PAN
ARCHIVES OF ACOUSTICS
Vol. **45**, No. 4, pp. 753–764 (2020)
DOI: 10.24425/aoa.2020.135281

Technical Note

Training RBF NN Using Sine-Cosine Algorithm for Sonar Target Classification

Yixuan WANG[(1)*], LiPing YUAN[(1),(2)*], Mohammad KHISHE[(3)**]
Alaveh MORIDI[(4)], Fallah MOHAMMADZADE[(3)]

[(1)] *School of Information Engineering, Wuhan University of Technology*
China

[(2)] *School of Information Engineering, Wuhan Huaxia University of Technology*
China

[(3)] *Department of Marine Electronic and Communication Engineering*
Imam Khomeini Marine Science University of Nowshahr
Nowshahr, Iran

[(4)] *Department of Electrical Engineering, Iran University of Science and Technology*
Narmak, Tehran, Iran

[*]Yixuan Wang and LiPing Yuan contributed equally to this work (co-first)
[**]Corresponding Author e-mail: m_khishe@alumni.iust.ac.ir

(*received February 15, 2019; accepted June 26, 2020*)

Radial basis function neural networks (RBF NNs) are one of the most useful tools in the classification of the sonar targets. Despite many abilities of RBF NNs, low accuracy in classification, entrapment in local minima, and slow convergence rate are disadvantages of these networks. In order to overcome these issues, the sine-cosine algorithm (SCA) has been used to train RBF NNs in this work. To evaluate the designed classifier, two benchmark underwater sonar classification problems were used. Also, an experimental underwater target classification was developed to practically evaluate the merits of the RBF-based classifier in dealing with high-dimensional real world problems. In order to have a comprehensive evaluation, the classifier is compared with the gradient descent (GD), gravitational search algorithm (GSA), genetic algorithm (GA), and Kalman filter (KF) algorithms in terms of entrapment in local minima, the accuracy of the classification, and the convergence rate. The results show that the proposed classifier provides a better performance than other compared classifiers as it classifies the sonar datasets 2.72% better than the best benchmark classifier, on average.

Keywords: classifiers; radial basis function neural network; sine-cosine algorithm; sonar.

1. Introduction

RBF NNs are one of the most useful tools for soft computing, by which non-linear optimisation problems can be solved. In general RBF NNs are used for pattern classification, data estimation, and function approximation (Mirjalili *et al.*, 2014; Abedifar *et al.*, 2013; Nguyen *et al.*, 2014). Not only RBF NNs have the advantage of strong global approximation capability, but also they benefit from other powerful characteristics such as a compact structure, the ability to estimate any continuous network, they have the benefits of easy design, good generalisation, strong tolerance to input noise, and online learning ability (Park, Sandberg, 1993; Du, Swamy, 2014; Yu *et al.*, 2011).

Regardless of its application, the special ability of RBF NNs is learning (Auer *et al.*, 2008). To be more precise, learning means that these networks, like the human brain, can learn from an experience or experiment. The target of this process is to minimise some error criterion by tuning the parameters of the network. An RBF NN with a common architecture has a single hidden layer which consists of three main parameters: the connection weights, widths, and centres. The

conventional method for training RBF NN is a training process with two sequential stages. In the first stage, some unsupervised clustering algorithms such as k-means (Yu-Qing *et al.*, 2016), vector quantisation (Vogt, 1993), or decision trees (Hyontai, 2011) are used to find the centres of the hidden layer and the widths. In the second stage, the connection weights between the hidden layer and the output layer are learned. Usually, the weights are discovered linearly using the simple linear least squares (LS), the orthogonal least squares (OLS) algorithms (Lin *et al.*, 2009; Chen *et al.*, 1999), or a gradient descent algorithm (Neruda, Kudova, 2005).

Although RBF NNs have many advantages, training a network using the conventional approaches comes with common limitations in the terms of convergence speed and prediction accuracy, especially for multimodal search space such as sonar target classification. For example, when using a classical gradient descent method, trapping in local minima is highly likely. Additionally, in most conventional algorithms and gradient descent methods the initial parameters' setting is very important (Fasshauer, Zhang, 2007). As a result, many researchers were motivated to investigate the use of meta-heuristic algorithms as an alternative approach for training RBF NNs due to the mentioned reasons.

The advantage of meta-heuristic algorithms is searching in a better range regardless of the size of the search area. Meta-heuristic algorithms have some outstanding features including being gradient free (Mirjalili *et al.*, 2012), solving large scale problems (Khishe *et al.*, 2017), achieving reliable and robust optima (Mosavi, Khishe, 2017), simple design, and flexible implementation (Mosavi *et al.*, 2016). They also have a higher efficiency in searching for a global solution when the search space is highly multi-modal and challenging such as sonar target classification (Yang, 2014; Mirjalili *et al.*, 2014; Faris *et al.*, 2016).

Some of these algorithms used for training RBF NNs as part of different training schemes are genetic algorithms (Ding *et al.,* 2012; Neruda, Kudova, 2005; Chen *et al.*, 1999; Zhang *et al.*, 2014), learning automata (Khishe, Aghababaee, 2013), particle swarm optimisation (Chen *et al.*, 2009; Wu *et al.*, 2010; Zhong *et al.*, 2014), ant colony optimisation (Chun-Tao *et al.*, 2007), differential evolution (Yu, He, 2006), biogeography based optimiser (Aljarah *et al.*, 2016), moth-flame optimiser (Faris *et al.,* 2017), and firefly algorithm (Horng *et al.*, 2012).

An interesting training method for RBF NNs is to search for all the required parameters simultaneously in just one stage. Despite the simple look of the training scheme of this approach, it might have some challenges. The main challenge is that in the case of the sonar target classification problem, the search space becomes too large and therefore the problem is considered a complex and non-linear optimisation task (Chen *et al.*, 2009). This is the main reason why some evaluation based meta-heuristic methods like GA perform slowly and require more iterations to converge (Gan *et al.*, 2012).

To sum up, on the one hand, there is a well known theorem in literature called No Free Lunch (NFL) theorem which proves that there is no meta-heuristic algorithm accruing the best answer for all optimisation problems. So, some algorithms are better for particular optimisation problems rather than other meta-heuristic algorithms (Ho, Pepyne, 2002). This is why this field of study is actively open and many researchers are working on it. On the other hand, reference (Mirjalili, 2016) proved that the SCA is a powerful algorithm to avoid getting stuck in local minima when working with a high-dimensional dataset. These two motivations convinced us to train an RBF NN using the SCA for the sake of designing an efficient and reliable sonar data set classifier. The current approach in this work is based on simultaneously optimising all the parameters of the network including the centres, widths, and the connection weights.

