MDPI

Article

Applying bounding techniques on Grammatical Evolution

Ioannis G. Tsoulos[1,*], Alexandros Tzallas[2], Evangelos Karvounis[3]

[1] Department of Informatics and Telecommunications, University of Ioannina, Greece;itsoulos@uoi.gr
[2] Department of Informatics and Telecommunications, University of Ioannina, Greece;tzallas@uoi.gr
[3] Department of Informatics and Telecommunications, University of Ioannina, Greece;ekarvounis@uoi.gr
* Correspondence: itsoulos@uoi.gr;

Abstract: The Grammatical Evolution technique has been successfully applied to a wide range of problems in various scientific fields. However, in Grammatical Evolution, the chromosomes can be initialized at wide value intervals, which can lead to a decrease in the efficiency of underlying technique. In this paper, a technique for discovering appropriate intervals for the initialization of chromosomes is proposed using partition rules guided by a genetic algorithm. This method has been applied to feature construction technique used in a variety of scientific papers. After successfully finding a promising interval, the feature construction technique is applied and the chromosomes are initialized within that interval. This technique was applied to a number of known problems in the relevant literature and the results were extremely promising.

Keywords: Grammatical Evolution; Bounding techniques; Neural networks; Evolutionary techniques; Stochastic methods.

1. Introduction

Genetic algorithms, initially suggested by John Holland[1] are a special case of evolutionary techniques, in which a group of candidate solutions (called also chromosomes) of any optimization problem, are evolved iteratively by applying procedures based on natural processes, such as selection, crossover and mutation [2–4]. Usually, chromosomes are arrays of decimal numbers that represent some possible solution to an optimization problem. A special case of genetic algorithms is Grammatical Evolution [5], where the chromosomes are series of integer numbers. The chromosomes in Grammatical Evolution stand for production rules of a given BNF grammar [6]. This process can be used to construct programs to the underlying language. This method has been used in a variety of real world problems, such as function approximation[7,8], credit classification [9], intrusion detection in computer networks [10], monitoring of water quality [11], solution of trigonometric equations [12], automatic music composition [13], construction of neural networks [14,15], creating numeric constraints [16], video games [17,18], estimation of energy demand [19], combinatorial optimization [20], cryptography [21], evolution of decision trees [22], automatic design of analog electronic circuits [23] etc.

A number of extensions have been proposed in recent years by a number of researchers on Grammatical Evolution, such as the Structured Grammatical Evolution [24,25], which utilizes one-to-one mapping between the values of the chromosomes and the nonterminal symbols of the grammar, incorporation of Particle Swarm Optimization(PSO) [26] to generate programs with grammatical evolution denoted as Grammatical Swarm [27,28], application of parallel programming techniques to Grammatical Evolution [29,30] etc. Furthermore, a series of software has been developed for Grammatical Evolution, such as the GEVA [31] which utilizes the Java programming language to implement various problems for Grammatical Evolution, and it provides also a simple GUI to control the evolutionary process, the gramEvol software [32] that provides a package in the R programming language, the GeLab [33] that implements a Matlab toolbox for Grammatical Evolution, the GenClass [34] software used to create classification programs in a C - like language, the

Citation: Tsoulos, I.G.; Tzallas A; Karvounis E; Applying bounding techniques on Grammatical Evolution. *Journal Not Specified* **2022**, *1*, 0. https://doi.org/

Received:
Accepted:
Published:

QFc software [35] used to produce artificial features from the original ones for classification and regression problems etc.

An important issue of the Grammatical Evolution technique is that usually the integer chromosomes take values in a large interval of values, e.g. $[0 \dots 255]$ as a result of which there is a significant delay in finding the most suitable programs, as many of the values in the above interval are not used. Furthermore, in many cases the performance of Grammatical Evolution is not as expected and a numerous chromosomes and/or generations are required in order to find the most suitable program. The present work proposes a two-step technique to solve the above problem. In the first step of the proposed technique, the value limits for the elements of the chromosomes are estimated using a genetic algorithm and a similar approach as in [36], and in the second stage, the Grammatical Evolution chromosomes are initialized within the best value interval of the first stage. The proposed approach was tested in the feature construction procedure initially suggested in [37]. This approach of creating artificial features with Grammatical Evolution has also been used in many cases, such as Spam Identification [38], Fetal heart classification [39], signal processing of EEG signals [40,41] etc. The usage of the proposed interval technique in the construction of features significantly increased the performance of the method both in classification problems and in regression problems, as will be seen in the experiments section. However, the proposed interval technique is quite general and can be applied to other problems dealt with by Grammatical Evolution.

The rest of this article is divided as follows: in section 2 the proposed method is outlined in detail, in section 3 the datasets used in the experimental results and the experimental results are listed and finally in section 4 some conclusions and guidelines for future research are presented.

2. The proposed method

2.1. Preliminaries

The chromosomes in Grammatical evolution stand for production rules of the given BNF grammar. Any BNF grammar can be described as a set $G = (N, T, S, P)$, where

- N is a set that contains the non-terminal symbols of the grammar.
- T is the set of terminal symbols.
- S represents that start symbol of the grammar.
- P is the production rules of the grammar. Usually these rules are in the form $A \to a$ or $A \to aB,\ A, B \in N,\ a \in T$.

The original BNF grammar is extended by enumerating the production rules. As an example, consider the extended BNF grammar shown in 1. The non - terminal symbols are enclosed in the <> symbols. The sequence numbers of the production rules are inside the parentheses of the extended grammar. The constant N denotes the dimension of the input data. **T**he Grammatical Evolution production procedure initiates from the start symbol of the grammar and through a series of steps creates a program, by replacing non - terminal symbols with the right hand of the selected production rule. Every rule is selected using the following two steps:

- Read the next element V from the chromosome.
- The next rule is selected through the equation Rule = V mod NR, where NR is the number of production rules for the current non – terminal symbol.

As an example of production consider the chromosome:

$$x = [9, 8, 6, 4, 16, 10, 17, 23, 8, 14]$$

with $N = 3$. The expression $f(x) = x_2 + \cos(x_3)$ is produced through the steps shown in Table 1.

Figure 1. An example of a BNF grammar.

```
S::=<expr>   (0)
<expr> ::=  (<expr> <op> <expr>)  (0)
          | <func> ( <expr> )    (1)
          |<terminal>            (2)
<op> ::=     +      (0)
          | -      (1)
          | *      (2)
          | /      (3)
<func> ::=   sin  (0)
          | cos  (1)
          |exp   (2)
          |log   (3)
<terminal>::=<xlist>                (0)
          |<digitlist>.<digitlist> (1)
<xlist>::=x1    (0)
          | x2 (1)
          .........
          | xN (N)
<digitlist>::=<digit>                   (0)
          | <digit><digit>              (1)
          | <digit><digit><digit>       (2)
<digit>  ::= 0 (0)
          | 1 (1)
          | 2 (2)
          | 3 (3)
          | 4 (4)
          | 5 (5)
          | 6 (6)
          | 7 (7)
          | 8 (8)
          | 9 (9)
```

Table 1. The steps performed to produce a program from the used grammar.

