Dynamic Adaptive Search Strategy Based Incremental Extreme Learning Machine Based on

Zuozhi Liu[1(✉)], Jianjun Jiao[1], and Quan Yuan[2]

[1] School of Mathematics and Statistics, Guizhou University of Finance and Economics, Guiyang, Guizhou, People's Republic of China
[2] Finance Department, Guizhou University of Finance and Economics, Guiyang, Guizhou, People's Republic of China

Abstract. Extreme learning machine (ELM) is a promising method for the learning of single-hidden layer feedforward network (SLFN) which is attractive for its simplicity and high efficiency. However, during the rapid development of ELM algorithm, the determination of suitable network architecture is still a challenging work. To deal with this issue, this work develops a modified ELM algorithm based on a novel adaptive optimization method. Specifically, we use the growth structure strategy to design the network architecture. During the learning process of the proposed algorithm, the grey wolf optimization (GWO) technique is then introduced to seek the optimal parameters for hidden nodes instead of random selection. In addition, to improve the convergence speed, we further ameliorate the traditional GWO approach. Experiment results over some benchmark applications indicate that our AI-ELM algorithm can dramatically reduce the scale of network and obtains the better generalization performance than other classical ELM algorithms.

Keywords: Single-hidden layer feedforward network · Incremental extreme learning machine · Enhanced grey wolf optimization · Universal approximation

1 Introduction

Artificial neural network (ANN) is one of the important research branches of machine learning and computational intelligence. Simple speaking, the purpose of this bionic network is to simulate the information processing function of the brain nervous system. In the past decades, ANN has been widely studied and has become a powerful information processing technique because of its self-learning, self-adapting, and fault-tolerant. Single-hidden-layer feedforward

This work was supported by the Fund of Guizhou University of Finance and Economics (No. 2019XYB02) and the Science and Technology Project of Guizhou Province (No. [2020]1Y253).

H. Song and D. Jiang (Eds.): SIMUtools 2020, LNICST 369, pp. 562–571, 2021.
https://doi.org/10.1007/978-3-030-72792-5_44

network (SLFN) is a special kind of ANN which only contains a single hidden layer. Due to its simplicity and good generalization performance (GP), SLFN has attracted great interest and has been applied to various fields ranging from scientific research to engineering application [1,2]. However, the most of existing learning algorithms for SLFN are usually designed based on gradient optimization strategy, which are proven to be complex and slow in computation. As a result, it is urgent to develop a learning algorithm which can satisfy the need of the rapid development of SLFN. To this end, Huang et al. developed a novel learning algorithm for SLFN called extreme learning machines (ELM) [3]. Generally speaking, the key idea of ELM is to take advantage of the random mechanism to select the parameters of hidden nodes and use the least-mean square method to resolve the output weights. Compared with traditional gradient-based learning algorithms, ELM not only learns faster but also shows better GP, especially in some practical applications [4,5].

Although ELM learns faster than the traditional learning algorithms, the scale of network architecture (NA) is still determined by trial and error. Therefore, how to build a suitable NA is a prerequisite for the successful application of ELM. To deal with this problem, various construction methods were currently proposed with their own advantages and drawbacks. Among these methods, incremental ELMs are the most famous and representative. I-ELM uses the incremental mechanism to obtain the optimal NA. In this algorithm, the hidden nodes are recruited one by one with randomly selected parameters [6]. In order to improve the convergence rate of I-ELM, CI-ELM referred in [7] proposes to constantly update the output weights of the added hidden nodes rather than unchanged. In addition, EI-ELM was developed to reduce the network complexity of I-ELM. Different from I-ELM, EI-ELM takes several randomly hidden nodes into consideration at each learning step and only recruits the one with the maximum error decreasing into the network [8]. In addition, a large number of experiments have shown that these algorithms can reduce the network scale to a great extent. However, during the learning procedure of all the above-mentioned algorithms, the parameters of hidden nodes are randomly assigned. Although this random mechanism can simplify the learning procedure, it also brings some redundant hidden nodes into the network, which will increase the network scale.

