ISSN: 2584-0495 Vol. 3, Issue 1, pp. 1513-1518

International Journal of Microsystems and IoT

ISSN: (Online) Journal homepage: https://www.ijmit.org

An Adaptive Growing Pruning Algorithm to Optimize Dynamic Feed Forward Neural Networks for Nonlinear Dynamic System Identification

Satvik Chaurasia, Shobana, R

Cite as: Chaurasia, S., & Shobana, R. (2025). An Adaptive Growing Pruning Algorithm to Optimize Dynamic Feed Forward Neural Networks for Nonlinear Dynamic System Identification. International Journal of Microsystems and IoT, 3(1), 1519–1525. https://doi.org/10.5281/zenodo.15493712

Published online: 20 January 2025

DOI: https://doi.org/10.5281/zenodo.15493712

International Journal of Microsystems and IoT
Vol.3, Issue 1, pp.1519-1525; DOI: https://doi.org/10.5281/zenodo.15493712

An Adaptive Growing Pruning Algorithm to Optimize Dynamic Feed Forward Neural Networks for Nonlinear Dynamic System Identification

Satvik Chaurasia, Shobana. R
Department of Electrical & Electronics Engineering, Galgotias College of Engineering & Technology, Greater Noida

ABSTRACT
Optimizing the structure is very crucial for effective identification and control of any nonlinear system. Optimization leads to a robust and more generalized structure. In this work, an effective adaptive growing-pruning algorithm scheme is proposed to optimize dynamic feed forward structures. The hidden layer of the static FFNN is made dynamic and the weights of the dynamic FFNN are trained using standard Back propagation algorithm. Firstly, the network is grown only when the MSE is found high and increasing. Likewise, unnecessarily neurons are pruned based on low activation variance. The learning rate is made dynamic by calculating them based on variance of each neuron. The performance of the proposed algorithm is tested by use of bench mark Mackey glass series problem. The result shows that proposed algorithm performs better than static FFNN with ALR.

KEYWORDS
optimization, FFNN, growing-pruning algorithm, gradient BP based approach

1. INTRODUCTION

Neural networks (NNs) are significantly valuable in artificial intelligence (AI) and machine learning (ML). They provide a robust approach for effectively addressing a range of intricate tasks such as regression, classification, pattern recognition, and decision-making. Neural Networks (NNs) are designed based on the human brain to provide an effective solution for data processing [7-10]. In these networks, a neuron takes in inputs, processes them, and transmits the output to the following layer. The structure allows the feed forward neural network to learn from the input and make predictions. Like all other neural networks, this also consists of three layers: the input layer, hidden layer, and output layer. The input layer handles the original data set, after which the hidden layer process that data , and ultimately the output layer generates the final result, such as a prediction or classification [1]. Neural networks perform better at various tasks than algorithms do. This is due to their ability to learn nonlinear connections between inputs and outputs, enabling them to attain high accuracy [2]-[13-16]. As the input nodes of a feed-forward neural network (FFNN) transmit data only in one direction without any

feedback, it represents the simplest form of neural network. FFNNs are often employed in areas such as identification, image recognition, natural language processing and control due to their ease of use and efficiency [3]. This also introduces constraints like, the incapacity to manage sequential data or patterns dependent on time [4]. FFNNs are often trained with the back propagation method, which adjusts the network's weights to reduce the discrepancy between the predicted and actual outputs [18]. The training process can become more resource-intensive as the network expands, increasing the chances of overfitting, in that case the model struggles with unfamiliar data due to being overly tailored to the training dataset [5]. To address these challenges, optimization methods such as pruning, growing and growing-pruning techniques are been developed. The aim is to optimize the neural networks by adding essential neurons or eliminating the duplicate or unnecessary neurons and connections without degrading the network's performance. A constructive algorithm dynamically adds neurons to improve learning, allowing it to adapt to increasing complexity in nonlinear system identification. While, pruning lowers the computational resources needed for both training and inference while also enhancing the

model's ability to generalize better [6]-[7]. As alone constructive may lead to overfitting likewise, alone pruning can increase the chances of underfitting and increase the chances of failure in identifying or capturing complex patterns, it may also lead to not so accurate results. Recent advances in deep learning have significantly improved the performance of neural networks across various tasks, such as identification of systems [8]-[12]. When combined together, constructive and pruning techniques help maintain an ideal and optimal network size, preventing the excessive expansion that can happen with a purely constructive method and avoiding underfitting that could arise from solely aggressive pruning alone. This blend of growing and pruning enables the network to adaptively respond to the problem's complexity, speeding up convergence, and guaranteeing effective resource use. By continuously expanding when necessary and contracting when feasible and required, the model stays both adaptable and computationally efficient, making it more effective than relying on either pruning or constructive alone.