To the best of our knowledge, this is the first time the SCA algorithm is used for training RBF NNs. In order to evaluate the performance of the proposed classifier, the experiments in this work are carried out in two steps: first, we assess the SCA with two well regarded sonar dataset classification problems and second, we compare it to a well known classical training method which is commonly used in literature for training RBF NNs. In this regard, we developed a real world challenging underwater sonar data set to benchmark and compare the training algorithms.

The rest of the paper is organised as follows: In Sec. 2, RBF NNs are trained by the proposed algorithm. In Sec. 3, the experimental results are described and finally, a conclusion is given in Sec. 4.

2. Training RBF NNs using meta-heuristic algorithms

In general, there are three methods for using meta-heuristic algorithms to train RBF NNs. The first method is to use meta-heuristic networks to find the combination of connection weight, output node bias, propagation parameters of the basic function, and vectors of the hidden layer centre for having the least amount of error in an RBF NN. The second method is to use meta-heuristic networks to find the proper structure of an RBF NN in a specific problem, and the last method involves using meta-heuristic networks to find the parameters of gradient learning algorithm such as learning rate and movement size. In this paper, the algorithm is applied to RBF NNs using the first method. In order to design a training algorithm for RBF NNs,

some parameters must be displayed properly in the algorithm. These parameters are the connection weights, the output node bias, propagation parameters of the hidden layer's basic function, and vectors of the hidden layer centres. Therefore, a general description of an RBF NN's training methodology can be represented as Fig. 1.

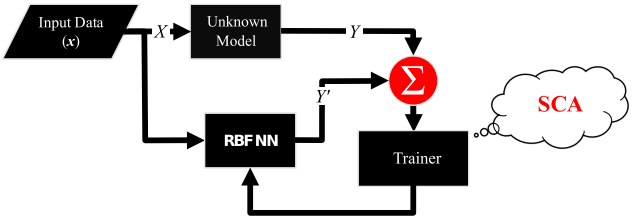

Fig. 1. Block diagram of a typical RBF NN.

2.1. Training RBF NN using the SCA

In general, there are three methods to display the combination of unknown parameters: vector, matrix, and binary state (Mirjalili *et al.*, 2014). To train an RBF NN, all weights, biases, propagation parameters, and centres must be known. Each of these representation modes has its advantages and disadvantages that can be useful for a particular problem.

In this paper, since we are not dealing with complex RBF NNs, the vector representation is used. As previously mentioned, RBF NNs can be trained by selecting the optimal values for parameters indicated in Table 1.

Table 1. The parameters of an RBF NN need to be optimised.

No	Parameters	Description
1	W	Weights between the hidden layers and the output layer
2	α	Propagation parameters of the hidden layer's basic function
3	c	Centr evectors of the hidden layer
4	β	Bias parameters of the output layer neurons

The number of neurons of the hidden layer in RBF NNs is very important. Using neurons more than needed, will lead to overlearning of network and increases structure complexity and algorithm execution time. According to the reference (Abu-Mouti, El-Hawary, 2012) and the investigations carried out, 10 is chosen for the number of the hidden layer neurons. By increasing the number of hidden layer neurons, the network performance is not significantly increased, although, space and time complexity of the network increases dramatically. The search agents of the SCA algorithm include weights ($\mathbf{w}$), propagation ($\boldsymbol{\alpha}$), centre vectors ($\mathbf{c}$), and bias vectors ($\boldsymbol{\beta}$). A search agent in the SCA can be represented as Eq. (1):

$$P_i = [\mathbf{w}\boldsymbol{\alpha}\,\mathbf{c}\boldsymbol{\beta}]. \tag{1}$$

As mentioned, the ultimate goal of the learning methods is to train RBF NNs. In this regard, each sample of training should include the value of the fitness function for all search agents. In this paper, the fitness function of the search agent (for all samples of training) is calculated using the mean squared error (MSE), which is represented by Eq. (2):

$$\text{MSE} = \frac{1}{k}\sum_{i=1}^{k}(y - \widehat{y})^2. \tag{2}$$

In this equation, as shown in Fig. 1, y demonstrates the desired training output and $\widehat{y}$ indicates the output of the evaluated RBF NN. Consequently, the ultimate goal is to set the optimal parameters to minimise the MSE value.

3. Experiments and results

In this section, to test the efficiency of the designed classifier, two benchmarks of underwater sonar classification problems were used. Also, an experimental underwater target classification was developed to practically evaluate the merits of the RBF based classifier in dealing with high-dimensional real word problems. To have a comprehensive comparison, in addition to the SCA algorithm, the PBIL, ES, ACO, GA, and PSO algorithms are also used to train this network. The functionality of these classifiers are tested in terms of classification rate, getting stuck in local minima, and convergence speed. The parameters and initial values of these algorithms are shown in Table 2.

All algorithms were performed under the same conditions to achieve fairness in comparative experiments. Among them, the population and the iteration time was set to 30 and 500 respectively. To reduce the impacts of random factors in the algorithm on the results, all the compared algorithms were run individually 30 times in each function and averaged as the final running result. On the purpose of measuring experiment results, standard deviation (STD), and average results (AVG) were employed to evaluate the results. Note that the best results will be bolded (take one in the case of juxtaposition). Wilcoxon sign-rank test method (Gorman, Sejnowski, 1998) was exerted to verify whether RBF-SCA has obvious advantages over pairwise comparison. If the p-value produced by the comparison is below the significant level of 0.05 in this case, it means that the achievements of the algorithm in pairwise comparison have obvious advantage in the statistical sense. On the other hand, it is considered that the difference between the two contestants is not noticeable in the statistical sense.

Table 2. Parameters and initial values of the training algorithms.

Parameter	Algorithm	Value
GA	Number of generation	4000
	Choice type	Roulette wheel
	Type of combination	Single point
	Rate of combination	0.8
	Type of growth	Steady
	Rate of growth	0.05
GD	Learning parameter	$\boldsymbol{\eta = 0.01}$
KF	Covariance matrix of state estimation error	$R = 40I$
	Covariance matrix of artificial noise ω_k	$Q = 40I$
	Covariance matrix of artificial noise ν_k	$PO = 40I$
GSA	Number of search agents	60
	G_0	1
	α	20
	Maximum iteration	500

The p-value achieved from this expression is shown in Tables 3–5 (see Subsec. 3.5), where the p-values were all lower than 0.05 when compared with traditional algorithms. Therefore, the RBF-SCA has significant advantage in these problems compared to the traditional algorithms.

