Neural Comput & Applic (2015) 26:1919–1928
DOI 10.1007/s00521-015-1847-6

ORIGINAL ARTICLE

Designing evolutionary feedforward neural networks using social spider optimization algorithm

Seyedeh Zahra Mirjalili · Shahrzad Saremi · Seyed Mohammad Mirjalili

Received: 1 July 2014 / Accepted: 11 February 2015 / Published online: 27 February 2015

Abstract Training feedforward neural networks (FNNs) is considered as a challenging task due to the nonlinear nature of this problem and the presence of large number of local solutions. The literature shows that heuristic optimization algorithms are able to tackle these problems much better than the mathematical and deterministic methods. In this paper, we propose a new trainer using the recently proposed heuristic algorithm called social spider optimization (SSO) algorithm. The trained FNN by SSO (FNNSSO) is benchmarked on five standard classification data sets: XOR, balloon, Iris, breast cancer, and heart. The results are verified by the comparison with five other well-known heuristics. The results prove that the proposed FNNSSO is able to provide very promising results compared with other algorithms.

Keywords Optimization · Training neural network · Social spider optimization · Feedforward neural networks · Learning

S. Z. Mirjalili · S. M. Mirjalili
Zharfa Pajohesh System (ZPS) Co.,
Unit 5, NO. 30, West 208 St., Third Sq. Tehranpars,
P. O. Box: 1653745696, Tehran, Iran

S. Saremi (✉)
School of Information and Communication Technology,
Griffith University, Nathan Campus,
Brisbane, QLD 4111, Australia
e-mail: shahrzad.saremi.mit@gmail.com;
shahrzad.saremi@griffithuni.edu.au

S. Saremi
Queensland Institute of Business and Technology,
Mt Gravatt, Brisbane, QLD 4122, Australia

1 Introduction

The problem of improving the learning of artificial neural networks (ANNs) is considered as a challenging task as the literature shows. This problem is to find optimal values for structural parameters of ANNs (mostly weights and biases) to achieve the minimum classification, predication, or approximation errors. In fact, learning is the main and key process in any type of ANNs. Due to the high dimensionality of this problem and varying search space with respect to the given data set, the problem of ANN learning enhancement is considered as a challenging task.

Generally speaking, there are three types of learning in ANNs: supervised [1, 2], unsupervised [3, 4], and reinforcement [5, 6]. As its name implies, in a supervised learning process, a supervisor provides feedbacks about the performance of an ANN based on the given training sample. So, an ANN is provided with its performance and allowed to adjust its structural parameters with respect to it. In contrary, there is no feedback from an external supervisor in unsupervised learning. In this case, an ANN has to assess its performance individually. Finally, ANNs are punished or rewarded with incorrect or correct actions during an enforced learning process. So, the feedbacks are limited to two types: right or wrong. In this case, an ANN has to adapt itself to the training samples based on the provided feedbacks. This type of learning is highly similar to learning mechanism of trained animals.

Regardless of the differences between the three types of learning in ANNs, the similarity is the objective. The ultimate goal of a learning process is the find the best structural parameters of NNs to achieve highest performance. In feedforward neural network (FNN) [7], which is the focus of this study, the most important structural parameters are weights of the connections between various

neurons in different layers and the biases of the neurons themselves. There are other parameters that are of interest as well: number of hidden neurons and number of hidden layers. Since the structure of a FNN is defined before the learning process, however, the weights and biases are mostly considered as the main variables in learning enhancement of FNN.

The conventional learning algorithm for FNNs is the so-called back-propagation (BP) algorithm [8]. This algorithm is a mathematical gradient-based algorithm that is indeed the most popular algorithm in this field. In this algorithm, the training samples are fed to the FNN and the difference between the desirable output and obtained output (error) is propagated backward to adjust the weights and biases. This iterative process is continued until the satisfaction of an end criterion, which is mostly a threshold of the error.

Although the BP algorithm can be very effective in simple and mostly linear data sets, there is one main drawback: premature convergence. The BP algorithm utilizes gradient descent. Therefore, the initial random solution is guided towards the steepest valley in the search space. In this case, the quality of the obtained solutions highly depends of the initial solution. In addition, there is high possibility of local optima stagnation. It is quite often that the error stays constant during the learning process for a long time and that there is no further improvement, which shows that the algorithm entraps in local optima. The literature shows that heuristic algorithms are promising alternatives for alleviating this main drawback in learning FNNs [9].

Heuristic optimization algorithms mostly start the optimization process by creating one or a set of random solutions [10–14]. They then improve the initial random solution(s) mostly using nature-inspired concepts until an end condition is met [15, 16]. Since heuristics methods randomly create solutions and improve them, they have high ability to avoid of local optima [17–21]. Some of the most popular algorithms in this field are as: ant lion optimizer (ALO) [22], genetic algorithm (GA) [23], particle swarm optimization (PSO) [24, 25], ant colony optimization (ACO) [26], differential evolutions (DE) [27], evolution strategy (ES) [28], and PSOGSA [29]. Nearly, the majority of these algorithms have been employed to improve the learning of FNNs.

