International Journal of Advanced Smart Convergence Vol.7 No.4 57-65 (2018)
http://dx.doi.org/10.7236/IJASC.2018.7.4.57

IJASC 18-4-7

Model Selection in Artificial Neural Network

Byung Joo Kim

School of Computer Engineering Youngsan University, Korea
bjkim@ysu.ac.kr

Abstract

Artificial neural network is inspired by the biological neural network. For simplicity, in computer science, it is represented as a set of layers. Many research has been made in evaluating the number of neurons in the hidden layer but still, none was accurate. Several methods are used until now which do not provide the exact formula for calculating the number of the hidden layer as well as the number of neurons in each hidden layer. In this paper model selection approach was presented. Proposed model is based on geographical analysis of decision boundary. Proposed model selection method is useful when we know the distribution of the training data set. To evaluate the performance of the proposed method we compare it to the traditional architecture on IRIS classification problem. According to the experimental result on Iris data proposed method is turned out to be a powerful one.

Keywords: *Model selection, Linear classifier, Classification boundary, Overfitting*

1. Introduction

Artificial neural network(ANN) is inspired by the biological neural network. For simplicity, in computer science, it is represented as a set of layers. These layers are categorized into three classes which are input, hidden, and output. Knowing the number of input and output layers and number of their neurons is the easiest part. Every network has a single input and output layers. The number of neurons in the input layer equals the number of input variables in the data being processed. The number of neurons in the output layer equals the number of outputs associated with each input. The hidden layer is the collection of neurons which has activation function applied on it as well as provide an intermediate layer between the input layer and the output layer. Many researches have been made in evaluating the number of neurons in the hidden layer but still none was accurate. Another question arises that how many hidden layers have to be used when dealing with the complex problem. If the data is linearly separable then there is no need to use more hidden layers. But in case of problems which deal with arbitrary decision boundary to arbitrary accuracy with rational activation functions then one has to use two or three hidden layer. Also the number of neurons that should be kept in each hidden layer need to be calculated. If the number of neurons are less as compared to the complexity of the problem data then underfitting may occur. Underfitting occurs when there are too few

Manuscript Received: Sep. 19, 2018 / Revised: Sep. 27, 2018 / Accepted: Oct. 5, 2018
Corresponding Author: bjkim@ysu.ac.kr
Tel: +82-55-380-9447, Fax: +82-55-380-9447
School of Computer Engineering Youngsan University, Korea

neurons in the hidden layers to adequately detect the signals in a complicated data set. If unnecessary more neurons are present in the network then overfitting may occur. Several methods are used utill now which do not provide the exact formula for calculating the number of hidden layer as well as number of neurons in each hidden layer. But the challenge is knowing the proper number of hidden layers and their neurons. However, there may be some hints to know in a classification problem. Following are the guidelines. (1) Based on the data, draw an expected decision boundary to separate the classes. (2) Express the decision boundary as a set of lines. Note that the combination of such lines must yield to the decision boundary. (3) The number of selected lines represents the number of hidden neurons in the first hidden layer. (4) To connect the lines created by the previous layer, a new hidden layer is added. Note that a new hidden layer is added each time you need to create connections among the lines in the previous hidden layer. (5) The number of hidden neurons in each new hidden layer equals the number of connections to be made. Paper is composed as follows. In section 2 we will apply above guidelines what we mentioned earlier and try to figure out the proper number of hidden layers and nodes on Iris data set. In section 3 geographical analysis of decision boundary is explained. Experimental result on proposed method is described in section 4, summary and future works will be described in section 5.

