[Array 14 (2022) 100173](https://doi.org/10.1016/j.array.2022.100173)



Contents lists available at [ScienceDirect](http://www.elsevier.com/locate/array)

Array

journal homepage: [www.elsevier.com/locate/array](http://www.elsevier.com/locate/array)

Optimising Multilayer Perceptron weights and biases through a Cellular Genetic Algorithm for medical data classification

Matías Gabriel Rojas [a](#_bookmark0),[∗](#_bookmark2), Ana Carolina Olivera [a](#_bookmark0),[b](#_bookmark1), Pablo Javier Vidal [a](#_bookmark0),[b](#_bookmark1)

a *Instituto Universitario para las Tecnologías de la Información y las Comunicaciones, Consejo Nacional de Investigaciones Científicas y Técnicas, Universidad Nacional de Cuyo, Padre Jorge Contreras 1300, Mendoza, M5502JMA, Mendoza, Argentina*

b *Facultad de Ingeniería, Universidad Nacional de Cuyo, Centro Universitario, Mendoza, M5502JMA, Mendoza, Argentina*

A R T I C L E I N F O A B S T R A C T

*Keywords:*

Multilayer Perceptron Training methods

Cellular Genetic Algorithm Metaheuristics

Medical data classification

In recent years, technology in medicine has shown a significant advance due to artificial intelligence becoming a framework to make accurate medical diagnoses. Models like Multilayer Perceptrons (MLPs) can detect implicit patterns in data, allowing identifying patients conditions that cannot be seen easily. MLPs consist of biased neurons arranged in layers, connected by weighted connections. Their effectiveness depends on finding the optimal weights and biases that reduce the classification error, which is usually done by using the Back Propagation algorithm (BP). But BP has several disadvantages that could provoke the MLP not to learn. Metaheuristics are alternatives to BP that reach high-quality solutions without using many computational resources. In this work, the Cellular Genetic Algorithm (CGA) with a specially designed crossover operator called Damped Crossover (DX), is proposed to optimise weights and biases of the MLP to classify medical data. When compared against state-of-the-art algorithms, the CGA configured with DX obtained the minimal Mean Square Error value in three out of the five considered medical datasets and was the quickest algorithm with four datasets, showing a better balance between time consumed and optimisation performance. Additionally, it is competitive in enhancing classification quality, reaching the best accuracy with two datasets and the second-best accuracy with two of the remaining.

# Introduction

Nowadays, it is impossible to imagine advances in medicine without talking about artificial intelligence. The incredible amount of data from different sources, such as medical images, data from clinical examinations, sensors and many others, outperforms by far the human capacity to process and analyse them [[1](#_bookmark56)]. For example, an average radiologist technician analyses about 215,000 radiography in about 40 years, while an artificial intelligence method processes that amount in about an hour [[2](#_bookmark57)].

Artificial Neural Networks (ANNs) undoubtedly are one of the arti- ficial intelligence methods that more contributions to the medical field have reported [[3](#_bookmark58)]. Examples of applications of ANNs to medicine are: diagnosis of diseases [[4](#_bookmark59),[5](#_bookmark60)], prediction of treatments behaviour [[6](#_bookmark61)–[8](#_bookmark62)] and preventive medicine [[9](#_bookmark63),[10](#_bookmark64)].

Multilayer Perceptron (MLP) [[11](#_bookmark65)] is a kind of ANN in which calculus units, called neurons, are organised in three types of layers. Each neuron is connected to all the neurons of the following layer, and data flow from the first to the last layer. Connections between neurons have a weight representing the strength of the linkage. In addition, both

hidden and output neurons have a bias that acts as a threshold of activation of the neuron.

What became attractive from the MLP was its ability to be a universal classifier that adapts to different distributions, features and complexities of data [[12](#_bookmark66)]. This quality is highly desirable in the medical field, considering that medical data can have noise, be imbalanced in the distribution of the classes and can have errors of registration [[13](#_bookmark67)]. The MLP effectiveness depends on its learning process, which iden- tifies the weights and biases values that minimise the classification error of training samples. The Back Propagation Algorithm (BP) is the standard way to make the MLP learn. However, BP has several weaknesses that can lead to a divergence in the MLP learning process, such as, a tendency to get stuck in local optima or dependency on initial

values of hyperparameters [[14](#_bookmark68),[15](#_bookmark69)].

In recent years, metaheuristics have gained attention as alterna- tives to the BP method. They are iterative algorithms able to find high-quality solutions in a reasonable time. One of their highlighted characteristics is that they can be applied to different kinds of prob- lems without needing specific knowledge [[16](#_bookmark70),[17](#_bookmark71)], which makes them

∗ Corresponding author.

*E-mail addresses:* [mrojas@mendoza-conicet.gob.ar](mailto:mrojas@mendoza-conicet.gob.ar) (M.G. Rojas), [acolivera@conicet.gov.ar](mailto:acolivera@conicet.gov.ar) (A.C. Olivera), [pjvidal@conicet.gov.ar](mailto:pjvidal@conicet.gov.ar) (P.J. Vidal).

<https://doi.org/10.1016/j.array.2022.100173>

Received 24 January 2022; Received in revised form 22 April 2022; Accepted 23 April 2022

Available online 30 April 2022

2590-0056/© 2022 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY-NC-ND license ([http://creativecommons.org/licenses/by-](http://creativecommons.org/licenses/by-nc-nd/4.0/) [nc-nd/4.0/](http://creativecommons.org/licenses/by-nc-nd/4.0/)).

# Nomenclature

*𝛽𝑘*

Bias for a neuron *𝑘* of the hidden or output layer.

*𝜂* Distribution index.

BP C9 CGA CS DE DX EO ES FWA GA GWO

IMGWO LOA MFO MLP MPA MSE MVO NUM PBIL PM

PSO RM SBX SCA

Sn Sp SSA TDE

UM VEWOA

WOA

Back Propagation Algorithm.

Type of Neighbourhood Compact-9. Cellular Genetic Algorithm.

Cuckoo Search. Differential Evolution. Damped Crossover.

Equilibrium Optimiser. Evolutionary Strategy. Fireworks Algorithms. Genetic Algorithm.

Grey Wolf Optimiser.

Inertia Motivated Grey Wolf Optimisation. Lion Optimiser Algorithm.

Moth–Flame Optimiser. Multilayer Perceptron. Marine Predators Algorithms. Mean Square Error.

Multi-Verse Optimisation. Non-Uniform Mutation.

Population-Based Incremental Learning. Polynomial Mutation.

Particle Swarm Optimisation. Random Mutation.

Simulated Binary Crossover. Sine–Cosine Algorithm.

Sensitivity. Specificity.

Salp Swarm Algorithm.

Trigonometric Mutation Differential Evolu- tion.

Uniform Mutation.

Velocity Enhanced Whale Optimisation Al- gorithm.

Whale Optimisation Algorithm.

**𝐁** Set of biases of the whole MLP.

**𝐋** Set of training samples of a given dataset.

**𝐖** Set of weights of the whole MLP.

*𝐻*

*𝜔*

*𝑖,𝑗*

A weight from the input neuron *𝑖* to the

hidden neuron

*𝑂*

*𝜔*

*𝑗,*1

*𝑗* in the MLP.

A weight from the hidden neuron *𝑗* to the

output neuron in the MLP.

*𝐵𝐼𝑖* Element *𝑖* of the best solution in the

population.

*𝑙𝑏* Lower variables boundary.

*𝑚* number of hidden neurons.

*𝑛* Number of input neurons.

*𝑂𝑓* 1*𝑖* Element *𝑖* of the first offspring of the

crossover operator.

*𝑂𝑓* 2*𝑖* Element *𝑖* of the second offspring of the

crossover operator.

*𝑝*1 First parent for the crossover operator.

*𝑝*2 Second parent for the crossover operator.

*𝑟* Random number.

*𝑆* A candidate solution found by an algorithm.

*𝑠𝑜𝑙𝑡*+1

*𝑖*

*𝑠𝑜𝑙𝑡*

*𝑖*

*𝑆𝑢𝑚𝑂*

1

*𝑆𝑢𝑚𝐻*

*𝑗*

Value of the position *𝑖* of the solution *𝑠𝑜𝑙* at evaluation *𝑡* + 1.

Value of the position *𝑖* of the solution *𝑠𝑜𝑙* at evaluation *𝑡*.

Result of the summation operation for the output neuron.

hidden neuron *𝑗*. Result of the summation operation for the

*𝑇* Total number of fitness evaluations to

perform.

*𝑡* Number of performed fitness evaluations.

*𝑢𝑏* Upper variables boundary.

*𝑋𝑖* An input to the MLP.

*𝑌* Binary output of the MLP.

CGA because they contribute to a better exploration and exploitation

*𝐻*

*𝑦*

*𝑗*

*𝑦𝑂*

1

Output of the hidden neuron *𝑗*.

Output of the unique output neuron.

of the search space [[23](#_bookmark77)]. Previous works demonstrated that the CGA gets competitive results compared to state-of-the-art algorithms when

*𝑦𝑙* Expected output for a sample *𝑙* of the

training set.

ABC Artificial Bee Colony.

ALO Ant Lion Optimiser.

|  |  |  |
| --- | --- | --- |
| ANN | Artificial Neural Network. |  |
| AX | Adjusted Crossover. |  |
| BAT | Bat Algorithm. |  |
| BBO | Biogeography-Based Algorithm. | Optimisation |

BOA Butterfly Optimisation Algorithm.

suitable for solving complex optimisation problems such as the learning process of the MLP [[18](#_bookmark72),[19](#_bookmark73)]. Studies have demonstrated that meta- heuristics can perform well in training MLP models, even when a large number of weights and biases must be optimised [[20](#_bookmark74),[21](#_bookmark75)].

Cellular Genetic Algorithm (CGA) [[22](#_bookmark76)] is a metaheuristic based on canonical Genetic Algorithm (GA), which works with a decentralised population where genetic operators act over a small overlapped sub- population per time. These features improve the performance of the

the optimisation process is applied in continuous search spaces [[24](#_bookmark78),[25](#_bookmark79)]. Nevertheless, evolving large-dimensional solutions could be challeng- ing for genetic operators, causing the optimiser to diverge. To address this problem is necessary to design new genetic operators that, by using external information such as the stage of the evolutionary process, can speed up and enhance the convergence of the algorithm [[26](#_bookmark80)].

In this paper, the CGA is used as an alternative to finding the optimal weights and biases of the MLP for medical data classification. A new crossover operator, called Damped Crossover (DX), is proposed to improve the performance of the CGA at traversing the search space. The purpose is to obtain a reliable method for training the MLPs and improving classification quality. The main contributions of this work can be summarised as follows:

1. A CGA is used for the optimisation of weights and biases of the MLP.
2. A novel genetic crossover operator called Damped Crossover (DX) is introduced. It uses the damped harmonic oscillation function and information related to the best current solution and the stage of the evolutionary process to determine the direction and magnitude of recombination.
3. Different configurations of the DX are evaluated against well- known genetic operators for the CGA. Five benchmark medical

calculated for a given neuron *𝑗* by Eq. ([2](#_bookmark3)). activation function. The most-used is the *sigmoid* function, which is

datasets were considered for experiments.

*𝑦𝐻* = *𝑓* (*𝑆𝑢𝑚𝐻* ) = 1

(2)

1. The CGA is compared deeply against state-of-the-art algorithms *𝑗*

*𝑗*

previously utilised for optimising a MLP. Comparisons are made

*𝑗* 1 + *𝑒*−*𝑆𝑢𝑚𝐻*

observing how much each one improves the classification quality of the MLP.

This paper is organised as follows. Section [2](#_bookmark6) introduces the MLP,

Being *𝑦𝐻* the final output of the hidden neuron *𝑗*. This output can

feed either another sub-layer of the hidden layer or the output layer.

*𝑗*

operation is performed by Eq. ([3](#_bookmark4)), where *𝑦𝐻* is the output of a neuron If there is a single output neuron, as shown in [Fig.](#_bookmark10) [1](#_bookmark10), the summation

*𝑗*

*𝑗,*1

gives a notion about the traditional ways of training, defines the

*𝑗* of the hidden layer, and *𝜔𝑂*

is the weight of the connection between

problem formally and presents the related works in literature. Section [3](#_bookmark11) describes the CGA, the representation of the solution and the DX. Section [4](#_bookmark22) is about the experiments configuration, the datasets used for tests, the state-of-the-art algorithms and genetic operators used in

experiments and the metrics to evaluate the quality of classification.

the hidden neuron *𝑗* and the output neuron. *𝛽𝑚*+1 is the bias of the

output neuron.

*𝑚*

∑

*𝑆𝑢𝑚𝑂* = *𝜔𝑂* × *𝑦𝐻* + *𝛽𝑚*+1 (3)

1

*𝑗*=0

*𝑗,*1

*𝑗*

Results and their analysis are shown in Section [5](#_bookmark47). Finally, conclusions and future work are presented in Section [6](#_bookmark55).

Finally, the result from the MLP to a given instance of a dataset is determined by the activation operation of the output neuron (see Eq. ([4](#_bookmark5))).

# Multilayer perceptron

*𝑦𝑂* = *𝑓* (*𝑆𝑢𝑚𝑂*) = 1

(4)

1 1 1 + *𝑒*

1

−*𝑆𝑢𝑚𝑂*

Multilayer Perceptron (MLP) is an Artificial Neural Network (ANN) belonging to the feed-forward neural network family. The MLP has a set of processing units called neurons that transform data to get an expected output [[27](#_bookmark81)].

Internally, neurons are organised by three well-differentiated layers. The first layer contains the input neurons, which receives the input

data and redirect them to the following layer. The number of input

When MLP is used for classification, the output of the MLP must be a discrete value able to distinguish between classes. Binary output

an ultimate binary output *𝑌* , that distinguishes between two classes, is commonly used. To convert the real output obtained by Eq. ([4](#_bookmark5)) to

Eq. ([5](#_bookmark7)) is used.

{0 if *𝑦𝑂 <* 0*.*5

neurons is usually the same as the number of features. The second layer, called the hidden layer, contains neurons that map the data using

*𝑌* =

1

1 if *𝑦𝑂* ≥ 0*.*5

1

(5)

mathematical functions. An MLP can be configured with one or more hidden layers, according to the complexity of the problem. Finally, the output layer receives the data transformed by the hidden layer and returns a result. The amount of neurons in the output layer depends on the codification of the expected result.

MLP is hierarchical and fully connected, meaning that neurons of one layer interact with all the neurons of the following layer by weighted connections, e.g., each input neuron is connected to all the neurons in the hidden layer.

Weights (*𝜔*) of each connection indicates how strong is the con-

neurons have an element called bias (*𝛽*), which is a threshold to adjust nection between two given neurons. In addition, hidden and output

the prediction by conditioning the neuron output. Depending on the

*𝛽* value, the response of a neuron will be excitatory (positive) or

* 1. *Problem definition*

Let a set of weights of connections between neurons **𝐖** and a set of biases **𝐁** for both hidden and output neurons. The objective is to find the best combination of weights **𝐖** and biases **𝐁** to minimise the Mean Squared Error (MSE) of the MLP. Thus, a solution *𝑆* to this problem

weights in **𝐖** and biases in **𝐁**, where each of its elements is a weight or is a real vector with a length equal to the sum of the total number of a bias to be optimised. The fitness function for a solution *𝑆* is shown

using the configuration in *𝑆*. in Eq. ([6](#_bookmark8)). It is calculated as the MSE of the output returned by the MLP

*𝐿*

1 ∑

*𝑓 𝑖𝑡𝑛𝑒𝑠𝑠*(*𝑆*) = *𝑀𝑆𝐸* = (*𝑦𝑙* − *𝑦𝑂* )2 (6)

**𝐋**

1*,𝑙*

| |

*𝑙*=1

where | | *𝑂*

inhibitory (negative) [[28](#_bookmark82),[29](#_bookmark83)]. The learning process of the MLP consist

**𝐋**

is the total amount of training samples, *𝑦*1*,𝑙* is the predicted

of finding the optimal set of weights and biases.

