[4] S. Jockusch and H. Ritter, "Self–organizing maps: Local competition and evolutionary optimization," *Neural Netw.*, vol. 7, pp. 1229–1239, 1994.
[5] T. Kohonen, *Self-Organization and Associative Memory*, ser. Information Sciences, 3rd ed. Berlin, Germany: Springer-Verlag, 1989, vol. 8.
[6] H. Lu and S.-I. Amari, "Global exponential stability of multitime scale competitive neural," *IEEE Trans. Neural Netw.*, vol. 17, no. 5, pp. 1152–1164, Sep. 2006.
[7] A. J. Laub, *Matrix Analysis for Scientists & Engineers*. Philadelphia, PA: SIAM, 2005.
[8] B. Linares-Barranco, E. Sanchez-Sinencio, A. Rodriguez-Vazquez, and J. L. Huertas, "Modular Analog Continuous-Time VLSI Neural Networks with On-Chip Hebbian Learning and Analog Storage," in *Proc. IEEE Int. Symp. Circuits Syst.*, 1992, vol. 3, pp. 1533–1536.
[9] M. Lemmon and V. Kumar, "Emulating the dynamics for a class of laterally inhibited neural networks," *Neural Netw.*, vol. 2, pp. 193–214, 1989.
[10] A. Meyer-Bäse, F. Ohl, and H. Scheich, "Singular perturbation analysis of competitive neural networks with different time scales," *Neural Comput.*, vol. 8, no. 9, pp. 1731–1742, 1996.
[11] A. Meyer-Bäse and S. Pilyugin, "Flow invariance for competitive multi-modal neural networks," in *Proc. Int. Joint Conf. Neural Netw.*, 2003, vol. 4, pp. 3101–3105.
[12] A. Meyer-Bäse, S. Pilyugin, and Y. Chen, "Global exponential stability of competitive neural networks with different time scales," *IEEE Trans. Neural Netw.*, vol. 14, no. 3, pp. 716–719, May 2003.
[13] A. Meyer-Bäse, S. Pilyugin, A. Wismüller, and S. Foo, "Local exponential stability of competitive neural networks with different time scales," *Eng. Appl. Artif. Intell.*, vol. 17, no. 3, pp. 227–232, 2004.
[14] A. Menon, K. Mehrotra, C. K. Mohan, and S. Ranka, "Characterization of a class of sigmoid functions with applications to neural networks," *Neural Netw.*, vol. 9, no. 5, pp. 819–835, 1996.
[15] P. C. Parks and V. Hahn, *Stability Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1993.
[16] M. Vidyasagar, *Nonlinear Systems Analysis*. Englewood Cliffs, NJ: Prentice-Hall, 1993.
[17] D. Willshaw and C. von der Malsburg, "How patterned neural connections can be set up by self–organization," *Proc. R. Soc. Lond. B*, vol. 194, pp. 431–445, 1976.
[18] X. Xie, R. Hahnloser, and S. Seung, "Learning winner–take–all competition between groups of neurons in lateral inhibitory networks," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2000, vol. 13.
[19] T. Yoshizawa, *Stability Theory by Liapunov's Second Method*. Tokyo, Japan: The Mathematical Society of Japan, 1966, vol. 9.

A Normalized Adaptive Training of Recurrent Neural Networks With Augmented Error Gradient

Wu Yilei, Song Qing, and Liu Sheng

***Abstract*—For training algorithms of recurrent neural networks (RNN), convergent speed and training error are always two contradictory performances. In this letter, we propose a normalized adaptive recurrent learning (NARL) to obtain a tradeoff between transient and steady-state response. An augmented term is added to error gradient to exactly model the derivative of cost function with respect to hidden layer weight. The influence of the induced gain of activation function on training stability is also taken into consideration. Moreover, adaptive learning rate is employed to improve the robustness of the gradient training. Fianlly, computer simulations of a model prediction problem are synthesized to give comparisons between NARL and conventional normalized real-time recurrent learning (N-RTRL).**

***Index Terms*—Adaptive learning rate, augmented error gradient, convergence, normalization.**

I. Introduction

Recurrent neural network (RNN) has been an active research topic in the past decades due to its promising performance of modeling nonstationary signals [1]–[4]. Numerous applications can be found in various disciplines [3], [5]. However, improvement of training method is necessary because conventional algorithm always suffers from poor transient response, such as real-time recurrent learning (RTRL). In literature, many works have been carried out to address the issue [6]–[9]. For example, Rupp *et al.* proposed an adaptive training for a single perceptron neuron to improve the convergence based upon analysis via small gain theorem [10], [11]. Recently, a normalized RTRL (N-RTRL) has been introduced by Mandic *et al.* [2], [3], [12].