In this work, we have proposed an adaptive growing pruning algorithm by adaptively growing them based on MSE and prune unnecessary neurons based on variance. We have also implemented an adaptive learning rate based on variance of each neuron to obtain an optimized FFNN structure and further extended them for identification of nonlinear dynamic system. The structure of the paper is organized as follows: Section II outlines the problem statement, while Section III explains the framework and specifics of the FFNN. Section IV includes the learning algorithm. Section V includes the proposed growing pruning algorithm scheme. Section VI includes the simulation results obtained. Ultimately, Section VII addresses the conclusions derived from this study and highlights possible avenues for future research.

2. PROBLEM STATEMENT

Let the input vector R (k) = [r (k), r (k-1), …, r (k-m)] and output vector $Y_{pp}(k) = [y_{pp}(k), y_{pp}(k-1), \ldots, y_{pp}(k-n)]$ of a non-linear plant. The differential equation of the plant at the k^{th} instant is:

$$y_{pp}(k) = f[R(k), Y_{pp}(k), k] \qquad (1)$$

When ADFNN is considered as an identifier, the identifier structure is:

$$y_{adfnn}(k) = \tilde{f}[y_{pp}(k-17), r(k)] \qquad (2)$$

Here, f represents the nonlinear mapping function. The current output $y_{pp}(k)$ relies on both the current and previous values of the plant as well as external input. m, n denotes the order of the plant respectively. The objective is to optimize the FFNN models using adaptive growing pruning algorithm to approximate the nonlinear function $\hat{f} \cong f$ while minimizing the output error between the actual system response $y_{pp}(k)$ and the predicted response $y_{adfnn}(k)$. This is formalized as:

$$|y_{pp}(k) - y_{adfnn}(k)| \leq \epsilon \qquad (3)$$

The classic back propagation (BP) technique with an adaptive learning rate is used to repeatedly update the trainable network weights as $\epsilon \rightarrow 0$ in order to satisfy the criterion given in Eq. (1).

3. FEEDFORWARD NEURAL NETWORK (FFNN)

The FFNN is a forward propagating structure that sends information from input to output layer. The structure is as shown as in Figure 1. The functions of the layer are as below:

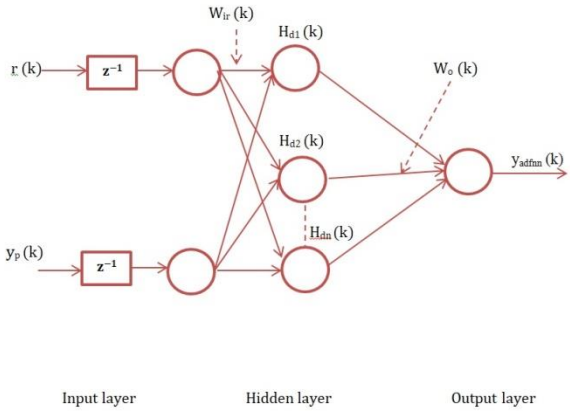

Figure 1 Feed forward neural network

Input Layer:

This layer is composed of input signals. In this work, we have considered two inputs y(k-1) and r(k-1) . The input signals are then forwarded to the next layer by multiplying with the input weight vector Wir (k).

Hidden layer:
The received input layer signals are then processed further. In this work, a single hidden layer is considered with the sigmoid activation function. The signals are further sent to the output layer.

Output layer:
This is the final layer that produces the output. It receives input from hidden layer and is multiplied with weight vector Wo (k). Here purelin activation function is considered. The output of the network is denoted as y_{adfnn} (k).

