Journal of Computers Vol. 29 No. 1, 2018, pp. 76-85
doi:10.3966/199115992018012901007

Effects of BP Algorithm-based Activation Functions on Neural Network Convergence

Junguo Hu[1], Lili Xu[1], Xin Wang[2], Xiaojun Xu[3*], Guangyun Su[1]

[1] 1 School of Information Technology, Zhejiang A & F University, Lin'an, 311300, China
hawkhjg@163.com, gdyashilandai@163.com, 276746009@qq.com

[2] Beijing LiuYi Biotechnology CO., LTD, Beijing, 100070, China
59336807@qq.com

[3] School of Environmental & Resource Sciences, Zhejiang A & F University, Lin'an, 311300, China
xuxiaojun3115371@163.com

Received 21 September 2016; Revised 3 March 2017; Accepted 3 April 2017

Abstract. Activation functions map data in artificial neural network computation. In an application, the activation function and selection of its gradient and translation factors are directly related to the convergence of the network. Usually, the activation function parameters are determined by trial and error. In this work, a Cauchy distribution (Cauchy), Laplace distribution (Laplace), and Gaussian error function (Erf) were used as new activation functions for the back-propagation (BP) algorithm. In addition, this study compares the effects of the Sigmoid type function (Logsig), hyperbolic tangent function (Tansig), and normal distribution function (Normal). The XOR problem was used in simulation experiments to evaluate the effects of these six kinds of activation functions on network convergence and determine their optimal gradient and translation factors. The results show that the gradient factor and initial weights significantly impact the convergence of activation functions. The optimal gradient factors for Laplace, Erf-Logsig, Tansig-Logsig, Logsig, and Normal were 0.5, 0.5, 4, 2, and 1, respectively, and the best intervals were [0.5, 1], [0.5, 2], [2, 6], [1, 4], and [1, 2], respectively. Using optimal gradient factors, the order of convergence speed was Laplace, Erf-Logsig, Tansig-Logsig, Logsig, and Normal. The functions Logsig (gradient factor = 2), Tansig-Logsig (gradient factor = 4), Normal (translation factor = 0, gradient factor = 1), Erf-Logsig (gradient factor = 0.5) and Laplace (translation factor = 0, gradient factor = 0.5) were less sensitive to initial weights, and as a result, their convergence performances were less influenced. As the gradient of the curve of the activation functions increased, the convergence speed of the networks showed an accelerating trend. The conclusions obtained from the simulation analysis can be used as a reference for the selection of activation functions for BP algorithm-based feedforward neural networks.

Keywords: activation functions, back-propagation (BP) algorithm, convergence, gradient factor, initial weights

1 Introduction

The feedforward neural network (FNN) is a widely-used artificial neural network model [1-3], especially the multilayer perceptron learning algorithm trained with error back propagation (BP) [4-5]. Research on the BP algorithm-based FNN has made tremendous progress, but also has exposed some important issues [4, 6], such as (1) it is easy to fall into local minima, (2) the learning algorithm converges very slowly, (3) there is no conclusive scientific theory to guide the selection of network hidden nodes, and (4) the generalization of networks with good learning ability is poor.

* Corresponding Author

To solve these problems, many scholars have proposed improvement strategies, which can be divided into the following categories. The first category is the use of heuristic techniques, such as the additional momentum [7], adaptive learning rate, and elastic feedback algorithms. The second is the use of numerical optimization techniques, such as the conjugate gradient, quasi-Newton, and Levenberg-Marquardt [8] methods. Riedmiller and Braun proposed the resilient BP algorithm in 1993, which is a local adaptive learning method that implements supervised batch learning in multilayer perceptrons [9-10]. The third category is the use of intelligent weight optimization. By utilizing a genetic algorithm to optimize the weights of a neural network, Whitley et al. [11] and Kuo et al. [12] accelerated the convergence speed of the network and reduced the probability of falling into a local minimum. The fourth category comprises the approaches that use new activation functions and their combinations [13-15]. It is generally believed that an activation function should be continuous, differentiable, and monotonically increasing [16]. However, Hornik theoretically proved that any continuous, bounded, non-constant function can be used as an activation function [17]. It has also been proven that the activation function plays an important role in network convergence.

In this paper, the selection of new activation functions was premised on the selection criteria and based on the range and convergence speed of the function itself and its derivative. The following three functions were selected as new activation functions: Cauchy distribution (Cauchy), Laplace distribution (Laplace), and Gaussian error function (Erf). They were compared with the Sigmoid type (Logsig), hyperbolic tangent (Tansig), and normal distribution (Normal) functions in simulations experiments using the classic XOR problem. The network convergence for all six activation functions was analyzed under different gradient factor and initial weight conditions. The optimal activation function with optimal parameter values and ranges was determined.