An structure of an MLP with *𝑛* input neurons, one hidden layer with

*𝑚* neurons and one output neuron is presented in [Fig.](#_bookmark10) [1](#_bookmark10). Weights for

real value in the range [0*,* 1] for the *𝑙*th training sample in the training set **𝐋**, obtained by Eq. ([4](#_bookmark5)). *𝑦𝑙* is the expected binary output for the *𝑙*th

training sample.

connections from input to hidden layer are represented by *𝜔𝐻*

*𝑖,𝑗*

and

MSE is frequently used to evaluate regressions. But, in the context

weights for connections from hidden to output layer are showed as

*𝜔𝑂* , with *𝑖* = {1*,* … *, 𝑛*} and *𝑗* = {1*,* … *, 𝑚*}. Biases appear as *𝛽𝑘* with

*𝑗,*1

predicted float output *𝑦𝑂* (previous to convert it binary) approximates of classifications, it provides a metric of how much error exist when the

*𝑂*

1*,𝑙*

*𝑘* = {1*,* … *, 𝑚* + 1}.

Hidden neurons transform data by performing two operations: Sum- mation and Activation. The former is the sum of the product between the outputs of neurons from the previous layer and the weights of the connections, added to the correspondent bias. Eq. ([1](#_bookmark9)) is used to apply

summation on a given neuron *𝑗* of the hidden layer.

∑*𝑛*

*𝑆𝑢𝑚𝐻* =

*𝜔𝐻* × *𝑋𝑖* + *𝛽𝑗* (1)

*𝑗*

*𝑖*=0

*𝑖,𝑗*

where *𝜔𝐻* is the weight of the connection between an input neuron *𝑖* of the input layer and the neuron *𝑗* of the hidden layer. *𝑋𝑖* is the output of neuron *𝑖* that feeds the neuron *𝑗*, and *𝛽𝑗* is the bias of the neuron *𝑗*.

*𝑖,𝑗*

The activation operation applies a mathematical function to map the result of the summation operation. This function is known as the

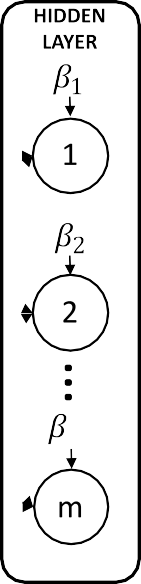
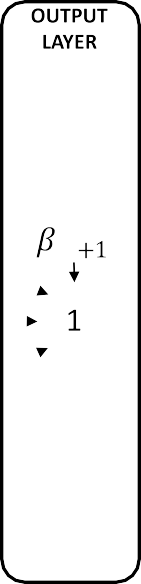
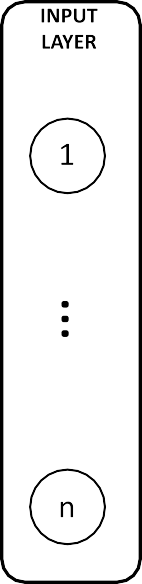
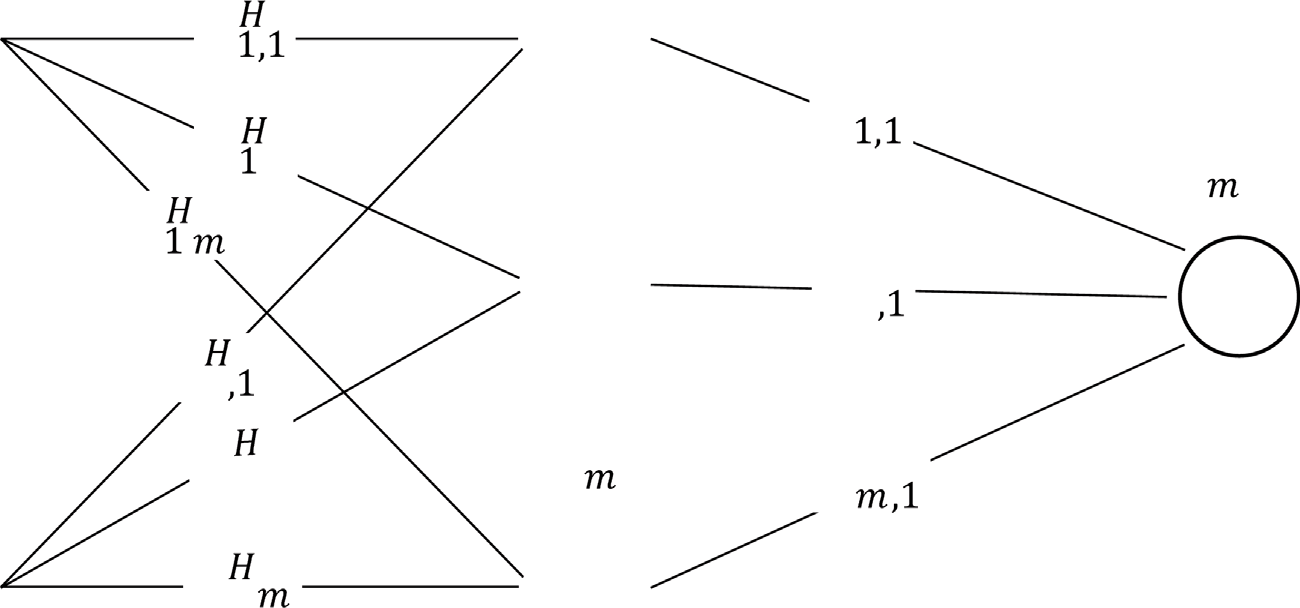
the expected binary output *𝑦𝑙* . The idea is, for negative samples *𝑦*1*,𝑙* has

to be near to 0 and for positive samples near to 1 [[15](#_bookmark69)].

Since MSE is a quadratic function, it strongly penalises when the MLP output is far from expected. It is the reason why MSE has been widely used as a fitness function to optimise weights and biases by metaheuristics [[14](#_bookmark68),[30](#_bookmark84)].

* 1. *Traditional approaches to train the MLP*

Learning is the process through which the ANN acquire knowledge. It is why they can perform and be effective in classification and regression tasks. In MLP, learning is reached by training the neural network, which is an iterative process for determining the optimal weights and biases to reduce the error between the obtained and the expected output.



**/ig. 1.** Example of structure of a Multilayer Perceptron (MLP).

One of the most conventional training methods for MLP is Back Propagation (BP) [[31](#_bookmark85)]. It starts setting random values for weights and biases. Classified samples (referred to as training set) are presented to the MLP to get an output value. Then, the error between the obtained and the desired values is calculated and propagated backwards to correct weights and biases. These steps are repeated until an acceptable error is reached [[29](#_bookmark83)].

The basic BP algorithm uses a first-order gradient descent for the optimisation of the MLP. Other existent approaches for optimising MLP are conjugate gradient [[32](#_bookmark86)] that is based on a second-order minimisation method, Quasi-Newton Method [[33](#_bookmark87)], Gauss–Newton [[34](#_bookmark88)] or Levenberg–Marquardt [[35](#_bookmark89)] that is based on the approximation by least-squares [[15](#_bookmark69)].

Although conventional approaches have shown to be effective in most of the problems they were applied to, there were situations in which they stuck in the same error value of MLP during extended peri- ods or even stuck in local optima. Furthermore, their success strongly depends on the initial values of weights, the values of momentum and the learning rate, which can provoke divergence if they are not right defined [[14](#_bookmark68)]. Finally, conventional methods put aside biases, focusing just on the values of the weights [[15](#_bookmark69),[36](#_bookmark90)].

Considering the disadvantages of traditional methods, in this paper, an MLP optimiser based on the Cellular Genetic Algorithm is proposed to determine the optimal combination of weights and biases values to improve the classification quality.

* 1. *Related works*

In the latest years, it has been demonstrated that metaheuristics are able to be applied for training MLPs, reaching even better results than traditional mathematical methods [[37](#_bookmark91)]. This motivated different authors to train MLPs by metaheuristics for different problems in real life, getting featured results.

The work by Kaveh et al. [[38](#_bookmark92)] uses the Biogeography-Based Op- timisation algorithm (BBO) to train an MLP that classifies sonar data into three different classes: noises, reverberation, and clutter. A novel mutation operator is introduced in this work to enhance the exploration capability of the BBO. Results have demonstrated that the proposal of new operators can positively impact the behaviour of the algorithm, increasing the resulting classification performance. Qiao et al. [[39](#_bookmark93)] also worked with sonar data, proposing a modified Whale Optimisation

Algorithm (WOA) to train an MLP that classifies sonar signals in real- time. This work introduces new ways to control the balance between exploration and exploitation during the evolutionary process using mathematical functions. This approach shows how the mathematical approach could help guide the seeking process of the metaheuristics in large search spaces. Results show that the proposal outperforms literature algorithms in terms of classification accuracy and speed of convergence.

An application to the field of robotics was made by Jalali et al. [[40](#_bookmark94)], who compared a set of nature-inspired algorithms to determine the optimal weights and biases of an MLP used for autonomous robot navigation. Considered algorithms were the Moth–Flame Algorithm (MFO), the Particle Swarm Optimisation (PSO), the Grey Wolf Opti- miser (GWO), the Cuckoo Search (CS) and the Multi-Verse Optimiser (MVO). The evaluated algorithms overcame the traditional approaches BP and Levenberg–Marquardt, demonstrating that metaheuristics per- form a better exploration and local-optima avoidance.

Mansouri et al. [[41](#_bookmark95)] implement a GWO hybridised to an Evolu- tionary Strategy (ES) algorithm to train an ANN to detect unusual sensor networks behaviour. The GWO is used when accuracy is critical, and ES is used when it is preferred to perform quickly detections. Results demonstrated that both approaches perform as expected, being the ANN capable of accurately recognising anomalies in industrial sensor networks. It also shows that different metaheuristics can provide distinct behaviour tailored to the particular context of the problem.

For the energy production problems, Aladejare et al. [[42](#_bookmark96)] developed an ANN trainer based on the PSO to predict the higher heat value of coal, biomass and other solid fuels to determine their energy con- tent. The ANN trained by PSO exhibits satisfactory predictive ability compared to multilinear and multi nonlinear models in the literature. Several proposals in the literature addressed classification problems over medical datasets from different specialities by using metaheuristics

for training ANNs.

In [[43](#_bookmark97)], a Butterfly Optimisation Algorithm (BOA) is used to train an MLP. Results showed that the BOA reached a performance similar to the existing approaches. Tests have just focused on the Parkinson and vertebral dataset, which suggests that different complexities and distributions of data need to be considered to confirm that the method can train the MLP effectively in complex situations. Furthermore, the author used the canonical version of the BOA, which suggests that bet- ter results would be achieved if a specific modification to the algorithm

is proposed. Another bio-inspired algorithm in the literature is the pro- posal by Das et al. [[44](#_bookmark98)] in that a Velocity Enhanced Whale Optimisation Algorithm (VEWOA) trains an ANN to classify data related to breast cancer, cervical cancer, and lung cancer. The VEWOA raises that each whale has to have a velocity, calculated as in PSO, where positions of the best and the previous positions of particles are considered. Results were compared against different machine learning approaches and the canonical WOA, making it difficult to observe whether this approach enhances the performance of metaheuristics specifically designed for MLP training.

Kumar et al. [[45](#_bookmark99)] propose an Inertia Motivated Gray Wolf Optimisa- tion Algorithm (IMGWO) that trains an MLP to classify data concerning breast cancer, heart disease, hepatitis and Parkinson’s disease. The IMGWO introduces a new method of calculating the balance between exploration and exploitation using a non-linear function. It also pro- poses to use velocity concepts similar to the PSO. The number of eval- uations required for the IMGWO to converge is significantly higher than the one used in the literature, indicating that those changes could have a negative effect on the convergence capacity of the algorithm. The IMGWO performed better than the canonical version of GA, PSO, and GWO. Despite this, comparisons didn’t make with the metaheuristics prepared to train MLPs, resulting in an unfair comparison.

Sharifi et al. [[46](#_bookmark100)] compared the GA and the PSO in defining the initial weights and biases of an MLP for detecting thyroid func- tional disease. After the initialisation phase, the MLP is trained by the Levenberg–Marquardt method. Results demonstrated that GA and PSO could contribute to accurate diagnoses, being the GA better than the PSO at improving the classification quality.

Salman et al. [[47](#_bookmark101)] compare GA, PSO and Fireworks Algorithms (FWA) at optimising weights and biases of an MLP for the classification of five benchmark medical datasets. Metaheuristics were tried over different MLP architectures, which contributed to understanding how metaheuristics performance could be affected by ANN architectural decisions and different parameters configurations. Results show that architectural changes did not significantly affect the performance of the algorithms. Nonetheless, increasing the iterations performed by each metaheuristic reported improvements in classification quality. Results inherently imply that metaheuristics perform well even when many weights or biases have to be optimised.

Bhattacharjee in [[48](#_bookmark102)] proposes five different hybridisations between GA and PSO to train an MLP that classifies human glioma from molec- ular brain neoplasia data. This paper provides an interesting point of view about how PSO and GA can be combined and which combination provides the best results. This work established that hybridisations between PSO and GA can report good results due to the synergetic effect generated.

In [[49](#_bookmark103)] authors compare eleven recently-proposed metaheuristics for training an ANN to classify fifteen different medical datasets. Al- gorithms included in the experiments were the Artificial Bee Colony (ABC), the Ant Lion Optimiser (ALO), the BBO, the Equilibrium Opti- miser (EO), the MFO, the Marine Predators Algorithm (MPA), the PSO, the Sine–Cosine Algorithm (SCA), the Salp Swarm Algorithm (SSA), the Trigonometric Mutation Differential Evolution (TDE), the WOA, a hybrid SSA with PSO, a hybrid SSA with SCA and the deterministic method for training ANN Levenberg–Marquardt. Evaluations focused on seven different classification quality metrics. Metaheuristics have proven to be highly effective in training ANNs. However, the evalu- ation has not considered the BP, overlooking one of the widely used options for training an ANN. The authors highlighted the EO among the metaheuristics, which obtained better values when considering all the classification metrics. Besides, the parameters of the EO were selected by observing which configuration provides the best results. The other algorithms were configured by adaptive approaches or setups used in literature, suggesting that comparisons could have been unfair.

Orozco et al. [[50](#_bookmark104)] applied multi-objective CGA to optimise an MLP. The multi-objective approach is focused on optimising the architecture

and weights of the connections of the MLP but not the biases. The method was tested over two breast cancer medical datasets, and the CGA was configured with standard genetics operators. The proposal has reached similar results to algorithms of the literature. However, since the algorithm did not take biases into account, new approaches should be raised for optimising all MLP parameters.

