Chapter 8
An Investigation into the Adaptive Capacity of Recurrent Neural Networks

N.H. Siddique[1] and B.P. Amavasai[2]

[1] School of Computing and Intelligent Systems, University of Ulster,
Northland Road, Londonderry, BT48 7JL
nh.siddique@ulster.ac.uk
http://www.infm.ulst.ac.uk/~siddique
[2] Reading Innovation Centre, Procter and Gamble, 460 Basingstoke Road,
Reading RG2 0QE, UK
amavasai.b@pg.com

Abstract. The characteristic feature of any intelligent system is its ability to appropriately adjust its behaviour in response to a change of the environment and/or change in the structure of the system. Such a feature or behaviour of a system is termed as "adaptive behaviour", which is a measure of the learning ability of a system. Feed-forward neural networks have commonly been used to model such behaviours. However the weights of feed-forward neural networks remain static once strained and so can hardly be categorised as adaptive. On the contrary, recurrent networks have the capability to exhibit dynamic behaviour. In general, the feedback connections in a recurrent network are made after the non-linear activation function. In this chapter we investigated network architectures with different feedback connections made before and after the non-linear activation function to observe adaptive capability of these networks. Backpropagation training algorithms are applied to these networks with a minimum number of recurrent neurons at which adaptive behaviour is attainable. Three benchmark problems are chosen to investigate the performances on learning ability of the proposed architectures and results are presented and analysed.

Keywords: Recurrent network, Learning capacity, Adaptive behaviour, Back propagation algorithm.

1 Introduction

All forms of multilayer perceptron networks (MLP), radial basis functions networks (RBF), generalised regression networks (GRN) and probabilistic networks (PN) are classified as feedforward networks. They are typically used as static networks in such varied applications like identification, control, prediction, speech generation and pattern recognition [1]-[7]. Research interest in recurrent neural networks (RNN) has grown over the years because of their capability of exhibiting dynamic behaviour and

I. Jordanov and L.C. Jain (Eds.): Innovations in Intelligent Machines -3, SCI 442, pp. 119–138.
springerlink.com

computational power [8]. RNNs have empirically shown the ability to perform inference in problems as diverse as grammar induction, demonstrated potential for applications such as time series prediction, process modelling, and process control [9]-[14]. These networks can be trained, depending on their architecture and weights, as oscillators, as associative memories and also as in finite automata. Generally, training of this class of neural networks has been difficult due to the excessive time required to converge [15], [16]. Stability issues of training in recurrent neural networks are addressed in [17], [18]. Some researchers have suggested improvements that can be made to well known algorithms, which can improve the training of recurrent networks. Practical problems during the learning process of recurrent networks are mainly due to the presence of local minima in the cost function [19]. The adaptation of recurrent neural networks with fixed weights has been investigated by many researchers. It has been demonstrated that dynamic networks can learn without changing their synaptic weights [20]-[24]. Bengio et al. showed that gradient-based learning algorithms face an increasingly difficult problem as the duration of the dependencies to be captured increases [25]. De Jesus et al. demonstrated that spurious minima are introduced into the error surface due to characteristics in the input sequence that make the training more difficult for gradient descent algorithms in recurrent network [15]. Other early results on recurrent training algorithms have been reported in [26]-[28].

In this chapter we investigated the RNN architectures with different feedback connections, mainly, before and after the application of the non-linear activation function to observe the adaptive capability of these networks with varying weights for the feed forward connections and fixed weights for the recurrent connections. Therefore, traditional backpropagation (BP) training algorithms can be applied to networks with minimum number of recurrent neurons at which adaptive behaviour is attainable. Three different benchmark problems were chosen to verify the performances of the proposed architectures of the recurrent neural networks.

The rest of the chapter is organised as follows. Section 2 presents an overview of different RNN architectures; the proposed architectures and their block diagrams are described in section 3. The training algorithm for the proposed RNN is presented in section 4. The benchmark problems, experimentation and analysis of results are presented in section 5. Finally the chapter concludes with some remarks in section 6.

2 Overview of Recurrent Architectures

In the past few decades several recurrent neural network models have been proposed [9], [10], [14], [29]-[36]. These recurrent architectures can be classified into three broad categories where feedback connections are taken from three locations in the feedback loop: synapse feedback, feedback from the neuron output before non-linearity and output feedback after non-linearity proposed by different researchers [9], [10], [13], [14], [37]. All of these architectures reported in the literature are summarised according to their feedback connectivity in Table 1.

Table 1. Summary of architectures

Architecture	Synapse feedback	Output feedback	
		before nonlinearity $f(.)$	after nonlinearity $f(.)$
Hopfield			yes
Elman (uses context units)	-	-	yes
Jordan (uses context units)	-	-	yes
William-Zipser (allows any neuron)	-	-	yes
Frasconi-Gori-Soda	-	yes	yes
De Vries-Principe	yes	-	-
Poddar- Unnikrishnan	-	-	yes
Back-Tsoi	yes	-	-

The structure of the original Jordan network has three layers, with the main feedback connections taken from the output layer to the context layer. It has been theoretically shown that the original Jordan network [32] is not capable of representing arbitrary dynamic systems. However, by adding the feedback connections from the hidden layer to the context layer, (similar to the case of the Elman network [9]) a modified Jordan network is obtained. The modified Jordan network can be trained using the standard BP algorithm to model different dynamic systems. The values of the feedback connection weights have to be fixed by the user if the standard BP algorithm is employed.

The Elman architecture differs from the above in that it uses an extra layer of context neurons to copy hidden layer outputs and after delaying these values for one time unit, feeds them back as additional inputs to hidden layer neurons [9]. The idea of introducing self-feedback connections for the context units was borrowed from Jordan [32]. The standard BP algorithm is employed to Elman networks.

The William-Zipser architecture allows any neuron in the network to be connected to any other neuron in the network [14]. The William-Zipser architecture typically suffers from a lack of stability, i.e., for a given set of initial values, activations of linear output units may grow without limit and it has a slow convergence [13]. Frasconi et al. experimented with a slightly different architecture where they introduced local activation feedback and output feedback taken from hidden layer neurons after it has passed through the non-linearity and weighted by a constant value [10]. Their architecture is represented by the following equation:

$$y(t) = f\left(k_1 y(t-1) + \sum_{i=1}^{n} w_i x_i + b \right) \tag{1}$$

Where x_i, $i = 1,2,...,n$ are the inputs, w_i is the weight, b is the bias, $y(t-1)$ is the output delayed by one time unit and k_1 is the weighting factor of the output.

