

GenClass: A parallel tool for data classification based on Grammatical Evolution

Nikolaos Anastasopoulos⁽¹⁾, Ioannis G. Tsoulos⁽²⁾, Alexandros Tzallas⁽²⁾

⁽¹⁾*Department of Electrical and Computer Engineering, University of Patras, Greece*

⁽²⁾*Department of Informatics and Telecommunications, University of Ioannina, 47100 Arta, Greece*

Abstract

A genetic programming tool is proposed here for data classification. The tool is based on Grammatical Evolution technique and it is designed to exploit multicore computing systems using the OpenMp library. The tool constructs classification programs in a C – like programming language in order to classify the input data, producing simple if – else rules and upon termination the tool produces the classification rules in C and Python files. Also, the user can use his own BNF grammar through a command line option. The tool is tested on a wide range of classification problems and the produced results are compared against traditional techniques for data classification, yielding very promising results.

Keywords: Genetic algorithm, Data classification, Grammatical evolution, Stochastic methods

1. Introduction

Data classification finds many applications on a series of practical problems from areas such as chemistry [1, 2, 3], biology [4, 5], economics [6, 7], physics [8, 9] etc. During the past years many methods have been proposed to problems of this category such as neural networks [10], radial basis functions networks [11], support vector machines [12], etc. The proposed method, which is initially described in [13], constructs classification programs in a human readable format using the technique of Grammatical Evolution [14]. Grammatical evolution is an evolutionary process that has been applied with success in many areas such as music composition [15], economics [16], symbolic regression [17], robot control [18] and caching algorithms [23].

This article introduces a software tool (named GenClass) coded in ANSI C++, for data classification using classification rules in human readable

form. The rules are created with the Grammatical evolution method and can be applied in any classification problem, without a priory knowledge of the dataset. Also, this tool has a series of command line options to control the various aspects of the software, such as mutation rate or number of chromosomes in the genetic population. Another similar software is the Parallel Program Induction (PPI) available from the relevant URL <https://github.com/daaugusto/ppi>, which constructs with Grammatical Evolution programs for regression and classification, but the PPI software has many dependencies on other software to be compiled. Also, GenClass produces classification problems in human readable format and it can export output in C++ and in Python. However, in the current version GenClass does not solve regression problems. Also, recently the software NNC has been published [24], which incorporates the Grammatical Evolution to construct neural networks for regression and classification problems. Even though the GenClass software can not used for regression problems in the current form it has similar execution runs as the NNC software and also it can discover the hidden relationships between the features of a dataset and the expected output. Also, the software GenClass can use every core of a multicore environment and it has the ability to load a BNF grammar from a separate input file.

The rest of this article is organized as follows: in section 2 the software is described in detail, in section 3 a series of experiments on some well - known classification datasets are demonstrated, in section 4 the impact of the software tool is discussed and finally in section 5 conclusions and guidelines for further expansion of the software are presented.

2. Software description

2.1. The proposed algorithm

The main steps of the algorithm are:

1. Initialization step.

- (a) Read the train data
- (b) Set N_G the maximum number of generations, N_C as the number of chromosomes, P_S the selection rate, P_M the mutation rate.
- (c) Initialize the chromosomes of the population.

2. Genetic step

- (a) **For** $i = 1, \dots, N_g$ **do**
 - i. Create for every chromosome in the population a classification program using Grammatical Evolution.

- 51 ii. Calculate the fitness for every chromosome of the population
- 52 iii. Execute the genetic operators of selection with rate P_S and
- 53 mutation with rate P
- 54 (b) **EndFor**
- 55 3. **Evaluation** step
- 56 (a) Create a classification program for the best chromosome in the
- 57 population.
- 58 (b) Apply the previous program to test set and report the induced
- 59 error.

60 The installation of the software is explained in detail in the main page <https://github.com/itsoulos/GenClass/>.

62 2.2. *The executable genclass*

63 The outcome of the software compilation and installation is the executable
64 **genclass** under the directory **bin**. The executable has the following com-
65 mand line parameters:

