Physics Letters A 432 (2022) 128008

Contents lists available at ScienceDirect

Physics Letters A

www.elsevier.com/locate/pla

Training threshold neural networks by extreme learning machine and adaptive stochastic resonance

Zejia Chen [a], Fabing Duan [a,*], François Chapeau-Blondeau [b], Derek Abbott [c]

[a] Institute of Complexity Science, Qingdao University, Qingdao 266071, PR China
[b] Laboratoire Angevin de Recherche en Ingénierie des Systèmes (LARIS), Université d'Angers, 62 avenue Notre Dame du Lac, 49000 Angers, France
[c] Centre for Biomedical Engineering (CBME) and School of Electrical & Electronic Engineering, The University of Adelaide, Adelaide, SA 5005, Australia

ARTICLE INFO

Article history:
Received 4 December 2021
Received in revised form 12 February 2022
Accepted 14 February 2022
Available online 21 February 2022
Communicated by M. Perc

Keywords:
Threshold network
Adaptive stochastic resonance
Extreme learning machine
Noise injection
Function approximation
Classification

ABSTRACT

Threshold neural networks are highly useful in engineering applications due to their ease of hardware implementation and low computational complexity. However, such threshold networks have non-differentiable activation functions and therefore cannot be trained by standard gradient-based algorithms. To circumvent this limitation, here we propose a hybrid training algorithm for threshold neural networks. The proposed hybrid training algorithm has two distinguishing features: the structural transformation of the hidden layer enables threshold networks to benefit from a noise-boosted learning capability via adaptive stochastic resonance (ASR), and by using the fast learning algorithm of the extreme learning machine (ELM) suitable generalization performance ensues for the threshold networks. Experimental results on regression and on multiclass classification demonstrate the realizability and practical efficiency of the proposed hybrid training algorithm, thereby demonstrating the beneficial role of artificial noise injection in threshold neural networks.

1. Introduction

Due to low-consumption and the ease of hardware implementation, the neural network with threshold activation functions, referred to as a threshold network, is highly useful in practical engineering problems [1–10]. For instance, a number of multi-threshold quantizers are usually deployed over a sensing field to compose a wireless sensor network in distributed estimation problems, whereby the analog observation is compressed into low-precision bit information in the sense of the occupied bandwidth [11–14]. However, for training threshold networks, the commonly used backpropagation algorithm of artificial neural networks is not directly applicable as the threshold activation functions are non-differentiable or with zero gradients [1–10]. Thus, alternative methods to train threshold networks have attracted the attention of researchers.

In order to train a threshold network using the gradient-descent based backpropagation algorithm, a series of studies have been carried out by approximating the threshold neuron to a smoothed activation function [1,3,4], randomizing network weights with a smooth distribution [2], computing gradients of real-valued neurons in backpropagation but performing forward propagation with binary neurons [6,7], etc. Recently, it is interesting to note that noise injection has become a useful alternative strategy in updating the weights of such threshold networks [6,8–11,15–17]. By injecting artificial noise into the saturated regime of the activation function [6] or smoothing the input-output characteristic of hard-limiting neurons with an ensemble of mutually independent noise components [8–11,15], enabled a proper definition of the gradients in non-differentiable threshold networks. Then, the gradient-based backpropagation algorithm can successfully perform in finely optimizing threshold networks in the training phase. Of special interest is the noise-modulated threshold network proposed by Ikemoto et al. [8] as an application of the suprathreshold stochastic resonance mechanism [18–21,27,34], wherein an optimal amount of deliberately added noise that facilitates threshold network learning is demonstrated to be non-zero and the beneficial role of injected noise to the generalization of the threshold network is manifested.

Beyond exploring the gradient-descent based learning algorithm for training threshold networks [1–4,6–10,15], another interesting learning method for single hidden layer feedforward neural networks named the extreme learning machine (ELM) [5,8,22,23] can be directly applied to train threshold networks with good generalization performance [5]. Due to the binary output of the threshold neuron, the hidden layer output matrix is very likely non-full column rank. This unstable shortcoming of the ELM algorithm can be resettled by introducing a regularization term of

* Corresponding author.
E-mail address: fabing.duan@hotmail.com (F. Duan).

https://doi.org/10.1016/j.physleta.2022.128008
0375-9601/

the norm of weights [22,23]. However, it is noted that the threshold network trained by the ELM algorithm requires a large number of threshold activation functions for attaining improved generalization performance [5]. Moreover, the required large number of threshold activation functions brings an equivalent order of the number of weight coefficients, and the overfitting of the threshold network may occur. Especially, both the gradient-descent based learning algorithms and the ELM approach have been implemented in the noise-modulated threshold network, and the latter is demonstrated to provide improved generalization performance at an extremely fast learning speed [8]. However, the determination of the optimal noise level is manually but not adaptively searched, the adaptive learning ability of the noise-boosted neuron is not fully appreciated.

In this paper, we propose a hybrid method for training threshold networks by combining the fast learning feature of the ELM [22,23] and the noise-boosted learning capability of adaptive stochastic resonance (ASR) [24,25]. More specifically, by injecting an ensemble of noise components into threshold activation functions, the noise-modulated threshold network is endowed with a non-zero gradient for the gradient-descent updating rule in the training phase. Interestingly, the noise level treated as a network parameter can be adaptively searched, and the converged non-zero noise level that results clearly confirms the learning capability of ASR for training and testing threshold networks in practical applications. In particular, in the training phase, the input weights of the noise-modulated threshold network are randomly chosen and the output weights are determined as the least-squares solution by the ELM algorithm. Thus, this novel hybrid method not only finely optimizes weights in an extremely fast way, but also can adaptively search the optimal non-zero noise level that achieves the beneficial role of noise injection in the designed threshold networks. Compared with implementations of straight substituting the sigmoid function with a large gain parameter for the threshold function [3] and directly training the threshold networks by the ELM algorithm [5], the regression and classification results demonstrate improved generalization performance of the threshold network achieved by the proposed hybrid method. These comparison results also suggest an interesting strategy for training neural networks with a much wider family of non-differentiable or discrete-valued activation functions in real-world applications.

