ACTIVATION FUNCTIONS IN SINGLE HIDDEN LAYER FEED-FORWARD NEURAL NETWORKS

Ender SEVİNÇ[+]
University of Turkish Aeronautical Association, Ankara Turkey
esevinc@thk.edu.tr

Abstract

Especially in the last decade, Artificial Intelligence (AI) has gained increasing popularity as the neural networks represent incredibly exciting and powerful machine learning-based techniques that can solve many real-time problems. The learning capability of such systems is directly related with the evaluation methods used. In this study, the effectiveness of the calculation parameters in a Single-Hidden Layer Feedforward Neural Networks (SLFNs) will be examined. We will present how important the selection of an activation function is in the learning stage. A lot of work is developed and presented for SLFNs up to now. Our study uses one of the most commonly known learning algorithms, which is Extreme Learning Machine (ELM). Main task of an activation function is to map the input value of a neural network to the output node with a high learning or achievement rate. However, determining the correct activation function is not as simple as thought. First we try to show the effect of the activation functions on different datasets and then we propose a method for selection process of it due to the characteristic of any dataset. The results show that this process is providing a remarkably better performance and learning rate in a sample neural network.

Keywords: Machine Learning, SLFN, ELM.

1. Introduction

Neural networks are widely used in many areas because of their powerful, fast and accurate learning capacities. Many new and different algorithms have been put forward and

[+] This paper has been presented at the ICENTE'18 (International Conference on Engineering Technologies) held in Konya (Turkey), October 26-28, 2018.

studies have been presented in Artificial Intelligence (AI). Additionally, many different types of machine learning methodologies are developing gradually. The purpose of our study is to present the role of activation function in a neural network. ELM is the one of the most popular learning methods used for such neural networks.

Despite many data and inputs available, AI mechanisms or smart systems are still lacking in powerful interpreter systems that can comment on these results. For example, in cancer detection cases, there might be many data; however, there are still cases that you can encounter with false positives or true negatives. Generally, it will not be so clear that which features you will use for better learning. Though there are many studies in this area, an optimization in this AI field seems to be valuable.

At this point, ELM capable of high and fast learning rate in neural networks can take part. The inputs and their selection methods are important; however, related two topics come forward. The first one is the importance levels of the inputs. Studies [1-4] can be examined on such a topic. This is so important that for example you might have 10 inputs for a case, and then if you select only third and fifth features of that dataset, learning rate might be more than the status which all features are involved.

The second topic is to get the *hidden neuron matrix* (H) that is to evaluated due to the activation function. This matrix is required for mapping the inputs to the output nodes. After determining the weight matrix, an activation function is selected and all the values in that matrix are manipulated due to it. This means the learning capacity of the network is directly related with both weight matrix and the activation function.

Although the number of the hidden neurons is important for getting higher learning rates, activation function in SLFN has also a remarkable effect for achieving higher learning levels. We think that this work can be used for determining the activation function and can form a base for another work.

In this study, small and medium sized data sets from [5] will be used, and in order to get rid of the effect of hidden number of neurons on the experiments, a varying range of hidden neuron number will be used in the experiments. Thus within those ranges, the effect of different types of activation functions will be seen and observed at the same time.

In the 2nd part related works will be mentioned, in the 3rd section the activation functions and formulations will be discussed. Experimental studies will be presented in the 4th section and a method for determining the most appropriate one for any dataset will be proposed. Lastly, comments about future work and results will be discussed.

2. Related Works

Up to now, there are many examples of machine learning studies using ELM. The process starts with instantiating the links between the input and output layers. Initially, hidden neurons are randomly assigned; weight and biases are calculated due to inputs. Then this matrix is processed with the activation function, which is one of the important levels of the process. Then the output connections are set and found by reducing the cost function to a minimum through a linear system. Also mentioned in the related website [6], ELM is used for deriving learning methodologies in a similar way in many AI areas because of having low computational complexity, accurate results and being very fast [1, 2, 4].

Especially effect of activation functions and saturation during training formulations are discussed in [7]. After making experiments with sigmoid, hyperbolic tangent functions and some other types, they declared that the logistic regression or conditional log-likelihood cost function worked better for any type of classification. It's also mentioned in the same study that classical neural networks with sigmoid or hyperbolic tangent functions, converge more slowly and apparently get stuck ultimately to a poorer local minima [7].

However, a new recurrent neural network with one-layer architecture and a discontinuous hard-limiting activation function for solving quadratic programming problems have been discussed in [8]. Study examining quadratic and nonlinear programming and SVM learning methods, puts forward the benefits of a recurrent neural network over a general one.

