Implementation of Decision Tree Classifiers
ID3 versus C4.5

Depuydt Antoine
Dany Efila
Mudura Mircea

May, 2017

Introduction

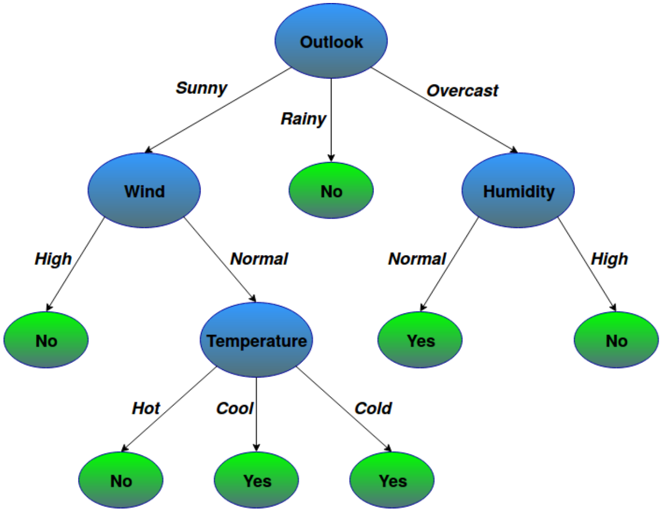
- Data mining: compress, understand and predict
  - Clustering
  - Classification
  - Regression
  - ...
- Techniques to find links
  - Linear Regression
  - Decision Trees
  - Neural Networks
  - ...

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



Premise

- Given a training data-set
- Recursively split on a node:
- If node is pure return leaf (class value)
- Else compute entropy & info gain:
  - Shannon's entropy: $E(S) = \sum_i -p_i log_2(p_i)$
  - Subtree gain: $Gain(T, X) = E(T) - E(T, X)$

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

ID3

- Initial implementation of decision trees
- Top down approach
- Split current node based on information gain:

ID3 - Learning algorithm

- Make a decision based on measurement of probability
  - How to decide? Split pocess
- Two main elements:
  - Entropy
  - Information gain

ID3 - Split procedure

- How to decide which node to split on?
- Two main elements:
  - Entropy
  - Information gain

ID3 - Entropy

- How to decide which node to split on?
- Two main elements:
  - Entropy
  - Information gain

ID3 - Information gain

- How to decide which node to split on?
- Two main elements:
  - Entropy
  - Information gain

ID3 - Pseudo code example

- How to decide which node to split on?
- Two main elements:
  - Entropy
  - Information gain

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

Missing data I

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

- Dataypes 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 $= 15$
  - Card$(2) = 1$
  - Card$(1) = 12$
  - $\rightarrow$ safest choice replace missing values with 1
- Column 3:
  - No instances $= 15$ (of course)
  - Card$(2) = 1$
  - Card$(1) = 1$
  - Card$(3) = 7$
  - Card$(4) = 3$
  - Missing $= 3$
  - $\rightarrow$ replace missing values with 3

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)
  - $\rightarrow$ Everithing equal or smaller than node value to the left
  - $\rightarrow$ 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
    - $\rightarrow$ 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

Pruning

- It's too complicated

K-fold cross validation

Demonstration