Dynamic Modification of Activation Function using the Backpropagation Algorithm in the Artificial Neural Networks

Marina Adriana Mercioni[1], Alexandru Tiron[2], Stefan Holban[3]
Department of Computer Science
Politehnica University Timisoara, Timisoara, Romania

***Abstract*—The paper proposes the dynamic modification of the activation function in a learning technique, more exactly backpropagation algorithm. The modification consists in changing slope of sigmoid function for activation function according to increase or decrease the error in an epoch of learning. The study was done using the Waikato Environment for Knowledge Analysis (WEKA) platform to complete adding this feature in Multilayer Perceptron class. This study aims the dynamic modification of activation function has changed to relative gradient error, also neural networks with hidden layers have not used for it.**

***Keywords*—Artificial neural networks; activation function; sigmoid function; WEKA; multilayer perceptron; instance; classifier; gradient; rate metric; performance; dynamic modification**

I. Introduction

The behavior of artificial neural networks has been an area that has been extensively studied over time, providing consistent results. It studied the computational power of artificial neural networks by activating recurring sigmoid. [1] This aspect occurs due to the diversity of systems which are described by datasets used for exercising them.

In applications, the activation function and gradient selection are impacted directly by network convergence. [2].

In this context, the universal behavior of Turing neural networks has been studied for a specific activation function which was called linear function by form:

$$f(x) = \begin{cases} 0, & x \le 0 \\ x, & 0 < x < 0 \\ 1, & x \ge 1 \end{cases} \qquad (1)$$

An activation function in computing of neural artificial networks plays an important role [3, 4].

Neural networks consist of an important method for analysis of activation function because of their capability to deal with data sets which change. For example, in Multilayer Perceptron (MLP), the most common system, neurons are organized in layers [5].

One of the activation functions studied on large scale in the literature is the sigmoid function:

$$g(x) = \frac{1}{1+e^{-x}} \qquad (2)$$

These functions represent the main activation functions, being currently the most used by neural networks, representing from this reason a standard in building of an architecture of Neural Network type [6,7].

The activation functions that have been proposed along the time were analyzed in the light of the applicability of the backpropagation algorithm. It has noted that a function that enables differentiated rate calculated by the neural network to be differentiated so and the error of function becomes differentiated [8].

The main purpose of activation function consists in the scalation of outputs of the neurons in neural networks and the introduction a non-linear relationship between the input and output of the neuron [9].

By the other hand, the sigmoid function is used usually for hidden layers because it combines the linear, curvilinear and constant behavior depends by the input value [10].

Also, it has demonstrated that sigmoid function is not efficiently for a single hidden unit, but when more hidden units are involved, it becomes more usefully [11].

Our approach in this paper consists in dynamic modifying of activation function using backpropagation algorithm for training an artificial neural network.

II. Backpropagation Algorithm

It is the most common algorithm used to train neural networks regardless of the nature of the data set used. The Neural network architecture is determined by repeat trials, the goal is to obtain the best possible classification of data set used in this context [8, 12].

Backpropagation algorithm is a supervised learning algorithm and his purpose is to correct the error. It compares the computed output value with the real value, and it tries to change the weights through the calculated error, and in analog manner until the size of obtained error becomes smaller than the error obtained in the first round. The backpropagation network composition is detailed in Fig. 1 [13].

As learning strategy, the backpropagation algorithm proved to be effectively in that it ensures a classification whose accuracy is generally satisfactory [14, 15].

