Wireless Personal Communications (2019) 108:2241–2260
https://doi.org/10.1007/s11277-019-06520-w

Classification of Sonar Targets Using an MLP Neural Network Trained by Dragonfly Algorithm

Mohammad Khishe[1] · **Abbas Safari**[1]

Published online: 10 May 2019

Abstract
Due to the compatibility of the designed classifiers with MLP Neural Networks (MLP NNs), in this article, MLP NNs have been used to identify and classify active and passive sonar targets. On the one hand, the great importance of precise and immediate classification of sonar targets, and on the other hand, being trapped in local minimums and the low convergence speed in classic MLP NNs have led the newly proposed Dragonfly Algorithm (DA) to be offered for training MLP NNs. In order to assess the performance of the designed classifier, this algorithm have been compared with BBO, GWO, ALO, ACO, GSA and MVO algorithms in terms of precision of classification, convergence speed and the ability to avoid local optimum. To have a comprehensive comparison, the three sets of active and passive data were used. Simulation results indicate that DA-based classification have better results in all three datasets compared to benchmark algorithms.

Keywords Sonar · Classification · Dragonfly · Multi-layer perceptron neural network

1 Introduction

Due to the complex physical properties of Sonar targets, classification of the real target and the false alarms has become a significant and widely used area for researchers and industrialists working in this field. Unwanted signals in the underwater environment have various sources including noise, reverberation and clutter [1]. Since the reverberation has the same domain and are homogenous with original ping, it is easy to distinguish them from the real target [2]. When the seabed changes a lot and when there are different kinds of beds, echoes returning from the seabed will have target-like properties, in a way that even probability density function of the real target and the seabed are very much alike. This false alarm is called clutter [3]. Due to the great similarity of their reflected echoes, it's very hard to classify clutter and real target.

In recent years the use of NNs to classify sonar targets has been very common and many efforts have been put into this area [4]. For most applications of MLP NNs, optimized [5] or standard back-propagation algorithms [6] are used as learning methods. Many methods

✉ Mohammad Khishe
m_khishe@alumni.iust.ac.ir

[1] Department of Electrical Engineering, Imam Khomeini Marine University, Nowshahr, Iran

based on differentiation have been used for training the neural network, including gradient descent, Kalman filter, decoupled Kalman filter, and back- propagation. Back-propagation algorithm is based on the gradient which has some deficiencies such as slow convergence [7] and applicability in small areas [8]; therefore, it's not reliable for practical applications.

The ultimate goal of the learning process in NNs is to find the best combination of their weights and biases, so that we'd have the lowest error in training the network and in the test samples [9]. Reference [10] shows that the meta-heuristic optimization algorithms can replace the gradient-based learning algorithms, since the Stochastic nature of these algorithms prevents getting stuck in local optimum, increases the convergence speed, and reduces classification errors [11].

Here we briefly mention the NFL theorem, in which the abbreviation stands for "No free lunch" [12, 13]. This theorem logically proves that there is no appropriate meta-heuristic algorithm for solving all the optimization problems. In other words, a specific meta-heuristic algorithm could have promising results for a set of problems, but do poorly over another set. Obviously, the NFL makes this field of study very active and leads to the promotion of current methods and proposition of new meta-heuristic algorithms every year. Meta-heuristic methods used for training NNs in recent years include Genetic Algorithms (GA) [14], Simulated Annealing (SA) [15, 16], Magnetic Optimization Algorithm (MOA) [17], Biogeography-Based Optimization (BBO) [18] and Gray Wolf and Modified Grey Wolf Optimizer (GWO) [19].

One of the aspects of the similarity of meta-heuristic algorithms is dividing the search space into two phases: exploration and exploitation. Exploration phase occurs when the algorithm converges towards reliable solutions. In this phase, the population undergoes slight changes. In many cases, due to the stochastic nature of meta-heuristic algorithms, there is no clear boundary between the two phases; in other words, the imbalance between the two phases makes the algorithm stuck in a local minimum. DA easily recognizes the boundary between the exploration and the exploitation phase and converges towards more reliable solutions. Due to the two aforementioned reasons, i.e. NFL theorem and the DA's ability for regulating exploitation and exploration phase, this paper suggest the use of AD for training an MLP NN to classify sonar targets with high accuracy and in real time.

The rest of the paper is organized as follows. Section 2 introduces MLP NNs. Section 3 discusses the general aspects of DA. The proposed training scheme is described in Sect. 4. Section 5 presents the various sonar data sets. Simulation results are discussed in Sect. 6. Finally, Sect. 7 concludes the article.

2 MLP NN

Figure 1 shows a three-layer MLP NN where, n represents the number of input nodes, h represents the number of hidden nodes and m denotes the number of output nodes [19]. As show, there are one-way connections between the nodes of the MLP NN which is from the family of Feed forward NNs. The output of the MLP NN is calculated by Eq. 1:

$$S_j = \sum_{i=1}^{n} \left(W_{ij} \cdot X_i\right) - \theta_j, \quad j = 1, 2 \ldots h \tag{1}$$

where n is the number of input nodes, W_{ij} is the weight connecting the i-th (in the input layer) to j-th-node (in the hidden layer), θ_{ij} denotes j-th bias node (in the hidden layer), and

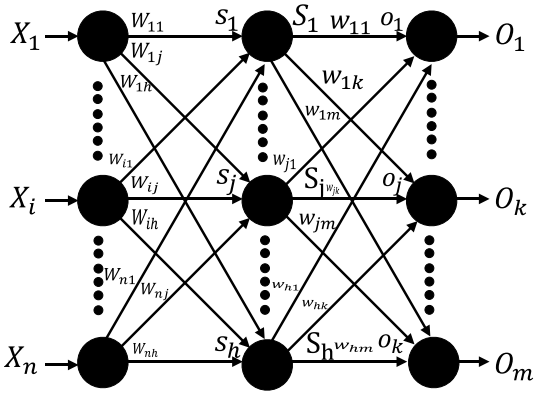

Fig. 1 An MLP NN with a hidden layer

$X\text{-}_i$ represents the input to the i-th node (in the input layer). The output of each hidden node is calculated using a sigmoid function as in Eq. 2.

$$S_j = sigmoid(s_j) = \frac{1}{(1 + \exp(-s_j))}, \quad j = 1, 2 \ldots h \tag{2}$$

$$o_k = \sum_{j=1}^{h} (W_{jk} \cdot S_j) - \theta_k^{'}, \quad k = 1, 2 \ldots m \tag{3}$$

After calculating the values of the hidden nodes, the finals output can be defined as follows:

$$O_k = sigmoid(o_k) = \frac{1}{(1 + exp(-o_k))}, \quad k = 1, 2 \ldots m \tag{4}$$

where W_{jk} is the weight connecting the j-th node (in the hidden layer) to the k-th node (output layer), and θ_k represents the k-th bias node (in the output layer). The most important parts of MLP NNs are weights and the bias of the nodes. As we observed above, weights and the biases, define the final output values. Training a MLP NN includes finding the best value for the weights and the biases, in order to obtain the desired output per specific input.

