Cihan University-Erbil Scientific Journal (CUESJ)

Research Article

Performance Comparison of Neural Networks Training Algorithms in Time Series Forecasting

Diyar M. Khalil

Department of Mathematics, Faculty of Science, Soran University, Kurdistan Region, Iraq

ABSTRACT

Research gaps exist in the area of forecasting since there has been no comparison between different algorithms for training the feed-forward neural network (FFNN) model. One of the main reasons for this existing gap is that it is hard to decide which training algorithm to choose. This study proposes to fill that gap by identifying the best method to train the FFNN (both shallow and deep learning), a technique for a univariate monthly time series forecast. A total of seven widely known techniques have been studied to compare their efficiency: Quick propagation algorithm, conjugate gradient descent algorithm, Quasi–Newton algorithm, limited memory Quasi–Newton algorithm, Levenberg–Marquardt algorithm, online back propagation algorithm (Stochastic Gradient Descent), and lastly the batch back propagation algorithm. The study was carried out using Alyuda NeuroIntelligence 2.2. In addition, four statistical measurements (mean absolute percentage error [MAPE], root mean square error [RMSE], mean absolute error [MAE], and coefficient of determination [R^2]) were used for comparison. The results indicate that conjugate gradient descent outperforms the other algorithms, yielding the highest R^2 and the lowest values for RMSE, MAPE, and MAE, proving its superior effectiveness in training the FFNN for forecasting monthly time series. However, batch back propagation was shown to be the least effective algorithm, with the lowest R^2 value and the highest values for RMSE, MAPE, and MAE.

Keywords: Artificial neural network, conjugate gradient descent, deep learning, forecasting, training algorithm

INTRODUCTION

Once the researchers have identified an artificial neural network (ANN) design that is of interest, it needs to be trained so that the network's parameters can be calculated from data. To achieve this effectively, a suitable training algorithm is required. Training a neural network (NN) can be viewed as a non-linear mathematical optimization problem, and various solving algorithms or methods might have very varied effects on the training outcome. Using various techniques can improve NN training performance.[1] NN training is an unrestricted non-linear minimization issue where a network's weights are repeatedly adjusted to reduce the average or total squared error between the predicted and actual output values for all output nodes across all input patterns. There are numerous optimization methods available, providing a variety of options for NN training.[2] In addition, there is currently no algorithm known that guarantees a global optimal solution to a general non-linear optimization issue in a reasonable period of time. As a result, all optimization algorithms in reality necessarily confront local optima issues, and the best that can do is utilize an appropriate optimization method that can provide the "best" local optima if the genuine global solution is not obtainable.[1]

In their famous and foundational work, Zhang *et al*.[3] identified a significant gap in the area of ANNs used for forecasting, specifically time series forecasting. They expressly requested further study to answer the fundamental question: "What is the best training method or algorithm for forecasting problems?" Building on the stated research gap, this study was conducted to fill this gap and contribute to the identification of the most effective learning algorithm for time series forecasting.

The key objective of this study is to determine the best algorithm for training a Feed Forward NN (FFNN) to forecast monthly time series data. To achieve this purpose, seven distinct training algorithms, including quick propagation (QP), conjugate gradient descent (CGD), Quasi–Newton (QN), limited memory Quasi–Newton (LM-QN), Levenberg–Marquardt (LM), online back propagation (OBP), and

Corresponding Author:
Diyar M. Khalil, Department of Mathematics, Faculty of Science, Soran University, Kurdistan Region, Iraq.
E-mail: diyar.khalil@soran.edu.iq

Received: October 17, 2025
Accepted: November 30, 2025
Published: January 1, 2026

DOI: 10.24086/cuesj.v10n1y2026.pp6-11

batch back propagation (BBP), were utilized for training the networks. To obtain more accurate results, each algorithm applied to train two different FFNN architecture: A shallow network with a single hidden layer and a deep network having two hidden layers, each with a different number of nodes; With each architecture tested on two different important financial datasets: The first is monthly Brent oil price from 1990 to 2023, and second is the monthly gold price covering the period from 1990 to 2024. These time series were selected based on their importance in the financial market, since they have significant effects on the global economy, investment strategies, and the stability of the market. The fluctuation of prices in this time series highlights the impact of economic, political, and environmental factors on supply and demand.[4,5]

This study's results will probably help optimize NN training methods and improve forecasting accuracy for complicated financial time series.