String	Chromosome	Operation
<expr>	9,8,6,4,16,10,17,23,8,14	9 mod 3 = 0
(<expr><op><expr>)	8,6,4,16,10,17,23,8,14	8 mod 3 = 2
(<terminal><op><expr>)	6,4,16,10,17,23,8,14	6 mod 2 = 0
(<xlist><op><expr>)	4,16,10,17,23,8,14	4 mod 3 = 1
(x2<op><expr>)	16,10,17,23,8,14	16 mod 4 = 0
(x2+<expr>)	10,17,23,8,14	10 mod 3 = 1
(x2+<func>(<expr>))	17,23,8,14	17 mod 4 = 1
(x2+cos(<expr>))	23,8,14	23 mod 2 = 1
(x2+cos(<terminal>))	8,14	8 mod 2 = 0
(x2+cos(<xlist>))	14	14 mod 3 = 2
(x2+cos(x3))		

2.2. The feature construction method

The feature construction technique constructs artificial features from the original ones using the Grammatical Evolution procedure and the main steps are graphically illustrated in 2.

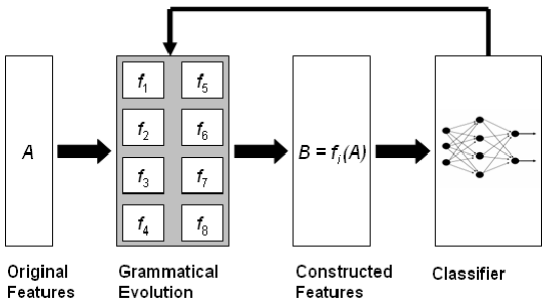

Figure 2. Schematic representation of the original feature construction technique that utilizes Grammatical Evolution.

This process creates artificial features which are evaluated using a classifier such as an artificial neural network [42,43] or an Radial Basis Function (RBF) network [44,45]. The main steps of the process are:

1. **Initialization** step.
   (a) **Read** the train data of the objective problem. The train data have M pairs (x_i, t_i), $i = 1..M$ where t_i is the expected output for pattern x_i.
   (b) **Set** the parameters of the method: N_G as the number of allowed generations, N_C as the total number of chromosomes, p_s as the selection rate and p_m as the mutation rate.
   (c) **Define** as N_f the number of features that will be constructed.
   (d) **Initialize** the chromosomes of the population as random integers.
   (e) **Set** iter=1
2. **Genetic step**
   (a) **For** $i = 1, \ldots, N_g$ **do**
      i. **Create** using the Grammatical Evolution procedure of mapping, a set of N_f artificial features for the corresponding chromosome g_i.
      ii. **Perform** a transformation of the original features using the previously produced features. Denote the new train set as $(x_{g_i,j}, t_j)$, $j = 1, ..M$
      iii. **Apply** a machine learning model C (such as an artificial neural network) on the new data and train the model C. The fitness f_i is calculated as:

$$f_i = \sum_{j=1}^{M} \left(C\left(x_{g_i,j}\right) - t_j \right)^2 \tag{1}$$

      iv. **Perform** the selection procedure, where initially the chromosomes are sorted according to their fitness. The best $(1 - p_s) \times N_C$ chromosomes are copied to the next generation of the population. The remaining chromosomes will be replaced by offsprings created during the crossover procedure.
      v. **Perform** the crossover procedure. This process will create $p_s \times N_c$ offsprings. Initially two distinct chromosomes are selected for every pair of constructed offsprings. The selection is performed using the tournament selection. For each pair of (z, w) parents, two offsprings $\tilde{z}$ and $\tilde{w}$ are produced utilizing the so - called one point crossover, which is graphically shown in Figure 3.

vi. **Perform** the mutation procedure. For each element of every chromosome, a random number $r \in [0,1]$ is selected. The corresponding element is altered if $r \leq p_m$.

(b) **EndFor**

3. **Set** iter=iter+1
4. **If** iter $\leq N_G$ goto **Genetic** Step, else the process is terminated and the best chromosome g^* is obtained.
5. **Apply** the N_f features that correspond to g^* to the test set.
6. **Apply** a machine learning model to the new test set and report the corresponding test error..

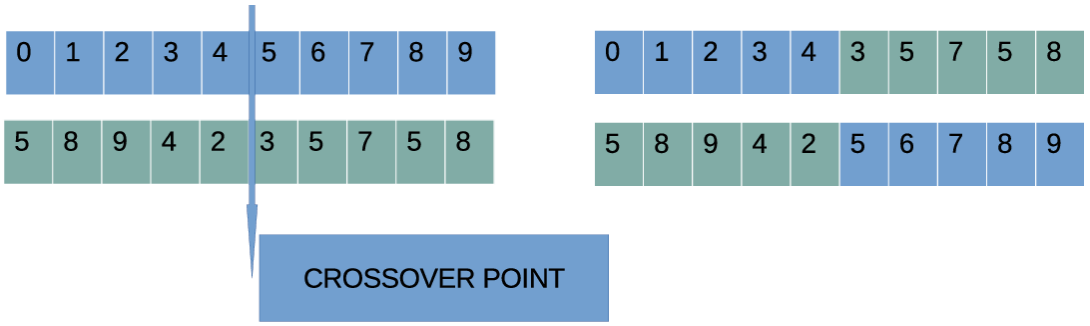

Figure 3. One point crossover method, used in the Grammatical Evolution procedure.

2.3. The proposed method

In the present technique, a procedure is conducted before the initialization of the chromosomes of the algorithm described in subsection 2.2. This process using a modified genetic algorithm aims to find the optimal value interval for the chromosome elements. In the proposed method the chromosomes are sets of intervals as well as the fitness values. Hence, a comparison operator should be used in order to compare intervals. In order to compare the intervals $a = [a_1, a_2]$, $b = [b_1, b_2]$, the operator is $L^*(a, b)$ used defined as:

$$L^*(a,b) = \begin{cases} \text{TRUE}, & a_1 < b_1, \text{OR } (a_1 = b_1 \text{ AND } a_2 < b_2) \\ \text{FALSE}, & \text{OTHERWISE} \end{cases} \quad (2)$$

The main steps of the proposed method are listed in Algorithm 1.

