数理解析研究所講究録
第1762巻 2011年 1-12

Analysis of a learning algorithm for a single phasor neuron

東京大学・生産技術研究所　田中　剛平 (Gouhei Tanaka)
Institute of Industrial Science, The University of Tokyo

Abstract

Numerical methods for computational intelligence are often described by dynamical systems. For instance, a learning algorithm for complex-valued neural networks can be regarded as a dynamical system with complex variables. Especially when the state of a complex-valued neuron is defined on the unit circle in the complex domain, it is called a phasor neuron, which is a special class of a complex-valued neuron. We have recently derived a supervised learning algorithm for multilayer feedforward phasor neural networks, by recursively updating complex-valued weight parameters. As a first step towards understanding the dynamics of the algorithm, the case of a single phasor neuron is examined in this report.

1 Introduction

In engineering and computer science, a variety of numerical algorithms have been presented to solve mathematical problems which are difficult to address analytically. Even with developments of computers in the several decades, there are still problems for which a rigorous solution can be found only with unrealistically much computation time. Therefore, efficient numerical techniques have been explored to obtain a good solution in a feasible computation time even if it is not the best one.

Some numerical methods represented by a recursive formula are closely related to discrete-time dynamical systems. One of such algorithms is the gradient method to find a local optimum of a nonlinear function. Let us consider the following nonlinear scholar function:

$$y = f(\mathbf{x}), \tag{1}$$

where $\mathbf{x} = (x_1, \ldots, x_n)^T$ is an n-dimensional real vector and the function f is differentiable with respect to $\mathbf{x}$. Then, the gradient vector of f, represented as

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n} \right), \tag{2}$$

is the normal vector to the constant-level surface of f. As the state $\mathbf{x}$ is moved in this direction, the function most increases. To reduce the functional value, it is valid to move the state in its inverse direction. Based on this notion, the steepest (gradient) descent method is summarized as follows:

- Step. 1: Set an initial condition $\mathbf{x}_0$ and $k = 0$.
- Step. 2: If $\nabla f(\mathbf{x}_k) = \mathbf{0}^T$, then end.
- Step. 3: The state is updated as follows:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha(\nabla f(\mathbf{x}_k))^T \tag{3}$$

  where α is the parameter corresponding to the step size of each update.

- Step. 4: $k := k + 1$ and go to Step. 2.

This algorithm guarantees that the functional value monotonically decreases and the state converges to one of the local minima of the nonlinear function f. The iterative formula (3) can be regarded as a discrete-time dynamical system. When α is too small, the convergence time would be very long. The initial condition is also a key component which influences the convergence time and the probability that the global minimum is found.

The backpropagation learning algorithm for artificial neural networks, which is the most popular one, is also based on the above gradient method. We have recently derived a backpropagation algorithm with multilayer feed-forward phasor neural networks for learning circular data [1]. As a first step towards understanding the dynamics of the algorithm, we analyze the learning method for a single phasor neuron in this report.

2 Learning algorithm for circular data

The gradient descent method has been applied to a learning algorithm for a neural network. The error backpropagation algorithm [2] based on a gradient descent method has provided a general method to optimize weight and bias parameters for minimizing the errors between a network output and a desired

one. Since the algorithm requires a gradient vector, the activation function of a neuron must be continuous and differentiable. Therefore, instead of the step function with discrete nonlinearity, the sigmoid activation function with continuous nonlinearity has been typically employed for learning algorithms of ordinary real-valued neural networks.

Recently, a growing attention has been paid to neural networks which are able to address information with special properties. For instance, complex-valued neural networks are useful for complex-valued information with amplitude and phase components, as found in wave phenomena. For deriving a backpropagation algorithm, the complex activation function requires to be bounded and holomorphic. However, such an entire function is limited to a constant function due to Liouville's theorem. Thus, activation functions which are bounded but not holomorphic in some singular points have been applied practically so far. In fact, the learning algorithms for complex-valued neural networks with real-imaginary type activation functions [3, 4] and amplitude-phase type ones [5, 6] have been presented.