In order to obtain a more compact NA for specific problem, we propose a new ELM algorithm based on a novel adaptive optimization method (referred as AI-ELM). Similar to I-ELM, in this work, we take advantage of the growth structure strategy to design the network architecture. In other words, we recruit one hidden node into network at each learning step. Differently, during the learning process of AI-ELM, a novel adaptive optimization method is introduced to seek the optimal parameters for hidden nodes instead of random selection. Considering the simplicity and great search capacity, the hot grey wolf optimization (GWO) method is taken as the optimizer. In addition, to improve the convergence speed, we further ameliorate the traditional GWO algorithm. Finally, we compare AI-ELM with the existing algorithms on some benchmark problems.

The corresponding experimental results indicate that our AI-ELM can obtain a more compact NA with the better GP.

2 Brief on I-ELM and GWO

2.1 I-ELM Algorithm

In terms of structure, SLFN is composed of three main parts: input layer, hidden layer and output layer. Mathematically, the SLFN as shown in Fig. 1 can be represented as

$$f_l(\mathbf{x}) = \sum_{i=1}^{l} \beta_i g_i(\mathbf{x}) = \sum_{i=1}^{l} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}) \tag{1}$$

where $\mathbf{x} \in R^d$ is the input sample, l is the number of the hidden nodes, $(\mathbf{a}_i, b_i)$ are the parameters of the ith hidden node, β_i is the ith output weight, and $g_i(\mathbf{x})$ or $G(\mathbf{a}_i, b_i, \mathbf{x})$ denotes the output of the ith hidden node [6].

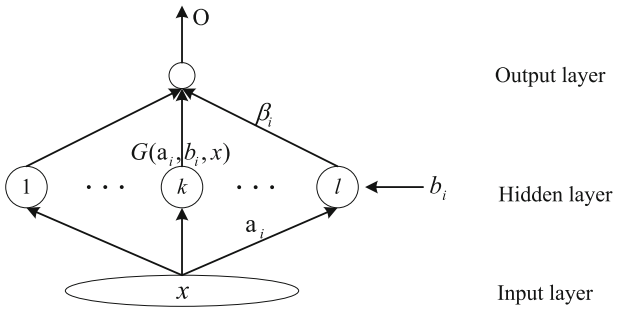

Fig. 1. Single-hidden layer feedforward network architecture.

I-ELM is an incremental constructive approach where the hidden nodes are recruited into network one after one with randomly selected parameters. Specifically, at each learning step of I-ELM, only one node is added to the hidden layer of the current network. The parameters of new added nodes are selected in a fixed interval at random. When lth hidden node is added to the network, the output of I-ELM can be expressed as

$$f_l(\mathbf{x}) = f_{l-1}(\mathbf{x}) + \beta_l g_l(\mathbf{x}). \tag{2}$$

It is worth noting that although I-ELM adopts random mechanism to seek the parameters of hidden nodes, it has been proven that I-ELM can accurately approximate the objective function. That is

Theorem 1. *Given a SLFN with any nonconstant piecewise continuous function* $g : R \to R$, *if* $span\{G(\mathbf{a}, b, \mathbf{x}) : (\mathbf{a}, b) \in R^d \times R\}$ *is dense in* L^2, *then for any continuous target function* f *and any randomly generated function sequence* $\{g_l(\mathbf{x}) = G(\mathbf{a}_l, b_l, \mathbf{x})\}$, $\lim_{l\to\infty} \parallel f - f_{l-1} - \beta_l g_l \parallel = 0$ *holds with probability one if*

$$\beta_l = \frac{< e_{l-1}, g_l >}{\parallel g_l \parallel^2}, \tag{3}$$

where $e_{l-1} = f - f_{l-1}$.

2.2 GWO Algorithm

Grey wolf optimization (GWO) is an emerging intelligent optimization algorithm developed by Mirjalili et al. in 2014 [9]. Due to its great search capacity, GWO has attracted increasing interest and has been successfully used to solve many real-world problems [10]. In addition, this technique only has few uncertain parameters to be adjusted and can be easy to implement.

The core idea of GWO algorithm can be generally summarized as two parts: social pyramid and hunting strategy. Firstly, grey wolves are grouped into four types, namely $\alpha, \beta, \delta, \omega$, where α, β, δ denote three levels of solutions from good to inferior, and ω is the solutions except for α, β, δ. As shown in Fig. 2, four types of grey wolves constitute a strict social hierarchy.

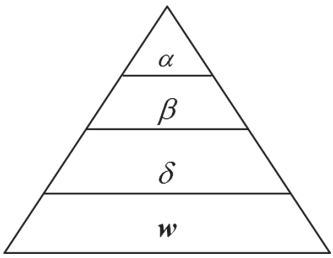

Fig. 2. Social pyramid of grey wolves.

