Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC-PapersOnLine 49-25 (2016) 049–054

Weight Initialization Possibilities for Feedforward Neural Network with Linear Saturated Activation Functions

Petr Dolezel * **Pavel Skrabanek** ** **Lumir Gago** ***

* *University of Pardubice, Faculty of Electrical Engineering and Informatics, Pardubice, Czech Republic (e-mail: petr.dolezel@upce.cz).*
** *(e-mail: pavel.skrabanek@upce.cz)*
*** *(e-mail: lumir.gago@student.upce.cz)*

Abstract: Initial weight choice is an important aspect of the training mechanism for feedforward neural networks. This paper deals with a particular topology of a feedforward neural network, where symmetric linear saturated activation functions are used in a hidden layer. Training of such a topology is a tricky procedure, since the activation functions are not fully differentiable. Thus, a proper initialization method for that case is even more important, than dealing with neural networks with sigmoid activation functions. Therefore, several initialization possibilities are examined and tested here. As a result, particular initialization methods are recommended for application, according to the class of the task to be solved.

Keywords: artificial neural network, initialization, linear-saturated activation function, linearization.

1. INTRODUCTION

In the past few years, artificial neural networks have attracted great attention both from a theoretical point of view and for their various applications in pattern recognition, function approximation, data processing, dynamic system behavior prediction, etc.

Although many different topologies of neural networks have been introduced, a classical feedforward neural network still remains the most popular architecture for a majority of engineers. Clearly, a feedforward neural network has to be trained substantially to perform sufficiently - and many well-known training algorithms have been introduced so far. Let us mention at least the Backpropagation algorithm introduced by Rumelhart et al. (1986) as the first one; and Levenberg-Marquardt algorithm - see Kollias and Anastassiou (1989) - which is widely considered as the most effective one for batch training.

Generally, initialization of weights significantly affects the performance of the training regardless of used training algorithm. Since the neural networks are applied to different highly nonlinear and complex problems, various initialization methods provide fluctuating performances. This implies two points. Firstly, the study of new weight initialization methods is a living research field; and secondly, pure random weight initialization is still the most common method due to its simplicity and general use.

This paper deals with a special topology of a feedforward neural network introduced in Dolezel and Taufer (2012) and Dolezel and Skrabanek (2012). In mentioned references, authors used a neural network approach for continuous linearization of nonlinear dynamic systems. In brief, they implemented a symmetric linear saturated activation function for neurons in hidden layers, to achieve a piecewise-linear behavior of the resulting dynamic neural model. However, the authors did not explicitly deal with, or test, the training possibilities of this network. Clearly, this insufficiency was reduced later, and several approaches suitable to train such a topology were suggested in Mariska et al. (2013) or in Gago and Dolezel (2016). Nevertheless, no particular initialization technique in combination with the topology used in Dolezel and Taufer (2012) and Dolezel and Skrabanek (2012) has been considered, so far. Thus, our current work aims on detection of an initialization technique suitable for feedforward neural networks with symmetric linear saturated activation functions.

As a first engineering approach, several known initialization techniques, commonly used with neural networks containing sigmoid activation functions, are considered. To be specific, four approaches (random initialization, Nguyen and Widrow (1990) method, Halawa (2014) method and Sodhi and Chandra (2013) method) are applied to the set of defined benchmark problems and the performances are then statistically evaluated.

The paper is structured as follows: section 2 scrictly formulates the problem; conditions of the experiments are defined in section 3 while results are shown in section 4. A text in section 5 concludes all the achieved observations.

2. PROBLEM FORMULATION

This contribution aims at practical features of an approximation procedure, using a feedforward neural network introduced in Dolezel and Taufer (2012) and Dolezel and Skrabanek (2012), and their extensions or modifications. Hornik et al. (1989) proved that a standard multilayer feedforward neural network with one hidden layer is capa-

2405-8963
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2016.12.009

ble of approximating any real measurable function to any desired degree of accuracy. Neurons in the hidden layer contain a squashing activation function, while the output neuron contains a non-squashing activation function - see Hornik et al. (1989) for formal definition. For practical applications, a continuous, bounded and monotonic activation function is used for the neurons in the hidden layer. A continuous and monotonic activation function is then used in the output neuron - for some examples, see Haykin (1999), Nguyen et al. (2003). Neural network used in Dolezel and Taufer (2012) and Dolezel and Skrabanek (2012) satisfies Hornik's condition. However, it does not fulfil the condition of all the gradient-based training algorithms (all the components of the network have to be fully analytically differentiable).

