ISSN: 2584-0495 Vol. 3, Issue 1, pp. 1487-1491

International Journal of Microsystems and IoT

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

Optimizing Feed forward Neural Networks for Nonlinear Dynamic System Identification with Adaptive Learning rate and Pruning algorithm

Shobana R, Aditya Sharma, Anurag Chauhan, Krishna Gupta and Pooja Rout

Cite as: Shobana, R., Sharma, A., Chauhan, A., Gupta, K., & Rout, P. (2025). Optimizing Feed forward Neural Networks for Nonlinear Dynamic System Identification with Adaptive Learning rate and Pruning algorithm. International Journal of Microsystems and IoT, 3(1), 1487–1491. https://doi.org/10.5281/zenodo.15472198

Published online: 20 January 2025

DOI: https://doi.org/10.5281/zenodo.15472198

International Journal of Microsystems and IoT
Vol.3, Issue 1, pp.1487-1491; DOI: DOI: https://doi.org/10.5281/zenodo.15472198

Optimizing Feed forward Neural Networks for Nonlinear Dynamic System Identification with Adaptive Learning rate and Pruning algorithm

Shobana.R, Aditya Sharma, Anurag Chauhan, Krishna Gupta and Pooja Rout

Department of Electrical & Electronics Engineering, Galgotias College of Engineering & Technology, Greater Noida

ABSTRACT

For effective training and enhancing any ANN's capacity for generalization, structure optimization is crucial. In this work, an effective a pruning strategy was created to identify nonlinear dynamic systems by optimizing the feed forward neural network's structure. The gradient-based back propagation technique is used to update the FFNN weights. A threshold is set and the weights between the input, hidden and output layers below the threshold limit are eliminated. A novel adaptive learning rate is created by calculating the score index Si after each epoch thereby increasing the speed of the algorithm. A nonlinear benchmark problem is used to demonstrate the effectiveness of the suggested algorithm. A comparative study of the algorithm is also done with simple FFNN with ALR and PFFNN with ALR.

KEYWORDS

FFNN, pruning algorithm, gradient BP based ANN

1. INTRODUCTION

Neural networks (NNs) play an essential role in artificial intelligence (AI) and machine learning (ML). They can solve complex tasks like regression, classification, pattern recognition, and decision-making efficiently. The working of the neural network is inspired by the human brain for efficient information processing. [1]. In these networks, the inputs received by the neurons is calculated and the output is sent to the next layer. These networks learn from the inputs provided to them and give a desired prediction all thanks to its structure. A neural network consists of mainly three layers: input layer, hidden layer, and an output layer. [2]. The input layer receives the initial data set and sends it to be hidden layer where it analyzes the data by finding pertinent patterns, finally it is sent to the output layer to generate the result like a classification or prediction. [3]. As compared to other algorithms neural networks are preferred because they can learn nonlinear relationship between inputs and outputs thus achieving a higher accuracy. [4]. FFNN is the simplest form of neural network which transmits data only in a forward direction without any feedback. Their user-friendliness and effectiveness lead to its frequent usage in fields like image recognition, natural language processing, identification, and control. [5]. However, this simplicity limits the FFNN's ability to handle sequential data or time dependent patterns. [6]. FFNNs are frequently trained using the back propagation technique, which optimizes the network's weights by minimizing the error between expected and actual outputs. The training process may become more computationally demanding as the network gets bigger, and there is a greater chance of over-fitting, in which the model performs poorly on unknown data because it is too particular to the training set [7]. In order to overcome these difficulties, optimization techniques like pruning approaches have been developed. Their goal is to simplify neural networks by removing unnecessary or redundant neurons and connections without appreciably compromising performance. Pruning reduces the computational resources required for training and inference while simultaneously improving the model's capacity for generalization [8]. Pruning techniques can be categorized into several groups: significance-based methods, mutual information (MI)-based methods, magnitude-based methods, cross-validation methods, evolutionary algorithms, sensitivity analysis (SA) methods, and penalty term approaches [9]. In order to create neural network models that are more effective and efficient, pruning methods have become a crucial element of contemporary deep learning. Neural networks now perform noticeably better on a variety of tasks, including system identification, thanks to recent

Developments in deep learning [10]. Building on these developments, pruning methods have been investigated to enhance neural networks' effectiveness and performance even further. To lower the computational cost and increase the effectiveness of neural networks, some pruning strategies have been investigated, for example, dynamic network surgery and

sparsity-inducing regularization [11-17]. The outline of the paper is as follows II describes the problem statement; section III describes the structure and details of the FFNN. The learning algorithm and weight update rules are also covered in this section, In Section IV, the simulation results are presented. Finally, Section V discusses the conclusions drawn from this study and outlines potential directions for future research.