4. LEARNING ALGORITHM

The network's output is given as:

$$y_{adffnn}(k) = f(\sum_{j=1}^{n} H_{dj}(k)W_o(k) + b_{oh}(k)W_{hr}(k)) \quad (4)$$

Where, H_{dj} signifies the hidden layer vector, W_o is the weight vector of the hidden layer, b_{oh} is the output bias vector and W_{hr} denotes the output bias vector.
The output of hidden layer for the network at k^{th} instant is computed as:

$$H_{dj}(k) = f_1\left(\sum_{j=1}^{n} X(k-j)W_i(k) + b_1(k)W_1(k)\right) \quad (5)$$

Where, W_{ir} signifies the input weight vector, W_i is the weight vector of the input layer, b_1 is the input bias vector.
The weights are updated through a gradient descent back propagation algorithm at every iteration. A cost function is defined at the beginning of the training and the training is carried out until the cost function is found to reach the minimum value. Here, Mean square error (MSE) is chosen as the cost function.
Mathematically the cost function is given as:

$$MSE = \sum_{i=1}^{N}(y_p - y_{adfnn})^2 \quad (6)$$

Using the chain rule, all the weights are updated at every iteration and the stochastic descent algorithm is applied to update the new weights using the formula:

$$W_{new} = W_{old} + \eta \frac{\partial E}{\partial W} \quad (7)$$

Where, $\frac{\partial E}{\partial W}$ indicates the weight updating derivative of W and η is the learning rate. It is kept close between 0 and 1.

5. PROPOSED GROWING PRUNING ALGORITHM SCHEME

In this work we have proposed an adaptive growing pruning network to construct feed forward neural network. It is denoted as ADFNN. The proposed scheme is as shown in Figure 2. The detailed steps are as shown below:

Step 1: The neural network is initialized with no hidden neurons or connections. Parameters like learning rate, initial weights, and biases are set.

Step 2: Train feed forward neural network. After training, the performance of the ANN is evaluated by calculating the MSE, which measures the average squared difference between the predicted and actual values. The algorithm also calculates the variance of each neuron $\mathrm{Var}\left(H_{dj}\right)$. Based on the variance, learning rate matrix is updated. Each neuron is provided with different learning rate based on its variance. It is calculated as below:

$$\eta_j^{(t+1)} = \eta_j{}^t\alpha \ \ if\ \mathrm{Var}\left(H_{dj}\right) > \mathrm{Var\ threshold}$$
$$\eta_j{}^t\beta \ if\ \mathrm{Var}\left(H_{dj}\right) < \mathrm{Var\ threshold}$$

Step 3: The algorithm now checks MSE for two consecutive iterations. If the MSE is found to increase continuously, the algorithm enters the growing step. Here based on MSE, the neurons are added one by one in each iteration cycle. i.e.

$MSE_t > MSE_{t-1}\ and\ MSE_{t-1} > MSE_{t-2}$, then add neuron

Step 4: After every growing step, there are chances that the network is growing unnecessarily large leading to over fitting. Hence now it enters the pruning loop.

Step 5: In pruning loop, it eliminates the neuron based on variance. If activation variance is very low of any neuron compared to threshold, it prunes such neuron.

$$\mathrm{Var}(H_{dj}) = \frac{1}{N}\sum_{i=1}^{N}(H_{dj} - \overline{H_{dj}})^2$$

$\mathrm{Var}(H_{dj}) < threshold$, then prune neuron

Step 6: The performance of the network is again validated. If network found to perform best, then it is considered as final network structure.

Step 7: The algorithm enters the final termination loop and ends. If the MSE has reached its minimum, the Average MSE (AMSE) and Root Mean Squared Error (RMSE) are calculated to evaluate the model's overall performance.

