RESEARCH ARTICLE OPEN ACCESS

Optimization of Number of Neurons in the Hidden Layer in Feed Forward Neural Networks with an Emphasis to Cascade Correlation Networks

Dr. R. Amal Raj
Associate Professor in Computer Science, Sri Vasavi College, Erode- 638 316. Tamil Nadu, South India.

ABSTRACT
The architectures of Artificial Neural Networks (ANN) are based on the problem domain and it is applied during the ‘training phase’ of sample data and used to infer results for the remaining data in the testing phase. Normally, the architecture consist of three layers as input, hidden, output layers with the number of nodes in the input layer as number of known values on hand and the number of nodes as result to be computed out of the values of input nodes and hidden nodes as the output layer. The number of nodes in the hidden layer is heuristically decided so that the optimum value is obtained with reasonable number of iterations with other parameters with its default values. This study mainly focuses on Cascade-Correlation Neural Networks (CCNN) using Back-Propagation (BP) algorithm which finds the number of neurons during the training phase itself by appending one from the previous iteration satisfying the error condition gives a promising result on the optimum number of neurons in the hidden layer.

Keywords: Cascade-Correlation Neural Network, Back-Propagation Algorithm, Double Dummy Bridge Problem.

I. INTRODUCTION

ANN are classified under a broad spectrum of Artificial Intelligence (AI) that attempts to imitate the way a human brain works and the Feed-Forward Neural Networks (FFNN) are one of the most common types of neural networks in use and these are often trained by the way of supervised learning supported by Cascade-Correlation Neural Network architecture using Back Propagation algorithm. Many FFNN were trained to solve the Double Dummy Bridge Problems (DDBP) in bridge game (Mandziuk, & Mossakowski, 2009a), and (Mandziuk, & Mossakowski 2004), and (Sarkar, Yegnanarayana and Khemani, 1995), and (Dharmalingam, & Amalraj, 2013a) and they have been formalized in the best defense model, which presents the strongest possible assumptions about the opponent. This is used by human players because modeling the strongest possible opponents provides a lower bound on the pay off that can be expected when the opponents are less informed.

The Bridge Baron is generally acknowledged to be the best available commercial program for the game of Contract Bridge developed by using Domain Dependent Pattern-matching techniques which has some limitations. Hence there was a need to develop more sophisticated AI techniques to improve the performance of the Bridge Baron which was supplemented by its previously existing routines for declarer, to play with routine, based on Hierarchical Task-Network (HTN) planning techniques. The HTN planning techniques used to develop game trees in which the number of branches at each node corresponds to the different strategies that a player might pursue rather than the different cards the player might be able to play (Smith, Nau, & Throop, 1998a). A Point Count method and Distributional Point methods are the two types of hand strength in human estimators. The Work Point Count System (WPCS) is an exclusive, most important and popular system which is used to bid a final contract in Bridge game. Many of the neural network architectures are used to solve the double dummy bridge problem in contract bridge. Among the various networks, cascade-correlation neural networks (CCNN) is focused in this paper with Back-Propagation (BP) algorithm used to train and test the data and the results are compared.

II. ARTIFICIAL NEURAL NETWORKS

ANN consists of several dispensation units which are interconnected according to some topology to accomplish a pattern classification task or data classification through learning process. The cascade-correlation architecture was introduced by (Fahlman & Lebiere, 1990) starts with a one layer neural network and hidden neurons are added depends on the need. The Cascade-Correlation begins with a minimal network, then mechanically trains and adds new hidden units one by one, creating a multi-layer configuration. Once a new hidden unit has been added to the network, its input-side weights are frozen. The new hidden neuron is added in each training set and weights are adjusted to minimize the magnitude of the correlation between the new hidden neuron output and the residual error signal on the network output that has to be eliminated. The cascade-correlation architecture has many rewards over its counterpart, as it learns at a faster rate, the network determines its own dimension and topology, it retains the structures it had built, still if the preparation set changes, and it requires no back-propagation of error signals through the associations of the network. During the learning progression, new neurons are added to the network one by one Fig.3 and each one of them is placed into a new hidden layer and connected to all the preceding input and hidden neurons. Once a neuron is finally further to the network and activated, its input connections become frozen and do not change anymore.