Algorithm 1 The main steps of the procedure aim to discover the optimal value interval.

1. **Initialization Step**
   (a) **Set** as I_C as the total number of chromosomes**,** as I_G the number of generations, p_s as the selection rate**,** p_m as the mutation rate. and as I_M the initial margin.
   (b) **Construct** the initial set of margins

$$N_M = [0, I_M]^{(n)} \tag{3}$$

   where n is the number of elements for the chromosomes of the Grammatical Evolution procedure.
   (c) **Initialize** a genetic population of I_C chromosomes. Every chromosome contains a set for intervals randomly initialized in N_M. An Example of chromosome with $n = 5$ can be the following: $[(0, 12), (0, 30), (0, 21), (0, 100), (0, 192)]$.
   (d) **Set** iter=0
2. **Evaluation Step**
   (a) **For** $i = 1 \ldots I_C$ **do**
      i. **Create** N_S random sets of intervals inside the range defined by chromosome g_i**.**
      ii. **Execute** the procedure of subsection 2.2, by initializing the chromosomes of the Grammatical Evolution procedure inside set every random set.
      iii. **Denote** by $E_{\min}(g_i)$ the minimum train error obtained by the Feature Construction procedure.
      iv. **Denote** by $E_{\min}(g_i)$ the maximum train error obtained by the Feature Construction procedure.
      v. **Set** $f_i = [E_{min}(g_i), E_{max}(g_i)]$, as the fitness value of chromosome g_i.
3. **EndFor**
4. **Genetic step**
   (a) **Perform** the crossover procedure of algorithm 2.
   (b) **Perform** the mutation procedure of algorithm 3.
5. **Termination Check Step**
   (a) **Set** $iter = iter + 1$
   (b) **If** $iter > I_G$ Then **Terminate** and **report** the chromosome g_{best} with the best fitness**, else Goto** Evaluation Step.
   (c) **Endif**

Algorithm 2 The steps of the crossover procedure used in the proposed method.

1. The chromosomes are sorted with respect to their fitness values.
2. The first $(1 - p_s) \times I_c$ chromosomes are copied intact to the next generation and the rest of the population are replaced by offprings created.
3. For every pair of offsprings, two chromosomes $z = (z_1, z_2, ..., z_n)$, $y = (y_1, y_2, ..., y_n)$ are selected by utilizing the procedure of tournament selection. The offsprings $\tilde{z}$ and $\tilde{y}$ are produced using the following operations:

$$\begin{aligned} \tilde{z}_i &= a_i z_i + (1 - a_i) y_i \\ \tilde{y}_i &= a_i y_i + (1 - a_i) z_i \end{aligned} \tag{4}$$

   where a_i is a random number in $[-0.5, 1.5]$[46].

Algorithm 3 Mutation procedure for the proposed algorithm.

1. **For** each chromosome g **Do**
   (a) **For** every element $g_i, i = 1..n$ **Do**
      i. **Draw** a random number $r \in [0, 1]$
      ii. If $r < p_m$ then $g_i = \text{left}(g_i) + z(\text{right}(g_i) - \text{left}(g_i))$, where z is a random number in $[0, 1]$. The function left(x) represents the lower value of the interval x and the function right(x) represents the the upper value of the interval x.
   (b) **EndFor**
2. **EndFor**

3. Experiments

The present methodology was applied to a number of problems in various scientific domains and the experimental results were compared with other machine learning techniques. The datasets used to run the experiments are freely available at the following websites:

1. The UCI dataset repository, https://archive.ics.uci.edu/ml/index.php(accessed on 18 February 2024)[47]
2. The Keel repository, https://sci2s.ugr.es/keel/datasets.php(accessed on 18 February 2024)[48].
3. The Statlib URL http://lib.stat.cmu.edu/datasets/(accessed on 18 February 2024).

3.1. Classification datasets

The classification datasets used in the conducted experiments are the following:

1. **Appendictis** a medical dataset, used in [49].
2. **Australian** dataset [50], about credit card transactions.
3. **Balance** dataset [51], which is a dataset used in psychological experiments.
4. **Circular** dataset, an artificial datasets that contains 1000 examples which belong to two categories (500 examples each).
5. **Cleveland** dataset, which is a medical dataset [52,53].
6. **Dermatology** dataset [54], which is a medical dataset about dermatological deceases.
7. **Haberman** dataset, a medical dataset about breast cancer.
8. **Heart** dataset [55], a medical dataset about heart diseases.
9. **Hayes roth** dataset [56].
10. **HouseVotes** dataset [57], related to votes in the U.S. House of Representatives Congressmen.
11. **Ionosphere** dataset, It was used in experiments related to the ionosphere [58,59].
12. **Liverdisorder** dataset [60], a medical dataset related to liver disorders.
13. **Mammographic** dataset [61], a medical dataset related to breast cancer.
14. **Parkinsons** dataset, a medical dataset related to Parkinson's disease (PD)[62].
15. **Pima** dataset [63], a medical dataset related to the diabetes presence.
16. **Popfailures** dataset [64], a dataset related to climate measurements.
17. **Regions2** dataset, medical dataset related to hepatitis C [65].
18. **Saheart** dataset [66], a medical dataset related to heart diseases.
19. **Segment** dataset [67], an image processing dataset.
20. **Spiral** dataset, which is an artificial dataset with two classes. The features in the first class are constructed as: $x_1 = 0.5t\cos(0.08t)$, $x_2 = 0.5t\cos\left(0.08t + \frac{\pi}{2}\right)$ and for the second class the used equations are : $x_1 = 0.5t\cos(0.08t + \pi)$, $x_2 = 0.5t\cos\left(0.08t + \frac{3\pi}{2}\right)$
21. **Student** dataset [68]. This data approach student achievement in secondary education of two Portuguese schools.
22. **Transfusion** dataset[69]: Data taken from the Blood Transfusion Service Center in Hsin-Chu City in Taiwan.

23. **Wdbc** dataset [70], a medical dataset related to breast tumors.
24. **Wine** dataset, used to detect the origin of wines [71,72].
25. **Eeg** datasets, a medical dataset related to EEG measurements [73]. There are 4 different cases from this dataset: Z_F_S, Z_O_N_F_S, ZO_NF_S and ZONF_S.
26. **Zoo** dataset [74], related to animal classification.

