PERGAMON

Neural Networks 12 (1999) 1341–1345

Neural Networks

www.elsevier.com/locate/neunet

Contributed article

Exploiting inherent relationships in RNN architectures

D.P. Mandic[1,*], J.A. Chambers[2]

Communications and Signal Processing Group, Department of Electrical and Electronic Engineering, Imperial College of Science, Technology and Medicine, Exhibition Road, London SW7 2BT, UK

Received 11 March 1999; accepted 5 July 1999

Abstract

We provide the relationship between the learning rate and the slope of a nonlinear activation function of a neuron within the framework of nonlinear modular cascaded systems realised through Recurrent Neural Network (RNN) architectures. This leads to reduction in the computational complexity of learning algorithms which continuously adapt the weights of such architectures, because there is a smaller number of independent parameters to optimise. Results are provided for the Gradient Descent (GD) learning algorithm and the Extended Recursive Least Squares (ERLS) algorithm, using a general nonlinear activation function of a neuron. The results obtained degenerate into the corresponding results for single RNNs, when considering only one module in such cascaded systems.

Keywords: Recurrent Neural Networks; Learning; Gradient Descent; Extended Recursive Least Squares; Extended Kalman filter; Pipelined Recurrent Neural Network; Modular RNNs; Equivalent networks

1. Introduction

There has been a recent interest in nonlinear cascaded systems realised as Recurrent Neural Networks (RNN)s (Haddad, Chellaboina & Fausz, 1998; Narendra & Parthasarathy, 1990; Suykens, DeMoor & Vanderwalle, 1997). An example of such systems is the so-called Pipelined Recurrent Neural Network (PRNN) (Haykin & Li, 1995). It is a complex modular network which consists of a number of small-scale RNNs, and has been shown to represent nonlinear Wiener–Hammerstein cascaded systems (Mandic & Chambers, 1999a). However, a learning algorithm for such a complex network has to perform a nonlinear optimisation task on a number of parameters (Baltersee & Chambers, 1998; Mandic & Chambers, 1998). For a learning algorithm to be optimal, it has to adjust adaptively all the parameters of the network, such as a learning rate, slope of the activation function, forgetting factor in a cost function, and even the number of modules in such networks, which is rather complex. Hence, if a relationship between some of the parameters inherent to the PRNN can be established, the number of the degrees of freedom in such an optimisation task would be reduced, as well as the corresponding computational complexity. This is particularly important for real-time applications, such as nonlinear prediction.

Thimm, Moerland and Fiesler (1996) provided the relationship between the slope of the logistic activation function β and the learning rate η for a class of general feedforward neural networks trained by backpropagation, which use the logistic nonlinear activation function. Similar relationships have also been derived for the case of single recurrent neural networks (Mandic & Chambers, 1999b). However, although these results help to reduce the problem of computational complexity in learning, they are dependent on the underlying learning algorithm, which was a gradient descent algorithm. Here, we provide the static and dynamic equivalence between a nonlinear cascaded system realised through a PRNN described by β, η and $\mathbf{W}(n)$, and a referent PRNN described by $\beta^{\mathrm{R}} = 1$, η^{R}, and $\mathbf{W}^{\mathrm{R}}(n)$, and show that there is a deterministic relationship between them, which allows one degree of freedom less in the nonlinear optimisation task of learning in the PRNN framework. The relationships are provided for both the Gradient Descent (GD) and the Extended Recursive Least Squares (ERLS) learning algorithms based upon Baltersee and Chambers (1998), Haykin, Sayed, Zeidler, Yee and Wei (1997), Mandic, Baltersee and Chambers (1998), and a general nonlinear activation function of a neuron. The results obtained boil down to the results for the RNN, for single module (cascade) systems.

* Corresponding author. Present address: School of Information Systems, University of East Anglia, Norwich NR4 7TJ, UK. Tel.: + 44-171-594-6300; fax: + 44-171-594-6234.

E-mail addresses: d.mandic@uea.ac.uk (D.P. Mandic), j.chambers@ic.ac.uk (J.A. Chambers)

[1] http://www.dsp.ee.ic.ac.uk/ ~ mandic.

[2] http://www.dsp.eeiic.ac.uk/ ~ jonc.

