Proceedings of the 24th International Conference on Automation & Computing, Newcastle University, Newcastle upon Tyne, UK, 6-7 September 2018

This is the Pre-Published Version.

The following publication W. A. Khan, S. -H. Chung and C. Y. Chan, "Cascade Principal Component Least Squares Neural Network Learning Algorithm," 2018 24th International Conference on Automation and Computing (ICAC), Newcastle Upon Tyne, UK, 2018, pp. 1-6 is available at https://doi.org/10.23919/IConAC.2018.8748964.

Cascade Principal Component Least Square Neural Network Learning Algorithm

Waqar Ahmed Khan, Sai-Ho Chung, Ching Yuen Chan
Department of Industrial and Systems Engineering
The Hong Kong Polytechnic University
Hong Kong
waqarahmed.khan@connect.polyu.hk, nick.sh.chung@polyu.edu.hk, cy.chan@connect.edu.hk

***Abstract*— Cascading correlation learning (CasCor) is a constructive algorithm which determines its own network size and typology by adding hidden unit one at a time based on covariance with output error. Its generalization performance and computational time depends on cascade architecture and iteratively tuning of connection weights. CasCor was developed to address the slowness of backpropagation (BP), however, recent studies have concluded that in many applications, CasCor generalization performance does not guarantee to be optimal. Apart from BP, CasCor learning speed can be considered slow because of iterative tuning of connection weights by numerical optimization techniques. Therefore, this paper addresses CasCor bottlenecks and aims to introduce a new algorithm with improved cascade architecture and tuning free learning method to achieve objectives of better generalization performance and fast learning ability. The proposed algorithm determines network input connection weights by orthogonally transforming a set of correlated input units into uncorrelated hidden units and output connection weights by considering hidden layer and the output units in a linear relationship. This research work is unique and innovative than previous because it does not need any prior random generation and repeatedly tuning of connection weights. Comparative study on complex nonlinear regression approximation function and classification tasks proof that proposed algorithm has achieved better generalization performance and learn twenty-five times faster than CasCor.**

Keywords- cascading correlation learning; connection weights; network topology; principle component analysis; ordinary least square; cascade principal component least square

I. INTRODUCTION *(HEADING 1)*

Feedforward neural networks (FNNs) are supervised universal approximator learning algorithms and have been extensively applied in many areas such as classification and regression. Selection of FNNs depend upon application area with purpose to achieve better generalization performance without overfitting and underfitting in shortest possible time. Too many hyperparameters and its adjustment by iterative learning algorithms make FNNs complex and generalization performance become inconsistent. The most popular learning algorithm for FNNs is backpropagation (BP) stochastic gradient descent, but it encounters low computational speed, adoption to local minima and convergence learning rate issues [1]. Researchers are extensively focused on two major areas for improvement of FNNs. The first area is to formulate methodologies that can improve generalization performance and, another area is to reduce the computational speed of algorithms to gain faster estimation.

Recently, the methodologies formulated to improve FNNs generalization performance are based on a trade-off between bias and variance. Srivastava et al. [2] proposed dropout technique to deal with overfitting by randomly dropping hidden units from the layers to reduce coadoption between hidden units and connection weights. The incorporation of ad hoc weight decays regularization shrink variance by penalizing connection weights towards zero to less likely fit training noise data. The suitable network size is always not obvious, too large and small size will not be able to learn, therefore, pruning involves training a network of large size than necessary and removing the parts from the final network that are unnecessary [3]. Krogh [4] suggested that overall accuracy and generalization performance of networks improves by neural networks ensembles. In uniform weights, the generalization error of ensembles is always less than individual neural network errors. Furthermore, the learning speed of FNNs has been improved by the development of second-order derivative numerical optimization learning algorithms such as conjugate gradient method [5], Marquardt-Levenberg [6] and quick prop [7]. The implementation of numerical optimization techniques in constructive learning algorithm has provided a promising faster learning speed as compared to BP fixed network topology.

