OPEN ACCESS Asian Journal of Scientific Research

ISSN 1992-1454
DOI: 10.3923/ajsr.2020.132.139

Research Article

Incremental Contribution Method to Determine the Size of Neural Network Optimized by Genetic Algorithm

Budi Warsito, Hasbi Yasin and Rukun Santoso

Department of Statistics, Diponegoro University, Semarang, Indonesia

Abstract

Background and Objective: Determining the optimal architecture and optimization method in neural network for time series modeling have been interesting open problem in recent years. Several studies have been developed to solve this problem, but only to a limited extent. Determination of optimal size with a specific method is only done in the search for weights with standard methods, as well as the use of heuristic optimization without including how to determine the optimal architecture. This paper focuses on the determining optimal architecture in neural network for time series model optimizing by heuristic optimization. **Materials and Methods:** In this method, the network built first with the big size and then the cells with low contribution, expressed by the R^2_{inc}, will be removed from network. To get an approach of global optimum, genetic algorithm was used as optimization method to obtain the optimal weights. This model was applied to the rainfall data. Starting with eight hidden units, NN (1,2,18,5) was chosen as the optimal size of the network, i.e., a network consists of lags 1, 2 and 18 as input and has 5 hidden units. Removing 3 units in the hidden layer is not too large reducing R^2. **Results:** The proposed procedure successfully reduced the size of the network so that the constructed model was simpler. **Conclusion:** The use of incremental contribution method can effectively reduce network size on neural network optimized by genetic algorithm and is no longer just based on trial and error techniques.

Key words: Neural network, architecture, incremental contribution, genetic algorithm, optimal weights, time series, rainfall

Citation: Budi Warsito, Hasbi Yasin and Rukun Santoso, 2020. Incremental contribution method to determine the size of neural network optimized by genetic algorithm. Asian J. Sci. Res., 13: 132-139.

Corresponding Author: Budi Warsito, Department of Statistics, Diponegoro University, Semarang, Indonesia

Competing Interest: The authors have declared that no competing interest exists.

Data Availability: All relevant data are within the paper and its supporting information files.

INTRODUCTION

There is a lot of flexibility in constructing Neural Network model, especially in the field of time series. The flexibility of this model makes it have more open issues to be further researched. How to determine the optimal architecture includes input model, number of hidden units, the activation function and the optimization method used in the training process to obtain the optimal weights are some of them. Many studies have developed methods used to obtain optimal architecture. In general, the architecture is not known and has to be determined heuristically[1]. Several approaches have been taken but none has been shown to be applicable in general. On the other hands, they are depending on complex parameter selection and fine-tuning[1]. In general, there are two main methods, the first is the general to specific that is started with a complex model and it will be reduced to the simpler model with a certain algorithm and the second is the contrary, i.e., specific to general. Some studies that belong to general to specific are Optimal Brain Surgeon and Optimal Brain Damage proposed by Hassibi and Stork[2] and LeCun *et al.*[3], respectively, Magnitude Base Pruning method developed by Hagiwara[4] and Pruning Neuroplasticity by Wagarachchi and Karunananda[5]. In these methods, the complex architecture is built first and then the units with low saliency will be removed to get simpler architecture. Another type of general to specific class is Incremental Contribution method developed by Kaashoek and Van Dick[6] that proposed a way to obtain optimal hidden unit and input unit by constructing a complex architecture, then the units with no significant weight will be removed from network result as a simple network. In this case, the removed unit is regarded not giving real contribution to the result. The analysis of this method is by using R^2 incremental quantity and graphic method. In this procedure, the optimization method used to estimate the weights is conventional method.

