Contents lists available at ScienceDirect

Physics Letters A

www.elsevier.com/locate/pla

Enhancing threshold neural network via suprathreshold stochastic resonance for pattern classification

Xiaojie Liu[a], Lingling Duan[a,b], Fabing Duan[a,*], François Chapeau-Blondeau[c], Derek Abbott[d]

[a] Institute of Complexity Science, Qingdao University, Qingdao 266071, PR China
[b] Department of Mathematics, Jining University, Jining 273155, PR China
[c] Laboratoire Angevin de Recherche en Ingénierie des Systèmes (LARIS), Université d'Angers, 62 avenue Notre Dame du Lac, 49000 Angers, France
[d] Centre for Biomedical Engineering (CBME) and School of Electrical & Electronic Engineering, The University of Adelaide, Adelaide, SA 5005, Australia

ARTICLE INFO

Article history:
Received 16 February 2021
Received in revised form 21 April 2021
Accepted 23 April 2021
Available online 1 May 2021
Communicated by M. Perc

Keywords:
Threshold neural network
Suprathreshold stochastic resonance
Backpropagation
Pattern classification
Noise-smoothed activation function

ABSTRACT

Hard-threshold nonlinearities are of significant interest for neural-network information processing due to their simplicity and low-cost implementation. They however lack an important differentiability property. Here, hard-threshold nonlinearities receiving assistance from added noise are pooled into a large-scale summing array to approximate a neuron with a noise-smoothed activation function. Differentiability that facilitates gradient-based learning is restored for such neurons, which are assembled into a feed-forward neural network. The added noise components used to smooth the hard-threshold responses have adjustable parameters that are adaptively optimized during the learning process. The converged non-zero optimal noise levels establish a beneficial role for added noise in operation of the threshold neural network. In the retrieval phase the threshold neural network operating with non-zero optimal added noise, is tested for data classification and for handwritten digit recognition, which achieves state-of-the-art performance of existing backpropagation-trained analog neural networks, while requiring only simpler two-state binary neurons.

1. Introduction

Neural networks with threshold or hard-limiting activation functions have attracted a lot of interest in past decades as they allow low-cost hardware implementation [1–7]. However, gradient-based learning algorithms are not applicable for training these networks as the threshold function is nondifferentiable with zero gradients [1–7]. A number of learning methods have been proposed for training these kinds of threshold neural networks, such as replacing the threshold activation functions with smoothed ones [2,4,5,8], randomizing weights with smooth distribution functions [1,3,7] and using extreme machine learning [6,8]. Of special interest is the noise-modulated neural network designed by Ikemoto et al. [8] as an application of the suprathreshold stochastic resonance (SSR) mechanism, wherein an optimal amount of deliberately added noise not only facilitates threshold network learning during the training process, but also benefits the network in terms of function approximations and regression problems at run time [8]. However, the determination of the optimal noise remains unsolvable, and a manual grid search on the noise level is employed [8].

Thus, how to successively adjust the noise level during threshold neural network training, as well as making adjustments to the weights by the backpropagation algorithm, become critical questions to be solved [9]. Notably, the SSR effect in a large-scale network can result in large information gains significantly greater than is attainable in a single threshold element [10–12]. It is emphasized that SSR differs from the subthreshold stochastic resonance—in the case of SSR, a single threshold device is subjected to a weak signal that is smaller than the threshold level, and SSR occurs regardless of whether the input signal is entirely subthreshold or suprathreshold (larger than the threshold level) [10–13]. When the number of threshold elements increases to infinity, the asymptotic behavior can be seen as an ensemble-averaging operation on the threshold element with respect to the added noise distribution. This distinguishing feature of the asymptotic behavior of SSR was first theoretically analyzed in order to obtain the most efficient signal-to-noise ratio amplifier [14].

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

https://doi.org/10.1016/j.physleta.2021.127387
0375-9601/© 2021 Elsevier B.V. All rights reserved.

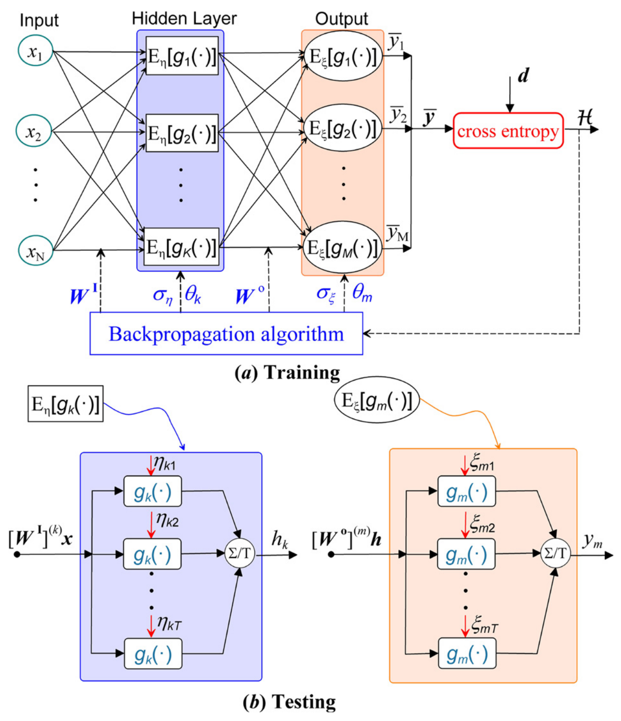

Fig. 1. Block diagram representations of (a) training the feedforward neural network with the noise-smoothed activation functions $\mathrm{E}_\eta[g_k(\cdot)]$ $(k = 1, 2, \ldots, K)$ in the hidden layer and $\mathrm{E}_\xi[g_m(\cdot)]$ $(m = 1, 2, \ldots, M)$ in the output layer, and (b) the pattern classification testing experiments operated by injecting noise components η_{kt} into threshold elements $g_k(\cdot)$ and noise components ξ_{mt} into threshold elements $g_m(\cdot)$ for $t = 1, 2, \ldots, T$.

