



# Implementation of Decision Tree Classifiers

## ID3 versus C4.5

Depuydt Antoine  
Dany Efila  
Mudura Mircea

May, 2017



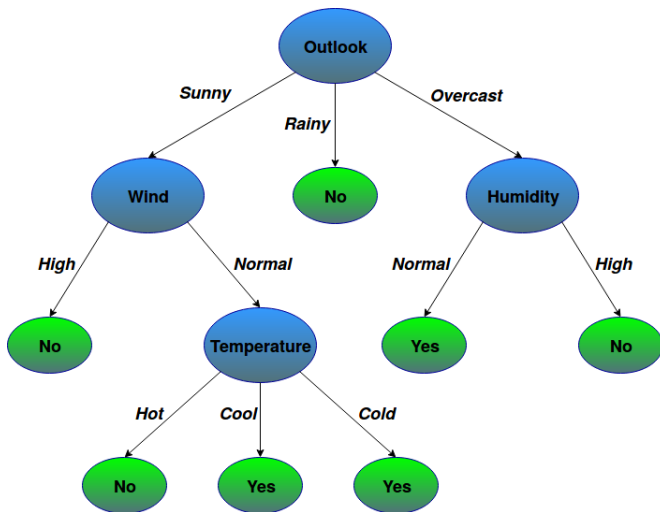
- ▶ Data mining: compress, understand and predict
  - ▶ Clustering
  - ▶ Classification
  - ▶ Regression
  - ▶ ...
  
- ▶ Techniques to find links
  - ▶ Linear Regression
  - ▶ Decision Trees
  - ▶ Neural Networks
  - ▶ ...

# Classification

- ▶ Classical example: play tennis today?
  - ▶ **Features:**
    - ▶ Outlook: sunny, overcast, rainy
    - ▶ Temperature: hot, cool, cold
    - ▶ Wind: high, weak
    - ▶ Humidity: high, normal
  - ▶ **Class labels:**
    - ▶ Yes
    - ▶ No

# Decision Tree

- ▶ Visual model, easily understandable
- ▶ Model: tree with **decision** and **leaf nodes**





## ID3 versus C4.5

- ▶ Goal: implement ID3 and C4.5 algorithms
- ▶ Objectives: compare ID3 and C4.5 output
  - ▶ Compare ID3 and C4.5
  - ▶ Create an application that classifies any data using both algorithms

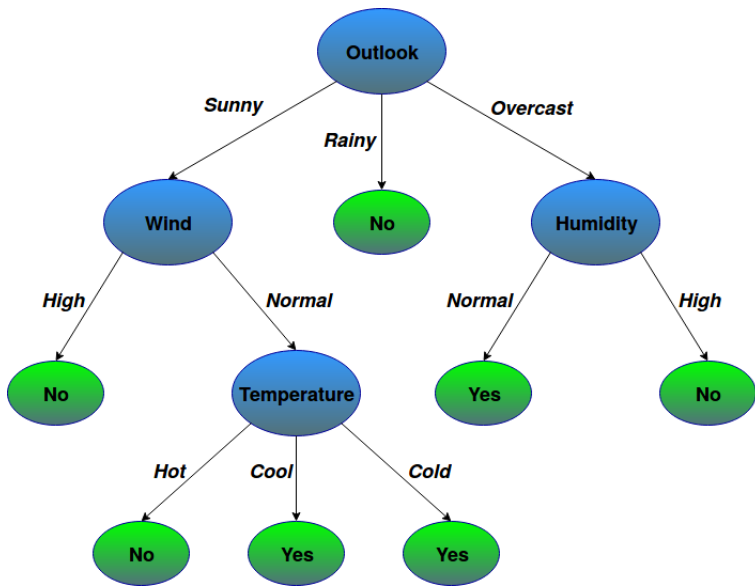


- ▶ Shannon's entropy:  $H(S) = -\sum_i (p_i * \log_2(p_i))$ 
  - ▶ Higher entropy more information gathered
- ▶ Example coin toss:
  - ▶ Entropy =  $-P(heads)\log_2 P(heads) - P(tails)\log_2 P(tails)$
- ▶ Information Gain:  $IG(A) = H(S) - \sum_t p(t)H(t)$ 
  - ▶  $H(S)$  = entropy of set S
  - ▶  $t$  Subset of S obtained by splitting S with A (T)
  - ▶  $p(t)$  proportion of elements in t to the no elements in S
  - ▶  $H(t)$  entropy of subset t

