Tek Gizli Katmanlı İleri Beslemeli Sinir Ağlarında Gizli Nöronların Etkisi

Araştırma Makalesi/Research Article

Ender SEVİNÇ

Bilgisayar Mühendisliği Bölümü, Türk Hava Kurumu Üniversitesi, Ankara, Türkiye
esevinc@thk.edu.tr

(Geliş/Received:30.09.2018; Kabul/Accepted:04.08.2019)
DOI: 10.17671/gazibtd.465886

Özet— Son on yılda Yapay Zeka, derin öğrenme ve sinir ağlarının gerçek zamanlı birçok problemi inanılmaz derecede hızla çözebilen güçlü makine öğrenmesi teknikleri kullanmaları, bu konuya giderek artan bir popülerlik kazandırmaktadır. Bu çalışmada, İleri Beslemeli Sinir Ağları'nda kullanılan parametrelerin iyileştirilmesi için hızlı öğrenme yeteneğine sahip Olağanüstü Öğrenme Makinaları (OÖM) kullanılmaktadır. Bu tür sistemlerin öğrenme kapasitesi, parametrelerin ve hesaplama yöntemlerinin etkinliği ile doğrudan ilişkilidir. Burada kullanılan parametrelerinden biri olan gizli nöron sayısı tartışılacak ve değişik durumlardaki performansı incelenecektir. Bu değerin uygun seçiminin önemini vurgulanacak ve uygun seçim için yeni bir yöntem önerilecektir. Önerilen yöntem, Normalize Ortalama Değer (NOD) bu alandaki istatistiki metotları temel alan basit ve etkili bir formülasyondur. Doğru gizli nöron sayısı (L) değerini belirlemeye yönelik yapılan deneysel sonuçlar, L'nin rastgele seçilmesinin, aşırı ya da yetersiz uyum gibi problemlere neden olabileceğini göstermektedir. NOD daha iyi öğrenme seviyelerine erişme imkanı sağlamaktadır. Deneysel sonuçlar, gizli nöronların sayısı belirlenmesi durumunda, %10-15'lik bir performans artışı sağlandığını göstermektedir.

Anahtar Kelimeler— Yapay Zeka, İleri Beslemeli Sinir Ağları, OÖM, girdi seçimi.

The Effect of Hidden Neurons in Single-Hidden Layer Feedforward Neural Networks

Abstract— Especially in the last decade, Artificial Intelligence is gaining popularity increasingly since deep learning and neural networks have fast and powerful machine learning-based techniques that can solve many real-time problems efficiently. In this study, Extreme Learning Machine (ELM), capable of high and fast learning is used for optimization parameters of Single hidden Layer Feedforward Neural networks (SLFN)s. The learning capability of such systems is directly related to the effectiveness of the parameters and the calculation methods. Hidden neurons number, one of the parameters in the calculations is discussed and its role is examined. The importance of the appropriate selection of this value will not only be emphasized but also a new method will be proposed for proper selection. The proposed method, Normalized Average Value (NAV) is a simple and effective formulation that originates from statistical methods in this field. Experimental results for determining a correct number of hidden neurons (L) show that random selection of this number causes either overfitting or under fitting problems. NAV can improve any algorithm in order to reach better learning rates. The results show that it provides a 10-15% performance increase due to random selection if the number of hidden neurons, L is determined according to the result of the study.

Keywords— Artificial Intelligence, SLFN, ELM, Feature Selection.

1. INTRODUCTION

Neural networks are commonly used in many areas because of their powerful, fast and accurate learning abilities. Many new and different algorithms have been put forward and studies have been presented in Artificial Intelligence (AI) field up to now.

Despite many data and inputs, AI mechanisms or smart systems are still lacking in powerful interpreter systems that can comment on these results as also stated in [1, 2]. For example, in detecting output nodes, there might be many different types of data and a lot of inconsistencies can be found in the input. Since the priority of each feature of the input is generally vague, the predictions are generally not so stable and promising. As a result, many studies and efforts were put forward in this field since the relation among the features of the input and the output classes drew attention any time.

Neural networks are popular especially as a supervised machine learning method. However, they have some drawbacks at the same time. Being dependent prior knowledge, selection of activation function and mapping input to output data, gradual learning and overfitting/underfitting problems, also mentioned in abstract, are some of them. But their fabulous power in multiple domains from data science to computer vision and asset at solving complex tasks that involve generalizations make them very popular and common in AI area. It can be claimed that they are very good at approximating behavior of any complex mathematical function.

Extreme Learning Machine (ELM) is a high and fast learning method, mostly used for single layer, i.e. *Single hidden Layer Feedforward Neural Networks (SLFN)*, and multi-layer networks. For mapping input nodes onto output ones, importance levels of the attributes are first to be examined. In order to provide some type of mapping and prioritization among features, networks such as sigmoid networks, RBF networks, threshold networks, trigonometric networks are commonly used. Studies in [1, 2] can be examined and given as the latest examples of providing prioritization for the features of the input. For example, you might have 10 inputs for a case and if you select only third and fifth features of the dataset, an increase in learning rate is quite probable.