The Back-Tsoi architecture is different from Frasconi et al. architecture in that the feedback is taken at the synapse output [13]. They introduced a synapse with a linear transfer function instead of a synapse with a constant weight. Poddar-Unikrishnan [33] introduced a memory neuron which remembers the past output values. This architecture has a feedback transfer function with one pole only. A critical review of the various feedback architectures that appeared in the literature can be found in [13].

The Hopfield network consists of a set of neurons and a corresponding set of unit delays, where the output of each neuron is fed back to each of the other neurons except itself via the delay units [38]. The number of feedback connections is equal to the number of neurons. It consists of n neurons. The discrete Hopfield network is described in discrete time as

$$y_i(t) = \Gamma\left(\sum_{j=1}^{n} w_{ij} y_j(t-1) + x_i - b_i \right) \quad for \quad j \neq i, i = 1,2,\cdots,n \tag{2}$$

Where $\Gamma()$ is the activation function defined as $\mathrm{sgn}(.)$, x_i is the external input to ith neuron and b_i is the bias. The feedback input to the ith neuron is equal to the weighted sum of neuron outputs y_j, where $j = 1,2,\cdots,n$. The matrix of synaptic weights W in the Hopfield model is an $n \times n$ symmetric matrix i.e. $w_{ij} = w_{ji}$ and the diagonal elements are zero i.e. $w_{ij} = 0$ for $\forall i = j$ and defined as

$$W = \begin{bmatrix} 0 & w_{12} & \cdots & w_{1n} \\ w_{21} & 0 & \cdots & w_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ w_{n1} & w_{n2} & \cdots & 0 \end{bmatrix} \tag{3}$$

Hence no connection exists from any neuron back to itself. Though the difference equation in (2) is highly coupled and nonlinear, the solution of the Hopfield model is stable when the weight matrix in (3) is symmetric and the biases are zero. This yields a non-increasing energy function for Hopfield network defined as

$$H(y) = -\frac{1}{2} \sum_i \sum_j w_{ij} y_i y_j \tag{4}$$

3 Proposed Network Architecture

The architecture considered in this study is a combination of Elman and Frasconi-Gori-Soda architecture with three layers of neurons in that it does not use any context layer and has a single unit time delay. Two types of feedback connections are investigated in this study: (i) output feedback before the activation function (non-linearity) and (ii) output feedback after the activation function (non-linearity). The two architectures can be described mathematically in the following equations. The

recurrent architecture with feedback connections before nonlinearity is described by the equations (5)-(7).

$$y_k(t) = f\left(\left(\sum_{j=1}^{m} w_{kj}\, y_j^h(t)\right) + \theta_k\right), \quad k = 1,2,\cdots,l \tag{5}$$

$$y_j^h(t) = f\left(net_j(t)\right) \tag{6}$$

$$net_j(t) = \left(\sum_{i=1}^{n} w_{ji} x_i + \theta_j\right) + \left(\sum_{j=1}^{m} w_{jj} K_s net_j(t-1)\right) \tag{7}$$

The architecture with feedback connections after nonlinearity can be described by the equations (5)-(6) and (8).

$$net_j(t) = \left(\sum_{i=1}^{n} w_{ji} x_i + \theta_j\right) + \left(\sum_{j=1}^{m} w_{jj} K_s y_j^h(t-1)\right) \tag{8}$$

where $i = 1,2,3,\cdots,n$, $j = 1,2,3,\cdots,m$ and $k = 1,2,3,\cdots,l$. n, m and l represent the maximum number of neurons in the input, hidden and output layer respectively. $y_k = [y_1, y_2, y_3, \cdots, y_l]^T$ are the outputs of the output-layer, $y_j^h(t) = \left[y_1^h(t), y_2^h(t), y_3^h(t), \cdots, y_m^h(t)\right]$ are the outputs of the hidden neurons after the nonlinear activation function, $net_j(t) = [net_1(t), net_2(t), net_3(t), \cdots, net_m(t)]^T$ are the outputs of the hidden neurons, $x_i = [x_1, x_2, x_3, \cdots, x_n]^T$ are the inputs, w_{kj} is the weight matrix of $l \times m$ dimension representing connectivity from hidden layer neurons to output layer neurons, w_{ji} is the weight matrix of $m \times n$ dimension representing connectivity from input layer to hidden layer neurons, $\theta_k = [\theta_{k1}, \theta_{k2}, \theta_{k3}, \cdots, \theta_{kl}]^T$ and $\theta_j = \left[\theta_{j1}, \theta_{j2}, \theta_{j3}, \cdots, \theta_{jm}\right]^T$ are biases of the output layer and the hidden layer neurons respectively. w_{jj} is the weight matrix of $m \times m$ dimension represents the recurrent connectivity between neurons in the hidden layer. The scaling factor K_s scales the feedback variables to neurons. The matrix w_{jj} can take the form of a full, symmetric, upper diagonal, lower diagonal, diagonal or a null matrix, which represents different connectivity between hidden-layer neurons. The two architectures proposed in this section, described by the equations (5)-(8), can be written in the following form, which will help understanding the block structure of the two networks.

$$y(t) = F_k\left(net_k(t)\right)I_k \tag{9}$$

$$net_k(t) = A_k\, y^h(t) + \Theta_k \tag{10}$$

$$y^h(t) = F\left(net_j(t)\right)I_j \tag{11}$$

$$net_j(t) = \left(A_j x + \Theta_j\right) + B\left(K_s net_j(t-1)\right) \tag{12}$$

$$net_j(t) = \left(A_j x + \Theta_j\right) + B\left(K_s y^h(t-1)\right) \tag{13}$$