The most popular constructive Cascade Correlation learning algorithm (CasCor) which determines its own network topology and size was first proposed by S.E. Fahlman and C. Lebiere [8]. It works by adding new hidden unit one at a time and maximize the covariance magnitude among hidden unit and output error. In most cases, second order quick prop iterative learning algorithm is used in CasCor to reach faster towards error function by taking much larger steps rather than infinitesimal small steps. CasCor constructive algorithm has several advantages over fixed networks and pruning methods that it can solve complex tasks, learn more quickly, determine its own network typology, more economical and no underfitting. All these advantages lead CasCor to attain training results much faster, however, it does not assure optimal generalization performance. Huang [9] explained that generalization performance of CasCor decreases with increasing network size. The input connection weights are

Identify applicable sponsor/s here. *(sponsors)*

trained to maximize the covariance magnitude of existing hidden units and output error which cannot guarantee maximum error reduction when a new hidden unit is added. Increasing the hidden units causes learning complicated and convergence becomes slow. The iteratively tuning of input and output connection weights before and after hidden unit generation is more time-consuming. The learning speed of CasCor is obvious over backpropagation neural network (BPNN) but generalization performance of CasCor depend upon the type of data structure and task. It has been observed that CasCor perform better in classification tasks as compared to regression tasks [10] and this makes it unsuitable for many application areas. Moreover, Kovalishyn et al. [11] experimental work on quantitative structure relationships data did not find any dramatically better performance compared to BPNN. The possible reasons are lack of criteria which need to stop the addition of hidden units when learning converges and improper handling of hidden units with too many parameters adjustments which cause poor generalization performance [12]. Different from comparison with BPNN, CasCor learning speed can be considered slow because of iterative numerical optimization learning algorithms.

This paper proposes new unique algorithm that addresses and improves CasCor generalization performance and learning ability bottlenecks by modifying its cascade architecture and introducing new tuning free learning method. The proposed algorithm named as Cascade Principal Component Least Square Neural Network Learning Algorithm (CPCLS) determine its own cascade architecture and connection weights at each layer of learning. The CPCLS is free from the implementation of any kind of numerical optimization learning algorithms which makes learning easy and efficient with no need of iteratively tuning parameters. CPCLS determines input connection weights by orthogonal transforming set of correlated input units into uncorrelated hidden units. The output connection weights are determined by linear transformation of uncorrelated hidden units into output units. The proposed algorithm converts nonlinearity of continuous function into linearity by optimally calculating weight parameters. This results in the selection of most appropriate hidden units that create a linear relationship to output. Previously, little attempt has been made to analytically calculate connection weights on both sides. Experimental results on popular extreme nonlinear benchmark problems of two spiral classification [13] task and SinC approximation function regression task reveals that CPCLS has obtained better generalization ability and learn twenty-five times faster than traditional CasCor.

The paper is organized as follows. Section II briefly explains learning methodology behind CasCor and its drawbacks, Orthogonal Linear Transformation and Ordinary Least Squares linear regression (OLS). Section III introduces newly proposed algorithm CPCLS. Section IV is about performance evaluation by comparing Casor with CPCLS on two spiral classification tasks and SinC approximation function regression task. Section V is related to discussion and conclusion.

II. LEARNING METHADOLOGIES

A. CasCor and its drawbacks

The main issue associated with FNNs applicability is their slow learning ability because of BP learning algorithm. The CasCor was proposed by S.E. Fahlman and C. Lebiere to address the slowness of FNNs [8]. They argue that FNNs faces constantly changing environment by changing connection weights at once which make it slow to move towards overall solution. Unlike traditional FNNs, CasCor combines two main concepts: Cascade architecture and learning algorithm. It architecture begins with a minimum network by adding hidden unit one at a time and learning connection weights to maximizes the covariance between added hidden unit and network output error.