The second item is obtaining the *Hidden neuron matrix* (H) which is evaluated due to an activation function. H matrix is calculated in the training phase and then used for predicting the output nodes in the testing phase. As a result, we can see how much learning rate can be accomplished on a sample data presented in [3] and this work can be used for the creation of H matrix by selecting the number of hidden neurons. In this study, medium-sized data sets are used, and the effect of the hidden neurons is put forward by using ELM. On the other hand, there is a remarkable effect of activation functions in SLFNs for achieving higher learning rates. The effect of different action functions on the learning rate is presented in [4].

In this study, small and medium sized data sets is used and the range of hidden neuron number is changed in order to minimize the effect of hidden neuron numbers. This is because of minimizing the effect of the activation function. Results will be discussed due to the averages of them. The datasets are taken from UCI repository [3], popular and commonly used in such studies of machine learning. Comparison with the latest studies having a multi-layer neural network architecture will also be discussed.

In the 2nd section, related works are mentioned, in the 3rd section why hidden neurons are so important in SLFN is discussed and a method is proposed for determining the most appropriate number of hidden neurons. We present experimental studies in the 4th section. A new method is also proposed since the main goal and the benefit of this study is presented. Finally in the last section, comments about future works and results are discussed.

2. RELATED WORKS

Up to now, there exist many examples of machine learning studies in SLFNs using ELM. The process starts with instantiating the links between the input and output layers. Initially, hidden neurons are randomly assigned, then weight and biases are calculated due to inputs. Then this matrix is processed with the activation function, which is one of the important levels in the process. With this method, the output connections are set and found by reducing the cost function to a minimum through a linear system. ELM, mentioned in the related website in [5], is known for its low computational complexity, accurate results and is extremely fast. [6, 7, 8]. The method proposed in the study, NAV is a simple, easy to implement. This method has been integrated into an algorithm in order to improve its capability for reaching better accuracy rates. The other important capability about NAV is that it can be implemented to in any algorithm. The rest of the implementation details will be mentioned in Section 4.

Most important related work with this study is other studies using ELM. In [9], Huang et al. proposed a method with ELM by using the regression based on the two-stage selection process. They tried to construct decision support tree by using information entropy method and to investigate the result. In another similar study, Feng et al. in [10] tried to predict the results by using the regression method, one-step but different formulations.

ELM is an example of supervised machine learning algorithms in SLFNs. It is commonly used and one of the most popular methods in the literature. It starts its process with randomized values, and then maps input nodes onto output counterparts in a linear system. Especially this mapping process is dependent upon the activation function and the related other parameters. One of the most important parameters for the calculation is the number of hidden

neurons. Though changes due to datasets, the effect of L in a neural network is important.

In order to find out correct L, studies as in [11, 12] have been proposed different methods. For example, they tried to find an upper bound for L in [11] or a technique, Singular Value Decomposition, is applied to the hidden layer output activation matrix in order to provide or estimate the weights of H matrix learned during the training stage of the SLFN. Many other similar and generic studies related to the selection methods of features have been put forward in [13, 14] in the literature.

A number of methods have been introduced to stabilize the hidden neuron number and a network design has been proposed in order to reduce learning mistakes in [15]. Some measures have been proposed to increase the stability and accuracy of the neural network. The approach in [15] is determined to implement the selection of proper hidden neuron number in Elman network for renewable energy systems.

One of the state-of-the-art and recent studies are presented in [16, 17]. A Teaching-Learning-Based Optimization algorithm, TLBO-ELM has been put forward for data classification. Study in [16] has two phases, in which the first is feature selection phase and then the second is the data classification phase. The proposed algorithm uses an implementation of classical evolutionary genetic algorithms on parallel/distributed computation environments. It has promising accuracy rates. Similarly, in [17], a parallel execution method for data classification is presented. The proposed algorithm executes data classification in parallel which remarkably improves the solution times and accuracy rates.

Finally, a different type of solution for neural networks is discussed. They might have a multi-layer architecture with a pruning method for the design of neural networks. The main purpose in them is to start with an oversized network and then prune it to smaller sizes. Thus, it will be easier to get less computational complexity and better performance in generalization. The study in [18] proposes such an approach using an algorithm that implements a pruning technique which removes the neurons with the least relevance by means of a quantified sensitivity measure. It is claimed that the correctness and effectiveness of the technique are quite promising by using a multilayer perceptron.

The study in [19] has a similar approach for data classification as well. The proposed algorithm, neural network pruning by significance (N2PS) has a pruning technique for each individual input node and then pruning them due to a significance threshold value. N2PS algorithm determines an optimal architecture of neural networks of arbitrary topology for classifying large datasets from UCI repository [3] by using sigmoidal function and will be used for comparison in Section 4.

3. IMPLEMENTATION

3.1. Extreme Learning Machine

The purpose of this study is to show the precise effect and role of hidden neurons in calculations, suggesting 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 used a learning algorithm called Extreme Learning Machine (ELM) whose learning speed is quite faster than traditional feedforward network learning algorithms. For detailed information, studies [6-10] can be examined for better understanding. However, we explain ELM briefly.