The next question is whether the optimum value is real global optimum or just local optimum. Some optimization methods are, often unable to get the optimal global. Besides, the optimization methods used are difficult to determine the initial value to get convergence and need trial and error. Even a run from a starting point close to the global optimum is practically not guaranteed to converge[7]. Genetic Algorithm (GA) is one of the approaches to get optimum global based on evolution theory. There are some reasons why the using of genetic algorithm is useful to train neural network[8]. The first is the peculiarity of genetic algorithm that is efficient in a large and complex searching space to get an approach of global optimum, especially if the objective function has some local optimum. Because the complexity of the searching space grows, the algorithm can be an alternative to the technique based on gradient method as back propagation. The second advantage is its generality. Genetic algorithm can be used to train all kinds of network with different transfer functions than sigmoid, which are discontinuous and hence not trainable by gradient techniques, just with a little minor modification. Furthermore, genetic algorithm can be used to optimize not only the weights and bias but also combination of weights, bias, topology and the transfer function[8]. The type of generality given by genetic algorithm is allowed to use arbitrary evaluation function. The third reason using genetic algorithm for learning neural network is that the algorithm is an important method used in nature. Weights in neural network are genetically essential and therefore learning method is used as well as the process of natural selection. This superiority makes genetic algorithm be used as optimization technique in NN model to obtain optimal weights. Some papers that already used genetic algorithm as optimization method to obtain weights in NN for time series Giovanis[9], Mahajan and Kaur[10], Idrissi *et al.*[11] and Chung and Shin[12]. In this paper, the incremental contribution method is used to determine the optimal architecture of the NN model which the weights are estimated by genetic algorithm.

MATERIALS AND METHODS

Data: The data used in this section is the ten daily rainfall data of ZOM 136 Cokro Tulung, Klaten, Central Java Indonesia from January, 2010 until July, 2018 with the length of 309. The network inputare three variables i.e., lags 1, 2 and 18.The eight hidden units are built first, i.e., NN(1,2,18;8). Logistic sigmoid and linear are the activation functions used in hidden layer and output layer, respectively. The specification of genetic algorithm used to estimate weights are: population size = 20, probability of crossover $p_c = 0.7$, probability of mutation $p_m = 0.1$ and the number of generations = 10000. Selection of the parents couple is using roulette wheel selection method.

Optimization of weights and architecture of neural network: Neural Network model were examined specifically in this paper is Feed Forward Neural Network (FFNN). Architecture of FFNN model for forecasting time series data with the configuration X_{t-1}, X_{t-2}, ..., X_{t-p} as input units and a constant (bias), one hidden layer with n neurons and one output unit is shown in Fig. 1. This FFNN model can be written as follows[13]:

$$x_t = w^b + \sum_{j=1}^{l} w_j^o f^h \left(w_j^b + \sum_{i=1}^{p} w_{ji}^h x_{t-i} \right) + \varepsilon_t \qquad (1)$$

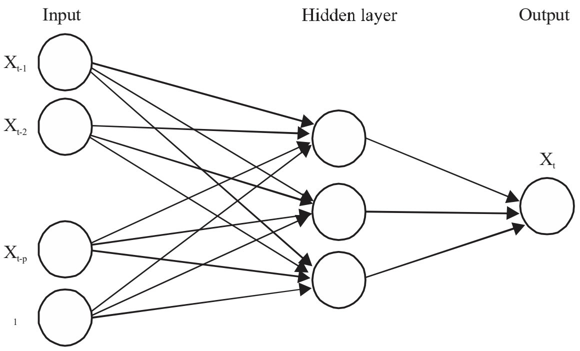

Fig. 1: Architecture of FFNN for forecasting the time series data with one hidden layer containing n neuron and the input variables at lags 1 until lag p

where, w^b_j is the weight between constant unit and neuron while w^b is the weight between constant and output unit. The symbols w^h_{ji} and w^o_j are weights connection from input to neuron and from neuron to output respectively. Both functions f^h is the activation function used at hidden layer. The notation used to write neural network model is NN (j_1, ..., j_k, n) that shows FFNN with input variables at lag j_1, ..., j_k and n neuron in one hidden layer.