After analysing all these contributions, it seems clear that very few approaches design and evaluate new operators to improve numerically the search along with the problem space. In this paper, a CGA approach is proposed to address the problem of optimising the weights and biases of the MLP. The idea is to take advantage of the properties of the CGA for better exploration and exploitation, such as the slow spread of the best solution and exploitation in neighbourhoods. In addition, a novel specially designed crossover operator is proposed. The aim is to make the evolutionary process more accurate as it runs and to consider the best solution in neighbourhoods that do not have it. To the best of our knowledge, no presented work has encompassed the components of this paper.

# Cellular genetic algorithm

The Cellular Genetic Algorithm (CGA) [[22](#_bookmark76)] is a decentralised evolu- tionary algorithm based on the canonical Genetic Algorithm (GA) [[51](#_bookmark105)] that differs in how the population is managed. In CGA, individuals are distributed on a toroidal mesh where border individuals, in columns or rows, are connected to those on the opposite border.

During the evolutionary process, where the genetic operators are applied, individuals can interact just with their neighbours. Neighbours are the closest individuals determined by a type of neighbourhood, considering the Manhattan distance. Through the use of neighbour- hoods, CGA is able to conduct a local search process within each one, which facilitates the discovery of better near solutions (exploitation of the search space). It is possible because genetic operators are applied to neighbourhoods as they were isolated from the whole population. In addition, neighbourhoods are superposed, which implies that an individual takes part in more than one neighbourhood. Thus, an in- dividual spreads the improvements in its genes throughout all of the neighbourhoods it belongs to. This quality provokes the slowly spread of better solutions through the population, enhancing the exploration process.

[Fig.](#_bookmark12) [2](#_bookmark12) presents the way in that CGA evolves a given individ- ual (white circle). First, the neighbourhood of the individual to be evolved is obtained. In [Fig.](#_bookmark12) [2](#_bookmark12), the type of neighbourhood used is called Compact-9 or C9, which includes all the individuals in the mesh surrounding the individual to be evolved (marked with black circles). Genetics operators are applied to two individuals from the neighbour- hood. The resulting individual is evaluated, and a replacement policy is applied to decide if it will replace the individual being evolved.

A more profound point of view of the operation of the CGA is presented in Algorithm [1](#_bookmark15). Three well-differentiated stages can explain the evolutionary process carried out:

* **Initialisation stage:** It involves lines [2](#_bookmark13) to [4](#_bookmark13). The population is randomly generated and then evaluated. Next, the auxiliary

variable *𝑡*, which is used to count the number of evaluations

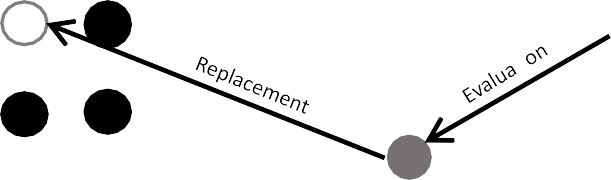
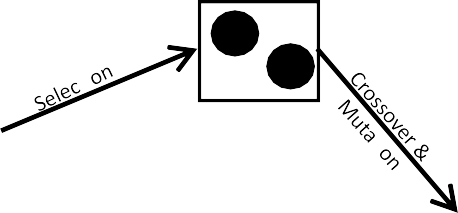
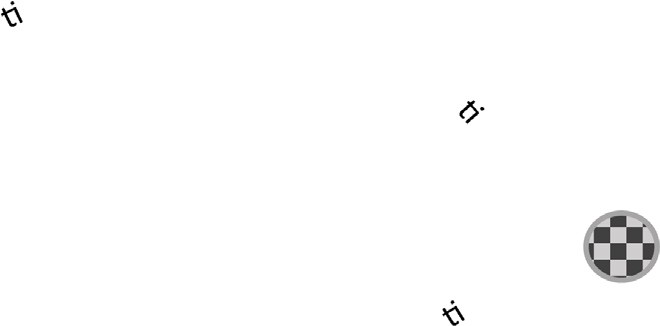
performed, is initialised.

* **Evolutionary stage:** It goes from line [5](#_bookmark13) to [15](#_bookmark13). If the stop condi- tion is not reached (checked on line [5](#_bookmark13)), the iteration is performed. Per step in the iteration, an individual and its neighbourhood are selected. Genetic operators are applied to them, generating two new individuals. After performing the fitness evaluations (line [11](#_bookmark13)), the selected individual is replaced if one of the gen- erated individuals is fitter. This process is shown in [Fig.](#_bookmark12) [2](#_bookmark12) and is performed on all the individuals in the population. In line

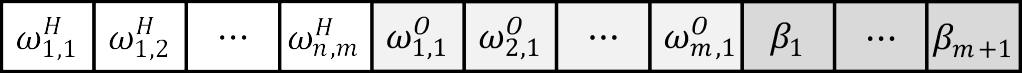
[13](#_bookmark13), the auxiliary variable *𝑡* is increased to reflect the number of

evaluations performed. The evolutionary stage is repeated while

the stop condition is not met.



**/ig. 2.** Type of neighbourhood C9 and application of genetics operators in an CGA.



**/ig. 3.** Example of vector representation of an MLP structure.

* **/inalisation stage:** Involves lines [16](#_bookmark13) and [17](#_bookmark13). When the evolution- ary stage has reached the stop condition, the best solution found is returned as a final solution.

**Algorithm 1** Pseudo-code of the Cellular Genetic Algorithm (CGA)

1: **function** CellularGA(popSize,crossoverRate, mutationRate, maxE- valuations)

2: *𝑝𝑜𝑝𝑢𝑙𝑎𝑡𝑖𝑜𝑛* ← *𝑖𝑛𝑖𝑡𝑖𝑎𝑙𝑖𝑠𝑒*(*𝑝𝑜𝑝𝑆𝑖𝑧𝑒*)

3: *𝑒𝑣𝑎𝑙𝑢𝑎𝑡𝑒*(*𝑝𝑜𝑝𝑢𝑙𝑎𝑡𝑖𝑜𝑛*)

4: *𝑡* ← *𝑝𝑜𝑝𝑆𝑖𝑧𝑒 ⊳* Evaluations counter

5: **while** *𝑡 < 𝑚𝑎𝑥𝐸𝑣𝑎𝑙𝑢𝑎𝑡𝑖𝑜𝑛𝑠* **do**

6: **for all** *𝑖𝑛𝑑𝑖𝑣𝑖𝑑𝑢𝑎𝑙* ∈ *𝑝𝑜𝑝𝑢𝑙𝑎𝑡𝑖𝑜𝑛* **do**

7: *𝑛𝑒𝑖𝑔ℎ𝑏𝑜𝑢𝑟𝑠* ← *𝑔𝑒𝑡𝑁 𝑒𝑖𝑔ℎ𝑏𝑜𝑢𝑟𝑠*(*𝑖𝑛𝑑𝑖𝑣𝑖𝑑𝑢𝑎𝑙, 𝑝𝑜𝑝𝑢𝑙𝑎𝑡𝑖𝑜𝑛*)

8: *𝑝𝑎𝑟𝑒𝑛𝑡𝑠* ← *𝑠𝑒𝑙𝑒𝑐𝑡𝑖𝑜𝑛*(*𝑛𝑒𝑖𝑔ℎ𝑏𝑜𝑢𝑟𝑠*)

9: *𝑜𝑓 𝑓 𝑠𝑝𝑟𝑖𝑛𝑔* ← *𝑐𝑟𝑜𝑠𝑠𝑜𝑣𝑒𝑟*(*𝑝𝑎𝑟𝑒𝑛𝑡𝑠, 𝑐𝑟𝑜𝑠𝑠𝑜𝑣𝑒𝑟𝑅𝑎𝑡𝑒*)

10: *𝑜𝑓 𝑓 𝑠𝑝𝑟𝑖𝑛𝑔* ← *𝑚𝑢𝑡𝑎𝑡𝑖𝑜𝑛*(*𝑜𝑓 𝑓 𝑠𝑝𝑟𝑖𝑛𝑔, 𝑚𝑢𝑡𝑎𝑡𝑖𝑜𝑛𝑅𝑎𝑡𝑒*)

11: *𝑒𝑣𝑎𝑙𝑢𝑎𝑡𝑒*(*𝑜𝑓 𝑓 𝑠𝑝𝑟𝑖𝑛𝑔*)

12: *𝑟𝑒𝑝𝑙𝑎𝑐𝑒*(*𝑖𝑛𝑑𝑖𝑣𝑖𝑑𝑢𝑎𝑙, 𝑜𝑓 𝑓 𝑠𝑝𝑟𝑖𝑛𝑔*)

13: *𝑡* ← *𝑡* + 2 *⊳* Increased by the number of offsprings

14: **end for**

The advantage of using the vector representation is that it results in a straightforward encoding and decoding process because elements of the individual do not need to be decoded to be applied to the MLP.

*3.2. The damped crossover*

This work proposes a novel crossover that considers different fea- tures of the evolutionary process that can lead the population to reach points of the search space near to a global optimum. The crossover operator is called The Damped Crossover (DX) because it is inspired by the damped harmonic oscillation function, which describes how an oscillating object tends to an equilibrium point as time runs [[52](#_bookmark106)].

The DX operator is based on two premises. The first one is that the knowledge acquired by the best individual during the evolutionary process is essential. So, this information must be considered when parents are crossed. The second premise is that influence from parents and the best solution has to be more specific as the evolutionary process runs because it is supposed that the solutions are near-optimal at the latest iterations.

In a given execution, the DX generates the *𝑖*th elements of two

offspring *𝑂𝑓* 1 and *𝑂𝑓* 2 by using Eqs. ([7](#_bookmark16)) and ([8](#_bookmark17)) respectively.

15: **end while**

*𝑂𝑓* 1 = *𝑝*1 + *𝑖𝑛𝑐*

(7)

16: *𝑏𝑒𝑠𝑡𝐼 𝑛𝑑𝑖𝑣𝑖𝑑𝑢𝑎𝑙* ← *𝑔𝑒𝑡𝐵𝑒𝑠𝑡𝐼 𝑛𝑑𝑖𝑣𝑖𝑑𝑢𝑎𝑙*(*𝑝𝑜𝑝𝑢𝑙𝑎𝑡𝑖𝑜𝑛*)

17: **return** *𝑏𝑒𝑠𝑡𝐼 𝑛𝑑𝑖𝑣𝑖𝑑𝑢𝑎𝑙*

18: **end function**

* 1. *Individual representation*

In literature, three kinds of individuals representation are used for MLP weights and biases optimisation: vector, matrix or binary [[14](#_bookmark68)].

The vector is used in this work. It represents each individual as a real array with a dimension equal to the total number of weights and biases in an MLP. Each element of the array corresponds to either a weight or a bias value. An example of this kind of representation, based on [Fig.](#_bookmark10) [1](#_bookmark10), is presented in [Fig.](#_bookmark14) [3](#_bookmark14).

*𝑖 𝑖 𝑖*

*𝑂𝑓* 2*𝑖* = *𝑝*2*𝑖* + *𝑖𝑛𝑐𝑖* (8)

where *𝑝*1*𝑖* and *𝑝*2*𝑖* are the *𝑖*th elements of the parents *𝑝*1 and *𝑝*2 respec- tively. *𝑖𝑛𝑐* is the increment performed over each element, calculated

by Eq. ([9](#_bookmark18)).

*𝑖𝑛𝑐𝑖* = *𝑑𝑖𝑓 𝑓𝑖* × (1 + *𝑟𝑎𝑡𝑖𝑜*) (9)

*𝑑𝑖𝑓 𝑓𝑖* is the difference between the *𝑖*th element of the best individual (*𝐵𝐼𝑖*) and the average of the *𝑖*th element of parents. It is obtained

by Eq. ([10](#_bookmark19)).

*𝑝*1*𝑖* + *𝑝*2*𝑖*

*𝑑𝑖𝑓 𝑓𝑖* = *𝐵𝐼𝑖* − 2 (10)



**/ig. 4.** Behaviour of damped harmonic oscillation function, presented in Eq. ([11](#_bookmark21)).

*𝑟𝑎𝑡𝑖𝑜* is determined by Eq. ([11](#_bookmark21)). It applies the damped harmonic oscillation function to determine how big the influence of parents and the best individual will be. This function provides a desirable

behaviour because it applies more variable changes at the beginning of the evolutionary process but is more precise at the end, doing minimal changes to the influence of the best individual and the parents.

*𝑟𝑎𝑡𝑖𝑜* = *𝐴* × *𝑒*(−*𝐶𝑡*) × sin (*𝑃* + 99*.*75 × *𝑡*) (11)

where *𝑡* is the number of evaluations performed at the moment of the execution of the DX. *𝐴* is the initial amplitude of the function set to

1.0. *𝐶* is the variable that controls how fast *𝐴* decreases. The bigger is

*𝐶*, the faster *𝐴* will decrease. *𝑃* is the phase of the function. The values of *𝐶* and *𝑃* used are 3 and 0.5, respectively. The Eq. ([11](#_bookmark21)) configured with the mentioned values of *𝐴*, *𝐶* and *𝑃* behaves like it is shown in [Fig.](#_bookmark20) [4](#_bookmark20), where a value of 0.0 in the *𝑥* axis represents the initial step

completed. The values of *𝐴*, *𝐶* and *𝑃* were selected to make the curve of the evolutionary process, and a value of 1.0 is when the process is

have a sustained decrease, thereby allowing precise variations at final evaluations but avoiding performing insignificant movements.

Both the *𝑟𝑎𝑡𝑖𝑜* and the *𝑑𝑖𝑓 𝑓𝑖* functions aim to improve the exploita-

tion of the DX by focusing on reaching the best individual and reducing

same time, *𝑑𝑖𝑓 𝑓* and *𝑟𝑎𝑡𝑖𝑜* act as direction indicators as they make the changes when the end of the evolutionary process is near. At the the new element increase or decrease the *𝑖*th element of the parents according to the difference with the *𝐵𝐼* .

# Experiments configuration

This section presents the experimental setup. It provides informa- tion related to the evaluations to be performed, considered algorithms, standard configurations of the MLP and characteristics of the used datasets.

Comparisons were performed against different variations of the CGA, introduced in Section [4.1](#_bookmark26), and state-of-the-art algorithms. Consid- ered algorithms and their configurations are presented in Section [4.2](#_bookmark32). The MLP uses the sigmoid function as the activation function. The fitness function for the learning process performed by metaheuristics is the MSE, also known as cost function in the ANN scope [[43](#_bookmark97),[53](#_bookmark107)]. The structure of the MLP is always the same. The number of input

neurons (*𝑛*) matches the number of features of the dataset. The MLP

neurons (*𝑚*) is determined by Eq. ([12](#_bookmark23)), following the rule established is configured with a unique hidden layer where the number of hidden

in [[14](#_bookmark68)].

*𝑚* = 2 × *𝑛* + 1 (12)

Each weight and biases values are restricted to the interval [−1,1]

according to the values handled in other approximations [[30](#_bookmark84),[54](#_bookmark108),[55](#_bookmark109)].

The training dataset aims to present the greatest amount of samples to the MLP to identify the optimal weights and biases values and obtain a reliable classification model. It is the only dataset involved in the training phase in traditional methods, so it is also used for the training process made by metaheuristics [[14](#_bookmark68),[37](#_bookmark91)]. The test dataset is used when the metaheuristics have finished their training phase to corroborate the ultimate performance of classification reached by the MLP, observe its generalisation ability and confirm whether there was or not over-fitting.