Figure 3 gives an intuitive illustration of how grey wolves hunt the prey. In general, the hunting strategy consists of the following two steps:

(1) *Encircling prey.* Before the hunting, grey wolves firstly move towards the prey and surround it. Mathematically, this phase can be modeled as

$$\mathbf{D} = |\mathbf{C}\mathbf{x}_p(t) - \mathbf{x}(t)|, \tag{3}$$

$$\mathbf{x}(t+1) = \mathbf{x}_p(t) - \mathbf{A}\mathbf{D}, \tag{4}$$

$$\mathbf{A} = 2\mathbf{a}\mathbf{r}_1 - \mathbf{a}, \tag{5}$$

$$\mathbf{C} = 2\mathbf{r}_2. \tag{6}$$

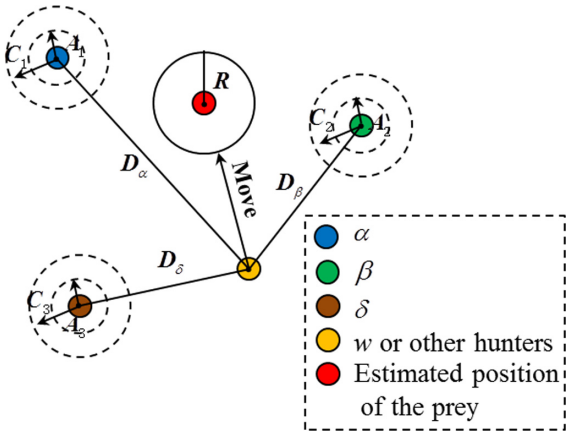

Fig. 3. Hunting strategy of grey wolves.

where $\mathbf{r}_1, \mathbf{r}_2$ are two random vectors, $\mathbf{a} = 2 - 2t/Iter_{\max}$, and $\mathbf{x}_p(t), \mathbf{x}(t)$ denote the current locations of prey and wolves, respectively.

(2) *Hunting prey.* During the hunting phase, grey wolves α usually conducts β, δ to attack the prey. Meanwhile, the rest of grey wolves ω adjust their locations according to α, β and δ. This phase can be expressed as follows,

$$\mathbf{D}_\alpha = |\mathbf{C}_1\mathbf{x}_\alpha - \mathbf{x}|, \tag{7}$$

$$\mathbf{D}_\beta = |\mathbf{C}_2\mathbf{x}_\beta - \mathbf{x}|, \tag{8}$$

$$\mathbf{D}_\delta = |\mathbf{C}_3\mathbf{x}_\delta - \mathbf{x}|, \tag{9}$$

$$\mathbf{x}_1 = \mathbf{x}_\alpha - \mathbf{A}_1\mathbf{D}_\alpha, \tag{10}$$

$$\mathbf{x}_2 = \mathbf{x}_\beta - \mathbf{A}_2\mathbf{D}_\beta, \tag{11}$$

$$\mathbf{x}_3 = \mathbf{x}_\delta - \mathbf{A}_3\mathbf{D}_\delta, \tag{12}$$

$$\mathbf{x}(t+1) = \frac{\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3}{3} \tag{13}$$

where $\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3, \mathbf{A}_1, \mathbf{A}_2$, and $\mathbf{A}_3$ are six random vectors, $\mathbf{x}_\alpha, \mathbf{x}_\beta, \mathbf{x}_\delta$ denote the locations of α, β and δ, respectively.

3 Proposed AI-ELM

As described earlier, the original I-ELM adopts random mechanism to seek the parameters for hidden nodes. Although this random mechanism can simplify learning process, it also brings some unnecessary hidden nodes into network. In other words, some randomly added hidden nodes may play a small part in the whole network. It means that I-ELM usually needs more hidden nodes to approximate target function.

In order to achieve a more compact network, this work proposes an improved I-ELM based on a novel adaptive optimization method, named adaptive incremental extreme learning machine (AI-ELM). Similar to traditional I-ELM, in this work, the growth structure strategy is also employed to design the network architecture. During the learning process, we recruit hidden nodes into current network one after another. Differently, during the learning process of AI-ELM, we introduce a novel adaptive optimization method to seek the optimal parameters for hidden nodes instead of random selection. Considering the simplicity and great search capacity, the hot grey wolf optimization (GWO) method is taken as optimizer. The clear schematic of our AI-ELM is shown in Fig. 4.