2 Related Work

An activation function maps the data in artificial neural network computation and is hence the most important function in neural network processing [18-20]. Currently, the linear, hard-limiting, hyperbolic tangent, and Sigmoid type functions are the most commonly used activation functions [21].In addition, the normal, exponential, periodic, translation, and wavelet functions can also be used as activation functions of neural networks.

Previous works have proposed many new and effective activation functions and proven that changes in activation functions can increase the convergence speed of the network. For example, Kenue proposed a new activation function with its first derivative as $\sec h^n(x), n = 1,2,\cdots$. Comparing it with the standard Sigmoid function, he found that the new activation function can significantly improve the convergence speed of the network [22]. Nambiar et al. proposed a novel and cost efficient sigmoid-like activation function for an evolvable block-based neural network, which executes up to 410 times faster than embedded software [15]. Miao et al. [23] proposed a new neuron model of an adjustable activation function.

The main contributions of this paper lie in the following two aspects.

Firstly, three new activation functions are proposed, which are well applied in the BP algorithm-based FNN and are superior to common activation functions. These new activation functions increase the convergence speed of the BP algorithm-based FNN.

Secondly, activation functions map data in artificial neural network computation, but are usually determined by trial and error. In this work, the optimal gradient factors and their ranges for these six activation functions are determined, which provides a reference for designers of back-propagation feedforward neural networks.

3 Materials and Methods

3.1 Activation Functions and Their First Derivatives

Normal, Cauchy, Laplace, and Erf satisfy the differentiable, bounded, and non-constant conditions, and can give a bounded output with bounded input. Thus, the above functions can all be used as a BP algorithm-based activation function for feedforward neural networks. Table 1 presents the expressions

for the above four activation functions, Logsig, Tansig, and their first derivatives.

Table 1. Six activation functions and their first derivatives

Activation function	$f(x)$	$f'(x)$	Parameter
Logsig	$\frac{1}{1+\exp(-\lambda x)}$	$\lambda f(x)(1-f(x))$	λ
Tansig	$\frac{1-\exp(-\lambda x)}{1+\exp(-\lambda x)}$	$\frac{\lambda}{2}(1-f(x)^2)$	λ
Normal	$\frac{1}{2}\left(1+erf(-\frac{x-\mu}{\sqrt{2}\lambda})\right)$	$\frac{1}{\sqrt{2\pi}}\exp(-\frac{(x-\mu)^2}{2\lambda^2})$	μ, λ
Cauchy	$\frac{1}{\pi}\arctan(\frac{x-u}{\lambda})+\frac{1}{2}$	$\frac{1}{\pi}\left(\frac{\lambda}{(x-u)^2+\lambda^2}\right)$	μ, λ
Erf	$erf(\frac{x}{\sqrt{\lambda}})$	$\frac{2}{\sqrt{\pi\lambda}}\exp(-\frac{x^2}{\lambda})$	λ
Laplace	$\begin{cases}\frac{1}{2}\exp(-\frac{u-x}{\lambda})\ ifx<u\\ 1-\frac{1}{2}\exp(-\frac{x-u}{\lambda})ifx\geq u\end{cases}$	$\frac{1}{2\lambda}\begin{cases}\exp(-\frac{u-x}{\lambda})\ ifx<u\\ \exp(-\frac{x-u}{\lambda})ifx\geq u\end{cases}$	μ, λ

Note: The Gaussian error function is $erf(x)=\frac{2}{\sqrt{2}}\int_0^x e^{-t^2}dt$ and its Taylor expansion is $erf(x)=\frac{2}{\sqrt{\pi}}\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n+1}}{(2n+1)n!}=\frac{2}{\sqrt{\pi}}(x-\frac{x^3}{3}+\frac{x^5}{10}-\frac{x^7}{42}+\frac{x^9}{216}-\cdots)$. Additionally, λ is the gradient factor, and μ is the translation factor.

