signals

MDPI

Article

Constructing Features Using a Hybrid Genetic Algorithm

Ioannis G. Tsoulos

Department of Informatics and Telecommunications, University of Ioannina, 45110 Ioannina, Greece; itsoulos@uoi.gr

Abstract: A hybrid procedure that incorporates grammatical evolution and a weight decaying technique is proposed here for various classification and regression problems. The proposed method has two main phases: the creation of features and the evaluation of these features. During the first phase, using grammatical evolution, new features are created as non-linear combinations of the original features of the datasets. In the second phase, based on the characteristics of the first phase, the original dataset is modified and a neural network trained with a genetic algorithm is applied to this dataset. The proposed method was applied to an extremely wide set of datasets from the relevant literature and the experimental results were compared with four other techniques.

Keywords: genetic algorithm; machine learning; neural networks; grammatical evolution

Citation: Tsoulos, I.G. Constructing Features Using a Hybrid Genetic Algorithm. *Signals* **2022**, *3*, 174–188. https://doi.org/10.3390/signals3020012

Academic Editors: Justin A. Blanco and Manuel Duarte Ortigueira

Received: 24 February 2022
Accepted: 31 March 2022
Published: 6 April 2022

1. Introduction

Artificial neural networks (ANNs) are programming tools [1,2], based on a series of parameters that are commonly called weights or processing units. They have been used in a variety of problems from different scientific areas such as physics [3–5], chemistry [6–8], economics [9–11] and medicine [12,13]. A common way to express a neural network is a function $N(\overrightarrow{x}, \overrightarrow{w})$, with $\overrightarrow{x}$ as the input vector (commonly called pattern) and $\overrightarrow{w}$ as the weight vector. A method that trains a neural network should be used to estimate the vector $\overrightarrow{w}$ for a certain problem. The training procedure can be also formulated as an optimization problem, wherein the objective is to minimize the so-called 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 Equation (1), the set $\left(\overrightarrow{x_i}, y_i\right)$, $i = 1, \ldots, M$ is the dataset used to train the neural network, with y_i being the actual output for the point $\overrightarrow{x_i}$. The neural network form used here was also considered in [14]. Suppose we have a neural network with a processing level that uses the sigmoid function as an output function. Every output of the network is defined as

$$o_i(x) = \sigma\left(p_i^T x + \theta_i\right), \tag{2}$$

where p_i is the weight vector and θ_i is the bias for the output i. For a neural network with H hidden nodes, the final output function can be written as

$$N(x) = \sum_{i=1}^{H} v_i o_i(x), \tag{3}$$

where v_i is the output weight for the processing unit i. Hence, by using one vector for all parameters (weights and biases), the neural network can be written in the following form:

$$N\left(\overrightarrow{x}, \overrightarrow{w}\right) = \sum_{i=1}^{H} w_{(d+2)i-(d+1)} \sigma\left(\sum_{j=1}^{d} x_j w_{(d+2)i-(d+1)+j} + w_{(d+2)i}\right) \tag{4}$$

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

$$\sigma(x) = \frac{1}{1+\exp(-x)} \tag{5}$$

From Equation (4), one can obtain that the dimension of the 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 [17–19], Quasi Newton methods [20,21] and particle swarm optimization [22,23]. All the previously mentioned methods have to overcome two major problems:

- Excessive computational times, because they require a 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 an optimization problem with dimension $w = (d+2)H = 50$. This means that the total number of network parameters is growing extremely quickly, which results in a longer computation time than the corresponding universal optimization method. An extensive discussion of the problems caused by the dimensionality of neural networks was presented in [24]. A common approach to overcome this problem is to use the PCA technique to reduce the dimensionality of the objective problem [25–27], i.e., the parameter d.
- The overfitting problem, which is quite common for these methods to produce poor results when they are applied to data (test data) not previously used in the training procedure. This problem was discussed in detail in the article by Geman et al. [28] as well as in the article of Hawkins [29]. A variety of methods have been proposed to overcome this problem, such as weight sharing [30], pruning [31–33], the dropout technique [34], early stopping [35,36], and weight decaying [37,38].

This article proposes a method that tackles both the above problems using two major steps. During the first step, a new set of features was created from the initial features using a procedure based on the grammatical evolution technique [39]. A feature is a measurement that defines a property of the objective problem and the series of all measurements forms a pattern. A feature can be an integer value, a double precision value or even a string literature. In our case, we only consider numeric values for the features. The number of features of each pattern is the dimensionality of the problem defined as d in this work. The procedure of feature construction with grammatical evolution was introduced in the work of Gavrilis et al. [40] and it has been used with success in spam identification [41], fetal heart classification [42], epileptic oscillations in clinical intracranial electroencephalograms [43], etc. The outcomes of the first phase are the training and testing data which have been modified 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.

Genetic algorithms are methods based on biological observations such as reproduction and mutation [44,45]. The genetic algorithms work by creating and maintaining a population of candidate solutions (chromosomes). This population is iteratively altered though operations such as crossover and mutation until some stopping criteria are met. They have many advantages, such as simplicity of implementation, endurance in noise, can be easily parallelized, etc. Furthermore, they have been applied to many problems such as aerodynamic optimization [46], steel structure optimization [47] and brain images [48]. They have been used to train neural networks in various research papers, such as in the work of Leung et al. [49] which estimates the structure and weights of a neural network through a genetic algorithm, the evolution of a neural networks for daily rainfall–runoff forecasting [50], and the evolution of neural networks to predict the deformation modulus of rock masses [51] etc.