Some data involved with spatial direction and time periodicity are called *circular data*. For dealing with circular data without amplitude components, the data is represented by a complex variable which is defined not on the whole complex domain but on the unit circle in the complex domain. The phasor neuron proposed by Noest [7, 8] enables such a topologically appropriate information representation and belongs to a special class of a complex-valued neuron. We have presented a backpropagation learning algorithm for multilayer feedforward phasor neural networks to learn circular data. The error function of this algorithm is given by the angular distance. In this sense, the phasor neural network is different from circular node networks [9] with an error function based on Euclidean distance.

In the remaining of this section, we review the algorithm for learning circular data [1]. The forward propagation is used to calculate the output of each neuron for a given input to the network and evaluate the output error. The backward propagation is used to optimize the weight and bias parameters for minimizing the output error. The forward and backward propagations are repeated for each sample data until the output error falls below a small threshold.

Figure 1(a) shows a schematic illustration of a single phasor neuron, which receives the sum of weighted inputs:

$$re^{i\varphi} \equiv w_0 + \sum_j w_j e^{i\theta_j}. \tag{4}$$

Here w_j denotes the weight parameter and w_0 denotes the bias parameter. These parameters are assumed to be complex numbers. The single phasor

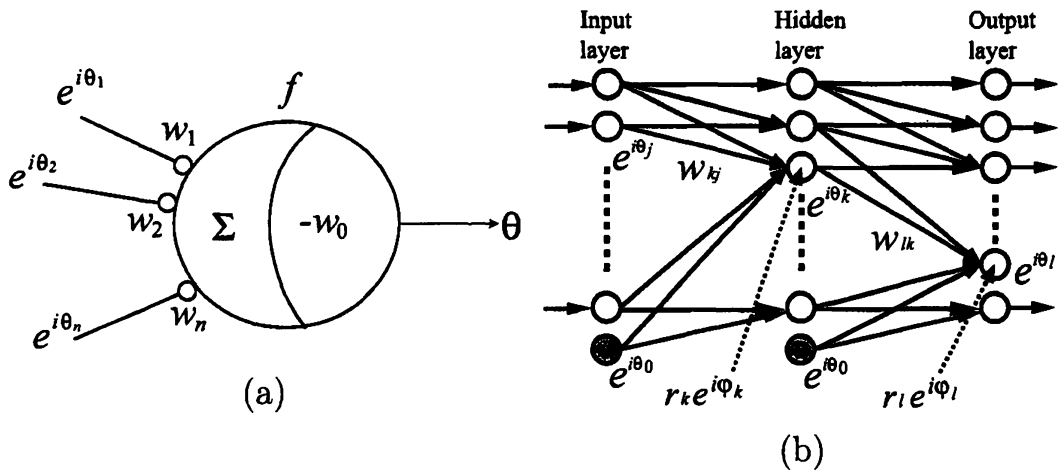

Figure 1: (a) A single phasor neuron. (b) A multilayer phasor neural network.

neuron operates only on the phase component φ of the above summation. The output of the neuron is given by

$$\theta = f(\varphi), \tag{5}$$

where f determines the characteristic of the phasor neuron. We assume that the function f is continuous and differentiable with respect to φ, to derive a gradient descent backpropagation algorithm. In the multilayer feedforward phasor neural network as illustrated in Fig. 1(b), input data is given to the first input layer and then the outputs of each layer are transmitted to the neurons in the next layer sequentially. Once the outputs of a layer are obtained, the outputs of the next layer can be calculated. As a result of sequential operations (forward propagation of the information), we finally obtain the network output.

We summarize the forward propagation for an L-layer phasor neural network. The number of neurons in the lth layer ($l = 1, \ldots, L$) is denoted by N_l. The state of the jth neuron in the lth layer is denoted by θ_j^l ($j = 1, \ldots, N_l$). The weight parameter between the jth neuron in the $(l-1)$th layer and the kth neuron in the lth layer is denoted by $w_{kj}^{(l)}$ ($j = 1, \ldots, N_{l-1}$, $k = 1, \ldots, N_l$). The bias constant parameters for the kth neuron in the lth layer are denoted by $w_{k0}^{(l)}$ ($k = 1, \ldots, N_l$). The weight and bias parameters are assumed to be complex numbers. We focus on the kth neuron in the lth layer. It receives the sum of weighted inputs from the previous layer as follows:

$$r_k^{(l)} \exp(i\varphi_k^{(l)}) \equiv w_{k0}^{(l)} + \sum_{j=1}^{N_{l-1}} w_{kj}^{(l)} \exp(i\theta_j^{(l-1)}), \quad k = 1, \ldots, N_l. \tag{6}$$

By using an activation function $f : \mathcal{S}^1 \to \mathcal{S}^1$, the output of the neuron is

given by

$$\theta_k^{(l)} = f(\varphi_k^{(l)}), \quad k = 1, \ldots, N_l. \tag{7}$$

When the weight and bias parameter values are randomly assigned, the network output obtained by the forward propagation is generally different from the desired output.

