Training Radial Basis Function Neural Network using Stochastic Fractal Search Algorithm to Classify Sonar Dataset

M. R. Mosavi*(C.A.), M. Khishe*, Y. Hatam Khani** and M. Shabani***

Abstract: Radial Basis Function Neural Networks (RBF NNs) are one of the most applicable NNs in the classification of real targets. Despite the use of recursive methods and gradient descent for training RBF NNs, classification improper accuracy, failing to local minimum and low-convergence speed are defects of this type of network. In order to overcome these defects, heuristic and meta-heuristic algorithms have been conventional to training RBF network in the recent years. This study uses Stochastic Fractal Search Algorithm (SFSA) for training RBF NNs. The particles in the new algorithm explore the search space more efficiently by using the diffusion property, which is observed regularly in arbitrary fractals. To assess the performance of the proposed classifier, this network will be evaluated with the two benchmark datasets and a high-dimensional practical dataset (i.e., sonar). Results indicate that new classifier classifies sonar dataset six percent better than the best algorithm and its convergence speed is better than the other algorithms. Also has better performance than classic benchmark algorithms about all datasets.

Keywords: Classifier, RBF, Stochastic Fractal, Meta-heuristic Algorithm.

1. Introduction

Radial Basis Function Neural Networks (RBF NNs) are one of the most applicable tools for soft computing. By using these networks, non-linear problems can be solved. In general, RBF NNs are used to pattern classification, time series prediction, and functions approximation [1-4]. Regardless of their applications, the distinct ability of NNs is learning [3]. Learning means that these networks can learn like the human's brain from experience or experiments. This feature (learning) is an essential part of all NNs that can be divided into two types: supervised learning [5] and unsupervised learning [6]. Generally, NNs' training is a challenging point for the optimization of non-linear problems. Many methods, including gradient descent [6], Kalman Filter (KF) [7], Decoupling Kalman Filter (DKF) [8] and Back Propagation (BP) [9] which are based on derivation have been used to teach RBF NNs. In addition to derivative-based methods, derivation without methods, such as Genetic Algorithm (GA) [10], Automata Learning (AL) [11] and Simulated Annealing (SA) [12], Convergent Decomposition Techniques (CDT) [13], Population Migration Algorithm (PMA) [14], Artificial Bee Colony (ABC) [15], Efficient Two-phase Algorithm (ETA) [16], Increment Design and Training (IDT) [17], Decay Method (DM) [18] and Unsupervised Feature Learning (UFL) [19] have been used in training RBF NNs.

The ultimate purpose of the learning process in NNs is distinguishing the best combination of network's parameters so that the training and testing samples produce the least amount of error. In this paper, our tool to reach this goal is Stochastic Fractal Search Algorithm (SFSA). Approximation algorithms have been divided into two categories: specific heuristic and meta-heuristics. Specific heuristics are designed for particular problems while meta-heuristics are applicable for a large variety of optimization problems. The competence of meta-heuristic algorithms depends on the fact that they imitate the best features of nature, especially the selection of the fittest biological system that is evolved over millions of years [20]. The main advantage of SFSA algorithms, which is one of

Iranian Journal of Electrical & Electronic Engineering, 2017.
Paper received 18 June 2016 and accepted 17 March 2017.

* The authors are with the Department of Electrical Engineering, Iran University of Science and Technology, Narmak, Tehran 13114-16846, Iran.
** The author is with the Department of Electrical Engineering, University of Imam Khomeini Marine Sciences, Noshahr, Iran.
*** The author is with the Department of Physics, University of Hormozgan, Bandar Abbas, Iran.
E-mails: m_mosavi@iust.ac.ir, m_khishe@elec.iust.ac.ir, yousofhatamkhani@yahoo.com and shabani.stu@Hormozgan.ac.ir.
Corresponding Author: M. R. Mosavi.

the most powerful meta-heuristic algorithms, is related to explore the space efficiently without sensitivity to the size of the search space. Basically, SFSA is designed based on three main purposes: solving problems faster than other methods, solving large problems, and obtaining a robust method for solving problems [21]. Furthermore, ease of design and hardware implementation along with flexibility is the other features of this algorithm.

In this paper, RBF NN has been trained by SFSA algorithm [22] to classify sonar data (including real target and false target). Our motivations for using RBF NN and SFSA learning algorithm are as follows:

- By using basis functions, RBF NN has the unique ability to work with datasets that are linearly inseparable. On the other hand, sonar dataset is linearly inseparable, so it needs for classifier with high-dimensional.
- Characteristics of clutter (false target) and real targets are very similar; therefore, an algorithm must be chosen that discovers the searching space, completely.
- The outstanding feature of the SFSA algorithm in comparison with other meta-heuristic methods is discovering of whole problem space so that it has the extra ability to avoid trapping into local optima than other meta-heuristic algorithms.
- SFSA is very low computational cost; therefore, it is very efficient for hardware implementation.
- On the one hand real-time processing is a critical requirement in the field of sonar dataset classification, and on the other hand, RBF NN and SFSA have parallel structure, which helps us to use the ability of the hardware with parallel structure (FPGA), for real-time processing.

Organization of the paper is as follows: Section 2 explains RBF NNs. Section 3 describes RBF NN training algorithm (i.e., SFSA). In section 4, RBF NN is trained by SFSA algorithm. The simulation results are described in section 5 and conclusions is presented in section 6, finally.

2. Radial Basis Function Neural Network

RBF NN is a kind of feed-forward NNs composed of three layers including an input layer, hidden layer, and output layer that have entirely different roles [23]. The block diagram of identification system based on RBF NN is shown in Fig.1. In a RBF NN, outputs of the input layer are specified by calculating the distance between inputs of the network and centers of the hidden layer. The second layer is a linear hidden layer. The outputs of this layer are weighted samples of the outputs of the input layer. Each neuron of hidden layer has a parameter vector so called center. Therefore, a general description of the network is given by Eq. (1) [24]:

$$\hat{y}_j = \sum_{i=1}^{I} w_{ij}\, \phi(\|x - c_i\|) + \beta_j \tag{1}$$

Standard distance is usually the Euclidean distance and RBF function is considered as Gaussian function as following:

$$\phi\ (\mathrm{r}) = \exp(-\alpha_i \parallel \mathrm{x} - \mathrm{c_i} \parallel^2) \tag{2}$$

In Eqs. (1) and (2), following definitions are considered: i∈{1, 2, 3, ..., I} so that I indicates the number of neurons in the hidden layer; w_{ij} is the connection weight from ith neuron in the hidden layer to jth neuron in the output layer; ϕ is Gaussian function; α_i shows spread parameter (amount of variance) for ith neuron; *x* indicates input data vector; c_i is the center vector of the ith neuron; β_j is the bias of jth neuron in the output layer and y^ indicates the output of the network.