The stop condition for all the algorithms is to reach 10 000 fitness evaluations. Due to the non-deterministic nature of metaheuristics, 30 independent runs are executed for each algorithm and with each dataset [[53](#_bookmark107)–[55](#_bookmark109)]. The tables in the results section mark a result with boldface when it is the best and with italic when it is the second-best according to the used performance metric. The Wilcoxon rank-sum [[56](#_bookmark110)] test is applied to check whether the differences between the CGA variations and the other algorithms are statistically significant or just a matter of chance. The statistically significant differences are high-

level of 99% (i.e., a significance level of *𝛼* = 0*.*01) for the statistical lighted in the corresponding tables. This work considers a confidence

tests.

Algorithms are executed in the Toko cluster[1](#_bookmark24) with an AMD Opteron/ Epyc processor (64 cores and 128 GB of RAM). The operating sys- tem is Ubuntu 18.04 LTS. Metaheuristics are implemented using the jmetalpy [[57](#_bookmark111)] library, and for MLPs the neurolab[2](#_bookmark25) library is utilised.

* 1. *CGA operators*

Experiments compare the CGA configured with the Damped Crossover(DX) against other configurations that vary the genetic op- erators. Crossover operators used for experiments are:

* + - **Adjusted Crossover (AX):** It was proposed by Yasojima et al.

[[58](#_bookmark112)] to deal with two problems of the crossover operators for real-coded solutions. The first problem is that crossover methods may be stuck in local optima or reach not feasible solutions. The second problem is that generated solutions are limited to

the values of their parents. To deal with it, elements *𝑂𝑓* 1*𝑖* and

*𝑂𝑓* 2*𝑖* of first and second offspring, are generated by Eqs. ([13](#_bookmark27))

and ([14](#_bookmark28)).

*𝑂𝑓* 1*𝑖* = *𝑝*1*𝑖* + ((*𝑝*1*𝑖* − *𝑝*2*𝑖*) × *𝛼*) × *𝑔𝑖* (13)

*𝑂𝑓* 2*𝑖* = *𝑝*2*𝑖* + (*𝑝*1*𝑖* − *𝑝*2*𝑖*) × *𝛼* × *𝑔𝑖* (14) where *𝑝*1*𝑖* is the element *𝑖* of the first parent, *𝑝*2*𝑖* is the element *𝑖*

( )

of the second parent. *𝑔𝑖* is the gradient value of the element *𝑖*. It

is 1 if the value of the element *𝑖* in the fittest parent is bigger than in the other parent. Otherwise *𝑔𝑖* is −1. Finally, *𝛼* is the weight of the crossover. In this paper, *𝛼* is 0.02 as recommended in [[58](#_bookmark112)].

* + - **Simulated Binary Crossover (SBX):** It was proposed by Deb and Agrawal [[59](#_bookmark113)]. SBX simulates the single-point crossover of binary

representations. The elements of the offspring *𝑂𝑓* 1*𝑖* and *𝑂𝑓* 2*𝑖* are

obtained by Eq. ([15](#_bookmark29)) and Eq. ([16](#_bookmark30)) respectively.

*𝑂𝑓* 1*𝑖* = 0*.*5 × [(1 + *𝛽𝑞* ) × *𝑝*1*𝑖* + (1 − *𝛽𝑞* ) × *𝑝*2*𝑖*] (15)

*𝑂𝑓* 2*𝑖* = 0*.*5 × [(1 − *𝛽𝑞* ) × *𝑝*1*𝑖* + (1 + *𝛽𝑞* ) × *𝑝*2*𝑖*] (16)

1 <https://toko.uncu.edu.ar/>.

2 <https://pythonhosted.org/neurolab/>.

where *𝑝*1*𝑖* is the element *𝑖* of the first parent, *𝑝*2*𝑖* is the element *𝑖*

of the second parent and *𝛽𝑞* is an ordinate obtained by Eq. ([17](#_bookmark31)).

In order to increase the rigour of evaluations and observe if DX crossover can improve the performance of the CGA, twelve possible variations of CGA are obtained by combining crossover and muta-

*𝛽𝑞* = ⎪⎨[ 1

1

*𝑟 𝜂*+1

⎧ (2 × )

1

]

*𝜂*+1

if *𝑟* ≤ 0*.*5

(17)

tion operators. The DX variations arise from combining the Damped Crossover, proposed in this paper, with the four mutation operators, re-

⎪⎩ 2×(1−*𝑟*)

*𝑂𝑡ℎ𝑒𝑟𝑤𝑖𝑠𝑒*

sulting in the DX+NUM, DX+PM, DX+RM and DX+UM variations. The

AX variations are the different combinations of the Adjusted Crossover,

Being *𝑟* a random number between 0 and 1, and *𝜂* a distribution index fixed to 20 as considered in [[60](#_bookmark114)].

CGA variations are produced using four different mutation operators that vary in the strategy to diversify the population. The mutation operators are:

* + - **Non-Uniform Mutation (NUM):** It was first proposed by Michalewicz et al. [[61](#_bookmark115)]. The objective of this operator is to avoid

generating new elements randomly. The mutated value *𝑠𝑜𝑙𝑡*+1 of

AX+NUM, AX+PM, AX+RM and AX+UM. Finally, the SBX variations

ators, obtaining the SBX+NUM, SBX+PM, SBX+RM and SBX+UM. combine the Simulated Binary Crossover with the four mutation oper-

* 1. *Algorithms for comparison*

Experiments compare the DX variations of the CGA against AX variations, SBX variations and state-of-the-arts approaches that repre- sent different strategies to optimise weights and biases of the MLP.

the

*𝑖*

*𝑖*th element of a solution in the evaluation *𝑡* is generated

Evaluations focus on the convergence ability of each metaheuristic

and the classification quality reached for the MLP configured with

by Eq. ([18](#_bookmark33)) according to a mutation probability.

*𝑠𝑜𝑙𝑡*+ ▵ (*𝑡, 𝑢𝑏* − *𝑠𝑜𝑙𝑡*) if *𝑟* ≤ 0*.*5

*𝑠𝑜𝑙𝑡*+1 =

*𝑖*

*𝑖*

(18)

{

*𝑖 𝑖*

*𝑖*

*𝑠𝑜𝑙𝑡*+ ▵ (*𝑡, 𝑠𝑜𝑙𝑡* − *𝑙𝑏*) *𝑂𝑡ℎ𝑒𝑟𝑤𝑖𝑠𝑒*

With *𝑙𝑏* and *𝑢𝑏* as the lower and the upper bounds of the element

*𝑠𝑜𝑙𝑡*, a random number *𝑟* ∈ [0*,* 1] and the function ▵ (*𝑡, 𝑑*)

*𝑖*

calculated by Eq. ([19](#_bookmark34)).

*𝑇*

weights and biases determined by metaheuristics. Eight state-of-the-art algorithms are considered for comparisons:

* **Bat Algorithm (BAT)** is a metaheuristic proposed by Yang and Gandomi [[63](#_bookmark117)]. It is based on the behaviour of bats, which detect prey by an echolocation mechanism. In the algorithm, each bat moves through the search space by modifying its speed and based

▵ (*𝑡, 𝑑*) = *𝑑* ×

(1 − *𝑟*

(1− *𝑡* )*𝑏* )

(19)

on the proximity of the prey.

* **Cuckoo Search Algorithm (CS)** was proposed by Yang and Deb [[64](#_bookmark118)]. Cuckoo birds substitute eggs of other nests to their

*𝑇* is the maximum number of evaluations, and *𝑏* establishes the

of *𝑏* is bigger, more disturbance is applied by the operator as dependency on the evaluations number, set to 0.5. As the value evaluations are performed. Thus, small values of *𝑏* mean that

more precise movements will be made.

* + - **Uniform Mutation (UM):** In this case, the element *𝑠𝑜𝑙𝑡* is substi- tuted by a mutated value *𝑠𝑜𝑙𝑡*+1 obtained by Eq. ([20](#_bookmark35)).

*𝑖*

*𝑖*

*𝑠𝑜𝑙𝑡*+1 = *𝑠𝑜𝑙𝑡* + (*𝑟* − 0*.*5) × *𝑢* (20)

*𝑖 𝑖*

With a random number *𝑟* ∈ [0*,* 1] and the disturbance level of the operator *𝑢* set to 0.5. If the value generated is out of the range

delimited by the lower and the upper bounds, the new value will be one of the bounds.

* + - **Polynomial Mutation (PM):** This mutation operator was pro- posed by Deb and Agrawal [[62](#_bookmark116)]. A polynomial probability dis- tribution is used in this proposal to mutate each element of the

individual (*𝑠𝑜𝑙𝑡*), taking into account the lower (*𝑙𝑏*) and upper (*𝑢𝑏*)

bounds. Eq. ([21](#_bookmark36)) is used to obtain the new value *𝑠𝑜𝑙𝑡*+1 for the *𝑖*th

*𝑖*

*𝑖*

element of the solution to mutate.

=

*𝑖*

*𝑖*

ones with the aim of another bird to breed them. If the host bird realises that an egg is not its own, the host can destroy the egg or leave the nest. The algorithm mimics this behaviour, considering that a cuckoo egg is laid in a nest if it is better than the egg in the nest.

* + - **Differential Evolution (DE)** proposed in [[65](#_bookmark119)], the DE evolve a current individual by generating each element of a new individual considering a differential weight and the elements of three par- ents selected by a selection method. The new individual replaces the current one if it is better.
    - **Genetic Algorithm (GA)** [[51](#_bookmark105)] is a metaheuristic based on the Darwinian theory of the evolution of species. In GA, a population evolves by iterations called generations. Three genetic operators are applied. The selection operator selects a set of individuals to go on to the next generation or be recombined by the crossover operator. The crossover operator exploits the shared space of two individuals. Finally, the mutation operator performs random changes to increase the diversity over the population.
    - **Gray Wolf Optimisation (GWO)** [[66](#_bookmark120)] is a bio-inspired algorithm based on the hunting mechanism of the grey wolf and the leader-

(21)

*𝑡*+1

*𝑠𝑜𝑙*

*𝑖*

{*𝑠𝑜𝑙𝑡* + *𝛾𝐿* × (*𝑠𝑜𝑙𝑡* − *𝑙𝑏*) if *𝑟* ≤ 0*.*5

*𝑠𝑜𝑙𝑡* + *𝛾𝑅* × (*𝑢𝑏* − *𝑠𝑜𝑙𝑡*) *𝑂𝑡ℎ𝑒𝑟𝑤𝑖𝑠𝑒*

ship hierarchy of the herd. The algorithm establishes four types of

grey wolves to represent the hierarchy. Three hunting behaviours

where *𝑟* is a random number belonging to the range [0*,* 1] and the functions *𝛾𝐿*(*𝑑*) and *𝛾𝑅*(*𝑑*) are calculated by Eqs. ([22](#_bookmark37)) and ([23](#_bookmark38))

*𝑖*

*𝑖*

respectively.

1

*𝛾𝐿*(*𝑟*) = (2 × *𝑑*) 1+*𝜂* (22)

are simulated in the optimisation process, search for prey, encircle the prey, and attack the prey.

* + - * **Moth–/lame Optimisation (M/O)** [[67](#_bookmark121)] is based on the moth behaviour. Moth usually navigates in the direction of the moon because it is an efficient way to go through considerable dis- tances. But, due to the artificial lights, moths get disoriented and

1

*𝛾* (*𝑟*) = 1 − 2 × (1 − *𝑑*)

( )

*𝑅*

1+*𝜂*

(23)

are kept in circles around the light up to they die. MFO mimics

With *𝜂* fixed to 5, which is a user-defined parameter that controls

this behaviour mathematically to perform optimisation.

the perturbation applied to elements of the individuals.

* + - **Random Mutation (RM):** This operator is one of the simplest ways to mutate real-coded individuals. It mutates the value of

an element *𝑠𝑜𝑙𝑡* by generating a completely new value between

*𝑖*

the lower bound (*𝑙𝑏*) and the upper bound (*𝑢𝑏*) of the element, considering a random number *𝑟* in the range [0*,* 1]. For this, it

uses Eq. ([24](#_bookmark39)).

*𝑠𝑜𝑙𝑡*+1 = *𝑙𝑏* + (*𝑢𝑏* − *𝑙𝑏*) × *𝑟* (24)

*𝑖*

* + - **Multi-Verse Optimisation (MVO)** was developed by Mirjalili et al. [[68](#_bookmark122)]. It performs optimisation by mathematically mod- elling the concepts of white hole for exploration, black hole for exploitation of the search space and wormhole for local search.
    - **Particle Swarm Optimisation (PSO)** was proposed by Kennedy et al. [[69](#_bookmark123)]. PSO works imitating the behaviour of different organ- isms like bird flocking. It begins generating a swarm of particles distributed in the search space. At each iteration, the position and velocity of each particle are updated according to its previous position and the position of the best particle.

Algorithms parameters configurations. Summary of datasets features.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Algorithm | Parameter | Value |  | Dataset | Attributes | Instances | Class 1 | Class 2 |
|  | Population size | 10 × 10 |  | Breast | 8 | 699 | Benign (458) | Malignant (241) |
|  | Neighbourhood | C9 |  | Diabetes | 8 | 768 | Positive (268) | Negative (500) |
| CGA (This paper) | Crossover probability | 0.9 |  | Liver | 6 | 345 | Liver Disorders (200) | Normal (145) |
|  | Mutation probability | 0.01 |  | Parkinson | 22 | 195 | Parkinson (147) | Healthy (48) |
|  | Selection operator | Binary tournament |  | Vertebral | 6 | 310 | Abnormal(210) | Normal (100) |
|  | Population size | 50 |  |  |  |  |  |  |
|  | Loudness | 0.5 |  |  |  |  |  |  |
| BAT | Pulse rate | 0.5 |  |  |  |  |  |  |

Frequency minimum 0

Frequency maximum 1

CS Number of nests 50

Discovery rate 0.25

|  |  |  |
| --- | --- | --- |
|  | Population size | 50 |
| DE | Crossover probability | 0.9 |
|  | Differential weight | 0.5 |
|  | Population size | 50 |

* + - * **Vertebral:** This dataset contains 310 patients classified as abnor-

mal (cases of disk hernia or spondylolisthesis) or normal. Each instance contains a set of six different variables [[77](#_bookmark131)].

Datasets were split into 66% for the training set and 34% for the test set. Stratified sampling is used to ensure that each subset respects the original distribution of classes. Features of every dataset were

normalised into the interval [0*,* 1] by using the max–min normalisation

GA Crossover operator SBX (Prob.: 0.9)

Mutation operator UM (Prob.: 0.01)

Selection operator Binary tournament

Population size 50

method, calculated by Eq. ([25](#_bookmark42)).

*𝐴*′ = *𝐴𝑖* − *𝑚𝑖𝑛𝐴*

(25)

GWO

*𝑎̂*

Decrease linearly from 2 to 0

*𝑖 𝑚𝑎𝑥𝐴* − *𝑚𝑖𝑛𝐴*

MFO

MVO

PSO

Population size 50

*𝑏* 1

*𝑡* [−1*,* 1]

Population size 50

Min. wormhole existence prob. 0.2 Max. wormhole existence prob. 1.0

Number of particles 50

Inertia weight 0.721

Cognitive component 1.193

where *𝐴𝑖* is the value for the feature *𝐴* corresponding to the instance *𝑖*,

*𝐴*′ is the new value resulting from the normalisation, *𝑚𝑖𝑛𝐴* and *𝑚𝑎𝑥𝐴* are the minimum and the maximum value for the feature *𝐴* respectively.