The CasCor is illustrated in Fig. 1 which begins initially with no hidden units in the network. All the input units are directly connected to output units with randomly generated connection weights. The connection weights are tuned by Fahlman “quick prop” learning algorithm due to its property of taking larger steps to converge more quickly towards minimal error rather than backpropagation infinitesimal small steps. When training approaches an asymptote and there is no further error reduction, a hidden unit is added. For hidden unit with characteristics of maximum error reduction, candidate units are initially added which receives incoming connections from all input units and any pre-existing hidden units. The objective is to maximise covariance ***S*** between candidate units and output error prior connecting to output units:

$$S = \sum_{o} \left| \sum_{p} (V_p - \bar{V})(E_{p,o} - \overline{E_o}) \right| \qquad (1)$$

The ***S*** magnitude is maximized by computing derivative of ***S*** with respect to each incoming connection weights to a candidate unit:

$$\frac{\partial S}{\partial w_i} = \sum_{p,o} \sigma_o \left(E_{p,o} - \overline{E_o}\right) f'_p I_{i,p} \qquad (2)$$

The quick prop gradient ascent is adopted to tune the incoming connection weights (which are also known as input connection weights) to achieve highest ***S.*** When ***S*** stop improving, the candidate unit with highest ***S*** is selected as hidden unit and is connected to the output units by output connection weights while incoming connections are kept frozen. Again, quick prop learning algorithm is adopted to train the network output connection weights. This process continues, and hidden units are added one by one until error converges.

The major drawbacks of CasCor is it complex cascade architecture and tuning based learning algorithm to find best connection weights. It determines its own network topology by adding hidden units one at a time based on the maximum covariance between the hidden unit and output error, whereas, the output error is reduced by connecting input and hidden units to output units. This iteratively tuning connection weights separately on the

input side to maximize covariance and output side to reduce error decreases the generalization ability of network on new testing data. Although quick prop CasCor is much faster than traditional BPNN but still the learning speed can be considered slow because of iteration steps to find optimal parameters. The main issue associated with CasCor is their isolation learning in each hidden layer. When a new hidden unit is added, the previously added hidden units become less significant for error minimization objective function because of fact that they had already performed utmost error minimization in their specific hidden layers. The output error changes after every hidden unit addition and newly added hidden unit becomes more important and correlated to error as compared to previously added hidden units. This results in complex network to handle additional information in the form of testing data which reduce the generalization performance.

Therefore, this paper proposes improved cascade architecture and new tuning free learning algorithm for CasCor to improve its generalization performance and learning ability. The proposed algorithm CPCLS works by adding multiple uncorrelated hidden units one at time based on orthogonal linear transformation of input units. Furthermore, the output connection weights are calculated by ordinary least square (OLS) by considering hidden units and output units in linear relationship. CPCLS improve cascade architecture by connecting only newly added multiple hidden units of final hidden layer to output units and eliminate previously added output connection weights, whereas, learning improves by analytically calculating both input and output connections weight without any need of iterative tuning.

B. Orthogonal Linear Transformation and OLS

Suppose a training data samples with $(\boldsymbol{X}, \boldsymbol{Y})$, where $\boldsymbol{X}$ be input units matrix *m*x*n* and $\boldsymbol{Y}$ be output units matrix *m*x*q* with hidden units $\boldsymbol{H}$ of matrix *m*x*p*. The input connection weights are represented by *W* of matrix *n*x*p*, whereas, output connection weights are represented by β of matrix *p*x*q*.

The *n*-features $\boldsymbol{X}$ are orthogonally linear transform into new *p*-features space of uncorrelated $\boldsymbol{H}$ by Principal Component Analysis (PCA) algorithm [14]. It takes only input units into consideration for dimensionality reduction by calculating unknown parameter *W*. Each principal component is coordinate describing the amount of variance in the data. The components are selected with first component describing largest possible variance and so on. It calculates eigenvalue λ and its corresponding eigenvector on the covariance matrix $\boldsymbol{S}$, which is equal dimensional *d*x*d* matrix with each quantity representing covariance between *n*-features with diagonal quantities representing covariance for the same feature. For *n*-features variable, let $\boldsymbol{X}$ be *m*x*n* matrix with *m* representing observations and *n* gives a data features. The $\boldsymbol{S}$ can be calculated as:

$$\boldsymbol{S} = \frac{1}{m-1}\boldsymbol{X}^T\boldsymbol{X} \tag{3}$$

Where $\boldsymbol{X} = x_i - \bar{x}$, such that $\bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_i$ with each value represents the mean of *n* features in the dataset.

