Determining the Number of Hidden Layers in Neural Network by Using Principal Component Analysis

Muh. Ibnu Choldun R.(✉), Judhi Santoso, and Kridanto Surendro

School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Jl. Ganecha 10, Bandung, Indonesia
ibnucholdun@poltekpos.ac.id, judhi@stei.itb.ac.id, endro@informatika.org

Abstract. One of the challenges faced in the success of Deep Neural Network (DNN) implementation is setting the values for various hyper-parameters, one of which is network topology that is closely related to the number of hidden layers and neurons. Determining the number of hidden layers and neurons is very important and influential in DNN learning performance. However, up to now, there has been no guidance on it. Determining these two numbers manually (usually through trial and error methods) to find fairly optimal arrangement is a time-consuming process. In this study, we propose the method used for determining the number of hidden layers was through the number of components formed on the principal component analysis (PCA). By using Forest Type Mapping Data Set, based on PCA analysis, it was found out that the number of hidden layers that provide the best accuracy was three. This is in accordance with the number of components formed in the principal component analysis which gave a cumulative variance of around 70%.

Keywords: Neural network · Hidden layer · PCA

1 Introduction

Neural network has been successfully applied in various fields such as computer science, finance, medicine, engineering, physics, and so forth. The main reason is that neural network could approach arbitrary function. Over the past 30 years, a number of researches showed that artificial neural network called as feedforward network with one hidden layer can approach all functions arbitrarily [1–6]. One of important aspects for designing neural network is architecture or its topology that is closely related to the capability of generalization [7, 8]. Since the era of deep learning, the use of neural network that has more than one hidden layer is a research topic that caught researchers' interest. One of the challenges in the success of deep neural network implementation is setting values for various hyper-parameters, one of which is network topology that is closely related to the number of hidden layers and neurons. Determining the number of hidden layers and neurons is very important and influential in the performance of deep neural network [7, 8]. Therefore research to determine both of them is very necessary.

Y. Bi et al. (Eds.): IntelliSys 2019, AISC 1038, pp. 490–500, 2020.
https://doi.org/10.1007/978-3-030-29513-4_36

According to some literatures aforementioned, determining the number of hidden layers or optimal number of neurons still has no clear guideline. Moreover, the roles and functions of both are explained minimally. Some literatures propose methods or ways to determine the number of hidden layers or neurons, but they are not generally applied due to the type of input and output data [9, 10]. Studies in this area still leave difficult research tasks [11]. Some of the methods mentioned are apparently not applicable for different types of data. The earlier researchers determined the number of hidden layers or neurons based on their experience. However, less experienced researchers even did it based on 'trial and error'. One of the methods that can be applied practically is through data mining techniques, clustering and regression [12–14]. In this study, the researcher did not explain the relationship between the number of clusters formed by feature extraction that is going to be used in machine learning process.

Although it is a difficult area of research, determining the number of hidden layers and neurons should be carried out. This is because they greatly determine the deep neural network learning performance. As a first step, a method or technique is needed to determine the number of hidden layers. Hidden layers in the neural network in stages represent increasingly complex features, while the main components of the PCA represent the information content that exists. By assuming that the complexity of features is in line with the size of the feature information, then in this study we propose a method specifically for determining the number of hidden layers in a neural network based on the number of main components formed on principal component analysis (PCA).

2 Related Work

Studies compared the use of one or two hidden layers focused on univariate and multivariate functions [4–6, 15]. Thomas [4, 5] got different result that the use of two hidden layers applied to predictive functions showed better performance. Guliyev and Ismailov [6] concluded that the use of one hidden layer was less capable of approaching the multivariate function, so the use of two hidden layers showed better performance.

To determine the number of hidden layers and neurons at once, the method is divided into two: automatically and manually [15]. Automatic determination uses models, but it also can be done without models. Manual determination uses certain formulas, some are done by trial and error. Automatic determination without a model consists of grid search (GS) and random search (RS). GS is generally used to optimize DNN parameters only if the number is very low. First, users choose a range of values to explore. Then, DNN is set for each specification along with other parameters. Usually, GS is carried out for logarithmic scale steps that the best combination is estimated progressively to search [16]. The obvious disadvantage of GS is its time complexity – parameter k takes different values (by assuming that n is the same for all parameters), the complexity grows exponentially at level $O(n^k)$. Random search (RS) is an alternative to GS. It integrates faster into parametricization. RS is not adaptive that it is not dynamically updated during the experiment (the solutions found did not do anything to

the search). Furthermore, there is also hybrid algorithm that combines RS with other techniques to improve its performance, for instance, manual updates provided by experts [16, 17]. The technique that has been done turns out to be not generally accepted.