3.2 Plots for Activation Functions and Their First Derivatives

To present a more intuitive understanding of the characteristics of the six activation functions, they and their first derivatives are plotted for different gradient factors in Fig. 1. As shown in Figs. 1(a) and (b), the range of output values of Logsig is [0, 1]. The output values of Tansig are in the range [−1, 1]. As the gradient factors increase, the function curves become steeper, the peak values of the corresponding derivative near zero gradually narrow, and the maxima increase. However, Normal, Cauchy, Laplace, Erf, and their corresponding derivative curves show the opposite changes with respect to various gradient factors. Figs. 1(c), (d), (e), and (f) show that the range of output values of Erf is [−1, 1], and the output ranges of the other four functions are [0, 1]. As the gradient factors are increased, these four function curves gradually slow, the corresponding peaks in the vicinity of the zero derivative gradually widen, and the maximum values gradually decrease.

3.3 Algorithms

In this work, only the standard activation function for the BP algorithm was replaced with Normal, Cauchy, Laplace, Erf, and their corresponding derivatives, therefore, the algorithm remains relatively simple. Taking Normal as an example, the algorithm process is described as follows. The algorithm training process uses online learning, which is a learning method that updates network weights for each mode (sample) within the training set. This method is characterized by its requirement for less storage in the learning process, but sometimes increases the overall output error of the network.

(1) The output node algorithm expressions for the hidden node output $y^{(k)}$ and node output $O^{(k)}$ are as follows:

$$y^{(k)}=\frac{1}{2}\left(1+erf(-\frac{X^{(k)}*IW-\mu}{\sqrt{2}\lambda})\right), \quad \textbf{(1)}$$