In this letter, we propose a new training algorithm of RNN, namely, normalized adaptive recurrent learning (NARL), which employs both adaptive learning rate and normalization factors to improve the robustness as we will show in Section III. Further, the effect of induced gain of activation function is taken into consideration to improve the necessity of derived stability criterion. Another important advantage of the new algorithm is that an augmented error gradient term is added to pursue an accurate calculation of derivative of cost function. Finally, we carry out simulations to show the performance of NARL is improved in terms of steady-state response in comparison with N-RTRL.

II. Normalized Adaptive Recurrent Learning

Consider a multiple-input–single-output (MISO) RNN with m hidden layer neurons as shown in Fig. 1 [13]. The estimated weight matrices of the RNN are expressed by $\hat{V}(k) \in R^{1\times n}$ (output layer) and $\hat{W}(k) \in R^{n\times m}$ (hidden layer), respectively. Upon an input $u(k)$, the corresponding RNN output is calculated as

$$\hat{y}(k) = \hat{V}(k)\Phi(\hat{W}(k)\cdot x(k)) \qquad (1)$$

where $\Phi(\cdot)$ is the nonlinear activation function, $\hat{y}(k)$ is output vector, and $x(k)$ is the state vector that consists of external inputs and delayed output feedback entries

$$x(k) = [u(k), \ldots, u(k-l+1), \hat{y}(k-1), \ldots, \hat{y}(k-m+l)]^T \quad (2)$$

Manuscript received March 30, 2007; revised July 4, 2007; accepted July 15, 2007.

The authors are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore (e-mail: bieber@pmail.ntu.edu.sg).

Digital Object Identifier 10.1109/TNN.2007.908647

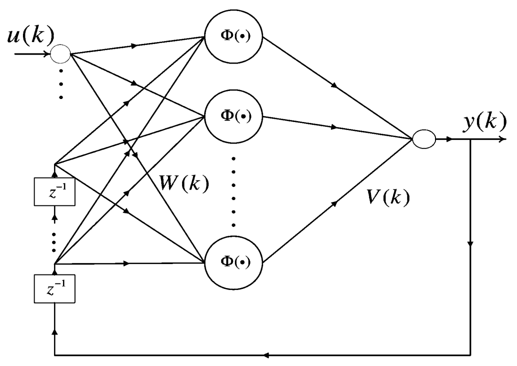

Fig. 1. Structure of an external feedback recurrent neural network.

where superscript T denotes transpose operation. To simplify the expression, we use notation $\Phi(k)$ instead of $\Phi(\hat{W}(k)\hat{x}(k))$ hereafter.

When estimating a command signal $d_n(k)$, the instantaneous estimation error of RNN is defined by (3). Without loss of generality, the signal $d_n(k)$ contains both background signal $d(k)$ and noise $\varepsilon(k)$. There is no assumption on the prior knowledge of noise statistics in analysis of this letter

$$\begin{cases} e(k) = d_n(k) - \hat{y}(k) \\ d_n(k) = d(k) + \varepsilon(k) \end{cases}. \tag{3}$$

In an environment of time-varying signal statistics, gradient-type algorithms can be used to reduce cost function $f(e) = e(k)^2/2$ by estimating the weight at each time instant. Now, we propose the NARL training algorithm shown in (4) at the bottom of the page, where adaptive $\alpha_v(k)$ and fixed α_w are learning rates, $\rho_v(k)$ and $\rho_w(k)$ are normalization factors, and $\gamma_w(k)$ is the so-called augmented error gradient that compensates the gradient approximation error.

III. Robustness Analysis

For the convenience of exposition, some notations are introduced as

$$\begin{aligned} \tilde{V}(k) &= V^* - \hat{V}(k) \\ \Delta\Phi(k+1) &= \Phi(k+1) - \Phi(k) \\ \Delta\hat{W}(k+1) &= \hat{W}(k+1) - \hat{W}(k) \quad \tilde{W}(k) = W^* - \hat{W}(k) \\ \Delta x(k+1) &= x(k+1) - x(k) \\ \Delta d_n(k+1) &= d_n(k+1) - d_n(k) \\ \tilde{\Phi}(k) &= \Phi^* - \Phi(k) = \Phi(W^*x^*) - \Phi(\hat{W}(k)x(k)) \end{aligned} \tag{5}$$

where $V^*(k)$ and $W^*(k)$ are optimal weights matrices and $x^*(k)$ is fixed local attractor of RNN corresponding to the expected signal $d(k)$.

