Constructing features using a hybrid genetic algorithm

Ioannis G. Tsoulos*

Department of Informatics and Telecommunications, University of Ioannina, Greece

Abstract

A hybrid procedure that incorporates Grammatical Evolution and a weight decaying technique is proposed here for classification and regression problems. The procedure is divided in two main phases: in the first phase new features are created from the original ones and in second phase a hybrid genetic algorithm is used to train neural networks on the previously created features. The proposed procedure is applied on a wide range of classification and regression problems from the relevant literature and the results are reported and discussed.

Keywords: Genetic algorithm, machine learning, neural networks, Grammatical Evolution.

1 Introduction

Neural networks are well - established parametric tools [1, 2], used with success in many areas such as physics [3, 4, 5], chemistry [6, 7, 8], economics [9, 10, 11], medicine [12, 13] etc. A neural network usually is expressed as a function $N(\overrightarrow{x}, \overrightarrow{w})$, where $\overrightarrow{x}$ is the input vector and $\overrightarrow{w}$ is the weight vector to be calculated through some optimization process. This optimization process should estimate the weight vector by minimizing the following error function:

$$E\left(N\left(\overrightarrow{x}, \overrightarrow{w}\right)\right) = \sum_{i=1}^{M} \left(N\left(\overrightarrow{x}_i, \overrightarrow{w}\right) - y_i\right)^2 \tag{1}$$

In the above equation the set $\left(\overrightarrow{x_i}, y_i\right),\ i = 1, ..., M$ is the data used to train the neural network, where symbol y_i represents the actual output for the point $\overrightarrow{x_i}$. In many cases the neural network $N(\overrightarrow{x}, \overrightarrow{w})$ can be expressed as a weighted

*Corresponding author. Email: itsoulos@uoi.gr

summation of processing units as proposed in [14] and defined such as:

$$N\left(\overrightarrow{x},\overrightarrow{w}\right)=\sum_{i=1}^{H}w_{(d+2)i-(d+1)}\sigma\left(\sum_{j=1}^{d}x_{j}w_{(d+2)i-(d+1)+j}+w_{(d+2)i}\right) \quad (2)$$

where H is the number of processing units of the neural network and d is the dimension of vector $\overrightarrow{x}$ and $\sigma(x)$ is the sigmoid function defined as:

$$\sigma(x)=\frac{1}{1+\exp(-x)} \quad (3)$$

From the equation 2 one can obtain that the dimension of weight vector w is computed as: $w=(d+2)H$. The function of equation 1 has been minimized with a variety of optimization methods during the past years such as: the Back Propagation method [15, 16], the RPROP method [17, 18, 19], Quasi Newton methods [20, 21], Genetic Algorithms [22, 23], Particle Swarm Optimization [24, 25] etc. All the previously mentioned methods have to overcome two major problems:

- Excessive computational times, because they require processing time proportional to the dimension of the objective problem and the number of processing units as well. For example, a neural network of $H=10$ processing units applied to a test data with $d=3$, is considered as an optimization problem with dimension $w=(d+2)H=50$. A discussion on the effects of the dimensionality on neural networks is provided in [26]. A common approach to overcome this problem is to use the PCA technique to reduce the dimensionality of the objective problem [27, 28, 29] i.e. the parameter d.

- The ovetfitting problem. It is common for these methods to produce poorer results when they are applied to data (test data) not previously used in the optimization process. This problem is discussed in detail in the article of Geman et all [30] as well as in the article of Hawkins [31]. A variety of methods have been proposed to overcome this problem such as are weight sharing [32], pruning [33, 34, 35], the dropout technique [36], early stopping [37, 38], and weight decaying [39, 40].

This article proposes a method that tackle both the above problems using two major steps. During the first step new features are created from the original ones using a procedure based on the Grammatical Evolution technique [41]. This procedure was introduced in the work of Gavrilis et al [42] and it has been with success in many areas such as Spam Identification [43], Fetal heart classification [44], epileptic oscillations in clinical intracranial electroencephalograms [45] etc. The outcomes of the first phase are the modified training and testing data according to the created features. During the second step, a genetic algorithm that incorporates a weight decaying procedure is used to train a neural network on the modified data of the first step.