The eigenvector describes the coordinate system for new *p* features by reducing its dimensions equal to or less than *n* features. However, eigenvector selection depends on the highest value of λ. The λ with smaller value can be omitted as it contains least variance information. The λ is calculated from $\boldsymbol{S}$ matrix:

$$|\boldsymbol{S} - \lambda\boldsymbol{I}| = 0 \tag{4}$$

The corresponding eigenvector based on λ can be found by calculating for the component $\boldsymbol{W}$ in the equation:

$$\boldsymbol{S}W = \lambda W \tag{5}$$

The matrix *W* linearly transform *n* features $\boldsymbol{X}$ into new uncorrelated *p* features space $\boldsymbol{H}$:

$$\boldsymbol{H} = \boldsymbol{X}W \tag{6}$$

OLS [15] reduces the sum of square (SSE) error between observed and predicted variables by estimating unknown parameter β based on hidden units $\boldsymbol{H}$ and actual output $\boldsymbol{Y}$:

$$\boldsymbol{Y} = \boldsymbol{H}\beta + \varepsilon \tag{7}$$

According to ordinary least square theories, β can be calculated by:

$$\beta = (\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{Y} \tag{8}$$

Where $(\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T$ is Moore Penrose pseudo-inverse of matrix $\boldsymbol{H}$. For optimal estimation, the assumption exists that there should be no multicollinearity among hidden units $\boldsymbol{H}$.

Knowing the value of β, the predicted output $\hat{\boldsymbol{Y}}$ can be calculated from the inner product of $\boldsymbol{H}$ and β:

$$\hat{\boldsymbol{Y}} = \boldsymbol{H}\beta \tag{9}$$

The model fitness objective function SSE can be accomplished by optimally calculating input and output connection weights in forward step. Equation (5) and (8) plays a significant role in determining weight connections for newly proposed CPCLS.

III. CPCLS LEARNING ALGORITHM

The objective of CasCor is to improve the slowness of backpropagation learning algorithm which does not assure it superior generalization performance. This paper introduced an efficient algorithm CPCLS with better generalization and fast learning speed than popular quick prop CasCor. CPCLS improves the existing CasCor by modifying its cascade architecture and introduces new method of tuning free learning. Unlike traditional CasCor which only determine its own network topology, CPCLS determine its own network typology and optimal weight connections by only forward steps to error reduction. CPCLS does not need to create a direct linkage between the input-output units and to add hidden unit one at a time, the number of hidden units addition depends upon the task and to move more quickly towards estimation. CPCLS basic idea is to convert nonlinear data structure into linear for fast and efficient computation.

Like CasCor, CPCLS also combine the two idea of cascade architecture and learning. CPCLS architecture is

represented in Fig. 2. The architecture of CPCLS is improved form of CasCor. Firstly, different from CasCor, it does not need to create initially linear combination of input and output units. All the input units are connected to output units by adding hidden units. Secondly, multiple hidden units can be added to converge faster to estimation. Thirdly, the last hidden layer is only connected to output units and previous connections are eliminated to make network simpler. In terms of learning, CasCor depends upon numerical optimization learning algorithms to repeatedly tune weight connection parameters which is more time consuming and difficult to handle it convergence. CPCLS is tuning free method which zimproves the learning by analytically calculating weight connections in forward step without any need of delta method. CPCLS different from CasCor by having better improved cascade architecture and analytically calculate input and output connections weight rather than tuning by numerical optimization algorithms.

