Soft Comput (2014) 18:1603–1614
DOI 10.1007/s00500-013-1166-8

METHODOLOGIES AND APPLICATION

Hybrid back-propagation training with evolutionary strategies

José Parra · Leonardo Trujillo · Patricia Melin

Published online: 8 November 2013

Abstract This work presents a hybrid algorithm for neural network training that combines the back-propagation (BP) method with an evolutionary algorithm. In the proposed approach, BP updates the network connection weights, and a (1 + 1) Evolutionary Strategy (ES) adaptively modifies the main learning parameters. The algorithm can incorporate different BP variants, such as gradient descent with adaptive learning rate (GDA), in which case the learning rate is dynamically adjusted by the stochastic (1 + 1)-ES as well as the deterministic adaptive rules of GDA; a combined optimization strategy known as memetic search. The proposal is tested on three different domains, time series prediction, classification and biometric recognition, using several problem instances. Experimental results show that the hybrid algorithm can substantially improve upon the standard BP methods. In conclusion, the proposed approach provides a simple extension to basic BP training that improves performance and lessens the need for parameter tuning in real-world problems.

Keywords Neural networks · Back-propagation · Evolutionary strategies

Communicated by W. Pedrycz.

J. Parra · L. Trujillo (✉) · P. Melin
Instituto Tecnológico de Tijuana, Av. Tecnolgíco, Fracc. Tomás Aquino, Tijuana, BC, México
e-mail: leonardo.trujillo@tectijuana.edu.mx
URL: http://www.tree-lab.org

J. Parra
e-mail: galavizjpg@gmail.com

P. Melin
e-mail: pmelin@tectijuana.mx

1 Introduction

Artificial neural networks (ANNs) are one of the most widely used paradigms in pattern analysis and machine learning research.In particular, the multi-layer perceptron (MLP) has proven to be a powerful and versatile tool in many domains, including recognition (Melin and Castillo 2005), classification (Zhang 2000) and time series prediction (Castillo and Melin 2002). MLPs are feed-forward and fully connected ANNs that are trained with supervised learning methods, the most common of which is the back-propagation (BP) algorithm (Rumelhart et al. 1986). BP is a gradient descent method that propagates an error measure (such as the mean square error) from the output layer of the network to the input layer, taking into account all hidden layers in between. For years this algorithm has been considered the standard approach for supervised MLP training. However, BP is also hampered by three noteworthy shortcomings. First, BP suffers from learning problems, such as overtraining and sometimes a slow convergence depending on the characteristics of the problem and the initial connection weights. Second, it can lead to network paralysis where the algorithm is unable to significantly modify the connection weights to achieve performance improvements. And third, it often gets trapped in local minima, a common problem for gradient-based methods. To overcome these shortcomings, several improvements to basic BP training have been proposed (Isasi Viñuela 2004). For instance, gradient descent with adaptive learning rate (GDA) or gradient descent with momentum (GDM).

Currently, the improved variants of BP are widely used when training MLPs. Nevertheless, one drawback of these methods is that they tend to require several ad-hoc decisions to correctly apply them in real-world problems. Furthermore, they introduce several new parameters to achieve adaptation during training, but the parameters themselves remain constant throughout the entire process. This constrains the manner in which the learning process searches for the minimum over the error surface. One common approach to overcome these limitations is using more robust variants of BP, such as the Rprop algorithm (Riedmiller 1994). Still others have turned to population based optimization methods, such as evolutionary algorithms, to search for the optimal set of connection weights, thereby avoiding gradient-based learning altogether (Yao 1999; Cantú-Paz and Kamath 2005; Kiranyaz et al. 2009). However, such an approach does not exploit the powerful local optimization that gradient-based methods can offer. Therefore, we propose a hybrid approach that combines the learning improvements of previously proposed methods with the global search of evolutionary algorithms.

The proposal developed here, dynamically modifies the main parameters of a BP algorithm in an unconstrained manner using a $(1 + 1)$ Evolutionary Strategy (ES). The canonical training methods are easily incorporated within the evolutionary loop of the $(1 + 1)$-ES, yielding a simple hybrid learning strategy. Indeed, Cantú-Paz and Kamath (2005) performed a comprehensive comparison of different evolutionary approaches for neural network training and found that simpler methods performed the best; therefore, simplicity was a guiding principle in the proposal presented here. The experimental work evaluates the performance of this proposal using benchmark problems of time series prediction, classification and biometric recognition. Results are promising, achieving a substantial improvement in most cases and never performing substantially worse.

The remainder of this paper is organized as follows. Section 2 contains a brief introduction to the BP algorithm and some of the most widely used variants. Then, Sect. 3 presents the problem addressed in this work and introduces the hybrid training proposal. Section 4 outlines previous work in hybrid evolutionary-neural network research, while Sect. 5 gives a detailed description of the proposed algorithm and its memetic variants. Afterwards, Sect. 6 contains the experimental setup and results on several benchmark problems. Finally, concluding remarks are given in Sect. 7.

2 Background and basic concepts

2.1 Back-propagation training

The standard supervised learning method for MLPs is BP, a gradient descent optimization algorithm that backwardly propagates the error between the network output and the desired output. BP uses this error to modify the network connection weights based on the gradient and on a learning rate parameter β, which modulates the step size of the weight updates. Despite the success of BP training, it is well-known that it suffers from several limitations. Therefore, many researchers have proposed algorithmic improvements.

2.1.1 Back-propagation algorithm

In an MLP, the input pattern is propagated forward through the network, and this results in a vector of activation values in the output layer, which means that the MLP basically acts as a function. The behavior of the function can be changed by modifying the connection weights between individual neurons. Starting from an initial set of random connection weights, the objective of BP is to adaptively modify these weights in order to achieve a particular input/output behavior. In fact, the overall goal is to minimize the error E given by

$$E = \frac{1}{2}(d_p - y_p)^2, \tag{1}$$

where d_p is the desired output and y_p is the actual network output obtained for each input pattern x_p.[1] In order to do so, the connection weight w_{ijl} between neuron i in layer l and neuron j in layer $l + 1$ is modified at each epoch t by the following rule

$$w_{ijl}(t + 1) = w_{ijl}(t) + \Delta w_{ijl}(t), \tag{2}$$

where

$$\Delta w_{ijl}(t) = \beta \delta_{i(l+1)p}(t) y_{jlp}(t) \tag{3}$$

$\delta_{i(l+1)p}(t)$ is the generalized error term at neuron i in layer $l+1$ for pattern p, given by the product of the first derivative of the activation function and error E, $y_{jlp}(t)$ is the output of neuron j in layer l for pattern p, and where β is the learning rate parameter. For a more complete description of BP training the interested reader is referred to (Rumelhart et al. 1986; Radi and Poli 2003).