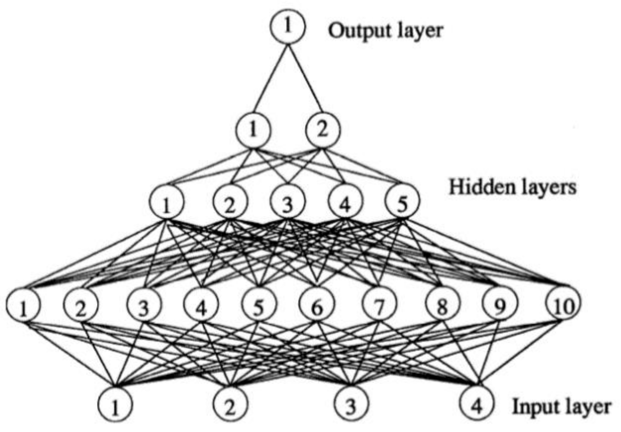


Fig. 1. Backpropagation Network [12].

The main disadvantage of this learning technique is the fact that they are required repeated attempts to establish network architecture, number of hidden layers and number of neurons in each hidden layer, in the context in which training requires a lot of resources as memory and runtime, as in Fig. 2.

The classical backpropagation algorithm is divided in two phases: the first phase consists in the propagation of useful information from the input layer toward the exit, then spread in the reverse direction of errors, and the second phase consists in updating the weights.

- In the first phase, the input data propagate through the network to generate the output values, then calculate the error and propagate back toward the hidden layers and toward the entry to generate the differences between the obtained values and actual values for each neuron.
- In the second phase, for each weight calculates the gradient of error using generated differences in the first step. This weight is updated in the opposite direction to the error by a coefficient of learning.

Backpropagation algorithm has been focused on theorems which have guaranteed „universal approximation", a property of neural networks [16].

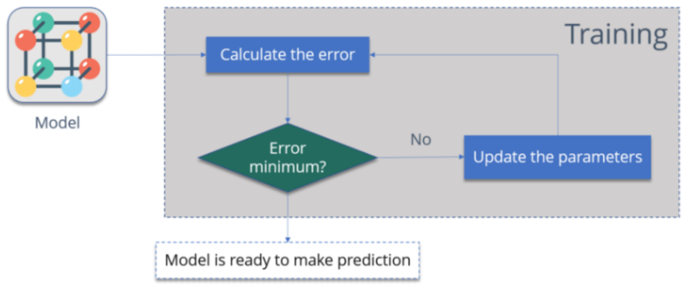


Fig. 2. Backpropagation Algorithm (Source: https://d1jnx9ba8s6j9r. cloudfront.net/blog/wp-content/uploads/2017/09/Training-A-Neural-Network-Backpropagation-Edureka-768x314.png).

III. Backpropagation Algorithm with Dynamic Activation Function

The main element of an artificial neural network is artificial neuron, which can be represented by a sum and a function, so the network will be composed of several interconnected functions. More clearly, these functions are the filters through which the information passes. And depending on how we want to learn neural network, these functions have specific characteristics to the chosen purpose [17,19].

One of the most important characteristics of an artificial neural network is the activation function [7].

One of the most important activation functions is sigmoid function. From point of view of evolution, this function has developed from Heaveside function as it is showed in Fig. 3.

The sigmoid function can be calculated as it is showed in Fig. 4; respectively its curve can be seen.

Among the most commonly available types of activation functions we mention them in Fig. 5:

Whatever type of activation function and its characteristics, it remains unchanged during the training process.

We consider that this aspect represents a serious restriction. How learning is a process of dynamic change of weights which characterize numerically the connections between neurons of network, it is normally that this learning process to be reflected in the dynamic activation function by dynamic changing of its characteristics [20].

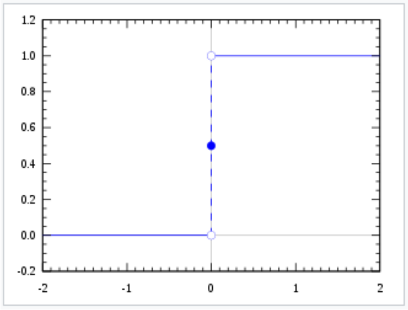


Fig. 3. Heaviside Function (Source: https://upload. wikimedia.org /wikipedia/commons/thumb/d/d9/Dirac_distribution_CDF.svg/1280px-Dirac_distribution_CDF.svg.png).