Model-based algorithm builds replacement models for hyper-parameter space. Then, hyper-parameters are set by using this model. Most of these techniques use Bayesian regression model that turns this problem into a trade-off between exploration (traversing unknown areas of space that classification of the performance is unknown), and exploitation (analyzing hyper-parameters that are likely to perform well and closed to "good" hyper-parameters that have been investigated). One of model-based algorithms is the use Particle Swarm Optimization (PSO) [3, 15]. PSO is an optimization technique done by continuously calculating potential solutions by using a quality reference. This algorithm optimizes problems by moving particles or prospective solutions for problems through certain functions for the position and velocity of the particles. The particles' movement is influenced by the best solution for the particles and the best solution in general that is obtained from other particles. This set of particles is called as a swarm that in the end it will move towards the best solution.

Although some researchers have proposed methods to calculate the number of hidden layers and neurons at once, but the methods were rarely used by other researchers, because they are difficult to apply or do not match the type of dataset used, therefore some researchers decided to use trial and error methods [8]. Therefore, a method or technique is needed to calculate the number of hidden layers or hidden neurons that can be generally accepted.

3 Neural Network

Architecture of back propagation neural network consists of several layers: input layer, hidden layer and output layer. Each neuron in each layer is connected to the next layer. The neurons in the input layer are connected to neurons in the hidden layer through synapses called as weight. In addition, hidden layers of neurons connected through synapses with neurons in the output layer. One of the neural network architectures is called as Multilayer Perceptron (MLP) [18]. MLP consists of: input layers where the neurons are fully connected to each neuron in the first hidden layer, each neuron in the hidden layer is connected to each neuron in the next hidden layer, and each neuron in the last hidden layer is fully connected with each neuron in the output layer. MLP, or often called as feedforward deep network, is a classic example of deep learning model.

Figure 1 is an example of a neural network with an MLP architecture consisting of input layers, two hidden layers, and an output layer. The closer the hidden layer to the output layer the better it identifies the complex features. For instance, MLP architecture that uses three hidden layers for face identification: the first hidden layer is used to identify the geometric shape that can distinguish the facial parts, the second hidden layer is used to identify eyes, nose, mouth, etc., and the third hidden layer is used to identify the face.

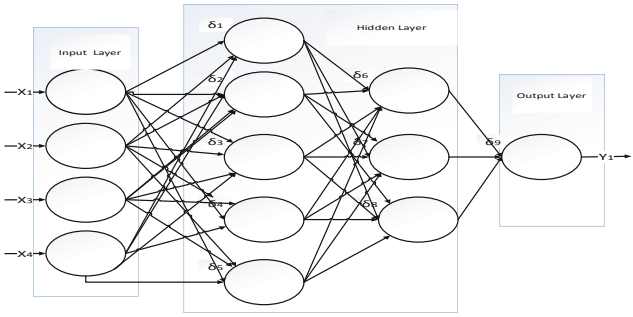

Fig. 1. Neural network

The training process for neural network is done by finding the value of each connection weight, which is done iteratively. Therefore, the output can correctly predict the target value of the training data examples. Backpropagation is an algorithm that is widely used for neural network learning, especially in MLP architecture. Each learning algorithm developed is based on inductive bias, a collection of assumptions that underlie the selection criteria of learning algorithm model. According to Mitchell [19], a learner's inductive bias is a set of additional assumptions that is sufficient to justify an inductive conclusion as a deductive conclusion. An example of an inductive bias that can be applied to MLP learning is to obtain the desired accuracy that the correct architecture must be chosen (the number of hidden layers and neurons for each hidden layer).

After obtaining the output from the perceptron based on the input given, at the error stage the perceptron calculation will be evaluated "whether the perceptron output matches the expected output or not". To calculate the difference between actual output and desired output can be used various functions of errors such as Squared Error (Eq. 1), Root Mean Squared Error (RMSE), or Cross Entropy (Eq. 2).

$$E = \frac{1}{2}\sum_{i=1}^{N}(t_i - y_i)^2 \tag{1}$$

for t = actual output, y = desired output, and N = number of data, then squared errors from t and y can be calculated using Eq. 1. The use of squared error often has problems if it is implemented in the case of classification with logistical output. Unlike the squared error, cross entropy does not experience problems in this case. In cross entropy, the greater the prediction error of the neural network, the greater the update value for the weight. The error cross entropy function can be formulated as follows:

$$E = -\frac{1}{N}\sum_{i=1}^{N} a_i \log(y_i) + (1 - a_i)\log(1 - y_i) \tag{2}$$

where y_i is the output of the neuron to i, a_i is the *actual output* to i, and N the total number of datasets.

