Extending MLP ANN hyper-parameters Optimization by using Genetic Algorithm

Fernando Itano
Dept. Electronic Systems Engineering
University of São Paulo
São Paulo, Brazil
itanofe@lsi.usp.br

Miguel Angelo de Abreu de Sousa
Federal Institute of Education, Science and Technology of São Paulo
Sao Paulo, Brazil
angelo@ifsp.edu.br

Emilio Del-Moral-Hernandez
Dept. Electronic Systems Engineering
University of São Paulo
São Paulo, Brazil
emilio@lsi.usp.br

***Abstract*—Optimizing the hyper-parameters of a multi-layer perceptron (MLP) artificial neural network (ANN) is not a trivial task, and even today the trial-and-error approach is widely used. Many works have already presented using the genetic algorithm (GA) to help in this optimization search including MLP topology, weights, and bias optimization. This work proposes adding hyper-parameters for weights initialization and regularization to be optimized simultaneously with the usually MLP topology and learning hyper-parameters. It also analyses which hyper-parameters are more correlated with classification performance, allowing a reduction in the search space, which decreases the time and computation needed to reach a good set of hyper-parameters. Results achieved with public datasets reveal an increase in performance when compared with similar works. Also, the hyper-parameters related to weights initialization and regularization are among the top 5 most relevant hyper-parameters to explain the accuracy performance in all datasets, showing the importance of including them in the optimization process.**

Keywords—artificial neural network, multi-layer perceptron, MLP, genetic algorithm, GA, hyper-parameters

I. Introduction

Once each problem has specificities about its data, to choose the optimal hyper-parameters of an MLP usually involves a trial-and-error approach, which consumes time, computational resources and requires the researcher to have great experience to properly tune the MLP. It is thus highly desirable to have a method to automatically search for the optimal hyper-parameters efficiently. By hyper-parameters we mean those responsible for defining the topology, learning, weights initialization and regularization options of an MLP.

GA has been widely used as an alternative to the classical Back-propagation (BP) algorithm [1] to tune the set of weight values of MLP with fixed neural topology such as in [2] [3]. Some works studied the use of GA to find only the MLP topology, i.e., the number of hidden layers and the number of neurons in each layer, as in [4]. Others use GA to search for the optimal values of MLP weights together with its topology, instead of using the classical BP, as in [5]. GA has also been used to tune MLP weights and topology such as in [6] [7], and to compose a hybrid training strategy with BP [8]. This hyper-parameter optimizations certainly increases the performance of the MLP. However, there are other essential hyper-parameters, such as weights initialization and regularization that also need to be tuned because they can improve the MLP performance.

The weights initialization hyper-parameters used in this work control the statistical distribution and the scale of initial weights. Poorly initialized weights may prevent to achieve a good performance, either leading to a slower training and requiring more epochs to train or to a faster training but with an increased risk of being trapped in a local minimum [9]. On the other hand, an optimized weight initialization will allow the MLP back-propagation to efficiently decrease the error through the epochs, reaching better performance.

The regularization hyper-parameters are especially essential to improve the generalization of a network with limited sample size and a large number of parameters [10]. With a large number of parameters, the MLP can memorize the training instances exactly and achieve a supposed error-free perfect fit (Fig. 1), compromising the capability of the network to generalize the acquired knowledge on prediction for the examples not used in training.

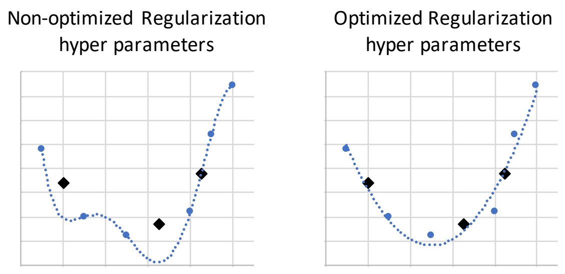

Fig. 1. The circle dots represent the training set and the diamond dots the test set. The non-optimized regularization hyper-parameters can result in the overfit of the MLP, i.e. with loss of generalization capability and higher error on the test set (fitting curve on left chart). The regularization method is especially important with limited sample size and a large number of parameters, leading to a better generalization and lower error on the test set (fitting curve on right chart). The symbols are in the same position on both charts.

To improve the classification performance, this work proposes adding the weights initialization and the regularization hyper-parameters to be optimized simultaneously with the MLP topology and learning hyper-parameters by using a GA. The proposed method named MLPGA+4 because of the 4 hyper-parameters categories to be optimized simultaneously.

978-1-5090-6014-6/18/$31.00 ©2018 IEEE

Moreover, the relationship between these added hyper-parameters and the classification performance will be analyzed to understand the effects of these hyper-parameters on the classification performance. It will allow identifying hyper-parameters space regions where best classification performance is achieved. With that, it will also be possible to restrict the search space and to develop a more efficient GA, which requires less time and computational resources to find a good set of hyper-parameters.

The remainder of this paper is organized as follows: Section II briefly presents some general concepts about MLP networks and GA. Section III details the methodology developed, followed by the experimental results in Section IV. Section V presents the conclusions and future works.

II. General Concepts and Related Works

This section presents some general concepts of MLP and the hyper-parameters we propose adding to the optimization process. General concepts of GA will be presented as well as the modifications made to optimize categorical, integer and real value hyper-parameters simultaneously. The Related Works section presents other methods of MLP hyper-parameter optimization.