In the final study, a number of activation functions have been examined. It has been declared that most deep neural networks generally use non-periodic and monotonic activation functions. Neural networks with a single hidden layer using sinusoidal activation functions is declared to be used largely and by this, higher learning rates are more probable with the networks relying on the periodicity of the sinusoids [9].

Our study shows and discuss all of the types of functions discussed in this part.

3. Implementation

3.1 Extreme Learning Machine (ELM)

The purpose of this study is to show the precise effect of hidden neurons in any neural network. We will also suggest a new method based on the nature of the data set to achieve a higher learning rate.

The problem solving technique here proposes a learning methodology for Single-hidden Layer Feedforward Neural network (SLFN)s. We will use a learning algorithm called *ELM* whose learning speed is much faster than traditional feedforward network learning algorithms like back-propagation (BP). For detailed information, studies [1-4] can be examined. However, for better understanding ELM will be explained briefly.

The output of SLFN having L number of hidden nodes is represented in (1) below;

$$f_L(x) = \sum_{i=1}^{L} \beta_i G(a_i, b_i, x), \quad x \in \mathbf{R}^n, a_i \in \mathbf{R}^n, \tag{1}$$

where a_i and b_i are learning parameters of hidden nodes and β_i is the weight connecting the i^th hidden node to the output node. $G(a_i, b_i, x)$ is the output of the i^th hidden node with respect to the input **x**.

Given a standard SLFN with L hidden nodes and activation function $g: R \rightarrow R$, which is infinitely differentiable in any interval and the hidden layer, output matrix H of the SLFN is invertible and defined as in (2);

$$\mathbf{H}\beta = \mathbf{T} \tag{2}$$

H is the output of hidden layer matrix of the neural network. The i^th column of **H** is the i^th hidden node's output vector with respect to inputs $x_1, x_2, \ldots, x_N$ and the j^th row of **H** is the output vector of the hidden layer with respect to input x_j. **T** is the training error matrix in (2) and the smallest training error can be reached by this solution: i.e, min $\|\beta\|$ and min $\|\mathbf{H}\beta - \mathbf{T}\|$ a simple representation of the solution of the system in (2). This is also the target of the whole network and explained in detail in [1, 2].

3.2. Activation functions

In SLFNs, hidden neurons are modeled using an *activation function* for output. This function is important to implement and help to understand functional mapping between input and output nodes in a SLFN. By applying properties of the activation function to the neural network, the input signal is transformed to the output counterpart. For estimating the role of an activation function in a neural network, a sample illustration has been shown in Figure 1.

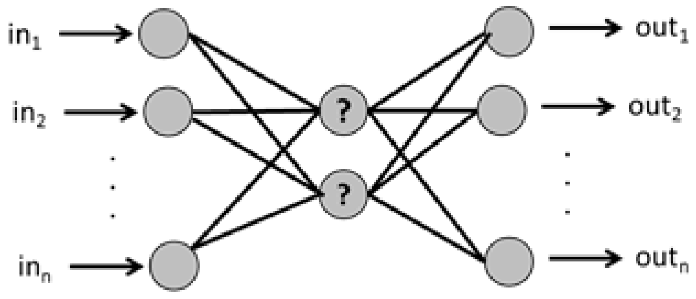

Figure 1. Role of activation function in a neural network

4 different types of activation function will be discussed in this study;

1) Sigmoid Function

Sigmoid is a non-linear, monotonic and S-shaped activation function that produces an output value in the range (0, 1). In this context, sigmoidal function is a special form of logistic function and is defined in (3);

$$\text{sig(x)} = \frac{1}{1+e^{-x}} \tag{3}$$

A sigmoid function is a bounded differentiable real function that is defined for all real input values and has a positive derivative at each point. Sigmoid functions are known to be one of the most commonly used and major hidden layer output functions used in a neural network. A wide variety of sigmoid functions have been used as the activation function of artificial neurons, including the logistic and hyperbolic tangent functions.

2) Tanh(hyperbolic tangent) Function

Tanh has a similar characteristic to that of sigmoid. It is nonlinear function and has a bound to range (-1, 1). Deciding between the sigmoid or *tanh* will depend on the requirement of gradient strength and is defined in (4);

$$\text{tanh (x)} = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{4}$$

Like sigmoid, *tanh* also has the vanishing gradient problem. This problem originates from multiplying many small numbers to compute gradients of the "front" layers in a neural network, ending in the gradient decreases exponentially. Since gradient is so small, computation is calculated down to limits of floating point data types and the system cannot learn any further or drastically degrade.