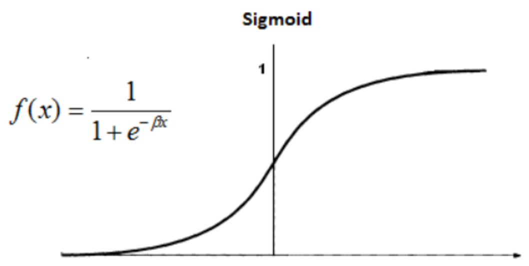
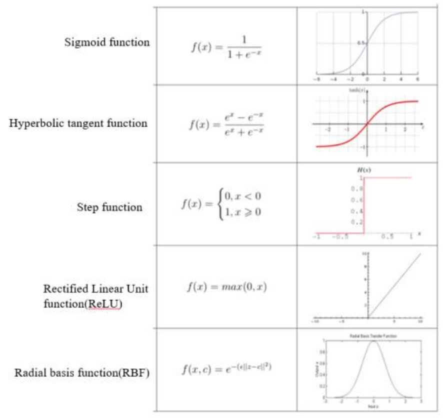


Fig. 4. Sigmoid function (Source: https://cdn-images-1.medium.com/max /1200/1*8SJcWjxz8j7YtY6K-DWxKw.png).



Sigmoid function	$f(x)=\frac{1}{1+e^{-x}}$	
Hyperbolic tangent function	$f(x)=\frac{e^x-e^{-x}}{e^x+e^{-x}}$	
Step function	$f(x)=\begin{cases}0, x<0\\1, x\geqslant 0\end{cases}$	
Rectified Linear Unit function(ReLU)	$f(x)=max(0,x)$	
Radial basis function(RBF)	$f(x,c)=e^{-(\lVert x-c\rVert^2)}$	

Fig. 5. Activation Function Types (Source: https://www.code-it.ro/wp-content/uploads/2018/11/Screenshot-2018-11-12-at-22.36.39.png).

In other words, learning is both dynamic modification of weights and also dynamic modification of activation function characteristics.

To achieve this goal we start from a classical activation function, more exactly sigmoid function (Fig. 4) wherein the β coefficient value is set dynamically during each epoch of learning simultaneously with the adjustment of the weights.

The initial value was considered to be 1 so that this value can change (increase or decrease) depending on the error level of present gradient in each of the learning epochs. The percentage change of this coefficient and its direction of adjustment (increase or decrease) is set by the user at the beginning of the learning process.

IV. Experimental Experiments

In this study we took in consideration an activation function by sigmoid type in the context of a neural network architecture without hidden layers. The study has been done considering:

- The β coefficient value of activation function decreases if the error increases during an epoch, respectively it increases. The direction can reverse (β coefficient value of activation function increases if the error increases during an epoch, respectively decreases, if the error decreases) is defined at the start of training.
- Percentage to scale this coefficient is set at the beginning of the training. It is defined by user.

In our experiments we consider the scenario:

- The β coefficient of activation function decreases if the error value increases during an epoch, the coefficient increases, if the error decreases.
- The percentage of modification for β coefficient was 10%, respectively 5% from actual value during training epoch.

To achieve the scenario above, Multilayer Perceptron Classifier has been changed through Weka 3.8.3 platform provided by adding a parameter that determines the direction of change of β value for activation function as the coefficient which is modified. The two parameters are:

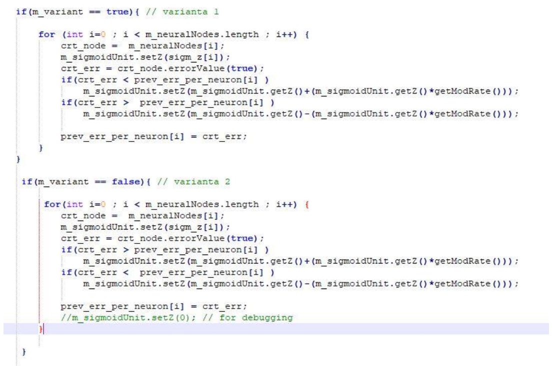
- modRate which specifies the percentage of modification for β coefficient.
- variant which defines the direction of change for β value (True - in case the value of β coefficient for activation function decreases, if the error increases during an epoch, respectively it increases, if the error decreases and False otherwise.

