ESANN'2009 proceedings, European Symposium on Artificial Neural Networks - Advances in Computational Intelligence and Learning. Bruges (Belgium), 22-24 April 2009, d-side publi., ISBN 2-930307-09-9.

Monotonic Recurrent Bounded Derivative Neural Network

Alexey Minin[1] and Bernhard Lang[1] *

1- Siemens OOO Fault Analysis and Prevention
191186, St. Petersburg, Volinskiy lane 3, Russia.

Abstract. Neural networks applied in control loops and safety-critical domains have to meet hard requirements. First of all, a small approximation error is required, then, the smoothness and the monotonicity of selected input-output relations have to be taken into account and finally, for some processes, time dependencies in time series should be induced into the model. If not then the stability of the control laws can be lost. In the following paper authors present new Monotonic Recurrent Bounded Derivative Network (RBDN) on the basis of the Bounded Derivative Network (BDN) [1]. Authors compared invented network with other known networks, investigated the influence of the back connection in recurrent network, stability and monotonicity of the new recurrent network. This paper is also an attempt to incorporate Input/Output monotonicity into the recurrent network nodes (weights).

1 Bounded Derivative Neural Network

Following the work [1] consider multi layer perceptron. This class of networks consists of multiple layers of computational units, usually interconnected in a feed-forward way. Each neuron in one layer has directed connections to the neurons of the subsequent layer. In many applications the units of these networks apply a hyper tangent function as an activation function (see Eq.1): General equation of the MLP is given in Eq.(1).

$$Y = \sum_{k=1}^{nh4} w_k^{4,6} \tanh\left(\sum_{l=1}^{nh2} w_{k,l}^{2,4} \tanh\left(\sum_{i=1}^{ni} w_i^{0,2} X_i + w_i^1\right) + w_l^3\right) + w^5 \qquad (1)$$

where upper index of the w shows layer number and lower index shows node number inside the layer, i stands for input number, l is the node number in 2^{nd} layer. The Bounded Derivative Network (BDN) is the analytical integral of a

*Some datasets used in the paper were downloaded from: Asuncion, A. and Newman, D.J. (2007). UCI Machine Learning Repository [http://www.ics.uci.edu/ mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science.

multi layer perceptron (2 hidden layers) neural network (see Eq.3).

$$\begin{aligned} Y = \quad & w_{1,1}^{6,1} + \sum_{i=1}^{ni} w_{1,i}^{6,2} w_{i,i}^{2,0} X_i + \sum_{j=1}^{nh} w_{1,j}^{6,5} w_{j,j}^{5,4} \times \\ & \times \left[\log \left(\cosh \left(w_{j,1}^{3,1} + \sum_{i=1}^{ni} w_{j,i}^{3,2} w_{i,i}^{2,0} X_i \right) \right) + \right. \\ & \left. + w_{j,j}^{5,3} \left(w_{j,1}^{3,1} + \sum_{i=1}^{ni} w_{j,i}^{3,2} w_{i,i}^{2,0} X_i \right) \right] \end{aligned} \tag{2}$$

For graphical visualization see Fig.1 below.

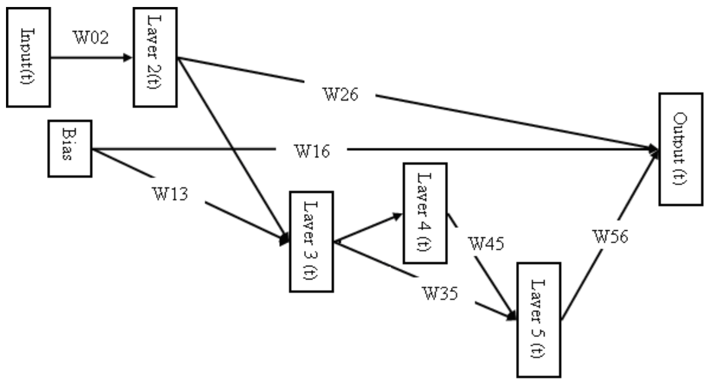

Fig. 1: Bounded derivative network is shown at the figure. Here "Input(t)" are inputs at time moment t, W are weights between layers, Y are outputs. Connections between nodes are presented with the lines

