Decision Tree - Project 2, EDAF70

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

March 19, 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:
['yes', 'no']
The example data used is:
['x1', 0, 1, 1, 0, 1, 2, 1, 0, 0, 0, 0]
['x2', 0, 1, 1, 0, 2, 0, 1, 1, 1, 2, 1]
['x3', 1, 0, 1, 0, 1, 0, 1, 1, 2, 0, 0]
['x4', 0, 1, 0, 0, 2, 0, 0, 1, 1, 1, 0]
['x5', 0, 1, 0, 1, 2, 2, 1, 0, 0, 3, 1]
['x6', 1, 0, 1, 0, 1, 1, 0, 0, 3, 0, 0]
['x7', 1, 0, 1, 1, 0, 0, 0, 1, 2, 0, 1]
['x8', 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0]
['x9', 1, 0, 0, 1, 2, 0, 0, 1, 2, 3, 1]
['x10', 0, 0, 0, 0, 2, 2, 1, 0, 3, 1, 1]
['x11', 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1]
['x12', 0, 0, 0, 0, 2, 0, 1, 1, 2, 2, 0]

decision tree:
 patrons  =  none :  NO
 patrons  =  some :  YES
 patrons  =  full
    hungry  =  yes
       type  =  french :  NO
```

```
       type  =  thai
          fri/sat  =  yes :  YES
          fri/sat  =  no :  NO
       type  =  burger :  YES
       type  =  italian :  NO
    hungry  =  no :  NO

pruned tree:
Attemped to prune root of tree
 patrons  =  none :  NO
 patrons  =  some :  YES
 patrons  =  full :  NO
```

Election data set and resulting Decision Trees

The data set "Election" contains 435 data points (examples). Due to its length the full data set is not included in the report but can be found in the directory named in the manual below. The data has the following attributes:

```
@attribute 'handicapped-infants' {'n','y'}
@attribute 'water-project-cost-sharing' {'n','y'}
@attribute 'adoption-of-the-budget-resolution' {'n','y'}
@attribute 'physician-fee-freeze' {'n','y'}
@attribute 'el-salvador-aid' {'n','y'}
@attribute 'religious-groups-in-schools' {'n','y'}
@attribute 'anti-satellite-test-ban' {'n','y'}
@attribute 'aid-to-nicaraguan-contras' {'n','y'}
@attribute 'mx-missile' {'n','y'}
@attribute 'immigration' {'n','y'}
@attribute 'synfuels-corporation-cutback' {'n','y'}
@attribute 'education-spending' {'n','y'}
@attribute 'superfund-right-to-sue' {'n','y'}
@attribute 'crime' {'n','y'}
@attribute 'duty-free-exports' {'n','y'}
@attribute 'export-administration-act-south-africa' {'n','y'}
@attribute class {'democrat','republican'}
```

From this data we generate a decision tree and a pruned decision tree (with p = 0.05).

```
decision tree:
 'physician-fee-freeze'  =  'n'
    'adoption-of-the-budget-resolution'  =  'n'
       'synfuels-corporation-cutback'  =  'n'
          'superfund-right-to-sue'  =  'n'
             'mx-missile'  =  'n' :  'REPUBLICAN'
```

```
            'mx-missile' = 'y'
               'religious-groups-in-schools' = 'n'
                  'crime' = 'n' : 'DEMOCRAT'
                  'crime' = 'y' : 'REPUBLICAN'
               'religious-groups-in-schools' = 'y' : 'DEMOCRAT'
         'superfund-right-to-sue' = 'y' : 'DEMOCRAT'
      'synfuels-corporation-cutback' = 'y' : 'DEMOCRAT'
   'adoption-of-the-budget-resolution' = 'y' : 'DEMOCRAT'
 'physician-fee-freeze' = 'y'
   'education-spending' = 'n'
      'water-project-cost-sharing' = 'n'
         'anti-satellite-test-ban' = 'n'
            'handicapped-infants' = 'n' : 'DEMOCRAT'
            'handicapped-infants' = 'y' : 'REPUBLICAN'
         'anti-satellite-test-ban' = 'y' : 'REPUBLICAN'
      'water-project-cost-sharing' = 'y'
         'el-salvador-aid' = 'n'
            'immigration' = 'n' : 'DEMOCRAT'
            'immigration' = 'y'
               'duty-free-exports' = 'n'
                  'aid-to-nicaraguan-contras' = 'n' : 'DEMOCRAT'
                  'aid-to-nicaraguan-contras' = 'y'
                     'export-administration-act-south-africa' = 'n' : 'DEMOCRAT'
                     'export-administration-act-south-africa' = 'y' : 'REPUBLICAN'
               'duty-free-exports' = 'y' : 'DEMOCRAT'
         'el-salvador-aid' = 'y' : 'REPUBLICAN'
   'education-spending' = 'y' : 'REPUBLICAN'

pruned tree:
 'physician-fee-freeze' = 'n' : 'DEMOCRAT'
 'physician-fee-freeze' = 'y'
   'education-spending' = 'n'
      'water-project-cost-sharing' = 'n' : 'REPUBLICAN'
      'water-project-cost-sharing' = 'y'
         'el-salvador-aid' = 'n'
            'immigration' = 'n' : 'DEMOCRAT'
            'immigration' = 'y'
               'duty-free-exports' = 'n'
                  'aid-to-nicaraguan-contras' = 'n' : 'DEMOCRAT'
                  'aid-to-nicaraguan-contras' = 'y'
                     'export-administration-act-south-africa' = 'n' : 'DEMOCRAT'
                     'export-administration-act-south-africa' = 'y' : 'REPUBLICAN'
               'duty-free-exports' = 'y' : 'DEMOCRAT'
         'el-salvador-aid' = 'y' : 'REPUBLICAN'
   'education-spending' = 'y' : 'REPUBLICAN'
```

Comments

The results of the restaurant example 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 to the restaurant example, the algorithm is trying to prune the whole tree. This is due to an overfitting 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.

However, in the election example we see that a pruning was done, but still some significant differences were found between some of the branches in the tree.

Manual

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

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 <data file name>* from the command line with the name of the input data file as argument. 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
%

@ATTRIBUTE Alternate    {yes,no}
@ATTRIBUTE Bar          {yes,no}
@ATTRIBUTE Fri/Sat      {yes,no}
@ATTRIBUTE Hungry       {yes,no}
@ATTRIBUTE Patrons      {none,some,full}
@ATTRIBUTE Price        {$,$$,$$$}
@ATTRIBUTE Raining      {yes,no}
@ATTRIBUTE Reservation  {yes,no}
@ATTRIBUTE Type         {french,thai,burger,italian}
@ATTRIBUTE WaitEstimate {0-10,10-30,30-60,>60}
@ATTRIBUTE class        {yes,no}

@DATA
yes,no,no,yes,some,$$$,no,yes,french,0-10,yes
yes,no,no,yes,full,$,no,no,thai,30-60,no
no,yes,no,yes,some,$,no,no,burger,0-10,yes
yes,no,yes,yes,full,$,yes,no,thai,10-30,yes
yes,no,yes,no,full,$$$,no,yes,french,>60,no
no,yes,no,yes,some,$$,yes,yes,italian,0-10,yes
no,yes,no,no,none,$,yes,no,burger,0-10,no
no,no,no,yes,some,$$,yes,yes,thai,0-10,yes
no,yes,yes,no,full,$,yes,no,burger,>60,no
yes,yes,yes,yes,full,$$$,no,yes,italian,10-30,no
no,no,no,no,none,$,no,no,thai,0-10,no
yes,yes,yes,yes,full,$,no,no,burger,30-60,yes
```