Performance Analysis of the Activation Neuron Function in the Flexible Neural Tree Model

Tomáš Buriánek and Sebastián Basterrech

IT4Innovation
VŠB–Technical University of Ostrava,
Czech Republic
{Tomas.Burianek.St1,Sebastian.Basterrech.Tiscordio}@vsb.cz

Abstract. The time series prediction and forecasting is an important area in the field of Machine Learning. Around ten years ago, a kind of Multilayer Neural Network was introduced under the name of *Flexible Neural Tree (FNT)*. This model uses meta-heuristic techniques to determinate its topology and its embedded parameters. The FNT model has been successfully employed on time-series modeling and temporal learning tasks. The activation function used in the FNT belongs to the family of radial basis functions. It is a parametric function and the parameters are set employing an heuristic procedure. In this article, we analyze the impact on the performance of the FNT model when it used other family of neuron activation functions. For that, we use the *hyperbolic tangent* and *Fermi activation* functions on the tree nodes. Both functions have been extensively used in the field of Neural Networks. Moreover, we study the FNT technique with a linear variant of the Fermi function. We present an experimental comparison of our approaches on two widely used time-series benchmarks.

Keywords: Neural Network, Flexible Neural Tree, Neuron Activation Function, Time-series modeling, Forecasting

1 Introduction

The *Flexible Neural Tree (FNT)* have been used for several time-series problems as learning predictor. The model consists of an interconnected nodes forming a tree architecture [9, 10]. There are two kind of nodes, functional and terminal nodes. The terminal nodes contains the information of the input patterns. The functional nodes process the information using a specific activation function. The parameters of the model are: the weight connections among the nodes, the parameters in the activation function and the pattern of connectivity on the tree. The method combines two heuristic algorithms in order to find theses parameters. Several bio-inspired methods can be used as meta-heuristic technique to find the topology of the tree such as: Probabilistic Incremental Program Evolution (PIPE), Genetic Programing (GP), Ant Programming (AP). In order to find the embedded parameters, it can be used: Genetic Algorithms (GA), Particle Swarm Optimization (PSO) and Differential Evolution (DE), and so on.

J. Pokorný, K. Richta, V. Snášel (Eds.): Dateso 2014, pp. 35–46, ISBN 978-80-01-05482-6.

The first FNT model was developed with a Gaussian activation function in the functional nodes.

In this paper we analyze the FNT performance when another kind of activation functions is used in the nodes. We study the nodes with hyperbolic tangent and Fermi activation function, both family of functions were extensively studied in the field of Neural Networks. Additionally, we test the model with a linear variant of the Fermi activation neurons. We present an experimental comparison of the different activation functions on two widely used time-series benchmarks. The first one is a simulated benchmark commonly used in the forecasting literature called 20-order fixed NARMA data set. The another one is a real data set about the Internet traffic from an European Internet service provider.

The paper is organized as follows. In Section 2, we present a description of the Flexible Neural Tree model. Section 3 contains the description of other activation functions. Next, we present our experimental results. Finally, the last part presents the article conclusion.

2 The Flexible Neural Tree model description

About ten years ago a new kind of *Neural Network (NN)* was introduced under the name of *Flexible Neural Tree (FNT)* [9, 10]. A FNT is a multilayer feed-forward NN with an irregular topology. In the original FNT model each neuron has a parametric activation function. The network architecture is designed in an automatic way considering a pre-defined set of instructions and functional operators. The automatic process involves the parameters of the activation function, defines the pattern of connectivity among the units and selects the input variables. An evolutionary procedure is used to evaluate the performance of the tree topology. In temporal learning tasks is hard to select the proper input variables, the FNT technique uses an automatic procedure for this selection. Another meta-heuristic algorithm is employed to find the neural tree parameters. The FNT model proposes to use a simple random search method for setting the tree parameters. For this task can be use some of the following techniques:*Particle Swarm Optimization (PSO)* [13, 19] and *Differential Evolution (DE)* [16, 20]. In the pioneering FNT approach was used the Probabilistic Incremental Program Evolution (PIPE) [17] for encoding the NN in a tree [9]. In the literature other techniques were studied to find the topology and tree parameters, such that the *Genetic Programing (GP)* [4–6, 8] and *Ant Programming (AP)* [**?**].