2. Results of the hybrid training algorithm for threshold networks

Let us consider a feedforward neural network with its size $N \times K \times M$, where a $N \times 1$ dimensional data vector $\boldsymbol{x}$ received in the input layer is applied to K neurons in the hidden layer, and then mapped into a $M \times 1$ dimensional output vector $\boldsymbol{y}$ in the output layer. The $K \times N$ weight matrix $\boldsymbol{W}$ connects the hidden neurons to the input vector $\boldsymbol{x}$, the $K \times 1$ bias vector $\boldsymbol{b}$ contains K biases b_k for hidden neurons, and the $K \times M$ weight matrix $\boldsymbol{U}$ connects the output layer to the hidden one.

2.1. Motivated example

We here focus on the k-th hidden neuron in the hidden layer as the threshold activation function [8–10,15,18,19] or the McCulloch-Pitts neuron [26] given by

$$\phi_k(u) = \begin{cases} 1, & u \geq \theta_k, \\ 0, & u < \theta_k, \end{cases} \tag{1}$$

with the threshold parameter θ_k for $k = 1, 2, \cdots, K$. For P examples of the training set $\{\boldsymbol{x}^{(p)}, \boldsymbol{t}^{(p)}\}_{p=1}^{P}$ and the linear activation functions in the output layer, the threshold network yields P output vectors

$$\boldsymbol{y}^{(p)} = \boldsymbol{h}^{(p)}\boldsymbol{U}, \tag{2}$$

for $p = 1, 2, \cdots, P$. For the p-th example of $\{\boldsymbol{x}^{(p)}, \boldsymbol{t}^{(p)}\}_{p=1}^{P}$, the corresponding $1 \times K$ dimensional output vector of the hidden layer can be written as

$$\boldsymbol{h}^{(p)} = [\phi_1(v_1), \phi_2(v_2), \cdots, \phi_K(v_K)], \tag{3}$$

where $v_k = [\boldsymbol{W}]^{(k)}\boldsymbol{x}^{(p)} + b_k$ is the local field of k-th neuron and $[\boldsymbol{W}]^{(k)}$ denotes the k-th row of the input weight matrix $\boldsymbol{W}$. The ELM algorithm [5] can be directly applied to the threshold network by assigning an arbitrary input weight matrix $\boldsymbol{W}$ and an arbitrary bias vector $\boldsymbol{b}$ to calculate the hidden layer output matrix

$$\boldsymbol{H} = [\boldsymbol{h}^{(1)\top}, \boldsymbol{h}^{(2)\top}, \cdots, \boldsymbol{h}^{(P)\top}]^{\top}. \tag{4}$$

Since the bias b_k plays an equivalent role of the threshold parameter θ_k for the k-th threshold neuron, here θ_k can be set to zero for simplicity. Letting the network output matrix $\boldsymbol{Y} = [\boldsymbol{y}^{(1)\top}, \boldsymbol{y}^{(2)\top}, \cdots, \boldsymbol{y}^{(P)\top}]^{\top} = \boldsymbol{H}\boldsymbol{U}$ and the matrix $\boldsymbol{T} = [\boldsymbol{t}^{(1)\top}, \boldsymbol{t}^{(2)\top}, \cdots, \boldsymbol{t}^{(P)\top}]^{\top}$, the ELM algorithm finds a least-squares solution of the output weight matrix as

$$\boldsymbol{U} = \boldsymbol{H}^{\dagger}\boldsymbol{T} \tag{5}$$

to minimize the cost function of the error energy

$$\mathcal{J} = \|\boldsymbol{Y} - \boldsymbol{T}\|^2 = \|\boldsymbol{H}\boldsymbol{U} - \boldsymbol{T}\|^2, \tag{6}$$

where $\|\cdot\|$ denotes the Euclidean norm [5,8,22,23]. Here, aiming to avoid a singular matrix $\boldsymbol{H}^{\top}\boldsymbol{H}$, the generalized inverse matrix $\boldsymbol{H}^{\dagger} = (\boldsymbol{H}^{\top}\boldsymbol{H} + \lambda\boldsymbol{I})^{-1}\boldsymbol{H}^{\top}$ $(N > K)$ or $\boldsymbol{H}^{\dagger} = \boldsymbol{H}^{\top}(\boldsymbol{H}\boldsymbol{H}^{\top} + \lambda\boldsymbol{I})^{-1}$ $(N < K)$ with a small positive constant $0 < \lambda \ll 1$ and the identity matrix $\boldsymbol{I}$ [5,22,23].

For instance, the approximation capability of the threshold network is evaluated on a benchmark unidimensional function [5,22,23]

$$f(x) = \begin{cases} \frac{\sin(x)}{x}, & x \neq 0, \\ 1, & x = 0. \end{cases} \tag{7}$$

The training set $\{\boldsymbol{x}^{(p)}, \boldsymbol{t}^{(p)}\}_{p=1}^{P}$ is generated equally spaced in the interval $[-10, 10]$ with the length of data $P = 41$. Here, for reference, the maximum difference of the target function in the interval $[a, b]$ is defined as $\Delta = \max f(x) - \min f(x), \forall x \in [a, b]$. For the unidimensional target function of Eq. (7) in the interval $[-10, 10]$, the maximum difference is given by $\Delta = 1.2172$.

It is shown in Fig. 1 (a) that, upon increasing the hidden neuron number K, the root-mean-square (RMS) of the error energy $\sqrt{\mathcal{J}}/\Delta$ decreases monotonically, and each point denotes a statistical averaged value for 100 trials of the randomly selected $\boldsymbol{W}$ and $\boldsymbol{b}$. However, it is noted in Fig. 1 (a) that $\sqrt{\mathcal{J}}/\Delta$ is sensitive to the distributed intervals of the weight matrix $\boldsymbol{W}$ and the bias vector $\boldsymbol{b}$. For instance, when $\boldsymbol{W}$ and $\boldsymbol{b}$ are both uniformly distributed in the range of $[0, 1]$, the positive input $\boldsymbol{x}^{(p)} \in [0, 10]$ will lead to the positive local field $\boldsymbol{W}\boldsymbol{x}^{(p)} + \boldsymbol{b} > \boldsymbol{0}$. Then the hidden layer output vector $\boldsymbol{h}^{(p)} = \phi[\boldsymbol{W}\boldsymbol{x}^{(p)} + \boldsymbol{b}]$ becomes the $1 \times K$ dimensional constant row vector $\boldsymbol{1} = [1, 1, \cdots, 1]$. Thus, all network outputs $\boldsymbol{y}^{(p)} = \boldsymbol{h}^{(p)}\boldsymbol{U}$ in Eq. (2) are equal to the sum of the $K \times 1$ weight vector of $\boldsymbol{U}$, as shown in Fig. 1 (b) (□). Therefore, the generalized performance of the threshold network trained by the ELM algorithm heavily depends on the "appropriately" selected $\boldsymbol{W}$ and $\boldsymbol{b}$. Moreover, even for the "appropriately" selected $\boldsymbol{W}$ and $\boldsymbol{b}$ that uniformly distribute in the range of $[-1, 1]$, $\sqrt{\mathcal{J}}/\Delta$ has an unsatisfactory statistical averaged value 0.1160 for a very large number $K = 500$ of threshold

