Scientific Research Publishing

Journal of Applied Mathematics and Physics, 2018, 6, 247-263
http://www.scirp.org/journal/jamp
ISSN Online: 2327-4379
ISSN Print: 2327-4352

Some Features of Neural Networks as Nonlinearly Parameterized Models of Unknown Systems Using an Online Learning Algorithm

Leonid S. Zhiteckii[1], Valerii N. Azarskov[2], Sergey A. Nikolaienko[1], Klaudia Yu. Solovchuk[1]

[1]Department of Intelligent Automatic Systems, International Centre of Information Technologies and Systems, Institute of Cybernetics, Kiev, Ukraine
[2]Aircraft Control Systems Department, National Aviation University, Kiev, Ukraine
Email: leonid_zhiteckii@i.ua

How to cite this paper: Zhiteckii, L.S., Azarskov, V.N., Nikolaienko, S.A. and Solovchuk, K.Yu. (2018) Some Features of Neural Networks as Nonlinearly Parameterized Models of Unknown Systems Using an Online Learning Algorithm. *Journal of Applied Mathematics and Physics*, **6**, 247-263.
https://doi.org/10.4236/jamp.2018.61024

Received: October 31, 2017
Accepted: January 26, 2018
Published: January 29, 2018

Abstract

This paper deals with deriving the properties of updated neural network model that is exploited to identify an unknown nonlinear system via the standard gradient learning algorithm. The convergence of this algorithm for online training the three-layer neural networks in stochastic environment is studied. A special case where an unknown nonlinearity can exactly be approximated by some neural network with a nonlinear activation function for its output layer is considered. To analyze the asymptotic behavior of the learning processes, the so-called Lyapunov-like approach is utilized. As the Lyapunov function, the expected value of the square of approximation error depending on network parameters is chosen. Within this approach, sufficient conditions guaranteeing the convergence of learning algorithm with probability 1 are derived. Simulation results are presented to support the theoretical analysis.

Keywords

Neural Network, Nonlinear Model, Online Learning Algorithm, Lyapunov Function, Probabilistic Convergence

1. Introduction

Design of mathematical models for technical, economic, social and other systems with uncertainties is the important problem from both theoretical and practical points of view. This problem attracts close attention of many researches. The significant progress in this scientific area has been achieved last time. Within this area, new methods and modern intelligent algorithms dealing with uncertain

systems have recently been proposed in [1] [2] [3] [4]. They include some new optimization approaches advanced, in particular, in the papers [2] [4].

Over the past decades, interest has been increasing toward the use multilayer neural networks applied among other as models for the adaptive identification of nonlinearly parameterized dynamic systems [5] [6] [7] [8]. This has been motivated by the theoretical works of several researches [9] [10] who proved that, even with one hidden layer, neural network can uniformly approximate any continuous mapping over a compact domain, provided that the network has sufficient number of neurons with corresponding weights. The theoretical background on neural network modeling may be found in the book [11].

Different learning methods for updating the weights of neural networks have been reported in literature. Most of these methods rely on the gradient concept [8]. One of these methods is based on utilizing the Lyapunov stability theory [6] [12].

The convergence of the online gradient training procedure dealing with input signals that have deterministic (non-stochastic) nature was studied by many authors [13]-[23]. Several of these authors assumed that training set must be finite whereas in online identification schemes, this set is theoretically infinite. Nevertheless, recently we observed a non-stochastic learning process when this procedure did not converge for certain infinite sequence of training examples [24].

The probabilistic asymptotic analysis on convergence of the online gradient training algorithms has been conducted in [25]-[33]. Several of their results make it possible to employ a constant learning rate [28] [30]. To the best of author's knowledge, there are no general results in literature concerning the global convergence properties of training procedures with a fixed learning rate applicable to the case of infinite learning set.

A popular approach to analyze the asymptotic behavior of online gradient algorithms in stochastic case is based on Martingale convergence theory [34]. This approach has been exploited by the authors in [24] to derive some local convergence in stochastic framework for standard online gradient algorithms with the constant learning rate.

The difficulties associated with convergence properties of online gradient learning algorithms are how to guarantee the boundedness of the network weights biases assuming the learning process to be theoretically infinite. To overcome these difficulties, the penalty term to an error function has been introduced in [33]. Recently we however established in [35] that the global convergence of these algorithms with probability 1 can be achieved without any additional term, at least, in the case when the activation function of the network output layer is linear.

This work has been motivated by the fact that the standard gradient algorithm is widely exploited for online updating the neural network weights in accordance with the gradient-descent principle whereas the following important question related to its ultimate properties remained in part open as yet: when does the sequential procedure based on this algorithm converge if the learning rate is

constant? As pointed out in [23], the answer to the question on convergence properties of this standard algorithm which should shed some light on asymptotic features of multilayer neural networks using the gradient-like training technique is the first step toward a full understanding of other more generic training algorithms based on regularization, conjugate gradient, and Newton optimization methods, etc.

Novelty of this paper which extends the basic ideas of [35] to the case where the activation function of the output layer is nonlinear, consists in establishing sufficient conditions under which the gradient algorithm for learning neural networks will globally converge in the sense almost sure for the case when the learning rate can be constant. The proposed approach to deriving these convergence results is based on utilizing the Lyapunov methodology [36]. They make it possible to reveal some new features of the multilayer neural networks with nonlinear activation function in output layer which use the online gradient-type training algorithms having a constant learning rate.

2. Description of Learning Neural Network System: Problem Formulation

Consider the typical three-layer feedforward neural network containing a hidden layer and p inputs, q hidden neurons, and one output neuron. Denote by

$$W = \left(w_{ij}\right)_{q\times p} = \left[w_1, \cdots, w_q\right]^{\mathrm{T}}$$

with

$$w_i = \left[w_{i1}, \cdots, w_{ip}\right]^{\mathrm{T}} \in \mathbf{R}^p, \ i = 1, \cdots, q$$

the weight matrix connecting the input and hidden layers, and define the so-called bias vector w_0 as

$$w_0 = \left[w_{01}, \cdots, w_{0q}\right]^{\mathrm{T}} \in \mathbf{R}^q,$$

which is the threshold in the hidden-layer output. Further, let

$$\omega = \left[\omega_1, \cdots, \omega_q\right]^{\mathrm{T}} \in \mathbf{R}^q,$$

be the weight vector between the hidden and output layers, and ω_0 be the bias in the output layer. As in [33], the activation functions used in the hidden neurons are all the same denoted by $g : \mathbf{R} \to \mathbf{R}$, and the activation function for the output layer is $f : \mathbf{R} \to \mathbf{R}$.