2. PROBLEM STATEMENT

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

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

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

$$y_{pffnn}(k) = f[y_{pp}(k-17), y_p(k)] \quad (2)$$

Here, f represents the nonlinear mapping function. The current output $y_p(k)$ 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 pruning algorithm to approximate the nonlinear function $f \hat{} \cong f$ while minimizing the output error between the actual system response $y_{pp}(k)$ and the predicted response $y_{pffn}(k)$. This is formalized as:

$$|y_{pp}(k) - y_{pffn}(k)| \leq \in \quad (3)$$

The classic back propagation (BP) technique with an adaptive learning rate is used to repeatedly update the trainable network weights as $\epsilon \rightarrow 0$ 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 functions of the layer are as follows:

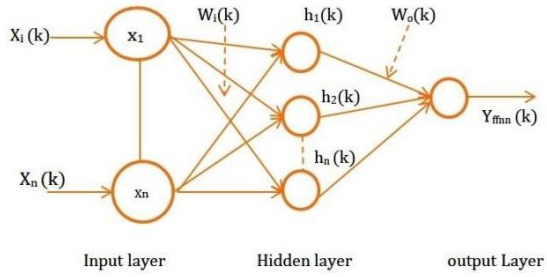

Figure 1 Feed forward network structure

Input Layer:

The input layer obtains both the past and present values of the external input and plant output. In this work, we have considered two inputs as y (k−17), and y (k). This layer receives these values as input and forwards it to the next layer.

Hidden layer:

This layer receives the inputs from the input layer and process the input. We have considered a single hidden layer in our work. The input signals are multiplied with input weight vector W_i (k) and sent to the output layer. In this layer, a nonlinear activation function.

Output layer:

Using linear activation function this layer generates the final output of the network which is denoted as $y_{pffnn}(k)$.

4. LEARNING ALGORITHM

The network's output is given as:

$$y_{pffnn}(k) = f(\sum_{i=1}^{m} H_j(k) w_o(k) + b_{oh}(k) w_{bh}(k) \quad (4)$$

where w_o(k) is the output weight of FFNN, w_{bh}(k) is the output bias weight and b_{oh}(k) is the output bias vector. The linear activation function used is denoted by f.

The output of hidden layer for the network at k^{th} instant is computed as:

$$H_j(k) = f_1\left(\sum_{j=1}^{n} X(k-j) w_i(k) + b_1(k) w_1(k)\right) \quad (5)$$

where w_i(k) is the input weight, b_1(k) denotes the associated input bias vector with f_1 is the nonlinear activation function of the FFNN network.

A gradient descent-based back propagation algorithm is used to optimize the tunable parameters of a feed-forward neural network (FFNN). The Mean Squared Error is used as the cost function in this process, which can be expressed as:

$$E(k) = \frac{1}{2}[y_{pp}(k) - y_{pffnn}(k)]^2 \quad (6)$$

Here, e (k) = ypp (k) − ypffnn (k) denotes the identification error.

To propagate errors from output layer back to hidden layer the chain rule is used which adjusts the weights in the output layer during training process.

$$\frac{\partial E(k)}{\partial w_o(k)} = \frac{\partial E(k)}{\partial y_{pffnn}(k)} \times \frac{\partial y_{pffnn}(k)}{\partial w_o(k)} \quad (7)$$

Additionally, the output weights w_o (k) are updated using Stochastic Gradient Descent (SGD) as follows:

$$w_o(k+1) = w_o(k) + \eta e(k) H_j(k) \quad (8)$$

where H_j(k) indicates the induced fields of PFFNN. Here, η

denotes the learning rate, typically set within a range of 0 to 1. By propagating the error backward all the way to the input layer, the weights wi (k) between the input layer and the hidden layer can be adjusted [19].

$$\frac{\partial E(k)}{\partial w_i(k)} = \frac{\partial E(k)}{\partial y_{pffnn}(k)} \times \frac{\partial y_{pffnn}(k)}{\partial R_j(k)} \times \frac{\partial R_j(k)}{\partial w_i(k)} \quad (9)$$