- 66 1. **-h**: The program prints a help screen.
- 67 2. **-c count**: The parameter **count** determines the number of chromo-
68 somes with default value 500.
- 69 3. **-f count**. Specify fold count for fold validation. Default value 0 (no
70 folding).
- 71 4. **-i grammarfile**. The string parameter **grammarfile** stands for the
72 user defined grammar in BNF format. If this options is empty, then
73 the default grammar will be used. An example of such a file is listed
74 in Figure 5 and it is included under examples subdirectory of the dis-
75 tribution.
- 76 5. **-g gens**: The parameter **gens** determines the maximum number of
77 generations with default value is 200.
- 78 6. **-d count**. The parameter **d** determines the maximum number of threads
79 used by the OpenMp library. The default value is 16.
- 80 7. **-s srate**: The parameter **srate** specifies the selection rate with default
81 value 0.10 (10%).
- 82 8. **-m mrate**: The parameter **mrate** specifies the mutation rate with de-
83 fault value 0.05 (5%).
- 84 9. **-l size**: The parameter **size** determines the size of every chromosome
85 with default value 100.
- 86 10. **-p train_file**: The string parameter **train_file** specifies the file con-
87 taining the points that will be used as train data for the algorithm.
88 The file should conform to the format outlined in figure 1.

- 89 11. **-t test_file**: The string parameter **test_file** specifies the file containing
 90 the test data for the particular problem. The file should be in the same
 91 format as the **train_file**.
- 92 12. **-o method**: The string parameter **method** specifies the output method
 93 for the executable. The available options are
 - 94 (a) **simple**. The program prints output only on termination.
 - 95 (b) **csv**. In every generation the program prints: number of genera-
 96 tions, train error and test error.
 - 97 (c) **full**. The program prints in every generation detailed information
 98 about the optimization procedure.
- 99 13. **-r seed**: The integer parameter **seed** specifies the seed for the random
 100 number generator. It can assume any integer value.

101 2.3. The output files

102 The software produces upon termination two distinct files that contains
 103 the classification rules. The first file is written in ANSI C++ named clas-
 104 sifier.c and an example is shown in Figure 2. The second file is written in
 105 Python named classifier.py and an example is displayed in Figure 3.

106 3. Experiments

107 3.1. A typical example

108 Consider the Ionosphere dataset available from the Machine Learning
 109 Repository. The ionosphere dataset contains data from the Johns Hop-
 110 kins Ionosphere database. The dataset has been divided into two files,
 111 **ionosphere.train** and **ionosphere.test** under directory **examples** of the
 112 distribution. A typical run for the **GenClass** will be

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

114 The output of this command is shown in Figure 4.

115 3.2. Experiments

116 In order to measure the efficiency of the proposed method a series of
 117 experiments were conducted on some common classification problems found
 118 in two major dataset databases:

- 119 1. UCI dataset repository, [https://archive.ics.uci.edu/ml/index.](https://archive.ics.uci.edu/ml/index.php)
 120 [php](https://archive.ics.uci.edu/ml/index.php)
- 121 2. Keel repository, <https://sci2s.ugr.es/keel/datasets.php>[19].

122 All the experiments were conducted 30 times using different seed for the
123 random generator each time and averages were taken. Also, to validate
124 the results 10 - fold cross validation was used in all experiments. In all
125 experiments we have used the parameters shown in table 1. The following
126 datasets were used

- 127 1. **Wine** dataset. The wine recognition dataset contains data from wine
128 chemical analysis.
- 129 2. **Glass** dataset. The dataset contains glass component analysis for glass
130 pieces that belong to 6 classes.
- 131 3. **Tae** dataset. The data consist of evaluations of teaching performance
132 for the University of Wisconsin-Madison.
- 133 4. **Spiral** dataset: The spiral artificial dataset contains 1000 two-dimensional
134 examples that belong to two classes (500 examples each). The num-
135 ber of the features is 2. The data in the first class are created using
136 the following formula: $x_1 = 0.5t \cos(0.08t)$, $x_2 = 0.5t \cos(0.08t + \frac{\pi}{2})$
137 and the second class data using: $x_1 = 0.5t \cos(0.08t + \pi)$, $x_2 =$
138 $0.5t \cos(0.08t + \frac{3\pi}{2})$
- 139 5. **Pima** dataset. The Pima Indians Diabetes dataset contains with two
140 categories: healthy and diabetic.
- 141 6. **Ionosphere** dataset. The ionosphere dataset (ION in the following
142 tables) contains data from the Johns Hopkins Ionosphere database.
- 143 7. **Appendictis** dataset, proposed in [20].
- 144 8. **Australian**, the dataset concerns credit card applications.
- 145 9. **Hayes roth** dataset. This dataset[21] contains 5 numeric-valued at-
146 tributes and 132 patterns.
- 147 10. **Alcohol**, a dataset about Alcohol consumption [22].
- 148 11. **Dermatology**. Dataset used for differential diagnosis of erythemat-
149 squamous diseases.
- 150 12. **Balance**. This data set was generated to model psychological experi-
151 mental results.
- 152 13. **Regions2** dataset. It is created from liver biopsy images of patients
153 with hepatitis C [26].
- 154 14. **Parkinsons**. This dataset is composed of a range of biomedical voice
155 measurements from 31 people, 23 with Parkinson’s disease (PD)[25].
- 156 15. **Wdbc**. The Wisconsin diagnostic breast cancer dataset (WDBC) con-
157 tains data for breast tumors.
- 158 16. **Popfailures**. This dataset contains records of simulation crashes en-
159 countered during climate model.
- 160 17. **Heart**. The task is to detect the absence or presence of heart disease.

