Array 14 (2022) 100173

Contents lists available at ScienceDirect

Array

journal homepage: www.elsevier.com/locate/array

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

Matías Gabriel Rojas [a,*], Ana Carolina Olivera [a,b], Pablo Javier Vidal [a,b]

[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

ARTICLE INFO

Keywords:
Multilayer Perceptron
Training methods
Cellular Genetic Algorithm
Metaheuristics
Medical data classification

ABSTRACT

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.

1. 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]. 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].

Artificial Neural Networks (ANNs) undoubtedly are one of the artificial intelligence methods that more contributions to the medical field have reported [3]. Examples of applications of ANNs to medicine are: diagnosis of diseases [4,5], prediction of treatments behaviour [6–8] and preventive medicine [9,10].

Multilayer Perceptron (MLP) [11] 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]. 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].

The MLP effectiveness depends on its learning process, which identifies 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 [14,15].

In recent years, metaheuristics have gained attention as alternatives 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 problems without needing specific knowledge [16,17], which makes them

[*] Corresponding author.
E-mail addresses: mrojas@mendoza-conicet.gob.ar (M.G. Rojas), acolivera@conicet.gov.ar (A.C. Olivera), 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

Nomenclature

β_k	Bias for a neuron k of the hidden or output layer.
η	Distribution index.
$\mathbf{B}$	Set of biases of the whole MLP.
$\mathbf{L}$	Set of training samples of a given dataset.
$\mathbf{W}$	Set of weights of the whole MLP.
$\omega^H_{i,j}$	A weight from the input neuron i to the hidden neuron j in the MLP.
$\omega^O_{j,1}$	A weight from the hidden neuron j to the output neuron in the MLP.
BI_i	Element i of the best solution in the population.
lb	Lower variables boundary.
m	number of hidden neurons.
n	Number of input neurons.
$Of1_i$	Element i of the first offspring of the crossover operator.
$Of2_i$	Element i of the second offspring of the crossover operator.
$p1$	First parent for the crossover operator.
$p2$	Second parent for the crossover operator.
r	Random number.
S	A candidate solution found by an algorithm.
sol_i^{t+1}	Value of the position i of the solution sol at evaluation $t+1$.
sol_i^t	Value of the position i of the solution sol at evaluation t.
Sum_1^O	Result of the summation operation for the output neuron.
Sum_j^H	Result of the summation operation for the hidden neuron j.
T	Total number of fitness evaluations to perform.
t	Number of performed fitness evaluations.
ub	Upper variables boundary.
X_i	An input to the MLP.
Y	Binary output of the MLP.
y_j^H	Output of the hidden neuron j.
y_1^O	Output of the unique output neuron.
y_l	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 Optimisation Algorithm.
BOA	Butterfly Optimisation Algorithm.
BP	Back Propagation Algorithm.
C9	Type of Neighbourhood Compact-9.
CGA	Cellular Genetic Algorithm.
CS	Cuckoo Search.
DE	Differential Evolution.
DX	Damped Crossover.
EO	Equilibrium Optimiser.
ES	Evolutionary Strategy.
FWA	Fireworks Algorithms.
GA	Genetic Algorithm.
GWO	Grey Wolf Optimiser.
IMGWO	Inertia Motivated Grey Wolf Optimisation.
LOA	Lion Optimiser Algorithm.
MFO	Moth–Flame Optimiser.
MLP	Multilayer Perceptron.
MPA	Marine Predators Algorithms.
MSE	Mean Square Error.
MVO	Multi-Verse Optimisation.
NUM	Non-Uniform Mutation.
PBIL	Population-Based Incremental Learning.
PM	Polynomial Mutation.
PSO	Particle Swarm Optimisation.
RM	Random Mutation.
SBX	Simulated Binary Crossover.
SCA	Sine–Cosine Algorithm.
Sn	Sensitivity.
Sp	Specificity.
SSA	Salp Swarm Algorithm.
TDE	Trigonometric Mutation Differential Evolution.
UM	Uniform Mutation.
VEWOA	Velocity Enhanced Whale Optimisation Algorithm.
WOA	Whale Optimisation Algorithm.

suitable for solving complex optimisation problems such as the learning process of the MLP [18,19]. 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 [20,21].

Cellular Genetic Algorithm (CGA) [22] 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 CGA because they contribute to a better exploration and exploitation of the search space [23]. Previous works demonstrated that the CGA gets competitive results compared to state-of-the-art algorithms when the optimisation process is applied in continuous search spaces [24,25]. Nevertheless, evolving large-dimensional solutions could be challenging 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].

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 datasets were considered for experiments.
4. The CGA is compared deeply against state-of-the-art algorithms previously utilised for optimising a MLP. Comparisons are made observing how much each one improves the classification quality of the MLP.

This paper is organised as follows. Section 2 introduces the MLP, gives a notion about the traditional ways of training, defines the problem formally and presents the related works in literature. Section 3 describes the CGA, the representation of the solution and the DX. Section 4 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. Results and their analysis are shown in Section 5. Finally, conclusions and future work are presented in Section 6.

2. Multilayer perceptron

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].

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 neurons is usually the same as the number of features. The second layer, called the hidden layer, contains neurons that map the data using 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 connection between two given neurons. In addition, hidden and output neurons have an element called bias (β), which is a threshold to adjust the prediction by conditioning the neuron output. Depending on the β value, the response of a neuron will be excitatory (positive) or inhibitory (negative) [28,29]. The learning process of the MLP consist 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. 1. Weights for connections from input to hidden layer are represented by $\omega^H_{i,j}$ and weights for connections from hidden to output layer are showed as $\omega^O_{j,1}$, with $i = \{1, \dots, n\}$ and $j = \{1, \dots, m\}$. Biases appear as β_k with $k = \{1, \dots, m+1\}$.

Hidden neurons transform data by performing two operations: Summation 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) is used to apply summation on a given neuron j of the hidden layer.

$$Sum^H_j = \sum_{i=0}^{n} \omega^H_{i,j} \times X_i + \beta_j \tag{1}$$

where $\omega^H_{i,j}$ is the weight of the connection between an input neuron i of the input layer and the neuron j of the hidden layer. X_i is the output of neuron i that feeds the neuron j, and β_j is the bias of the neuron j.

