An Adaptive Recurrent Network Training Algorithm Using IIR Filter Model and Lyapunov Theory

[1]SENG KAH PHOOI, [2] ZHIHONG MAN, [3]H.R.WU, [4]KAI MING TSE
[1]Monash University (Malaysia) 2 Jalan Kolej, Bandar Sunway, 46150 PJ, Selangor, MALAYSIA
[2]School of Computer Engineering, Nanyang Technological University, SINGAPORE
[3] School of CS & SE, Monash University, Clayton VIC 3168, AUSTRALIA
[1]School of Microelectronics, Griffith University, Kessels Rd, Nathan QLD 4111,AUSTRALIA

Abstract: - A new approach for the adaptive algorithm of a fully connected recurrent neural network (RNN) based upon the digital filter theory is proposed. Each recurrent neuron is modeled by using an infinite impulse response (IIR) filter. The weights of each layer in the RNN are updated adaptively so that the error between the desired output and the RNN output can converge to zero asymptotically. The proposed optimization method is based on the Lyapunov theory-based adaptive filtering (LAF) method [9]. The merit of this adaptive algorithm can avoid computation of the dynamic derivatives that is rather complicated in the RNN. The design is independent of the stochastic properties of the input disturbances and the stability is guaranteed by the Lyapunov stability theory. Simulation example of the nonstationary time series prediction problem is performed. The simulation results have validated the fast tracking property of the proposed method.

Key-Words: Recurrent Neural Network, IIR filter, Lyapunov stability theory

1 Introduction

Numbers of training methods for RNNs have been proposed in different literatures. Backpropagation through time (BPTT) [1] is an efficient learning algorithm but it cannot be run on-line and is impractical for applying on input signals of unknown lengths. The real-time recurrent learning (RTRL) [2] is a powerful on-line learning algorithm but it is extremely inefficient. Dynamic backpropagation [3] is a more useful one but complicated computation is required for the fully connected recurrent links. All of the above algorithms are, however, based on the steepest decent optimization and the speed of convergence is relatively low although different optimization techniques and adaptation of learning rate and momentum factor were developed, for example in [4]. Up to now, there is still a very little number of fast training algorithms applying at RNN architectures.

Recently, Kuan proposed a recurrent Newton algorithm and showed that the performance could be more effective than the conventional BP [5]. The approach showed that the search direction required the computation of Hessian matrix (second order derivative), which is rather computational complex and requires a rather a long computational time per iteration. The rate of convergence is not substantially faster than the other fast algorithms. In the past few years, recursive least squares (RLS) or extended Kalman filter (EKF) training algorithm are widely used for fast algorithm of RNNs. Williams has proposed the training algorithm for RNNs using the EKF [6]. Puskorius and Feldkamp have also proposed the decoupled extended Kalman filter (DEKF) algorithm for RNN that provides robust performance in the modeling of nonlinear dynamical system [7],[8]. This approach creates disjoint subsets of weights that are considered to be decoupled, that allow modeling of the interactions amongst the weights to be redundant. As a result, the computational complexity and the storage requirements of DEKF can be significantly less than those of the general EKF, It is, however, noted that these EKF type algorithm all require the computation of the dynamic derivatives (that is the derivatives with respect to the weights at real time training). Most cases, the computational complexity of the dynamic derivatives can be significantly affected on the algorithm efficiency.

In this paper, a fast adaptive training algorithm for RNN based on the digital IIR filter design is presented and each recurrent neuron is modeled by an IIR filter. The method of determining the filter coefficients is based upon the Lyapunov technique [9]. The synapses and the feedback weights are updated in the fashion of layer by layer which is similar to the process of determining the filter coefficients, The computation of the dynamic derivatives is not required. As a result, the computational complexity is significantly reduced. Comparing to other conventional algorithm the proposed adaptive algorithm exhibits very tracking behavior. The filter stability is guaranteed because the error dynamic asymptotically converges to zero. The results of simulation example are included. These results have shown that the proposed method performs well.