Proposition 1: If the learning rate α_w of NARL in (4) is chosen in the interval $(0, 1)$, and $\alpha_v(k)$ is calculated by

$$\alpha_v(k) = \begin{cases} 1 + \dfrac{\Delta d_n(k+1)}{e(k)}, & |e(k)| > \xi \\ 1, & |e(k)| \le \xi \end{cases} \tag{6}$$

and the normalization factors $\rho_v(k)$ and $\rho_w(k)$ and the augmented error gradient $\gamma_w(k) \in R^{m\times n}$ are determined as

$$\begin{aligned} \rho_v(k) &= \|\Phi(k)\|^2 \\ \rho_w(k) &= \phi_1 \cdot \phi_2 \cdot \|\hat{V}(k+1)\|^2 \|x(k)\|^2 \end{aligned} \tag{7}$$

$$\begin{aligned} \gamma_w(k) = &-\frac{x(k)}{x(k)^T x(k+1)} \\ &\cdot \left[\frac{\rho_w(k)}{\alpha_w e(k^+)}\hat{W}(k)\Delta x(k+1) \right. \\ &\left. + \operatorname{diag}\{\Phi(k)\}\hat{V}(k+1)^T x(k)^T \Delta x(k+1)\right]^T \end{aligned} \tag{8}$$

where ξ is a small positive number, ϕ_1 and ϕ_2 are induced norm of activation function of RNN and its first-order derivative, respectively, which is defined by

$$\phi_1 = \max\{|\Phi(\cdot)|\} \quad \phi_2 = \max\{|\Phi'(\cdot)|\} \tag{9}$$

then the training will be stable in the sense that $e(k)$, $\tilde{V}(k)$, and $\tilde{W}(k)$ are bounded (convergent) as long as $\Delta d_n(k)$ is bounded (convergent).

Proof: To study the stability of the algorithm, we start with establishing the error dynamics of the training. Because the training is separated into output and hidden layer, respectively, we should also partition the analysis into two subsystems. First, by decomposing the posterior error and then substituting $\hat{V}(k+1)$ and $\tilde{V}(k+1)$ into it, we can derive

$$\begin{aligned} e(k^+) &= d_n(k) - \hat{V}(k+1)\Phi(k) \\ &= V^*(k)\Phi^*(k) + \varepsilon(k) - \hat{V}(k+1)\Phi(k) \\ &= \tilde{V}(k+1)\Phi(k) + \tilde{V}(k+1)\tilde{\Phi}(k) + \hat{V}(k+1)\tilde{\Phi}(k) + \varepsilon(k) \\ &= [1 - \alpha_v(k)]e(k). \end{aligned} \tag{10}$$

Next, we expand the modeling error at time step $k+1$ as

$$\begin{aligned} e(k+1) &= d_n(k+1) - \hat{y}(k+1) \\ &= d_n(k) + \Delta d_n(k+1) - \hat{V}(k+1)\Phi(k+1) \\ &= d_n(k) - \hat{V}(k+1)\Phi(k) + \hat{V}(k+1)\Phi(k) \\ &\quad - \hat{V}(k+1)\Phi(k+1) + \Delta d_n(k+1) \\ &= e(k^+) - \hat{V}(k+1)\Delta\Phi(k+1) + \Delta d_n(k+1) \end{aligned} \tag{11}$$

by the mean value theorem

$$\begin{aligned} \hat{V}(k+1)\Delta\Phi(k+1) &= \hat{V}(k+1)[\Phi(k+1) - \Phi(k)] \\ &= \hat{V}(k+1)\Big[\Phi(\hat{W}(k+1)x(k+1)) \\ &\qquad - \Phi(\hat{W}(k)x(k))\Big] \\ &= \hat{V}(k+1)\operatorname{diag}\{\Phi'(\delta(k+1))\} \\ &\quad \cdot [\hat{W}(k+1)x(k+1) - \hat{W}(k)x(k)] \\ &= \hat{V}(k+1)\operatorname{diag}\{\Phi'(\delta(k+1))\} \\ &\quad \cdot [\hat{W}(k+1)\Delta x(k+1) + \Delta\hat{W}(k+1)x(k)] \end{aligned} \tag{12}$$

$$\begin{cases} \hat{V}(k+1) = \hat{V}(k) + \dfrac{\alpha_v(k)e(k)}{\rho_v(k)}\Phi(k)^T \\ \hat{W}(k+1) = \hat{W}(k) + \dfrac{\alpha_w e(k^+)}{\rho_w(k)}[x(k)\hat{V}(k+1)\operatorname{diag}\{\Phi(k)\} + \gamma_w(k)]^T \end{cases} \tag{4}$$

where $\Phi'(\cdot)$ is the first-order derivative of activation function $\Phi(k)$ on input space, $\delta(k+1)$ is an unknown vector with all the entries between those of $\hat{W}(k+1)x(k+1)$ and $\hat{W}(k)x(k)$, and $\mathrm{diag}\{\Phi'(\cdot)\}$ means a diagonal matrix which has each entry of $\Phi'(\cdot)$ on the main diagonal. Further, by the training of (4), we have

$$\begin{aligned}\Delta\hat{W}(k+1)x(k) &= \frac{\alpha_w e(k^+)}{\rho_w(k)}\Big[x(k)\hat{V}(k+1)\mathrm{diag}\{\Phi(k)\} \\ &\qquad + \gamma_w(k)\Big]^T x(k) \\ &= \frac{\alpha_w e(k^+)}{\rho_w(k)}\mathrm{diag}\{\Phi(k)\}\hat{V}(k+1)^T x(k)^T x(k) \\ &\quad + \frac{\alpha_w e(k^+)}{\rho_w(k)}\gamma_w(k)^T x(k). \qquad (13)\end{aligned}$$