The idea of feature construction has been examined by various researchers in the relevant literature, such as the work of Smith and Bull [52], who used a tree genetic programming approach to construct features from the original ones. Another approach to constructing features using genetic programming was proposed by Neshatian et al. [53], where the genetic programming utilizes an entropy-based fitness function that maximizes the purity of class intervals. Another evolutionary approach was proposed by Li and Yin for feature selection using gene expression data [54]. Finally, a recent work that utilizes a genetic programming approach and the information gain ratio (IGR) was proposed by Ma and Teng [55] to construct features from the original ones.

In problems of classification and regression, as the number of features increases, additional examples are needed in order to achieve good results in training a model but also to maintain good generalization skills in unknown data. Of course, adding new examples to the training process is almost never possible and this results in the poor performance of the control data. For this reason, the original dataset must be transformed into a new one, which gives better generalization skills to the learning models. According to Cover's theorem [56], there is at least one non-linear extension of the original feature vector, so that with this extension, a linear separation of the set of patterns can be made. Many techniques have been proposed in this direction that try to detect such non-linear extensions. The proposed method uses a hybrid approach, in which first new features are constructed using grammatical evolution and then these features are evaluated by a neural network that appropriately trains a genetic algorithm. In the first phase, the creation of new features is performed in such a way as to achieve the best possible learning accuracy. The methods that can be used to convert attributes are grouped into three categories: feature selection, feature construction, and feature reduction. The second case is the most difficult, as it does not simply require reducing the size of the problem, but also the non-linear creation of new features from old ones.

The proposed technique can outperform other techniques from the modern literature as it does not require prior knowledge of the objective problem, and can thus be applied with the exact same procedure to both categorization problems and function learning problems. In addition, it can be used to discover hidden function dependencies between the original features of the problem and, because it is based on grammatical evolution, the user can add and subtract functions or even allow the algorithm to construct new functions to better learn the dataset. Nonetheless, the final characteristics of the method can be evaluated by any computational intelligence model without any additional processing. In the present method, these characteristics are evaluated by an artificial neural network, but this is something that could change.

The rest of this paper 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-known 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. In the first phase, a procedure that exploits the grammatical evolution technique is used in order to create new features from the old ones. The new features are evaluated using a radial basis function (RBF) [57] 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 much faster than those of neural networks. In the second phase, a hybrid genetic algorithm trains a neural network using the constructed features of the first phase.

2.1. The Usage of Grammatical Evolution

Grammatical evolution is an evolutionary procedure whereby the chromosomes represent production rules from a BNF (Backus–Naur form) grammar [58], which is very often

used to describe the syntax of programming languages, document formats, etc. These grammars are defined as the set $G = (N, T, S, P)$ where:

- N is the set of non-terminal symbols, which produce a series of terminal symbols through production rules.
- T is the set of terminal symbols.
- S is a non-terminal symbol which is also called the start symbol.
- P is a set of production rules in the form $A \to a$ or $A \to aB,\ A, B \in N,\ a \in T$.

The production procedure starts from the start symbol of the BNF grammar and iteratively 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 Figure 1 was used to create a new feature from the initial features. The symbols that are in <> are considered non-terminal symbols. The parameter N denotes the number of original features. Typically, a chromosome x in grammatical evolution is expressed as a series of binary values 0 or 1. In the current work, in order to simplify the mapping procedure and to increase the speed of the algorithm, every element of each chromosome is considered an integer in a predefined range. In our case, the range $[0, 255]$ was used but of course, this could be easily changed.

Take, for example, the chromosome $x = [9, 8, 6, 4, 16, 10, 17, 23, 8, 14]$ and $N = 3$. The valid expression $f(x) = x_2 + \cos(x_3)$ is created using a series of production steps shown in Table 1. An expression is considered valid if it only contains terminal symbols. Each number in the parentheses stands for the sequence number of the production rule. Hence, the process to produce N_f features from the original is 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 Figure 1.
3. Create a mapping function:

$$G(\overrightarrow{x}, Z) = \left(t_1(\overrightarrow{x}, Z), t_2(\overrightarrow{x}, Z), \ldots, t_{N_f}(\overrightarrow{x}, Z)\right) \tag{6}$$

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

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))		

```
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)
```

Figure 1. BNF grammar of the proposed method.

2.2. Feature Construction

The first phase of feature construction presented below has also been used as a feature construction mechanism in the initial work of Gavrilis et al. [40]. In this phase, a genetic algorithm constructs new features from the original and the training error of an RBF network on the dataset created with the new features is used. The steps of the algorithm for the first phase are:

1. **Initialization step**
   (a) **Set** iter = 0, generation number.
   (b) **Construct** the set TR $= \{(\overrightarrow{x_1}, y_1), (\overrightarrow{x_2}, y_2), \ldots, (\overrightarrow{x_M}, y_M)\}$, which is the original training set.
   (c) **Set** N_c as the number of chromosomes and N_f as the number of desired constructed features. These options are defined by the user.
   (d) **Initialize** randomly in range $[0, 255]$ the integer chromosomes $Z_i, i = 1 \ldots N_c$
   (e) **Set** N_g as the maximum number of generations allowed.
   (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 $\geq N_g$ **go to** step 6.
3. **Estimate** the fitness f_i of every chromosome Z_i with the following procedure:
   (a) **Use** the procedure described in Section 2.1 and create N_f features.
   (b) **Create** a modified training set :

$$\text{TN} = \{(G(\overrightarrow{x_1}, Z_i), y_1), (G(\overrightarrow{x_2}, Z_i), y_2), \ldots, (G(\overrightarrow{x_M}, Z_i), y_M)\} \quad (7)$$

