Decision Tree - Project 2, EDAF70

Ola Johansson (jur10ojo@student.lu.se)
Niklas Lundström (tpi13nlu@student.lu.se)

February 23, 2018

1 Decision Tree Learning

DTL The assignment consists creating a program centered around *supervised learning.* Specifically solvning a classification problem using *decision trees.* the task entails the reading of an ARFF-formated file containing attributes, classes and a data set mapping attribute values to a binary value of ”yes” or ”no” (1/0), creating a decision tree from this data using the ID3-algorithm, and finally pruning the tree and representing it visually.

2 Pruning

Using χ^2-pruning it is possible to statistically calculate which leaf-nodes are unnecessary to represent the tree created by the DTL-algorithm. By using the degrees of freedom (number of branches from a leaf-node), the desired precision, and the distribution of positive and negative outcomes for the examples at a given node in the tree, it is possible to decide whether or not to prune the node.

Design - Script Files

TreeNode.py A bare bone node class which enables the building of a decision tree through a list of children from each node, and what examples correspond to which child from any given node. The bulk of code in the TreeNode class is used for the visual representation of the complete tree.

Reader.py A script which contains only two functions. The main of these two is *read_file(file_name)* which takes the ARFF-formated data files path as its input and return the tree values: *attributes, classes*, and *data.*

Main.py This script file holds the bulk of the implementation of the ID3 algorithm. The main functionality is in the function *DTL(examples, attributes, parent_examples, orig_examples).* This mimics the structure of the algorithm given in *Artificial Intelligence, A Modern Approach* and calls serveral other functions to make the necessarty calculations. The *Importance*-function is implemented in the function *get_next_attribute(attributes, examples)*, which in turn uses a seperate function for calculation of gain.

The pruning algorithm is implemented in the functions *prune(curr_node, node_parent, data)* and *get_node_examples(node, examples)* where a table of χ^2-values is used to compare to the calculated Δ-values in order to determine which leaves to prune.

Data set - 'Restaurant' The data file *Restaurant* contains the data for this application. In the comments of this file the conventions used for translating string values in the examples into integers can be found. The file contents can be seen in the appendix.

Restaurant data set and resulting Decision Trees

The first section of the print shows the read attributes, classes, and data from the ARFF file. Following this is a decision tree generated by the application. Finally a tree were pruning using a p-value of 0.05 is shown.

```
The attributes are:
['alternate', 'bar', 'fri/sat', 'hungry', 'patrons', 'price',
'raining', 'reservation', 'type', 'waitestimate']
The classes are:
['0', '1']
The example data used is:
['x1', '1', '0', '0', '1', '1', '2', '0', '1', '0', '0', '1']
['x2', '1', '0', '0', '1', '2', '0', '0', '0', '1', '2', '0']
['x3', '0', '1', '0', '0', '1', '0', '0', '0', '2', '0', '1']
['x4', '1', '0', '1', '1', '2', '0', '1', '0', '1', '1', '1']
['x5', '1', '0', '1', '0', '2', '2', '0', '1', '0', '3', '0']
['x6', '0', '1', '0', '1', '1', '1', '1', '1', '3', '0', '1']
['x7', '0', '1', '0', '0', '0', '0', '1', '0', '2', '0', '0']
['x8', '0', '0', '0', '1', '1', '1', '1', '1', '1', '0', '1']
['x9', '0', '1', '1', '0', '2', '0', '1', '0', '2', '3', '0']
['x10', '1', '1', '1', '1', '2', '2', '0', '1', '3', '1', '0']
['x11', '0', '0', '0', '0', '0', '0', '0', '0', '1', '0', '0']
['x12', '1', '1', '1', '1', '2', '0', '0', '0', '2', '2', '1']
decision tree:
 patrons  =  0 :  0 ['x7', 'x11']
 patrons  =  1 :  1 ['x1', 'x3', 'x6', 'x8']
 patrons  =  2 ['x2', 'x4', 'x5', 'x9', 'x10', 'x12']
    hungry  =  0 :  0 ['x5', 'x9']
    hungry  =  1 ['x2', 'x4', 'x10', 'x12']
       type  =  0 :  1 []
       type  =  1 ['x2', 'x4']
          fri/sat  =  0 :  0 ['x2']
          fri/sat  =  1 :  1 ['x4']
       type  =  2 :  1 ['x12']
       type  =  3 :  0 ['x10']
pruned tree with p-value = 0.05 :
 patrons  =  0 :  0 ['x7', 'x11']
 patrons  =  1 :  1 ['x1', 'x3', 'x6', 'x8']
 patrons  =  2 :  0 ['x2', 'x4', 'x5', 'x9', 'x10', 'x12']
```

Comments

The results seem to be in line with those displayed in the textbook *Artificial Intelligence, A Modern Approach*, at least regarding the initial tree devoid of any pruning. Given this size of data the algorithm performs very fast and the generated tree is limited in size.

When applying pruning, no pruning is performed until the p-value $\leqslant 0.05$ at which point the χ^2-pruning removes all but the first node. This is due to an over-fitting of the given data. The limited available examples results in insignificant classifications in the tree which are removed by the pruning algorithm. Closer inspection of the algorithm showed that the pruning was done not necessarily because the attribute did not split the given examples well, but because the number of examples was small and the discrimination performed was thereby not deemed statistically significant.

Manual

To run the program *python3.5* and *scipy* is required. The

The main script *Main.py* is found at path /h/dk/x/jur10ojo/edaf70/decisiontree and is run by executing the command *python3.5 Main.py* from the command line. The program requires no input arguments and is not interactive. The application will terminate after printing the data and the generated trees, as can be seen in the printout above.

Appendix

Restaurant

```
_% 1. Title: Restaurant decisions
_%
_% 2. Sources:
_%       AIMA 2010
```

```
_%
_% For Patrons the following encoding is used:
_% None  = 0
_% Some  = 1
_% Full  = 2
_%
_% For Price the following encoding is used:
_% $     = 0
_% $$    = 1
_% $$$   = 2
_%
_% For Type the following encoding is used:
_% French    = 0
_% Thai      = 1
_% Burger    = 2
_% Italian   = 3
_%
_% For WaitEstimate the following encoding is used:
_% 0-10      = 0
_% 10-30     = 1
_% 30-60     = 2
_% >60       = 3
_%
_% All other attribute values are represented as
_% No = 0
_% Yes = 1

@ATTRIBUTE Alternate    NUMERIC
@ATTRIBUTE Bar          NUMERIC
@ATTRIBUTE Fri/Sat      NUMERIC
@ATTRIBUTE Hungry       NUMERIC
@ATTRIBUTE Patrons      NUMERIC
@ATTRIBUTE Price        NUMERIC
@ATTRIBUTE Raining      NUMERIC
@ATTRIBUTE Reservation  NUMERIC
@ATTRIBUTE Type         NUMERIC
@ATTRIBUTE WaitEstimate NUMERIC
@ATTRIBUTE class        {0,1}

@DATA
1,0,0,1,1,2,0,1,0,0,1
1,0,0,1,2,0,0,0,1,2,0
0,1,0,0,1,0,0,0,2,0,1
1,0,1,1,2,0,1,0,1,1,1
1,0,1,0,2,2,0,1,0,3,0
0,1,0,1,1,1,1,1,3,0,1
```

```
0,1,0,0,0,0,1,0,2,0,0
0,0,0,1,1,1,1,1,1,0,1
0,1,1,0,2,0,1,0,2,3,0
1,1,1,1,2,2,0,1,3,1,0
0,0,0,0,0,0,0,0,1,0,0
1,1,1,1,2,0,0,0,2,2,1
```