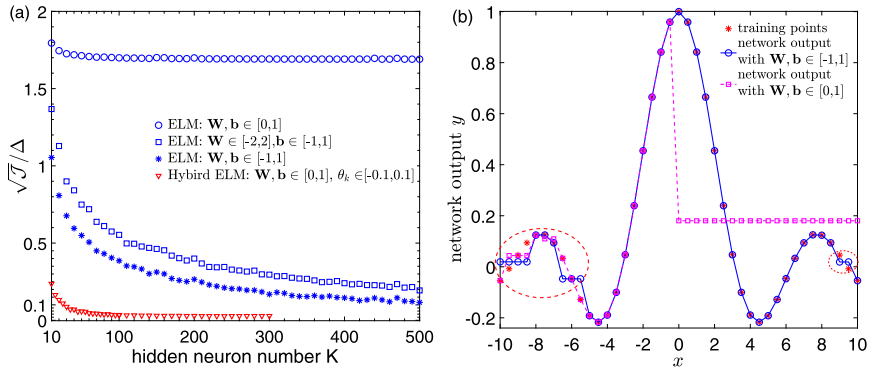

Fig. 1. (a) Statistical averaged values of $\sqrt{\mathcal{J}}/\Delta$ obtained by the ELM algorithm and the proposed hybrid **Algorithm** 1 versus the hidden neuron number K for different distributed intervals of the weight matrix $\boldsymbol{W}$ and the bias vector $\boldsymbol{b}$. (b) Approximation (blue dashed line) of the target function of Eq. (7) obtained by the ELM algorithm for the threshold networks with the size $1 \times 500 \times 1$. The $L = 41$ training data ($*$) sampled from the target function of Eq. (7) are also plotted. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

neurons. An example of the approximation ($\circ$) of the target function of Eq. (7) obtained by the trained threshold neural network is shown in Fig. 1 (b). It is seen in Fig. 1 (b) that, for the $1 \times 500 \times 1$ trained threshold network, the difference between the approximation ($\circ$) and the training data ($*$) of Eq. (7) is clearly distinct in the bounded ranges delineated by the two dashed circles. Although the ELM algorithm can learn much faster than the traditional back-propagation method for training threshold networks, the improved approximation capability of threshold network does need a large network size and the finely selected random weight matrix $\boldsymbol{W}$ and bias vector $\boldsymbol{b}$.

2.2. Hybrid training method of the noise-modulated threshold network

In order to further improve the generalization performance of the threshold network, we here propose a novel hybrid training method that combines the efficiency of ELM and the benefit of injecting noise for smoothing threshold neurons. In the training phase, each neuron in the hidden layer of the threshold network is calculated as a smooth activation function

$$\psi_k = \mathrm{E}_\eta[\phi_k(v_k+\eta)] = \int_{-\infty}^{\infty} \phi(v_k+\eta) f_\eta(\eta) d\eta = \int_{\frac{\theta_k - v_k}{\sigma}}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx, \tag{8}$$

where the artificially injected Gaussian noise is considered and has the probability density function (PDF) $f_\eta(\eta) = \exp(-\eta^2/2\sigma^2)/\sqrt{2\pi\sigma^2}$ with the learnable noise level $\sigma > 0$.

It is seen from Eq. (8) that the noise-smoothed activation function ψ_k is differentiable and varies from zero to unity. Then, for P training examples, we have P hidden layer output vectors $\widetilde{\boldsymbol{h}}^{(p)} = [\psi_1^{(p)}, \psi_2^{(p)}, \cdots, \psi_K^{(p)}]$ and the hidden layer output matrix $\widetilde{\boldsymbol{H}} = [(\widetilde{\boldsymbol{h}}^{(1)})^\top, (\widetilde{\boldsymbol{h}}^{(2)})^\top, \cdots, (\widetilde{\boldsymbol{h}}^{(P)})^\top]^\top$. Furthermore, we regard the noise level σ as a learnable network parameter as well as the threshold parameter θ_k. Then, we propose a novel hybrid training algorithm that combines the fast ELM algorithm for solving the output weight matrix $\boldsymbol{U}$ and the powerful learning ability of the noise-smoothed activation function ψ_k. For clarity, a detailed description of the proposed training method is presented in **Algorithm** 1. It is noted in **Algorithm** 1 that, for each training epoch $\ell = 1, 2, \cdots, L$, the partial derivative of the error energy $\mathcal{J}(\ell)$ with respect to the noise level σ can be expressed as

Algorithm 1: Hybrid training algorithm for threshold networks.

Input: Training set $\{\boldsymbol{x}(p), \boldsymbol{t}(p)\}_{p=1}^{P}$, initial Gaussian noise level $\sigma(0)$, initial threshold parameter vector $\boldsymbol{\theta}(0)$, learning rates α and β, epoch number L, uniform random matrix $\boldsymbol{W}$ and bias vector $\boldsymbol{b}$.

Output: weight matrix $\boldsymbol{U}$, noise level σ.

for *training epoch* $\ell = 1 \to L$ **do**
- ELM learning:
- $\widetilde{\boldsymbol{H}}(\ell) = \{[\widetilde{\boldsymbol{h}}^{(1)}(\ell)]^\top, [\widetilde{\boldsymbol{h}}^{(2)}(\ell)]^\top, \cdots, [\widetilde{\boldsymbol{h}}^{(P)}(\ell)]^\top\}$;
- $\boldsymbol{U}(\ell) \leftarrow \widetilde{\boldsymbol{H}}^{\dagger}(\ell) T$;
- ASR learning:
- $\mathcal{J}(\ell) \leftarrow \|\widetilde{\boldsymbol{H}}(\ell)\boldsymbol{U}(\ell) - \boldsymbol{T}\|^2$;
- $\sigma(\ell) \leftarrow \sigma(\ell-1) - \alpha \dfrac{\partial \mathcal{J}(\ell)}{\partial \sigma}\Big|_{\sigma=\sigma(\ell-1)}$;
- $\boldsymbol{\theta}(\ell) \leftarrow \boldsymbol{\theta}(\ell-1) - \beta \dfrac{\partial \mathcal{J}(\ell)}{\partial \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}(\ell-1)}$;