The most widely used of training algorithm for feed forward neural network is back propagation[14]. It involves three stages: feed forward of the input pattern, the calculation and back propagation of error and adjustments the weights. At the first stage, every input unit receives an input signal x_i and passes it to the hidden units z_1, ..., z_p. Each hidden unit then calculates the activation and weighted summation is considered as:

$$z_{in_j} = \sum w_{ji} x_i + w_{bj} \tag{2}$$

where, x_i is the activation of the i-th input unit that sends the signal to hidden unit j and w_j is the weight of the signal sent and j = 1,2, ..., q is the number of hidden units, whereas w_{bj} is the weight from bias to hidden unit j. The sum was transformed by nonlinear activation function f(.) to get activation z_j from hidden unit j in the form:

$$z_j = f\left(z_{in_j}\right) \tag{3}$$

After all of the hidden units calculate the activation and then they send the signal z_j to the output unit. The output unit then calculates the activation to give response from the network to input pattern in the form:

$$g\,(w, z) = \Sigma\, w_j z_j + w_{bo} \tag{4}$$

The function at Eq. 4 is the output of the network:

$$y = \Sigma\, w_j\, f\,(a_j) + w_{bo} \tag{5}$$

where, w_{bo} is the weight from bias to the output unit. During the training process, output unit compares the calculated activation y with the target t to establish the error of the pattern.

Incremental contribution: The act of determining the size of a network can be done by analysis of incremental contribution proposed by Kaashoek and van Dick[6]. The explanation of the method can be described below. Consider a network with H cell (unit) at hidden layer. The network output will be as in Eq. 5 and the network performance quantity can be declared as squared of the correlation coefficient from y and ŷ:

$$R^2 = \frac{(\hat{y}'y)^2}{(y'y)(\hat{y}'\hat{y})} \tag{6}$$

where, ŷ is the vector of network output. The stage done compares the observed data with the network output in which one cell is removed to the one which all the cells are entered in the network. For instance, if the contribution from hidden cell h is put to zero (c_h = 0) then the network will produce the output $\hat{y}_{-h}$, with error:

$$e_{-h} = y - \hat{y}_{-h} \tag{7}$$

The squared of correlation coefficient from the network reduced by one cell in the hidden layer, that is correlation between y and $\hat{y}_{-h}$, is abbreviated as[6]:

$$R^2_{-h} = \frac{(\hat{y}'_{-h}\, y)^2}{(y'y)(\hat{y}'_{-h}\, \hat{y}_{-h})} \tag{8}$$

The incremental contributions of cell h can be explained as the difference of R^2 and R^2_{-h}:

$$R^2_{inc} = R^2 \text{-} R^2_{-h} \tag{9}$$

If the result of R^2_{inc} from cell h is lower than R^2_{inc} from all other cells then this cell is a candidate to be removed from the network. The cell with the lowest contribution is the first candidate for exclusion. From this pronouncement, if, a number of cell H and there are H' cells that have low R^2_{inc}

compared to all other cells, then the number of cells that were maintained in the network are H-H' cells. This value is an optimal number of hidden units in the network due to the cell with low R^2_{inc} basically does not have a significant contribution, in terms of the network with addition of the cell is not unduly affect network performance.

Reduction cells can also be done with graph analysis. Graph of {t, $\hat{y}_{-h}$ (t)} is compared with the graph of {t, y(t)} and this comparison will provide clues for the contribution of hidden cell h in explaining the variance of y(t). If the graph of {t, $\hat{y}_{-h}$ (t)} has shown a good prediction to {t, y(t)}, then cell h can be excluded from the network because the network without cell h has resulted a good prediction. In other words, the addition of cell h has no significant contribution to the network performance.

The same procedure can be applied to reduce the number of input layer cells. In this case, $\hat{y}_{-i}$ is network output without input cell i. The contribution of input cell i is put to zero ($A_{hi} = 0$, h = 1, 2, ..., H), then the reduced network can be quantified by the square of the correlation coefficient R^2_{-i} between y and $\hat{y}_{-i}$ as in Kaashoek and van Dick[6]:

$$R^2_{-i} = \frac{(\hat{y}'_{-h}\, y)^2}{(y'y)(\hat{y}'_{-i}\, \hat{y}_{-i})} \qquad (10)$$

The contribution of cell i is measured as R^2-R^2_{-i}. The relative value of incremental contributions in R^2 can be used in evaluating whether an input cell can be omitted or not.