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

$$f_i = \sum_{j=1}^{M} \left(C\left(G\left(\overrightarrow{x_j}, Z_i \right) \right) - y_j \right)^2 \tag{8}$$

4. **Genetic Operators**
   (a) **Selection procedure:** initially, the chromosomes are sorted according to their fitness value. The best chromosomes are placed in the beginning of the population and the worst at the end. The best $(1 - p_s) \times N_c$ chromosomes are transferred to the next generation intact. The remaining chromosomes are substituted by offspring created through the crossover and mutation procedures.
   (b) **Crossover procedure:** in this process, for every produced offspring, two mating chromosomes (parents) are selected from the previous population using tournament selection. Tournament selection is a rather simple selection mechanism defined as: first a set of $K > 1$ randomly selected chromosomes is constructed and subsequently the chromosome with the best fitness value in the previous set is selected as the mating chromosome. Having selected the two parents for the offspring, the offspring is formed using the one point crossover. In one-point crossover, a random point is selected for the two parents and their right-hand side subchromosomes are exchanged.
   (c) **Mutation procedure:** for every element of each chromosome, a random number $r \in [0, 1]$ is taken. If $r \leq p_m$, then this element is randomly altered by producing a new integer number.
5. **Set** iter = iter+1 and **go to** Step 2.
6. **Get** the best chromosome in the population defined as Z_l with the corresponding fitness value f_l and **Terminate**.

2.3. Weight Decay Mechanism

The quantity x in Equation (5) of the sigmoid function is calculated through many calculations involving the input patterns as well as the weight vector. If the value within the function is excessively large, then the sigmoid function tends towards one and this will result in the neural network losing what generalization possibilities it has. In order to estimate the effect of this issue, the quantity $B\left(N\left(\overrightarrow{x}, \overrightarrow{w}\right), F\right)$ is defined as shown in Algorithm 1.

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

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

2.4. Application of Genetic Algorithm

The following is a hybrid genetic algorithm used to train artificial neural networks in the modified dataset. The purpose of this algorithm is to train the artificial neural network in such a way that it does not lose its generalizing abilities. For this purpose, it uses a fitness function that consists of the neural network training error, but also a punitive factor

is added. This penalty factor aims to ensure that network weights do not attain excessively high values during training. This technique can be directly applied to a neural network without having previously performed the first phase of feature construction. The main steps of the hybrid genetic algorithm used in the second phase are:

1. **Initialization step**
   (a) **Set** iter = 0 as the generation number.
   (b) **Set** TN as the modified training set, where:

$$\mathrm{TN} = \left\{ \left(G\left(\overrightarrow{x_1}, Z_l\right), y_1\right), \left(G\left(\overrightarrow{x_2}, Z_l\right), y_2\right), \ldots, \left(G\left(\overrightarrow{x_M}, Z_l\right), y_M\right) \right\} \quad (9)$$

   (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 $W = \left(N_f + 2\right) H$.
2. **Termination check. If** iter $\geq N_g$ **goto** step 6
3. **Fitness calculation step**.
   (a) **For** every chromosome D_i:
      i. **Calculate** the quantity $B_i = \sum_{x \in \mathrm{TN}} (B(N(x, D_i), F))$ using Algorithm 1.
      ii. **Calculate** the quantity $E_i = \sum_{(x,y) \in \mathrm{TN}} (N(x, D_i) - y)^2$, 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 Section 2.2.
5. **Set** iter = iter + 1 and **go to** step 2
6. **Local search step.**
   (a) **Get** the best chromosome D^* of the population.
   (b) **For** i = 1...W **Do**
      i. **Set** $p_i = D_i^*$
      ii. **Set** $LM_i = -\alpha |p_i|$
      iii. **Set** $RM_i = \alpha |p_i|$, $\alpha > 1$.
   (c) **End for**
   (d) **Set** $L^* = \mathcal{L}(D^*, LM, RM)$ where $\mathcal{L}()$ is a local optimization method procedure that searches for a local optimum of $N(x, D^*)$ inside the bounds $\left[\overrightarrow{LM}, \overrightarrow{RM}\right]$. The TOLMIN [59] local optimization procedure used in the above algorithm is a modified version of the BFGS local optimization procedure [60].
   (e) **Apply** the optimized neural network $N(x, D^*)$ to the test set, which has been modified using the same transformation procedure as in the train set, and report the final results.

3. Experiments

The software for the algorithm was coded using ANSI C++ and was utilized for parallelization and to accelerate the genetic algorithm for all of the OpenMP library [61]. Every experiment was executed 30 times with a different speed for the random generator each time and averages were measured and reported. The function used for random numbers was the drand48() function of the C programming language. The classification error is reported for the classification datasets on the test set and the average mean squared error for regression datasets. Furthermore, for more reliability in the results, the commonly used method of 10-fold cross-validation was used. The values for the parameters of the used algorithms are reported in Table 2.

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

3.1. Experimental Datasets

The method was tested on a series of regression and classification datasets which were mostly obtained from two repositories:

1. The Machine Learning Repository http://www.ics.uci.edu/~mlearn/MLRepository.html (accessed on 1 April 2022);
2. The Keel Repository https://sci2s.ugr.es/keel/ (accessed on 1 April 2022).