2.2 BP with adaptive learning rate

One of the earliest improvements to BP was to adaptively modify the learning rate on-line during training (Hagan and Beale 1996). In standard BP the learning rate is constant during the entire training process, it is thus imperative to choose a correct initial value. For instance, if the learning rate is set too high the algorithm may oscillate and become unstable. Conversely, if it is too small then convergence will be slow. However, setting an optimal value for β is not trivial. Moreover, the optimal value might change during the training process. Therefore, if β is allowed to change during training the quality of the learning process might improve. The idea is to make β responsive to the structure of the local error surface.

[1] The error measure E given above is just one possible measure that can be used.

In order to implement this idea, the GDA algorithm modifies BP in the following ways. First, the initial network output and error are calculated. At each epoch new weights are computed using the current β. New outputs and errors are then measured. If the new error exceeds the old error by more than a predefined threshold then the new weights are discarded and β is decreased by a fixed amount, call this parameter $\beta_{\Delta-}$. Otherwise, the new weights are kept, and if the new error is less than the old error, the learning rate β is increased by a constant parameter $\beta_{\Delta+}$. Therefore, if a larger β could result in stable learning then it is increased. On the other hand, if the learning rate is too high to guarantee a decrease in error then it is decreased until stable learning resumes (Hagan and Beale 1996).

2.3 BP with momentum

Another proposed improvement is the GDM algorithm, which takes a different approach towards overcoming some of the shortcomings of BP. It is equivalent to BP, with an added parameter called the momentum coefficient γ which is used to modify the weight update rule as follows (Samarasinghe 2006)

$$\Delta w_{ijl}(t) = \gamma \Delta w_{ijl}(t-1) + (1-\gamma)\beta\delta_{i(l+1)p}(t)y_{jlp}(t). \tag{4}$$

In essence, γ produces an averaging effect by which changes in connection weights consider both the current error value as well as past weight changes.

2.4 BP with momentum and adaptive learning rate

Based on the previous methods, GDA and GDM, the gradient descent algorithm with momentum and adaptive learning rates was proposed (GDX), which combines the advantages of both (Hagan and Beale 1996).

3 Problem description and main proposal

We have outlined three of the most common improvements to BP training, GDA, GDM and GDX. However, the manner in which these methods operate also raises other issues. For instance, in GDA the learning rate is either increased or decreased by the fixed parameters $\beta_{\Delta+}$ and $\beta_{\Delta-}$. One could argue that the value of these parameters should also be subject to an adaptive process during training. Furthermore, in GDM the γ coefficient is held constant, and there is no a priori reason to assume that such a strategy is optimal. Therefore, in this work we hypothesize that a better learning strategy would be able to adaptively modify all of the main parameters of the algorithm in an unconstrained manner. This follows from the basic argument behind the GDA method, where it is assumed that because the error surface changes during training then the optimal β should also change. Therefore, we argue that the same logic must hold for parameters such as $\beta_{\Delta+}$ and $\beta_{\Delta-}$, and the γ coefficient. For example, in GDA β should be able to increase or decrease during training without the need of constant step sizes, and in GDM γ could also be adaptively modified.

Therefore, we propose an improvement to BP that can dynamically change the main parameters of the learning algorithm without constant step values. In order to achieve this we develop our proposal using a hybrid algorithm that combines an evolutionary search process and standard BP. Concretely, we use evolutionary strategies as a global search method that adapts the main learning parameters during training and allows the gradient descent algorithm to perform a local search over the MLP error surface. With the hybrid approach, we combine the exploration capabilities of evolutionary search with the local optimization provided by gradient descent. Moreover, the proposal can incorporate the basic BP algorithm as well as any of the previously proposed improvements (GDA, GDM and GDX) without requiring substantial modifications. To contextualize the current contribution, the following contains a brief overview of how evolutionary computation (EC) has intersected with ANN research in previous works.

4 Evolutionary computation and ANNs

Evolutionary computation encompasses a large variety of global search and optimization methods that are based on an abstract model of Neo-Darwinian evolutionary theory. Some of the most widely known paradigms are genetic algorithms, evolutionary strategies and genetic programming, all of which are based on similar conceptual principles (DeJong 2002), and are closely related to other population-based meta-heuristics such as particle swarm optimization (Kiranyaz et al. 2009) and ant colony optimization (Dorigo and Stützle 2004). These methods have proven to be quite robust and flexible, applicable to a large variety of application domains and problem instances.

In the case of ANNs, many attempts have been made to use evolutionary methods to optimize a specific characteristic of an ANN (Yao 1999; Cantú-Paz and Kamath 2005). Probably the most common strategy is to use EC to determine the optimal connection weights of an ANN (Yao 1999; Fogel et al. 1990), in some cases combining an EC algorithm with standard learning techniques (Alba and Chicano 2004). For instance, this approach has found strong acceptance in robotics applications, an approach referred to as evolutionary robotics, where a well posed error gradient is not feasible (Nolfi and Floreano 2000). Recently, these approaches have allowed researchers to train extremely large networks by exploiting concepts from developmental biology and indirect encoding schemes (Stanley et al. 2009). Another possibility is to use EC to search for an optimal network topology (Miller 1989) and then use standard learning methods to determine the connection weights of the network. Still yet, others have attempted to reduce the amount of a priori knowledge as much as possible by concurrently searching for the optimal network topology and connection weights within a single evolutionary loop (Harp et al. 1989; Stanley and Miikkulainen 2002).