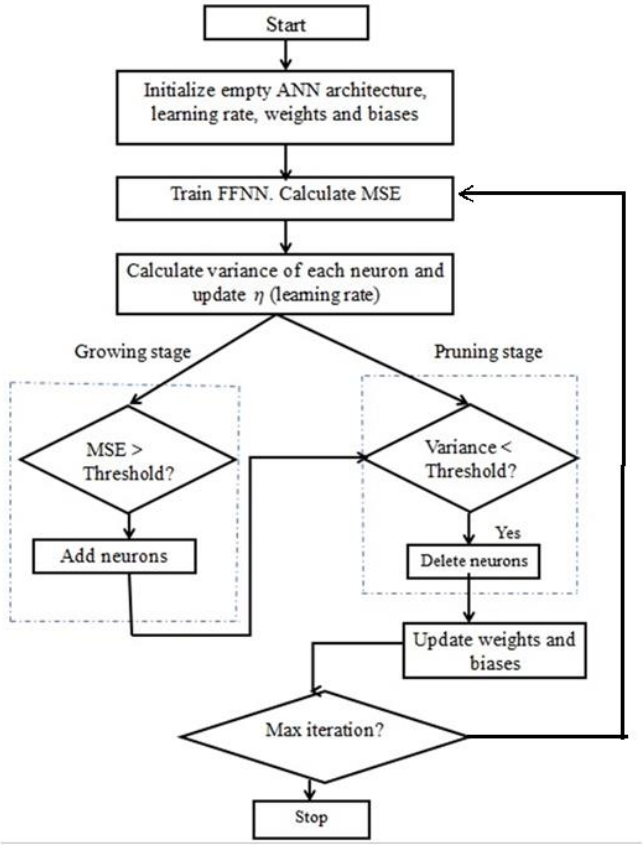

Figure 2 Adaptive growing-pruning algorithm scheme

6. SIMULATION RESULTS

The simulation results of nonlinear plant using ADFNN algorithm with ALR is presented and discussed in this section. The proposed algorithm is tested on the Mackey-glass series prediction, considering it as a benchmark problem.

Consider the following Mackey-glass series prediction problem given by [18]:

$$y_{pp}(k) = -\beta \times (k) + \frac{\alpha \times y_{pp}(k-\tau)}{1 + y_{pp}{}^{10}(k-\tau)}$$

The values of α= 0.2 and β = 0.3. The simulation is carried for 700 epochs. The initial weights and biases are taken random. The initial learning rate is considered as 0.001. Figure 3 shows the validation plot of the performance of ADFNN with ALR, showing optimal number of hidden neurons required to predict the Mackey-glass series. Figure 4 shows the comparison between the prediction of ADFNN with and without ALR for the given benchmark problem. Table 1 shows the tabular comparison between ADFNN with ALR to that of ADFNN without ALR. On analysis of the Figure 3, Figure 4 and Table 1, it is clearly evident that the ADFNN with ALR provides accurate prediction and is a better option for the identification of non-linear system as compared to ADFNN without ALR. Table 2 shows the comparison of RMSEs obtained by other algorithms to that of ADFNN.

S. No.	Structures	Learning rate	No. of hidden neurons (after Growing-Pruning)	MSE	AMSE
1.	ADFNN	Adaptive	18	0.0096	0.0100
2.	ADFNN	Non-Adaptive	20	0.1004	0.0163

