Training Multilayer Perceptrons with a Gaussian Artificial Immune System

Pablo A. D. Castro and Fernando J. Von Zuben
School of Electrical and Computer Engineering (FEEC)
University of Campinas (Unicamp)
Campinas, São Paulo, Brazil
{pablo,vonzuben}@dca.fee.unicamp.br

Abstract— **In this paper we apply an immune-inspired approach to train Multilayer Perceptrons (MLPs) for classification problems. Our proposal, called Gaussian Artificial Immune System (GAIS), is an estimation of distribution algorithm that replaces the traditional mutation and cloning operators with a probabilistic model, more specifically a Gaussian network, representing the joint distribution of promising solutions. Subsequently, GAIS utilizes this probabilistic model for sampling new solutions. Thus, the algorithm takes into account the relationships among the variables of the problem, avoiding the disruption of already obtained high-quality partial solutions (building blocks). Besides the capability to identify and manipulate building blocks, the algorithm maintains diversity in the population, performs multimodal optimization and adjusts the size of the population automatically according to the problem. These attributes are generally absent from alternative algorithms, and all were shown to be useful attributes when optimizing the weights of MLPs, thus guiding to high-performance classifiers. GAIS was evaluated in six well-known classification problems and its performance compares favorably with that produced by contenders, such as opt-aiNet, IDEA and PSO.**

I. Introduction

Artificial Neural Networks (ANN) are parameterized models for processing information in a distributive and parallel way. Among the several neural network models, the Multilayer Perceptron (MLP) is the most known and has been widely and successfully applied to regression and classification problems [1].

In an MLP, the weights are optimized to minimize an error function, such as Mean Square Error (MSE) between the output of the network and the target output. Generally, this task is accomplished iteratively by a gradient descent method. Despite its popularity, the gradient descent based method presents some limitations: it often gets trapped in a local minimum, being sensitive to the initial values attributed to the weights (free parameters of the neural networks), and it is not capable of dealing with non-differentiable error functions [2].

To overcome the drawbacks of the approaches over the last decades considerable efforts have been carried out on weight training by means of computational intelligence techniques, particularly in the field of bio-inspired algorithms such as genetic algorithms [3], ant colony optimization [4], particle swarm optimization [5], estimation of distribution algorithm [6] and artificial immune system [7]. These algorithms are population-based search strategies capable of dealing successfully with complex and large search spaces.

Among these appealing approaches, artificial immune systems (AISs) have received special attention due to their interesting features: (*i*) dynamical control of population size; (*ii*) efficient mechanism of exploration/exploitation of the search space, which allows to find and preserve the local optima as well as to insert and maintain diversity in the population [8] [9].

Despite their high performance as general problem solving tools, there are some shortcomings associated with these immune-inspired algorithms. Firstly, as the complexity and scale of the problem increase, the performance of the algorithms becomes more and more associated with a proper choice of the design parameters, such as mutation rate.

In addition, it is noticeable that, when the solution is represented by a vector of attributes, the population of candidate solutions may contain partial high-quality solutions to the problem, called building blocks [10]. With the preservation of building blocks, the chances of finding the global optimum and local optima increases over the time. However, the existing AISs suffer from the lack of ability to identify and effectively manipulate these partial solutions [11].

One way to allow AISs to handle building blocks effectively is to make the algorithm learn the relationships among the variables by using statistical information extracted from a set of promising solutions. From a conceptual point of view, the selected set of promising solutions can be viewed as a sample drawn from an unknown probability distribution. An estimation of this probability distribution would allow the immune algorithm to generate new solutions that are somehow similar to the ones contained in the original selected solutions [12]. Additionally, a probability distribution can effectively capture a building block structure of a problem, even without any prior information.

In this context, recently we have proposed a general framework to implement high-performance artificial immune systems which replaces the traditional mutation operator with a probabilistic model representing the probability distribution of the promising solutions found so far. Then the obtained probabilistic model is used to generate new solutions. Founded on this framework, some algorithms was developed and successfully applied to different optimization problems.

978-1-4244-7835-4/11/$26.00 ©2011 IEEE

For instance, the Bayesian Artificial Immune System (BAIS) [11] [13] [14] and Multi-objective Bayesian Artificial Immune System (MOBAIS) [15][16][17] for single and multi objective optimization in discrete domains, respectively, and the the Gaussian Artificial Immune System (GAIS) [18] and Multi-objective Gaussian Artificial Immune System (MO-GAIS) [19] for single and multi objective optimization in continuous domains, respectively. Besides the capability to deal with building blocks, these algorithms still preserve the aforementioned advantages of AISs.