Note that interconnection is performed only between layers 2 and 3 (see Eq. 3). BDN is an "interesting" network since according to [1] it incorporates monotonicity rules by its structure. Under monotonicity authors understand the following rules: if A increases then B will increase as well can be transferred to a monotonicity constraint dB/dA > 0. The derivative of the BDN activation function (see Eq.3 and Fig.2) is limited due to the limitation of hyperbolic tangent function. The derivative is presented at the Eq.3:

$$\begin{aligned} \frac{\partial Y}{\partial X_k} = w_{k,k}^{2,0} \times \\ \times \left(w_{1,k}^{6,2} + \sum_{j=1}^{nh} w_{1,j}^{6,5} w_{j,k}^{3,2} \left(w_{j,j}^{5,3} + w_{j,j}^{5,4} \tanh \left[w_{j,1}^{3,1} + \sum_{i=1}^{ni} w_{j,i}^{3,2} w_{i,i}^{2,0} X_i \right] \right) \right) \end{aligned} \tag{3}$$

From the Fig.2 it is clear that BDN has constrained derivative and unconstrained monotone activation function. In case BDN one can see that extrapola-

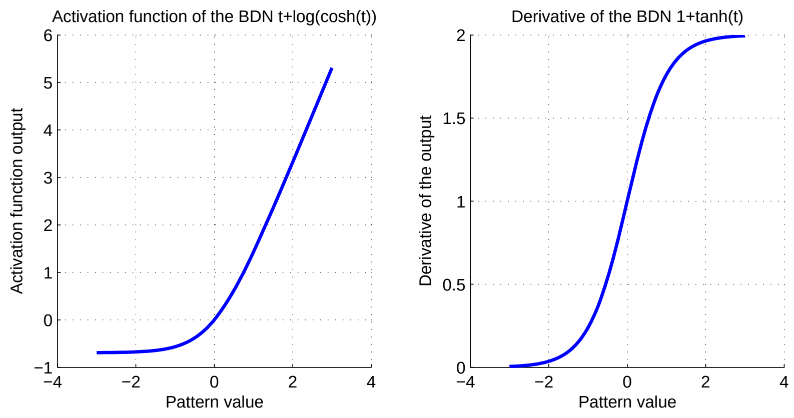

Fig. 2: BDN activation function and its derivative

tion (activation function) is linear in its major part and has very small nonlinear domain [1]. Thus having such type of activation function monotonicity is guaranteed.

2 Recurrent Bounded Derivative Neural Network

Taking into account [4], namely ability of neural networks to provide monotonic behavior for input-output relations, and inability to extract time structure of the training patterns [5], authors decided to create monotonic recurrent neural network, which should incorporate monotonicity. For this purpose RBDN was created. To make BDN of recurrent type (to provide the ability for time structure extraction out of the training patterns) one should provide back connection [5] for the 3rd layer (see fig. 1). The resulting architecture is presented at the Fig.3 and the resulting equation is presented at the eq.4. Here Layer 3(t-1) contains previous state for the 3rd layer (Layer3(t)). Layer3(t-2) contains state of the Layer3 at the time moment (t-2) etc till Layer3(t-n).

$$\begin{aligned} Y &= w^{6,1}_{1,1} + \sum_{i=1}^{ni} w^{6,2}_{1,i} w^{2,0}_{i,i} X_i + \sum_{j=1}^{nh} w^{6,5}_{1,j} \times \\ &\times \left[w^{5,4}_{j,j} \log\left(\cosh\left(w^{3,1}_{j,1} + T\right)\right) + w^{5,3}_{j,j}\left(w^{3,1}_{j,1} + T\right)\right] \end{aligned} \tag{4}$$

In the Eq.4 T stands for the follwoing term:

$$T = \sum_{i=1}^{ni} \tanh\left(w^{3,2}_{j,i} w^{2,0}_{i,i} X_i\left(t-1\right)\right) + w^{3,2}_{j,i} w^{2,0}_{i,i} X_i\left(t\right)$$