The equations (9)-(11) and (12) represent the recurrent architecture with feedback connections before non-linearity and the equations (9)-(11) and (13) represent the recurrent architecture with feedback connections after non-linearity. $A_j = w_{ji}$, $A_k = w_{kj}$ and $B = w_{jj}$ are connection matrices, $\Theta_j = [\theta_1, \theta_2, \theta_3, \cdots, \theta_m]^T$ and $\Theta_k = [\theta_1, \theta_2, \theta_3, \cdots, \theta_l]^T$ are vectors of biases. $I_j = [1 \quad 1 \quad \cdots \quad 1]^T$ and $I_k = [1 \quad 1 \quad \cdots \quad 1]^T$ are vectors of $m \times 1$ and $l \times 1$ dimensions respectively. F_j and F_k are diagonal matrices of non-linear activation functions of hidden layer and output layer defined as

$$F_j = \begin{bmatrix} f_1(.) & 0 & \cdots & 0 \\ 0 & f_2(.) & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & f_m(.) \end{bmatrix} \tag{14}$$

$$F_k = \begin{bmatrix} f_1(.) & 0 & \cdots & 0 \\ 0 & f_2(.) & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & f_l(.) \end{bmatrix} \tag{15}$$

Thus, the equations (9)-(13) along with the definitions of the non-linear activation matrices defined by equations (14)-(15) can be described by the block notation form of the networks shown in Figures (1) and (2). While z^{-1} represents one unit of time delay element in the feedback path resulting in a recurrent network. The block notation, first used by Santini et al. [39] derived from Narendra's [2], are developed for the two network architectures to provide a clearer representation of the feedforward and feedback connectivity.

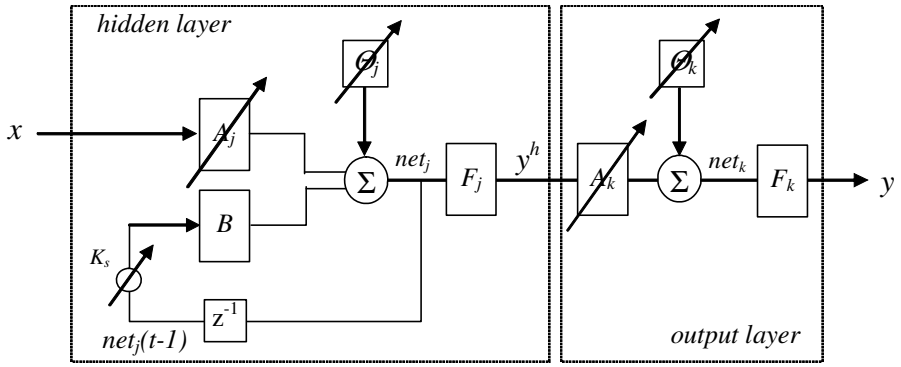


Fig. 1. Block diagram of the recurrent network with feedback before activation function

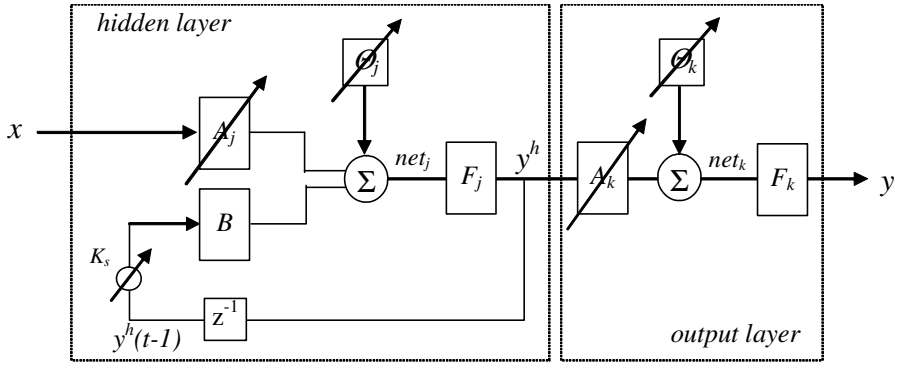


Fig. 2. Block diagram of the recurrent network with feedback after activation function

$B = w_{jj}$ in Figure 1 and Figure 2 represents the feedback connection matrix describing the feedback connectivity between hidden neurons and can take various forms depending on the connectivity described by the equations (16)-(26). The feedback connection matrix can be a full-matrix shown in equation (16), where the elements $\{b_{11}, b_{12}, \cdots, b_{mm}\} \in \Re$ are real values. The full-matrix implies that all neurons in the hidden layer are connected to each other in all possible ways, which yields a connection matrix (B_F) of the form

$$B_F = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mm} \end{bmatrix} \tag{16}$$

When the connectivity matrix is a full matrix (B_F), it can be seen as a special case of the Williams-Zipser architecture and its performance is well-known for its slow convergence [14].

The connectivity matrix between hidden neurons can be symmetric, which means that the outputs are fed back to other neurons except themselves. Such feedback connectivity can be described by the matrix (B_S) given by equation (22) with diagonal elements zero.

$$B_S = \begin{bmatrix} 0 & b_{12} & \cdots & b_{1m} \\ b_{21} & 0 & \cdots & b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & 0 \end{bmatrix} \tag{17}$$

The feedback connection matrix can be an upper diagonal matrix (B_U) shown in equation (18) where neurons are connected to the next neuron in the layer. For example, neuron 1 is connected to neurons 2 to *n*, neuron 2 is connected to 3 to *n*, and neuron *n* does not have any connections.

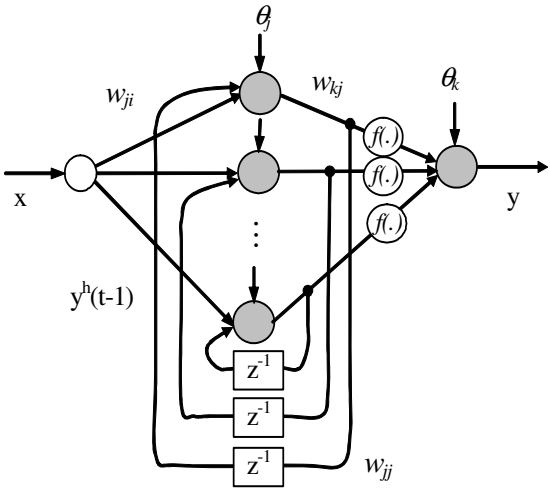
$$B_U = \begin{bmatrix} 0 & b_{12} & \cdots & b_{1m} \\ 0 & 0 & \cdots & b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \tag{18}$$

The connectivity matrix can be a lower diagonal matrix (B_L) as shown in equation (19). The connections are just reversed to connections in matrix (B_U).