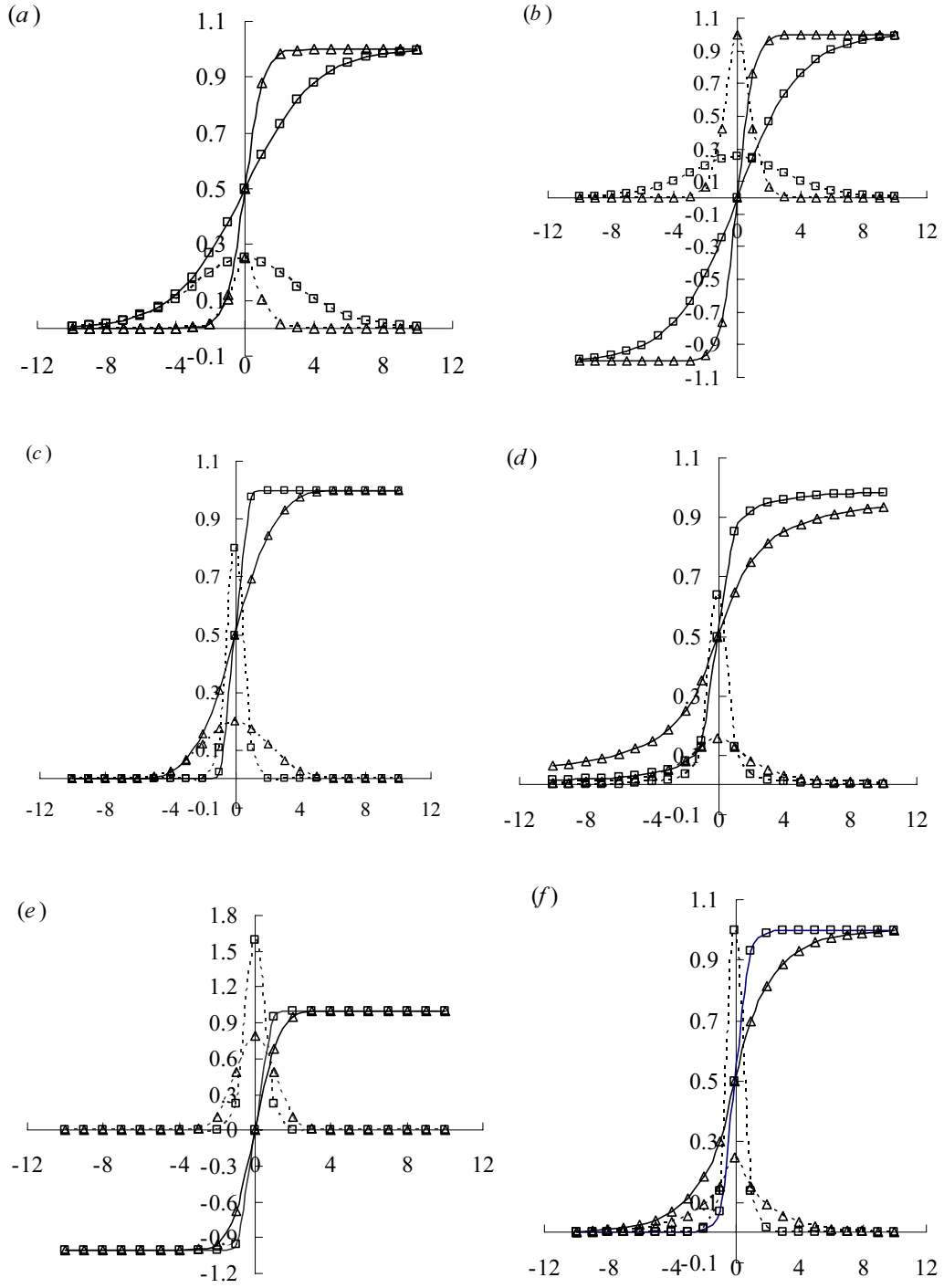

Fig. 1. Activation functions and their first derivatives: (a) Logsig, (b) Tansig, (c) Normal, (d) Cauchy, (e) Erf, and (f) Laplace. In all plots, □: λ=0.5, Δ: λ=2, solid lines indicate activation functions, and dashed lines indicate their first derivatives

$$O^{(k)} = \frac{1}{2}\left(1 + erf(-\frac{y^{(k)} * LW - \mu}{\sqrt{2}\lambda})\right), \tag{2}$$

where $X^{(k)}$ represents the k-th mode, $k = 1,2,\cdots n$, IW is the connection weight between input and hidden layers (including the threshold), LW is the connection weight between hidden and output layers (including the threshold), and μ and λ are the parameters for Normal.

(2) The error correction algorithm for the output layer (between the hidden and output nodes) uses the

formula

$$\delta^{(k)} = (t^{(k)} - O^{(k)}) * \frac{1}{\sqrt{2\pi}} \exp(-\frac{(y^{(k)} * LW - \mu)^2}{2\lambda^2}), \quad \textbf{(3)}$$

where $\delta^{(k)}$ represents the error correction for the k-th mode, and $t^{(k)}$ is the expected output for the k-th mode.

(3) The error correction algorithm for the hidden layer (between the input and hidden nodes) uses the formula

$$\varphi_{\bullet h}^{(k)} = \delta^{(k)} LW'_{h\bullet} * \frac{1}{\sqrt{2\pi}} \exp(-\frac{(\gamma_{\bullet h} - \mu)^2}{2\lambda^2}), \ h = 2,3,\cdots q+1, \quad \textbf{(4)}$$

$$\gamma = \mathrm{X}^{(k)} * IW, \quad \textbf{(5)}$$

where $\varphi_{\bullet h}^{(k)}$ is the error correction value for the *h*-th column of the *k*-th sample in the hidden layer, $LW'_{h\bullet}$ is the transpose of the *h*-th row of the connection weights between the hidden and output layers (including the threshold), and $\gamma_{\bullet h}$ is the *h*-th column of the integral of the *k*-th sample and IW.

4 Experimental Simulations

The classic XOR problem is used as an example to evaluate the convergence of the above-mentioned six activation functions. The same number of hidden layers, initial weights, learning rate, momentum factor, and training samples were used for these six activation functions. Ten groups of initial weights were randomly generated, ranging between [−1, 1]. This weight initialization was repeated ten times with different activation functions under different conditions and parameters. The initial weights of these ten groups are not presented here because of space limitations. The network structure includes a ratio of input, hidden, and output layers of 4:4:1; learning rate of 0.5; and momentum factor of 0.9. The network error performance function was set to 0.01 using sum of squared errors. In order to analyze the effect of the activation function gradient factor on the convergence rate and select an appropriate range, gradient factors of 0.1, 0.2, 0.3, 0.5, 1, 2, 4, and 6 were used. Zhou et al. [21] found in their experiments that the network oscillated and could not converge when the target value was within the range 0−1 when Tansig was used as the activation function for the output layer, which could be related to the range [−1, 1] for both the activation function and output value. Therefore, a combination of Tansig-Logsig and Erf-Logsig was used, with parameter λ =1 unchanged.

The number of epochs to convergence, the difference between the maximum and minimum convergence epochs for various gradient factor conditions, and the epoch variation coefficient for all conditions are used as indicators to evaluate the convergence of the activation functions. When the convergence epochs are less in number, the activation function converges faster; when the difference between the maximum and minimum epochs as well as the variation coefficient are smaller, the activation function is less sensitive to the initial weights. Based on the margin between the minimum and maximum epochs, it can be inferred that the different initial weights have a significant impact on the convergence speed of the network. A greater margin indicates that the activation function is more sensitive to the initial weights. The F test was used to evaluate the significance of the effect of different gradient factors on the convergence of the activation function.