Montana and Davis was first utilized GA to improve learning of NNs [30]. In 1990, the GA algorithm was again applied to FNN [31]. The weights and biases of FNN were the variables. Belew et al. showed that this algorithm is able to effectively enhance the learning of FNN. There are also other studies that integrate GA to NN for improving learning using different methods [32–35]. The PSO algorithm was also employed as the trainer for FNN in many studies [36–41]. Although ACO is suitable for combinatorial problems, it was shown in [42–45] that this algorithm is able to provide very promising results when applying to NNs as well. Some other heuristic-based learning algorithms in the literature are as follows: DE-based trainer [46, 47], ABC-based trainer [48, 49], gravitational search algorithm (GSA)-based trainer [50, 51], Tabu search (TS)-based trainer [52, 53], bio-geography-based optimization (BBO)-based trainer [54], and ES-based trainer [55], magnetic optimization algorithm (MOA)-based trainer [56], and grey wolf optimizer (GWO) [57].

Despite the merits of the above-mentioned works, the problem of local optima entrapment still persists. In addition, there is a theorem in the field of heuristics called No Free Lunch [58] that says there is no optimization algorithm for solving all problems. Since FNNs are trained for different data sets, there are possibilities that one algorithm performs well on a data set but worse on another. These reasons allow researcher to investigate the efficiencies of new algorithms in enhancing learning of FNNs. This is also the contribution of this study, in which the recently proposed social spider optimization (SSO) algorithm [59, 60] is chosen to be embedded to FNNs. The only similar work in the literature is that of Pereira et al. [61], in which the SSO algorithm has been employed to train neural networks and classify Ionosphere, Satimage, Diadol, Mea, and Spiral data sets. They only optimized the weights of the connections, but this work optimizes the weights and biases of MLPs simultaneously. Therefore, the problem of learning MLP is more difficult in this work. We also solve five different standard classification data sets. Other contribution of this work is the comparison of different heuristic algorithms on classification data sets. The rest of the paper is written as follows.

Section 2 presents the rudimentary concepts of FNNs. The general concepts and mechanism of SSO algorithm is provided in Sect. 3. The new SSO-based learning process of FNN is proposed in Sect. 4. Section 5 includes the results, discussion, and analysis. Eventually, Sect. 6 concludes the study and advises a couple of directions for future studies.

2 Feedforward neural network

As the name of FNN show, data are cascaded in one direction between the neurons in the network [7]. The neurons are arranged in parallel layers, and each neuron in tth layer receives data from the neurons $t-1$th layer and delivers data to the neurons in $t+1$th layer. The structure of FNN with 1 input, 1 hidden, and 1 output layer is illustrated in Fig. 1.

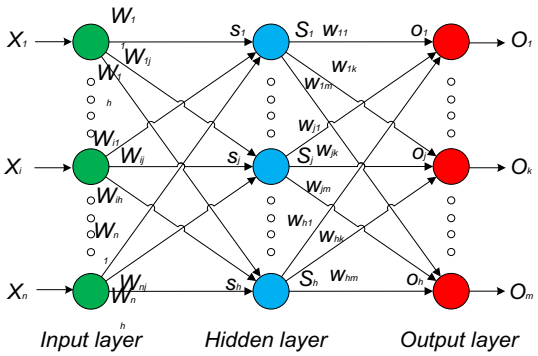

Fig. 1 Three-layer FNN

As shown in Fig. 1, the input layer is the layer that is given the inputs. Then, the inputs are multiplied by the weights and delivered as the inputs of the hidden layer. The same scenario occurs for the data transition between the hidden layer and output. The actual outputs of neurons are calculated by a transfer function. Interested authors are referred to [7] for more details. Although the structure and workflow of FNN is very simple, it has been proven that a three-layer FNN is able to approximate any given function [62].

When providing inputs for a FNN, the weighted sum of inputs are first calculated as follows:

$$s_{\mathrm{j}} = \sum_{\mathrm{i}=1}^{n} (W_{\mathrm{ij}}.X_{\mathrm{i}}) - \theta_{\mathrm{j}}, \quad j = 1, 2, \ldots h \tag{2.1}$$

where n is the number of the input nodes, W_{ij} shows the connection weight from the ith node in the input layer to the jth node in the hidden layer, θ_j is the bias (threshold) of the jth hidden node, and X_i indicates the ith input.

Then, a sigmoid function defines the final output of each hidden node as follows:

$$S_{\mathrm{j}} = \mathrm{sigmoid}\left(s_{\mathrm{j}}\right) = \frac{1}{\left(1 + \exp\left(-s_{\mathrm{j}}\right)\right)}, \quad j = 1, 2, \ldots h \tag{2.2}$$

Finally, the same two steps define the output of the network as follows:

$$o_{\mathrm{k}} = \sum_{\mathrm{j}=1}^{\mathrm{h}} (w_{\mathrm{jk}}.S_{\mathrm{j}}) - \theta_{\mathrm{k}}', \quad k = 1, 2, \ldots, m \tag{2.3}$$

$$O_{\mathrm{k}} = \mathrm{sigmoid}(o_{\mathrm{k}}) = \frac{1}{(1 + \exp(-o_{\mathrm{k}}))}, \quad k = 1, 2, \ldots, m \tag{2.4}$$

where w_{jk} is the connection weight from the jth hidden node to the zth output node, and $\theta_k^{'}$ is the bias (threshold) of the kth output node.

As discussed in Sect. 1, the most important structural parameters of FNN are weights and biases. Equations (2.1) and (2.3) show how they define the final output of FNN. We propose a new learning algorithm in the next section to find the optimal values for weights and biases.

3 Social Spider Optimization Algorithm

The social spider optimization (SSO) algorithm was proposed by Cuevas et al. in [60]. This algorithm is a swarm-based algorithm, which mimics the social intelligence of spiders who live in a colony. In fact, the social communication of spiders using the vibration throughout the web was the main inspiration of this algorithm. In this algorithm, the search space of optimization problems is considered as the web and the search agents as the spiders of the colony. The search agents are divided to two types: males (M) and females (F). The weight of each spider is defined as its fitness. The general operators that apply for modifying spiders (candidate solutions) are defined based on the gender. In the SSO algorithm, similar to a spider colony, the number of females is higher than the number of males and initially considered as 60 to 90 % of the population.