The output of SLFN having L number of hidden nodes is shown with Eq.1 below;

$$f_L(x) = G(a_i, b_i, x), \qquad x \in \mathbf{R}^n, a_i \in \mathbf{R}^n \quad (1)$$

where, a_i and b_i are weight and bias 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. The activation function $G(a_i, b_i, x)$ is;

$$G(a_i, b_i, x) = g(a_i.x + b_i), \; b_i \in R \quad (2)$$

In Eq.2, $a_i.x$ denotes the inner product of the vectors a_i and x where both are element of R. In a SLFN with L hidden nodes with activation function g(x) can approximate these L samples with zero error means that $\sum_{i=1}^{L} || o_i - t_i || = 0$ which means that there exists β_i , a_i and bi in Eq.3 such that;

$$\sum_{i=1}^{L} \beta_i \, . \, g(a_i . x_j + b_i) = t_j \qquad , j=1,..,N \quad (3)$$

N is the number of samples, i.e. inputs. This equation can be compactly written as in Eq.4;

$$H \beta = T \quad (4)$$

where

$H(a_1,\ldots,a_L, b_1,\ldots,b_L, x_1,\ldots,x_N)$

$$= \begin{bmatrix} g(a_1.x_1 + b_1) & \cdots & g(a_L.x_1 + b_L) \\ \vdots & \vdots & \vdots \\ g(a_1.x_N + b_1) & \cdots & g(a_L.x_N + b_L) \end{bmatrix}_{NxL} \quad (5)$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{Lxm} \quad \text{and} \quad T = \begin{bmatrix} t_1^T \\ \vdots \\ t_N^T \end{bmatrix}_{Nxm} \quad (6)$$

In Eq.5, H is the output of hidden layer matrix of the network. β^T is the transpose of a matrix or vector β as in Eq.6. The i^{th} column of H is the i^{th} hidden node's output vector with respect to inputs x_1, x_0, …, x_N and the j^{th} row of H is the output vector of the hidden layer with respect to input x_j. However, the number L is generally much smaller than the number of training data. If this is checked under the constraint of minimum norm least square, i.e., min $\|\beta\|$ and min$\|H.\beta-T\|$ a simple representation of the solution of the system was given explicitly by Huang et al. [7, 8, 11] in Eq.7

$$\tilde{\beta} = \mathbf{H}^{\dagger}T, \quad (7)$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse [20] of the hidden layer output matrix H. Huang et al.[11] had further shown that H is column full rank with probability one when L<N if the N training data are distinct. In real life applications, the number of hidden nodes is mostly less than the instance number of training data, L<N. Thus, $\mathbf{H}^{\dagger}$ can be re-written as $(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T$ [10].

3.2. Selection of Hidden Neurons

The hidden neurons in SLFNs are modeled using an activation function for output. This function is important to learn and understand functional mapping between input and output node points for all neural networks. By applying nonlinear properties to the neural network, they transform the input signal to the output counterpart.

In our study, *Sigmoid* function is used as the activation function which is known to be one of the most popular one. Sigmoid is a non-linear, monotonic and S-shaped activation function that produces a value in the range [0,1]. In this context, sigmoidal function is a special form of logistic function and is defined in Eq.8;

$$\text{sig}(x) = \frac{1}{1+e^{-x}} \quad (8)$$

The effect of hidden neurons begins with the formation of the weight matrix and biases are added. These values are initially assigned with random values as mentioned. After implementing the activation function on H matrix, output values of SLFN are calculated as the result. Though activation has an effect on the learning, the matrices formed by the help of hidden neurons, L and inputs have remarkable effect for predicting the output nodes [21].

In order to get rid of the effect of any probability, number of hidden neurons is changed proportional to the number of the dataset instances and are tested from 10% to 200% of them. This range is covered with 10% increments in the experiments. After getting the results, each data set is averaged in the same percentile.

Although you can reach better learning rates as the number of hidden neurons, L increase, but this does not mean if L is increased to twice as many as instance number, you get better result. There is an optimum value for the number L and our goal is to find a generic method to get optimum value. In other words, using twice of instance number as L value may not be so meaningful [22]. It increases both execution time and degrades the accuracy. The reason for including hose percentile is for covering the search space as much as possible. Additionally, it will burden extra complexity for multi-layer neural network architectures.

4. EXPERIMENTAL RESULTS

All experiments are performed on a computer having 64-bit Windows 7 operating system, i5 4200u 1.6 Ghz. processor with 8 GB RAM. Datasets are obtained from UCI web page [3]. The details of the datasets are presented in Table 1. An ascending order of datasets is presented due to the number of features and the instances. Datasets will be stated with their IDs.

K-fold cross validation is a statistical method as explained in [23]. It is also called as *rotation estimation* that shows how you can obtain a generalized result of a statistical analysis from an independent data set. The number for k-fold is selected as 10 as in most of the studies. Additionally, the number, 10 is selected because of being the optimum value for avoiding the effect of randomness and having a reasonable execution time in the process.