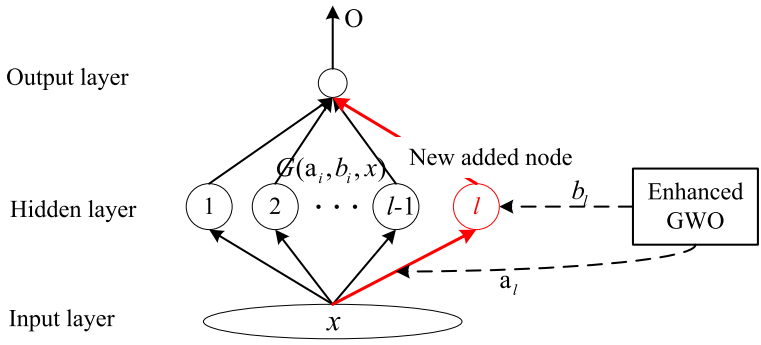

Fig. 4. Block diagram of the proposed AI-ELM.

However, researches have demonstrated that similar to other optimization algorithms, the traditional GWO may fall into local minima in practical applications. To address this issue, we make the following two improvements to ameliorate the traditional GWO algorithm. Firstly, we further modify the convergence factor **a** to enhance the nonlinear dynamic capability of it, that is

$$\mathbf{a} = \frac{2}{1 + e^{\gamma(\frac{2t}{Iter_{\max}} - 1)}}, \tag{14}$$

where γ is a generalized coefficient that is used to restrict **a** to the interval $[0, 2]$. Here, we set $\gamma = 8$. With the help of the modified convergence factor, it can effectively balance the global search and local exploration.

Considering the fact that the traditional GWO may fall into local minima, we introduce the dynamic learning rates to update the position information. That is

$$\mathbf{x}(t+1) = \begin{cases} \frac{\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3}{3} & |\frac{\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3}{3}| \leq |\frac{w_1\mathbf{x}_1 + w_2\mathbf{x}_2 + w_3\mathbf{x}_3}{3}| \\ \frac{w_1\mathbf{x}_1 + w_2\mathbf{x}_2 + w_3\mathbf{x}_3}{3} & \text{other} \end{cases} \tag{15}$$

where

$$w_1 = \frac{|\mathbf{x}_1|}{|\mathbf{x}_1| + |\mathbf{x}_2| + |\mathbf{x}_3|}, \tag{16}$$

$$w_2 = \frac{|\mathbf{x}_2|}{|\mathbf{x}_1| + |\mathbf{x}_2| + |\mathbf{x}_3|}, \tag{17}$$

$$w_3 = \frac{|\mathbf{x}_3|}{|\mathbf{x}_1| + |\mathbf{x}_2| + |\mathbf{x}_3|}. \tag{18}$$

By doing so, the search ability of GWO algorithm can be further enhanced so as to avoid falling into local minima.

Note that in the proposed AI-ELM, the position of each grey wolf is composed of input weight and hidden bias, and residual error is adopted as the fitness function. The details of AI-ELM algorithm are presented as follows.

AI-ELM Algorithm:

1. **Initialization:** Suppose that $l = 0$ and target error $E = t$;
2. **Learning step:**
   While $l < l_{\max}$ and $\| E \| > \epsilon$
   (a) Add one hidden node to network $l = l + 1$;
   (b) Seek the optimal parameters $(\mathbf{a}_i, b_i)$ for the lth hidden node by means of the enhanced GWO algorithm;
   (c) Compute the output weight β_l :

$$\beta_l = \frac{E \cdot H_l^T}{H_l \cdot H_l^T}; \tag{19}$$

   (d) Update the target error E when the lth hidden node is added:

$$E = E - \beta_l \cdot H_l. \tag{20}$$

   Endwhile

As stated above, each hidden node recruited by the proposed AI-ELM plays an important part in the whole network, which means that AI-ELM needs less hidden nodes to complete the learning task. Therefore, it can be concluded that AI-ELM achieves a better NA than the traditional I-ELM in real applications.

4 Experimental Verification

In this section, we test our method in terms of NA and GP. Specifically, the proposed AI-ELM is compared with three classical ELM algorithms (I-ELM [6], EI-ELM [8], and D-ELM [11]) on some benchmark problems. Table 1 gives a detailed description of these eight UCI benchmark problems. For the fairness, all the tests are conducted in the same environment.

