Vladimír STENCHLÁK[1], Ivan ZAJAČKO[2], Ivan KURIC[3]

WPŁYW WAG I PROGÓW NA WYDAJNOŚĆ NEURONÓW

Streszczenie: Artykuł dotyczy problematyki aproksymacji funkcji przez sieć neuronową. Pokazano, jak każdy parametr w warstwie sieci neuronowej wpływa na sygnał wyjściowy każdego neuronu w tej warstwie. Wiedzę z tego artykułu można wykorzystać na przykład w uniwersalnym twierdzeniu o aproksymacji.

Słowa kluczowe: sieci neuronowe, aproksymacja, sztuczna inteligencja, dopasowanie krzywki

IMPACT OF WEIGHTS AND BIASES ON THE OUTPUT OF NEURONS

Summary: This article deals with problematics of function approximation by neural network. There is shown how every parameter in layer of the neural network affects output signal of every neuron in that layer. These knowledge from this article can be used for example in universal approximation theorem.

Keywords: neural networks, approximation, artificial intelligence, curve fitting

Introduction

The usage of the neural networks has expanded sharply nowadays. Neural networks are way how to create dynamic systems, which behavior depends on parameters like weights and biases. There is no complete methodology of designing structures of these neural networks and the methodology of choosing parameters for training processes. There are a many ways how to optimize the behavior of the neural network using the best optimization method for the specific problem, which is our neural network solving. For example the commonly used method is called „Backpropagation“ and it has many parameters which affect the final result of the training process. Understand how single neurons in neural network work is the basic prerequisite to design the most efficient structure with specific parameters. At least when neural network is solving approximation of some specified function.

---

[1] inž., Department of Automation and Production Systems, Faculty of Mechanical Engineering, University in Žilina, e-mail: vladimir.stenchlak@fstroj.uniza.sk
[2] dr. inž., Department of Automation and Production Systems, Faculty of Mechanical Engineering, University in Žilina, e-mail: ivan.zajacko@fstroj.uniza.sk
[3] prof. dr. hab. inž. University of Bielsko-Biala, Faculty of Mechanical Engineering and Computer Science, Department of Industrial Engineering: kuric.ivan@gmail.com

1. Artificial neuron and its calculation in network

Artificial model of neuron is based on biological neuron. This model executes two single calculations. The first one is usually called „weighted input“. In this calculation we are feeding the neuron with n inputs. We can notate those inputs x_i. The significance of every input x_i is expressed by their weights w_i [3]. Every input has his own weight. In biology those weights w_i are also called „synaptic connections“, the output y is called „axon“ and the inputs are called „dendrites“. The weighted input can be expressed as shown below

$$in = \sum_{i=1}^{n} w_i x_i \tag{1}$$

Mathematical model of artificial neuron and the weighted input is described in figure 1. Every neuron is specific by his activation function. Activation functions are different in each layer [3].

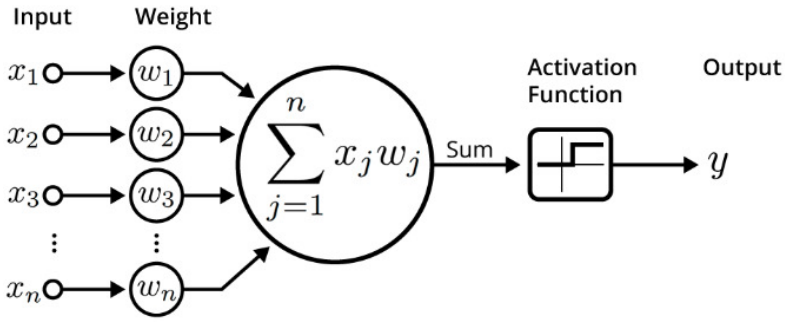

Figure 1. Mathematical model of artificial neuron[1]

The second calculation which neuron does is the transfer of weighted input via activation function. The output value of the activation function is also the final value of the neuron´s signal and it can be expressed as shown in equation 2 [3].