3.2. Regression datasets

The regression datasets used in the conducted experiments are the following:

1. **Abalone** dataset [75], a dataset related to the prediction of age of abalones.
2. **Airfoil** dataset, a dataset used in NASA [76].
3. **Baseball** dataset, It was used to predict financial earnings of baseball players.
4. **Concrete** dataset [77], which is a civil engineering dataset.
5. **Dee** dataset, used to predict the price of electricity.
6. **ELE** dataset, Electrical Length data set downloaded from the KEEL repository.
7. **HO** dataset, downloaded from the STALIB repository.
8. **Housing** dataset, provided in [78].
9. **Laser** dataset, which is related to laser experiments.
10. **LW** dataset. This dataset is produced from a study to identify risk factors associated with low weight babies.
11. **MORTGAGE** dataset, a dataset related to economic measurements from USA.
12. **PL** dataset, downloaded from the STALIB repository.
13. **SN** dataset, downloaded from the STALIB repository.
14. **Treasury** dataset, a dataset related to economic measurements from USA.
15. **TZ** dataset, downloaded from the STALIB repository.

3.3. Experimental results

All the methods used in the experiments were coded in ANSI C++ using the freely available optimization package of OPTIMUS, that can be downloaded from https://github.com/itsoulos/OPTIMUS/(accessed on 14 February 2024). All the experiments were conducted 30 times with different initialization for the random generator each time. Also, the drand48() function of the C++ was used to produce random numbers. To validate the experimental results the 10 - fold validation method was utilized. For the case of classification datasets, the average classification error as measured on the test set is reported. The average regression error as measured on the test set is reported for the case of regression datasets. The values for the experimental parameters are shown in Table 2. The results for the classification datasets are listed in Table 3 and the results for the regression datasets ae shown in Table 4. The following applies to the tables with the experimental results:

1. The column DATASET stands for the name used dataset.
2. The column represents the results obtained by the training of an artificial neural network with $H = 10$ processing nodes. The neural network was trained using the BFGS optimization method [79].
3. The column RBF stands for the results obtained by an RBF network with $H = 10$ processing nodes.
4. The column FC represents the results obtained by the Feature Construction procedure of subsection 2.2 without the proposed modification.
5. The column IFC10 stands for the incorporation of the proposed technique with the Feature Construction technique. The parameter I_G was set to 10.
6. The column IFC20 stands for the incorporation of the proposed technique with the Feature Construction technique. The parameter I_G was set to 20.
7. The line AVERAGE denotes the average classification or regression error.

NAME	PURPOSE	VALUE
N_C	Number of chromosomes	500
N_G	Number of generations	200
p_s	Selection rate	0.10
p_m	Mutation rate	0.05
N_f	Number of features	2
H	Number of hidden nodes	10
I_C	Number of chromosomes (proposed method)	200
I_G	Number of generations (proposed method)	10
I_M	Initial right bound (proposed method)	1024

Table 2. The values for the parameters used in the conducted experiments.

Table 3. Experimental results for the classification datasets. The numbers in cells represent average classification error as measured in the test set.

DATASET	**MLP**	**RBF**	**FC**	**IFC10**	**IFC20**
APPENDICITIS	18.10%	12.23%	15.86%	16.18%	15.94%
AUSTRALIAN	32.21%	34.89%	16.48%	13.80%	14.05%
BALANCE	8.97%	33.42%	11.16%	0.38%	0.00%
CIRCULAR	5.99%	6.30%	3.69%	3.27%	3.04%
CLEVELAND	51.60%	67.10%	47.74%	44.79%	44.86%
DERMATOLOGY	30.58%	62.34%	27.47%	11.23%	13.43%
HABERMAN	28.66%	25.10%	26.62%	23.03%	22.94%
HAYES ROTH	56.18%	64.36%	32.70%	18.50%	16.96%
HEART	28.34%	31.20%	18.13%	15.08%	15.53%
HOUSEVOTES	6.62%	6.13%	11.20%	6.77%	8.09%
IONOSPHERE	15.14%	16.22%	10.09%	9.83%	11.38%
LIVERDISORDER	31.11%	30.84%	31.01%	28.93%	30.03%
MAMMOGRAPHIC	19.88%	21.38%	16.77%	14.91%	14.72%
PARKINSONS	18.05%	17.41%	10.60%	11.09%	9.39%
PIMA	32.19%	25.78%	24.43%	23.26%	24.12%
POPFAILURES	5.94%	7.04%	5.33%	4.78%	4.64%
REGIONS2	29.39%	38.29%	29.69%	26.38%	26.58%
SAHEART	34.86%	32.19%	29.45%	29.40%	29.93%
SEGMENT	57.72%	59.68%	47.81%	31.19%	30.27%
SPIRAL	40.21%	44.87%	31.69%	26.06%	22.41%
STUDENT	5.61%	7.52%	5.29%	3.57%	3.61%
TRANSFUSION	25.84%	27.36%	22.54%	19.99%	20.76%
WDBC	8.56%	7.27%	3.66%	3.42%	2.52%
WINE	19.20%	31.41%	7.49%	7.97%	8.92%
Z_F_S	10.73%	13.16%	5.31%	6.01%	5.37%
Z_O_N_F_S	64.81%	60.40%	37.97%	32.78%	32.23%
ZO_NF_S	8.41%	9.02%	4.74%	4.04%	4.22%
ZONF_S	2.60%	4.03%	2.66%	2.49%	2.24%
ZOO	16.67%	21.93%	25.33%	11.32%	10.40%
AVERAGE	**24.10%**	**27.86%**	**19.31%**	**15.53%**	**15.47%**

Table 4. Experimental results for the regression datasets. The numbers in cells stand for the average regression error as measured on the test set.

DATASET	MLP	RBF	FC	IFC10	IFC20
ABALONE	7.17	7.37	4.66	3.73	3.70
AIRFOIL	0.003	0.27	0.002	0.001	0.001
BASEBALL	103.60	93.02	71.45	55.64	56.91
CONCRETE	0.0099	0.011	0.006	0.005	0.005
DEE	1.013	0.17	0.17	0.18	0.19
ELE	75.06	49.95	43.54	36.60	26.71
HO	2.78	0.03	0.009	0.013	0.017
HOUSING	43.26	57.68	27.58	13.14	14.60
LASER	0.59	0.024	0.009	0.024	0.031
LW	1.90	1.14	0.73	0.013	0.015
MORTGAGE	2.41	1.45	0.58	0.015	0.014
PL	0.28	0.083	0.028	0.019	0.018
SN	2.95	0.90	0.79	0.038	0.022
TREASURY	2.93	2.02	0.63	0.06	0.05
TZ	5.38	4.10	3.41	0.40	0.65
AVERAGE	**16.20**	**14.27**	**9.87**	**7.13**	**6.60**

From the experimental results and their careful study, one can deduce that the original feature construction method significantly outperforms simple machine learning techniques in terms of classification error or function approximation error. The feature construction method achieved low errors by creating only 2 artificial features from the existing ones and thus significantly reduced the required information needed for successful classification or data fitting. In addition, the proposed value interval generation technique (IFC10 or IFC20) significantly improves the feature construction technique, both on classification data and data fitting data. Furthermore, a statistical comparison was performed using scatter plot and the results are outlined in Figure 4 for the classification datasets and in Figure 5 for the regression datasets.