Later, this asymptotic feature was incorporated in constructing the noise-enhanced estimators [12,15–17], detectors [18–22] and noise-modulated neural network [8,23–25]. Recently, a visual perception algorithm combined with stochastic resonance has also been proposed to choose a suitable spiking threshold for spiking neural networks [26].

Note that each neuron in feedforward threshold neural networks approximates a noise-smoothed function when it consists of a sufficiently large number of threshold elements. Then, each neuron in the designed neural network becomes a differentiable function of both the noise level and the network parameters, and the gradient-based backpropagation learning algorithm can be executed for training the threshold neural network. We here argue that the noise level in different layers of the threshold network, as well as the connected weights, undergoes adjustment in accordance with the steepest descent of the cost function. After a certain number of training epochs, the trained weights and the converged non-zero noise levels are employed to establish the threshold neural network for the testing session. It is emphasized in testing that noise-smoothed activation functions of the neural network are replaced with a finite number of threshold elements that are activated by mutually independent noise components. Since these noise components in a layer have the same non-zero noise level, then the mechanism of SSR is the key to implementing the trained threshold neural network in the testing experiments. The testing results of data classification and handwritten digit recognition, including the implementation on the threshold neural network, validate the realizability and practical utility of the proposed gradient-based backpropagation learning algorithm.

2. Large-scale threshold neural network

2.1. Threshold network and noise-smoothed activation functions

Let us consider a feedforward neural network with a single hidden layer as illustrated in Fig. 1, which has an input vector $\boldsymbol{x} = [x_1, x_2, \ldots, x_N]^\top$ in the input layer, K neurons in the hidden layer and M neurons in the output layer. The $K \times N$ matrix $\boldsymbol{W}^{\mathrm{I}}$ contains the weights of the connections between the hidden neurons and inputs, and the $M \times K$ weight matrix $\boldsymbol{W}^{\mathrm{o}}$ connects the output and hidden layers. Each neuron is composed of the classic SSR model of a summing array of threshold elements [10], and the k-th neuron in the hidden layer has T identical threshold elements denoted by

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

with the same threshold parameter θ_k as shown in Fig. 1. It is noted that these threshold elements in Eq. (1) have a common input $[\boldsymbol{W}^{\mathrm{I}}]^{(k)}\boldsymbol{x}$, but are activated by T mutually independent Gaussian noise components η_{kt} for $t = 1, 2, \ldots, T$, respectively. Here, $[\boldsymbol{W}^{\mathrm{I}}]^{(k)}$ denotes the k-th row of the weight matrix $\boldsymbol{W}^{\mathrm{I}}$, and η_{kt} have the common probability density function (PDF) $f_\eta(\eta) = \exp(-\eta^2/2\sigma_\eta^2)/\sqrt{2\pi\sigma_\eta^2}$ and the same noise level σ_η. Then, the output h_k of the k-th neuron in the hidden layer is given by

$$h_k = \frac{1}{T}\sum_{t=1}^{T} g_k\left([\boldsymbol{W}^{\mathrm{I}}]^{(k)}\boldsymbol{x} + \eta_{kt}\right). \tag{2}$$

Then, the $K \times 1$ output vector of the hidden layer can be expressed as $\boldsymbol{h} = [h_1, h_2, \dots, h_K]^\top$. Similarly, the m-th neuron in the output layer is expressed as

$$y_m = \frac{1}{T}\sum_{t=1}^{T} g_m\left([\boldsymbol{W}^{\mathrm{o}}]^{(m)}\boldsymbol{h} + \xi_{mt}\right), \tag{3}$$

where T threshold elements $g_m(\cdot)$ have the same threshold θ_m, $[\boldsymbol{W}^{\mathrm{o}}]^{(m)}$ is the m-th row of the $M \times K$ weight matrix $\boldsymbol{W}^{\mathrm{o}}$ between the output layer with the hidden one, and T mutually independent Gaussian noise components ξ_{mt} having common PDF $f_\xi(\xi) = \exp(-\xi^2/2\sigma_\xi^2)/\sqrt{2\pi\sigma_\xi^2}$ and the same noise level σ_ξ. Then, the $M \times 1$ output vector of the designed neural network becomes $\boldsymbol{y} = [y_1, y_2, \dots, y_M]^\top$.

2.2. Backpropagation algorithm based on SSR

It is noted that the classical backpropagation algorithm [1–5,8] cannot be directly applied to training the threshold neural network, since these functions in Eqs. (1)–(3) are non-differentiable at zero and have zero gradients at the non-zero domain. Aiming to train the threshold neural network, we here consider a large-scale summing array with $T \to \infty$. Using the scale transformation of $\eta_0 = \eta/\sigma_\eta$, we asymptotically represent the neuron output h_k as

$$\begin{aligned}\overline{h}_k &= \lim_{T\to\infty} h_k = \mathrm{E}_\eta\left[g_k\left([\boldsymbol{W}^{\mathrm{I}}]^{(k)}\boldsymbol{x} + \eta\right)\right] = \int g_k\left([\boldsymbol{W}^{\mathrm{I}}]^{(k)}\boldsymbol{x} + \eta\right) f_\eta(\eta) d\eta \\ &= \int_{\frac{\theta_k - [\boldsymbol{W}^{\mathrm{I}}]^{(k)}\boldsymbol{x}}{\sigma_\eta}}^{\infty} \frac{1}{\sqrt{2\pi}} \exp\Big(-\frac{\eta_0^2}{2}\Big) d\eta_0.\end{aligned} \tag{4}$$