Fig. 2 shows the precise architecture of used RBF network in this paper.

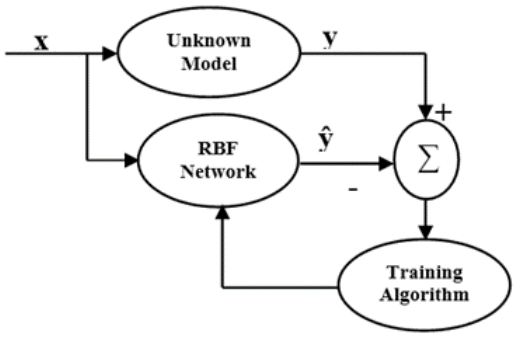


Fig.1. The block diagram of identification system based on RBF NN.

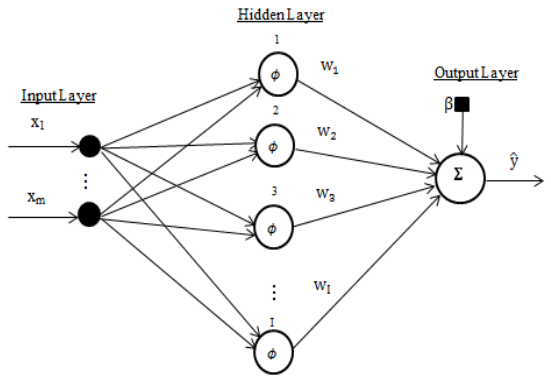


Fig. 2. Architecture of used RBF network in this paper.

The design method of RBF NN should include determining the number of neurons in the hidden layer. In order to obtain the desired output of RBF NN, parameters W, α, C and β must be set, correctly. Root Mean Square Error (RMSE) can be used to evaluate network performance and define as follows:

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{I}(y-\hat{y})^2}{I}} \tag{3}$$

where, y is the desired output and y^ shows the output of RBF NN. Minimizing the error function is the aim of RBF NN training method.

3. Stochastic Fractal Search Algorithm

In this section, we explain SFSA algorithm that is exerted for RBF NN training.

3. 1. Fractal

A property of a quantity that remains steady on all scales in technical concept is called fractal. Mandelbrot used the concept of fractal to describe geometric designs in nature [25]. Far-from-equilibrium growth phenomenon is an important field including fractals that is linked to various fields of science and technology. Some instances of such processes are consisting of dendritic solidification in a very cold ambient, viscous fingering when a viscous fluid is injected into a more viscous fluid, and electrode-positing of ions into an electrode [26]. We simulate some samples of fractal phenomena in Fig.3.

3.2. Fractal Search Algorithm

FSA has applied both fractal growth and potential theory. FSA uses three simple rules to discover a solution:

1. Each particle has a potential energy.
2. Each particle diffuses, and causes to generate other particles randomly, and the energy of the original particle is divided between other particles.
3. Only a few of the best particles subsist in each generation and the rest of the particles are dissembled.

Consider P (where $1 \leq P \leq 20$) particles are taken care of finding the solution for a problem. Initially, each particle P_i has been located stochastically in the search space with equal energy E_i acquired from Eq. (4):

$$E_i = \frac{E}{P} \tag{4}$$

where, E shows the maximum supposed electric potential energy. To run fractal optimizer, each particle diffuses in each generation and produces others based on Levy flight. Levy flight is a special type of Brownian distribution that includes flying particles from one region into other region of space in multi-steps so that they have overlap with each other. Also, Levy flight can be applied to the spread of an epidemic model [27]. Eq. (5) indicates the Levy distribution.

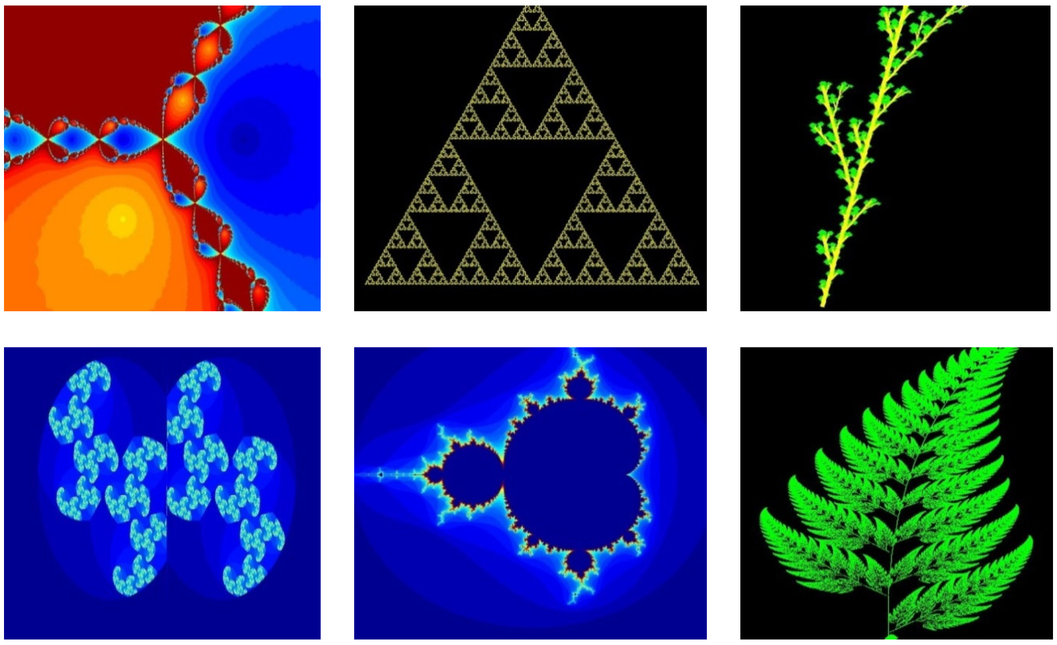


Fig.3. Examples of the fractal phenomenon.