Table 1. Specification of regression problems.

Data sets	Attributes	Cases	Training data	Testing data
Abalone	8	4177	2000	2177
Boston Housing	13	506	250	256
California Housing	8	20640	8000	12640
Census (House8L)	8	22784	10000	12784
Delta Ailerons	6	7129	3000	4129
Delta Elevators	6	9517	4000	5517
Japanese Vowels	12	9961	4275	5686
Kin8nm	9	8192	4000	4192

In order to compare the NA and GP of each algorithm, for each problem, we first set the same expected accuracy (stopping RMSE) for all four algorithms, and the specification of expected accuracy is presented in Tables 2 and 3. In addition, the maximum learning step is set to 200 for all eight problems. Note that the less hidden nodes a method needs, the more compact NA it has. Meanwhile, the smaller the error a method achieves, the better GP it has. Therefore, a better algorithm must has less hidden nodes and residual error at the same time. In order to reduce the instability of the results, for each algorithm, the average result over fifty trails are taken as the final comparison results. Note that the better results in tables are emphasized in bold to facilitate the intuitive analysis.

The performance of four ELM algorithms on eight real-world problems is shown in Tables 2, 3, 4, and 5. Tables 2 and 3 show the NA comparison of four different algorithms with sigmoid nodes and sine nodes, respectively. It can be seen that the number of hidden nodes needed by the proposed AI-ELM is much less than those of other three algorithms for all cases. In view of this fact, we

Table 2. NA comparison of different algorithms with sigmoid nodes.

Datasets	Stop	I-ELM		EI-ELM		D-ELM		AI-ELM	
	RMSE	Nodes	Dev	Nodes	Dev	Nodes	Dev	Nodes	Dev
Abalone	0.09	177.78	52.84	66.80	40.65	4.75	1.41	**3.82**	0.61
Boston Housing	0.11	183.48	30.14	68.12	41.57	11.84	3.50	**8.44**	2.62
California Housing	0.16	197.98	12.60	48.96	14.52	5.74	1.37	**3.66**	1.12
Census (House8L)	0.09	194.34	21.04	48.12	19.04	5.80	1.34	**4.52**	0.94
Delta Ailerons	0.05	182.08	35.12	26.16	17.21	4.02	0.88	**3.12**	0.08
Delta Elevators	0.06	189.72	21.43	30.88	11.10	5.26	0.96	**3.36**	0.07
Japanese Vowels	0.11	199.10	6.36	74.94	31.10	7.52	1.31	**5.14**	1.92
Kin8nm	0.14	194.16	18.88	137.96	66.16	12.34	3.53	**8.68**	3.13

can find that our AI-ELM achieves a more compact NA. Tables 4 and 5 show the GP comparison of four different ELM algorithms. Apparently, AI-ELM performs better than the other three algorithms. Therefore, it can be concluded that the proposed AI-ELM achieves a more compact NA with better GP.

Table 3. NA comparison of different algorithms with sine nodes.

Datasets	Stop	I-ELM		EI-ELM		D-ELM		AI-ELM	
	RMSE	Nodes	Dev	Nodes	Dev	Nodes	Dev	Nodes	Dev
Abalone	0.09	139.88	59.65	20.54	12.97	6.20	1.50	**4.24**	0.19
Boston Housing	0.11	189.68	26.75	53.58	17.58	21.56	3.45	**13.44**	1.52
California Housing	0.16	111.24	55.60	18.04	13.41	6.74	1.42	**3.92**	1.32
Census (House8L)	0.09	78.44	24.30	13.80	4.33	6.70	1.65	**4.56**	0.25
Delta Ailerons	0.05	148.22	63.90	53.42	72.23	5.80	1.12	**3.40**	0.07
Delta Elevators	0.06	194.84	25.56	134.38	69.80	8.68	1.50	**5.10**	0.07
Japanese Vowels	0.11	198.46	10.89	89.24	55.09	11.46	2.05	**7.86**	0.32
Kin8nm	0.14	200.00	0.00	100.38	22.94	26.68	5.15	**16.76**	0.78

Table 4. GP comparison of different algorithms with sigmoid nodes.