The results published in Mariska et al. (2013) indicate that certain approximations of nondifferentiable parts of the used network are able to provide sufficient behavior of standard training algorithms such as Backpropagation algorithm or Levenberg-Marquardt algorithm. The aim of this paper is to test several known initialization techniques to even further improve the training results.

2.1 Piecewise-linear neural network

A neural network used in Dolezel and Taufer (2012) and Dolezel and Skrabanek (2012) is called a piecewise-linear neural network (PWLNN) and it transforms the input vector $\mathbf{x}$ into a scalar value y - see (1).

$$y = \sum_{i=1}^{H} v_i \phi(\mathbf{w}_i^{\mathrm{T}} \mathbf{x} + b_i) + v_0, \tag{1}$$

where H is the number of hidden neurons, $\mathbf{w}_i$ is the weight vector of the i^{th} neuron of the hidden layer, b_i is the bias weight of the i^{th} neuron, v_i is the weight of the output neuron which connects the i^{th} hidden unit to the output neuron and v_0 is the bias of the output neuron.

Only one function is considered in Dolezel and Taufer (2012) and Dolezel and Skrabanek (2012) as an activation function ϕ. It is called a symmetric linear saturated activation function and is defined as follows.

$$y_i = \begin{cases} 1 & \text{for} \quad y_{a,i} > 1 \\ y_{a,i} & \text{for} \quad -1 \leq y_{a,i} \leq 1 \\ -1 & \text{for} \quad y_{a,i} < 1 \end{cases}, \tag{2}$$

where y_i is the output from the activation function while $y_{a,i}$ is its input.

Clearly, the projection between the input vector $\mathbf{x}$ and the output scalar y indicates a piecewise-linear characteristic.

2.2 Tested initialization methods

As mentioned above, four approaches to initialization are examined in this paper. Although they are originally proposed to initialize neural networks with sigmoid activation functions, they can be applied to PWLNN, too.

Random initialization Random initialization provides uniformly distributed random values symmetric around a zero value. In other words, each weight value is picked from the interval $[-\delta; \delta]$.

Nguyen-Widrow method The Nguyen-Widrow method generates initial weight and bias values, so that the active regions of the neurons are distributed approximately evenly over the input space. Although it was introduced in 1990, it is probably still the most applied initialization method, except for the random initialization.

Halawa method This approach uses weight selections based on determination of the variability of the function to be approximated, within various fragments of its domain. However, the method provides weights only for some of the neurons. The rest of them have to be initialized using the Nguyen-Widrow method. See Halawa (2014) for detailed information.

Sodhi-Chandra method In 2013, Sartaj Singh Sodhi and Pravin Chandra published a simple initialization method. They proposed to initialize weights such, that the input layer to the hidden layer weights are set to random values in a manner that weights for distinct hidden nodes belong to distinct intervals.

2.3 Experiment procedure

The procedure of neural network design involves training and testing set acquisition, neural network training and pruning, and neural network validating.

The data sets for our experiments consist of four distinct points and are described in following sections.

Training of the neural network means using a set of observations to find optimal (in some sense) values of weights and biases of a trained neural network. For the purposes of this paper, the Levenberg-Marquardt training algorithm is used since it is broadly considered as the first choice for most of the applications (see Kollias and Anastassiou (1989) for detailed information).

Pruning (i.e. optimization of the topology) is not considered here, as the topologies for the experiments are invariant.

Two sets of experiments are performed - convergence speed during training and performance over a defined number of epochs.

Convergence speed In this set of experiments, PWLNN, initialized by all the mentioned approaches, are trained to achieve a defined goal, while the number of epochs is measured.

Performance Here, PWLNN, initialized by all the mentioned approaches, are trained over the defined number of epochs, while the performance of the nets at the end of the training procedure is observed.