To estimate a system characterizing the sample data, we consider an error function to evaluate the difference between a network output and a desired output as follows:

$$\begin{aligned} E_p &= \frac{1}{2}\sum_{k=1}^{N_L} |e^{i\theta_k^{(L)}} - e^{i\hat{\theta}_k^{(L)}}|^2 \\ &= \sum_{k=1}^{N_L} \{1 - \cos(\theta_k^{(L)} - \hat{\theta}_k^{(L)})\}, \end{aligned} \tag{8}$$

which is based on the angular distance. The non-negative error function E_p vanishes if and only if the current and desired outputs are the same.

The weight and bias parameters are corrected so that the error function decreases, as follows:

$$\tilde{w}_{kj}^{(l)} = w_{kj}^{(l)} - \eta \left(\frac{\partial E_p}{\partial w_{kj}^{(l)R}} + i \frac{\partial E_p}{\partial w_{kj}^{(l)I}} \right), \tag{9}$$

where $w_{kj}^{(l)} = w_{kj}^{(l)R} + i w_{kj}^{(l)I}$ represents the current parameter value. The real and imaginary parts of the complex numbers are indicated by the superscripts R and I, respectively. The variation of the parameter is proportional to the negative value of the gradient vector. The learning rate $\eta > 0$ corresponds to the step size of the weight updates. Using the chain rule, we obtain

$$\left(\frac{\partial E_p}{\partial w_{kj}^{(l)R}} + i \frac{\partial E_p}{\partial w_{kj}^{(l)I}} \right) = \frac{\partial E_p}{\partial \theta_k^{(l)}} f'(\varphi_k^{(l)}) \exp\{i(\varphi_k^{(l)} - \theta_j^{(l-1)} + \pi/2)\}/r_k^{(l)}, \tag{10}$$

The derivative $\partial E_p / \partial \theta_k^{(l)}$ is a local error which can be computed from the local errors in the post-layer. The local error is propagated backwards. Hence, this is called a backpropagation algorithm. The local error is calculated by the following recursive formula:

$$\frac{\partial E_p}{\partial \theta_k^{(L)}} = \begin{cases} \sin(\theta_k^{(L)} - \hat{\theta}_k^{(L)}) & (l = L), \\ \sum_{m=1}^{N_{l+1}} \frac{\partial E_p}{\partial \theta_m^{(l+1)}} \frac{\partial \theta_m^{(l+1)}}{\partial \theta_k^{(l)}} & (l = L-1, \ldots, 1), \end{cases} \tag{11}$$

where

$$\frac{\partial\theta_m^{(l+1)}}{\partial\theta_k^{(l)}} = f'(\varphi_m^{(l+1)})\frac{w_{mk}^{(l+1)R}\cos(\varphi_m^{(l+1)}-\theta_k^{(l)}) + w_{mk}^{(l+1)I}\sin(\varphi_m^{(l+1)}-\theta_k^{(l)})}{r_m^{(l+1)}}.$$

The forward propagation is used to calculate the current output error and the backward propagation is used to calculate the local errors. The sequence of updates in every neuron is repeated until the output error becomes sufficiently small. The above algorithm is represented as a discrete-time dynamical system where weight and bias parameters are the state variables. Since all the weight and bias parameters are updated, the total number of the state variables is given by $\sum_{l=1}^{L} N_{l+1}(N_l+1)$. Analyzing the system from dynamical systems viewpoint could be useful to understand the performance of the algorithm, e.g. convergence speed and effective initial conditions. However, the analysis is difficult when the number of state variables is large. To get an insight into the dynamics of the algorithm, we consider the algorithm for a single phasor neuron in the next section.