4 Principal Component Analysis

Principal Component Analysis (PCA) is a multivariate analysis that transforms correlated origin variables into new variables that do not correlate with each other by reducing a number of the variables mentioned, so that the variables have smaller dimension but can explain most of the diversity of the origin variables [20]. The PCA steps are as follows.

1. There is a data matrix ($X = [x_1, x_2, \ldots, x_N]$), N is the total number of samples and xi represents the sample to i.
2. Calculate the average of all samples, as follows:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{3}$$

3. Subtract all data samples with the average, as follows:

$$D = \{d_1, d_2, \ldots, d_N\} = \sum_{i=1}^{N} x_i - \mu \tag{4}$$

4. Calculate the covariance matrix, as follows:

$$\sum = \frac{1}{N-1} D \times D^T \tag{5}$$

5. Calculate the eigenvectors V and eigenvalues λ from the covariance matrix (Σ)
6. Sort the eigenvectors based on the eigenvalues that correspond
7. Select eigenvectors which have the largest eigenvalues $W = \{v_1, \ldots, v_k\}$. The selected Eigenvectors (W) represent the PCA projection space.
8. All samples are projected on the dimensional space lower than PCA (W), as follows:

$$Y = W^T D \tag{6}$$

The number of the main components formed is equal to the number of original variables. The selection of the main components used was based on the eigenvalue value that the main components will be used if the eigenvalue is greater than one. However, the reduction (simplification) of the dimension can also be done by finding the percentage criteria of the data diversity explained by the first few main components. For example, the number of the main components used was if the cumulative variance of the components was at least 70%.

5 Research Method

5.1 Research Framework

The more hidden layers in a neural network the more they can represent the increasingly complex features [19]. Deep architecture in neural network can detect features

arranged in each layer. The lower layers (that are closer to the input layer) could detect simpler features, while the layers closer to the ouput layer could detect more complex features [21]. Increasingly complex feature represents higher information content received. The high information in PCA is represented in the main component that has high variance. In other words, increasingly complex feature in hidden layers of the neural network is in accordance with the main components that have higher variance. As a result, the number of hidden layers in the required neural network will be in accordance with the number of the main components in the principal component analysis. Therefore, in this study, the number of hidden layers in the neural network could be determined based on the number of the main components formed through PCA.

5.2 Preparing the Dataset

The data set used in this study was Forest Type Mapping Data Set that was converted into numeric data. The data sets could be downloaded on UCI Machine Learning Repository website[1]. The number of input attribute was 27 and the output category was 4 classes: 's' ('Sugi' forest), 'h' ('Hinoki' forest), 'd' ('Mixed deciduous' forest), 'o' ('Other' non-forest land). The number of dataset used was 325 records, 70% of which was used as training data and the remaining is for testing. The output function used in the data set was classification.

5.3 Standardization and Normalization

Standardization was done by deleting data that had empty attributes and also eliminating extreme values. Normalization was done with the aim of getting data with attributes that are scaled accordingly. The normalization done was Min-Maks with a value between 0 and 1. Min-Maks normalization formula is as follows:

$$v' = \frac{v - min_A}{max_A\_min_A}(new\_max_A - new\_min_A) + new\_min_A \tag{7}$$

v′ = new data, v = old data, $\min_A$ = the smallest value of an attribute, $\max_A$ = the biggest value of an attribute, new_min_A = the smallest value of a new attribute (=0), new_max_A = the biggest value of a new attribute (=1).

5.4 Determining Neural Network Topology

Changing network topology was done by varying the number of hidden layers by referring to the number of the components formed in PCA. The number of hidden neurons in each hidden layer is changed by trial and error.

[1] UCI Machine Learning Repository Homepage, https://archive.ics.uci.edu/ml/datasets/Forest+type+mapping, last accessed 2019/01/10.

5.5 Training and Testing

Training was carried out with 70% data from the dataset, while testing was carried out with 30% data from the dataset. The training and testing process would stop when the iteration had reached 500 times.