LITERATURE REVIEWS

Several studies have been conducted to compare the performance of training algorithms in different fields, including classification, pattern recognition, and medication delivery. However, research that analyzed and compared these algorithms in the context of time series forecasting is still limited. Furthermore, among the limited studies available, very few have focused on forecasting financial time series, revealing a significant gap in the research. In this part, we highlight some of the current studies, regardless of their limited application to financial forecasting.

Awang *et al*.[6] compared the performance of nine NN training algorithms using a dataset from a large Malaysian telecom provider. A multilayer perceptron with 14 input nodes, one hidden layer, and one output node was trained using different training algorithms. Their results showed that, among these algorithms, the L-M algorithm had the greatest accuracy in forecasting of 94.82% throughout training and testing, suggesting its usefulness in churn forecasting. Tabbussum and Dar[7] conducted a study to build enhanced flood forecasting models utilizing ANN algorithms and verified them on the Jhelum River. The L-M algorithm performed best among the five learning algorithms employed, having the lowest MSE and the greatest R^2. Arthur *et al*.[8] compared the performance of different NN training algorithms for forecasting ground vibration caused by blasting. They concluded that the QN approach had the highest prediction accuracy of all training algorithms evaluated.

TRAINING ALGORITHM

One of the most obvious characteristics of a NN is its capability to learn from the presentation of patterns. The procedure for training involves a series of stages employed to adjust the weights of the network's neurons. After training, the network has learnt the association between inputs and outputs and may therefore generate output that is approximately equal to the predicted outcome of any specific input.[9,10] NNs, such as humans, acquire information through examples. A NN learns using a collection of input data known as a training set. The purpose of training is to reduce the value of the average error or minimize the difference between the NN's output and the desired output.[11]

No algorithm can guarantee the ideal global solution in an acceptable duration of time for some large non-linear optimization problems. The most common training method is back-propagation, which is essentially a gradient steepest descent strategy.[12] A brief overview of the learning algorithms used in this study is given below.

QP

The QP algorithm is a repetitive second-order optimization technique. In each phase, the technique approximates the target function with a quadratic function for each variable, independent of the others. In NNs, the objective functions' input variables are the network's parameters or weights.[13]

CGD

The CGD is an advanced algorithm for training multilayer NNs.[14] It is an optimization algorithm that learns an NN by identifying the best path to minimize error (loss), skipping unnecessary steps. Instead of simply moving downward (as with standard gradient descent), it moves in more intelligent, more efficient directions to reach the minimum faster, which is especially effective for large and complex problems. When training a NN, conjugate gradient optimizes paths to discover the best weights quickly and effectively.[15]

QN

The QN algorithm is similar to Newton's approach for line search, with the exception that the inverse Hessian is approximated and adjusted between iterations.[16] The QN algorithm is an optimization algorithm for training NNs or minimizing functions that approximate the second derivative of the loss function rather than calculating it directly. It increases the search direction over gradient descent, allowing the model to converge faster without the computational overhead of true second-order approaches such as Newton's method.[17]

LM-QN

This method of optimization, also known as the LM Broyden Fletcher–Goldfarb–Shanno (LM-BFGS) algorithm, is a powerful tool for the optimization of complex problems. Traditional optimization algorithms involve storing the inverse of the Hessian matrix of second derivatives of the loss function; however, this powerful technique uses a small set of vectors that denote changes to the parameter and gradients of recent modifications to the function's parameterization of the target function.[18,19] This approach enables the LM-BFGS method to estimate curvature with better accuracy and search directions with faster convergence compared to Gradient Descent. This technique can be efficiently implemented for a big-scale optimization task, where storing the entire matrix of a function's second derivative may not be possible or feasible. This technique is widely used to solve complex optimization problems across a wide range of industries.[20]