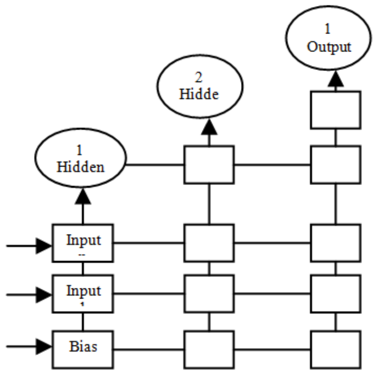

Fig. 1 Cascade-Correlation Neural Network (CCNN)

The neuron to be added to the existing network can be made in the following two steps: (i) The candidate neuron is connected to all the input and hidden neurons by trainable input connections, but its output is not connected to the network. Then the weights of the candidate neuron can be trained while all the other weights in the network are frozen. (ii) The candidate is connected to the output neurons and then all the output connections are trained. The whole process is repeated until the desired network accuracy is obtained. The equation (1) correlation parameter 'S' defined as below is to be maximized.

$$S = \sum_{0=1}^{O} \left| \sum_{p=1}^{P} (V_p - \bar{V})(E_{po} - \overline{E_o}) \right| \qquad (1)$$

where *O* is the number of network outputs, *P* is the number of training patterns, V_p is output on the new hidden neuron and E_{po} is the error on the network output. In the equation (2) the weight adjustment for the new neuron can be found by gradient descent rule as

$$\Delta_{w_i} = \sum_{0=1}^{O} \sum_{p=1}^{P} \sigma_o \left(E_{po} - \overline{E_o}\right) f_p' \, x_{ip} \qquad (2)$$

The output neurons are trained using the generalized delta learning rule for faster convergence in Back -Propagation algorithm. Each hidden neuron is trained just once and then its weights are frozen. The network learning building process is completed when satisfied results are obtained. The cascade-correlation architecture needs only a forward sweep to compute the network output and then this information can be used to train the candidate neurons.

III. CASCADE-CORRELATION NEURAL NETWORK WITH BACK-PROPAGATION TRAINING ALGORITHM

The cascade correlation neural network is a widely used type of network architecture, Instead of just adjusting the weights in a network of fixed topology, with supervised learning. This network consists of an input layer, a hidden layer, an output layer and two levels of adaptive connections. It is also fully interconnected, i.e. each neuron is connected to all the neurons in the next level. The overall idea behind back propagation is to make large change to a particular weight, '*w'* the change leads to a large reduction in the errors observed at the output nodes. Let *'y'* be a smooth function of several variables x_i, we want to know how to make incremental changes to initial values of each x_i, so as to increase the value of y as fast as possible. The change to each initial x_i value should be in proportion to the partial derivative of y with respect to that particular x_i. Suppose that *'y'* is a function of a several intermediate variables x_i and that each x_i is a function of one variable *'z'*. Also we want to know the derivative of '*y*' with respect to *'z*', using the chain rule.

$$\Delta x_i \infty \frac{\partial y}{\partial x_i} \qquad (3)$$

$$\frac{dy}{dz} = \sum_i \frac{\partial y}{\partial x_i}\frac{dx_i}{dz} = \sum_i \frac{dx_i}{dz}\frac{\partial y}{\partial x_i} \qquad (4)$$

The standard way of measuring performance is to pick a particular sample input and then sum up the squared error at each of the outputs. We sum over all sample inputs and add a minus sign for an overall measurement of performance that peaks at o.

$$P = -\sum_s \left(\sum_z (d_{sz} - o_{sz})^2 \right) \qquad (5)$$

Where *'P'* is the measured performance, *'S'* is an index that ranges over all sample inputs, *'Z'* is an index that ranges overall output nodes, d_{sz} is the desired output for sample input 's' at the z^{th} node, o_{sz} is the actual output for sample input *'s'* at the z^{th} node. The performance measure *'P'* is a function of the weights. We can deploy the idea of gradient ascent if we can calculate the partial derivative of performance with respect to each digit. With these partial derivatives in hand, we can climb the performance hill most rapidly by altering all weights in proportion to the corresponding partial derivative. The performance is given as a sum over all sample inputs. We can compute the partial derivative of performance with respect to a particular weight by adding up the partial derivative of performance for each sample input considered separately. Each weight will be adjusted by summing the adjustments derived from each sample input.

Consider the partial derivative

$$\frac{\partial P}{\partial w_{i\to j}}$$

where the weight *w (i→j)* is a weight connecting i^{th} layer of nodes to j^{th} layer of nodes. Our goal is to find an efficient way to compute the partial derivative of P with respect to *w(i→j)*. The effect of *w(i→j)* on value P, is through the intermediate variable o_j, the output of the j^{th} node and using the chain rule, it is express as

$$\frac{\partial P}{\partial w_{i\to j}} = \frac{\partial P}{\partial o_j}\frac{\partial o_j}{\partial w_{i\to j}} = \frac{\partial o_j}{\partial w_{i\to j}}\frac{\partial P}{\partial o_j} \qquad (7)$$

Determine o_j by adding up all the inputs to node *'j'* and passing the results through a function.