CPCLS initialize with training data input units $\boldsymbol{X}$, output units $\boldsymbol{Y}$, and hidden units $\boldsymbol{H}$ such that $p \leq n$. For *input connection weights* determination, it orthogonally linearly transforms set of correlated $\boldsymbol{X}$ n-features into new space of uncorrelated $\boldsymbol{H}$ p-features by eigen decomposition of its covariance square matrix (3) such that $p \leq n$. The corresponding eigenvectors (5) with highest λ (4) value that explains maximum variance in data are selected as input connection weights W. $\boldsymbol{H}$ are computed (6) by performing activation function on the inner product of $\boldsymbol{X}$ and W. The advantage of this process is that it generates $\boldsymbol{H}$ explaining maximum variance in data with no multicollinearity to reduce target error more fast and efficient due to less complicated calculations. Next step is to determine *output connection weights* by considering the hidden unit in linear relationship to output units. The unknown output connection weights parameters β are determined (8) by the inner product of Moore Penrose pseudo-inverse of $\boldsymbol{H}$ and $\boldsymbol{Y}$. Knowing the value of β, output $\widehat{\boldsymbol{Y}}$ is estimated by linear conversion of hidden unit $\boldsymbol{H}$ through β such as $\boldsymbol{H}\beta$. The objective is to minimize error function at fast learning speed with better generalization performance:

$$E = \frac{1}{m}\sum_{i=1}^{m}\left(\widehat{\boldsymbol{Y}} - \boldsymbol{Y}\right)^2 \qquad (10)$$

If E is less than defined target error e, the CPCLS will stop, else, hidden layers will be added until the desired performance is not achieved. In proceeding hidden layers, the newly added H receives all incoming connections from input units and any existing hidden units, whereas, output unit receives connections from only newly added hidden layer and diminish its previous connections. Connecting previously added hidden layers to output units plays no significant role in the network. It only adds burden on the network and reduces the generalization performance as well as learning speed. The amount of hidden units addition in each layer is not restricted to one at a time, but it depends upon the task to converge faster to target error. Hidden units are generated from orthogonal liner transformation of input units, therefore, the hidden

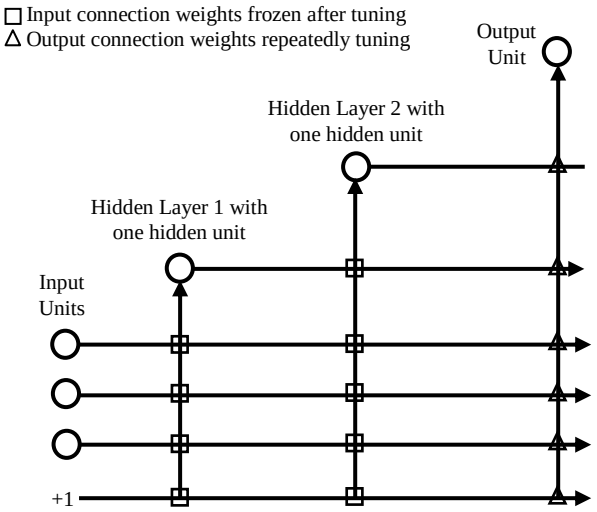

Figure 1. CasCor architecture

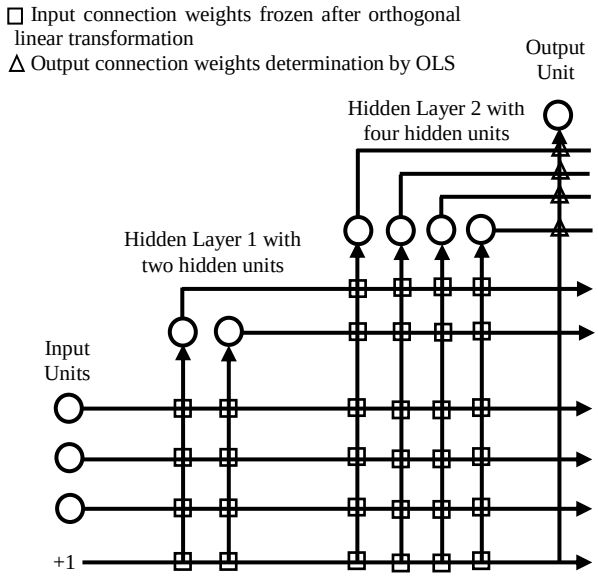

Figure 2. CPCLS architecture

units feature generation will be always less than or equal to input units features such that $p \leq n$.