Genetic algorithm: The genetic algorithm (GA) is a search heuristic that is routinely used to generate useful solutions to optimization and search problems[15,16]. This method works by imitating the mechanism of natural genetic, that finds the structure of the best gen in creature's body. The basic of genetic algorithm is evolution theory that explains base principal of the conceived species in the world. Species who are able to adapt better would have more chance to hold out. The stage of genetic algorithm began with determining a set of potential solution and adjusts it by some iteration to get the best solution. The set of potential solution is established earlier and called chromosome, that is formed randomly in the form of generated binary alphabet. All sets of observed chromosomes represent a population. The chromosomes will evolve in some stages of iteration called generation. Evolution process to get new generation (offspring) comprises selection, crossover and mutation process. In principal, this algorithm has several characteristics such as working a set of encoding parameters, not the parameter set itself; searching from a population of points, not just from a single point; using the information of objective function (fitness function), not derivative, using random operations with probabilistic adjustment rules, not operation with certain derivative rules at every iteration.

Genetic algorithm works with encodes candidate solutions in the form of chromosomes, each of has the same length and consists of elements such as symbols of the chosen set, for instance is binary alphabet. Each chromosome x corresponds to the fitness function f(x). The election of chromosome length, alphabet and coding that constitutes a mapping from Ω (set with certain universe of discourse) to the set of chromosomes is called representation scheme of a problem.

The first stage of genetic algorithm after initialization of early population P(0) is selection operation, that is forming mating pool set M(k) that many elements are equal to the number of elements in P(k). Each point $m^{(k)}$ in M(k) is taken from the points $x^{(k)}$ in P(k). The second stage is evolution in the form of cross-breeding operations (crossover) and mutation. Crossover operation took a pair of chromosomes as the parent that give birth a pair of new chromosomes(offspring). A parent couple is chosen from M(k) randomly with probability p_c. The completion of crossover is followed by the mutation process by changing (flipping) the value of one or more genes in a single chromosome. The mutation process occurred randomly with small probability, p_m<<1. The next stage is to select a chromosome to survive to the next generation. An aid strategy that can be used is elitism, i.e., by storing some good chromosomes that have been maintained in the next generation. Another useful strategy is linear fitness ranking (LFR) to scale fitness values obtained from the individual evaluations. LFR is done to avoid inclination to converge to the local optimum solution by the new fitness values obtained with the larger variance. This process is repeated until it gets a best fitness chromosome as best solution.

The stages of genetic algorithm can be simplified into the following:

- Shape the early population (set: k = 0 → P(0))
- Evaluate P(k)
- If stopping criteria is fulfilled then stop
- Select M(k) from P(k)
- Arrange M(k) to the form P(k+1)
- Back to stage 2 (set: k = k+1)

The stages can also be illustrated as shown in the flowchart[17].