end

$$\frac{\partial \mathcal{J}(\ell)}{\partial \sigma} = \sum_{k=1}^{K}\sum_{m=1}^{M} \sqrt{\frac{2}{\pi}} \left[y_m(\ell) - t_m\right] [\boldsymbol{U}]_{km} \frac{(\theta_k - v_k)}{\sigma^2} e^{-\frac{(\theta_k - v_k)^2}{2\sigma^2}}, \tag{9}$$

where $[\boldsymbol{U}]_{km}$ denotes the weight connecting the neuron m in the output layer with the k-th neuron ψ_k in the hidden layer. Similarly, the partial derivative of $\mathcal{J}(\ell)$ with respect to the threshold parameter θ_k is given by

$$\frac{\partial \mathcal{J}(\ell)}{\partial \theta_k} = \sum_{m=1}^{M} -\sqrt{\frac{2}{\pi}} \frac{\left[y_m(\ell) - t_m\right] [\boldsymbol{U}]_{km}}{\sigma} e^{-\frac{(\theta_k - v_k)^2}{2\sigma^2}}. \tag{10}$$

Using Eqs. (9) and (10), the hybrid training **Algorithm** 1 can be implemented for training the noise-modulated threshold network with the smoothed function ψ_k defined in Eq. (8).

2.3. Function approximation

For comparison, using the proposed hybrid training **Algorithm** 1, $\sqrt{\mathcal{J}}/\Delta$ versus the hidden neuron number K is also plotted in Fig. 1 (a) ($\triangledown$). It is seen in Fig. 1 (a) that the threshold network with a small size of $1 \times 25 \times 1$ can attain the statistical averaged value of $\sqrt{\mathcal{J}}/\Delta \approx 0.1095$, which is improved over $\sqrt{\mathcal{J}}/\Delta \approx 0.1160$ achieved by the threshold network ($1 \times 500 \times 1$) directly trained by the ELM algorithm. An example of the approximation of the target function of Eq. (7) obtained by the trained $1 \times 25 \times 1$ threshold neural network is illustrated in Fig. 2 (a). It

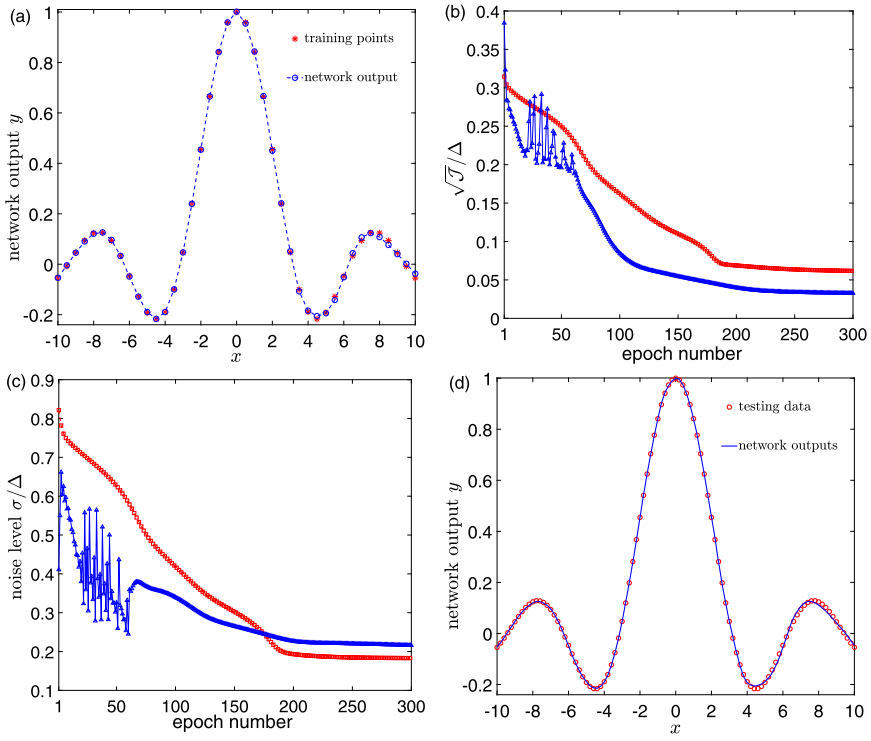

Fig. 2. (a) Approximation (blue dashed line) of the target function in Eq. (7) obtained by the trained threshold neural network via the hybrid **Algorithm** 1. The $L = 41$ training data ($*$) sampled from the target function of Eq. (7) are also plotted. (b) The $\sqrt{\mathcal{J}}/\Delta$ and (c) the noise level σ/Δ versus the epoch number of training. Here, the learning rates $\alpha = \beta = 1$, $\boldsymbol{W}$ and $\boldsymbol{b}$ are uniformly distributed in the interval $[-1, 1]$. Two initial values of the noise level $\sigma(0)/\Delta = 0.8215$ (□) and 0.4108 (△) are considered. (d) Outputs (blue solid line) of the trained threshold network for approximating the target function of Eq. (7). The testing data (○) of Eq. (7) are also plotted.

is seen in Fig. 2 (a) that the trained threshold neural network assisted by the addition of noise performs well for approximating the training set sampled from the target unidimensional function of Eq. (7).

In Figs. 2 (b) and (c), upon increasing the training epoch number, both $\sqrt{\mathcal{J}}/\Delta$ and the noise level σ/Δ first greatly decrease and then reach convergence effectively. Interestingly, two initial values of the noise level $\sigma(0)/\Delta = 0.8215$ (□) and 0.4108 (△) are considered, and the corresponding converged noise levels $\sigma/\Delta = 0.1832$ and 0.2163 are adaptively found by the proposed hybrid **Algorithm** 1, respectively. Correspondingly, the local convergence of $\sqrt{\mathcal{J}}/\Delta$ also approaches different small values, as shown in Fig. 2 (b). This result indicates the error energy $\mathcal{J}$ is a nonconvex function of the network parameters containing the noise level σ and weights and exhibiting several local minima.