During optimization, spiders communicate with vibrating the strings in web. The vibration that a spider receives is defined with respect to the size of the sender spider and its distance. The mathematical model that was proposed by Cuevas et al. is as follows [60]:

$$Vibs_{\mathrm{i}} = w_{\mathrm{j}} e^{\mathrm{d}_{\mathrm{i,j}}^2}$$

where w_j indicates the weight of the jth spider, and $d_{i,j}$ is the Euclidean distance between ith and jth spiders.

It is assumed in SSO that every spider is only able to feel three vibrations from other spiders as follows:

- The nearest spider subject to having higher fitness ($Vibc_{\mathrm{i}}$).
- The best spider in the swarm ($Vibb_{\mathrm{i}}$).
- The nearest female, which is applicable for men ($Vibf_{\mathrm{i}}$).

As mentioned above, the operator for position updating of search agents is defined based on their gender. A female updates its position as follows [59]:

$$X_{\mathrm{i}}(t+1) = X_{\mathrm{i}}(t) + (\alpha \cdot Vibc_{\mathrm{i}} \cdot (S_c - X_{\mathrm{i}}(t)) + \beta \cdot Vibb_{\mathrm{i}} \cdot (S_b - X_{\mathrm{i}}(t)) + \delta \cdot (r - 0.5)) \tag{3.1}$$

where α, β, δ, and r are random values in [0,1], S_c indicates the closest best neighbour, and S_b shows the fittest spider in the swarm.

This formula is applicable when a female is attracted towards the source of vibrations. In SSO, it is assumed that females can make a decision randomly to move towards or outwards the source. If they decide to move away from the source, the following formula is utilized [59]:

$$X_i(t+1) = X_i(t) - (\alpha \cdot Vibc_i \cdot (s_c - X_i(t)) + \beta \cdot Vibb_i \cdot (S_b - X_i(t)) + \delta \cdot (r - 0.5)) \quad (3.2)$$

where α, β, δ, and r are random values in [0,1], S_c indicates the closest best neighbour, and S_b shows the fittest spider in the swarm.

Male spiders have different position updating procedure. Firstly, they are divided to two groups: dominated (D) and non-dominated (ND). ND spiders tend to be attracted toward females, whereas D spiders move towards the centre of male population. This behaviour is inspired by the tendency of ND spider to go to the more populated male areas and feed from the leftovers of other males and grow up to become an ND spider in future [59]. In order to define what make spider D, we have to calculate the median of the fitness (weight) of all male spiders in every iteration. Then, the spiders with fitness below the calculated median are considered as D-type and the rest are ND-type spiders. The mathematical model that was proposed to mimic the movement of males is as follows [59]:

For ND-type spiders:

$$X_i(t+1) = X_i(t) + \left(\alpha \cdot Vibf_i \cdot \left(S_f - X_i(t)\right) + \delta \cdot (r - 0.5)\right) \quad (3.3)$$

For D-type spiders:

$$X_i(t+1) = X_i(t) + \left(\alpha \cdot \left(\frac{\sum_{j=1}^{N_m} m_j^k \cdot W_{N_f} + j}{\sum_{j=1}^{N_m} W_{N_f}}\right) - X_i(t)\right) \quad (3.4)$$

where S_f is the closest female to ith make, and β, δ, and r are random values in [0,1].

It may be seen in these equations that ND-type spiders are only able to move toward females and that there is no backward movement in contrast to female's movement methods.

The last mechanism of the SSO algorithm for modifying the search agents is the mating operator. The ND males are required to mate with females who are within a certain radius called mating radius. There might be more than one male and female in the mating radius. Therefore, a roulette wheel mechanism randomly chooses parents proportional to their fitness values. A new spider is then constructed by the combination of genes (variables) of males and females. After producing new spiders, the fitness of them are calculated and compared to the worst spiders in the population. If any of the new spiders are fitter than a spider, it is added to the population with eliminating the less fit spider.

The SSO algorithm follows the following steps to solve optimization problems:

(a) Begin the optimization process by generating spiders in random positions on the search space.
(b) The spiders are assigned a gender (65–90 % female and the rest male).
(c) The fitness of spiders is calculated by the objective function.
(d) The best spider in the swarm, best female, and closest spider to each spider are defined.
(e) Position of a spider is updated by (3.1) or (3.2) if it is female.
(f) Position of a spider is updated by (3.3) or (3.4) if it is male.
(g) The males and females located in the mating radius mate to create new spiders.
(h) Produced new spiders are substituted with the worst spiders if they have better fitness value.
(i) Steps c–h is iterated until the satisfaction of an end criterion.

Cuevas et al. [60] proved that the SSO algorithm is able to provide very competitive results compared with the well-known algorithms such as PSO and ABC. They observed and concluded that the division of spiders based on gender maintains the diversity of the balance between exploration and exploitation by each group. In addition, the vibration mechanism and movement methods promote exploration.

These observations also motivates our attempts to propose a SSO-based FNN trainer and investigate the performance of this algorithm in training NNs.

4 SSO-based Feedforward Neural Network

Ad discussed in Sect. 1, the learning of FNN mostly refers to the process of finding the best values of weights and biases. However, improving the learning of FNN is done with three methods when using heuristic algorithms [31]:

1. Defining the weights and biases.
2. Defining the structure of FNN.
3. Tuning the parameters of other learning methods (e.g. learning rate and momentum in BP).