On the other hand, EC has also been used to develop improvements to the traditional learning process. For instance, some researchers use EC techniques to determine the initial connection weights of an ANN, instead of using random weights, from which the learning algorithm can then proceed (Lee 1996). Still yet, others have focused on optimizing the BP parameters off-line, to find optimal values that can be used throughout the entire learning process (Patel 1996) or by considering the learning parameters and connection weights concurrently (Merelo et al. 1993). Finally, some researchers have used automatic program induction with genetic programming to derive new learning rules through and evolutionary search (Radi and Poli 2003). In our work, however, we are interested in developing an adaptive strategy similar to what is done in GDA, with an additional evolutionary search that allows for dynamic modifications of the BP parameters. A similar approach was proposed in (Kim et al. 1996) with several important differences that are addressed in Sect. 5.5.

5 The proposed hybrid learning approach

This section presents our hybrid approach for BP learning using evolutionary computation.

5.1 Evolutionary strategies

Evolutionary strategies are an optimization technique based on the core principles of EC highlighted in the previous section (Schwefel 1981). In ES, candidate solutions are coded as real-valued parameter vectors. In the canonical version only one operator is used to generate new parameter vectors (offspring), a Gaussian mutation that perturbs the value of each parameter; for a detailed introduction see (Eiben and Smith 2003). Moreover, two selection strategies are commonly used with ESs, $(\mu + \lambda)$ and (μ, λ), where μ is the number of individuals in a population and λ is the number of offspring generated at each generation. In a $(\mu + \lambda)$-ES, the individuals contained in the next generational loop are chosen from the best solutions from both the past population and the offspring. Conversely, in (μ, λ) the λ offspring replace all of the μ individuals in the previous population. In other words, $(\mu + \lambda)$ is an elitist strategy while (μ, λ) is not (Eiben and Smith 2003).

5.2 Evolutionary strategies for BP learning

As stated above, our proposal is to combine BP with an evolutionary search. The goal is to provide a mechanism by which the main parameters of a BP algorithm can be adaptively modified on-line during network training. For this task we have chosen the $(1 + 1)$-ES, because:

- It is well-known and understood.
- It is particularly well suited for real-valued parameter optimization.
- It is very simple to implement, which allows us to maintain the basic structure of BP unchanged. From this it follows that the method will not dramatically increase the computational cost of training an ANN. Simplicity was explicitly considered given the conclusions derived by the extensive comparison of previously proposed evolutionary approaches carried out by Cantú-Paz and Kamath (2005).

The proposal is a hybrid learning process, such that a $(1+1)$-ES adaptively changes the BP parameters after a specified number of epochs, during which the BP training algorithm carries out the standard weight updates. The $(1+1)$-ES represents the simplest type of evolutionary search, that lacks the intrinsic parallel nature of other population based evolutionary techniques, such as genetic algorithms or genetic programming. Therefore, the $(1 + 1)$-ES is closely related to other heuristic search methods such as simulated annealing (Kirkpatrick et al. 1983). Nonetheless, the aim of the proposal is to develop a simple and robust improvement to BP training with a minimal amount of computational overhead, and, as we shall see in our results, the $(1+1)$-ES fulfills these requirements.

The proposed algorithm proceeds as follows. First, generate a BP parameter vector x with standard initial values that are commonly used in the literature. In this case, the number of parameters depends on the version of BP used. For instance, if we use GD or GDA then x would contain only the β parameter. On the other hand, if we use GDM or GDX, then x would contain β and the momentum coefficient γ. Afterwards, randomly generate the initial connection weights of the ANN called Ax, just as it would be done in standard BP. This leads to the first generation (iteration) of the $(1 + 1)$-ES. Within the evolutionary loop, create a mutated version of x called y, using a Gaussian mutation with the same σ for all elements. Then, make a copy of Ax, call this Ay. Train Ax using the BP parameters specified in x for a total of ρ epochs, and the same is done for Ay with the parameter values y. After training both networks we obtain a corresponding convergence error from each, call these Ex

and Ey, respectively. The error values are used to determine which ANN and which parameter vector will survive for the following generation. This process is repeated until one of two conditions is met: (1) the total number of generations is reached; or (2) the goal error for the ANN is achieved. In this algorithm there are two new parameters. One is the number of epochs per generation denoted by ρ. The other is the value of the step size σ of the Gaussian mutation. In this work, σ is set to a constant value of 0.2, while ρ is set using an extensive experimental evaluation described in the following section. It should be mentioned that some of the most advanced, and successful, variants of ES are those that adaptively modify σ by including it as another decision variable optimized by the evolving process (Eiben and Smith 2003). In fact, in such a case an ES can fit σ to the structure of the fitness landscape, with the slight tradeoff of increasing the dimensionality of the search space and possibly making it more complex. However, in our case the search space is not fixed, that is, fitness evaluation is dynamic because BP modifies the connection weights at each time step (possibly several times) and thus changes the underlying structure of the fitness landscape. Therefore, we choose not to include an adaptive mutation step size in the evolutionary process.

5.3 Memetic search

Since the proposal in this work does not depend on a specific BP variant, it is also possible to use learning algorithms that can also modify the learning parameters, in particular GDA and GDX which adapt the learning rate β. If such is the case, this parameter can be adapted by two separate mechanisms, the more unconstrained global search carried out by ES and by local optimization with gradient descent. This type of combined optimization strategy is called a memetic algorithm within EC literature (Eiben and Smith 2003), and two variants exist. The first one is a Lamarckian memetic algorithm (Husbands 1994), where inheritance of acquired traits is possible, while the other is based on Baldwin's theory of inherited learning ability (Paul et al. 1987). The difference between both approaches is based on whether or not the modifications made to β by the underlying BP algorithm (in this case GDA or GDX) are encoded back to the parameter vectors (x or y) and thus propagated to the next generation. In the Lamarckian case the updated β is inserted into the parameter vector, while in the Baldwinian case it is not. In our work, we test both variants, thus we have a total of ten different algorithms that are tested, these are:

- Standard algorithms: Basic back-propagation (GD), GDA, GDM and GDX.
- Non-memetic BP with ES: GD-ES and GDM-ES.
- Lamarckian memetic algorithms: GDA-ES/L and GDX-ES/L.
- Baldwinian memetic algorithms: GDA-ES/B and GDX-ES/B.