Let $\overline{\boldsymbol{h}} = [\overline{h}_1, \overline{h}_2, \dots, \overline{h}_K]^\top$ and $\xi_0 = \xi/\sigma_\xi$, the neuron output y_m also asymptotically converges to

$$\begin{aligned}\overline{y}_m &= \lim_{T\to\infty} y_m = \mathrm{E}_\xi\left[g_m\left([\boldsymbol{W}^{\mathrm{o}}]^{(m)}\overline{\boldsymbol{h}} + \xi\right)\right] = \int g_m\left([\boldsymbol{W}^{\mathrm{o}}]^{(m)}\overline{\boldsymbol{h}} + \xi\right) f_\xi(\xi) d\xi \\ &= \int_{\frac{\theta_m - [\boldsymbol{W}^{\mathrm{o}}]^{(m)}\overline{\boldsymbol{h}}}{\sigma_\xi}}^{\infty} \frac{1}{\sqrt{2\pi}} \exp\Big(-\frac{\xi_0^2}{2}\Big) d\xi_0.\end{aligned} \tag{5}$$

It is seen that the outputs $\overline{h}_k$ in Eq. (4) and $\overline{y}_m$ in Eq. (5) become differentiable, and the backpropagation algorithm is applicable to the training of the neural network. However, it is noted that, besides the weight matrices and the threshold parameters, two new parameters of noise levels σ_η and σ_ξ are introduced and need to be updated in the on-line training process.

Let $\mathcal{S} = \{\boldsymbol{x}(\ell), \boldsymbol{d}(\ell)\}_{\ell=1}^{L}$ denote the training set, which consists of L samples $\boldsymbol{x}(\ell)$ and L desired classifications $\boldsymbol{d}(\ell)$ for training the network in the way of supervised learning. Here, the loss function is chosen as the total cross entropy

$$\mathcal{H}_{\text{tot}} = \sum_{\ell=1}^{\mathcal{L}} \mathcal{H}(\ell) = \sum_{\ell=1}^{\mathcal{L}}\sum_{m=1}^{M} -\Big\{d_m(\ell)\ln\big[\overline{y}_m(\ell)\big] + \big[1 - d_m(\ell)\big]\ln\big[1 - \overline{y}_m(\ell)\big]\Big\}, \tag{6}$$

where the cross entropy $\mathcal{H}(\ell)$ measures the log-likelihood between the network output vector $\overline{\boldsymbol{y}} = [\overline{y}_1, \overline{y}_2, \dots, \overline{y}_M]^\top$ and the desired classification $\boldsymbol{d} = [d_1, d_2, \dots, d_M]^\top$ at the ℓ-th training epoch [27,28]. Compared with the quadratic cost of the mean square error (MSE), the cross entropy function is much more sensitive to the error between the network output and the desired output, and tends toward zero as the neuron gets better at computing the desired output for all training inputs.

Using Eqs. (4)–(6), the partial derivative of $\mathcal{H}(\ell)$ with respect to the noise level σ_ξ of the output layer can be computed as

$$\begin{aligned}\frac{\partial\mathcal{H}(\ell)}{\partial\sigma_\xi} &= \sum_{m=1}^{M}\frac{\partial\mathcal{H}(\ell)}{\partial\overline{y}_m(\ell)}\frac{\partial\overline{y}_m(\ell)}{\partial\sigma_\xi} \\ &= \sum_{m=1}^{M}\Big[\frac{1 - d_m(\ell)}{1 - \overline{y}_m(\ell)} - \frac{d_m(\ell)}{\overline{y}_m(\ell)}\Big]\frac{\theta_m - [\boldsymbol{W}^{\mathrm{o}}]^{(m)}\overline{\boldsymbol{h}}}{\sqrt{2\pi}\sigma_\xi^2}\exp\Big[-\frac{(\theta_m - [\boldsymbol{W}^{\mathrm{o}}]^{(m)}\overline{\boldsymbol{h}})^2}{2\sigma_\xi^2}\Big].\end{aligned} \tag{7}$$

Similarly, we have partial derivatives

$$\begin{aligned}\frac{\partial\mathcal{H}(\ell)}{\partial\theta_m} &= \frac{\partial\mathcal{H}(\ell)}{\partial\overline{y}_m(\ell)}\frac{\partial\overline{y}_m(\ell)}{\partial\theta_m} \\ &= \Big[\frac{d_m(\ell)}{\overline{y}_m(\ell)} - \frac{1 - d_m(\ell)}{1 - \overline{y}_m(\ell)}\Big]\frac{1}{\sqrt{2\pi}\sigma_\xi}\exp\Big[-\frac{(\theta_m - [\boldsymbol{W}^{\mathrm{o}}]^{(m)}\overline{\boldsymbol{h}})^2}{2\sigma_\xi^2}\Big],\end{aligned} \tag{8}$$