3 Analysis of the algorithm for a single phasor neuron

3.1 System reduction

We consider a two-layer network with one input and one output, i.e. $N_1 = N_2 = 1$. The nonlinear transformation of this network is essentially performed by a single phasor neuron. We suppose that a single pair of input pattern $\hat{\theta}_1^{(1)} = \theta_{in}$ and output one $\hat{\theta}_1^{(2)} = \theta_{out}$ is given. The weighted sum transmitted to the neuron is given by

$$r_1^{(2)}e^{i\varphi_1^{(2)}} = w_{10}^{(2)} + w_{11}^{(2)}e^{i\theta_1^{(1)}}. \tag{12}$$

By assuming $w_{10}^{(2)} = 0$, we obtain

$$r_1^{(2)} = |w_{11}^{(2)}|, \tag{13}$$

$$\varphi_1^{(2)} = \arg(w_{11}^{(2)}) + \theta_1^{(1)}. \tag{14}$$

Hence, the output is given by

$$\theta_1^{(2)} = \varphi_1^{(2)} = \arg(w_{11}^{(2)}) + \theta_1^{(1)}. \tag{15}$$

From Eqs. (9)-(11), the algorithm is written as

$$\tilde{w}_{11}^{(2)} = w_{11}^{(2)} - i\eta\sin(\theta_1^{(2)} - \hat{\theta}_1^{(2)})\exp\{i(\varphi_1^{(2)} - \theta_1^{(1)})\}/r_1^{(2)}. \tag{16}$$

Substituting Eqs. (13)-(15) into the above equation, we get

$$\tilde{w}_{11}^{(2)} = w_{11}^{(2)} - i\eta \sin(\arg(w_{11}^{(2)}) + \hat{\theta}_1^{(1)} - \hat{\theta}_1^{(2)}) e^{i\arg(w_{11}^{(2)})}/|w_{11}^{(2)}|. \quad (17)$$

By rewriting $w_{11}^{(2)}$ as w_k to represent the weight parameter value after k iterations and introducing $\delta \equiv \theta_{out} - \theta_{in} = \hat{\theta}_1^{(2)} - \hat{\theta}_1^{(1)}$, the update scheme is represented by the following dynamical system:

$$w_{k+1} = w_k - i\eta \sin(\arg(w_k) - \delta) w_k/|w_k|^2. \quad (18)$$

The learning rate $\eta > 0$ and the input-output difference δ are the system parameters. In the rest of this section, we analyze the system (18).

3.2 Stability analysis

To examine the dynamics of system (18) with any δ, it is enough to investigate the case of $\delta = 0$. It is because the transformation of $w_k \to w_k e^{i\theta}$ results in the following system:

$$w_{k+1} = w_k - i\eta \sin(\arg(w_k)) w_k/|w_k|^2. \quad (19)$$

Fixed points From Eq. (19), a fixed point w^* satisfies

$$i\eta \sin(\arg(w^*)) w^*/|w^*|^2 = 0. \quad (20)$$

Therefore, the set of fixed points is given by

$$A^+ = \{w^* \mid \arg(w^*) = 0\}, \quad (21)$$
$$A^- = \{w^* \mid \arg(w^*) = \pi\}. \quad (22)$$

It should be noted that a convergence to a fixed point in A^+ implies a successful learning of the sample data.

Symmetric property The behavior of the system (19) is symmetric with respect to the real axis. Suppose that $w_k = r_k e^{i\varphi_k}$ is transformed into $w_k' = r_k e^{i\varphi_k'} = r_k e^{-i\varphi_k}$. Then, it follows $w_k = r_k e^{-i\varphi_k'}$. By substituting this equation into Eq. (19), we obtain the equivalent dynamical system for w_k'. Hence, the system is invariant under the above transformation.

System decomposition The system (19) can be rewritten as follows:

$$r_{k+1}e^{\varphi_{k+1}} = (1+d_k^2)^{1/2} r_k e^{i(\varphi_k+\alpha_k)}, \tag{23}$$

where $d_k = \eta \sin\varphi_k / r_k^2$ and α_k is defined as an argument satisfying $\cos\alpha_k = 1/(1+d_k^2)^{1/2}$ and $\sin\alpha_k = -d_k/(1+d_k^2)^{1/2}$. Hence, the amptitude and phase components can be decomposed as follows:

$$r_{k+1} = \left(r_k^2 + \frac{\eta^2 \sin^2\varphi_k}{r_k^2}\right)^{1/2}, \tag{24}$$