0893-6080/99/$ - see front matter
PII: S0893-6080(99)00076-3

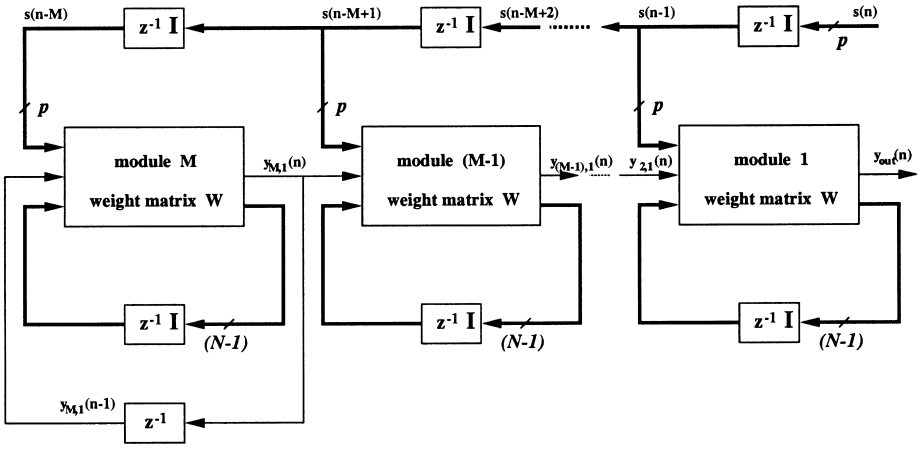

Fig. 1. Pipelined recurrent neural network.

In Section 2, the PRNN is briefly introduced. In Section 3, conditions of static equivalence between an arbitrary PRNN and the referent PRNN are derived. Further, in Section 4, conditions of the dynamic equivalence between an arbitrary PRNN and the referent PRNN are provided for the most commonly used learning algorithms. Section 5 concludes the letter.

2. The PRNN

The PRNN is a modular neural network, and consists of a certain number M of RNNs as its modules, with each module consisting of N neurons. In the PRNN configuration, the M modules, which are RNNs, are connected as shown in Fig. 1. The $(p \times 1)$ dimensional external signal vector $\mathbf{s}^{\mathrm{T}}(n) = [s(n-1), \ldots, s(n-p)]$ is delayed by m time steps ($z^{-m}\mathbf{I}$) before feeding the module m. All the modules operate using the same weight matrix $\mathbf{W}$. The overall output signal of the PRNN is $y_{\mathrm{out}}(n) = y_{1,1}(n)$, i.e. the output of the first neuron of the first module, as shown in Fig. 1. The equations which describe the PRNN are

$$y_{i,k}(n) = \Phi(v_{i,k}(n)), \qquad i = 1, \ldots, M, \;\; k = 1, \ldots, N \tag{1}$$

$$v_{i,k}(n) = \sum_{l=1}^{p+N+1} w_{k,l}(n) u_{i,l}(n), \quad i = 1, \ldots, M, \;\; k = 1, \ldots, N \tag{2}$$

$$\begin{aligned} u_i^{\mathrm{T}}(n) &= [s(n-i), \ldots, s(n-i-p+1), 1, y_{i+1,1}(n), \\ &\quad y_{i,2}(n-1), \ldots, y_{i,N}(n-1)], \qquad 1 \le i \le M-1 \\ u_M^{\mathrm{T}}(n) &= [s(n-M), \ldots, s(n-M-p+1), 1, y_{M,1}(n-1), \\ &\quad y_{M,2}(n-1), \ldots, y_{M,N}(n-1)], \qquad i = M \end{aligned} \tag{3}$$

where $y_{i,k}$ corresponds to the output of the kth neuron of the ith module in the PRNN. The nonlinear activation function commonly used is the logistic function

$$\Phi(x) = \frac{1}{1 + \mathrm{e}^{-\beta x}} \tag{4}$$

3. Static equivalence between an arbitrary and a referent PRNN

By static equivalence between two PRNNs, we consider the equivalence between the outputs of the neurons in an arbitrary PRNN, and the outputs of the corresponding neurons in a referent PRNN, for a fixed time-instant n, i.e.