The activation operation applies a mathematical function to map the result of the summation operation. This function is known as the activation function. The most-used is the *sigmoid* function, which is calculated for a given neuron j by Eq. (2).

$$y^H_j = f(Sum^H_j) = \frac{1}{1 + e^{-Sum^H_j}} \tag{2}$$

Being y^H_j the final output of the hidden neuron j. This output can feed either another sub-layer of the hidden layer or the output layer. If there is a single output neuron, as shown in Fig. 1, the summation operation is performed by Eq. (3), where y^H_j is the output of a neuron j of the hidden layer, and $\omega^O_{j,1}$ is the weight of the connection between the hidden neuron j and the output neuron. β_{m+1} is the bias of the output neuron.

$$Sum^O_1 = \sum_{j=0}^{m} \omega^O_{j,1} \times y^H_j + \beta_{m+1} \tag{3}$$

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)).

$$y^O_1 = f(Sum^O_1) = \frac{1}{1 + e^{-Sum^O_1}} \tag{4}$$

When MLP is used for classification, the output of the MLP must be a discrete value able to distinguish between classes. Binary output is commonly used. To convert the real output obtained by Eq. (4) to an ultimate binary output Y, that distinguishes between two classes, Eq. (5) is used.

$$Y = \begin{cases} 0 & \text{if } y^O_1 < 0.5 \\ 1 & \text{if } y^O_1 \geq 0.5 \end{cases} \tag{5}$$

2.1. 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 is a real vector with a length equal to the sum of the total number of weights in **W** and biases in **B**, where each of its elements is a weight or a bias to be optimised. The fitness function for a solution S is shown in Eq. (6). It is calculated as the MSE of the output returned by the MLP using the configuration in S.

$$fitness(S) = MSE = \frac{1}{|\mathbf{L}|} \sum_{l=1}^{L} (y_l - y^O_{1,l})^2 \tag{6}$$

where $|\mathbf{L}|$ is the total amount of training samples, $y^O_{1,l}$ is the predicted real value in the range $[0, 1]$ for the lth training sample in the training set $\mathbf{L}$, obtained by Eq. (4). y_l is the expected binary output for the lth training sample.

MSE is frequently used to evaluate regressions. But, in the context of classifications, it provides a metric of how much error exist when the predicted float output $y^O_{1,l}$ (previous to convert it binary) approximates the expected binary output y_l. The idea is, for negative samples $y^O_{1,l}$ has to be near to 0 and for positive samples near to 1 [15].

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 [14,30].

2.2. 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.

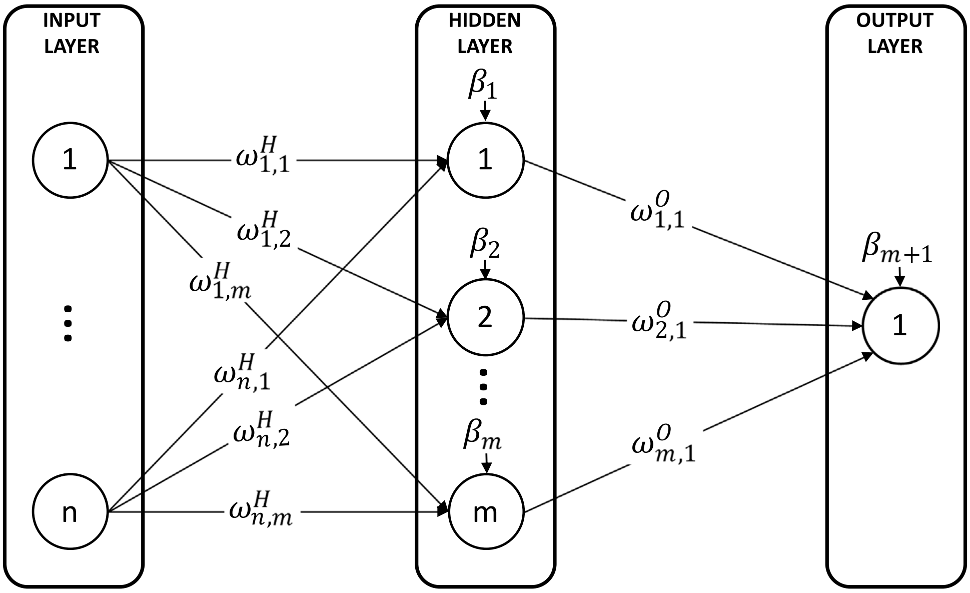


Fig. 1. Example of structure of a Multilayer Perceptron (MLP).

One of the most conventional training methods for MLP is Back Propagation (BP) [31]. 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].

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] that is based on a second-order minimisation method, Quasi-Newton Method [33], Gauss–Newton [34] or Levenberg–Marquardt [35] that is based on the approximation by least-squares [15].

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 periods 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]. Finally, conventional methods put aside biases, focusing just on the values of the weights [15,36].

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.

2.3. 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]. 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] uses the Biogeography-Based Optimisation 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] 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], 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 Optimiser (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 perform a better exploration and local-optima avoidance.

Mansouri et al. [41] implement a GWO hybridised to an Evolutionary 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] 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 content. 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], 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 better results would be achieved if a specific modification to the algorithm is proposed. Another bio-inspired algorithm in the literature is the proposal by Das et al. [44] 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] propose an Inertia Motivated Gray Wolf Optimisation 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 proposes to use velocity concepts similar to the PSO. The number of evaluations 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] compared the GA and the PSO in defining the initial weights and biases of an MLP for detecting thyroid functional 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] 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] proposes five different hybridisations between GA and PSO to train an MLP that classifies human glioma from molecular 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] authors compare eleven recently-proposed metaheuristics for training an ANN to classify fifteen different medical datasets. Algorithms included in the experiments were the Artificial Bee Colony (ABC), the Ant Lion Optimiser (ALO), the BBO, the Equilibrium Optimiser (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 evaluation 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] 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.

3. Cellular genetic algorithm

The Cellular Genetic Algorithm (CGA) [22] is a decentralised evolutionary algorithm based on the canonical Genetic Algorithm (GA) [51] 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 neighbourhoods, 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 individual 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. 2 presents the way in that CGA evolves a given individual (white circle). First, the neighbourhood of the individual to be evolved is obtained. In Fig. 2, 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 neighbourhood. 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. Three well-differentiated stages can explain the evolutionary process carried out:

- **Initialisation stage:** It involves lines 2 to 4. 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 to 15. If the stop condition is not reached (checked on line 5), 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), the selected individual is replaced if one of the generated individuals is fitter. This process is shown in Fig. 2 and is performed on all the individuals in the population. In line 13, 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.

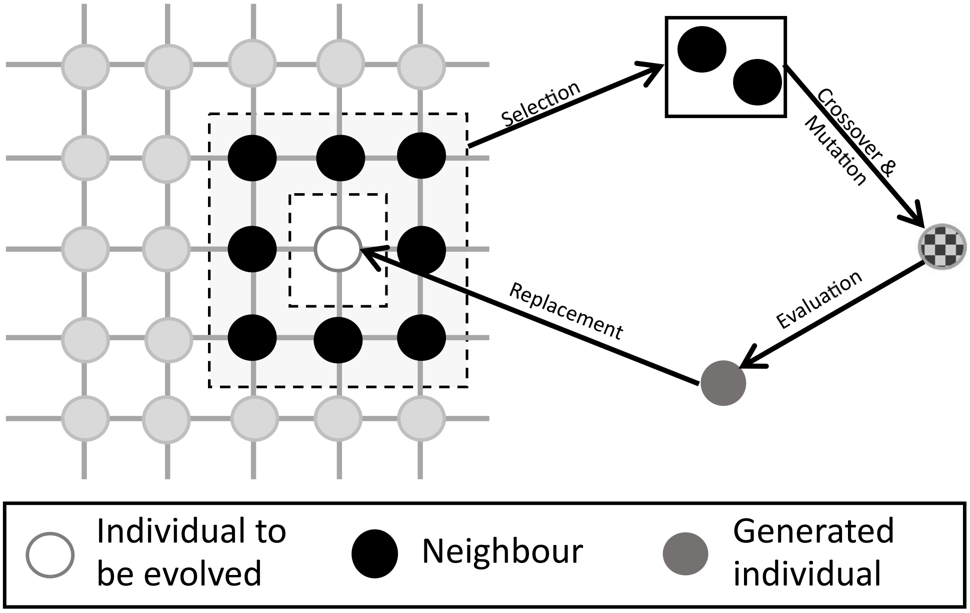


Fig. 2. Type of neighbourhood C9 and application of genetics operators in an CGA.