3 Dragonfly Algorithm

3.1 The Operators for Exploration and Exploitation

According to Reynolds' theory, collective (group) behavior follows three basic principles [21]:

- Separation, which refers to avoiding collision with neighbors.
- Alignment, which represents the speed adjustment relative to the neighbors.
- Cohesion, which points to the tendency of moving toward the average position of the neighbors.

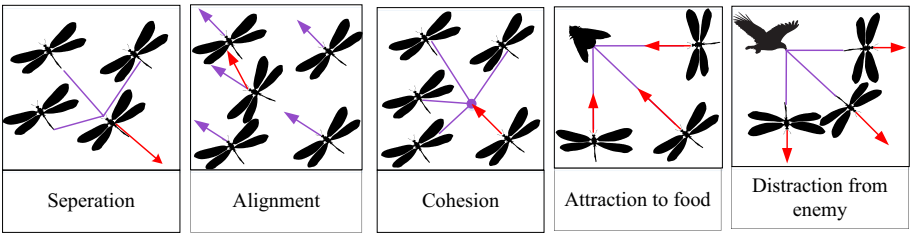

Fig. 2 Five main factors of the members of the updating group

The main purpose of collective life is to survive; therefore, all the members should be attracted to the food sources and divert the attention of the external enemies. According to these two behaviors and following Fig. 2 five main factors of the group members will be updated.

Each of these behaviors is mathematically modeled as follows. Separation is calculated as follows [22].

$$S_i = -\sum_{j=1}^{N} X - X_j \tag{5}$$

where X is the specific position of each member, X_j represents the j-th position of adjacency for each member and N is the number of members' neighbors. The alignment is calculated as follows:

$$A_i = \frac{\sum_{j=1}^{N} V_j}{N} \tag{6}$$

where V_j is the speed of j-th adjacency for each member. Cohesion is calculated as follows:

$$C_i = \frac{\sum_{j=1}^{N} X_j}{N} - X \tag{7}$$

where X is the specific position for each member, N represents the number of neighbors, and X_j denotes the j-th position of adjacency for each. Gravity and attraction toward a food source is calculated as follows:

$$F_i = X^+ - X \tag{8}$$

where X is the specific position of each member and X^+ shows the location of a food source. Distracting the enemy is calculated as follows:

$$E_i = X^- + X \tag{9}$$

where X is the specific position of each member and X^- is the position of the enemy. It is assumed that the behavior of dragonfly is a combination of the five patterns mentioned in this article. We consider two vectors to update the position of the artificial dragonfly in a search space and simulate its movements: step vector (ΔX) and position vector (X).

Step vector is similar to the velocity vector in PSO, and DA has been developed based on the framework of PSO algorithm. The step vector shows the direction toward which the

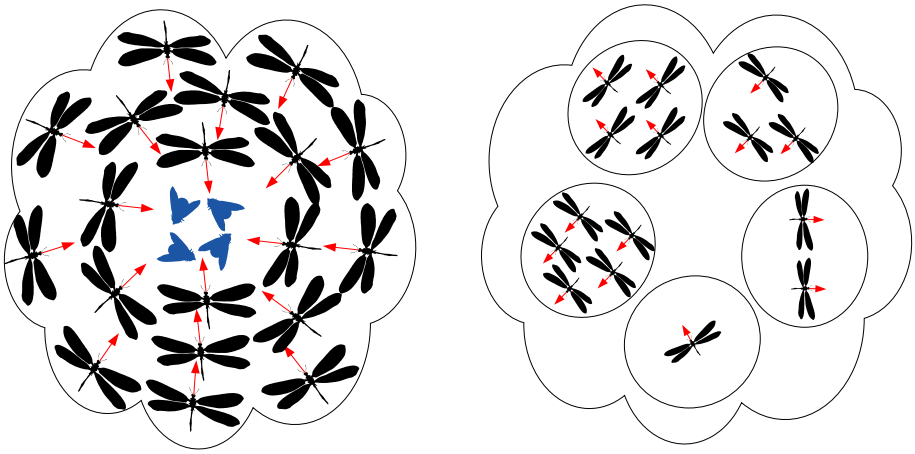

Fig. 3 Dynamic versus static dragonfly swarms

dragonfly moves and it is defined as follows. It should be noted that the position update model for artificial dragonflies has been defined in one dimension; however, it can be developed to higher dimensions.

$$X_{t+1} = \left(sS_i + aA_i + cC_i + fF_i + eE_i\right) + w\Delta X_t \tag{10}$$

where s, a, and c are the separation, alignment, and cohesion weights, respectively while S_i, A_i and C_i, respectively denote the separation, alignment, and cohesion of the i-th member. f is a factor for food and Fi represents the food source for the i-th member. e is a factor for enemy and E_i is the position of the i-th member of the enemy and also w is the inertia weight and t counts the number of repetitions. After calculating the step vector, the position vector is calculated as follows:

$$X_{t+1} = X_t + \Delta X_{t+1} \tag{11}$$

where t is the number of repetitions. By incorporating separation, alignment, cohesion, absorption of food and repulsing the enemy agents (s, a, c, f and e), we can obtain various exploration and exploitation behaviors during the course of optimization. The neighbors of a dragonfly are very important, so that an adjacency is defined as a specific radius around each dragonfly. As Fig. 3 shows, dragonflies show only two types of mass movement: the static and the dynamic movements.

It may be seen in Fig. 3 that dragonflies are willing to adapt their flight, and this occurs when the collective and dynamic separation and adhesion are appropriate. In a group of static dragonflies compliance is low while cohesion, due to the attack on the prey, is high. As a result, while exploring the search space, we assign high compliance and low cohesion to dragonflies. When compliance is low and the cohesion is high, the search space is being exploited.

To transfer between exploration and exploitation, the radius of adjacency increases proportional to the number of iterations. Another way to balance exploration and exploitation, is the compatibility of the adjustments of collective factors (s, a, c, f, w, and e) during the optimization. Here the question is how to ensure convergence of the DA during the optimization. Dragonflies need to change their alignment weight in order to transfer from exploration

Initialize the dragonflies population $X_i(i = 1.2.\ldots.n)$
Initialize step vectors $\Delta X_i(i = 1.2.\ldots.n)$
While the end condition is not satisfied
Calculate the objective values of all dragonflies
Update the food source and enemy
Update w, s, a, c , f and e
Calculate S, A, C, F and E using Eqs.5 to 9
Update neighboring radius
if a dragonfly has at least one neighboring dragonfly
Update velocity vector using Eq.10
Update position vector using Eq.11
else
Update position vector using Eq.12
end if
check and correct the new positions based on the boundaries of variables
end while

Fig. 4 Pseudo-codes of DA

to exploitation phase in the search space. It is assumed that the dragonflies tend to adjust their flight path as an optimization improvement. In other words, the adjacency expands; thereby, in order to allow for the optimization and converge towards the local optimum, a large herd of them become one group.