*𝑖*

This process is fundamental because it prevents that variables with big range values affect the other features of the dataset.

*4.4. Classification measures*

[Table](#_bookmark40) [1](#_bookmark40) presents a summary of the configuration for each considered algorithm. All the CGA versions use the same parameters configuration. Parameters utilised by state-of-the-art algorithms are established based on previous works oriented to adjust weights and biases of the MLP with metaheuristics [[30](#_bookmark84),[55](#_bookmark109),[70](#_bookmark124),[71](#_bookmark125)].

* 1. *Datasets*

A set of classification quality measures evaluates the performance of every MLP trained by the considered algorithms. Each metric provides a different point of view about how well the classification was performed, taking as a basis the amounts of True Positives (TP) or positive in- stances classified as positive, False Positives (FP) or negative instances classified as positive, True Negatives (TN) or negative instances classi- fied as negative and False Negatives (FN) or positive instances classified as negative.

**Accuracy**: it is the proportion of instances well classified to the total amount of instances. This metric is calculated by Eq. ([26](#_bookmark44)).

Evaluations of the MLP optimised by metaheuristics were performed

using five different medical datasets obtained from the UCI machine learning repository.[3](#_bookmark43) Each dataset is described in the following para-

*𝐴𝑐𝑐𝑢𝑟𝑎𝑐𝑦* = *𝑇 𝑃* + *𝑇 𝑁*

*𝑇 𝑃* + *𝐹 𝑁* + *𝐹 𝑃* + *𝑇 𝑁*

(26)

graphs and a summary is provided in [Table](#_bookmark41) [2](#_bookmark41).

* + - **Breast:** This dataset is composed of 699 instances, where each one corresponds to a patient submitted to surgery. Eight vari-

**Specificity (Sp)**: it is the proportion of negative instances classified

as negative to the total of negative instances. It provides an idea about the ability of the MLP to identify patients that do not have a given disease rightly. It is obtained by Eq. ([27](#_bookmark45)).

ables were measured, and the instances are distinguished between benign or malignant cases [[72](#_bookmark126),[73](#_bookmark127)].

*𝑆𝑝𝑒𝑐𝑖𝑓 𝑖𝑐𝑖𝑡𝑦* = *𝑇 𝑁*

*𝑇 𝑁* + *𝐹 𝑃*

(27)

* + - **Diabetes:** It is composed of 768 instances which are classified as positive or negative diabetes cases. Data were collected from a population of Pima-Indian women at least 21 years old living near Phoenix, Arizona, USA [[74](#_bookmark128)].
    - **Liver:** Instances come from blood tests performed over 345

**Sensitivity (Sn)**: it is the proportion of positive instances classified as positive to the total of positive instances. It provides a notion about the ability to detect positive cases of a given disease. It is calculated by Eq. ([28](#_bookmark46)).

male patients with apparent liver disorders by excessive alco- hol consumption. Instances are split into positives and negative classes [[75](#_bookmark129)].

* + - **Parkinson:** This dataset was obtained from voice analysis per- formed over thirty-one patients. Specialists took almost 6 record-

*𝑆𝑒𝑛𝑠𝑖𝑡𝑖𝑣𝑖𝑡𝑦* = *𝑇 𝑃*

*𝑇 𝑃* + *𝐹 𝑁*

# Experimental results

(28)

ings for each individual. Recordings have 22 different metrics, and each one is classified as Parkinson’s disease or normal [[76](#_bookmark130)].

3 <https://archive.ics.uci.edu/>.

Experiments are divided into two parts. The first part analyses the convergence ability of each metaheuristic to optimal points of the search space and their time consumption. The second part evaluates the quality of the classification of the MLP configured with the weights and biases yielded by the metaheuristics.

**Table 3**

Mean and standard deviation of the fitness quality indicator.

Algorithm Breast Diabetes Liver Parkinsons Vertebral

**Table 4**

Vertebral). A confidence level of 99% (a significance level *𝛼* = 0*.*01) was considered. Wilcoxon values of the fitness quality indicator (Breast, Diabetes, Liver, Parkinsons,

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BAT | 0*.*0360*.*002 | 0*.*1520*.*002 | 0*.*1960*.*006 | 0*.*0940*.*007 | 0*.*1320*.*004 | Algorithm | DX+NUM | DX+PM | DX+RM | DX+UM |
| CS | 0*.*0530*.*004 | 0*.*1710*.*005 | 0*.*2110*.*004 | 0*.*2200*.*027 | 0*.*1480*.*006 | BAT | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ – ▵ | ▵ ▵ ▵ ▵ ▵ |
| DE | 0*.*0480*.*002 | 0*.*1640*.*004 | 0*.*2080*.*004 | 0*.*1820*.*018 | 0*.*1440*.*003 | CS | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ |
| GA | 0*.*0360*.*002 | 0*.*1530*.*001 | 0*.*1990*.*002 | 0*.*1040*.*006 | 0*.*1330*.*002 | DE | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ |
| GWO | 0*.*0350*.*002 | 0*.*1530*.*001 | 0*.*2000*.*003 | 0*.*0920*.*004 | 0*.*1340*.*002 | GA | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ |
| MFO | 0*.*0350*.*002 | 0*.*1530*.*002 | 0*.*1950*.*004 | 0*.*1070*.*010 | 0*.*1310*.*002 | GWO | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ – ▵ | ▵ ▵ ▵ ▵ ▵ |
| MVO | 0*.*0300*.*002 | 0*.*1470*.*002 | **𝟎***.***𝟏𝟕𝟔𝟎***.***𝟎𝟎𝟑** | 0*.*0770*.*007 | **𝟎***.***𝟏𝟏𝟖𝟎***.***𝟎𝟎𝟑** | MFO | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ |
| PSO | 0*.*0300*.*001 | 0*.*1480*.*001 | 0*.*1860*.*002 | *0.0760.005* | 0*.*1250*.*002 | MVO | ▵ ▵ *⊲ ⊲ ⊲* | ▵ ▵ *⊲ ⊲ ⊲* | – – *⊲ ⊲ ⊲* | ▵ ▵ *⊲* ▵ *⊲* |
| AX+NUM | 0*.*0360*.*002 | 0*.*1550*.*002 | 0*.*2020*.*002 | 0*.*1070*.*007 | 0*.*1370*.*003 | PSO | ▵ ▵ ▵ *⊲* ▵ | ▵ ▵ ▵ *⊲* ▵ | – – ▵ *⊲* ▵ | ▵ ▵ ▵ ▵ ▵ |
| AX+PM | 0*.*0360*.*002 | 0*.*1550*.*002 | 0*.*2020*.*002 | 0*.*1080*.*007 | 0*.*1370*.*002 | AX+NUM | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ |
| AX+RM | 0*.*0360*.*002 | 0*.*1560*.*002 | 0*.*2040*.*002 | 0*.*1090*.*007 | 0*.*1380*.*003 | AX+PM | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ |
| AX+UM | 0*.*0380*.*002 | 0*.*1570*.*003 | 0*.*2070*.*004 | 0*.*1150*.*013 | 0*.*1410*.*004 | AX+RM | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ |
| SBX+NUM | 0*.*0370*.*002 | 0*.*1560*.*001 | 0*.*2050*.*003 | 0*.*1260*.*008 | 0*.*1390*.*002 | AX+UM | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ |
| SBX+PM | 0*.*0370*.*001 | 0*.*1560*.*001 | 0*.*2050*.*002 | 0*.*1270*.*009 | 0*.*1390*.*002 | SBX+NUM | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ |
| SBX+RM | 0*.*0370*.*001 | 0*.*1560*.*001 | 0*.*2060*.*002 | 0*.*1260*.*009 | 0*.*1400*.*002 | SBX+PM | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ |
| SBX+UM | 0*.*0370*.*001 | 0*.*1550*.*001 | 0*.*2060*.*002 | 0*.*1170*.*008 | 0*.*1390*.*002 | SBX+RM | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ |
| DX+NUM | 0*.*0290*.*002 | 0*.*1460*.*002 | 0*.*1820*.*002 | 0*.*0870*.*006 | 0*.*1220*.*002 | SBX+UM | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ | ▵ ▵ ▵ ▵ ▵ |
| DX+PM | *0.0280.001* | *0.146*0.001 | 0*.*1820*.*002 | 0*.*0870*.*006 | 0*.*1220*.*002 | DX+NUM |  | – – – – – | – *⊲ ⊲ ⊲* – | ▵ ▵ ▵ ▵ ▵ |
| DX+RM | 0*.*0300*.*002 | 0*.*1470*.*001 | 0*.*1840*.*002 | 0*.*0920*.*005 | 0*.*1230*.*002 | DX+PM |  |  | *⊲ ⊲ ⊲ ⊲* – | ▵ ▵ – ▵ ▵ |
| DX+UM | **𝟎***.***𝟎𝟐𝟖𝟎***.***𝟎𝟎𝟏** | **𝟎***.***𝟏𝟒𝟓𝟎***.***𝟎𝟎𝟏** | *0.1810.002* | **𝟎***.***𝟎𝟔𝟔𝟎***.***𝟎𝟎𝟒** | *0.1210.001* | DX+RM |  |  |  | ▵ ▵ ▵ ▵ ▵ |

* 1. *Fitness and time analysis*

This section begins with an analysis of the numerical performance of metaheuristics. Then, a comparison of the time consumed in seconds to reach the stop criteria is presented. Tests are made by applying the algorithms to the five considered benchmarks datasets during the 30 independent runs.

[Table](#_bookmark48) [3](#_bookmark48) reports the mean fitness and the standard deviation reached by the evaluated algorithms. The first column contains the considered algorithms, and the results obtained are presented in columns two to six. Results show that the CGA configured with DX crossover and the

UM mutation (DX+UM) overcomes the other metaheuristics in three

out of the five datasets (Breast, Diabetes and Parkinsons). A possible

explanation for these results could be that the UM does not apply a significant disturbance over the genes, contributing to finding better near solutions and not diverting the seek. Furthermore, UM is not influenced by the number of evaluations done (already considered by

the DX), which helps when the algorithm is stuck. DX+UM also reached

the second-best solution for liver and vertebral datasets, falling behind

the MVO, which got the best average of fitness values in both instances. These results appear to be related to how the neural network learns, which can be affected when a small number of attributes is used for

so if those attributes are noisy. Beyond that, the DX+UM kept very near training, as in liver and vertebral datasets. Results could be even worse

the best results obtained by the MVO, meaning that it achieves minimal values of MSE.

In general, DX variations have shown similar behaviour, suggesting that the DX characteristics of taking into account the best current solution and making more minor changes as the population evolves can increase in a significant way the exploration and exploitation capability of the CGA.

[Table](#_bookmark49) [4](#_bookmark49) shows the results of the statistical analysis obtained by applying the Wilcoxon rank-sum test over the fitness value obtained by each algorithm. The table compares the DX variations of the CGA (located in columns) against the other metaheuristics (placed in rows). Each table cell contains a set of five symbols, representing the result

Diabetes, Liver, Parkinson and Vertebral. A leftward triangle (*⊲*) means of the comparison using each of the datasets, namely Breast Cancer,

that the row algorithm gets statistically better values than the column algorithm. An upward triangle (▵) means that the column algorithm gets better values than the row metaheuristic. If no significant differ- ences are found, the place is completed with a dash (–). For example, the first upward triangle in the table means that with the dataset Breast,

the DX+NUM was statistically better in the fitness value than the BAT

algorithm over the 30 runs.

The variation DX+UM has beaten all the other algorithms at fitness The Wilcoxon test results confirm the tendency shown in [Table](#_bookmark49) [4](#_bookmark49).

value, except for the liver and vertebral datasets where the MVO was better.

DX+NUM and DX+PM have shown a similar performance, overcom-

and SBX as crossover operators. The DX+RM performed a little worse ing the BAT, CS, DE, GA, GWO, MVO, and all the CGA versions using AX

than the other DX variations. These results confirm the capability of the CGA and, specifically, the DX operator to be a robust alternative in that featured results are obtained most of the time they are executed.

The convergence curves for DX+UM (which obtained the best fitness

results) compared to the state-of-the-art algorithm are presented from

evolutionary process advance. As can be seen, the DX+UM converges [Figs.](#_bookmark50) [5(a)](#_bookmark50) to [5(e)](#_bookmark51). Each plot shows how fitness is enhanced as the

to minimal values of the fitness function (MSE) rapidly, independent of the dataset to classify. Furthermore, it reached the minimal value of MSE in four datasets and was the second-best with the liver dataset ([Fig.](#_bookmark51) [5(c)](#_bookmark51)). The second algorithm that better converges to optimal

points is the MVO which performed similarly to the DX+UM and was

the best with the liver dataset. The worst algorithm was the CS, which

appears to be stuck in local optima in all the datasets.

Finally, [Table](#_bookmark52) [5](#_bookmark52) shows the average and standard deviation of time consumed (in seconds) by each algorithm to reach the stop condition. The first column shows the considered metaheuristics, and the follow- ing columns inform the results of the time evaluations for each dataset. The best algorithm was the CGA with the DX crossover and the NUM

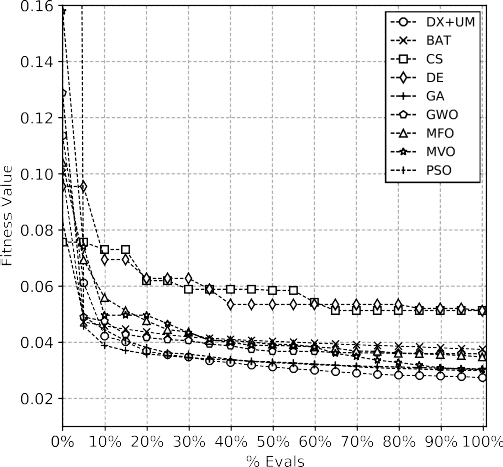
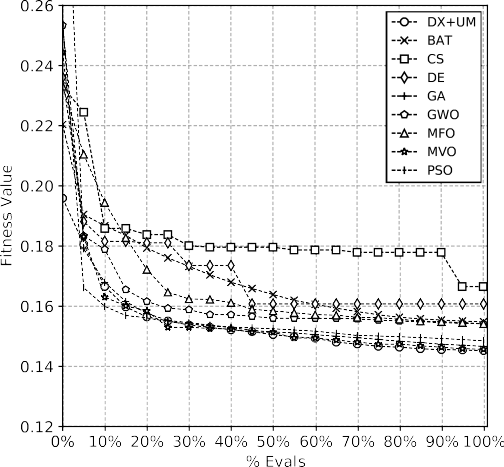
mutation in four datasets. The second-best was the DX+PM which stood

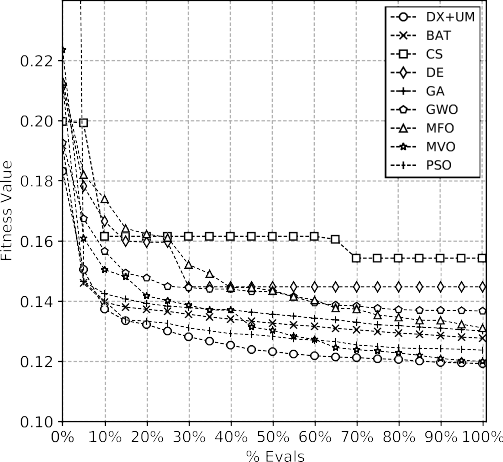
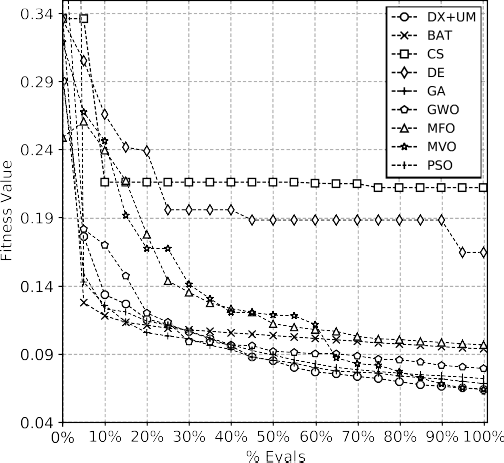
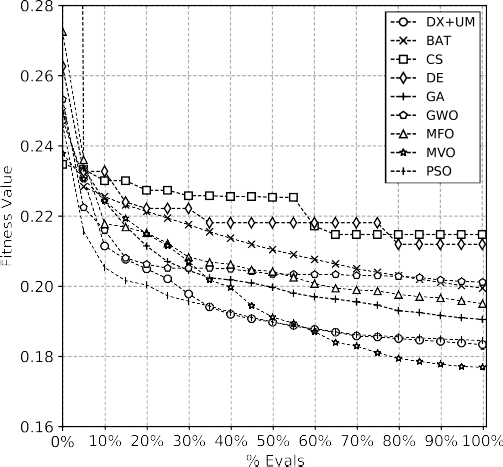
out in the same datasets as DX+NUM. However, all the DX variations

have spent similar execution times, showing that they can reach op-

timal points of the search space quicker than the other approaches. The CS performed better in the parkinsons dataset, but it is irrelevant because CS could not reach acceptable fitness values. The MVO, which had the best average fitness with liver and vertebral datasets, showed considerable high execution times (except with the Parkinson dataset), reflecting that the DX variations are better at balancing time consumed and performance.