We substitute (12) and (13) into (11), then the error dynamics of hidden layer is obtained

$$\begin{aligned}e(k+1) = e(k^+) &- \frac{\alpha_w e(k^+)}{\rho_w(k)}\hat{V}(k+1)\mathrm{diag}\{\Phi'(\delta(k+1))\} \\ &\cdot \mathrm{diag}\{\Phi(k)\}\hat{V}(k+1)^T x(k)^T x(k) + \Delta d_n(k+1) \\ &- \hat{V}(k+1)\mathrm{diag}\{\Phi'(\delta(k+1))\} \\ &\cdot \left[\hat{W}(k+1)\Delta x(k+1) + \frac{\alpha_w e(k^+)}{\rho_w(k)}\gamma_w(k)^T x(k)\right]. \qquad (14)\end{aligned}$$

Then, we substitute $\hat{W}(k+1)$ of (4) into the last term on the right-hand side of (14)

$$\begin{aligned}&\hat{W}(k+1)\Delta x(k+1) + \frac{\alpha_w e(k^+)}{\rho_w(k)}\gamma_w(k)^T x(k) \\ &= \hat{W}(k)\Delta x(k+1) + \frac{\alpha_w e(k^+)}{\rho_w(k)} \\ &\quad \cdot [x(k)\hat{V}(k+1)\mathrm{diag}\{\Phi(k)\} + \gamma_w(k)]^T \Delta x(k+1) \\ &\quad + \frac{\alpha_w e(k^+)}{\rho_w(k)}\gamma_w(k)^T x(k) \\ &= \hat{W}(k)\Delta x(k+1) + \frac{\alpha_w e(k^+)}{\rho_w(k)} \\ &\quad \cdot \mathrm{diag}\{\Phi(k)\}\hat{V}(k+1)^T x(k)^T \Delta x(k+1) \\ &\quad + \frac{\alpha_w e(k^+)}{\rho_w(k)}\gamma_w(k)^T x(k+1). \qquad (15)\end{aligned}$$

According to the definition of $\gamma_w(k)$ in (8) and by some trivial computations

$$\begin{aligned}&\frac{\alpha_w e(k^+)}{\rho_w(k)}\gamma_w(k)^T x(k+1) \\ &\quad = -\frac{\alpha_w e(k^+)}{\rho_w(k)x(k)^T x(k+1)} \\ &\qquad \cdot \left[\frac{\rho_w(k)}{\alpha_w e(k^+)}\hat{W}(k)\Delta x(k+1)\right. \\ &\qquad\quad \left. + \mathrm{diag}\{\Phi(k)\}\hat{V}(k+1)^T x(k)^T \Delta x(k+1)\right] \\ &\qquad \cdot x(k)^T x(k+1) \\ &\quad = -\hat{W}(k)\Delta x(k+1) - \frac{\alpha_w e(k^+)}{\rho_w(k)} \\ &\qquad \cdot \mathrm{diag}\{\Phi(k)\}\hat{V}(k+1)^T x(k)^T \Delta x(k+1). \qquad (16)\end{aligned}$$

Thus, by the results of (15) and (16), the following condition can be derived:

$$\hat{W}(k+1)\Delta x(k+1) + \frac{\alpha_w e(k^+)}{\rho_w(k)}\gamma_w(k)^T x(k) = 0. \qquad (17)$$

If we define the operator

$$\begin{aligned}\Lambda(k) = 1 - \frac{\alpha_w}{\rho_w(k)}&\hat{V}(k+1)\mathrm{diag}\{\Phi'(\delta(k+1))\} \\ &\cdot\mathrm{diag}\{\Phi(k)\}\hat{V}(k+1)^T x(k)^T x(k) \qquad (18)\end{aligned}$$

then the error dynamics of the overall NARL algorithm in (14) can be simplified as

$$\begin{aligned}e(k+1) &= [1-\Lambda(k)]e(k^+) + \Delta d_n(k+1) \\ &= [1-\Lambda(k)][1-\alpha_v(k)]e(k) + \Delta d_n(k+1). \qquad (19)\end{aligned}$$

Equation (19) actually reflects the propagation of the modeling error between each full training step. The dynamics not only subject to the external disturbance, but also the normalized operator $\Lambda(k)$ and $\alpha_v(k)$. Because $\Lambda(k)$ is unknown due to that there is no feasible access to obtain $\delta(k)$, the dynamics can only be controlled by learning rate $\alpha_v(k)$. Under this framework, we may proceed to study the stability of NARL by dividing both sides of (19) by $\Delta d_n(k+1)$, which will result in

$$\frac{e(k+1)}{\Delta d_n(k+1)} = \frac{e(k)}{\Delta d_n(k+1)}\cdot[1-\alpha_v(k)][1-\Lambda(k)] + 1. \qquad (20)$$