Food sources and the enemies are among the best and the worst solutions found by swarming and collective movement. It makes the convergence move towards the mentioned search spaces while the divergence to be outside. To reach stochastic improvement and when there is no solution around the adjacency, we use the random behavior and exploration of this dragonfly which should fly around the search space. In this case, the position of dragonfly is updated using the following relations.

$$X_{t+1} = X_t + Le'vy(d) \times X_t \tag{12}$$

where t is the current iteration and d is the distance from the position vector (Fig. 4).

$$Le'vy(x) = 0.01 \times \frac{r_1 \times \sigma}{\left|r_2\right|^{\frac{1}{\beta}}} \tag{13}$$

where r_1, r_2 are two random numbers in the range of [0,1], β is a constant (in this work, equal to 1.5) and σ is calculated as follows:

$$\sigma = \left(\frac{\Gamma(1+\beta) \times \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\left(\frac{\beta-1}{2}\right)}} \right)^{\frac{1}{\beta}} \tag{14}$$

where $\Gamma(x) = (x-1)!$

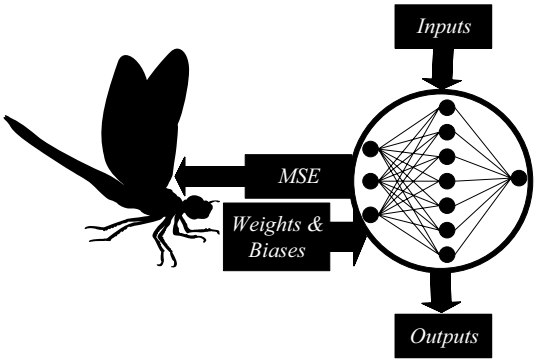

Fig. 5 The training scheme used in the paper

4 Training an MLP NN Using Dragonfly Algorithm

In general, there are three methods of using meta-heuristic algorithms to train MLP NNs. The first method consists of applying meta-heuristic algorithms in order to find a combination of weights and bias nodes to have the least amount of errors in an MLP NN. The second method is the use of meta-noise curves received by hydrophone, Fourier transformheuristic algorithms to find an appropriate structure for an MLP NN in a particular problem, and the last method includes the use of meta-heuristic algorithms to find the gradient descent based methods' parameters like learning rates and momentums. In this paper, DA, is applied to an MLP NN using the first approach. In order to design an algorithm for training an MLP NN, it is necessary to appropriately present the connection weights and the node biases (Fig. 5).

Generally, there are three ways to show the connection weights and the node biases including vectors, matrices and the binary method. In vectors, matrices and binary method, each element is represented respectively by a vector, a matrix, and a string of binary bits. Each of these methods has deficiencies and advantages which could be of use in certain cases.

In the first method, it is easy to convert the elements to a vector, a matrix, or to strings of binary bits; however, the retrieving process will be complicated. That's why this method is often used in simple NNs. In the second method for NNs with complex structures, the decoding process is easier than coding the elements. This method of learning algorithms in generic NNs is very appropriate. In the third method, variables need to be displayed in the binary form. In this case, when the structure of the network becomes complicated, the length of each element also increases. So the process of encoding and decoding would be very complicated.

Since we are not dealing with complex MLP NNs in this work, the vector method is used. In order to reduce the execution time of MLP NN, the general toolboxes of MATLAB are not used. As an example of this style of coding, we have used the final vector of MLP NN shown in Fig. 6 which is presented in Eq. 15.

$$Position = [w_{13}w_{23}w_{14}w_{24}w_{15}w_{25}w_{36}w_{46}w_{56}\theta_1\theta_2\theta_3\theta_4] \tag{15}$$

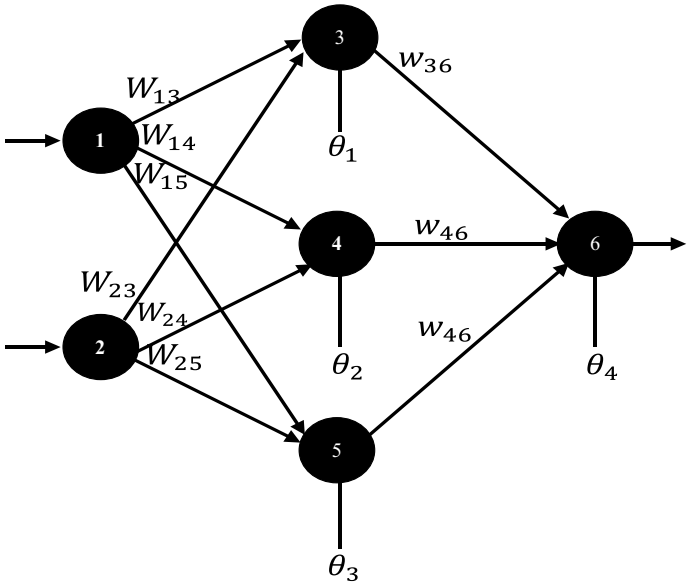

Fig. 6 An MLP NN with structure 2-3-1

5 Dataset

The main strategy of this study is to use three dataset as follows:

- Using Sejnowski & Gorman's [24, 25] dataset in order to have a reference dataset to compare the proposed classifiers with the works of other researchers.
- Planning test scenarios and collecting data in the cavitation tunnel in order to use as inactive sonar data [26].
- Planning test scenarios and collecting data by the designed sonobuoy in order to use as an active sonar dataset.

In the following, the data collection mentioned is briefly described.

5.1 The Sonar Dataset of Sejnowski & Gorman

Sonar data used in this section have been derived from Sejnowski and Gorman's test in [24, 25]. In this experiment, there are two types of echoes; the first is of a metallic cylinder (the real target) and the second, a rock with the same size as the cylinder (clutter or the false target). In this test, a metal cylinder with the length of 5 feet and a rock with the same size have been placed at a sandy seabed and a chirped wide-band linear FM (ka = 55.6) is sent towards them. Based on the received echo's SNR, 208 echoes of 1200 collected echoes which had a SNR in the range of 4-15 db were selected. One hundred and eleven of these 208 echoes belong to the metallic cylinder and 97 to the rock. Figure 7 shows samples of echoes received from the rock and the metal cylinder. As it is observed, echoes of the real