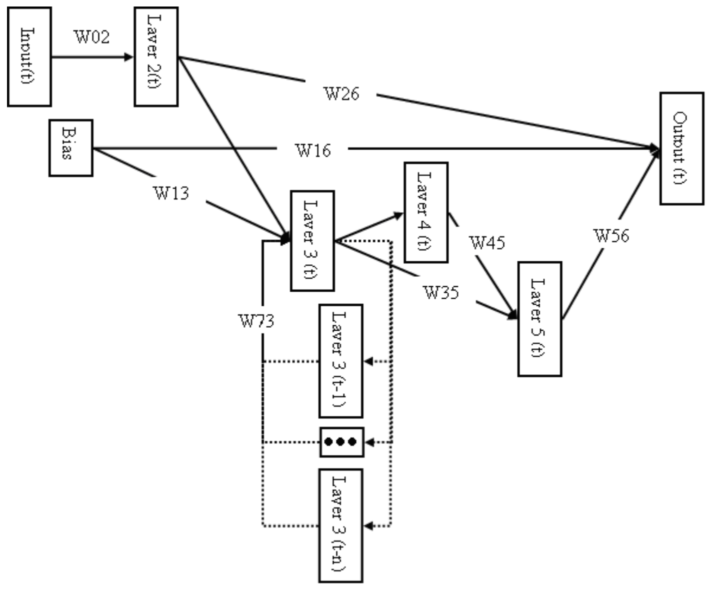

Fig. 3: Recurrent Bounded Derivative Network. Here "Input (t)" is input at time moment t, W are weights between layers, Y are outputs. Connections between nodes are presented with the lines. Dotted line shows back connection.

Note that interconnection is performed only between layers 2 and 3 (see Eq. 4). The derivative of the RBDN activation function (see Eq.5 and Fig.4) is limited due to the superposition of the hyperbolic tangent function and derivative for the hyperbolic tangent function (see Eq.5). The derivative of the RBDN consists of the derivative for the Eq.1 (feed forward perceptron) and also consists of the derivative for the BDN (see Eq.5).

$$\frac{\partial Y}{\partial X_k} = w_{1,j}^{6,5}\left(\left(w_{j,j}^{5,3}+T\right)\frac{\partial T}{\partial X_k} + w_{j,j}^{5,4}\tanh\left(w_{j,1}^{3,1}+T\right)\frac{\partial T}{\partial X_k} + w_{1,k}^{6,2}w_{k,k}^{2,0}\right) \tag{5}$$

where

$$\frac{\partial T}{\partial X_k} = w_{k,k}^{2,0}w_{j,k}^{3,2}\left[\frac{1}{cosh^2\left(w_{k,k}^{2,0}w_{j,k}^{3,2}\right)}+1\right]$$

Activation function and its derivative are presented at the figure 4 below. As one can see from the figure 4 activation function of the RBDN is very similar to the BDN (see fig. 2). Hyper tangent activation function is used in order to limit the output of recurrent layer; otherwise it can lead to the unlimited growth of the output. One should note that hyper tangent function and its derivative are

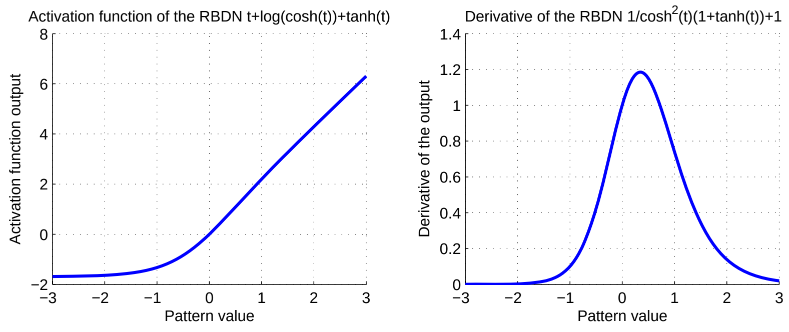

Fig. 4: Activation function and its derivative for the RBDN

limited. RBDN derivative is still limited and activation function is linear in its major part and is monotonic.