A. *Multilayer Perceptron hyper-parameter effects*

MLP is one of the most widely used architectures of MLP due to its versatility in classification and regression problems and its universal function approximation characteristic [1] [11] [12] [13]. An example of one MLP is presented in Fig. 2. The small circles represent the input and output neurons in their respective layers, and the large circles represent the neurons present in the hidden layers. This structure is also called topology of the MLP. The connections between the neurons, also known as synaptic weights, are represented by arrows and contains the knowledge of the MLP. This knowledge is acquired during the training phase, usually by using the BP algorithm, when the MLP learns from examples. Below we present the conceptual effects of the MLP hyper-parameters used in this work.

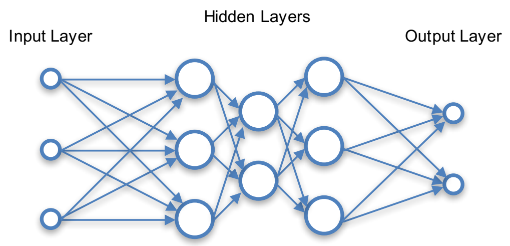

Fig. 2. Example of an MLP with 3 inputs, 3 hidden layers; the first with 3 neurons, the second with 2 neurons and the third with 3 neurons and an output layer with 2 neurons.

1) *Topology hyper-parameters*

The increase in the number of neurons, allocated in one or more hidden layers, allows an MLP mapping more complex relationships between input and output. However, a great number of neurons increases the time and computation needed to train the network and raises the probability of overfitting. On the other hand, fewer neurons and fewer hidden layers limit the MLP to map only simpler relationships.

Even today, there is no established rule to define the number of neurons and hidden layers, so the trial-and-error approach is widely used.

2) *Learning Rate hyper-parameters*

There are several optimization methods to properly adjust the synaptic weights. The Gradient Descent was the first one and it was necessary to manual tune the learning rate, a critical hyper-parameter to obtain high performance. More recently some methods with dynamic adaptation of the learning rate were proposed, such as ADADELTA [14], used in this work. This method dynamically changes the learning rate to optimize the synaptic weights faster and without being trapped in local minima. It requires two hyper-parameters to be set: 1) ρ represents a decay constant, similar to that used in the momentum method, 2) ε is a constant to avoid the division by zero. Also, it appears to be robust to different topologies, datasets and the other hyper-parameters selection [14].

3) *Synaptic Weights Initialization hyper-parameters*

The speed and convergence of the learning process of an MLP, as in many multidimensional optimization problems, is strongly influenced by the initial condition [15] [16]. In an MLP, these initial states are given by the initial synaptic weights. In this work, we used the hyper-parameters that allow the initial values to be sampled from different probabilistic distributions, such as Gaussian and Uniform, and different scales, as depicted in Fig. 3. Thus, we allow each data set to obtain the optimal initial condition to increase the classification performance.

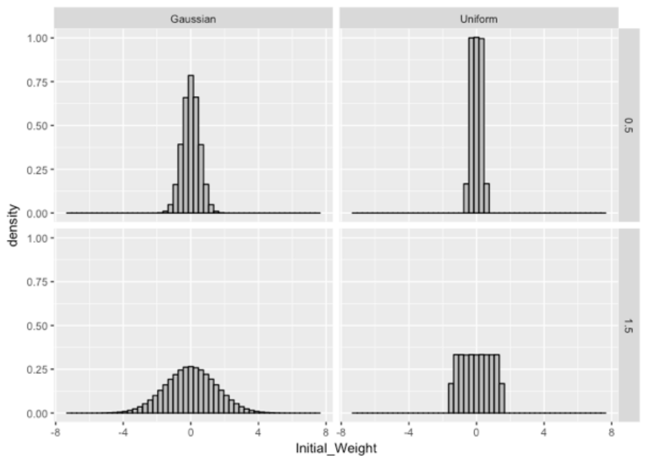

Fig. 3. Initial weight distribution comparison: Gaussian and Uniform with scales of 0.5 and 1.5.

4) *Regularization hyper-parameters*

With the increase of computational power, it is easy to create an MLP with several layers and a large number of neurons in each layer. However, this large number of parameters may lead to overfitting, i.e., the generalization capability loss [10]. The regularization hyper-parameters considered in this work are used to preserve the generalization capability in 2 main ways:

a) *Input dropout ratio*

The input dropout ratio constrains the online optimization so that during forward propagation, for a given training example, each neuron in the network suppresses its activation with probability P [17].

b) l1 and l2

These hyper-parameters modify the loss function so as to minimize loss. l1 constraints the absolute value of the synaptic weights, while l2 constrains the sum of the squared weights; both may cause synaptic weights to become zero, also known as pruning process. If l1 and l2 hyper-parameters are well adjusted, they reduce the number of synaptic weights, appropriately reducing the overfit effect. On the other hand, if l1 and l2 hyper-parameters are poorly optimized, too many parameters may become zero, oversimplifying the acquired knowledge, or too little parameters may become zero, increasing the overfit effect.

B. *Genetic Algorithm optimizing an MLP*

GA is one of the evolutionary computation search algorithms based on the natural evolutionary theory [18]. It states that individuals who are best adapted to their environment are more likely to survive and to reproduce. The next generation represented by their offspring will inherit a mix of the parents' characteristics, and generate individuals who are enhanced and worse. The enhanced ones will be even more likely to survive and to reproduce whereas the worse ones will disappear. After several generations in this process, the population is expected to evolve and find an individual whose characteristics allow it to be the best-adapted individual.

The main characteristics of GA include (1) robustness to discontinuities of the fitness function because GA does not require the fitness function to have a derivative, (2) robustness to local minima due to its global search characteristic and (3) directed search which does not require exploring the complete solutions space. The following are the details on using GA for MLP hyper-parameters optimization.

1) *Fitness Function*

A fitness function calculates a measure that allows assessing how adapted an individual is to the environment. This metric is used to direct the search for the characteristics that will result in a better-adapted individual, i.e., with better performance in a task. In this paper context, we defined the fitness function as the accuracy rate in classification tasks.