Hence,

$$o_j = f\left(\sum_i 0_i w_{i\to j}\right) \qquad (8)$$

where is a threshold function. Let

$$\sigma_j = \sum_i 0_i w_{i\to j}$$

We can apply the chain rule again.

$$\frac{\partial o_j}{\partial w_{i\to j}} = \frac{df(\sigma_j)}{d\sigma_j}\frac{\partial \sigma_j}{\partial w_{i\to j}} \qquad (9)$$

$$\frac{\partial P}{\partial o_j} \quad \frac{df(\sigma_j)}{d\sigma_j}\, o_i \quad \frac{\partial P}{\partial o_k} \qquad (10)$$

$$\frac{\partial P}{\partial w_{i\to j}} = o_i \frac{df(\sigma_j)}{d\sigma_j}\frac{\partial P}{\partial o_j} \qquad (11)$$

Substituting Equation (8) in Equation (5), we have

$$\frac{\partial P}{\partial o_j} = \partial w_{i\to j}\frac{df(\sigma_k)}{d\sigma_k}\frac{\partial P}{\partial o_k} \qquad (12)$$

$$\frac{\partial P}{\partial w_{i\to j}} = o_i \frac{df(\sigma_j)}{d\sigma_j} w_{j\to k} \frac{df(\sigma_k)}{d\sigma_k}\frac{\partial P}{\partial o_k} \qquad (13)$$

Thus, the two important consequences of the above equations are, 1) The partial derivative of performance with respect to a weight depends on the partial derivative of performance with respect to the following output. 2) The partial derivative of performance with respect to one output depends on the partial derivative of performance with respect to the outputs in the next layer. The system error will be reduced if the error for each training pattern is reduced. Thus, at step 's+1' of the training process, the weight adjustment should be proportional to the derivative of the error measure computed on iteration 's'. This can be written as

$$\Delta w(s+1) = -n\partial P / \partial w(s) \qquad (14)$$

where η is a constant learning coefficient, and there is another possible way to improve the rate of convergence by adding some inertia or momentum to the gradient expression, accomplished by adding a fraction of the previous weight change with current weight change. The addition of such term helps to smooth out the descent path by preventing extreme changes in the gradient due to local anomalies. Hence, the partial derivatives of the errors must be accumulated for all training patterns. This indicates that the weights are updated only after the presentation of all of the training patterns.

IV. PROBLEM DESCRIPTION

In bridge games, basic representation includes value of each card (Ace (A), King (K), Queen (Q), Jack (J), 10, 9, 8, 7, 6, 5, 4, 3, 2) and suit as well as the assignment of cards into particular hands and into public or hidden subsets, depending on the game rules. In the learning course, besides acquiring these basic information, several other more sophisticated features need to be developed by the learning system (Francis, Truscott and Francis (1994) and Root ,(1998)).

4.1. The game of contract bridge

Contract bridge, simply known as bridge, is a trick-taking card game, where there are four players in two fixed partnerships as pairs facing each other (Mandziuk & Mossakowski 2007) and referred according to their position at the table as North (N), East (E), South (S) and West (W), so N and S are partners playing against E and W. A standard fifty two pack is used and the cards in each suit rank from the highest to the lowest as *Ace (A), King (K), Queen (Q), Jack (J),* 10, 9, 8, 7, 6, 5, 4, 3, 2. The dealer deals out all the cards one at a time so that each player receives 13 of them. The team who made the final bid will at the moment try to make the contract. The first player of this group who mentioned the value of the contract becomes the declarer. The player to the left of the declarer leads to the first trick and instantly after this opening lead, the dummy's cards are shown. The play proceeds clockwise and each player must, if potential, play a card of the suit led. A trick consists of four cards and is won by the maximum trump in it or if no trumps were played by the maximum card of the suit led. The champion of a trick leads to the next stage and the aim of the declarer is to take at least the number of tricks announced during the bidding phase when the opponents try to prevent from doing it. In bridge, special focus in game representation is on the fact that players cooperate in pairs, thus sharing potentials of their hands (Mandziuk & Mossakowski, 2009b).

To estimate the number of tricks to be taken by one pair of bridge players in DDBP, an attempt is made to solve the problem in which the solver is presented with all four hands and is asked to determine the course of play that will achieve or defeat at a particular contract. The partners of the declarer, whose cards are placed face up on the table and may be played by declarer. The dummy has few rights and may not participate in choices concerning the play of the hand and estimating hands strength is a decisive aspect of the bidding phase of the game of bridge, since the contract bridge is a game with incomplete information. This incompleteness of information might allow for many variants of a deal in cards distribution and the player should take into account all these variants and speedily approximate the predictable number of tricks to be taken in each case (Dharmalingam & Amalraj 2013b).