3.1. Gorman and Sejnowski Dataset

The first data set used in this paper is derived from the Gorman and Sejnowski experiment in references (Gorman, Sejnowski, 1998; connectionist bench). In this experiment, there are two types of echoes (return signal), the first one is related to a metal cylinder (plays the role of a real target) and the second one is from a rock of the same size as the rock (plays the role of clutter or a false target). In this experiment, a 5-foot-long metal cylinder and a rock of the same size were placed in the sandy seabed, and a wide-band linear FM chirp pulse (ka = 55/6) was transmitted to them. Based on the SNR of the received echo, out of 1200 collected echoes, 208 of those whose SNR was between 4 dB and 15 dB, were selected. 111 echoes, out of these 208 selected ones, are related to the metal cylinder and 97 of them are related to the rock. Figure 2 shows a sample of the received echoes from the rock and the metal cylinder. It can be seen that the echoes from the real target (metal cylinder) and the clutter (rock) are very similar, therefore they cannot be classified by a linear or a low degree non-linear classifier.

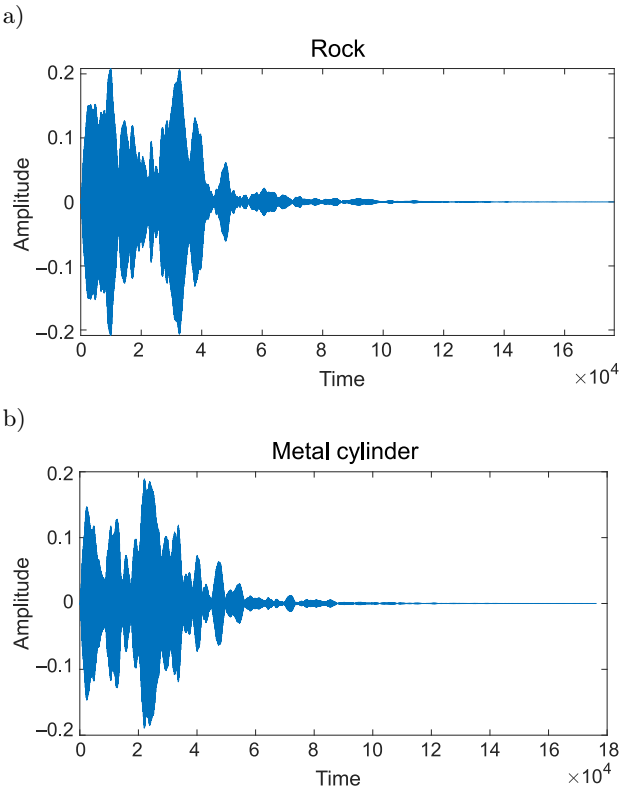

Fig. 2. Sample of the received echoes from the rock (a) and the metal cylinder (b).

The preprocessing used to obtain the spectral coverage is shown in Fig. 3. In Fig. 3a a set of sampling windows is shown; Fig. 3b illustrates a set of sampling windows on the two-dimensional spectrogram of the Fourier transform of the sonar echo. Consequently, spectral coverage is obtained by accumulating the effects of each window. In this experiment, the spectral coverage is formed by a set of 60 spectral samples that were normalised between 0 and 1. Each of these numbers represents the total energy in the relevant sampling window; for instance, the energy of the first window (η = 0) after normalisation forms the first number of the 60 numbers in the feature vector. The detailed information of the return echoes from the rock

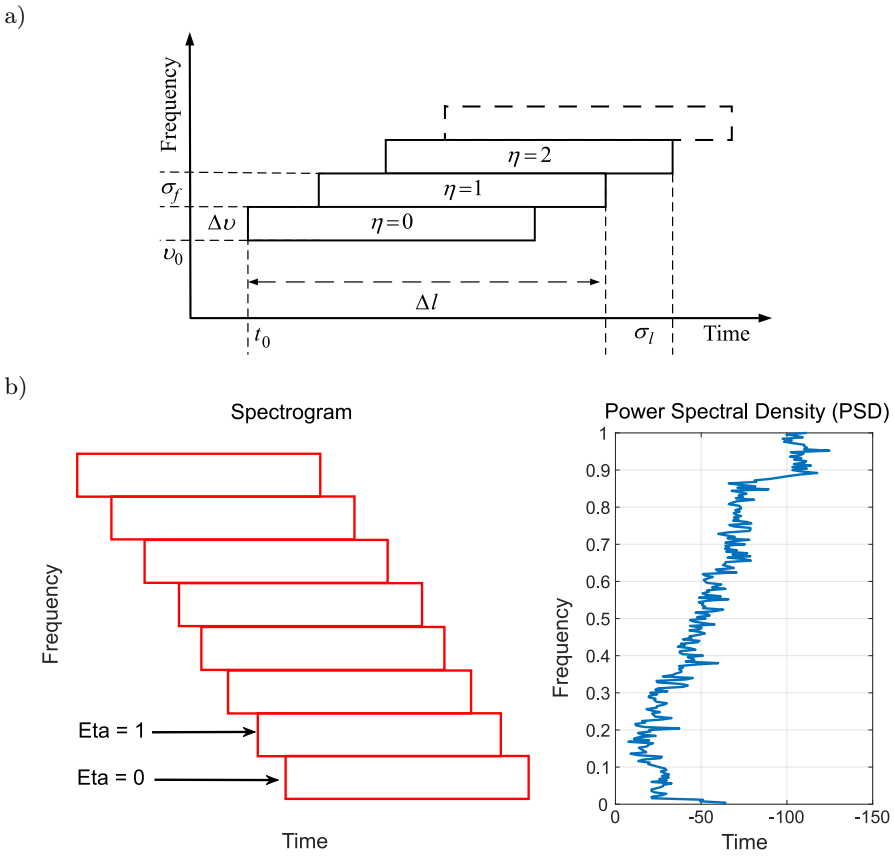

Fig. 3. Preprocessing used to obtain spectral coverage: a) sampling windows, b) applying a set of sampling windows on the two-dimensional spectrogram.

and the metal cylinders can be observed in references (Gorman, Sejnowski, 1998; connectionist bench).

3.2. Common Data Set 2015 (CDS2015)

The second data set used to evaluate the designed classifier is the Common Data set (CDS2015). This is a range of bathymetric and backscatter datasets collected using the latest shallow water survey techniques so that comparisons can be made and the merits of the different approaches are investigated. To this end, areas were chosen in Plymouth Sound and Wembury Bay, which offered a good variety of depth, seabed types, and sub-sea conditions, as shown in Fig. 4.