$$\begin{aligned} y_{i,k}(n) = y_{i,k}^{\mathrm{R}}(n) &\Leftrightarrow \Phi(\beta, \mathbf{w}_{i,k}(n), \mathbf{u}_{i,k}(n)) \\ &= \Phi(1, \mathbf{w}_{i,k}^{\mathrm{R}}(n), \mathbf{u}_{i,k}(n)) \Leftrightarrow \beta \mathbf{w}_{i,k}(n) = \mathbf{w}_{i,k}^{\mathrm{R}}(n) \end{aligned} \tag{5}$$

where $\mathbf{w}_{i,k}(n)$ and $\mathbf{u}_{i,k}(n)$ are, respectively, the set of weights and the set of inputs which belong to the neuron k in module i. An example for the logistic nonlinearity (Thimm et al., 1996)

$$\frac{1}{1 + \mathrm{e}^{-\beta \mathbf{u}_{i,k}^{\mathrm{T}} \mathbf{w}_{i,k}}} = \frac{1}{1 + \mathrm{e}^{-\mathbf{u}_{i,k}^{\mathrm{T}} \mathbf{w}_{i,k}^{\mathrm{R}}}} \Leftrightarrow \beta \mathbf{w}_{i,k} = \mathbf{w}_{i,k}^{\mathrm{R}} \tag{6}$$

where the time index (n) is neglected, supports the result for a general nonlinear activation function, given in expression (5).

The following Lemma, which is independent of the underlying learning algorithm for the PRNN, comprises the above analysis.

Lemma 1. *An arbitrary PRNN with weight matrix $\mathbf{W}(n)$, and a slope in the activation function β, is equivalent in the static sense to the isomorphic referent PRNN, characterised by $\mathbf{W}^{\mathrm{R}}(n)$, and $\beta^{\mathrm{R}} = 1$, if*

$$\beta \mathbf{W}(n) = \mathbf{W}^{\mathrm{R}}(n) \tag{7}$$

for every discrete time instant n while the networks are running.

4. Dynamic equivalence between an arbitrary and a referent PRNN

While by static equivalence between two PRNNs we consider the calculation of the outputs of the networks, for a given weight matrix and input vector, by dynamic equivalence, we consider the equality between two PRNNs in the sense of adaptation of the weights, i.e. of the underlying learning algorithm (Mandic & Chambers, 1996b). The overall cost function of the PRNN is given by Baltersee and Chambers (1998), Haykin and Li (1995)

$$E(n) = \sum_{i=1}^{M} \lambda^{i-1} e_i^2(n) \tag{8}$$

where $e_i(n)$ is the error from module i, and a forgetting factor λ, $\lambda \in (0,1)$, is introduced which determines the weighting of the individual modules (Baltersee & Chambers, 1998; Haykin & Li, 1995). For a general class of nonlinear activation functions, the weight updating includes the first derivative of a nonlinear activation function of a neuron (Baltersee & Chambers, 1998; Mandic et al., 1998; Williams & Zipser, 1989), where

$$\Phi'(\beta, \mathbf{w}, \mathbf{u}) = \beta \Phi'(1, \mathbf{w}^{\mathrm{R}}, \mathbf{u}) \tag{9}$$

For the logistic function, for instance, we have

$$\Phi'(x) = \frac{\beta \mathrm{e}^{-\beta x}}{(1+\mathrm{e}^{-\beta x})^2} = \beta \Phi'(x^{\mathrm{R}}) \tag{10}$$

where $x^{\mathrm{R}} = \beta x$ denotes the argument of the referent logistic function (with $\beta^{\mathrm{R}} = 1$), so that the network considered is equivalent in the static sense to the referent network.

4.1. Dynamic equivalence for a GD learning algorithm

For a gradient descent learning algorithm (Baltersee & Chambers, 1998; Haykin & Li, 1995) for a PRNN, which is based upon the Real Time Recurrent Learning (RTRL) algorithm (Haykin, 1994; Williams & Zipser, 1989), the weight updating is given by

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \Delta \mathbf{W}(n) \tag{11}$$

$$\begin{aligned} \Delta w_{k,l}(n) &= -\eta \frac{\partial}{\partial w_{k,l}(n)} \left(\sum_{i=1}^{M} \lambda^{i-1} e_i^2(n) \right) \\ &= -2\eta \sum_{i=1}^{M} \lambda^{i-1} e_i(n) \frac{\partial e_i(n)}{\partial w_{k,l}(n)} \end{aligned} \tag{12}$$

and the error gradient becomes (Baltersee & Chambers, 1998; Haykin & Li, 1995)

$$\frac{\partial e_i(n)}{\partial w_{k,l}(n)} = -\frac{\partial y_{i,1}(n)}{\partial w_{k,l}(n)} \tag{13}$$