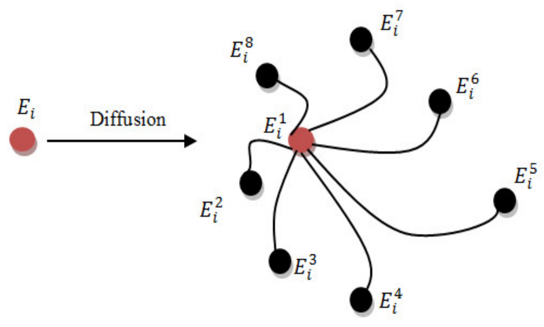


Fig. 4. Particle diffusion.

$$L(x) = \frac{1}{\pi}\int_0^{\infty} e^{(-\alpha q^{\beta})}\cos(qx)dx \tag{5}$$

where, β is the distribution index, which is confined to 0 ≤ β≤ 2, and α is the scale factor. As shown in Fig.4, particle diffusion causes the generation of new particles with stochastic locations around it.

To create each particle in the diffusion process, both Levy and Gaussian distributions are used based on Eqs. (6) and (7), respectively:

$$\chi_i^q = \chi_i + \alpha_i^q \otimes levy(\lambda) \tag{6}$$

where q is an index (1 ≤ q ≤ Maximum diffusion number).

$$\chi_i^q = \chi_i + \beta \times Gaussian(P_{i,}|BP|) - (\gamma \times BP - \gamma' \times P_i) \tag{7}$$

where, q is the number of particles generated by the diffusion of the main particle. The product ⊗ means entry-wise multiplications, β in Eq. (7) is equal to (log(g))/g, where g is assumed the number of iterations. Gaussian (P_i, |BP|) is the Gaussian distribution so that P_i and BP are mean and standard deviations, respectively. BP is hinted as the location of the best particle. Moreover, γ and γ′ are casual parameters between 0 and 1. To use better of both Levy and Gaussian walks, FSA substitute between them, randomly. Thus Levy distribution is often used to guaranty fast convergence algorithms while Gaussian walk is applied to extract a better estimation of the final result. Since the search method depends entirely on random walks, a fast convergence cannot be assurance. Therefore, the α parameter plays an important role for fast convergence. Two equations are supposed for α, one of them is applied to search in a wider space, and the other is used to gain a solution with higher accuracy. Two equations used for α are as follow:

$$\alpha_i = \frac{log(min(\hat{E}))\times(U-D)}{g\times log(E_i)} \tag{8}$$

$$\alpha_i = \frac{(U-D)}{(g\times \log(E_i))^{\varepsilon}} \tag{9}$$

where, min (Ê) is considered as the minimum energy of a particle. U and D are the upper and lower bound of the search space. g indicates the generation number. E_i Shows the particle energy of P_i, and ε is generally fixed by 3/2 and is considered as the power. After diffusing the particle, the main problem is how to distribute the original particle energy between generated particles. A simple theory to distribute energy, states the the created fitness particle has the greater the energy. Suppose q is the number of particles generated by diffusing the particle P_i with energy E_i. Every diffused particle has a fitness value of f_j, where j=1, 2… q. The equation of the distribution energy can be described as follows:

$$E_i^j = \left[\left(\frac{f_j}{f_i+\sum_{k=1}^{q} f_k}\right)\right] \times E_i \tag{10}$$

where, f_i is the fitness value of the original particle before diffusion. Although this model is performed well in both local and universal searching, but complexity of searching increase based on the generation of new particles In effect diffusion. To solve this problem, just a few of the best particles are passed into the oncoming generation (lower than 15% of the particles in each generation). The acquired energy from the banishing particles is applied for the remained particles and the generation of new particles.

Suppose ϕ is the sum of all acquired energy from the banishing particles. μ is the distribution rate of energy between remained particles and newly generated particles. Eq. (11) shows the distribution energy equation for the remained particles:

$$E_{new}^t = E_{old}^t + [\left[\left(\frac{f_t}{\sum_{k=1}^{\xi} f_k}\right)\right] \times \phi] \times \mu \tag{11}$$

where E^t_{new} and E^t_{old} are the particle's energy after and before distribution. On the other hand, ξ is the number of all particles in the iteration. For any particle that diffuse the numbers of generated particles and located in the search space, is calculated by following equation [22]:

$$\vartheta = \frac{\log(\text{numberof discarded particles})}{\log(\text{maximum diffusion})} \tag{12}$$

The distributed energy for generating each particle is according to Eq. (13):

$$E'_c = \frac{\phi\times(1-\mu)}{\vartheta} \quad ; \quad c = 1, 2 \dots \vartheta \tag{13}$$

3. 3. Stochastic Fractal for FSA

Although FSA applies well in finding the solution, but this method has a weak performance in solving high-dimensional data such as sonar. Therefore, we use another version of Fractal Search so-called Stochastic Fractal Search Algorithm (SFSA).

Two main processes that happened in the SFSA algorithms are the diffusion and the updating processes. In the first process similar fractal search, each particle diffuses to obtain exploitation property and this process increases the probability of finding the global minima, and prevents being trapped in the local minima. In the second process, the algorithm displays how a particle in the group updates its situation based on the position of other particles. Unlike the diffusing stage in FSA which causes a dramatic growth in the number of participating particles, we assess a static diffusion process for SFSA. It means that the best-created particle in the diffusing stage is selected, and the rest of the particles are ignored. Also, SFSA uses some random methods as updating processes for exploration properties in meta-heuristic algorithms to efficient detection of the problem space.

To generate new particles from the diffusion process, two statistical models called Levy flight and Gaussian are considered. Primary studies over advantage of Levy and Gaussian distributions individually represent, that although Levy flight converges faster than Gaussian walk in a few generations. The Gaussian walk is more answerable than Levy flight in finding global minima. Therefore, unlike FSA, which uses the Levy flight distribution, SFSA uses Gaussian distribution, which is the random walk applied in the Diffusion Limited Aggregation (DLA) growth process. A series of Gaussian walks taking part in the diffusion process have been mentioned in Eqs. (14) and (15) [27]:

$$GW_1 = \text{Gaussian}\ (\mu_{BP}, \sigma) + (\mathcal{E} \times BP - \mathcal{E}' \times P_i) \tag{14}$$

$$GW_2 = \text{Gaussian}\ (\mu_P, \sigma) \tag{15}$$

where, $\mathcal{E}$ and $\mathcal{E}'$ are uniformly distributed random numbers limited to [0, 1]. BP and P_i are mentioned as the position of the best particle and the ith particle in the group, respectively. The two primary parameters of the Gaussian distribution are μ_{BP} and σ where μ_{BP} is really equal to BP. Parameters μ_P and σ, where μ_P is equal to P_i. According to the Gaussian parameters, the standard deviation is calculated by Eq. (16):