Shallow water term has been clearly defined as water depths lower than 200 meters. However, in this particular collection, the depths never exceed 40 meters. In order to obtain such a performance and quality in the collected data, an exceptionally high standard given the safety of navigation implication of sub-40-meter surveying is required.

This shallow water dataset provides a unique opportunity to experience the current technological capabilities and advancements made by Kongsberg Maritime in the shallow water survey market. Raw data have been provided throughout, with no modifications or deviations, from the reference (Gutiérrez, Zhao, 2015). Therefore, CDS2015 is completely raw, and it has had no filtering, either statistical or manual one.

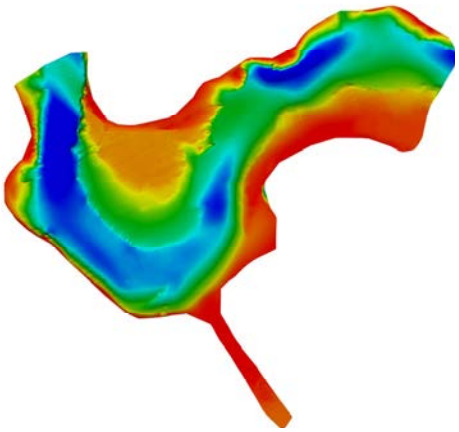

Fig. 4. Processed survey area 1.

As outlined by the data collection instruction, the four mentioned targets were traversed using the guidelines at set speeds, set swath angles, set direction, and within a set distance to the line. For more details, the descriptions of the targets can be found in reference (Gutiérrez, Zhao, 2015). To better understand the types of objects, these four targets are illustrated in Fig. 5.

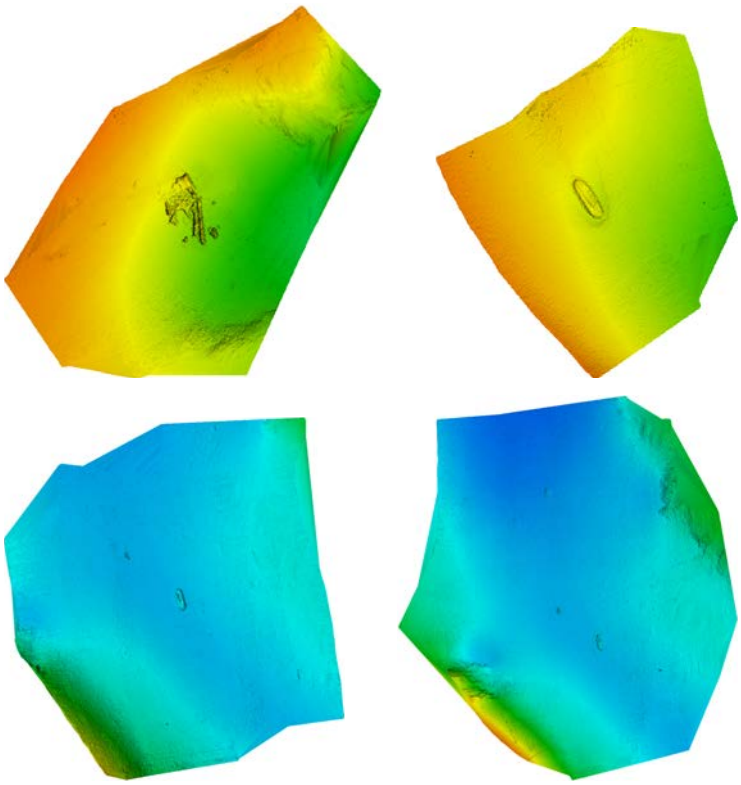

Fig. 5. Four various targets in CDS2015.

3.3. Experimental sonar dataset

This section tries to briefly introduce the utilised sonar dataset, collected by the authors. The sonar data set was obtained from experiments conducted on the shores of the Caspian sea. This region, with a depth of 40 to 100 meters, is assumed to be a shallow water area. This data set was collected by the sonobuoy shown in Fig. 6a. Also, information about environmental parameters such as temperature, salinity, water depth, wind speed, and seabed type has been obtained from the oceanographic buoy of Nowshahr Ports and Maritime Organization (PMO), which is shown in Fig. 6b (Khishe, Mosavi, 2017).

Fig. 6. Sonobuoys used for sonar dataset acquisition: a) designed sonobuoy, b) oceanographic sonobuoy.

In this experiment, 6 objects including 4 targets and 2 non-targets have been placed on the homogenous seabed. The transmitted ping is a wideband linear frequency modulated signal which covers the frequency range of 5–110 Hz. An electrical motor rotates the objects (at the bottom of the sea) through 180 degrees, with one degree precision.

A good data set plays a vital role in the sonar data classification. Due to the high volume of raw data obtained in the previous step, a great load of computing shall be expected. In order to reduce the computational load of the classifier and the feature extractor, the process of revealing the probable targets from the total received data is necessary. For this purpose, the intensity of the received signal is used.

Due to the shallow depth of the sea in this area, phenomena such as multi-path propagation, secondary reflections, and reverberation are inevitable. The effects of these artifacts will be eliminated after the detection step and before the feature extraction step, using a filter in the field of matched filters. In the next step, the inverse filtering will be used to recover the original reflection signal. This stage is based on the fact that it is much easier to separate these artifacts in the field of the matched filter than in the time domain. The whole preprocessing is executed in four steps as follows:

- **Scaling**: In order to eliminate the effect of the amplifier's gain and filter in the data collection stage, the raw signal is converted to the scaled signal.

- **Down-sampling**: The main sampling rate is 2 MHz, which is much higher than the bandwidth of the main signal. In order to reduce the sampling rate without losing useful information, reference (Preston, 2004) has been used. In this reference, for each ping a fixed number of points is selected in the sampling stage, using environmental information such as water depth, operating frequency, and monitored area. Here, 2048 points are selected so that valuable information for feature extraction is not lost.
- **The process of removing artifacts and multi-path effects**: In this method, by using the cross-correlation of the back-scattered signal with the signal sent at each angle, the location of the maximum output of the matched filter named x is determined. A window that covers the range [x – left: x + right] is then applied to the signal. In this example, the right point is equal to 300 and the left point is equal to 211, which forms a 512-point window. This segmented signal is zero-padded in order to maintain the original signal size and to eliminate the effect of the sent signal removal by Eq. (3)

$$H(k) = \frac{X(k)}{|X(k)|^2 + c}. \tag{3}$$

In this equation $X(k)$ is the Fourier transform of the sent signal and $c = 0.0025 \cdot \max(|X(k)|^2)$ is added to the relation to solve the singularity problem. The output of this step is a pure back-scattered signal without the effects of the artifacts.