$$y = a(in) \tag{2}$$

$$a(in) = \frac{1}{1+\exp(-in)} \tag{3}$$

$$a(in) = \frac{\exp(in) - \exp(-in)}{\exp(in) + \exp(-in)} \tag{4}$$

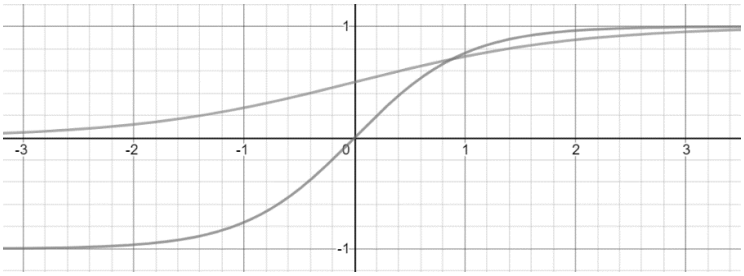

Figure 2. Hyperbolic tangent and sigmoidal activation function

There are several types of activation functions which are giving good results in final. Often you can see activations like logical sigmoid (3) hyperbolic tangent (4). Main advantage of hyperbolic tangent is the first derivation of the function [4]. In comparison with logical sigmoid it has stronger gradient which can increase effectiveness of the learning optimization method [4]. Neural networks can solve two types of tasks in general – classification and regression [5]. We will look closely on regression in this article. We will see how neural network can easily approximate any function. We need to understand weight and bias parameters and their meaning firstly so we can understand fully a simple approximation by feed forward neural network [2].

2. Changing values of weights and biases

In this part of article we will take a look how bias and weight parameters are affecting the final output of the neuron. The activation function we used in this example is logical sigmoid. We will increase and decrease values of those parameters and see how simple neuron will behave.

2.1 Bias parameter and its meaning

As we can see on the figure 3 - bias parameter is offsetting the inactive and active state output of the neuron. It means that if you have bias with high value and if you want to get simple neuron in an active state (so his output value is 1), you need greater input values. You need to have greater weighted input in this neuron. If you have bias with negative big number it means you don´t have to feed big input values in that neuron. Output state of this neuron will be active even when the weighted input is also negative [2].

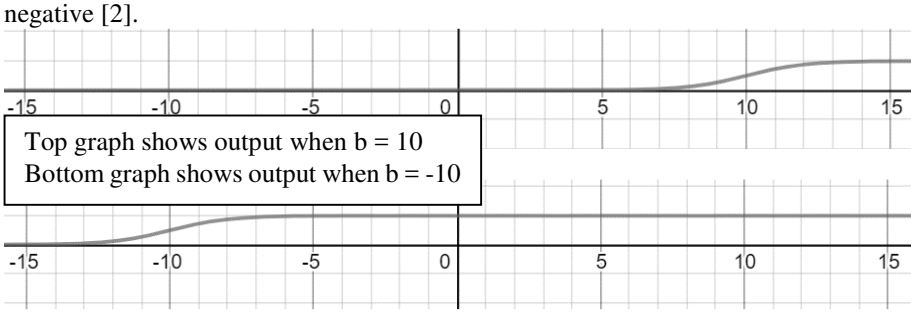

Figure 3. Impact of bias parameter on the final output from neuron

2.2 Weight parameter and its meaning

We can see on the fig. 4 that the size of weight parameters is affecting compression of the activation function and the polarity of this parameters inverts this output according to vertical axis of the graph. You can compare fig. 4 with fig. 2 and see the difference. The size of the weight is affecting the steepness of transition. The greater the weight parameter is, the bigger steepness of transition we have. The meaning of weight parameter in artificial neural networks is that the greater value of weight parameter between two neurons is, the more they affecting each other [2]. For example

when we have neuron *n.1*, and we are feeding the input of this neuron by output from neuron *n.2* and the weight of the connection is 0. The input has no effect to this neuron, because $w_i x_i = 0 x_i = 0$.