$$\sigma = \left| \frac{\log(g)}{g} \times (P_i - BP\ ) \right| \tag{16}$$

To get closer to the solution, the term (log(g))/g is applied to decrease the step of Gaussian mutations, as the number of generation increases. Suppose, there is a global optimization problem with D dimension; therefore, each individual assumption to solve the problem has been constructed based on a D-dimensional vector. During the initialization procedure, each point is randomly initialized based on matter constraints by defining maximum and minimum limits. The initialization equation of the jth point P_j is presented as follows:

$$P_j = LB + \mathcal{E} \times (UB - LB) \tag{17}$$

where, LB and UB are the lower and the upper boundary vectors in problem, respectively. $\mathcal{E}$ is a uniformly distributed random number which is limited to [0, 1]. After initializing all points, the fitness function of each point is calculated to find the Best Point (BP) in between all points. According to the exploitation property in the diffusion process, all points have been considered about their running position to apply problem search space and also two statistical methods have been consider to increase the exploration of space. The first statistical method executes on each vector index individually while the second statistical procedure is practical to all points. In the first statistical method, initially, all the points are graded based on the value of the cost function. Then, a probability value is given for each point i in the group, which follows a simple uniform distribution according to Eq. (18):

$$Pa_i = \frac{rank(P_i)}{N} \tag{18}$$

where rank(P_i) is considered as the rank of point P_i among the other points in the group, and N is applied as the number of all points in the group. In fact, Eq. (18) clarifies the better point that has the higher the probability. This equation is used to increase the opportunity of changing the position of points. Also, the chance of passing good solutions will increase in the next generation. For each point P_i in group, depend on the condition $\mathcal{E} > Pa_i$ is satisfied, and where e is a uniform random number between [0, 1], the jth component of P_i is updated according to Eq. (19), otherwise it subsists unchanged.

$$P'_i(j) = P_r(j) - \mathcal{E} \times (P_t(j) - P_i(j)) \tag{19}$$

where P'_i is the new modified position of P_i. P_t and P_r are randomly selected points in the group, , $\mathcal{E}$ is the selected random number from the uniform distribution in the continuous space [0, 1]. While the first statistical method is performed on the elements of the points, the second statistic algorithm is aimed to change the position of a point regarding of the position other points in the group. This property causes to increase the quality of exploration, and improvement the diversification property. Before beginning the second method, once again, all points acquired from the first statistical method are ranked based on Eq. (19). If the condition) $\mathcal{E}>Pa_i$ (is preserved for a new point P'_i, the current position of P'_i is modified according to Eqs. (20) and (21), otherwise no update occurs.

$$P''_i = \acute{P}_i - \hat{\varepsilon} \times (\acute{P}_t - BP)\ ;\ \acute{\mathcal{E}} \le 0.5 \tag{20}$$

$$P''_i = \acute{P}_i - \hat{\varepsilon} \times (\acute{P}_t - \acute{P}_r)\ \ ;\ \acute{\mathcal{E}} > 0.5 \tag{21}$$

where P'_r and P'_t are randomly selected points concluded from the first method and $\mathcal{E}'$ is stochastic number produced by the normal distribution. The new point P''_i is replaced by P'_i if its fitness function value is better than P'_i [22]. The whole trend of the algorithm is indicated as Fig. 5.

4. Training RBF NN using SFSA

Generally, there are three methods for use of meta-heuristic algorithms in order to train RBF NN. The first method uses meta-heuristic algorithms to find a combination of connection weights, external nodes bias, emission parameters of the basis function and the vector of centers of hidden layer for having a minimum error in a RBF NN. The second method is to use meta-heuristic algorithms in order to find the proper structure of the RBF NN on a specific problem and the latest method uses meta-heuristic algorithms in order to find training algorithm parameters based upon gradients such as learning rate and momentum. In this paper, SFSA algorithm is exerted to a RBF NN by the use of the first method. In order to design a trainer algorithm for RBF NNs, it is necessary to properly define the connection weights, external nodes bias, emission parameters of the hidden layer's basis function, and the vector of centers as the particles of SFSA.

4. 1. Presentation of Training RBF NN using SFSA

Generally, there are three methods to present combination of passive parameters: 1) vector, 2) matrix and 3) binary state [1]. In vector presentation, each element presents only one vector. All weights, biases, emission parameters and centers should be obvious to train a RBF NN. In matrix presentation, each element is presented by a matrix. In binary presentation, each element is presented in the form of a string of binary bits. Any of these presentations has advantages and disadvantages that can be useful within a particular problem.

In the first method, it is simple to transform elements into a vector, matrix, and a string of binary bits but the process of their recycling is complicated. For this reason, this method has been used in typical NNs. In the second method, for complicated NNs recycling is simpler than encoding elements. This method fits very well for training algorithm of the general NNs. In the third method, it is fundamental to demonstrate variables in the form of binary. In this method, the length of each element gets increased when the structure of network gets complicated. Therefore, the encoding and decoding processes get profoundly complicated.

In this paper, vector method is applied to the NN, because its structure is not complicated. Also, MATLAB general toolbox is not used in order to reduce the time of