- 161 18. **Ecoli**. The goal here is to predict the localization site of proteins.
- 162 19. **Haberman**. A dataset about breast cancer from a study at the Uni-
- 163 university of Chicago's Billings Hospital.
- 164 20. **HouseVotes**. This data set includes votes for each of the U.S. House
- 165 of Representatives Congressmen on the 16 key votes.
- 166 21. **Shuttle**. The task is to decide what type of control of a spacecraft
- 167 should be employed.
- 168 22. **Lymography**. The aim here is to detect the presence of a lymphoma
- 169 in patients.
- 170 23. **Mammographic**. This dataset be used to identify the severity (benign
- 171 or malignant) of a mammographic mass lesion.
- 172 24. **OptDigits**. Optical Recognition of Handwritten Digits data set.
- 173 25. **Page Blocks**. The dataset contains blocks of the page layout of a
- 174 document that has been detected by a segmentation process.
- 175 26. **Penbased**. This is a Pen-Based Recognition of Handwritten Digits
- 176 data set.
- 177 27. **Saheart**. The dataset is about to categorize persons if have a coronary
- 178 heart disease.
- 179 28. **Segment**. This database contains patterns from a database of 7 out-
- 180 door images (classes).
- 181 29. **Thyroid**. The goal in this dataset is to detect if a patient is normal
- 182 or suffers from hyperthyroidism or hypothyroidism.
- 183 30. **Eeg** datasets. As an real word example, consider an EEG dataset
- 184 described in [27] is used here. The dataset consists of five sets (denoted
- 185 as Z, O, N, F and S) each containing 100 single-channel EEG segments
- 186 each having 23.6 sec duration. With different combinations of these sets
- 187 the produced datasets are Z_F_S, ZO_NF_S, ZONF_S and Z_O_N_F_S.

188 The results from the experiments are displayed in table 2. The column Neural
189 stands for the application of a Neural network with 10 processing nodes and
190 the usage of a BFGS variant due to Powell [28] as the training method.
191 The column Mlp-Gen stands for the application of a neural network with 10
192 processing nodes and the utilization of a genetic algorithm as the training
193 method. The genetic algorithm used has the same parameters as in the case
194 of GenClass method, in order to make the results comparable. The column
195 RBF stands for the application of an RBF network using 10 centers. Also,
196 the column RBF-Gen represents the application of an RBF network with
197 10 processing units that is trained using a genetic algorithm with the same
198 set of parameters as in GenClass. Also a statistical comparison is displayed
199 in Figure 6. It is evident, that the proposed method outperforms the other

200 methods in terms of efficiency in the majority of the objective problems.
 201 Of course, the method could be slower than RBF or Neural network due
 202 to the usage of Grammatical Evolution, however this increase in time can
 203 be significantly reduced with the use of threads. In addition, the software
 204 appears to be slightly better in some datasets than originally published [13]
 205 due to the use of far more chromosomes in the genetic population.

206 4. Impact

207 This software tool produces human readable classification rules in ANSI
 208 C++ as well as in Python format. This rules used to classify any classification
 209 problem without a priory knowledge about the nature of the problem. The
 210 implemented software utilizes the Grammatical Evolution technique to pro-
 211 duce the classification rules based on simple BNF grammar. The software is
 212 guided by a series of command line options that control many critical options
 213 of the underlying method such as the number of chromosomes, the mutation
 214 rate etc. Also, the software is designed to be use in multi core computational
 215 environments through the public available library of OpenMP. The proposed
 216 method proved to be very reliable on a series of simple and hard classifica-
 217 tion problems and hence the software can be used in different areas such as
 218 physics, chemistry, medicine etc. The user need to provide only the patterns
 219 of the classification problem in a simple text format and if it is required a
 220 series of command line options. Also, the user can provide his own grammar
 221 in BNF format in a separate text file. Finally, the software can be easily
 222 extended to be more friendly for all users, through a graphical interface.