The objective of this paper is to apply the GAIS algorithm to optimize the weights of MLPs, aiming at investigating its usefulness for accomplishing this task. The probabilistic model used by our algorithm is a Gaussian network, due to its capability to properly capture the most expressive interactions among the variables of the problem.

Experiments on six well-known classification problems have been carried out to evaluate the effectiveness of the proposed methodology when compared to other algorithms. The results have shown that GAIS is a competent algorithm with qualitative and quantitative advantages over the contenders.

This paper is organized as follows. In Section II, we describe the GAIS in details. Section III describes the application of GAIS to train weights of MLPs. The experimental results are outlined and analyzed in Section IV. Finally, in Section V we draw some concluding remarks and present the further steps of the research.

II. Gaussian Artificial Immune System

Recently, we proposed a novel immune-inspired algorithm for solving continuous optimization problems which has the mutation and cloning operators replaced by a probabilistic model in order to generate new antibodies. The probabilistic model used in our proposal is a Gaussian network due to its capability to properly represent the most expressive interactions among the variables. The algorithm, called Gaussian Artificial Immune System (GAIS), works with a population of potential solutions (antibodies) which can grows and shrinks according to the problem.

The pseudo-code of GAIS is presented in Algorithm 1.

Algorithm 1 Gaussian Artificial Immune System

Begin
  Initialize the population of antibodies;
while stopping condition is not met **do**
  Evaluate the population;
  Select the best antibodies;
  Build the Gaussian network at every p iterations;
  Sample new antibodies;
  Suppress antibodies with fitness lower than a threshold;
  Eliminate similar antibodies;
  Insert new antibodies randomly;
end while
End

In GAIS, the initial population is generated at random. From the current population, the best solutions are selected. A Gaussian network that properly fits the selected antibodies is constructed. A number of new antibodies sampled from the network are then inserted into the population and those similar ones and with lower fitness are eliminated. We computed similarity based on the vector of solutions. Next, few individuals are generated randomly and inserted into the population in order to favour diversity, yielding a better exploration of the search space.

Aiming at decreasing the computational cost of the algorithm and looking for a speed up in the execution time, the structure of the network is rebuilt at every p iterations ($p > 1$), whereas the redefinition of the probabilities remains at every iteration.

Besides the capability to maintain diversity in the population, GAIS also performs multimodal optimization, adjusts dynamically the size of the population according to the problem, and, most importantly, identifies and manipulates building blocks. As a direct consequence of these characteristics, the search process becomes more robust in the sense that there is no much variation in performance when the algorithm is run several time for the same problem. In addition, a considerable reduction in the number of evaluations of candidate solutions occurs until the algorithm reaches a certain level of performance.

Two aspects should receive special attention. The first is related to the way of building the Gaussian network from the selected individuals; and the other is how to use the network to generate new solutions. In the next subsections we explain in detail how to perform these two tasks.

A. Gaussian Network - Learning and Sampling

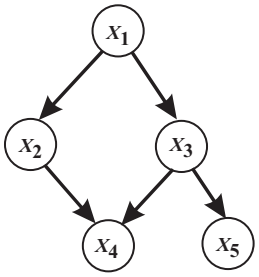
The two tasks, learning and sampling of a Gaussian network, are fundamental to the GAIS algorithm. Learning the Gaussian network for a given set of promising solutions corresponds to estimating their joint distribution. Sampling new instances according to the network guides to new candidate solutions to the problem.

Gaussian networks are probabilistic graphical models often used to estimate probabilistic dependencies among variables belonging to continuous domains [20]. Formally, a Gaussian network for a set of variables $X = \{X_1, X_2, ..., X_n\}$ is a directed acyclic graph whose nodes are variables of the problem and the edges indicate relationships of dependence among the connected variables. In Figure 1, there is an example of a Gaussian network for five variables.

The Gaussian network receives this name because the joint probability for X is represented by an n-dimensional Gaussian distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$:

$$f(x) = (2\pi)^{-n/2}|\boldsymbol{\Sigma}|^{-1/2}e^{-1/2(x-\boldsymbol{\mu})^t\boldsymbol{\Sigma}^{-1}(x-\boldsymbol{\mu})}, \quad (1)$$

where $|\boldsymbol{\Sigma}|$ is the determinant of the covariance matrix and $\boldsymbol{\Sigma^{-1}}$ is the inverse of the covariance matrix, also called precision matrix and denoted by W.