3) Sine Function

Sine is one of the commonly used nonlinear functions and has a bound to range (-1, 1). It is a periodic function and has rises and falls. However, the function saturated and its output converges to zero for large positive and negative inputs. These sinusoids are commonly defined as a generalized Fourier decomposition and have the potential to converge to a local minimum in their approximation capacity.

4) Hardlim Function

The hard limit transfer function forces a neuron to output a "1" if its net input reaches a threshold, otherwise it outputs "0". This allows a neuron to make a decision or classification. It can say yes or no finally and is defined in (6);

$$\text{hardlim}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Hardlim is a neural transfer function that calculates a layer's output from its net input.

4. Experiment Results

All experiments are done on a computer having 64-bit Windows 7 operating system, i5 4200u 1.6 Ghz. processor and 8 GB. RAM. Datasets are taken from UCI web page [1] which is commonly used and referenced in this field. An ascending order of datasets is presented due to the number of features and the instances in Table 1. Datasets will be referred with their IDs as shown from now on.

For each test, 10-fold cross validation method has been used for avoiding the effect of chance. This method, as mentioned in [10], whole data set is equally divided into defined fold number, i.e. 10 for our case. Then among 10 equal parts, the first 9 pieces are used for learning and the final part is used for testing. In this way, all parts are subjected to the same process by exchanging one by one. Thus, all sub parts are rotated 10 times in the same way, i.e. 9 for learning, 1 for testing. Finally, the averages of all runs are taken and the results are calculated.

Table 1. Used Datasets and Characteristics.

Dataset	**ID**	**# instances**	**# features**	**# output**
Iris	IRI	150	4	3
Monk1	MK1	432	7	2
Monk2	MK2	432	7	2
Monk3	MK3	432	7	2
Ionosphere	ION	351	34	2
WDBC	WDB	569	32	2
Pima Indian Dia.	PID	768	8	2
Wis.Bre.Cancer(Org)	WIS	699	10	2
Waveform	WAV	569	21	3
Spambase	SPM	4601	57	2

The test results are shown in Figure 2. All 10 datasets in Table 1 are put together and one whole figure is created. Each dataset has its own ID name on top of itself. Horizontal scale shows number of hidden neurons changing form 10% to 200%. This percentage is obtained by dividing the used number of hidden neurons to the number of instances in the dataset. Vertical scale shows the accuracy level achieved. It is between (0.0 - 1.0) as in (7).

$$\text{Accuracy} = \frac{\#\, truly\_known\_instances}{\#\, all\,\_instances} \qquad (7)$$

Figure 2 shows the learning rate of activation functions with respect to datasets. In the experiments, the number of hidden neurons is also changing and inevitably has an effect on the learning rate of the system. In order to get rid of those effects, commonly used ranges are assigned and experiments are done due to these ranges, varying from 10% to 200% of the instance number of the dataset. All 10 datasets in Table 1 are included in Figure 2, and each dataset are applied with same hidden number of neurons and the same activation functions. Consequently, we will take the sample mean of all results and be able to see the real, purified effect of the activation function on the learning rate.

If we examine Figure 2, activation functions using sigmoid and sinusoid seem to be more successful for predicting the output. Another point to take attention is that the values tend to come closer after 40%-50%'s of horizontal scale, which means deviation gets smaller. Because the more hidden number of neurons, you put in H matrix, the more accuracy you get.

As a result, this effect caused a variance of the distribution of the sample mean, and it can be clearly observed that values between 10% and 50% have more deviance and cause more error. Because of that, values greater than 60% will be taken into consideration and added to calculations to get the average. After taking the average of the values in Figure 2 as stated, we will get Figure 3. Here we can see the performance results of activation functions due to the datasets.

For example, in Figure 3, WIS dataset shows the average of values of 4 activation functions of WIS dataset that are changing between 60% and 200% stated in Figure 2. In addition, we can observe which activation function has the most learning rate separately for that and the other datasets together.

If to talk about execution times of activation functions, they are almost close to each other, changing from 0.5 sec to 30 sec. In fact, the fundamental difference for execution times is originating from the size of dataset. The smaller dataset used, the shorter time it is to spent for execution. It can be inferred that the effect of activation function is negligible for the execution times.