Algorithm 1: Backpropagation learning based on SSR.

```
Input: Training set $\{\boldsymbol{x}(\ell), \boldsymbol{d}(\ell)\}_{\ell=1}^{L}$, initial weight matrices $\boldsymbol{W}^{\mathrm{I}}(0)$, $\boldsymbol{W}^{\mathrm{o}}(0)$ and the initial noise levels $\sigma_\eta(0)$, $\sigma_\xi(0)$, the epoch number $P$, the learning rate $\alpha$.
Output: Trained parameter $\Theta \in \{[\boldsymbol{W}^{\mathrm{o}}]_{mk}, [\boldsymbol{W}^{\mathrm{I}}]_{kn}, \sigma_\eta, \sigma_\xi, \theta_k, \theta_m\}$.
1  for training epoch $p = 1 \to P$ do
2      $\mathcal{H}_{\mathrm{tot}} \leftarrow 0$;
3      for training example $\ell = 1 \to L$ do
4          $\overline{h}_k(\ell) \leftarrow \mathrm{E}_\eta\left[g_k\left([\boldsymbol{W}^{\mathrm{I}}]^{(k)}(\ell-1)\boldsymbol{x}(\ell)+\eta\right)\right]$;
5          $\overline{y}_m(\ell) \leftarrow \mathrm{E}_\xi\left[g_m\left([\boldsymbol{W}^{\mathrm{o}}]^{(m)}(\ell-1)\overline{\boldsymbol{h}}(\ell)+\xi\right)\right]$;
6          $\mathcal{H}(\ell) \leftarrow -\sum_{m=1}^{M}\left\{d_m(\ell)\ln\left[\overline{y}_m(\ell)\right]+\left[1-d_m(\ell)\right]\ln\left[1-\overline{y}_m(\ell)\right]\right\}$;
7          $\Theta(\ell) \leftarrow \Theta(\ell-1) - \alpha\frac{\partial\varepsilon(\ell)}{\partial\Theta}\Big|_{\Theta=\Theta(\ell-1)}$;
8          $\mathcal{H}_{\mathrm{tot}} \leftarrow \mathcal{H}_{\mathrm{tot}} + \mathcal{H}(\ell)$.
9      end
10     $\Theta(0) \leftarrow \Theta(L)$;
11 end
```

$$\begin{aligned}
\frac{\partial\mathcal{H}(\ell)}{\partial[\boldsymbol{W}^{\mathrm{o}}]_{mk}} &= \frac{\partial\mathcal{H}(\ell)}{\partial\overline{y}_m(\ell)}\frac{\partial\overline{y}_m(\ell)}{\partial[\boldsymbol{W}^{\mathrm{o}}]_{mk}} \\
&= \left[\frac{1-d_m(\ell)}{1-\overline{y}_m(\ell)} - \frac{d_m(\ell)}{\overline{y}_m(\ell)}\right]\frac{\overline{h}_k}{\sqrt{2\pi}\sigma_\xi}\exp\left[-\frac{(\theta_m-[\boldsymbol{W}^{\mathrm{o}}]^{(m)}\overline{\boldsymbol{h}})^2}{2\sigma_\xi^2}\right], \qquad (9)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial\mathcal{H}(\ell)}{\partial\sigma_\eta} &= \sum_{k=1}^{K}\sum_{m=1}^{M}\frac{\partial\mathcal{H}(\ell)}{\partial\overline{y}_m(\ell)}\frac{\partial\overline{y}_m(\ell)}{\partial\overline{h}_k}\frac{\partial\overline{h}_k}{\partial\sigma_\eta} \\
&= \sum_{k=1}^{K}\sum_{m=1}^{M}\left[\frac{1-d_m(\ell)}{1-\overline{y}_m(\ell)} - \frac{d_m(\ell)}{\overline{y}_m(\ell)}\right]\frac{[\boldsymbol{W}^{\mathrm{o}}]_{mk}}{\sqrt{2\pi}\sigma_\xi}\exp\left[-\frac{(\theta_m-[\boldsymbol{W}^{\mathrm{o}}]^{(m)}\overline{\boldsymbol{h}})^2}{2\sigma_\xi^2}\right] \\
&\quad\times\frac{\theta_k-[\boldsymbol{W}^{\mathrm{I}}]^{(k)}\boldsymbol{x}}{\sqrt{2\pi}\sigma_\eta^2}\exp\left[-\frac{(\theta_k-[\boldsymbol{W}^{\mathrm{I}}]^{(k)}\boldsymbol{x})^2}{2\sigma_\eta^2}\right], \qquad (10)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial\mathcal{H}(\ell)}{\partial[\boldsymbol{W}^{\mathrm{I}}]_{kn}} &= \sum_{m=1}^{M}\frac{\partial\mathcal{H}(\ell)}{\partial\overline{y}_m(\ell)}\frac{\partial\overline{y}_m(\ell)}{\partial\overline{h}_k}\frac{\partial\overline{h}_k}{\partial[\boldsymbol{W}^{\mathrm{I}}]_{kn}} \\
&= \frac{x_n}{\sqrt{2\pi}\sigma_\eta}\exp\left[-\frac{(\theta_k-[\boldsymbol{W}^{\mathrm{I}}]^{(k)}\boldsymbol{x})^2}{2\sigma_\eta^2}\right]\sum_{m=1}^{M}\left[\frac{1-d_m(\ell)}{1-\overline{y}_m(\ell)} - \frac{d_m(\ell)}{\overline{y}_m(\ell)}\right] \\
&\quad\times\frac{[\boldsymbol{W}^{\mathrm{o}}]_{mk}}{\sqrt{2\pi}\sigma_\xi}\exp\left[-\frac{(\theta_m-[\boldsymbol{W}^{\mathrm{o}}]^{(m)}\overline{\boldsymbol{h}})^2}{2\sigma_\xi^2}\right] \qquad (11)
\end{aligned}$$