2.1 The tree construction

The topology of the tree is generated using a pre-defined instruction set. We consider the following *function set* $\mathcal{F} = \{+_2, +_3, \ldots, +_{\mathrm{N_f}}\}$. A *node operator* $+_i$ is an internal vertex instruction with i inputs. In the original FNT, the activation neuron function of any unit i is the following parametric function:

$$f(a_i, b_i, x) = e^{-(\frac{x-a_i}{b_i})^2}, \tag{1}$$

where the parameters a_i and b_i are adjustable parameters. The *terminal set* of the tree consists of $\mathcal{T} = \{x_1, x_2, \ldots, x_{\mathrm{N_t}}\}$. The instruction set $\mathcal{S}$ is defined as follows:

$$\mathcal{S} = \mathcal{F} \cup \mathcal{T} = \{+_2, +_3, \ldots, +_{\mathrm{N_f}}\} \cup \{x_1, x_2, \ldots, x_{\mathrm{N_t}}\}. \tag{2}$$

Each functional node $+_i$ has i random nodes as its inputs, which can be internal and terminal nodes. The model outputs can be computed using a depth-first strategy for traversing the tree. Given a functional node $+_i$ with inputs $\{x_1, \ldots, x_i\}$, the total input charge of $+_i$ is defined as:

$$net_i = \sum_{j=1}^{i} w_j x_j, \tag{3}$$

where w_j is the weight connection between x_j and $+_i$. The output of the node $+_i$ is obtained using the expression (1)

$$\mathrm{out_i} = \mathrm{e}^{-(\frac{\mathrm{net_i} - \mathrm{a_i}}{\mathrm{b_i}})^2}. \tag{4}$$

Let i be a functional node if some of its inputs are other functional nodes, then the output of i is computed in a recursive way following the expressions (3) and (4). The algorithm complexity for traversing the tree is $O(N)$ algorithmic time where N is the number of unit in the tree. Figure 1 illustrates an example of an FNT.

2.2 The fitness function

In order to evaluate the accuracy of the FNT model in learning tasks an error distance is considered. In this context, this performance measure is called *fitness function*. In a supervised learning problem given a training set $\{(\mathbf{x}(t), \mathbf{y}_{\mathrm{target}}(t)), t = 1, \ldots, T\}$, the goal is to infer a mapping $\varphi(\cdot)$ in order to predict $\mathbf{y}_{\mathrm{target}}$, such that the fitness function is minimized. Most often is used the *Mean Square Error* (MSE) or *Root Mean Square Error* (RMSE) [10], given by the expression:

$$MSE = \frac{1}{T} \sum_{t=1}^{T} (y(t) - \mathbf{y}_{\mathrm{target}}(t))^2. \tag{5}$$

Another important parameter of the model performance is the number of nodes in the tree. Obviously, between two trees with equal accuracy the smaller one is preferred.

2.3 The parameter optimization using meta-heuristic techniques

Several strategies have been used to find "good" parameters and topology [4, 7–10]. The model parameters embedded in the tree are the activation function parameters (a_i and b_i, for all $+_i$) and the weight connections. In this paper, we use Genetic Programming (GP) for finding a "good" connectivity among the