The fifty two input card representation deals were implemented in this architecture. The card values were determined in rank card (2, 3, K, A) and suit card (♠ (S), ♥ (H), ♦ (D), ♣(C)). The rank card was transformed using a uniform linear transformation to the range from 0.10 to 0.90. The Smallest card value is 2(0.10) and highest card value is A (0.90). The suit cards were a real number using the following mapping: Spades (0.3), Hearts (0.5), Diamonds (0.7) and Clubs (0.9).There were 52 input values and each value represented one card from the deck. Positions of cards in the input layer were fixed, i.e. from the leftmost input neuron to the rightmost one the following cards were represented: 2♠, 3♠, K♠, A♠, 2♥, A♥, 2♦, A♦, 2♣,. .. , A♣. A value presented to this neuron determined the hand to which the respective card belonged, i.e. 1.0 for North, 0.8 for South, −1.0 for West, and −0.8 for East. The game then proceeds through a bidding and playing phase. The purpose of the biding phase is to identification of trumps and declarer of the contract. The playing phase consists of 13 tricks, with each player contributing one card to each trick in a clockwise fashion with another level bid to decide who will be the declarer. A bid recognizes a number of tricks and a trump suit or no-trump. The side which bids highest will try to win at least that number of tricks bid, with the specified suit as trumps. There are 5 possible trump suits: spades (♠), hearts (♥), diamonds (◆), clubs (♣) and "no-trump" which is the term for contracts played without a trump. After three successive passes, the last bid becomes the contract.

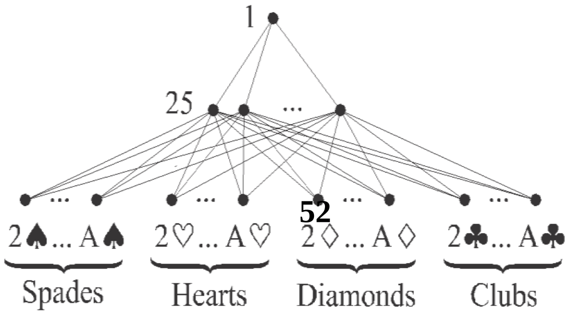

Fig. 2 Neural network architecture with 52 input neurons

Layers were fully connected, i.e., in the 52 − 32 − 1 network all 52 input neurons where connected to all 32 hidden ones, and all hidden neurons were connected to a single output neuron.

4.2. The bidding and playing phases

The bidding phase is a conversation between two cooperating team members against an opposing partnership which aims to decide who will be the declarer. Each partnership uses an established bidding system to exchange information and interpret the partner's bidding sequence as each player has knowledge of his own hand and any previous bids only. A very interesting aspect of the bidding phase is the cooperation of players in North with South and West with East. In each, player is modeled as an autonomous, active agent that takes part in the message process (Yegnanarayana, Khemani,and Sarkar,1996) and (Amit & Markovitch 2006).

The play phase seems to be much less interesting than the bidding phase. The player to the left of the declarer leads to the first trick and may play any card and instantly after this opening lead, the dummy's cards are exposed. The play proceeds clockwise and each of the other three players in turn must, if found potential, play a card of the same suit that the person in-charge played. A player with no card of the suit may play any card of his selection. A trick consists of four cards, one from each player, declared won by the maximum trump in it, or if no trumps were played by the maximum card of the suit. The winner of a trick leads subsequently with any card as the dummy takes no active role in the play and not permitted to offer any advice or observation. Finally, the scoring depends on the number of tricks taken by the declarer team and the contract (Smith, Nau and Throop (1998b) and (Frank & Basin, 2001) and (Ando, Sekiya, and Uehara (2000)).

4.3 No-trump & Trump-suit

A trick contains four cards one contributed by each player and the first player starts by most important card, placing it face up on the table. In a clockwise direction, each player has to track suit, by playing a card of the similar suit as the one led. If a heart is lead, for instance, each player must play a heart if potential. Only if a player doesn't have a heart he can discard. The maximum card in the suit led wins the trick for the player who played it. This is called playing in no-trump. No-trump is the maximum ranking denomination in the bidding, in which the play earnings with no-trump suit. No-trump contracts seem to be potentially simpler than suit ones, because it is not possible to ruff a card of a high rank with a trump card. Though it simplifies the rules, it doesn't simplify the strategy as there is no guarantee that a card will take a trick, even Aces are useless in tricks of other suits in no-trump contracts. The success of a contract often lies in the hand making the opening lead. Hence even knowing the location of all cards may sometimes be not sufficient to indicate cards that will take tricks (Jamroga, 1999). A card that belongs to the suit has been chosen to have the highest value in a particular game, since a trump can be any of the cards belonging to any one of the players in the pair. The rule of the game still necessitates that if a player can track suit, the player must do so, otherwise a player can no longer go behind suit, however, a trump can be played, and the trump is higher and more influential than any card in the suit led (Mandziuk, 2008).

4.4 Work Point Count System

The Work Point Count System (WPCS) which scores 4 points for Ace, 3 points for King, 2 points for Queen and 1point for a Jack is followed in which no points are counted for 10 and below. During the bidding phase of contract bridge, when a team reaches the combined score of 26 points, they should use WPCS for getting final contract and out of thirteen tricks in contract bridge, there is a possibility to make use of eight tricks by using WPCS.