2. Previous Research

In this section we will review the previous research on choosing the proper number of hidden layer and nodes. Many researchers put their best effort in analyzing the solution to the problem that how many neurons are kept in hidden layer in order to get the best result, but unfortunately no body succeed in finding the optimal formula for calculating the number of neurons. Basically when dealing with the number of neurons in the input layer, one has to analyze about the data which is trained. For example, while dealing with handwritten numeral recognition using neural network for pin code recognition[1], the box size in which the pin number is written is taken under consideration for determining the number of neurons in the input layer. If the box size is 10x15 then 150 neurons must be taken as a input layer, so that every pixel contribute its value to the input layer individually. When the output layer is considered, then it depends on the chosen model configuration. If the neural network is applied to regression problem, then the output layer has a single node. Now the number of neurons in the intermediate layer is taken under consideration which is the motivation of this paper. Before that one fact should be kept in mind that if the data is linearly separable then there is not at all any use of hidden layer. Linear activation function can be directly implemented on the input and output layer. One hidden layer will be used when any function that contains a continuous mapping from one finite space to another. Two hidden layer can represent an arbitrary decision boundary to arbitrary accuracy with rational activation functions and can approximate any smooth mapping to any accuracy [2]. When multilayer perceptron is under consideration then the number of hidden layers and approximation of neurons in each hidden layer need to be calculated. Unnecessary increasing hidden layer may causes increase in the complexity of network. This is because in the weight balancing stage, weights between the first hidden layer and second hidden layer are also considered for weight updating which depends upon the error gap between the actual and target output. Usually some rule-of-thumb methods are used for determining the number of neurons in the hidden nodes. Multiple hidden layers are used in the applications where accuracy is the criteria and no limit for the training time is mentioned. Even the drawback of using multiple hidden layers in the neural network is that they are more prone to fall in bad local minima [3]. In [3] they try to optimize the number of neurons in the hidden layer using benchmark datasets and estimation of the signal-to- noise-ratio figure. The method utilizes a quantitative criterion based on the SNRF to detect overfitting automatically using the training error only, and it does not require a separate validation or testing set. This method reduces the overfitting problem

in the network. Rivals I. & Personnaz L.[4] used techniques based on least squares estimation and statistical tests for estimating the number of neurons in the hidden layer. Two approaches were used phase wise , first the bottom-up phase in which initially the number of neurons are increased up to an extent till the network becomes ill-conditioned. Afterwards in the next phase i.e. the top-down approach the neural network has to go through statistical Fisher tests. The second phase usually reduces the complexity of the neural network. F. Fnaiech in [5] make an attempt to prune the hidden nodes of the feed forward architecture by initially creating a non linear activation function of hidden nodes as Taylor's expansion and then NARX (nonlinear auto regressive with exogenous input) model is used. Afterwards nonlinear order selection algorithm proposed by Kortmann-Unbehauen most relevant signal of the NARX are selected and finally BPNN algorithm is used in order to prune the hidden nodes. Stuti Asthana and Rakesh K Bhujade in [1] made an approach to use two hidden layer for recognizing multiscript number recognition using artificial neural network on postcard, keeping accuracy as a chief criteria. More than 95% accuracy was obtained due to the use of two hidden layer. Moreover equal number of neurons are used in both layers so as to get best accuracy. This work has been tested on five different popular Indian scripts namely Hindi, Urdu, Tamil ,English and Telugu and satisfactory results were obtained under ideal condition. The decision method for selecting number of neurons to getting optimal solution using two phase method is describe in [6] by Kazuhiro Shin-ike. In two phase method the termination condition is same as the trial and error method but in new approach dataset is divided into four groups in which two groups of data are used in first phase to train the network and one group of remaining data set is used in second phase to train the network and last group of data set is used predict the output values of the trained network and this experiment is repeated for different number of neurons to get the minimum number of error terms for selecting the number of neurons in the hidden layer.

3. Theory : Geographical analysis of boundary decision

Choosing the proper number of hidden layer and number of neurons in each layer is not easy. However there may be some hints to know in a classification problem. There is a very interesting analysis of calculating the number of hidden layers and nodes in each layer[7][9]. We will briefly introduce the method. Following are the guidelines. (1) Based on the data, draw an expected decision boundary to separate the classes. (2) Express the decision boundary as a set of lines. Note that the combination of such lines must yield to the decision boundary. (3) The number of selected lines represents the number of hidden neurons in the first hidden layer. (4) To connect the lines created by the previous layer, a new hidden layer is added. Note that a new hidden layer is added each time you need to create connections among the lines in the previous hidden layer. (5) The number of hidden neurons in each new hidden layer equals the number of connections to be made. To make things clearer, let's apply the above guidelines for a number of examples. We will start with a simple and famous XOR problem shown in figure 1.