LM

This technique is a type of iterative method that can calculate the minimum value for a given function expressed in the form of a sum of squared non-linear functions. This technique uses a combination of the steepest descent method and the QN algorithms; hence, it is a common method for the solution of non-linear least squares problems.[21,22] Furthermore, depending on the value of the difference between the actual and desired values of the coefficients, the L-M method can be a gradient descent technique; however, by approximating the ideal value of the coefficients, it may resemble the QN technique. This technique of optimization offers flexibility.[21]

OBP

This technique is a modification of the BP technique for training ANNs. This technique is also known as the Stochastic Gradient Descent with Back Propagation. This technique updates the weights of the network after every training instance instead of updating after going through all of the training examples, as BP does.[23] This technique provides faster updates and more regular updates, making it suited for applications that involve real-time or streaming inputs. This technique begins with the initialization of weights. Then it tours the network to calculate the output values. Next, the error margin is computed at the last layer and then backpropagated to every layer of the network. Finally, the weights are updated simultaneously using the computed gradient value.[24]

BBP

It is a kind of supervised learning that uses gradient descent to lower the error margin on the training dataset. Instead of the weights being updated after each pattern, the error for the entire dataset was computed, and the weights of the network were updated for every epoch of the dataset. This learning technique works well for small to medium-scale datasets.[25] However, for a larger dataset, it may be inefficient and space-consuming, too. Furthermore, it does not have the flexibility of online learning algorithms.[26]

MATERIALS AND METHODS

In this study, seven of the most important algorithms for training the FFNN to forecast monthly time series were compared. The algorithms are: QP, CGD, QN, LM-QN, L-M, OBP, and BBP. All algorithms have been tested under similar conditions, with all other factors constant, such as the number of neurons in the input layer, the number of hidden layers and their neurons, and the activation functions between the layers. These parameters were similar for all algorithms, ensuring a fair comparison.

To enable a comparative investigation of algorithm performance across shallow and deep network architectures. The seven algorithms were used in two scenarios: A standard NN with one hidden layer and a deep learning architecture with two hidden layers. Furthermore, to evaluate the algorithms, two of the most well-known and trusted datasets were used. The first dataset, containing 408 observations, consists of Brent crude oil monthly prices from January 1990 to December 2023. The second dataset, which includes 420 observations, covers the monthly gold price in USD from 1990 to 2024. Both datasets were extracted from the World Bank website. In addition, Alyuda NeuroIntelligence 2.2 was used for analyzing the data.

The architecture of the deep FFNN for the first dataset is (12:7:9:1), whereas for the second dataset it is (12:6:8:1). Similarly, the shallow FFNN designs are (12:4:1) for the first dataset and (12:6:1) for the second dataset (all of these networks are represented in Figure 1). Because the data are monthly, the most appropriate number of input neurons is 12. All four networks were trained under the same conditions regarding data partitioning and activation functions: The logistic activation function was applied between all layers. For training the networks, 85% of the data was used as a training set, whereas 15% was reserved for testing the network's accuracy. Furthermore, to obtain accurate results, all networks with each algorithm were trained for 1000 iterations and retrained 10 times. In addition, to guarantee consistency across trials for all algorithms that need momentum and learning rate, a momentum value of 0.5 and a learning rate of 0.5 were used.

Finally, to compare algorithms' performance, four statistical measures were used: Root mean square error