V. THE DATA REPRESENTATION OF GIB LIBRARY

The data used in this game of DDBP was taken from the Ginsberg's Intelligent Bridge (GIB) Library, which includes 7,00,000 deals and for each of the tricks, it provides the number of tricks to be taken by N S pair for each combination of the trump suit and the hand which makes the opening lead. There are 20 numbers of each deal i.e. 5 trump suits by 4 sides as No-trumps, spades, Hearts, Diamonds and Clubs. The term 'No-trump' is used for contracts played without trump in the four sides West, North, East and South.

VI. IMPLEMENTATION

In this layer only one output was received and getting the result, decision boundaries are normalized 0 to 1 in the range. The results are defined a priori and target ranges from 0 to 13 for all possible number of tricks. The results produced are represented in Table 1 and Table 2 respectively.

Table1 Training deals sample 20

S.No	Actual value	Calcuted value
1	0.75000	0.64566
2	0.83000	0.78789
3	1.00000	0.94189
4	0.83000	0.88712
5	0.75000	0.68881
6	0.50000	0.50331
7	0.58000	0.68419
8	0.75000	0.72846
9	0.50000	0.62448
10	0.83000	0.79120

11	0.58000	0.74925
12	1.00000	0.86405
13	0.58000	0.59701
14	0.50000	0.51491
15	0.91000	0.51458
16	0.50000	0.79397
17	0.50000	0.50773
18	0.83000	0.69075
19	0.66000	0.71014
20	0.58000	0.82686

Table 2 Test deals sample 10(Even)

S.No	Actual value	Calcuted value
1	0.83000	0.82689
2	0.83000	0.77306
3	0.50000	0.64252
4	0.75000	0.82796
5	0.83000	0.82953
6	1.00000	0.97059
7	0.50000	0.54102
8	0.50000	0.79967
9	0.83000	0.82698
10	0.58000	0.57828

VII. REPRESENTATION OF RESULTS

A total of twenty deals from GIB library for training and testing are considered and all the 20 deals are trained on CCNN with fifty two input neurons, twenty five, twenty six, thirty two hidden neurons and only one output neuron (52-25/26/32-1). The Back Propagation Algorithm was used for training and testing through MATLAB 2008a. The results are represented in Fig.3 and Fig.4.

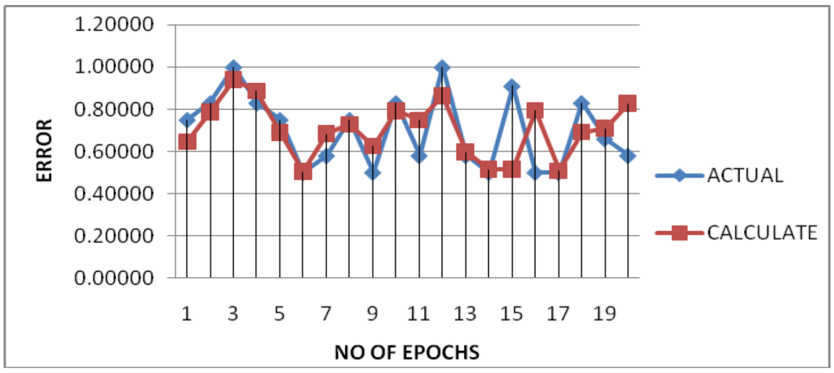

Fig.3. Training deals sampling 1000 epochs

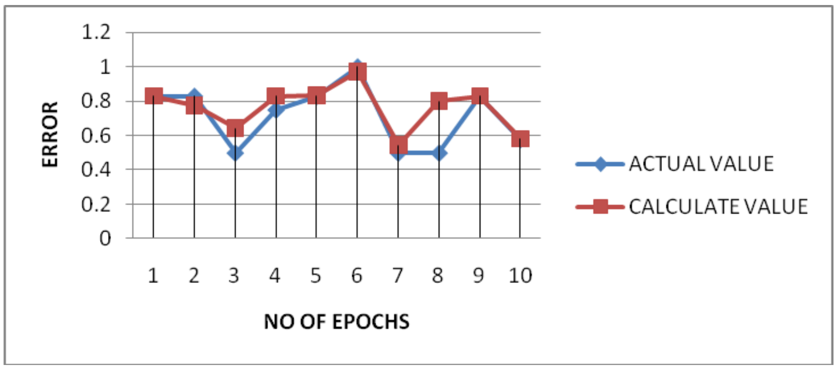

Fig.4. Testing deals sampling 1000 epochs

The results revealed that, the data trained and tested through above architecture show significantly better performance and the time taken for training and testing the data were relatively minimum which also converged to the error steadily during the whole process.