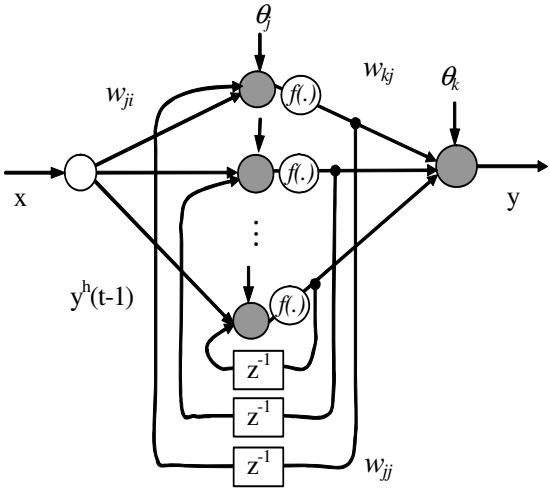
$$B_L = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ b_{21} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & 0 \end{bmatrix} \tag{19}$$

The outputs of the hidden neurons can be fed back only to the neuron itself. In this case the weight matrix will be a diagonal matrix (B_D) described by equation (20). The connectivity is shown in Figure 3. The connectivity before non-linearity is shown in Figure 3(a) and after non-linearity is shown in the Figure 3(b).

$$B_D = \begin{bmatrix} b_{11} & 0 & \cdots & 0 \\ 0 & b_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & b_{mm} \end{bmatrix} \tag{20}$$



(a) Feedback connection before nonlinearity.



(b) Feedback connection after nonlinearity.

Fig. 3. Feedback connections to neurons themselves

If there are no feedback connections, it implies a feedforward network and yields a null matrix (B_N) in equation (21). In this case, the network is simply a feedforward network.

$$B_N = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \tag{21}$$

The symmetric matrix B_S , upper diagonal B_U and lower diagonal matrix B_L cases are abandoned for computational simplicity. The diagonal matrix B_D is chosen for detailed investigation in this chapter.

4 Training Algorithm of the RNN

Generally the training of RNNs have been difficult due to the time dependencies present in the architecture and difficulties in gradient descent algorithms. The most common error (criterion) function E for the training of dynamic neural networks is given by

$$E = \frac{1}{2}\sum_{t=0}^{T}\left(y_d(t) - y(t)\right)^2 \tag{22}$$

$$\Delta A_j = -\eta \frac{\partial E}{\partial A_j} \tag{23}$$

$$\Delta A_k = -\eta \frac{\partial E}{\partial A_k} \tag{24}$$

Where T is the maximum training time at which error function E approaches very close to zero i.e. for $T \to \infty$, $E \to 0$. η is a fixed positive learning rate. A varying feedback connection weights may produce better results but for simplicity feedback connection weights (B) are considered fixed in the both cases. Therefore, ∂B is constant and hence the learning of the weights does not occur. Since the weights of the feedback connections are considered fixed, the weight update rule for B is

$$\Delta B = -\eta \frac{\partial E}{\partial B} = 0 \tag{25}$$

From the derivations above, the recurrent network can be considered as a feedforward network and by using the weight-update rules in (23)-(25) the network can be trained with standard BP algorithm.

To apply the BP procedure to RNNs, the ordered list of dependencies for the recurrent topologies needs to be adjusted. The present value of the activation of the state denoted as $y(t)$ depends on the previous value $y(t-1)$. The sensitivity of the present state also depends on the previous sensitivities.

5 Experimentation and Analysis of Results

The recurrent network architectures and training procedures described in Section 3 were trained using the BP algorithm to demonstrate the capabilities of the networks. Three well-known benchmark problems were chosen for this task:

(i) Two-input XOR,
(ii) Three-input XOR, and
(iii) Shape eight

The XOR problem has historically been considered a good test of a network model and its associated learning algorithm. There are many reasons for this choice. Firstly, the XOR problem is one of the simplest problems, which is not linearly separable and complex enough for BP algorithm to be trapped in local minima without reaching the global optimum. Secondly, there has been a significant number of research work which claimed that the XOR problem exhibits local minima, a view that is widely accepted in neural network literature [40]-[42]. The configuration of the network and the truth table for the two-input XOR are shown in Figure 4 and Table 2 respectively. Only the symbolic representations of the networks are produced here.

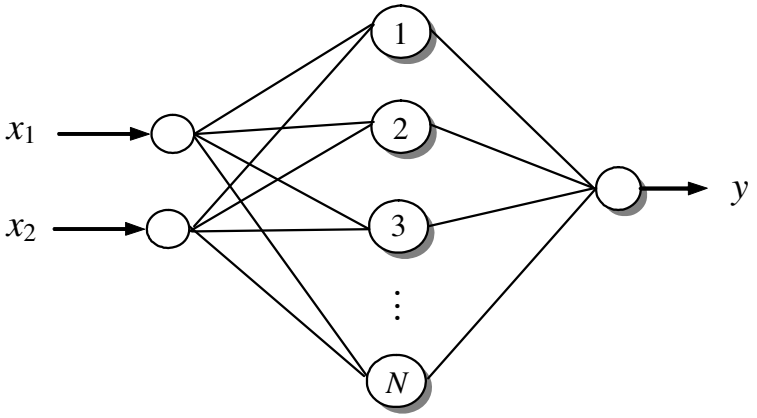


Fig. 4. Neural network for two-input XOR problem

Table 2. Truth-table for two-input XOR

inputs		output
x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0

The parity problem can be considered as a generalised XOR. The generalised XOR problem is good test for computer simulation, because we can systematically vary the mismatch between the degrees of freedom for the problem (number of weights in an unconstrained network). The configurations of the network and truth tables for the three-input generalised XOR is shown in Figure 5 and Table 3.