Added code can be viewed in Fig. 6:

From point of view of user, the modification sets initial settings for training process which can be visualized in Fig. 7 with the implicit values which are β=10% and modification direction variant=True.

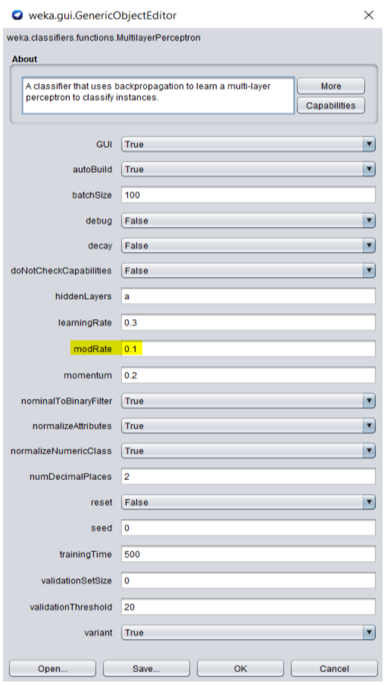
The study was performed taking into account a total of 10 sets of data provided by this platform as follows.

For these date sets we can observe the experimental results in the following tables and diagrams, which are centralized in Tables 1 and 2 and Fig. 8.



```
if(m_variant == true){ // varianta 1

    for (int i=0 ; i < m_neuralNodes.length ; i++) {
        crt_node =  m_neuralNodes[i];
        m_sigmoidUnit.setZ(sigm_z[i]);
        crt_err = crt_node.errorValue(true);
        if(crt_err < prev_err_per_neuron[i] )
            m_sigmoidUnit.setZ(m_sigmoidUnit.getZ()+(m_sigmoidUnit.getZ()*getModRate()));
        if(crt_err >  prev_err_per_neuron[i] )
            m_sigmoidUnit.setZ(m_sigmoidUnit.getZ()-(m_sigmoidUnit.getZ()*getModRate()));

        prev_err_per_neuron[i] = crt_err;
    }
}

 if(m_variant == false){ // varianta 2

     for(int i=0 ; i < m_neuralNodes.length ; i++) {
        crt_node =  m_neuralNodes[i];
        m_sigmoidUnit.setZ(sigm_z[i]);
        crt_err = crt_node.errorValue(true);
        if(crt_err > prev_err_per_neuron[i] )
            m_sigmoidUnit.setZ(m_sigmoidUnit.getZ()+(m_sigmoidUnit.getZ()*getModRate()));
        if(crt_err <  prev_err_per_neuron[i] )
            m_sigmoidUnit.setZ(m_sigmoidUnit.getZ()-(m_sigmoidUnit.getZ()*getModRate()));

        prev_err_per_neuron[i] = crt_err;
        //m_sigmoidUnit.setZ(0); // for debugging
    }

 }
```

Fig. 6. Changes Added to Multilayer Perceptron Class..



weka.gui.GenericObjectEditor

weka.classifiers.functions.MultilayerPerceptron

About

A classifier that uses backpropagation to learn a multi-layer perceptron to classify instances.