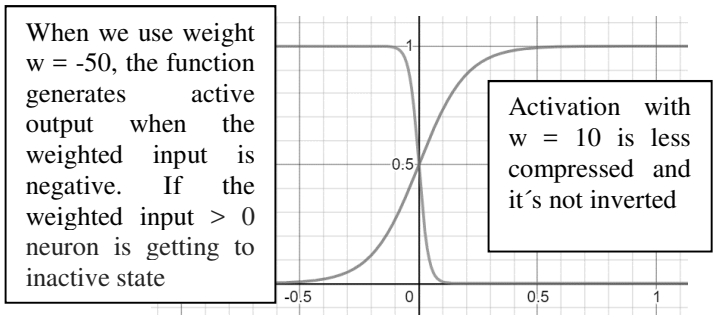

Figure 4. Impact of weight parameter on the output from neuron

3. Approximation using simple neural network

We can approximate any function what we want and the accuracy of the approximation depends on the number of neurons in network. What we do is that we are summing outputs from two neurons and that is how we get tower function from each 2 neurons [2].

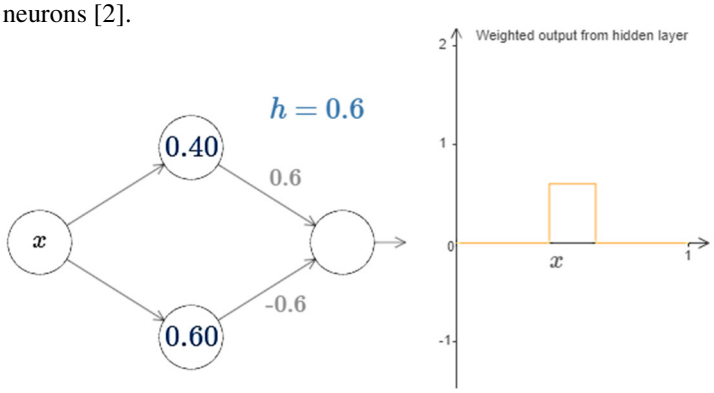

Figure 5. Simple tower function [2]

We are using neural networks to create many of these tower functions so they can approximate any function. Learning algorithm is setting up parameters like biases and weights so the output from the network is corresponding with our targets with minimal error [2].

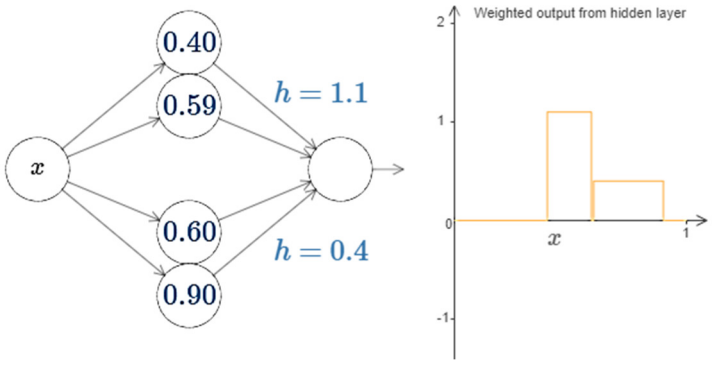

Figure 6. Tower functions from 4 neurons [2]

4. Practical use

We created training data for neural network. Created data set is simulating measured values. Input to the network is angle in radians. The dependence between inputs and outputs is given by the function (5). We will use neural network to approximate this function. We can say that the neural network will find the dependencies between input and output data without knowing the function (5). This knowledge will be acquired from training data set. Training data set consists of 1201 training samples.

$$output(input) = \cos(2input) + \sin(input^2) \tag{5}$$

You can see the graph of this function in fig. 7.

