Cascade correlation neural network with deterministic weight modification for predicting stock market price

K VELUSAMY¹ and R AMALRAJ²

¹²Department of Computer Science
¹²Sri Vasavi College, Erode, Tamilnadu, India
¹kvelusamy85@gmail.com and ²r_amalraj05@yahoo.co.in

Abstract. A new weight updating approach is proposed for cascade correlation neural network (CCNN) in this paper. The deterministic weight modification (DWM) algorithm is used to adjust the connection weights of CCNN. The introduced new method can improve the global convergence capability of the conventional CCNN and optimally reduced the system error. The proposed DWM+CCNN prediction algorithm is applied well-known stock market dataset in order to evaluate the robustness and efficiency. The experimental results are confirmed that the proposed DWM+CCNN algorithm is achieves higher performance in terms of convergence rate and the capability of global converges.

Keywords: Artificial Neural Networks, Back Propagation Neural Network, Cascade Correlation Neural Network, Deterministic Weight Modification, Stock Index Prediction

1. Introduction

Artificial Neural Networks (ANNs) is well-known information processing system that can approximate any continues function and its universal approximation property makes suitable for nonlinear model [1]. The architecture of the ANNs is an organized by the number of network layers including the input layer, hidden layer, and output layer. An efficient architecture of ANNs is determined by the network layers, connection weights, learning rate, and learning algorithm [2]. However, the number of neurons in both input and output layer is decided according to a given solution. In the hidden layer, deciding the number of neurons is also a crucial and challenging task and another N-P hard problem [3, 4]. If selecting a more number of neurons in the hidden layer, the learning process falls into over fitting. Similarly, if selecting a small number of neurons in the hidden layer, the learning process falls into under fitting [5, 6].

In ANN, the back propagation neural network (BPNN) is well-known neural network architecture because of its straightforwardness and high problem-solving capability. It is
supervised and feed forward neural network (FFNN) architecture. However, the BPNN algorithm is to decide its structure before starting the learning process. In this circumstance, there are many drawbacks, including slow convergence rate, failure to find global minima and fall into overtraining or undertraining [5]. Generally, the size of the BPNN architecture is determined by using trial-and-error technique [7]. On the other hand, many research works have been carried out to choose the proper network structure of BPNN [3, 5, 8]. However, the convergence rate of these enhanced BPNN architectures is slow because there is no thumb rule to decide the proper structure [9].

The pruning and constructive neural networks are an alternative way to determine the network sizes that can be automatically found the optimal size of architecture [4, 10]. From these two methods, the pruning algorithm starts with large networks then removing the hidden neuron one by one until the network achieves the optimal error. The constructive neural network algorithm learns begin with zero hidden neuron, then add the hidden neuron one by one until the network is obtaining optimal results. Compared with these two architectures, the constructive neural network is more efficient [11].

The pruning and constructive neural networks are an alternative way to determine the network sizes that can be automatically found the optimal size of architecture [4, 10]. From these two methods, the pruning algorithm starts with large networks then removing the hidden neuron one by one until the network achieves the optimal error. The constructive neural network algorithm learns begin with zero hidden neuron, then add the hidden neuron one by one until the network is obtaining optimal results. Compared with these two architectures, the constructive neural network is more efficient [11].

The pruning and constructive neural networks are an alternative way to determine the network sizes that can be automatically found the optimal size of architecture [4, 10]. From these two methods, the pruning algorithm starts with large networks then removing the hidden neuron one by one until the network achieves the optimal error. The constructive neural network algorithm learns begin with zero hidden neuron, then add the hidden neuron one by one until the network is obtaining optimal results. Compared with these two architectures, the constructive neural network is more efficient [11].

The CCNN is well-known CNN technique developed by S.E Fahlman and C. Lebiere (1990) [14]. The major goal of the CCNN is automatically increased the size of network structure during the learning process [14, 15]. Two types of the key ideas are in the conventional CCNN. First, cascade structure is leads to create a powerful feature detector. Second, the objective function is used to create and train the new hidden units is a new one. The aim of an objective function is to maximize the degree of correlation between the recent hidden units’ output and the remaining network output error.