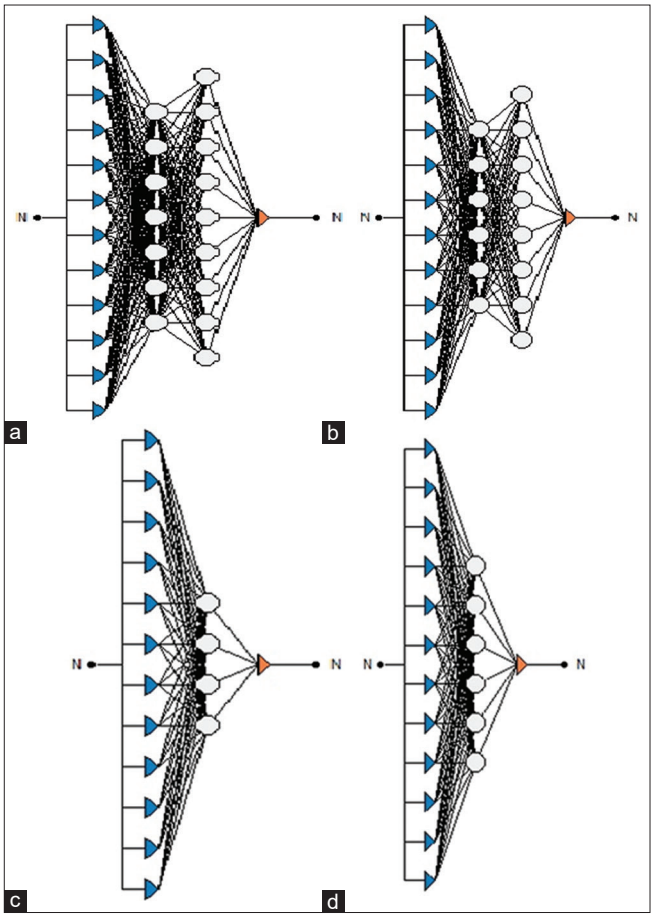

Figure 1: The shallow and deep learning architecture of the feed-forward neural network for both datasets: (a) Deep learning architecture to forecast the 1st dataset; (b) deep learning architecture to forecast the 2nd dataset; (c) shallow architecture to forecast the 1st dataset; (d) shallow architecture to forecast the 2nd dataset

(RMSE), mean absolute percentage error (MAPE), mean absolute error (MAE), and coefficient of determination (R^2).

RESULTS

This section presents the results individually for the two types of network topologies employed in this study: The shallow NN (one hidden layer) and the deep learning model (two hidden layers). Each was applied separately to the datasets.

The Results of Deep Learning NN

As it is clear from Table 1, after training the FFNN with the architecture (12:7:9:1) to forecast the Brent oil price using all training algorithms it was found that the CGD algorithm is the most suitable algorithm for training the network and produces best results, as its R^2 value (0.9897) is greater than the other algorithms, and the RMSE (3.5087), MAPE (5.7216) and MAE (2.4388) values of this algorithm are the lowest when compared to the other algorithms. Furthermore, the second most effective algorithm is QP, which achieved an R^2 value of 0.9867and had lower error rates compared to other algorithms. In contrast, the BBP is the worst-performing algorithm, with the lowest R^2 (0.951) value and high RMSE (6.6861), MAPE (13.5471), and MAE (4.8566) values.

Table 1: Algorithms' performance on Brent oil dataset using feed-forward neural network (12:7:9:1)

Algorithm	R^2	RMSE	MAPE	MAE
Quick propagation	0.9867	3.6110	5.8465	2.4852
Conjugate gradient descent	0.9897	3.5087	5.7216	2.4388
Quasi–Newton	0.9836	3.9090	5.7437	2.6895
Limited memory Quasi–Newton	0.9840	3.9424	6.6992	2.8747
Levenberg–Marquardt	0.9812	4.2435	7.2016	3.1503
Online back propagation	0.9700	5.1191	12.6149	4.1301
Batch back propagation	0.9510	6.6861	13.5471	4.8566

Table 2: Results of training algorithms on the gold price dataset using a feed-forward neural network (12:6:8:1)

Algorithm	R^2	RMSE	MAPE	MAE
Quick propagation	0.9961	38.4165	3.3307	27.0678
Conjugate gradient descent	0.9971	33.2896	2.6668	23.0659
Quasi–Newton	0.9968	35.0563	2.7367	23.7265
Limited memory Quasi–Newton	0.9965	36.3542	2.9268	25.2957
Levenberg–Marquardt	0.9930	50.8586	6.5662	40.4949
Online back propagation	0.9807	82.7830	11.7119	67.5922
Batch back propagation	0.9767	90.2093	10.9131	68.2092

As for the gold price dataset, similar analyses were conducted using the different network architecture, which is FFNN (12:6:8:1). As indicated in Table 2, the algorithm's performance differs significantly among measurements. Having the highest value of R^2 (0.9971) and the lowest value of RMSE (33.2896), MAPE (2.6668), and MAE (23.0659), the CGD is identified as the best performing algorithm. The second-best method is QN, which performed well across all criteria for evaluation, yet slightly lower than CGD. In addition, the BBP algorithm has the lowest R^2 (0.9767) value and the biggest measurement of error RMSE (90.2093), MAPE (10.9131), and MAE (68.2092) compared to all other algorithms.