As a matter of rule of thumb, when fifty two input nodes are added in the input layer, all requiring a single output node in the output layer as the result, exactly half the number of input nodes minus one are assumed to be in the hidden layer and values are trained and the weight matrix is computed. With the weight values obtained so far in the above training session, the testing phase is executed and the results are compared, which are not at all to the level of satisfaction and hence discarded. The results obtained for the training phases are illustrated as below. Thus fifty input neurons with twenty five hidden neurons and one output neuron are trained and the results are compared. The following Table illustrates the performance of the architectures with 25 hidden neuron as

Table .3 Actual value Vs Calculated value for 25 hidden neurons

Sl. No	Actual Value	Calculated Value
1	0.75000	0.51231
2	0.83000	0.94955
3	1.00000	0.95303
4	0.83000	0.72074
5	0.75000	0.57246
6	0.50000	0.51066
7	0.58000	0.61328
8	0.75000	0.92103
9	0.50000	0.93981
10	0.83000	0.59870
11	0.58000	0.89859
12	1.00000	0.87241
13	0.58000	0.84650
14	0.50000	0.83622
15	0.91000	0.93451
16	0.50000	0.85116
17	0.50000	0.51979
18	0.83000	0.84266
19	0.66000	0.69493
20	0.58000	0.63764

The Chart depicting the accuracy of actual value and the calculated value of the output obtained out of training with 25 hidden neurons is in pictorially represented as

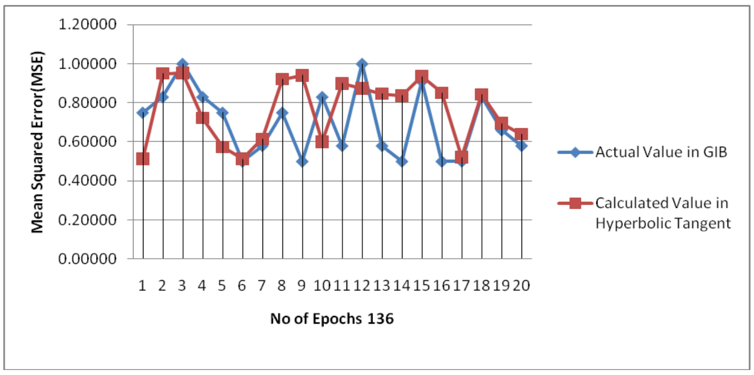

Fig. 5 Mean Square error between actual value and calculated value with 25 hidden neuron

The results obtained for the training phases with 26 hidden neurons are illustrated as below. Thus fifty input neurons with twenty six hidden neurons and one output neuron are trained and the results are compared. The following Table illustrates the performance of the architectures with 26 hidden neuron as

Table .4 Actual value Vs Calculated value for 26 hidden neurons

Sl. No	Actual Value	Calculated Value
1	0.75000	0.57056
2	0.83000	0.74493
3	1.00000	0.76269
4	0.83000	0.75261
5	0.75000	0.83763
6	0.50000	0.57597
7	0.58000	0.70295
8	0.75000	0.83670
9	0.50000	0.65969
10	0.83000	0.68574
11	0.58000	0.69158
12	1.00000	0.56964
13	0.58000	0.73977
14	0.50000	0.70528
15	0.91000	0.60019
16	0.50000	0.66657
17	0.50000	0.66947
18	0.83000	0.77297
19	0.66000	0.74093
20	0.58000	0.61806

The Chart depicting the accuracy of actual value and the calculated value of the output obtained out of training with 26 hidden neurons is in pictorially represented as

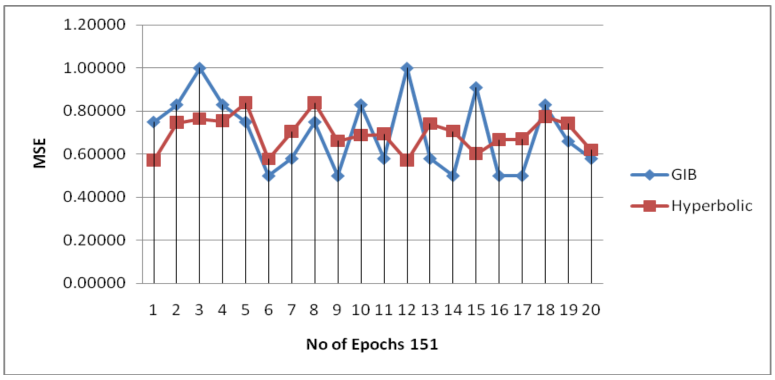

Fig. 6 Mean Square error between actual value and calculated value with 26 hidden neuron

The results obtained for the training phases with 32 hidden neurons are illustrated as below. Thus fifty input neurons with thirty two hidden neurons and one output neuron are trained and the results are compared. The following Table illustrates the performance of the architectures with 32 hidden neuron as