To this end, we investigated the definition of $\rho_w(k)$ and $\alpha_v(k)$ and derived that

$$\begin{aligned}&\frac{\alpha_w}{\rho_w(k)}\hat{V}(k+1)\mathrm{diag}\{\Phi'(\delta(k+1))\}\cdot\mathrm{diag}\{\Phi(k)\}\hat{V}(k+1)^T \\ &\qquad\qquad x(k)^T x(k) < 1 \qquad (21)\\ &\frac{e(k)}{\Delta d_n(k+1)}\cdot[1-\alpha_v(k)] = \begin{cases}-1, & |e(k)| > \xi \\ 0, & |e(k)| \le \xi\end{cases}. \qquad (22)\end{aligned}$$

Thus, the following two inequalities can be obtained:

$$0 < \Lambda(k) < 1 \quad -1 \le \frac{e(k)}{\Delta d_n(k+1)}\cdot[1-\alpha_v(k)] \le 0 \qquad (23)$$

which actually ensures $|(e(k+1)/\Delta d_n(k+1))| \le 1$. Thus, the boundedness (convergence) of modeling error of NARL algorithm is guaranteed as long as $\Delta d_n(k)$ is bounded (convergent). Hence, we conclude the proof.

Remark 1: Indeed, the previously proposed NARL is a generalized form of training and many of existing algorithms can be derived as its special forms. For example, if we set $\rho_v = \rho_w = 1$ and $\gamma_w(k) = 0$, then the algorithm becomes the RTRL introduced by Williams and Zipser [14]. If we take into account the normalization factor, then it will become an N-RTRL algorithm. The most remarkable feature of NARL is the improved robustness compared to the conventional training algorithms. This point can be explained by looking at the ratio $(e(k+1)/\varepsilon(k+1)) = (e(k+1)/\Delta d_n(k+1))\cdot(\Delta d_n(k+1)/\varepsilon(k+1))$, which is desired to be as small as possible. When a constant learning rate is employed, say, $0 < \alpha < 1$, then the term $(e(k)/\Delta d_n(k+1))[1-\alpha_v(k)]$ can be probably larger than zero if $e(k)$ and $\Delta d_n(k+1)$ have the same sign. In this case, the ratio of $(e(k+1)/\Delta d_n(k+1))$ will be larger than 1. In other words, there is no rejection of the input disturbance, or the influence of the disturbance variation is amplified. With the proposed adaptive learning rate, the sensitivity can be ensured to be less than unit. Hence, in this manner, the robustness is improved.

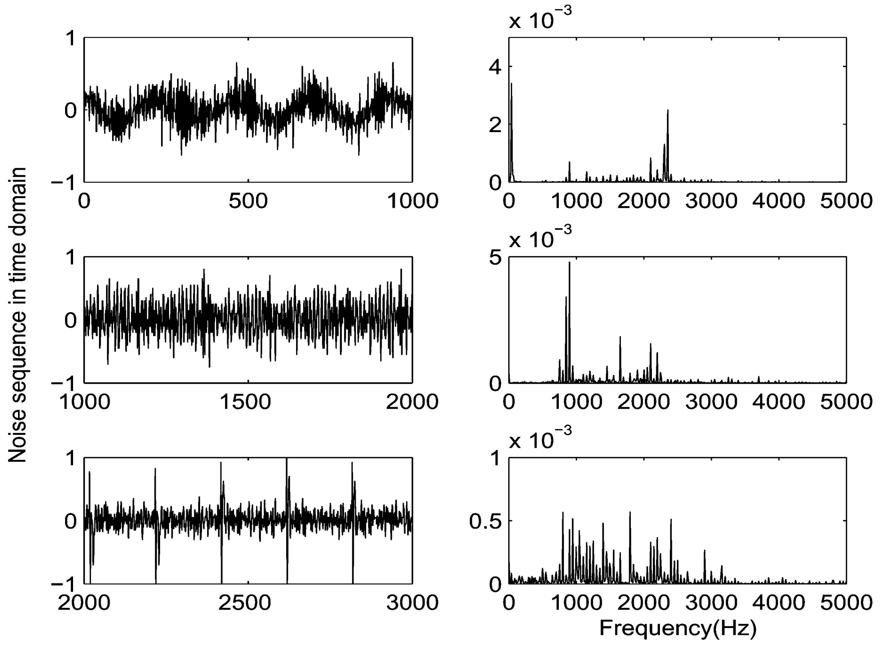

Fig. 2. Time sequence of noise and its corresponding power spectrum.

TABLE I
COEFFICIENTS OF HAMMERSTEIN–WIENER MODEL

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9
0.768	0.287	0.115	0.214	0.644	0.356	0.020	0.020	0.095
b_1	b_2	b_3						
0.006	0.03	0.008						

IV. EXPERIMENTAL SIMULATIONS

In this section, we study NARL training algorithm via simulations on a model prediction problem. RNN is utilized to predict the output of unknown plant upon input signal $u(k)$. We employ a Hammerstein–Wiener model to generate the reference signal $d_n(k)$ [15], which is described by (24), shown at the bottom of the page, where $u(k)$ is the white Gaussian sequence, $\varepsilon(k)$ is an external disturbance with statistics varied at time steps $k = 1000$ and $k = 2000$, respectively (statistical nonstationary), as shown in Fig. 2, and the coefficients of finite-input response (FIR) filer is defined in Table I.