Results of the Shallow NN

Similarly, in the deep learning case, two different FFNN structures with one hidden layer were used, one for Brent oil price prediction (12:4:1) and one for gold price prediction (12:6:1); seven training algorithms were applied to evaluate which is most effective for each model. As clearly shown in Table 3 and Figure 2, the CGD algorithm once again outperforms the others across both datasets, producing the best results in terms of performance. CGD achieved an R^2 value of 0.9827 for the FFNN (12:4:1) model and 0.9915 for the FFNN (12:6:1) model, which were much higher than those obtained by the other training algorithm. CGD also achieved the lowest error values for the FFNN (12:4:1), with an RMSE of 4.1322, MAPE of 6.5347, and MAE of 2.8663. Similarly, the FFNN (12:6:1) outperformed all other algorithms, with an RMSE of 55.1578, MAPE of 3.0332, and MAE of 30.6076. The QN algorithm came in second, with results similar to those of CGD. In contrast, the BBP algorithm performed the poorest of all the algorithms tested, because it yielded the lowest R^2 value and highest RMSE, MAE, and MAPE values.

DISCUSSION

The results of this study contribute to the continuing discussion over the best training procedures for FFNN in time series forecasting, notably for financial datasets such as Brent oil and gold prices. The study demonstrates that the choice of training

Table 3: Results of training algorithms on the gold price dataset using a feed-forward neural network (12:6:1)

Algorithm	Coefficient of determination	Root mean square error	Mean absolute percentage error	Mean absolute error
Quick propagation	0.9837	73.7238	5.02772	41.6158
Conjugate gradient descent	0.9915	55.1578	3.0332	30.6076
Quasi–Newton	0.9901	59.1371	3.0339	30.9725
Limited memory Quasi–Newton	0.9837	74.3231	4.3138	39.1710
Levenberg–Marquardt	0.9727	91.6437	7.9645	56.5298
Online back propagation	0.9733	91.6789	7.1293	62.1992
Batch back propagation	0.9747	94.7512	13.1164	77.5222

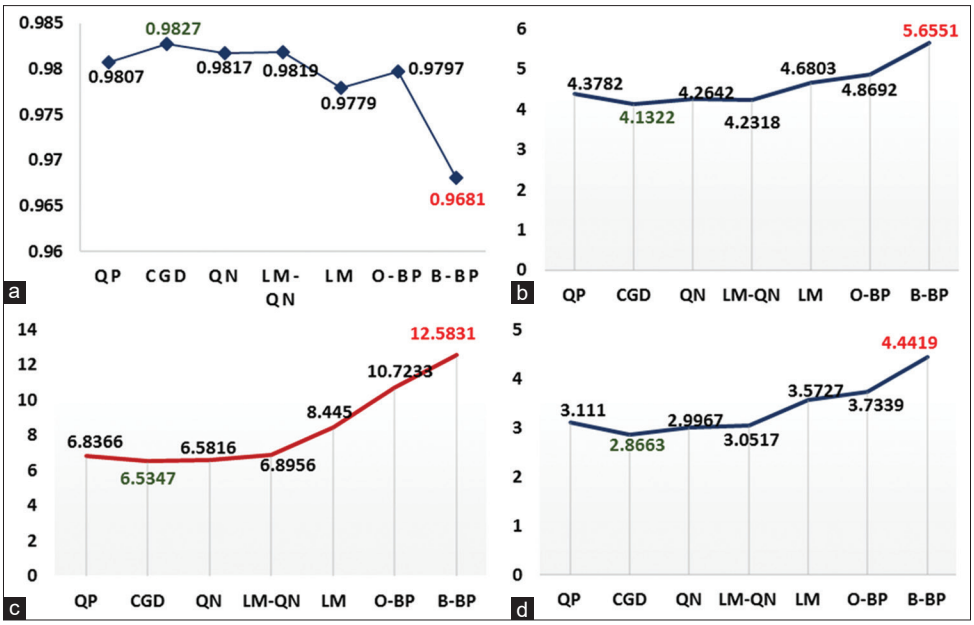

Figure 2: Performance comparison of training algorithms on the Brent oil dataset feed-forward neural network (12:4:1); (a) coefficient of determination values of the algorithms; (b) root mean square error values of the algorithms; (c) mean absolute percentage error values of the algorithms; (d) mean absolute error values of the algorithms

technique has a significant influence on the accuracy and reliability of forecasting by systematically evaluating seven commonly recognized training methods that use both deep and shallow NN designs.