GUI	True
autoBuild	True
batchSize	100
debug	False
decay	False
doNotCheckCapabilities	False
hiddenLayers	a
learningRate	0.3
modRate	0.1
momentum	0.2
nominalToBinaryFilter	True
normalizeAttributes	True
normalizeNumericClass	True
numDecimalPlaces	2
reset	False
seed	0
trainingTime	500
validationSetSize	0
validationThreshold	20
variant	True

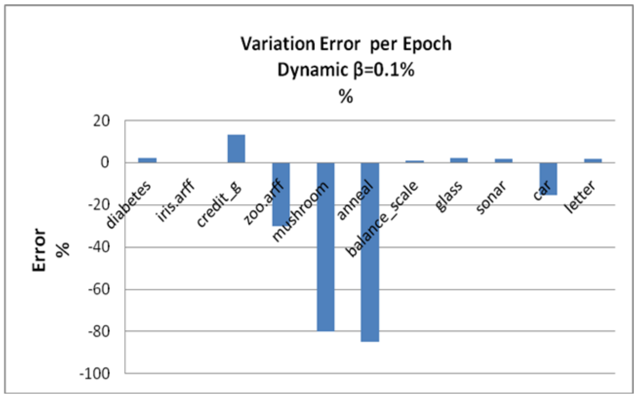
Fig. 7. β Values Set (modRate) and Modification Direction (Variant).

TABLE I. ERROR PER EPOCH

Data set	Variation Error per Epoch β=5% %	Variation Error per Epoch β=10% %
diabetes.arff	0.064609	2.172752
iris.arff	-0.4377358	-0.398922
credit_g.arff	27.853203	13.34323
zoo.arff	136.95652	-30
mushroom.arff	-25	-80
anneal.arff	-2.3637525	-85.01997
balance_scale.arff	0.3788186	1.084578
glass.arff	0.9818232	2.139382
sonar.arff	1.0156026	1.707687
car.arff	-1.2320329	-15.19507
letter.arff	2.1341222	1.921839

TABLE II. INSTANCE CLASSIFICATION

Data set	Dynamic Variation of Instances Classified Correctly β=5% %	Dynamic Variation of Instances Classified Correctly β=10% %
diabetes.arff	0.646161863	-1.777069
iris.arff	0	0
credit_g.arff	-0.201409869	-0.100705
zoo.arff	0	0
mushroom.arff	0	0
anneal.arff	1.283500796	4.426815
balance_scale.arff	-0.173611111	-0.173611
glass.arff	0.543490271	-1.086981
sonar.arff	0	0
car.arff	0	0
letter.arff	-0.891395307	-0.770117



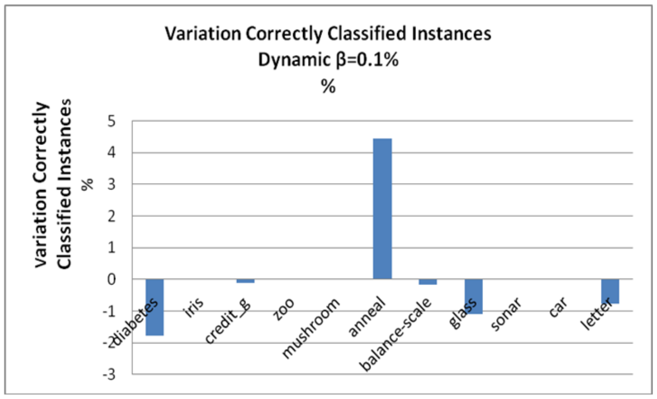


Fig. 8. Variation Error Per Epoch and Correctly Clasified Instances in Dynamic Mode.

V. Evaluation Impact of Training Process by Dynamic Introducing Activation Function

After analyzing the experimental results several observations can be made:

1) Introduction of the concept according to the dynamic activation leads to a change in drive definite process that quickly leads to a reduction of the error learning.

For example, this aspect is visible in the graph of error evolution in the case of using the iris.arff data set (β = 10%, variant = True) as it is showed in Fig. 9.

It can be seen in the first 50 epochs the modification of characteristics activation function leads to a search process directed toward a lower error in the training process. This aspect is present also in the case (β=5%, variant=True), it can be visualized in the graph in Fig. 10.