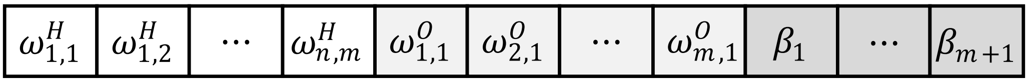


Fig. 3. Example of vector representation of an MLP structure.

- **Finalisation stage:** Involves lines 16 and 17. When the evolutionary 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, maxEvaluations)
2:     $population \leftarrow initialise(popSize)$
3:     $evaluate(population)$
4:     $t \leftarrow popSize$                          ▷ Evaluations counter
5:     while $t < maxEvaluations$ do
6:         for all $individual \in population$ do
7:             $neighbours \leftarrow getNeighbours(individual, population)$
8:             $parents \leftarrow selection(neighbours)$
9:             $offspring \leftarrow crossover(parents, crossoverRate)$
10:            $offspring \leftarrow mutation(offspring, mutationRate)$
11:            $evaluate(offspring)$
12:            $replace(individual, offspring)$
13:            $t \leftarrow t + 2$        ▷ Increased by the number of offsprings
14:        end for
15:    end while
16:    $bestIndividual \leftarrow getBestIndividual(population)$
17:    return $bestIndividual$
18: end function
```

3.1. Individual representation

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

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. 1, is presented in Fig. 3.

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 features 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].

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 ith elements of two offspring $Of1$ and $Of2$ by using Eqs. (7) and (8) respectively.

$$Of1_i = p1_i + inc_i \tag{7}$$

$$Of2_i = p2_i + inc_i \tag{8}$$

where $p1_i$ and $p2_i$ are the ith elements of the parents $p1$ and $p2$ respectively. inc is the increment performed over each element, calculated by Eq. (9).

$$inc_i = diff_i \times (1 + ratio) \tag{9}$$

$diff_i$ is the difference between the ith element of the best individual (BI_i) and the average of the ith element of parents. It is obtained by Eq. (10).

$$diff_i = BI_i - \frac{p1_i + p2_i}{2} \tag{10}$$

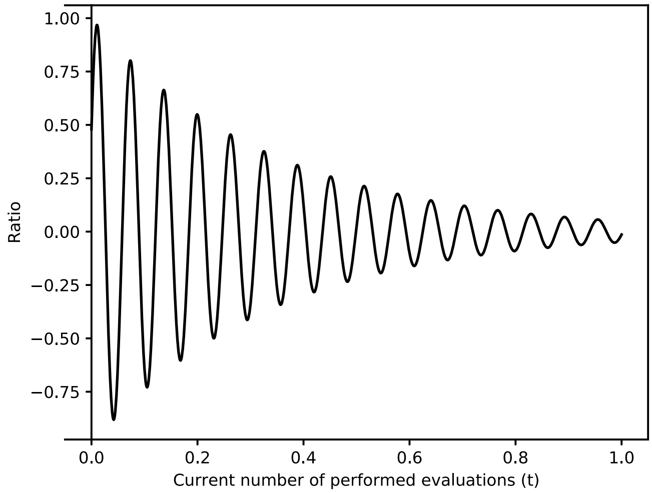


Fig. 4. Behaviour of damped harmonic oscillation function, presented in Eq. (11).

ratio is determined by Eq. (11). 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.

$$ratio = A \times e^{(-Ct)} \times \sin(P + 99.75 \times t) \tag{11}$$

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) configured with the mentioned values of A, C and P behaves like it is shown in Fig. 4, where a value of 0.0 in the x axis represents the initial step of the evolutionary process, and a value of 1.0 is when the process is completed. The values of A, C and P were selected to make the curve have a sustained decrease, thereby allowing precise variations at final evaluations but avoiding performing insignificant movements.

Both the *ratio* and the $diff_i$ functions aim to improve the exploitation of the DX by focusing on reaching the best individual and reducing the changes when the end of the evolutionary process is near. At the same time, $diff$ and *ratio* act as direction indicators as they make the new element increase or decrease the ith element of the parents according to the difference with the BI.

4. Experiments configuration

This section presents the experimental setup. It provides information 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, and state-of-the-art algorithms. Considered algorithms and their configurations are presented in Section 4.2.

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 [43,53]. 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 is configured with a unique hidden layer where the number of hidden neurons (m) is determined by Eq. (12), following the rule established in [14].

$$m = 2 \times n + 1 \tag{12}$$

Each weight and biases values are restricted to the interval [−1,1] according to the values handled in other approximations [30,54,55].

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 [14,37]. 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 [53–55]. 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] 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 highlighted in the corresponding tables. This work considers a confidence level of 99% (i.e., a significance level of $\alpha = 0.01$) for the statistical tests.

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

4.1. CGA operators

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

- **Adjusted Crossover (AX):** It was proposed by Yasojima et al. [58] 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 $Of1_i$ and $Of2_i$ of first and second offspring, are generated by Eqs. (13) and (14).

$$Of1_i = p1_i + \big((p1_i - p2_i) \times \alpha\big) \times g_i \tag{13}$$

$$Of2_i = p2_i + \big((p1_i - p2_i) \times \alpha\big) \times g_i \tag{14}$$

where $p1_i$ is the element i of the first parent, $p2_i$ is the element i of the second parent. g_i 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 g_i is −1. Finally, α is the weight of the crossover. In this paper, α is 0.02 as recommended in [58].

- **Simulated Binary Crossover (SBX):** It was proposed by Deb and Agrawal [59]. SBX simulates the single-point crossover of binary representations. The elements of the offspring $Of1_i$ and $Of2_i$ are obtained by Eq. (15) and Eq. (16) respectively.

$$Of1_i = 0.5 \times [(1 + \beta_q) \times p1_i + (1 - \beta_q) \times p2_i] \tag{15}$$

$$Of2_i = 0.5 \times [(1 - \beta_q) \times p1_i + (1 + \beta_q) \times p2_i] \tag{16}$$

[1] https://toko.uncu.edu.ar/.
[2] https://pythonhosted.org/neurolab/.

where $p1_i$ is the element i of the first parent, $p2_i$ is the element i of the second parent and β_q is an ordinate obtained by Eq. (17).