## Attributes, Classifier, Class features

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

# Decision Tree





- ▶  $Entropy(S) = -\frac{9}{14} \log_2(\frac{9}{14}) - \frac{5}{14} \log_2(\frac{5}{14})$
- ▶  $Gain(S, Windy) = Entropy(S) - \frac{8}{14} Entropy(S_{false}) - \frac{6}{14} Entropy(S_{true})$
- ▶ Windy:
  - ▶ False 8  $\rightarrow (6_+, 2_-)$
  - ▶ True 6  $\rightarrow (3_+, 3_-)$
- ▶  $Entropy(S_{false}) = -\frac{6}{8} \log_2(\frac{6}{8}) - \frac{2}{8} \log_2(\frac{2}{8}) = 0.811$
- ▶  $Entropy(S_{true}) = -\frac{3}{6} \log_2(\frac{3}{6}) - \frac{3}{6} \log_2(\frac{3}{6}) = 1$
- ▶  $\rightarrow Gain(S, Windy) = 0.940 - \frac{8}{14} * 0.811 - \frac{6}{14} * 1 = 0.048$
- ▶  $Gain(S, Humidity) = 0.151$
- ▶  $Gain(S, Temperature) = 0.029$
- ▶  $Gain(S, Outlook) = 0.246 \leftarrow$

# Stopping criteria

- ▶ Split the set according to the maximum gain
- ▶ Repeat until homogeneous or pure nodes (ie only leaves or entropy =0)



# Pseudo-code

- ▶ Simplified overview
- ▶ ID3(tree)
  - ▶ Find best attribute to split
  - ▶ Split dataset according to attribute (left-right) branches
  - ▶ Recurse on each branch until dataset is pure or entropy is 0

## Improvements?

- ▶ Entropy & information gain not sufficient metrics
- ▶ Missing data has to be handled
- ▶ Numerical values could provide order or dimension to a problem set
- ▶ Tree can be simplified



```

2,*,*,*,*,2
1,2,*,*,*,1
1,1,2,*,*,1
1,1,1,*,*,1
1,1,3,2,2,*,1
1,*,*,*,*,4,1
2,1,4,*,*,1,1
2,1,4,*,*,2,1
2,1,4,*,*,3,1
2,1,3,1,1,1,1
2,1,3,1,1,2,1
2,1,3,1,2,1,1
2,1,3,1,2,2,1
1,1,3,1,1,3,1
2,1,3,1,2,3,1
    
```

## Missing data II

- ▶ Datatypes can co-exist (eg. strings, integer/float)
- ▶ Solutions
  - ▶ Replace missing values in column with most frequent
  - ▶ For numerical values replace with mean/mode/median
- ▶ Column 2:
  - ▶ No instances = 15
  - ▶  $\text{Card}(2) = 1$
  - ▶  $\text{Card}(1) = 12$
  - ▶ → safest choice replace missing values with 1



# Numerical & continuous variables

- ▶ General approach separate categorical and continuous
- ▶ Our implementation:
  - ▶ Treat all numerical variables as continuous
  - ▶ C45 implementation based on a binary tree (computational gain)
  - ▶ → Everything equal or smaller than node value to the left
  - ▶ → Everything else to the right



## C4.5 I

- ▶ Simplifying a tree
  - ▶ Given a target gain level (generic or user-defined)
  - ▶ Prune (condense) the subtree
    - ▶ Might induce overclassification or errors
    - ▶ Decreases the depth of the tree
- ▶ 2 strategies:
  - ▶ **Pre-prune:**
    - ▶ Using statistical significance
    - ▶ → stop growing/building when no statistical significant association between any attribute and class at a node
    - ▶ chi-squared test ( too much statistics for us)
    - ▶ Pre-pruning may stop growing prematurely (eg. XOR stops at root node)



## C4.5 II

▶ **Post-prune:**

- ▶ Subtree replacement
- ▶ → Replace subtree with leaf
- ▶ → Stop when additional pruning is harmful
- ▶ → Accuracy default/user given
- ▶ → Using a validation set (derived from original)
- ▶ Subtree raising
- ▶ → Delete node & redistribute instances
- ▶ → Slower than replacement strategy

## C4.5

- ▶ Implementation differences to ID3
  - ▶ First test for missing values and replace
  - ▶ When splitting apply rule for numerical values
  - ▶ After growing the tree: prune