$$\varphi_{k+1} = \varphi_k + \alpha_k. \tag{25}$$

Therefore, it is obvious that the radius of the orbit monotonically increases unless the state is a fixed point. This fact can be confirmed in Fig. 2 where the orbits starting from initial conditions with a fixed radius are shown. In Figs. 2(a) and (b), all the orbits seem to stay in the one side of the real axis. However, in Fig. 2(c), there is an orbit which changes the side with respect to the real axis as marked by the filled circles. We investigate the condition for the orbit to change the side before and after an update, by comparing the signs of $\sin\varphi_{k+1}$ and $\sin\varphi_k$. We obtain

$$\frac{\sin\varphi_{k+1}}{\sin\varphi_k} = \frac{\sin(\varphi_k+\alpha_k)}{\sin\varphi_k} = \frac{1}{(1+d_k^2)^{1/2}}\left(1 - \eta\cos\varphi_k/r_k^2\right). \tag{26}$$

Therefore, the sign changes if $\cos\varphi_k > r_k^2/\eta$. Such a region is shown in Fig. 3. When $r_0 > \sqrt{\eta}$, the orbit holds the side until its convergence to a fixed point because r_k is monotonically increasing. This condition is satisfied by the cases in Figs. 2(a) and (b), while not in Fig. 2(c).

Stability analysis The local stability of the fixed points in A^+ and A^- is investigated. Suppose that a fixed point $w^* = r$ with $\arg(w^*) = 0$ is perturbed by ϵ in the phase direction. By substituting $\varphi_k = 0 + \epsilon_k$ into Eq. (25), we get

$$\epsilon_{k+1} \sim \epsilon_k - \eta \sin\epsilon_k / r_k^2 (1+d_k^2)^{1/2}. \tag{27}$$

Here we adopted an approximation $\alpha_k \sim \sin\alpha_k$ for $\alpha_k \ll 1$. Since ϵ_k converges to 0, the fixed point r (≥ 0) is stable. Therefore, the set A^+ is composed of stable fixed points.

Similarly, suppose that a fixed point $w^* = -r$ with $\arg(w^*) = \pi$ is perturbed by ϵ in the phase direction. By substituting $\varphi_k = \pi + \epsilon_k$ into Eq. (25), we get

$$\epsilon_{k+1} \sim \epsilon_k + \eta \sin\epsilon_k / r_k^2 (1+d_k^2)^{1/2}. \tag{28}$$