The new weights are updated using the SGD as:

$$w_i(k+1) = w_i(k) - \eta e(k) \frac{\partial E(k)}{\partial w_i(k)} \quad (10)$$

5. PROPOSED PRUNING ALGORITHM

In this work, the FFNN structure is optimized using a pruning scheme with adaptive learning rate. Figure 2 shows the pruning algorithm scheme. The detailed procedure is as described below:

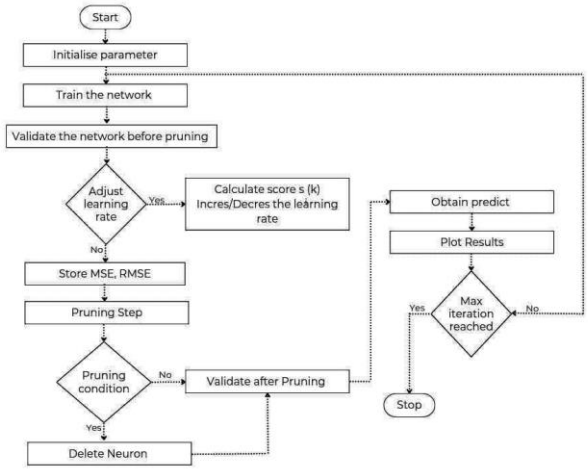

Figure 2 Proposed pruning algorithm scheme

Step 1: Build the FFNN architecture and set up all the settings, including the network's learning rate, weights, and biases. The number of outputs in the network is equal to the number of predictions, and the number of input neurons is equal to the number of observations. In the hidden neurons, the maximum number of neurons is set to 50.

Step 2: Train and validate the network. Adjust learning rate if required by calculating the score Si (k) using eqn. no. 10. Store MSE, AMSE of the epoch.

Step 3: Execute the pruning step by removing unnecessary weights of neurons from input, hidden and output layer with magnitudes below the threshold.

Step 4: Validate the network after pruning procedure is executed. Obtain the predictions and plot the results.

Step 5: Stop the execution after termination condition is met. When maximum iterations are reached, the network terminates in this work.

Novel Adaptive learning rate: During training, the learning rate controls how much a neural network's weights are altered. Training will be sluggish or may not function at its best if the learning rate is too low, whereas convergence issues will arise if the learning rate is too high. Adaptive learning rate approaches were developed to address this issue by automatically modifying the learning rate throughout the training process in accordance with the gradient of the error function. In our approach, we introduced an adjustable learning rate by computing a score index S_i (k). We have used the performance matrices to increase or decrease the learning rate of the next consecutive iteration based on previous performance. We have considered MSE and RMSE as the performance metrics. The calculation of the score index S_i (k) is as below:

$$S_i(k) = w_1 \times MSE(k) + w_2 \times RMSE(k) \quad (11)$$

Where w_1 and w_2 are the weights based on their importance and they are made equal to 0.5 Based on the score index S_i (k), the learning rate adapts as below:

$$\text{If } S_i(k) < S_i(k-1) \text{ then } \eta(k+1) = \eta(k)(1+\alpha)$$

(Or)

$$\text{If } S_i(k) \geq S_i(k-1) \text{ then } \eta(k+1) = \eta(k)(1-\alpha) \quad (12)$$

6. SIMULATION RESULTS

The simulation results of nonlinear plant using PFFNN algorithm with ALR is presented and discussed in this section. The initial number of hidden neurons is selected to be 50 for PFFNN algorithm with ALR, PFFNN algorithm with FLR and FFNN algorithm with FLR. The proposed algorithm is tested on the Mackey series prediction bench mark problem. A total of 1500 samples are considered. 600 values are considered for testing and remaining 900 values for validation.

A. Mackey glass series prediction

The performance of the algorithm is tested on the following Mackey-glass series prediction problem given by [18]

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

When series-parallel based identification is considered, the plant takes the following structure:

$$y_{pp}(k) = f[y_{pp}(k-17), y_{pp}(k)] \quad (13)$$

Now, if FFNN is considered as identifier, the identification structure will be:

$$y_{ffnn}(k) = f_2[y_{pp}(k-17), y_p(k) \quad (14)$$

Here, f_2 is the nonlinear mapping function.