2) *Chromosomic Representation*

The binary chromosomic representation [18] is the most widely used representation and consists in encoding the information in genes that can assume zero and one values. To avoid one of its caveats, like the Hamming Abyss, this work used a mixed representation in which each gene can assume integer values for categorical and discrete characteristics and real values for continuous characteristics.

3) *Initial Population*

The initial population contains all the individuals that will be assessed by the fitness function and submitted to the genetic operators. Usually, it is generated by random sampling, but in this work, additional care was taken to ensure that there are no replicate individuals at the beginning of the optimization process.

4) *Genetic Operators*

a) Selection

The selection operator chooses 2 individuals from the population based on its fitness function value to submit to cross-over operator. We here used the tournament selection with 2 individuals. These 2 individuals are randomly sampled from the population and their fitness function values are compared, the one with the highest value is then selected. This process is repeated for another random sample of 2 individuals and the winner is also selected. These 2 selected winner individuals are then submitted to the cross-over operator.

b) Cross-over

This operator exchanges the chromosomic information between 2 individuals to generate offspring with mixed characteristics. We here used the uniform cross-over, which consists in sampling a random variable from the Bernoulli distribution [19] with p = 0.5 for each gene. Then defining which parent will send its genes to the offspring based on this sample. For example, the first offspring will inherit the genes from parent #1 if the random variable value is "0" and inherit the genes from parent #2 if the random variable value is "1". The second offspring will inherit the opposing genes. An example is depicted in Fig. 4.

The probability of the cross-over to occur was defined as 80%; otherwise, the selected individuals proceed to the next operator without modifications.

Once the chromosomic representation adopted here contains categorical, discrete and continuous genes, instead of the widely used binary representation, the cross-over operator was customized to properly exchange the genetic information, as shown below.

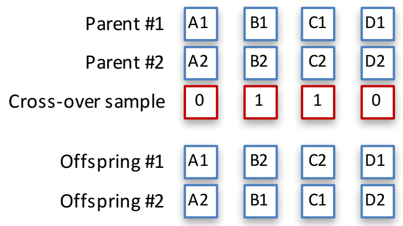

Fig. 4. Example of a uniform cross-over – the parent's genes inheritance are stochastically defined by sampling from a Bernoulli distribution. The genes A, B, C and D, each one representing one characteristic, are exchanged or not between 2 parents. The 2 offspring will inherit a combination of parent's characteristics.

c) Mutation

The mutation operator introduces a stochastic factor to the artificial evolution algorithm and randomly changes the information contained in a gene. Here, the probability of a mutation to occur was defined as 20%. Also, the mutation and the cross-over operators occur independently.

C. *Related Works*

There are some works optimizing the MLP by using GA with different strategies.

The G-Prop method, proposed by [20], uses GA to select the initial weights and optimize the number of neurons in a single hidden layer. This is a hybrid approach where the synaptic weights are initialized by a GA and optimized by BP algorithm. The NNC method, proposed by [21], uses grammatical evolution to encode the network topology and the synaptic weights. The GE-BP method, proposed by [22], also uses grammatical evolution to design the MLP topology but uses BP for training. The NN-SGE method, proposed by [4], uses structured grammatical evolution to optimize both topology and synaptic weights.

The proposed method, MLPGA+4, optimizes not only the topology and the learning rate but initial weights and regularization hyper-parameters also. Consequently, it is expected the resulting MLP is better optimized and have a higher accuracy performance in classification.

III. Method

The classification performance of the proposed method and the effect of these hyper-parameters on accuracy ratio was assessed by using five public datasets.

A. Datasets

Five public datasets from UCI Machine Learning Repository were used with no pre-processing or data augmentation methodologies, i.e., the datasets are used as obtained. The features column shows how many inputs are available and the instances column shows how many observations from each category each dataset contains.

TABLE I. UCI Datasets used in experiments

Dataset	Features	Instances [a]
Breast Cancer Wisconsin Diagnostic (BCWD)	30	357+212=569
Ionosphere (Iono)	34	225+126=351
Connectionist Bench - Sonar, Mines vs. Rocks (Sonar)	60	97+111=208
Heart Disease (Heart)	14	164+139=303
Iris	4	50+50+50=150

a. Number of observations in each category and the total

B. MLP Neural Network hyper-parameters

Table II presents the hyper-parameters to be optimized by GA, the range of permitted values and gene locations. The numbers in parenthesis represent the types of genes: (1) Categorical, (2) Discrete and (3) Continuous. The hyper-parameters this work proposes to include in the optimization process, weights initialization and regularization, are presented in genes 1 to 2 and 12 to 14, respectively.

TABLE II. Hyper-parameters to be optimized by GA, the range of permitted values and gene locations

Hyper-parameter (type of gene)	Range of permitted values	Gene
Initial Weight Distr. (1)	1 – Gaussian 2 – Uniform 3 – Uniform Adaptive	1
Initial Weight Scale (3)	[0.5, 1.5]	2
Nº of hidden layers (2)	[1, 5]	3
Nº of neurons in hidden layer 1 (2)	[1, 16]	4
Nº of neurons in hidden layer 2 (2)	[1, 16]	5
Nº of neurons in hidden layer 3 (2)	[1, 16]	6
Nº of neurons in hidden layer 4 (2)	[1, 16]	7
Nº of neurons in hidden layer 5 (2)	[1, 16]	8
Activation Function (1)	1 – Tanh 2 – Rectifier 3 – Maxout	9
Adaptive learning rate: time decay factor – ALRTD (3)	[0.985, 0.995]	10
Adaptive learning rate: smoothing factor – ALRSM (3)	[1e-9, 1e-7]	11
Input dropout ratio (3)	[0.0, 0.8]	12
l1 (3)	[0.0, 1e-3]	13
l2 (3)	[0.0, 1e-3]	14