2 IIR Filter Neuron Model

In the fully connected RNN, each neuron in both hidden layer and output layer has a feedback loop from the neuron itself and the recurrent links from the other neurons at the layer which input are coming from the previous state of nonlinear output that they *y(t* -1) of the corresponding neurons. Assume are *m* number of inputs and *n* number of outputs at the corresponding layer.

$$z_j(t) = \sum_{i=1}^{m} b_{ij} + \sum_{k=1}^{n} a_{kj} y_k(t-1) \qquad (1)$$

$$y_j(t) = f\{z_j(t)\}, \quad 1 \le j \le n \qquad (2)$$

where *f{x}* is the nonlinear sigmoids function, $1/(1+e^{-x})$; *z_j(t)* is the summing output inside the neuron; $y_j(t)$ is the nonlinear output of the neuron and $(t-1)$ means the previous state; $x_i(t)$ be the input of the neuron; $a_{kj}(t)$ and $b_{ij}(t)$ are the weight of the recurrent synapses links, respectively. Each neuron can be separated into linear and nonlinear part as shown in both (1) and (2). We consider (1) a recursive difference equation that the linear output is a weighted sum of the past nonlinear function of output and the present input. Therefore, the output can produce an IIR if all inputs are unit impulse signal. Thus, from the digital filter theory point of view, all neurons can be considered as an IIR filter combining with the sigmoids function. The weights of each neuron can then updated by determining the IIR filter coefficients.

According to (1), we can reformulate in a simple vector form for applying in digital filter theory.

$$z_j^L(t) = u_j^{\,L}(t) h_j^L(t) \qquad (4)$$

where

$u_j^{\,L}(t) = [x_1(t), x_2(t), ..., x_m(t), y_1(t-1), y_2(t-1) ... y_n(t-1)]$

in *L*th layer.

$h_j^L(t) = [b_{1j}, b_{2j}, ..., b_{mj}, a_{1j}, a_{2j}, ..., a_{nj}]^T$ in *L*th layer

$h_j^L(t)$ in (4) can be computed by the adaptive algorithms.

In our approach, the problem of determining the adaptive algorithm be formulated as follow. We define the Lyapunov function

$$V(t) = \sum_{j=1}^{N} e^2(t) = \sum_{j=1}^{N} \left(s_j^L(t) - u_j(t) h_j^L(t)\right)^2 \qquad (5)$$

where $s_j^L(t)$ is an inversion form of monotonic function of *f{x}* by the desired outputs $d_j^L(t)$.

Therefore

$$s_j^L(t) = f^{-1}\{d_j^L(t)\} \qquad (6)$$

An adaptive algorithm is then designed so that $\Delta V(t) = V(t) - V(t-1) < 0$. The adaptive training algorithm for the multiplayer perceptron model based on the new method is detailed in the next section.

3 Adaptive Algorithm For Multilayer Perceptron Model

The objective of the adaptive training algorithm multiplayer perceptron model is to update the weights in each layer in order to get error convergence to zero asymptotically. As a result, based on (4), weights in each layer can be adaptively updated based on the Lyapunov theory in [10]. For simplicity, the superscript *L* can be omitted.

Let us define the estimated weight be $h_j^L(t)$, let

$$h_j(t) = h_j(t-1) + g_j(t)\alpha_j(t) \qquad (7)$$

where $g_i(t)$ is the adaptation gain and $\alpha_i(t)$ is a priori estimation error defined as

$$\alpha_j(t) = s_j(t) - h_j(t-1)u_j(t) \qquad (8)$$

The adaptation gain $g_i(t)$ in (8) is adaptively adjusted using Lyapunov stability theory [10] as

$$g_j(t) = \frac{u_j(t)}{\|u_j(t)\|^2}\left(1 - \kappa_j \frac{|e_j(t-1)|}{|\alpha_j(t)|}\right) \qquad (9)$$

where $0 \le \sum_{j=1}^{N} \kappa_j < 1$, so that the sum of squared error $\sum_{j=1}^{N} e_j^{\,2}(t)$ asymptotically converges to zero.

If the network has three layers (i.e., only one hidden layer), the adaptive learning procedure at every iteration can then be summarized as follows.