and

$$\begin{aligned}
\frac{\partial\mathcal{H}(\ell)}{\partial\theta_k} &= \sum_{m=1}^{M}\frac{\partial\mathcal{H}(\ell)}{\partial\overline{y}_m(\ell)}\frac{\partial\overline{y}_m(\ell)}{\partial\overline{h}_k}\frac{\partial\overline{h}_k}{\partial\theta_k} \\
&= \sum_{m=1}^{M}\left[\frac{d_m(\ell)}{\overline{y}_m(\ell)} - \frac{1-d_m(\ell)}{1-\overline{y}_m(\ell)}\right]\frac{[\boldsymbol{W}^{\mathrm{o}}]_{mk}}{\sqrt{2\pi}\sigma_\xi}\exp\left[-\frac{(\theta_m-[\boldsymbol{W}^{\mathrm{o}}]^{(m)}\overline{\boldsymbol{h}})^2}{2\sigma_\xi^2}\right] \\
&\quad\times\frac{1}{\sqrt{2\pi}\sigma_\eta}\exp\left[-\frac{(\theta_k-[\boldsymbol{W}^{\mathrm{I}}]^{(k)}\boldsymbol{x})^2}{2\sigma_\eta^2}\right], \qquad (12)
\end{aligned}$$

where $[\boldsymbol{W}^{\mathrm{o}}]_{mk}$ is the weight that neuron m in the output layer applies to the k-th hidden unit $\overline{h}_k$, and $[\boldsymbol{W}^{\mathrm{I}}]_{kn}$ denotes the weight connecting the hidden unit $\overline{h}_k$ with the n-th element x_n of the input vector $\boldsymbol{x}$.

Combining Eqs. (7)–(12), we can write the learning rule of parameters $\Theta \in \{\sigma_\xi, [\mathbf{W}^{\mathrm{o}}]_{mk}, [\mathbf{W}^{\mathrm{I}}]_{kn}, \sigma_\eta, \theta_k, \theta_m\}$ as

$$\Theta(\ell) = \Theta(\ell-1) - \alpha\frac{\partial\mathcal{H}(\ell)}{\partial\Theta}\Big|_{\Theta=\Theta(\ell-1)} \qquad (13)$$

for the ℓ-th training epoch and the learning rate $\alpha > 0$. To summarize the modified backpropagation learning algorithm for the considered threshold neural network, the pseudo code of the algorithm is shown in Algorithm 1.

3. Training and testing results of the threshold network

3.1. Data pattern classification

The proposed backpropagation algorithm indicated in Eq. (13) is applied to a classical data classification problem [28], as shown in Fig. 2. There is a pair of regions A and B facing each other in an asymmetrically arranged manner, which represents two data patterns.

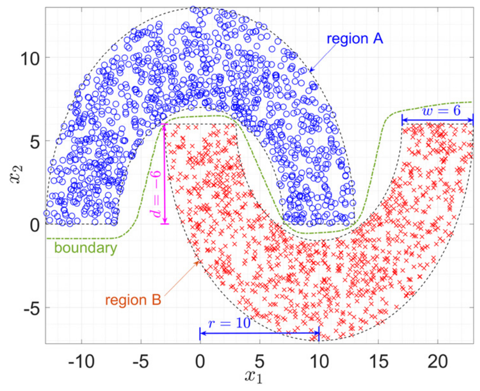

Fig. 2. Results of the computer experiment on the backpropagation algorithm applied to the feedforward threshold network for data classification. The training data are with distance $d = -6$, radius $r = 10$ and width $w = 6$ [28]. The decision boundary is determined by finding the coordinates x_1 and x_2 of the input vector $\boldsymbol{x}$, for which the response $\overline{y}_m$ of the output neuron exceeds 0.5 for classifying the input vector $\boldsymbol{x}$ into the region A, and otherwise, the input belongs to the region B.

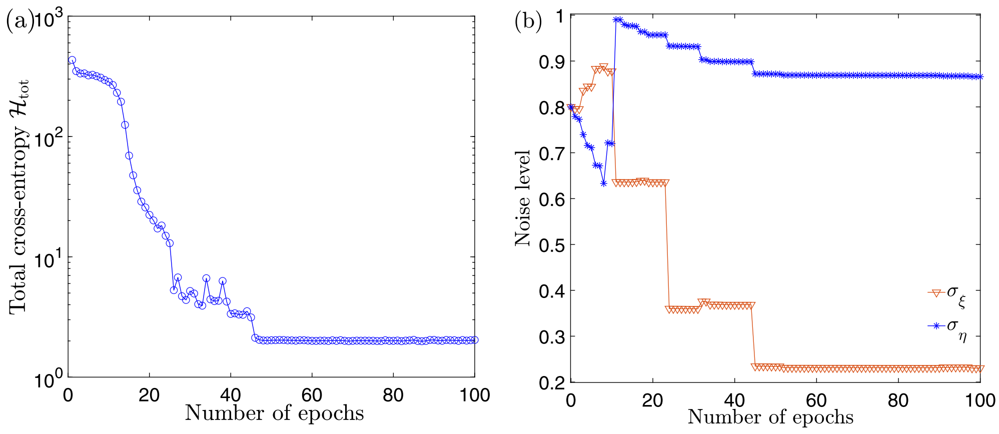

Fig. 3. Learning curves of (a) the total cross entropy $\mathcal{H}_{\rm tot}$ and (b) the added noise levels σ_ξ and σ_η for training the feedforward neural network to the data classification problem indicated in Fig. 2.

Two regions have the identical radius $r = 10$ and width $w = 6$, and the data are uniformly scattered over each region. The data points are represented by the coordinates x_1 and x_2 that form the input vector $\boldsymbol{x}$ of the neural network. The vertical distance $d = -6$ separates two regions with respect to the x_2 axis, and the smaller d means the larger area the two regions overlap, resulting into a difficult nonlinear separability problem [28]. The coordinate x_2 distributes in region A over $[0, 13]$ and in region B over $[-7, 6]$ when $d = -6$, so that the two classes overlap for $x_2 \in [0, 6]$.