The proposed CPCLS algorithm has serval advantages over traditional quick prop Cascor that it learns faster, no need of iterations to optimized parameters, better generalization performance, determine its own network typology and parameters, work directly with both differentiable and non- differentiable activation function, no need of ad hoc learning rate methods for faster convergences, no candidate unit's generation to avoid overfitting and free from complex learning algorithms calculations.

Algorithm CPCLS

Suppose a training set $(\boldsymbol{X}, \boldsymbol{Y})$ with input units matrix $\boldsymbol{X}$ mxn, output units matrix $\boldsymbol{Y}$ mxq, hidden units matrix $\boldsymbol{H}$ mxp, and target error e:

Step 1) **Initialization:** Let initial number of $\boldsymbol{H}$=N such that $p \leq n$ and target error = e

Step 2) **Learning Step:**

While $E > e$

a) Calculate *input connection weights W* matrix *nxp*:

1. Determine covariance $\boldsymbol{S}$ matrix *dxd* among input *n* features:

$$\boldsymbol{S} = \frac{1}{m-1}\boldsymbol{X}^T\boldsymbol{X}$$

$$\boldsymbol{X} = x_i - \bar{x}$$

$$\bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_i$$

2. Generate eigenvalue λ and corresponding eigenvector from $\boldsymbol{S}$:

$$|\boldsymbol{S} - \lambda \boldsymbol{I}| = 0$$

$$\boldsymbol{SW} = \lambda \boldsymbol{W}$$

Where selected eigenvector $\boldsymbol{W}$ are the input connection weights for $\boldsymbol{H}$

b) Calculate $\boldsymbol{H}$ by taking activation of inner product of $\boldsymbol{X}$ and $\boldsymbol{W}$:

$$\boldsymbol{H} = \boldsymbol{X}W$$

c) Calculate the *output connection weights β pxq* by taking inner product of Moore Penrose pseudo-inverse of $\boldsymbol{H}$ and $\boldsymbol{Y}$:

$$\beta = (\boldsymbol{H^T H})^{-1}\boldsymbol{H^T Y}$$

d) Calculate the output matrix $\hat{\boldsymbol{Y}}$ by linearly transferring $\boldsymbol{H}$ through β:

$$\hat{\boldsymbol{Y}} = \boldsymbol{H}\beta$$

e) Estimate error E by subtracting the actual output $\boldsymbol{Y}$ from predicted output $\hat{\boldsymbol{Y}}$:

$$E = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{\boldsymbol{Y}} - \boldsymbol{Y}\right)^2$$

f) Stack the calculated $\boldsymbol{H}$ with $\boldsymbol{X}$:

$$\boldsymbol{X} = (\boldsymbol{X}, \boldsymbol{H})$$

g) Increase the number of $\boldsymbol{H}$ by n' such that $p \leq n$:

$$N = N + n'$$

endwhile

IV. PERFORMANCE EVALUATION

The generalization performance and learning speed of proposed algorithm CPCLS is compared with quick prop CasCor on popular highly nonlinear benchmarking problems of two spiral classification and function approximation of SinC regression task. The simulation work is performed in Anaconda Spyder Python v3.2.6 and Netmaker v0.9.5.2. Netmaker is a simulation software designed in C programming for building neural networks and has build-in quick prop CasCor programming code. CPCLS simulation is carried out in python, whereas, CasCor simulation is carried out in Netmaker. Simulation in different programming code will not affect the comparison because C programming execution is faster than Python. The activation function used is sigmoid for CPCLS and CasCor.

A. Classification Benchmark: Two Spiral Problem

The generalization performance and learning speed of CPCLs and CasCor are evaluated on well-known extreme nonlinear benchmarking task of two spiral classification. The task consists of 194 training points twisted in two spirals with three turns for each spiral. It consists of two inputs units and one output unit with two classes as 0's and 1's. The algorithms must classify all data points as 0's and 1's as shown in Fig. 3 in black and white dots respectively. The generalization performance of algorithms is evaluated on newly generated dense spiral of 770 testing data points other than training points [13]. All the data points of input and output are normalized in the range [0,1].