Table 1 Comparison of performance of adaptive and non-adaptive ADFNN

Parameter	ADFNN	SORBF[19]	ESRNN [20]
RMSE	0.00238	0.20107	0.019710

Table 2 Comparing our result with other algorithms

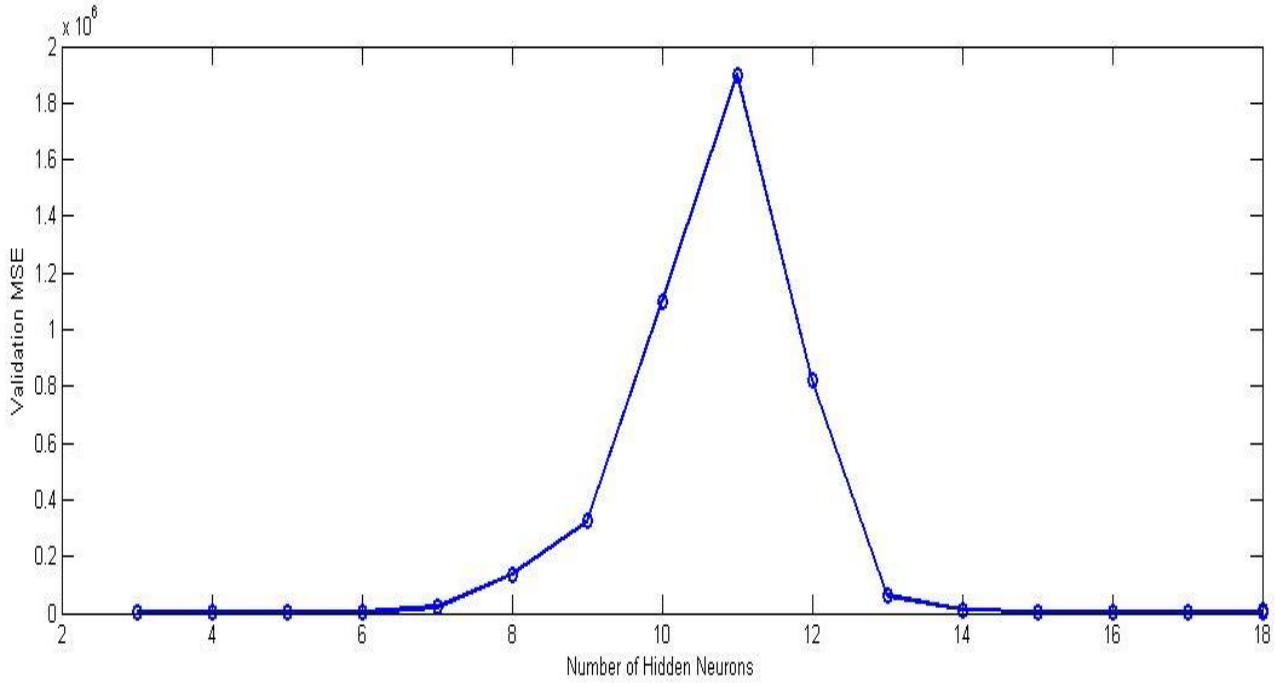

Figure 3 MSE obtained from ADFNN with ALR at different number of neurons

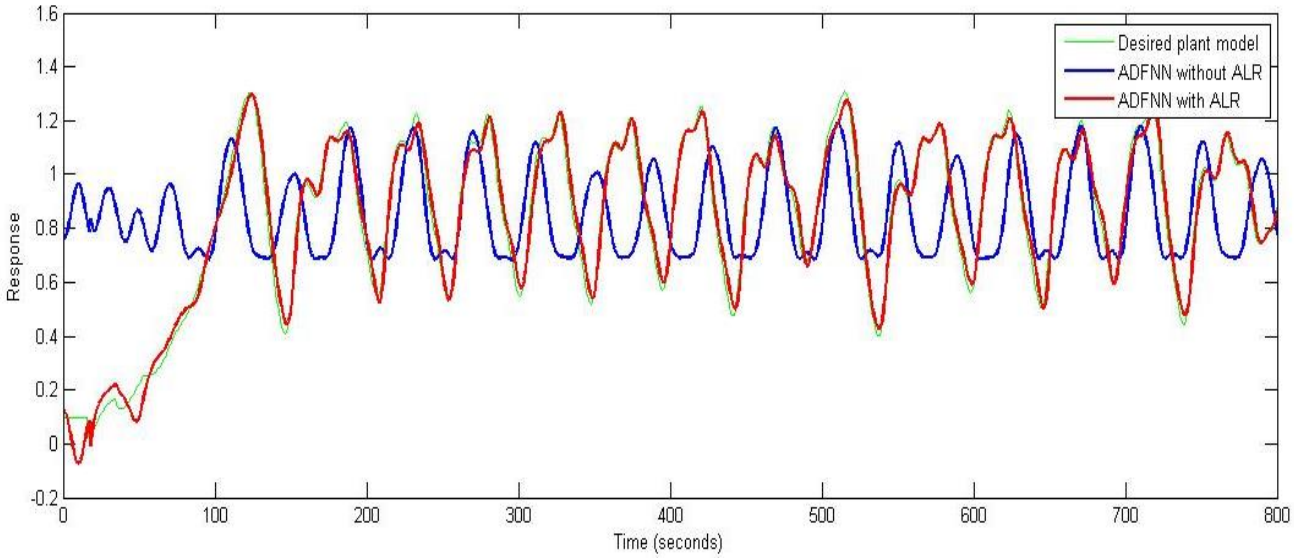

Figure 4 Comparison of response attained by ADFNN with ALR to that of without ALR

7. CONCLUSION AND FUTURE WORK

In this work, we have proposed a growing pruning algorithm with adaptive learning rate based on variance to obtain an optimized FFNN structure for identification of nonlinear dynamic system. The efficiency of the proposed algorithm is demonstrated using a nonlinear benchmark Mackey-Glass series prediction problem. The results of ADFNN with ALR are compared with ADFNN without ALR. The results show that the ADFNN with ALR structure predicts the series more efficiently as compared to the one without ALR. Our future work would focus on developing a new optimization algorithm combined with growing-pruning method.