6 Result and Discussion

6.1 Principal Component Anaysis Result

By analyzing the main components of the input attributes on the dataset that was going to be used for neural network training, the following results were obtained (see Table 1):

Table 1. Principal component analysis processing results

Component	Total	% of variance	Cumulative %
1	10.146	37.578	37.578
2	5.014	18.571	56.149
3	3.772	13.971	70.120
4	1.980	7.333	77.453
5	1.501	5.560	83.013
6	1.328	4.919	87.932
7	.875	3.240	91.172
8	.621	2.301	93.473
9	.511	1.893	95.366
10	.368	1.365	96.731
11	.318	1.179	97.910
12	.176	.652	98.561
13	.112	.416	98.977
14	.069	.256	99.233
15	.066	.244	99.477
16	.036	.132	99.609
17	.030	.112	99.721
18	.028	.104	99.826
19	.022	.081	99.907
20	.011	.042	99.948
21	.007	.026	99.974
22	.003	.013	99.987
23	.002	.009	99.996
24	.001	.003	99.999
25	.000	.001	100.000
26	8.995E−5	.000	100.000
27	2.354E−5	8.718E−5	100.000

If a minimum total diversity of 70% is taken, the number of hidden layers is three, if 80% of the total diversity of hidden layers is five, and if the total diversity is 90%, the number of hidden layers is seven. The number of components that will be used as the basis for determining the number of hidden layers in a neural network is chosen from one to four.

6.2 Neural Network Training Result

Neural network training was done by using multilayer perceptron architecture with the number of hidden layers chosen from one to four, while the number of neurons for each hidden layer was determined by the researcher. With the number of hidden layers selected and the number of neurons in each hidden layer determined, the neural network architecture to be trained can be seen in the following Table 2.

Table 2. The architecture of neural network

Architecture	Number of hidden layer	Number of neuron
I	1	36
II	1	100
III	2	15, 8
IV	3	15, 10, 5
V	3	20, 10, 5
VI	4	20, 15, 10, 5

The researcher gave an example as an explanation of the architecture in the above table that if architecture IV is a multilayer perceptron with three hidden layers, the first hidden layer has 15 neurons, the second hidden layer has 10 neurons, and the third hidden layer has 5 neurons. The training and testing were carried out on each of the above architectures, and the following results were obtained (see Table 3):

Table 3. The accuracy of neural network

Architecture	Accuracy
I	90.45%
II	89.17%
III	87.90%
IV	92.36%
V	92.36%
VI	27.39%

For architecture IV, the accuracy obtained was 92.36% which means that from the test data used, about 7.64% of the output was classified as incorrect.

6.3 Discussion

The experiments carried out used several different network topologies, both in terms of the number of hidden layers and neurons in each hidden layer, each of which has a certain level of accuracy by using the Forest Type Mapping dataset. From the experiments performed, it was shown that by using one hidden layer by varying the number of neurons 36(I) and 100(II), the accuracy were 90.44% and 89.18%, respectively. In the architecture of one hidden layer, the addition of the number of neurons did not increase the accuracy. For the use of two hidden layers, architecture III had an accuracy rate of 87.89% that also did not improve the accuracy compared to the use of one hidden layer. By using three hidden layers, architecture IV and V had an accuracy rate of 92.36%, which means that the use of three hidden layers was more accurate than the use of one or two hidden layers. The use of four hidden layers, in architecture VI, decreased the accuracy.

The experiments showed that the use of three hidden layers provided the best level of accuracy. If it is associated with PCA result, where it is assumed that the number of hidden layers selected was in accordance with the number of the components selected in PCA, the cumulative variance of the selected component was around 70%. This result indicated that the cumulative variance classification needed to determine the number of components does not have to be close to 100%. The classification of a group of different outputs, but still within a certain range, will be grouped in the same classification. It would be different if the regression objective function of each different output was considered as a different value.

7 Conclusion and Future Work

The right number of hidden layers chosen in neural network would provide high learning accuracy. Increasing the number of hidden layers does not guarantee an increase in their accuracy. The determination of the number of hidden layers could be done by using the number of components produced on the PCA principal component analysis by considering the cumulative variance. For the classification function, the number of cumulative variances could be done with a cut off value of 70%. For future works, the researchers need to consider the way to determine the number of neurons in each hidden layer besides determining the number of hidden layers. In addition, determining the number of hidden layers by using the number of PCA components also needs to be applied for the regression objective function. In the research that has been done, each neural network topology chosen is only conducted one-time experiments, in future studies for each topology the experiment repetition must be performed for example at least ten times, then each topology is compared. Experiments also need to be done for different types of datasets.