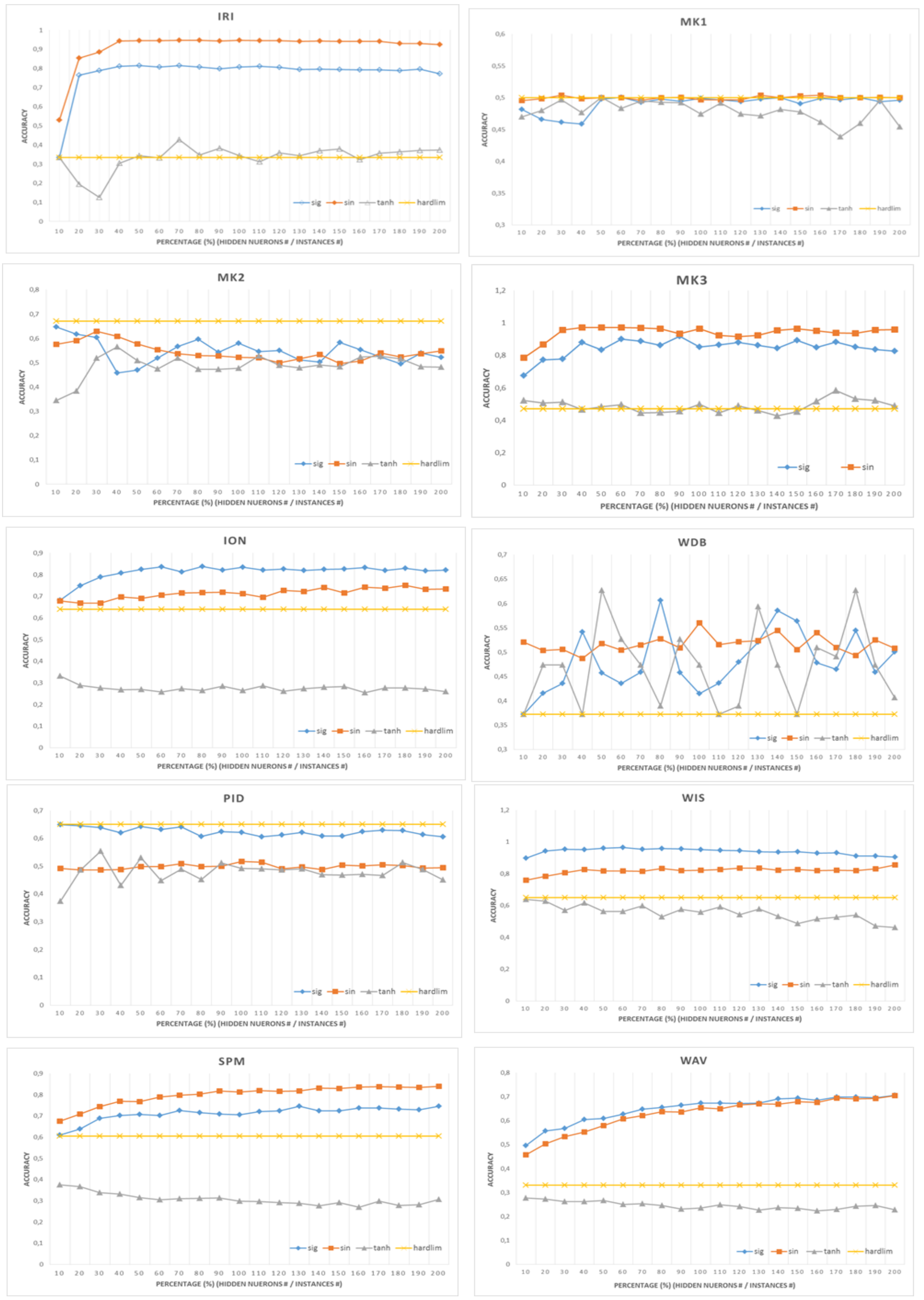

Figure 2. Results of activation functions with changing hidden neurons due to datasets.

Now we can propose a method for the data used in the experiments. If we examine Figure 3, sigmoid activation function proves to be the most successful among others. Because many natural processes, such as those of complex system learning curves, exhibit a progression from small beginnings that accelerates and approaches a climax over time and a sigmoid function most probably best suits for such a case. Thus, a mapping process of a sigmoid function might be most successful on such conventions.

Likewise, when a detailed description or information is lacking, sigmoid function can often be used and is acceptable among others [11].