$f(x_1x_2x_3x_4x_5) = f(x_1)f(x_2|x_1)f(x_3|x_1)f(x_4|x_2x_3)f(x_5|x_3)$

Fig. 1. Example of a Gaussian network for 5 variables and the joint probability

For instance, if there is an edge from node X_1 to node X_2, we say that variable X_1 is parent of variable X_2 and, therefore, the value of X_2 is in a conditional dependence on the value of X_1. If a variable X_i has a set of parents, denoted here by $\boldsymbol{pa_i}$, its probability distribution is characterized by a conditional probability density function (pdf) and it is expressed by $f(X_i|\boldsymbol{pa_i})$. On the other hand, the probability distribution of a variable X_j which has no parents is expressed by its unconditional pdf, $f(X_j)$. In this sense, the joint probability distribution for X can be re-written as a product of conditional pdf, each of which belongs to an independent Gaussian distribution. That is:

$$f(x) = \prod_{i=1}^{n} f(x_i|\boldsymbol{pa_i}), \tag{2}$$

where

$$f(x_i|\boldsymbol{pa_i}) = \mathcal{N}(\mu_i + \sum_{x_k \in \boldsymbol{pa_i}} b_{ki}(x_k - \mu_k), \sigma_i^2), \tag{3}$$

where μ_i is the mean of X_i, σ_i^2 is the variance of X_i conditioned to the parents of X_i and b_{ki} is a linear coefficient reflecting the strength of the relationship between X_k and X_i. If $b_{ki} = 0$, then there is no relationship between the variables X_k and X_i.

The transformation of b_{ki} and σ_i^2 of the Gaussian network into the precision matrix W of the equivalent Gaussian distribution is achieved by the following recursive formula [21]:

$$W(i+1) = \begin{pmatrix} W(i) + \frac{b_{i+1}b_{i+1}^t}{\sigma_{i+1}^2} & \frac{-b_{i+1}}{\sigma_{i+1}^2} \\ \frac{-b_{i+1}^t}{\sigma_{i+1}^2} & \frac{1}{\sigma_{i+1}^2}, \end{pmatrix} \tag{4}$$

where $W(i)$ is the $i \times i$ upper left submatrix of W, $W(1) = \frac{1}{\sigma_i^2}$, b_i is the column vector $(b_{1i}, ..., b_{(i-1)i})^t$ and b_i^t is the transposed vector.

- **Learning**

The Gaussian network learning from a dataset can be stated as follows. Given a collection of observed data, find the network model to explain these data with maximum likelihood. By finding the network we mean to provide the structure of the graph, as well as the probability distribution of each variable that best fits the data.

One usual approach to this task is to adopt a procedure for searching the space of all possible candidate network structures, given a metric that can provide a relative score to each point in the space. The heuristic search utilized by GAIS begins with an empty network, i.e. with no edges. Next, the probability distribution of each variable is estimated using the dataset and the score of the network is computed. The search process proposes small changes to the structure in order to obtain a network with higher score than the previous one. These small changes can be accomplished by adding or deleting an edge, or reversing the edge direction. Every time a change is made it is necessary to compute the probability distribution of the variables for the modified network. Regarding the scoring metrics, GAIS utilizes the *Bayesian Information Criterion* (BIC) for Gaussian networks [22]:

$$BIC = p(D|G) - g(N) * dim(G) \tag{5}$$

where $p(D|G) =$

$$\prod_{l=1}^{N} \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi v_i}} e^{-1/2v_i(x_{li} - m_i - \sum_{x_k \in \boldsymbol{pa_i}} b_{ki}(x_{lk} - m_k))^2}, \tag{6}$$

D represents the dataset, G is the network under evaluation, N is the number of instances, n is the number of variables, b_{ki} is the linear coefficient of relationship between X_k and X_i, $\boldsymbol{pa_i}$ is the set of parents of X_i and v_i is the conditional variance of X_i given $X_1, ..., X_{i-1}$ $\forall i, k$. The function $g(N) = \frac{1}{2}\ ln\ N$ is responsible for penalizing complex models and $dim(G) = 2n + \sum_{i=1}^{n} \boldsymbol{pa_i}$ is the number of parameters to be estimated.

- **Sampling**

Once the Gaussian network is built, we can generate new instances using the joint probability distribution encoded by the network (Eq. 1). To accomplish this task, we utilize a method that finds an ancestral ordering of the nodes in the Gaussian network and instantiates one variable at a time in a forward manner, that is, a variable is not sampled until all its parents have already been sampled [23]. In GAIS, the number of sampled instances varies with the problem.

III. GAIS Applied to Learn Weights of MLPs

In this section we will show that GAIS is a promising tool for training weights of MLPs. The remaining of this paper is restricted to classification problems and the topologies of the MLPs are restricted to single hidden layer fully connected.

Next, we describe the codification scheme, the fitness function and the suppression mechanism of GAIS.

