New Approach of the Recurrent Neural Network Training

V. Golovko, Y. Savitsky
Brest Polytechnic Institute
Moskowskaja str. 267, Brest 224017
Belarus, e-mail: cm@brpi.belpak.brest.by

Abstract

In this work the technique of creation of adaptive training algorithms for recurrent neural networks (RNN) is considered. These algorithms have high convergence and accuracy on a comparison with traditional backpropagation. The original technique of calculation of an adaptive training step with use of steepest descent method is resulted. The features of calculation of an adaptive pitch for neural elements with recurrent connections are discussed. Are considered the neural units with various functions of activation, used in architectures neural systems of forecasting. The indicated computing experiments demonstrate advantage of the developed RNN training methods.

1: Introduction

The training of RNN with plenty of recurrent connections with use backpropagation is a large problem. It is connected to defects of training algorithm and complexity of neural network architecture. Therefore there is a problem of development of adaptive training algorithms permitting to execute RNN training of a complicated structure with a high exactitude and a speed [3-6]. In this work the technique of calculation of an adaptive training step is resulted which can be applied to any activation of neural elements. The adaptive step for sigmoid, logarithmic and linear functions of activation used by the authors for construction of systems of forecasting is considered.

2: Basic RNN Architecture

This work is focused in the fully connected third-layered recurrent neural network (RNN) architecture, as shown in a Fig 1 [3]. The output activity of the neural network is defined by expression:

$$Y_j(t) = F\left(\sum_{i=1}^{N_h} w_{ij} y_i(t) - T_j \right) \quad (1)$$

where N_h - number of units of the hidden level, $y_i(t)$ - output activity of hidden units, s_j - threshold for output unit, w_{ij} - weights from hidden input units i to the output unit j, $j = \{1, N_o\}$.

The output activity of the hidden units on the current training iteration t for training exemplar p is defined as:

$$y_j(t) = F\left(\sum_{i=1}^{N_I} w_{ij} x_i(t) + \sum_{k=1}^{N_h} w_{kj} y_k(t-1) + \sum_{l=1}^{N_o} w_{lj} Y_l(t-1) - s_j \right) \quad (2)$$

where $x_i(t)$ is a i'th element of the input vector $x(t)$, N_I - size of an input vector, w_{ij} - weights from external units i to hidden units j, w_{kj} - weights from hidden units k to hidden units j, $y_k(t-1)$ - output activity of a hidden unit k for the previous moment of time, w_{lj} - weight to the hidden unit j from an output unit l, $Y_l(t-1)$ - network output activity for the previous moment of time and s_j - thresholds of the hidden units.

In this work we use three types of neural units transfer function:

- *Linear* activation function:

$$F(S_j) = M \cdot S_j \quad (3)$$

This function is used as RNN output units in the some types of the neural prediction systems.

- *Logarithmic* activation function:

$$F(S_j) = \ln\left((S_j + \sqrt{(S_j)^2 + a}) / \sqrt{a} \right), \ (a > 0) \quad (4)$$

The choice by this transfer function is stipulated by that it is unlimited on all define area. It allows better to simulate and predict complex non-stationary processes. We propose to use this function in the hidden units of RNN. The parameter a defines declination of the activation function (see Fig. 2).