- **Normalisation**: At last, we scale each target in such a way that each one of them has the same target strength. For this purpose, each back-scattered signal is divided by the SRA, which is the largest domain that is less than 90% of the maximum received domains.

Figure 7 demonstrates a set of signals received from different targets and non-targets that are a function of the frequency and direction of the objects. Furthermore, after target detection, feature extraction stage must be executed.

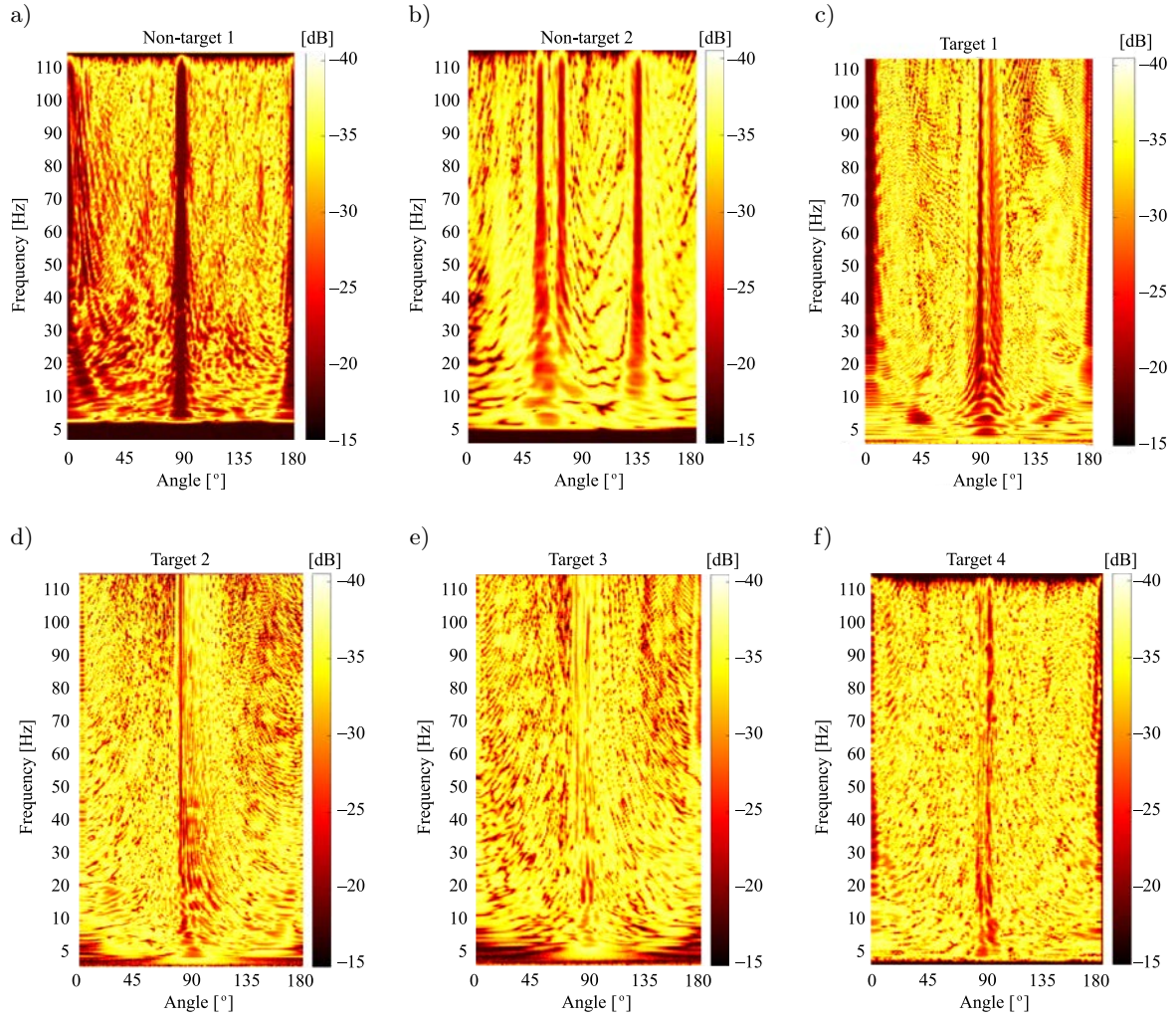

Fig. 7. Sample echoes returned from the target and non-target objects: a) non-target 1, b) non-target 2, c) target 1, d) target 2, e) target 3, f) target 4.

3.4. Feature extraction

After preprocessing, the detected echoes are transformed into the frequency domain (named $S(k)$) and then delivered to the feature extraction stage. In this step, first, the power spectral density (PSD) function of the detected signal is calculated by Eq. (4):

$$|S(k)|^2 = S_r^2(k) + S_i^2(k), \tag{4}$$

where $S_r^2(k)$ and $S_i^2(k)$ are the real and the imaginary parts of the detected signal's Fourier transform, respectively. In the next step, the spectral energy is filtered by a Mel-scaled triangular filter. Therefore, the output energy of the l-th filter is calculated by Eq. (5):

$$E(l) = \sum_{k=0}^{N-1} |S(k)|^2 \; H_l(k), \tag{5}$$

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 $l = 0, 1, ..., M_l$.

The dynamic range of the Mel-scale filtered energy spectrum is compacted by the logarithmic function as Eq. (6):

$$E(l) = \log(E(l)). \tag{6}$$

Mel-scaled frequency cepstral coefficients (MFCC) are converted back to the time domain using discrete cosine transform (DCT), shown in Eq. (7):

$$C(n) = \sum_{l=1}^{M} e(l) \cos\left(n\left(l - \frac{1}{2}\right)\frac{\pi}{M}\right). \tag{7}$$

Finally, the feature vector will be in the form of Eq. (8):

$$X_m = (c(0)\, c(1) \; \ldots \; c(P-1))^{\mathrm{T}}. \tag{8}$$

The entire classification process includes preprocessing, detection, feature extraction, and classifier designing, which is shown in Fig. 8.

3.5. The simulation result

After preprocessing on sonar recursive echoes and obtaining a normalised dataset within the range of (0, 1), in this part of the paper the dataset with dimensions 208×60 (400 samples, each of which has 128 features), 625×144, and 400×128 for Gorman & Sejnowski, CDS2015, and developed data set, respectively, is applied to an RBF NN which is trained by various algorithms. The results are shown in Figs 9–11 and Tables 3–5 below.