In the first method, which is the most common method, the weights and biases are optimized by a meta-heuristic. The second method deals with optimizing the structure of FNN such as connections between neurons, number of neurons, number of hidden nodes, and number of hidden layers. The third method employs a heuristic algorithm as auxiliary method for another learner. In fact, meta-heuristics plays the role of parameter tuner in this case. In this study, we concentrate on the first method.

In order to formulate the problem of learning enhancement of FNN for a meta-heuristic, two steps should be taken:

1. Representing the weights and biases in a suitable format.
2. Defining the objective function.

The first step represents the variable of the problem for a given meta-heuristic. There are three main methods in the literature: vector, binary, and matrix approaches. Since the SSO algorithm employed in this work considers solutions in a vector, we choose the vector representation method [31, 39, 50]. An example of this approach is shown in Fig. 2.

Figure 2 shows that the vector representation is a very simple method, in which the weights and biases are just added to a vector in order to be delivered to heuristic algorithms. After defining the vector of variables, an objective function should define its fitness. In FNN, generally speaking, the performance is defined by looking at the desirable output and the actual output of the network. The most common performance metric in the literature is mean squared error (MSE), which is calculated as follows:

$$\mathrm{MSE} = \sum_{i=1}^{m} \left(o_{\mathrm{i}}^{\mathrm{k}} - d_{\mathrm{i}}^{\mathrm{k}}\right)^2 \tag{4.1}$$

where m is the number of outputs, d_i^k is the desired output of the ith input unit when the kth training sample is used, and o_i^k is the actual output of the ith input unit when the kth training sample appears in the input.

The key point here is that there is always more than one training sample in data sets. Therefore, a given FNN should be assessed based on its performance on all training samples. In this case, average of the MSE on all training samples is fruitful.

With problem representation and the objective function, the FNN is ready to be trained by the SSO algorithm. The flow chart of the proposed training algorithm is illustrated in Fig. 3.

5 Results and discussion

In this section, we employ five data sets to benchmark the performance of the proposed method as presented in Table 1. This table show that we select five standard classification data sets from the University of California at Irvine (UCI) Machine Learning Repository [63]: XOR, balloon, Iris, breast cancer, and heart. It should be noted that the number of attributes is increased from the first to the last data set. We deliberately choose this set of data sets with different attributes to challenge the proposed algorithm and observe its performance. Obviously, the number of hidden nodes, weights, and biases are increased proportional to the number of attributes of data sets. To define the structure of FNNs for solving each data set, we chose $2 \times N+1$ hidden nodes where N is the number of attributes as per the recommendation of [64]. Other details of the data sets are available in Table 1.

For data verification, we compare the results with PSO and ACO as the best candidates and most well-known algorithms in the family of swarm-based optimization techniques. In addition, GA, ES, and PBIL are chosen as the best representatives of evolutionary algorithms. The initial values for the main parameters of SSO and these algorithms are provided in Table 2.

For generating the results, each algorithm is run 10 times and the average (AVE) and standard deviation (STD) of MSE are chosen as the comparison metrics. Average of MSE will indicate the ability of algorithms in avoiding local solutions. In addition, the standard deviations show the variation of the results and stability of algorithm in avoiding local solutions. Another comparison metric