Now, denoting by

$$G(z) = \left[g(z_1), \cdots, g(z_q)\right]^{\mathrm{T}}$$

the vector-valued function which depends on the vector $z = \left[z_1, \cdots, z_q\right]^{\mathrm{T}} \in \mathbf{R}^q$, introduce the extended matrix $\tilde{W} = \left[W \vdots w_0\right] \in \mathbf{R}^{q\times(p+1)}$ by adding the column w_0 to W and the extended vector $\tilde{\omega} = \left[\omega^{\mathrm{T}}, \omega_0\right]^{\mathrm{T}} \in \mathbf{R}^{q+1}$, and also the function $\tilde{G}(z) = \left[g(z_1), \cdots, g(z_q), 1\right]^{\mathrm{T}}$ of z. Then the for an input vector

$$x = \left[x_1, \cdots, x_p\right]^{\mathrm{T}} \in \mathbf{R}^p,$$

the output vector of hidden layer can be written as $\tilde{G}\left(\tilde{W}\tilde{x}\right)$, where the notation $\tilde{x}=\left[x^{\mathrm{T}},1\right]^{\mathrm{T}}$ of the extended vector $\tilde{x}\in\mathbf{R}^{p+1}$ is used, and the final output $y_{\mathrm{NN}}\in\mathbf{R}$ of the neural network can be expressed as follows:

$$y_{\mathrm{NN}}=f\left(\tilde{\omega}^{\mathrm{T}}\tilde{G}\left(\tilde{W}\tilde{x}\right)\right). \tag{1}$$

Let

$$y=\varphi\left(x\right) \tag{2}$$

with $\varphi:\mathbf{R}^{p}\to\mathbf{R}$ be an unknown and bounded nonlinearity given over the bounded either finite or infinite sets $X\subset\mathbf{R}^{p}$ which are depicted in Figure 1 for the case $p=2$. This function needs to be approximated by the neural network (1) via suitable choice of $\tilde{\omega}$ and $\tilde{W}$. By virtue of (2) the approximation error

$$e\left(\tilde{\omega},\tilde{W},x,y\right)=y-f\left(\tilde{\omega}^{\mathrm{T}}\tilde{G}\left(\tilde{W}\tilde{x}\right)\right) \tag{3}$$

depends on x for any fixed $\left(\tilde{\omega},\tilde{W}\right)$.

Now, suppose that some complex system to be identified is described at each nth time instant by the equation

$$y^{n}=\varphi\left(x^{n}\right)\quad\left(n=0,1,2,\cdots\right) \tag{4}$$

in which $x^{n}\in X$ and $y^{n}\in\mathbf{R}$ are its input and output signals, respectively available for measurement.

Based on the infinite sequence of the training examples $\left\{x^{n},y^{n}\right\}_{n=0}^{\infty}$ that is generated by (4), the outline learning algorithm for updating the weight and biases in (1) is defined as the standard gradient-descent iteration procedure

$$\tilde{\omega}^{n+1}=\tilde{\omega}^{n}-\eta_{n}\nabla_{\tilde{\omega}}e^{2}\left(\tilde{\omega}^{n},\tilde{W}^{n};y^{n},\tilde{x}^{n}\right), \tag{5}$$

$$w_{i}^{n+1}=w_{i}^{n}-\eta_{n}\nabla_{\tilde{w}_{i}}e^{2}\left(\tilde{\omega}^{n},\tilde{W}^{n};y^{n},\tilde{x}^{n}\right), \tag{6}$$

$$i=1,\cdots,q,\quad n=0,1,\cdots.$$

In these equations, $\nabla_{\tilde{\omega}}e^{2}\left(\cdot,\cdot;\cdot,\cdot\right)$ and $\nabla_{\tilde{w}_{i}}e^{2}\left(\cdot,\cdot;\cdot,\cdot\right)$ denote the current gradients of the error function $e^{2}\left(\tilde{\omega},\tilde{W};y,\tilde{x}\right)$ with respect to $\tilde{\omega}$ and w_{i},

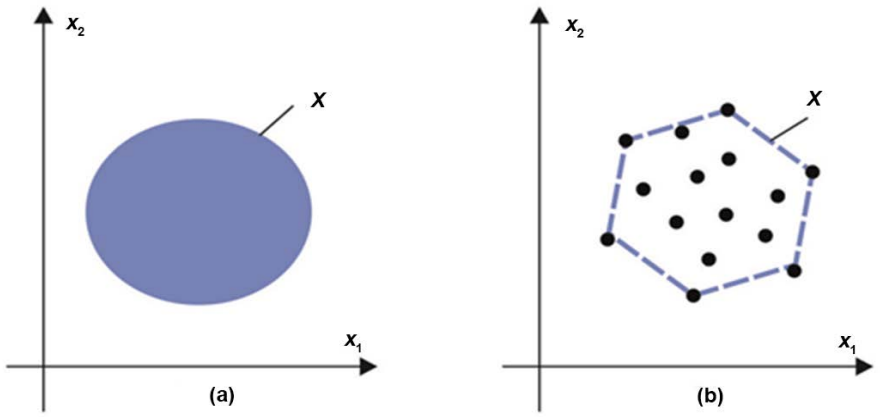

Figure 1. Training sets: (a) X is an infinite set of xs; (b) X is a finite set of xs.

respectively, obtained after substituting $\tilde{\omega} = \tilde{\omega}^n$, $\tilde{W} = \tilde{W}^n$, $y = y^n$, and $\tilde{x} = \tilde{x}^n$ into (3), and $\eta_n > 0$ represents the step size (the learning rate). Note that the expressions of $\nabla_{\tilde{\omega}} e^2\left(\tilde{\omega}^n, \tilde{W}^n; y^n, \tilde{x}^n\right)$ and $\nabla_{\tilde{w}_i} e^2\left(\tilde{\omega}^n, \tilde{W}^n; y^n, \tilde{x}^n\right)$ may be written in detail similar to that in [23] [33]. (Due to space limitation, they are here omitted.)

Introducing the notation

$$\theta = \left[\tilde{\omega}^{\mathrm{T}}, w_1^{\mathrm{T}}, \cdots, w_q^{\mathrm{T}}, w_0^{\mathrm{T}}\right]^{\mathrm{T}}$$

of the extended weight and bias vector $\theta \in \mathbf{R}^{q(p+1)}$, and considering the Equations (5) and (6) in conjunction, rewrite the online gradient learning algorithm for updating θ^n in a general form (as in [33])

$$\theta^{n+1} = \theta^n - \eta_n \nabla_\theta e^2\left(\theta^n; y^n, \tilde{x}^n\right), \tag{7}$$

where $\nabla_\theta e^2(\cdot;\cdot,\cdot)$ represents the gradient of $e^2\left(\theta^n; y^n, \tilde{x}^n\right)$ with respect to θ calculated at the nth time instant.