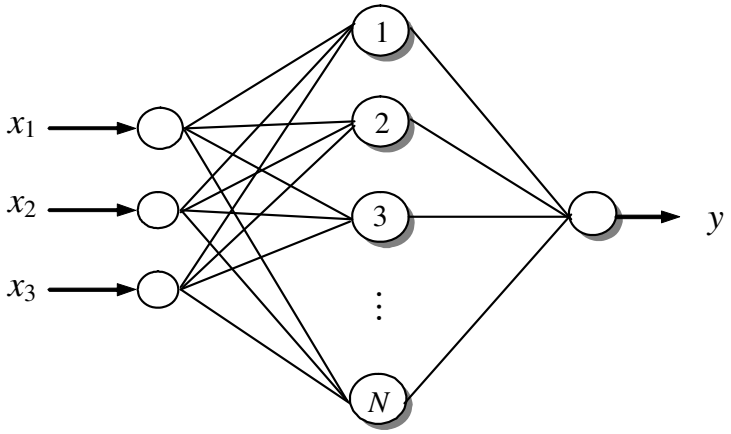


Fig. 5. Neural network for three-input generalised XOR problem

Table 3. Truth-table for three-input generalised XOR

inputs			output
x_1	x_2	x_3	y
0	0	0	0
1	0	0	1
0	1	0	1
1	1	0	0
0	0	1	1
1	0	1	0
0	1	1	0
1	1	1	1

A continuous target trajectory like shape eight is also used as benchmark problem by many researchers [43], [44]. The analytical equations of the corresponding target trajectory of figure eight are given by equations (26)-(27).

$$r1(t) = 0.5 * (1 + 0.9 * \sin(2 * t)) \tag{26}$$

$$r2(t) = 0.5 * (1 + 0.9 * \sin(t)) \tag{27}$$

where $t \in \Re$ and chosen between $[1,8]$. The network configuration and the target trajectory are shown in Figure 6 and Figure 7 respectively.

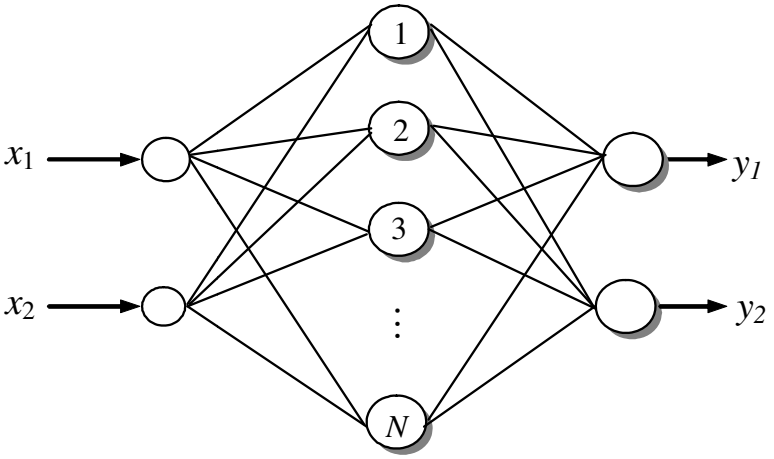


Fig. 6. Neural network for figure eight problem

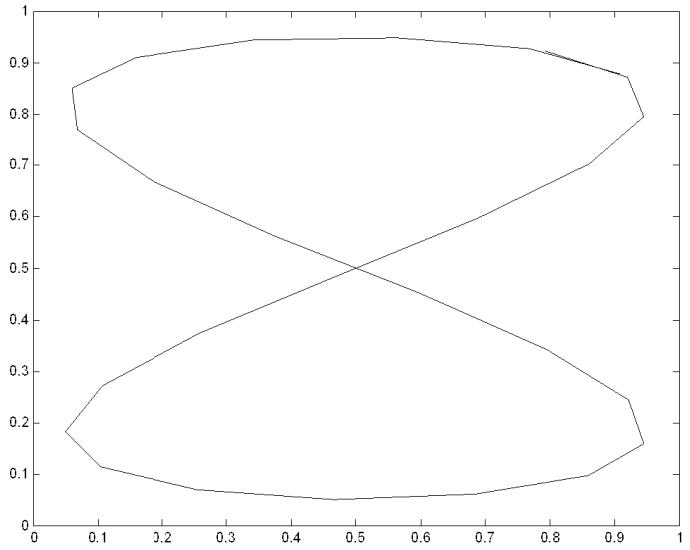


Fig. 7. The shape eight problem

In this study, sigmoidal activation functions were used for the hidden layer and linear activation functions were used for the output layer. Experiments on different architectures with 1 to 18 hidden neurons were investigated on the three benchmark problems. For each problem an error goal of 0.009 and a maximum iteration of 25000 epochs were set. Training parameters such as learning rate, momentum, and acceleration were different for each problem. 11 patterns were chosen for two-input XOR (2XOR) and three-input XOR (3XOR) problem. 27 patterns were used for the figure eight problem, which is found to be the minimum for representing a reasonably

smooth curve. For even smoother curve representations, more training time will be required, and so it not considered in this study. The initial weights of the network are sensitive to the convergence of the learning profile and hence found by trial and error for each case of the three benchmark problems.

For the two-input XOR with feedback after non-linearity, a 2-12R-1 RNN achieved an error goal of 0.1955 at 25000 epochs. For the case of feedback before non-linearity, a 2-4R-1 RNN achieved an error goal of 0.0517 at 25000 epochs. For the latter case, full feedback caused instability of the network output and hence a scaling factor was introduced. The RNN was found to be stable for a scaling factor $0 < K_s < 1$ and produced the best result at a value of $K_s = 0.2$. In both cases, learning rate, momentum, and acceleration were kept constant, as shown in Tables 4 and 5. The weight matrices A_j and A_k were initialised within the range (-0.5, +0.5) and yielded better result than if initialised between [0, 1]. The learning profiles for two-input XOR problem are shown in Figure 8. It has been reported by many researchers that the existence of local minima depends on both the problem and the architecture of the network being investigated. Rumelhart et al. found local minima to be very rare [45]. In a study by Hamey, it is shown that a feedforward neural network for the XOR problem has no local minima, which provides a valuable refinement of theories of the occurrence of local minima [46]. It is clearly evident from Figure 8 that the performance of 2-4R-1 RNN with feedback before non-linearity is better than that of 2-12R-1 RNN with feedback after non-linearity. Although feedback before