The rest of this article is organized as follows: in section 2 the proposed method is described in detail, in section 3 the proposed method is tested on a series of well know datasets from the relevant literature and the results are compared to those of a simple genetic algorithm and finally in section 4 some conclusions are presented.

2 Method description

The proposed method has two major phases. During the first phase a procedure based on the Grammatical Evolution technique is utilized in order to create new features from the old ones. The new features are evaluated using an RBF [46] neural network with H hidden nodes. The RBF network is used during this phase instead of a neural network because the training procedure for RBF networks are must faster than these of neural networks of the Equation 2. In the second phase, a hybrid genetic algorithm is to a neural network trained using the constructed features of the first phase.

2.1 The usage of Grammatical Evolution

Grammatical evolution is a genetic programming procedure where the chromosomes are represent production rules from a BNF grammar. The production procedure starts from the start symbol of the grammar and produces programs by replacing non terminal symbols with the right hand of the production rules that will be selected according to the value of each element in the chromosome. In the proposed method the BNF grammar of the Figure 1 was used to create a new feature from the original ones. The parameter N denotes the number of original features. $[0, 255]$. For example, consider the chromosome $x = [9, 8, 6, 4, 16, 10, 17, 23, 8, 14]$ and $N = 3$. The steps to produce the valid expression $f(x) = x_2 + \cos(x_3)$ are listed in Table 1. Each number in the parentheses denotes the sequence number of the production rule. The process to produce N_f features from the original have as follows:

1. Every chromosome Z is split into N_f parts. Each part g_i will be used to construct a feature.
2. For every part g_i construct a feature t_i using the grammar given in 1
3. Create a mapping function

$$G(\overrightarrow{x}, Z) = \left(t_1\left(\overrightarrow{x}, Z\right), t_2\left(\overrightarrow{x}, Z\right), \ldots, t_{N_f}\left(\overrightarrow{x}, Z\right)\right) \tag{4}$$

   where $\overrightarrow{x}$ is a pattern from the original set and Z is the chromosome.

2.2 Feature construction

The main steps of the algorithm used in the first phase are listed below:

Figure 1: BNF grammar of the proposed method.

```
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: Steps to produce a valid expression from the BNF 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))		

1. **Initialization step**
   (a) **Set** iter=0, the current generation number.
   (b) Set TR $= \{(\overrightarrow{x_1}, y_1), (\overrightarrow{x_2}, y_2), \ldots, (\overrightarrow{x_M}, y_M)\}$ the original train set.
   (c) **Set** N_c the number of chromosomes and N_f the number of constructed features.
   (d) **Initialize** randomly in range $[0, 255]$ the integer chromosomes $Z_i, i = 1 \ldots N_c$
   (e) **Set** N_g the maximum number of generations.
   (f) **Set** $p_s \in [0, 1]$ as the selection rate and $p_m \in [0, 1]$ the mutation rate.
2. **Termination check. If** iter>=N_g **goto** step 6.
3. **Calculate** the fitness f_i for every chromosome Z_i of the population:
   (a) **Use** the procedure described in subsection 2.1 and create N_f features.
   (b) **Create** a modified training set

$$\text{TN} = \{(G(\overrightarrow{x_1}, Z_i), y_1), (G(\overrightarrow{x_2}, Z_i), y_2), \ldots, (G(\overrightarrow{x_M}, Z_i), y_M)\} \quad (5)$$

   (c) **Train** an RBF neural network C with H processing units on the modified training set TN using the following train error

$$f_i = \sum_{j=1}^{M} \left(C(G(\overrightarrow{x_j}, Z_i)) - y_j\right)^2 \quad (6)$$

4. **Genetic Operators**

(a) **Selection procedure**: The chromosomes are sorted in descending order according to their fitness value. The first $(1-p_s)\times N_c$ chromosomes are transferred to the next generation. The rest of the chromosomes are substituted by offsprings created through crossover procedure: For every offspring two chromosomes (parents) are selected from the old population using tournament selection. The procedure of tournament selection has as follows: A set of $K > 1$ randomly selected chromosomes is produced and the chromosome with the best fitness value in this set is selected and the others are discarded. Each offspring is created by the parents using the one point crossover. During one point crossover the parent chromosomes are cut at a randomly selected point and their right-hand side subchromosomes are exchanged.

(b) **Mutation procedure:** For every element of each chromosome a random number r in range $[0,1]$ is produced. If $r \leq p_m$ then the corresponding element is randomly altered.

(c) **Replace** the $p_s \times N_c$ worst chromosomes in the population with the offsprings created by the genetic operators.

5. **Set** iter=iter+1 and **goto** Step 2.

6. **Obtain** the best value in the population, denoted as f_l for the corresponding chromosome Z_l and Terminate.

2.3 Weight decay mechanism

The quantity x in the equation 3 of the sigmoid function is calculated through many calculations involving the input patterns as well as the weight vector. If the absolute value of x is too large i.e greater than 20, then the sigmoid function tends to be zero or one and the neural network may lost its generalization capabilities. In order to measure the above effect we can define the bound quantity $B\left(N\left(\overrightarrow{x},\overrightarrow{w}\right),F\right)$ as shown in the Algorithm 1.

2.4 Application of genetic algorithm

The steps of the hybrid genetic algorithm used in the second phase of the proposed algorithm are the following:

1. **Initialization step**

   (a) **Set** iter=0, the current generation number.

   (b) **Set** TN the modified training set, where

   $$\mathrm{TN} = \left\{\left(G\left(\overrightarrow{x_1},Z_l\right),y_1\right),\left(G\left(\overrightarrow{x_2},Z_l\right),y_2\right),\ldots,\left(G\left(\overrightarrow{x_M},Z_l\right),y_M\right)\right\} \quad (7)$$

   (c) **Initialize** randomly the double precision chromosomes $D_i, i = 1\ldots N_c$ in range $[L_N, R_N]$. The size of each chromosome is set to $W = (N_f+2)H$

Algorithm 1 Calculation of the bounding quantity for neural network $N(x, w)$.

1. **Set** $b = 0$
2. **For** $i = 1..K$ **Do**
   (a) **For** $j = 1..M$ **Do**
      i. **Set** $v = \sum_{kT=1}^{d} w_{(d+2)i-(d+i)+k} x_{jk} + w_{(d+2)i}$