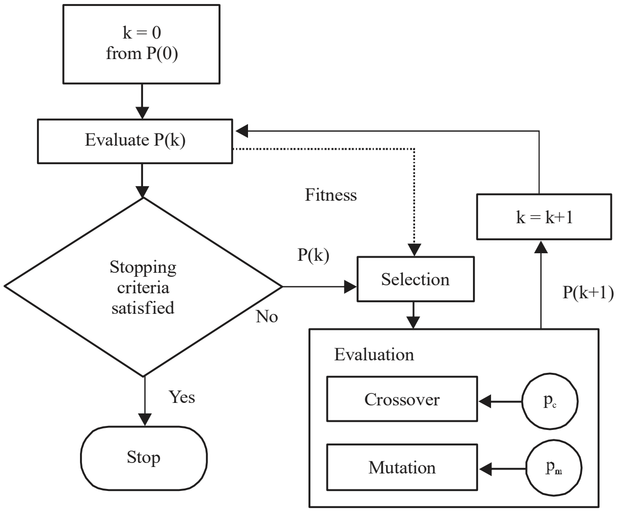

Fig. 2: Summarize of genetic algorithm

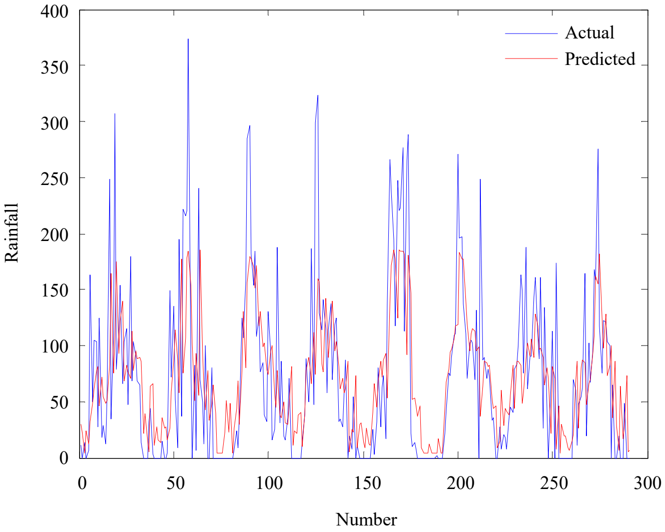

Fig. 3: Plot of actual and the prediction of NN (1,2,18;8) model

As shown in Fig. 2, some criteria that have been arranged are selection method, crossover technique, probability of crossover and mutation and the number of generations. Evaluation is done by using an error criteria. If stopping criteria has not been satisfied yet, selection process is carried out to choose two chromosomes as parent. Crossover and mutation are then be enforced for getting new generation. This processes are carried out until the fulfillment of the stop criteria. In this case, time series data X_t and FFNN model is built as in (1), the input variables are from lag 1 until lag p (X_{t-1}, X_{t-2}, ..., X_{t-p}), φ_n and φ_o are logistic sigmoid f(x)=1/(1+exp(-x)) and linear function y = x, respectively. The estimation of network weights w = (w_{bn}, w_{bo}, w_{in} and w_{no}) that are $\hat{w}$ = ($\hat{w}_{bn}$, $\hat{w}_{bo}$, $\hat{w}_{in}$ and $\hat{w}_{no}$) can be searched by genetic algorithm to minimize MSE.

RESULTS AND DISCUSSION

The result of simulation shows that the squared of correlation coefficient of this model is R^2 = 0.9864. The squared of correlation coefficient of the network which the hidden unit removed one cell h, i.e. R^2_{-h} and the contribution of each hidden cell h (h = 1,2, ..., 8), i.e., R^2_{inc} are shown in Table 1.

The list of Table 1 is the result of NN (1,2,18;8), i.e., the architecture which three input units and eight hidden units. R^2 of NN (1,2,18;8) refer to the squared of correlation coefficient of neural network model optimized by genetic algorithm. Values in the row of R^2_{-h} indicate the squared of correlation coefficients when the hidden unit is removed one by one and R^2_{inc} follows the equation 9. Hidden cells 3, 5 and 8 are the candidates to be removed from the network due to the R^2_{inc} of these cells are lower than the others. If cells 3, 5 or 8 is excluded from the network, still have a high values of R^2_{-3}, R^2_{-5} and R^2_{-8}, respectively. After deleting hidden cells 3, 5 and 8, there are still five cells that will be used. Figure 3 shows the plot of full 8 hidden units. By this composition, the initial architecture contains 41 weights that must be obtained. Figure 4 shows the plot after removing hidden unit number 1-8, respectively. Attention should also b paid to the results of these figures for visually checking the comparison of the plots after deleting cells with the previous.

In Fig. 4, the top plots are actual and the prediction of the series which removing hidden units 1 and 2, respectively. They are the most extreme values. After excluding the hidden unit 1 the prediction gave a bad result. So, the hidden unit 1 should be maintained in the model. The same condition is for second unit. On the contrary, exclusion of the 3rd unit doesn't greatly change goodness of the prediction results. It means that the third unit could be excluded from the model. The procedure was repeated until the last unit in the hidden layer. This was the last stage of the proposed procedure and it given NN (1,2,18;5) as an optimal size of the network, i.e., NN which has lags 1, 2 and 18 as input units and five hidden units. It means that the optimal network architecture chosen was not the smallest architecture and also not the greatest. This method could remove about 36.58% weights, i.e., from 41-26. This result was similar with Kavzoglu[18] which was the second smallest network considered, appeared to be the best choice. Similarly, in Wijayasekara *et al.*[19] the optimal architectures were networks with a number of hidden units in the middle of a number of experiments.