RNN is configured as a ten-input–one-output network with 50 neurons. Input vector consists of five entries of output feedback (1–5 steps delay, respectively) and five entries of external input ($d_n(k)$ and its delays). The weight matrices of RNN are initialized as uniformly distributed in the interval of $(-1, 1)$. Sigmoid function is chosen as the activation function

$$\Phi(x) = \frac{1}{1+e^{-\lambda x}} \leq 1 \quad \Phi'(x) = \frac{\lambda e^{-\lambda x}}{(1+e^{-\lambda x})^2} \leq \frac{\lambda}{4}. \tag{25}$$

The simulation runs for 3000 steps and the squared prediction error is chosen as the measure of training performance. To obtain a comparative idea about the performance, the filtering results of RTRL algorithm with various fixed learning rate and N-RTRL ($\alpha_v = 1$ and $\gamma_w = 0$) are also provided. In order to present a clear illustration on both transient and steady-state performance, the first 100 steps and full 3000 steps of training error are shown separately and the time axis of transient response plot is expressed in logarithmic form as shown in Fig. 3. Moreover, the steady-state prediction errors in Fig. 4 are expressed in decibels such that performance difference can be more apparent (between NARL and N-RTRL). The traces of $\gamma_w(k)$ and $\alpha_v(k)$ are displayed in Fig. 5.

The figures indicate that the final trained RNNs, using all the algorithms, are successfully stabilized. Moreover, there is no much difference between converging speed of N-RTRL and NARL among various learning rates (both within 20 steps). On the contrary, NARL can achieve better steady-state response than N-RTRL. The mean of squared prediction error of NARL is -29.04 dB, while N-RTRL is -25.88 dB, which is improved by about 12%. This result is tallied with the theoretical analysis in Section III.

V. CONCLUSION

In this letter, NARL algorithm is investigated. The algorithm improved the training performance of RNN in the sense of robustness, on basis of introducing three new elements: normalization factors, adaptive learning rate, and augmented error gradient. Theoretical analysis is presented to justify the stability of NARL. We emphasize that the induced gain of activation function is added in normalization factor to improve the necessity of the stability criterion. Simulation experiments are carried out to verify the theoretical derivations. The results show that the learning performance of NARL is improved in terms of the steady-state response in comparison with N-RTRL algorithm.

$$\begin{cases} x(k) = 0.3\tanh(0.5u(k)) + 0.5\tanh(0.8u(k)) \\ y(k) = a_1 x(k) + a_2 e^{-b_1 k} x(k-1) - a_3 e^{-b_2 k} x(k-2) + a_4 x(k-3) - a_5 \sin(b_3 k) x(k-4) \\ \qquad +a_6 x(k-5) + a_7 x(k-6) - a_8 x(k-7) + a_9 x(k-8) + \varepsilon(k) \end{cases} \tag{24}$$

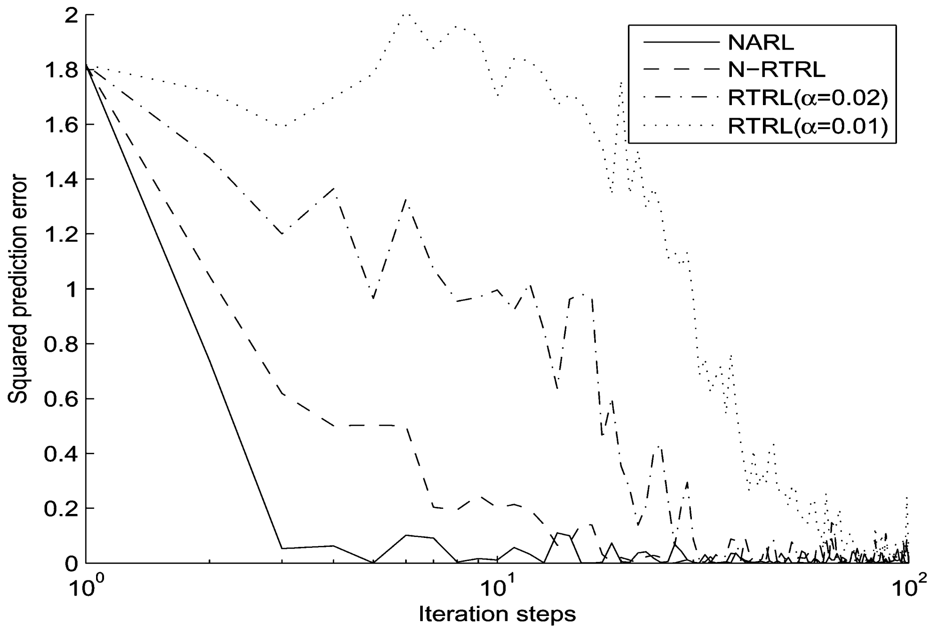

Fig. 3. Squared prediction errors of first 100 steps of NARL, RTRL, and N-RTRL training results, respectively (transient response), with time axis in logarithmic form.

