Hunt Algorithm
Predicting by DT

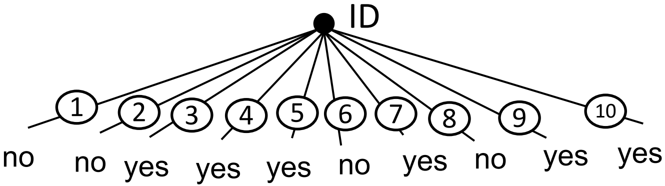
Attribute Selection
Information Gian
Gain Ratio

More DT Algorithms

Overfitting

Classifier Combination
Bagging
Random Tree
Random Forest

Summary
Resources



ID	Outlook	Temperature	Humidity	Windy	Play
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes

Shortcomings of Information Gain

- Information gain tends to prefer highly-branching attributes:
  - A subset of instances is more likely to be homogeneous (all of a single class) if there are only a few instances
  - Attribute with many values will have fewer instances at each child node

Shortcomings of Information Gain

- Information gain tends to prefer highly-branching attributes:
  - A subset of instances is more likely to be homogeneous (all of a single class) if there are only a few instances
  - Attribute with many values will have fewer instances at each child node
- This may result in **overfitting/fragmentation**

Shortcomings of Information Gain

- Information gain tends to prefer highly-branching attributes:
  - A subset of instances is more likely to be homogeneous (all of a single class) if there are only a few instances
  - Attribute with many values will have fewer instances at each child node
- This may result in **overfitting/fragmentation**
- Solution → **Gain Ratio**

Gain Ratio

- **Gain ratio (GR)** reduces the bias for information gain towards highly-branching attributes by normalizing relative to the split information
- **Split info (SI)** is the entropy of a given split (evenness of the distribution of instances to attribute values)

$$
\begin{aligned}
GR(R_A|R) &= \frac{IG(R_A|R)}{SI(R_A|R)} = \frac{IG(R_A|R)}{H(R_A)} \\
&= \frac{H(R) - \sum_{i=1}^{m} P(x_i) H(x_i)}{-\sum_{i=1}^{m} P(x_i) \log_2 P(x_i)}
\end{aligned}
$$

- **Gain ratio (GR)** reduces the bias for information gain towards highly-branching attributes by normalising relative to the split information
- **Split info (SI)** is the entropy of a given split (evenness of the distribution of instances to attribute values)

$$GR(R_A|R) = \frac{IG(R_A|R)}{SI(R_A|R)} = \frac{IG(R_A|R)}{H(R_A)}$$
$$= \frac{H(R) - \sum_{i=1}^{m} P(x_i)H(x_i)}{-\sum_{i=1}^{m} P(x_i)\log_2 P(x_i)}$$

- **Note**: Split Info is entropy of distribution of instances to attribute values (disregarding classes, unlike Mean Info)

Gain Ratio

$$\text{Mean Info}(x_1, .., x_m) = \sum_{i=1}^{m} P(x_i) H(x_i)$$

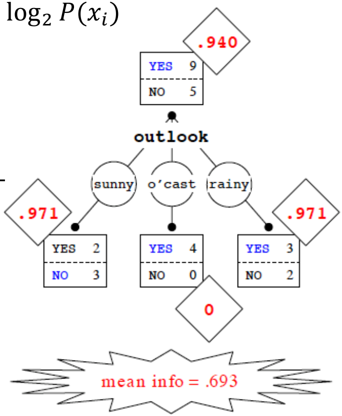
$$\text{Split Info}(x_1, .., xm|R) = -\sum_{i=1}^{m} P(x_i) \log_2 P(x_i)$$

Gain Ratio

$$\text{Mean Info}(x_1, .., x_m) = \sum_{i=1}^{m} P(x_i) H(x_i)$$

$$\text{Split Info}(x_1, .., xm|R) = -\sum_{i=1}^{m} P(x_i) \log_2 P(x_i)$$

$$\text{Split Info}(\text{outlook}|R) =$$
$$-\left(\left(\frac{5}{14}\right)\log_2\left(\frac{5}{14}\right) + \left(\frac{4}{14}\right)\log_2\left(\frac{4}{14}\right) + \left(\frac{5}{14}\right)\log_2\left(\frac{5}{14}\right)\right) = 1.577$$