Under the assumption, also used in the RTRL algorithm (Narendra & Parthasarathy, 1990; Williams & Zipser, 1989), that when the learning rate η is sufficiently small, we have

$$\frac{\partial y_{i,\alpha}(n-1)}{\partial w_{k,l}(n)} \approx \frac{\partial y_{i,\alpha}(n-1)}{\partial w_{k,l}(n-1)} \tag{14}$$

and a quadruply indexed set of variables $\pi_{k,l}^{ij}(n) = \partial y_{i,j}(n)/\partial w_{k,l}(n)$ can be introduced to characterize the GD algorithm for the PRNN, as

$$\begin{aligned} &\pi_{k,l}^{ij}(n) = \frac{\partial y_{i,j}(n)}{\partial w_{k,l}(n)} \\ &1 \le i \le M, \;\; 1 \le j, \;\; k \le N, \;\; 1 \le l \le p+1+N \end{aligned} \tag{15}$$

Note that $\pi_{k,l}^{j1}(n) = (\partial y_{i,1}(n)/\partial w_{k,l}(n))$. The variant of the RTRL algorithm suitable for the PRNN is used to recursively compute the values of $\pi_{k,l}^{ij}$ for every time step n and all appropriate i, j, k, and l as follows:

$$\pi_{k,l}^{ij}(n+1) = \Phi'(v_{i,j}) \left[\sum_{m=1}^{N} w_{j,m}(n) \pi_{k,l}^{im}(n) + \delta_{kj} u_{i,l}(n) \right] \tag{16}$$

with the values for i, j, k, and l as in Eq. (15) and the initial conditions

$$\pi_{k,l}^{ij}(0) = 0 \tag{17}$$

where

$$\delta_{kj} = \begin{cases} 1, & k = j \\ 0, & k \neq j \end{cases}. \tag{18}$$

From the equivalence in the static sense, the weight update equation for the referent network, can be written as

$$\mathbf{W}^{\mathrm{R}}(n) = \mathbf{W}^{\mathrm{R}}(n-1) + \beta \Delta \mathbf{W}(n) \tag{19}$$

which gives

$$\begin{aligned} \Delta \mathbf{W}^{\mathrm{R}}(n) = \beta \Delta \mathbf{W}(n) &= \beta \left(2\eta \sum_{i=1}^{M} e_i(n) \frac{\partial y_{i,1}(n)}{\partial \mathbf{W}(n)} \right) \\ &= 2\eta\beta \sum_{i=1}^{M} e_i(n) \mathbf{\Pi}_{i,1}(n) \end{aligned} \tag{20}$$

where $\mathbf{\Pi}_{i,1}(n)$ is the matrix of $\pi_{k,l}^{i,1}(n)$.

The matrix $\mathbf{\Pi}(n)$, which comprises all the terms $\partial y_{i,j}/\partial w_{k,l}$, $\forall w_{k,l} \in \mathbf{W}$, $i = 1, \ldots, M$, $j = 1, \ldots, N$ must be related to the appropriate matrix $\mathbf{\Pi}^{\mathrm{R}}(n)$, which represents the gradients of the referent network. Thus, we have

$$\begin{aligned} (\pi_{k,l}^{i,j}(n+1))^{\mathrm{R}} &= \Phi'(v_{i,j}^{\mathrm{R}}(n)) \left[\sum_{m=1}^{N} w_{j,m}^{\mathrm{R}}(n) \pi_{k,l}^{i,m}(n) + \delta_{kj} u_{i,l}(n) \right] \\ &= \frac{1}{\beta} \Phi'(v_{i,j}(n)) \left[\sum_{m=1}^{N} \beta w_{j,m}(n) \frac{1}{\beta} \pi_{k,l}^{i,m}(n) + \delta_{kj} u_l(n) \right] \\ &= \frac{1}{\beta} \pi_{k,l}^{i,j}(n+1) \end{aligned} \tag{21}$$

which gives

$$\mathbf{\Pi}^{\mathrm{R}}(n+1) = \frac{1}{\beta}\mathbf{\Pi}(n+1) \tag{22}$$

Following the procedure of Mandic and Chambers (1999b), the weight adaptation process for the referent PRNN can be now expressed as

$$\Delta\mathbf{W}^{\mathrm{R}}(n) = \beta\Delta\mathbf{W}(n) = 2\beta\eta\sum_{i=1}^{M} e_i(n)\mathbf{\Pi}_{i,1}(n)$$

$$= 2\beta^2\eta\sum_{i=1}^{M} e_i(n)\mathbf{\Pi}^{\mathrm{R}}_{i,1}(n) = 2\eta^{\mathrm{R}}\sum_{i=1}^{M} e_i(n)\mathbf{\Pi}^{\mathrm{R}}_{i,1}(n) \tag{23}$$

which gives the required dynamic relationship, and is encompassed in the following Lemma.