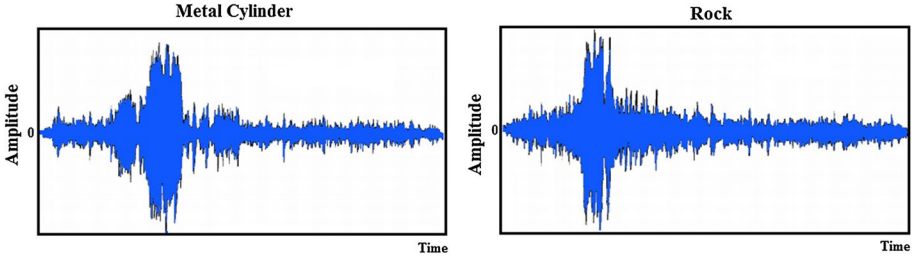

Fig. 7 An example of the echoes returned from the rock and the metal cylinder

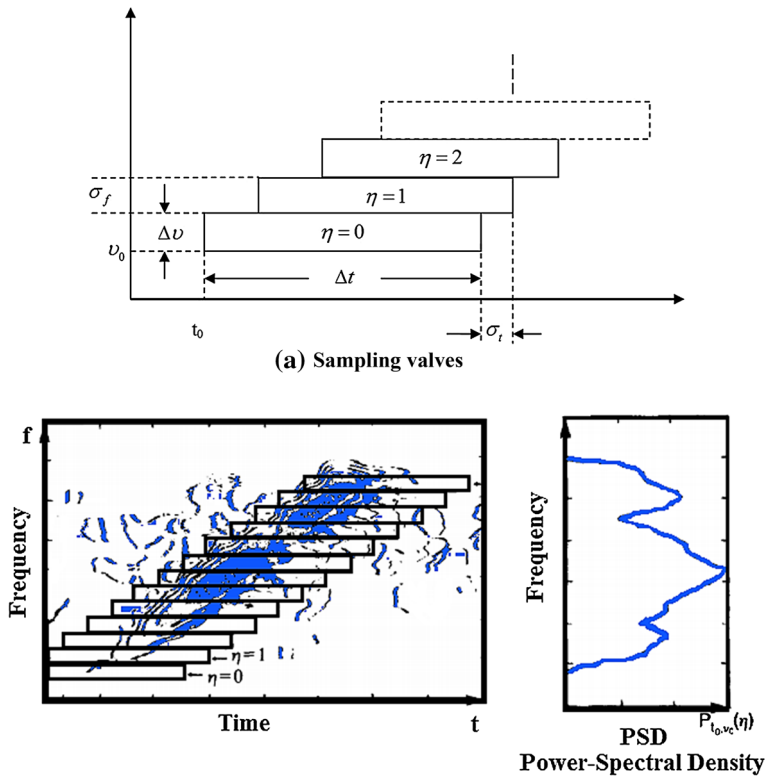

Fig. 8 Preprocessing used to obtain the spectral envelope

target (the cylinder) and the clutter (rock) are very similar to each other and cannot be separated by a linear or non-linear low-order classifier.

Preprocessing used to obtain the spectral envelope is shown in Fig. 8. Figure 8a presents a set of sampling valves. In Fig. 8b, the set of sampling valves have been located on the two-dimensional spectrogram for Fourier transform of Sonar data. The spectral

envelope is obtained by collecting the effects of each valve. In this test, the spectral envelope consists of 60 samples which have been normalized in the range of 0–1. Each of these numbers represents the total energy contained in its associated sampling valve; for instance, the energy contained, after being normalized, would be the first number of the sixty numbers in the characteristics vector.

5.2 The Passive Datasets (Cavitation Tunnel)

To perform this test and obtain reliable datasets, three types of propellers in classes A, B and C were tested in the cavitation tunnel. At this test, which was performed in the cavitation tunnel of the hydrodynamic, three propellers with different speeds to simulate the working conditions of different vessels were examined. This dataset was collected by the authors using the cavitation tunnel model B&K 8103. The complementary information and whole dataset is available in [26].

5.2.1 The Presentation of the Collected Noise

From left to right, respectively, Fig. 9 presents noise curves received by hydrophone, Fourier transform, and noise power spectrum in decibels for the various modeled propellers [25–27].

5.2.2 Active Datasets (Sonobouy)

This dataset has been collected by the Port and Maritime Organization's sonobuoy [29]. In this experiment, 6 objects including 4 targets and 2 non-targets have been placed at the sandy bed of the sea. In this study, the transmitted signal is a wide-band linear frequency modulated pulse which covers the frequency range of 5-110 Hertz. An electric motor rotates the objects (at the bottom of the sea) through 180 degrees, with a precision of one degree. In this position, the returning echoes are collected within 10 m of them.

A proper dataset plays a critical role in the classification of sonar data. Due to the high volume of raw data obtained in the previous step, a high complexity in calculations is anticipated. In order to reduce the complexity of the feature classifier and extractor, conducting a possible target detection process for all the incoming data is necessary. For this purpose, the intensity of the received signal is used.

Due to the shallowness of the sea, phenomena such as multipath propagation, secondary reflections and echoes are inevitable in this environment. After the detection phase and before the feature extraction, we remove the effects of theses artifacts, using a filter in the field of matched filters. After this step, we use inverse filtering to restore the original reflected signal. At this point, we have taken advantage of the fact that separating these artificial processes in the field of matched filter is much simpler than in the time domain. The whole process of pre-processing takes place in four stages as follows:

1. Scaling: To eliminate the effect of amplifier gain, filter and etc. in the data collection stage, it converts the raw signal to a scaled signal.
2. Downsampling: the main sampling rate is 2 MHz which is much higher than the main bandwidth. In order to reduce the sampling rate so that the useful information is not lost, we have used Ref. [26]. In this reference, using environmental data such as water