- **Coding**

As the MLP error surface is a function of the weights and biases, training the network becomes the problem of searching for the most suitable weight set that minimizes

the error. Therefore, each antibody (solution vector) will correspond to the weight vector of the whole network, including the bias terms for the hidden and output neurons. As a consequence, the antibodies are real-coded vectors.

- **Initial Population**

To initialize the population, all antibodies are coded using a uniformly distribution in the range $[-0.5, 0.5]$.

- **Fitness Function**

The fitness function is defined based on the performance of the neural networks, calculated as a function of the number of training patterns correctly classified. The fitness function is expressed by:

$$Fit(Ab_i) = \frac{NPCC(Ab_i)}{NP} \tag{7}$$

where $NPCC(Ab_i)$ is the Number of Patterns Correctly Classified by the network coded in the antibody Ab_i and NP is the total number of instances in the training set.

- **Suppression**

In this phase, similar antibodies are eliminated in order to avoid redundancy and thus maintain diversity.

The preliminary tests showed that the suppression mechanism depends on the dimension of the problem to be solved, i.e. the number of weights and biases to be adjusted. This is because the suppression mechanism considers the Euclidean distance among solutions and, for normalized vectors, the greater the number of dimensions, the larger the distance may be. As the problems to be studied here require varying neural network dimensions, it is relevant to make suppression independent of this parameter. Thus, the following steps are proposed to perform suppression:

a) The weight vectors are normalized in the [0,1] interval.

b) The maximal distance between the normalized solutions is determined.

c) The distance matrix among solutions is divided by the maximal distance.

Antibodies with Euclidean distance smaller than ϵ are removed from the population.

IV. Experimental Results

In this section we evaluate the proposed methodology by applying it to six classification problems widely used in machine learning tasks. Next we describe the datasets, the algorithms used for comparison and then present and discuss the obtained results.

A. Description of datasets

During the experiments, we considered six well-known benchmarks from the UCI Repository of Machine Learning Databases [24]: Iris, Ionosphere, Pima, Sonar, Wine and Glass. Table I summarizes the characteristics of each one, giving the total number of instances, the number of attributes, the number of classes and the architecture of the networks (number of inputs, number of hidden neurons and number of outputs). The number of hidden neurons was defined based on preliminary experiments considering each dataset.

TABLE I
Characteristic of the datasets

Dataset	# Instances	# Attributes	# Classes	Architecture
Iris	150	4	3	4:2:3
Ionosphere	350	34	2	34:3:2
Pima	768	8	2	8:3:2
Sonar	208	60	2	60:5:2
Wine	178	13	3	13:2:3
Glass	214	9	6	9:5:6

The attributes of all data were normalized within the $[-1, 1]$ interval.

B. Methodology

The parameters utilized during the experiments are the same for the six problems. The initial population of GAIS contains 50 antibodies and the stopping condition of the algorithm is the maximum number of generations, defined here as 500.

We have compared the GAIS algorithm with three alternative approaches: opt-aiNet, IDEA and PSO.

The first one is the Artificial Immune Network for Optimization (opt-aiNet) [25]. It is an immune-inspired approach which evolves the antibodies utilizing the clonal selection principle and immunological network theory. The initial population contains 100 antibodies and each one generates 10 clones per iteration. The stopping condition is the maximum number of iterations, defined as 500.

The other algorithm considered for comparative analysis is the Iterated Density Estimation Algorithm (IDEA) [26], that also utilizes a probabilistic model, particularly a Gaussian mixture model. IDEA applies the *Leader* clustering algorithm [27] to generate the components of the mixture. The probability density function (pdf) of each component is encoded by a Gaussian network. The number of clusters was defined as 3 (so, 3 components of the mixture) and the population size was equal to 500. Again, the stopping condition is the maximum number of iterations, defined as 500.

For GAIS and IDEA, the Gaussian network is built at every 10 iterations, but the probabilities of the network are updated at every iteration. In order to penalize the complexity of the model, we have imposed a constraint on the number of parents a node can have. It corresponds to a maximal order of interactions that can be covered and it directly influences the complexity of the model. By our previous experience on Gaussian network learning, we know that when the complexity of the network is too high, it is more likely to detect spurious correlations on the data. Thus, each variable can have only two parents. For GAIS, 80% of the best solutions are utilized to build the probabilistic model. The number of samples generated is half of the size of the current population. The number of random individuals inserted into the population in order to create diversity is around 3% of the current population size.

Notice that opt-aiNEt is an immune-inspired algorithm not endowed with probabilistic models, and IDEA uses probabilistic models but are not endowed with the search capabilities of immune-inspired algorithms. The authors believe that using these two algorithms as contenders is justifiable in this inaugural paper, because it allows two reasonable contrasts: (*i*) immune-inspired against non-immune-inspired Gaussian algorithms; (*ii*) Gaussian against non-Gaussian immune-inspired algorithms.