2) For a number of four data sets (zoo.arff, mushroom.arff, anneal.arff, carr.arff) reveals a definite reduction of the error obtained at the end of the 500 epochs of training.

The error of training is reduced as follows: 30% for zoo.arff, 80% for mushroom.arff, 85% for anneal.arff data set respectively with 15 for carr.arff compared to the situation where the training to perform a static activation function whose features remain constant in the training process.

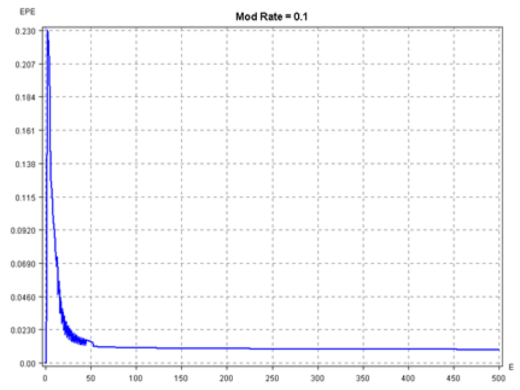


Fig. 9. The Graph of Error Variation/Epoch (β=10%, Variant=True).

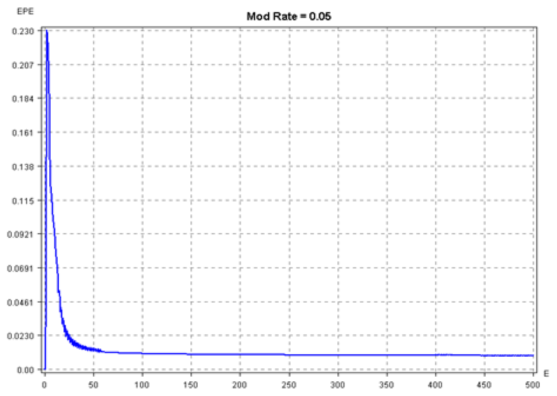


Fig. 10. The Graph of Error Variation / Epoch (β=5%, Variant=True).

The only data set where error of training increases is credit_g.arff, whose error increases by 15%. For other datasets variation of error is insignificant.

3) An interesting aspect is related to the following thing: for credit_g.arff data set, whose error increases, we can see that in testing phase it keeps the same number of instance correct classified as in the case of using an unchanged activation function. This aspect is remarked for zoo.arff data set, which for β=5% coefficient we have on the hand an increase of training error equal to 136.9%, but on the other hand when we pass to testing phase, doesn't appear any variation in the number of instances classified correctly. This aspect looks to be a result in fact the instances from data set for a static activation function aren't classified correctly, so in the case of activation function some data set are classified correctly meanwhile other instances are incorrectly classified.

4) We can see that significant differences occur between a dynamic activation of function with β = 10% and a dynamic activation of function with β = 5%. But what we can also see in both cases is an improvement in accuracy of classification. It is also specified that will be necessary to find the optimal of dynamic modification for activation function from a training epoch to another training epoch.

VI. Conclusions

Introducing the concept of dynamic modification during the training process leads to an increase of accuracy for classification. We can remark that a classical training algorithm such as backpropagation algorithm becomes more flexible in the meaning that the learning algorithm is faster guided toward a classification error generally lower.

The proposed method is attempted to monitor the error and check the status for accuracy for different data sets.

Influence of the nature of the data set is no longer reflected in the weights of the neural network but it can be reflected in the activation function. For this reason, the dynamic modification of the activation function may represent a significant way to improve the classification neural network in the wide range of directions where can advance this concept.

References

[1] J. Kilian, H. T. Siegelmann, "The Dynamic Universality of Sigmoidal Neural Networks", New Jersey, Israel, information and computation 128, pp. 48-56, 1996.