After 300 training epochs illustrated in Figs. 2 (b) and (c), the randomly assigned weight matrix $\boldsymbol{W}$ and the bias vector $\boldsymbol{b}$, the converged weight matrix $\boldsymbol{U}$, threshold parameters θ_k and the (local) optimal noise level σ define the trained threshold network. However, in the testing phase, the noise-smoothed activation function ψ_k is a limiting expression with respect to the injected noise PDF and needs to be asymptotically implemented by injecting a sufficiently large number of mutually independent noise components into the threshold neuron of Eq. (1). In practice, since the injected noise is stationary, then we can artificially manufacture a sufficiently large number T of mutually independent noise components and numerically simulate the trained threshold network for T trials. The size of the threshold network is the same as in the training phase. In line with this, we can treat the average value $\sum_{t=1}^{T} \boldsymbol{y}_t^{(p)}/T = (\sum_{t=1}^{T} \boldsymbol{h}_t^{(p)}/T)\boldsymbol{U} \approx \boldsymbol{h}^{(p)}\boldsymbol{U}$ of the T experimental outputs of the network as the approximation of the target function in Eq. (7). Here, the subscript t denotes the t-th trial of the network output.

An illustrative example of the approximation (blue solid line) of the $1 \times 25 \times 1$ threshold network is plotted in Fig. 2 (d) by averaging $T = 10^4$ times of experimental results of the network outputs. For comparison, the testing points (○) are also shown in Fig. 2 (d), and the corresponding $\sqrt{\mathcal{J}}/\Delta = 0.1115$ for the converged Gaussian noise level $\sigma/\Delta = 0.1832$ in testing phase. It is noted that, as the length of testing data is $N = 81$, $\sqrt{\mathcal{J}}/\Delta = 0.1115$ is larger than the result of $\sqrt{\mathcal{J}}/\Delta \approx 0.1095$ obtained by $N = 41$ training data. By dividing the data length N into the error energy $\mathcal{J}$, $\sqrt{\mathcal{J}/N}/\Delta \sim 0.01$ in both training and testing phases represents the RMS of mean square error. It is seen in Fig. 2 (d) that the trained threshold neural network assisted by the addition of noise performs well on the test for approximating the target unidimensional function of Eq. (7). The injection of noise into the threshold activation function does improve the learning capacity of threshold networks.

Furthermore, using the proposed hybrid training **Algorithm** 1, the non-zero value of the converged σ shows the general usefulness of injecting noise in the threshold network, and the occurrence of the ASR effect is evidently demonstrated. For visualizing the ASR effect, Figs. 3 (a) and (b) illustratively show the learning curve of $\sqrt{\mathcal{J}}/\Delta$ as a function of the noise level σ and the threshold parameter θ_{16} of the 16-th hidden neuron. It is clearly seen in Figs. 3 (a) and (b) that the local optima of the noise level σ obtained by the adaptive learning **Algorithm** 1 in the "resonance" domain of the landscape of $\sqrt{\mathcal{J}}/\Delta$ is really not zero. This also indicates that noise injection becomes an essential composition of the designed threshold neural network.

Furthermore, we also test a two-dimensional function

$$f(x_1, x_2) = 3(1 - x_1)^2 e^{-x_1^2 - (x_2+1)^2} - 10\Big(\frac{x_1}{5} - x_1^3 - x_2^5\Big)e^{-x_1^2 - x_2^2} - \frac{1}{3}e^{-(x_1+1)^2 - x_2^2}. \quad (11)$$

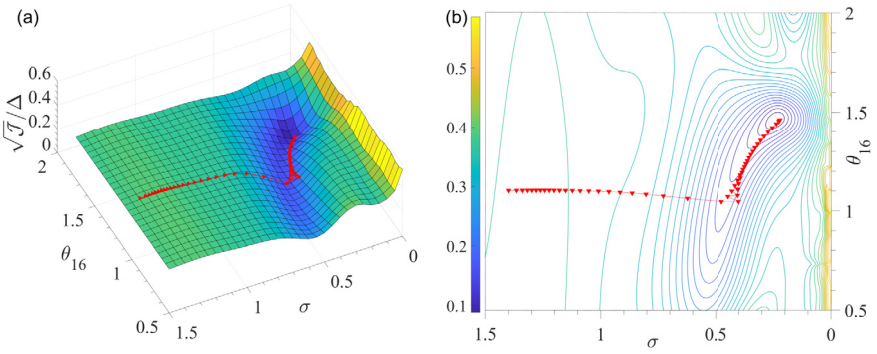

Fig. 3. (a) Landscape surface of $\sqrt{\mathcal{J}}/\Delta$ and (b) its contour in the parameter space of the noise level σ and the threshold parameter θ_{16} of the 16-th hidden neuron. The learning curve of $\sqrt{\mathcal{J}}/\Delta$ (▶) versus σ and θ_{16} is also plotted. Other parameters are in accordance with Fig. 2.

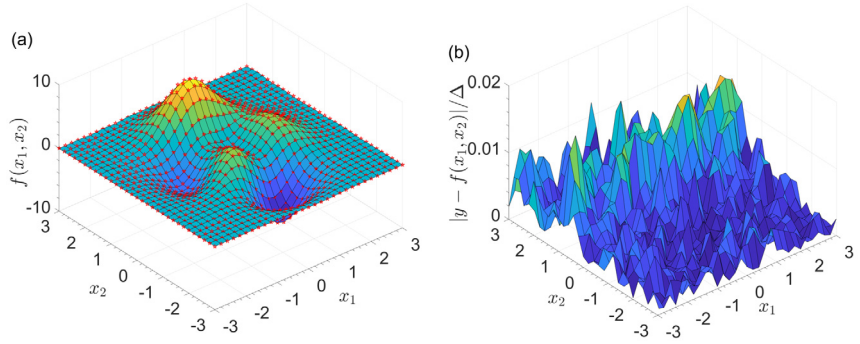

Fig. 4. (a) Network outputs y of the trained neural network via the hybrid **Algorithm** 1 as the approximation (patched surface) to the 32×32 testing data (⋆) of the two-dimensional function $f(x_1,x_2)$ in Eq. (11) in the range $[-3,3]\times[-3,3]$. (b) The corresponding relative error $|y-f(x_1,x_2)|/\Delta$ between the threshold network output and the testing data.