Results presented in this section have demonstrated that the CGA variations can successfully tackle optimising weights and biases of the MLP, reaching fitness values comparable to or even better than other state-of-the-art algorithms. Furthermore, the results suggest that the characteristics of the DX crossover, of considering the best solution and the number of evaluations carried out, enable the CGA to achieve featured results in a short period of time, being even better than using a different crossover operator.



**/ig. 5.** Convergence curves of fitness value (MSE) of the five datasets.

* 1. *Classification metrics analysis*

This section provides a point of view about the performance of the MLP configured by the metaheuristics. First of all, the accuracy reached by each MLP is informed. Next, Sensitivity (Sn) and Specificity (Sp) are analysed to observe if the classification model has been balanced

general, both metrics should be near 1, which suggests a minor number in classifying both classes or has had a preference for one of them. In

of classification errors. In medicine, it is more relevant to inform the results of Sn and Sp because, depending on the situation, it could be necessary to obtain a high sensitivity (e.g. when it is crucial not to miss

**Table 5**

Mean and standard deviation of the time quality indicator.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Algorithm | Breast | Diabetes | Liver | Parkinsons | Vertebral |
| BAT | 670*.*0853*.*469 | 731*.*0708*.*384 | 326*.*2351*.*927 | *273.507 5.402* | 296*.*5532*.*318 |
| CS | 671*.*2013*.*404 | 733*.*28010*.*611 | 324*.*9891*.*397 | **𝟐𝟓𝟑***.***𝟏𝟔𝟖𝟑***.***𝟔𝟑𝟑** | 289*.*59410*.*815 |
| DE | 679*.*6063*.*443 | 741*.*7857*.*983 | 332*.*5021*.*428 | 339*.*1174*.*780 | 297*.*4329*.*723 |
| GA | 675*.*9244*.*141 | 732*.*09610*.*269 | 329*.*2436*.*266 | 333*.*2895*.*370 | 299*.*9224*.*780 |
| GWO | 858*.*6347*.*135 | 922*.*1998*.*422 | 442*.*4382*.*194 | 1503*.*95962*.*336 | 409*.*9006*.*772 |
| MFO | 694*.*2845*.*010 | 755*.*43611*.*762 | 339*.*7925*.*065 | 389*.*8377*.*562 | 311*.*2313*.*031 |
| MVO | 688*.*6382*.*860 | 751*.*5527*.*176 | 337*.*8465*.*612 | 426*.*7264*.*205 | 306*.*3357*.*593 |
| PSO | 684*.*3853*.*963 | 739*.*88011*.*074 | 332*.*4986*.*024 | 392*.*3564*.*975 | 302*.*3035*.*761 |
| AX+NUM | 664*.*88824*.*317 | 728*.*18129*.*326 | 330*.*5785*.*546 | 311*.*9291*.*758 | 297*.*8356*.*342 |
| AX+PM | 665*.*01823*.*633 | 731*.*38026*.*913 | 324*.*70010*.*418 | 312*.*5431*.*231 | 297*.*1187*.*528 |
| AX+RM | 669*.*45522*.*040 | 728*.*37328*.*904 | 327*.*5958*.*761 | 310*.*7511*.*378 | 297*.*4606*.*379 |
| AX+UM | 667*.*60621*.*882 | 726*.*21525*.*157 | 329*.*5816*.*725 | 311*.*3411*.*342 | 295*.*2098*.*767 |
| SBX+NUM | 666*.*91023*.*772 | 729*.*47824*.*990 | 330*.*4256*.*960 | 319*.*0252*.*191 | 299*.*6255*.*178 |
| SBX+PM | 672*.*32721*.*499 | 728*.*70629*.*179 | 330*.*3326*.*679 | 318*.*3611*.*847 | 299*.*1045*.*860 |
| SBX+RM | 668*.*75820*.*317 | 726*.*25926*.*142 | 330*.*4066*.*855 | 317*.*4621*.*934 | 299*.*7605*.*351 |
| SBX+UM | 664*.*90923*.*683 | 725*.*25326*.*871 | 330*.*3016*.*435 | 318*.*3812*.*309 | 298*.*6486*.*704 |
| DX+NUM | **𝟒𝟏𝟎***.***𝟑𝟑𝟔𝟑𝟒***.***𝟎𝟒𝟓** | **𝟒𝟑𝟓***.***𝟒𝟕𝟖𝟒𝟗***.***𝟑𝟕𝟐** | **𝟐𝟏𝟕***.***𝟓𝟒𝟒𝟖***.***𝟖𝟕𝟗** | 621*.*92888*.*362 | **𝟏𝟗𝟗***.***𝟑𝟖𝟗𝟏𝟓***.***𝟐𝟏𝟕** |
| DX+PM | *425.737 20.098* | *454.12825.014* | *219.9896.717* | 674*.*76367*.*467 | *203.6918.301* |
| DX+RM | 427*.*16819*.*789 | 456*.*10227*.*252 | 220*.*7005*.*058 | 673*.*41964*.*273 | 204*.*1958*.*599 |
| DX+UM | 426*.*24624*.*491 | 461*.*05335*.*099 | 221*.*6834*.*913 | 676*.*34064*.*095 | 204*.*6558*.*437 |

**Table 6**

Mean and standard deviation of accuracy metric reached by each algorithm.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Algorithm | Breast | Diabetes | Liver | Parkinsons | Vertebral |
| BAT | 0*.*9760*.*005 | 0*.*7540*.*007 | 0*.*7510*.*018 | 0*.*8610*.*017 | *0.8750.013* |
| CS | 0*.*9570*.*011 | 0*.*7240*.*030 | 0*.*6910*.*030 | 0*.*6990*.*060 | 0*.*8070*.*033 |
| DE | 0*.*9620*.*006 | 0*.*7380*.*021 | 0*.*7270*.*028 | 0*.*7540*.*060 | 0*.*8380*.*031 |
| GA | 0*.*9730*.*005 | 0*.*7510*.*007 | *0.7580.016* | 0*.*8480*.*036 | 0*.*8730*.*014 |
| GWO | 0*.*9780*.*004 | 0*.*7510*.*005 | **𝟎***.***𝟕𝟔𝟎𝟎***.***𝟎𝟏𝟗** | **𝟎***.***𝟖𝟕𝟔𝟎***.***𝟎𝟐𝟏** | **𝟎***.***𝟖𝟕𝟔𝟎***.***𝟎𝟏𝟎** |
| MFO | 0*.*9760*.*006 | 0*.*7520*.*009 | 0*.*7570*.*014 | 0*.*8420*.*037 | 0*.*8710*.*014 |
| MVO | 0*.*9730*.*007 | 0*.*7570*.*009 | 0*.*7280*.*017 | 0*.*8540*.*032 | 0*.*8690*.*014 |
| PSO | 0*.*9780*.*005 | 0*.*7530*.*008 | 0*.*7530*.*018 | 0*.*8670*.*026 | 0*.*8730*.*011 |
| BP | 0*.*9530*.*011 | 0*.*7160*.*026 | 0*.*6470*.*043 | 0*.*8610*.*059 | 0*.*8180*.*036 |
| AX+NUM | 0*.*9740*.*005 | 0*.*7470*.*010 | 0*.*7440*.*019 | 0*.*8320*.*036 | 0*.*8640*.*017 |
| AX+PM | 0*.*9750*.*007 | 0*.*7500*.*010 | 0*.*7530*.*021 | 0*.*8320*.*027 | 0*.*8720*.*020 |
| AX+RM | 0*.*9750*.*005 | 0*.*7470*.*012 | 0*.*7400*.*020 | 0*.*8460*.*032 | 0*.*8680*.*019 |
| AX+UM | 0*.*9710*.*005 | 0*.*7440*.*008 | 0*.*7310*.*021 | 0*.*8240*.*043 | 0*.*8470*.*031 |
| SBX+NUM | 0*.*9710*.*006 | 0*.*7470*.*011 | 0*.*7430*.*019 | 0*.*8160*.*038 | 0*.*8570*.*026 |
| SBX+PM | 0*.*9730*.*005 | 0*.*7480*.*013 | 0*.*7390*.*019 | 0*.*8050*.*049 | 0*.*8690*.*023 |
| SBX+RM | 0*.*9750*.*007 | 0*.*7510*.*011 | 0*.*7400*.*020 | 0*.*8020*.*034 | 0*.*8610*.*020 |
| SBX+UM | 0*.*9720*.*006 | 0*.*7490*.*010 | 0*.*7430*.*016 | 0*.*8400*.*026 | 0*.*8680*.*020 |
| DX+NUM | *0.9810.004* | 0*.*7540*.*007 | 0*.*7480*.*019 | *0.8700.021* | 0*.*8730*.*009 |
| DX+PM | *0.9810.005* | *0.757 0.011* | 0*.*7510*.*015 | 0*.*8630*.*025 | 0*.*8730*.*013 |
| DX+RM | 0*.*9800*.*006 | 0*.*7540*.*011 | 0*.*7520*.*014 | 0*.*8600*.*021 | *0.8750.015* |
| DX+UM | **𝟎***.***𝟗𝟖𝟐𝟎***.***𝟎𝟎𝟓** | **𝟎***.***𝟕𝟓𝟖𝟎***.***𝟎𝟏𝟎** | 0*.*7490*.*018 | *0.8700.025* | *0.8750.016* |

a diagnosis) or a high specificity value (e.g. when mislabelling a sample as positive is detrimental) [[78](#_bookmark132)].

[Table](#_bookmark53) [6](#_bookmark53) shows the average of the accuracy values and the standard deviation obtained by the MLP configured with weights and biases generated by metaheuristics. Results were calculated using the test subset of every dataset.

Metaheuristics have overcome the mean accuracy of the Back Prop- agation algorithm (BP) in all the considered datasets. Focusing on the

DX variations, the DX+UM has stood out in two out of the five datasets

(Breast and Diabetes) and has had the second-best mean accuracy in two other datasets (Parkinson and Vertebral). The GWO has emerged as the best solution with Liver, Parkinsons and Vertebral datasets. Because the algorithms that achieved the best fitness value did not reach the best accuracy with these datasets, it is evident that there is no relationship between fitness value and accuracy. As can be seen, all the metaheuristics have obtained very similar results. In particular, CGA with DX crossover has proven to achieve competitive accuracy results with all the datasets.

[Table](#_bookmark54) [7](#_bookmark54) displays the mean of the Sn and the Sp for the DX variations and the algorithms of the state-of-the-art. The best results are marked with bold font, and the second-best is marked with italic font.

**Table 7**

Averages of classification metrics, Specificity (Sp) and Sensitivity (Sn), reached by each algorithm.

Algorithm Breast Diabetes Liver Parkinsons Vertebral

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Sp | Sn |  | Sp | Sn |  | Sp | Sn |  | Sp | Sn |  | Sp | Sn |
| BAT | 0.981 | 0.967 |  | 0.505 | **𝟎***.***𝟖𝟗𝟖** |  | 0.661 | 0.818 |  | *0.988* | 0.458 |  | *0.916* | 0.777 |
| CS | 0.950 | 0.970 |  | 0.431 | 0.894 |  | 0.539 | 0.802 |  | 0.713 | *0.652* |  | 0.848 | 0.709 |
| DE | 0.957 | 0.972 |  | 0.468 | 0.894 |  | 0.607 | 0.816 |  | 0.809 | 0.577 |  | 0.889 | 0.714 |
| GA | 0.976 | 0.967 |  | 0.502 | 0.895 |  | 0.662 | **𝟎***.***𝟖𝟐𝟖** |  | 0.955 | 0.506 |  | **𝟎***.***𝟗𝟏𝟖** | 0.765 |
| GWO | **𝟎***.***𝟗𝟖𝟒** | 0.967 |  | 0.503 | 0.895 |  | **𝟎***.***𝟔𝟖𝟐** | 0.818 |  | **𝟎***.***𝟗𝟗𝟏** | 0.510 |  | **𝟎***.***𝟗𝟏𝟖** | 0.776 |
| MFO | 0.977 | 0.975 |  | 0.502 | 0.896 |  | *0.673* | 0.819 |  | 0.920 | 0.592 |  | 0.911 | 0.775 |
| MVO | 0.979 | 0.961 |  | *0.519* | 0.895 |  | 0.607 | 0.816 |  | 0.952 | 0.540 |  | 0.900 | 0.792 |
| PSO | 0.981 | 0.973 |  | 0.514 | 0.892 |  | 0.664 | 0.819 |  | 0.973 | 0.531 |  | 0.905 | 0.794 |
| BP | 0.956 | 0.947 |  | 0.446 | 0.872 |  | 0.435 | 0.803 |  | 0.844 | **𝟎***.***𝟗𝟏𝟓** |  | 0.827 | 0.798 |
| DX+NUM | 0.982 | 0.977 |  | 0.511 | 0.894 |  | 0.651 | 0.820 |  | 0.982 | 0.510 |  | 0.905 | 0.797 |
| DX+PM | 0.982 | **𝟎***.***𝟗𝟖𝟏** |  | 0.517 | 0.896 |  | 0.654 | *0.823* |  | 0.980 | 0.487 |  | 0.901 | *0.805* |
| DX+RM | 0.983 | 0.974 |  | 0.506 | *0.987* |  | 0.659 | 0.820 |  | 0.975 | 0.492 |  | 0.903 | **𝟎***.***𝟖𝟎𝟗** |
| DX+UM | *0.983* | *0.980* |  | **𝟎***.***𝟓𝟐𝟑** | 0.894 |  | 0.653 | 0.819 |  | 0.977 | 0.529 |  | 0.905 | 0.803 |

Regarding the breast dataset, all the compared algorithms have shown an outstanding balance between sensitivity and specificity met- rics, which implies that samples of both classes mostly were well classified, reducing the rate of false positives and false negatives.