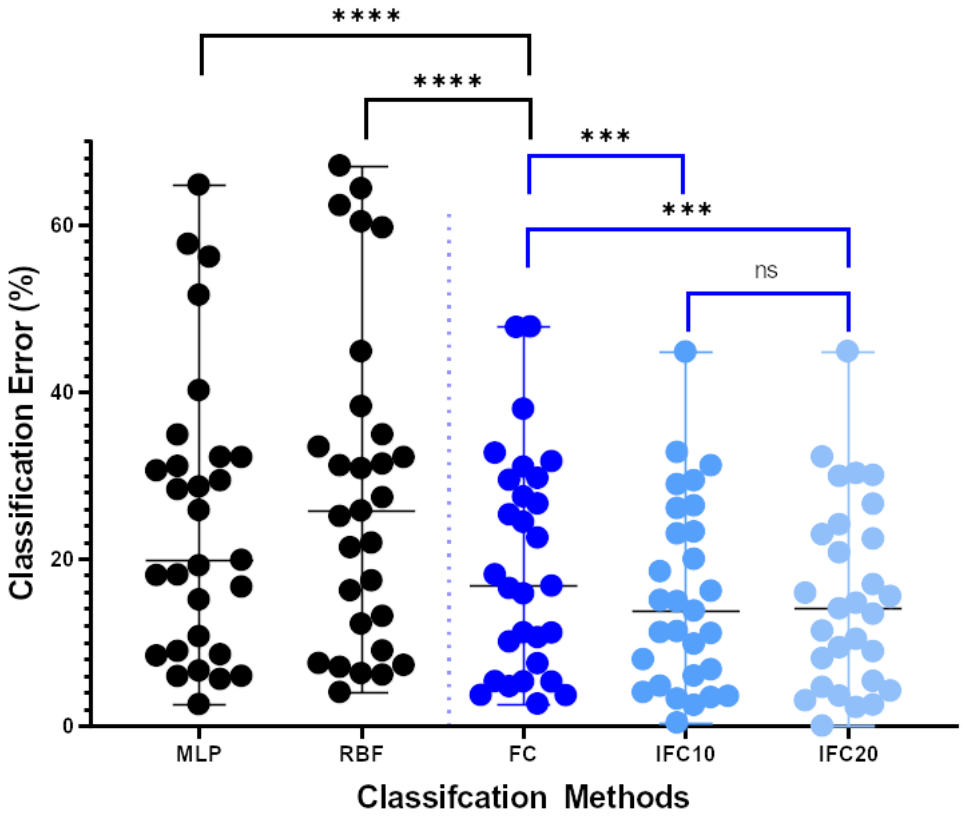

Figure 4. Comparison of Classification Error Percentages for Different Machine Learning Methods. The scatter plot displays individual classification error measurements for each method (MLP, RBF, FC, IFC10, IFC20) with lines indicating median values. Statistical significance is denoted by asterisks, with ” indicating $p < 0.0001$, ’*’ indicating $p < 0.001$, ” $p < 0.01$, and ’ns’ indicating no significant difference. The results show that FC significantly outperforms both MLP and RBF, IFC10 improves upon FC, and there is no significant difference between the IFC10 and IFC20 methods.

The scatter plot (Figure 4) effectively demonstrates the comparative performance of various classification methods in terms of classification error percentage. The methods evaluated are Multi-Layer Perceptron (MLP), Radial Basis Function (RBF), the simple feature construction technique (FC), and two variants of the proposed modification named IFC10 and IFC20 respectively. The FC method shows a lower median classification error compared to both MLP and RBF, with the differences being statistically significant (as indicated by the asterisks). This suggests that the FC method outperforms MLP and RBF in terms of classification accuracy. When comparing FC to IFC10, IFC10 displays a lower median classification error, and the difference is statistically significant, suggesting that IFC10 is an improvement over the standard FC method. Between IFC10 and IFC20, the median classification errors are close, and the lack of asterisks (’ns’ for not significant) between these two methods indicates that the difference in classification error is not statistically significant. This suggests that the critical parameter change from IFC10 to IFC20 does not have a significant impact on classification accuracy.

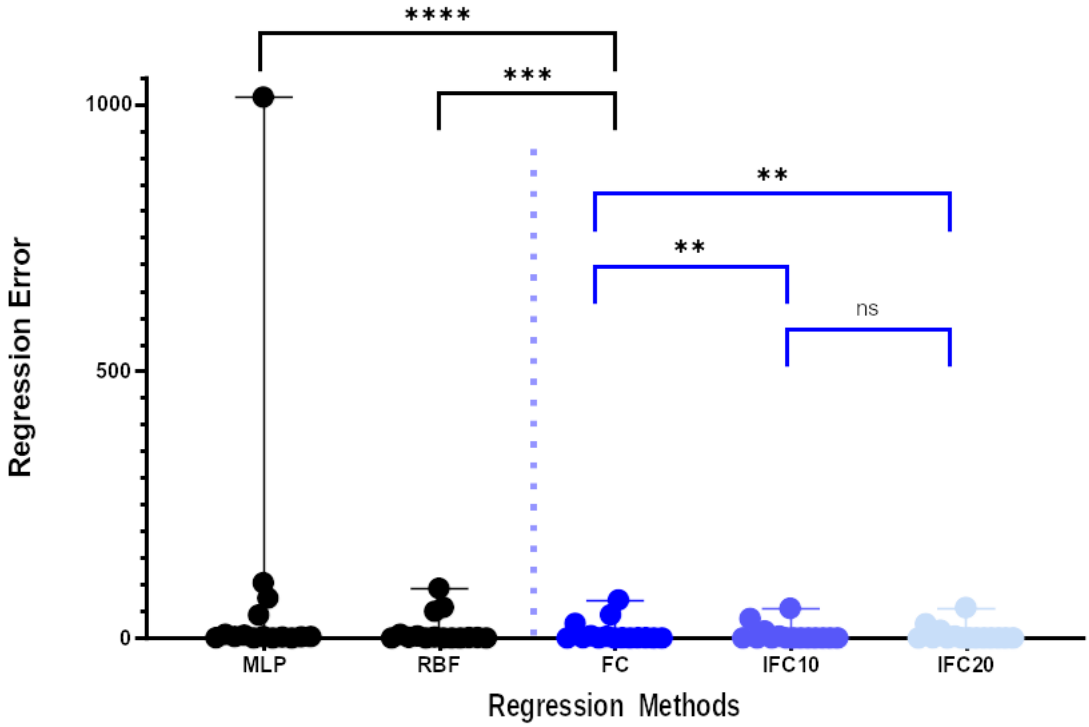

Figure 5. Comparative scatter plot of regression errors for machine learning methods: Multi-Layer Perceptron (MLP), Radial Basis Function (RBF), FC and Interval Feature Construction with parameters 10 (IFC10) and 20 (IFC20). The data points represent individual measurements of regression error, with lines indicating the median values. Statistical significance is denoted by asterisks, with FC showing significantly lower errors than MLP and RBF, IFC10 improving upon FC, and no significant difference between IFC10 and IFC20.