$$\beta_q = \begin{cases} (2 \times r)^{\frac{1}{\eta+1}} & \text{if } r \leq 0.5 \\ \left[\frac{1}{2\times(1-r)}\right]^{\frac{1}{\eta+1}} & \mathit{Otherwise} \end{cases} \tag{17}$$

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

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]. The objective of this operator is to avoid generating new elements randomly. The mutated value sol_i^{t+1} of the ith element of a solution in the evaluation t is generated by Eq. (18) according to a mutation probability.

$$sol_i^{t+1} = \begin{cases} sol_i^t + \vartriangle(t, ub - sol_i^t) & \text{if } r \leq 0.5 \\ sol_i^t + \vartriangle(t, sol_i^t - lb) & \mathit{Otherwise} \end{cases} \tag{18}$$

With lb and ub as the lower and the upper bounds of the element sol_i^t, a random number $r \in [0,1]$ and the function $\vartriangle(t,d)$ calculated by Eq. (19).

$$\vartriangle(t,d) = d \times \left(1 - r^{(1-\frac{t}{T})^b}\right) \tag{19}$$

T is the maximum number of evaluations, and b establishes the dependency on the evaluations number, set to 0.5. As the value of b is bigger, more disturbance is applied by the operator as evaluations are performed. Thus, small values of b mean that more precise movements will be made.

- **Uniform Mutation (UM):** In this case, the element sol_i^t is substituted by a mutated value sol_i^{t+1} obtained by Eq. (20).

$$sol_i^{t+1} = sol_i^t + (r - 0.5) \times u \tag{20}$$

With a random number $r \in [0,1]$ 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 proposed by Deb and Agrawal [62]. A polynomial probability distribution is used in this proposal to mutate each element of the individual (sol_i^t), taking into account the lower (lb) and upper (ub) bounds. Eq. (21) is used to obtain the new value sol_i^{t+1} for the ith element of the solution to mutate.

$$sol_i^{t+1} = \begin{cases} sol_i^t + \gamma_L \times (sol_i^t - lb) & \text{if } r \leq 0.5 \\ sol_i^t + \gamma_R \times (ub - sol_i^t) & \mathit{Otherwise} \end{cases} \tag{21}$$

where r is a random number belonging to the range $[0,1]$ and the functions $\gamma_L(d)$ and $\gamma_R(d)$ are calculated by Eqs. (22) and (23) respectively.

$$\gamma_L(r) = (2 \times d)^{\frac{1}{1+\eta}} \tag{22}$$

$$\gamma_R(r) = 1 - \left(2 \times (1-d)\right)^{\frac{1}{1+\eta}} \tag{23}$$

With η fixed to 5, which is a user-defined parameter that controls 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 sol_i^t by generating a completely new value between 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).

$$sol_i^{t+1} = lb + (ub - lb) \times r \tag{24}$$

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 mutation operators. The DX variations arise from combining the Damped Crossover, proposed in this paper, with the four mutation operators, resulting in the DX+NUM, DX+PM, DX+RM and DX+UM variations. The AX variations are the different combinations of the Adjusted Crossover, AX+NUM, AX+PM, AX+RM and AX+UM. Finally, the SBX variations combine the Simulated Binary Crossover with the four mutation operators, obtaining the SBX+NUM, SBX+PM, SBX+RM and SBX+UM.

4.2. Algorithms for comparison

Experiments compare the DX variations of the CGA against AX variations, SBX variations and state-of-the-arts approaches that represent different strategies to optimise weights and biases of the MLP. Evaluations focus on the convergence ability of each metaheuristic and the classification quality reached for the MLP configured with 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]. 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 on the proximity of the prey.
- **Cuckoo Search Algorithm (CS)** was proposed by Yang and Deb [64]. Cuckoo birds substitute eggs of other nests to their 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], the DE evolve a current individual by generating each element of a new individual considering a differential weight and the elements of three parents selected by a selection method. The new individual replaces the current one if it is better.
- **Genetic Algorithm (GA)** [51] 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] is a bio-inspired algorithm based on the hunting mechanism of the grey wolf and the leadership hierarchy of the herd. The algorithm establishes four types of grey wolves to represent the hierarchy. Three hunting behaviours are simulated in the optimisation process, search for prey, encircle the prey, and attack the prey.
- **Moth–Flame Optimisation (MFO)** [67] 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 distances. But, due to the artificial lights, moths get disoriented and are kept in circles around the light up to they die. MFO mimics this behaviour mathematically to perform optimisation.
- **Multi-Verse Optimisation (MVO)** was developed by Mirjalili et al. [68]. It performs optimisation by mathematically modelling 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]. PSO works imitating the behaviour of different organisms 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.

Table 1
Algorithms parameters configurations.

Algorithm	Parameter	Value
CGA (This paper)	Population size	10 × 10
	Neighbourhood	C9
	Crossover probability	0.9
	Mutation probability	0.01
	Selection operator	Binary tournament
BAT	Population size	50
	Loudness	0.5
	Pulse rate	0.5
	Frequency minimum	0
	Frequency maximum	1
CS	Number of nests	50
	Discovery rate	0.25
DE	Population size	50
	Crossover probability	0.9
	Differential weight	0.5
GA	Population size	50
	Crossover operator	SBX (Prob.: 0.9)
	Mutation operator	UM (Prob.: 0.01)
	Selection operator	Binary tournament
GWO	Population size	50
	$\hat{a}$	Decrease linearly from 2 to 0
MFO	Population size	50
	b	1
	t	$[-1, 1]$
MVO	Population size	50
	Min. wormhole existence prob.	0.2
	Max. wormhole existence prob.	1.0