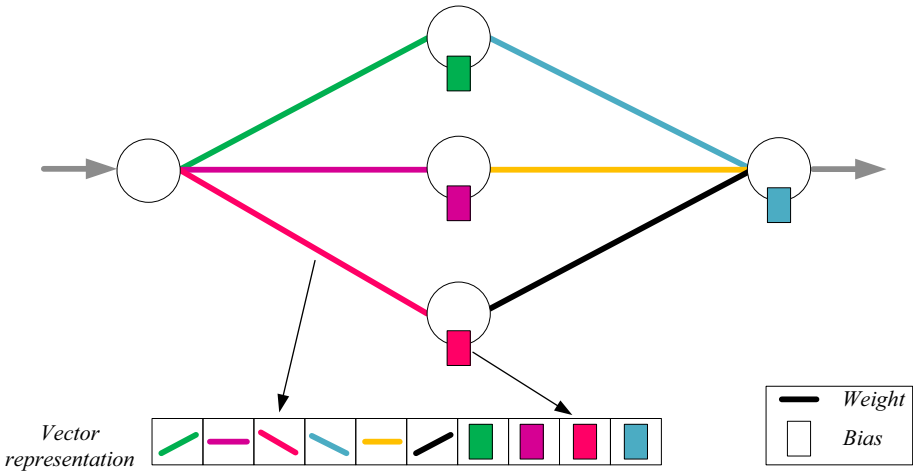

Fig. 2 Vector representation of weights and biases for heuristic algorithms

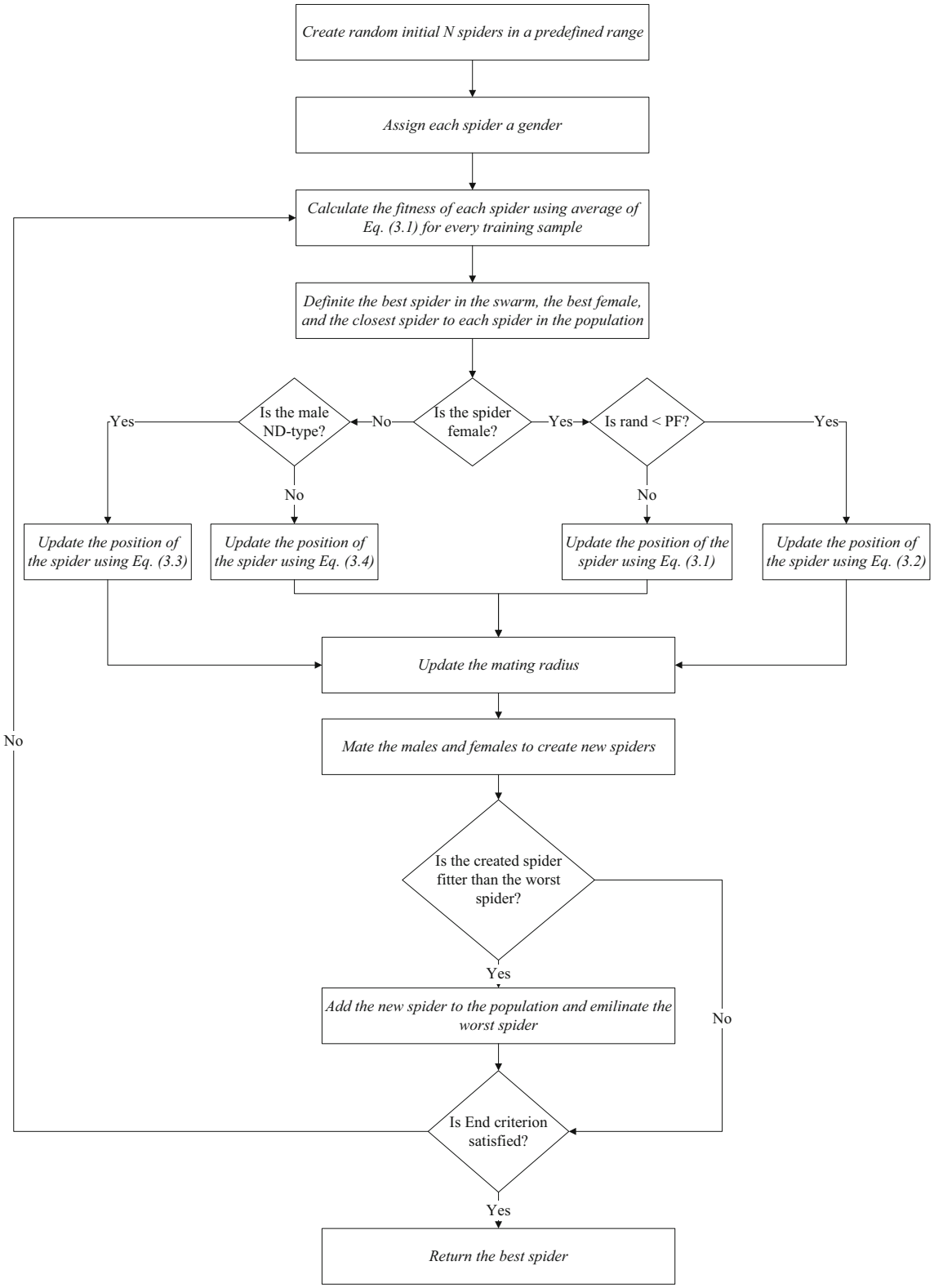

Fig. 3 General steps of the proposed SSO-based trainer

reported in the results is the classification rate. We chose the best structure over 10 runs and calculate the classification rate using the test samples. This metric will assist us to see how well an algorithm in providing accurate results is. After all, we report the results in Tables 3, 4, 5, 6 and 7. Please note that we name the proposed learning method of

Table 1 Data sets

Classification data sets	Number of attributes	Number of training samples	Number of test samples	Number of classes
3-Bit XOR	3	8	8 as training samples	2
Balloon	4	16	16 as training samples	2
Iris	4	150	150 as training samples	3
Breast cancer	9	599	100	2
Heart	22	80	187	2

this work FNNSSO and highlight the best results in boldface in the table of results.

The results of the algorithm on XOR data set are provided in Table 3. This table shows that the minimum average of MSE is provided by the proposed FNNSSO algorithm. The standard deviations show that this algorithm performs very stable on this data set. The results of this algorithm is significantly better than FNNPSO and FNNACO in this data set, so this proves the merits of this algorithm in avoiding local solutions compared with other swarm-based optimization techniques. The FNNGA algorithm shows very competitive results compared with FNNPSO in terms of local optima avoidance and accuracy. It can be seen in Table 3 that the classification rate for FNNGA and FNNSSO are both equal to 100 %.

The results of the algorithms on the balloon data sets are almost similar to those of the XOR data set. Table 4 shows that the best result belong to FNNGA, closely followed by FNNSSO. Although the classification accuracies are similar for all algorithms, the average and standard deviation of MSE are different. Again, the results show that the local optima avoidance of the majority of evolutionary algorithms are higher than that of FNNPSO and FNNACO. As a swarm-based algorithm, however, FNNSSO provides very competitive results compared to evolutionary-based FNN trainers.

The results of Iris data set are shown in Table 5. The results are highly consistent with those of Table 3, in which FNNSSO shows the best results for AVE, STD, and classification rate. The significantly improved MSE and classification accuracy of FNNSSO algorithm are noticeable. The results of evolutionary algorithms (especially classification accuracy) are again better than FNNPSO and FNNACO.