      ii. **If** $|v| > F$ **set** $b = b + 1$
   (b) **EndFor**
3. **EndFor**
4. **Return** $\frac{b}{K \star M}$

---

2. **Termination check. If** iter>=N_g **goto** step 6
3. **Fitness calculation step**.
   (a) **For** every chromosome D_i
      i. **Calculate** the quantity $B_i = \sum_{x \in \mathrm{TN}} \left(B\left(N\left(x, D_i \right), F \right) \right)$ using the algorithm 1.
      ii. **Calculate** the quantity $E_i = \sum_{(x,y) \in \mathrm{TN}} \left(N\left(x, D_i \right) - y \right)^2$, the training error of the neural network where the chromosome D_i is used as the weight vector.
      iii. **Set** $f_i = -E_i \left(1 + \lambda B_i^2 \right)$, where $\lambda > 0$ as the fitness of D_i.
   (b) **End For**
4. **Genetic operations Step**. Apply the same genetic operations as in the first algorithm of subsection 2.2.
5. **Set** iter=iter+1 and **goto** step 2
6. **Local Search step.**
   (a) **Obtain** the best chromosome D^* in the genetic population.
   (b) **For** i=1..W **Do**
      i. **Set** $p_i = D_i^*$
      ii. **Set** $LM_i = -\alpha \left| p_i \right|$
      iii. **Set** $RM_i = \alpha \left| p_i \right|$, α being a positive number with $\alpha > 1$.
   (c) **EndFor**
   (d) **Set** $L^* = \mathcal{L}\left(D^*, LM, RM \right)$ where $\mathcal{L}()$ is a local optimization method that searches for a local optimum of $N\left(x, D^* \right)$ inside the bounds $\left[\overrightarrow{LM}, \overrightarrow{RM} \right]$. The TOLMIN [47] local optimization procedure used

Table 2: Experimental parameters.

PARAMETER	VALUE
H	10
N_c	500
N_f	2
p_s	0.10
p_m	0.05
N_g	200
L_N	-10.0
R_N	10.0
F	20.0
λ	100.0
α	5.0

in the above algorithm, which is modified version BFGS (Broyden–Fletcher–Goldfarb–Shanno) local optimization procedure[48].

(e) **Apply** the optimized neural network $N(x, D^*)$ to the test set, that has been modified using the same transformation procedure as in the train set and report the final results.

3 Experiments

The proposed method was tested against neural network trained by a genetic algorithm (denoted as MLP GEN) on a series of classification and regression datasets. The software for the algorithm was coded using ANSI C++ and all the experiments were conducted using the OpenMP library [49] for parallelization. The experiments were executed 30 times using different seed for the random generator each time and averages were taken. For classification datasets the average classification error on the test set is reported and for regression datasets the average mean squared error on the test set is shown. 10 fold cross validation is used in the conducted experiments. The parameters used in the experiments are listed in Table 2.

3.1 Experimental datasets

The following classification datasets were acquired from the Machine Learning Repository `http://www.ics.uci.edu/~mlearn/MLRepository.html` and from `https://sci2s.ugr.es/keel/`:

1. **Balance** dataset: Used to model psychological experimental results. The dataset has 625 instances with 4 features each.

2. **Dermatology** dataset: Dataset used for differential diagnosis of erythemato-squamous diseases. The dataset has 366 instances of 34 features each.

3. **Glass** dataset. This dataset contains glass component analysis for glass pieces that belong to 6 classes. The dataset contains 214 examples with 10 features each.

4. **Hayes Roth** dataset: This dataset[50] contains **5** numeric-valued attributes and 132 patterns.

5. **Heart** dataset: Dataset used to discriminate between absence or presence of heart disease. The dataset has 270 instances of 13 features each.

6. **Ionosphere** dataset: The ionosphere dataset (ION in the following tables) contains data from the Johns Hopkins Ionosphere database. The two-class dataset contains 351 examples of 34 features each.

7. **Parkinsons** dataset: This dataset[51] is composed of a range of biomedical voice measurements from 31 people, 23 with Parkinson's disease (PD).The dataset has 22 features.

8. **Pima** dataset: The Pima Indians Diabetes dataset contains 768 examples of 8 attributes each that are classified into two categories: healthy and diabetic.