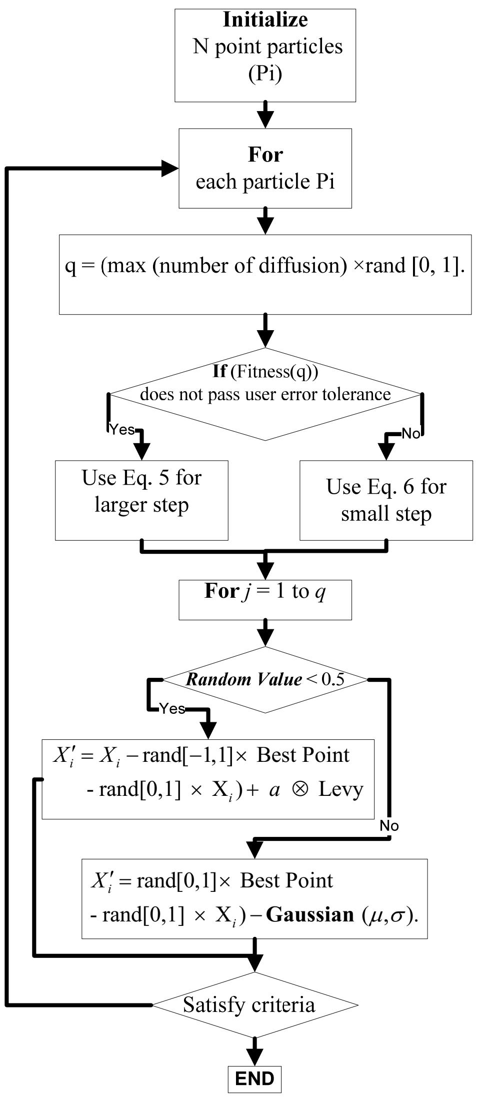

Initialize
N point particles
(Pi)
For
each particle Pi
q = (max (number of diffusion) ×rand [0, 1].
If (Fitness(q))
does not pass user error tolerance
Yes
No
Use Eq. 5 for larger step
Use Eq. 6 for small step
For j = 1 to q
Random Value < 0.5
Yes
X'i = Xi − rand[−1,1]× Best Point - rand[0,1] × Xi) + a ⊗ Levy
No
X'i = rand[0,1]× Best Point - rand[0,1] × Xi) − Gaussian (μ,σ).
Satisfy criteria
END

Fig. 5. The whole trend of the SFSA.

running RBF program. As previously represented, training RBF NN can be achieved by selecting optimum values for the following parameters:

- Weights between the hidden layer and output layer (W)
- Emission parameters of basis function of hidden layer (α)
- Vectors center of the hidden layer (c)
- Bias parameters of neurons in output layer (β)

The number of hidden layer's neurons is very important. Using more neurons than the normal number, leads to over-learned network, to increase structural sensibility and execution time of the algorithm. According to reference [28] and studies that have been done, four neurons are selected in the hidden layer. SFSA algorithm particles are weight ($\vec{w}$), emission ($\vec{\alpha}$), vector of center ($\vec{c}$) and bias vectors (β). A particle of SFSA algorithm can be expressed in vector Eq. (22):

$$P_i = [\vec{w}\,\vec{\alpha}\,\vec{c}\,\vec{\beta}] \tag{22}$$

As mentioned before, the ultimate goal of learning methods is training. Each training sample should include the appropriate calculation of the fitness of all particles. In this paper, particle's fitness function index (for all training samples) is calculated by Sum Squared Error (SSE) method as shown in equation (23):

$$f = E^{SSE} \tag{23}$$

5. Setting Parameters and Testing

In this section, in order to test the efficiency of SFSA algorithm in training RBF NN, in addition to SFSA algorithm, the network are taught by benchmark algorithms such as Population-Based Incremental Learning (PBIL), Evolution Strategy (ES), Ant Colony Optimization (ACO), Genetic Algorithm (GA), and Particle Swarm Optimization (PSO). The necessary parameters and initial values of these algorithms are shown in Table 1. Firstly, designed classifier is exerted on iris and lenses dataset (described in Table 2) and the performance of the classifier is tested in terms of the convergence speed and classification rate. Each algorithm is executed 10 times and then the average classification rate is illustrated in Table

Table 1. Parameters and initial values of the training algorithms.

Algorithm	Parameters	Value
SFSA	Maximum Diffusion Number (MDN)	1
	Population size	100
	Maximum number of iterations	250
PSO	Topology	Fully connected
	Cognitive constant (C1)	1
	Social constant (C2)	1
	Inertia constant (w)	0.3
	Population size	208
GA	Type	Real coded
	Selection	Roulette wheel
	Integration	Single point (1)
	Mutation	Uniform (0,0,1)
	Population size	208
	Maximum number of iterations	250
ACO	Primary pheromone (τ_0)	0.000001
	Pheromone updating constant (Q)	20
	Pheromone constant (q_0)	1
	Global pheromone reduction rate (p_g)	0.9
	Local pheromone reduction rate (p_t)	0.5
	Pheromone sensitivity (α)	1
	Visible sensitivity	5
	Population size	208
ES	λ	10
	σ	1
	Visible sensitivity	5
	Population size	208
PBIL	Learning rate	0.05
	Good member of population	1
	Bad member of population	0
	Elite selection parameter	1
	Mutation probability	0.1

Table 2. Benchmark dataset used in the paper.

Name	Default task	Attribute characteristics	Number of attributes	Number of samples	Year
Iris	Classification	Multivariate	4	150	1988
Lenses	Classification	Multivariate	4	24	1990

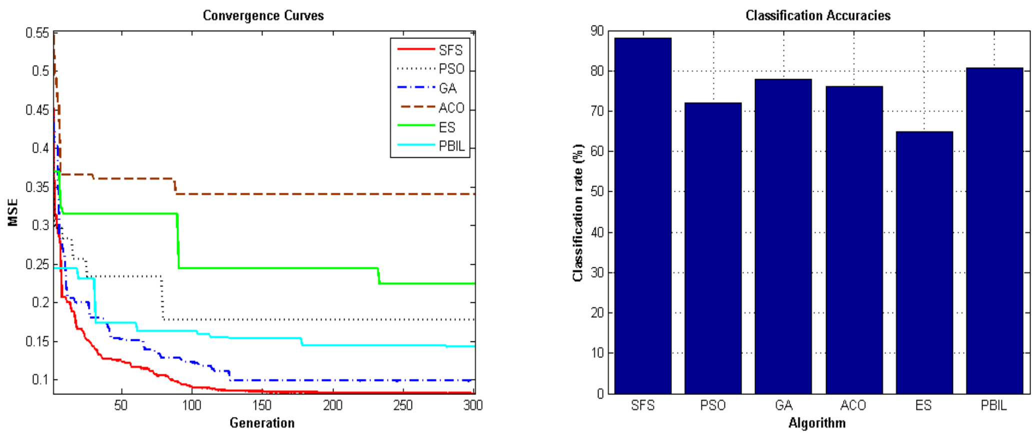


Fig. 6. Classification rate and convergence speed for iris dataset.