The most difficult data sets in terms of dimensionality of learning problems are breast cancer and heart data sets. It may be observed in Tables 6 and 7 that the average and standard deviation of MSE are much better for FNNSSO algorithm This highly proves that the SSO algorithm is

Table 2 Initial parameters of algorithms

Algorithm	Parameter	Value
SSO	*PF*	0.7
	Population size	50 for XOR and balloon, 200 for the rest
	Maximum number of generations	250
PSO	Topology	Fully connected
	Cognitive constant (C_1)	1
	Social constant (C_2)	1
	Inertia constant *(w)*	0.3
	Population size	50 for XOR and balloon, 200 for the rest
	Maximum number of iterations	250
ACO	Initial pheromone (τ_0)	1e-06
	Pheromone update constant (Q)	20
	Pheromone constant (q_0)	1
	Global pheromone decay rate (p_g)	0.9
	Local pheromone decay rate (p_l)	0.5
	Pheromone sensitivity (α)	1
	Visibility sensitivity (β)	5
	Population size	50 for XOR and balloon, 200 for the rest
	Maximum number of iterations	250
GA	Type	Real coded
	Selection	Roulette wheel
	Crossover	Single point (probability = 1)
	Mutation	Uniform (probability = 0.01)
	Population size	50 for XOR and balloon, 200 for the rest
	Maximum number of generations	250
ES	λ	10
	σ	1
	Population size	50 for XOR and balloon, 200 for the rest
	Maximum number of generations	250
PBIL	Learning rate	0.05
	Good population member	1
	Bad population member	0
	Elitism parameter	1
	Mutational probability	0.1
	Population size	50 for XOR and balloon, 200 for the rest
	Maximum number of generations	250

Table 3 Statistical results for XOR data set over 10 independent runs

Algorithm	MSE (AVE ± STD)	Classification rate (%)
FNNSSO	**2.8075e-05 ± 0.0000**	**100.00**
FNNPSO	0.084050 ± 0.035945	37.50
FNNACO	0.180328 ± 0.025268	62.50
FNNGA	0.000181 ± 0.000413	**100.00**
FNNES	0.118739 ± 0.011574	62.50
FNNPBIL	0.030228 ± 0.039668	62.50

Table 4 Statistical results for balloon data set over 10 independent runs

Algorithm	MSE (AVE ± STD)	Classification rate (%)
FNNSSO	1.7395e-15 ± 0.0000	**100.00**
FNNPSO	0.000585 ± 0.000749	**100.00**
FNNACO	0.004854 ± 0.007760	**100.00**
FNNGA	**5.08e-24 ± 1.06e-23**	**100.00**
FNNES	0.019055 ± 0.170260	**100.00**
FNNPBIL	2.49e-05 ± 5.27e-05	**100.00**

Table 5 Statistical results for iris data set over 10 independent runs

Algorithm	MSE (AVE ± STD)	Classification rate
FNNSSO	**0.0210 ± 3.6571e-18**	**91.3333**
FNNPSO	0.228680 ± 0.057235	37.33 %
FNNACO	0.405979 ± 0.053775	32.66 %
FNNGA	0.089912 ± 0.123638	89.33 %
FNNES	0.314340 ± 0.052142	46.66 %
FNNPBIL	0.116067 ± 0.036355	86.66 %

Table 6 Statistical results for breast cancer data set over 10 independent runs

Algorithm	MSE (AVE ± STD)	Classification rate (%)
FNNSSO	**0.001600 ± 2.28 e-19**	**98.00**
FNNPSO	0.034881 ± 0.002472	11.00
FNNACO	0.013510 ± 0.002137	40.00
FNNGA	0.003026 ± 0.001500	**98.00**
FNNES	0.040320 ± 0.002470	06.00
FNNPBIL	0.032009 ± 0.003065	07.00

suitable for training FNNs due the high dimensional nature of this problem. The high exploration of this algorithm resulted in such a performance on these two challenging data sets. The classification accuracy of this algorithm is also good on these two data sets.

To sum up, the FNNSSO shows superior results compared with FNNGA on the majority of data sets. This is due to high exploration ability of this algorithm. Another behaviour observed is the better results of the evolutionary algorithms compared to the swarm-based optimization techniques employed in this work except SSO. This is because of the recombination operators of evolutionary algorithm which highly promote exploration. Despite this high exploration, the result of this work evidences that SSO algorithm also has a very high exploration.

Table 7 Statistical results for heart data set over 10 independent runs

Algorithm	MSE (AVE ± STD)	Classification rate
FNNSSO	**0.0627 ± 1.4628e-17**	67.5000
FNNPSO	0.188568 ± 0.008939	68.75 %
FNNACO	0.228430 ± 0.004979	00.00 %
FNNGA	0.093047 ± 0.022460	58.75 %
FNNES	0.192473 ± 0.015174	**71.25** %
FNNPBIL	0.154096 ± 0.018204	45.00 %

Another behaviour observed in the results is the better performance of the FNNSSO algorithm on data sets as the difficulty increases. The results show that FNNSSO provide very good results on Iris, cancer, and heart data sets. The number of variables for these data sets to be optimized is 75, 209, and 1081, respectively. This evidences that the SSO algorithm is very good in optimizing high dimensional problems, which again is due to the high exploration of this algorithm. However, the classification accuracies are not as good as local optima avoidance, which is due to the exploitation of the SSO algorithm. Therefore, it seems that exploration of SSO is better that its exploitation, which assist this algorithm to show very high local optima avoidance and reasonable classification accuracy.