Thus, the Equation (7) together with the expression

$$e\left(\theta^n; y^n, x^n\right) = y^n - y_{\mathrm{NN}}^n$$

in which y^n is given by (4), and

$$y_{\mathrm{NN}}^n = \psi\left(\theta^n, x^n\right)$$

describe the learning neural network system necessary to identify the nonlinearity (2). For better understanding the performance of this system, its structure is depicted in **Figure 2**, where the notation $e^n = e\left(\theta^n; y^n, x^n\right)$ is used.

The problem formulated in this paper consists in analyzing asymptotic properties of the learning neural network system presented above. More certainly, it is required to derive conditions under which the learning procedure will be convergent meaning the existence of a limit

$$\lim_{n\to\infty} \theta^n = \theta^\infty \tag{8}$$

in some sense [24].

3. Preliminaries

Suppose that there is a multilayer neural network described by

$$y_{\mathrm{NN}} \equiv \psi(\theta, x),$$

where θ is some fixed parameter vector. According to [9] [10], the requirement

$$\max_{x\in X} \left|\varphi(x) - \psi(\theta, x)\right| \le \varepsilon$$

evaluating an accuracy of the approximation of $\varphi(x)$ by $\psi(\theta, x)$ can be satisfied for any $\varepsilon > 0$ via suitable choice of θ and the number of the neurons in its layers. On the other hand, the performance index of the neural network model with a fixed number of these neurons defining its approximation capability might naturally be expressed as follows:

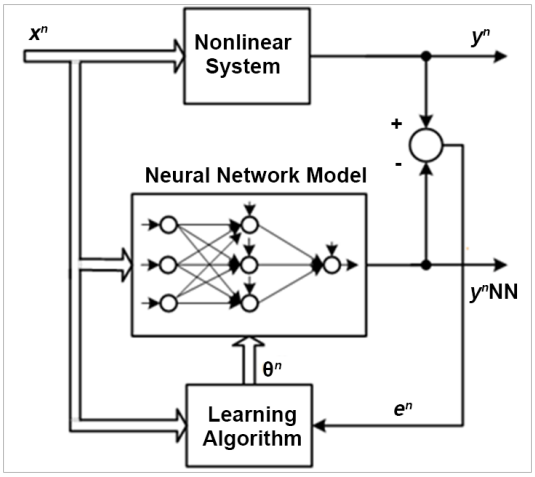

Figure 2. Configuration of learning neural network system.

$$J^0(\theta) = \max_{x \in X} \left| \varphi(x) - \psi(\theta, x) \right|. \tag{9}$$

In fact, the desired (optimal) vector $\theta = \theta_0^*$ will then be specified from (9) as the variable θ minimizing $J^0(\theta)$:

$$\theta_0^* = \arg\min_{\theta} \max_{x \in X} \left| \varphi(x) - \psi(\theta, x) \right|. \tag{10}$$

Nevertheless, all researches which employ online learning procedures in stochastic environment "silently" replace $J^0(\theta)$ by

$$J(\theta) = E_x \left\{ e^2(\theta; y, \tilde{x}) \right\},$$

where $E_x \left\{ e^2(\theta; y, \tilde{x}) \right\}$ denotes the expected value of $e^2(\theta; y, \tilde{x})$.

Indeed, the learning algorithm (7) does not minimize (9): namely, it minimizes $J(\theta)$ (instead of $J^0(\theta)$) [37]. This observation means that (7) will at best yield

$$\theta^* := \arg\min_{\theta} J(\theta).$$

but not θ_0^* given by (10) as $n \to \infty$.

Now, consider a special case when the unknown function (2) can exactly be approximated by the neural network $\psi(\theta, \tilde{x})$ implying

$$\varphi(x) \equiv \psi(\theta^*, \tilde{x}) \quad \forall x \in X. \tag{11}$$

In this case called in ([8], p. 304) by the ideal case, we have $e(\theta^*, \tilde{x}) \equiv 0$ for any x from X and, consequently, $J(\theta^*) = 0$.

If the condition given in identity (11) is satisfied, then the learning rate η_n in (7) may be constant:

$$\eta_n \equiv \eta = \text{const} > 0;$$

see ([37], sect. 3.13).

Note that the property (11) may take place, in particular, when $X = \left\{ x^{(1)}, \cdots, x^{(K)} \right\}$ contains certain number $K = \text{card } X$ of training examples

provided that their number does not exceed the dimension of θ. To understand this fact, according to (11) write the set of K equations

$$\left.\begin{array}{c}\psi\left(\theta,\tilde{x}^{(1)}\right)=y^{(1)}\\ \vdots\\ \psi\left(\theta,\tilde{x}^{(K)}\right)=y^{(K)}\end{array}\right\}$$

with respect to the unknown θ. They are compatible if $K\le q(p+2)+1$. Due to (2) together with the definition of θ^* it can be concluded that their solution is just $\theta=\theta^*$ yielding $J\left(\theta^*\right)=0$ because in this special case, $\psi\left(\theta^*,\tilde{x}^{(k)}\right)=y^{(k)}$ for all $k=1,\cdots,K$.

4. Main Results

4.1. Some Feature of Multilayer Neural Network

It turns out that if the activation functions g of the hidden layer are nonlinear, then for an arbitrary fixed vector θ' there is, at least, one vector θ'' such that the network outputs for these different vectors are the same even though the output activation function f is linear, *i.e.* if $f(\zeta)=\zeta$:

$$\psi\left(\theta',\tilde{x}\right)\equiv\psi\left(\theta'',\tilde{x}\right)\quad\forall x\in X. \tag{12}$$

The feature (12) gives that in the presence of nonlinear g there exist, at least, two different θ^*s. For example, let $p=1, q=1$ and

$$g\left(z_1\right)=\frac{1}{1+\exp\left(-z_1\right)}$$

in which $z_1=w_{11}x_1+w_{01}$, and $f(\zeta)=\zeta$ with $\zeta=\omega_1 g\left(z_1\right)+\omega_0$. Fix a $\theta'=\left[w'_{11},w'_{01},\omega'_1,\omega'_0\right]^{\mathrm{T}}$. Then $\theta''=\left[-w'_{11},-w'_{01},-\omega'_1,\omega'_1+\omega'_0\right]^{\mathrm{T}}$ will also satisfy (12); see [35]. Therefore, the set of θ^*s will be not one-point if g is nonlinear.

