GenClass Manual

Ioannis G. Tsoulos[(1)][*], Alexandros Tzallas[(1)], Dimitris Tsalikakis[(2)]

[(1)]Department of Computer Engineering, School of Applied Technology, Technological Educational Institute of Epirus, 47100 Arta, Greece
[(2)]University of Western Macedonia, Department of Engineering Informatics and Telecommunications, Greece

1 Introduction

GenClass is a software entirely written in ANSI C++ that constructs classification programs in a C – like programming language in order to classify the input data, producing simple if – else rules. The dataset should conform to the format outlined below:

D
M

$$\begin{array}{ccccc} x_{11} & x_{12} & \cdots & x_{1D} & y_1 \\ x_{21} & x_{22} & \cdots & x_{2D} & y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{M1} & x_{M2} & \cdots & x_{MD} & y_M \end{array}$$

The integer value D determines the dimensionality of the problem and the value M determines the number of points in the file. Every subsequent line contains a pattern and the final column is the real output (category) for this pattern. The number of the classes is induced from the file. The software scans the file and identifies the number of problem's classes. The classes should be integer numbers with number 0 assigned to the first class.

2 Distribution

The package is distributed in a zip file from Github `https://github.com/itsoulos/GenClass` named `GenClass-master.zip` and under UNIX systems the user must execute the command: `unzip GenClass-master`. This command creates a directory named `GenClass` with the following contents:

[*]Corresponding author. Email: itsoulos@teiep.gr. Phone: +302681050344

1. **bin**: A directory which is initially empty. After compilation of the package, it will contain the executable **genclass**
2. **doc**: This directory contains the documentation of the package (this file) in different formats: A LyX file, A LaTeX file and a PostScript file.
3. **examples**: A directory that contains some test functions.
4. **include**: A directory which contains the header files for all the classes of the package.
5. **src**: A directory containing the source files of the package.
6. **Makefile**: The input file to the `make` utility in order to build the tool. Usually, the user does not need to change this file.
7. **Makefile.inc**: The file that contains some configuration parameters, such as the name of the C++ compiler etc. The user must edit and change this file before installation.

3 Installation

The following steps are required in order to build the tool:

1. Uncompress the tool as described in the previous section.
2. `cd GenClass`
3. Edit the file `Makefile.inc` and change (if needed) the configuration parameters.
4. Type `make`.

The parameters in `Makefile.inc` are the following:

1. **CXX**: It is the most important parameter. It specifies the name of the C++ compiler. In most systems running the GNU C++ compiler this parameter must be set to g++.
2. **ROOTDIR**: Is the location of the GenClass directory.

4 The executable genclass

The outcome of the compilation is the executable `genclass` under the directory `bin`. The executable has the following command line parameters:

1. `-h`:The program prints a help screen and afterwards the program terminates.

2. `-c count`: The integer parameter `count` determines the number of chromosomes for the genetic population. The default value for this parameter is 500.

3. -g **gens**: The integer parameter **gens** determines the maximum number of generations allowed for the genetic algorithm. The default value is 200.

4. `-s srate`: The double parameter `srate` specifies the selection rate used in the genetic algorithm. The default value for this parameter is 0.10 (10%).

5. `-m mrate`: The double parameter `mrate` specifies the mutation rate used in the genetic algorithm. The default value for this parameter is 0.05 (5%).

6. -l **size**: The integer parameter **size** determines the size of every chromosome in the genetic population. The default value for this parameter is 100.

7. -p **train_file**: The string parameter **train_file** specifies the file containing the points that will be used as train data for the algorithm.

8. -t **test_file**: The string parameter **test_file** specifies the file containing the test data for the particular problem. The file should be in the same format as the **train_file**.

9. -w **wrapping**. The integer parameter **wrapping** determines the maximum number of wrapping events allowed. The default value for this parameter is 1.

10. -f **foldcount**. The integer parameter **foldcount** specifies the number of fold to be used for cross validation. The default value for this parameter is 0 (no cross validation).

11. -o **method**: The string parameter method specifies the output method for the executable. The available options are

    (a) simple. The program prints output only on termination.

    (b) csv. The program prints in csv (comma separated value) format information in every generation. In every generation the program prints: number of generations, train error and test error. This is the default value for the string parameter **method**.

    (c) full. The program prints in every generation detailed information about the optimization procedure as well as classification error for every distinct class of the problem.

12. `-r seed`: The integer parameter `seed` specifies the seed for the random number generator. It can assume any integer value.

5 A typical example

Consider the Ionosphere dataset available from the Machine Learning Repository in the following URL: http://www.ics.uci.edu/~mlearn/MLRepository.html. The ionosphere dataset contains data from the Johns Hopkins Ionosphere database. The two-class dataset contains 351 examples of 34 features each. The datasets has been divided into two files, `ionosphere.train` and `ionosphere.test` under directory `examples`. A typical run for the `GenClass` will be

```
../bin/genclass -p ionosphere.train -t ionosphere.test -g 10 -o csv
```

The output of this command is:

```
1,     15.43,     19.32
   2,     15.43,     19.32
   3,     15.43,     19.32
   4,     13.71,     17.05
   5,     12.57,     15.34
   6,     12.57,     15.34
   7,     12.57,     15.34
   8,        12,     13.64
   9,        12,     13.64
  FINAL OUTPUT EXPRESSION= if(!(x7<log(cos(cos(((-788.787)+
  ((sin(x28)/sin(cos(((-7.17)/x34))))+(-83.6))))))|x6>x13&x7<log(x5))) CLASS=0.00
else  CLASS=1.00
TRAIN ERROR = 12.00%
CLASS ERROR = 13.64%
** CONFUSION MATRIX ** Number of classes: 2
 102   3
  21  50
```