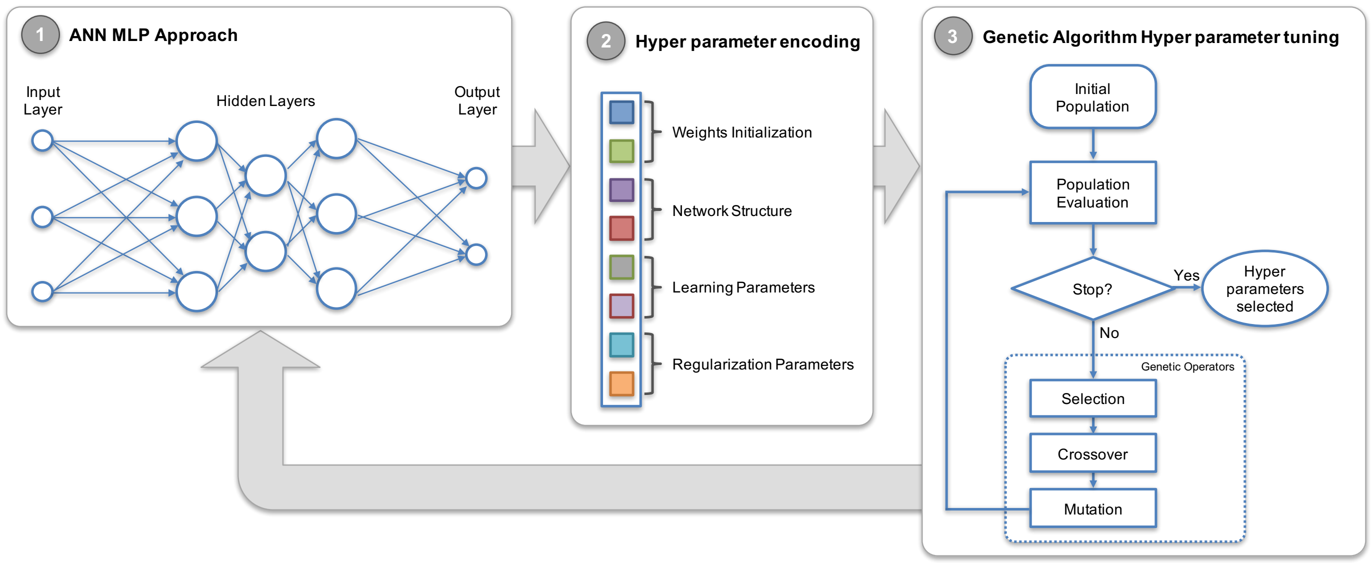

Fig. 5. The flowchart diagram of MLP hyper-parameters being optimized by GA. Here, the weights initialization and regularization hyper-parameters are encoded in a chromosomic vector to be optimized by GA, simultaneously with network structure (topology) and learning hyper-parameters.

The selected method to optimize the synaptic weights and biases is BP with adaptive learning rate and a train limited by 1,000 epochs. Also, a 3-fold cross-validation technique was used to assess the generalization capabilities of the MLP. The 3-fold was chosen due to datasets number of instances limitation, since training with 90% of limited instances may increase the risk of overfitting.

C. GA Setup

The mixed chromosomic representation was adopted to allow the GA to evolve with different types of genes. On creating the initial population, the categorical and discrete genes were sampled with equal probabilities for each category or integer number. The continuous genes assumed a real value from a uniform distribution within the range of permitted values. The details about how the GA operators works are described in section II.B.4.

The fitness function adopted calculates the mean accuracy rate of the validation sets of a 3-fold cross-validation setup. The initial population was defined to contain 100 individuals, and the number of generations is determined as 20. The elitism technique was adopted in 5%. This method, presented in Fig. 5, was applied to 30 independent runs, generating 60,000 trained MLP for each dataset.

To assess which of the hyper-parameters are more correlated with the classification performance and to quantify its effects, a non-linear random forest model [23] was fitted to the data. This model was chosen due to his ability to calculate the variable importance. So, we can easily identify which hyper-parameter has more influence in classification performance. The data used in the model contains 60,000 observations of each dataset, resulting from a population of 100 individuals from 20 generations and 30 independent runs. The MLP hyper-parameters setup were the inputs, and the mean accuracy rate of the 3-fold cross-validation was the target.

The importance of each hyper-parameter and its effects were analyzed graphically and statistically using the Mann-Whitney U test [24], which is a well established nonparametric test used to compare the data distribution of 2 groups. The statistical distribution of the results will be presented in a boxplot graphic [25], which allows a quick comparison between groups, such as median, 1st and 3rd quartiles, and dispersion, and visually identifying patterns. The essays were conducted on a Linux Ubuntu 16.04 operational system with R 3.4.1 statistical computing platform and H2O machine learning library version 3.16.0.2.

IV. EXPERIMENTAL RESULTS

Table III presents the classification performance of the best MLP after 20 generations in each of the 30 independent runs on each dataset with its respective number of neurons using the MLPGA+4, and the results from similar works. The mean accuracy rate and the number of neurons are presented with their respective standard deviation (±).

Considering the BCWD dataset, the proposed method shows a slightly higher accuracy rate in classifying cancer as benign or malign when compared with the G-Prop method, but with a considerably lower standard deviation, evidencing better stability in achieving this performance. Conversely, the topology of the proposed method shows a substantially higher number of neurons, among those who presented this information.

Analyzing the Ionosphere dataset results, MLPGA+4 presented a considerably higher accuracy rate when compared with NNC method. As the previous datasets, the standard deviation of the proposed method remains the smallest and the number of neurons, considerably higher.