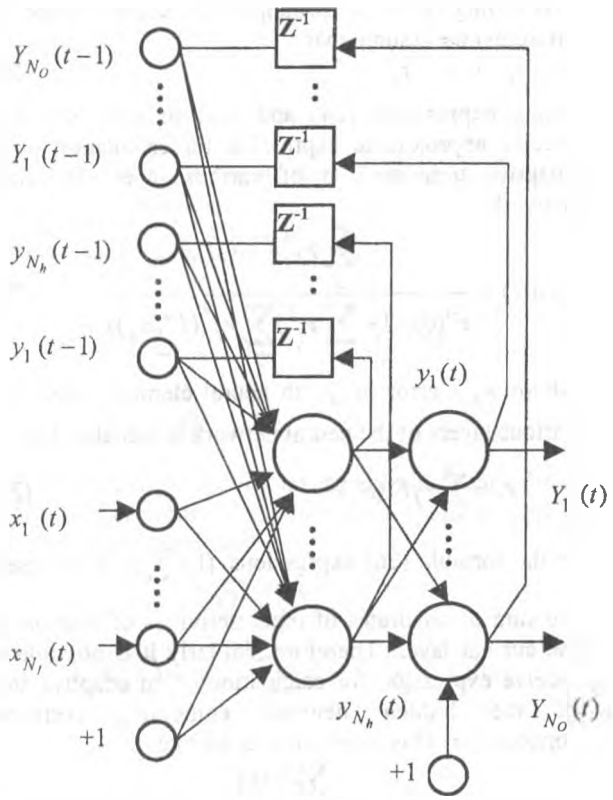

Figure 1. The RNN architecture

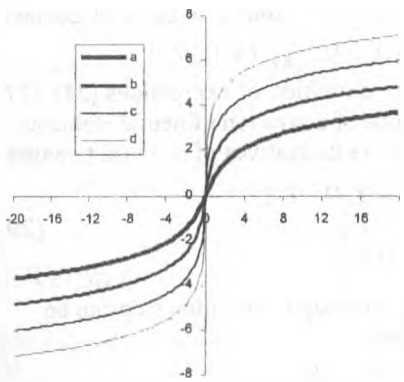

Figure 2. The logarithmic activation function for various parameters a: a) $a = 1.0$, b) $a = 0.1$, c) $a = 0.01$, d) $a = 0.001$

- At other case in hidden units is used standard *sigmoid* transfer function, defined as:

$$F(S_j) = \left((1 + e^{-S_j}\right)^{-1} \tag{5}$$

Let's consider the adaptive training step calculation for mentioned above transfer functions.

3: Adaptive Training Step Calculation for RNN

In standard backpropagation algorithm there is a problem of choice of an optimal training step to increase speed of training. For choice of adaptive step it would be possible to use a method of steepest descent. According to it, on each iteration of neural network training, the training step for each layer by such is necessary to select to minimize a root-mean-square error of a neural network:

$$\alpha(t) = \min E(y_j(t+1)) \tag{6}$$

where $j = \{1, N_O\}$, N_O - number of neural units of the output layer.

The output value of the j'th neuron depends on function of activation of units and is generally determined as follows:

$$y_j(t+1) = F(w_{ij}(t+1), T_j(t+1)) \tag{7}$$

The weights and thresholds are modified as:

$$w_{ij}(t+1) = w_{ij}(t) - \alpha(t)\frac{\partial E}{\partial w_{ij}(t)}, \qquad T_j(t+1) = T_j(t) - \alpha\ (t)\frac{\partial E}{\partial s_j(t)}, \tag{8}$$

Root-mean-square error of the RNN is defined as:

$$E = \frac{1}{2}\sum_{j=1}^{N_O}(Y_j - t_j)^2 \tag{9}$$

Then for determination $\alpha(t)$ it is necessary to find

$$\frac{\partial E}{\partial \alpha(t)} = \frac{\partial E}{\partial Y_j(t+1)}\frac{\partial Y_j(t+1)}{\partial \alpha(t)}. \tag{10}$$

The given equation cannot be decided rather $\alpha(t)$ by an analytical way. Therefore in the series of publications for definition of an adaptive training step it is offered to use methods of linear search [1]. However it is connected to difficult calculations. Therefore it is possible to offer an approximate method of a determination of a training step $\alpha(t)$. It bases on expansion of the neural unit activation function in a number Tailor serial. Let's consider it explicitly.

Let output value of the j'th neuron of the output layer is equaled:

$$Y_j(t) = F(S_j(t)), \qquad S_j(t) = \sum_i y_i(t) w_{ij}(t) - T_j(t), \tag{11}$$

where $y_i(t)$ - output value of the i'th neuron of the hidden layer.

For definition of the weighted sum of the j'th neuron in the moment of time $t+1$ we shall use in (11) expressions (8):

$$S_j(t+1) = \sum_i y_i(w_{ij} - \alpha\frac{\partial E}{\partial w_{ij}}) - T_j + \alpha\frac{\partial E}{\partial T_j} =$$
$$= \sum_i y_i w_{ij} - T_j - \alpha\cdot(\sum_i y_i\cdot\frac{\partial E}{\partial w_{ij}} - \frac{\partial E}{\partial T_j}). \quad (12)$$

We shall designate

$$a_j = \sum_i y_i\frac{\partial E}{\partial w_{ij}} - \frac{\partial E}{\partial T_j}. \quad (13)$$