5 Results and Discussion

The number of epochs to convergence for each activation function under different gradient factor conditions are shown in Fig. 2. The F test showed a very significant effect ($p < .01$) on the convergence speed of the activation function, which provides theoretical support to the argument that an adjustable activation function gradient factor can accelerate the convergence speed of the network. In an application, the gradient factor can be continuously adjusted to accelerate network convergence. Different gradient

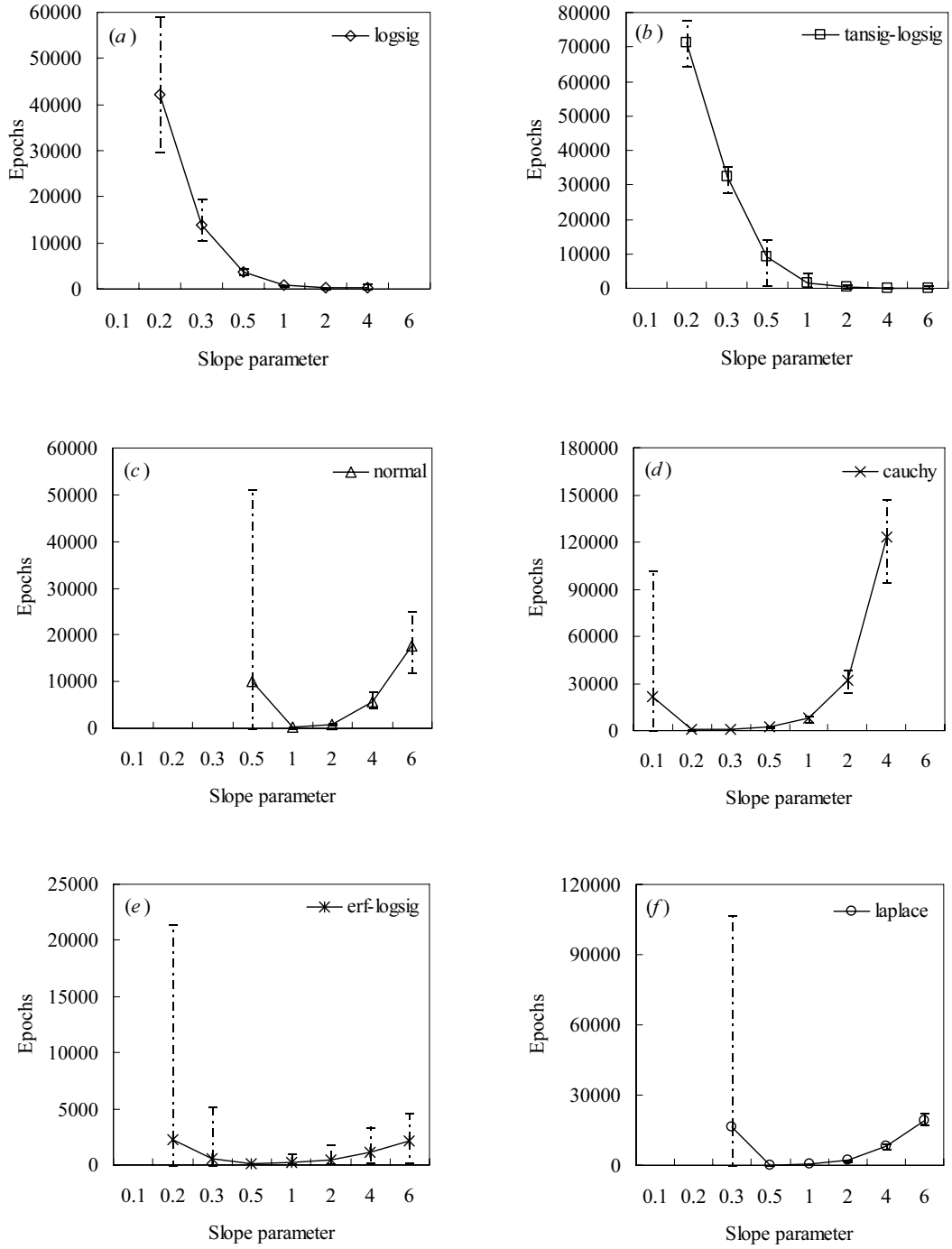

Fig. 2. Epochs of each activation function under different gradient factor conditions

factors caused the convergence speed of the activation functions to vary widely. As the gradient factor varied from 0.1 to 6, the convergence speed of the activation functions first reduced and then increased. Each activation function has its own optimal gradient factor and a reasonable range of values. Once the gradient factor exceeds a certain value, the convergence speed of the network will be very slow or the network may not converge at all. The optimal gradient factors for Logsig and Tansig-Logsig were greater than those for Normal, Cauchy, Erf-Logsig and Laplace. This is related to their unique characteristics. As shown in Fig. 1, the gradients of the Logsig and Tansig function curves increase with increasing gradient factor, while the other four activation functions show the opposite trend. Therefore, it is believed that within a certain range of gradient factors, the activation function converges faster if the gradient is higher, which may be caused by the fact that high gradient can speed up the initial weights when approaching the optimal weights.