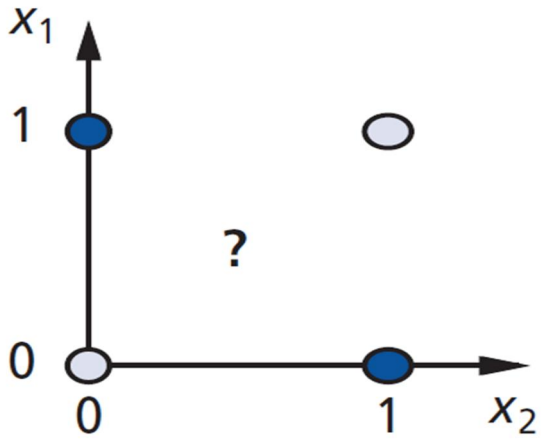

Figure 1. XOR problem

The first question to answer is whether hidden layers are required or not. A rule to follow in order to determine whether hidden layers are required or not is as follows: in artificial neural networks, hidden layers are required if and only if the data must be separated non-linearly. Looking at figure 2, it seems that the classes must be non-linearly separated. A single line will not work. As a result, we must use hidden layers in order to get the best decision boundary. In such case, we may still not use hidden layers but this will affect the classification accuracy. So, it is better to use hidden layers. Knowing that we need hidden layers to make us need to answer two important questions. Following the previous procedure, the first step is to draw the decision boundary that splits the two classes. There is more than one possible decision boundary that splits the data correctly as shown in figure 2. Following the guidelines, next step is to express the decision boundary by a set of lines. The idea of representing the decision boundary using a set of lines comes from the fact that any ANN is built using the single layer perceptron as a building block. The single layer perceptron is a linear classifier which separates the classes using a line created according to the following equation:

$$\mathrm{Y} = \mathrm{w}_1 x_1 + \mathrm{w}_2 x_2 + \cdots \mathrm{w}_n x_n + b \tag{1}$$

Where x_n is the input, w_n is its weight, **b** is the bias, and **Y** is the output. In this case, the decision boundary is replaced by a set of lines. The lines start from the points at which the boundary curve change direction. At such point, two lines are placed, each in a different direction.

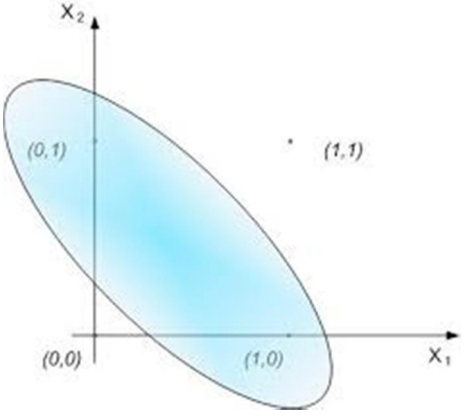

Figure 2. Nonlinear classifier is necessary in XOR problem

Because there is just one point at which the boundary curve change direction as shown in figure 3 by a donut shape circle, then there will be just two lines required. In other words, there are two single layer perceptron networks. Each perceptron produces a line.