4.2. An Observation

To study some asymptotic properties of sequence $\left\{\theta^n\right\}$ caused by the learning algorithm (7) in the non-stochastic case, simulation experiments with the scalar nonlinear system (2) having the nonlinearity

$$\varphi(x)=\frac{3.75+0.05\exp(-7.15x)}{1+0.19\exp(-7.15x)}$$

were conducted. This nonlinearity can explicitly be approximated by the two-layer neural network model described by $\psi\left(\theta^*,\tilde{x}\right)$ as in Subsection 4.1 with $\theta^{*(1)}=\left[7.15,1.65,3.45,0.3\right]^{\mathrm{T}}$ and $\theta^{*(2)}=\left[-7.15,-1.65,-3.45,3.75\right]^{\mathrm{T}}$.

Figure 3 illustrates the results of the one simulation experiment with $\eta=0.01$, where $\left\{x^n\right\}$ was chosen as a non-stochastic sequence. It can be observed that in this example, the variable $\min_{i=1,2}\left\|\theta^{*(i)}-\theta^n\right\|$ shown in **Figure 3(b)** has no limit implying that the learning algorithm (7) may not be convergent: in this case, the limit (8) does not exist, see **Figure 3(c)**.

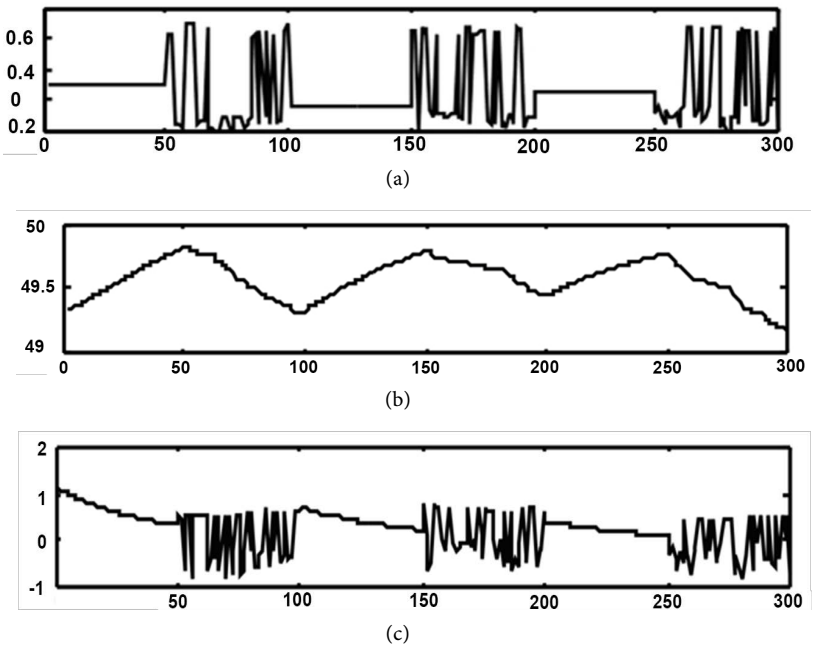

Figure 3. Behaviour of learning algorithm (7) in non-stochastic case: (a) inputs e^n; (b) the variable $\min\left\{\left\|\theta^{*(1)}-\theta^n\right\|,\left\|\theta^{*(2)}-\theta^n\right\|\right\}$; (c) current model error e^n.

4.3. Sufficient Conditions for the Probabilistic Convergence of Learning Procedure

The following basic assumption concerning $\left\{x^n\right\}_{n=0}^{\infty}$ which is bounded stochastic sequence (since X is bounded) is made:

(A1) x^ns arise randomly in accordance with a probability distribution $P(x)$ if X is finite, and with probability density $p(x)$ if X is infinite.

Within assumption (A1), the expected value (mean) of $e^2(\theta;y,\tilde{x})=\left(y-\psi(\theta,\tilde{x})\right)^2$ is given by

$$E_x\left\{e^2(\theta;y,\tilde{x})\right\}=\begin{cases}\sum\limits_{x\in X}e^2(\theta;y,\tilde{x})P(x) & \text{if } X \text{ is finite set,}\\ \int_X e^2(\theta;y,\tilde{x})\,p(x)\,\mathrm{d}x & \text{if } X \text{ is infinite set.}\end{cases}$$

To derive the main theoretical result we need Assumption (A1) and the following additional assumptions:

(A2) the identity (11) holds;

(A3) the activation functions used in the hidden neurons and output neuron are the same $\left(f(\cdot)=g(\cdot)\right)$, twice continuously differentiable on $\mathbf{R}$ and also uniformly bounded on $\mathbf{R}$.

Further, we introduce a scalar function $V(\theta)$ playing a role of the Lyapunov function [36] with the features:

(a) $V(\theta)$ is nonnegative, *i.e.*,

$V(\theta)\geq 0$; (13)

(b) $V(\theta)$ is the Lipschitz function in the sense that

$$\left\|\nabla V(\theta')-\nabla V(\theta'')\right\| \le L\left\|\theta'-\theta''\right\| \tag{14}$$

for any θ', θ'' from $\mathbf{R}^{q(p+1)}$, where $\nabla V(\theta)$ denotes its gradient, and $L>0$ represents the Lipschitz constant.

Now, the global stochastic convergence analysis of the gradient learning algorithm (7) is based on employing the fundamental convergence conditions established in the following Key Technical Lemma which is the slightly reformulated Theorem 3 of [36].

Key Technical Lemma. Let $V(\theta)$ be a function satisfying (13) and (14). Define the scalar variable

$$H(\theta)=\nabla_\theta V(\theta)^{\mathrm{T}}\nabla_\theta E\{Q(x,\theta)\} \tag{16}$$

with some $Q(x,\theta)\ge 0$, and denote

$$H_n(\theta):=\nabla_\theta V(\theta^n)^{\mathrm{T}}\nabla_\theta E\{Q(x,\theta^n)\}.$$

Suppose:

1) $H_n(\theta)\ge\Theta_n V(\theta^{n-1}), \Theta_n>0,$

2) $E\left\{\left\|\nabla_\theta Q(x,\theta^n)\right\|^2\right\}\le\tau_n V(\theta^n), \tau_n\ge 0.$

Introduce the additional variable

$$\nu_n=\eta_n(\Theta_n-L\eta_n\tau_n/2). \tag{17}$$

Then the algorithm (7) yields

$\lim_{n\to\infty} V_n=0$ a.s.,

where $V_n:=V(\theta^n)$ provided that $E\{\theta^0\}<\infty$ and

$$0\le\nu_n\le 1, \tag{18}$$

$$\sum_{n=0}^{\infty}\nu_n=\infty. \tag{19}$$

Related results followed from the Theorem 3' of [36] are:

Corollary. Under the conditions of the Key Technical Lemma, if $\Theta_n\equiv\Theta=\text{const}$ and $\tau_n\equiv\tau=\text{const}$, and $\eta_n\equiv\eta=\text{const}$, then $V_n\xrightarrow[n\to\infty]{}0$ with probability 1 provided that

$$0<\eta\le 2(\Theta-\varepsilon)/L\tau \quad (0<\varepsilon<\Theta) \tag{20}$$

is satisfied. ■

Next, we are able to present the convergence result summarized in the theorem below.