Finally, the Particle Swarm Optimization (PSO) is quite simple and can be performed using the following steps: (*i*) generate some particles randomly distributed over the search space; (*ii*) generate a velocity vector for each particle. These vectors describe the movement of the particles in the space; and (*iii*) update the velocity vector. The neighborhood is used to define how the particles influence one another. In the present implementation the interactions of particles is global, in a star-like neighborhood. The velocity of the particle, denoted by v is updated taking into account the cognitive and social terms:

$$v_i(t+1) = \omega.v_i(t)+\varphi_1\times(p_i(t)-w_i(t))+\varphi_2\times(p_g(t)-w_i(t)),$$

where ω is the inertia weight that controls the convergence of the particles, $p_i(t)$ is the vector containing the best position (i.e., the best position that led to the best performance) of the particle, $p_g(t)$ is the vector containing the best position among all particles with a predefined neighborhood, φ_1 is the stochastic vector that will weigh the influence of the cognitive components, and φ_2 is the weight vector that will weigh the influence of the social component. Thus, particle i moves in a direction that is the weighted sum between its best position so far and the best position of other socially interacting particles. Here, $\omega = 0.730$, φ_1 and φ_2=2.05. The stopping condition is also the maximum number of iterations, established as 500.

The parameters of the algorithms were adjusted by preliminary experiments. All algorithms utilized the same topology for the neural networks, as specified in the 5th column of Table I.

C. Results

The k-fold cross-validation method with k = 10 was used in the experiments. The k-fold cross-validation consists of dividing the data set into k folders, training the network with the data from k-1 folders and testing it with the data from the remaining folder not used for training. After that, the average classification errors are calculated. Table II presents the average taken in five executions of the 10-fold cross-validation (thus, the average of 50 values) of the neural networks' performance for the test dataset.

To evaluate the statistical significance of the difference among the performance of the algorithms, the *t-test* was employed with a significance level of 95% (α=0.05). The symbol $\star$ indicates that there is a statistical difference between the results obtained by GAIS and the compared algorithms.

TABLE II
NEURAL NETWORKS' PERFORMANCE ON TEST DATASETS

Dataset	GAIS	opt-aiNet	IDEA	PSO
Iris	**97.4 ± 0.6**	96.1 ± 0.4 $\star$	97.2 ± 0.8	95.4 ± 0.8 $\star$
Ionosphere	**91.7 ± 0.9**	89.6 ± 1.0 $\star$	90.4 ± 0.6 $\star$	87.1 ± 0.9 $\star$
Pima	**76.3 ± 1.2**	74.7 ± 1.3 $\star$	73.9 ± 1.2 $\star$	73.3 ± 1.4 $\star$
Sonar	**81.6 ± 1.4**	79.2 ± 1.2 $\star$	80.2 ± 1.1 $\star$	78.7 ± 1.1 $\star$
Wine	**98.3 ± 0.6**	97.9 ± 0.5 $\star$	**98.3 ± 0.8**	96.1 ± 0.9 $\star$
Glass	**66.2 ± 1.1**	64.3 ± 0.9 $\star$	64.9 ± 0.7 $\star$	62.8 ± 0.8 $\star$

We can see from Table II that all the algorithms were able to generate neural networks with high generalization capability in the six classification tasks, with a slight advantage for GAIS.

When compared with opt-aiNet and IDEA, GAIS has produced a superior performance in all cases due to its efficient mechanism to explore/exploit the search space. The opt-aiNet, with its mutation operator, cannot capture the relationships among the variables. Besides, it is more likely to disrupt partial solutions found occasionally. On the other hand, IDEA can deal with building blocks but there are some difficulties in applying it effectively. One difficulty is related to the loss of diversity in the population, leading the algorithm to get stuck at local minima easily.

The PSO was the algorithm that presented worst performance in all cases. This was already expected because PSO have not an efficient mechanism to maintain diversity in the population and it is not able to manipulate building blocks effectively.

We also present the boxplot comparing the four algorithms for all datasets. The graphical results can be observed in Figures 2, 3, 4, 5, 6 and 7.