The Figure 5 depicts a scatter plot that compares regression errors across the used machine learning regression methods in the experiments. From the plot, we can see that:

- The FC method has a lower median regression error than both the MLP and RBF methods. This is statistically significant, as indicated by the asterisks (** for $p < 0.01$ between FC and RBF, and *** for $p < 0.001$ between FC and MLP), suggesting that FC is indeed a better method compared to MLP and RBF.
- The IFC10 method shows a lower median regression error compared to the FC method, with this difference being statistically significant (** for $p < 0.01$). This suggests that IFC10 is a better method compared to FC.
- Between IFC10 and IFC20, the median regression error is similar, and the difference is not statistically significant (ns for not significant). This indicates that there is no significant difference in regression error between IFC10 and IFC20 when the parameter is changed from 10 to 20.

4. Conclusions

In the current work, an efficient technique was proposed to identify the bounding box of the values of chromosomes in Grammatical Evolution. The proposed method precedes Grammatical Evolution and aims to discover a promising value interval for the chromosome values and, in this way, will significantly reduce the computational time required afterward but at the same time make the research more efficient. The method was tested on the Feature Construction technique, which is guided by Grammatical Evolution. Experimental results and statistical comparison show a significant reduction in classification error and data fitting error using the proposed technique. In addition, from the statistical comparison of the results, it appears that there is no direct effect of the number of generations of

the genetic algorithm on the effectiveness of the method, which means that only a few generations are enough to effectively find the promising value interval. Although the proposed technique was applied to the feature generation method, it is quite general and in the future it will be able to be applied to other application areas of Grammatical Evolution such as the construction of artificial neural networks, the rule generation method, etc. However, a major drawback of the present technique is that it requires a significant amount of computing resources for its execution, but this can be alleviated by using parallel computing techniques, such as the MPI programming library [80] or the OpenMP library [81].

Author Contributions: I.G.T., A.T. and E.K. conceived the idea and methodology and supervised the technical part regarding the software. I.G.T. conducted the experiments, employing several datasets, and provided the comparative experiments. A.T. performed the statistical analysis. E.K. and all other authors prepared the manuscript. E.K. and I.G.T. organized the research team and A.T. supervised the project. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Institutional Review Board Statement: Not applicable.

Acknowledgments: This research has been financed by the European Union : Next Generation EU through the Program Greece 2.0 National Recovery and Resilience Plan , under the call RESEARCH – CREATE – INNOVATE, project name “iCREW: Intelligent small craft simulator for advanced crew training using Virtual Reality techniques" (project code:TAEDK-06195

References

1. J.H. Holland, Genetic algorithms. Scientific american **267**, pp. 66-73, 1992.
2. J. Stender, Parallel Genetic Algorithms:Theory & Applications. Edition: IOS Press, 1993.
3. D. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Publishing Company, Reading, Massachussets, 1989.
4. Z. Michaelewicz, Genetic Algorithms + Data Structures = Evolution Programs. Springer - Verlag, Berlin, 1996.
5. M. O’Neill, C. Ryan, Grammatical evolution, IEEE Trans. Evol. Comput. **5,**pp. 349–358, 2001.
6. J. W. Backus. The Syntax and Semantics of the Proposed International Algebraic Language of the Zurich ACM-GAMM Conference. Proceedings of the International Conference on Information Processing, UNESCO, 1959, pp.125-132.
7. C. Ryan, J. Collins, M. O’Neill, Grammatical evolution: Evolving programs for an arbitrary language. In: Banzhaf, W., Poli, R., Schoenauer, M., Fogarty, T.C. (eds) Genetic Programming. EuroGP 1998. Lecture Notes in Computer Science, vol 1391. Springer, Berlin, Heidelberg, 1998.
8. M. O’Neill, M., C. Ryan, Evolving Multi-line Compilable C Programs. In: Poli, R., Nordin, P., Langdon, W.B., Fogarty, T.C. (eds) Genetic Programming. EuroGP 1999. Lecture Notes in Computer Science, vol 1598. Springer, Berlin, Heidelberg, 1999.
9. A. Brabazon, M. O’Neill, Credit classification using grammatical evolution, Informatica **30.3**, 2006.
10. S. Şen, J.A. Clark. A grammatical evolution approach to intrusion detection on mobile ad hoc networks, In: Proceedings of the second ACM conference on Wireless network security, 2009.
11. L. Chen, C.H. Tan, S.J. Kao, T.S. Wang, Improvement of remote monitoring on water quality in a subtropical reservoir by incorporating grammatical evolution with parallel genetic algorithms into satellite imagery, Water Research **42**, pp. 296-306, 2008.
12. C. Ryan, M. O’Neill, J.J. Collins, Grammatical evolution: Solving trigonometric identities, proceedings of Mendel. Vol. 98. 1998.
13. A.O. Puente, R. S. Alfonso, M. A. Moreno, Automatic composition of music by means of grammatical evolution, In: APL ’02: Proceedings of the 2002 conference on APL: array processing languages: lore, problems, and applications July 2002 Pages 148–155.
14. Lídio Mauro Limade Campo, R. Célio Limã Oliveira,Mauro Roisenberg, Optimization of neural networks through grammatical evolution and a genetic algorithm, Expert Systems with Applications **56**, pp. 368-384, 2016.
15. K. Soltanian, A. Ebnenasir, M. Afsharchi, Modular Grammatical Evolution for the Generation of Artificial Neural Networks, Evolutionary Computation **30**, pp 291–327, 2022.
16. I. Dempsey, M.O’ Neill, A. Brabazon, Constant creation in grammatical evolution, International Journal of Innovative Computing and Applications **1** , pp 23–38, 2007.
17. E. Galván-López, J.M. Swafford, M. O’Neill, A. Brabazon, Evolving a Ms. PacMan Controller Using Grammatical Evolution. In: , et al. Applications of Evolutionary Computation. EvoApplications 2010. Lecture Notes in Computer Science, vol 6024. Springer, Berlin, Heidelberg, 2010.