Then the expression (12) can be presented as follows:

$$S_j(t+1) = S_j(t) - \alpha\cdot a_j. \quad (14)$$

The target value of j th neuron in the moment of time $t+1$ is equaled:

$$Y_j(t+1) = F(S_j(t+1)). \quad (15)$$

Let's decompose this expression under the Tailor formula and limits by first two members:

$$Y_j(t+1) = F(0) + F'(0)\cdot S_j(t+1), \quad (16)$$

where

$$F'(0) = \frac{\partial F}{\partial S_j} \text{ for } S_j = 0.$$

We shall use in (16) expressions (14). Then

$$Y_j(t+1) = F(0) + F'(0)S_j(t) - \alpha F'(0)a_j \quad (17)$$

As

$$Y_j(t) = F(0) + F'(0)S_j(t), \quad (18)$$

that expression (17) can be presented as follows:

$$Y_j(t+1) = Y_j(t) - \alpha F'(0)a_j \quad (19)$$

For definition of an adaptive training step it is necessary to supply:

$$E = \frac{1}{2}\sum_j (Y_j(t+1) - t_j)^2 \to \min \quad (20)$$

Then

$$\frac{\partial E}{\partial \alpha} = \sum_j (Y_j(t) - t_j - \alpha F'(0)a_j)\cdot(-F'(0)a_j) = 0 \quad (21)$$

Expressing from the last equation, we shall receive:

$$\alpha(t) = \frac{\sum_j (Y_j(t) - t_j)a_j}{F'(0)\sum_j a_j^2} \quad (22)$$

As $\frac{\partial^2 E}{\partial \alpha^2} > 0$, for want of given α the minimum of a root-mean-square error is ensured. Let's find expression for a_j. For this purpose we shall define:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial Y_j}\cdot\frac{\partial Y_j}{\partial s_j}\cdot\frac{\partial S_j}{\partial w_{ij}} = (Y_j - t_j)F'(S_j)\cdot y_i,$$
$$\frac{\partial E}{\partial T_j} = \frac{\partial E}{\partial Y_j}\cdot\frac{\partial Y_j}{\partial S_j}\cdot\frac{\partial S_j}{\partial T_j} = -(Y_j - t_j)F'(S_j). \quad (23)$$

Using (22) in expression (12), we shall receive:

$$a_j = (1+\sum_i y_i^2)\cdot(Y_j - t_j)\cdot F'(S_j) \quad (24)$$

Proceeding from a principle of independence of stratums, we assume that

$$\gamma_j = Y_j - t_j \quad (25)$$

Using expressions (24) and (25) in (22), we shall receive approximate expression for calculation of an adaptive training step of various layers if neural network:

$$\alpha(t) = \frac{\sum_j \gamma_j^2 F'(S_j)}{F'(0)\cdot(1+\sum_j y_i^2)\sum_j \gamma_j^2 (F'(S_j))^2} \quad (26)$$

where γ_j - error of j 'th neural element, which for various layers of the neural network is calculated as:

$$\gamma_i = \sum_{j=1}^{N_{i+1}} \gamma_j F'(S_j)w_{ij} \quad (27)$$

In the formula (26) expressions $(1+\sum_i y_i^2)$ represent the sum of quadrates of input activities of neurons of the current layer. Therefore similarly it is possible to receive expression for calculation of an adaptive step of the hidden neurons containing recurrent connections. This expression looks like:

$$\alpha(t) = \frac{\sum_j \gamma_j^2 F'(S_j)}{F'(0)\cdot(1+\sum_i x_i^2(t) + \sum_k (y_k(t-1))^2 + \sum_l (y_l(t-1))^2)\sum_j \gamma_j^2 (F'(S_j))^2}, \quad (28)$$

where $y_k(t-1), y_l(t-1)$, - source activity of context-sensitive neurons, $k = \{1, N_h\}, l = \{1, N_O\}$.

Let's consider definition of expressions (25), (27) for various functions of activation of neural elements.

Sigmoid function. As derivatives of sigmoid function:

$$Y_j' = F'(S_j) = Y_j(1 - Y_j),$$
$$Y_j'(0) = F'(0) = \frac{1}{4}, \quad (29)$$

that expression for an adaptive training step can be presented as follows:

$$\alpha(t) = \frac{4\sum_j \gamma_j^2 Y_j(1 - y_j),}{(1+\sum_i y_i^2)\cdot\sum_j \gamma_j^2 Y_j^2 (1 - Y_j)^2} \quad (30)$$

For neurons of the hidden layer, with allowance for of recurrent connections, the adaptive training step is determined by expression:

$$\alpha(t) = \frac{4\sum_j \gamma_j^2 y_j(1-y_j),}{(1+\sum_i x_i^2 + \sum_k (y_k(t-1))^2 + \sum_l (y_l(t-1))^2)\cdot\sum_j \gamma_j^2 y_j^2 (1-y_j)^2} \quad (31)$$