6 Conclusion

This work proposed a new training algorithm based on the recently proposed SSO algorithm for FNNs. The vector representation method was chosen to provide SSO with the weights and biases for optimization. The objective function was to minimize the average of MSE on all training samples. The performance of the proposed FNN trainer was benchmarked on five standard classification problems: XOR, balloon, Iris, breast cancer, and heart. The results of the proposed FNNSSO algorithm were compared to five other algorithms in the literature for verification: PSO, ACO, GA, ES, and PBIL. The results showed that the proposed method is very efficient in learning FNN, which is due to exploration and high local optima avoidance of this algorithm. We also observed that the results of FNNSSO were better on the majority of the data sets in terms of classification accuracy. Another finding was the

relevancy of the performance of the FNNSSO algorithm and difficulty of the data set in terms of number of features. It was observed that the proposed algorithm outperforms other algorithms in Iris, breast cancer, and heart data sets, which is again due to high exploration of this algorithm. The paper also considered the comparison of other algorithms employed in this work. We found evolutionary algorithms outperformed swarm-based algorithms. The reason was discussed in terms of the intrinsic promotion in exploration using recombinations operators in evolutionary algorithms.

For future work, it is recommended to see the effectiveness of the proposed FNNSSO in training other types of NNs. In addition, methods for improving the exploitation of SSO algorithm are worth studying.

References

1. Reed RD, Marks RJ (1998) Neural smithing: supervised learning in feedforward artificial neural networks. Mit Press, Cambridge
2. Caruana R, Niculescu-Mizil (2006) An empirical comparison of supervised learning algorithms. In: Proceedings of the 23rd international conference on Machine learning, pp 161–168
3. Hinton GE, Sejnowski TJ (1999) Unsupervised learning: foundations of neural computation. MIT press, Cambridge
4. Wang D (2001) Unsupervised learning: foundations of neural computation. AI Mag 22:101
5. Barto A (1997) Reinforcement learning. Neural systems for control, pp 7–29
6. Barto AG (1998) Reinforcement learning: an introduction. MIT press, Cambridge
7. Bebis G, Georgiopoulos M (1994) Feed-forward neural networks. Potentials, IEEE 13:27–31
8. Hertz J (1991) Introduction to the theory of neural computation, vol 1. Basic Books, New York
9. Branke J (1995) Evolutionary algorithms for neural network design and training. In: Proceedings of the first nordic workshop on genetic algorithms and its applications
10. Saremi S, Mirjalili S (2013) Integrating chaos to biogeography-based optimization algorithm. Int J Comput Commun Eng 2:655–658
11. Saremi S, Mirjalili S, Lewis A (2014) How important is a transfer function in discrete heuristic algorithms. Neural Comput Appl. doi:10.1007/s00521-014-1743-5
12. Saremi S, Mirjalili S, Lewis A (2014) Biogeography-based optimisation with chaos. Neural Comput Appl 25(5):1077–1097
13. Saremi S, Mirjalili SM, Mirjalili S (2014) Chaotic krill herd optimization algorithm. Procedia Technol 12:180–185
14. Saremi S, Mirjalili SM, Mirjalili S (2014) Unit cell topology optimization of line defect photonic crystal waveguide. Procedia Technol 12:174–179
15. Mirjalili S, Hashim SM (2011) BMOA: binary magnetic optimization algorithm. 3rd international conference on machine learning and computing (ICMLC 2011), Singapore, pp 201–206
16. Mirjalili S, Lewis A (2014) Adaptive gbest-guided gravitational search algorithm. Neural Comput Appl 25(7–8):1569–1584
17. Mirjalili S, Lewis A, Sadiq AS (2014) Autonomous particles groups for particle swarm optimization. Arab J Sci Eng 39(6): 4683–4697
18. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61
19. Mirjalili S, Mirjalili SM, Yang XS (2013) Binary bat algorithm. Neural Comput Appl 25(3–4):663–681
20. Mirjalili S, Rawlins T, Hettenhausen J, Lewis A (2013) A comparison of multi-objective optimisation metaheuristics on the 2D airfoil design problem. ANZIAM J 54:C345–C360
21. Mirjalili S, Wang GG, Coelho LDS (2014) Binary optimization using hybrid particle swarm optimization and gravitational search algorithm. Neural Comput Appl 25(6):1423–1435
22. Mirjalili S (2015) The ant lion optimizer. Adv Eng Softw. doi:10.1016/j.advengsoft.2015.01.010
23. Holland JH (1992) Genetic algorithms. Sci Am 267:66–72
24. Eberhart RC, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proceedings of the sixth international symposium on micro machine and human science, pp 39–43
25. Mirjalili S, Lewis A (2012) S-shaped versus V-shaped transfer functions for binary particle swarm optimization. Swarm Evolut Comput 9:1–14. doi:10.1016/j.swevo.2012.09.002
26. Dorigo M, Birattari M (2010) Ant colony optimization. In: Encyclopedia of machine learning. Springer, New York, pp 36–39
27. Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11:341–359
28. Beyer H-G, Schwefel H-P (2002) Evolution strategies—a comprehensive introduction. Nat Comput 1:3–52
29. Mirjalili S, Mohd Hashim SZ (2010) A new hybrid PSOGSA algorithm for function optimization. In: IEEE international conference on computer and information application (ICCIA 2010), China, pp 374–377
30. Montana DJ, Davis L (1989) Training feedforward neural networks using genetic algorithms. In: IJCAI, vol 89, pp 762–767
31. Belew RK, McInerney J, Schraudolph NN (1990) Evolving networks: using the genetic algorithm with connectionist learning
32. Leung FH-F, Lam H-K, Ling S-H, Tam PK-S (2003) Tuning of the structure and parameters of a neural network using an improved genetic algorithm. Neural Netw IEEE Trans 14:79–88
33. Kitano H (1994) Neurogenetic learning: an integrated method of designing and training neural networks using genetic algorithms. Phys D 75:225–238
34. Chen M-S, Liao FH (1998) Neural networks training using genetic algorithms. In: IEEE international conference on systems, man, and cybernetics, 1998, vol 3. IEEE, pp 2436–2441
35. Whitley D, Starkweather T, Bogart C (1990) Genetic algorithms and neural networks: optimizing connections and connectivity. Parallel Comput 14:347–361
36. Meissner M, Schmuker M, Schneider G (2006) Optimized particle swarm optimization (OPSO) and its application to artificial neural network training. BMC Bioinform 7:125
37. Gudise VG, Venayagamoorthy GK (2003) Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. In: Swarm intelligence symposium, 2003. SIS'03. Proceedings of the 2003 IEEE, pp 110–117
38. Al-kazemi B, Mohan CK (2002). Training feedforward neural networks using multi-phase particle swarm optimization. In: Neural information processing, 2002. ICONIP'02. Proceedings of the 9th international conference on, 2002, pp 2615–2619
39. Zhang J-R, Zhang J, Lok T-M, Lyu MR (2007) A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training. Appl Math Comput 185:1026–1037
40. Mendes R, Cortez P, Rocha M, Neves J (2002) Particle swarms for feedforward neural network training. Learning 6(1)
41. Ismail A, Engelbrecht AP (1999) Training product units in feedforward neural networks using particle swarm optimization. In: Proceedings of the international conference on artificial intelligence, pp 36–40