Table I shows the generalization performance and learning ability simulation results of both algorithms average over 25 trials. For better simulation results, number of candidate units in CasCor are set to 8 Nos. As shown in Table I, CPCLS spend 5.07 s to train a network with generalization performance of 95.61%, whereas, CasCor spend 135.84 s to train a network with generalization performance of 91.00%. The results indicate that CPCLS has obtain better generalization performance and faster learning speed than CasCor.

B. Regression Benchmark: SinC Function Approximation

In regression, nonlinear SinC function task is approximated by CPCLS and CasCor to check its generalization performance and learning ability. To make task more complex and nonlinear, data points of 4000 observations are generated from below SinC function in the range (-20,20):

$$y(x) = \begin{cases} \dfrac{\sin(x)}{x}, & x \neq 0 \\ 1, & x = 0 \end{cases} \tag{11}$$

Table II shows the generalization performance and learning ability simulation results of CPCLS and CasCor average over 25 trials by randomly splitting data points in to training and testing dataset at a 50:50 ratio during each trial. All the data points of input and output are normalized in the range [0,1]. As observed from Table II, CPCLS spend 10.07 s to train a network with generalization performance of 0.00080 RMSE, whereas, CasCor spend 318.82 s to train a network with generalization performance of 0.02425 RMSE. CPCLS can truly approximate all data points of SinC as illustrate in Fig. 4, whereas, CasCor loss its generalization ability as illustrated in Fig 5.

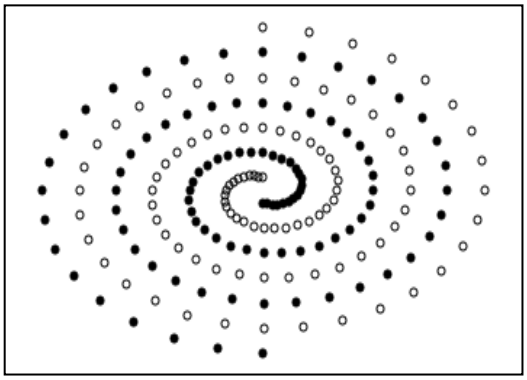

Figure 3. Two Spiral Classification Task

TABLE I. GENERALIZATION PERFORMANCE AND LEARNING SPEED OF TWO SPIRAL CLASSIFICATION TASK

Algorithm	Testing Accuracy (%)		Training Time (Sec)		Hidden Units (nos.)	
	Mean	*Stdev*	*Mean*	*Stdev*	*Mean*	*Stdev*
CPCLS	95.61	0.47	5.07	0.51	142	4.47
CasCor	91.00	2.25	135.84	39.72	20.5	2.06

V. DISCUSSIONS AND CONCLUSIONS

In this paper, simulation work demonstrates that new proposed algorithm known as Cascade Principal Component Least Squares Neural Network (CPCLS) has achieved better generalization performance and learning speed with improved cascade architecture and tuning free learning method as compared to traditional Cascading Correlation learning (CasCor) algorithm. CPCLS algorithm analytically calculate weight connections on improved cascade architecture to convert nonlinear complex tasks to linear for quick and efficient reduction of sum of squares error (SSE). CPCLS has several noteworthy features as compared to CasCor:

- The generalization performance of proposed algorithm is better than CasCor. Simulation results shows that improvement in cascade architecture create less burden on network and converges more quickly towards target error. The proposed algorithm works more analytically by determining optimal weight connections rather than CasCor which need numerical optimization techniques, ad hoc adjustments such as weight decay, and candidate unit's generation to improve generalization performance and prevent issues like underfitting and overfitting.
- The learning speed is fast due to no adoption of numerical optimization technique such as quick prop. Unlike CasCor that is involved in both forward and backward delta rule tuning of weight connections, CPCLS take only forward steps to calculate target error by analytically calculating weight connections. Furthermore, CPCLS can work with all type of non-differentiable activation function (such as threshold) directly to move more faster towards results.