Table .5 Actual value Vs Calculated value for 32 hidden neurons

Sl. No	Actual Value	Calculated Value
1	0.75000	0.79554
2	0.83000	0.93323
3	1.00000	0.91831
4	0.83000	0.80402
5	0.75000	0.73769
6	0.50000	0.55078
7	0.58000	0.59921
8	0.75000	0.69213
9	0.50000	0.50566
10	0.83000	0.61028

11	0.58000	0.52415
12	1.00000	0.70065
13	0.58000	0.58546
14	0.50000	0.53056
15	0.91000	0.90011
16	0.50000	0.79344
17	0.50000	0.57977
18	0.83000	0.76868
19	0.66000	0.67927
20	0.58000	0.92717

The Chart depicting the accuracy of actual value and the calculated value of the output obtained out of training with 32 hidden neurons is in pictorially represented as

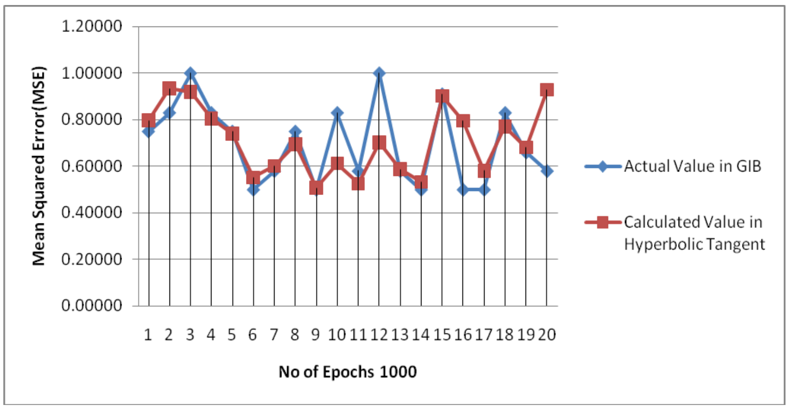

Fig. 7 Mean Square error between actual value and calculated value with 32 hidden neuron

As a rule of thumb, the number of hidden neurons equal to the number of input neurons say fifty, instead of fifty two was tested for the behaviour similar to the twenty five neurons in earlier case and also twenty six neurons with its calculated value and actual value as in the above case was also done and the results are given in the following Table as

Table .6 Actual value Vs Calculated value for 50 hidden neurons

Sl. No	Actual Value	Calculated Value
1	0.75000	0.88385
2	0.83000	0.93545
3	1.00000	0.94133
4	0.83000	0.81202
5	0.75000	0.74827
6	0.50000	0.50538
7	0.58000	0.58669
8	0.75000	0.74794
9	0.50000	0.50766
10	0.83000	0.68974
11	0.58000	0.56994
12	1.00000	0.81799
13	0.58000	0.67714
14	0.50000	0.55779
15	0.91000	0.87769
16	0.50000	0.57663
17	0.50000	0.52860
18	0.83000	0.80026
19	0.66000	0.68407
20	0.58000	0.54160

The Chart depicting the accuracy of actual value and the calculated value of the output obtained out of training with 50 hidden neurons is in pictorially represented as

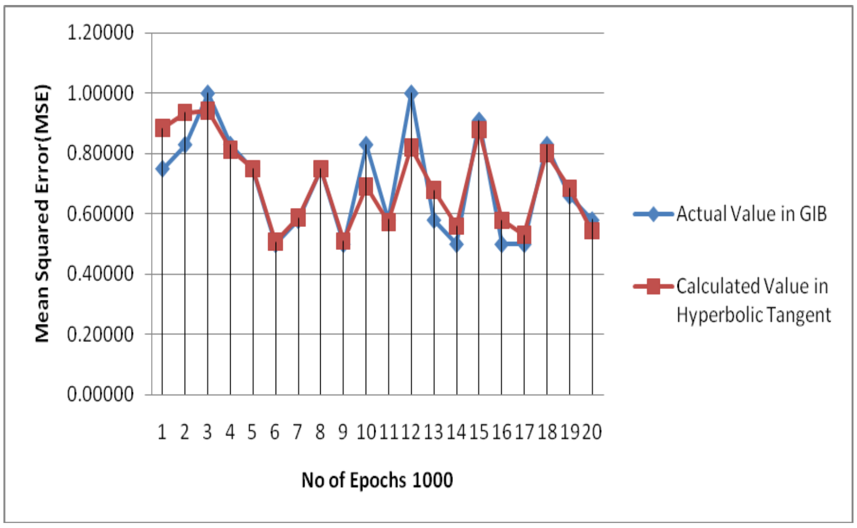

Fig. 8 Mean Square error between actual value and calculated value with 50 hidden neuron