DX+UM was the second-best at classifying both positive and negative

samples. DX+PM was the best at classifying positives samples, while

GWO was the best with negatives samples.

For the diabetes dataset, DX+UM was the best at classifying negative

the best was the BAT algorithm, while the second-best was DX+RM. samples, showing a high specificity value. At classifying positive cases,

For the liver dataset, all the algorithms showed to better classify positive samples (high sensitivity). The best algorithm for classify-

ues. The best with sensitivity values was the GA. DX+PM was the ing negative samples was GWO which showed higher specificity val-

second-best in sensitivity values and was near to the best algorithm in specificity values.

Concerning the Parkinson dataset, the BP was better at classify- ing positive cases but kept under the other algorithms at classifying negative samples. The GWO was the second-best, obtaining the best

Among the DX variations, DX+NUM showed to be competitive at specificity and a similar sensitivity value as the other metaheuristics.

classifying positive and negative samples. Metaheuristics performed better at determining negative samples, suggesting they could learn patterns from a few samples (48 samples). But, the performance with positive samples was poor, which might have been related to noise in the data. In particular, BP appears to better tolerate noise in the Parkinson dataset.

Finally, with the vertebral dataset, the DX+RM showed the best

result on sensitivity and an acceptable specificity value, which suppose

a better classification. GA and GWO shared the best result of specificity. All the previous analyses indicate that CGA using the DX crossover offers better capabilities for exploring and exploiting solutions than the other approach, which enhances the classification ability of the MLP. It is necessary to remark that all the metaheuristics overcame the typical

BP, except in the Parkinsons dataset.

# Conclusion

This work proposes a CGA approach to determine the optimal weights and biases of a MLP to classify medical data accurately. The idea was to take advantage of the properties of the CGA that im- prove exploration and exploitation of the search space. One of the main contributions of this paper was the Damped Crossover (DX), a specially designed crossover operator, which based on the damped harmonic oscillation function, determines the magnitude and direction of recombination between two parents, using external information as the knowledge of the best solution and the stage of the evolutionary

process. DX operator has demonstrated to improve the exploration and exploitation of the search space of CGA even more.

Experiments use five well-known benchmark medical datasets. Com- parisons are against state-of-the-art algorithms and CGA versions con- figured with well-known genetic operators. Two aspects were eval- uated, the convergence capability of the algorithms and the quality of classification achieved by the MLP optimised by the evaluated metaheuristics.

In general, DX variations have demonstrated that they can rapidly converge to optimal points of the search space. Regarding MSE values,

the CGA+DX was the best in three out of the five considered datasets

and the second-best in the remaining two. The DX operator combined

with the UM mutation achieved better results than the other algorithms and reached minimal fitness values. Considering the times consumed by each algorithm, the DX variations were the best in four out of five datasets, overwhelming even to the CGA with other crossover operators. It is reliable proof that the DX operator performs its task efficiently, making the CGA work quicker.

The DX+UM produced better accuracy results in two datasets and

showed to be very near to the results of the DX+UM and the state-of- was the second-best in the other two datasets. The other DX variations

the-art algorithms. These results suggest that the optimisation process depends on the fitness function definition, because minimal values of MSE do not necessarily imply better accuracy values. Despite that, these results confirm that DX variations can have a featured performance in optimising weights and biases of the MLP, being able to improve the classification.

Besides, metrics of classification quality showed that solutions CGA variations with DX crossover get competitive results of specificity and sensitivity, deriving in a level of learning and generalisation com- parable to other approaches. Specifically, DX variations highlight in the Breast dataset by reaching the second-best result. With the other datasets, the performance was very similar to the state-of-the-art algo- rithms.

To conclude, results demonstrate that the CGA can be a robust and reliable tool for identifying the optimal weights and biases of the MLP for classifying medical data.

For future work, extending the CGA application to optimise the parameters and structure of neural networks is proposed. Furthermore, it is desirable to study alternatives to MSE function as fitness functions to obtain a better relationship between the fitness function of the metaheuristic and the accuracy reached by the MLP.

# CRediT authorship contribution statement

**Matías Gabriel Rojas:** Conceptualization, Methodology, Software, Investigation, Formal analysis, Writing – original draft, Writing – re- view & editing. **Ana Carolina Olivera:** Conceptualization, Method- ology, Validation, Investigation, Formal analysis, Writing – original draft, Writing – review & editing. **Pablo Javier Vidal:** Conceptualiza- tion, Methodology, Validation, Investigation, Formal analysis, Writing

- original draft, Writing – review & editing.

# Declaration of competing interest

The authors declare that they have no known competing finan- cial interests or personal relationships that could have appeared to influence the work reported in this paper.

# Acknowledgements

This work was supported in part by Universidad Nacional de Cuyo,

Investigaciones Científicas *𝑦* Técnicas, Argentina. Argentina (Project No. 06/B081-B). Authors thank Consejo Nacional de

# References

1. Topol EJ. High-performance medicine: the convergence of human and artificial intelligence. Nat Med 2019;25(1):44–56. [http://dx.doi.org/10.1038/s41591-018-](http://dx.doi.org/10.1038/s41591-018-0300-7)

[0300-7](http://dx.doi.org/10.1038/s41591-018-0300-7).

1. Mintz Y, Brodie R. Introduction to artificial intelligence in medicine. Minim Invasive Ther Allied Technol 2019;28(2):73–81. [http://dx.doi.org/10.1080/](http://dx.doi.org/10.1080/13645706.2019.1575882) [13645706.2019.1575882](http://dx.doi.org/10.1080/13645706.2019.1575882).
2. Kaul V, Enslin S, Gross SA. History of artificial intelligence in medicine. Gastrointest Endosc 2020;92(4):807–12. [http://dx.doi.org/10.1016/j.gie.2020.](http://dx.doi.org/10.1016/j.gie.2020.06.040) [06.040](http://dx.doi.org/10.1016/j.gie.2020.06.040).
3. Guan Q, Huang Y, Zhong Z, Zheng Z, Zheng L, Yang Y. Thorax disease clas- sification with attention guided convolutional neural network. Pattern Recognit Lett 2020;131:38–45. <http://dx.doi.org/10.1016/j.patrec.2019.11.040>.
4. Poudel S, Kim YJ, Vo DM, Lee S-W. Colorectal disease classification us- ing efficiently scaled dilation in convolutional neural network. IEEE Access 2020;8:99227–38. <http://dx.doi.org/10.1109/access.2020.2996770>.
5. Annunziata S, Pelliccioni A, Hohaus S, Maiolo E, Cuccaro A, Giordano A. The prognostic role of end-of-treatment FDG-PET/CT in diffuse large B cell lymphoma: a pilot study application of neural networks to predict time-to- event. Ann Nucl Med 2020;35(1):102–10. [http://dx.doi.org/10.1007/s12149-](http://dx.doi.org/10.1007/s12149-020-01542-y) [020-01542-y](http://dx.doi.org/10.1007/s12149-020-01542-y).
6. Chu CS, Lee NP, Adeoye J, Thomson P, Choi S-W. Machine learning and treat- ment outcome prediction for oral cancer. J Oral Pathol Med 2020;49(10):977–85. <http://dx.doi.org/10.1111/jop.13089>.
7. Koo KC, Lee KS, Kim S, Min C, Min GR, Lee YH, Han WK, Rha KH, Hong SJ, Yang SC, Chung BH. Long short-term memory artificial neural network model for prediction of prostate cancer survival outcomes according to initial treatment strategy: development of an online decision-making support system. World J Urol 2020;38(10):2469–76. <http://dx.doi.org/10.1007/s00345-020-03080-8>.
8. Cui S, Li C, Chen Z, Wang J, Yuan J. Research on risk prediction of dyslipidemia in steel workers based on recurrent neural network and LSTM neural network. IEEE Access 2020;8:34153–61. <http://dx.doi.org/10.1109/access.2020.2974887>.
9. Zeleznik R, Foldyna B, Eslami P, Weiss J, Alexander I, Taron J, Parmar C, Alvi RM, Banerji D, Uno M, Kikuchi Y, Karady J, Zhang L, Scholtz J-E, Mayrhofer T, Lyass A, Mahoney TF, Massaro JM, Vasan RS, Douglas PS, Hoffmann U, Lu MT, Aerts HJWL. Deep convolutional neural networks to predict cardiovascular risk from computed tomography. Nature Commun 2021;12(1). <http://dx.doi.org/10.1038/s41467-021-20966-2>.
10. Murtagh F. Multilayer perceptrons for classification and regression. Neurocom- puting 1991;2(5–6):183–97. [http://dx.doi.org/10.1016/0925-2312(91)90023-](http://dx.doi.org/10.1016/0925-2312(91)90023-5)
12. Soria E, Martín JD, Lisboa PJG. Classical training methods. In: Metaheuristic procedures for training neutral networks. Boston, MA: Springer US; 2006, p. 7–36. <http://dx.doi.org/10.1007/0-387-33416-5_1>, Ch. 1.
13. Chicco D. Ten quick tips for machine learning in computational biology. BioData Min 2017;10(1). <http://dx.doi.org/10.1186/s13040-017-0155-3>.
14. Mirjalili S, Mirjalili SM, Lewis A. Let a biogeography-based optimizer train your multi-layer perceptron. Inform Sci 2014;269:188–209. [http://dx.doi.org/](http://dx.doi.org/10.1016/j.ins.2014.01.038) [10.1016/j.ins.2014.01.038](http://dx.doi.org/10.1016/j.ins.2014.01.038).
15. Ojha VK, Abraham A, Snášel V. Metaheuristic design of feedforward neu- ral networks: A review of two decades of research. Eng Appl Artif Intell 2017;60:97–116. <http://dx.doi.org/10.1016/j.engappai.2017.01.013>.
16. Huang C, Li Y, Yao X. A survey of automatic parameter tuning methods for metaheuristics. IEEE Trans Evol Comput 2020;24(2):201–16. [http://dx.doi.org/](http://dx.doi.org/10.1109/tevc.2019.2921598) [10.1109/tevc.2019.2921598](http://dx.doi.org/10.1109/tevc.2019.2921598).
17. Swan J, Adriaensen S, Brownlee AE, Hammond K, Johnson CG, Kheiri A, Kraw- iec F, Merelo J, Minku LL, Özcan E, Pappa GL, García-Sánchez P, Sörensen K, Voß S, Wagner M, White DR. Metaheuristics ‘‘In the Large’’. European J Oper Res 2021. <http://dx.doi.org/10.1016/j.ejor.2021.05.042>.
18. Akay B, Karaboga D, Akay R. A comprehensive survey on optimizing deep learning models by metaheuristics. Artif Intell Rev 2021;2021:1–66. [http://dx.](http://dx.doi.org/10.1007/S10462-021-09992-0) [doi.org/10.1007/S10462-021-09992-0](http://dx.doi.org/10.1007/S10462-021-09992-0).
19. Galván E, Mooney P. Neuroevolution in deep neural networks: Current trends and future challenges. IEEE Trans Artif Intell 2021;2(6):476–93. [http://dx.doi.](http://dx.doi.org/10.1109/TAI.2021.3067574) [org/10.1109/TAI.2021.3067574](http://dx.doi.org/10.1109/TAI.2021.3067574).
20. Ding S, Su C, Yu J. An optimizing BP neural network algorithm based on genetic algorithm. Artif Intell Rev 2011;36(2):153–62. [http://dx.doi.org/10.](http://dx.doi.org/10.1007/s10462-011-9208-z) [1007/s10462-011-9208-z](http://dx.doi.org/10.1007/s10462-011-9208-z).
21. Such FP, Madhavan V, Conti E, Lehman J, Stanley KO, Clune J. Deep neu- roevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. 2018, [arXiv:1712.06567](http://arxiv.org/abs/1712.06567). URL [https:](https://arxiv.org/abs/1712.06567)

[//arxiv.org/abs/1712.06567](https://arxiv.org/abs/1712.06567).

1. Alba E, Dorronsoro B. Introduction to cellular genetic algorithms. In: Cellular genetic algorithms. Boston, MA: Springer US; 2008, p. 3–20. [http://dx.doi.org/](http://dx.doi.org/10.1007/978-0-387-77610-1_1) [10.1007/978-0-387-77610-1\_1](http://dx.doi.org/10.1007/978-0-387-77610-1_1), Ch. 1.
2. Salto C, Alba E. Cellular genetic algorithms: Understanding the behavior of using neighborhoods. Appl Artif Intell 2019;33(10):863–80. [http://dx.doi.org/10.1080/](http://dx.doi.org/10.1080/08839514.2019.1646005) [08839514.2019.1646005](http://dx.doi.org/10.1080/08839514.2019.1646005).
3. Alba E, Dorronsoro B. Continuous optimization. In: Cellular genetic algorithms. Boston, MA: Springer US; 2008, p. 167–74. [http://dx.doi.org/10.1007/978-0-](http://dx.doi.org/10.1007/978-0-387-77610-1_12)

[387-77610-1\_12](http://dx.doi.org/10.1007/978-0-387-77610-1_12), Ch. 12.

1. Dorronsoro B, Alba E. A simple cellular genetic algorithm for continuous opti- mization. In: 2006 IEEE international conference on evolutionary computation. 2006, p. 2838–44. <http://dx.doi.org/10.1109/CEC.2006.1688665>.
2. Tinós R. Artificial neural network based crossover for evolutionary algorithms. Appl Soft Comput 2020;95:106512. [http://dx.doi.org/10.1016/J.ASOC.2020.](http://dx.doi.org/10.1016/J.ASOC.2020.106512) [106512](http://dx.doi.org/10.1016/J.ASOC.2020.106512).
3. Mirjalili S. Evolutionary multi-layer perceptron. In: Evolutionary algorithms and neural networks: Theory and applications. Cham: Springer International Publishing; 2019, p. 87–104. <http://dx.doi.org/10.1007/978-3-319-93025-1_7>,

Ch. 7.

1. Principe JC, Lefebvre C, Fancourt CL. Dataflow learning in coupled lattices: An application to artificial neural networks. In: Handbook of global optimization: Volume 2. Boston, MA: Springer US; 2002, p. 363–86. [http://dx.doi.org/10.](http://dx.doi.org/10.1007/978-1-4757-5362-2_10) [1007/978-1-4757-5362-2\_10](http://dx.doi.org/10.1007/978-1-4757-5362-2_10), Ch. 10.
2. Krogh A. What are artificial neural networks? Nature Biotechnol 2008;26(2):195–

7. <http://dx.doi.org/10.1038/nbt1386>.

1. Heidari AA, Faris H, Aljarah I, Mirjalili S. An efficient hybrid multi- layer perceptron neural network with grasshopper optimization. Soft Comput 2018;23(17):7941–58. <http://dx.doi.org/10.1007/s00500-018-3424-2>.
2. Werbos PJ. Generalization of backpropagation with application to a recurrent gas market model. Neural Netw 1988;1(4):339–56. [http://dx.doi.org/10.1016/0893-](http://dx.doi.org/10.1016/0893-6080(88)90007-X) [6080(88)90007-X](http://dx.doi.org/10.1016/0893-6080(88)90007-X).
3. [Hestenes MR, Stiefel E, et al. Methods of conjugate gradients for solving linear](http://refhub.elsevier.com/S2590-0056(22)00033-9/sb32) [systems. J Res Natl Bur Stand 1952;49(6):409–36.](http://refhub.elsevier.com/S2590-0056(22)00033-9/sb32)
4. Chen O-C, Sheu BJ. Optimization schemes for neural network training. In: Pro- ceedings of 1994 IEEE international conference on neural networks (ICNN’94), Vol. 2. 1994, p. 817–22. <http://dx.doi.org/10.1109/ICNN.1994.374284>.
5. Bertsekas DP. Nonlinear programming. J Oper Res Soc 1997;48(3):334. [http:](http://dx.doi.org/10.1057/palgrave.jors.2600425)

[//dx.doi.org/10.1057/palgrave.jors.2600425](http://dx.doi.org/10.1057/palgrave.jors.2600425).