PSO	Number of particles	50
	Inertia weight	0.721
	Cognitive component	1.193

Table 1 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 [30,55,70,71].

4.3. Datasets

Evaluations of the MLP optimised by metaheuristics were performed using five different medical datasets obtained from the UCI machine learning repository.[3] Each dataset is described in the following paragraphs and a summary is provided in Table 2.

- **Breast:** This dataset is composed of 699 instances, where each one corresponds to a patient submitted to surgery. Eight variables were measured, and the instances are distinguished between benign or malignant cases [72,73].
- **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].
- **Liver:** Instances come from blood tests performed over 345 male patients with apparent liver disorders by excessive alcohol consumption. Instances are split into positives and negative classes [75].
- **Parkinson:** This dataset was obtained from voice analysis performed over thirty-one patients. Specialists took almost 6 recordings for each individual. Recordings have 22 different metrics, and each one is classified as Parkinson's disease or normal [76].

Table 2
Summary of datasets features.

Dataset	Attributes	Instances	Class 1	Class 2
Breast	8	699	Benign (458)	Malignant (241)
Diabetes	8	768	Positive (268)	Negative (500)
Liver	6	345	Liver Disorders (200)	Normal (145)
Parkinson	22	195	Parkinson (147)	Healthy (48)
Vertebral	6	310	Abnormal(210)	Normal (100)

- **Vertebral:** This dataset contains 310 patients classified as abnormal (cases of disk hernia or spondylolisthesis) or normal. Each instance contains a set of six different variables [77].

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 method, calculated by Eq. (25).

$$A'_i = \frac{A_i - min_A}{max_A - min_A} \tag{25}$$

where A_i is the value for the feature A corresponding to the instance i, A'_i is the new value resulting from the normalisation, min_A and max_A are the minimum and the maximum value for the feature A respectively. This process is fundamental because it prevents that variables with big range values affect the other features of the dataset.

4.4. Classification measures

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 instances classified as positive, False Positives (FP) or negative instances classified as positive, True Negatives (TN) or negative instances classified 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).

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{26}$$

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).

$$Specificity = \frac{TN}{TN + FP} \tag{27}$$

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).

$$Sensitivity = \frac{TP}{TP + FN} \tag{28}$$

5. Experimental results

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.

[3] https://archive.ics.uci.edu/.

Table 3
Mean and standard deviation of the fitness quality indicator.

Algorithm	Breast	Diabetes	Liver	Parkinsons	Vertebral
BAT	$0.036_{0.002}$	$0.152_{0.002}$	$0.196_{0.006}$	$0.094_{0.007}$	$0.132_{0.004}$
CS	$0.053_{0.004}$	$0.171_{0.005}$	$0.211_{0.004}$	$0.220_{0.027}$	$0.148_{0.006}$
DE	$0.048_{0.002}$	$0.164_{0.004}$	$0.208_{0.004}$	$0.182_{0.018}$	$0.144_{0.003}$
GA	$0.036_{0.002}$	$0.153_{0.001}$	$0.199_{0.002}$	$0.104_{0.006}$	$0.133_{0.002}$
GWO	$0.035_{0.002}$	$0.153_{0.001}$	$0.200_{0.003}$	$0.092_{0.004}$	$0.134_{0.002}$
MFO	$0.035_{0.002}$	$0.153_{0.002}$	$0.195_{0.004}$	$0.107_{0.010}$	$0.131_{0.002}$
MVO	$0.030_{0.002}$	$0.147_{0.002}$	$\mathbf{0.176_{0.003}}$	$0.077_{0.007}$	$\mathbf{0.118_{0.003}}$
PSO	$0.030_{0.001}$	$0.148_{0.001}$	$0.186_{0.002}$	$\mathit{0.076_{0.005}}$	$0.125_{0.002}$
AX+NUM	$0.036_{0.002}$	$0.155_{0.002}$	$0.202_{0.002}$	$0.107_{0.007}$	$0.137_{0.003}$
AX+PM	$0.036_{0.002}$	$0.155_{0.002}$	$0.202_{0.002}$	$0.108_{0.007}$	$0.137_{0.002}$
AX+RM	$0.036_{0.002}$	$0.156_{0.002}$	$0.204_{0.002}$	$0.109_{0.007}$	$0.138_{0.003}$
AX+UM	$0.038_{0.002}$	$0.157_{0.003}$	$0.207_{0.004}$	$0.115_{0.013}$	$0.141_{0.004}$
SBX+NUM	$0.037_{0.002}$	$0.156_{0.001}$	$0.205_{0.003}$	$0.126_{0.008}$	$0.139_{0.002}$
SBX+PM	$0.037_{0.001}$	$0.156_{0.001}$	$0.205_{0.002}$	$0.127_{0.009}$	$0.139_{0.002}$
SBX+RM	$0.037_{0.001}$	$0.156_{0.001}$	$0.206_{0.002}$	$0.126_{0.009}$	$0.140_{0.002}$
SBX+UM	$0.037_{0.001}$	$0.155_{0.001}$	$0.206_{0.002}$	$0.117_{0.008}$	$0.139_{0.002}$
DX+NUM	$0.029_{0.002}$	$0.146_{0.002}$	$0.182_{0.002}$	$0.087_{0.006}$	$0.122_{0.002}$
DX+PM	$\mathit{0.028_{0.001}}$	$\mathit{0.146}_{0.001}$	$0.182_{0.002}$	$0.087_{0.006}$	$0.122_{0.002}$
DX+RM	$0.030_{0.002}$	$0.147_{0.001}$	$0.184_{0.002}$	$0.092_{0.005}$	$0.123_{0.002}$
DX+UM	$\mathbf{0.028_{0.001}}$	$\mathbf{0.145_{0.001}}$	$\mathit{0.181_{0.002}}$	$\mathbf{0.066_{0.004}}$	$\mathit{0.121_{0.001}}$

Table 4
Wilcoxon values of the fitness quality indicator (Breast, Diabetes, Liver, Parkinsons, Vertebral). A confidence level of 99% (a significance level $\alpha = 0.01$) was considered.

Algorithm	DX+NUM	DX+PM	DX+RM	DX+UM
BAT	△ △ △ △ △	△ △ △ △ △	△ △ △ – △	△ △ △ △ △
CS	△ △ △ △ △	△ △ △ △ △	△ △ △ △ △	△ △ △ △ △
DE	△ △ △ △ △	△ △ △ △ △	△ △ △ △ △	△ △ △ △ △
GA	△ △ △ △ △	△ △ △ △ △	△ △ △ △ △	△ △ △ △ △
GWO	△ △ △ △ △	△ △ △ △ △	△ △ △ – △	△ △ △ △ △
MFO	△ △ △ △ △	△ △ △ △ △	△ △ △ △ △	△ △ △ △ △
MVO	△ △ ◁ ◁ ◁	△ △ ◁ ◁ ◁	– – ◁ ◁ ◁	△ △ ◁ △ ◁
PSO	△ △ △ ◁ △	△ △ △ ◁ △	– – △ ◁ △	△ △ △ △ △
AX+NUM	△ △ △ △ △	△ △ △ △ △	△ △ △ △ △	△ △ △ △ △
AX+PM	△ △ △ △ △	△ △ △ △ △	△ △ △ △ △	△ △ △ △ △
AX+RM	△ △ △ △ △	△ △ △ △ △	△ △ △ △ △	△ △ △ △ △
AX+UM	△ △ △ △ △	△ △ △ △ △	△ △ △ △ △	△ △ △ △ △
SBX+NUM	△ △ △ △ △	△ △ △ △ △	△ △ △ △ △	△ △ △ △ △
SBX+PM	△ △ △ △ △	△ △ △ △ △	△ △ △ △ △	△ △ △ △ △
SBX+RM	△ △ △ △ △	△ △ △ △ △	△ △ △ △ △	△ △ △ △ △
SBX+UM	△ △ △ △ △	△ △ △ △ △	△ △ △ △ △	△ △ △ △ △