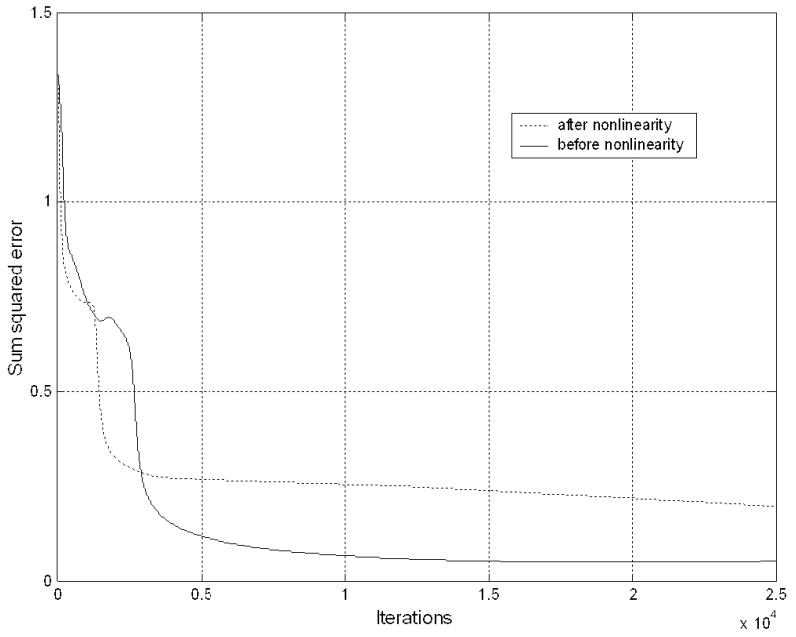


Fig. 8. Learning profile for two-input XOR problem

nonlinearity shows a small local minimum basin at a very early stage of learning, it exhibits a smooth learning curve with a much smaller sum squared error. In the experimentations, we also investigated cases with different odd and even number combinations of hidden neurons. It was found that odd numbers of hidden neurons produced poorer results in the case of two-input XOR.

For the three-input generalised XOR with feedback after non-linearity, a 3-6R-1 RNN achieved an error goal of 0.0089 at 8792 epochs. The weight matrices A_j and A_k were initialised within the range of [-0.5, 0.5]. For the case of feedback before non-linearity, a 3-4R-1 RNN achieved an error goal of 0.00899 at 8080 epochs with a scaling factor of $K_s = 0.2$ In this case, weight matrices A_j and A_k initialised within range of [0, 1] produced better result than within the range of [-0.5, 0.5]. Training parameters were exactly the same in the both cases. The learning profiles for three-input XOR problem are shown in Figure 9. The learning profile for feedback after non-linearity shows two local minima. Training could be easily trapped in those local minima for wrong choice of parameters whereas the learning profile for feedback before non-linearity shows smoother training and much better performance with a smaller architecture with 4 hidden neurons. Investigations further showed that an even number of hidden neurons produce better result than that of odd numbers.