The 16×16 training set is equally spaced in the range $[-3,3]\times[-3,3]$, and the maximum difference is $\Delta = 13.4628$ for the target function in Eq. (11). Using the proposed learning **Algorithm** 1 for training a $2\times 100\times 1$ threshold network, $\sqrt{\mathcal{J}}/\Delta = 0.1082$ and the converged noise level $\sigma_\eta/\Delta = 0.0858$ after 500 training epochs. Then, for 32×32 testing set of Eq. (11), Fig. 4 (a) illustrates the outputs of the trained threshold network as the approximation (patched surface) of the two-dimensional function $f(x_1,x_2)$ in Eq. (11) in the range $[-3,3]\times[-3,3]$. $\sqrt{\mathcal{J}}/\Delta = 0.1614$ is obtained by $T = 10^4$ times of experimental results of the threshold network with its size $2\times 100\times 1$. By dividing the data length N into the error energy $\mathcal{J}$, the RMS of mean square error $\sqrt{\mathcal{J}/N}/\Delta$ is roughly equal to 0.005 in both training and testing phases. The relative errors $|y-f(x_1,x_2)|/\Delta$ between the threshold network output y and the testing data are also plotted in Fig. 4 (b). The maximum relative error $\max|y-f(x_1,x_2)|/\Delta = 0.0170$ is obtained. For comparison, the threshold network with the same size $2\times 100\times 1$ trained by the ELM algorithm has $\sqrt{\mathcal{J}}/\Delta = 0.9096$ for 16×16 training set of Eq. (11). For 32×32 testing set of Eq. (11), $\sqrt{\mathcal{J}}/\Delta = 1.9523$ for the trained threshold network by the ELM algorithm, and a large relative error $\max|y-f(x_1,x_2)|/\Delta = 0.2492$ between the network output and the testing data is obtained. Aiming to reach the same level of $\sqrt{\mathcal{J}}/\Delta = 0.1082$ in the testing phase, the threshold network with a relatively large size $2\times 1200\times 1$ trained by the ELM algorithm is required. Therefore, the proposed learning **Algorithm** 1 can greatly improve the threshold network with a small size to fulfill the accuracy requirement for function approximation.

2.4. Real world data set of the function regression

We further validate the noise-modulated threshold network trained by the proposed **Algorithm** 1 in five real world data sets. The $N\times 50\times 1$ threshold network is employed for the real-world N-dimensional data sets of Auto MPG ($N=7$) [28], Housing ($N=13$) [29], Airfoil noise ($N=5$) [28], Wine quality ($N=11$) [30] and QSAR fish toxicity ($N=6$) [31] in a computer equipped with CPU of Intel Core i7-6700@3.40 Ghz and 16G RAM DDR4@2133 Mhz. Here, the data sets contain 198, 253, 300, 980 and 908 examples, respectively. Note that 50% of data is used for training, while 50% of data is employed to test the trained threshold neural network. For comparison, besides directly applying the ELM to the threshold network and the proposed hybrid **Algorithm** 1, replacing the threshold activation function by the sigmoid one $1/(1+e^{-\lambda x})$ with a large parameter $\lambda = 10$ in the training phase is also considered.

Table 1 lists the $\sqrt{\mathcal{J}}/\Delta$ obtained by three learning algorithms in training and testing phases. It is seen in Table 1 that, for training the threshold network by three considered algorithms, the proposed hybrid algorithm can achieve a smaller $\sqrt{\mathcal{J}}/\Delta$ in both training and testing phases. Moreover, for randomly chosen input weight matrix $\boldsymbol{W}$ and bias vector $\boldsymbol{b}$ (not learning in the training phase), the proposed hybrid **Algorithm** 1 is also able to harness the constructive role of injected noise to improve the learning ability of the threshold network, and the generalization performance of threshold network is unaffected by the randomly assigned $\boldsymbol{W}$ and $\boldsymbol{b}$. Meanwhile, for inappropriately selected $\boldsymbol{W}$ and $\boldsymbol{b}$, the threshold network trained by the ELM algorithm will yield a degraded $\sqrt{\mathcal{J}}/\Delta$. For instance, when $\boldsymbol{W}$ and $\boldsymbol{b}$ are both uniformly distributed in the range of $[-1,0]$, the threshold network has a degraded training with $\sqrt{\mathcal{J}}/\Delta = 16.2022$ and a worse testing with $\sqrt{\mathcal{J}}/\Delta = 16.6513$ for the "Airfoil noise" data set. Under the same condition of $\boldsymbol{W}$ and $\boldsymbol{b}$, the proposed hybrid **Algorithm** 1 can still train the threshold network to achieve an improved $\sqrt{\mathcal{J}}/\Delta = 2.9585$, and also perform well in the testing phase with $\sqrt{\mathcal{J}}/\Delta = 2.9913$. Of course, the ELM algorithm runs extremely fast, and the proposed hybrid **Algorithm** 1 needs more

Table 1
Comparison of results of $\sqrt{\mathcal{J}}/\Delta$ of three training algorithms for threshold networks.

$\sqrt{\mathcal{J}}/\Delta$ / Data set	Sigmoid ($\lambda = 10$)		ELM		Hybrid ELM	
	Training	Testing	Training	Testing	Training	Testing
Airfoil noise	9.9582	19.3847	3.7551	4.1532	2.6479	2.9899
Auto MPG	9.5361	16.9982	1.179	1.3984	0.9121	1.0164
Boston house prices	11.1299	18.6579	1.7283	2.3261	1.309	1.7265
Wine quality	8.8003	10.9927	6.2301	6.3571	5.8825	6.0615
QSAR fish toxicity	9.1754	14.062	2.1438	2.3669	1.8652	2.0604

Table 2
Comparison of training time of three training algorithms for threshold networks.

Time (s) / Data set	Sigmoid ($\lambda = 10$)	ELM	Hybrid ELM
Airfoil noise	0.2801	0.0019	0.1648
Auto MPG	0.0732	0.0015	0.0656
Boston house prices	0.1235	0.0018	0.0780
Wine quality	0.8780	0.0269	0.4286
QSAR fish toxicity	0.1545	0.0023	0.1109

time to finely optimize the noise-modulated threshold network. It is seen in Table 2 that the training time of the proposed hybrid algorithm is found between the computing times taken by the ELM algorithm and the conventional backpropagation approach.