Gain Ratio

$$IG(\texttt{outlook}|R) = 0.247$$

$$IG(\texttt{humidity}|R) = 0.152$$

$$IG(\texttt{temperature}|R) = 0.029$$

$$IG(\texttt{windy}|R) = 0.048$$

Gain Ratio

$$IG(\texttt{outlook}|R) = 0.247$$
$$SI(\texttt{outlook}|R) = 1.577$$

$$IG(\texttt{humidity}|R) = 0.152$$
$$SI(\texttt{humidity}|R) = 1.000$$

$$IG(\texttt{temperature}|R) = 0.029$$
$$SI(\texttt{temperature}|R) = 1.557$$

$$IG(\texttt{windy}|R) = 0.048$$
$$SI(\texttt{windy}|R) = 0.985$$

Gain Ratio

$$
\begin{aligned}
IG(\texttt{outlook}|R) &= 0.247\\
SI(\texttt{outlook}|R) &= 1.577\\
GR(\texttt{outlook}|R) &= 0.156
\end{aligned}
$$

$$
\begin{aligned}
IG(\texttt{humidity}|R) &= 0.152\\
SI(\texttt{humidity}|R) &= 1.000\\
GR(\texttt{humidity}|R) &= 0.152
\end{aligned}
$$

$$
\begin{aligned}
IG(\texttt{temperature}|R) &= 0.029\\
SI(\texttt{temperature}|R) &= 1.557\\
GR(\texttt{temperature}|R) &= 0.019
\end{aligned}
$$

$$
\begin{aligned}
IG(\texttt{windy}|R) &= 0.048\\
SI(\texttt{windy}|R) &= 0.985\\
GR(\texttt{windy}|R) &= 0.049
\end{aligned}
$$

Gain Ratio

$$IG(\texttt{outlook}|R) = 0.247$$
$$SI(\texttt{outlook}|R) = 1.577$$
$$GR(\texttt{outlook}|R) = 0.156$$

$$IG(\texttt{humidity}|R) = 0.152$$
$$SI(\texttt{humidity}|R) = 1.000$$
$$GR(\texttt{humidity}|R) = 0.152$$

$$IG(\texttt{temperature}|R) = 0.029$$
$$SI(\texttt{temperature}|R) = 1.557$$
$$GR(\texttt{temperature}|R) = 0.019$$

$$IG(\texttt{windy}|R) = 0.048$$
$$SI(\texttt{windy}|R) = 0.985$$
$$GR(\texttt{windy}|R) = 0.049$$

Gain Ratio

$$
\begin{aligned}
IG(\texttt{outlook}|R) &= 0.247 \\
SI(\texttt{outlook}|R) &= 1.577 \\
GR(\texttt{outlook}|R) &= 0.156
\end{aligned}
$$

$$
\begin{aligned}
IG(\texttt{humidity}|R) &= 0.152 \\
SI(\texttt{humidity}|R) &= 1.000 \\
GR(\texttt{humidity}|R) &= 0.152
\end{aligned}
$$

$$
\begin{aligned}
IG(\texttt{temperature}|R) &= 0.029 \\
SI(\texttt{temperature}|R) &= 1.557 \\
GR(\texttt{temperature}|R) &= 0.019
\end{aligned}
$$

$$
\begin{aligned}
IG(\texttt{windy}|R) &= 0.048 \\
SI(\texttt{windy}|R) &= 0.985 \\
GR(\texttt{windy}|R) &= 0.049
\end{aligned}
$$

Other Decision Tree Implementations

Decision Tree & Random Forest

COMP90049 Knowledge Technologies

Decision Tree
Definition
Hunt Algorithm
Predicting by DT

Attribute Selection
Information Gian
Gain Ratio

More DT Algorithms

Overfitting

Classifier Combination
Bagging
Random Tree
Random Forest

Summary
Resources

Ross Quinlan

- **ID3** uses the Hunt's algorithm with information gain criterion and gain ratio
    - Available in WEKA (no discretization, no missing values)
- **C4.5** improves **ID3**
    - Needs entire data to fit in memory
    - Handles missing attributes and continuous attributes
    - Performs tree post-pruning
    - Available in WEKA as **J48**

Other Decision Tree Implementations

Breiman et al.

- **CART** builds multivariate decision (binary) trees
    - Supports numerical target variables (regression)
    - Available in WEKA as **SimpleCART**

Determine When to Stop Splitting

- When a node is homogenous

Determine When to Stop Splitting

- When a node is homogenous
- When the subsample size is smaller than a threshold

Determine When to Stop Splitting

- When a node is homogenous
- When the subsample size is smaller than a threshold

Remember: our goal is not to sub-divide the data perfectly

- Over-subdivision leads to a complicated decision boundary (over-fitting)

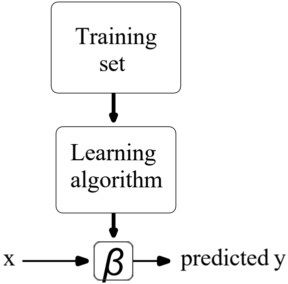
Determine When to Stop Splitting

Metrics to control the complexity of the tree

- Total number of nodes
- Tree depth
- Minimum number of data points for a split
- IG/GR is smaller than a threshold

So far..