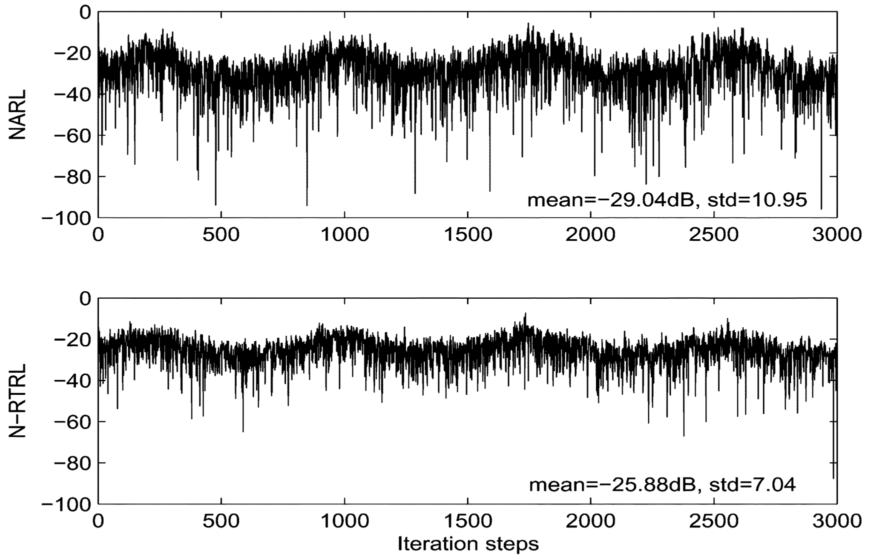

Fig. 4. Prediction errors (in decibels) of full 3000 steps of NARL and N-RTRL training results, respectively (steady-state response).

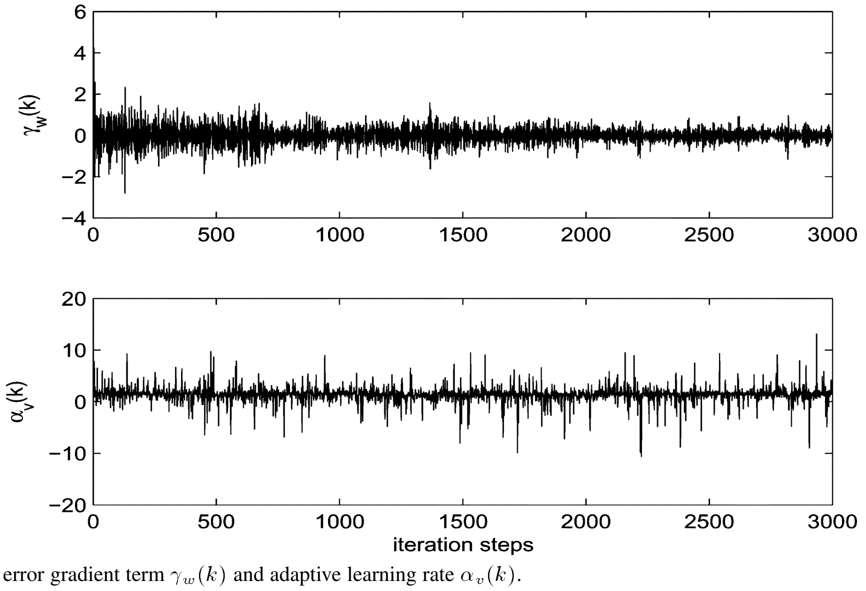

Fig. 5. Traces of augmented error gradient term $\gamma_w(k)$ and adaptive learning rate $\alpha_v(k)$.

Acknowledgment

The authors would like to thank two anonymous reviewers for their insightful comments and valuable suggestions on the earlier versions of this letter.

References

[1] S. Haykin, *Neural Networks*. Upper Saddle River, NJ: Prentice-Hall, 1999.

[2] "A normalised real time recurrent learning algorithm," *Elsevier Signal Processing*, vol. 80, no. 9, pp. 1909–1916, 2000.

[3] D. P. Mandic and J. A. Chambers, *Recurrent Neural Networks for Prediction: Learning Algorithms, Architecture and Stability*. Chichester, U.K.: Wiley, 2001.

[4] Oliver and Nelles, *Nonlinear System Identification*. New York: Springer-Verlag, 2001.

[5] F. C. Sun, Z. Q. Sun, and P. Y. Woo, "Neural network-based adaptive controller design of robotic manipulators with an observer," *IEEE Trans. Neural Netw.*, vol. 12, no. 1, pp. 54–67, Jan. 2001.