18. N. Shaker, M. Nicolau, G. N. Yannakakis, J. Togelius, M. O'Neill, Evolving levels for Super Mario Bros using grammatical evolution, 2012 IEEE Conference on Computational Intelligence and Games (CIG), 2012, pp. 304-31.
19. D. Martínez-Rodríguez, J. M. Colmenar, J. I. Hidalgo, R.J. Villanueva Micó, S. Salcedo-Sanz, Particle swarm grammatical evolution for energy demand estimation, Energy Science and Engineering **8**, pp. 1068-1079, 2020.
20. N. R. Sabar, M. Ayob, G. Kendall, R. Qu, Grammatical Evolution Hyper-Heuristic for Combinatorial Optimization Problems, IEEE Transactions on Evolutionary Computation **17**, pp. 840-861, 2013.
21. C. Ryan, M. Kshirsagar, G. Vaidya, G. et al. Design of a cryptographically secure pseudo random number generator with grammatical evolution. Sci Rep **12**, 8602, 2022.
22. P.J. Pereira, P. Cortez, R. Mendes, Multi-objective Grammatical Evolution of Decision Trees for Mobile Marketing user conversion prediction, Expert Systems with Applications **168**, 114287, 2021.
23. F. Castejón, E.J. Carmona, Automatic design of analog electronic circuits using grammatical evolution, Applied Soft Computing **62**, pp. 1003-1018, 2018.
24. N. Lourenço, F.B. Pereira, E, Costa, Unveiling the properties of structured grammatical evolution, Genetic Programming and Evolvable Machines **17** , pp. 251-289, 2016.
25. N. Lourenço, F. Assunção, F.B. Pereira, E. Costa, P. Machado, Structured grammatical evolution: a dynamic approach, Handbook of grammatical evolution, pp. 137-161, 2018.
26. Riccardo Poli, James Kennedy kennedy, Tim Blackwell, Particle swarm optimization An Overview, Swarm Intelligence **1**, pp 33-57, 2007.
27. M. O'Neill, A. Brabazon, Grammatical swarm: The generation of programs by social programming. Natural Computing **5**, pp. 443-462, 2006.
28. E. Ferrante, E. Duéñez-Guzmán, A.E. Turgut, T. Wenseleers, GESwarm: Grammatical evolution for the automatic synthesis of collective behaviors in swarm robotics. In Proceedings of the 15th annual conference on Genetic and evolutionary computation, pp. 17-24, 2013.
29. O. Popelka, P. Osmera, Parallel Grammatical Evolution for Circuit Optimization. In: Hornby, G.S., Sekanina, L., Haddow, P.C. (eds) Evolvable Systems: From Biology to Hardware. ICES 2008. Lecture Notes in Computer Science, vol 5216. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-85857-7_40
30. P. Ošmera, Two level parallel grammatical evolution, Advances in Computational Algorithms and Data Analysis pp. 509-525, 2009.
31. M. O'Neill, E. Hemberg, C. Gilligan, E. Bartley, J. McDermott, A. Brabazon, GEVA: grammatical evolution in Java. ACM SIGEVOlution **3**, pp. 17-22, 2008.
32. F. Noorian, A.M. de Silva, P.H.W. Leong, gramEvol: Grammatical Evolution in R, Journal of Statistical Software **71**, pp. 1–26, 2016.
33. M.A. Raja, C. Ryan, GELAB - A Matlab Toolbox for Grammatical Evolution. In: Yin, H., Camacho, D., Novais, P., Tallón-Ballesteros, A. (eds) Intelligent Data Engineering and Automated Learning – IDEAL 2018. IDEAL 2018. Lecture Notes in Computer Science(), vol 11315, 2018. Springer, Cham. https://doi.org/10.1007/978-3-030-03496-2_22
34. N.Anastasopoulos, I.G. Tsoulos, A. Tzallas, GenClass: A parallel tool for data classification based on Grammatical Evolution, SoftwareX **16**, 100830, 2021.
35. I.G. Tsoulos, QFC: A Parallel Software Tool for Feature Construction, Based on Grammatical Evolution, Algorithms **15**, 295, 2022.
36. N. Anastasopoulos, I.G. Tsoulos, E. Karvounis, A. Tzallas, Locate the Bounding Box of Neural Networks with Intervals, Neural Process Lett **52**, pp. 2241–2251, 2020.
37. Dimitris Gavrilis, Ioannis G. Tsoulos, Evangelos Dermatas, Selecting and constructing features using grammatical evolution, Pattern Recognition Letters **29**,pp. 1358-1365, 2008.
38. Dimitris Gavrilis, Ioannis G. Tsoulos, Evangelos Dermatas, Neural Recognition and Genetic Features Selection for Robust Detection of E-Mail Spam, Advances in Artificial Intelligence Volume 3955 of the series Lecture Notes in Computer Science pp 498-501, 2006.
39. George Georgoulas, Dimitris Gavrilis, Ioannis G. Tsoulos, Chrysostomos Stylios, João Bernardes, Peter P. Groumpos, Novel approach for fetal heart rate classification introducing grammatical evolution, Biomedical Signal Processing and Control **2**,pp. 69-79, 2007
40. Otis Smart, Ioannis G. Tsoulos, Dimitris Gavrilis, George Georgoulas, Grammatical evolution for features of epileptic oscillations in clinical intracranial electroencephalograms, Expert Systems with Applications **38**, pp. 9991-9999, 2011
41. A. T. Tzallas, I. Tsoulos, M. G. Tsipouras, N. Giannakeas, I. Androulidakis and E. Zaitseva, Classification of EEG signals using feature creation produced by grammatical evolution, In: 24th Telecommunications Forum (TELFOR), pp. 1-4, 2016.
42. C. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995.
43. G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of Control Signals and Systems **2**, pp. 303-314, 1989.
44. J. Park and I. W. Sandberg, Universal Approximation Using Radial-Basis-Function Networks, Neural Computation 3, pp. 246-257, 1991.
45. H. Yu, T. Xie, S. Paszczynski, B. M. Wilamowski, Advantages of Radial Basis Function Networks for Dynamic System Design, in IEEE Transactions on Industrial Electronics **58**, pp. 5438-5450, 2011.
46. Z. Michaelewicz, Genetic Algorithms + Data Structures = Evolution Programs. Springer - Verlag, Berlin, 1996.