5.4 Discussion and design choices

A reasonable question arises regarding the proposed algorithm. In the development of an algorithm that can automatically adjust the learning rate and momentum coefficient for a BP training algorithm, we have introduced several new parameters related to the $(1+1)$-ES, namely μ, λ, σ and ρ. Then, it might seem that the original concern arises again: is it necessary to dynamically adjust these parameters based on the learning process to achieve optimal performance? However, care must be taken regarding such concerns, if not we might fall into an infinite regress of algorithm design if an algorithm without parameters cannot be derived. Consider that the goal of an evolutionary algorithm is to automate a real-world problem solving process, allowing system designers to transfer the responsibility of determining some solution features to an independent process. This, as stated before, comes at a cost, with a possible increase in system parameters and (for most problems) no guarantee of optimality. However, in practice these shortcomings are mostly overcome by the fact that evolutionary algorithms tend to be quite robust, achieving strong results in a wide range of parameter values (Eiben and Smith 2003; DeJong 2002). The implicit assumption made when evolutionary algorithms are used is that the search process will be easier to setup and configure than the underlying problem that needs to be solved.

Returning to the problem addressed in this work, the following design choices are made. First, population sizes μ and λ are set to 1, to maintain computational cost low with respect to standard BP variants. Second, the Gaussian mutation step size σ is set to a constant value to maintain a simple search process and because the search presents a dynamic fitness landscape. Third, the step size is set $\sigma = 0.2$ since the domain of both BP parameters that are evolved, β and γ, is basically $[0, 1]$,[2] thus the step size provides a good exploratory search. Finally, the number of epochs per generation ρ can be seen as the only new parameter in the proposed $(1+1)$-ES. Therefore, the proposed experimental study will first focus on analyzing the influence of ρ on the quality of network training over a wide range of values. It should be noted that such an experimental validation is definitely not exhaustive (there might be a plausible interdependence between the $(1+1)$-ES parameters), but any further analysis is left as future research. Nonetheless, the experimental work is aimed at showing that the $(1+1)$-ES provides a sufficiently robust algorithm that simplifies parameter setting and tuning.

[2] For the momentum coefficient γ this is strictly the case, while for the learning rate β most published works suggest values within this range.

5.5 Comparison with previous work and design choices

The proposal developed in this work, is one of a handful of methods that address the problem of MLP training with artificial evolution. In fact, to our knowledge, only the work in Kim et al. (1996) presents a similar approach, with several noteworthy differences. First, Kim et al. (1996) only considers the learning rate, it does not include the momentum coefficient. Second, in Kim et al. (1996) the learning rate is updated every generation by the evolutionary algorithm, thereby not exploiting a local optimization. In fact, this configuration is contained as a particular case of our approach and is evaluated in the experimental work presented below. However, the experimental results strongly suggest that the best performance is achieved by a trade-off that exploits both optimization strategies using a memetic approach, instead of the pure evolutionary search of Kim et al. (1996). Third, Kim et al. (1996) use an evolutionary algorithm with a population of 20 or 50 individuals, thus raising the number of different networks that must be trained concurrently. On the other hand, by using a $(1 + 1)$-ES we are at most doubling the total computation required to train a single network, a negligible increase for many real-world scenarios. Finally, while Kim et al. (1996) only presents preliminary results on simple synthetic problems, we consider several benchmark and real world problems from ANN and machine learning literature.

6 Experiments and results

The aim of the experimental tests is twofold. Firstly, we want to estimate a range of values for which the ρ parameter gives the best performance. Secondly, perform a series of comparative tests between the basic BP algorithms (GD, GDA, GDM and GDX) and the corresponding ES variants. The goal of the first set of experiments is to examine how ρ influences the performance of the learning process and to determine a proper range of values for which ρ gives the best performance. The second set of experiments will provide a comparative statistical validation of the type of improvements that the hybrid algorithm yields. These experiments are carried out using three different problem types, and six datasets in total. Finally, all of the algorithms are coded and tested using Matlab 2009a.

6.1 Datasets and experimental setup

Three different problem types are used: time series prediction, classification and biometric pattern recognition. With several different instances for each case.

The Mackey and Glass (1977) time-delay differential equation is used for time series prediction, given by

Table 1 Mackey–Glass time series problem, from left to right: dataset, network architecture and initial parameters for BP algorithm

Time series	Dataset	Network	Params
Mackey–Glass	800 samples 500/training, 300/testing	1 Hidden layer 25 neurons, 3 inputs: $x(t-1)$ $x(t-2), x(t-3)$ Tan-Sigmoid activation Prediction horizon: $x(t)$	$\beta = 0.01$ $\gamma = 0.9$ Epochs = 4,000 Goal = 1e−10

$$\frac{dx}{dt} = \frac{0.2x(t-\phi)}{1 + x(t-\phi)^{10}} - 0.1x(t). \qquad (5)$$

We construct three datasets by using ϕ values of 16, 17 and 30, where the the first produces no chaotic behavior and the last produces the most, we respectively call each dataset MKG1, MKG2 and MKG3. The initial conditions in each case are $x(0) = 1.2$ and $x(t) = 0$ for $t < 0$, and the data is generated using a Matlab 2009a implementation of the 4th order Runge–Kutta method.

Additionally, three different classification problems are tested, taken from the UCI repository,[3] these are forensic glass, red wine quality, and breast tissue. Finally, the algorithm is also tested on a recognition problem, using the ORL face dataset (Samaria and Harter 2002). For face recognition, each image is given as input to the neural network in vector form without any preprocessing, where each is of size 92×112 pixels and in greyscale.

The datasets are summarized in Tables 1, 2 and 3, along with the neural network architecture and parameters used in each experiment. The learning rate and momentum coefficient use the default values provided with the implementations of the training algorithms from the Neural Network Toolbox for Matlab, except for the classification problems where they are set based on the best performance achieved by GD on the forensic glass problem. For GDA and GDX $\beta_{\Delta+} = a\beta$ with $a = 1.05$ and $\beta_{\Delta-} = b\beta$ with $b = 0.7$, as suggested in the Neural Network Toolbox. Additionally, the neural network architecture was also set empirically for all problems based on the best performance achieved by GD, except for the face recognition problem where GD performs quite poorly, so the architecture is set based on the performance of GDA. In summary, network architecture and initial BP parameters are set based on the performance of standard BP variants or based on the suggested values from a widely used Matlab toolbox. This provides an initial baseline configuration that can be used to fairly compare the proposed ES-based variants.

[3] UCI Machine Learning Repository, http://archive.ics.uci.edu/ml.