The description for these datasets is as follows:

1. **Balance** dataset [62], used in psychological experiments.
2. **Dermatology** dataset [63], which is used for differential diagnosis of erythemato-squamous diseases.
3. **Glass** dataset. This dataset contains a glass component analysis for glass pieces that belong to 6 classes.
4. **Hayes Roth** dataset [64].
5. **Heart** dataset [65], used to detect heart disease.
6. **Ionosphere** dataset, a meteorological dataset used in various research papers [66,67].
7. **Parkinsons** dataset,[68] which is created using a range of biomedical voice measurements from 31 people, among which 23 have Parkinson's disease (PD). The dataset has 22 features.
8. **Pima** dataset, related to diabetes.
9. **PopFailures** dataset [69], used in meteorology.
10. **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: $x_1 = 0.5t\cos(0.08t + \pi)$, $x_2 = 0.5t\cos\left(0.08t + \frac{3\pi}{2}\right)$
11. **Wine** dataset, which is related to the chemical analysis of wines and it has been used in comparison in various research papers [70,71].
12. **Wdbc** dataset, which contains data for breast tumors.
13. As a real-world example, consider an EEG dataset described in [72,73] which is used here. This dataset consists of five sets (denoted as Z, O, N, F and S), each containing 100 single-channel EEG segments which each have 23.6 s duration. With different combinations of these sets, the produced datasets are Z_F_S, ZO_NF_S, ZONF_S.

The regression datasets are available from the Statlib URL http://lib.stat.cmu.edu/datasets/ (accessed on 1 April 2022) and other sources:

1. **BK** dataset. This dataset comes from smoothing methods in statistics [74] 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 effects of machine adjustments on the time to count bolts. It contains 40 patters of 7 features each.
3. **Housing** dataset, described in [75].
4. **Laser** dataset, which is related to laser experiments.
5. **NT** dataset [76], which is related to body temperature measurements.
6. **Quake** dataset, used to estimate the strength of an earthquake.
7. **FA** dataset, which contains a percentage of body fat and ten body circumference measurements. The goal is to fit body fat to the other measurements.
8. **PY** dataset [77], used to learn quantitative structure–activity relationships (QSARs).

The numbers of features and patterns for every dataset used in the experiments are listed in Table 3.

Table 3. Features and patterns for every experimental dataset.

Dataset	Features	Patterns
Balance	4	625
BK	4	96
BL	7	41
Dermatology	34	359
Glass	9	214
Hayes Roth	5	132
Heart	13	270
Housing	13	506
Ionosphere	34	351
Laser	4	993
NT	2	131
Parkinson's	22	195
Pima	8	768
PopFailures	18	540
PY	27	74
Quake	3	2178
FA	18	252
Sprial	2	2000
Wine	13	179
Wdbc	30	569
Z_F_S	21	300
Z_O_N_F_S	21	500
ZO_NF_S	21	500

3.2. Experimental Results

Table 4 represents the comparative results for the classification datasets and Table 5 shows the results for the regression problems. For the case of classification problems, the average classification error is reported, while for the regression problem, the average per

point error is reported. The proposed method is denoted as FC MLP and it is compared against four other approaches from the relevant literature:

1. The minimum redundancy maximum relevance feature selection method [78,79] with two selected features. This approach is denoted as MRMR in the experimental tables. The features selected by MRMR are evaluated using an artificial neural network trained by a genetic algorithm with N_c chromosomes.
2. The principal component analysis (PCA) method as implemented in the Mlpack software [80]. The PCA method is used to construct two features from the original dataset. Subsequently, these features are evaluated using an artificial neural network trained by a genetic algorithm with N_c chromosomes.
3. A genetic algorithm with N_c chromosomes and the parameters of Table 2 used to train a neural network with H hidden nodes. This approach is denoted as MLP GEN in the experimental tables.
4. A particle swarm optimization (PSO) with N_c particles and a N_g number of generations used to train a neural network with H hidden nodes. This method is denoted as MLP PSO in the experimental tables.

Table 4. Experimental results for classification datasets.

Dataset	MRMR	PCA	MLP GEN	MLP PSO	FC MLP
Balance	56.80%	56.48%	8.23%	8.07%	0.30%
Dermatology	68.54%	62.11%	10.01%	17.57%	4.98%
Glass	58.35%	50.16%	58.03%	57.35%	45.84%
Hayes Roth	61.21%	61.13%	35.26%	36.69%	23.26%
Heart	38.04%	35.84%	25.46%	25.67%	17.71%
Ionosphere	12.93%	21.22%	13.67%	15.14%	8.42%
Parkinson's	17.16%	16.96%	17.47%	18.35%	10.10%
Pima	26.29%	39.43%	32.98%	30.45%	23.76%
PopFailures	7.04%	31.42%	7.66%	6.24%	4.66%
Spiral	44.87%	45.94%	45.71%	42.10%	26.53%
Wine	30.73%	30.39%	20.82%	19.31%	7.31%
Wdbc	12.91%	10.28%	6.32%	6.95%	3.47%
Z_F_S	32.71%	44.81%	9.42%	10.38%	5.52%
Z_O_N_F_S	43.04%	56.45%	60.38%	63.56%	31.20%
ZO_NF_S	33.79%	40.02%	8.06%	8.84%	4.00%

Table 5. Experiments for regression datasets.

Dataset	MRMR	PCA	MLP GEN	MLP PSO	FC MLP
BK	0.03	0.17	0.21	0.15	0.03
BL	0.15	0.19	0.84	2.15	0.005
Housing	67.97	319.08	30.05	33.43	10.77
Laser	0.031	0.145	0.003	0.038	0.002
NT	1.79	0.69	1.11	0.03	0.01
Quake	0.06	0.59	0.07	0.28	0.03
FA	0.02	0.08	0.04	0.08	0.01
PY	1.56	0.30	0.21	0.07	0.02

Furthermore, the average classification errors for some of the classification datasets are graphically illustrated in Figure 2. An additional experiment was performed, wherein the number of chromosomes for the genetic algorithm of feature construction (Section 2.2) varied from 50 to 500. This experiment was performed on four datasets and the results are presented in Table 6. This table shows the reliability and durability of the proposed method, and partially because of a low number of chromosomes, it achieves quite good generalization results.