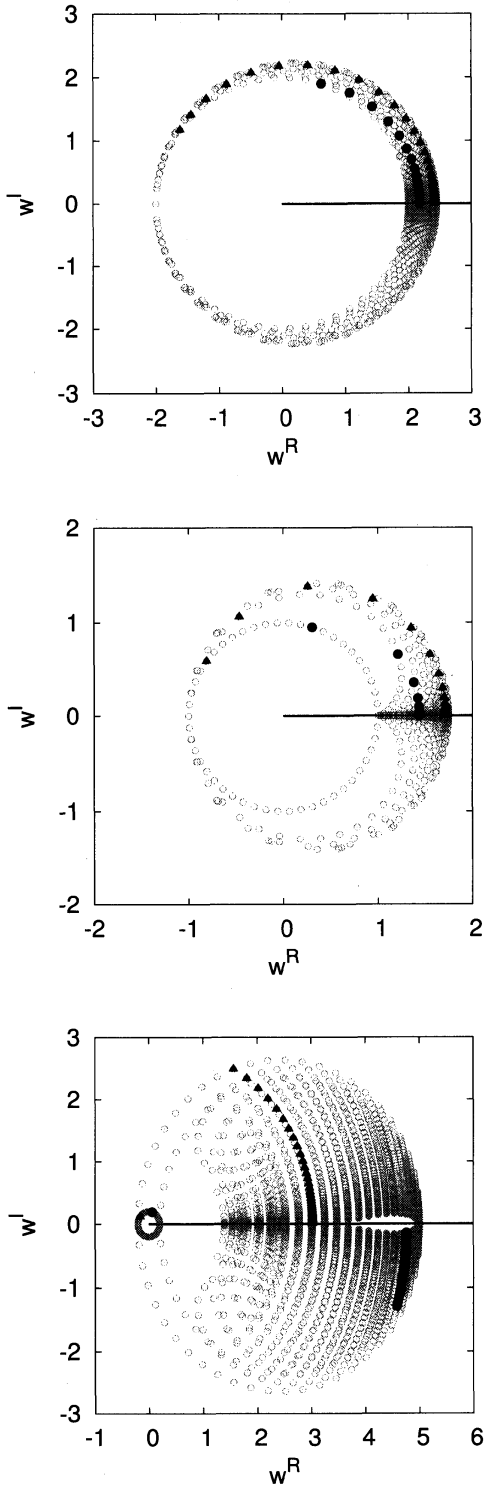

Figure 2: Behavior of the dynamical system (18) with $\eta = 1$. The solid line indicates the set A^+ of stable fixed points. In each panel, 50 initial conditions are distributed on the circles with $|w_0| = 2$ (top), $|w_0| = 1$ (middle), and $|w_0| = 0.2$ (bottom).

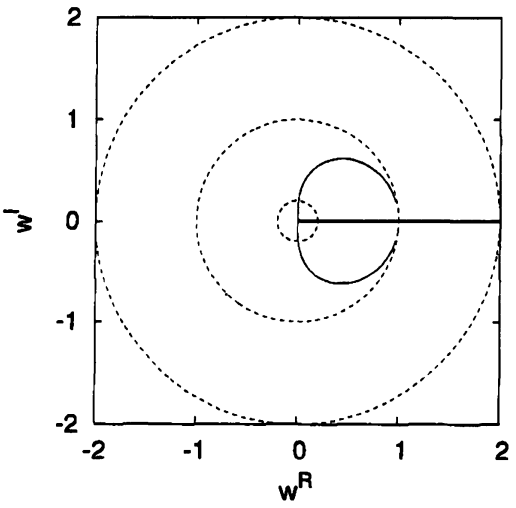

Figure 3: The region (bounded by the solid curve) where the orbit moves to the opposite side of the real axis by the next update.

Since ϵ_k does not converge to 0, the fixed point $-r$ (< 0) is unstable. Therefore, the set A^- are composed of unstable fixed points.

In conclusion, any initial condition excluding the point at infinity converges to the global attractor A^+ where the argument of w is zero. This means that the learning is successful for any initial condition in the original system (18).

3.3 Convergence time

It has been guaranteed that the learning is successful for almost all initial conditions. Next we focus on transient dynamics of the system (19). As shown in Fig. 2, the time needed for an orbit to reach the stable fixed point is dependent on the initial condition of the algorithm. The convergence time is closely related to the learning performance of the phasor neural network. Figure 4 shows the convergence time for different values of the learning rate η. We can see that the initial conditions close to the origin require a relatively long time steps for convergence. As the learning rate η increases, the area of such initial conditions increases. Therefore, the weight parameters which are initially randomly assigned should be distributed apart from the origin. If a learning rate is changed, accordingly the initial weight parameter values should also be appropriately given. The numerical result for a single phasor neuron would be useful to improve the effectiveness of the backpropagation algorithm for phasor neural networks.