VIII. CONCLUSION

In Cascade-Correlation neural network, during training process new hidden nodes are added to the network one by one. For each new hidden node, the correlation magnitude between the new node output and the residual error signal is maximized. During the time when the node is being added to the network, the input weights of hidden nodes are frozen, and only the output connections are trained repeatedly. Thus, so far any formula to find the exact number of hidden neurons for any architecture is not yet available and it is not feasible to find out one such formula also. In such cases in order to make the architecture to learn continuously data after data for a better prediction of untrained data may yield in memorizing the data instead of learning the pattern in the data. Also when the number of neurons becomes very less, then after successful training, it may not be able to predict the correct results because of poor learning. Hence, to find the optimum number of hidden neurons in an architecture which balances both extremes and simultaneously able to predict the best possible output with respect to expected and calculated value of the output neuron, cascade correlation neural network behaves sufficiently better, since the architecture itself demands appending of hidden nodes one after another from the initial condition of hidden layer with no neuron at all. The Back Propagation Algorithm which produced better results and used to bid a final contract is a good information system and it provides some new ideas to the bridge players and helpful for beginners and semi professional players too in improving their bridge skills.

REFERENCES

The author wishes to express special thanks to the authorities of University Grants Commission for having sanctioned a Minor Research Project during the academic year 2014-2016 in which the author as a Principal Investigator has done this Research as a main focus of his study.

[1]. Amit,A.,& Markovitch,S. (2006). Learning to bid in bridge. Machine Learning, 63(3), 287-327.
[2]. Ando,T, Sekiya,Y., & Uehara,T. (2000). Partnership bidding for Computer Bridge. Systems and Computers in Japan, 31(2), 72-82.
[3]. Dharmalingam,M., & Amalraj,R (2013a). Neural Network Architectures for Solving the Double Dummy Bridge Problem in Contract Bridge. In Proceeding of the PSG-ACM National Conference on Intelligent Computing.1(1),31-37.
[4]. Fahlman,S.E.& Lebiere.V. (1990). The cascade-correlation learning architecture. Advances in Neural Information Processing. 524-532.
[5]. Francis,H.,Truscott,A & Francis,D (1994). The Official Encyclopedia of Bridge, American Contract Bridge League.(5th) Edition.
[6]. Frank, I., & Basin, D.A. (2001). A Theoretical and Empirical Investigation of Search in Imperfect Information Game. Theoretical Computer Science, 252(1), 217-256.
[7]. Frank, I., & Basin, D.A. (1999). Optimal play against best Defence: Complexity

and Heuristics. In Lecture Notes in Computer Science, vol, 1558. 1999 (pp.50-73). Germany: Springer – Verlag.

[8]. Jamroga, W. (1999). Modeling Artificial Intelligence on a case of bridge card play bidding. In Proceedings of International Workshop on Intelligent Information System. 276-277.

[9]. Mandziuk,J. (2008). Some thoughts on using Computational Intelligence methods in classical mind board games. In Proceedings of the International Joint Conference on Neural Networks. 4001-4007.

[10]. Mandziuk,J.(2007). Computational Intelligence in Mind Games. Springer, (Chapter 2).

[11]. Mandziuk,J.(2010). Knowledge-free and Learning – Based methods in Intelligent Game Playing. Springer, (Chapter 3).

[12]. Mandziuk,J., & Mossakowski,K.(2009a). Neural Networks compete with expert human players in solving the double dummy bridge problem. In Proceedings of the Conference on Computational Intelligence and games, (5), 117-124.

[13]. Mossakowski,K., & Mandziuk,J.(2004). Artificial Neural Networks for solving double dummy bridge problems. In L. Rutkowski, J. H. Siekmann, R. Tadeusiewicz, and L. A. Zadeh,(Eds.), Lecture Notes in Artificial Intelligence,LNAI:vol,3070. Artificial Intelligence and Soft Computing ICAISC 2004 (pp.915-921).Springer.

[14]. Mossakowski,K., & Mandziuk,J.(2009b). Learning without human expertise: A case study of Double Dummy Bridge Problem. IEEE Transactions on Neural Networks, 20(2), 278-299.

[15]. Sarkar,M., Yegnanarayana,B.,& Khemani,D (1995). Application of neural network in contract bridge bidding. In Proceeding of National Conference on Neural Networks and Fuzzy Systems.144-151.

[16]. Smith,S.J.J & Nau,D.S. (1994). An Analysis of Forward Pruning. In Proceeding of the National Conference on Artifitical Intelligence. 1386-1391.

[17]. Smith,S.J.J, Nau,D.S & Throop,T.A. (1998a). Success in Spades: Using AI Planning Techniques to Win the World Championship of Computer Bridge. In Proceeding of the National Conference on Artifitical Intelligence. 1079-1086.

[18]. Smith,S.J.J, Nau,D.S & Throop,T.A. (1998b). Computer Bridge - A Big Win for AI planning. Artificial Intelligence Magazine, 19(2), 93-106.

[19]. Yegnanarayana,B., Khemani,D.,& Sarkar,M. (1996). Neural Network for contract bridge bidding. 21(3), 395-413.