As is clear from Tables 1-3 and Figure 2, across all of the algorithms, the CGD algorithm consistently outperformed others. CGD achieved the greatest R^2 values for the Brent oil and gold price datasets, indicating more explanatory ability. Furthermore, it provided the lowest value for RMSE, MAPE, and MAE. This demonstrates the capacity of the technique to efficiently remove systematic and random errors associated with forecasting by the resulting model. In the deep learning designs of 12:7:9:1 for Brent oil and 12:6:8:1 for gold, the performance of CGD outperformed all other algorithms tested, with R^2 of 0.9897 and 0.9971, respectively. Once again, CGD demonstrated excellence in shallow designs with indicative performance metrics that affirm its continued efficiency at varying levels of complexity of the model.

QN was the second-best outperforming algorithm for the most part and was similar to CGD, but with a slightly larger error margin. This clearly indicates that although QN can serve as a good substitute for algorithms due to its accuracy and reliability, it still lags behind CGD with regard to accuracy and consistency of optimization results for non-linear patterns common to financial time series. QP was also quite good and followed the performance patterns of CGD and QN. On the other hand, the poorest-performing method appeared to be BBP. The small values of R^2 and high error rates for all datasets and network topologies indicate that BBP does not prove to be efficient with convergence as well as error minimization, specifically for non-stationary data sets such as gold and Brent oil prices.

An interesting finding that has been obtained here is that the ordering of algorithms remained quite stable irrespective of the design being shallow or deep, and that the CGD algorithm was always the best training algorithm for every design. This points toward the fact that the performance differences are a natural part of the optimization strategy of the algorithms concerned. This has been found to have the following implications. First, they show that algorithm selection is equally important as network architecture design in obtaining high predictive accuracy. Second, they emphasize the significance of using optimization methods that balance computational efficiency with convergence dependability. CGD's constant dominance across numerous datasets and settings makes it an excellent choice for researchers looking to use NNs for forecasting financial tasks.

CONCLUSION

The study's results indicate that the CGD algorithm is the best algorithm for training the FFNN to forecast financial time series, in both cases, the shallow NN and the deep learning model. Because it gives the highest value of R^2 and the lowest value of RMSE, MAPE, and MAE compared to the other algorithms. Furthermore, QN is ranked the second algorithm, whereas BBP, with the lowest value of R^2 and the highest value of RMSE, MAPE, and MAE, is the poorest algorithm for training the FFNN for forecasting monthly time series.

DATA AVAILABILITY

Both datasets were extracted from the World Bank website.

FUNDING

The author did not receive support from any organization for the submitted work.