DX+NUM		– – – – –	– ◁ ◁ ◁ –	△ △ △ △ △
DX+PM			◁ ◁ ◁ ◁ –	△ △ – △ △
DX+RM				△ △ △ △ △

5.1. 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 3 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 training, as in liver and vertebral datasets. Results could be even worse so if those attributes are noisy. Beyond that, the DX+UM kept very near 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 4 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 of the comparison using each of the datasets, namely Breast Cancer, Diabetes, Liver, Parkinson and Vertebral. A leftward triangle (◁) means 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 differences 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.

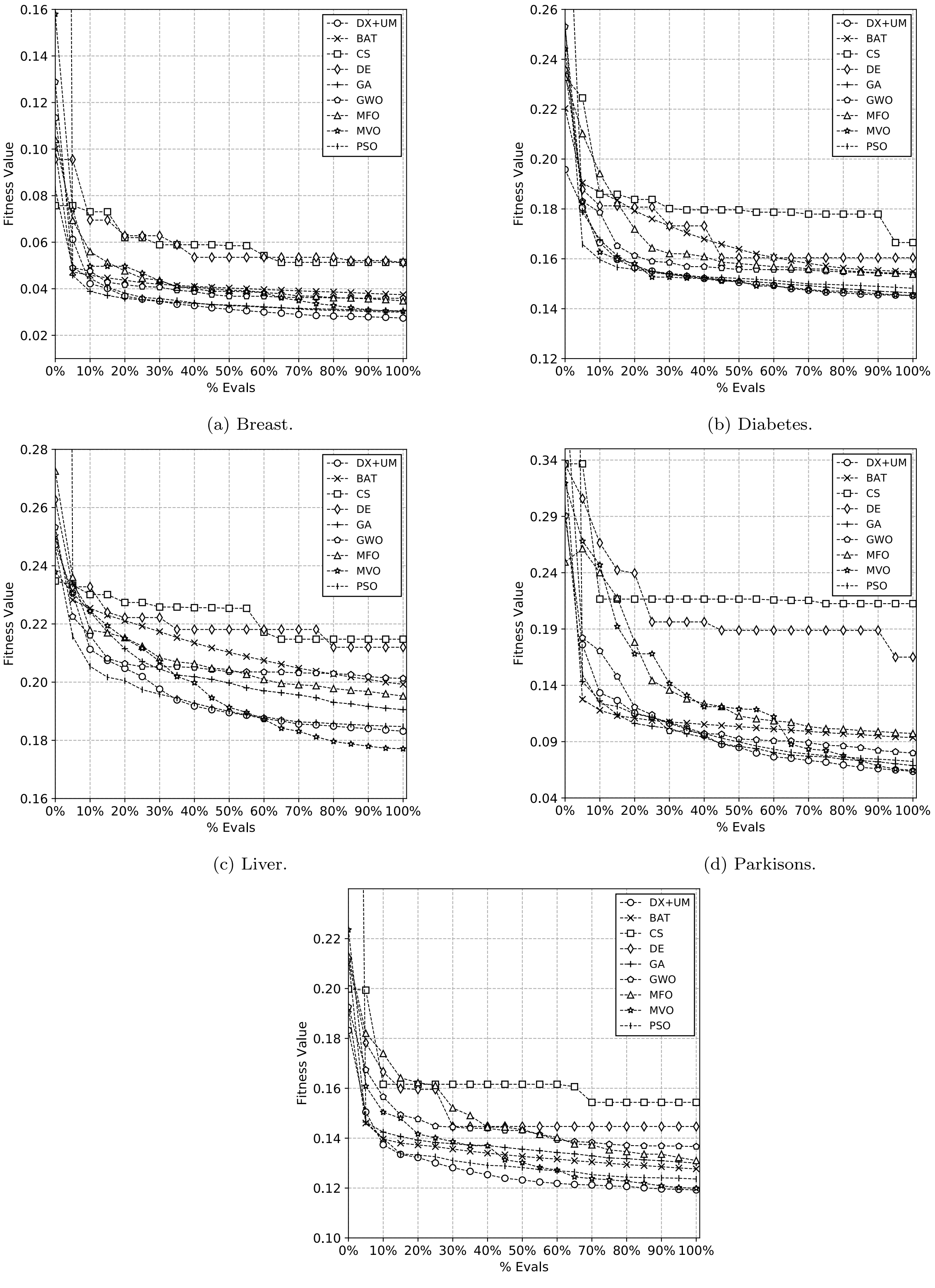
The Wilcoxon test results confirm the tendency shown in Table 4. The variation DX+UM has beaten all the other algorithms at fitness value, except for the liver and vertebral datasets where the MVO was better.

DX+NUM and DX+PM have shown a similar performance, overcoming the BAT, CS, DE, GA, GWO, MVO, and all the CGA versions using AX and SBX as crossover operators. The DX+RM performed a little worse 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 Figs. 5(a) to 5(e). Each plot shows how fitness is enhanced as the evolutionary process advance. As can be seen, the DX+UM converges 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. 5(c)). 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 5 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 following 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 optimal 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.



(a) Breast.

(b) Diabetes.

(c) Liver.

(d) Parkisons.

(e) Vertebral.

Fig. 5. Convergence curves of fitness value (MSE) of the five datasets.

5.2. 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 in classifying both classes or has had a preference for one of them. In general, both metrics should be near 1, which suggests a minor number 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	$670.085_{3.469}$	$731.070_{8.384}$	$326.235_{1.927}$	*$273.507_{5.402}$*	$296.553_{2.318}$
CS	$671.201_{3.404}$	$733.280_{10.611}$	$324.989_{1.397}$	**$253.168_{3.633}$**	$289.594_{10.815}$
DE	$679.606_{3.443}$	$741.785_{7.983}$	$332.502_{1.428}$	$339.117_{4.780}$	$297.432_{9.723}$
GA	$675.924_{4.141}$	$732.096_{10.269}$	$329.243_{6.266}$	$333.289_{5.370}$	$299.922_{4.780}$
GWO	$858.634_{7.135}$	$922.199_{8.422}$	$442.438_{2.194}$	$1503.959_{62.336}$	$409.900_{6.772}$
MFO	$694.284_{5.010}$	$755.436_{11.762}$	$339.792_{5.065}$	$389.837_{7.562}$	$311.231_{3.031}$
MVO	$688.638_{2.860}$	$751.552_{7.176}$	$337.846_{5.612}$	$426.726_{4.205}$	$306.335_{7.593}$
PSO	$684.385_{3.963}$	$739.880_{11.074}$	$332.498_{6.024}$	$392.356_{4.975}$	$302.303_{5.761}$
AX+NUM	$664.888_{24.317}$	$728.181_{29.326}$	$330.578_{5.546}$	$311.929_{1.758}$	$297.835_{6.342}$
AX+PM	$665.018_{23.633}$	$731.380_{26.913}$	$324.700_{10.418}$	$312.543_{1.231}$	$297.118_{7.528}$
AX+RM	$669.455_{22.040}$	$728.373_{28.904}$	$327.595_{8.761}$	$310.751_{1.378}$	$297.460_{6.379}$
AX+UM	$667.606_{21.882}$	$726.215_{25.157}$	$329.581_{6.725}$	$311.341_{1.342}$	$295.209_{8.767}$
SBX+NUM	$666.910_{23.772}$	$729.478_{24.990}$	$330.425_{6.960}$	$319.025_{2.191}$	$299.625_{5.178}$
SBX+PM	$672.327_{21.499}$	$728.706_{29.179}$	$330.332_{6.679}$	$318.361_{1.847}$	$299.104_{5.860}$
SBX+RM	$668.758_{20.317}$	$726.259_{26.142}$	$330.406_{6.855}$	$317.462_{1.934}$	$299.760_{5.351}$
SBX+UM	$664.909_{23.683}$	$725.253_{26.871}$	$330.301_{6.435}$	$318.381_{2.309}$	$298.648_{6.704}$
DX+NUM	**$410.336_{34.045}$**	**$435.478_{49.372}$**	**$217.544_{8.879}$**	$621.928_{88.362}$	**$199.389_{15.217}$**
DX+PM	*$425.737_{20.098}$*	*$454.128_{25.014}$*	*$219.989_{6.717}$*	$674.763_{67.467}$	*$203.691_{8.301}$*
DX+RM	$427.168_{19.789}$	$456.102_{27.252}$	$220.700_{5.058}$	$673.419_{64.273}$	$204.195_{8.599}$
DX+UM	$426.246_{24.491}$	$461.053_{35.099}$	$221.683_{4.913}$	$676.340_{64.095}$	$204.655_{8.437}$

Table 6
Mean and standard deviation of accuracy metric reached by each algorithm.

Algorithm	Breast	Diabetes	Liver	Parkinsons	Vertebral
BAT	$0.976_{0.005}$	$0.754_{0.007}$	$0.751_{0.018}$	$0.861_{0.017}$	*$0.875_{0.013}$*
CS	$0.957_{0.011}$	$0.724_{0.030}$	$0.691_{0.030}$	$0.699_{0.060}$	$0.807_{0.033}$
DE	$0.962_{0.006}$	$0.738_{0.021}$	$0.727_{0.028}$	$0.754_{0.060}$	$0.838_{0.031}$
GA	$0.973_{0.005}$	$0.751_{0.007}$	*$0.758_{0.016}$*	$0.848_{0.036}$	$0.873_{0.014}$
GWO	$0.978_{0.004}$	$0.751_{0.005}$	**$0.760_{0.019}$**	**$0.876_{0.021}$**	**$0.876_{0.010}$**
MFO	$0.976_{0.006}$	$0.752_{0.009}$	$0.757_{0.014}$	$0.842_{0.037}$	$0.871_{0.014}$
MVO	$0.973_{0.007}$	$0.757_{0.009}$	$0.728_{0.017}$	$0.854_{0.032}$	$0.869_{0.014}$
PSO	$0.978_{0.005}$	$0.753_{0.008}$	$0.753_{0.018}$	$0.867_{0.026}$	$0.873_{0.011}$
BP	$0.953_{0.011}$	$0.716_{0.026}$	$0.647_{0.043}$	$0.861_{0.059}$	$0.818_{0.036}$
AX+NUM	$0.974_{0.005}$	$0.747_{0.010}$	$0.744_{0.019}$	$0.832_{0.036}$	$0.864_{0.017}$
AX+PM	$0.975_{0.007}$	$0.750_{0.010}$	$0.753_{0.021}$	$0.832_{0.027}$	$0.872_{0.020}$
AX+RM	$0.975_{0.005}$	$0.747_{0.012}$	$0.740_{0.020}$	$0.846_{0.032}$	$0.868_{0.019}$
AX+UM	$0.971_{0.005}$	$0.744_{0.008}$	$0.731_{0.021}$	$0.824_{0.043}$	$0.847_{0.031}$
SBX+NUM	$0.971_{0.006}$	$0.747_{0.011}$	$0.743_{0.019}$	$0.816_{0.038}$	$0.857_{0.026}$
SBX+PM	$0.973_{0.005}$	$0.748_{0.013}$	$0.739_{0.019}$	$0.805_{0.049}$	$0.869_{0.023}$
SBX+RM	$0.975_{0.007}$	$0.751_{0.011}$	$0.740_{0.020}$	$0.802_{0.034}$	$0.861_{0.020}$
SBX+UM	$0.972_{0.006}$	$0.749_{0.010}$	$0.743_{0.016}$	$0.840_{0.026}$	$0.868_{0.020}$
DX+NUM	*$0.981_{0.004}$*	$0.754_{0.007}$	$0.748_{0.019}$	*$0.870_{0.021}$*	$0.873_{0.009}$
