A Brief History of Decision Tree Implementation

MAX AUSTIN

Overview

- Famous Decision Tree Algorithms
  - Chi-squared Automatic Interaction Detector (CHAID)
  - Classification and Regression Tree (CART)
  - Iterative Dichotomiser 3 (ID3)
  - C4.5
- Personal Implementation

CHAID

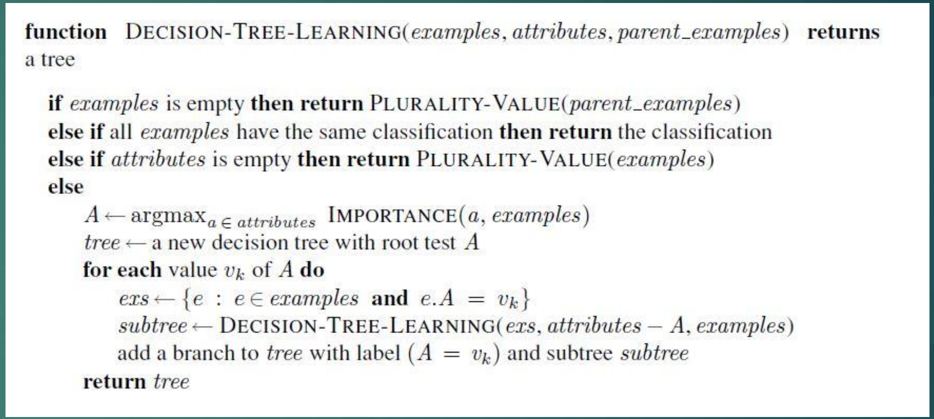
- Developed by Gordon V. Kass in 1980
- Builds non-binary trees
- Based on Bonferroni method
    - Allows multiple comparisons without a rise in Type I error
- Used particularly for analysis of large data sets
    - i.e. marketing research

CART

- Developed by Leo Breiman in 1984
- Binary tree
- Produces either classification trees or regression trees based on data
  - Classification trees predict the class or attribute of data
  - Regression trees predict the actual data value
- Split using Gini Index
  - $G = 1 – p_1^2 - p_2^2$
  - G = 0 -> purity

ID3 and C4.5

- Developed by John Ross Quinlan in 1986 and 1993
- Uses entropy to split data sets
- C4.5 implemented pruning and handles discrete and continuous data



```
function DECISION-TREE-LEARNING($examples, attributes, parent\_examples$) returns
a tree
  if $examples$ is empty then return PLURALITY-VALUE($parent\_examples$)
  else if all $examples$ have the same classification then return the classification
  else if $attributes$ is empty then return PLURALITY-VALUE($examples$)
  else
      $A \leftarrow \text{argmax}_{a \in attributes}$ IMPORTANCE($a, examples$)
      $tree \leftarrow$ a new decision tree with root test $A$
      for each value $v_k$ of $A$ do
          $exs \leftarrow \{e : e \in examples$ and $e.A = v_k\}$
          $subtree \leftarrow$ DECISION-TREE-LEARNING($exs, attributes - A, examples$)
          add a branch to $tree$ with label $(A = v_k)$ and subtree $subtree$
      return $tree$
```

Famous Decision Tree Implementation

Personal Implementation

- Mainly based off of C4.5 algorithm
- Does not prune tree
- Handles specifically nominal data
- Input files have possible attributes and features pre-defined

Basics of Algorithm

- Determines best split by calculating entropy and information gain
- Loops over all possible features for attribute
- Recurse through tree until a pure feature is found or you run out of possible attributes
- If no more attributes are available and there are multiple solutions possible, return the first one that occurs in the data

Results of Algorithm

- Weather = 71.43%
- Class = 60.00%
- DeerHunter = 55.36%