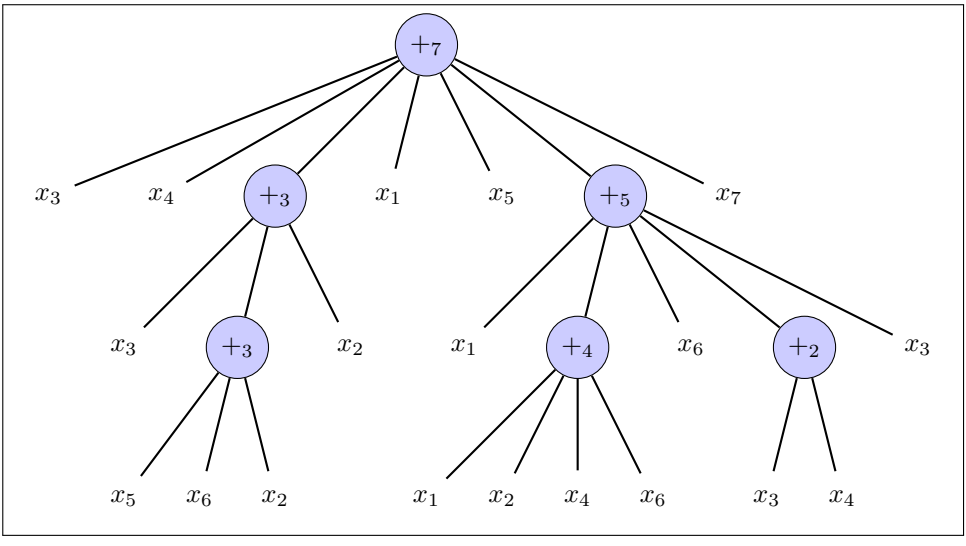

Fig. 1: Example of a Flexible Neural Tree with the function instruction set $\mathcal{F} = \{+_2, +_3, +_4, +_5, +_6, +_7\}$ and the terminal instruction set $\mathcal{T} = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$. The root in the tree $(+_7)$ is the output layer and the input layer is composed by the leaves in the bottom level of the tree.

neurons in the tree and we use the PSO method to find the embedded tree parameters.

The GP theory was intensely developed in the 90s [15]. A GP algorithm starts defining an initial population of same specific devices. The procedure is iterative, at each epoch it transforms a selected group of individuals producing a new generation. This transformation consists in applying some bio-inspired rules to the individuals. In our problem the individuals are the flexible trees. A set of individuals are probabilistically selected and a set of genetic rules is applied. The operating rules arise from some biological genetic operations, which basically consist of: *reproduction*, *crossover* and *mutation*. The reproduction is the identity operation, an individual i of a generation at time t is also presented at the generation at time $t + 1$. Given two sub-tree the crossover consists in interchanging their parents. The mutation consists in selecting a tree and realizing one of the following operations: to change a leaf node to another leaf node, to replace a leaf node for a sub-tree, and to replace a functional node by a leaf node. The GP algorithms applied for our specific problem is described in Algorithm 1.

There are two kinds of adjustable parameters on the tree: the activation function parameters a_i and b_i for each functional node i presented in the expression (1), another one refers to the weight connections between the nodes in the tree. In this paper, we use *Particle Swarm Optimization (PSO)* [13] for finding these parameters. The PSO algorithm is an evolutionary computation technique

Algorithm 1: Specification of the Genetic Programming algorithm used for finding the optimal flexible tree topology.

Inputs : $\mathrm{N_t}, \mathrm{N_f}$, training data set, algorithm parameters
Outputs: Tree

```
Initialize the population of trees;
Compute the fitness function for all trees;
while (Termination criterion is not satisfied) do
    // Creation of the new population
    while (Size of population is not satisfied) do
        Select genetic operation;
        if (Selection is crossover) then
            Select two trees from population using tournament selection;
            Realize the crossover operation;
            Insert the two new offspring into the new population;
        if (Selection is mutation) then
            Select one tree from population using tournament selection;
            Realize the mutation operation;
            Insert the new offspring into the new population;
        if (Selection is reproduction) then
            Select one tree from population;
            Insert it to the new population;
        Compute the fitness function of the new trees;
    Replace old population by the new population;
Return the tree with best fitness function.
```

based on the social behaviors in a simplified social environment. A *swarm* is a set of particles, which are characterized by their position and their velocity in a multidimensional space. We denote the position of a particle i with the column N_{x}-vector $\mathbf{x}^{(i)} = (x_1^{(i)}, \ldots, x_{N_{\mathrm{x}}}^{(i)})$. The velocity of i is defined by the column N_{x}-vector $\mathbf{ve}^{(i)} = (v_1^{(i)}, \ldots, v_{N_{\mathrm{x}}}^{(i)})$. Besides, we use auxiliary vectors $\mathbf{p}^*$ and $\mathbf{p}^{(i)}$, $\forall i$, each one has dimension $N_{\mathrm{x}} \times 1$. The vector $\mathbf{p}^{(i)}$ denotes the best position of i presented until the current iteration. The best swarm position is represented by the vector $\mathbf{p}^*$. Besides, we use two auxiliary random weights $\mathbf{r}_1^{(i)}$ and $\mathbf{r}_2^{(i)}$ of dimensions $N_{\mathrm{x}} \times 1$, which are randomly initialized in $[0, 1]$ for each particle i. At any iteration t, the simulation of the dynamics among the particles is given by the following expressions [18]:

$$v_j^{(i)}(t+1) = c_0 v_j^{(i)}(t) + c_1 r_{1_j}(t)^{(i)}(p_j^{(i)}(t) - x_j^{(i)}(t)) + c_2 r_{2_j}(t)^{(i)}(p_j^*(t) - x_j^{(i)}(t)), j \in [1, N_{\mathrm{x}}] \tag{6}$$

and

$$x_j^{(i)}(t+1) = x_j^{(i)}(t) + v_j^{(i)}(t+1), j \in [1, N_{\mathrm{x}}], \tag{7}$$

where the constant c_0 is called the *inertia weight* the constants c_1 and c_2 regulate local and global position of the swarm, respectively.