As the experimental results clearly show, the proposed method is significantly superior to the other techniques and in many cases the percentage gain reaches 90%. The proposed technique for each dataset created two artificial features with non-linear combinations of the original features. This process is based on grammatical evolution. Because the previous procedure is extremely time consuming, it was chosen to evaluate the characteristics to train a radial basisnetwork which has a fast training time. Then, another genetic algorithm is used to train an artificial neural network on the new features. The overall process is the same regardless of the type of data and this means that the method can be applied to a wide range of datasets. However, because the method requires the presence of two phases using genetic algorithms, it is considered a very slow method compared to other techniques in the literature. Execution times, however, could be drastically reduced by using parallel techniques such as the OpenMP technique used during the experiments. Furthermore, as was clear from the additional experiments performed with the number of chromosomes, this method is quite robust, even for a small number of chromosomes.

Table 6. Experiments with the number of chromosomes for the algorithm of Section 2.2.

Dataset	$N_g = 50$	$N_g = 100$	$N_g = 200$	$N_g = 500$
Heart	21.26%	21.34%	18.73%	17.71%
BK	0.02	0.02	0.02	0.03
BL	0.02	0.02	0.01	0.005
FA	0.01	0.01	0.01	0.01

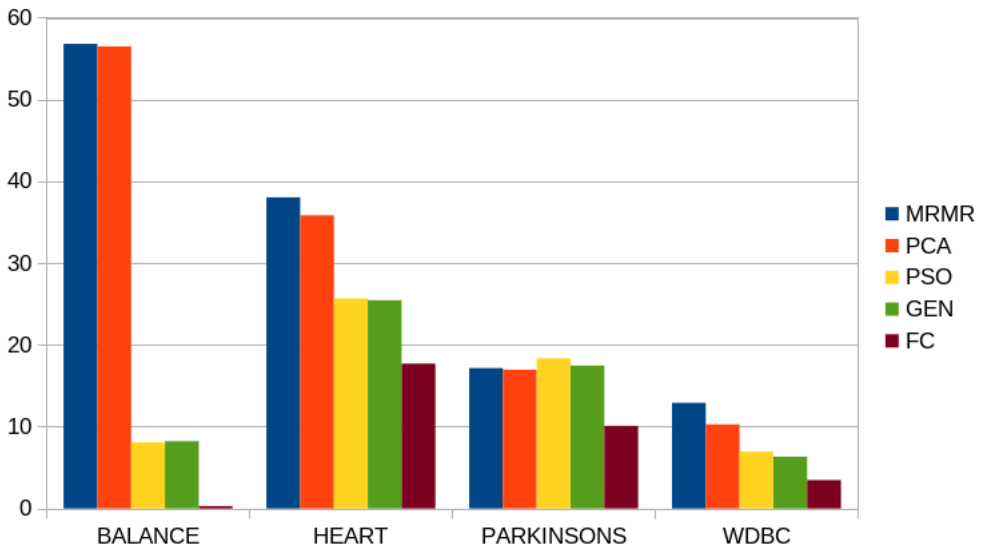

Figure 2. Graphic representation of some of the classification datasets.

4. Conclusions

In the present work, a hybrid feature construction technique was presented with two phases: (a) feature construction and (b) feature evaluation. In the first phase, new features were created as non-linear combinations of old features using grammatical evolution and

radial basis networks. In the second phase, the original dataset was transformed based on the new features and an artificial neural network with a genetic algorithm was trained to learn the new dataset. The genetic algorithm used tried to train the artificial neural network in such a way that it did not lose its generalizing abilities. The proposed technique was applied to a number of datasets from the relevant literature and the results were more than satisfactory. Furthermore, with a series of additional experiments, the stability of the proposed methodology was shown, since it produces satisfactory results even with a small number of chromosomes. However, the proposed technique is much slower than other processes as it requires two computational phases to reach a conclusion. However, with the use of parallel techniques, acceleration can be achieved. The method can be made more efficient in a number of ways. For example, this can be achieved by using parallel genetic algorithms; smarter evaluators to construct features instead of radial basis networks such as SVM; and more sophisticated termination techniques for genetic algorithms to achieve the acceleration of the export of the results.