Datasets	I-ELM		EI-ELM		D-ELM		AI-ELM	
	Mean	Dev	Mean	Dev	Mean	Dev	Mean	Dev
Abalone	0.0943	0.0045	0.0903	0.0032	0.0873	0.0043	**0.0851**	0.0020
Boston Housing	0.1256	0.0119	0.1200	0.0131	0.1170	0.0115	**0.1142**	0.0124
California Housing	0.1711	0.0081	0.1607	0.0022	**0.1556**	0.0043	0.1564	0.0027
Census (House8L)	0.0927	0.0024	0.0901	0.0030	0.0883	0.0035	**0.0877**	0.0016
Delta Ailerons	0.0527	0.0042	0.0492	0.0019	0.0462	0.0029	**0.0411**	0.0006
Delta Elevators	0.0641	0.0054	0.0596	0.0011	0.0572	0.0020	**0.0543**	0.0005
Japanese Vowels	0.1234	0.0089	0.1102	0.0013	0.1158	0.0044	**0.1118**	0.0010
Kin8nm	0.1445	0.0027	0.1413	0.0023	0.1398	0.0023	0.1396	0.0026

Table 5. GP comparison of different algorithms with sine nodes.

Datasets	I-ELM		EI-ELM		D-ELM		AI-ELM	
	Mean	Dev	Mean	Dev	Mean	Dev	Mean	Dev
Abalone	0.0915	0.0040	0.0898	0.0032	0.0888	0.0034	**0.0846**	0.0031
Boston Housing	0.1435	0.0146	0.1360	0.0183	0.1302	0.0150	**0.1228**	0.0124
California Housing	0.1553	0.0040	0.1542	0.0028	0.1547	0.0039	**0.1512**	0.0034
Census (House8L)	0.0896	0.0029	0.0892	0.0027	0.0882	0.0032	**0.0858**	0.0018
Delta Ailerons	0.0528	0.0045	0.0498	0.0024	0.0466	0.0028	**0.0414**	0.0012
Delta Elevators	0.0727	0.0115	0.0634	0.0046	0.0587	0.0017	**0.0554**	0.0015
Japanese Vowels	0.1301	0.0140	0.1108	0.0016	0.1079	0.0030	**0.1038**	0.0036
Kin8nm	0.1862	0.0159	0.1418	0.0025	0.1408	0.0029	**0.1377**	0.0029

5 Conclusion

In this work, an modified I-ELM called AI-ELM is developed, which automatically determines the optimal network architecture. In AI-ELM, the hidden nodes are recruited one after one, but the parameters of each hidden node are sought based on the enhanced GWO algorithm instead of random selection. Compared with the other ELMs, AI-ELM achieves a more compact NA with better GP. This fact has been further verified by a performance comparison on some real-world applications.

References

1. Bishop, C.-M.: Neural Networks for Pattern Recognition. Oxford University Press, New York (1995)
2. Li, T.-S., et al.: A spintronic memristor-based neural network with radial basis function for robotic manipulator control implementation. IEEE Trans. Syst. Man Cybern. Syst. **46**(4), 582–588 (2017)
3. Huang, G.-B., et al.: Extreme learning machine: theory and applications. Neurocomputing **70**(1–3), 489–501 (2006)
4. Duan, M., et al.: A parallel multiclassification algorithm for big data using an extreme learning machine. IEEE Trans. Neural Netw. Learn. Syst. **29**(6), 2337–2351 (2018)
5. Chen, K., Laghrouche, S., Djerdir, A.: Proton exchange membrane fuel cell prognostics using genetic algorithm and extreme learning machine. Fuel Cells **20**(3), 263–271 (2020)
6. Huang, G.-B., Chen, L., Siew, C.-K.: Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans. Neural Netw. **17**(4), 879–892 (2006)
7. Huang, G.-B., Chen, L.: Convex incremental extreme learning machine. Neurocomputing **70**(16), 3056–3062 (2007)
8. Huang, G.-B., Chen, L.: Enhanced random search based incremental extreme learning machine. Neurocomputing **71**(16), 460–468 (2008)
9. Mirjalili, S., et al.: Grey wolf optimizer. Adv. Eng. Soft. **69**, 46–61 (2014)
10. Martin, B., Marot, J., Bourennane, S.: Mixed grey wolf optimizer for the joint denoising and unmixing of multispectral images. Appl. Soft. Comput. **74**, 385–410 (2019)
11. Zhang, R., et al.: Dynamic extreme learning machine and its approximation capability. IEEE Trans. Cybern. **43**(6), 2054–2065 (2013)