Theorem. Suppose Assumption (A2) holds. Then the gradient algorithm (7) with a constant learning rate, $\eta_n\equiv\eta$, will converge with probability 1 (in the sense that $V_n\xrightarrow[n\to\infty]{}0$ a.s.) and

$$\lim_{n\to\infty} e(\theta^n;y^n,\tilde{x}^n)=0 \quad \text{a.s.} \tag{21}$$

for any initial θ^0 chosen randomly so that $E\{Q(x,\theta^0)\}<\infty$ if η satisfies the conditions (20) with Θ and τ determined by

$$\Theta := \inf_{\theta} \frac{\left\| \nabla_{\theta} E\left\{ Q(x,\theta) \right\} \right\|^2}{E\left\{ Q(x,\theta) \right\}} > 0, \tag{22}$$

$$\tau := \sup_{\theta} \frac{E\left\{ \left\| \nabla_{\theta} Q(x,\theta) \right\|^2 \right\}}{E\left\{ Q(x,\theta) \right\}} < \infty. \tag{23}$$

Proof. Set $V(\theta) = E\left\{ Q(x,\theta) \right\}$. Then condition (13) and (14) can be shown to be valid. This indicates that this function may be taken as the Lyapunov function. By virtue of (16) such a choice of $V(\theta)$ gives $H(\theta) = \left\| \nabla_{\theta} E\left\{ Q(x,\theta) \right\} \right\|^2$. Putting $\Theta_n \equiv \Theta$ and $\tau_n \equiv \tau$ with Θ and τ determined by (22) and (23), respectively, we can conclude that the conditions 1), 2) of the Key Technical Lemma are satisfied. Applying its Corollary it proves that $\lim_{n\to\infty} V_n = 0$ with probability 1.

Due to the fact that $V(\theta) = E_x\left\{ e^2(\theta; y, \tilde{x}) \right\}$ together with Assumption (A2), result (21) follows. ■

4.4. Simulations and a Discussion

To demonstrate theoretical result given in Subsection 4.3, several simulations were conducted. First, we dealt with the same neural network and the same training samples as in ([33], p. 1052). Namely, they were chosen as follows:

$$x^{(1)} = [0,0]^{\mathrm{T}},\ y^{(1)} = 1;$$

$$x^{(2)} = [0,1]^{\mathrm{T}},\ y^{(2)} = 0;$$

$$x^{(3)} = [1,0]^{\mathrm{T}},\ y^{(3)} = 0;$$

$$x^{(4)} = [1,1]^{\mathrm{T}},\ y^{(4)} = 1.$$

The two numerical examples with different initial θ^0 were considered. In Example 1 we set $w_{11}^0 = 0.95$, $w_{12}^0 = -0.084$, $w_{21}^0 = 0.079$, $w_{22}^0 = -0.079$, $w_{01}^0 = -0.089$, $w_{02}^0 = 0.075$, $\omega_1^0 = 0.357$, $\omega_2^0 = -0.357$, $\omega_0^0 = 0.354$. In Example 2 we set $w_{11}^0 = -0.090$, $w_{12}^0 = 0.225$, $w_{21}^0 = -0.138$, $w_{22}^0 = 0.139$, $w_{01}^0 = 0.222$, $w_{02}^0 = -0.084$, $\omega_1^0 = -0.356$, $\omega_2^0 = 0.357$, $\omega_0^0 = 0.353$.

Contrary to [33] the learning rate was chosen as $\eta = 0.01$ in order to implement the algorithms (5), (6) with no penalty term.

Results of two simulation experiments whose durations were 10000 iteration steps are presented in **Figure 4** and **Figure 5** in which the components of θ^n and $J(\theta^n)$ are shown.

Further, another simulation experiments were also conducted. In contrast with previous experiments, they dealt with an infinite training sets X Namely, the two simulations with the same nonlinear function as in Subsection 4.2 were first conducted, provided that X is the infinite bounded set given by $X \in [-2,2]$. However, $\{x^n\}$ was now chosen as the stochastic sequence. Namely, it was generated as a pseudorandom i.i.d. sequence.

Two numerical examples were considered. In Example 3, the initial values of neural network weights and biases were taken as follows: $w_1^0 = 0.529$,

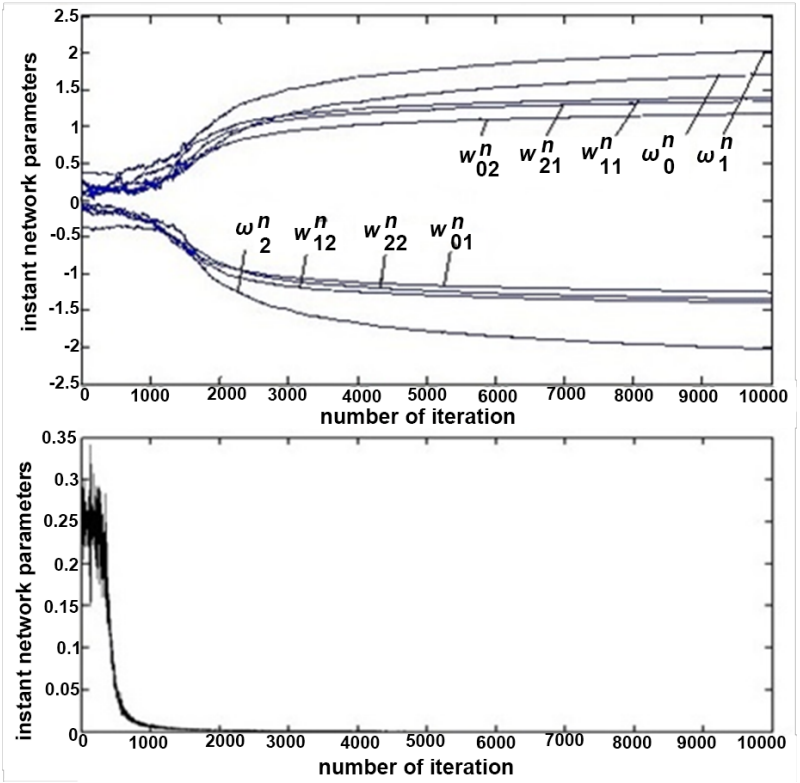

Figure 4. Behavior of gradient learning algorithm (7) in Example 1.