Table 2 Description of the classification problems, from left to right: dataset, network architecture and initial parameters for BP algorithm

Problem	Dataset	Network	Params
Forensic glass	214 samples	2 hidden layers	$\beta = 0.5$
	K-fold = 10	64 neurons	$\gamma = 0.5$
	7 classes	7 output neurons	Epochs = 4,000
	10 attributes	Log-Sigmoid activation	Goal = 1e−10
Red wine quality	1,599 samples	2 hidden layers	$\beta = 0.5$
	K-fold = 10	15,11 neurons	$\gamma = 0.5$
	6 classes	6 output	Epochs = 4,000
	11 attributes	Log-Sigmoid activation	Goal = 1e−10
Breast tissue	106 samples	2 hidden layers	$\beta = 0.5$
	K-fold = 3	60 neurons	$\gamma = 0.5$
	6 classes	4 output neurons	Epochs = 4,000
	9 attributes	Log-Sigmoid activation	Goal activation = 1e−5

Table 3 Description of the biometric face recognition problem

Problem	Dataset	Network	Params
Face recognition	400 samples	2 hidden layers	$\beta = 0.01$
	K-fold = 10	84,45 neurons	$\gamma = 0.9$
	40 person	40 output neurons	Epochs = 1,000
	10 images of each person		Goal = 0.00001

Note that the learning rate and momentum parameters define the initial individual of the $(1 + 1)$-ES. In all three cases, a desired goal error is used as the primary stopping criterion and a maximum number of epochs as a secondary criterion, both specified in the corresponding tables referenced above.

6.2 Effects of the number of epochs per generation on network training

It is imperative to analyze how the number of epochs per generation, specified by the ρ parameter, effects the performance on learning for each algorithm. To do so, we test each algorithm on a specific problem and vary ρ over a fixed range of values in a discrete logarithmic scale, given by $[2, 3, \ldots, 10, 20, 30, \ldots, 100, 200, 300, \ldots, 1000, 2000]$. This allows us to test the effect of ρ at different orders of magnitude. We perform this test on a dataset from each problem type: for time series we use MKG1, for classification we use forensic glass, and for biometric recognition we use face recognition. Moreover, in order to obtain statistically valid results, for time series we compute the average performance over thirty independent runs.On the other hand, for classification and biometric recognition we use the average performance of tenfold cross-validation as suggested by Refaeilzadeh et al. (2009). For instance, Fig. 1a shows the average classification accuracy and standard error on the test set of forensic glass for GDM-ES over the entire range of ρ values, presented in a log scale. In this case the performance reaches a maximum plateau in the range of [200−1,000], which can be considered as a proper operating range for GDM-ES. This is done for all of the ES variants and all of the problem types, these results are summarized in the second, third and fourth columns of Table 4. For most of methods, in all of the problems, the best vale for ρ is a few hundred epochs per generation, with some exceptions that achieved better performance with smaller ρ values. Additionally, we can count the number of times that evolution produced improvements on the learning parameters, which is the number of times that the offspring vector substituted the parent vector. This is shown in Fig. 1b for the GDM-ES algorithm, the results for all of the methods are given in the final three columns of Table 4 that shows the minimum and maximum values obtained within the established range for ρ. Obviously, the number of times that the offspring vector improves upon the parent depends on ρ which effectively gives the total number of times that an offspring is generated. Therefore, the results are presented as the proportion of times that the offspring outperformed the parent, where a value of 1 would indicate that the offspring always replaces the parent. For GDM-ES, Fig. 1 shows that there is a correlation between the maximum performance achieved and the higher proportion of instances in which the offspring outperformed the parent. This indicates that the ES is allowing the network learning process to improve upon what a standard BP run would have achieved. Moreover, similar results were obtained for all ES variants of BP. In the following section, we focus on determining the significance of the improvements, if any, that are achieved by the evolutionary competition between the parent vector and its offspring when we compare the $(1+1)$-ES proposals with the standard algorithms.

6.3 Comparative tests

This section compares each of the baseline BP algorithms with their ES variants on all of the test problems, setting the ρ parameter based on the preceding tests. On the other hand, for the standard BP algorithms the parameters are set to common values that have been determined after years of theoretical and empirical work, which is why we do not optimize them for each specific problem. It is important to consider that parameter selection and tuning is an important open problem in machine learning, and most real-world implementations are based on an initial "good guess" made by the system

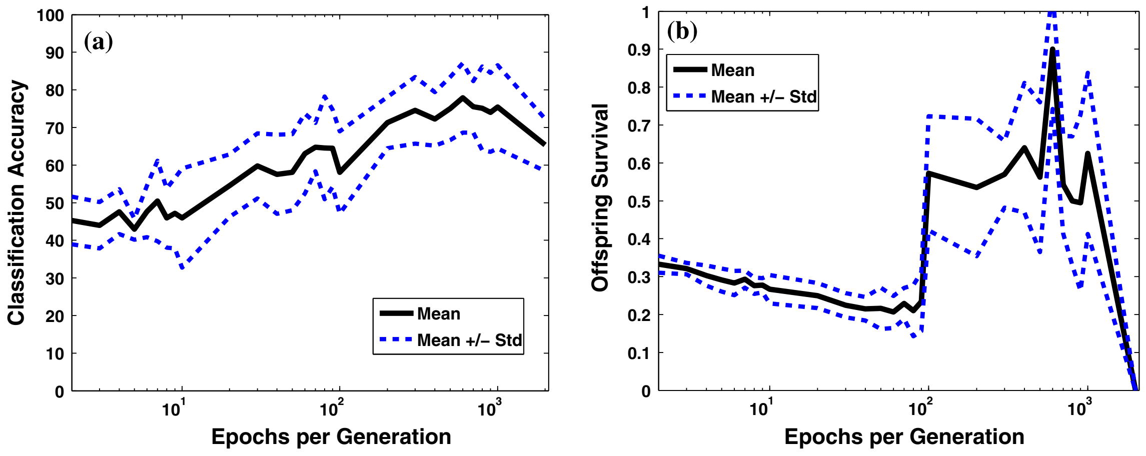

Fig. 1 Performance of GDM-ES on the forensic glass classification problem. **a** Accuracy, **b** offspring survival

Table 4 Ranges in which ρ produced the best results for each ES-based BP algorithm