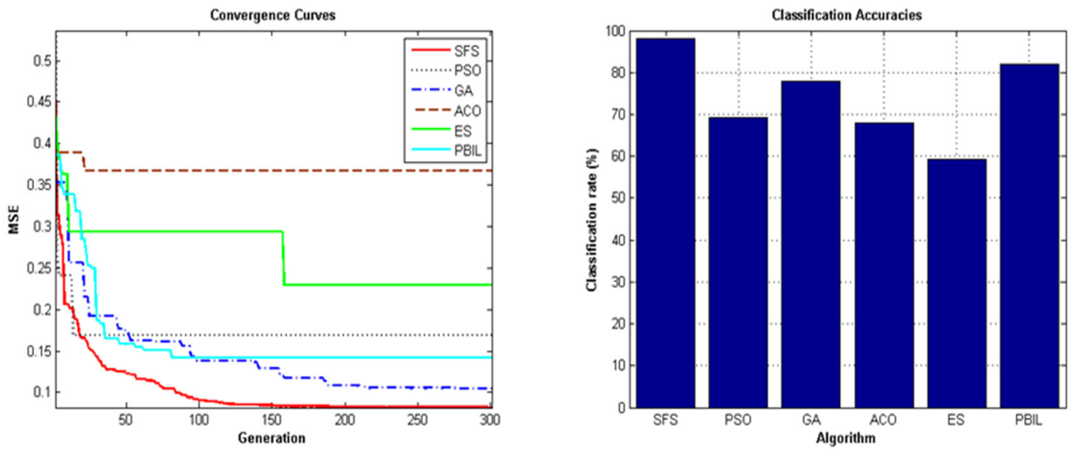


Fig. 7. Classification rate and convergence speed for lenses dataset.

3. The typical results for iris and lenses dataset are shown in Figs.6 and 7, respectively. In the subsequent section, sonar dataset will be intensely explained and then our designed classifiers will be tested on this type of dataset.

As can be seen in Figs.6 and 7, designed classifiers with SFSA algorithm have better performance than others in terms of convergence speed and classification rate. About iris dataset, SFSA algorithm categorizes dataset with an accuracy of about %90. This is while the best algorithm of between benchmark algorithms (in this case PBIL) offers accuracy of about %84. This improvement in performance is due to the a) exploitation ability of SFSA algorithm, which is created by stochastic nature of the fractals, and b) utilization power, where is created by the best-selected particles.

SFSA algorithm classifies lenses dataset with accuracy %97. This is while as the best algorithm of among compared algorithms (in this case PBIL) provides an accuracy

Table 3. Average classification rate of different algorithms for 10 runs (in term percent).

Algorithm	PBIL	ES	ACO	GA	PSO	SFSA
Iris	84%	65%	76%	78%	72%	90%
Lenses	86%	60%	68%	78%	70%	97%

of about %86. This example as well shows that the SFSA algorithm on data collection with little samples (here 24) also shows very good results and classifies data with an acceptable rate.

5. 1. Sonar Dataset

Sonar dataset used in this paper are extracted from Gorman and Sejnowski tests available in references [28, 29]. There are two types of echo (return signal) in this test: the first relates to the metallic cylinder (takes the role of real target) and the second relates to a rock as the same size as the cylinder (takes the role of false target). So that can be seen in Fig.8, the real and clutter target attribute is closely same and they cannot be separated by a lower linear or non-linear classifier (to be observable, dataset are reduced to three dimensions).

In this test a metal cylinder with a length of 5 feet and a rock with same size placed on the sea sand bed and a wide-band linear FM chirp pulse (ka=55.6) has been sent to them. Returned echoes have been collected in the distance 10 yards of them. Based on the SNR of received echo, of 1,200 collected echo has been selected 208 echoes that their SNR between 4dB to 15dB. From this 208 echoes, 111 echoes are of metal cylinders and 97 echoes are related to rocks. Fig.9 shows samples of received echoes from the rock and metal cylinder.

Pre-processing applied for acquiring spectrum envelope is showed in Fig.10. Fig.10a display a set of sampled apertures and Fig.10b show a set of sampling apertures that are been laid on bilateral spectrogram of the Fourier function of sonar echo. Spectral envelope is obtained from collecting each aperture's effects. In the test, spectral envelope is constructed from 60 spectrum samples which are normalized between 0 and 1. Each number expresses summation existing energy in sampled aperture. For instance, existence energy in the first aperture (η=0) after normalizing, constitutes the first number of 60 numbers within feature vector.

5. 2. Sonar Targets Classification

After pre-processing on sonar returned echoes and obtaining normalized dataset between 0 and 1, in this part of paper got exerted dataset of 60 *208 (208 samples which have 60 features) in RBF network which is trained by various algorithms. Outcomes are illustrated in Table 4 and Fig.11.

According to the Fig. 10, SFSA algorithm with %94 has the best performance and ES algorithm with %68 has the weakest performance. Regarding fluctuating nature and extra local maximum and minimum the possibility of failing within local maximum is too much for an algorithm such as ES. Whereas algorithms such as SFSA,

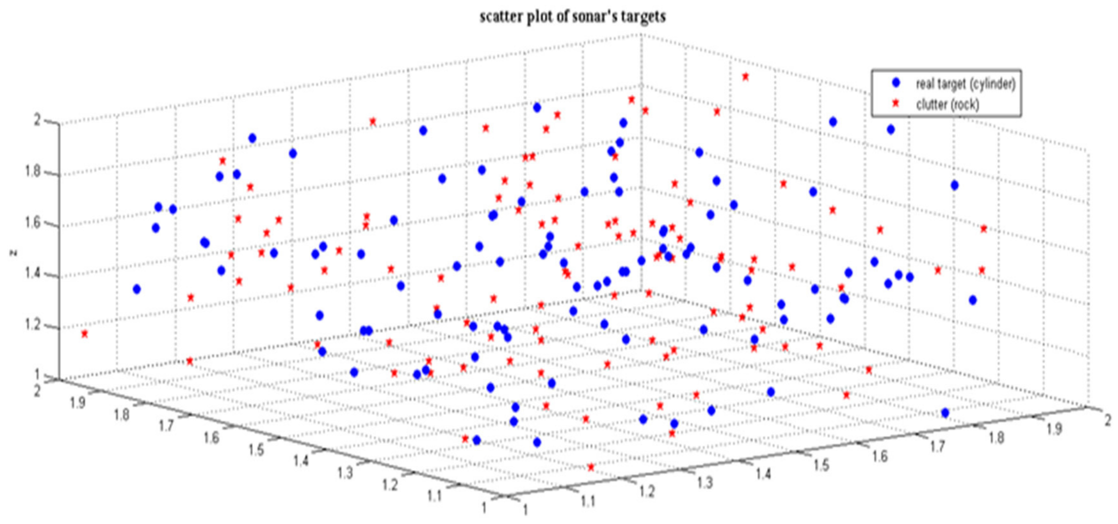


Fig. 8. Sonar dataset dispersion drawing.