The Figure 3 illustrate that the proposed PFFNN with ALR effectively predicts the Mackey-Glass chaotic series. The analysis of Figures 3–4 clearly shows that the PFFNN with ALR provides accurate predictions. Additionally, to assess the model's performance, a comparison was made with the PFFNN using FLR, with detailed results summarized in the table. The table 1 highlights the final count of hidden neurons, along with

the RMSE and MSE values. It is clear from Table 1 that the PFFNN with ALR uses fewer hidden neurons than its counterpart, the PFFNN with FLR. This example implies that the PFFNN with ALR algorithm has better prediction ability than some other existing algorithms. Further the proposed PFFNN is also compared with other proposed algorithms for pruning in literature. It is evident from table 2 that the proposed PFFNN is effective for identification of Mackey glass series.

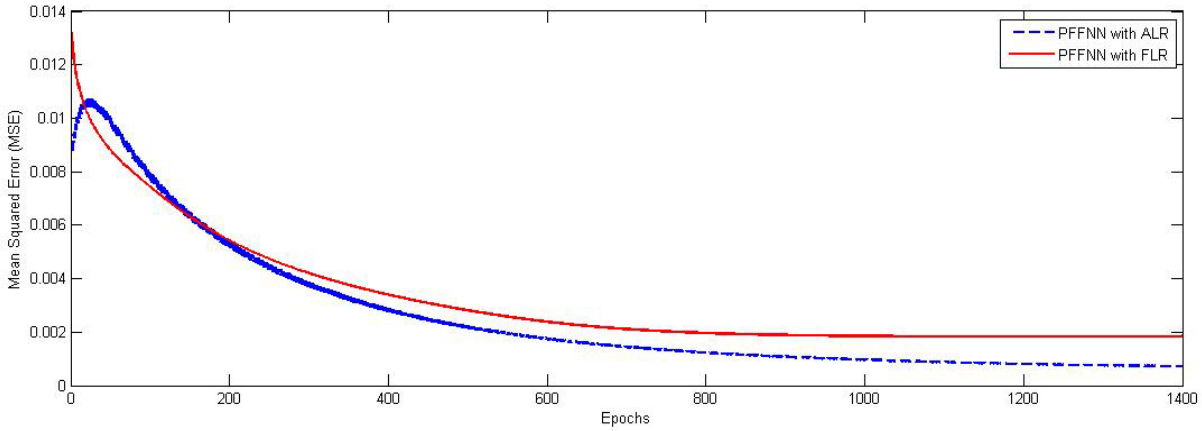

Figure 3 MSE obtained from PFFNN with ALR and PFFNN with FLR structures

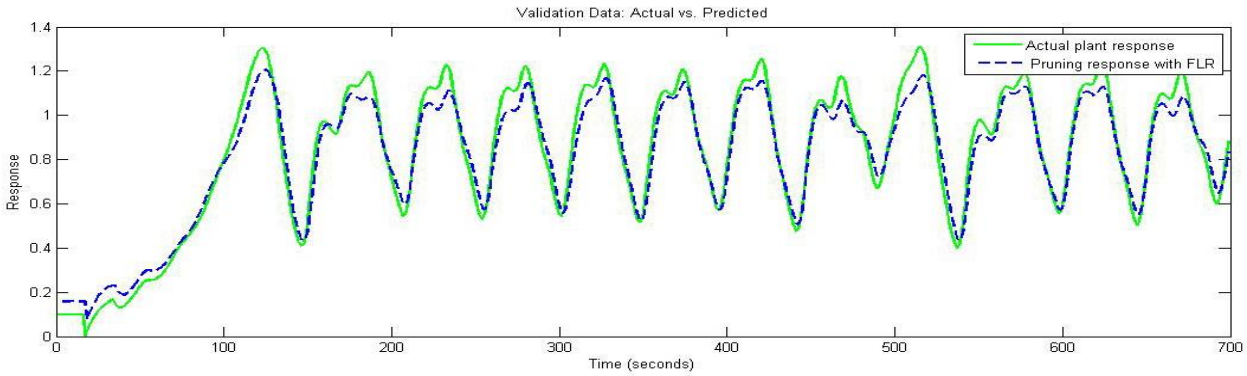

Figure 4 Response obtained from plant during final stages of training

S. No.	Structures	Learning rate	No of hidden neurons (before pruning)	No of hidden neurons (after pruning)	MSE (before Pruning)	MSE (after Pruning)
1.	PFFNN	Adaptive	50	**18**	**0.0015**	**0.0007**
2.	PFFNN	Fixed	50	21	0.0021	0.0018