47. M. Kelly, R. Longjohn, K. Nottingham, The UCI Machine Learning Repository. 2023. Available online: https://archive.ics.uci.edu (accessed on 18 February 2024).
48. 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.
49. 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.
50. J.R. Quinlan, Simplifying Decision Trees. International Journal of Man-Machine Studies **27**, pp. 221-234, 1987.
51. T. Shultz, D. Mareschal, W. Schmidt, Modeling Cognitive Development on Balance Scale Phenomena, Machine Learning **16**, pp. 59-88, 1994.
52. Z.H. Zhou,Y. Jiang, NeC4.5: neural ensemble based C4.5," in IEEE Transactions on Knowledge and Data Engineering **16**, pp. 770-773, 2004.
53. R. Setiono , W.K. Leow, FERNN: An Algorithm for Fast Extraction of Rules from Neural Networks, Applied Intelligence **12**, pp. 15-25, 2000.
54. G. Demiroz, H.A. Govenir, N. Ilter, Learning Differential Diagnosis of Eryhemato-Squamous Diseases using Voting Feature Intervals, Artificial Intelligence in Medicine. **13**, pp. 147–165, 1998.
55. I. Kononenko, E. Šimec, M. Robnik-Šikonja, Overcoming the Myopia of Inductive Learning Algorithms with RELIEFF, Applied Intelligence **7**, pp. 39–55, 1997
56. 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.
57. R.M. French, N. Chater, Using noise to compute error surfaces in connectionist networks: a novel means of reducing catastrophic forgetting, Neural Comput. **14**, pp. 1755-1769, 2002.
58. J.G. Dy , C.E. Brodley, Feature Selection for Unsupervised Learning, The Journal of Machine Learning Research **5**, pp 845–889, 2004.
59. S. J. Perantonis, V. Virvilis, Input Feature Extraction for Multilayered Perceptrons Using Supervised Principal Component Analysis, Neural Processing Letters **10**, pp 243–252, 1999.
60. J. Garcke, M. Griebel, Classification with sparse grids using simplicial basis functions, Intell. Data Anal. **6**, pp. 483-502, 2002.
61. M. Elter, R. Schulz-Wendtland, T. Wittenberg, The prediction of breast cancer biopsy outcomes using two CAD approaches that both emphasize an intelligible decision process, Med Phys. **34**, pp. 4164-72, 2007.
62. M.A. Little, P.E. McSharry, E.J. Hunter, J. Spielman, L.O. Ramig, Suitability of dysphonia measurements for telemonitoring of Parkinson's disease. IEEE Trans Biomed Eng. **56**, pp. 1015-1022, 2009.
63. J.W. Smith, J.E. Everhart, W.C. Dickson, W.C. Knowler, R.S. Johannes, Using the ADAP learning algorithm to forecast the onset of diabetes mellitus, In: Proceedings of the Symposium on Computer Applications and Medical Care IEEE Computer Society Press, pp.261-265, 1988.
64. D.D. Lucas, R. Klein, J. Tannahill, D. Ivanova, S. Brandon, D. Domyancic, Y. Zhang, Failure analysis of parameter-induced simulation crashes in climate models, Geoscientific Model Development **6**, pp. 1157-1171, 2013.
65. N. Giannakeas, M.G. Tsipouras, A.T. Tzallas, K. Kyriakidi, Z.E. Tsianou, P. Manousou, A. Hall, E.C. Karvounis, V. Tsianos, E. Tsianos, A clustering based method for collagen proportional area extraction in liver biopsy images (2015) Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS, 2015-November, art. no. 7319047, pp. 3097-3100.
66. T. Hastie, R. Tibshirani, Non-parametric logistic and proportional odds regression, JRSS-C (Applied Statistics) **36**, pp. 260–276, 1987.
67. M. Dash, H. Liu, P. Scheuermann, K. L. Tan, Fast hierarchical clustering and its validation, Data & Knowledge Engineering **44**, pp 109–138, 2003.
68. P. Cortez, A. M. Gonçalves Silva, Using data mining to predict secondary school student performance, In Proceedings of 5th FUture BUsiness TEChnology Conference (FUBUTEC 2008) (pp. 5–12). EUROSIS-ETI, 2008.
69. I-Cheng Yeh, King-Jang Yang, Tao-Ming Ting, Knowledge discovery on RFM model using Bernoulli sequence, Expert Systems with Applications **36**, pp. 5866-5871, 2009.
70. W.H. Wolberg, O.L. Mangasarian, Multisurface method of pattern separation for medical diagnosis applied to breast cytology, Proc Natl Acad Sci U S A. **87**, pp. 9193–9196, 1990.
71. M. Raymer, T.E. Doom, L.A. Kuhn, W.F. Punch, Knowledge discovery in medical and biological datasets using a hybrid Bayes classifier/evolutionary algorithm. IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society, **33** , pp. 802-813, 2003.
72. P. Zhong, M. Fukushima, Regularized nonsmooth Newton method for multi-class support vector machines, Optimization Methods and Software **22**, pp. 225-236, 2007.
73. R.G. Andrzejak, K. Lehnertz, F. Mormann, C. Rieke, P. David, and C. E. Elger, Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state, Phys. Rev. E **64**, pp. 1-8, 2001.

74. M. Koivisto, K. Sood, Exact Bayesian Structure Discovery in Bayesian Networks, The Journal of Machine Learning Research **5**, pp. 549–573, 2004.
75. W. J Nash, T.L. Sellers, S.R. Talbot, A.J. Cawthor, W.B. Ford, The Population Biology of Abalone (_Haliotis_ species) in Tasmania. I. Blacklip Abalone (_H. rubra_) from the North Coast and Islands of Bass Strait, Sea Fisheries Division, Technical Report No. 48 (ISSN 1034-3288), 1994.
76. T.F. Brooks, D.S. Pope, A.M. Marcolini, Airfoil self-noise and prediction. Technical report, NASA RP-1218, July 1989.
77. I.Cheng Yeh, Modeling of strength of high performance concrete using artificial neural networks, Cement and Concrete Research. **28**, pp. 1797-1808, 1998.
78. D. Harrison and D.L. Rubinfeld, Hedonic prices and the demand for clean ai, J. Environ. Economics & Management **5**, pp. 81-102, 1978.
79. M.J.D Powell, A Tolerant Algorithm for Linearly Constrained Optimization Calculations, Mathematical Programming **45**, pp. 547-566, 1989.
80. W. Gropp, E. Lusk, N. Doss, A. Skjellum, A high-performance, portable implementation of the MPI message passing interface standard, Parallel Computing 22, pp. 789-828, 1996.
81. L. Dagum, R. Menon, OpenMP: an industry standard API for shared-memory programming, IEEE Computational Science and Engineering 5, pp. 46-55, 1998.