For training the feedforward threshold neural network in Fig. 1 by the backpropagation algorithm of Eq. (13), we consider the training samples consisting of 10^3 pairs of data points, i.e. one point picked from the region A and another point picked from the region B. Before entering into the network, the coordinates x_1 and x_2 are both normalized by

$$\widetilde{x}_n(\ell) = \frac{x_n(\ell) - \sum_\ell^L x_n(\ell)/L}{\max\{|x_n(\ell)|\}}, \; n = 1, 2. \tag{14}$$

Here, the feedforward threshold network is with $N = 2$ input neurons receiving the normalized coordinates $\widetilde{x}_1$ and $\widetilde{x}_2$, respectively, $K = 20$ hidden neurons $\overline{h}_k$ and $M = 1$ output neuron $\overline{y}_1$. The learning rate in Eq. (13) is $\alpha = 0.1$, the initial noise level $\sigma_\eta(0) = \sigma_\xi(0) = 0.8$, and the initial weight matrixes $\boldsymbol{W}^{\rm I}(0)$ and $\boldsymbol{W}^{\rm o}(0)$ are uniformly distributed in the interval $[-0.1, 0.1]$. It is seen in Fig. 3 (a) that the learning curve of the total cross entropy $\mathcal{H}_{\rm tot}$ reaches convergence as $\mathcal{H}_{\rm tot}(50) = 2.05$ effectively in about 50 epochs of training. Moreover, it is shown in Fig. 3 (b) that the noise levels $\sigma_\eta(100) = 0.87$ and $\sigma_\xi(100) = 0.23$ converge to non-zero values respectively, which also demonstrates the benefits of noise to the training of the feedforward threshold network. Since the neuron output $y_m \in [0, 1]$ in Eq. (3), the decision boundary is determined by finding the coordinates x_1 and x_2 of the input vector $\boldsymbol{x}$, for which the response y_m of the output neuron exceeds 0.5 for classifying the input vector $\boldsymbol{x}$ into the region A, and otherwise, the input belongs to the region B, as shown in Fig. 2.

In the testing experiments, the trained feedforward threshold network is employed with the converged parameters $\Theta \in \{[\mathbf{W}^{\rm o}]_{mk}, [\mathbf{W}^{\rm I}]_{kn}, \theta_k, \theta_m\}$ obtained after the 100 training epochs. However, the neuron outputs h_k of Eq. (2) in the hidden layer and the output y_m of Eq. (3) in the output layer are calculated with a finite number of T threshold elements, respectively. It is noted that the added Gaussian noise components η_{kt} in h_k are mutually independent and with the common level $\sigma_\eta(100) = 0.87$, and the mutually independent Gaussian noise components ξ_{mt} in y_m have the common level $\sigma_\xi(100) = 0.23$ for $t = 1, 2, \ldots, T$. For different numbers T, the testing results of classification errors $N_{\rm error}$ in a testing set of 2×10^3 pairs of data points are experimentally obtained for 10^2 trials, represented by an error rate of $N_{\rm error}/2000$ percent, as shown in Fig. 4. It is shown in Fig. 4 that, when the number of threshold elements $T \geq 450$, the error rate reaches zero, which is better than the error rate 1% obtained by the network with the same size of $N \times K \times M = 2 \times 20 \times 1$

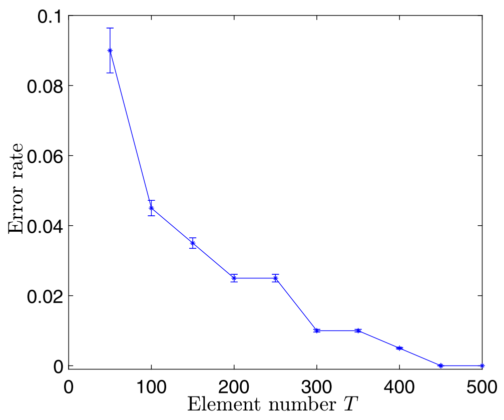

Fig. 4. Classification error rate in a testing set of 2×10^3 pairs of data points versus the number T of threshold elements. Here, 10^2 trials are realized for each point of experimental results.

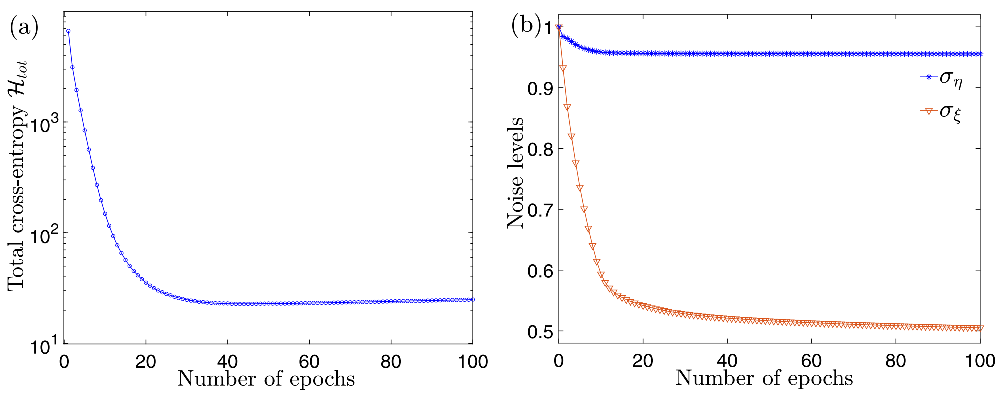

Fig. 5. Learning curves of (a) the total cross entropy $\mathcal{H}_{\text{tot}}$ and (b) noise levels σ_η and σ_ξ versus the number of epochs for training the feedforward neural network to recognize handwritten digits.

but hyperbolic tangent activation functions [28]. This result also demonstrates the beneficial effects of added noise in the threshold neural network for the data pattern classification problem.