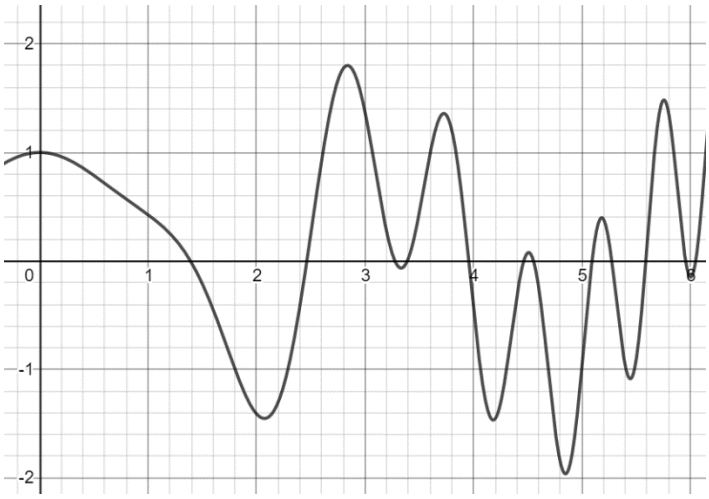

Figure 7. Graph of function (5)

We used neural network toolbox in MATLAB to create a neural network. We will proof that the accuracy off the network is dependent mainly on the number of the neurons in hidden layer. Firstly we have tested the learning algorithm for the neural

network with 5 neurons in hidden layer. We have used Levenberg-Marquardt backpropagation algorithm for training.

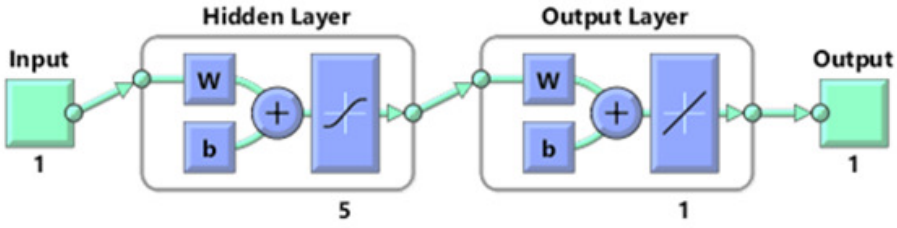

Figure 8. Feed forward neural network with 5 neurons in hidden layer

This network uses hyperbolic tangent function in hidden layer and linear function in output layer. Fig. 9 shows that the error of the network is very high. Network can approximate the function with 78% accuracy. The training was ended in iteration 26 and was terminated because of validation checks. Time of the training was 41 seconds (with visualizing the output from learning algorithm). In fig. 9 you can see that this network can approximate this function only in range of input data < 0,1; 1,7>.