3. CONDITIONS OF THE NUMERICAL EXPERIMENTS

3.1 Data for training

The experiments are performed with four datasets; the first one represents a continuous function, the second one a discontinuous function, the third one is a simulated first order time series, and the last one is a time series gained

as a response of the real dynamic system - twin rotor aerodynamic device. See Fig. 1 for some examples of the courses.

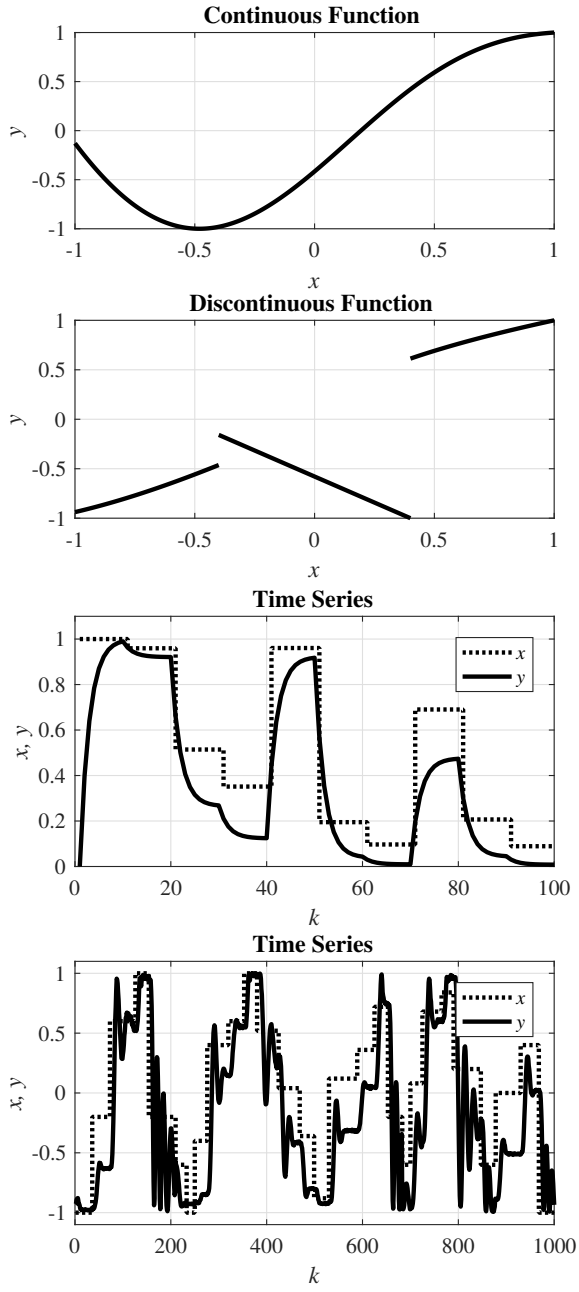

Fig. 1. Data sets (x - input to the system, y - output, k - discrete time)

A twin rotor aerodynamic system (see Fig. 2) is a significantly nonlinear device with two inputs (power of main rotor and power of tail rotor) and two outputs (vertical elevation and yaw motion). All quantities are pronounced as unified voltage signals (0-5V). For the aims of this contribution, all the quantities are transformed to the interval $[-1; 1]$ and yaw motion is locked. According to Dolezel et al. (2012), this device can be treated as a second order dynamic system.

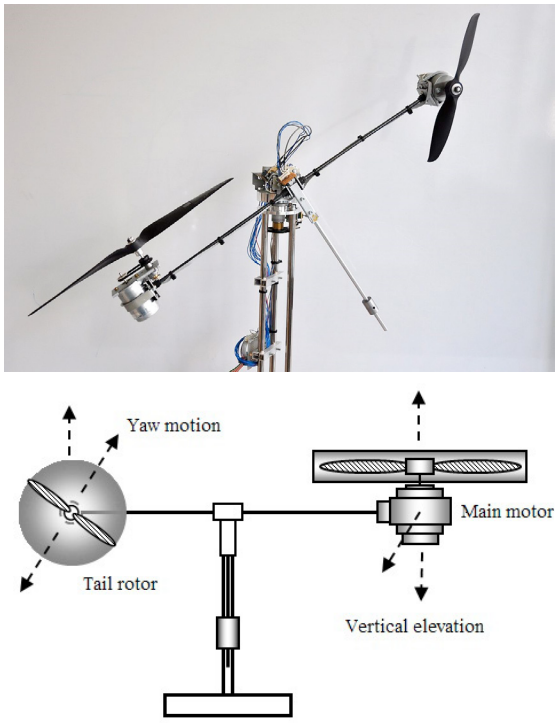

Fig. 2. Twin rotor aerodynamic system

The data from the datasets is divided into three subsets. The training set contains 70 % of the samples, while both testing and validation sets contain 15 %.