1 Initialize the weights between input and hidden layer randomly.
2 Propagate input signal through hidden layer to obtain output of the hidden layer to obtain output of the hidden layer (i.e., input of the output layer).
3 Evaluate the linear component of desired output by using (6).
4 Update weights between the hidden and the output layer using the set of recursive equation (7)-(9).
5 Apply the input of this output layer (i.e., the output of the hidden layer) to evaluate the linear components of the hidden layer by using (6) again.
6 Update the weights between the input and the hidden layer by using the set of recursive equations (7)-(9).

The design procedure of this adaptive RNN algorithm is similar to that of [10].

Define a Lyapunov function

$$V(t) = \sum_{j=1}^{N} e_j^{\,2}(t) \qquad (10)$$

and the increment

$$\begin{aligned}
\Delta V(t) &= V(t) - V(t-1) \\
&= \sum_{j=1}^{N} e_j^{\,2}(t) - \sum_{j=1}^{N} e_j^{\,2}(t-1) \\
&= \sum_{j=1}^{N} (d_j(t) - h_j(t)u_j(t))^2 - \sum_{j=1}^{N} e_j^{\,2}(t-1) \\
&= \sum_{j=1}^{N} (d_j(t) - (h_j(t-1) + g_j(t)\alpha_j(t))u_j(t))^2 - \sum_{j=1}^{N} e_j^{\,2}(t-1) \\
&= \sum_{j=1}^{N} (d_j(t) - h_j(t-1)u_j(t) - g_j(t)\alpha_j(t)u_j(t))^2 - \sum_{j=1}^{N} e_j^{\,2}(t-1) \\
&= \sum_{j=1}^{N} (\alpha_j(t) - g_j(t)\alpha_j(t)u_j(t))^2 - \sum_{j=1}^{N} e_j^{\,2}(t-1) \\
&= \sum_{j=1}^{N} \alpha_j(t)(1 - g_j(t)u_j(t))^2 - \sum_{j=1}^{N} e_j^{\,2}(t-1) \qquad (11)
\end{aligned}$$

Using the equation (9) in the equation (11), we have

$$\Delta V(t) = -(1 - \sum_{j=1}^{N} \kappa^2) \sum_{j=1}^{N} e_j^{\,2}(t-1) < 0 \qquad (12)$$

According to Lyapunov theory [10], if a positive definite function, eg. $V(t) = \sum_{j=1}^{N} e_j^{\,2}(t)$ is found such that its discrete time difference taken along a trajectory is always negative, $\Delta V(t) < 0$, then as time *t* increases, *V(t)* will finally converge to zero. Therefore the sum of error $\sum_{j=1}^{N} e_j^2(t)$ will also converge to zero asymptotically and the stability of the error dynamics is guaranteed.

It is noticeable that the values of $u_j(t)$ and $\alpha_j(t)$ in (9) may be zero and rise singularities problem. Therefore the adaptation gain may be modified as in the adaptation law (13)

$$g_j(t) = \frac{u_j(t)}{\| u_j(t) \|^2 + \lambda_1} \left(1 - \kappa_j \frac{| e_j(t-1) |}{| \alpha_j(t) | + \lambda_2} \right) \qquad (13)$$

where $0 \leq \sum_{j=1}^{N} \kappa_j < 1$, and λ_1, λ_2 are small positive numbers.

4 Simulation

Nonstationary Time Series Prediction – Simulations have been done for a one-step ahead prediction of a nonlinear and nonstationary speech signal which is identical to those in [11]. The signal is downloaded from the WWW [12] and is described as follow: *S1* speech sample "*When recording audio data …*", length 10000, sampled at 8kHz.The NN is expected to be able to track the nonstationary signal characteristic. Fig. 1 shows the speech signal and the RNN output between iterations (1 … 5000). Fig. 2 illustrates the square predictor error.