As shown in Figs 9–11, the convergence rates of RBF-SCA have the fastest convergence rate among all the benchmark classifiers, followed by RBF-GSA and RBF-GA, respectively. At the same time, RBF-GD and RBF-KF, with steady convergence curves, fail to reach a convincing solution after a certain amount of iterations. This is because of local optima stagnation which indicates that RBF-SCA can still show better capability in dealing with high-dimensional real world problems.

From the data in Tables 3–5, it can be observed that the average results of the RBF-SCA classifier with 91.8944% of correct classification rate has the best performance, and the RBF-GD classifier with 86.9231% has the weakest functionality. Due to having oscillation nature and too many local minima, the probability of getting trapped in local minima for algorithms such as GD is very high. The weak performance of GD and KF algorithms confirms this issue, while SCA, GSA, and GA algorithms with random nature and not using derivatives have better performance than other

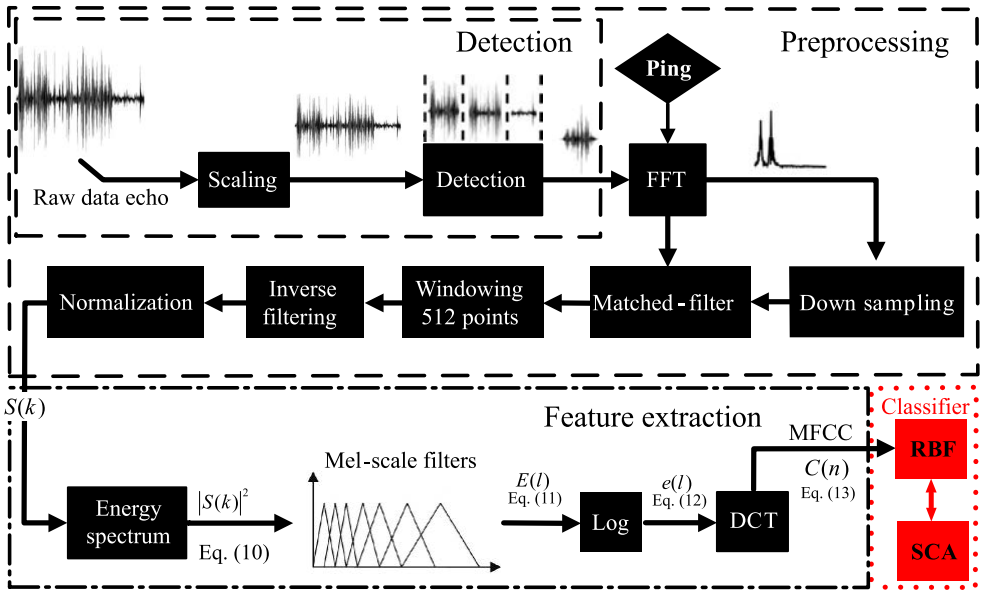

Fig. 8. Entire classification process: preprocessing, detection, feature extraction, and classifier.

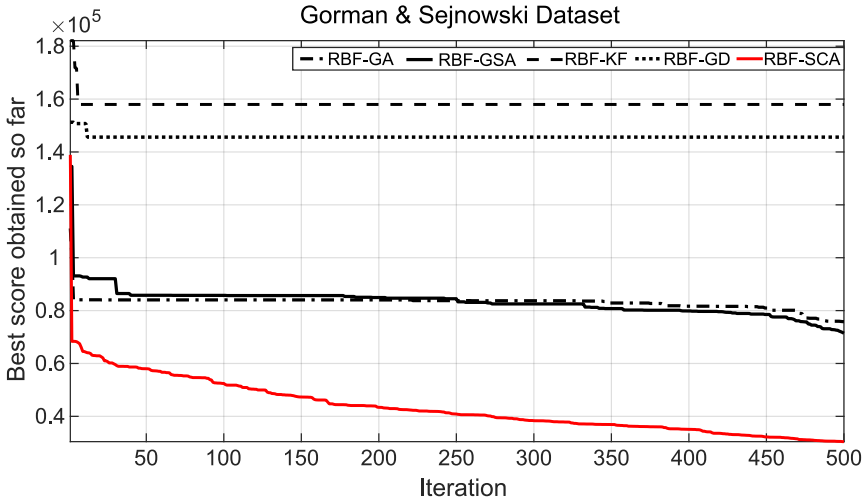

Fig. 9. Convergence curves of various classifiers applied to Gorman & Sejnowski dataset.

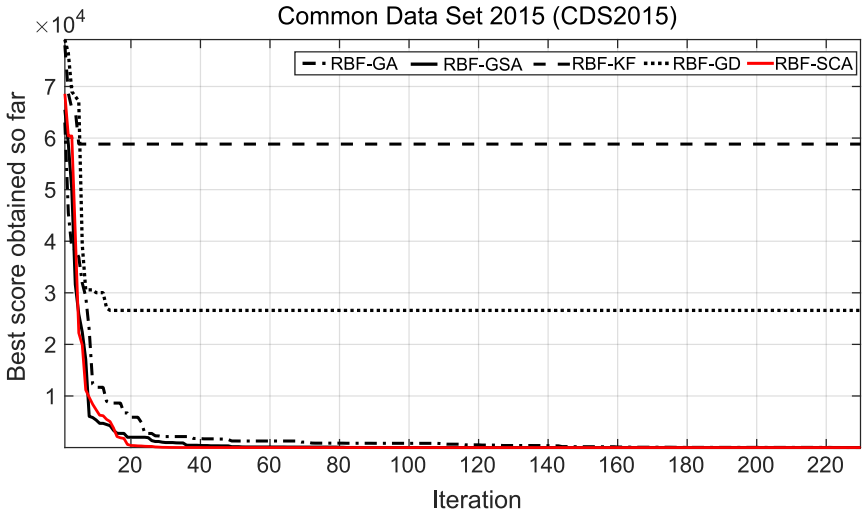

Fig. 10. Convergence curves of various classifiers applied to CDS2015 dataset.