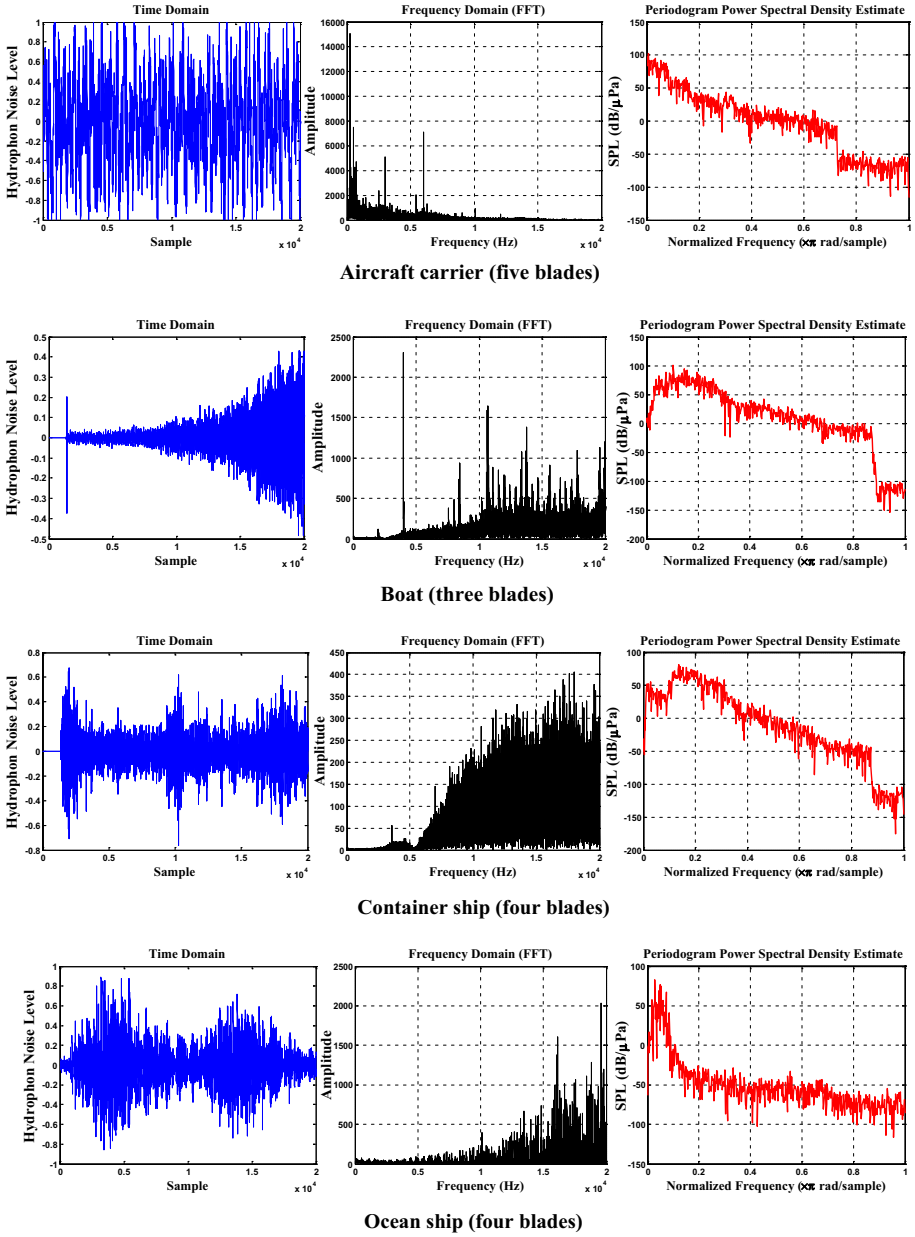

Fig. 9 From left to right, respectively, noise curves received by hydrophone, Fourier transform, and noise power spectrum in decibels versus sound power reference of water(1 μPa) for various modeled floaters

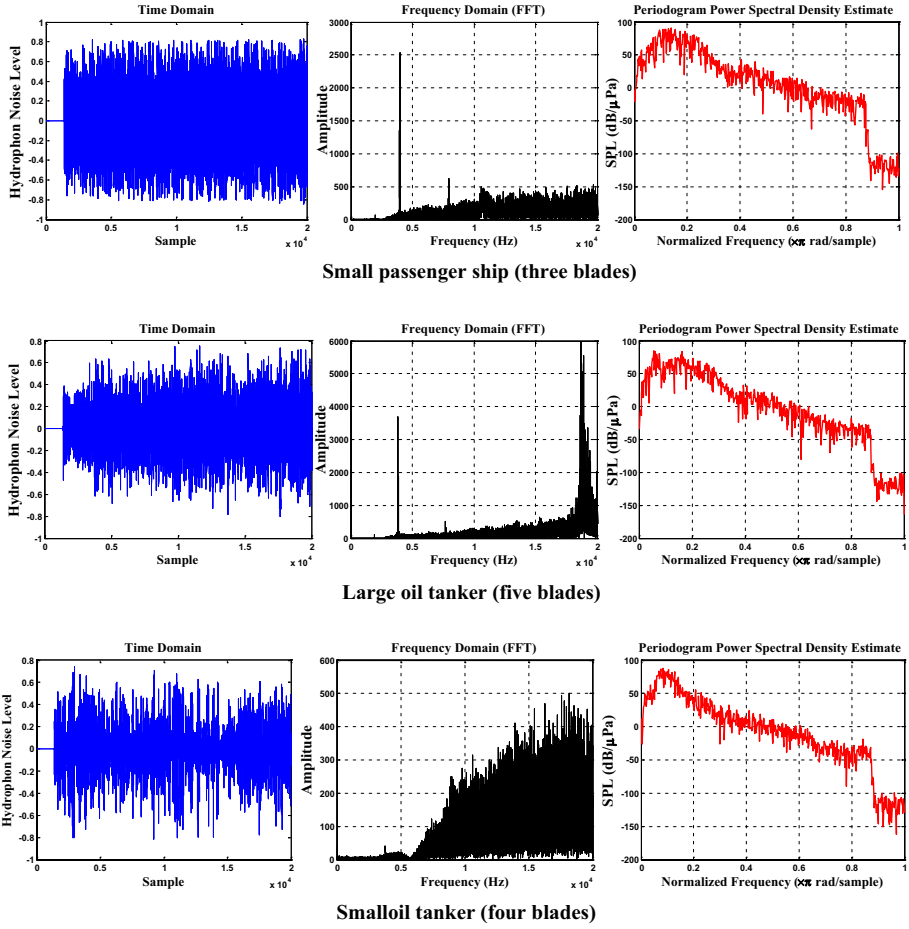

Fig. 9 (continued)

depth, frequency, area under surveillance etc., a fixed number of points are selected for each ping in the sampling stage [30]. Here, 2048 points are selected in a way that the useful information for feature extraction is not missed.

3. The process of removing the artifacts and multipath: In this method, using cross-correlation of backscattering signal with transmitted signal at every angle, the maximum output of matched filter is assigned as x. Then a window covering [x-left: x + right] is applied on the signal. Here, the right and the left are equal to 300 and 211, respectively, resulting in a window with 512 points. In order to maintain the original size of the signal, it is segmented, zero-padded and then inverse-filtered through Eq. 16 to eliminate the effect of transmitted signals.

$$H(k) = \frac{X(k)}{|X(k)|^2 + c} \tag{16}$$

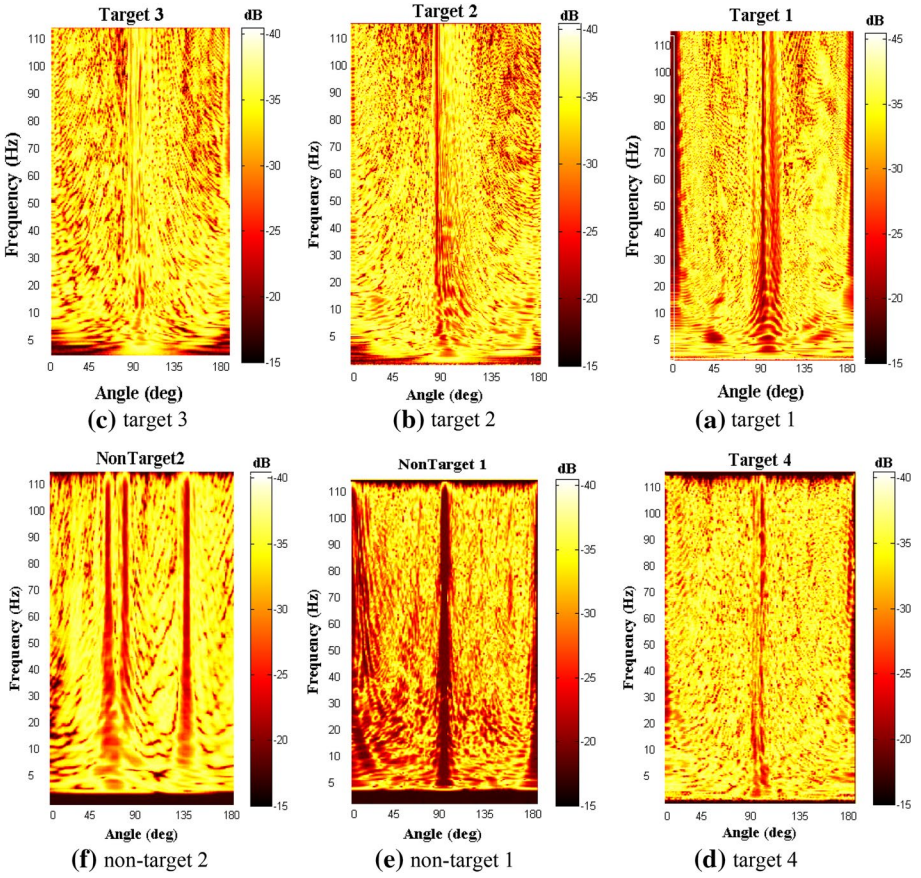

Fig. 10 Examples of echoes returning from different target and non-target objects

where $X(k)$ the Fourier transform of the transmitted signal and $c = 0.0025(|X(k)|^2)$ for solving the singularity problem are added to the Eq. The output of this step is the net backscattered signal without the effects of artificial phenomena.