The CCNN is well-known CNN technique developed by S.E Fahlman and C. Lebiere (1990) [14]. The major goal of the CCNN is automatically increased the size of network structure during the learning process [14, 15]. Two types of the key ideas are in the conventional CCNN. First, cascade structure is leads to create a powerful feature detector. Second, the objective function is used to create and train the new hidden units is a new one. The aim of an objective function is to maximize the degree of correlation between the recent hidden units’ output and the remaining network output error.

In this research work, the DWM algorithm is used to change the weights of the CCNN in order to enhance the performances. The major aim of this research work is to implement an efficient prediction model that can automatically adjust the network size according to a given solution with desirable accuracy. The contributions of proposed research work are as follows:

- The proposed model is enhancing the convergence rate and the prediction accuracy.
- The proposed model is managing the over fitting or under fitting during learning process.
- The efficient prediction model is constructed with the fast-learning process and low memory consumptions.

The paper is structured as follows: In section 2, conventional CCNN architecture is described. The detail of the proposed method is described in section 3. The experimental outcome and discussion are demonstrated in section 4. Finally, the conclusion is presented in the section 5.

2. Cascade Correlation Neural Networks (CCNN)

The CCNN is well-known CNN technique developed by S.E Fahlman and C. Lebiere (1990) [14]. The major goal of the CCNN is automatically increased the size of network structure during the learning process [14, 15]. Two types of the key ideas are in the conventional CCNN. First, cascade structure is leads to create a powerful feature detector. Second, the objective function is used to create and train the new hidden units is a new one. The aim of an objective function is to maximize the degree of correlation between the recent hidden units’ output and the remaining network output error.
The CCNN has many advantages compared with non-constructive neural networks, including learning speed is high, own size, depth and topologies are determined dynamically. There is not perform the back propagation of the error and only one-layer weights are trained when hidden neurons is added to the active network. Initially, the learning process of the CCNN is started with minimal network. The input unit is directly linked to the one or more output units through adjustable weight and bias input is +1, but without hidden neurons. At end of the iteration, calculates the error values. If the network error achieves a valid error, then discontinue the learning process. Otherwise, the new hidden unit adds one by one to the active networks up to reach reasonable errors [16].

3. Proposed DWM+CCNN Prediction model

The goal of the weight adjustment is to reduce the network error and improving the architecture performances. In some of the improved CCNN, fast learning algorithms, namely QPROP [14, 16, 17], RPROP [18], and GA [19] are used for its weight adjustments during the learning process for reducing error and convergence speed. Still, the above-mentioned algorithms have natural shortcomings such as poor convergence, local mini-ma. To enhance the performance of the DWM+CCNN prediction algorithm, DWM is applied for adjusting weights of the standard CCNN. The purpose of DWM is to adjust the weights of network in a controlled mode for the period of the learning process. The DWM learning algorithm is used to find the global value, least error signal and very rapid convergence rate. Thus, DWM process is to decrease the error by altering the weights of the CCNN in a deterministic way [20]. The figure 1 shows the overview of proposed DWM+CCNN prediction model.

3.1 Weight adjustment between hidden and output layer

In order to make the outputs so close to the expected outcome, the weight modification is implemented between the hidden and output layer. The specific output \( o \) neuron of the output layer of the error is selected above the MSE as follows,

\[
E_o(i) = \frac{1}{2} \sum_{p=1}^{P} (y_{po} - y_{po(i)})^2 \geq E(i)/o
\]  

(1)

The objective is to change the weights \( W_{po(i+1)} \) associated to the output \( o \) neuron and remaining weights in the network will unaffected. The outcome of the after that iteration will become close to the predictable output. This weight change process can speed up of convergence rate and get away from local minimum. It is noted that \( o \) is any output neuron for fulfilling the equation (1). Thus \( o \) can be either randomly chosen among all appropriate values fulfilling equation (1) or can be one \( i^{th} \) the largest MSE. (i.e \( E_o(i) > E_o(i) \forall 0 \)). The output \( y_{po(i+1)} \) in next iteration wants to be close up to the objective value \( t_{po} \) the predictable output is measured as

\[
y_{po(i+1)} = y_{po(i)} + \beta (t_{po(i)} - y_{po(i)}) = \beta t_{po(i)} - (1 - \beta) p_{po(i)}
\]  