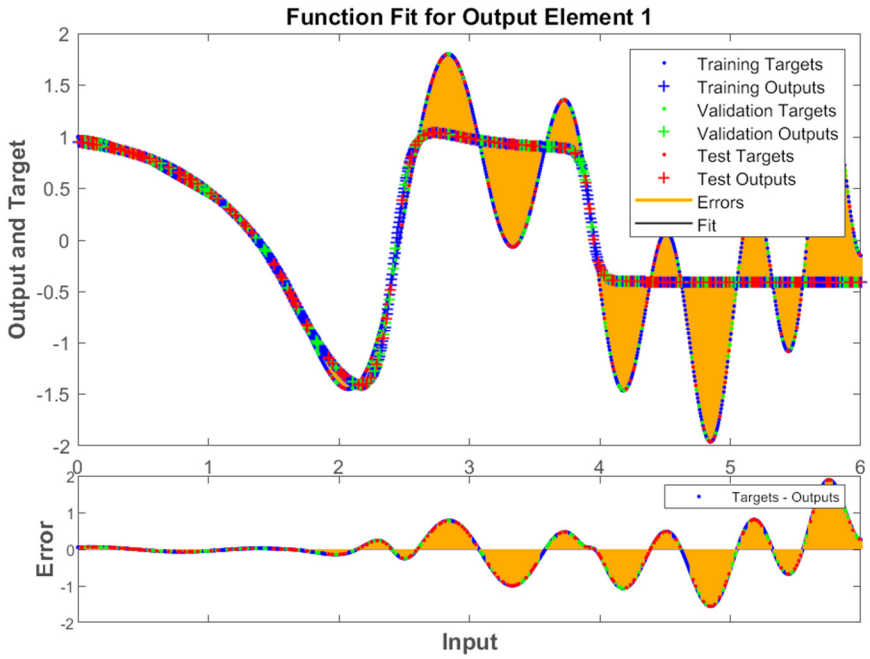

Figure 9. Approximation of the network with 5 neurons

We will use the same activation functions and number of layers of the neural network. Also we will use the Levenberg-Marquardt learning algorithm. We will change only number of the hidden neurons. Structure of the new neural network in MATLAB is shown below.

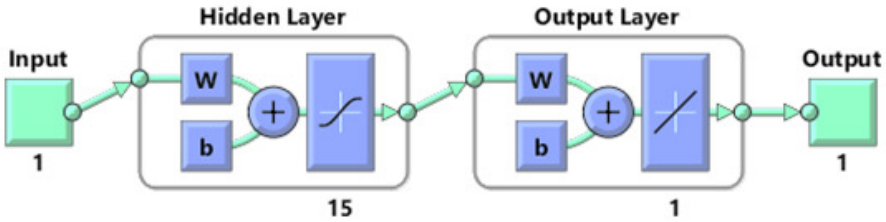

Figure 10. Feed forward neural network with 15 neurons in hidden layer

The learning of this network took 227 iterations and lasted for 5 minutes and 5 seconds (with visualizing every iteration process). We can see that training of bigger network is time-consuming and it is caused by the amount of weight and bias parameters. The more of these parameters you´ve got in network the more calculations you need (calculations to set up weight and biases to the correct values). You can see on Fig.11 that this bigger neural network can approximate very well in comparison with the first network. Accuracy of this network is 99%. You can see the accuracy in this regression plot on the fig.12. In this figure you can see the regression of the actual output from network in accordance to the desired output from training data set. Training data was divided in 3 groups – training (70%), validation(15%) and test data set (15%).