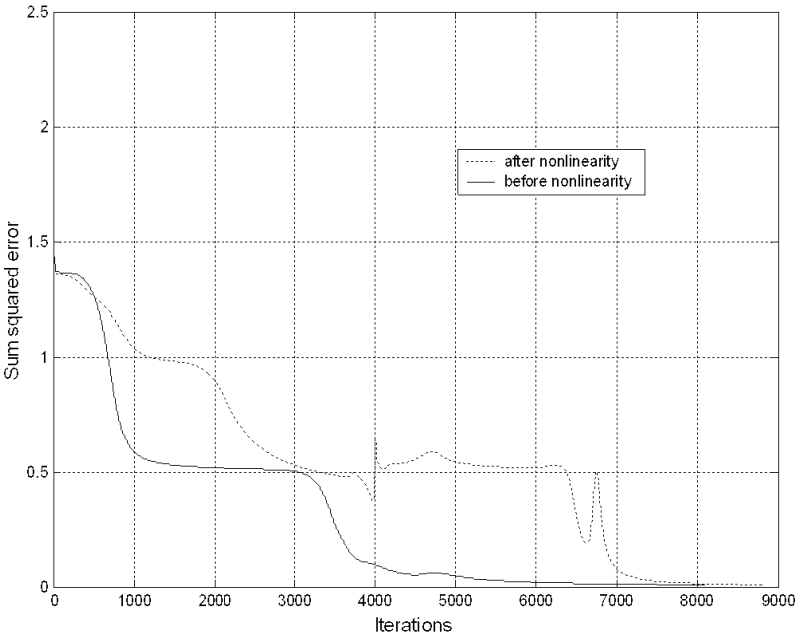


Fig. 9. Learning profile for three-input XOR problem

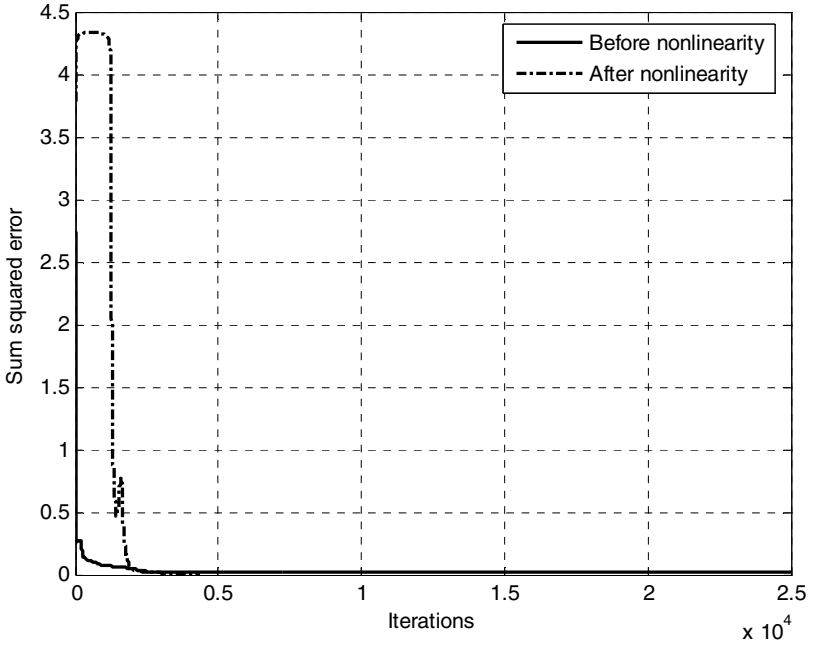


Fig. 10. Learning profile for figure 8 problem

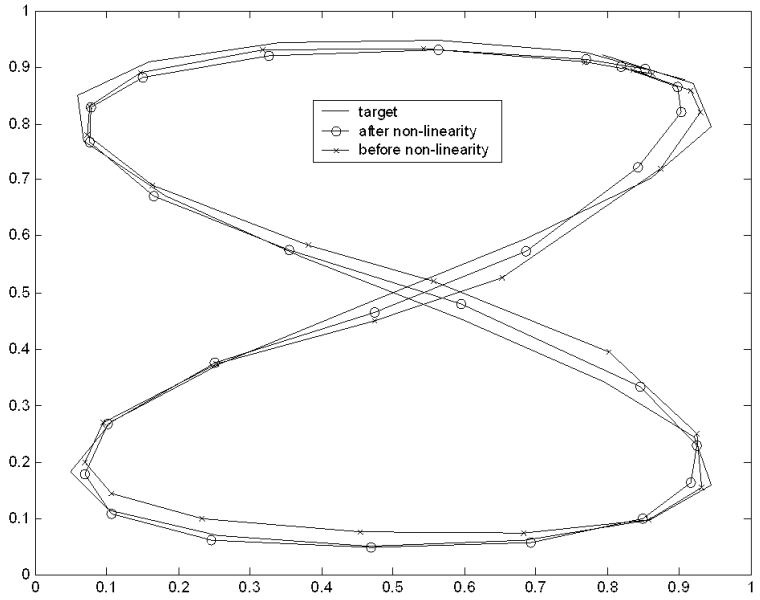


Fig. 11. Trajectory tracking for figure 8

For the figure-eight problem with feedback after non-linearity, a 2-7R-2 RNN achieved an error goal of 0.00899 at 3339 epochs. For the case of feedback before non-linearity, a 2-4R-2 RNN achieved an error goal of 0.01562 at 25000 epochs with a scaling factor of $K_s = 0.2$. Training parameters were kept same in the both cases. The weight matrices A_j and A_k were initialised within the range of [-1, 1]. The learning profiles for the shape-eight problem are shown in Figure 10. The learning profile for feedback after non-linearity shows local minima. Training could be easily trapped for incorrect choice of parameters where as the learning profile for feedback before non-linearity shows smoother training and much better performance with a smaller number of hidden neurons, i.e. 4 hidden neurons. Investigations show that odd number of hidden neurons produced better results in the case of feedback after non-linearity and even number produced better result in the case of feedback before non-linearity. It is found that such architectures of RNN can follow a continuous trajectory as shown in Figure 11.

A summary of network training parameters used for the three benchmark problems are shown in Table 4 and Table 5.

Table 4. Feedback after non-linearity

	Learning rate	Momentum	Acceleration	Hidden neuron	K_s	Epochs	Error
2XOR	.04	.8	-0.1	12	1	25000	.1955
3XOR	.04	.8	-0.1	6	1	8792	.0089
Fig8	1.2	.7	-0.7	7	1	3339	.0089

Table 5. Feedback before non-linearity

	Learning rate	Momentum	Acceleration	Hidden neuron	K_s	Epochs	Error
2XOR	.04	.8	-0.1	4	.2	25000	.0517
3XOR	.04	.8	-0.1	4	.2	8080	.0089
Fig8	1.2	.7	-0.7	4	.2	25000	.0156

As can be seen from Table 5 that feedback connection before non-linearity is more stable than that of feedback connection after non-linearity, require less hidden layer neurons, and achieve better error goal within fewer epochs.

6 Conclusion

The adaptive capacity of the RNN was investigated on two different neural architectures. The three benchmark problems have been used to investigate the performance of the proposed RNN architectures. Improved performances have been

demonstrated by RNN architectures with feedback connections before the application of the non-linear function when compared with RNN architectures with feedback connection applied after the non-linear function. The experiments have shown that the RNN can be trained with a reduced risk of the stability, which is mainly caused by recurrent connection after non-linearity. The results from this investigation will thus be useful in designing future applications that incorporates RNNs.

References

[1] Parlos, A.G., Parthasarathy, S., Atiya, A.F.: Neuro-Predictive Process Control using On-line Controller Adaptation. IEEE Transaction on Control Systems Technology 9(5), 741–755 (2001)
[2] Narendra, K.S., Parthsarathy, K.: Identification and control of dynamical systems using Neural Network. IEEE Transaction on Neural Networks 1(1), 4–27 (1993)
[3] Narendra, K.S., Mukhopadhyay, S.: Adaptive Control using Neural Networks and Approximate Models. IEEE Transactions on Neural Networks 8, 475–485 (1997)
[4] Powell, M.J.D.: Radial Basis Functions for Multivariable Interpolation: A review. IMA Conference Algorithm for Approximation of Functions and Data, RMCS Shrivenham (1985)
[5] Specht, D.F.: Probabilistic Neural Networks. Neural Networks, International Neural Network Society 3, 109–118 (1990)
[6] Specht, D.F.: A Generalized Regression Neural Network. IEEE Transactions on Neural Networks 2(6), 568–576 (1991)
[7] Millan, J.R., Mourino, J., Franze, M., Cincotti, F., Varsta, M., Heikkonen, J., Babiloni, F.: A Local Neural Classifier for the Recognition of EEG Patterns Associated to Mental Tasks. IEEE Transaction on Neural Networks 13(3), 678–686 (2002)
[8] Feldkamp, L.A., Prokhorov, D.V., Feldkamp, T.M.: Simple and Conditioned Adaptive Behaviour from Kalman Filter Trained Recurrent Networks. Neural Networks 16, 683–689 (2003)
[9] Elman, J.L.: Finding Structure in Time. Cognitive Science 14, 179–211 (1990)
[10] Frasconi, P., Gori, M., Soda, G.: Local feedback multilayered networks. Neural Computing 4, 120–130 (1992)
[11] Patan, K.: Stability Analysis and the Stabilization of a Class of Discrete-Time Dynamic Neural Networks. IEEE Transaction on Neural Networks 18(3), 660–673 (2007)
[12] Spiegel, R., Suret, M., Le Pelley, M.E., McLaren, I.P.L.: Analysing State Dynamics in a Recurrent Neural Network. In: IEEE International Conference on Neural Networks, pp. 834–839 (2002)
[13] Tsoi, A.C., Back, A.D.: Locally Recurrent Globally Feedforward Networks: A Critical Review of Architectures. IEEE Transaction on Neural Networks 5(2), 229–239 (1994)
[14] Williams, R.J., Zipser, D.: A Learning Algorithm for Continually Running Fully Recurrent Networks. Neural Computing 1, 270–280 (1989)
[15] De Jesus, O., Horn, J.M., Hagan, M.T.: Analysis of Recurrent Network Training and Suggestions for Improvements. In: IEEE International Joint Conference on Neural Networks, pp. 2632–2637 (2001)
[16] Towntey, S., Ilchmann, A., Weib, M.G., Mcclements, W., Ruiz, A.C., Owens, D.H., Pratzel-Wolters, D.: Existence and Learning of Oscillation in Recurrent Neural Networks. IEEE Transactions on Neural Networks 11(1), 205–214 (2000)
[17] Suykens, J.A.K., De Moor, B., Vandewalle, J.: Robust Local Stability of Multilayer Recurrent Neural Networks. IEEE Transactions on Neural Networks 11(1), 222–229 (2000)