1. Marquardt DW. An algorithm for least-squares estimation of nonlinear parame- ters. J Soc Ind Appl Math 1963;11(2):431–41, URL [http://www.jstor.org/stable/](http://www.jstor.org/stable/2098941) [2098941](http://www.jstor.org/stable/2098941).
2. Devikanniga D, Vetrivel K, Badrinath N. Review of meta-heuristic optimiza- tion based artificial neural networks and its applications. J Phys Conf Ser 2019;1362:012074. <http://dx.doi.org/10.1088/1742-6596/1362/1/012074>.
3. Hemeida AM, Hassan SA, Mohamed A-AA, Alkhalaf S, Mahmoud MM, Senjyu T, El-Din AB. Nature-inspired algorithms for feed-forward neural network classifiers: A survey of one decade of research. Ain Shams Eng J 2020;11(3):659–75. <http://dx.doi.org/10.1016/j.asej.2020.01.007>.
4. Kaveh M, Khishe M, Mosavi MR. Design and implementation of a neighborhood search biogeography-based optimization trainer for classifying sonar dataset using multi-layer perceptron neural network. Analog Integr Circuits Signal Process 2018;100(2):405–28. <http://dx.doi.org/10.1007/s10470-018-1366-3>.
5. Qiao W, Khishe M, Ravakhah S. Underwater targets classification using local wavelet acoustic pattern and Multi-Layer Perceptron neural network optimized by modified Whale Optimization Algorithm. Ocean Eng 2021;219:108415. [http:](http://dx.doi.org/10.1016/j.oceaneng.2020.108415)

[//dx.doi.org/10.1016/j.oceaneng.2020.108415](http://dx.doi.org/10.1016/j.oceaneng.2020.108415).

1. Jalali SMJ, Hedjam R, Khosravi A, Heidari AA, Mirjalili S, Nahavandi S. Autonomous robot navigation using moth-flame-based neuroevolution. In: Evo- lutionary machine learning techniques: Algorithms and applications. Singapore: Springer Singapore; 2020, p. 67–83. [http://dx.doi.org/10.1007/978-981-32-](http://dx.doi.org/10.1007/978-981-32-9990-0_5)

[9990-0\_5](http://dx.doi.org/10.1007/978-981-32-9990-0_5), Ch. 5.

1. Mansouri A, Majidi B, Shamisa A. Metaheuristic neural networks for anomaly recognition in industrial sensor networks with packet latency and jitter for smart infrastructures. Int J Comput Appl 2018;43(3):257–66. [http://dx.doi.org/](http://dx.doi.org/10.1080/1206212x.2018.1533613) [10.1080/1206212x.2018.1533613](http://dx.doi.org/10.1080/1206212x.2018.1533613).
2. Aladejare AE, Onifade M, Lawal AI. Application of metaheuristic based artificial neural network and multilinear regression for the prediction of higher heating values of fuels. Int J Coal Prep Util 2020;1–22. [http://dx.doi.org/10.1080/](http://dx.doi.org/10.1080/19392699.2020.1768080) [19392699.2020.1768080](http://dx.doi.org/10.1080/19392699.2020.1768080).
3. Jalali SMJ, Ahmadian S, Kebria PM, Khosravi A, Lim CP, Nahavandi S. Evolv- ing artificial neural networks using butterfly optimization algorithm for data classification. In: Gedeon T, Wong KW, Lee M, editors. Neural information processing. Cham: Springer International Publishing; 2019, p. 596–607. [http:](http://dx.doi.org/10.1007/978-3-030-36708-4_49)

[//dx.doi.org/10.1007/978-3-030-36708-4\_49](http://dx.doi.org/10.1007/978-3-030-36708-4_49).

1. Das S, Mishra S, Senapati MR. New approaches in metaheuristic to classify medical data using artificial neural network. Arab J Sci Eng 2019;45(4):2459–71. <http://dx.doi.org/10.1007/s13369-019-04026-y>.
2. Kumar N, Kumar D. An improved grey wolf optimization-based learning of artificial neural network for medical data classification. J Inf Commun Technol 2021;20(Number 2):213–48. <http://dx.doi.org/10.32890/jict2021.20.2.4>.
3. Sharifi A, Alizadeh K. Comparison of the particle swarm optimization with the genetic algorithms as a training for multilayer perceptron technique to diagnose thyroid functional disease. Shiraz E-Med J 2020;22(1). [http://dx.doi.org/10.](http://dx.doi.org/10.5812/semj.100351) [5812/semj.100351](http://dx.doi.org/10.5812/semj.100351).
4. Salman I, Ucan O, Bayat O, Shaker K. Impact of metaheuristic iteration on artificial neural network structure in medical data. Processes 2018;6(5):57. [http:](http://dx.doi.org/10.3390/pr6050057)

[//dx.doi.org/10.3390/pr6050057](http://dx.doi.org/10.3390/pr6050057).

1. Bhattacharjee K, Pant M. Hybrid particle swarm optimization-genetic algorithm trained multi-layer perceptron for classification of human glioma from molecular brain neoplasia data. Cogn Syst Res 2019;58:173–94. [http://dx.doi.org/10.1016/](http://dx.doi.org/10.1016/j.cogsys.2019.06.003) [j.cogsys.2019.06.003](http://dx.doi.org/10.1016/j.cogsys.2019.06.003).
2. Si T, Bagchi J, Miranda PB. Artificial neural network training using metaheuris- tics for medical data classification: An experimental study. Expert Syst Appl 2022;116423. <http://dx.doi.org/10.1016/J.ESWA.2021.116423>.
3. Orozco-Monteagudo M, Taboada-Crispí A, Del Toro-Almenares A. Training of multilayer perceptron neural networks by using cellular genetic algorithms. In: Martínez-Trinidad JF, Carrasco Ochoa JA, Kittler J, editors. Progress in pattern recognition, image analysis and applications. Berlin, Heidelberg: Springer Berlin Heidelberg; 2006, p. 389–98. <http://dx.doi.org/10.1007/11892755_40>.
4. Holland JH. Genetic algorithms. Sci Am 1992;267(1):66–73. [http://dx.doi.org/](http://dx.doi.org/10.2307/24939139) [10.2307/24939139](http://dx.doi.org/10.2307/24939139), URL <http://www.jstor.org/stable/24939139>.
5. Kleppner D, Kolenkow R. The harmonic oscillator. In: An introduction to mechanics. 2. Cambridge: Cambridge University Press; 2013, p. 411–38. [http:](http://dx.doi.org/10.1017/cbo9781139013963.013)

[//dx.doi.org/10.1017/cbo9781139013963.013](http://dx.doi.org/10.1017/cbo9781139013963.013), Ch. 13.

1. Faris H, Aljarah I, Mirjalili S. Improved monarch butterfly optimization for unconstrained global search and neural network training. Appl Intell 2017;48(2):445–64. <http://dx.doi.org/10.1007/s10489-017-0967-3>.
2. Aljarah I, Faris H, Mirjalili S. Optimizing connection weights in neural networks using the whale optimization algorithm. Soft Comput 2016;22(1):1–15. [http:](http://dx.doi.org/10.1007/s00500-016-2442-1)

[//dx.doi.org/10.1007/s00500-016-2442-1](http://dx.doi.org/10.1007/s00500-016-2442-1).

1. Aljarah I, Faris H, Mirjalili S, Al-Madi N, Sheta A, Mafarja M. Evolving neural networks using bird swarm algorithm for data classification and regression applications. Cluster Comput 2019;22(4):1317–45. [http://dx.doi.org/10.1007/](http://dx.doi.org/10.1007/s10586-019-02913-5) [s10586-019-02913-5](http://dx.doi.org/10.1007/s10586-019-02913-5).
2. Gibbons JD, Chakraborti S. The general two-sample problem. In: Nonparametric statistical inference. 6th ed.. Boca Raton: CRC Press; 2020, p. 247–300. [http:](http://dx.doi.org/10.1201/9781315110479-6)

[//dx.doi.org/10.1201/9781315110479-6](http://dx.doi.org/10.1201/9781315110479-6), Ch. 6.

1. Benítez-Hidalgo A, Nebro AJ, García-Nieto J, Oregi I, Del Ser J. jMetalPy: A Python framework for multi-objective optimization with metaheuristics. Swarm Evol Comput 2019;51:100598. <http://dx.doi.org/10.1016/j.swevo.2019.100598>, URL <https://www.sciencedirect.com/science/article/pii/S2210650219301397>.
2. Yasojima EKK, de Oliveira RCL, Teixeira ON, Pereira RL. CAM-ADX: A new genetic algorithm with increased intensification and diversification for design optimization problems with real variables. Robotica 2019;37(9):1595–640. [http:](http://dx.doi.org/10.1017/s026357471900016x)

[//dx.doi.org/10.1017/s026357471900016x](http://dx.doi.org/10.1017/s026357471900016x).

1. [Deb K, Agrawal R. Simulated binary crossover for continuous search space.](http://refhub.elsevier.com/S2590-0056(22)00033-9/sb59) [Complex Syst 1995;9.](http://refhub.elsevier.com/S2590-0056(22)00033-9/sb59)
2. Chacón J, Segura C. Analysis and enhancement of simulated binary crossover. In: 2018 IEEE congress on evolutionary computation (CEC). 2018, p. 1–8. <http://dx.doi.org/10.1109/CEC.2018.8477746>.
3. Michalewicz Z. GAs: Selected topics. In: Genetic algorithms + Data struc- tures=Evolution programs. Berlin, Heidelberg: Springer Berlin Heidelberg; 1992,

p. 55–72. <http://dx.doi.org/10.1007/978-3-662-02830-8_5>, Ch. 5.

1. Deb K, Agrawal S. A niched-penalty approach for constraint handling in genetic algorithms. In: Artificial neural nets and genetic algorithms. Vienna: Springer Vienna; 1999, p. 235–43. <http://dx.doi.org/10.1007/978-3-7091-6384-9_40>.
2. Yang X-S, Gandomi AH. Bat algorithm: a novel approach for global engineer- ing optimization. Eng Comput 2012;29(5):464–83. [http://dx.doi.org/10.1108/](http://dx.doi.org/10.1108/02644401211235834) [02644401211235834](http://dx.doi.org/10.1108/02644401211235834).
3. Yang X-S, Deb S. Cuckoo search via Lévy flights. In: 2009 world congress on nature biologically inspired computing (NaBIC). 2009, p. 210–4. [http://dx.doi.](http://dx.doi.org/10.1109/NABIC.2009.5393690) [org/10.1109/NABIC.2009.5393690](http://dx.doi.org/10.1109/NABIC.2009.5393690).
4. Storn R, Price K. Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 1997;11(4):341–59. http://dx.doi.org/10.1023/a:1008202821328.
5. Mirjalili S, Mirjalili SM, Lewis A. Grey wolf optimizer. Adv Eng Softw 2014;69:46–61. <http://dx.doi.org/10.1016/j.advengsoft.2013.12.007>.
6. Mirjalili S. Moth-flame optimization algorithm: A novel nature-inspired heuris- tic paradigm. Knowl-Based Syst 2015;89:228–49. [http://dx.doi.org/10.1016/j.](http://dx.doi.org/10.1016/j.knosys.2015.07.006) [knosys.2015.07.006](http://dx.doi.org/10.1016/j.knosys.2015.07.006).
7. Mirjalili S, Mirjalili SM, Hatamlou A. Multi-verse optimizer: a nature-inspired algorithm for global optimization. Neural Comput Appl 2015;27(2):495–513. <http://dx.doi.org/10.1007/s00521-015-1870-7>.
8. Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of ICNN’95- International conference on neural networks, Vol. 4. 1995, p. 1942–8. [http:](http://dx.doi.org/10.1109/ICNN.1995.488968)

[//dx.doi.org/10.1109/ICNN.1995.488968](http://dx.doi.org/10.1109/ICNN.1995.488968).

1. Mirjalili S. How effective is the Grey Wolf optimizer in training multi-layer perceptrons. Appl Intell 2015;43(1):150–61. [http://dx.doi.org/10.1007/s10489-](http://dx.doi.org/10.1007/s10489-014-0645-7) [014-0645-7](http://dx.doi.org/10.1007/s10489-014-0645-7).
2. Yamany W, Fawzy M, Tharwat A, Hassanien AE. Moth-flame optimization for training multi-layer perceptrons. In: 2015 11th international computer engineer- ing conference (ICENCO). 2015, p. 267–72. [http://dx.doi.org/10.1109/ICENCO.](http://dx.doi.org/10.1109/ICENCO.2015.7416360) [2015.7416360](http://dx.doi.org/10.1109/ICENCO.2015.7416360).
3. Mangasarian OL, Street WN, Wolberg WH. Breast cancer diagnosis and prognosis via linear programming. Oper Res 1995;43(4):570–7. [http://dx.doi.org/10.1287/](http://dx.doi.org/10.1287/opre.43.4.570) [opre.43.4.570](http://dx.doi.org/10.1287/opre.43.4.570).
4. Wolberg WH, Mangasarian OL. Multisurface method of pattern separa- tion for medical diagnosis applied to breast cytology. Proc Natl Acad Sci 1990;87(23):9193–6. <http://dx.doi.org/10.1073/pnas.87.23.9193>.
5. Smith JW, Everhart JE, Dickson WC, Knowler WC, Johannes RS. Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In: Proceedings of the annual symposium on computer application in medical care. 1988, p. 261–5. PMC2245318[pmcid].
6. McDermott J, Forsyth RS. Diagnosing a disorder in a classification benchmark. Pattern Recognit Lett 2016;73:41–3. [http://dx.doi.org/10.1016/j.patrec.2016.01.](http://dx.doi.org/10.1016/j.patrec.2016.01.004) [004](http://dx.doi.org/10.1016/j.patrec.2016.01.004).
7. Little MA, McSharry PE, Roberts SJ, Costello DA, Moroz IM. Exploiting nonlinear recurrence and fractal scaling properties for voice disorder detection. BioMed Eng OnLine 2007;6(1):23. <http://dx.doi.org/10.1186/1475-925x-6-23>.
8. Rego da Rocha Neto A, de Alencar Barreto G. On the application of ensembles of classifiers to the diagnosis of pathologies of the vertebral column: A comparative analysis. IEEE Latin Am Trans 2009;7(4):487–96. [http://dx.doi.org/10.1109/](http://dx.doi.org/10.1109/TLA.2009.5349049) [TLA.2009.5349049](http://dx.doi.org/10.1109/TLA.2009.5349049).
9. Chu K. An introduction to sensitivity, specificity, predictive values and like- lihood ratios. Emerg Med 1999;11:175–81. [http://dx.doi.org/10.1046/J.1442-](http://dx.doi.org/10.1046/J.1442-2026.1999.00041.X) [2026.1999.00041.X](http://dx.doi.org/10.1046/J.1442-2026.1999.00041.X).