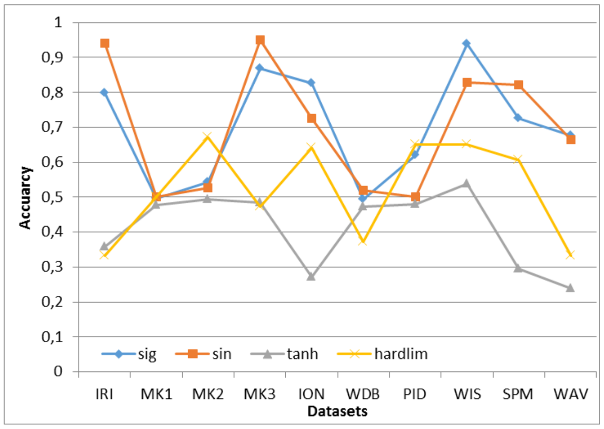

Figure 3. Dataset Performance due to Activation Functions.

On the other hand, *Sine* function has a similar learning capacity, which is very close to that of sigmoid. This is mainly because of having the same origins. They are both non-linear, monotonic and have similar ranges for negative and positive infinity values.

Additionally, back propagation relies on the use of a differentiable activation function and commonly use sigmoid. However, replacing it with another differentiable function such as sine does not affect the convergence so much and thus we get similar results for these two functions.

Tanh function turns out that the logistic sigmoid that might "stuck" to a minimum during training. This is partially due to the fact that if a strongly-negative input is provided to the logistic sigmoid, it outputs values very close to zero that degrades learning. Because of that, *tanh* function has the worst performance among the others.

Hard limit (*hardlim*) transfer function forces a neuron to output a 1 if its net input reaches a threshold, otherwise it outputs 0. This allows a rough calculation for the output and lower performance is observed. As seen in Table 2, general averages of 4 types of activation functions are given due to average performance value of accuracy levels of all datasets. The highest one is sigmoid and sinusoid functions respectively.

Table 2. General Performance of Activation Functions

Function Name	**Learning Rate (%)**
Sigmoid	69,86
Sine	69,75
Tanh	41,11
Hardlim	52,29

Table 2 shows the learning capabilities of the activation functions that form the goal of this study. Besides that, as the dataset sizes increase in terms of instances and feature numbers, sigmoid function seems to achieve better learning rates as seen in Figure 3. This result is also similar for sine function since they are similar. Consequently, most appropriate activation function for our study is either sigmoid or a sine function as in Table 2.

Finally, our method to propose is that if you do not have enough information about the characteristic of our data you can use sigmoid as activation function without any problem. If you have idea, for example if it is medium-sized or a huge dataset, you can prefer non-linear, monotonic and a derivative function that can smooth gradient of the data which is an important parameter.

5. Conclusion

In this study, we examined the effects of activation functions in a neural network. However, *sigmoid* function which is also called as standard logistic function is still very popular in classification problems and can be used on any occasion. On the other hand, *sine* function is also acceptable.

One of the most probable future works might be to examine the effect of hidden neurons in neural networks. It is also observed in our experiments that any changes in this number might have a remarkable effect on the learning capacity of the system. An optimized point on this topic might be a promising future work.

References

[1] UCI Irvine Machine Learning Repository [online], http://archive.ics.uci.edu/ml/datasets.html , [Last visited: 02 Jan 2019],

[2] Huang GB, Zhu QY, Siew CK, Extreme learning machine, Theory and applications, Neurocomputing 70, 2006; 489-501.

[3] Huang GB, Ding X, Zhou H, Optimization method based extreme learning machine for classification, Neurocomputing 74, 2010; 155-163.

[4] Feng G, Qian Z, Zhang X, Evolutionary selection extreme learning machine optimization for regression, Soft Computing 16, 2012; 1485-1491.

[5] Huang GB, Zhou H, Ding X, Extreme learning machine for regression and multiclass classification, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 42, 2012; 513-529.

[6] ELM Classification web page [online], *http://www.ntu.edu.sg/home/egbhuang/elm_random_hidden_nodes.html* [Last visited: 02 Jan 2019],

[7] Glorot X, Bengio Y, Understanding the difficulty of training deep feedforward neural networks, Proceedings of the 13th International Conference on Artificial Intelligence and Statistics, 2010.

[8] Liu Q and Wang J, A One-Layer Recurrent Neural Network with a Discontinuous Hard-Limiting Activation Function for Quadratic Programming, IEEE Transactions On Neural Networks, Vol. 19, No. 4, April 2008.

[9] Parascandolo G, Huttunen H, Virtanen T, Taming the Waves: Sine as activation function in deep neural networks, ICLR 2017 conference submission, 2017.

[10] Kohavi R. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. Conference, International Joint Conference on Artificial Intelligence (IJCAI), 1995

[11] Gibbs M.N., Variational Gaussian process classifiers. IEEE Transactions on Neural Networks. 11 (6), Nov 2000; 1458–1464.