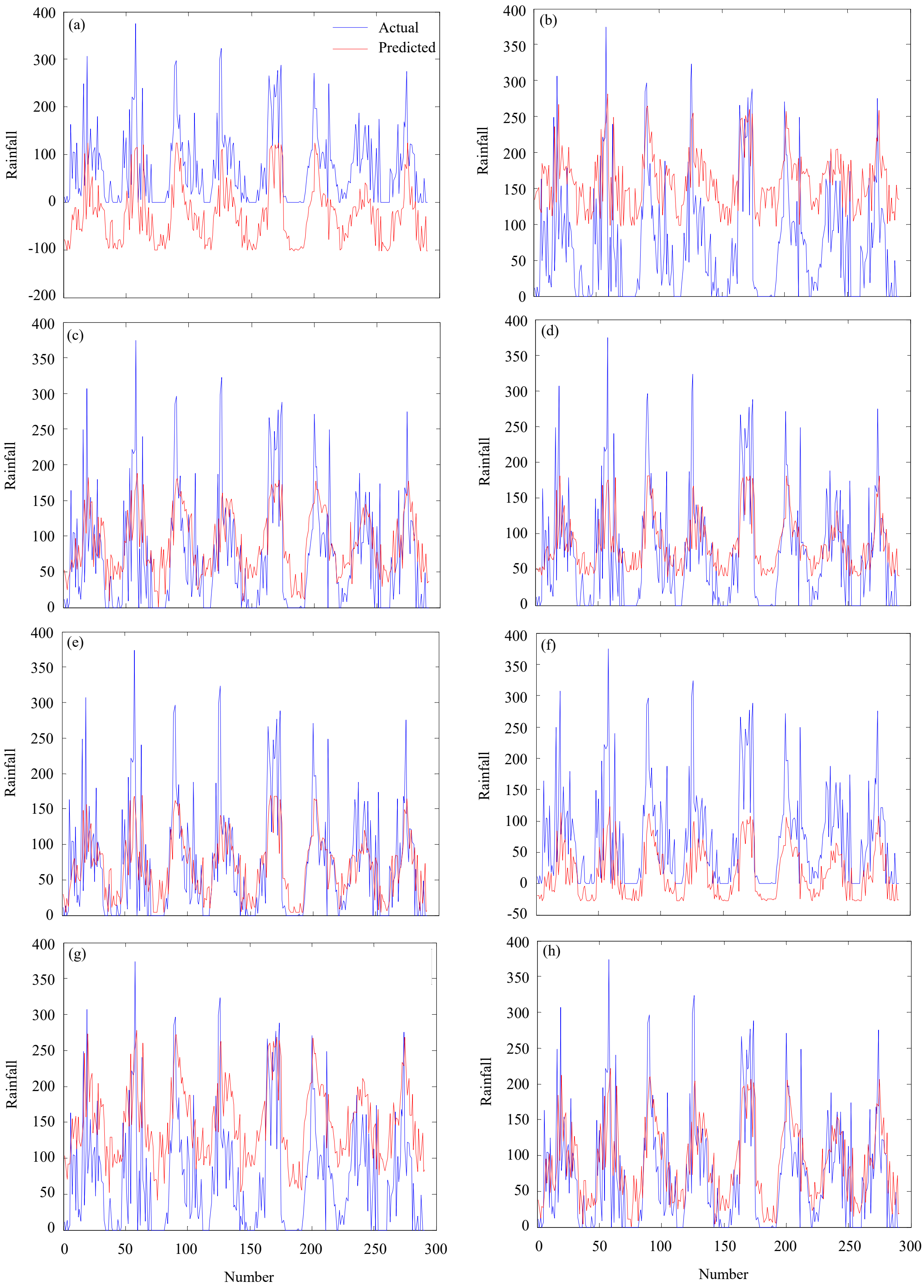

Fig. 4(a-h): Incremental contribution of input units of NN (1,2,18;8) and NN (1,2,18;1-8), plot actual vs. predicted of FFNN-GA without hidden units, (a) 1, (b) 2, (c) 3, (d) 4, (e) 5, (f) 6, (g) 7 and (h) 8