9. **PopFailures** dataset: Dataset used in meteorology. The dataset has 540 instances of 18 features each.

10. **Spiral** dataset: The spiral artificial dataset contains 1000 two-dimensional examples that belong to two classes (500 examples each). The number of the features is 2. The data in the first class are created using the following formula: $x_1 = 0.5t\cos(0.08t)$, $x_2 = 0.5t\cos\left(0.08t + \frac{\pi}{2}\right)$ and the second class data using**:** $x_1 = 0.5t\cos(0.08t + \pi)$, $x_2 = 0.5t\cos\left(0.08t + \frac{3\pi}{2}\right)$

11. **Wine**: The wine recognition dataset (WINE) contains data from wine chemical analysis. It contains 178 examples of 13 features each that are classified into three classes.

12. **Wdbc** dataset: The Wisconsin diagnostic breast cancer dataset (WDBC) contains data for breast tumors. The dataset has 30 features.

Also an EEG dataset was used, that is is available from [52, 53] and includes recordings for both healthy and epileptic subjects, is used. The dataset includes five subsets (denoted as Z, O, N, F, and S) each containing 100 single-channel EEG segments, each one having 23.6-second duration. In the current work two variants of the dataset was used:

1. In the first (**ZO_NF_S**), all the EEG segments from the dataset were used and they were classified into three different classes: Z and O types of EEG segments were combined to a single class, N and F types were also combined to a single class, and type S was the third class. This set is the one closest to real medical applications including three categories; normal (i.e., types Z and O), seizure free (i.e., types N and F) and seizure (i.e., type S).

2. In the second (**Z_F_S**) dataset, which is similar with the first one, a subset of the EEG segments from the dataset were employed. The normal class includes only the Z-type EEG segments, the seizure-free class the F-type EEG segments, and the seizure class the S-type.

3. In the third (**Z_O_N_F_S**) dataset The dataset consists of five sets (denoted as Z, O, N, F and S) each containing 100 single-channel EEG segments each having 23.6 sec duration. Sets Z and O have been taken from surface EEG recordings of five healthy volunteers with eye open and closed, respectively. Signals in two sets have been measured in seizure-free intervals from five patients in the epileptogenic zone (F) and from the hippocampal formation of the opposite hemisphere of the brain (N). Set S contains seizure activity, selected from all recording sites exhibiting ictal activity. Sets Z and O have been recorded extracranially, whereas sets N, F and S have been recorded intracranially.

The regression datasets are available from the Statlib URL `ftp://lib.stat.cmu.edu/datasets/index.html` and other sources:

1. **BK** dataset. This dataset comes from Smoothing Methods in Statistics [54] and is used to estimate the points scored per minute in a basketball game. The dataset has 96 patterns of 4 features each.

2. **BL** dataset: This dataset can be downloaded from StatLib. It contains data from an experiment on the affects of machine adjustments on the time to count bolts. It contains 40 patters of 7 features each.

3. **Housing** dataset. This dataset was taken from the StatLib library which is maintained at Carnegie Mellon University and it is described in [55].

4. **Laser** dataset. Dataset used in laser experiments. The dataset has 993 instances of 4 features each.

5. **NT** dataset. This dataset contains data from [56] that examined whether the true mean body temperature is 98.6 F. The number of patterns is 131 and the number of features 2.

6. **Quake** dataset. The objective here is to approximate the strength of a earthquake. The dataset has 2178 instances of 3 features each.

7. **FA** dataset. The FA dataset contains percentage of body fat, age, weight, height, and ten body circumference measurements. The goal is to fit body fat to the other measurements. The number of the features is 18. The total number of patterns is 252.