In order to use PSO for estimating the embedded tree parameters, the position $\mathbf{p}^{(i)}$ of the particle i is associated with the embedded parameters in one flexible tree (the weights and a_j, b_j, $\forall j \in \mathcal{F}$). The PSO algorithm return the global optimal position according to the fitness function, presented in the expression (5). The relationship between the tree parameters and the particle position is given by:

$$(p_1^{(i)}, \ldots, p_{N_x}^{(i)}) = (a_1, \ldots, a_{N_f}, b_1, \ldots, b_{N_f}, \mathbf{w}), \tag{8}$$

where $\mathbf{w}$ is a vector with the tree weights.

We present in the Algorithm 2 the PSO method used as meta-heuristic technique for finding the tree parameters.

Algorithm 2: Specification of the Particle Swarm Optimization used for finding the embedded tree parameters.

Inputs : N_x, number of particles, $\mathcal{S}$, training data set, algorithmic parameters
Outputs: Tree parameters

```
t = 0;
Random initialization of p^(i)(t) and v^(i)(t) for all i;
Compute the fitness value associated with i using (8) and the fitness function;
Set p*(t) and p^(i)(t) for all i;
while (Termination criterion is not satisfied) do
    for (Each particle i) do
        Compute v^(i)(t+1) using the expression (6);
        Compute x^(i)(t+1) using the expression (7);
        Compute the fitness value associated with i using (8) and the fitness function;
        Compute p^(i)(t+1);
    Compute p*(t+1);
    t=t+1;
Return the parameters using p*(t) and the expression (8);
```

3 Performance of other neuron activation functions in the nodes of the tree

In this paper we analyze the impact of other neuron activation function in the performance of the FNT model. The original model uses a Gaussian function

presented in the expression (4). This kind of function has been widely used in Support Vector Machine (SVM) where the model uses the radial basis function (RBF) kernel [11]. Additionally it has been used in Self-Organizing Maps (SOM) as *neighbourhood function* among the neurons on the Kohonen networks [14]. In the area of Neural Network two kind of activation function have been extensively used: the tanh(·) function and the Fermi function. The sigmoid activation function is

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{9}$$

The Fermi activation function is

$$f(x) = \frac{1}{1 + e^{-x}}. \tag{10}$$

In this paper we test the FNT model using the tanh(·) and Fermi function. Moreover, we analyze a parametric variation of these functions, given by:

$$g(x) = a_i f(x) + b_i, \tag{11}$$

where $f(x)$ is the function of expression (9) and (10) and the parameters a_i and b_i are specifics for each tree node i. We use PSO in order to find the parameters a_i and b_i.

4 Empirical results

4.1 Description of the benchmarks

We use two benchmarks, a simulated data set which has been widely used in the forecasting literature and a real data set about the Internet traffic data.

(1) Fixed kth order NARMA data set. This data set presents a high non-linearity, for this reason has been extensively analyzed in the literature [1,3],

$$b(t+1) = 0.3b(t) + 0.05b(t)\sum_{i=0}^{k-1} b(t-i) + 1.5s(t-(k-1))s(t) + 0.1,$$

where $k = 20$ and $s(t) \sim Unif[0, 0.5]$. The task consists to predict the value $b(t+1)$ based on the history of $b(t)$ up to time t. The size of the training data was 3980 and the test data numbered 780.

(2) The Internet traffic data from United Kingdom Education and Research Networking Association (UKERNA). The data was collected from 19 November and 27 January, 2005. This data set was analyzed in [2, 12]. The problem was studied collecting the data in five minute scale, The goal is to predict the traffic at time t using the information of the Traffic in the time $\{t-1, t-2, t-3, t-4, t-5, t-6\}$. This sliding window was studied in [12]. The training data corresponds to the initial 66% of the data. The size of the training set is the 13126 and the test set has 6762 patterns. We normalized the data in $[0, 1]$.