Funding: Not applicable.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Bishop, C. *Neural Networks for Pattern Recognition*; Oxford University Press: Oxford, UK, 1995.
2. Cybenko, G. Approximation by superpositions of a sigmoidal function. *Math. Control Signals Syst.* **1989**, *2*, 303–314. [CrossRef]
3. Baldi, P.; Cranmer, K.; Faucett, T.; Sadowski, P.; Whiteson, D. Parameterized neural networks for high-energy physics. *Eur. Phys. J. C* **2016**, *76*, 1–7. [CrossRef]
4. Valdas, J.J.; Bonham-Carter, G. Time dependent neural network models for detecting changes of state in complex processes: Applications in earth sciences and astronomy. *Neural Netw.* **2006**, *19*, 196–207. [CrossRef]
5. Carleo, G.; Troyer, M. Solving the quantum many-body problem with artificial neural networks. *Science* **2017**, *355*, 602–606. [CrossRef] [PubMed]
6. Shen, L.; Wu, J.; Yang, W. Multiscale Quantum Mechanics/Molecular Mechanics Simulations with Neural Networks. *J. Chem. Theory Comput.* **2016**, *12*, 4934–4946. [CrossRef]
7. Manzhos, S.; Dawes, R.; Carrington, T. Neural network-based approaches for building high dimensional and quantum dynamics-friendly potential energy surfaces. *Int. J. Quantum Chem.* **2015**, *115*, 1012–1020. [CrossRef]
8. Wei, J.N.; Duvenaud, D.; Aspuru-Guzik, A. Neural Networks for the Prediction of Organic Chemistry Reactions. *ACS Cent. Sci.* **2016**, *2*, 725–732. [CrossRef]
9. Falat, L.; Pancikova, L. Quantitative Modelling in Economics with Advanced Artificial Neural Networks. *Procedia Econ. Financ.* **2015**, *34*, 194–201. [CrossRef]
10. Namazi, M.; Shokrolahi, A.; Maharluie, M.S. Detecting and ranking cash flow risk factors via artificial neural networks technique. *J. Bus. Res.* **2016**, *69*, 1801–1806. [CrossRef]
11. Tkacz, G. Neural network forecasting of Canadian GDP growth. *Int. J. Forecast.* **2001**, *17*, 57–69. [CrossRef]
12. Baskin, I.I.; Winkler, D.; Tetko, I.V. A renaissance of neural networks in drug discovery. *Expert Opin. Drug Discov.* **2016**, *11*, 785–795. [CrossRef] [PubMed]
13. Bartzatt, R. Prediction of Novel Anti-Ebola Virus Compounds Utilizing Artificial Neural Network (ANN). *Chem. Fac.* **2018**, *49*, 16–34.
14. Tsoulos, I.G.; Gavrilis, D.; Glavas, E. Neural network construction and training using grammatical evolution. *Neurocomputing* **2008**, *72*, 269–277. [CrossRef]
15. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]
16. Chen, T.; Zhong, S. Privacy-Preserving Backpropagation Neural Network Learning. *IEEE Trans. Neural Netw.* **2009**, *20*, 1554–1564. [CrossRef] [PubMed]
17. Riedmiller, M.; Braun, H. A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP algorithm. In Proceedings of the IEEE International Conference on Neural Networks, San Francisco, CA, USA, 28 March–1 April 1993; pp. 586–591.
18. Pajchrowski, T.; Zawirski, K.; Nowopolski, K. Neural Speed Controller Trained Online by Means of Modified RPROP Algorithm. *IEEE Trans. Ind. Inform.* **2015**, *11*, 560–568. [CrossRef]