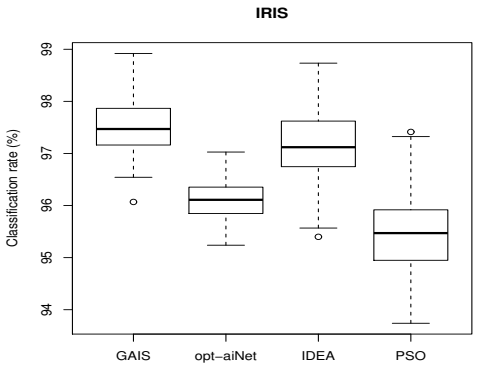


Fig. 2. Boxplot for the Iris dataset

From the perspective of the search process, it is interesting to notice an important characteristic of GAIS: the population size grows and shrinks dynamically along the search, allowing a more efficient exploration/exploitation of the search space. This particular characteristic plays an important role

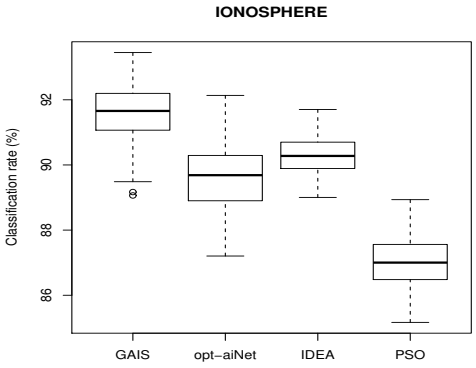


Fig. 3. Boxplot for the Ionosphere dataset

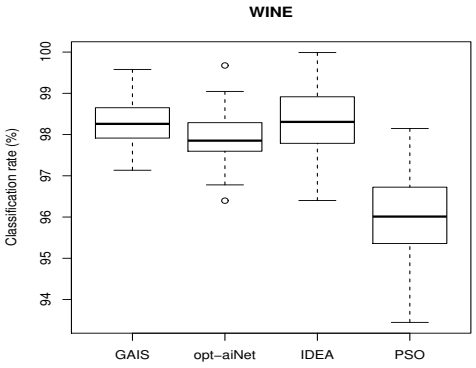


Fig. 6. Boxplot for the Wine dataset

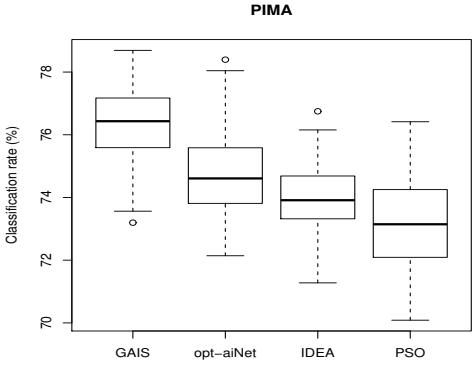


Fig. 4. Boxplot for the Pima dataset

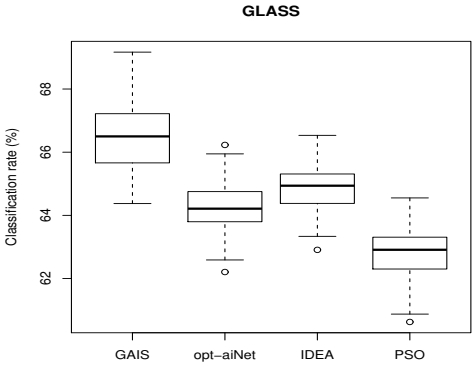


Fig. 7. Boxplot for the Glass dataset

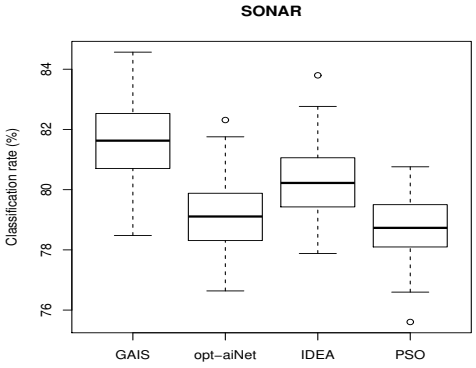


Fig. 5. Boxplot for the Sonar dataset

on the algorithm's capability to perform multimodal search. During the experiments, we could observe that GAIS is able to generate not just one but several different high-quality neural networks in a single run.

We take the best four neural networks generated in a random execution of GAIS. The results are presented in Tables III and IV. Notice that these four networks have different weight vectors, since the suppression mechanism eliminates similar networks.

TABLE III
RESULTS PRODUCED BY FOUR DIFFERENT NEURAL NETWORKS OBTAINED IN A SINGLE RUN FOR GAIS, FOR THE IRIS, IONOSPHERE AND PIMA DATA

Classifier	Iris	Ionosphere	Pima
Neural network 1	98.6 ± 0.9	94.3 ± 0.8	78.2 ± 1.3
Neural network 2	98.2 ± 1.1	94.1 ± 1.1	76.6 ± 0.9
Neural network 3	97.6 ± 0.7	92.4 ± 1.3	76.8 ± 1.1
Neural network 4	95.1 ± 1.2	91.1 ± 0.8	75.2 ± 0.7

In addition, we have also observed that GAIS found high-quality solutions in a short number of iterations in all the cases. This indicates that identifying and preserving building blocks, together with the capability to perform multimodal search, make the algorithm to converge quickly.