Briefly, for each test 10-fold cross validation method has been used for avoiding the effect of chance. In the method each data set is divided into 10 equal parts, the first 9 pieces are used for learning and the last part is used for testing. In this way, all parts are subject to the same treatment by turning. Due to 10 folds, all sub parts are rotated 10 times in the same way as 9 for learning 1 for testing. Then the average values are calculated and results are found. The averages of all operations are presented in Figure 1.

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
Vehicle	VEH	846	18	4
WDBC	WDB	569	32	2
Pima Ind.Dia.	PID	768	8	2
Wis.Bre.Can.(Or.)	WIS	699	10	2
Chess(King)	CHS	3196	36	4
Waveform	WAV	569	21	3
Spambase	SPM	4601	57	2

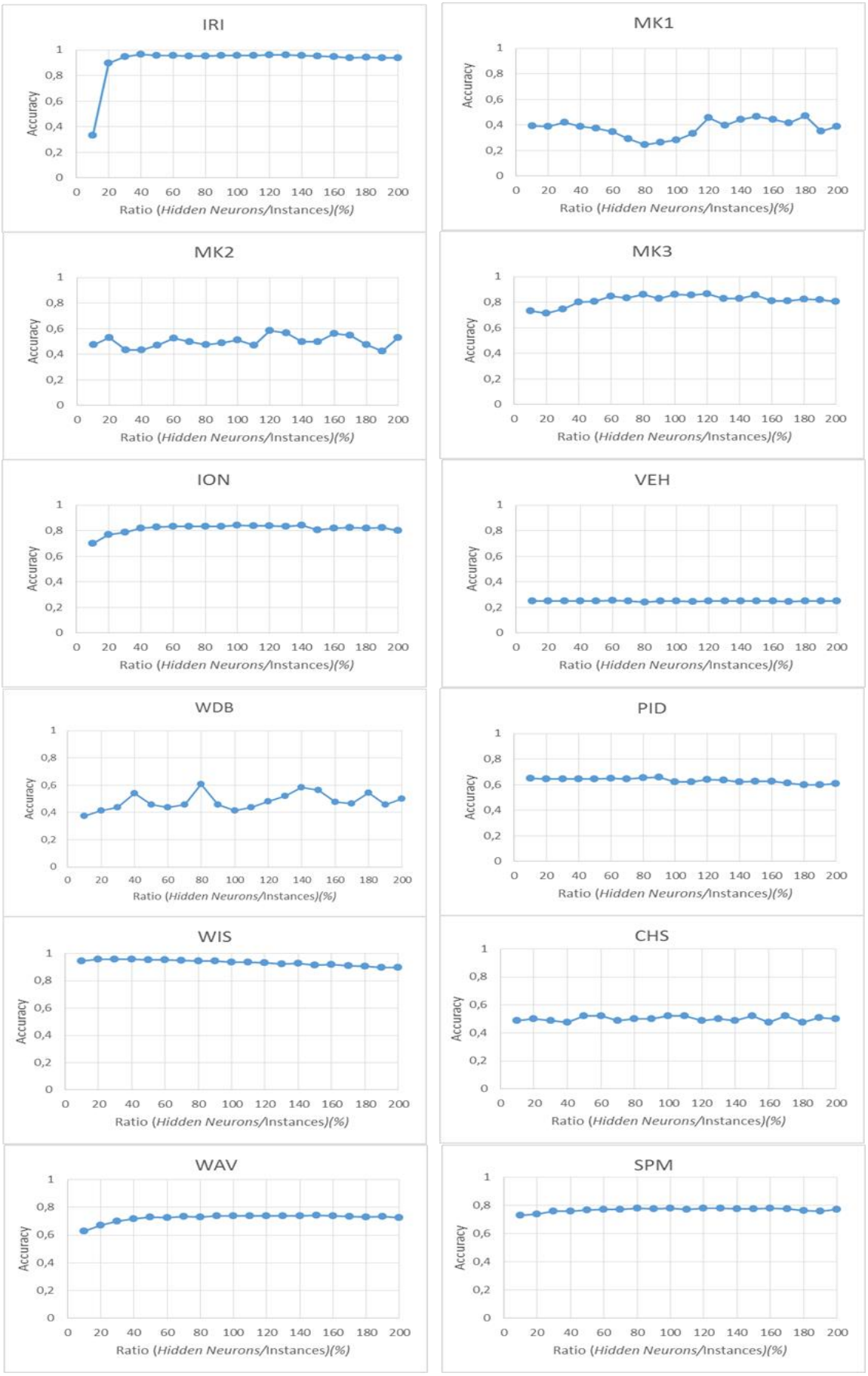

Figure 1. Performance with respect to the changing ratio of hidden neurons

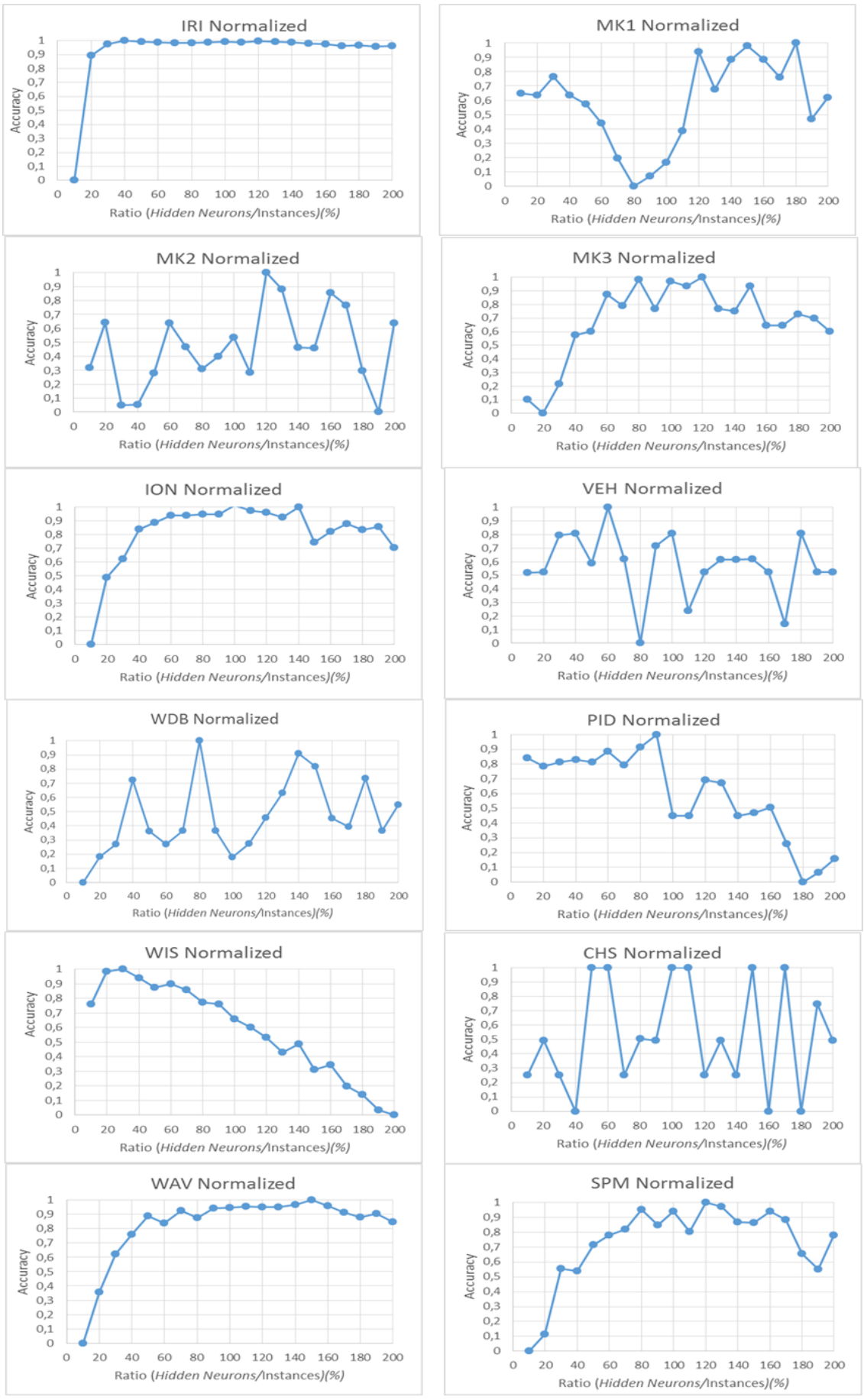

Figure 2. Normalized Learning Rates with respect to changing ratio of hidden neurons

Table 2. Normalized Average Values (NAV) of datasets

Percentile (%)	SPM	WAV	CHS	WIS	PID	WDB	VEH	ION	MK3	MK2	MK1	IRI	Average
10	0,000	0,000	0,254	0,758	0,843	0,000	0,520	0,000	0,104	0,316	0,648	0,000	0,28691
20	0,113	0,356	0,493	0,984	0,786	0,183	0,526	0,489	0,000	0,643	0,634	0,892	0,50814
30	0,553	0,621	0,254	1,000	0,814	0,271	0,795	0,625	0,219	0,050	0,766	0,976	0,57875