4. Normalization: finally, we scale each objective so that they all have the same target strength. To do this, we divide each backscattered signal using SAR with the highest range which is less than 90% of the maximum received ranges.

Figure 10 shows examples of the signals received from different objectives which are functions of their frequency and direction.

5.3 Feature extraction

After the preprocessing and receiving detected frames which contain the sounds of backscattered signals, the detected sounds whose effects of artificial phenomena has been eliminated and transferred to the frequency domain (named $S(k)$) are delivered to the feature section (Figs. 11, 12). At this step, first the spectral of the signal is calculated by Eq. 17.

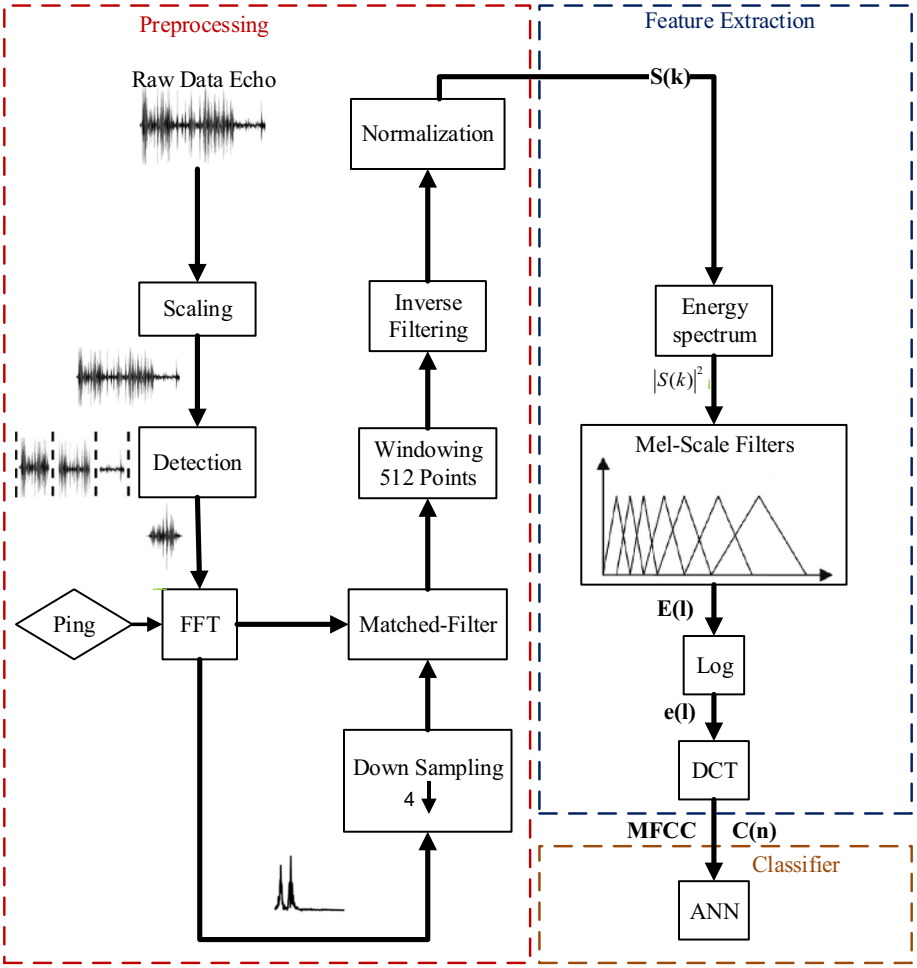

Fig. 11 Block diagram of preprocessing and feature extraction steps

$$|S(k)|^2 = S_r^2(k) + S_i^2(k) \tag{17}$$

where S_r^2(k) and S_i^2(k) are the real and the imaginary parts of the Fourier transform for the detected signal. Then the spectral energy is filtered by a Mel-scaled triangular filter. The output energy of the lth filter is calculated by Eq. 18.

$$E(l) = \sum_{k=0}^{N-1} |S(k)|^2 H_l(k) \tag{18}$$

where N is the number of discrete frequencies used for FFT in the preprocessing stage and $H_l(k)$ is the transfer function of the given filter where l = 0, 1,…, Ml.

The dynamic range of Mel-filtered energy spectrum is compacted by the logarithmic function as Eq. 19.

$$E(l) = log(E(l)) \tag{19}$$

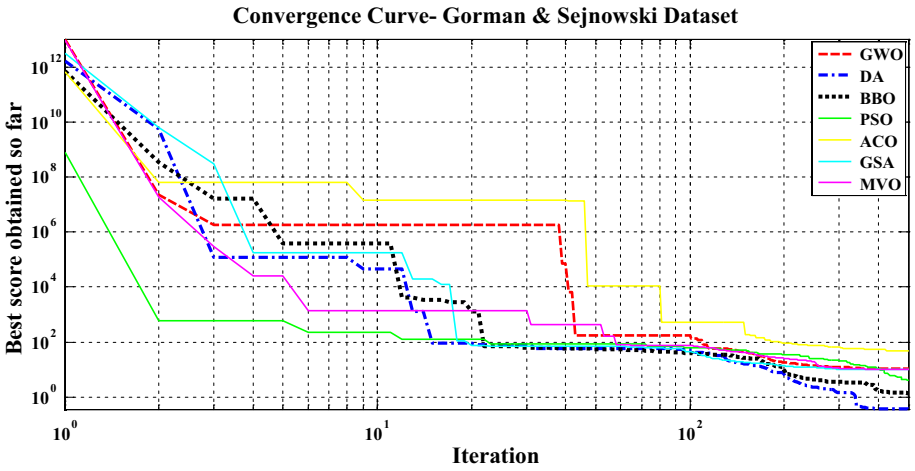

Fig. 12 Convergence curves for Gorman & Sejnowshki dataset

Finally Mel-frequency cepstral coefficients are converted back to the time domain using Eq. 20 and discrete cosine transform (DCT).

$$C(n) = \sum_{l=1}^{M} e(l)\cos\left(n\left(l - \frac{1}{2}\right)\frac{\pi}{M}\right) \tag{20}$$

In this case and for all detected objectives, the characteristic vector will be in the form of Eq. 21.

$$X_m = [c(0)\cdot c(1) \ldots c(P-1)]^T \tag{21}$$

All the steps outlined in the preprocessing and the feature extraction have been presented as a block diagram in Fig. 11.