Logarithmic function. Derivatives of logarithmic function is:

$$Y_j' = F'(s_j) = \frac{1}{\sqrt{S_j^2 + a}}.$$
$$Y_i'(0) = F'(0) = \frac{1}{\sqrt{a}}, \quad (32)$$

that expression for an adaptive training step can be presented as follows:

$$\alpha(t) = \frac{\sqrt{a}\sum_j \gamma_j^2 \left(\sqrt{(S_j)^2 + a}\right)^{-1}}{(1 + \sum_i y_i^2) \cdot \sum_j \gamma_j^2 \left((S_j)^2 + a\right)^{-1}}, \quad (33)$$

$$\gamma_i = \sum_j \gamma_j y_j (1 - y_j) w_{ij} \quad (34)$$

For neurons of the hidden layer, with allowance for of recurrent connections, the adaptive training step is determined by expression:

$$\alpha(t) = \frac{\sqrt{a}\sum_j \gamma_j^2 \left(\sqrt{(S_j)^2 + a}\right)^{-1}}{(1 + \sum_i y_i^2 + \sum_k (y_k(t-1))^2 + \sum_l (y_l(t-1))^2) \cdot \sum_j \gamma_j^2 \left((S_j)^2 + a\right)^{-1}}, \quad (35)$$

$$\gamma_i = \sum_j \gamma_j \frac{1}{\sqrt{S_j^2 + a}} w_{ij} \quad (36)$$

Linear function. As derivatives of linear function:

$$Y_j' = F'(S_j) = M,$$
$$Y_j'(0) = F'(0) = M, \quad (37)$$

then we receive the next expression:

$$\alpha(t) = \frac{M\sum_j \gamma_j^2}{M(1 + \sum_i y_i^2) \cdot \sum_j \gamma_j^2 M^2} = \frac{1}{M^2(1 + \sum_i y_i^2)} \quad (38)$$

4: Testing

For simulation two types of RNN architectures were taken. One of them consists from neurons with sigmoid activation function in the hidden layer. In the other case were used hidden units with logarithmic transfer function. Both neural networks contained 20 inputs, 5 nonlinear elements in the hidden layer and one output linear neuron. For training set organization were used the time serial of passenger airtransportations described in [2]. It size is 144 units. The testing task is forecasting of the described below time serial. For training both neural networks 3000 training iterations were executed. In procedures of training the appropriate expressions for calculation of an adaptive pitch were used. For a comparison the training with various constant steps were executed. The quality of training was inspected by a root-mean-square training error, as (9). In all cases higher efficiency of adaptive algorithms was observed (see Table 1).

Table 1. RNN training results

Type of training step	Training error for RNN with logarithmic transfer function a=0.01	Training error for RNN with sigmoid transfer function
Adaptive	$1.78E^{-5}$	$6.92E^{-4}$
$\alpha = 0.0099$	$3.91E^{-2}$	$3.11E^{-1}$
$\alpha = 0.099$	$7.13E^{-3}$	$5.01E^{-2}$
$\alpha = 0.99$	$9.11E^{-4}$	$9.78E^{-3}$
$\alpha = 1.99$	$5.21E^{-4}$	$6.97E^{-3}$
$\alpha = 9.99$	$2.87E^{-1}$	$9.98E^{-4}$
$\alpha = 19.99$	1.09	$8.78E^{-3}$

5: Conclusion

In this work the problem of fast training of recurrent multilayer neural networks with complicated structure is partially decided. The circumscribed original technique of calculation of an adaptive training step can be applied for any functions of activation of neuroelements. In work the application of the developed algorithms on an example of concrete types of RNN architectures is shown. The computing experiments demonstrate potential abilities of the developed adaptive algorithms for training of complicated neural networks.

Acknowledgments
This paper is supported by INTAS program "INTAS OPEN 97-0606". The authors express gratitude to the European Union for financial support.

6: References

1. Hortz J., Krogh A., Palmer R. Introduction to the theory of neural computation. – Addison Wesley Publishing Company.–1991.–327p.
2. Box G.E.P., Genkins G. M. Time-Series Analysis, Forecasting and Control. New York, 1970.
3. Pedersen M. Training Recurrent Networks // Proceeding of the IEEE Workshop on Neural Networks for Signal Processing VII. –New Jersey: IEEE.–1997.
4. T.Lin, B.G. Horne, P.Tino., C.L. Giles. Learning Loln-Term Dependies with NARX Recurrent Neural Networks. // IEEE Transactions on Neyral Networks., vol. 7, no. 6, p. 1329, - 1996.
5. M.W. Pedersen, L.K. Hansen. Recurrent Neural Networks: Second-Order Properties and Pruning. // Advanced in Neural Information Processing Systems 7, Cambridge, MA: The MIT Press, 1995, pp. 673-680.
6. Pineda F. Generalization of back-propagation to recurrent neural network. // Physical Review Letters, 19(59), 2229-2232.- 1987.