8. **PY** dataset (Pyrimidines problem). The source of this dataset is the URL: `https://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html` and it is a problem of 27 attributes and 74 number of patterns. The task consists of Learning Quantitative Structure Activity Relationships (QSARs) and provided by [57].

Table 3: Experimental results for classification datasets.

DATASET	MLP GEN	FC MLP	**GAIN**
BALANCE	8.23%	0.30%	**96.35%**
DERMATOLOGY	10.01%	4.98%	**50.25%**
GLASS	58.03%	45.84%	**21.00%**
HAYES ROTH	35.26%	23.26%	**34.03%**
HEART	25.46%	17.71%	**30.44%**
IONOSPHERE	13.67%	8.42%	**38.41%**
PARKINSONS	17.47%	10.10%	**36.46%**
PIMA	32.98%	23.76%	**27.96%**
POPFAILURES	7.66%	4.66%	**39.16%**
SPIRAL	45.71%	26.53%	**41.96%**
WINE	20.82%	7.31%	**64.89%**
WDBC	6.32%	3.47%	**45.09%**
Z_F_S	9.42%	5.52%	**41.40%**
Z_O_N_F_S	60.38%	31.20%	**48.33%**
ZO_NF_S	8.06%	4.00%	**50.37%**

Table 4: Experiments for regression datasets.

DATASET	MLP GEN	FC MLP	**GAIN**
BK	0.21	0.03	**85.71%**
BL	0.84	0.005	**99.40%**
Housing	30.05	10.77	**64.16%**
Laser	0.003	0.002	**33.33%**
NT	1.11	0.01	**99.10%**
Quake	0.07	0.03	**57.14%**
FA	0.04	0.01	**75.00%**
PY	0.21	0.02	**90.47%**

3.2 Experimental results

For the case of classification datasets the results are reported in Table 3 and for the case of regression datasets the results are reported in Table 4. The column MLP GEN denotes the neural network with H hidden nodes that was trained by a genetic algorithm with N_g chromosomes and the column FC MLP denotes the proposed method. Also, the column GAIN indicates the percentage gain in test error (classification or regression) obtained by the proposed method. From the conducted experiments it is evident that the proposed method outperforms the simple genetic algorithm in terms of efficiency. Also, the percentage gain is too high in many cases like in the regression datasets. On the other hand the proposed method is slower than a simple genetic algorithm because it relies on two steps and every step involves the usage of a genetic algorithm.

4 Conclusions

A hybrid method was proposed here for classification and regression problems. The method is composed by two steps: during the first step a recently introduced method was used to create artificial features from the original ones. The feature construction method has utilized the commonly used technique of Grammatical Evolution to produce artificial features. The outcome of the first step was the modified train and test sets of the objective problem. During the second step, a hybrid genetic algorithm which preserves the generalization abilities of the neural network was incorporated. The proposed method was tested on a variety of classification and regression problems and the results were very promising.

Compliance with Ethical Standards

All authors declare that they have no has no conflict of interest.

References

[1] C. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995.

[2] G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of Control Signals and Systems **2**, pp. 303-314, 1989.

[3] P. Baldi, K. Cranmer, T. Faucett et al, Parameterized neural networks for high-energy physics, Eur. Phys. J. C **76**, 2016.

[4] J. J. Valdas and G. Bonham-Carter, Time dependent neural network models for detecting changes of state in complex processes: Applications in earth sciences and astronomy, Neural Networks **19**, pp. 196-207, 2006

[5] G. Carleo,M. Troyer, Solving the quantum many-body problem with artificial neural networks, Science **355**, pp. 602-606, 2017.

[6] Lin Shen, Jingheng Wu, and Weitao Yang, Multiscale Quantum Mechanics/Molecular Mechanics Simulations with Neural Networks, Journal of Chemical Theory and Computation **12**, pp. 4934-4946, 2016.

[7] Sergei Manzhos, Richard Dawes, Tucker Carrington, Neural network-based approaches for building high dimensional and quantum dynamics-friendly potential energy surfaces, Int. J. Quantum Chem. **115**, pp. 1012-1020, 2015.

[8] Jennifer N. Wei, David Duvenaud, and Alán Aspuru-Guzik, Neural Networks for the Prediction of Organic Chemistry Reactions, ACS Central Science **2**, pp. 725-732, 2016.