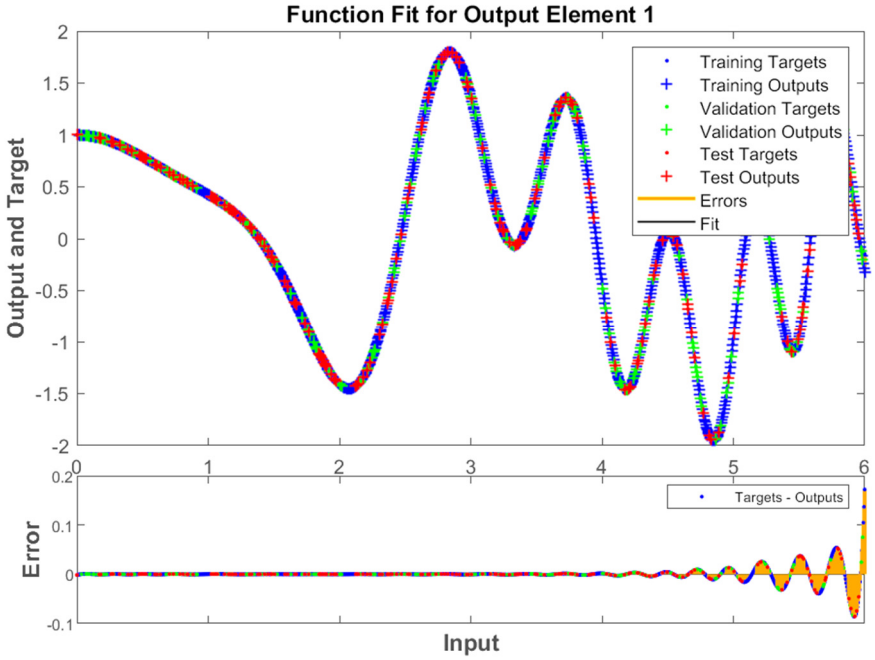

Figure 11. Approximation of the network with 15 neurons in hidden layer

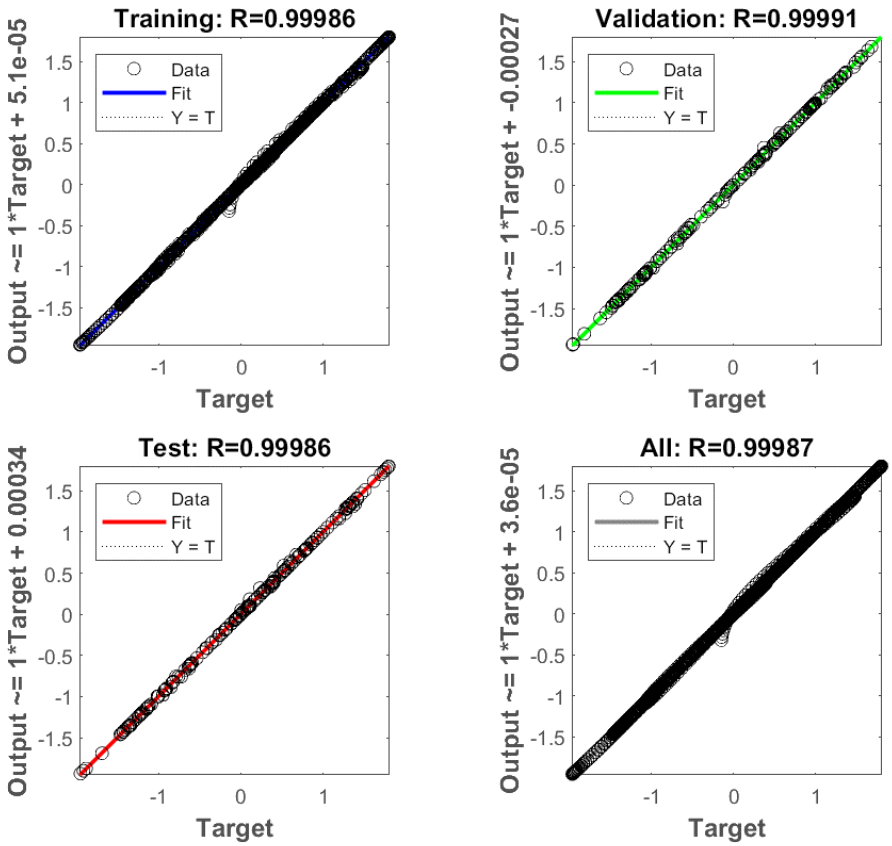

Figure 12. Regression plot of the network with 15 neurons in hidden layer

5. Conclusion

In this paper we had a close look at the weight and bias parameters of the neural network and their meaning. By understanding fully of those two parameters we have used this knowledge to prove that every function can be approximated by a simple feed forward neural network. We compared accuracy of two neural networks with different numbers of neurons in hidden layers in MATLAB. Experiment proved that increasing number of neurons in the hidden layer increases the accuracy of the network but also increases the training time of the neural network.

ACKNOWLEDGEMENT

This work was supported by the Slovak Research and Development Agency under the contract No. APVV-16-0283.

REFERENCES

1. SESTILLI C.: Deep Learning: Going Deeper toward Meaningful Patterns in Complex Data: *https://insights.sei.cmu.edu/sei_blog/2018/02/deep-learning-going-deeper-toward-meaningful-patterns-in-complex-data.html*, 15.10.2019.
2. NIELSEN M.: A visual proof that neural nets can compute any function: *http://neuralnetworksanddeeplearning.com/chap4.html*, 10.10.2019.
3. SINČÁK, Peter, 1996. Neurónové siete: inžiniersky prístup. Košice: Elfa, 1996. ISBN 808878638X.
4. SHARMA, Avinash. Understanding Activation Functions in Neural Networks: *https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0*, 19.10.2019.
5. BROWNLEE, Jason. Difference Between Classification and Regression in Machine Learning: *https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/*