3.2 Used topologies

PWLNN according to section 2.1 are used for training. The exact topologies are shown in Fig. 3. The number of hidden neurons for the first, second and third dataset is equal to 10. For the last dataset, the number is set to 20 since the Halawa initialization method requires "far more" neurons in the hidden layer than inputs.

3.3 Parameters of the training algorithm

Levenberg-Marquardt algorithm is used for all the trainings, as mentioned above. The formula for iterative weights update, as a fact of common knowledge, is defined as follows.

$$\mathbf{v}(new) = \mathbf{v}(old) - (\mathbf{J}^{\mathrm{T}}\mathbf{J} + \mu\mathbf{I})^{-1}\mathbf{J}^{\mathrm{T}}\varepsilon, \tag{3}$$

where $\mathbf{v}$ is a vector of all the tunable values, $\mathbf{J}$ is a Jacobian matrix of derivatives of the elements from the vector ε by elements of the vector $\mathbf{v}$, $\mathbf{I}$ is an identity matrix and ε is a vector of differences between target value and output from a neural network. In addition, another important parameter is defined for this training algorithm - μ_Δ. The parameter μ is divided by this parameter if the new weights provide better performance than the old ones. Otherwise, μ is multiplied by μ_Δ. For our experiments, $\mu = 0.1$ and $\mu_\Delta = 10$.

3.4 Parameters of the initialization methods

According to the original proposals, the Nguyen-Widrow method, as well as Sodhi-Chandra method, have no tunable parameters. For a random initialization method, only

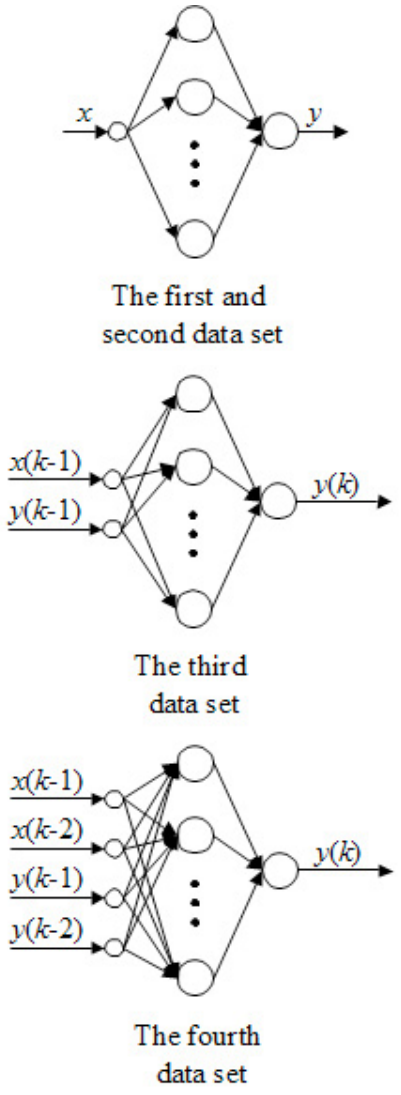

Fig. 3. Used topologies (a) - first and second dataset, (b) - trird dataset, (c) - fourth dataset

boundaries of the $[-\delta;\delta]$ interval have to be selected. For experiments performed in this contribution, δ is set empirically according to the variance of the values provided by other methods; $\delta = 5$.

On the other hand, the Halawa method requires determining if the number of the fragments, n, into which the input data space is divided. There are a number of recommendations mentioned in Halawa (2014) about how to do it. Thus, after a couple of trials, n is set to 4 for the first and second data set, $n = 9$ for the third data set and $n = 16$ for the last data set.