Method	ρ for time series	ρ for classification	ρ for biometric pattern	Avg. of offspring time series	Avg. of offspring classification	Avg. of offspring biometric pattern
GD-ES	50–100	200–1,000	2–10	0.08–0.10	0.32–0.67	0.13–0.68
GDA-ES/B	70–500	100–800	200–500	0.14–0.77	0.44–0.86	0.30–0.38
GDA-ES/L	70–500	10–90	100–500	0.16–0.87	0.03–0.24	0.20–0.25
GDM-ES	2–90	200–1,000	2–9	0.05–0.29	0.53–0.55	0.32–0.80
GDX-ES/B	90–700	100–1,000	200–500	0.28–0.29	0.15–0.92	0.42–0.70
GDX-ES/L	60–400	60–100	200–500	0.20–0.32	0.34–0.37	0.25–0.40

designer considering what has previously been reported to achieve good results in relevant literature. Indeed, the default values provided in software tools are based precisely on such an analysis, but it is not correct to describe these parameters as random, they are set within a specific range of values were good performance can be expected. Nonetheless, as stated before, in general these initial values cannot guarantee optimal performance in the general case, so researches must usually struggle with a tedious trial and error process to tune system parameters. Therefore, the goal of the proposal made in this work is precisely to allow the designer to use "standard" initial values, that are then tuned automatically through the ES-based search. On the other hand, an initial "good guess" for ρ cannot be derived from previous experiences, it is therefore based on the range of values where performance peeked on some representative test cases. Hence, for the experiments reported below, a value within these, quite large, ranges of values is chosen for each problem. Indeed, Table 4 confirms what is commonly accepted as a desirable property of evolutionary algorithms, their robustness to parametrization within a wide range of values. Moreover, it is important to point out that ρ is not chosen based on the performance achieved on all problems, only on single representative instances.

6.3.1 Time series

Table 5 shows the comparisons on each of the time series problems, with statistics of NRMSE[4] computed over thirty runs, showing average performance and standard error. The results show that in all cases the ES algorithms obtained a substantial improvement over the baseline methods. For instance, Fig. 2 shows the time series prediction for MKG2 using GDM and GDM-ES on the test data, this illustrates the improvement in performance achieved by the ES algorithm. To validate these results we perform a one sided t-test between each BP algorithm and its ES variant (Zimmerman and Williams 1986). For all of the cases there is a statistically significant improvement at the $\alpha = 0.01$ significance level, except on the MKG2 time series with GDA and GDA-ES/L. This is strong evidence that the proposed algorithm is able to substantially improve upon standard BP training.

In comparison with other published results, the NRMSE for time series prediction on Mackey–Glass is quite good. For instance, for MKG2 performance the proposed algorithm outperforms a particle swarm algorithm as reported by Samanta (2011). However, the method does not outper-

[4] Normalized Root Mean Square Error.

Table 5 Comparative result for the Mackey–Glass problem showing the average NRMSE and standard deviation

Method	ρ	Error MKG1	Std MKG1	Error MKG2	Std MKG2	Error MKG3	Std MKG3
GD	–	2.57	1.18	0.38	0.18	1.87	1.08
GD-ES	80	0.30	0.15	0.20	0.11	0.30	0.16
GDA	–	1.84	0.80	0.23	0.14	1.25	0.77
GDA-ES/B	200	0.14	0.03	0.18	0.09	0.25	0.16
GDA-ES/L	70	0.15	0.07	0.20	0.11	0.20	0.06
GDM	–	2.88	1.19	0.45	0.26	2.04	1.15
GDM-ES	90	0.23	0.09	0.15	0.06	0.27	0.11
GDX	–	2.93	1.43	0.21	0.12	2.07	1.19
GDX-ES/B	700	0.15	0.05	0.14	0.08	0.19	0.07
GDX-ES/L	60	0.16	0.07	0.13	0.07	0.21	0.09

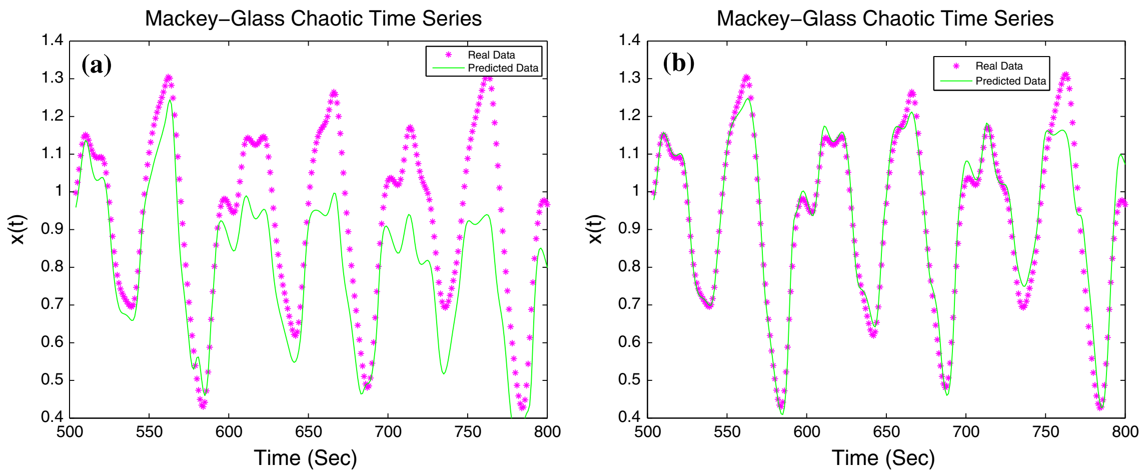

Fig. 2 Time series prediction on the test data for the MKG2 time series on **a** GDM and **b** GDM-ES

form more complex prediction strategies that were specifically designed for time series forecasting, such as an ensemble of ANIFS (Melin et al. 2012). Nonetheless, the goal here is only to improve the BP training process, domain specific systems can then use the learning algorithm to design more complex ensemble or hybrid methods.

6.3.2 Classification

For the classification problems we use k-fold cross-validation, Table 6 shows the comparisons on each dataset with statistics for classification accuracy, showing average performance and standard error. The results show that in the forensic glass problem the ES algorithms obtained a substantial improvement over the baseline methods. This is also apparent for the breast tissue classification problem, particularly for the GDA-ES variants. However, for red wine most of the ES algorithms only achieve a minimum improvement over the basic BP algorithms.