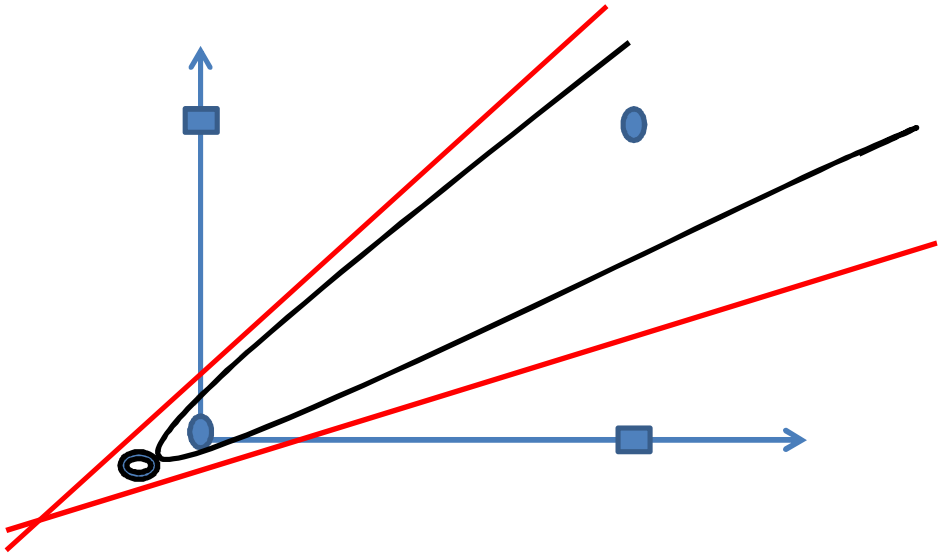

Figure 3. Each perceptron produces a line in XOR problem

Knowing that there are just two lines required to represent the decision boundary tells us that the first hidden layer will have two hidden neurons. Up to this point, we have a single hidden layer with two hidden neurons. Each hidden neuron could be regarded as a linear classifier that is represented as a line as in figure 3. There will be two outputs, one from each classifier (i.e. hidden neuron). But we are to build a single classifier with one output representing the class label, not two classifiers. As a result, the outputs of the two hidden neurons are to be merged into a single output. In other words, the two lines are to be connected by another neuron. Fortunately, we are not required to add another hidden layer with a single neuron to do that job. The output layer neuron will do the task. Such neuron will merge the two lines generated previously so that there is only one output from the network. After knowing the number of hidden layers and their neurons, the network architecture is now complete as shown in figure 4.

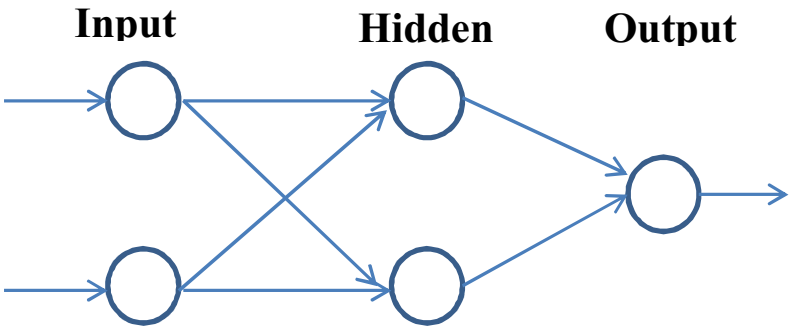

Figure 4. Classifier architecture proposed by guidelines

Another classification example is shown in figure 5. It is similar to the previous example in which there are two classes where each sample has two inputs and one output. The difference is in the decision boundary.

The boundary of this example is more complex than the previous example.

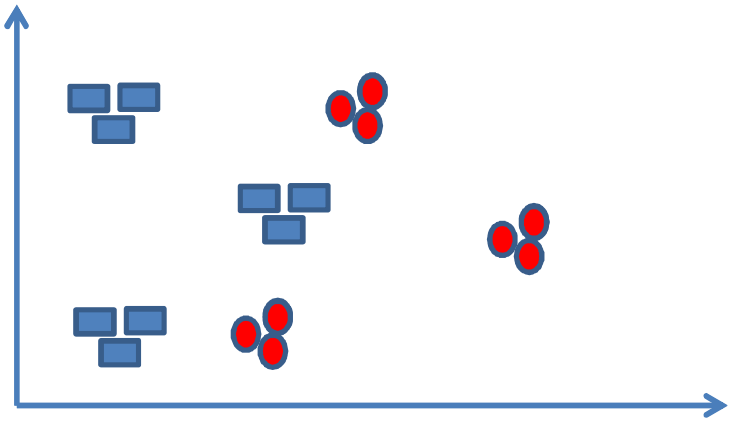

Figure 5. More complex problem

According to the guidelines, the first step is to draw the decision boundary. The decision boundary to be used in our discussion is shown in figure 6. The next step is to split the decision boundary into a set of lines, where each line will be modeled as a perceptron in the ANN. Before drawing lines, the points at which the boundary change direction should be marked as shown in figure 6.