The superior generalization performance and learning ability of CPCLS on nonlinear benchmark problems validate that that it can be efficiently applied in many real application tasks.

TABLE II. GENERALIZATION PERFORMANCE AND LEARNING SPEED OF SINC REGRESSION TASK

Algorithm	Testing RMSE		Training Time (Sec)		Hidden Units (nos.)	
	Mean	*Stdev*	*Mean*	*Stdev*	*Mean*	*Stdev*
CPCLS	0.00080	0.00013	10.07	2.64	105.5	7.66
CasCor	0.02425	0.00086	318.82	67.09	17.3	1.89

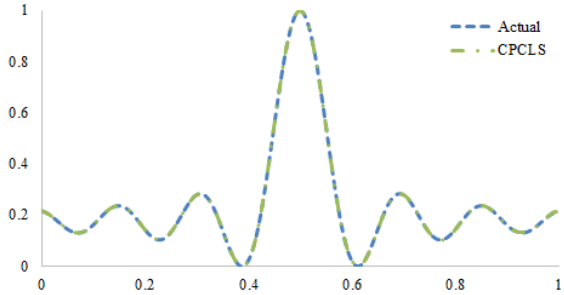

Figure 4. CPCLS generlization of SinC Function

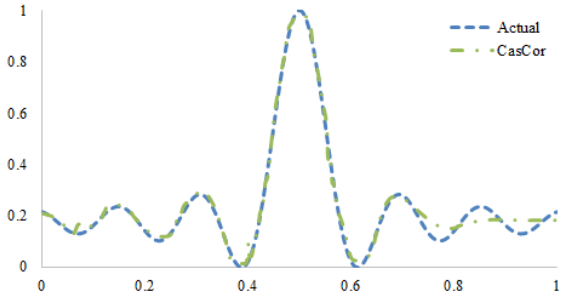

Figure 5. CasCor generlization of SinC Function

REFERENCES

[1] R. Hecht-Nielsen, "Theory of the Backpropagation Neural Network."

[2] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929-1958, 2014.

[3] R. Reed, "Pruning algorithms-a survey," *IEEE transactions on Neural Networks*, vol. 4, no. 5, pp. 740-747, 1993.

[4] A. Krogh and J. Vedelsby, "Neural network ensembles, cross validation, and active learning," in *Advances in neural information processing systems*, 1995, pp. 231-238.

[5] C. Charalambous, "Conjugate gradient algorithm for efficient training of artificial neural networks," *IEE Proceedings G (Circuits, Devices and Systems)*, vol. 139, no. 3, pp. 301-310, 1992.

[6] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE transactions on Neural Networks*, vol. 5, no. 6, pp. 989-993, 1994.

[7] S. E. Fahlman, "An empirical study of learning speed in back-propagation networks," 1988.

[8] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," in *Advances in neural information processing systems*, 1990, pp. 524-532.

[9] G. Huang, S. Song, and C. Wu, "Orthogonal least squares algorithm for training cascade neural networks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 11, pp. 2629-2637, 2012.

[10] M. Lehtokangas, "Modified cascade-correlation learning for classification," *IEEE Transactions on Neural Networks*, vol. 11, no. 3, pp. 795-798, 2000.

[11] V. V. Kovalishyn, I. V. Tetko, A. I. Luik, V. V. Kholodovych, A. E. Villa, and D. J. Livingstone, "Neural network studies. 3. variable selection in the cascade-correlation learning architecture," *Journal of Chemical Information and Computer Sciences*, vol. 38, no. 4, pp. 651-659, 1998.

[12] T.-Y. Kwok and D.-Y. Yeung, "Constructive algorithms for structure learning in feedforward neural networks for regression problems," *IEEE transactions on neural networks*, vol. 8, no. 3, pp. 630-645, 1997.

[13] K. J Lang and M. Witbrock, *Learning to Tell Two Spirals Apart*. 1988.

[14] PCA

[15] OLS