3.5 Parameters for the experiments

As mentioned above, in the "convergence speed" experiments, PWLNN are trained to achieve a defined goal, while the number of epochs is measured. The goal is defined as the particular value of a mean square error function computed over validation data, while the nets are trained using only training sets. The mean square error is defined as follows.

$$E_{val} = \frac{1}{N}\sum_{i=1}^{N}[o(i) - y(i)]^2, \tag{4}$$

where N is the number of the samples in the validation set, $o(i)$ is the i^{th} desired output and $y(i)$ is the actual output from the net.

The goals are set as follows - see Table 1. These values have been determined empirically after several tests. Clearly, they should be a challenge to achieve, while on the other hand, they must be feasible.

Table 1. Goals for convergence speed experiment

Data set	Goal
1	$3.0 \cdot 10^{-5}$
2	$3.0 \cdot 10^{-3}$
3	$5.0 \cdot 10^{-5}$
4	$1.5 \cdot 10^{-3}$

For the other set of experiments (i.e. "performance" experiments), 1000 epochs of training are performed and the final value of the error function (4) is analyzed.

Since all the initialization methods are stochastic, one hundred training experiments are performed for each scenario and the results are described using statistical quantities (mean, median, standard deviation, etc.) - see below.

4. RESULTS AND DISCUSSION

4.1 Convergence speed experiments

The results for this set of experiments are summarized in the following table (Table 2). The required goal is not met every time for some initialization methods. Thus, note that the statistical quantities are computed only for successful instances. These circumstances can lead to the misinterpretation of the results and must be taken into consideration.

For better illustration, the box graphs of mentioned data are shown in Fig. 4. The central marks of the box graphs are medians, the edges of the boxes are 25^{th} and 75^{th} percentiles, and the whiskers extend to the most extreme data points (except outliers).

4.2 Performance experiments

Table 3 summarizes the results of the second set of experiments. In this case, the performance is defined as the mean square error (4) and all the results are included in the statistical quantities. As in the previous case, the box graphs of mentioned data are shown in Fig. 5 (outliers are discarded).

Looking at the results, we can say, that using a more sophisticated initialization method is definitely worthwhile, since a simple random initialization provides the worst results in almost every experiment. Focusing on the Sodhi-Chandra method, the performance is unclear. It provides very good results for some experiments while it fails when applied to others. On the other hand, the Nguyen-Widrow method as well as the Halawa method provide high-quality and reliable results across all the experiments. Besides,

Table 2. Results for the first set of experiments

	Random				Nguyen-Widrow				Halawa				Sodhi-Chandra			
Dataset	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Number of exps.	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
Goals met	4	69	98	100	100	100	100	100	100	95	100	98	39	68	100	100
Mean	24	36	12	7	7	18	7	5	20	16	7	4	63	72	6	9
Median	17	22	10	7	7	16	7	5	19	15	7	4	63	52	6	7
Stand. deviation	21	67	9	2	2	12	2	2	6	6	2	2	34	82	1	14
Minimum	9	8	5	4	5	8	4	2	10	8	3	1	12	22	4	4
Maximum	inf	inf	inf	11	14	114	23	12	42	inf	15	inf	inf	inf	8	144