TABLE IV
RESULTS PRODUCED BY FOUR DIFFERENT NEURAL NETWORKS OBTAINED IN A SINGLE RUN FOR GAIS, FOR THE SONAR, WINE AND GLASS DATA

Classifier	Sonar	Wine	Glass
Neural network 1	84.1 $\pm$ 0.9	99.3 $\pm$ 1.1	68.7 $\pm$ 1.3
Neural network 2	83.6 $\pm$ 1.2	97.8 $\pm$ 0.8	67.2 $\pm$ 1.1
Neural network 3	83.0 $\pm$ 1.3	97.3 $\pm$ 0.8	65.8 $\pm$ 0.9
Neural network 4	81.2 $\pm$ 0.8	97.2 $\pm$ 0.9	65.1 $\pm$ 1.1

D. Discussion

As stated before, GAIS offers conceptual advantages over the contenders and they are described in what follows.

The first advantage is related to the effective mechanism of GAIS to perform a multimodal search, finding diverse high-quality local optima quickly. In addition, the initial population size is not crucial to GAIS due to its capability to control the population size along the search process in response to the particularities of the problem.

Other aspect to be highlighted is the effective maintenance of building blocks. With this mechanism, GAIS avoids disrupting the partial solutions found so far, leading to a great improvement in the quality of the candidate solutions even in the first iterations. While GAIS has found high-quality solutions in few generations, the other methodologies needed more generations to achieve a similar performance.

Moreover, with the absence of cloning and mutation operators, we do not have to worry about the proper choice of parameter values. The same does not occur with opt-aiNet, since we need to specify parameter values such as mutation rate, population size in order to avoid premature convergence, and number of clones.

Regarding the implementation of GAIS, we notice that the algorithm does not require a large amount of computational resources [28]. Although a Gaussian network has to be produced at every p iterations, the proposed methodology still preserves the computational tractability due to the restriction of at most two parents for each node in the network.

It is difficult to present here a comparison between the computational time required by the algorithms, since they were implemented using different languages and they were executed in different computers. GAIS was written using the C++ language and each algorithm was executed in a different computer, all with processor Intel Core 2 Duo. However, roughly comparing the computational cost of GAIS, opt-aiNet, IDEA and PSO, we could observe that GAIS requires much less fitness evaluations than the other algorithms, and thus tends to produce a slightly better execution time.

Nonetheless, we are currently investigating some enhancements to be incorporated into the algorithm in order to alleviate the computational cost to design the Gaussian network, such as parallelization and incremental learning of the network.

V. CONCLUSIONS

In this paper, we have applied the Gaussian Artificial Immune System (GAIS) for optimizing the weights of MLPs. Our algorithm replaces the traditional mutation and cloning operators with a probabilistic model representing the joint distribution of promising solutions and, subsequently, utilizes this model for sampling new solutions. The probabilistic model used is a Gaussian network due to its capability to properly capture the most relevant interactions among the variables of the problem.

Experiments on six datasets have demonstrated the effectiveness of the algorithm in designing accurate network classifiers. When compared with other approaches reported in the literature, our algorithm presented better results, in terms of performance of the obtained models.

Although promising results have been obtained, we are currently investigating some aspects that can be further improved, such as alternative metrics for evaluating the Gaussian networks and other algorithms for sampling new individuals. We are also analyzing the performance of GAIS when applied to synthesize neural networks for regression problems. Furthermore, since the GAIS algorithm generates several local optima neural networks, we intend to build a committee of networks, where the overall classification of a new sample is the combined individual classification of many networks.

ACKNOWLEDGMENT

The authors would like to thank CNPq for the financial support.

REFERENCES

[1] S. Haykin, "Neural Networks: A Comprehensive Foundation", 2nd edition, Prentice Hall PTR, 1998.

[2] X. Yao, "Evolving Artificial Neural Networks", *Proc. IEEE*, vol. 87, pp. 1423-1447, 1999.

[3] J. Bao, B. Zhou and Y. Yan, "A Genetic-Algorithm-Based Weight Discretization Paradigm for Neural Networks", *Proc. World Congress on Computer Science and Information Engineering*, pp. 655-659, 2009.

[4] C. Blum and K. Socha, "Training feed-forward neural networks with ant colony optimization: An application to pattern classification", *Proc. 5th Intern. Conf. on Hybrid Intelligent Systems(HIS-2005)*, pp. 10-15, 2005.