Table 1 Comparison of PFRNN with ALR with other selected structures

Table 2. Comparing our result with other algorithms

Parameter	PFFNN	Bilal Shoaiba.et.al [19]	LuLu.et.al[20]
MSE	0.0007	0.20107	0.019710

7. CONCLUSION AND FUTURE WORK

In this work, we have proposed a pruning algorithm with novel adaptive learning rate to optimize the FFNN structure for identification of nonlinear dynamic system. We have made learning rate dynamic by calculating a score index S_i (k) which is based on the combination of importance of performance metrics. The efficiency of the proposed algorithm is demonstrated using a nonlinear benchmark Mackey series prediction problem. The results of PFFNN with ALR are compared with PFFNN with FLR and standard FFNN with ALR. The results show that the pruned FFNN with ALR structure outperforms the fixed structures with FLR. It also shows better prediction accuracy and MSE than fixed adaptive learning rate. To further improve the overall performance of the algorithm, BP can be combined with any other new optimization technique. Our future work would will focus on

developing a new optimization algorithm combined with BP method.

REFERENCES

1. O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, A. M. Umar, and O. U. Linus, "Comprehensive Review of Artificial Neural Network Application to Pattern Recognition, "IEEE Access, vol. 7, pp. 108512-108546, 2019.
2. I. A. Basheer and M. Hajmeer, "Artificial neural networks: fundamentals, computing, design, and application," Journal of Microbiological Methods, vol. 43, no. 1, 1 pp. 3-31, Oct. 2000.
3. McCulloch, W.S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 5, 115-133.
4. Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1986). Learning representations by back- propagating errors. Nature,323(6088),533- 536
5. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
6. Hecht-Nielsen, R. (1992). Theory of the backpropagation neural network. Neural Networks for Perception, 65-93.
7. LeCun, Y., Denker, J.S., & Solla, S.A. (1990). Optimal brain damage. Advances in Neural Information Processing Systems, 2, 598-605.
8. Hassibi, B., & Stork, D.G. (1993). Second order derivatives for network pruning: Optimal brain surgeon. Advances in Neural Information Processing Systems, 5, 164-171.
9. M. Gethsiyal Augasta1_, T. Kathirvalavakumar (2013) Pruning algorithms of neural networks - a comparative study, 2-3.
10. 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.
11. Molchanov, P., Tyree, S., Karras, T., Aila, T., & Kautz, J. (2017). Pruning convolutional neural networks for resource-efficient inference. International Conference on Learning Representations.
12. Zhang, T., 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.
13. 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.
14. Kingma, D.P., & Ba, J. (2015). Adam: A method for stochastic optimization. International Conference on Learning Representations.
15. Zhu, M., & Gupta, S. (2017). To prune, or not to prune: Exploring the efficacy of pruning for model compression. arXiv preprint arXiv:1710.01878.
16. Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. International Conference on Machine Learning, 37, 448-456.
17. Frankle, J., & Carbin, M. (2019). The Lottery Ticket Hypothesis: Finding sparse, trainable neural networks. International Conference on Learning Representations.
18. Ibrahim, Rasha Mahmoud "Sentiment Analysis of Arabic Tweets – Implicit Semantics.
19. B. Shoaib, I. M. Qureshi, Shafqatullah, and Ihsanulhaq, "Adaptive step-size modified fractional least mean square algorithm for chaotic time series prediction," Chinese 1 Physics B, vol. 23, no. 4, p. 040501, 2014.
20. Lu, L., Zhao, H., & Chen, B. (2017). Time series prediction using kernel adaptive filter with least mean absolute third loss function. Neural Processing Letters, 46(3), 903-918.

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

Aditya Sharma is a B.Tech student in the Electrical and Electronics Engineering (EEE) department at Galgotias College of Engineering and Technology (GCET). His area of interests is soft computing projects.

Pooja Rout is currently pursuing a Bachelor of Technology (B.Tech) degree in electrical and electronics engineering at Galgotias College of Engineering and Technology, Greater Noida. Her research interests center around the neural networks.

E-mail id: -
poojarout3408@gmail.com

Anurag Chauhan is a B.Tech student in the Electrical and Electronics Engineering (EEE) department at Galgotias College of Engineering and Technology (GCET). His area of interests in control systems.

Krishna Gupta is currently pursuing a Bachelor of Technology (BTech) degree in electrical and electronics engineering at Galgotias College of Engineering and Technology, Greater Noida. His research interests center around the ANN.