References

1. Madhiarasan, M., Deepa, S.N.: A novel criterion to select hidden neuron numbers in improved back propagation networks for wind speed forecasting. Appl. Intell. **44**(4), 878–893 (2016). https://doi.org/10.1007/s10489-015-0737-z
2. Madhiarasan, M., Deepa, S.N.: Comparative analysis on hidden neurons estimation in multi layer perceptron neural networks for wind speed forecasting. Artif. Intell. Rev. **48**(4), 449–471 (2017). https://doi.org/10.1007/s10462-016-9506-6
3. Qolomany, B., Maabreh, M., Al-Fuqaha, A., Gupta, A., Benhaddou, D.: Parameters optimization of deep learning models using particle swarm optimization. In: 13th International Wireless Communications and Mobile Computing Conference (IWCMC), pp. 1285–1290 (2017). https://doi.org/10.1109/IWCMC.2017.7986470
4. Thomas, A.J., Petridis, M., Walters, S.D., Gheytassi, S.M., Morgan, R.E.: Eng. Appl. Neural Netw. **744**, 279–290 (2017). https://doi.org/10.1007/978-3-319-65172-9
5. Thomas, A.J., Walters, S.D., Gheytassi, S.M., Morgan, R.E., Petridis, M.: On the optimal node ratio between hidden layers: a probabilistic study. Int. J. Mach. Learn. Comput. **6**(5), 241–247 (2016). https://doi.org/10.18178/ijmlc.2016.6.5.605
6. Guliyev, N.J., Ismailov, V.E.: On the approximation by single hidden layer feedforward neural networks with fixed weights. Neural Netw. **98**, 296–304 (2018). https://doi.org/10.1016/j.neunet.2017.12.007
7. Nitta, T.: Resolution of singularities introduced by hierarchical structure in deep neural networks. IEEE Trans. Neural Netw. Learn. Syst. **28**(10), 2282–2293 (2017). https://doi.org/10.1109/TNNLS.2016.2580741
8. Koutsoukas, A., Monaghan, K.J., Li, X., Huan, J.: Deep-learning: investigating deep neural networks hyper-parameters and comparison of performance to shallow methods for modeling bioactivity data. J. Cheminform. **9**(1), 1–13 (2017). https://doi.org/10.1186/s13321-017-0226-y
9. Bunjongjit, S., Ngaopitakkul, A., Pothisarn, C., Jettanasen, C.: Improvement to reduce training time of back-propagation neural networks for discrimination between external short circuit and internal winding fault. In: International Conference on Information Science, Electronics and Electrical Engineering, Sapporo, pp. 614–618 (2014)
10. Chhachhiya, D., Sharma, A., Gupta, M.: Designing optimal architecture of neural network with particle swarm optimization techniques specifically for educational dataset. In: 7th International Conference on Cloud Computing, Data Science and Engineering - Confluence, Noida, pp. 52–57 (2017)
11. Lee, S., Ha, J., Zokhirova, M., Moon, H., Lee, J.: Background information of deep learning for structural engineering. Arch. Comput. Methods Eng. **25**(1), 121–129 (2018). https://doi.org/10.1007/s11831-017-9237-0
12. Tej, M.L., Holban, S.: Comparative study of clustering distance measures to determine neural network architectures. In: IEEE 12th International Symposium on Applied Computational Intelligence and Informatics (SACI), pp. 189–194 (2018)
13. Tej, M.L., Holban, S.: Determining optimal neural network architecture using regression methods. In: International Conference on Development and Application Systems (DAS), pp. 180–189 (2018)
14. Tej, M.L.: Neural network architecture through data mining techniques. In: Proceedings of the IIER International Conference, Bucharest, Romania (2017)

15. Lorenzo, P.R., Nalepa, J., Kawulok, M., Ramos, L.S., Pastor, J.R.: Particle swarm optimization for hyper-parameter selection in deep neural networks. In: Proceedings of the 2017 on Genetic and Evolutionary Computation Conference - GECCO 2017, vol. 8, pp. 481–488 (2017). https://doi.org/10.1145/3071178.3071208
16. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**, 281–305 (2012)
17. Larochelle, H., Erhan, D., Courville, A., Bergstra, J., Bengio, Y.: An empirical evaluation of deep architectures on problems with many factors of variation. In: ICML (2007)
18. Bengio, Y., Goodfellow, I., Courville, A.: Deep Learning. MIT Press, Cambridge (2016)
19. Mitchell, T.M.: Machine Learning. McGraw-Hill, Boston (1997)
20. Jollife, I.T.: Principal Component Analysis. Springer, New York (2002)
21. Di, W., Bhardwaj, A., Wei, J.: Deep Learning Essentials. Packt Publishing, Birmingham (2018)