(2)
Where, \( 0 < \beta < 1 \) for all data points \( p \). To obtain the next iteration weights \( W_{ko}^{(i+1)} \) from the above, the output becomes

\[
y_{po}(i+1) = f(\sum_{k=1}^{K} w_{ko} y_{pk}(i+1)) = f(\sum_{k=1}^{K} w_{ko} \gamma_{pk}(i+1) + \Delta w_{ko}(i+1) \gamma_{pk}(i+1))
\]

(3)

\[
\sum_{k=1}^{K} \Delta w_{ko}(i+1) y_{pk}(i+1) = \varepsilon_{po}
\]

(4)

Where,

\[
\varepsilon_{po} = f^{-1}(y_{po}(i+1)) - \sum_{k=1}^{K} w_{ko}(i) \gamma_{pk}(i+1)
\]

(5)

\[
\varepsilon_{po} = f^{-1} \left( \ln \left( \frac{1}{1 - \delta} \right) \right)
\]

(6)

Therefore, the following matrices to compute \( W_{ko}^{(i+1)} \) by considering Equation (3),

\[
\begin{bmatrix}
\tau_{11}(i+1) & \ldots & \tau_{1k}(i+1) & \ldots & \tau_{1K}(i+1) \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
\tau_{P1}(i+1) & \ldots & \tau_{P1}(i+1) & \ldots & \tau_{PK}(i+1) \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
\tau_{P1}(i+1) & \ldots & \tau_{P1}(i+1) & \ldots & \tau_{PK}(i+1)
\end{bmatrix}
\begin{bmatrix}
\Delta w_{ko}(i+1) \\
\vdots \\
\Delta w_{ko}(i+1) \\
\vdots \\
\Delta w_{ko}(i+1)
\end{bmatrix}
= \begin{bmatrix}
\gamma_{po} \\
\vdots \\
\gamma_{po} \\
\vdots \\
\gamma_{po}
\end{bmatrix}
\]

(7)

The expected output by updating \( W_{ko}^{(i+1)} \) with the pseudo inverse obtained least squared error condition [21].

\[
W_{ko}^{(i+1)} = \frac{\sum_{p=1}^{P} \gamma_{po} \tau_{pk}(i+1)}{\sum_{p=1}^{P} \tau_{pk}(i+1)}
\]

(8)

both \( W_{ko}^{(i+1)} \) and its squared error are calculate for each hidden neurons \( k \) as follows,

\[
E_{o}^{(k)}(i+1) = \frac{1}{2} \sum_{p=1}^{P} \gamma^{2}_{po} |y_{po} - y_{po}(i+1)|^2
\]

(9)

The current hidden neurons, is selected to update \( W_{ko}^{(i+1)} \) and if its root MSE is the smallest than the previous iteration error, stop the training the network model, Otherwise the training process goes on and new hidden neuron adds to the current networks. i.e \( E_{o}^{(k)}(i+1) \leq E_{o}^{(k)}(i+1) \) for all \( k \).

3.2 Weight adjustment between the input layer and hidden layer

The idea of the weight changing between input layer and hidden layer is to get away from the problem of local minima by reducing error of network into a targeted value in a deterministic way. The predictable value of the network is defined in the next iteration. The network error
of the subsequently iteration will be earlier to the predictable value. The DWM algorithm is fall into local optima at \( i \)th iteration. An error of the output layer is selected above the MSE

\[
E_p(i) = \frac{1}{2} \sum_{v=1}^{p} (t_{po} - y_{po}(i+1))^2 \geq \frac{E(i)}{p}
\]

(10)

The goal of the weight modification of \( W_{ko}(i+1) \) for \( n=1,2,...N \) is the output of next iteration so closed with predictable output and escaped from local minima. Since the output \( y_{po}(i+1) \) of the output layer in subsequently iteration required to be close to the objective value \( t_{po}(i+1) \), consider the expected output as

\[
E_{po}(i+1) = \frac{1}{2} \sum_{v=1}^{p} (t_{po} - y_{po}(i+1))^2 \geq \frac{\lambda E(i)}{p}
\]

(11)

Where, \( 0<\lambda<1 \). The error of each data point is measured by the average values. Thus, for any exacting output neurons \( O \).