6 Simulation results

In this section DA algorithm is measured by three different datasets. The datial of datasets are shown as Table 1. In each test, the efficiency of DA algorithm in terms of classification accuracy, convergence speed, and the probability of getting stuck in a local minimum

Table 1 The mean classification rate of various algorithms

Dataset	DA	GWO	BBO	MVO	GSA	PSO	ACO
Gorman and Sejnowski	92.1457	90.1111	92.0011	89.8896	86.2314	91.2213	85.9456
Passive	94.2154	89.2541	93.2145	88.2534	90.2145	90.0014	86.2541
Active	96.6523	94.4258	94.0124	93.9584	91.2584	88.2548	89.254

Table 2 Parameters and initial values of the applied algorithms

Algorithm	Parameter	Value
BBO	The probability of correcting the habitants	1
	The probability range for migrating into for each gene	[0, 1]
	Step size for the probability numerical integral	1
	Maximum migration into (I) and migrating out of (E) coefficient	1
	Mutation probability	005/0
	Population size	200
PSO	Layout	Full connection
	Cognitive constant ($C1$)	1
	Social constant ($C2$)	1
	Local constant (W)	0.3
	Population size	200
ACO	Primary pheromone ($\tau 0$)	0.000001
	Pheromone updating constant (Q)	20
	Pheromone constant ($q0$)	1
	Decreasing rate of the overall pheromone (Pg)	0.9
	Decreasing rate of local pheromone (Pt)	0.5
	Pheromone sensitivity (a)	1
	Observable sensitivity (β)	5
	Population size	208
GWO	The number of gray wolf	12
	Limit down	30
	Limit up	-30
GSA	Coefficient (α)	20
	Gravitational constant ($G_\circ$)	1
	The initial speed of the masses	[0, 1]
	The initial value of the acceleration	0
	Initial value of mass	0
	Population size	208
DA	Velocity (V)	5 m/s
	Initial velocity ($V_\circ$)	3 m/s
	Total weight	10^{-3} kg
	Wing area	10^{-4} m^2
	Population size	208
MVO	w_{max}	1
	w_{min}	0.2
	Precision operation (P)	6

are compared with other meta-heuristic algorithms. Comparison algorithms include: BBO, PSO, ACO, GWO, GSA, MVO and DA.

Required parameters and initial values have been presented in Table 2. Each network has been tested 10 times. After 10 runs, the best trained NN is selected and used for comparison. Classification rate and error percentage are two measures to compare the algorithms. To have a fairly good comparison, all algorithms stop after reaching a maximum of

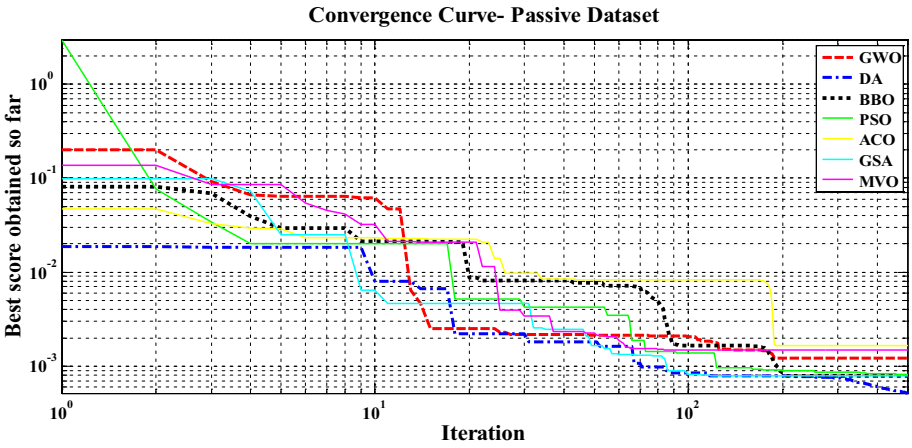

Fig. 13 Convergence curves for passive dataset

250 iterations. Finally, to have a comprehensive comparison, the convergence of the results is examined. Since there are no standards regarding the selection of hidden nodes in the classification of datasets, we use the proposition in [30–33] and Eq. 22 based on the structure of MLP NNs.

$$H = 2N + 1 \tag{22}$$

where N is the number of inputs and H represents the number of hidden nodes. It should be noted that the simulations were performed in MATLAB using a PC with a 2.3 GHz processor and 4 GB RAM. The simulation results have been presented in Figs. 12, 13 and 14.

As shown in Fig. 6, DA algorithm, compared to the classical algorithms, has much better results in all cases in terms of classification accuracy and convergence speed. The poor results of the ACO, PSO, and GWO algorithms are due to the nature of these algorithms.

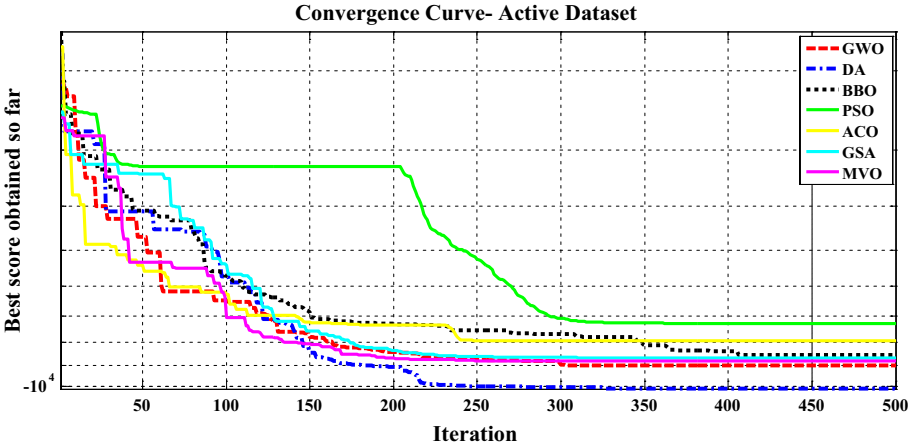

Fig. 14 Convergence curves for active dataset

These algorithms have no operators for dramatic changes; therefore, they get stuck in local minimums. Moreover, PSO algorithm uses pheromone matrix increasing the learning capacity and efficiency of the algorithm, which is an advantage in combination problems, but also increasing the probability of getting stuck in a local minimum. PSO algorithms depend too much on the initial distribution of the particles and their primary drivers based on the attraction between them. If a large number of particles are caught in the local minimums, the algorithm to a low extent prevents the trapping of other particles.

The exploration and exploitation in this problem provide a heuristic behavior and a different recognition power. Shortly, we can say that the exploration power carries much weight in MLP NNs problems. Therefore, in order to avoid getting stuck in local minimums when solving complex problems using MLP NNs, we need random and instant search algorithms.

7 Conclusion

In this article, the sonar targets were categorized using MLP NNs. DA, BBO, GWO, ALO, ACO, GSA and MVO algorithms were used to train the MLP classifier. As it was shown, DA has a good ability in general optimal search. Due to the unique ability of DA in the exploration and exploitation phases, it has been used for solving the problems mentioned above which are more apparent in the case of data with high dimensions such as sonar data. The results show that the classifier based on DA has the most optimum classification accuracy for three different values of 92.14, 94.21 and 96.65 respectively. The speed of convergence of the algorithm is also higher than the six algorithms used.