[6] R. Williams and J. Peng, "Gradient-based learning algorithms for recurrent neural networks," *Neural Comput.*, vol. 2, pp. 490–501, 1990.

[7] M. W. Mak, K. W. Ku, and Y. L. Lu, "On the improvement of the real time recurrent learning algorithm for recurrent neural networks," *Neurocomputing*, vol. 24, pp. 13–36, Feb. 1999.

[8] Q. Song, J. Xiao, and Y. C. Soh, "Robust back-propagation training algorithm for multi-layered neural tracking controller," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 1133–1141, Sep. 1999.

[9] Y. L. Wu, Q. Song, and X. L. Yang, "Robust recurrent neural network control of biped robot," *J. Intell. Robot. Syst.*, vol. 49, no. 2, pp. 151–169, Jun. 2007.

[10] M. Rupp and A. H. Sayed, "A time-domain feedback analysis of filtered-error adaptive gradient algorithms," *IEEE Trans. Signal Process.*, vol. 44, no. 6, pp. 1428–1439, Jun. 1996.

[11] M. RuppAli and H. Sayed, "Supervised learning of perceptron and output feedback dynamic networks: a feedback analysis via the small gain theorem," *IEEE Trans. Neural Netw.*, vol. 8, no. 3, pp. 612–622, May 1997.

[12] D. P. Mandic, "Data-reusing recurrent neural adaptive filters," *Neural Comput.*, vol. 14, pp. 2693–2707, 2002.

[13] A. D. Back, "Locally recurrent globally feedforward networks: A critical review of architectures," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 229–239, Mar. 1994.

[14] R. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.*, vol. 1, pp. 270–280, 1989.

[15] A. E. Nordsjo and L. H. Zetterberg, "Identification of certain time-varying nonlinear Wiener and Hammerstein systems," *IEEE Trans. Signal Process.*, vol. 49, no. 3, pp. 577–592, Mar. 2001.

Cline: A New Decision-Tree Family

M. Fatih Amasyalı and Okan Ersoy

***Abstract*—A new family of algorithm called Cline that provides a number of methods to construct and use multivariate decision trees is presented. We report experimental results for two types of data: synthetic data to visualize the behavior of the algorithms and publicly available eight data sets. The new methods have been tested against 23 other decision-tree construction algorithms based on benchmark data sets. Empirical results indicate that our approach achieves better classification accuracy compared to other algorithms.**

***Index Terms*—Classifier combination, decision forests, decision trees, learning (artificial intelligence), machine learning, multivariate decision trees, pattern classification, pattern recognition.**

I. Introduction

Decision-tree algorithms are among the most used tools in pattern recognition applications. A typical decision-tree algorithm begins with all the data, splits the data into two subsets based on the values of one or more attributes on decision nodes, and then recursively splits each subset into finer ones, until each subset contains a single class. These final subsets form the leaf nodes of the decision tree. The classification process starts at the root node. The data follows the directions of the decision nodes until it reaches a leaf. Upon reaching a leaf, the class of the data is assigned as the class of the leaf. Decision trees may be univariate, linear multivariate, or nonlinear multivariate depending on whether a single attribute, a linear function of all the attributes, or a nonlinear function of all the attributes is used for the partitioning at each node of the decision tree [13]. If the tree is univariate, its decision nodes hold only one feature and one threshold value. The data coming into a particular node is directed to a particular branch according to whether the feature of the data on this node is greater than the threshold value. Multivariate trees have hyperplanes in their nodes. Each hyperplane $(w_{1..p+1})$ consists of a linear combination of all features [1], [14], and implements (1). In (1), each data sample has p features

$$f(x) : w_m^T x + w_{m0} = \sum_{j=1}^{p} w_{mj} x_j + w_{m0} > 0. \tag{1}$$

An incoming data of a node is directed to a particular branch according to which part of the hyperplane the data is on. Univariate and multivariate decision nodes which are constructed for the same data set are illustrated in Fig. 1. In Fig. 1, $u1$, $u2$, and $u3$ univariate splitters are needed for a decision which can be carried out by only one multivariate splitter $(m1)$ shown. Many studies show that the multivariate trees are generally smaller and more accurate but less understandable [2].

To be able to compute the weighted sum in (1), all the features should be numeric and discrete or nonordered values need be represented numerically (usually by 1-of-L encoding) beforehand [1].

Manuscript received October 14, 2006; revised February 18, 2007 and June 22, 2007; accepted July 2, 2007. An early version of this work was presented at the ICSC Congress on Computational Intelligence Methods and Applications (CIMA'05), Istanbul, Turkey.

M. F. Amasyalı is with the Computer Engineering Department, Yıldız Technical University, Istanbul 34349, Turkey (e-mail: mfatih@ce.yildiz.edu.tr).

O. Ersoy is with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: ersoy@purdue.edu).

Color versions of one or more of the figures in this letter are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2007.910729