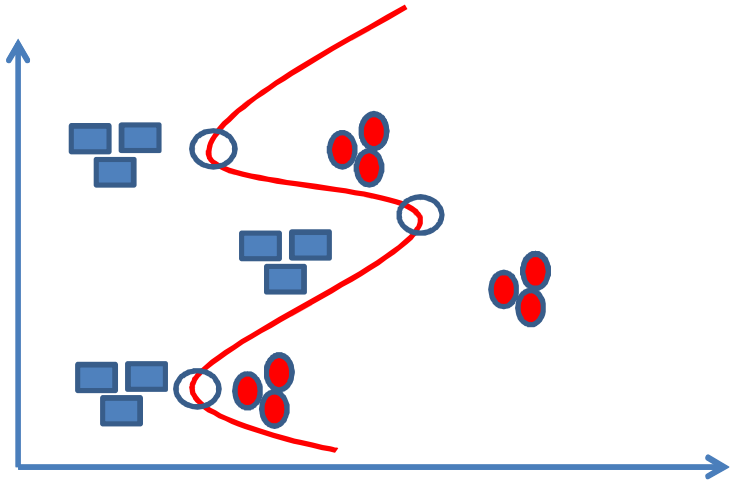

Figure 6. Classification boundary in more complex problem

Each of top and bottom points will have two lines associated to them for a total of 4 lines. The in-between point will have its two lines shared from the other points. The lines to be created are shown in figure 7. Because the first hidden layer will have hidden layer neurons equal to the number of lines, the first hidden layer will have 4 neurons. In other words, there are 4 classifiers each created by a single layer perceptron. At the current time, the network will generate 4 outputs, one from each classifier. Next is to connect these classifiers together in order to make the network generating just a single output. In other words, the lines are to be connected together by other hidden layers to generate just a single curve.

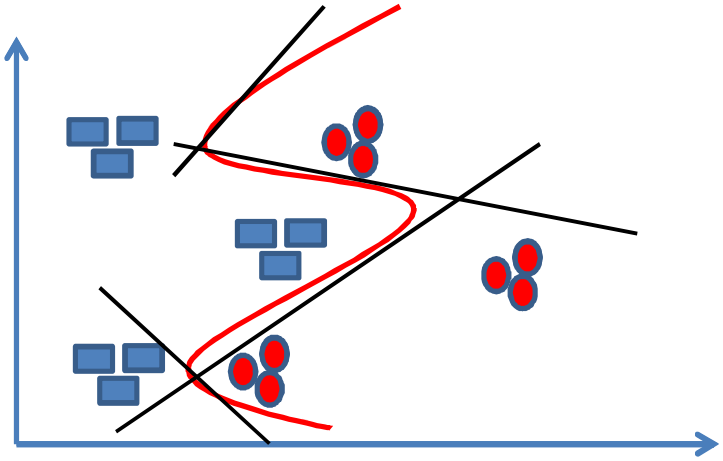

Figure 7. Four lines and 2 in-between lines are necessary to classify

It is up to the model designer to choose the layout of the network. One feasible network architecture is to build a second hidden layer with two hidden neurons. The first hidden neuron will connect the first two lines and the last hidden neuron will connect the last two lines. Up to this point, there are two separated curves. Thus there are two outputs from the network. Next is to connect such curves together in order to have just a single output from the entire network. In this case, the output layer neuron could be used to do the final connection rather than adding a new hidden layer. After network design is complete, the complete network architecture is shown in figure 8.