References

1. Mosavi, M. R., Khishe, M., & Ebrahimi, E. (2015). Classification of sonar targets using OMKC, genetic algorithms and statistical moments. *Journal of Advances in Computer Research, 7*(1), 50–59.
2. Mosavi, M. R., Khishe, M., Aghababaei, M., & Mohammadzadeh, F. (2015). Approximation of active sonar Clutter's statistical parameters using array's effective beam-width. *Iranian Journal of Marine Science and Technology, 73*(1), 11–22.
3. Simon, D. (2008). Biogeography-based optimization. *IEEE Ttransactions on Evolutionary on Computation, 12*(6), 702–713.
4. Auer, P., Burgsteiner, H., & Maass, W. (2008). A learning rule for very simple universal approximators consisting of a single layer of perceptrons. *Neural Networks, 21*(5), 786–795.
5. Mirjalili, S. M., & Mirjalili, S. (2014). Oval-shaped-hole photonic crystal waveguide design by MoMIR framework. *Photonics Technol Letters Issue, 99,* 1.
6. Mirjalili, S. M., Mirjalili, S., & Lewis, A. (2014). A novel multi-objective optimization framework for designing photonic crystal waveguides. *Photonics Technol Letters, 26,* 146–149.
7. Ng, S. C., Cheung, C. C., Leung, S. H., & Luk, A. (2003). Fast convergence for backpropagation network with magnified gradient function. *IEEE Joint Conference on Neural Networks, 3,* 1903–1908.
8. Magoulas, G., Vrahatis, M., & Androulakis, G. (1997). On the alleviation of the problem of local minima in backpropagation. *Nonlinear Analysis, Theory, Methods & Applications, 30*(7), 4545–4550.
9. Ho, Y. C., & Pepyne, D. L. (2002). Simple explanation of the no-free-lunch theorem and its implications. *Journal of Optimization Theory and Applications, 115*(3), 549–570.
10. Wang, P., Yu, X., & Lu, J. (2014). Identification and evolution of structurally dominant nodes in protein-protein interaction networks. *IEEE Transactions on Biomedical Circuits and Systems, 8*(1), 87–97.

11. Gudise, V. G., & Venayagamoorthy, G. K. (2003) Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. In *IEEE Swarm intelligence symposium* (pp. 110–117).
12. Culberson, J. L. (1998). On the futility of blind search: An algorithmists view of the no free lunch. *Evolutionary Computation, 6,* 109–127.
13. Ho, Y. C., & Pepyne, D. L. (2002). Simple explanation of the no-free-lunch theorem of optimization. *Cybernetics and Systems Analysis, 38*(2), 292–298.
14. Montana, D. J., & Davis, L. (1989). Training feed-forward neural networks using genetic algorithms. *In 11th International Joint Conference on Artificial Intelligence, 1,* 762–767.
15. Shaw, S., & Kinsner, W. (1996). Chaotic simulated annealing in multilayer feedforward networks. *Canadian Conference on Electrical and Computer Engineering, 1,* 265–269.
16. Chang, S. K., Mohammed, O. A., & Hahn, S. Y. (1994). Detection of magnetic body using article neural network with modified simulated annealing. *IEEE Transactions on Magnetics, 30*(5), 3644–3647.
17. Mirjalili, S., & Sadiq, A. S. (2011) Magnetic Optimization Algorithm for Training Multilayer Perceptron. In: IEEE 3rd International Conference on Communication Software and Networks (ICCSN) (pp. 42–46).
18. Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Let a biogeography-based optimizer train your multilayer perceptron. *Journal of Information Science, 269,* 188–209.
19. Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software, 69,* 46–61.
20. Auer, P., Burgsteiner, H., & Maass, W. (2008). A learning rule for very simple universal approximators consisting of a single layer of perceptrons. *Neural Networks, 21*(5), 786–795.
21. Wikelski, M., Moskowitz, D., Adelman, J., & Cochran, J. (2006). Simple rules guide dragonfly migration. *Biology Letters, 2,* 325–329.
22. Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *ACM Siggraph Computer Graphics, 21,* 25–34.
23. Yang, X. S. (2010). *Nature-inspired metaheuristic algorithms* (2nd ed.). Bristol: Luniver Press.
24. Gorman, R. P., & Sejnowski, T. J. (1988). Analysis of hidden units in a layered network trained to classify sonar targets. *Neural Networks, 1,* 75–89.
25. http://archive.ics.uci.edu/ml/datasets/Connectionist+Bench+(Sonar,+Mines+vs.+Rocks).
26. Khishe, M., Mosavi, M., & Safarpoor, B. (2017). *Sound of propeller*. Mendeley Data, v1, 2017. http://dx.doi.org/10.17632/rryjwssc53.1.
27. Gaggero, S., Savio, L., Brizzolara, S., Viviani, M., Ferrando, M., & Conti, F. (2009). *An experimental study on measuring and localzating propeller noise behind a body in a cavitations tunnel*. Trondheim, Norway: First International Symposium on Marine Propulsors.
28. Carlton, J. (2012). *Marine propeller and propulsion* (3rd ed.). Oxford: Butterworth-Heinemann.
29. Naseri, M. J. (2015). *Floating buoy controller design and implementation by using special sonobuoys*. M. S. Thesis, Marine Sciences University of Nowshahr Imam Khomeini.
30. Preston, M. (2004). *Resampling sonar echo time series primarily for seabed sediment*. United State Patent, US 6,801,474 B2, October 2004.
31. Branke, J. (1995). *Evolutionary Algorithms for Neural Network Design and Training*. Citeseer.
32. Mendes, R., Cortez, P., Rocha, M., & Neves, J. (2002). Particle Swarms for Feedforward Neural Network Training. In *IEEE joint conference on neural networks* (pp. 1895–1899).
33. Mirjalili, S. (2011). *Hybrid Particle Swarm Optimization and Gravitational Search Algorithm for Multilayer Perceptron Learning*. Master: Universiti Teknologi Malaysia (UTM).

Mohammad Khishe was born on 1985 in Nahavand, Iran. He received the B.Sc. degree in Maritime Electronic and Communication Engineering from Imam Khomeini Maritime Sciences University, Nowshahr, Iran in 2007 and also, M.Sc. and Ph.D. degrees in Electronic Engineering from Islamic Azad University, Qazvin branch and Iran University of Science and Technology, Tehran in 2012 and 2017, respectively. Since 2011, he is a faculty member of Maritime Electrical and Electronic Engineering at Imam Khomeini Maritime Sciences University, Nowshahr, Iran. His research interests are Digital Signal Processing, Artificial Neural Networks, Meta-heuristic Algorithms, Sonar and Radar Signal Processing, and FPGA design.

Abbas Safari was born in Touyserkan, Iran on 1992. He received his B.Sc. degree in 2013 and M.Sc.degree in 2016, from Imam Khomeini Marine University, Department of Electrical Engineering. His research interests are in the areas of Neural Networks, Meta-Heuristi Algorithm, video and image processing, Sonar, Classification.