Under different gradient factor conditions, the variation coefficient of the convergence epochs are shown in Fig. 3. A larger variation coefficient indicates the activation function is more sensitive to the initial weights, but it does not indicate the convergence speed. As shown in Fig. 3, with different gradient factors, the variation coefficients significantly differ, namely, the activation function at different gradient factors has different levels of sensitivity to the initial weights. This can also be reflected as the margin between the maximum and minimum convergence epochs in Fig. 2. On the other hand, the initial weights can also affect the convergence rate of the activation function. Erf-Logsig is highly sensitive to the initial weights, while Logsig is less sensitive. The variation coefficient of activation function convergence epochs is low at its optimal gradient factor, indicating that an appropriate gradient factor could reduce the sensitivity of an activation function to the initial weights, thus ensuring convergence stability. On the contrary, if the initial weights are inappropriately chosen, it could result in a phenomenon where network training cannot converge or converges very slowly. The sensitivity of the activation function to the initial weights could indirectly prove that methods that adjust the initial weights, such as genetic algorithms and simulated annealing methods, effectively accelerate network convergence. In an application, the activation function gradient factor should first be selected within the proper range, then the initial weights can be optimized by intelligent optimization algorithms such as a genetic algorithm or simulated annealing method.

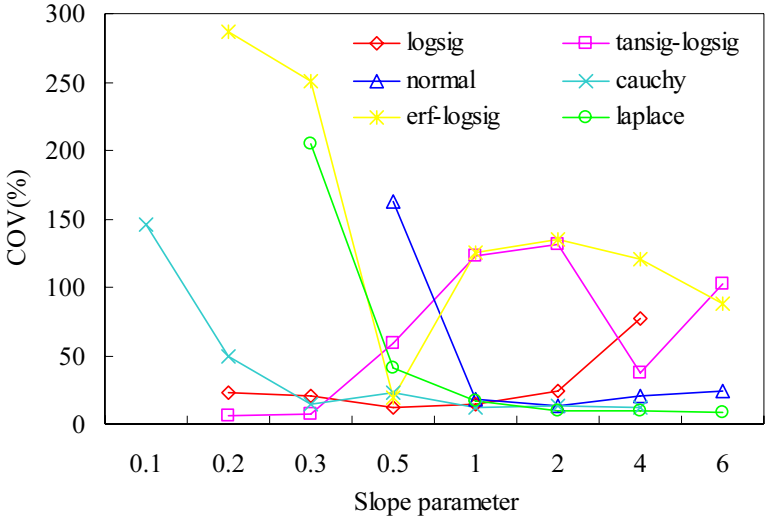

Fig. 3. Epoch variation coefficients for activation functions for various gradient factors

Fig. 4 shows the convergence epochs of the network for 10 repeated simulations of the Normal, Cauchy, and Laplace activation functions with translation factors $\mu = 0$ (no shift) and $\mu = 1$ (a shift to the right by one unit). The results show that the translation factor has no significant effect on the convergence speed of the network. Hence, in an application, the translation factor is not needed.