[18] Xia, Y., Wang, J.: Global Exponential Stability of Recurrent Networks for Solving Optimisation and Related Problems. IEEE Transactions on Neural Networks 11(4), 1017–1022 (2000)
[19] Bianchini, M., Gori, M., Maggini, M.: On the Problem of Local Minima in Recurrent Neural Networks. IEEE Transactions on Neural Networks 5(2), 167–177 (1994)
[20] Feldkamp, L., Puskorius, G., Moore, P.: Adaptation from Fixed Weight Dynamic Networks. In: Proceedings of IEEE International Conference on Neural Networks, pp. 155–160 (1996)
[21] Younger, A.S., Conwell, P.R., Cotter, N.E.: Fixed-Weight on-line Learning. IEEE Transaction on Neural Networks 10(2), 272–283 (1999)
[22] Prokhorov, D.V., Feldkamp, L.A., Tyukin, I.Y.: Adaptive Bahaviour with Fixed Weights in RNN: An Overview. In: IEEE International Joint Conference on Neural Networks, pp. 2018–2022 (2002)
[23] Younger, A.S., Hochreiter, S., Conwell, P.R.: Meta Learning with Backpropagation. In: IEEE International Conference on Neural Networks, pp. 2001–2006 (2001)
[24] Lo, J.T., Bassu, D.: Adaptive vs. Accommodative Neural Networks for Adaptive System Identification. In: IEEE International Joint Conference on Neural Networks, pp. 1279–1284 (2001)
[25] Bengio, Y., Simard, P., Frasconi, P.: Learning Long-Term Dependencies with Gradient Descent is Difficult. IEEE Transactions on Neural Networks 5(2), 157–166 (1994)
[26] Atiya, A.F., Parlos, A.G.: New Result on Recurrent Network Training: Unifying the Algorithms and Accelerating Convergence. IEEE Transactions on Neural Networks 11(3), 697–709 (2000)
[27] Campolucci, P., Uncini, A., Piazza, F., Rao, B.D.: On-Line Learning Algorithms for Locally Recurrent Neural Networks. IEEE Transactions on Neural Networks 10(2), 253–271 (1999)
[28] Leistritz, L., Galicki, M., Witte, H., Kochs, E.: InitialState Training Procedure Improves Dynamic Recurrent Networks with Time-Dependent Weights. IEEE Transaction on Neural Networks 12(6), 1513–1518 (2001)
[29] Back, A., Tsoi, A.: FIR and IIR synapses, a new neural network architecture for time series modelling. Neural Computation 3(3), 375–385 (1991)
[30] Billings, S.A., Jamaluddin, H.B., Chen, S.: Properties of neural networks with applications to modelling non-linear dynamical systems. International Journal of Control 55(1), 193–224 (1992)
[31] Arai, K., Nakano, R.: Stable Behaviour in a Recurrent Neural Network for a FiniteState Machine. Neural Networks 13, 667–680 (2000)
[32] Jordan, M.I.: Attractor dynamics and parallelism in a connectionist sequential machines. In: Proceedings of the 8th Conference of the Cognitive Science Society, pp. 531–546. Erlbaum (1986)
[33] Poddar, P., Unnikrishnan, K.P.: Non-linear prediction of speech signals using memory neuron networks. In: Juang, B.H., Kung, S.Y., Kammedts, C.A. (eds.) Proceedings of the 1991 IEEE Workshop on Neural Networks for Signal Processing, pp. 1–10 (1991)
[34] Gers, F.A., Schmidhuber, J.: LSTM Recurrent Networks Learn Simple Context-Free and Context-Sensitive Languages. IEEE Transactions on Neural Networks 12(6), 1333–1340 (2001)
[35] De Vries, B., Principe, J.C.: The gamma model – A new neural network for temporal processing. Neural Networks 5, 565–576 (1992)
[36] Chalup, S.K., Blair, A.D.: Incremental Training of First Order Recurrent Neural Networks to Predict a Context-Sensitive Language. Neural Networks 16, 955–972 (2003)
[37] Lin, T., Horne, B.G., Giles, C.L.: How embedded memory in recurrent neural network architectures helps learning long-term temporal dependencies. Neural Networks 11, 861–868 (1998)

[38] Hopfield, J.: Neural networks and physical systems with emergent collective computational abilities. Proceedings National Academy of Sciences, USA 79, 2554–2558 (1982)
[39] Santini, S., Del Bimbo, A., Jain, R.: Block-Structured Recurrent Neural Networks. Neural Networks 8(1), 135–147 (1995)
[40] Cetin, B.C., Burdick, J.W., Barhen, J.: Global descent replaces gradient descent to avoid local minima problem in learning with artificial neural networks. In: Proceedings of of the IEEE International Conference on Neural Networks, Piscataway, NJ, USA, vol. 2, pp. 836–842 (1993)
[41] Dayhoff, J.E.: The exclusive-OR: A classic problem. In: Neural Network Architectures: An Introduction, pp. 76–79. Van Nostrand Reinhold, New York (1990)
[42] Gori, M., Tesi, A.: On the problem of local minima in backpropagation. IEEE Transactions on Pattern Analysis and Machine Intelligence 14, 76–85 (1992)
[43] Galiki, M., Leistritz, L., Witte, H.: Learning Continuous Trajectories in Recurrent Neural Networks with Time-Dependent Weights. IEEE Transaction on Neural Networks 10(4), 741–756 (1999)
[44] Leistritz, L., Galicki, M., Witte, H., Kochs, E.: Training Trajectories by Continuous Recurrent Multilayer Networks. In: IEEE International Joint Conference on Neural Networks, pp. 283–291 (2002)
[45] Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. In: Parallel Distributed Processing, pp. 318–362. MIT Press, Cambridge (1986)
[46] Hamey, L.G.C.: XOR has no local minima: A case study in neural network error surface analysis. Neural Networks 11, 669–681 (1998)