DX+PM	*$0.981_{0.005}$*	*$0.757_{0.011}$*	$0.751_{0.015}$	$0.863_{0.025}$	$0.873_{0.013}$
DX+RM	$0.980_{0.006}$	$0.754_{0.011}$	$0.752_{0.014}$	$0.860_{0.021}$	*$0.875_{0.015}$*
DX+UM	**$0.982_{0.005}$**	**$0.758_{0.010}$**	$0.749_{0.018}$	*$0.870_{0.025}$*	*$0.875_{0.016}$*

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

Table 6 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 Propagation 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 7 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 metrics, 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 samples, showing a high specificity value. At classifying positive cases, the best was the BAT algorithm, while the second-best was DX+RM.

For the liver dataset, all the algorithms showed to better classify positive samples (high sensitivity). The best algorithm for classifying negative samples was GWO which showed higher specificity values. The best with sensitivity values was the GA. DX+PM was the second-best in sensitivity values and was near to the best algorithm in specificity values.

Concerning the Parkinson dataset, the BP was better at classifying positive cases but kept under the other algorithms at classifying negative samples. The GWO was the second-best, obtaining the best specificity and a similar sensitivity value as the other metaheuristics. Among the DX variations, DX+NUM showed to be competitive at 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.

6. 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 improve 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. Comparisons are against state-of-the-art algorithms and CGA versions configured with well-known genetic operators. Two aspects were evaluated, 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 was the second-best in the other two datasets. The other DX variations showed to be very near to the results of the DX+UM and the state-of-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 comparable 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 algorithms.

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 – review & editing. **Ana Carolina Olivera:** Conceptualization, Methodology, Validation, Investigation, Formal analysis, Writing – original draft, Writing – review & editing. **Pablo Javier Vidal:** Conceptualization, Methodology, Validation, Investigation, Formal analysis, Writing - original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial 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, Argentina (Project No. 06/B081-B). Authors thank Consejo Nacional de Investigaciones Científicas *y* Técnicas, Argentina.

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/s41591-018-0300-7.

[2] Mintz Y, Brodie R. Introduction to artificial intelligence in medicine. Minim Invasive Ther Allied Technol 2019;28(2):73–81. http://dx.doi.org/10.1080/13645706.2019.1575882.

[3] 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.2020.06.040.

[4] Guan Q, Huang Y, Zhong Z, Zheng Z, Zheng L, Yang Y. Thorax disease classification with attention guided convolutional neural network. Pattern Recognit Lett 2020;131:38–45. http://dx.doi.org/10.1016/j.patrec.2019.11.040.

[5] Poudel S, Kim YJ, Vo DM, Lee S-W. Colorectal disease classification using efficiently scaled dilation in convolutional neural network. IEEE Access 2020;8:99227–38. http://dx.doi.org/10.1109/access.2020.2996770.

[6] 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/s12149-020-01542-y.

[7] Chu CS, Lee NP, Adeoye J, Thomson P, Choi S-W. Machine learning and treatment outcome prediction for oral cancer. J Oral Pathol Med 2020;49(10):977–85. http://dx.doi.org/10.1111/jop.13089.

[8] 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.

[9] 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.

[10] 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.

[11] Murtagh F. Multilayer perceptrons for classification and regression. Neurocomputing 1991;2(5–6):183–97. 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/10.1016/j.ins.2014.01.038.

[15] Ojha VK, Abraham A, Snášel V. Metaheuristic design of feedforward neural 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/10.1109/tevc.2019.2921598.

[17] Swan J, Adriaensen S, Brownlee AE, Hammond K, Johnson CG, Kheiri A, Krawiec 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.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.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/10.1007/s10462-011-9208-z.

[21] Such FP, Madhavan V, Conti E, Lehman J, Stanley KO, Clune J. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. 2018, arXiv:1712.06567. URL https://arxiv.org/abs/1712.06567.

[22] 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/10.1007/978-0-387-77610-1_1, Ch. 1.

[23] 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/10.1080/08839514.2019.1646005.

[24] Alba E, Dorronsoro B. Continuous optimization. In: Cellular genetic algorithms. Boston, MA: Springer US; 2008, p. 167–74. http://dx.doi.org/10.1007/978-0-387-77610-1_12, Ch. 12.

[25] Dorronsoro B, Alba E. A simple cellular genetic algorithm for continuous optimization. In: 2006 IEEE international conference on evolutionary computation. 2006, p. 2838–44. http://dx.doi.org/10.1109/CEC.2006.1688665.

[26] Tinós R. Artificial neural network based crossover for evolutionary algorithms. Appl Soft Comput 2020;95:106512. http://dx.doi.org/10.1016/J.ASOC.2020.106512.

[27] 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.

[28] 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/10.1007/978-1-4757-5362-2_10, Ch. 10.

[29] Krogh A. What are artificial neural networks? Nature Biotechnol 2008;26(2):195–7. http://dx.doi.org/10.1038/nbt1386.