In the Sonar dataset, the proposed approach shows an increase of 15.79% on accuracy rate when compared with NN-SGE, and with a considerably lower standard deviation. Yet again with a noticeably higher number of neurons.

Analyzing the Heart dataset, MLPGA+4 presented an increase of 8.30% on accuracy rate when compared with the GE-BP method. Also presented a considerably lower standard deviation.

Finally, considering the Iris dataset, the proposed method shows a slightly higher accuracy rate when compared with the GE-BP method, and a considerably lower standard deviation.

TABLE III. CLASSIFICATION PERFORMANCE AND Nº OF NEURONS

Dataset	Method	Accuracy	Neurons
BCWD	G-Prop	99.00% ± 0.50%	3.20 ± 0.8
	NNC	95.44%	-
	GE-BP	95.90% ± 3.14%	-
	NN-SGE	93.00% ± 2.00%	3.73 ± 1.53
	MLPGA+4	**99.19% ± 0.08%** [a]	11.67 ± 2.15 [b]
Ionosphere	NNC	90.34%	-
	GE-BP	89.90% ± 3.16%	-
	NN-SGE	87.00% ± 10.00%	3.53 ± 1.36
	MLPGA+4	**96.73% ± 0.30%** [a]	36.03 ± 12.08 [b]
Sonar	NN-SGE [1]	78.00% ± 5.00%	4.23 ± 1.33
	MLPGA+4	**93.79% ± 0.41%** [a]	29.47 ± 16.11 [b]
Heart	GE-BP	80.20% ± 5.24%	-
	MLPGA+4	**88.50% ± 0.30%** [a]	26.57 ± 14.78 [b]
Iris	GE-BP	96.60 ± 6.14%	-
	MLPGA+4	**98.87% ± 0.33%** [a]	13.10 ± 11.30 [b]

[a] Mean accuracy rate from 3-fold cross-validation. [b] Average of the nº of neurons in MLP

The best classification performance was achieved at different generations for each dataset, considering the 30 independent runs, as depicted in Fig. 6.

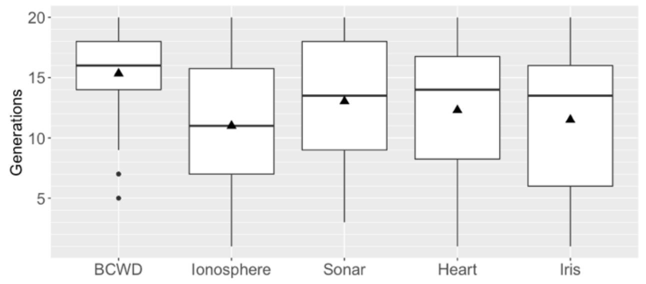

Fig. 6. The boxplots with the distribution of generations needed to achieve the best set of hyper-parameters. The triangles show the average number of generations. The Ionosphere dataset required fewer generations to achieve a good set of hyper-parameters, with an average of 11. On the other hand, BCWD dataset required 15.33 generations, on average.

The Ionosphere dataset required an average of 11 generations to reach the best set of hyper-parameters, the smallest number of generations among those considered in this work. The standard deviation of 5.36 is similar to the other datasets, except for the BCWD dataset, which presented 4.03. The BCWD dataset required an average of 15.33 generations to achieve the best set of hyper-parameters, the largest number of generations among the five datasets.

The results show that although some of the 30 independent runs achieved the best set of hyper-parameters before the 20th generation, the reduction in the number of generations drastically reduces the probability to reach this optimal set. This means that to keep a probability to achieve the best set of hyper-parameters in 90%, it is necessary to evolve at least to the 17th generation in the Ionosphere dataset and 20th generation in the BCWD dataset.

Table IV presents the scaled importance of the top 5 hyper-parameters by each dataset. The hyper-parameters in bold are the ones proposed to be included in the optimization process by this work.

TABLE IV. SCALED IMPORTANCE OF THE TOP 5 HYPER-PARAMETERS BY DATASET

Dataset	Hyper-parameter	Scaled Importance
BCWD	# Neurons Hidden Layer 3	1.0000
	# Hidden Layers	0.5075
	Input Dropout Ratio	**0.4693**
	# Neurons Hidden Layer 4	0.4615
	# Neurons Hidden Layer 2	0.4582
Sonar	**Input Dropout Ratio**	**1.0000**
	Initial Weight Distribution	**0.7384**
	# Hidden Layers	0.6003
	# Neurons Hidden Layer 1	0.5671
	Activation Function	0.3091
Ionosphere	# Neurons Hidden Layer 5	1.0000
	Initial Weight Scale	**0.8349**
	# Hidden Layers	0.7985
	Initial Weight Distribution	**0.5723**
	Activation Function	0.5376
Heart	Activation Function	1.0000
	# Hidden Layers	0.8232
	# Neurons Hidden Layer 5	0.6943
	# Neurons Hidden Layer 1	0.4752
	Initial Weight Scale	**0.4723**
Iris	**Input Dropout Ratio**	**1.0000**
	# Hidden Layers	0.2657
	# Neurons Hidden Layer 3	0.1064
	# Total Neurons	0.0749
	Activation Function	0.0684

Analyzing the hyper-parameters effect in the BCWD dataset, the input dropout ratio hyper-parameter is the third most important factor to explain the mean accuracy rate. The boxplot depicted in Fig. 7 shows that the MLP that used the input dropout ratio hyper-parameter in the interval (0.2, 0.3] presented the highest median of the mean accuracy rate. This optimal range gives an increase of 1.1% in the mean accuracy rate when compared with the worst interval (0.7, 0.8]. Contrasting with its neighbor intervals, (0.1, 0.2] and (0.3, 0.4], the p-values of the Mann-Whitney U test are inferior to 0.1%. This value shows that the optimal interval is statistically superior to its neighbors. The null hypothesis of this statistical test is that the groups have the same distribution. Although 1,1% may appear to be minor, some problems that require high precision may benefit from the optimization of this hyper-parameter.