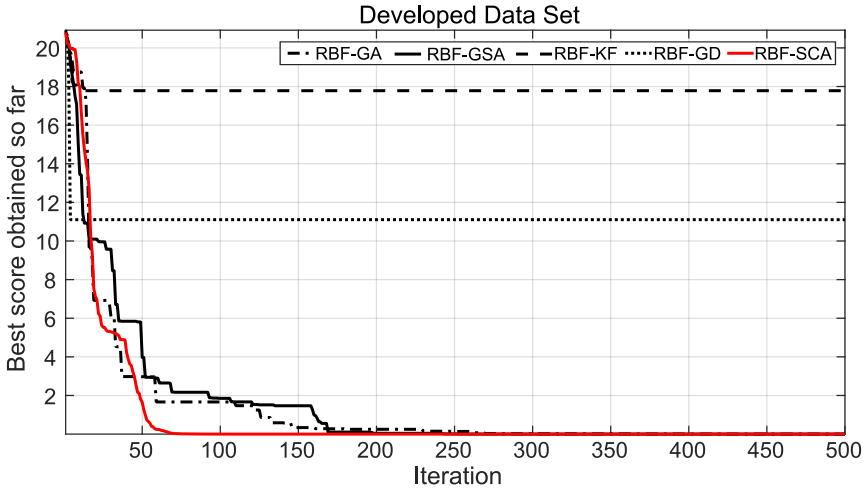

Fig. 11. Convergence curves of various classifiers applied to developed dataset.

algorithms. On the other hand, according to this study, the SCA algorithm performs better in this type of data, due to its high performance in both exploration and exploitation phases. As it has been stated, an algorithm with high efficiency in the exploration stage is required because sonar data covers the entire search area, and the SCA algorithm is much better than other meta-heuristic algorithms in this field.

Table 3. Experimental results for Gorman & Sejnowski dataset.

Classifier	MSE (AVE ± STD)	p-values	Classification rate [%]
RBF-SCA	0.0711 ± 0.0325	N/A	91.0117
RBF-GSA	0.0911 ± 0.1111	0.0145	90.0012
RBF-KF	0.1179 ± 0.0999	1.2119e–03	87.1167
RBF-GA	0.1127 ± 0.1184	0.0039	89.1010
RBF-GD	0.1295 ± 0.1112	1.1108e–20	84.8147

Table 4. Experimental results for CDS2015 dataset.

Classifier	MSE (AVE ± STD)	p-values	Classification rate [%]
RBF-SCA	0.0519 ± 0.0211	N/A	93.2147
RBF-GSA	0.0987 ± 0.0207	1.0019e–07	91.9147
RBF-KF	0.1001 ± 0.0997	0.0014	90.0119
RBF-GA	0.1227 ± 0.1004	0.0497	89.9147
RBF-GD	0.1391 ± 0.1113	0.0466	88.9997

Table 5. Experimental results for developed dataset.

Classifier	MSE (AVE ± STD)	p-values	Classification rate [%]
RBF-SCA	0.0919 ± 0.0012	N/A	91.4568
RBF-GSA	0.1311 ± 0.0098	7.2239e–04	87.8794
RBF-KF	0.3149 ± 0.0587	9.2798e–20	78.1457
RBF-GA	0.2227 ± 0.0458	0.0039	80.9874
RBF-GD	0.1994 ± 0.1202	0.0466	86.9546

4. Conclusion

In this paper, a new meta-heuristic method called the SCA was used to train RBF NNs. To evaluate the designed classifier, two benchmark underwater sonar classification problems were used. Also, an experimental underwater target classification was developed to practically evaluate the merits of the RBF-based classifier in dealing with high-dimensional real word problems. Then, results were compared with the GD, KF, GA, and GSA standard algorithms. Simulation results showed that the SCA algorithm provides better results in terms of convergence rate and classification accuracy compared to the standard algorithms. Due to the complexity of RBF NNs, we can use simpler networks such as multilayer perceptron in future work.

References

1. Abedifar V., Eshghi M., Mirjalili S., Mirjalili S.M. (2013), An optimized virtual network mapping using PSO in cloud computing, *21st Iranian Conference on Electrical Engineering*, pp. 1–6, doi: 10.1109/IranianCEE.2013.6599723.

2. Abu-Mouti F.S., El-Hawary M.E. (2012), Overview of Artificial Bee Colony (ABC) algorithm and its applications, *2012 IEEE International Systems Conference SysCon*, pp. 1–6, doi: 10.1109/SysCon.2012.6189539.

3. Aljarah I., Faris H., Mirjalili S., Al-Madi N. (2016), Training radial basis function networks using biogeography-based optimizer, *Neural Computing and Applications*, **29**(7): 529–553, doi: 10.1007/s00521-016-2559-2.

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, doi: 10.1016/j.neunet.2007.12.036.

5. Chen S., Hong X., Luk B.L., Harris C.J. (2009), Non-linear system identification using particle swarm optimisation tuned radial basis function models, *International Journal of Bio-Inspired Computation*, **1**(4): 246–258, doi:10.1504/IJBIC.2009.024723.

6. Chen S., Wu Y., Luk B.L. (1999), Combined genetic algorithm optimization and regularized orthogonal least squares learning for radial basis function networks, *IEEE Transactions on Neural Networks*, **10**(5): 1239–1243, doi: 10.1109/72.788663.

7. Chun-Tao M., Xiao-Xia L., Li-Yong Z. (2007), Radial basis function neural network based on ant colony optimization, *IEEE International Conference on Computational Intelligence and Security Workshops (CISW 2007)*, pp. 59–62, doi: 10.1109/CISW.2007.4425446.

8. Ding S., Xu L., Su C., Jin F. (2012), An optimizing method of RBF neural network based on genetic algorithm, *Neural Computing and Applications*, **21**(2): 333–336, doi: 10.1007/s00521-011-0702-7.

9. Du K.L., Swamy M.N.S. (2014), Radial basis function networks, [in:] *Neural Networks and Statistical Learning*, Springer Publishing Company, Inc., pp. 299–335, doi: 10.5555/2578631.

10. Dua D., Graff C. (2019), *UCI Machine Learning Repository*, Irvine, CA: University of California, School of Information and Computer Science, Connectionist Bench (sonar, mines vs. rocks), http://archive.ics.uci.edu/ml/datasets/Connectionist+Bench+(Sonar,+Mines+vs.+Rocks).

11. Faris H., Aljarah I., Mirjalili S. (2016), Training feedforward neural networks using multi-verse optimizer for binary classification problems, *Applied Intelligence*, **45**(2): 322–332, doi: 10.1007/s10489-016-0767-1.

12. Faris H., Aljarah I., Mirjalili S., Samui P., Sekhar S., Balas V.E. (2017), Evolving radial basis function networks using moth-flame optimizer, [in:] *Handbook of Neural Computation*, P. Samui, S. Sekhar, V.E. Balas [Eds], Academic Press, pp. 537–550, doi: 10.1016/B978-0-12-811318-9.00028-4.