Lemma 2. *An arbitrary PRNN represented by β, η, and $\mathbf{W}(n)$ is equivalent in the dynamic sense in terms of gradient descent-based learning to a referent isomorphic PRNN represented by $\beta^{\mathrm{R}} = 1$, η^{R} and $\mathbf{W}^{\mathrm{R}}(n) = \beta\mathbf{W}(n)$, if*

$$\eta^{\mathrm{R}} = \beta^2\eta \tag{24}$$

4.2. Dynamic equivalence for the ERLS learning algorithm

The extended recursive least squares algorithm, as introduced in Baltersee and Chambers (1998) and Haykin et al. (1997), is based upon representing the dynamics of the PRNN in the state space form

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{q}(n) \qquad \mathbf{x}(n) = \mathbf{h}(\mathbf{w}(n)) + \mathbf{v}(n) \tag{25}$$

where $\mathbf{w}(n)$ is the $N(N+p+1)\times 1$ vector obtained by rearranging the weight matrix $\mathbf{W}(n)$, $\mathbf{x}(n)$ is an $M\times 1$ observation vector, $\mathbf{q}(n) \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$ is a vector of white Gaussian noise (WGN), as well as $\mathbf{v}(n) \sim \mathcal{N}(\mathbf{0}, \mathbf{C})$. The first equation in Eq. (25) is the system equation, represented by a random walk, and satisfies the properties of the static equivalence, given in Lemma 1. The measurement equation in Eq. (25) is linearised using a first-order Taylor expansion, i.e. (Baltersee & Chambers, 1998; Iiguni & Sakai, 1992)

$$\mathbf{h}(\mathbf{w}(n)) \approx \mathbf{h}(\hat{\mathbf{w}}(n|n-1)) + \nabla\mathbf{h}^{\mathrm{T}}(\hat{w}(n|n-1))$$

$$\times[\mathbf{w}(n) - \hat{\mathbf{w}}(n|n-1)] \tag{26}$$

where the gradient of $\mathbf{h}(\cdot)$ can be expressed as

$$\nabla\mathbf{h}^{\mathrm{T}} = \frac{\partial\mathbf{h}(\hat{\mathbf{w}}(n|n-1))}{\partial\hat{\mathbf{w}}(n|n-1)} = \mathbf{H}(n) \tag{27}$$

Furthermore, the vector $\mathbf{h}(n)$, which is the result of the non-linear mapping $\mathbf{h}(n) = \mathbf{h}(\mathbf{w}(n))$ is actually the $M\times 1$ vector of the outputs of the PRNN modules (Baltersee & Chambers, 1998)

$$\mathbf{h}^{\mathrm{T}}(n) = [y_{1,1}(n), y_{2,1}(n), \ldots, y_{M,1}(n)] \tag{28}$$

That means that the equivalence needed for the observation equation boils down to the dynamic equivalence derived for the GD learning Eqs. (21)–(23).

Lemma 3. *An arbitrary PRNN represented by β, η, and* $\mathbf{W}$(n) *is equivalent in the dynamic sense in terms of the extended recursive least squares learning algorithm to a referent isomorphic PRNN represented by $\beta^{\mathrm{R}} = 1, \eta^{\mathrm{R}}$, and $\mathbf{W}^{\mathrm{R}}(n)$, if*

(i) They are equivalent in the static sense, $\mathbf{W}^{\mathrm{R}}(n) = \beta\mathbf{W}(n)$.
(ii) The learning rates are related as $\eta^{\mathrm{R}} = \beta^2\eta$.

Notice that the condition (i) is correspondent to the system equation of (25), whereas the condition (ii) corresponds to the measurement equation of (25).

4.3. Equivalence between an arbitrary and the referent PRNN

Some other learning algorithms for training the RNN and PRNN rest upon either general backpropagation, such as the Backpropagation Through Time (BPTT) algorithm (Werbos, 1990), or combine general backpropagation and the RTRL algorithm, such as the Recurrent Backpropagation (RB) algorithm (Pineda, 1987). Naturally, from Mandic and Chambers (1999b), Thimm et al. (1996) and the above analysis, the relationships derived are valid for both the BPTT and BP algorithms. From the static equivalence given in Lemma 1, and dynamic equivalence given in Lemmas 2 and 3, the following Theorem encompasses a general equivalence between an arbitrary PRNN, and the referent, isomorphic PRNN.