Table 1: Incremental contribution of hidden units of NN (1,2,18)

R^2 of NN (1,2,18;8) = 0.9864

Excluded hidden cell	-H1	-H2	-H3	-H4	-H5	-H6	-H7	-H8
R^2-$_h$	0.1304	0.1105	0.9782	0.6257	0.9703	0.4234	0.4421	0.9811
R^2_{inc}	0.8560	0.8759	0.0082	0.3607	0.0161	0.5730	0.5443	0.0053

In line with this research, Cantú-Paz[20] has compared several methods and successfully pruning between 30 and 50% of the total weights. It has used genetic algorithms as optimizers, but has been applied in the case of classification. On the other hand, Alshahrani *et al.*[21] has done pruning with a web-based tool, DANNP and has succeeded in reducing the amount of weight to 99%. However, the main focus of this study was on input reduction and is applied to the classification problem. The optimization used was also a gradient based method, not a heuristic. It was not discussed whether the method also succeeded in reducing such hidden units. In cases of reducing the network size, it is also interesting to study the affect of removing too many hidden units to the network's learning ability.

It has not considered the speed needed to do pruning with this method, as well as the change in speed from the initial architecture to the final architecture, which are limitations of this study. For developing, an automatic architecture design algorithm by using this procedure is important and interesting rule for speed up the process of choosing the optimal network. It was successfully conducted in traditional optimization[22]. It has also not yet pruned the neural network model with genetic optimization to reduce input and hidden layers simultaneously, which will increase the likelihood of reducing network weights. It would be interesting to compare various pruning methods in terms of increasing speed, successfully reduced weights, accuracy and its application in various types of data, not limited to seasonal data.

CONCLUSION

The using of incremental contribution to obtain the optimal size of Neural Network became a good progress in modeling NN so that the act of determining the optimal architecture is no longer just based on trial and error techniques. The graph analysis conducted also helps to decide which cells should be excluded from the network. The superiority of the use of genetic algorithm as optimization method to obtain the optimal weights in this procedure is the stability of the estimation result in each experiment. From some experiments done, in each step, the number of hidden layer cell excluded from network almost always the same, from each other. Moreover, in determining the cells used in the input layer, several experiments carried out always get the same results, i.e., the input units except lag 1 are excluded from the network. This procedure also could be adopted for various neural networks modelling which were use the other heuristic optimization methods.

SIGNIFICANCE STATEMENT

The main finding is how to obtain the optimal architecture on neural network model whose optimization uses the heuristic method, in this case is genetic algorithm. In much previous work, procedure for obtaining optimal architecture is limited to the gradient based methods. The proposed procedure was applied in the seasonal data which also very rarely discusses the selection of optimal architecture.

ACKNOWLEDGMENT

This research is supported by Directorate of Research and Community Service, Ministry of Research, Technology and Higher Education through the scheme of PDUPT and the contract number 101-24/UN7.P4.3/PP/2019.

REFERENCES

1. Kuri-Morales, A.F., 2014. The Best Neural Network Architecture. In: Nature-Inspired Computation and Machine Learning. MICAI 2014. Lecture Notes in Computer Science, Vol. 8857, Gelbukh, A., F.C. Espinoza and S.N. Galicia-Haro (Eds.)., Springer, Cham, ISBN: 978-3-319-13650-9, pp: 72-84.
2. Hassibi, B. and D.G. Stork, 1992. Second Order Derivatives for Network Pruning: Optimal Brain Surgeon. In: Advances in Neural Information Processing Systems 5, Hanson, S.J., J.D. Cowan and C.L. Giles (Eds.)., Morgan Kaufmann Publishers Inc., San Francisco, USA., pp: 164-171.
3. LeCun, Y., J. Denker, S. Solla, R.E. Howard and L.D. Jackel, 1990. Optimal Brain Damage. In: Advances in Neural Information Processing Systems 2, Touretzky, D. (Ed.). CA Morgan Kaufman, San Francisco, pp: 598-605.
4. Hagiwara, M., 1993. Removal of hidden units and weights for back propagation networks. Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan), October 25-29, 1993, IEEE., Los Alamitos.