3 Stability and training issues

To optimize weights of the RBDN Sequential Quadratic Programming, namely SQP was used. In order to calculate the constraints of the model one should consider the Eq.5 more precise. Since $tanh(x) \in (-1;1)$ and $1/cosh^2(x) \in (0;1]$ one can rewrite Eq.5 in the following way (see Eq. 6):

$$\frac{\partial Y}{\partial X_k} = w_{k,k}^{2,0} \times \left(w_{1,k}^{6,2} + \sum_{j=1}^{nh} w_{j,k}^{3,2} w_{1,j}^{6,5} \left(w_{j,j}^{5,3} + w_{j,j}^{5,4} \right) \right) \geq 0 \tag{6}$$

To train RBDN back propagation algorithm was used. During the training the error decay of the RBDN is robust. SQP iterations converge to the optima in case feasible starting point (initial nodes values). Calculation of the feasible starting point is easy to do and is described in [1]. To give the impression about the speed of the RBDN training (feasible starting point) note that to train RBDN for the Abalone dataset one will need 5 min at 2GHz CPU with 2 GB of RAM (1 recurrent layer with 5 hidden nodes, 1200 training patterns).

4 Results and outlook

RBDN was tested together with BDN onto different examples. Several dataset were time series and some were not. For the results see table 1. For each dataset 1 step forecast (for the whole test set) was provided. To estimate the modelling quality R^2, the so called determination coefficient, was used. Number of nodes in the table 1 stands for the amount of hidden nodes the 3rd layer (see fig. 3).

The main advantage of the RBDN is that it implies different models in one(linear regression, recurrent networks).The good property is that RBDN

		RMS	RMS	R^2	R^2
Dataset	$node(s)$	BDN	$RBDN$	BDN	$RBDN$
Abalone	5	2.10	2.20	0.10	0.12
See [2], 1200TP	10	1.90	2.34	0.30	0.44
Dow Jones,	1	14.10	14.10	0.64	0.64
3000TP,	5	14.10	14.00	0.65	0.64
TS	10	14.10	14.00	0.64	0.65
EEG, 2000TP,TS	5	0.53	0.34	0.99	0.99
Ball Bearings,	1	9.6e-3	9.7e-3	0.74	0.74
see [3],3000TP,TS	5	9.7e-3	9.9e-3	0.74	0.72

Table 1: Summary for RBDN and BDN. Time series are marked in the table in the filed "Dataset". TP stands to show number of training patterns. Number of nodes is presented for the 3rd layer (see fig. 3). TS stands to show time series datasets.

needs less patterns for the training then BDN. Convergence of the SQP solver for the RBDN is stable. In case process model is unknown it is better to have the architecture like RBDN which implies different models in one in order to have some guaranteed output quality. For the non time series (or non cause-and-effect relations data) recurrent layer is some kind of noise layer and sometimes it allows SQP solver not to stack in local optimum (random perturbation of weights). Therefore RBDN can be considered as a universal architecture which can be used for big variety of problems.

References

[1] P. Turner, J. Guiver,L.: Brian, Introducing The State Space Bounded Derivative Network For Commercial Transition Control, *Proceedings of the American Control Conference*, Denver, Colorado June 4-6 (2003)

[2] A. Minin, B. Lang: Comparison of Neural Networks Incorporating Partial monotonicity by Structure, *International Conference on Artificial Neural Networks 2008*, Springer, vol.2, pp. 597-607, 2008

[3] I. Mokhov, A. Minin: Advanced forecasting and classification technique for condition monitoring of rotating machinery, *Proceedings of the 8th International Conference On Intelligent Data Engineering and Automated Learning (IDEAL'07)*, Birmingham, UK, December 16-19, 2007, Springer, pp. 37-46

[4] Lang, B.: Monotonic Multi-layer Perceptron Networks as Universal Approximators. Formal Models and Their Applications,*International Conference on Artificial Neural Networks*, Springer (2005), vol. 3697, pp. 31-37

[5] Boden, M. A guide to recurrent neural networks and backpropagation, *in The DALLAS project. Report from the NUTEK-supported project AIS-8: Application of Data Analysis with Learning Systems*, 1999-2001. Holst, A. (ed.), SICS Technical Report T2002:03, SICS, Kista, Sweden, 2002.