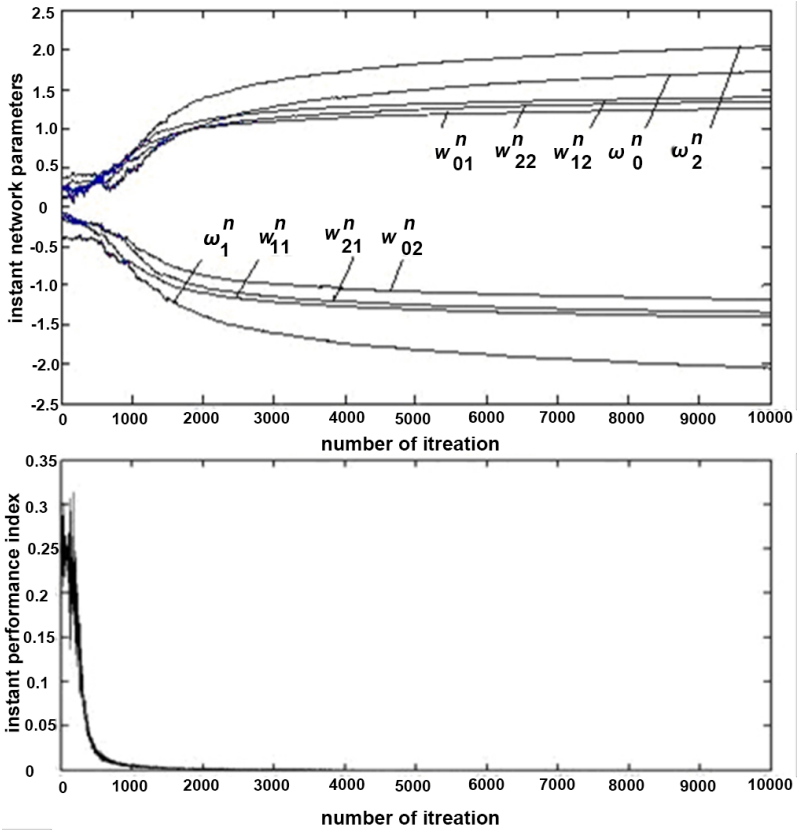

Figure 5. Behavior of gradient learning algorithm (2) in Example 2.

$w_2^0 = -0.5012$, $\omega_1^0 = -0.9168$, $\omega_2^0 = 1.0409$. In Example 4 we set $w_1^0 = -0.3756$, $w_2^0 = -0.572$, $\omega_1^0 = -0.9798$, $\omega_2^0 = 1.1436$. Figure 6 and Figure 7 demonstrate results of the two simulation experiments conducted with the initial estimates θ^0 given above. In both experiments, η_n was also chosen as $\eta_n \equiv \eta = 0.01$.

Next, another nonlinearity

$$\varphi(x) = \frac{1}{1+\exp\left[-a_1(x) - a_2(x) - 1\right]}$$

with $a_1(x) = \left[1+\exp(-10x-5)\right]^{-1}$ and $a_2(x) = \left[1+\exp(-10x+5)\right]^{-1}$ to be exactly approximated by a suitable neural network was chosen as in [11, p. 12-4]. The following initial estimates were taken: $w_1^0 = 2.8$, $w_2^0 = -5.6$, $w_3^0 = -2.8$, $w_4^0 = -5.6$, $\omega_1^0 = 5.33$, $\omega_2^0 = 1.71$, $\omega_3^0 = -3.52$ (Example 5), and $w_1^0 = 0.27$, $w_2^0 = 0.19$, $w_3^0 = -3.09$, $w_4^0 = 3.96$, $\omega_1^0 = 1.64$, $\omega_2^0 = 0.72$, $\omega_3^0 = -2.21$ (Example 6).

Results of the two simulation experiments conducted with the initial estimates θ^0 given above are depicted in Figure 8 and Figure 9.

From Figures 4-9 we can see that the learning processes converge and the performance index $J(\theta^n)$ tends to zero while the penalty term is absent. It can be observed that if the initial vectors θ^0s are different then the sequences $\{\theta^n\}$ may converge to different final θ^∞s.

The simulation experiments show that the penalty term is not necessary, in principle, to achieve the convergence of the online gradient learning procedure in the three-layer neural networks if certain conditions given by Assumption (A1)-(A3) are satisfied. This fact supports our theoretical results.

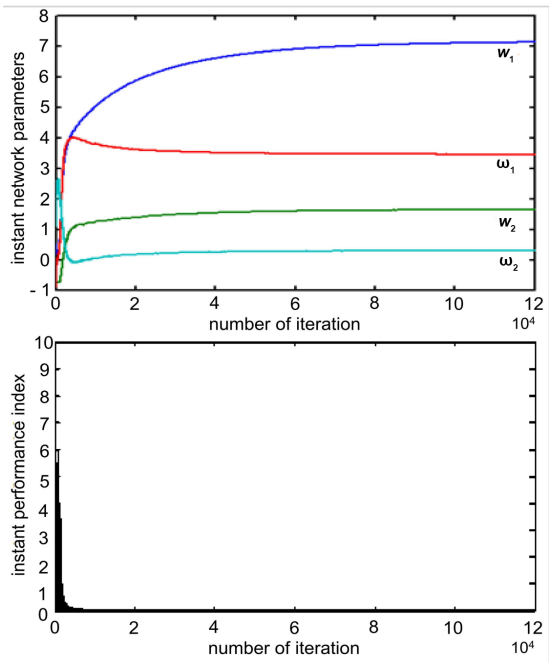

Figure 6. Behavior of gradient learning algorithm (7) in Example 3.

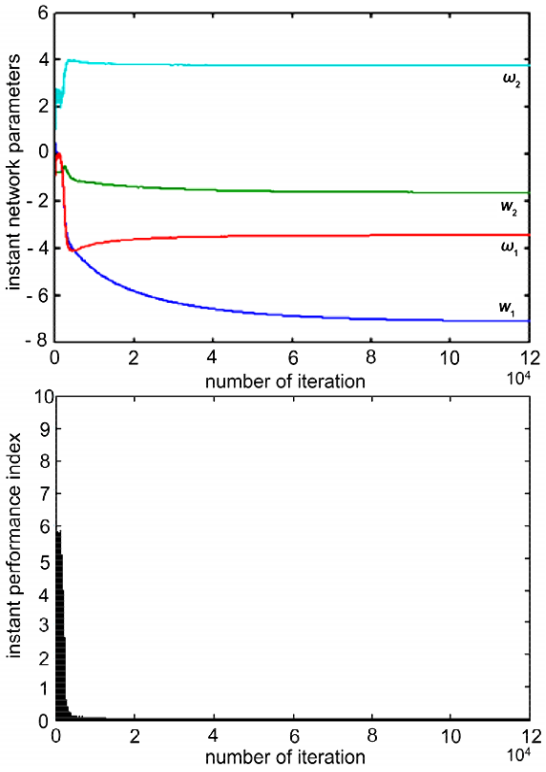

Figure 7. Behavior of gradient learning algorithm (7) in Example 4.