2.5. Multiclass classification

We also compare the performances of threshold networks trained by the three considered algorithms for multiclass classification problems. Seven real world multiclass data sets of Balance scale, Haberman, Wine, Iris, Data user modeling and Hayes-Roth are taken from UCI Machine Learning Repository [28]. The corresponding class features are $3, 2, 3, 3, 4, 2$ and 3, respectively. Here, 80% of data are used for training, while 20% of data are employed to test the trained threshold neural network. The number of hidden neurons is $K = 100$. Table 3 shows the classification success rates of the threshold network trained by the three considered algorithms in training and testing phases, respectively. It is shown in Table 3 that the proposed hybrid learning algorithm, compared with the two other algorithms, can train the threshold network with higher success rates in the training stage, and achieve higher testing success rates for each real world multiclass data set. Similarly, Table 4 indicates that the proposed hybrid algorithm takes more training time than the ELM method, but less than the approach of replacing the threshold activation function by the sigmoid one in threshold networks.

3. Conclusion

Although the ELM algorithm can be directly applied to train the threshold network, this fast learning method suffers two limitations when dealing with threshold networks with a very large size and the "appropriately" randomized input weight matrix and bias vector not learning in the training phase. By means of a beneficial role of noise for smoothing the threshold activation function, noise injection can transform the threshold network into a conventional neural network with differentiable activation functions. Then, the present paper proposes a hybrid learning algorithm that combines both characteristics of the fast learning rate of the ELM algorithm and the noise boosted capability of the ASR phenomenon. Moreover, for absolutely randomized weight matrix and bias vector, the proposed hybrid learning algorithm can adaptively search for the optimal value of noise to implement a smooth input-output nonlinearity of neurons, and train the noise-modulated threshold network to be relatively insensitive to randomized input network parameters. Experimental results on regression and multiclass classification in real world data sets demonstrate that the proposed algorithm can optimize the threshold network with better generalization performance than the ELM algorithm and the conventional backpropagation algorithm.

This proposed hybrid learning algorithm, benefiting from the noise injected into the threshold activation function, can train the threshold network with a much better generalization performance. As a follow-up, some open questions remain to be further investigated. For example, the search of the optimal noise level occupies the main training time of the threshold network via the proposed hybrid learning method. Thus, how to shorten the training time and find the optimal noise level faster are crucial problems for the implementation of the noise-modulated threshold network. It is also noted that the shallow threshold network with only one hidden layer is considered in this paper, and then whether the proposed learning algorithm can potentially be applied to more deeper threshold networks [7,10] remains to be examined. As the threshold network becomes deep, the hidden layers increase and the injected noise level in each hidden layer to be searched also increases. Improved generalization performance of deep threshold network with multiple injected noise sources is expected by implementing the proposed hybrid learning method. In addition, when the trained threshold network is applied in the testing phase, a sufficiently large number of trials needs to be carried out and the same number of mutually independent noise components is required. Addressing the issue of heavy computation in the testing phase will be a key problem to pursue. As another extension to this study, it may be of interest to further investigate the possibility and application of the proposed algorithm based on the ASR mechanism when the activation functions in neural networks employ a much wider family of activation functions that are non-differentiable or with zero gradients. The stochastic resonance effect has been extensively explored in dynamical systems [21,24,27,32,33,35–37] and complex networks [38–42] under various complex environments, and then the noise injection in these complex systems deserves to be further investigated on its adaptive learning capability.

CRediT authorship contribution statement

Zejia Chen: Methodology, Software, Writing – original draft. **Fabing Duan:** Conceptualization, Investigation, Methodology, Writing – original draft. **François Chapeau-Blondeau:** Formal analysis, Methodology, Writing – review & editing. **Derek Abbott:** Writing – review & editing, Methodology, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Table 3
Comparison of classification success rates of three training algorithms for threshold networks.

Rate(%) / Data set	Sigmoid ($\lambda = 10$)		ELM		Hybrid ELM	
	Training	Testing	Training	Testing	Training	Testing
Balance scale [28]	89.20	66.40	90.4	87.20	91.80	87.20
Haberman [28]	74.28	74.28	79.59	73.77	77.14	78.68
Wine [29]	100	68.57	99.30	91.43	100	94.28
Iris [30]	85.83	63.33	95.83	93.33	96.66	100
Data user modeling [31]	70.09	35.00	82.55	67.50	90.03	72.50
Banknote [31]	97.81	53.28	95.08	94.89	99.36	100
Hayes-Roth [31]	60.37	23.07	98.11	61.53	92.45	88.46

Table 4
Comparison of training time of three training algorithms for threshold networks.

Time (s) / Data set	Sigmoid ($\lambda = 10$)	ELM	Hybrid ELM
Balance scale [28]	0.3416	0.0022	0.2248
Haberman [28]	0.1645	0.0026	0.1275
Wine [29]	0.1645	0.0020	0.0969
Iris [30]	0.0790	0.0018	0.0796
Data user modeling [31]	0.2286	0.0023	0.1668
Banknote [31]	0.7390	0.0044	0.4276
Hayes-Roth [31]	0.0855	0.0014	0.0785

Acknowledgements

This work is supported by the Natural Science Foundation of Shandong Province (No. ZR2021MF051) and the Australian Research Council (No. DP200103795).