4.2 Results

We can see in the two graphics of Figure 2 the performance of the FNT prediction for the NARMA test data set using Gaussian activation function and Fermi activation function. Figure 3 shows the prediction of the model for the NARMA data set using the Gaussian activation function. In this last case we present the estimation on the last 200 samples. A example of estimation of the FNT using Gaussian is showed in the left figure of 4. The right figure of 4 shows the estimation of the model when the activation function is Fermi function. A example of prediction of the FNT model using the Fermi activation function for the Internet traffic data set is illustrated in Figure 5. We can see in Figure 4 the FNT estimation with the Fermi activation and the Gaussian activation function are very close. The Fermi function is not parametric, then the PSO algorithm is used only to estimate the weights in the tree. As a consequence the FNT model with Fermi function is faster in the training process than the FNT with exponential parametric function. The accuracy for the Internet Traffic data using the FNT technique and Fermi activation function was better than when we used the Gaussian activation function. Table 1 shows a comparison of the accuracy of the model using the different kinds of activation function in the nodes.

Function	Narma	Internet Traffic data
Gaussian	2.25646×10^{-3}	10.0642×10^{-5}
$\tanh(\cdot)$	3.47218×10^{-3}	10.2117×10^{-5}
Fermi function	2.37082×10^{-3}	8.67417×10^{-5}
Linear Fermi	3.23204×10^{-3}	9.95649×10^{-5}

Table 1: Accuracy of the FNT models for different kind of activation function in the tree nodes. The first and second column show the MSE obtained by the model, the error is presented using a scientific notation. First row refers the function used in the original FNT. The last three rows refer to the functions 9, 10 and 11.

5 Conclusions and future work

Ten years ago, the *Flexible Neural Tree (FNT)*, a specific type of Neural Network was presented. The method has proved to be a very powerful tool for time series processing and forecasting problems. The model uses meta-heuristic techniques for defining the topology of the tree and for finding "good" parameters in learning tasks. The FNT uses activation function with exponential form in the nodes. In this paper we analyze the performance of the model with another family of function in its nodes, we studied the hyperbolic tangent and the Fermi functions. We tested the performance in two widely used benchmark data: a simulated and

a real data set. The results are promising, specifically for the FNT model that uses Fermi function its the nodes. In future works, we will use statistical tests for comparing the different approaches presented here, as well as we will compare our results with other forecasting techniques.

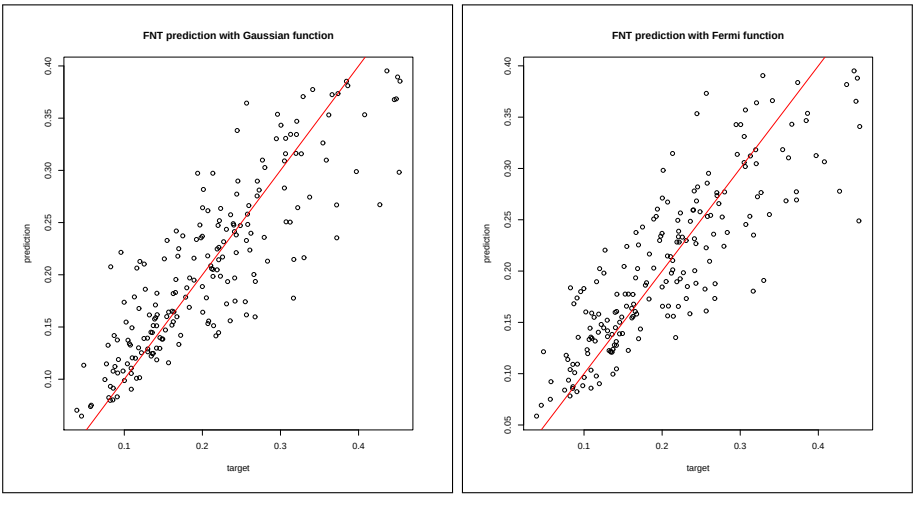

Fig. 2: Example of the FNT prediction for the NARMA data set. Both figures show the estimation of the last 80 time units of the test data set. The left figure shows the FNT prediction using the Gaussian activation function. The right figure illustrates the estimation of the model using the Fermi activation function. The red line corresponds the identity function.