3.2. Recognition of handwritten digits

We further apply the proposed backpropagation learning algorithm in Eq. (13) to train the feedforward threshold neural network for recognizing handwritten digits. The data set used to train the designed network is the MNIST handwritten image database, which contains 6×10^4 images used for training and 10^4 images for testing. Here, we employ 10^4 images with the training set and test set in a $4:1$ ratio. Each black-and-white handwritten image has 28×28 pixels and so can be mapped into a 784×1 input vector $\boldsymbol{x}$ for the feedforward neural network. The categories of the digits are expressed by the target vector set $\boldsymbol{d} = \{d_i\}$ for $i = 0, 1, \ldots, 9$. For instance, the vector $d_0 = [0, 0, 0, 0, 0, 0, 0, 0, 0, 1]^\top$ represents the digit 0, the vector $d_1 = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0]^\top$ indicates the digit 1, the vector $d_2 = [0, 1, 0, 0, 0, 0, 0, 0, 0, 0]^\top$ signifies the digit 2, and so on. The learning rate $\alpha = 0.1$ in Eq. (13) for the noise levels σ_ξ and σ_η, but $\alpha = 0.02$ for other network parameters. The initial noise levels $\sigma_\eta(0) = \sigma_\xi(0) = 1$. The initial weights $[\boldsymbol{W}^1]_{kn}$, $[\boldsymbol{W}^o]_{mk}$, the thresholds θ_k and θ_m are uniformly distributed in the interval $[-0.1, 0.1]$. The designed feedforward neural network is with the layer size of $N \times K \times M = 784 \times 100 \times 10$.

Using the proposed backpropagation learning algorithm in Eq. (13) and 8×10^3 handwritten images for training the designed network, we obtain the total cross entropy $\mathcal{H}_{\text{tot}}(100) = 24.96$, the converged noise levels $\sigma_\eta(100) = 0.96$ and $\sigma_\xi(100) = 0.51$ after 100 training epochs, as shown in Fig. 5 (a) and (b), respectively. Then, the test set containing 2×10^3 handwritten images is used to test the generalization ability of the trained neural network. Note that for testing, the network is operated by the injection of mutually independent Gaussian noise components η_{kt} in hidden neuron h_k at the noise level $\sigma_\eta(100)$ and ξ_{mt} in output neuron y_m at the noise level $\sigma_\xi(100)$, respectively. The number of threshold elements $T = 10^2$. In terms of precision, an accuracy rate up to 97% on the test set is obtained, which remains competitive with the results achieved by other classification networks, such as the convolutional neural network of the size $784 \times 1000 \times 10$ with the accuracy rate 95.5% [29,30].

4. Conclusion

In this paper, aiming to implement a feedforward threshold neural network for pattern classification, we intentionally add noise into a large-scale summing array of threshold elements. The reason is the asymptotical approximation of a large-scale summing array of threshold elements can be viewed as a differentiable noise-smoothed activation function giving rise to nonlinear neuron behavior for the feedforward neural network. As a consequence, during the training phase, the added noise sources, as well as the weights, are

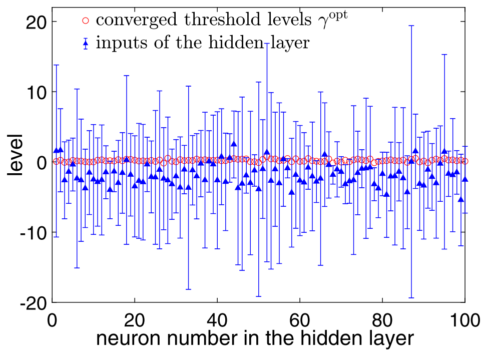

Fig. 6. Comparison of the input signals of the hidden layer and the converged threshold levels γ^{opt}.

treated as adjustable parameters that are adaptively learned. Moreover, the learning process converges to non-zero optimal added noise levels, establishing a beneficial role for noise on the operation of the network. The trained neural network is operated and activated by mutually independent noise components at run-time. Experimental results show that the feedforward threshold neural network trained by the proposed backpropagation algorithm can achieve state-of-the-art accuracy on two benchmark data sets. In the classification 2×10^3 handwritten digits, the input signals of the hidden layer and the converged threshold levels for 10^2 neurons are plotted in Fig. 6, respectively. The mechanism of SSR naturally arises in the optimal configuration obtained in Fig. 6, with the stochastic inputs of each neuron in the hidden layer that are not always less but partly larger than the corresponding converged threshold levels. Moreover, as indicated in Fig. 4, the beneficial effects of noise in the neural networks for pattern classification become more efficient as the element number increases. This conclusion reveals the beneficial effects of noise in the neural networks for pattern classification, which can be attributed to the SSR mechanism. In addition, the obtained results also extend the applicability of the backpropagation algorithm to train neural networks with a much wider family of non-differentiable activation functions by artificially introducing the random noise into them.

As an extension to this study, it may be of interest to further investigate the possibility and application of the pattern classification algorithm based on the stochastic resonance mechanism when the activation functions in neural networks employ other typical functions, such as Sigmoid, Rectified linear unit, Softmax and so on. Generally, the activation functions can be classified into two categories: with saturation (e.g. Sigmoid) and without saturation (e.g. Rectified Linear unit). For a parallel array of neurons with saturation, it has been found that the stochastic resonance effect can be reinforced by a sufficient amount of added noise in the intermediate and the saturation parts of the neuronal response [31]. Therefore, the effects of noise in the neural networks with both saturated and unsaturated activation functions can potentially be further investigated for additional benefits. Naturally, the pattern classification algorithm based on the stochastic resonance mechanism can be extended to such conditions.