40	0,539	0,758	0,000	0,938	0,829	0,723	0,810	0,842	0,578	0,053	0,634	1,000	0,64204
50	0,718	0,889	1,000	0,875	0,814	0,363	0,589	0,888	0,604	0,279	0,572	0,990	0,71510
60	0,782	0,837	1,000	0,899	0,886	0,271	1,000	0,941	0,875	0,636	0,441	0,987	0,79621
70	0,820	0,927	0,253	0,859	0,793	0,368	0,622	0,941	0,792	0,466	0,193	0,983	0,66802
80	0,951	0,875	0,507	0,773	0,914	1,000	0,000	0,947	0,984	0,310	0,000	0,983	0,68712
90	0,847	0,942	0,493	0,758	1,000	0,367	0,715	0,947	0,771	0,401	0,069	0,986	0,69130
100	0,940	0,945	1,000	0,656	0,450	0,181	0,810	1,013	0,969	0,534	0,166	0,990	0,72107
110	0,803	0,957	1,000	0,601	0,450	0,273	0,240	0,974	0,937	0,286	0,386	0,986	0,65785
120	1,000	0,950	0,253	0,531	0,693	0,458	0,523	0,961	1,000	1,000	0,938	0,997	0,77526
130	0,974	0,949	0,493	0,429	0,671	0,633	0,615	0,928	0,771	0,881	0,676	0,993	0,75120
140	0,867	0,968	0,253	0,484	0,450	0,910	0,617	1,000	0,750	0,462	0,883	0,990	0,71954
150	0,864	1,000	1,000	0,312	0,471	0,819	0,619	0,744	0,937	0,459	0,979	0,979	0,76537
160	0,942	0,959	0,000	0,344	0,507	0,452	0,525	0,823	0,646	0,854	0,883	0,975	0,65902
170	0,886	0,912	1,000	0,195	0,257	0,395	0,140	0,881	0,646	0,765	0,759	0,960	0,64964
180	0,656	0,879	0,000	0,140	0,000	0,734	0,810	0,835	0,729	0,298	1,000	0,968	0,58761
190	0,551	0,903	0,746	0,031	0,064	0,368	0,526	0,856	0,698	0,000	0,469	0,957	0,51399
200	0,779	0,848	0,493	0,000	0,157	0,549	0,525	0,706	0,604	0,639	0,621	0,960	0,57341