[9] Lukas Falat and Lucia Pancikova, Quantitative Modelling in Economics with Advanced Artificial Neural Networks, Procedia Economics and Finance **34**, pp. 194-201, 2015.

[10] Mohammad Namazi, Ahmad Shokrolahi, Mohammad Sadeghzadeh Maharluie, Detecting and ranking cash flow risk factors via artificial neural networks technique, Journal of Business Research **69**, pp. 1801-1806, 2016.

[11] G. Tkacz, Neural network forecasting of Canadian GDP growth, International Journal of Forecasting **17**, pp. 57-69, 2001.

[12] Igor I. Baskin, David Winkler and Igor V. Tetko, A renaissance of neural networks in drug discovery, Expert Opinion on Drug Discovery **11**, pp. 785-795, 2016.

[13] Ronadl Bartzatt, Prediction of Novel Anti-Ebola Virus Compounds Utilizing Artificial Neural Network (ANN), Chemistry Faculty Publications **49**, pp. 16-34, 2018.

[14] I.G. Tsoulos, D. Gavrilis, E. Glavas, Neural network construction and training using grammatical evolution, Neurocomputing **72**, pp. 269-277, 2008.

[15] D.E. Rumelhart, G.E. Hinton and R.J. Williams, Learning representations by back-propagating errors, Nature **323**, pp. 533 - 536 , 1986.

[16] T. Chen and S. Zhong, Privacy-Preserving Backpropagation Neural Network Learning, IEEE Transactions on Neural Networks **20**, , pp. 1554-1564, 2009.

[17] M. Riedmiller and H. Braun, A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP algorithm, Proc. of the IEEE Intl. Conf. on Neural Networks, San Francisco, CA, pp. 586–591, 1993.

[18] T. Pajchrowski, K. Zawirski and K. Nowopolski, Neural Speed Controller Trained Online by Means of Modified RPROP Algorithm, IEEE Transactions on Industrial Informatics **11**, pp. 560-568, 2015.

[19] Rinda Parama Satya Hermanto, Suharjito, Diana, Ariadi Nugroho, Waiting-Time Estimation in Bank Customer Queues using RPROP Neural Networks, Procedia Computer Science **135**, pp. 35-42, 2018.

[20] B. Robitaille and B. Marcos and M. Veillette and G. Payre, Modified quasi-Newton methods for training neural networks, Computers & Chemical Engineering **20**, pp. 1133-1140, 1996.

[21] Q. Liu, J. Liu, R. Sang, J. Li, T. Zhang and Q. Zhang, Fast Neural Network Training on FPGA Using Quasi-Newton Optimization Method,IEEE Transactions on Very Large Scale Integration (VLSI) Systems **26**, pp. 1575-1579, 2018.

[22] F. H. F. Leung, H. K. Lam, S. H. Ling and P. K. S. Tam, Tuning of the structure and parameters of a neural network using an improved genetic algorithm, IEEE Transactions on Neural Networks **14**, pp. 79-88, 2003

[23] X. Yao, Evolving artificial neural networks, Proceedings of the IEEE, 87(9), pp. 1423-1447, 1999.

[24] C. Zhang, H. Shao and Y. Li, Particle swarm optimisation for evolving artificial neural network, IEEE International Conference on Systems, Man, and Cybernetics, , pp. 2487-2490, 2000.

[25] Jianbo Yu, Shijin Wang, Lifeng Xi, Evolving artificial neural networks using an improved PSO and DPSO **71**, pp. 1054-1060, 2008.

[26] Verleysen M., Francois D., Simon G., Wertz V., On the effects of dimensionality on data analysis with neural networks. In: Mira J., Álvarez J.R. (eds) Artificial Neural Nets Problem Solving Methods. IWANN 2003. Lecture Notes in Computer Science, vol 2687. Springer, Berlin, Heidelberg. 2003.

[27] Burcu Erkmen, Tülay Yıldırım, Improving classification performance of sonar targets by applying general regression neural network with PCA, Expert Systems with Applications **35**, pp. 472-475, 2008.

[28] Jing Zhou, Aihuang Guo, Branko Celler, Steven Su, Fault detection and identification spanning multiple processes by integrating PCA with neural network, Applied Soft Computing **14**, pp. 4-11, 2014.

[29] Ravi Kumar G., Nagamani K., Anjan Babu G., A Framework of Dimensionality Reduction Utilizing PCA for Neural Network Prediction. In: Borah S., Emilia Balas V., Polkowski Z. (eds) Advances in Data Science and Management. Lecture Notes on Data Engineering and Communications Technologies, vol 37. Springer, Singapore. 2020

[30] S. Geman, E. Bienenstock and R. Doursat, Neural networks and the bias/variance dilemma, Neural Computation 4 , pp. 1 - 58, 1992.