CRediT authorship contribution statement

Fabing Duan (Corresponding author): Conceptualization, Methodology, Creation of models.
Xiaojie Liu and Lingling Duan: Data curation, Programming, Writing-Original draft preparation.
François Chapeau-Blondeau and Derek Abbott: Investigation, Validation, Writing-Reviewing and Editing.

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.

Acknowledgements

This work is supported in part by the Taishan Scholar Project of Shandong Province of China (No. ts20190930). Fabing Duan and Lingling Duan have no relationship.

References

[1] S. Hanson, A stochastic version of the delta rule, Physica D 42 (1–3) (1990) 265–272.
[2] D. Toms, Training binary node feedforward neural networks by back propagation of error, Electron. Lett. 26 (21) (1990) 1745–1746.
[3] P. Bartlett, T. Downs, Using random weights to train multilayer networks of hard-limiting units, IEEE Trans. Neural Netw. 3 (2) (1992) 202–210.
[4] 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.
[5] E. Wilson, S. Rock, Gradient-based parameter optimization for systems containing discrete-valued functions, Int. J. Robust Nonlinear Control 12 (9) (2002) 1009–1028.
[6] 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.
[7] N. Frazier-Logue, S. Hanson, The stochastic delta rule: faster and more accurate deep learning through adaptive weight noise, Neural Comput. 32 (5) (2020) 1018–1032.
[8] S. Ikemoto, F. Dallalibera, K. Hosoda, Noise-modulated neural networks as an application of stochastic resonance, Neurocomputing 277 (2) (2018) 29–37.
[9] M. Plappert, R. Houthooft, P. Dhariwal, S. Sidor, R. Chen, X. Chen, T. Asfour, P. Abbeel, M. Andrychowicz, Parameter space noise for exploration, in: 6-th International Conference on Learning Representations (ICLR), Vancouver, BC, Canada, 2018, https://arxiv.org/pdf/1706.01905.pdf.
[10] N.G. Stocks, Suprathreshold stochastic resonance in multilevel threshold systems, Phys. Rev. Lett. 84 (11) (2000) 2310–2313.
[11] N. Stocks, Information transmission in parallel threshold arrays: suprathreshold stochastic resonance, Phys. Rev. E 63 (4) (2001) 041114.
[12] 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.

[13] J. Zhu, C. Kong, X. Liu, Subthreshold and suprathreshold vibrational resonance in the Fitzhugh-Nagumo neuron model, Phys. Rev. E 94 (3) (2016) 032208.
[14] 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.
[15] A. Gershman, J. Böhme, A pseudo-noise approach to direction finding, Signal Process. 71 (1) (1998) 1–13.
[16] S. Uhlich, Bayes risk reduction of estimators using artificial observation noise, IEEE Trans. Signal Process. 63 (20) (2015) 5535–5545.
[17] F. Duan, Y. Pan, F. Chapeau-Blondeau, D. Abbott, Noise benefits in combined nonlinear Bayesian estimators, IEEE Trans. Signal Process. 67 (17) (2019) 4611–4623.
[18] V. Hari, G. Anand, A. Premkumar, A. Madhukumar, Design and performance analysis of a signal detector based on suprathreshold stochastic resonance, Signal Process. 92 (7) (2012) 1745–1757.
[19] J. Liu, B. Hu, Y. Wang, Optimum adaptive array stochastic resonance in noisy grayscale image restoration, Phys. Lett. A 383 (13) (2019) 1457–1465.
[20] R. Liu, Y. Kang, Stochastic resonance in underdamped periodic potential systems with alpha stable Lévy noise, Phys. Lett. A 382 (25) (2018) 1656–1664.
[21] F. Duan, F. Chapeau-Blondeau, D. Abbott, Weak signal detection: condition for noise induced enhancement, Digit. Signal Process. 23 (5) (2013) 1585–1591.
[22] D. Guo, M. Perc, T. Liu, D. Yao, Functional importance of noise in neuronal information processing, Europhys. Lett. 124 (5) (2018) 50001.
[23] W. Li, K. Hirota, Y. Dai, Z. Jia, An improved fully convolutional network based on post-processing with global variance equalization and noise-aware training for speech enhancement, J. Adv. Comput. Intell. Intell. Inform. 25 (1) (2021) 130–137.
[24] S. Lu, G. Qian, Q. He, F. Liu, Y. Liu, Q. Wang, In situ motor fault diagnosis using enhanced convolutional neural network in an embedded system, IEEE Sens. J. 20 (15) (2020) 8287–8296.
[25] B. Kosko, K. Audhkhasi, O. Osoba, Noise can speed backpropagation learning and deep bidirectional pretraining, Neural Netw. 129 (2020) 359–384.
[26] Y. Fu, Y. Kang, G. Chen, Stochastic resonance based visual perception using spiking neural networks, Front. Comput. Neurosci. 14 (2020) 24.
[27] C. Bishop, Pattern Recognition and Machine Learning, Springer, New York, 2006.
[28] S. Haykin, Neural Networks and Learning Machines, Prentice Hall, New York, 2009.
[29] Y. LeCun, L. Bottou, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[30] A. Neelakantan, L. Vilnis, V. Sutskever, L. Kaiser, K. Kurach, J. Martens, Adding gradient noise improves learning for very deep networks, in: 4-th International Conference on Learning Representation, San Juan, Puerto Rico, 2016, pp. 1–11.
[31] S. Blanchard, D. Rousseau, F. Chapeau-Blondeau, Noise enhancement of signal transduction by parallel arrays of nonlinear neurons with threshold and saturation, Neurocomputing 71 (2007) 333–341.