Acknowledgments

This article has been elaborated in the framework of the project *New creative teams in priorities of scientific research*, reg. no. CZ.1.07/2.3.00/30.0055, supported by Operational Program Education for Competitiveness and co-financed by the European Social Fund and the state budget of the Czech Republic. Additionally, this work was partially supported by the Grant of SGS No. SP2014/110, VŠB - Technical University of Ostrava, Czech Republic, and was supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070) and by the Bio-Inspired Methods: research, development and knowledge transfer project, reg. no. CZ.1.07/2.3.00/20.0073 funded by Operational Programme Education for Competitiveness, co-financed by ESF and state budget of the Czech Republic.

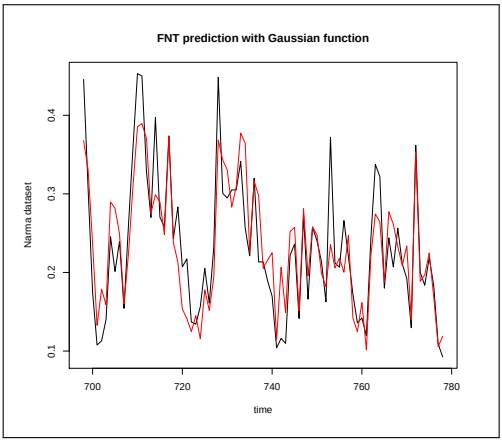

Fig. 3: FNT prediction using the Gaussian activation function on the NARMA data set. The estimation was realized on the last 200 time units of the test data. The red line is the prediction o the data and the black line is the target data.

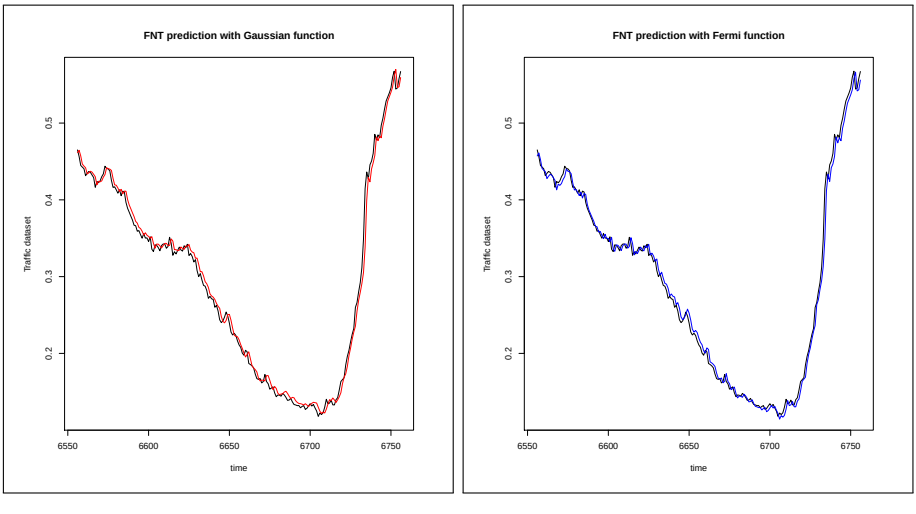

Fig. 4: FNT prediction on the Internet traffic dataset. Both figures show the estimation of the last 200 time units of the test data set. The left figure shows the prediction with the Gaussian activation function (red line). The right figure illustrates the estimation of the model (blue line) using the Fermi activation function. In both cases the black line is the target data.

References

1. Sebastián Basterrech, Colin Fyfe, and Gerardo Rubino. Self-Organizing Maps and Scale-Invariant Maps in Echo State Networks. In *Intelligent Systems Design and*

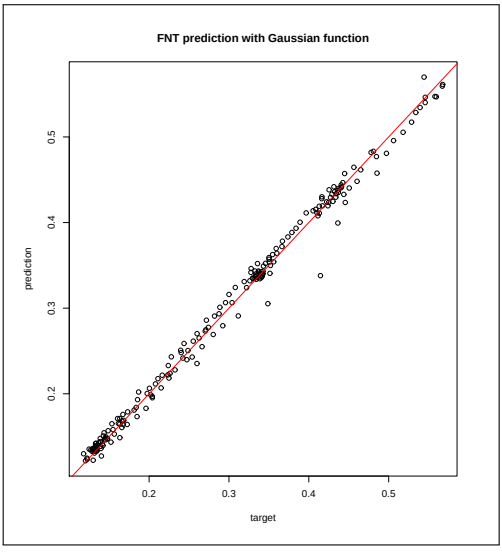

Fig. 5: FNT prediction using the Gaussian activation function on the Internet traffic data set. The estimation was realized on the last 80 time units of the test data set. The red line is the identity function.