223 5. Conclusions

224 A software which implements a method for data classifications was intro-
 225 duced. The method is based on grammatical evolution and the software is
 226 designed to be portable. Future versions of the software will include

- 227 • Input from various formats such as CSV, Json etc.
- 228 • Usage of improvement stopping rules for the genetic algorithm.
- 229 • Additional output formats.

230 References

- 231 [1] C. Güler, G. D. Thyne, J. E. McCray, K.A. Turner, Evaluation of graph-
 232 ical and multivariate statistical methods for classification of water chem-
 233 istry data, Hydrogeology Journal **10**, pp. 455-474, 2002

- 234 [2] E. Byvatov ,U. Fechner ,J. Sadowski , G. Schneider, Comparison of
235 Support Vector Machine and Artificial Neural Network Systems for
236 Drug/Nondrug Classification, *J. Chem. Inf. Comput. Sci.* **43**, pp 1882–
237 1889, 2003.
- 238 [3] Kunwar P. Singh, Ankita Basant, Amrita Malik, Gunja Jain, Arti-
239 ficial neural network modeling of the river water quality—A case study,
240 *Ecological Modelling* **220**, pp. 888-895, 2009.
- 241 [4] I. Guyon, J. Weston, S. Barnhill, V. Vapnik, Gene Selection for Cancer
242 Classification using Support Vector Machines, *Machine Learning* **46**, pp.
243 389-422, 2002.
- 244 [5] R. Marabini, J.M. Carazo, Pattern recognition and classification of im-
245 ages of biological macromolecules using artificial neural networks, *Bio-
246 physical Journal* **66**, pp. 1804-1814, 1994.
- 247 [6] I. Kaastra, M. Boyd, Designing a neural network for forecasting financial
248 and economic time series, *Neurocomputing* **10**, pp. 215-236, 1996.
- 249 [7] Moshe Leshno, Yishay Spector, Neural network prediction analysis: The
250 bankruptcy case, *Neurocomputing* **10**, pp. 125-147, 1996.
- 251 [8] S. R. Folkes,O. Lahav, S. J. Maddox, An artificial neural network ap-
252 proach to the classification of galaxy spectra, *Montly Notices of the
253 Royal Astronomical Society* **283**, pp 651-665, 1996.
- 254 [9] C.Z. Cai, W.L. Wang, Y.Z. Chen, Support vector machine classification
255 of physical and biological datasets, *Int. J. Mod. Phys. C* **14**, pp. 575,
256 2003
- 257 [10] K. Hornik, Multilayer feedforward networks are universal approxima-
258 tors, *Neural Networks* **2**, pp. 359-366, 1989
- 259 [11] M.D. Buhmann, Radial basis functions, Cambridge monographs on Ap-
260 plied and Computational Mathematics, 2003.
- 261 [12] I. Steinwart, A. Christmann, Support Vector Machines, *Information Sci-
262 ence and Statistics*, Springer, 2008.
- 263 [13] Ioannis G. Tsoulos, Creating classification rules using grammatical evo-
264 lution, *International Journal of Computational Intelligence Studies* **9**,
265 pp. 161-171, 2020.

- [14] M. O'Neill, C. Ryan, Grammatical evolution, *IEEE Trans. Evol. Comput.* **5**, pp. 349–358, 2001.
- [15] Alfonso Ortega, Rafael Sánchez, Manuel Alfonseca Moreno, Automatic composition of music by means of grammatical evolution, *APL '02 Proceedings of the 2002 conference on APL: array processing languages: lore, problems, and applications* Pages 148 - 155.
- [16] Michael O'Neill, Anthony Brabazon, Conor Ryan, J. J. Collins, Evolving Market Index Trading Rules Using Grammatical Evolution, *Applications of Evolutionary Computing Volume 2037 of the series Lecture Notes in Computer Science* pp 343-352.
- [17] M. O'Neill, C. Ryan, Grammatical Evolution: Evolutionary Automatic Programming in a Arbitrary Language, *Genetic Programming*, vol. 4, Kluwer Academic Publishers, Dordrecht, 2003
- [18] J.J. Collins, C. Ryan, in: *Proceedings of AROB 2000, Fifth International Symposium on Artificial Life and Robotics*, 2000.
- [19] J. Alcalá-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera. KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. *Journal of Multiple-Valued Logic and Soft Computing* **17**, pp. 255-287, 2011.
- [20] Weiss, Sholom M. and Kulikowski, Casimir A., *Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*, Morgan Kaufmann Publishers Inc, 1991.
- [21] B. Hayes-Roth, B., F. Hayes-Roth. Concept learning and the recognition and classification of exemplars. *Journal of Verbal Learning and Verbal Behavior* **16**, pp. 321-338, 1977.
- [22] Tzimourta, Katerina D. and Tsoulos, Ioannis and Bilerio, Thanasis and Tzallas, Alexandros T. and Tsipouras, Markos G. and Giannakeas, Nikolaos, Direct Assessment of Alcohol Consumption in Mental State Using Brain Computer Interfaces and Grammatical Evolution, *Inventions* **3**, pp. 1-12, 2018.
- [23] M. O'Neill, C. Ryan, in: K. Miettinen, M.M. Mkel, P. Neittaanmki, J. Periaux (Eds.), *Evolutionary Algorithms in Engineering and Computer Science*, Jyväskylä, Finland, 1999, pp. 127–134.