The statistical tests confirm these empirical observations, because according to the t-tests for the forensic glass problem all of the ES variants achieve a statistically significant improvement at the $\alpha = 0.01$ significance level. For red wine quality only the two memetic variants of GDA-ES showed a statistically significant improvement over GDA at the $\alpha = 0.05$ significance level, with GDA-ES/L also achieving it at $\alpha = 0.01$. In all other cases, the performance is very similar, with all methods exhibiting a poor performance of around 50 % accuracy. For the breast tissue problem, given the amount of available data and classes only threefold cross validation was performed, which excludes strong statistical testing and explains the large performance variations exhibited by two methods (GDA-ES/B and GDM-ES). Nonetheless, the results indicate a noticeable improvement in average performance, especially considering the observed standard error for the GDA-ES and GD-ES variants.

Finally, to provide an additional comparison, we compare with the accuracy results published by other authors on these sets. This is an informal comparison, since the algorithms were not executed under the same conditions. Nonetheless, it provides a rough estimate of the quality of the results reported here. The performance achieved by

Table 6 Comparative results for the classification problems, showing average classification accuracy on test data and the standard error over the k-fold cross-validation

Method	ρ	Acc. glass	Std glass	Acc. wine	Std wine	Acc. breast	Std breast
GD	–	34.5	4.3	48.8	7.2	46.21	0.87
GD-ES	500	77.2	8.7	48.2	7.1	54.65	7.99
GDA	–	35.0	2.1	46.4	3.9	46.21	0.87
GDA-ES/B	100	74.1	8.0	50.8	4.1	70.82	20.88
GDA-ES/L	100	63.0	8.9	53.6	2.8	60.42	5.09
GDM	–	36.8	2.4	50.0	6.7	46.21	0.87
GDM-ES	600	77.8	9.2	51.4	5.3	52.88	11.13
GDX	–	35.5	1.8	50.0	5.5	50.97	7.84
GDX-ES/B	500	77.9	9.3	50.4	5.4	55.68	8.93
GDX-ES/L	100	66.2	5.6	50.5	4.4	53.80	10.53

Table 7 Comparative results for the face recognition problem

Method	ρ	Recognition %	Std
GD	–	3.45	1.1654
GD-ES	3	67	2.8087
GDA	–	81.15	2.6357
GDA-ES/B	400	78.75	3.9033
GDA-ES/L	400	80.1	3.5103
GDM	–	3	0.9128
GDM-ES	2	65.8	4.2242
GDX	–	79	2.6457
GDX-ES/B	300	79.65	4.6430
GDX-ES/L	300	79.45	3.1837

GDM-ES, GDX-ES/B and GD-ES on forensic glass compares favorably with other published results, see for instance (Denoeux 2000; Zhong 2007). For red wine the improved ES variants outperform low-tolerance regression methods, such as Support Vector Machines (Cortez et al. 2009). Moreover, GDA-ES/L achieves comparable results to those of Naive Bayes and Artificial Immune Recognition Systems, but is slightly behind Logistics Regression and another MLP that achieves around 60 % accuracy[5] according to Dogan and Tanrikulu (2010).

6.3.3 Biometric recognition

Finally, we present comparative tests for biometric recognition on the ORL dataset, these are summarized in Table 7. In these tests our proposal achieves different results for each method. For instance, for GD and GDM the improvements are substantial, a difference of one order of magnitude in performance, and the t-tests confirm this above the $\alpha = 0.01$ significance level. On the other hand, for GDA and GDX the performance is basically the same with and without ES, and the t-tests confirms it. It appears that for this problem the learning process must be able to change the learning rate at almost every epoch, thus the good performance by GDA and GDX which is comparable with other approaches (Ayinde and Yang 2002). Moreover, notice that GD and GDM have an extremely low accuracy, while their ES variants require a small value for ρ and thus a frequent update of the learning rate, see Table 4. However, the random modifications offered by the ES algorithm cannot achieve the same performance than the deterministic heuristic followed by GDA, when we compare GD with GD-ES and GD with GDA.

7 Discussion and conclusions

This paper presents a hybrid approach towards neural network training that combines the standard back-propagation algorithm with a $(1 + 1)$ Evolutionary Strategy that adaptively modifies the main learning parameters. The algorithm can easily accommodate any of the standard BP variants, and can thus perform a memetic search when coupled with an adaptive learning rate method, such as GDA or GDX. The proposed algorithm was tested on various benchmark problems and compared with standard methods. The results show that the overall performance of an MLP in many cases is increased with the hybrid learning approach, particularly for time series prediction and classification problems. In fact, even in the worst test cases, the performance of the proposed algorithm was never substantially worse than standard methods. We do not claim that the proposed algorithm will achieve the best performance on all possible tests, indeed the many degrees of freedom these algorithms present coupled with well-known theoretical (Wolpert and Macready 1997) and experimental (Cantú-Paz and Kamath 2005) results, should lead us towards the conclusion that the proposal can outperform other algorithms on some problems, but will probably perform worse in other cases. However, the results pre-

[5] The authors do not provide the network architecture, thus a full comparison is not possible.

sented here are encouraging, considering that the ES variants mostly outperform simple BP algorithms, while Cantú-Paz and Kamath (2005) showed that in their experiments the standard BP algorithms actually performed better in many test cases. It appears that the unconstrained adaptations of the BP learning parameters provided by the $(1 + 1)$-ES does indeed allow the learning process to escape local optima and find a better overall optimization of network connection weights.

Future work derived from this paper should first focus on further validating the algorithm in other application domains. Moreover, in principle the proposed strategy could be used to enhance other learning algorithms, even those based on EC techniques, thus providing a self tuning or self adapting mechanism, which is a prerequisite for the development of truly autonomous learning systems.

Acknowledgments First author was supported by scholarship 263888 from Consejo Nacional de Ciencia y Tecnología (CONACYT) of México. Corresponding author also thanks the Departamento de Ingeniería Eléctrica y Electrónica at the Instituto Tecnológico de Tijuana. Additionally, partial funding for this work was given by CONACYT (Mexico) Basic Science Research Grant No. 178323.