In the tests, the number of hidden neurons is decided proportional to the number of records of each data set. For example in Table 1, it is seen that there are 846 records in the data set "VEH". Values ranging from 10% to 200%, was used with 10% increments, corresponding to hidden neuron numbers of the related dataset. For example, 20 totally different and independent experiments are held for each dataset and the number of hidden neurons varies from 85 to 1692 in the experiments for the "VEH" dataset having 846 instances. The value, L changes for the other datasets as stated.

Naturally, different learning scales are obtained because of the general characteristics of different data sets. So with the same learning machine and the same hidden neurons; for example, in the "MK1" data set, a learning level of around 30-40% is observed, while in the "IRI" set this level appears to be around 90%. Therefore, in order to get rid of this dataset characteristic effect, each dataset is normalized to achieve the correct formulation. This normalization method enables us to see the adjusted effect of the hidden neurons discarding the characteristics of the dataset.

Normalization process is performed according to a common statistical method called as *Minimum-Maximum Normalization*. In this method, the largest and smallest values of the levels are used as basis. All other data are normalized according to these values. The purpose of this process is to get the smallest value as 0 and the largest value as 1. The process is done with Eq.9 as stated below[24];

$$\text{Normalized} = \frac{(x - \min(x))}{(\max(x) - \min(x))} \qquad (9)$$

This is done to see the real effect of hidden number of hidden neurons on the accuracy level. This change has an effect on the learning rate as the number of hidden neurons change. The result of this process is shown in Figure 2.

Having normalized values, the average values are obtained as in Eq.10 with the name *Normalized Average Value (NAV)*.

$$\text{NAV}(x) = \frac{\sum_{x=1}^{m} k_x}{m} \quad , k_x \in R, m \in Z \qquad (10)$$

where k_x is NAV of the related dataset and m is the changing ratio of the hidden neurons.

The accuracy values of all data sets are normalized according to the relevant percentile, and the values of the averages are presented in Table 2. It must be kept in mind that these values are not real learning accuracy values.

They are relatively calculated and its purpose is to decide the number of hidden neurons in a SLFN in order to reach a higher accuracy.

If examined, the intervals at which the highest learning level is achieved can easily be determined. For example, minimum number of hidden neurons can be selected depending on the time-critical nature of the work being done or the execution time of the program. Datasets accuracies are presented in Table 2 and in the last column, that is "Average", the mean of that percentile is given including all datasets.

When Table 2 is examined, the highest learning rate is achieved with the value 0,79621 which belongs to 60% percentile in the "*Average*" column of the same table. In addition to this, we can observe closer values to 60% but we have to keep in mind that we try to select the minimum number. As the number of hidden neurons increase, the size of the matrices used in the calculations increases proportionally and any rise in the matrix size increases the execution time hyperbolically. Therefore, the less number hidden neurons you use, the shorter time of execution you get.

The NAV of the datasets is the proposed method in this study. Therefore, after deciding on 60% as the hidden neuron number, we can reach a relatively high learning rate on average. On the other hand, 60% being the NAV of datasets might change due to the characteristic of other different datasets. Nevertheless, our method has the benefit of normalization to adjust the effect of the hidden neurons discarding the characteristics of the dataset. This is one of the important goals of our work.

The overall value of the learning rate of the datasets is presented in Figure 3, which is formed by the values in the "*Average*" column of Table 2.