42. Socha K, Blum C (2007) An ant colony optimization algorithm for continuous optimization: application to feed-forward neural network training. Neural Comput Appl 16:235–247
43. Blum C, Socha K (2005) Training feed-forward neural networks with ant colony optimization: An application to pattern classification. In: Hybrid Intelligent Systems, 2005. HIS'05. Fifth international conference on, p. 6
44. Hong B-R, Jin F-H, Gao Q-J (2003) Multi-layer feedforward neural network based on ant colony system. J Harbin Inst Technol 7:016
45. Liu YP, Wu MG, Qian JX (2006) Evolving neural networks using the hybrid of ant colony optimization and BP algorithms. In: Advances in neural networks-ISNN 2006, ed: Springer, pp 714–722
46. Ilonen J, Kamarainen J-K, Lampinen J (2003) Differential evolution training algorithm for feed-forward neural networks. Neural Process Lett 17:93–105
47. Slowik A, Bialko M (2008) Training of artificial neural networks using differential evolution algorithm. In: Human system interactions. Conference on, pp 60–65
48. Karaboga D, Akay B, Ozturk C (2007) Artificial bee colony (ABC) optimization algorithm for training feed-forward neural networks. In Modeling decisions for artificial intelligence, ed: Springer, 2007, pp 318–329
49. Ozturk C, Karaboga D (2011) Hybrid artificial bee colony algorithm for neural network training. In: IEEE congress on evolutionary computation (CEC), 2011. IEEE, New Orleans, pp 84–88
50. Mirjalili S, Mohd SZ (2012) Hashim, and H. Moradian Sardroudi, "Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm,". Appl Math Comput 218:11125–11137
51. Ghalambaz M, Noghrehabadi A, Behrang M, Assareh E, Ghanbarzadeh A, Hedayat N (2011) A hybrid neural network and gravitational search algorithm (HNNGSA) method to solve well known Wessinger's equation. World Acad Sci Eng Technol 73:803–807
52. Battiti R, Tecchiolli G (1995) Training neural nets with the reactive tabu search. Neural Netw IEEE Trans 6:1185–1200
53. Kalinli A, Karaboga D (2004) Training recurrent neural networks by using parallel tabu search algorithm based on crossover operation. Eng Appl Artif Intell 17:529–542
54. Mirjalili S, Mirjalili SM, Lewis A (2014) Let a biogeography-based optimizer train your multi-layer perceptron. Inf Sci 269:188–209
55. Wienholt W (1993) Minimizing the system error in feedforward neural networks with evolution strategy. In: ICANN'93. Springer, London, pp. 490–493
56. Mirjalili S, Sadiq AS (2011) Magnetic optimization algorithm for training multi layer perceptron. In: Communication software and networks (ICCSN), 2011 IEEE 3rd international conference on, 2011, pp 42–46
57. Mirjalili S (2015) How effective is a Grey wolf optimizer in training multi-layer perceptrons. Appl Intell. doi:10.1007/s10489-014-0645-7
58. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. Evolutionary Comput IEEE Trans 1:67–82
59. Cuevas E, Cienfuegos M (2014) A new algorithm inspired in the behavior of the social-spider for constrained optimization. Exp Syst Appl: Int J. 41:412–425
60. Cuevas E, Cienfuegos M, Zaldívar D, Pérez-Cisneros M (2013) A swarm optimization algorithm inspired in the behavior of the social-spider. Expert Syst Appl 40:6374–6384
61. Pereira L, Rodrigues D, Ribeiro P, Papa J, Weber SA (2014) Social-spider optimization-based artificial neural networks training and its applications for parkinson's disease identification. In: Computer-based medical systems (CBMS), 2014 IEEE 27th International symposium on, 2014, pp 14–17
62. Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. Neural Netw 2:359–366
63. Asuncion A, Newman D (2007) UCI machine learning repository
64. Wdaa I, Sttar A (2008) Differential evolution for neural networks learning enhancement. Universiti Teknologi Malaysia, Faculty of Computer Science and Information System