Figure 1: Data format. 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.

D
M
 x_{11} x_{12} \dots x_{1D} y_1
 x_{21} x_{22} \dots x_{2D} y_2
 \vdots \vdots \vdots \vdots \vdots
 x_{M1} x_{M2} \dots x_{MD} y_M

Table 1: Parameters for the experiments.

PARAMETER	VALUE
N_C	500
N_G	500
P_S	90%
P_M	5%

- 301 [24] Ioannis G. Tsoulos, Alexandros Tzallas, Dimitris Tsalikakis, NNC: A
302 tool based on Grammatical Evolution for data classification and differ-
303 ential equation solving, Software X **10**, 2019.
- 304 [25] Little MA, McSharpy PE, Hunter EJ, Spielman J, Ramig LO.
305 Suitability of dysphonia measurements for telemonitoring of
306 Parkinson’s disease. IEEE Trans Biomed Eng. 2009;56(4):1015.
307 doi:10.1109/TBME.2008.2005954
- 308 [26] Giannakeas, N., Tsipouras, M.G., Tzallas, A.T., Kyriakidi, K., Tsianou,
309 Z.E., Manousou, P., Hall, A., Karvounis, E.C., Tsianos, V., Tsianos,
310 E. A clustering based method for collagen proportional area extraction
311 in liver biopsy images (2015) Proceedings of the Annual International
312 Conference of the IEEE Engineering in Medicine and Biology Society,
313 EMBS, 2015-November, art. no. 7319047, pp. 3097-3100.
- 314 [27] R.G. Andrzejak, K. Lehnertz, F. Mormann, C. Rieke, P. David, and C.
315 E. Elger, Indications of nonlinear deterministic and finite-dimensional
316 structures in time series of brain electrical activity: Dependence on
317 recording region and brain state, Phys. Rev. E **64**, pp. 1-8, 2001.
- 318 [28] M.J.D Powell, A Tolerant Algorithm for Linearly Constrained Optimiza-
319 tion Calculations, Mathematical Programming **45**, pp. 547-566, 1989.

Figure 2: A typical produced C file of the tool. The double array x is the input and the value class the estimated class of the method.

```
#include <math.h>
int classifier(double *x)
{
    int CLASS = 0;
    if (!(x[6]<x[25] || x[4]<(((x[5]-((-5.79)/x[0]))/
        ((-503.562)/x[22])/ (x[20]/x[24])))/(x[0]+(x[23]/x[4]))
        ||!(x[23]<=x[2]))) CLASS=0;
    else CLASS=1;
    return CLASS;
}
```

Figure 3: A typical output file in Python language.

```
import ctypes
import numpy as np from typing
import List
def classifier(input: List):
    fun = ctypes.CDLL("./classifier.so")
    fun.classifier.argtypes = [ctypes.POINTER(ctypes.c_double)]
    fun.classifier.restype = ctypes.c_int
    a = np.array(input)
    input_ptr = a.ctypes.data_as(ctypes.POINTER(ctypes.c_double))
    return fun.classifier(input_ptr)
```

Table 2: Results.