19. Hermanto, R.P.S.; Nugroho, A. Waiting-Time Estimation in Bank Customer Queues using RPROP Neural Networks. *Procedia Comput. Sci.* **2018**, *135*, 35–42. [CrossRef]
20. Robitaille, B.; Marcos, B.; Veillette, M.; Payre, G. Modified quasi-Newton methods for training neural networks. *Comput. Chem. Eng.* **1996**, *20*, 1133–1140. [CrossRef]
21. Liu, Q.; Liu, J.; Sang, R.; Li, J.; Zhang, T.; Zhang, Q. Fast Neural Network Training on FPGA Using Quasi–Newton Optimization Method. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2018**, *26*, 1575–1579. [CrossRef]
22. Zhang, C.; Shao, H.; Li, Y. Particle swarm optimisation for evolving artificial neural network. In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Melbourne, Australia, 17–20 October 2000; pp. 2487–2490.
23. Yu, J.; Wang, S.; Xi, L. Evolving artificial neural networks using an improved PSO and DPSO. *Neurocomputing* **2008**, *71*, 1054–1060. [CrossRef]
24. Verleysen, M.; Francois, D.; Simon, G.; Wertz, V. On the effects of dimensionality on data analysis with neural networks. In *Artificial Neural Nets Problem Solving Methods*; IWANN 2003; Lecture Notes in Computer Science; Mira, J., Alvarez, J.R., Eds.; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2687.
25. Erkmen, B.; Yıldırım, T. Improving classification performance of sonar targets by applying general regression neural network with PCA. *Expert Syst. Appl.* **2008**, *35*, 472–475. [CrossRef]
26. Zhou, J.; Guo, A.; Celler, B.; Su, S. Fault detection and identification spanning multiple processes by integrating PCA with neural network. *Appl. Soft Comput.* **2014**, *14*, 4–11. [CrossRef]
27. Kumar, G.R.; Nagamani, K.; Babu, G.A. A Framework of Dimensionality Reduction Utilizing PCA for Neural Network Prediction. In *Advances in Data Science and Management*; Lecture Notes on Data Engineering and Communications Technologies; Borah, S., Emilia Balas, V., Polkowski, Z., Eds.; Springer: Singapore, 2020; Volume 37.
28. Geman, S.; Bienenstock, E.; Doursat, R. Neural networks and the bias/variance dilemma. *Neural Comput.* **1992**, *4*, 1–58. [CrossRef]
29. Hawkins, D.M. The Problem of Overfitting. *J. Chem. Inf. Comput. Sci.* **2004**, *44*, 1–12. [CrossRef] [PubMed]
30. Nowlan, S.J.; Hinton, G.E. Simplifying neural networks by soft weight sharing. *Neural Comput.* **1992**, *4*, 473–493. [CrossRef]
31. Hanson, S.J.; Pratt, L.Y. Comparing biases for minimal network construction with back propagation. In *Advances in Neural Information Processing Systems*; Touretzky, D.S., Ed.; Morgan Kaufmann: San Mateo, CA, USA, 1989; Volume 1, pp. 177–185.
32. Mozer, M.C.; Smolensky, P. Skeletonization: A technique for trimming the fat from a network via relevance assesment. In *Advances in Neural Processing Systems*; Touretzky, D.S., Ed.; Morgan Kaufmann: San Mateo, CA, USA, 1989; Volume 1, pp. 107–115.
33. Augasta, M.; Kathirvalavakumar, T. Pruning algorithms of neural networks—A comparative study. *Cent. Eur. Comput. Sci.* **2003**, *3*, 105–115. [CrossRef]
34. Srivastava, N.; Hinton, G.E.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R.R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
35. Prechelt, L. Automatic early stopping using cross validation: Quantifying the criteria. *Neural Netw.* **1998**, *11*, 761–767. [CrossRef]
36. Wu, X.; Liu, J. A New Early Stopping Algorithm for Improving Neural Network Generalization. In Proceedings of the 2009 Second International Conference on Intelligent Computation Technology and Automation, Changsha, China, 10–11 October 2009; pp. 15–18.
37. Treadgold, N.K.; Gedeon, T.D. Simulated annealing and weight decay in adaptive learning: The SARPROP algorithm. *Ieee Transactions Neural Netw.* **1998**, *9*, 662–668. [CrossRef]
38. Carvalho, M.; Ludermir, T.B. Particle Swarm Optimization of Feed-Forward Neural Networks with Weight Decay. In Proceedings of the 2006 Sixth International Conference on Hybrid Intelligent Systems (HIS'06), Rio de Janeiro, Brazil, 13 December 2006; p. 5.
39. Neill, M.O.; Ryan, C. Grammatical evolution. *IEEE Trans. Evol. Comput.* **2001**, *5*, 349–358. [CrossRef]
40. Gavrilis, D.; Tsoulos, I.G. Evangelos Dermatas, Selecting and constructing features using grammatical evolution. *Pattern Recognition Lett.* **2008**, *29*, 1358–1365. [CrossRef]
41. Gavrilis, D.; Tsoulos, I.G. Evangelos Dermatas, Neural Recognition and Genetic Features Selection for Robust Detection of E-Mail Spam. In *Advances in Artificial Intelligence Volume 3955 of the Series Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 498–501.
42. Georgoulas, G.; Gavrilis, D.; Tsoulos, I.G.; Stylios, C.; Bernardes, J.; Groumpos, P.P. Novel approach for fetal heart rate classification introducing grammatical evolution. *Biomed. Signal Process. Control.* **2007**, *2*, 69–79. [CrossRef]
43. Smart, O.; Tsoulos, I.G.; Gavrilis, D.; Georgoulas, G. Grammatical evolution for features of epileptic oscillations in clinical intracranial electroencephalograms. *Expert Syst. Appl.* **2011**, *38*, 9991–9999. [CrossRef] [PubMed]
44. Goldberg, D. *Genetic Algorithms in Search, Optimization and Machine Learning*; Addison-Wesley Publishing Company: Reading, MA, USA, 1989.
45. Michaelewicz, Z. *Genetic Algorithms + Data Structures = Evolution Programs*; Springer: Berlin, Germany, 1996.
46. Doorly, D.J.; Peiro, J. *Supervised Parallel Genetic Algorithms in Aerodynamic Optimisation, Artificial Neural Nets and Genetic Algorithms*; Springer: Vienna, Austria, 1997; pp. 229–233.
47. Sarma, K.C. Hojjat Adeli, Bilevel Parallel Genetic Algorithms for Optimization of Large Steel Structures. *Comput.-Aided Civil Infrastruct. Eng.* **2001**, *16*, 295–304. [CrossRef]
48. Fan, Y.; Jiang, T.; Evans, D.J. Volumetric segmentation of brain images using parallel genetic algorithms. *IEEE Trans. Med. Imaging* **2002**, *21*, 904–909.