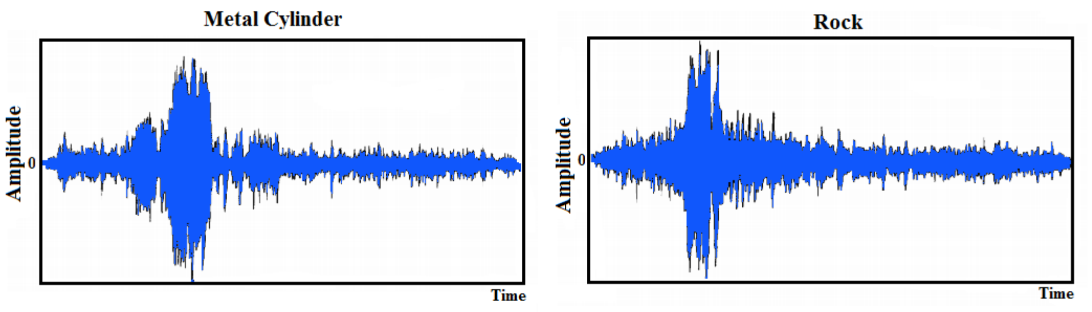
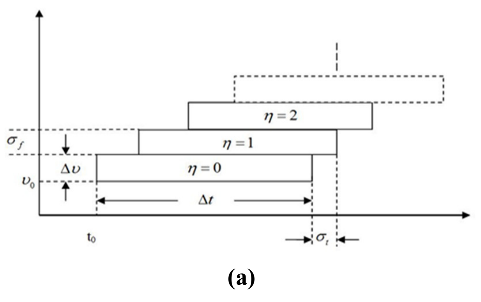


Fig. 9. Sample of received echo from the clutter (rock) and real target (metal cylinder).



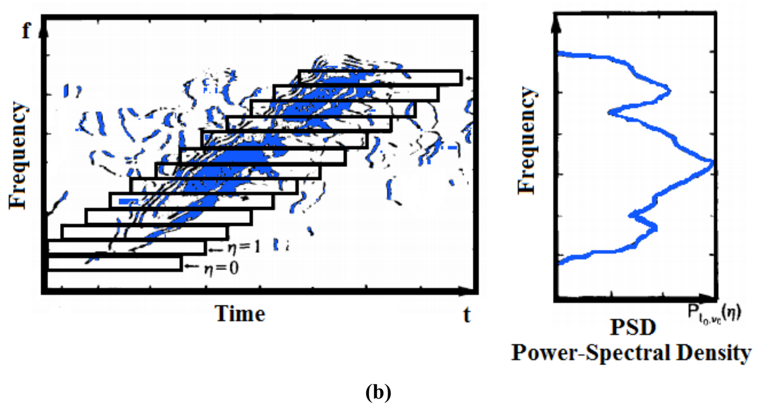


Fig. 10. Used pre-processing for obtaining spectral envelope.

Table 4. Average classification rate exerted algorithm different for 10 runs algorithms

Algorithm	PBIL	ES	ACO	GA	PSO	SFS
Sonar	72%	68%	70%	85%	88%	94%

PSO, and GA with random nature and having functions that change suddenly the problem space, have better performance than other algorithms. From another side, it can be observed that in this test, SFS algorithm due to high capability in detection has better performance for this type of dataset. As mentioned, sonar dataset due to covering whole searching space for classification requires algorithm which is strong in detection phase. SFSA is better than the other meta-heuristic ones in this field.

6. Conclusions

In this paper, a new meta-heuristic algorithm known as SFSA algorithm is firstly used to train a RBF NN. For the evaluation of the performance of designed classifier, a few datasets such as iris, lenses, and sonar dataset have been used and then obtained result are compared with PSO,

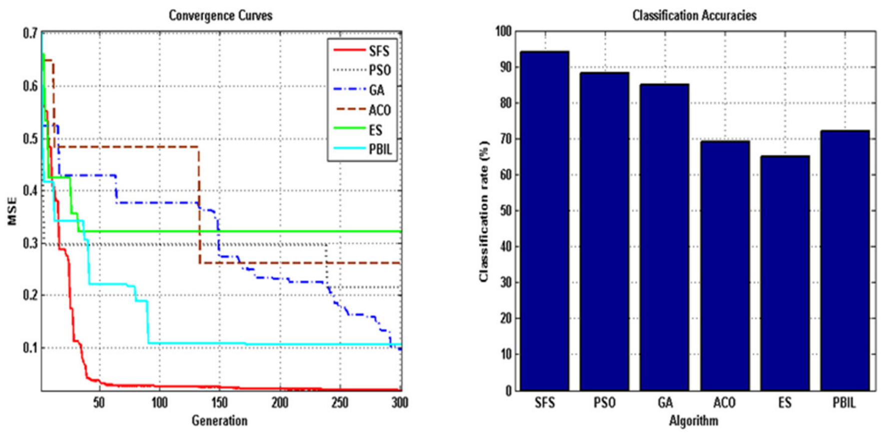


Fig. 11. Classification rate and convergence speed for sonar dataset.

ACO, ES, GA and PBIL benchmark algorithms. Results showed that the SFSA algorithm due to a simple structure and the ability to explore the search space is able to provide much better results in terms of convergence speed and classification accuracy in compare to benchmark algorithms. Also due to the simple structure of multi-layer perceptron NN, it can be used as a classifier in future works instead of RBF network.

References

[1] S. Mirjalili, S. M. Mirjalili and A. Lewis, "*Let a Biogeography-based Optimizer Train your Multi-Layer Perceptron,*" Journal of Information Sciences, Vol.269, pp.188-209, 2014.

[2] M. R. Mosavi, M. Khishe and E. Ebrahimi, "*Classification of Sonar Targets using OMKC, Genetic Algorithm and Statistical Moments*", Journal of Advances in Computer Research, Vol.7, No.1, pp.143-156, 2016.

[3] L. S. Nguyen, D. Frauendorfer, M. S. Mast and D. Gatica-Perez, "*Hire Me: Computational Inference of Hirability in Employment Interviews based on Nonverbal Behavior*," IEEE Transactions on Multimedia, Vol.16, No.4, pp.1018-1031, 2014.

[4] M. Khishe, M. R. Mosavi, and M. Kaveh, "*Improved Migration Models of Biogeography-based Optimization for Sonar Data Set Classification using Neural Network*," Applied Acoustic, Vol.118, pp.15-29, 2017.

