Department of Computer Science

Artificial Intelligence
COS314

Project 2: Decision Trees

Due: Wednesday 1 May, 08:00

For this project you will develop a decision tree induction algorithm.

Below are the rules for this project:

- The project is done by each student individually, and all code has to be written by yourself.
- You may implement the project either in C++, C, Java, Python, or Scala, and must run on Linux. You have to provide a Makefile, ant file, or any other alternative project script that will run on Linux. Describe in the pdf file how the project should be compiled and used.
- Your program has to accept command line parameters, to be executed as follows:

  `./DecisionTree <options> data.spec data.dat`

  where `DecisionTree` is the name of the program, `<options>` specifies one of a number of options (see the different options below), `data.spec` provides a specification of the attributes and class(es), and `data.dat` is the name of your data file.
- Although this is not a programming course, you should always provide well-structured, modular, well-written and documented code.
- Submit your project before the deadline via the online submission system as a compressed tar archive. Provide a pdf file, *project.pdf* in the root file to which your tar ball extracts, wherein you explain how the program should be compiled and used, give your name, surname and student number. Indicate in this pdf file which options have been implemented.
- You can use any of the classification data sets available at the UCI Machine Learning Repository: http://archive.ics.uci.edu/ml/
  It will be good if you test on more than one of these data sets.

Your decision tree induction algorithm should be able to do the following:

- Induce a decision tree on a given data set, by making use of the gain ratio criterion as given on slide 7 of the decision tree slides. The given data set has to be randomly split into two subsets: A training set containing 70% of the data patterns and a test set containing 30% of the test set.

- The following options will be used to run your decision tree algorithm:
  - `d`: To induce the decision tree on data sets that have only discrete-valued attributes
  - `c`: To induce a decision tree on data sets that, in addition to discrete-valued attributes, also contain continuous-valued attributes.
  - `md`: To induce a decision tree on data sets that have only discrete-valued attributes, but may contain missing values.
  - `mc`: As for option `c`, but the data set may contain missing values.
  - `pd`: As for option `d`, but after induction of the tree, you have to prune the tree to prevent overfitting.
  - `pc`: As for option `pd`, but there may be continuous-valued attributes.

  When any of these options does not work (you have not implemented them), then just display an appropriate error message.
- Your decision tree should handle any number of classes.
- After each tree induction, the following output has to be displayed on the screen, and be outputted to the data file `data.out`, in a readible format:
  - The classification error on the training set and the test set.
  - A list of each of the classification rules, in the following format:

    ```
    IF (attr1 relop value) AND (attr2 relop value) AND ... AND (attrN relop value) THEN
       className is classValue
    ```

    where `attr1, attr2, ...` refer to attribute names, `relop` is a relational operator, `value` is the value on which the split is done, `className` is the name of the class, and `classValue` is the value of the class variable.

    Remember that the number of rules is equal to the number of leaf nodes. So, one rule is one traversal from the root to a leaf.
  - For each rule, indicate the classification error on the training set and the test set, and indicate the total number of data patters associated with that rule, and for each class, the number of data patterns associated for that class.
  - For options `pc` and `pd`, display the above information for both the original tree and the pruned tree.

If you implement pruning of the decision tree, do it as follows: When considering to replace an intermediate (decision) node with a leaf node, calculate the classification error on the test set before the pruning and after the pruning. If the generalization error does not deteriorate more than 5%, then do the pruning. Otherwise, do not prune.

Your program will receive two files, i.e. `data.spec` and `data.dat`. The first file provides a specification of the attributes and classes, while the latter contains the actual data patterns. For the specification file, the format is as follows (note that the files that I will use to test your program will use these formats):

- Each line specifies one attribute.
- The first line specifies the class attribute as follows:

  ```
  className: value1 value2 value3 ... valueN
  ```

  where `className` is the name of the class, and `value1`, ..., `valueN` are the finite set of values that the class attribute can assume.
- The remainder of the lines specify an attribute each, as follows:

  ```
  attributeName: Type
  ```

  where `attributeName` is the name of the attribute and `Type` indicates the type of that attribute. For continuous-valued attributes, the type will be given as `Real`, and for discrete-valued attributes the type will be given as `{ value1, value2, ..., valueM }`, where the M finite values that the attribute can have, are listed.

For the data file, use the following format:

```
attrVal1 attrVal2 ... attrValM classValue
```

where each row contains the values for each attribute in the same order that they are provided in the specification file, and `classValue` is the associated target class. If an attribute has a missing value for a specific pattern, then indicate this with a `?`.

The project will be evaluated using the following marking scheme.

Decision tree induction option:	d	50
	c	15
	md	30
	mc	15
	pd	20
	pc	10
	Sub-total	**140**
Output:	On screen	10
	Output file	10
	Sub-total	**20**
Coding style:		
	Modularity	10
	Documentation	10
	Sub-total	**20**
Total:		**180**