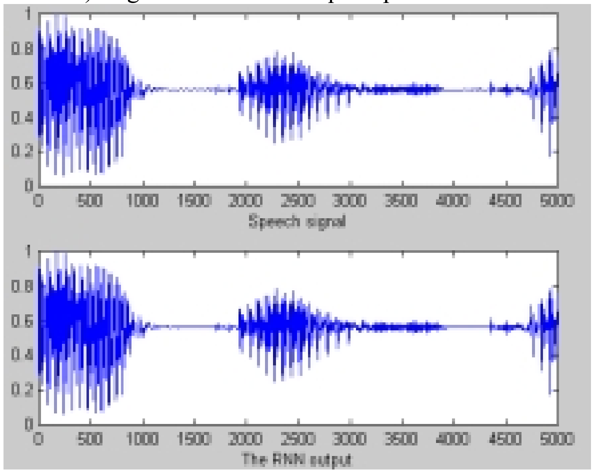

Fig. 1: Speech signal (top) & RNN output (bottom)

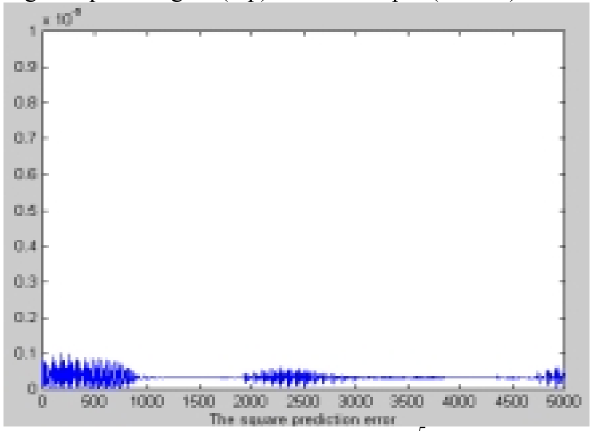

Fig. 2: Square output error *(y-axis – x10^{-5})*

5 Conclusion

In this paper, an adaptive learning algorithm of fully connected RNN is proposed. The proposed algorithm is based on the digital filter design theory that each recurrent neuron is considered as an IIR filter. The weights can be updated by using the new adaptive algorithm. The merit of this adaptive algorithm can avoid computation of the dynamic derivatives that is rather complicated in the RNN. As a result, the computational complexity of the proposed algorithm is significant less than other reported. The proposed algorithm has fast tracking rate and it is also capable of providing a low adaptive training error.

References:

[1] R. E. Rumelhard, G. E. Hinton and R. J. Williams, "Learning internal representation by error backpropagation", in Parallel Distributed Processing: Explorations Microstructure of Cognition, vol. 1. Cambridge, MA: MIT Press, 1986, ch. 8.

[2] R. J. Williams and D. Ziper, "A learning algorithm for continually running fully recurrent neural networks", Neural Computat., vol. 1. no. 2, pp 270-280, 1989.

[3] F. J. Pineda, "Generalization of backpropagation to recurrent neural network", Phys. Rev. Lett., vol. 59, pp. 2229-2232, 1987.

[4] K. P. Unnikrishnan and K. P. Venugopal, "Alopex: A correlation-based learning algorithm for feedforward and recurrent neural networks", Neural Computation, vol. 6, pp. 469-490, 1994.

[5] C. M. Kuan, "A recurrent Newton algorithm and its convergence properties", IEEE Trans. Neural Networks, vol. 6, pp. 7790-783, 1995.

[6] R. J. Williams, "Training recurrent networks training with the decoupled extended Kalman filter", in Int. Joint Conf. Neural network, vol. 1. no. 2, pp 270-280, 1989.

[7] G. V. Puskorius and L. A. Feldkamp, "Recurrent networks training with the decoupled extended Kalman filter algorithm", in Proc. 1992 SPIE Conf. Sci. Artificial Neural Networks, Baltimore, MD, vol. IV, 1992, pp. 461-473.

[8] ___, "Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks", IEEE Trans. Neural Networks, vol. 5, pp. 279-297, 1994.

[9] Man ZhiHong, H.R.Wu, W.Lai and Thong Nguyen, "Design of Adaptive Filters Using Lyapunov Stability Theory", The 6th IEEE International Workshop on Intelligent Signal Processing and Communication Systems, vo1, pp. 304-308, 1998

[10] Slotine, J-J. E. and Li, W. Applied nonlinear control, Prentice-Hall, Englewood Cliffs, NJ, 1991.

[11] Simon Haykin, "Nonlinear Adaptive Prediction of Nonstationary Signals", IEEE Trans. Signal Processing, Vol. 43, No. 2, February, 1995.

[12] http://www.ert.rwth-aachen.de/Presonen/balterse.html.