[5] J. Moody and C. Darken, "*Fast Learning in Networks of Locally-Tuned Processing Units*," Neural Computing, Vol.1, No.1, pp.289-303, 1989.

[6] N. Karayiannis, "*Reformulated Radial Basis Neural Networks Trained by Gradient Descent,*" IEEE Transactions on Neural Networks, Vol.10, No.3, pp.657-671, 1999.

[7] C. Liu, H. Wang and P. Yao, "*On Terrain-Aided Navigation for Unmanned Aerial Vehicle using B-spline Neural Network and Extended Kalman Filter,*" IEEE Conference on Guidance, Navigation and Control (CGNCC), pp.2258- 2263, 2014.

[8] D. Simon, "*Training Radial Basis Neural Networks with the Extended Kalman Filter,*" Neurocomputing, Vol.48, No.1-4, pp.455-475, 2002.

[9] Q. Zhang and B. Li, "*A Low-Cost GPS/INS Integration based on UKF and BP Neural Network*," Fifth International Conference on Intelligent Control and Information Processing (ICICIP), pp.100-107, 2014.

[10] X. Li, T. Zhang, Z. Deng and J. Wang, "*A Recognition Method of Plate Shape Defect based on RBF-BP Neural Network Optimized by Genetic Algorithm*," International Conference on Control and Decision, pp.3992-3996, 2014.

[11] K. Narendra and M. Thathachar, "*Learning Automata-An Introduction*," Prentice-Hall, Englewood Cliffs, NJ, 1989.

[12] S. Kirkpatrick, Cl. Gelatt, and M. Vecchi, "*Optimization by Simulated Annealing,*" Science, New Series, Vol.220, No.4598, pp. 671-680, 1983.

[13] C. Buzzi, L. Grippo and M. Sciandrone, "*Convergent Decomposition Techniques for Training RBF Neural Networks*," Neural Computation, Vol.13, No.8, pp.1891-1920, 2001.

[14] W. Zhang, Q. Luo and Y. Zhou, "*A Method for Training RBF Neural Networks Based on Population Migration Algorithm*," IEEE Conference on Artificial Intelligence and Computational, 2009 (DOI: 10.1109/AICI.2009.35).

[15] T. Kurban and E. Beşdok, "*A Comparison of RBF Neural Network Training Algorithms for Inertial Sensor Based Terrain Classification*," Sensors, Vol.9, No.8, pp.6312-6329, 2009.

[16] H. X. Huan, D. T. T. Hien and H. H. Tue, "*Efficient*

Algorithm for Training Interpolation RBF Networks with Equally Spaced Nodes," IEEE Transactions on Neural Networks, Vol.22, No.6, pp.982-988, 2011.

[17] H. Yu, P. D. Reiner, T. Xie, T. Bartczak and B. M. Wilamowski, "*An Incremental Design of Radial Basis Function Networks,*" IEEE Transactions on Neural Networks and Learning Systems, Vol.25, No.10, pp.1793-1803, 2014.

[18] C. Cecati, J. Kolbusz, P. Różycki, P. Siano and B. M. Wilamowski, "*A Novel RBF Training Algorithm for Short-Term Electric Load Forecasting and Comparative Studies,*" IEEE Transactions on Industrial Electronics, Vol.62, No.10, pp.6519-6529, 2015.

[19] D. Lam and D. Wunsch, "*Unsupervised Feature Learning Classification with Radial Basis Function Extreme Learning Machine using Graphic Processors*," IEEE Transactions on Cybernetics, Vol.47, No.1, pp.224-231, 2017.

[20] M. R. Mosavi, M. Khishe and A. Ghamgosar, "*Classification of Sonar Data Set using Neural Network Trained by Gray Wolf Optimization*," Journal of Neural Network World, Vol.26, No.4, pp.393-415, 2016.

[21] E. G. Talbi, "*Metaheuristics: From Design to Implementation*", Willy, 2009.

[22] H. Salimi, "*Stochastic Fractal Search: A Powerful Metaheuristic Algorithm*", Knowledge-Based Systems, Vol.75, pp.1-18, 2015.

[23] B. Yu and X. He, "*Training Radial Basis Function Networks with Differential Evolution,*" IEEE International Conference on Granular Computing, pp.369-372, 2006.

[24] S. Mirjalili, S. Z. M. Hashim and H. M. Sardroudi, "*Training Feedforward Neural Networks using Hybrid Particle Swarm Optimization and Gravitational Search Algorithm,*" Applied Mathematics and Computation, Vol.218, pp.11125-11137, 2012.

[25] B. B. Mandelbrot, "The Fractal Geometry of Nature", Times Books, 1983.

[26] T. T Vicsek, "*Fractal Growth Phenomena*", World Scientific Publishing Company Incorporated, 1992.

[27] K. J. Falconer, "*Random Fractals*", Mathematical Proceedings of the Cambridge Philosophical Society, Vol.100, No.03, pp. 559-582, 1986.

[28] R. P. Gorman and T. J. Sejnowski, "*Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets,*" Neural Networks, Vol.1, pp.75-89, 1988.

[29] http://archive.ics.uci.edu/ml/datasets.

Mohammad Reza Mosavi received his B.Sc., M.Sc., and Ph.D. degrees in Electronic Engineering from Iran University of Science and Technology (IUST), Tehran, Iran, in 1997, 1998, and 2004, respectively. He is currently faculty member of Department of Electrical Engineering of IUST as professor. He is the author of more than 300 scientific publications on journals and international conferences. His research interests include circuits and systems design.

Mohammad Khishe received his B.Sc. degree in University of Imam Khomeini Marine Sciences, Noshahr, Iran, M.Sc. degrees in Islamic Azad University, Qazvin Branch in 2007 and 2011, respectively. He is currently a Ph.D. student in the Electronic Engineering at Iran University of Science and Technology. His research interests include neural networks, meta-heuristic algorithms and digital design.

Yousof Hatam Khani received his B.Sc. degrees in University of Imam Khomeini Marine Sciences, Noshahr, Iran, in 2002. He is currently a M.Sc. student in the Department of Electrical Engineering at University of Imam Khomeini Marine Sciences, Noshahr, Iran. His research interests include nueral networks and cryptography.

Mahtab Shabani received her B.Sc. degree in Physics from Iran University of Science and Technology, Tehran, Iran and M.Sc. degrees in the Department of Physic University of Hormozgan, Bandar Abbas, Iran in 2003 and 2015, respectively. Her research interests include siesmology and neural networks.