References

Alba E, Chicano JF (2004) Training neural networks with GA hybrid algorithms. In: Deb K, et al (eds), GECCO (1). Lecture Notes in computer science, vol 3102. Springer, Berlin, pp 852–863

Ayinde O, Yang Y (2002) Face recognition approach based on rank correlation of Gabor-filtered images. Pattern Recognit 36(6):1275–1289

Cantú-Paz E, Kamath C (2005) An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems. IEEE Syst Man Cybern Soc 35(5):915–927

Castillo O, Melin P (2002) Hybrid intelligent systems for time series prediction using neural networks, fuzzy logic, and fractal theory. IEEE Trans Neural Netw 13(6):1395–1408

Cortez P, Cerdeira A, Almeida F, Matos T, Reis J (2009) Modeling wine preferences by data mining from physicochemical properties. Decis Support Syst 47(4):547–553

DeJong K (2002) Evolutionary computation: a unified approach. The MIT Press, Cambridge

Denoeux T (2000) A neural network classifier based on Dempster–Shafer theory. IEEE Syst Man Cybern Soc 30(2):131–150

Dogan N, Tanrikulu Z (2010) A comparative framework for evaluating classification algorithms. In: Proceedings of the world congress on engineering 2010, vol 1. WCE, pp 379–384

Dorigo M, Stützle T (2004) Ant colony optimization. Bradford Company, Scituate

Eiben AE, Smith JE (2003) Introduction to evolutionary computing. Springer, Berlin

Fogel DB, Fogel LJ, Porto VW (1990) Evolving neural networks. Biol Cybern 63(6):487–493

Hagan M, Demuth H, Beale M (1996) Neural Network Design. PWS Publishing Company, Boston

Harp SA, Samad T, Guha A (1989) Towards the genetic synthesis of neural network. In: Proceedings of the third international conference on Genetic algorithms. Morgan Kaufmann Publishers Inc., San Francisco, pp 360–369

Hoel PG, Port SC, Stone CJ (1987) Introduction to stochastic processes. Waveland Press, Long Grove

Husbands P (1994) Distributed coevolutionary genetic algorithms for multi-criteria and multi-constraint optimisation. In: Selected papers from AISB workshop on evolutionary computing. Springer, London, pp 150–165

Isasi Viñuela P (2004) Redes neuronales artificiales: un enfoque práctico. TPearson Educacion, Upper Saddle River

Kim HB, Jung SH, Kim TG, Park KH (1996) Fast learning method for back-propagation neural network by evolutionary adaptation of learning rates. Neurocomputing 11(1):101–106

Kiranyaz S, Ince T, Yildirim A, Gabbouj M (2009) Evolutionary artificial neural networks by multi-dimensional particle swarm optimization. Neural Netw 22(10):1448–1462

Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680

Lee S-W (1996) Off-line recognition of totally unconstrained handwritten numerals using multilayer cluster neural network. IEEE Trans Pattern Anal Mach Intell 18(6):648–652

Mackey M, Glass L (1977) Oscillation and chaos in physiological control systems. Science 197(4300):287–289

Melin P, Castillo O (2005) Hybrid intelligent systems for pattern recognition using soft computing: an evolutionary approach for neural networks and fuzzy systems (Studies in fuzziness and soft computing). Springer-Verlag New York Inc., Secaucus

Melin P, Soto J, Castillo O, Soria J (2012) A new approach for time series prediction using ensembles of ANFIS models. Expert Syst Appl 39(3):3494–3506

Merelo J, Patón M, Cañas A, Prieto A, Morán F (1993) Optimization of a competitive learning neural network by genetic algorithms. In: Proceedings of the international workshop artificial neural networks (IWANN93). Lecture notes in computer science, vol 686. Morgan Kaufmann Publishers Inc., Berlin, pp 185–192

Miller GF, Todd PM, Hegde SU (1989) Designing neural networks using genetic algorithms. In: Proceedings of the third international conference on Genetic algorithms. Morgan Kaufmann Publishers Inc., San Francisco, pp 379–384

Nolfi S, Floreano D (2000) Evolutionary robotics: the biology, intelligence, and technology. MIT Press, Cambridge

Patel D (1996) Using genetic algorithms to construct a network for financial prediction. In: Proceedings of SPIE: applications of artificial neural networks in image processing. pp 204–213

Radi A, Poli R (2003) Discovering efficient learning rules for feedforward neural networks using genetic programming, chap 7. Physica-Verlag GmbH, Heidelberg, pp 133–159

Refaeilzadeh P, Tang L, Liu H (2009) Cross-validation. In: Liu L, Özsu MT (eds) Encyclopedia of database systems. Springer, Berlin, pp 532–538

Riedmiller M (1994) Rprop—description and implementation details. Technical report, University of Karlsruhe

Rumelhart DE, Hinton GE, Williams RJ (1986) Learning internal representations by error propagation, chap 8. MIT Press, Cambridge, pp 318–362

Samanta B (2011) Prediction of chaotic time series using computational intelligence. Expert Syst Appl 38(9):11406–11411

Samarasinghe S (2006) Neural networks for applied sciences and engineering. Auerbach Publications, Boston

Samaria FS, Harter AC (2002) Parameterisation of a stochastic model for human face identification. In: Applications of computer vision, vol 1994. Proceedings of the second IEEE workshop on Sarasota, FL. pp 138–142

Schwefel H-P (1981) Numerical optimization of computer models. John Wiley & Sons Inc., New York

Stanley KO, D'Ambrosio DB, Gauci J (2009) A hypercube-based encoding for evolving large-scale neural networks. Artif Life 15(2):185–212

Stanley KO, Miikkulainen R (2002) Evolving neural networks through augmenting topologies. Evol Comput 10(2):99–127

Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. Trans Evol Comp 1(1):67–82

Yao X (1999) Evolving artificial neural networks. Proc IEEE 87(9):1423–1447

Zhang G (2000) Neural networks for classification: a survey. IEEE Syst Man Cybern Soc 30(4):451–462

Zhong P, Fukushima M (2007) Regularized nonsmooth newton method for multi-class support vector machines. Optim Methods Softw 22(1):225–236

Zimmerman DW, Williams RH (1986) Modern elementary statistics, with theoretical supplement and BASIC programming. American Sciences Press, Syracuse