REFERENCES

1. G. P. Zhang. Business forecasting with artificial neural networks. In: *Neural Networks in Business Forecasting*. IGI Global Scientific Publishing, United States, 2004.
2. R. Fletcher. *Practical Methods of Optimization*. John Wiley and Sons, United States, 2000.
3. G. Zhang, B. E. Patuwo and M. Y. Hu. Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, vol. 14, pp. 35-62, 1998.
4. D. G. Baur and T. K. McDermott. Is gold a safe haven? International evidence. *Journal of Banking and Finance*, vol. 34, no. 8, pp. 1886-1898, 2010.
5. L. Kilian. The economic effects of energy price shocks. *Journal of Economic Literature*, vol. 46, no. 4, pp. 871-909, 2008.
6. M. K. Awang, M. R. Ismail, M. Makhtar, M. N. A. Rahman and A. R. Mamat. Performance comparison of neural network training algorithms for modeling customer churn prediction. *International Journal of Engineering and Technology*, vol. 7, no. 2.15, pp. 35-37, 2018.
7. R. Tabbussum and A. Q. Dar. Comparative analysis of neural network training algorithms for the flood forecast modelling of an alluvial Himalayan river. *Journal of Flood Risk Management*, vol. 13, pp. e12656, 2020.
8. C. K. Arthur, V. A. Temeng and Y. Y. Ziggah. Performance evaluation of training algorithms in backpropagation neural network approach to blast-induced ground vibration prediction. *Ghana Mining Journal*, vol. 20, no. 1, pp. 20-33, 2020.
9. R. Bergendal and A. Rohlén. *A Comparison of Training Algorithms When Training a Convolutional Neural Network for Classifying Road Signs*. KTH Royal Institute of Technology, Sweden, 2019.
10. I. N. Da Silva, D. H. Spatti, R. A. Flauzino, L. H. B. Liboni and S. F. *Dos Reis Alves. Artificial Neural Networks: A Practical Course*. Springer, New York, 2016.
11. S. C. Wang. *Interdisciplinary Computing in Java Programming*. Springer, New York, 2003.
12. X. H. Yu, G. A. Chen and S. X. Cheng. Dynamic learning rate optimization of the backpropagation algorithm. *IEEE Transactions on Neural Network and Learning*, vol. 6, pp. 669-677, 1995.
13. S. E. Fahlman. *An Empirical Study of Learning Speed in Back-Propagation Networks*. *Techical Report*, Carnegie-Mellon University, Pittsburgh, 1988.
14. T. M. Bafitlhile, Z. Li and Q. Li. Comparison of Levenberg Marquardt and conjugate gradient descent optimization methods for simulation of streamflow using artificial neural network. *Advances in Ecological and Environmental Research*, vol. 3, pp. 217-237, 2018.
15. J. R. Shewchuk. *An Introduction to the Conjugate Gradient Method without the Agonizing Pain*. Carnegie Mellon University, Pittsburgh, PA, 1994.
16. C. J. Li and L. Yan. Mechanical system modelling using recurrent neural networks via quasi-Newton learning methods. *Applied Mathematical Modelling*, vol. 19, no. 7, pp. 421-428, 1995.
17. Z. Cömert and A. Kocamaz. A study of artificial neural network training algorithms for classification of cardiotocography signals. *Bitlis Eren University Journal of Science and Technology*, vol. 7, no. 2, pp. 93-103, 2017.
18. R. Malouf. A Comparison of Algorithms for Maximum Entropy Parameter Estimation. In: *COLING-02: The 6th Conference on Natural Language Learning*, 2002.
19. G. Yu, C. H. Farquharson, Q. Xiao and M. Li. Two-dimensional anisotropic magnetotelluric inversion using a limited-memory quasi-Newton method. *Geophysics*, vol. 87, pp. E13-E34, 2022.
20. D. R. S. Saputro and P. Widyaningsih. Limited memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) method for the parameter estimation on geographically weighted ordinal logistic regression model (GWOLR). *AIP Conference Proceedings*, vol. 1868, pp. 040009, 2017.
21. H. P. Gavin. *The Levenberg-Marquardt Algorithm for Nonlinear Least Squares Curve-Fitting Problems*. *Technical Reports*. Duke University, United States, 2019.
22. M. I. Lourakis. A brief description of the Levenberg-Marquardt algorithm implemented by levmar. *Foundation Research Technology*, vol. 4, no. 1, pp. 1-6, 2005.
23. Y. Tian, Y. Zhang and H. Zhang. Recent advances in stochastic gradient descent in deep learning. *Mathematics*, vol. 11, pp. 682, 2023.
24. L. Bottou. Stochastic gradient descent tricks. In: *Neural Networks: Tricks of the Trade*. 2nd ed. Springer, Berlin, pp. 421-436, 2012.
25. M. S. Al-Duais and F. S. Mohamad. A review on enhancements to speed up training of the batch back propagation algorithm. *Indian Journal of Science and Technology*, vol. 9, no. 46, pp. 1-10, 2016.
26. M. S. Al-Duais, F. S. Mohamad, M. Mohamad and M. N. Husen. Enhancement processing time and accuracy training via significant parameters in the batch BP algorithm. *International Journal of Intelligent Systems and Applications*, vol. 10, no. 1, pp. 43, 2020.