\[
E_{po}(i+1) = \frac{1}{p} \sum (t_{po} - y_{po}(i+1))^2 = \frac{\sum E(i)}{p}
\]

(12)

It can select randomly or \( E_{po}(i+1) \leq E_{po}(i+1) \) for all \( O \). After choosing

\[
O_{po}(i+1) = \begin{cases} 1 - \frac{\sum E(i)}{PO} & \text{if } t_{po} = 1 \\ \frac{\sum E(i)}{PO} & \text{if } t_{po} = 0 \end{cases}
\]

(13)

We also have,

\[
O_{po}(i+1) = f^K \sum_{k=1}^{K} w_{ko}(i) O_{pk}(i+1) = f^K \sum_{k=1}^{K} w_{ko} O_{pk}(i+1) + w_{po}(i+1) O_{po}(i+1)
\]

(14)

Every hidden neuron will not be customized, however only one \( k \) hidden neuron will be chosen at a time. Such that \( O_{pk}(i) \) is close to the 2-tail area for all \( k \).

\[
k \cdot \min(\sigma_{pk}(i),1-\sigma_{pk}(i)) \leq \min(\sigma_{pk}(i-1),1-\sigma_{pk}(i-1))
\]

(15)

Additionally, while now near to the local minimum, therefore, \( \Delta \sigma_{pk}(i+1) = 0 \) for \( \forall k \neq k \). With help of these above two simplifications, we have

\[
\Delta \sigma_{pk}(i+1) = f^{-1}(\sigma_{pk}(i+1) - \sum_{k=1}^{K} w_{ko}(i+1) \sigma_{pk}(i+1)) / w_{ko}(i+1)
\]

(16)

Hence, by using equation, \( \sigma_{pk}(i+1) \) can be obtained as follows,

\[
\sigma_{pk}(i+1) = \sigma_{pk}(i) + \Delta \sigma_{pk}(i+1)
\]

(17)

Furthermore,
\[
\sigma_{pk}(i+1) = f\left(\sum_{n=1}^{N} \pi_{nk}(i) \cdot x_{mn}\right) = f\left(\sum_{n=1}^{N} \pi_{nk}(i) \cdot y_{pm} + \sum_{n=1}^{N} \pi_{nk}(i+1) \cdot y_{pm}\right)
\]  

(18)

And, thus, \(\Delta \sigma_{nk}(i+1)\) can be obtained for all as

\[
\Delta \sigma_{nk}(i+1) = \frac{\pi_{pk} \cdot \sum_{n=1}^{N} \pi_{nk}}{\sum_{n=1}^{N} \pi_{nk}^2}
\]

(19)

Where,

\[
\pi_{pk} = f^{-1}(\sigma_{pk}(i+1) - f^{-1}(\sigma_{pk}(i))
\]

(20)

The process of weight modification is done in two related ways in CCNN using DWM. That is, they both changes of weight are to decrease the total error of the network model.

However, the DWM is used to update weight between hidden layer and output layer for decreasing comprehensively by affecting the output of the output layer. In the output neurons

![Figure. 3.1: Overview of the proposed DWM+CCNN prediction model](image-url)
the target output and predictable output is \( y_{po(i)} + \beta (t_{po(i)} - y_{po(i)}) \). \( \beta \) must not be more near to 1. If not, it will cause unbalanced behavior. When \( \beta \) is set to 1, the predictable yield will become the objective value. The DWM has an improved ability of global convergence rate and accelerate the convergence speed of the learning algorithm.

4. Experimental results and analysis

In this section, the efficiency of the proposed DWM-CCNN prediction is applied on two benchmark stock market indices datasets. Experimental outcomes are conducted with the help of MATLAB R2015b. The robustness of proposed DWM+CCNN prediction model is compared with conventional CCNN and BPNN.

4.1 Datasets collections

The data sets are covering the periods between January 3, 2005 and June 30, 2018. The proposed DWM-CCNN prediction model is applied to forecast the two Indian stock market indexes including Nifty 50 dataset and S&P BSE Sensex dataset. The stock market index prediction is a most challenging task in order to support investors to produce more profit. The dataset has many prices such as opening price, the highest price, the average price, low price and closing price.