13. Fasshauer G.E., Zhang J.G. (2007), On choosing "optimal" shape parameters for RBF approximation, *Numerical Algorithms*, **45**(1): 345–368, doi: 10.1007/s11075-007-9072-8.

14. Gan M., Peng H., Dong X.P. (2012), A hybrid algorithm to optimize RBF network architecture and parameters for nonlinear time series prediction, *Applied Mathematical Modeling*, **36**(7): 2911–2919, doi: 10.1016/j.apm.2011.09.066.

15. Gorman R.P., Sejnowski T.J. (1998), Analysis of hidden units in a layered network trained to classify sonar targets, *Neural Networks*, **1**(1): 75–89, doi: 10.1016/0893-6080(88)90023-8.

16. Gutiérrez F.J., Zhao A. (2015), *Common data set 2015*, Kongsberg GeoAcoustics Ltd.

17. 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, doi: 10.1023/A:1021251113462.

18. Horng M.H., Lee Y.X., Lee M.C., Liou R.J. (2012), Firefly metaheuristic algorithm for training the radial basis function network for data classification and disease diagnosis, [in:] *Theory and New Applications of Swarm Intelligence*, R. Parpinelli, H.S. Lopes [Eds], IntechOpen, Rijeka, pp. 115–132, doi: 10.5772/39084.

19. Hyontai S. (2011), Using quick decision tree algorithm to find better RBF networks, [in:] *Intelligent Information and Database Systems. ACIIDS 2011. Lecture Notes in Computer Science*, Nguyen N.T., Kim CG., Janiak A. [Eds], Vol. 6591, Springer, Berlin, Heidelberg, pp. 207–217, doi: 10.1007/978-3-642-20039-7_21.

20. Khishe M., Aghababaee M. (2013), Identifying and controlling sonar clutter by clutter indelible metho, *Iranian Conference on Electrical and Computer Engineering*, pp. 523–529, Sarvestan, Shiraz.

21. Khishe M., Mosavi M. (2017), Active sonar data set, *Mandeley Data*, v1, doi: 10.17632/fyxjjwzphf.1

22. Khishe M., Mosavi M.R., Kaveh M. (2017), Improved migration models of biogeography-based optimization for sonar data set classification using neural network, *Applied Acoustic*, **118**: 15–29, doi: 10.1016/j.apacoust.2016.11.012.

23. Lin C.L., Wang J., Chen C.Y., Chen C.W., Yen C. (2009), Improving the generalization performance of RBF neural networks using a linear regression technique, *Expert Systems with Applications*, **36**(10): 12049–12053, doi: 10.1016/j.apacoust.2016.11.012.

24. Mirjalili S. (2016), SCA: a Sine Cosine Algorithm for solving optimization problems, *Knowledge-Based Systems*, **96**: 120–133, doi: 10.1016/j.knosys.2015.12.022.

25. Mirjalili S., Hashim S.Z.M., Sardroudi H.M. (2012), Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm, *Applied Mathematics and Computation*, **218**(22): 11125–11137, doi: 10.1016/j.amc.2012.04.069.

26. Mirjalili S., Mirjalili S.M., Lewis A. (2014), Let a biogeography-based optimizer train your multilayer perceptron, *Journal of Information Sciences*, **269**: 188–209, doi: 10.1016/j.ins.2014.01.038.

27. Mosavi M.R., Khishe M. (2017), Training a feedforward neural network using particle swarm optimizer with autonomous groups for sonar target classification, *Journal of Circuits, Systems, and Computers (JCSC)*, **26**(11): 1750185:1-1750185:20, doi: 10.1142/S0218126617501857.

28. Mosavi M.R., Khishe M., Moridi A. (2016), Classification of sonar target using hybrid particle swarm and gravitational search, *Marine Technology*, **3**(1): 1–13, https://www.sid.ir/en/journal/ViewPaper.aspx?id=532112.

29. Neruda R., Kudová P. (2005), Learning methods for radial basis function networks, *Future Generation Computer Systems*, **21**(7): 1131–1142, doi: 10.1016/j.future.2004.03.013.

30. Nguyen L.S., Frauendorfer D., Mast M.S., Gatica-Perez D. (2014), Hire me: Computational inference of hirability in employment interviews based on nonverbal behavior, *IEEE Transactions on Multimedia*, **16**(4): 1018–1031, doi: 10.1109/TMM.2014.2307169.

31. Park J., Sandberg I.W. (1993), Approximation and radial-basis-function networks, *Neural Computation*, **5**(2): 305–316, doi: 10.1162/neco.1993.5.2.305.

32. Preston M. (2004), *Resampling sonar echo time series primarily for seabed sediment*, United State Patent, US 6,801,474 B2.

33. Vogt M. (1993), Combination of radial basis function neural networks with optimized learning vector quantization, *IEEE International Conference on Neural Networks*, San Francisco, CA, USA, Vol. 3, pp. 1841–1846, doi: 10.1109/ICNN.1993.298837.

34. Wu D. *et al.* (2010), Prediction of Parkinson's disease tremor onset using a radial basis function neural network based on particle swarm optimization, *International Journal of Neural Systems*, **20**(2): 109–116, doi: 10.1142/S0129065710002292.

35. Yang X.S. (2014), *Nature-Inspired Optimization Algorithms*, Elsevier.

36. Yu B., He X. (2006), Training radial basis function networks with differential evolution, *Proceedings of IEEE International Conference on Granular Computing*, pp. 369–372.

37. Yu H., Xie T., Paszczyński S., Wilamowski B.M. (2011), Advantages of radial basis function networks for dynamic system design, *IEEE Transactions on Industrial Electronics*, **58**(12): 5438–5450, doi: 10.1109/TIE.2011.2164773.

38. Yu-Qing S., Jun-Fei Q., Hong-Gui H. (2016), Structure design for RBF neural network based on improved K-means algorithm, *Chinese Control and Decision Conference (CCDC),* Yinchuan, pp. 7035–7040, doi: 10.1109/CCDC.2016.7532265.

39. Zhang D. *et al.* (2014), A new optimized GA-RBF neural network algorithm, *Computational Intelligence and Neuroscience*, **2014**: article ID 982045, 6 pages, doi: 10.1155/2014/982045.

40. Zhong Y., Huang X., Meng P., Li F. (2014), PSO-RBF neural network PID control algorithm of electric gas pressure regulator, *Abstract and Applied Analysis*, **2014**: article ID 731368, 7 pages, doi: 10.1155/2014/731368.