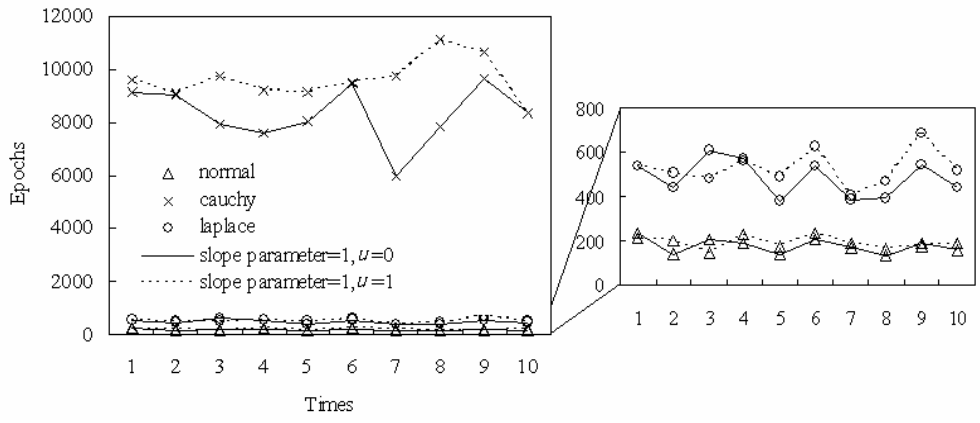

Fig. 4. Effects of the translation factor μ on convergence speed

Based on the convergence number and variation coefficient (Fig. 2 and Fig. 3), the optimal gradient factor and a reasonable range of values for each activation function were selected (Table 2). Among these six activation functions, the best activation function was Erf-Logsig with an average convergence number of at least 110 (gradient factor = 0.5), followed by Tansig-Logsig (gradient factor = 4), Laplace, Logsig, Normal, and Cauchy in that order. In fact, the convergence performance of the network is not only related to the gradient factor of the activation function, but also is related to its hidden layer, neuron number, learning rate, momentum factor, and other network parameters. Only if these parameters have achieved an optimal combination will the result of the entire network be optimized. The hidden layer of the network and its neuron number is related to the properties of training data such as dimension size. Currently, there is no scientific method to determine such information, which is mostly obtained by trial and error. The learning rate and momentum factor also have an appropriate range, generally between 0.1 and 0.9. In this study, under the same condition of other parameters, the choice of optimal gradient factor and a reasonable range for different activation functions was effective. When other parameters change, the conclusions derived from this study are still applicable. The only difference is that with different parameters, the convergence speed of the network will speed up or slow down. However, the trend of changes in the network convergence rate will not change with respect to changes in the activation function gradient factor.

Table 2. Optimal gradient factor and reasonable ranges for each activation function

Activation function	Optimal	Optimal range
logsig	2	[1,4]
tansig-logsig	4	[2,6]
Normal	1	[1,2]
Cauchy	0.2	[0.2,0.5]
Erf-logsig	0.5	[0.5,2]
Laplace	0.5	[0.5,1]

6 Conclusions and Future Work

In this paper, six kinds of BP algorithm-based activation functions and their convergence for different parameters were comparatively analyzed. Some conclusions and directions for follow-up work were obtained, as follows:

(1) The network convergence rate is significantly different for an activation function under different gradient factor conditions, reflecting that the adjustable gradient factor algorithm affects the convergence performance of the network. The translation parameter showed no significant effect on the convergence speed of the network. Therefore, in applications, an appropriate gradient factor should be selected to train the network.

(2) Different initial weights also showed a significant impact on the convergence speed of the network. If initial weights were inappropriately chosen, the network training did not converge or converged very slowly. This suggests an appropriate range of initial weights for network training should be selected. Considering the fast network convergence speed, a relatively small margin between the minimum and maximum convergence epochs, and a large number of convergence times, Logsig ($\lambda = 2$), Tansig-Logsig ($\lambda = 2$), Normal ($\mu = 0$, $\lambda = 2$), Erf-Logsig ($\lambda = 0.5$), and Laplace ($\mu = 0$, $\lambda = 0.5$) were less sensitive to initial weights.

(3) Based on the convergence speed, Laplace ($\mu = 0$, $\lambda = 0.5$), Erf-Logsig ($\lambda = 0.5$), Tansig-Logsig ($\lambda = 4$), Logsig ($\lambda = 2$), and Normal ($\mu = 0$, $\lambda = 1$) showed better performance with appropriate gradient factors. Combined with the plots of the activation functions, we conclude that an activation function converges faster with higher gradients.

(4) There is a great uncertainty in the results of artificial neural networks because of the complexity of the algorithms. The results of this study were only based on one example of the XOR problem, and the simulation results may be dependent on the selected example. Other examples will need to be included to further verify the reliability of the results.

(5) In this work, the effects of the gradient factor and initial weights of the activation function on the network convergence performance were analyzed based on the convergence speed of the network. In future studies, the impact of the activation function parameters on the generalization capability of the

network should be analyzed.

(6) It was found that the network convergence speeds of activation functions with hidden and output layers using a combination of functions (Erf-Logsig and Tansig-Logsig) were superior to the those using the same function. Therefore, in a follow-up study, the convergence speed and generalization ability of the network will be comparatively analyzed using further pairs of these six activation functions.