[2] J. Hu, Lili Xu, X. Wang, X. Xu , G. Su, "Effects of BP Algorithm-based Activation Functions on Neural Network Convergence", Journal of Computers Vol. 29 No. 1, pp. 76-85, 2018.

[3] V. Tiwari, N. Khare, "Hardware implementation of neural network with Sigmoidal activation functions using CORDIC, Microprocessors and Microsystems", 39(6), pp. 373-381, 2015.

[4] Z.J. Jia, Y.D. Song, D.Y. Li, P. Li, „Tracking control of nonaffine systems using bio-inspired networks with autotuning activation functions and self-growing neurons", Information Sciences pp. 388-389, 191-208, 2017.

[5] A. F. Sheta, S. E. M. Ahmed, H. Faris, „A Comparison between Regression, Artificial Neural Networks and Support Vector Machines for Predicting Stock Market Index", (IJARAI) International Journal of Advanced Research in Artificial Intelligence, Vol. 4, No.7, 2015.

[6] Siegelmann, H. T., and Sontag, „Turing computability with neural networks", Appl. Math. Lett. 4, 6, 1991.

[7] C. Özkan, F. S. Erbek, "The Comparison of Activation Functions for Multispectral Landsat TM Image Classification", Istanbul Photogrammetric Engineering & Remote Sensing Vol. 69, No. 11, pp. 1225–1234, 2003.

[8] R. Rojas, „Neural Networks", Berlin, Springer-Verlag, 1996.

[9] Oda, Tetsuya, et al, "A Neural Network Based User Identification for Tor Networks: Comparison Analysis of Different Activation Functions Using Friedman Test", Network-Based Information Systems (NBiS), 19th International Conference on. IEEE, 2016.

[10] M. Cilimkovic, „Neural Networks and Back Propagation Algorithm", Ireland, 2015.

[11] K. Hara, K. Nakayamma, „Comparison of activation functions in multilayer neural network for pattern classification", Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94), Orlando, USA, pp. 2997-3002 vol.5. doi: 10.1109/ICNN.1994.374710, 1994.

[12] J. Roman, A. Jameel, "Backpropagation and Recurrent Neural Networks in Financial Analysis of Multiple Stock Market Returns", Xavier University of Louisiana, Proceedings of the 29th Annual Hawaii International Conference on System Sciences – 1996.

[13] Z. Soltani, A. Jafarian, „A New Artificial Neural Networks Approach for Diagnosing Diabetes Disease Type I", (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 7, No. 6, 2016.

[14] P. Koehn, "Combining Genetic Algorithms and Neural Networks: The Encoding Problem", A Thesis Presented for the Master of Science Degree The University of Tennessee, Knoxville, 1994.

[15] V. N. Manohar, G.U. Chaudhari, B. Mohanty, "Function approximation using back propagation algorithm in artificial neural networks", Rourkela, 2007.

[16] Han J., Moraga C. „The influence of the sigmoid function parameters on the speed of backpropagation learning", Berlin, Heidelberg, In: Mira J., Sandoval F. (eds) From Natural to Artificial Neural Computation. Lecture Notes in Computer Science, vol 930. Springer, 1995.

[17] Siegelmann, H. T., and Sontag, E. D., „On the computational power of neural networks", J. Comput. System Sci. 50, 132150, 1995.

[18] T.Jayalakshmi, Dr.A.Santhakumaran, "Statistical Normalization and Back Propagation for Classification", International Journal of Computer Theory and Engineering, Vol.3, No.1, pp. 1793-8201, 2011.

[19] A. Gupts, M. Lam, "The Weight Decay backpropagation for generalizations with missing values", Annals of Operations Research, Science publishers, pp.165-187, 1998.

[20] Liu, Qiang, Yu, Feng, Wu, Shu, Wang, Liang, „A convolutional click prediction model. Proceedings of the 24th ACM International on Conference on Information and Knowledge Management", pp. 1743–1746, 2015.