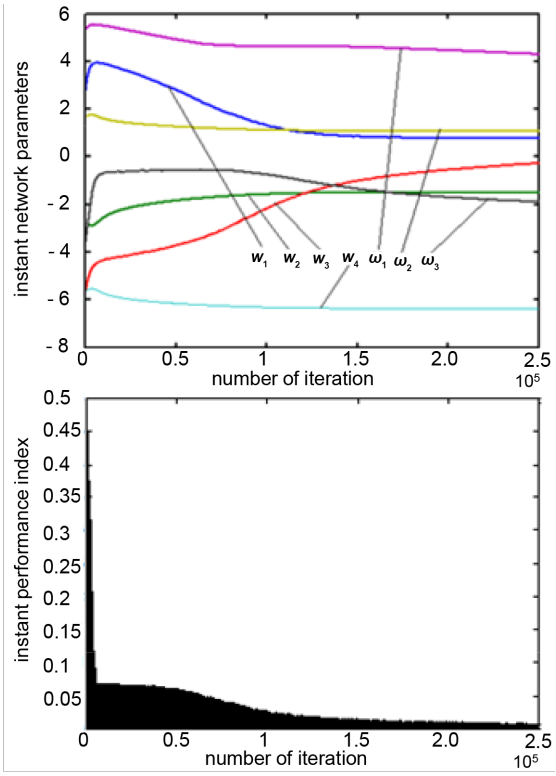

Figure 8. Behavior of gradient learning algorithm (7) in Example 5.

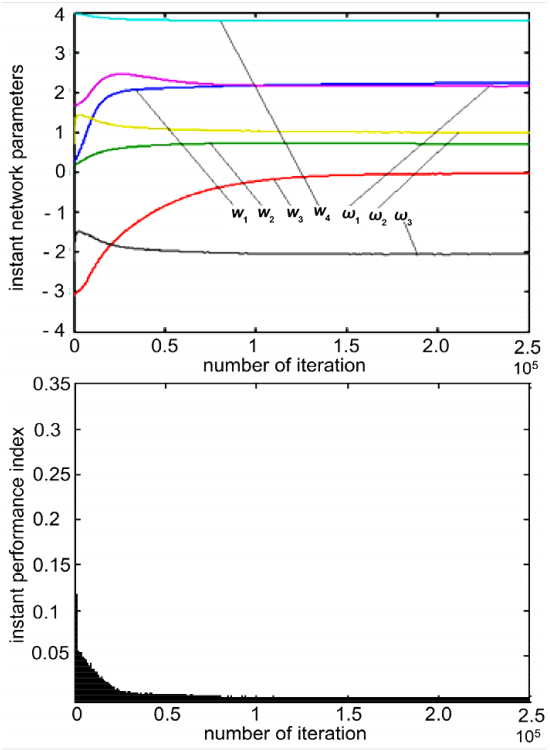

Figure 9. Behavior of gradient learning algorithm (7) in Example 6.

5. Conclusion

In this paper, some important features of multilayer neural networks which are utilized as nonlinearly parameterized models of unknown nonlinear systems to be identified have been derived. A special case where the nonlinearity can exactly be approximated by a three-layer neural network has been studied. Contrary to previous author's papers we dealt with the neural network having a nonlinear activation function for its output layer. It was shown that if the activation function of the hidden layer is nonlinear, then, for any input variables, there are, at least, two different network parameter vectors under which the network outputs will be the same even though the output activation function is linear. This feature gives that the standard gradient online training algorithm with a constant learning rate may not be convergent if the training sequence is non-stochastic. Nevertheless, provided that this sequence is stochastic, it has theoretically been established that, under certain conditions, such algorithm will converge with probability one. However, ultimate values of network parameters may be different. These facts were confirmed by simulation experiments.

Acknowledgements

The authors are grateful to anonymous reviewer for his valuable comments.

References

[1] Chen, L., Peng, J., Zhang, B. and Rosyida, I. (2017) Diversified Models for Portfolio

Selection Based on Uncertain Semivariance. *International Journal of Systems Science*, **3**, 637-648. https://doi.org/10.1080/00207721.2016.1206985

[2] Draa, A., Bouzoubia, S. and Boukhalfa, I. (2015) A Sinusoidal Differential Evolution Algorithm for Numerical Optimisation. *Applied Soft Computing*, **27**, 99-126. https://doi.org/10.1016/j.asoc.2014.11.003

[3] Zhang, B., Peng, J., Li, S. and Chen, L. (2016) Fixed Charge Solid Transportation Problem in Uncertain Environment and Its Algorithm. *Computers & Industrial Engineering*, **102**, 186-197. https://doi.org/10.1016/j.cie.2016.10.030

[4] Sun, G., Zhao, R. and Lan, Y. (2016) Joint Operations Algorithm for Large-Scale Global Optimization. *Applied Soft Computing*, **38**, 1025-1039. https://doi.org/10.1016/j.asoc.2015.10.047

[5] Suykens, J. and Moor, B.D. (1993) Nonlinear System Identification Using Multilayer Neural Networks: Some Ideas for Initial Weights, Number of Hidden Neurons and Error Criteria. *Proceedings of the* 12*th IFAC World Congress*, *Sydney*, *Australia*, **3**, 49-52.

[6] Kosmatopoulos, E.S., Polycarpou, M.M., Christodoulou, M.A. and Ioannou, P.A. (1995) High-Order Neural Network Structures for Identification of Dynamical Systems. *IEEE Transactions on Neural Networks*, **6**, 422-431. https://doi.org/10.1109/72.363477

[7] Levin, A.U. and Narendra, K.S. (1995) Recursive Identification Using Feedforward Neural Networks. *International Journal of Control*, **61**, 533-547. https://doi.org/10.1080/00207179508921916

[8] Tsypkin, Ya.Z., Mason, J.D., Avedyan, E.D., Warwick, K. and Levin, I.K. (1999) Neural Networks for Identification of Nonlinear Systems Under Random Piecewise Polynomial Disturbances. *IEEE Transactions on Neural Networks*, **10**, 303-311. https://doi.org/10.1109/72.750559

[9] Cybenko, G. (1989) Approximation by Superpositions of a Sigmoidal Functions. *Mathematics of Control*, *Signals*, *and Systems,* **2**, 303-313.

[10] Funahashi, K. (1989) On the Approximate Realization of Continuous Mappings by Neural Networks. *Neural Networks*, **2**, 182-192. https://doi.org/10.1016/0893-6080(89)90003-8

[11] Hagan, M.T., Demuth, H.B. and Beale, M.H. (1996) Neural Network Design. PWS Publishing, Boston.