[30] Heidari AA, Faris H, Aljarah I, Mirjalili S. An efficient hybrid multilayer perceptron neural network with grasshopper optimization. Soft Comput 2018;23(17):7941–58. http://dx.doi.org/10.1007/s00500-018-3424-2.

[31] Werbos PJ. Generalization of backpropagation with application to a recurrent gas market model. Neural Netw 1988;1(4):339–56. http://dx.doi.org/10.1016/0893-6080(88)90007-X.

[32] Hestenes MR, Stiefel E, et al. Methods of conjugate gradients for solving linear systems. J Res Natl Bur Stand 1952;49(6):409–36.

[33] Chen O-C, Sheu BJ. Optimization schemes for neural network training. In: Proceedings 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.

[34] Bertsekas DP. Nonlinear programming. J Oper Res Soc 1997;48(3):334. http://dx.doi.org/10.1057/palgrave.jors.2600425.

[35] Marquardt DW. An algorithm for least-squares estimation of nonlinear parameters. J Soc Ind Appl Math 1963;11(2):431–41, URL http://www.jstor.org/stable/2098941.

[36] Devikanniga D, Vetrivel K, Badrinath N. Review of meta-heuristic optimization 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.

[37] 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.

[38] 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.

[39] 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://dx.doi.org/10.1016/j.oceaneng.2020.108415.

[40] Jalali SMJ, Hedjam R, Khosravi A, Heidari AA, Mirjalili S, Nahavandi S. Autonomous robot navigation using moth-flame-based neuroevolution. In: Evolutionary machine learning techniques: Algorithms and applications. Singapore: Springer Singapore; 2020, p. 67–83. http://dx.doi.org/10.1007/978-981-32-9990-0_5, Ch. 5.

[41] 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/10.1080/1206212x.2018.1533613.

[42] 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/10.1080/19392699.2020.1768080.

[43] Jalali SMJ, Ahmadian S, Kebria PM, Khosravi A, Lim CP, Nahavandi S. Evolving 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://dx.doi.org/10.1007/978-3-030-36708-4_49.

[44] 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.

[45] 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.

[46] 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/10.5812/semj.100351.

[47] 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://dx.doi.org/10.3390/pr6050057.

[48] 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/j.cogsys.2019.06.003.

[49] Si T, Bagchi J, Miranda PB. Artificial neural network training using metaheuristics for medical data classification: An experimental study. Expert Syst Appl 2022;116423. http://dx.doi.org/10.1016/J.ESWA.2021.116423.

[50] 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.

[51] Holland JH. Genetic algorithms. Sci Am 1992;267(1):66–73. http://dx.doi.org/10.2307/24939139, URL http://www.jstor.org/stable/24939139.

[52] Kleppner D, Kolenkow R. The harmonic oscillator. In: An introduction to mechanics. 2. Cambridge: Cambridge University Press; 2013, p. 411–38. http://dx.doi.org/10.1017/cbo9781139013963.013, Ch. 13.

[53] 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.

[54] 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://dx.doi.org/10.1007/s00500-016-2442-1.

[55] 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/s10586-019-02913-5.

[56] Gibbons JD, Chakraborti S. The general two-sample problem. In: Nonparametric statistical inference. 6th ed.. Boca Raton: CRC Press; 2020, p. 247–300. http://dx.doi.org/10.1201/9781315110479-6, Ch. 6.

[57] 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.

[58] 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://dx.doi.org/10.1017/s026357471900016x.

[59] Deb K, Agrawal R. Simulated binary crossover for continuous search space. Complex Syst 1995;9.

[60] 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.

[61] Michalewicz Z. GAs: Selected topics. In: Genetic algorithms + Data structures=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.

[62] 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.

[63] Yang X-S, Gandomi AH. Bat algorithm: a novel approach for global engineering optimization. Eng Comput 2012;29(5):464–83. http://dx.doi.org/10.1108/02644401211235834.

[64] 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.org/10.1109/NABIC.2009.5393690.

[65] 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.

[66] 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.

[67] Mirjalili S. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. Knowl-Based Syst 2015;89:228–49. http://dx.doi.org/10.1016/j.knosys.2015.07.006.

[68] 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.

[69] Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of ICNN'95-International conference on neural networks, Vol. 4. 1995, p. 1942–8. http://dx.doi.org/10.1109/ICNN.1995.488968.

[70] 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/s10489-014-0645-7.

[71] Yamany W, Fawzy M, Tharwat A, Hassanien AE. Moth-flame optimization for training multi-layer perceptrons. In: 2015 11th international computer engineering conference (ICENCO). 2015, p. 267–72. http://dx.doi.org/10.1109/ICENCO.2015.7416360.

[72] 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/opre.43.4.570.

[73] Wolberg WH, Mangasarian OL. Multisurface method of pattern separation 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.

[74] 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].

[75] 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.2016.01.004.

[76] 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.

[77] 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/TLA.2009.5349049.

[78] Chu K. An introduction to sensitivity, specificity, predictive values and likelihood ratios. Emerg Med 1999;11:175–81. http://dx.doi.org/10.1046/J.1442-2026.1999.00041.X.