Hence, four variables are considered as an input value to the prediction algorithm such as the Open Price (OP), High Price (HP), Average Price (AP), Low Price (LP) and then one variable is considered as a target output value such as closing price (CP). The data sets are collected from Bombay Stock Exchange (BSE) [22] and National Stock Exchange (NSE) [23] websites respectively. Each data set consists of 3350 trading days and it transformed into two phases, namely training phase and testing phase. From the datasets, 2850 data points are used for training part and the remains 500 data samples are used for the testing part.

4.2 Performance measures

The strength of the proposed DWM+CCN prediction method is calculated by statistical indicators such as Root MSE (RMSE), Mean Absolute Error (MAE), and Directional Symmetry (DS).

\[
RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{N} (t_i - y_i)^2}
\]  
(21)

\[
MAE = \frac{1}{n} \sum_{i=1}^{N} |t_i - y_i|
\]  
(22)

\[
DS = \sqrt{\frac{100}{n} \sum_{i=1}^{N} d_i \cdot d_i} = \begin{cases} 
1 & (y_i - y_{i-1}) (t_i - t_{i-1}) \\
0 & otherwise
\end{cases}
\]  
(23)
Where, \( t_i \) is represents the target value of given datasets, \( y_i \) is represents the predicted output of ANN prediction algorithm and \( N \) - represent the total number of data samples. The smallest value of RMSE and MAE are considered as the best performance of the time series analysis predictor. The DS present the exactness of the prediction model in term of percentage. In this study, the weights are initialized by small random values and termination conditions is set 0.005 (Error value). The maximum epoch is set as 2000 for all compared algorithms.

4.3 Discussion: The results of the proposed model that the DWM-CCNN is compared with existing standard CCNN and BPNN. In this research paper, single hidden layer is considered for stock market prediction and the quantity of hidden neurons is set 15 [24,25]. The experimental outcomes are obtained by the average values of 50 trials for comparisons in each algorithm.

| Methods     | S&P BSE Sensex | Nifty 50 |
|-------------|----------------|----------|
|             | RMSE | MAE | DS(%) | RMSE | MAE | DS(%) |
| DWM-CCNN    | 0.0185 | 0.0100 | 87.42 | 0.0096 | 0.0023 | 91.34 |
| CCNN        | 0.0626 | 0.0244 | 83.56 | 0.0124 | 0.0089 | 82.23 |
| BPN         | 0.0975 | 0.0844 | 80.36 | 0.0139 | 0.0094 | 79.34 |

Figure 4.1: Performance analysis of the prediction models for Nifty 50 datasets
Figure 4.2: Performance analysis of the proposed prediction models for S & P datasets

The statistical parameter values of the standard prediction algorithms are demonstrated in Table 1 and graphical representation shows in Figure 2 and Figure 3 for both S & P Sensex and Nifty 50 data sets respectively. The proposed model is produced higher prediction accuracy compared with CNN and BPNN prediction methods. The optimal hidden neurons are an essential task which is determines the efficiency of the architecture. Hence, the proposed prediction model obtained the higher forecast accuracy and low error with 9 hidden neurons when applying for S & P Sensex and 8 hidden neurons when applying Nifty 50 datasets. Commonly, the proposed system has reduced the slight enhanced prediction accuracy compared with CCNN and average enhancement accuracy compared with BPNN on both benchmark stock market indices datasets in terms of prediction accuracy and choosing appropriate hidden neurons. From the research works, we conclude that proposed method can prevent from the overtraining or undertraining of the learning process and produced higher prediction accuracy with less error.

5. Conclusions

The present research work is empirically applied to the stock market predication. The performance of DWM+CCNN is compared with standard CCNN and BPNN. The performance of proposed prediction model is analyzed using statistical measures such RMSE, MAE and DS. The proposed DWM-CCNN prediction algorithm is produced high convergence rate with optimal error, finding a suitable optimal number of hidden neurons and producing higher prediction accuracy. The proposed model has formed a significant enhancement with higher accuracy when compared with standard CCNN and BPNN. Future enhancement of this research work is applied to deep neural network (DNN) for stock market prediction in order to produce more efficient predication results. Because the conventional CCNN takes more computation time and produced low accuracy. Hence, the DNN is significant method that can be produced high accuracy when compared with normal neural network algorithms.