Theorem 1. *An arbitrary PRNN represented by β, η, and $\mathbf{W}(n)$ is equivalent to a referent isomorphic PRNN represented by $\beta^{\mathrm{R}} = 1, \eta^{\mathrm{R}}$ and $\mathbf{W}^{\mathrm{R}}(n)$, if and only if*

(i) They are equivalent in the static sense, i.e. $\mathbf{W}^{\mathrm{R}}(n) = \beta\mathbf{W}(n)$.
(ii) They are equivalent in the dynamic sense, i.e. $\eta^{\mathrm{R}} = \beta^2\eta$.

5. Conclusions

We have derived the relationship between the learning rate η and the slope in the general activation function β for a nonlinear optimisation algorithm which adapts the weights of nonlinear cascaded systems realised through a Pipelined Recurrent Neural Network (PRNN). The relationship is derived both in the static sense (equality of the outputs of the neurons) and the dynamic sense (equality in learning processes), for both the Gradient Descent (GD) and the Extended Recursive Least Squares (ERLS) algorithms.

Such a result enables one degree of freedom less when adjusting variable parameters of the PRNN, and hence reduces computational complexity of learning. The results provided are shown to be easily extended for the backpropagation through time, and the recurrent backpropagation algorithms, when applied in the PRNN framework. For nonlinear systems with only one module, the results boil down to the results for recurrent neural networks.

References

Baltersee, J., & Chambers, J. A. (1998). Non-linear adaptive prediction of speech signals using a pipelined recurrent neural network. *IEEE Transactions on Signal Processing, 46* (8), 2207–2216.

Haddad, W. M., Chellaboina, V. S., Fausz, J. L., & Abdallah, C. (1998). Optimal discrete-time control for non-linear cascade systems. *Journal of the Franklin Institute, 335B* (5), 827–839.

Haykin, S. (1994). *Neural networks—a comprehensive foundation*, Englewood Cliffs, NJ: Prentice Hall.

Haykin, S., & Li, L. (1995). Nonlinear adaptive prediction of nonstationary signals. *IEEE Transactions on Signal Processing, 43* (2), 526–535.

Haykin, S., Sayed, A. H., Zeidler, J. R., Yee, P., & Wei, P. C. (1997). Adaptive tracking of linear time-variant systems by extended RLS algorithms. *IEEE Transactions on Signal Processing, 45* (5), 1118–1128.

Iiguni, Y., & Sakai, H. (1992). A real-time learning algorithm for a multilayered neural network based on the extended Kalman filter. *IEEE Transactions on Signal Processing, 40* (4), 959–966.

Mandic, D. P., & Chambers, J. A. (1998). A posteriori real time recurrent learning schemes for a recurrent neural network based non-linear predictor. *IEE Proceedings—Vision, Image and Signal Processing, 145* (6), 365–370.

Mandic, D. P., & Chambers, J. A. (1999a). A nonlinear adaptive predictor realised via recurrent neural networks with annealing. *Digest of the IEE Colloquium Statistical Signal Processing*, 2/1–2/6.

Mandic, D. P., & Chambers, J. A. (1999b). Relationship between the slope of the activation function and the learning rate for the RNN. *Neural Computation, 11* (5), 1069–1077.

Mandic, D. P., Baltersee, J., & Chambers, J. A. (1998). Nonlinear prediction of speech with a pipelined recurrent neural network and advanced learning algorithms. In A. Prochazka, J. Uhlir, P. J. W. Rayner & N. G. Kingsbury, *Signal analysis and prediction*, (pp. 291–309). Boston, MA: Birkhauser.

Narendra, K. S., & Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks, 1* (1), 4–27.

Pineda, F. J. (1987). Generalization of backpropagation to recurrent neural networks. *Physical Review Letters, 59*, 2229–2232.

Suykens, J. A. K., DeMoor, L. R., & Vanderwalle, J. (1997). NLq theory: a neural control framework with global asymptotic stability criteria. *Neural Networks, 10* (4), 615–637.

Thimm, G., Moerland, P., & Fiesler, E. (1996). The interchangeability of learning rate and gain in backpropagation neural networks. *Neural Computation, 8*, 451–460.

Werbos, P. (1990). Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE, 78* (10), 1550–1560.

Williams, R., & Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Computation, 1*, 270–280.