49. Leung, F.H.F.; Lam, H.K.; Ling, S.H.; Tam, P.S. Tuning of the structure and parameters of a neural network using an improved genetic algorithm. *IEEE Trans. Neural Netw.* **2003**, *14*, 79–88. [CrossRef]
50. Sedki, A.; Ouazar, D.; Mazoudi, E.E. Evolving neural network using real coded genetic algorithm for daily rainfall–runoff forecasting. *Expert Syst. Appl.* **2009**, *36*, 4523–4527. [CrossRef]
51. Majdi, A.; Beiki, M. Evolving neural network using a genetic algorithm for predicting the deformation modulus of rock masses. *Int. J. Rock Mech. Min. Sci.* **2010**, *47*, 246–253. [CrossRef]
52. Smith, M.G.; Bull, L. Genetic Programming with a Genetic Algorithm for Feature Construction and Selection. *Genet. Program Evolvable Mach* **2005**, *6*, 265–281. [CrossRef]
53. Neshatian, K.; Zhang, M.; Andreae, P. A Filter Approach to Multiple Feature Construction for Symbolic Learning Classifiers Using Genetic Programming. *IEEE Trans. Evol. Comput.* **2012**, *16*, 645–661. [CrossRef]
54. Li, X.; Yin, M. Multiobjective Binary Biogeography Based Optimization for Feature Selection Using Gene Expression Data. *IEEE Trans. Nanobiosci.* **2013**, *12*, 343–353. [CrossRef] [PubMed]
55. Ma, J.; Teng, G. A hybrid multiple feature construction approach for classification using Genetic Programming. *Appl. Soft Comput.* **2019**, *80*, 687–699. [CrossRef]
56. Cover, T.M. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Trans. Electron. Comput. EC* **1965**, *14*, 326–334. [CrossRef]
57. Park, J.; Sandberg, I.W. Universal Approximation Using Radial-Basis-Function Networks. *Neural Comput.* **1991**, *3*, 246–257. [CrossRef]
58. Backus, J.W. The Syntax and Semantics of the Proposed International Algebraic Language of the Zurich ACM-GAMM Conference. In Proceedings of the International Conference on Information Processing, UNESCO, Paris, France, 15–20 June 1959; pp. 125–132.
59. Powell, M.J.D. A Tolerant Algorithm for Linearly Constrained Optimization Calculations. *Math. Program.* **1989**, *45*, 547. [CrossRef]
60. Fletcher, R. A new approach to variable metric algorithms. *Comput. J.* **1970**, *13*, 317–322. [CrossRef]
61. Chandra, R.; Dagum, L.; Kohr, D.; Maydan, D.; Menon, J.M.R. *Parallel Programming in OpenMP*; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2001.
62. Shultz, T.; Mareschal, D.; Schmidt, W. Modeling Cognitive Development on Balance Scale Phenomena. *Mach. Learn.* **1994**, *16*, 59–88. [CrossRef]
63. Demiroz, G.; Govenir, H.A.; Ilter, N. Learning Differential Diagnosis of Eryhemato-Squamous Diseases using Voting Feature Intervals. *Artif. Intell. Med.* **1998**, *13*, 147–165.
64. Hayes-Roth, B.; Hayes-Roth, B.F. Concept learning and the recognition and classification of exemplars. *J. Verbal Learn. Verbal Behav.* **1977**, *16*, 321–338. [CrossRef]
65. Kononenko, I.; Šimec, E.; Robnik-Šikonja, M. Overcoming the Myopia of Inductive Learning Algorithms with RELIEFF. *Appl. Intell.* **1997**, *7*, 39–55. [CrossRef]
66. Dy, J.G.; Brodley, C.E. Feature Selection for Unsupervised Learning. *J. Mach. Learn. Res.* **2004**, *5*, 845–889.
67. Perantonis, S.J.; Virvilis, V. Input Feature Extraction for Multilayered Perceptrons Using Supervised Principal Component Analysis. *Neural Process. Lett.* **1999**, *10*, 243–252. [CrossRef]
68. Little, M.A.; McSharry, P.E.; Hunter, E.J.; Ramig, L.O. Suitability of dysphonia measurements for telemonitoring of Parkinson's disease. *IEEE Trans. Biomed.* **2009**, *56*, 1015–1022. [CrossRef] [PubMed]
69. Lucas, D.D.; Klein, R.; Tannahill, J.; Ivanova, D.; Brandon, S.; Domyancic, D.; Zhang, Y. Failure analysis of parameter-induced simulation crashes in climate models. *Geosci. Model Dev.* **2013**, *6*, 1157–1171. [CrossRef]
70. Raymer, M.; Doom, T.E.; Kuhn, L.A.; Punch, W.F. Knowledge discovery in medical and biological datasets using a hybrid Bayes classifier/evolutionary algorithm. IEEE transactions on systems, man, and cybernetics. *Part B Cybern. Publ. IEEE Syst. Man Cybern. Soc.* **2003**, *33*, 802–813. [CrossRef] [PubMed]
71. Zhong, P.; Fukushima, M. Regularized nonsmooth Newton method for multi-class support vector machines. *Optim. Methods Furth. Softw.* **2007**, *22*, 225–236. [CrossRef]
72. Andrzejak, R.G.; Lehnertz, K.; Mormann, F.; Rieke, C.; David, P.; Elger, C.E. 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* **2001**, *64*, 061907. [CrossRef]
73. Tzallas, A.T.; Tsipouras, M.G.; Fotiadis, D.I. Automatic Seizure Detection Based on Time-Frequency Analysis and Artificial Neural Networks. *Comput. Neurosci.* **2007**, *2007*, 80510. [CrossRef]
74. Simonoff, J.S. *Smooting Methods in Statistics*; Springer: New York, NY, USA, 1996.
75. Harrison, D.; Rubinfeld, D.L. Hedonic prices and the demand for clean ai. *J. Environ. Econ. Manag.* **1978**, *5*, 81–102. [CrossRef]
76. Mackowiak, P.A.; Wasserman, S.S.; Levine, M.M. 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.* **1992**, *268*, 1578–1580 [CrossRef]
77. King, R.D.; Muggleton, S.; Lewis, R.; Sternberg, M.J.E. Drug design by machine learning: The use of inductive logic programming to model the structure-activity relationships of trimethoprim analogues binding to dihydrofolate reductase. *Proc. Nat. Acad. Sci. USA* **1992**, *89*, 11322–11326. [CrossRef] [PubMed]
78. Peng, H.; Long, F.; Ding, C. Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Trans. Pattern Anal. Mach. Intell.* **2005**, *27*, 1226–1238. [CrossRef]

79. Ding, C.; Peng, H. Minimum redundancy feature selection from microarray gene expression data. *J. Bioinform. Comput. Biol.* **2005**, *3*, 185–205. [CrossRef] [PubMed]
80. Curtin, R.R.; Cline, J.R.; Slagle, N.P.; March, W.B.; Ram, P.; Mehta, N.A.; Gray, A.G. MLPACK: A Scalable C++ Machine Learning Library. *J. Mach. Learn.* **2013**, *14*, 801–805.