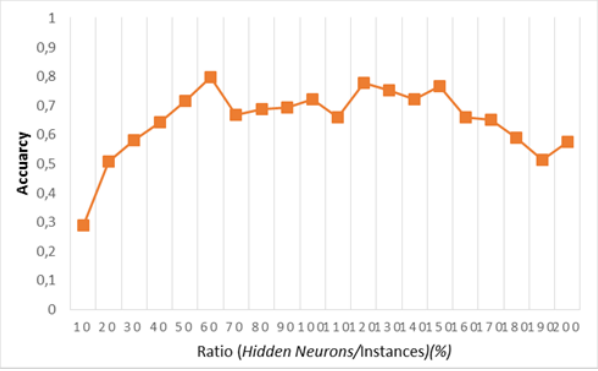

Figure 3. The performance of NAV for all datasets

In Figure 3, the highest accuracy is observed in the 60% percentile with 79.62% of accuracy value. This value is obtained by taking the mean of the NAV values of our datasets. This can be implemented for any other algorithm. This means that in a SLFN, if 60% of the instance number is taken as the number of hidden neuron number L, then you will probably get a higher learning rate on the average for any algorithm used.

Table 3 shows the comparison of a sample random selection. Due to the random selection, 70% value is chosen. You can do the selection however you like, even this selection can be out of the range of 10%-200%. Finally, the value to be compared is got by NAV, which is 60%.

In Table 3, the last column is the mean value of all datasets for related percentile. If we take the average value of all those mean values, we reach a general value including all datasets and their NAVs. When we examine Table 3, an increase of 12.82% of higher learning rate is achieved with our NAV formulation.

Table 3. Comparison of NAV with random selection

ID	Random	NAV	Ratio (%)
IRI	0,983	0,987	0,37
MK1	0,193	0,441	24,83
MK2	0,466	0,636	17,04
MK3	0,792	0,875	8,33
ION	0,941	0,941	0,00
VEH	0,622	1,000	37,84
WDB	0,368	0,271	-9,66
PID	0,793	0,886	9,29
WIS	0,859	0,899	3,91
CHS	0,253	1,000	74,68
WAV	0,927	0,837	-8,94
SPM	0,820	0,782	-3,86
		Average	12,82

Table 3 shows the main purpose of the study. When you implement NAV for determining the number L, the algorithm can reach higher learning values. This can be inferred regardless of the dataset since the normalized values have been used. The values presented in Table 3 are not real accuracy values; they are normalized and shows a relative comparison in the testbed.

In the study, a genetic algorithm based feature selection algorithm, which uses ELM, has been presented for experiments. NAV is an approach, a statistical method rather than being an algorithm. In other words, a comparison with another algorithm will not produce correct results. In order to get a better idea, a comparison with real accuracy values is presented in Table 4.

The accuracy results of four different algorithms are presented (*HGEFS, FSS, N2PS and NAV*). The first algorithm is hybrid genetic algorithm (GA) and extreme learning machine (HGEFS). It is a wrapper feature selection based on genetic algorithm and extreme learning

machine using SLFN as well in [1]. Second one is feature subset selection (FSS) is presented in [2] and one the most recent studies using SLFN. Third is Neural Network Pruning by Significance (N2PS), which prunes the insignificant neurons of a multi-layer neural network in [19]. Finally, NAV method uses Teaching-Learning-Based Optimization (TLBO) algorithm. It gets use of ELM and uses SLFN as stated in [16].

N2PS algorithm is chosen as a neural network having a multi-layer design as stated in [19]. It has been chosen for 2 main purposes. First, N2PS uses the same datasets in UCI [3] with a few exceptions.

Table 4. Comparison of single & multilayer algorithms

ID	HGEFS	FSS	N2PS	NAV
IRI	N/A	N/A	**0,9867**	**0,9867**
ION	0,913	0,89	**0,949**	0,926
VEH	**0,82**	N/A	N/A	0,662
WDB	**0,971**	0,963	N/A	0,606
PID	N/A	0,771	0,703	**0,773**
WIS	N/A	0,975	0,971	**0,976**
CHS	0,987	N/A	N/A	0,522
WAV	N/A	N/A	**0,855**	0,806
SPM	N/A	**0,893**	N/A	0,781

Secondly, N2PS works on multilayer feedforward neural networks, which means it has a multi-layer architecture, i.e. a deep neural network (DNN). This becomes popular among researchers and left as future work of this study. It will be informative for researchers to compare SLFN and DNN algorithms in a sample experimental setup.

It can be seen that the relatively highest values are bolded in Table 4. Since NAV finds the optimum hidden neuron number, found *best values* of TLBO algorithm is presented. Although algorithms using DNN have reported with better accuracy rates. Table 4 is formed due to the reported values of related algorithms and it can be inferred that NAV is a promising method in machine learning.

Additionally, NAV originates from statistical methods which means that it can be used with any algorithm preferred. The main goal of this study is to put forward an optimum hidden neuron number for a SLFN. That value can also be used with DNN as well.

5. CONCLUSION AND FUTURE WORKS

In this study, we examine the effect of hidden neurons in neural networks. We make comparisons with random selection and selecting with our method in order to show the effect of hidden neuron number in SLFNs. Then, we propose a new method from statistical analysis in order to increase the level of learning. This proposed method is the normalized averaged value of datasets, which is the main purposes of this study. We use a statistical method and get a normalized value. Finally, we observe that an improvement about 10-15% of better learning rates by only selecting the proper number of hidden neurons is obtained.