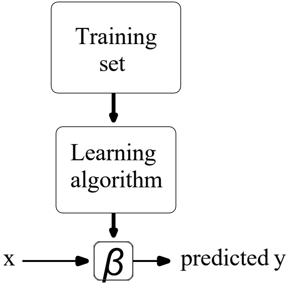
- We have discussed individual classification algorithms and considered each of them in isolation
- We have discussed ways of comparing the performance of individual classifiers over a given dataset/task, which allows us to choose the "dataset optimal" classifier



So far..

- We have discussed individual classification algorithms and considered each of them in isolation
- We have discussed ways of comparing the performance of individual classifiers over a given dataset/task, which allows us to choose the "dataset optimal" classifier
- **Classifier combination (ensemble learning)**

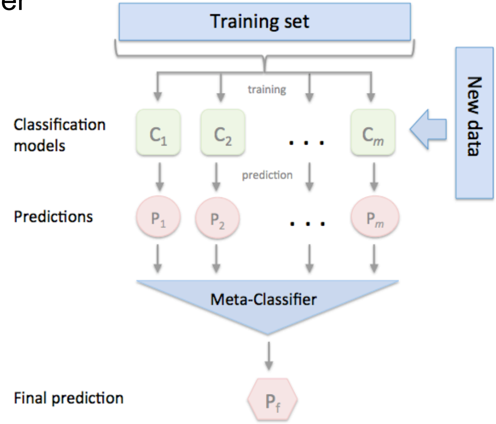
  constructs a set of base classifiers from a given set of training data and aggregates the outputs into a single meta-classifier



Classifier Combination

- **Classifier combination (ensemble learning)**

  constructs a set of base classifiers from a given set of training data and aggregates the outputs into a single meta-classifier



Classifier Combination

- **Motivation 1**: the combination of lots of weak classifiers can be at least as good as one strong classifier
- **Motivation 2**: the combination of a selection of strong classifiers is (usually) at least as good as the best of the base classifiers

Classifier Combination

- **Instance manipulation:** generate multiple training datasets through sampling, and train a base classifier over each
- **Feature manipulation:** generate multiple training datasets through different feature subsets, and train a base classifier over each
- **Class label manipulation:** generate multiple training datasets by manipulating the class labels in a reversible manner
- **Algorithm manipulation:** semi-randomly “tweak” internal parameters within a given algorithm to generate multiple base classifiers over a given dataset

Bagging

- **Basic intuition**: the more data, the better the performance, so how can we get ever more data out of a fixed training dataset?
- **Method**: Randomly sample the original dataset N times, with replacement

Original training dataset:

1	2	3	4	5	6	7	8	9	10

Bootstrap samples:

7	2	6	7	5	4	8	8	1	10

1	3	8	10	3	5	8	10	1	9

2	9	4	2	7	9	3	10	1	10

Random Tree

A “Random Tree” is a Decision Tree where:

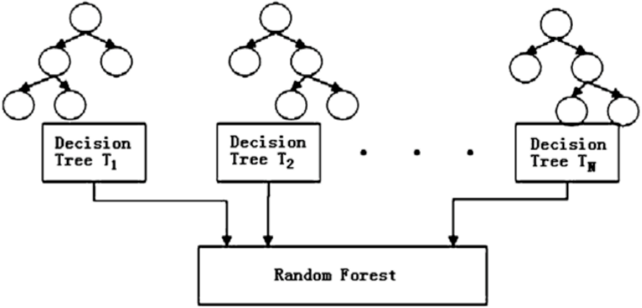
- At each node, only some of the possible attributes are considered
- For example, a fixed proportion of all of the attributes, except the ones used earlier in the tree
- Attempts to control for unhelpful attributes in the feature set
- Much faster to build than a “deterministic” Decision Tree, but increases model variance

Random Forest

A “Random Forest” is:

- An ensemble of Random Trees (many trees = forest)
- Each tree is built on a random subset of records of the data: **Tree bagging**
- Each tree is built on a random subset of features of the data: **Random subspace**
- Decision via **majority voting**

Random Forest



Community of Experts

Practical Properties of Random Forests

- Generally a very strong performer
- Embarrassingly parallelisable
- Surprisingly efficient
- Robust to overfitting
- Interpretability sacrificed

Tools and Resources

This lecture was prepared using some material adapted from:

- https://www-users.cs.umn.edu/~kumar/dmbook/ch4.pdf
- http://www-users.cs.umn.edu/~kumar/dmbook/dmslides/chap4_basic_classification.ppt
- Mitchell, Tom (1997). Machine Learning. Chapter 3: Decision Tree Learning.
- Tan et al (2006) Introduction to Data Mining. Section 4.3, pp 150-171.
- Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. Introduction to Data Mining. Addison Wesley, 2006