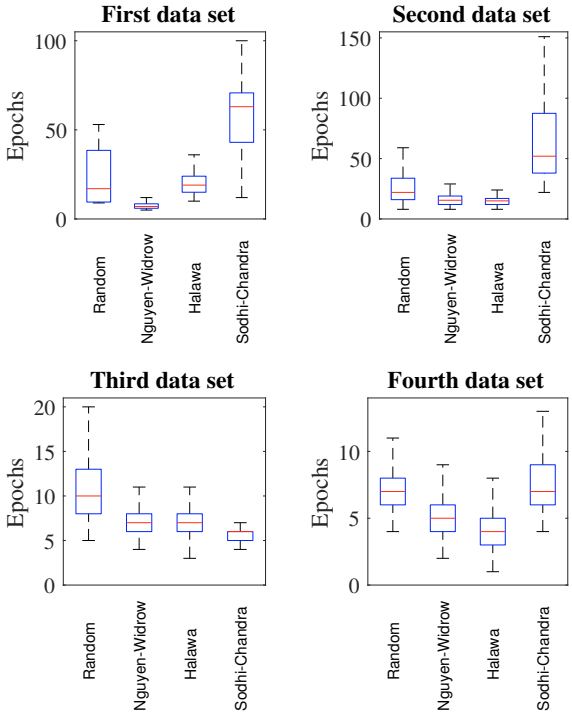

Fig. 4. Box graphs for the first set of experiments

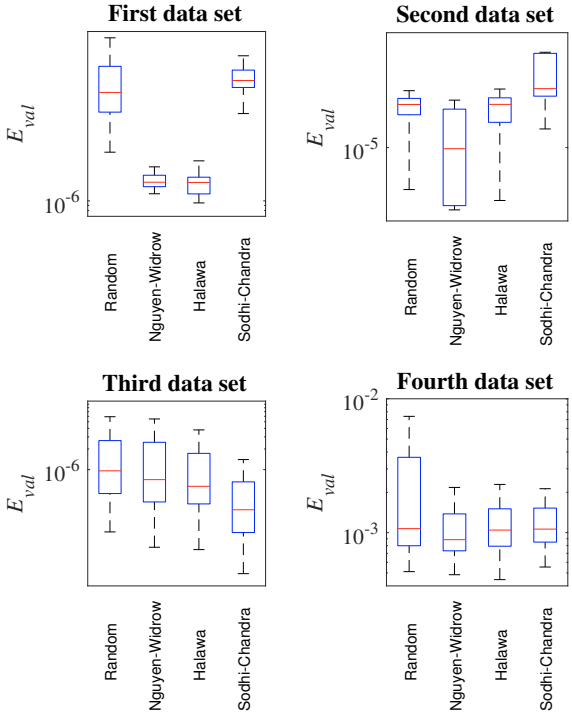

Fig. 5. Box graphs for the second set of experiments

note that the differences in performance quality are much bigger for approximations of static and highly nonlinear systems than for dynamic systems.

5. CONCLUSION

In this contribution, four initialization methods are tested and analyzed. They are applied to piecewise-linear neural network training using four different data sets. Statistical data gained from 3 200 training experiments in aggregate, indicate, that it is worthwhile to use a more sophisticated initialization method than s simple random initialization in the most cases. In a matter of fact, the Nguyen-Widrow method together with the Halawa method manifest the most reliable behavior. From those two methods, we recommend using Nguyen-Widrow method. It is computationally simpler and, considering the first set of 6 experiments, it did not even once stall in the shallow local minimum.

ACKNOWLEDGEMENTS

The work has been supported by the funds of the IGA, University of Pardubice, Czech Republic, project number SGS_2016_019. This support is very gratefully acknowledged.

REFERENCES

Dolezel, P., Havlicek, L., and Mares, J. (2012). Piecewise-linear neural model for helicopter elevation control. *International Journal of Control Science and Engineering*, 2(3), 42–46.

Dolezel, P. and Skrabanek, P. (2012). Piecewise-linear neural network - possible tool for nonlinear process control. 245–250.

Dolezel, P. and Taufer, I. (2012). Piecewise-linear artificial neural networks for pid controller tuning. *Acta Montanistica Slovaca*, 17(3), 224–233.

Gago, L. and Dolezel, P. (2016). Piecewise-linear neural networks training using gradient descent approach. In *ARTEP 2016*, 1–7.

Halawa, K. (2014). A new multilayer perceptron initialisation method with selection of weights on the basis of the function variability. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8467 LNAI(PART 1), 47–58.