We believe that this study can form a base for other upcoming studies on this issue. The number of hidden neurons might vary due to the characteristic of the dataset as well. Even with large data sets, some other parameters may be adjusted in a way similar to ours.

As a future work, the selection of input parameters can be optimized due to any criterion that can increase the learning rate. In neural networks, there are many parameters that can affect the learning rate, and higher learning levels always deserve to be searched. Second future work can be the implementation of a DNN architecture for reaching better results.

REFERENCES

[1] X. Xue, M. Yao, Z. Wu, "A novel ensemble-based wrapper method for feature selection using extreme learning machine and genetic algorithm", *Knowledge and Information Systems*, 1-24, 2017.

[2] A. Deniz, H. E. Kiziloz, T. Dokeroglu, A. Coşar, "A Robust multi-objective evolutionary feature subset selection algorithm for binary classification using machine-learning techniques", *Neurocomputing*, 241, 128-146, 2017.

[3] Internet: UCI Irvine Machine Learning Repository, http://archive.ics.uci.edu/ml/datasets.html, October 2018.

[4] E. Sevinç, "Activation Functions in Single Hidden Layer Feed-forward Neural Networks", *Selçuk-Teknik Dergisi*, 1-13, 2018.

[5] Internet: ELM Classification web page, http://www.ntu.edu.sg/home/egbhuang/elm_random_hidden_nodes.html, October 2018.

[6] G. B. Huang, QY. Zhu, CK. Siew, "Extreme learning machine, Theory and applications", *Neurocomputing*, 70, 489-501, 2006.

[7] G. B. Huang, X. Ding, H. Zhou, "Optimization method based extreme learning machine for classification", *Neurocomputing*, 74, 155-163, 2010.

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

[9] H. C. Yuan, F. L. Xiong, X. Y. Huai, "A method for estimating the number of hidden neurons in feed-forward neural networks based on information entropy", *Computers and Electronics in Agriculture*, 40, 57-64, 2003.

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

[11] G. B. Huang, H. A. Babri, "Upper Bounds on the Number of Hidden Neurons in Feedforward Networks with Arbitrary Bounded Nonlinear Activation Functions", *IEEE Transactions on Neural Networks*, 9(1), January 1998.

[12] E. J. Teoh, K. C. Tan, C. Xiang, "Estimating the Number of Hidden Neurons in a Feedforward Network Using the Singular Value Decomposition", *IEEE Transactions on Neural Networks*, 17(6), November 2006.

[13] A. Akbaş, H. U. Yildiz, A. M. Ozbayoglu, B. Tavli," Neural network based instant parameter prediction for wireless sensor network optimization models", *Wireless Networks*, 1–14, 2018.

[14] M. Karakaya, E. Sevinç, "An Efficient Genetic Algorithm for Routing Multiple UAVs under Flight Range and Service Time Window Constraints", *Bilişim Teknolojileri Dergisi*, 10(1), 113, 2017.

[15] K. G. Sheela, S. N. Deepa, "Review on Methods to Fix Number of Hidden Neurons in Neural Networks", *Mathematical Problems in Engineering*, 2013, 2013.

[16] E. Sevinc, T. Dokeroglu, "A novel hybrid teaching learning based optimization algorithm for the classification of data by using extreme learning machines", *Turkish Journal of Electrical Engineering & Computer Sciences*, 27(2), 1523-1533, 2019.

[17] T. Dokeroglu, E. Sevinc, "Evolutionary parallel extreme learning machines for the data classification problem", *Computers & Industrial Engineering*, 130, 237-249, 2019.

[18] X. Zeng, D. S. Yeung, "Hidden neuron pruning of multilayer perceptrons using a quantified sensitivity measure", *Neurocomputing*, 69(7-9), 825-837, 2006.

[19] M. G. Augasta, T. Kathirvalavakumar, "A novel pruning algorithm for optimizing feedforward neural network of classification problems", *Neural processing letters*, 34(3), 241, 2011.

[20] C. R. Rao, S. K. Mitra, **Generalized Inverse of Matrices and its Applications**, John Wiley and Sons Inc., New York, USA, 1971.

[21] Y. Lan, Y. C. Soh, G. B. Huang, "Two-stage extreme learning machine for regression", *Neurocomputing*, 73, 3028-3038, 2010.

[22] C. Hamzaçebi, F. Kutay, "Durağan Zaman Serilerinin Yapay Sinir Ağları Ile Tahmininde Girdi Nöronu Ve Gizli Nöron Sayısının Belirlenmesi", *TÜİK İstatistik Araştırma Dergisi*, 4, 2005.

[23] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection", **IJCAI'95 Proceedings of the 14th international joint conference on Artificial intelligence**, vol. 2, 1137-1143, 1995.

[24] M. H. Calp, "İşletmeler için personel yemek talep miktarının yapay sinir ağları kullanılarak tahmin edilmesi", *Politeknik Dergisi*, 22(3): 675-686, (2019).