Acknowledgments

This research is supported by The National Natural Science Fund (31570629, 31300539) and Public welfare technology research industry project from Zhejiang Province (2015C31004).

References

[1] S.T. Wang, F.L. Chung, J. Wang, J Wu, A fast learning method for feedforward neural networks, Neurocomputing 149(2015) 295-307.

[2] N. Vuković, Z. Miljković, Robust sequential learning of feedforward neural networks in the presence of heavy-tailed noise, Neural Networks 63(2015) 31-47.

[3] L.H. Huang, X.Q. Zeng, S. Zhong, L. Han, Sensitivity study of binary feedforward neural networks, Neurocomputing 136(2014) 268-280.

[4] L. Wang, Y. Zeng, T. Chen, Back propagation neural network with adaptive differential evolution algorithm for time series forecasting, Expert Systems with Applications 42(2)(2015) 855-863.

[5] L. Zhu, G. Wang, J. Fu, Z. Dong, A method of Android malware detection based on BP neural network combining static and dynamic features, Journal of Computers 27(3)(2016) 21-31.

[6] H.M. Shao, G.F. Zheng, Convergence analysis of back-propagation algorithm with adaptive momentum, Neurocomputing 74(5)(2011) 749-752.

[7] D.P. Xu, H.M. Shao, H.S. Zhang, A new adaptive momentum algorithm for split-complex recurrent neural networks, Neurocomputing 93(2012) 133-136.

[8] A.P. Piotrowski, J.J. Napiorkowski, Optimizing neural networks for river flow forecasting – evolutionary computation methods versus the levenberg–marquardt approach, Journal of Hydrology 407(1-4)(2011) 12-27.

[9] R. Martin, B. Heinrich, A direct adaptive method for faster backpropagation learning: the RPROP algorithrm, in: Proc. the IEEE International Conference on Neural Networks (ICNN), 1993.

[10] A.K. Pani, H.K. Mohanta, Online monitoring and control of particle size in the grinding process using least square support vector regression and resilientback propagation neural network, ISA Transactions 56(2015) 206-221.

[11] D. Whitley, T. Starkweather, C. Bogart, Gentic algorithms and neural networks:optimizing connections and connectivity, Parallel Computing 14(3)(1990) 347-361.

[12] R.J. Kuo, A sales forecasting system based on fuzzy neural network with initial weights generated by genetic algorithm, European Journal of Operational Research 129(3)(2001) 496-517.

[13] A.L. Wu, Z.G. Zeng, C.J. Fu, W.W. Shen, Global exponential stability in Lagrange sense for periodic neural networks with various activation functions, Neurocomputing 74(2011) 831-837.

[14] L.L. Wang, T.P. Chen, Multistability and complete convergence analysis on high-order neural networks with a class of nonsmooth activationfunctions, Neurocomputing 152(2015) 222-230.

[15] V.P. Nambiar, M. Khalil-Hani, R. Sahnoun, M.N. Marsono, Hardware implementation of evolvable block-based neural

networks utilizing a cost efficient sigmoid-like activation, Neurocomputing 140(2014) 228-241.

[16] D.G. Stork, J.D. Allen, How to solve the N-bit parity problem with two hidden units, Neural Networks 5(1992) 923-926.

[17] K. Hornik, Approximation capabilities of multilayer feed forward networks, Neural Networks 4(1991) 251-257.

[18] V. Tiwari, N. Khare, Hardware implementation of neural network with Sigmoidal activation fu-nctions using CORDIC, Microprocessors and Microsystems 39(6)(2015) 373-381.

[19] P. Doležel, P. Skrabanek, L. Gago, Weight initialization possibilities for feedforward neural network with linear saturated activation functions, IFAC-PapersOnLine 49(2016) 49-54.

[20] Z.J. Jia, Y.D. Song, D.Y. Li, P. Li, Tracking control of nonaffine systems using bio-inspired networks with auto-tuning activation functions and self-growing neurons, Information Sciences 388-389(2017) 191-208.

[21] L. Zhou, J.Sun, Y.B. Yuan, X.Q. Ding, Effects of Combined Activation Function on BP Algorithm's Convergence Speed, Journal of Hohai University(Natural Sciences) 27(1999)107-108.

[22] S.K. Kenue, Efficient Activation Functions for the Back-Propagation Neural Network, in: Proc. Submitted to the International Joint Conference of Neural Networks, 1991.

[23] P. Miao, Y.J. Shen, X.H. Xia, Finite time dual neural networks with a tunable activation function for solving quadratic programming problems and its application, Neurocomputing 143(2014) 80-89.