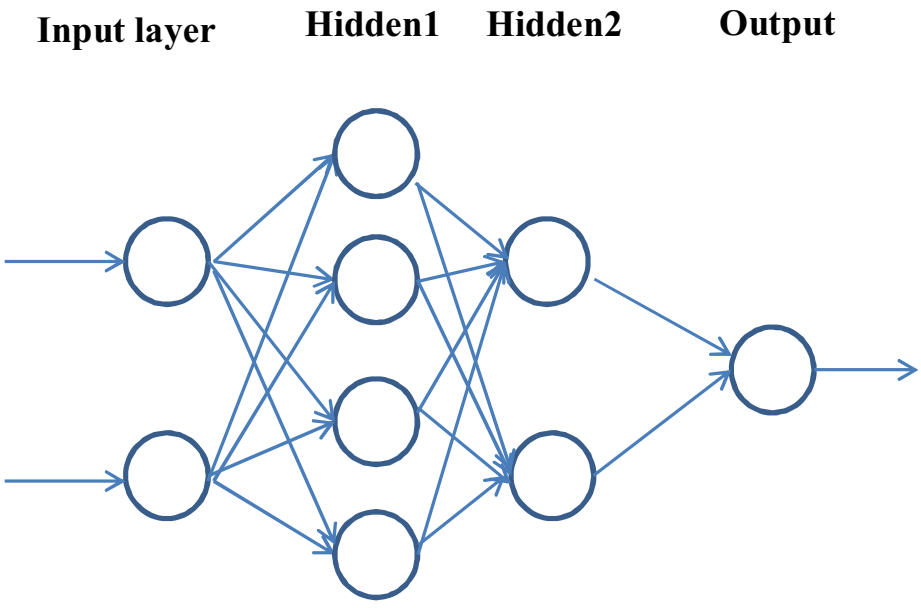

Figure 8. Generated network architecture

4. Experiments.

To evaluate proposed the method we apply it to the iris classification problem. Iris data set is one of the most popular and best known databases of the neural network application data set which is obtained from

UCI Machine Learning Repository and created by R.A. The IRIS dataset classifies three different classes of IRIS plant by performing pattern classification [8]. The IRIS data set includes three classes of 50 objects each, where each class refers to a type of IRIS plant. The attributed that already been predicted belongs to the class of IRIS plant. The list of attributes present in the IRIS can be described as categorical, nominal and continuous. The experts have mentioned that there isn't any missing value found in any attribute of this data set. The data set is complete. This project makes use of the well known IRIS dataset, which refers to three classes of 50 instances each, where each class refers to a type of IRIS plant. The first of the classes is linearly distinguishable from the remaining two, with the second two not being linearly separable from each other. The 150 instances, which are equally separated between the three classes, contain the following four numeric attributes: sepal length, sepal width, petal length petal width and the fifth attribute is the predictive attributes which is the class attribute that means each instance also includes an identifying class name, each of which is one of the following: IRIS Setosa, IRIS Versicolour, or IRIS Virginica. Figure 9 shows the distribution of the IRIS data according to the sepal length, sepal width, petal length petal width.