[31] Douglas M. Hawkins, The Problem of Overfitting, J. Chem. Inf. Comput. Sci. **44**, pp. 1–12, 2004.

[32] S.J. Nowlan and G.E. Hinton, Simplifying neural networks by soft weight sharing, Neural Computation 4, pp. 473-493, 1992.

[33] S.J. Hanson and L.Y. Pratt, Comparing biases for minimal network construction with back propagation, In D.S. Touretzky (Ed.), Advances in Neural Information Processing Systems, Volume 1, pp. 177-185, San Mateo, CA: Morgan Kaufmann, 1989.

[34] M.C. Mozer and P. Smolensky, Skeletonization: a technique for trimming the fat from a network via relevance assesment. In D.S. Touretzky (Ed.), Advances in Neural Processing Systems, Volume 1, pp. 107-115, San Mateo CA: Morgan Kaufmann, 1989.

[35] M. Augasta and T. Kathirvalavakumar, Pruning algorithms of neural networks — a comparative study, Central European Journal of Computer Science, 2003.

[36] Nitish Srivastava, G E Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan R Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, Journal of Machine Learning Research **15**, pp. 1929-1958, 2014.

[37] Lutz Prechelt, Automatic early stopping using cross validation: quantifying the criteria, Neural Networks **11**, pp. 761-767, 1998.

[38] X. Wu and J. Liu, A New Early Stopping Algorithm for Improving Neural Network Generalization, 2009 Second International Conference on Intelligent Computation Technology and Automation, Changsha, Hunan, 2009, pp. 15-18.

[39] N. K. Treadgold and T. D. Gedeon, Simulated annealing and weight decay in adaptive learning: the SARPROP algorithm,IEEE Transactions on Neural Networks **9**, pp. 662-668, 1998.

[40] M. Carvalho and T. B. Ludermir, Particle Swarm Optimization of Feed-Forward Neural Networks with Weight Decay, 2006 Sixth International Conference on Hybrid Intelligent Systems (HIS'06), Rio de Janeiro, Brazil, 2006, pp. 5-5.

[41] M. O'Neill, C. Ryan, Grammatical evolution, IEEE Trans. Evol. Comput. **5,**pp. 349–358, 2001.

[42] Dimitris Gavrilis, Ioannis G. Tsoulos, Evangelos Dermatas, Selecting and constructing features using grammatical evolution, Pattern Recognition Letters **29**,pp. 1358-1365, 2008.

[43] 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.

[44] 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

[45] 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

[46] J. Park and I. W. Sandberg, Universal Approximation Using Radial-Basis-Function Networks, Neural Computation 3, pp. 246-257, 1991.

[47] M.J.D. Powell, A Tolerant Algorithm for Linearly Constrained Optimization Calculations, Mathematical Programming **45**, pp 547, 1989.

[48] R. Fletcher, A new approach to variable metric algorithms, Computer Journal **13**, pp. 317-322, 1970.

[49] R. Chandra, L. Dagum, D. Kohr, D. Maydan,J. McDonald and R. Menon, Parallel Programming in OpenMP, Morgan Kaufmann Publishers Inc., 2001.

[50] 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.

[51] Max A. Little, Patrick E. McSharry, Eric J. Hunter, Lorraine O. Ramig (2008), 'Suitability of dysphonia measurements for telemonitoring of Parkinson's disease', IEEE Transactions on Biomedical Engineering **56**, pp. 1015-1022, 2009.

[52] 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," Physical Review E, vol. 64, no. 6, Article ID 061907, 8 pages, 2001.

[53] A. T. Tzallas, M. G. Tsipouras, and D. I. Fotiadis, "Automatic Seizure Detection Based on Time-Frequency Analysis and Artificial Neural Networks," Computational Intelligence and Neuroscience, vol. 2007, Article ID 80510, 13 pages, 2007. doi:10.1155/2007/80510

[54] J.S. Simonoff, Smooting Methods in Statistics, Springer - Verlag, 1996.

[55] D. Harrison and D.L. Rubinfeld, Hedonic prices and the demand for clean ai, J. Environ. Economics & Management **5**, pp. 81-102, 1978.

[56] Mackowiak, P.A., Wasserman, S.S., Levine, M.M., 1992. A critical appraisal of 98.6 degrees f, the upper limit of the normal body temperature, and other legacies of Carl Reinhold August Wunderlich. J. Amer. Med. Assoc. 268, 1578–1580

[57] R.D. King, S. Muggleton, R. Lewis, M.J.E. Sternberg, Proc. Nat. Acad. Sci. USA **89**, pp. 11322–11326, 1992.