Table 3. Results for the second set of experiments

Dataset	1	2	3	4
Number of exps	100	100	100	100
Epochs	10^3	10^3	10^3	10^3
Initialization	Random			
Mean	$2.65 \cdot 10^{-5}$	$1.10 \cdot 10^{-3}$	$2.27 \cdot 10^{-6}$	$3.40 \cdot 10^{-3}$
Median	$1.22 \cdot 10^{-5}$	$5.79 \cdot 10^{-4}$	$9.58 \cdot 10^{-7}$	$1.10 \cdot 10^{-3}$
Stand. deviation	$4.93 \cdot 10^{-5}$	$2.60 \cdot 10^{-3}$	$3.67 \cdot 10^{-6}$	$8.16 \cdot 10^{-3}$
Minimum	$3.08 \cdot 10^{-6}$	$1.92 \cdot 10^{-7}$	$1.22 \cdot 10^{-7}$	$5.11 \cdot 10^{-4}$
Maximum	$3.33 \cdot 10^{-4}$	$2.47 \cdot 10^{-2}$	$3.21 \cdot 10^{-5}$	$6.13 \cdot 10^{-2}$
Initialization	Nguyen-Widrow			
Mean	$1.59 \cdot 10^{-6}$	$1.44 \cdot 10^{-4}$	$1.43 \cdot 10^{-6}$	$1.40 \cdot 10^{-3}$
Median	$1.53 \cdot 10^{-6}$	$9.01 \cdot 10^{-6}$	$7.12 \cdot 10^{-7}$	$8.86 \cdot 10^{-4}$
Stand. deviation	$2.38 \cdot 10^{-7}$	$2.16 \cdot 10^{-4}$	$1.51 \cdot 10^{-6}$	$1.20 \cdot 10^{-3}$
Minimum	$1.17 \cdot 10^{-6}$	$2.84 \cdot 10^{-8}$	$7.33 \cdot 10^{-8}$	$4.84 \cdot 10^{-4}$
Maximum	$2.19 \cdot 10^{-6}$	$8.72 \cdot 10^{-4}$	$7.20 \cdot 10^{-6}$	$4.29 \cdot 10^{-3}$
Initialization	Halawa			
Mean	$1.60 \cdot 10^{-6}$	$6.86 \cdot 10^{-4}$	$1.16 \cdot 10^{-6}$	$1.41 \cdot 10^{-3}$
Median	$1.53 \cdot 10^{-6}$	$5.84 \cdot 10^{-4}$	$5.69 \cdot 10^{-7}$	$1.04 \cdot 10^{-3}$
Stand. deviation	$5.72 \cdot 10^{-7}$	$5.85 \cdot 10^{-4}$	$1.20 \cdot 10^{-6}$	$1.11 \cdot 10^{-3}$
Minimum	$9.54 \cdot 10^{-7}$	$6.88 \cdot 10^{-8}$	$6.78 \cdot 10^{-8}$	$4.45 \cdot 10^{-4}$
Maximum	$3.27 \cdot 10^{-6}$	$2.49 \cdot 10^{-3}$	$4.19 \cdot 10^{-6}$	$5.08 \cdot 10^{-3}$
Initialization	Sodhi-Chandra			
Mean	$1.89 \cdot 10^{-5}$	$2.49 \cdot 10^{-2}$	$6.13 \cdot 10^{-7}$	$1.57 \cdot 10^{-3}$
Median	$1.61 \cdot 10^{-5}$	$2.57 \cdot 10^{-3}$	$2.58 \cdot 10^{-7}$	$1.06 \cdot 10^{-3}$
Stand. deviation	$9.19 \cdot 10^{-6}$	$3.39 \cdot 10^{-2}$	$9.89 \cdot 10^{-7}$	$1.22 \cdot 10^{-3}$
Minimum	$7.51 \cdot 10^{-6}$	$5.78 \cdot 10^{-5}$	$3.01 \cdot 10^{-8}$	$5.53 \cdot 10^{-4}$
Maximum	$5.72 \cdot 10^{-5}$	$7.92 \cdot 10^{-2}$	$5.97 \cdot 10^{-6}$	$5.14 \cdot 10^{-3}$

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation*. Prentice Hall. ISBN: 0132733501.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5), 359–366.

Kollias, S. and Anastassiou, D. (1989). An adaptive least squares algorithm for the efficient training of artificial neural networks. *IEEE Transactions on Circuits and Systems*, 36(8), 1092–1101.

Mariska, M., Dolezel, P., and Havlicek, L. (2013). Training of the piecewise linear neural network used for process control. *Cybernetic Letters*, 11(1), 1–6.

Nguyen, D. and Widrow, B. (1990). Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. 21–26.

Nguyen, H., Prasad, N., and Walker, C. (2003). *A First Course in Fuzzy and Neural Control*. Chapman and Hall/CRC. ISBN: 1584882441.

Rumelhart, D., Hinton, G., and Williams, R. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088), 533–536.

Sodhi, S.S. and Chandra, P. (2013). Interval based weight initialization method for sigmoidal feedforward artificial neural networks. 19–25.