8. REFERENCES

1. Hecht-Nielsen, R. (1992). Theory of the backpropagation neural network. Neural Networks for Perception, 65-93.

8. Yeung, D.Y., & Xu, J. (2018). L1-regularized neural network pruning with lagrangian optimization. IEEE Transactions on Neural Networks and Learning Systems, 29(10), 4996-5007.

9. Blalock, D., Ortiz, J.J.G., Frankle, J., & Guttag, J. (2020). What is the state of neural network pruning? Proceedings of Machine Learning and Systems, 2, 129-146.

10. Kingma, D.P., & Ba, J. (2015). Adam: A method for stochastic optimization. International Conference on Learning Representations.

11. Zhu, M., & Gupta, S. (2017). To prune, or not to prune: Exploring the efficacy of pruning for model compression. arXiv preprint arXiv:1710.01878.

12. Ioffe, S., & Szegedy, C. (2015). Batch

13. normalization: Accelerating deep network training by reducing internal covariate shift. International Conference on Machine Learning, 37, 448-456.

14. Frankle, J., & Carbin, M. (2019). The Lottery Ticket Hypothesis: Finding sparse, trainable neural networks. International Conference on Learning Representations.

2. LeCun, Y., Denker, J.S., & Solla, S.A. (1990). Optimal brain damage. Advances in Neural Information Processing Systems, 2, 598-605.

3. Hassibi, B., & Stork, D.G. (1993). Second order derivatives for network pruning: Optimal brain surgeon. Advances in Neural Information Processing Systems, 5, 164-171.

4. M. Gethsiyal Augasta1_, T. Kathirvalavakumar (2013) Pruning algorithms of neural networks - a comparative study, 2-3.

5. Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. Advances in Neural Information Processing Systems, 28, 1135-1143.

6. Molchanov, P., Tyree, S., Karras, T., Aila, T., & Kautz, J. (2017). Pruning convolutional neural networks for resource-efficient inference. International Conference on Learning Representations.

7. Zhang, T.,

15. Wen, W., Wu, C., Wang, Y., Chen, Y., & Li, H. (2016). Learning structured sparsity in deep neural networks. Advances in Neural Information Processing Systems, 29, 2074-2082.

16. Gale, T., Elsen, E., & Hooker, S. (2019). The state of sparsity in deep neural networks. arXiv preprint arXiv:1902.09574.

17. Kumpati SN, Kannan P et al (1990) Identification and control of dynamical systems using neural networks. IEEE Trans Neural Netw 1(1):4−27

18. Ibrahim, Rasha Mahmoud. "Sentiment Analysis of Arabic Tweets – Implicit Semantics

19. Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533-536.

20. H.G. Han, J.F. Qiao, Prediction of activated sludge bulking based on a self-organizing RBF neural network, Journal of Process Control 22(6) (2012) 1103-1112.

21. J.Yin, Y. Meng, Y.C. Jin, A developmental approach to structural self-organization in reservoir computing, IEEE Transaction on Autonomous Mental Development 4(4) (2012) 273 – 289.

AUTHORS

Shobana. R currently working as AP in GCET, greater Noida. She received her B.E degree from Anna University, Chennai and M.E degree in control and instrumentation from Anna University, Chennai. She is currently pursuing PhD from Delhi Technological University. Her areas of interest are Neural networks, modeling and control of nonlinear systems. E-mail: r.shobana@galgotiacollege.edu

Satvik Chaurasia is a B.Tech student in the Electrical and Electronics Engineering (EEE)department at Galgotias College of Engineering and Technology. His interests' centers around Control Systems. E-mail: satvikpc14@gmail.com