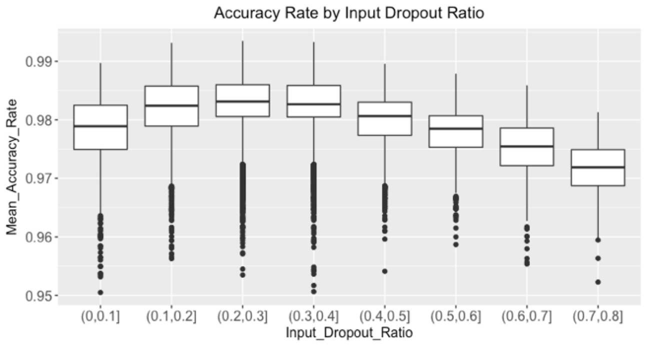

Fig. 7. The boxplot of the mean accuracy rate by input dropout ratio in the BCWD dataset shows an optimal range in the interval (0.2, 0.3], with an increase of 1.1% in the mean accuracy rate when compared with the worst interval (0.7, 0.8].

In the Sonar dataset, the input dropout ratio and the initial weight distribution are the 2 most relevant factors related to the mean accuracy rate. The boxplot presented in Fig. 8 shows that the MLP that used the input dropout ratio hyper-parameter in the interval (0.3, 0.4] presented the highest median of the mean accuracy rate. This optimal range presents an increase of 10.5% in the mean accuracy rate when compared with the worst interval (0.7, 0.8]. The comparison with its neighbor intervals using the Mann-Whitney U test shows the range (0.3, 0.4] is statistically superior, considering a significance level of 5%. Unlike the BCWD dataset, the dispersion of the mean accuracy rate, represented by the size of the boxplot, increases when the input dropout ratio rises.

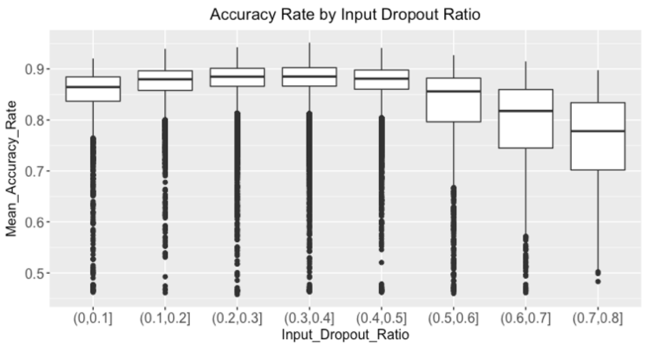

Fig. 8. The boxplot of the mean accuracy rate by input dropout ratio in the Sonar dataset shows an optimal range in the interval (0.3, 0.4], with an increase of 10.5% in the mean accuracy rate when compared with the worst interval (0.7, 0.8].

Analyzing the initial weight distribution in the Sonar dataset, Uniform Adaptive [26] presented the highest median of the mean accuracy rate (Fig. 9). This distribution showed an estimated increase of 3.6% when compared with the Gaussian distribution. Also, the Uniform Adaptive mean accuracy rate standard deviation is 44% smaller than Uniform and 61% smaller than the Gaussian distribution. Considering a significance level of 5%, Uniform Adaptive presented a mean accuracy rate statistically superior to the Gaussian and Uniform distributions.

In the Ionosphere dataset, the initial weight scale and the initial weight distribution are the second and fourth most relevant factors related to the mean accuracy rate. After grouping all the statistically equal intervals, the MLP which used the initial weight scale in the range (0.5, 0.9] presented an estimated increase of 0.2% in the mean accuracy rate when compared with the MLP that used the worst interval (1.3, 1.5] (Fig. 10). Considering a significance level of 5%, the optimal interval, (0.5, 0.9], is statistically superior to the other ranges.

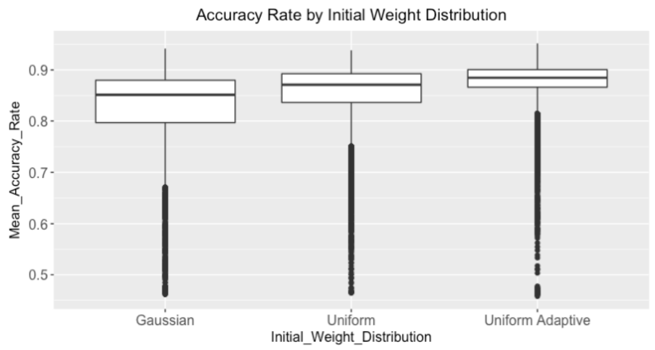

Fig. 9. The boxplot of the mean accuracy rate by initial weight distribution in the Sonar dataset shows Uniform Adaptive as the optimal distribution, with an increase of 3.6% in the accuracy performance when compared with the Gaussian distribution.

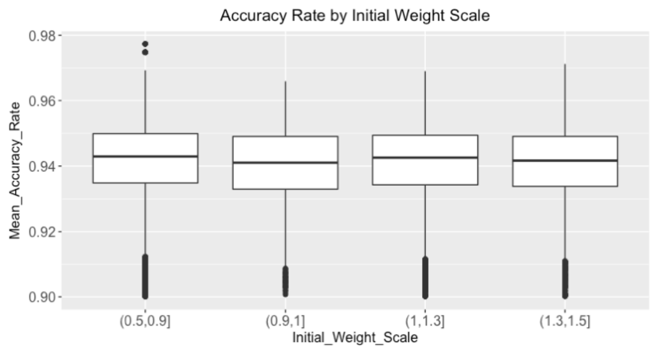

Fig. 10. The boxplot of the mean accuracy rate by initial weight scale intervals in the Ionosphere dataset shows an optimal range in the interval (0.5, 0.9], with an increase of 0.2% in the mean accuracy rate when compared with the worst interval (1.3, 1.5].