References

[1] D. Toms, Training binary node feedforward neural networks by back propagation of error, Electron. Lett. 26 (21) (1990) 1745–1746.
[2] P. Bartlett, T. Downs, Using random weights to train multilayer networks of hard-limiting units, IEEE Trans. Neural Netw. 3 (2) (1992) 202–210.
[3] E. Corwin, A. Logar, W. Oldham, An iterative method for training multilayer networks with threshold functions, IEEE Trans. Neural Netw. 5 (3) (1994) 507–508.
[4] E. Wilson, S. Rock, Gradient-based parameter optimization for systems containing discrete-valued functions, Int. J. Robust Nonlinear Control 12 (9) (2002) 1009–1028.
[5] G. Huang, Q. Zhu, K. Mao, C. Siew, P. Saratchandran, N. Sundararajan, Can threshold networks be trained directly?, IEEE Trans. Circuits Syst. II, Express Briefs 53 (3) (2006) 187–191.
[6] C. Gulcehre, M. Moczulski, M. Denil, Y. Bingio, Noisy activation functions, arXiv, available: https://arxiv.org/abs/1603.00391v3, 2016.
[7] M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi, XNOR-net: ImageNet classification using binary convolutional neural networks, in: European Conference on Computer Vision, Springer, 2016, pp. 525–542.
[8] S. Ikemoto, F. Dallalibera, K. Hosoda, Noise-modulated neural networks as an application of stochastic resonance, Neurocomputing 277 (2017) 29–37.
[9] S. Ikemoto, Noise-modulated neural networks for selectively functionalizing sub-networks by exploiting stochastic resonance, Neurocomputing 448 (2021) 1–9.
[10] X. Liu, L. Duan, F. Duan, F. Chapeau-Blondeau, D. Abbott, Enhancing threshold neural network via suprathreshold stochastic resonance for pattern classification, Phys. Lett. A 403 (2021) 127387.
[11] S. Uhlich, Bayes risk reduction of estimators using artificial observation noise, IEEE Trans. Signal Process. 63 (20) (2015) 5535–5545.
[12] D. Simon, J. Sulam, Y. Romano, Y. Lu, M. Elad, MMSE approximation for sparse coding algotithms using stochastic resonance, IEEE Trans. Signal Process. 67 (17) (2019) 4597–4610.
[13] H. Chen, P.K. Varshney, Nonparametric one-bit quantizers for distributed estimation, IEEE Trans. Signal Process. 58 (7) (2010) 3777–3787.
[14] J. Liu, F. Duan, F. Chapeau-Blondeau, D. Abbott, Distributed Bayesian vector estimation using noise-optimized low-resolution sensor observations, Digit. Signal Process. 118 (2021) 103224.
[15] L. Duan, F. Duan, F. Chapeau-Blondeau, D. Abbott, Noise-boosted backpropagation learning of feedforward threshold neural networks for function approximation, IEEE Trans. Instrum. Meas. 70 (2021) 1010612.
[16] N. Frazier-Logue, S.J. Hanson, The stochastic delta rule: faster and more accurate deep learning through adaptive weight noise, Neural Comput. 32 (5) (2020) 1018–1032.
[17] H. Qin, R. Gong, X. Liu, X. Bai, J. Song, N. Sebe, Binary neural networks: a survey, Pattern Recognit. 105 (2020) 107281.
[18] N.G. Stocks, Suprathreshold stochastic resonance in multilevel threshold systems, Phys. Rev. Lett. 84 (11) (2000) 2310–2313.
[19] M.D. McDonnell, N.G. Stocks, C.E.M. Pearce, D. Abbott, Stochastic Resonance: From Suprathreshold Stochastic Resonance to Stochastic Signal Quantization, Cambridge University Press, New York, 2008.
[20] D. Rousseau, F. Chapeau-Blondeau, Suprathreshold stochastic resonance and signal-to-noise ratio improvement in arrays of comparators, Phys. Lett. A 321 (5–6) (2004) 280–290.
[21] Y. Fu, Y. Kang, G. Chen, Stochastic resonance based visual perception using spiking neural networks, Front. Comput. Neurosci. 14 (2020) 24.
[22] G. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification, IEEE Trans. Syst. Man Cybern., Part B, Cybern. 42 (2) (2011) 513–529.
[23] G. Huang, D. Wang, Y. Lan, Extreme learning machines: a survey, Int. J. Mach. Learn. Cybern. 2 (2) (2011) 107–122.
[24] S. Mitaim, B. Kosko, Adaptive stochastic resonance, Proc. IEEE 86 (11) (1998) 2152–2183.
[25] B. Kosko, K. Audhkhasi, O. Osoba, Noise can speed backpropagation learning and deep bidirectional pretraining, Neural Netw. 129 (2020) 359–384.
[26] W. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, Bull. Math. Biophys. 5 (4) (1943) 115–133.
[27] D. Guo, M. Perc, T. Liu, D. Yao, Functional importance of noise in neuronal information processing, Europhys. Lett. 124 (5) (2018) 50001.
[28] D. Dua, C. Graff, UCI machine learning repository [Online], available: http://archive.ics.uci.edu/ml, 2017.
[29] M. Rafiei, H. Adeli, A novel machine learning model for estimation of sale prices of real estate units, J. Constr. Eng. Manage. 142 (2) (2016) 04015066.
[30] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, J. Reis, Modeling wine preferences by data mining from physicochemical properties, Decis. Support Syst. 47 (4) (2009) 547–553.
[31] M. Cassotti, D. Ballabio, R. Todeschini, V. Consonni, A similarity-based QSAR model for predicting acute toxicity towards the fathead minnow (Pimephales promelas), SAR QSAR Environ. Res. 26 (3) (2015) 217–243.
[32] Q. Yu, X. Liu, Self-induced stochastic resonance in an excitable potential well, Phys. Lett. A 410 (17) (2021) 127520.
[33] F. Gao, Y. Kang, X. Chen, G. Chen, Fractional Gaussian noise enhanced information capacity of a nonlinear neuron model with binary input, Phys. Rev. E 97 (5) (2018) 052142.
[34] M.B. Ghori, Y. Kang, Y. Chen, Emergence of stochastic resonance in a two-compartment hippocampal pyramidal neuron model, J. Comput. Neurosci. (2022), https://doi.org/10.1007/s10827-021-00808-2 [Online].
[35] F. Gao, Y. Kang, Positive role of fractional Gaussian noise in FitzHugh–Nagumo neuron model, Chaos Solitons Fractals 146 (5) (2021) 110914.
[36] W. Zhang, P. Shi, M. Li, D. Han, A novel stochastic resonance model based on bistable stochastic pooling network and its application, Chaos Solitons Fractals 145 (4) (2021) 110800.
[37] H. Dong, X. Shen, K. He, H. Wang, Nonlinear filtering effects of intrawell matched stochastic resonance with barrier constrainted Duffing system for ship radiated line signature extraction, Chaos Solitons Fractals 141 (12) (2020) 110428.
[38] Z. Liao, Z. Wang, H. Yamahara, H. Tabata, Echo state network activation function based on bistable stochastic resonance, Chaos Solitons Fractals 153 (12) (2021) 111503.
[39] D. Guo, M. Perc, Y. Zhang, P. Xu, D. Yao, Frequency-difference-dependent stochastic resonance in neural systems, Phys. Rev. E 96 (2) (2017) 022415.
[40] E. Yilmaz, M. Uzuntarla, M. Ozer, M. Perc, Stochastic resonance in hybrid scale-free neuronal networks, Physica A 392 (22) (2013) 5735–5741.
[41] M. Perc, Stochastic resonance on weakly paced scale-free networks, Phys. Rev. E 78 (3) (2008) 036105.
[42] H. Yu, X. Guo, J. Wang, C. Liu, B. Deng, X. Wei, Adaptive stochastic resonance in self-organized small-world neuronal networks with time delay, Commun. Nonlinear Sci. Numer. Simul. 29 (1) (2015) 1007–5704.