[5] L. A. Teixeira, F.T.G. Oliveira, A.L.I. Oliveira and C.J.A. Bastos Filho, "Adjusting Weights and Architecture of Neural Networks through PSO with Time-Varying Parameters and Early Stopping", Brazilian Symposium on Neural Networks, pp. 33-38, 2008.

[6] Y. Chen, Y. Zhang and A. Abraham, "Estimation of distribution algorithm for optimization of neural networks for intrusion detection system", *Proc. 8th Conference on Artificial Intelligence and Soft Computing*, pp. 9-18, 2006.

[7] R. Pasti and L.N. de Castro, "Bio-inspired and gradient-based algorithms to train MLPs: The influence of diversity", *I*nformation Sciences, vol. 179, no. 10, pp. 1441-1453, 2009.

[8] J. Timmis and A. Hone and T. Stibor and E. Clark, "Theoretical advances in artificial immune systems", *Theoretical Computer Science*, 403(1), pp. 11-32, 2008.

[9] L.N. de Castro and F. J. Von Zuben, "Learning and optimization using the clonal selection principle," *IEEE Trans. Evolutionary Computation*, 6(3), pp. 239–251, 2002.

[10] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, MIT Press, 1992.

[11] P. A. D. Castro and F. J. Von Zuben, "BAIS: A Bayesian Artificial Immune System for the effective handling of building blocks," *Information Sciences*, 179(10), pp. 1426 – 1440, 2009.
[12] M. Pelikan and D. Goldberg and F. Lobo, "A survey of optimization by building and using probabilistic models," *Comput. Optim. Appl.*, 21(1), pp. 5–20, 2002.
[13] P. A. D. Castro and F. J. Von Zuben, "Learning Ensembles of Neural Networks by Means of a Bayesian Artificial Immune System", *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 304-316, 2011.
[14] P. A. D. Castro and F. J. Von Zuben, "Feature subset selection by means of a Bayesian artificial immune system", *Proc. 8th Int. Conf. on Hybrid Intelligent Systems*, pp. 561–566, 2008.
[15] P. A. D. Castro and F. J. Von Zuben, "MOBAIS: A Bayesian artificial immune system for multi-objective optimization," Lecture Notes in Computer Science, 5132, pp. 48–59, Springer, 2008.
[16] P. A. D. Castro and F. J. Von Zuben,"Multi-objective Bayesian artificial immune system: Empirical evaluation and comparative analyses," *Journal of Mathematical Modelling and Algorithms*, 1, pp. 151–173, 2009.
[17] P. A. D. Castro and F. J. Von Zuben, "Multi-objective feature selection using a Bayesian artificial immune system," *Journal of Intelligent Computing and Cybernetics*, 3(2), pp. 235-256, 2010.
[18] P. A. D. Castro and F. J. Von Zuben, "GAIS: A Gaussian Artificial Immune System for Continuous Optimization," *Proc. 9th Int. Conf. on Artificial Immune Systems*, 2010.
[19] P. A. D. Castro and F. J. Von Zuben, "A Gaussian Artificial Immune System for Multi-Objective Optimization in Continuous Domains", *Proc. 10th International Conference on Hybrid Intelligent Systems*, pp. 159-164, Atlanta, USA, 2010.
[20] D. Geiger and D. Heckerman, "Learning Gaussian Networks", Technical Report MSR-TR-94-10, Microsoft Research, 1994.
[21] R. D. Shachter and C. R. Kenley, "Gaussian influence diagrams". *Management Science*, 35(5), 527–550, 1989.
[22] Q. Lu, and X. Yao, "Clustering and learning Gaussian distribution for continuous optimization,*IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 35(2), pp. 195-204, 2005.
[23] B. D. Ripley, *Stochastic simulation*. John Wiley & Sons, 1987.
[24] A. Frank and A. Asuncion, UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences, http://archive.ics.uci.edu/ml, 2010.
[25] L. N. de Castro and J. I. Timmis, "An Artificial Immune Network for Multimodal Function Optimization", *Proc. IEEE Congress of Evolutionary Computation*, vol. 1, pp. 699-674, 2002.
[26] P. Bosman and D. Thierens, "Expanding from discrete to continuous estimation of distribution algorithms: The IDEA". *Parallel Problem Solving From Nature*, 6, pp. 767–776, 2000.
[27] J. A. Hartigan, "Clustering algorithms", Wiley New York, 1975.
[28] B. Broom and D. Subramanian, "Computational Methods for Learning Bayesian Networks from High-Throughput Biological Data", *B*ayesian Inference for Gene Expression and Proteomics, Cambridge University, 2006.