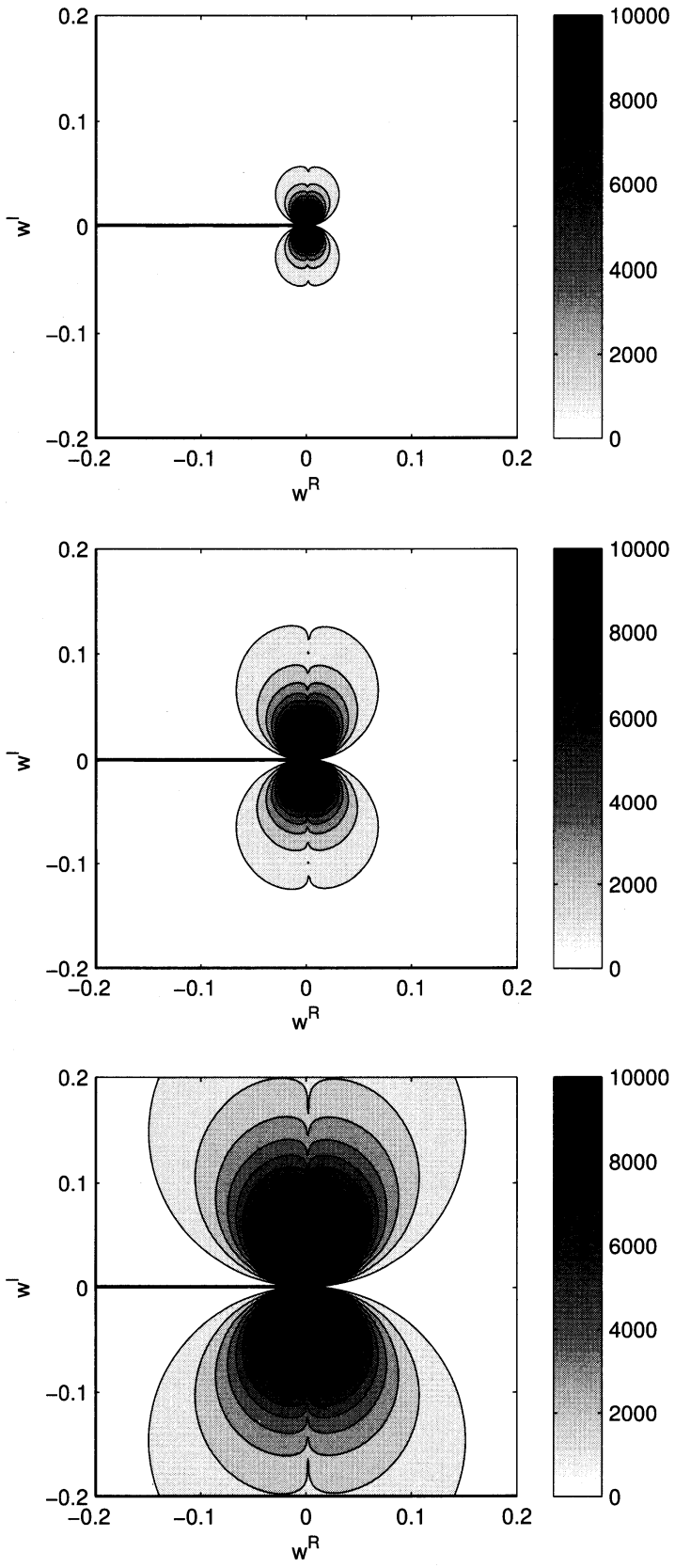

Figure 4: Dependence of the convergence time on the initial conditions. (Top) $\eta = 0.2$. (Middle) $\eta = 1$. (Bottom) $\eta = 2$.

4 Summary

We have investigated the performance of a learning algorithm for a phasor neural network from the viewpoint of dynamical system. The analysis of the simplest case has shown that the gradient descent algorithm has a set of globally stable fixed points, where the learning error vanishes. Therefore, the learning is successful for almost all initial conditions for the algorithm. However, the convergence time is largely influenced by the initial condition. We have demonstrated that the initial conditions close to the origin require a very long convergence time. The numerical simulation has shown that the area of such initial conditions increases as the learning rate increases.

References

[1] G. Tanaka and K. Aihara. Backpropagation learning algorithm for multilayer phasor neural networks. In *Proc. 16th Int. Conf. Neural Information Processing: Part 1*, Lecture Notes in Computer Science, pages 484–493, 2009.

[2] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. *Parallel Distributed Processing*, 1:318–362, 1986.

[3] N. Benvenuto and F. Piazza. On the complex backpropagation algorithm. *IEEE Trans. Signal Processing*, 40(4):967–969, 1992.

[4] T. Nitta. An extension of the back-propagation algorithm to complex numbers. *Neural Networks*, 10:1391–1415, 1997.

[5] G. M. Georgiou and C. Koutsougeras. Complex domain backpropagation. *IEEE Trans. CAS-II*, 39(5):330–334, 1992.

[6] A. Hirose. Continuous complex-valued back-propagation learning. *Electronics Lett.*, 28(20):1854–1855, 1992.

[7] A. J. Noest. Associative memory in sparse phasor neural networks. *Europhys. Lett.*, 6(6):469–474, 1988.

[8] A. J. Noest. Discrete-state phasor neural networks. *Phys. Rev. A*, 38:2196–2199, 1988.

[9] M. J. Kirby and R. Miranda. Circular nodes in neural networks. *Neural Comp.*, 8:390–402, 1996.