DATASET	NEURAL	MLP-GEN	RBF	RBF GEN	GENCLASS
Alcohol	26.85%	25.45%	46.63%	20.29%	14.68%
Appendicitis	22.50%	21.30%	12.23%	11.50%	15.00%
Australian	30.48%	29.83%	34.89%	33.30%	14.57%
Balance	8.19%	8.02%	33.42%	12.96%	0.00%
Dermatology	14.33%	8.75%	62.34%	34.94%	3.72%
Ecoli	53.60%	52.86%	59.59%	56.64%	24.54%
Glass	54.24%	49.30%	50.16%	45.54%	33.81%
Haberman	29.94%	26.98%	25.10%	24.06%	27.33%
Hayes Roth	35.16%	33.28%	64.36%	33.54%	25.39%
Heart	25.95%	24.20%	31.20%	21.00%	17.55%
HouseVotes	7.54%	7.28%	6.13%	5.29%	3.72%
Ionosphere	15.83%	11.30%	16.22%	8.63%	8.29%
Liverdisorder	33.82%	31.37%	30.84%	28.33%	32.06%
Lymography	25.57%	24.55%	25.31%	20.93%	19.29%
Mammographic	27.08%	16.74%	21.38%	17.16%	17.35%
OptDigits	50.37%	45.96%	81.22%	70.93%	24.33%
Page Blocks	7.10%	7.06%	10.09%	9.28%	3.82%
Parkinsons	19.44%	14.26%	17.42%	15.47%	9.47%
Penbased	42.91%	41.79%	82.47%	73.16%	25.32%
Pima	29.07%	28.76%	25.78%	24.12%	25.59%
Popfailures	6.57%	6.38%	7.04%	4.95%	7.04%
Regions2	33.32%	28.56%	38.29%	33.40%	19.52%
Saheart	33.22%	31.40%	32.19%	29.86%	31.30%
Segment	24.54%	21.01%	59.68%	48.70%	10.52%
Shuttle	31.29%	15.70%	32.97%	21.73%	0.15%
Spiral	42.60%	41.57%	44.87%	41.27%	36.90%
Tae	47.53%	47.22%	60.07%	55.07%	37.33%
Thyroid	4.28%	4.16%	10.52%	7.42%	1.89%
Wdbc	20.55%	6.76%	7.27%	6.22%	4.11%
Wine	46.63%	12.43%	31.41%	12.84%	4.71%
Z_F_S	14.17%	10.68%	13.16%	9.65%	7.00%
Z_O_N_F_S	78.73%	59.93%	48.71%	45.20%	32.20%
ZO_NF_S	9.97%	7.33%	9.02%	6.81%	2.60%
ZONF_S	3.48%	3.02%	4.03%	3.98%	1.60%

Figure 4: Typical output of the GenClass command. In every line the tool prints the generation number, the train error as well as the test error. At the end the tool prints the final classification rule and the estimated train and test error.

```

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%
```

320 Required Metadata

321 Current code version

Nr.	Code metadata description	
C1	Current code version	1.0
C2	Permanent link to code/repository used for this code version	https://github.com/itsoulos/GenClass/
C3	Legal Code License	GNU General Public License (GPL)
C4	Code versioning system used	git
C5	Software code languages, tools, and services used	C++
C6	Compilation requirements, operating environments & dependencies	Linux
C7	If available Link to developer documentation/manual	https://github.com/itsoulos/GenClass/wiki
C8	Support email for questions	itsoulos@uoi.gr

Figure 5: An example of user defined grammar.

```

<S> ::= <BEXPR>
<BEXPR> ::= <XLIST><BOOLOP><EXPR>
           | !( <BEXPR> )
           | <XLIST><BOOLOP><EXPR>&<BEXPR>
           | <XLIST><BOOLOP><EXPR>|<BEXPR>
<BOOLOP> ::= >
           | >=
           | <
           | <=
<EXPR> ::= ( <EXPR><BINARYOP><EXPR> )
           | <FUNCTION>( <EXPR> )
           | <TERMINAL>
<BINARYOP> ::= +
              | -
              | *
              | /
<FUNCTION> ::= sin
              | cos
              | exp
              | log
<TERMINAL> ::= <XLIST>
              | <DIGITLIST>.<DIGITLIST>
              | ( -<DIGITLIST>.<DIGITLIST> )
<DIGITLIST> ::= <DIGIT>
               | <DIGIT><DIGIT>
               | <DIGIT><DIGIT><DIGIT>
<DIGIT> ::= 0
            | 1
            | 2
            | 3
            | 4
            | 5
            | 6
            | 7
            | 8
            | 9
<XLIST> ::= x1
           | x2
           | x3
           | x4
           | x5
           | x6
           | x7
           | x8

```

Figure 6: Statistical comparison between the training methods.