[12] Behera, L., Kumar, S. and Patnaik, A. (2006) On Adaptive Learning Rate That Guarantees Convergence in Feedforward Networks. *IEEE Transactions on Neural Networks*, **17**, 1116-1125. https://doi.org/10.1109/TNN.2006.878121

[13] Mangasarian, O.L. and Solodov, M.V. (1994) Serial and Parallel Backpropagation Convergence via Nonmonotone Perturbed Minimization. *Optimization Methods of Software*, **4**, 103-116, 199. https://doi.org/10.1080/10556789408805581

[14] Luo, Z. and Tseng, P. (1994) Analysis of an Approximate Gradient Projection Method with Application to the Backpropagation Algorithm. *Optimization Methods of Software*, **4**, 85-101. https://doi.org/10.1080/10556789408805580

[15] Ellacott, S.W. (1993) The Numerical Analysis Approach. In: Taylor, J.G., Ed., *Mathematical Approaches to Neural Networks*, Elsevier Science Publisher B.V., Amsterdam, 103-137. https://doi.org/10.1016/S0924-6509(08)70036-9

[16] Wu, W. and Shao, Z. (2003) Convergence of an Online Gradient Methods for Continuous Perceptrons with Linearly Separable Training Patterns. *Applied Mathematics Letters*, **16**, 999-1002. https://doi.org/10.1016/S0893-9659(03)90086-3

[17] Wu, W. and Xu, Y.S. (2002) Deterministic Convergence of an Online Gradient Method for Neural Networks. *Journal of Computational and Applied Mathematics*, **144**, 335-347. https://doi.org/10.1016/S0377-0427(01)00571-4

[18] Wu, W., Feng, G.R., Li, X. and Xu, Y.S. (2005) Deterministic Convergence of an Online Gradient Method for BP Neural Networks. *IEEE Transactions on Neural Networks*, **16**, 1-9. https://doi.org/10.1109/TNN.2005.844903

[19] Wu, W., Feng, G. and Li, X. (2002) Training Multilayer Perceptrons via Minimization of Ridge Functions. *Advances in Computational Mathematics*, **17**, 331-347. https://doi.org/10.1023/A:1016249727555

[20] Wu, W., Shao, H. and Qu, D. (2005) Strong Convergence for Gradient Methods for BP Networks Training. *Proceedings of* 2005 *International Conference on Neural Networks and Brain*, Beijing, 13-15 October 2005, 332-334. https://doi.org/10.1109/ICNNB.2005.1614626

[21] Zhang, N., Wu, W. and Zheng, G. (2006) Convergence of Gradient Method with Momentum for Two-Layer Feedforward Neural Networks. *IEEE Transactions on Neural Networks*, **17**, 522-525. https://doi.org/10.1109/TNN.2005.863460

[22] Shao, H., Wu, W. and Liu, L. (2007) Convergence and Monotonicity of an Online Gradient Method with Penalty for Neural Networks. *WSEAS Transactions on Mathematics*, **6**, 469-476.

[23] Xu, Z.B., Zhang, R. and Jing, W.-F. (2009) When Does Online BP Training Converge? *IEEE Transactions on Neural Networks*, **20**, 1529-1539. https://doi.org/10.1109/TNN.2009.2025946

[24] Zhiteckii, L.S., Azarskov, V.N. and Nikolaienko, S.A. (2012) Convergence of Learning Algorithms in Neural Networks for Adaptive Identification of Nonlinearly Parameterized Systems. *Proceedings* 16*th IFAC Symposium on System Identification* Brussels, 11-13 July 2012, 1593-1598. https://doi.org/10.3182/20120711-3-BE-2027.00150

[25] Li, Z., Wu, W. and Tian, Y. (2004) Convergence of an Online Gradient Method for FNN with Stochastic Inputs. *Journal of Computational and Applied Mathematics*, **163**, 165-176. https://doi.org/10.1016/j.cam.2003.08.062

[26] White, H. (1989) Some Asymptotic Results for Learning in Single Hidden-Layer Feedforward Neural Network Models. *Journal of the American Statistical Association*, **84**, 1003-1013. https://doi.org/10.1080/01621459.1989.10478865

[27] Fine, T.L. and Mukherjee, S. (1999) Parameter Convergence and Learning Curves for Neural Networks. *Neural Computing and Applications*, **11**, 749-769. https://doi.org/10.1162/089976699300016647

[28] Finnoff, W. (1994) Diffusion Approximations for the Constant Learning Rate Backpropagation Algorithm and Resistance to Local Minima. *Neural Computing and Applications*, **6**, 285-295. https://doi.org/10.1162/neco.1994.6.2.285

[29] Oh, S.H. (1997) Improving the Error BP Algorithm with a Modified Error Function. *IEEE Transactions on Circuits and Systems*, **8**, 799-803.

[30] Kuan, C.M. and Hornik, K. (1991) Convergence of Learning Algorithms with Constant Learning Rates. *IEEE Transactions on Neural Networks*, **2**, 484-489. https://doi.org/10.1109/72.134285

[31] Gaivoronski, A.A. (1994) Convergence Properties of Backpropagation for Neural Nets via Theory of Stochastic Gradient Methods. *Optimization Methods of Software*, **4**, 117-134. https://doi.org/10.1080/10556789408805582

[32] Tadic, V. and Stankovic, S. (2000) Learning in Neural Networks by Normalized

Stochastic Gradient Algorithm: Local Convergence. *Proceedings of the* 5*th Seminar on Neural Network Applications in Electrical Engineering*, Yugoslavia, 26-27 September 2000, 11-17. https://doi.org/10.1109/NEUREL.2000.902375

[33] Zhang, H., Wu, W., Liu, F. and Yao, M. (2009) Boundedness and Convergence of Online Gradient Method with Penalty for Feedforward Neural Networks. *IEEE Transactions on Neural Networks*, **20**, 1050-1054. https://doi.org/10.1109/TNN.2009.2020848

[34] Loeve, M. (1963) Probability Theory. Springer-Verlag, New York.

[35] Azarskov, V.N., Kucherov, D.P., Nikolaienko, S.A. and Zhiteckii, L.S. (2015) Asymptotic Behaviour of Gradient Learning Algorithms in Neural Network Models for the Identification of Nonlinear Systems. *American Journal of Neural Networks and Applications*, **1**, 1-10. https://doi.org/10.11648/j.ajnna.20150101.11

[36] Polyak, B.T. (1976) Convergence and Convergence Rate of Iterative Stochastic Algorithms, I: General Case. *Automation and Remote Control*, **12**, 1858-1868.

[37] Tsypkin, Ya.Z. (1971) Adaptation and Learning in Automatic Systems. Academic Press, New York.