Applications (ISDA), 2011 11th International Conference on, pages 94–99, nov. 2011.

2. Sebastián Basterrech and Gerardo Rubino. Echo State Queueing Network: a new Reservoir Computing learning tool. *IEEE Consumer Comunications & Networking Conference (CCNC'13)*, pages 118–123, January 2013.
3. Sebastián Basterrech and Václav Snášel. Initializing Reservoirs With Exhibitory And Inhibitory Signals Using Unsupervised Learning Techniques. In *International Symposium on Information and Communication Technology (SoICT)*, pages 53–60, Danang, Viet Nam, December 2013. ACM Digital Library.
4. Yuehui Chen, Ajith Abraham, and Bo Yang. Feature selection and classification using flexible neural tree. *Neurocomputing*, 70(1-3):305–313, 2006.
5. Yuehui Chen, Ajith Abraham, and Bo Yang. Hybrid flexible neural-tree-based intrusion detection systems. *International Journal of Intelligent Systems*, 22(4):337–352, 2007.
6. Yuehui Chen, Ajith Abraham, and Ju Yang. Feature selection and intrusion detection using hybrid flexible neural tree. In Jun Wang, Xiao-Feng Liao, and Zhang Yi, editors, *Advances in Neural Networks*, volume 3498 of *Lecture Notes in Computer Science*, pages 439–444. Springer Berlin Heidelberg, 2005.
7. Yuehui Chen, Lizhi Peng, and Ajith Abraham. Exchange rate forecasting using flexible neural trees. In Jun Wang, Zhang Yi, JacekM. Zurada, Bao-Liang Lu, and Hujun Yin, editors, *Advances in Neural Networks - ISNN 2006*, volume 3973 of *Lecture Notes in Computer Science*, pages 518–523. Springer Berlin Heidelberg, 2006.

8. Yuehui Chen, Bo Yang, and Ajith Abraham. Flexible neural trees ensemble for stock index modeling. *Neurocomputing*, 70(4-6):697–703, 2007.
9. Yuehui Chen, Bo Yang, and Jiwen Dong. Nonlinear System Modelling Via Optimal Design of Neural Trees. *International Journal of Neural Systems*, 14(02):125–137, 2004.
10. Yuehui Chen, Bo Yang, Jiwen Dong, and Ajith Abraham. Time-series forecasting using flexible neural tree model. *Inf. Sci.*, 174(3-4):219–235, August 2005.
11. Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Mach. Learn.*, 20(3):273–297, September 1995.
12. P. Cortez, M. Rio, M. Rocha, and P. Sousa. Multiscale Internet traffic forecasting using Neural Networks and time series methods. *Expert Systems*, 2012.
13. J. Kennedy and R. Eberhart. Particle Swarm Optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948, 1995.
14. Teuvo Kohonen. *Self-Organizing Maps*. Springer Series in Information Sciences, third edition, 2001.
15. John R. Koza, Forrest H. Bennett III, and Oscar Stiffelman. Genetic Programming as a Darwinian Invention Machine. In Riccardo Poli, Peter Nordin, WilliamB. Langdon, and TerenceC. Fogarty, editors, *Genetic Programming*, volume 1598 of *Lecture Notes in Computer Science*, pages 93–108. Springer Berlin Heidelberg, 1999.
16. Kenneth Price, Rainer M. Storn, and Jouni A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.
17. Rafal Salustowicz and Jürgen Schmidhuber. Probabilistic incremental program evolution. *Evolutionary Computation*, 5(2):123–141, 1997.
18. Yuhui Shi and R. Eberhart. A modified particle swarm optimizer. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pages 69–73, 1998.
19. Yuhui Shi and RussellC. Eberhart. Parameter Selection in Particle Swarm Optimization. In V.W. Porto, N. Saravanan, D. Waagen, and A.E. Eiben, editors, *Evolutionary Programming VII*, volume 1447 of *Lecture Notes in Computer Science*, pages 591–600. Springer Berlin Heidelberg, 1998.
20. Rainer Storn and Kenneth Price. Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.