5. Wagarachchi, M. and A. Karunananda, 2017. Optimization of artificial neural network architecture using neuroplasticity. Int. J. Artif. Intell., 15: 112-125.
6. Kaashoek, J.F. and H.K. van Dick, 2002. Neural network pruning applied to real exchange rate analysis. J. Forecast., 21: 559-577.
7. Gilli, M. and E. Schumann, 2010. A note on 'good starting values' in numerical optimisation. https://comisef.eu/files/wps044.pdf
8. Montana, D.J., 1995. Neural Network Weight Selection Using Genetic Algorithms. In: Intelligent Hybrid System, Goonatilake, S. and S. Khebbal (Eds.)., John Wiley and Sons, New York, pp: 85-104.
9. Giovanis, E., 2010. Application of feed-forward neural networks autoregressive models in gross domestic product prediction. World Acad. Sci. Eng. Technol. Int. J. Social Behav. Educ. Econ. Bus. Ind. Eng., 4: 418-422.
10. Mahajan, R. and G. Kaur, 2013. Neural networks using genetic algorithms. Int. J. Comput. Applic., 77: 6-11.
11. Idrissi, M.A.J., H. Ramchoun, Y. Ghanou and M. Ettaouil, 2016. Genetic algorithm for neural network architecture optimization. Proceedings of the 3rd International Conference on Logistics Operations Management (GOL), Fez, Morocco, May 23-25, 2016, IEEE., pp: 1-4.
12. Chung, H. and K.S. Shin, 2018. Genetic algorithm-optimized long short-term memory network for stock market prediction. Sustainability, Vol. 10, No. 10. 10.3390/su10103765.
13. Wang, X. and M. Meng, 2012. A hybrid neural network and ARIMA model for energy consumption forcasting. J. Comput., 7: 1184-1190.
14. Konomi, M. and G.M. Sacha, 2014. Influence of the learning method in the performance of feedforward neural networks when the activity of neurons is modified. https://arxiv.org/ftp/arxiv/papers/1404/1404.5144.pdf
15. Bajpai, P. and M. Kumar, 2010. Genetic algorithm-an approach to solve global optimization problems. Indian J. Comput. Sci. Eng., 1: 199-206.
16. Mishra, S., S. Sahoo and M. Das, 2017. Genetic algorithm: An efficient tool for global optimization. Adv. Comput. Sci. Technol., 10: 2201-2211.
17. Chong, E.K. and S.H. Zak, 2013. An Introduction to Optimization. Vol. 76. John Wiley and Sons, New York.
18. Kavzoglu, T., 1999. Determining optimum structure for artificial neural networks. Proceedings of the 25th Annual Technical Conference and Exhibition of the Remote Sensing Society, September 8-10, 1999, Cardiff, UK., pp: 675-682.
19. Wijayasekara, D., M. Manic, P. Sabharwall and V. Utgikar, 2011. Optimal artificial neural network architecture selection for performance prediction of compact heat exchanger with the EBaLM-OTR technique. Nuclear Eng. Design, 241: 2549-2557.
20. Cantú-Paz, E., 2003. Pruning Neural Networks with Distribution Estimation Algorithms. In: Genetic and Evolutionary Computation-GECCO 2003. Lecture Notes in Computer Science, Vol. 2723, Cantú-Paz, E., J.A. Foster, K. Deb, L.D. Davis and R. Roy *et al.* (Eds.)., Springer, Berlin, Heidelberg, pp: 790-800.
21. Alshahrani, M., O. Soufan, A. Magana-Mora and V.B. Bajic, 2017. DANNP: An efficient artificial neural network pruning tool. PeerJ Comput. Sci., Vol. 3.
22. Luo, R., F. Tian, T. Qin, E. Chen and T.Y. Liu, 2018. Neural Architecture Optimization. In: Advances in Neural Information Processing Systems 31, Bengio, S., H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi and R. Garnett (Eds.)., Neural Information Processing Systems Foundation, San Diego, pp: 7816-7827.