Also, in the Ionosphere dataset, the MLP which achieved the higher median of the mean accuracy rate had the Uniform Adaptive as the initial weight distribution (Fig. 11). The optimal statistical distribution presented an estimated increase of 0.5% when compared with the Gaussian distribution. When compared with the other distributions using the Mann-Whitney U test, the p-value was inferior to 0.1%, evidencing Uniform Adaptive is statistically superior to the Gaussian and Uniform distributions.

In the Heart dataset, the input dropout ratio is the 5th most relevant factor related to the mean accuracy rate. After grouping all the statistically equal intervals, the MLP which used the initial weight scale in the range (0.5, 0.9] presented an estimated increase of 0.4% in the mean accuracy rate when compared with the MLP that used the worst interval (1.1, 1.2] (Fig. 12). Considering a significance level of 5%, the optimal interval, (0.5, 0.9], is statistically equal to the interval (1.0, 1.1] and statistically superior to the other ranges.

Finally, in the Iris dataset, the input dropout ratio is the most relevant factor related to the mean accuracy rate. After grouping all the statistically equal intervals, the MLP which used the input dropout ratio in the range (0.0, 0.1] presented an estimated increase of 15.2% in the mean accuracy rate when compared with the MLP that used the worst interval (0.6, 0.8] (Fig. 13). Considering a significance level of 5%, the optimal interval, (0.0, 0.1], is statistically superior to the other ranges. Also, the optimal interval presented the smaller standard deviation of the mean accuracy rate (0.026). It is 4.31 times smaller than the standard deviation of the mean accuracy rate of the worst interval (0.113).

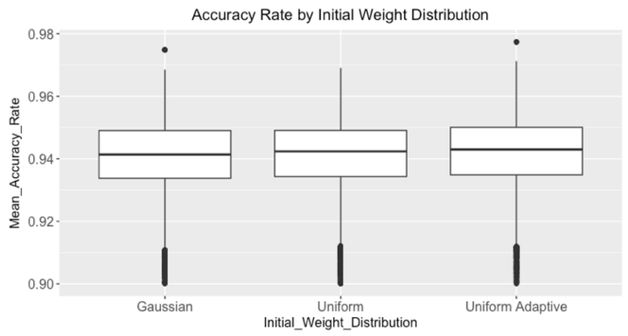

Fig. 11. The boxplot of the mean accuracy rate by initial weight distribution in the Ionosphere dataset shows Uniform Adaptive as the optimal distribution, with an increase of 0.5% in the mean accuracy rate when compared with the Gaussian distribution.

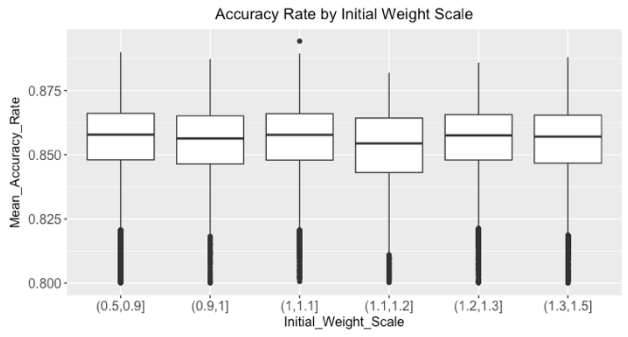

Fig. 12. The boxplot of the mean accuracy rate by initial weight scale intervals in the Heart dataset shows an optimal range in the interval (0.5, 0.9], with an increase of 0.4% in the mean accuracy rate when compared with the worst interval (1.1, 1.2].

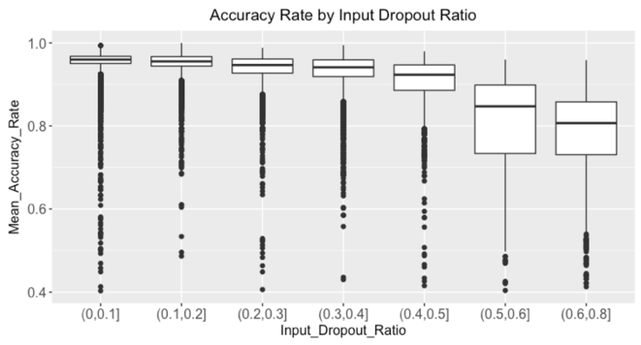

Fig. 13. The boxplot of the mean accuracy rate by input dropout ratio intervals in the Iris dataset shows an optimal range in the interval (0.0, 0.1], with an increase of 15.2% in the mean accuracy rate when compared with the worst interval (0.6, 0.8].

V. Conclusions and Future Works

This work presented the addition of hyper-parameters for weights initialization and regularization to be optimized simultaneously with topology and learning parameters of an MLP. It also proposed analyzing how these hyper-parameters affect the classification performance.

The results from the five datasets show the proposed method allows training an MLP with better performance in classification task when compared with similar works. Moreover, the standard deviation of the mean accuracy rate presented by the proposed method is the smallest, demonstrating the stability of the approach.

In the five datasets, the added hyper-parameters of weights initialization and regularization are found between the top 5 most relevant hyper-parameters to explain the accuracy rate of the MLP on classification tasks. The greatest difference in the mean accuracy rate occurred in Iris dataset with an increase of more than 15% from the worst to the best interval of input dropout ratio. Even with the higher number of neurons in all datasets, the MLP with the proposed method presented the highest accuracy rate in 3-fold cross-validation, showing the importance of the regularization hyper-parameters in controlling overfit.