References
[1] Li F, Qiao J, Han H, and Yang C 2016 A self-organizing cascade neural network with random weights for nonlinear system modeling Applied soft computing 42 184-193
[2] Amalraj R and Dharmalingam M 2015 A work point count system coupled with back-propagation for solving double dummy bridge problem Neurocomputing 168 160-178.
[3] Sheela K G and Deepa S N 2013 Review on methods to fix number of hidden neurons in neural networks Mathematical Problems in Engineering.
[4] Kwok T-Y and Yeung D-Y 1997 Constructive algorithms for structure learning in feedforward neural networks for regression problems IEEE transactions on neural networks 8 630-645
[5] Madhiarasan M and Deepa S 2016 A novel criterion to select hidden neuron numbers in improved back propagation networks for wind speed forecasting Applied intelligence 44 878-893
[6] Teoh E J, Tan K C, and Xiang C 2006 Estimating the number of hidden neurons in a feedforward network using the singular value decomposition IEEE Transactions on Neural Networks 17 1623-1629
[7] Huang G, Song S and Wu C 2012 Orthogonal least squares algorithm for training cascade neural networks IEEE Transactions on Circuits and Systems I: Regular Papers 59 2629-2637
[8] Madhiarasan M and Deepa S 2017 Comparative analysis on hidden neurons estimation in multi layer perceptron neural networks for wind speed forecasting Artificial Intelligence Review 48 449-471
[9] Kalaiselvi K, Velusamy K, and Gomathi C 2018 Financial prediction using back propagation neural networks with opposition based learning Journal of Physics: Conference Series 1142 012008.
[10] Castellano G, Fanelli A M, and Pelillo M 1997 An iterative pruning algorithm for feedforward neural networks IEEE transactions on Neural networks 8 519-531
[11] Xi L, Muzhou H, Lee M H, Li J, Wei D, Hai H, and Wu Y 2014 A new constructive neural network method for noise processing and its application on stock market prediction Applied Soft Computing 15 57-66
[12] Vivarelli F, Fariselli P, and Casadio R 1997 The prediction of protein secondary structure with a cascade correlation learning architecture of neural networks Neural Computing & Applications 6 57-62
[13] Ng S, Cheung C, and Leung S 2004 Deterministic weight modification algorithm for efficient learning IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541) IEEE
[14] Fahlman S E and Lebiere C 1990 The cascade-correlation learning architecture," in Advances in neural information processing systems
[15] Velusamy K and Amalraj R 2017 Performance of the cascade correlation neural network for predicting the stock price Second International Conference on Electrical, Computer and Communication Technologies (ICECCT), IEEE.
[16] Shultz T R and Fahlman S E Cascade-Correlation Encyclopedia of Machine Learning Editor, Springer US.
[17] Fahlman S E 1988 An empirical study of learning speed in back-propagation networks
[18] Treadgold N K and Gedeon T D 1997 A cascade network algorithm employing progressive RPROP International Work-Conference on Artificial Neural Networks Edittor Springer

[19] Liang H and Dai G 1998 Improvement of cascade correlation learning Information Sciences 112 1-6.

[20] Ng S-C Cheung C-C and Leung S-H 2004 Magnified gradient function with deterministic weight modification in adaptive learning IEEE Transactions on Neural Networks 15 1411-1423

[21] Ben-Israel A and Greville T N 2003 Generalized inverses: theory and applications (Springer Science & Business Media).

[22] Wang J Wu L Choo K-K R and He D 2019 Blockchain Based Anonymous Authentication with Key Management for Smart Grid Edge Computing Infrastructure IEEE Transactions on Industrial Informatics

[23] Zhang Y Deng R H Liu X and Zheng D 2018 Blockchain based efficient and robust fair payment for outsourcing services in cloud computing Information Sciences 462 262-277

[24] Sin-Chun N and Chi-Cheung S-H L 2004 Magnified gradient function with deterministic weight modification in adaptive learning Neural Networks 6 1411-1423

[25] Dai W Wu J-Y and Lu C-J 2012 Combining nonlinear independent component analysis and neural network for the prediction of Asian stock market indexes Expert systems with applications 39, 4444-4452.