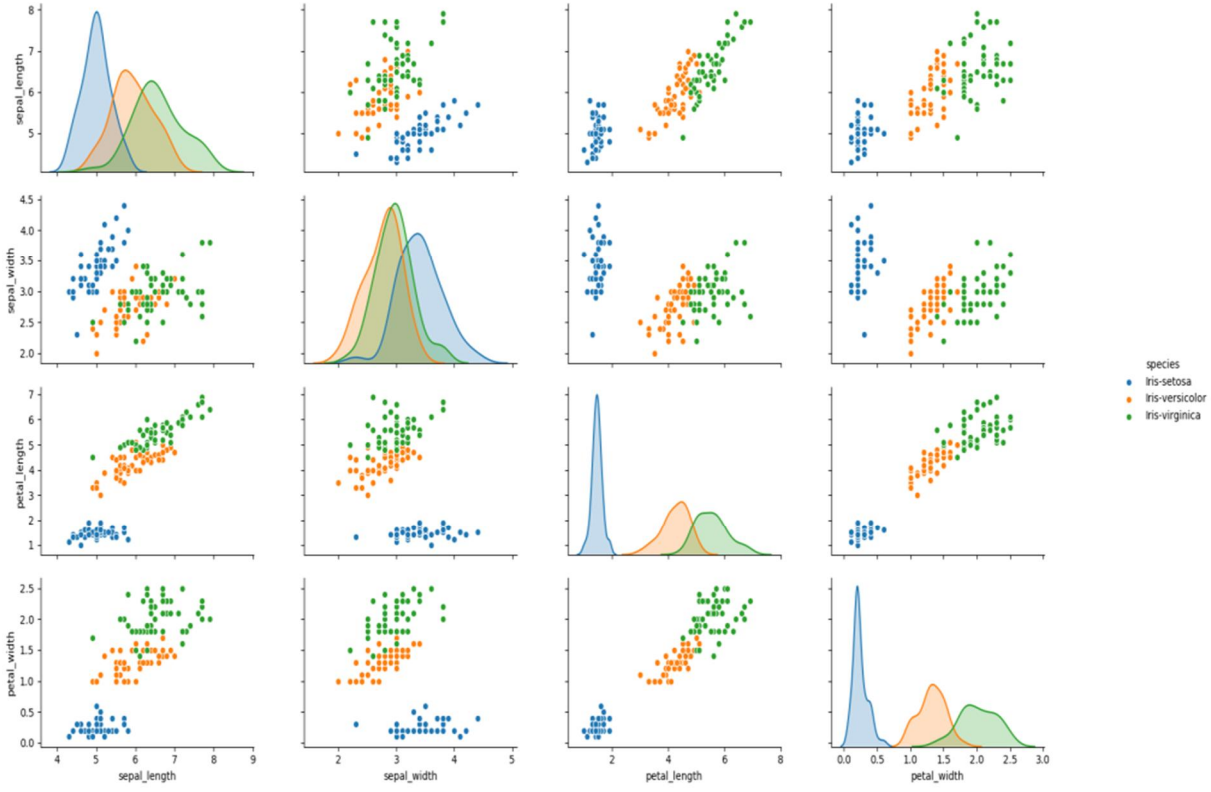

Figure 9. Distribution of IRIS data set according to each attribute

Applying the guidelines which were explained in section 3 we get the same architecture in more complex problem in figure 8 that is two hidden layers, 4 and 2 nodes in each layers. To test the behavior of proposed system, we compared the performance as traditional IRIS classification architecture.

Table 1. Experimental results on traditional method and proposed one

	Hidden-1	Hidden-2	Accuracy
Proposed architecture	4	2	97.33
Traditional architecture	16	0	96.00

5. Summary and Conclusion

In this paper, model selection approach was presented. Proposed model is based on geographical analysis of decision boundary. Proposed model selection method is useful when we know the distribution of the training data set. To evaluate the performance of the proposed method we compare it to the traditional architecture on IRIS classification problem. According to the experimental result proposed method is turned out to be a powerful one. Future work is applying proposed method to more complex problem.

Acknowledgment

This study was supported by a grant of Youngsan University(in 2018).

References

[1] S. Asthana, R.K. Bhujade, “Handwritten Multiscript Pin Code Recognition System having Multiple hidden layers using Back Propagation Neural Network”, Journal of Electronics Communication and Computer Engineering, Vol. 2, No. 1 2011

[2] http://www.heatonresearch.com/node/707

[3] Y. Liu, J. A. Starzyk, Z. Zhu, “Optimizing Number Of Hidden Neurons in Neural Networks”, http://www.ohio.edu/people/starzykj/network/Research/Papers/Recent%20conferences../Hidden%20Neurons%20AIA2007_549-204.pdf

[4] I.Rivals, L.Personnaz “A statistical procedure for determining the optimal number of hidden neurons of a neuralmodel”, Second International Symposium on Neural Computation (NC’2000), Berlin, May 23-26, 2000.

[5] F.Fnaiech, N.Fnaiech, M.Najim, “A new feedforward neural network hidden layer neuron pruning algorithm” IEEE International Conference on Acoustics, Speech, and Signal Processing, 2001.

[6] K. Shinike, “A Two Phase Method for Determining the Number of Neurons in the Hidden Layer of a 3-Layer Neural Network”, SICE Annual Conference 2010, August 18-21, 20 1 0, The Grand Hotel, Taipei, Taiwan.

[7] https://www.linkedin.com/pulse/beginners-ask-how-many-hidden-layersneurons-use-artificial-ahmed-gad?fbclid=IwAR2p Kz8hTj82n3XPt5v2LrVStMF2VZJPQRSjVlb_RfM1AZIup5es1mnWhc8

[8] https://archive.ics.uci.edu/ml/datasets/iris

[9] S.B. Lee, H.G. Kim, H.K.Seok, J.H. Nang, “Comparison of Fine-Tuned Convolutional Neural Networks for Clipart Style Classification” International Journal of Internet, Broadcasting and Communication(IJIBC), Vol.9 No.4, pp.1-7, 2017
DOI: https://doi.org/10.7236/IJIIBC.2017.9.4.1