The initial weight distribution and initial weight scale are found between the top 5 most relevant hyper-parameters in 3 out of 5 datasets. In the Sonar dataset, the initial weight distribution optimization increased the mean accuracy rate by 3,6%. This result shows the importance of this hyper-parameter to be included in the optimization process.

Due to the peculiarities of each problem, each dataset benefited from a different set of hyper-parameters and achieved the best set in different generations. Therefore, a pattern that could be used to make the search by GA more efficient was not found. On the other hand, it shows how important it is to optimize these hyper-parameters in each dataset with a minimum of 20 generations to achieve high performance.

Future extensions to this work include adding the hidden layer dropout hyper-parameter to be optimized with the ones proposed herein and analyzing the performance of the MLPGA+4 in regression tasks. The correlation between the hyper-parameters themselves is to be analyzed to search for a pattern that may possibly be used to reduce the hyper-parameters space of search, thus reducing the time needed to find the optimal set of hyper-parameters.

References

[1] S. Haykin, *Neural Networks: A Comprehensive Foundation (2nd ed.)*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998.

[2] J. Gill, B. Singh, and S. Singh, “Training back propagation neural networks with genetic algorithm for weather forecasting,” in *2010 IEEE 8th International Symposium on Intelligent Systems and Informatics* (SISY), Subotica, 2010, pp. 465-469.

[3] K. Y. Huang, L. C. Shen, K. J. Chen and M. C. Huang, “Multilayer perceptron with genetic algorithm for well log data inversion,” in *2013 IEEE International Geoscience and Remote Sensing Symposium* (IGARSS), Melbourne, 2013, pp. 1544-1547.

[4] F. Assunção, N. Lourenço, P. Machado and B. Ribeiro, "Automatic generation of neural networks with structured grammatical evolution," in *2017 IEEE Congress on Evolutionary Computation* (CEC), San Sebastian, 2017, pp. 1557-1564.

[5] P. P. Palmes, T. Hayasaka, and S. Usui, “Mutation-Based Genetic Neural Network,” *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 587-600, May 2005.

[6] S. Zhang, H. Wang, L. Liu, C. Du and J. Lu, “Optimization of neural network based on genetic algorithm and BP,” in *Proceedings of 2014 International Conference on Cloud Computing and Internet of Things*, Changchun, 2014, pp. 203-207.

[7] J. T. Tsai, J. H. Chou, and T. K. Liu, “Tuning the structure and parameters of a neural network by using hybrid Taguchi-genetic algorithm,” *IEEE Transactions on Neural Networks*, vol. 17, no. 1, pp. 69-80, Jan 2006.

[8] X. Yao, “Evolving artificial neural networks,” *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.

[9] Y. Bengio, “Practical recommendations for gradient-based training of deep architectures,” In G. Montavon, G.B. Orr, KR. Müller (eds) *Neural Networks: Tricks of the Trade*. Lecture Notes in Computer Science, vol 7700. Springer, Berlin, Heidelberg, 2012.

[10] G. N. Karystinos and D. A. Pados, "On overfitting, generalization, and randomly expanded training sets," *IEEE Transactions on Neural Networks*, vol. 11, no. 5, pp. 1050-1057, Sep 2000.

[11] G. Cybenko, Approximation by superpositions of a sigmoidal function, Math. *Control Signals Systems 2* (1989) 303–314.

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

[13] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural Netw.*, vol. 2, pp. 359–366, 1989.

[14] M. D. Zeiler, “ADADELTA: An Adaptive Learning Rate Method,” CoRR abs/1207.0151, 2012.

[15] H. Lari-Najafi, M. Nasiruddin, and T. Samad, "Effect of initial weights on back-propagation and its variations," *Conference Proceedings., IEEE International Conference on Systems, Man and Cybernetics*, Cambridge, MA, pp. 218-219 vol.1, 1989.

[16] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, “On the importance of initialization and momentum in deep learning,” in *Proceedings of the 30th International Conference on Machine Learning*, in PMLR 28(3), pp. 1139-1147, 2013.

[17] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” arXiv preprint arXiv:1207.0580, 2012.

[18] M. Mitchell, *An Introduction to Genetic Algorithm*. MIT Press, Cambridge, 1999.

[19] S. Ross, *A First Course in Probability*. Pearson, London, 2014.

[20] P. A. Castillo, J. J. Merelo, A. Prieto, V. Rivas, G. Romero, “G-Prop: Global optimization of multilayer perceptrons using GAs,” *Neurocomputing* 35, 2000, pp. 149-163.

[21] I. Tsoulos, D. Gavrilis, and E. Glavas, “Neural network construction and training using grammatical evolution,” *Neurocomputing*, vol. 72, no. 1, pp. 269–277, 2008.

[22] K. Soltanian, F. A. Tab, F. A. Zar, I. Tsoulos, “Artificial Neural Networks Generation Using Grammatical Evolution,” 21st Iranian Conference on Electrical Engineering (ICEE), Mashhad, pp. 1-5, 2013.

[23] L. Breiman, “Random Forests,” Machine Learning, vol. 45, pp. 5-32, 2001.

[24] W. J. Conover, *Practical Nonparametric Statistics*, Willey, New York, 1999.

[25] McGill, R., Tukey, J., and Larsen, W, "Variations of Box Plots," *The American Statistician, 32*(1), 12-16, 1978.

[26] A. Candel, V. Parmar, E. LeDell, A. Arora, “Deep Learning with H2O,” in *http://h2o.ai/resources*, 2018.