FAST SUPERVISED LEARNING ARCHITECTURE – A WORK POINT COUNT SYSTEM COUPLED WITH RESILIENT BACK – PROPAGATION ALGORITHM FOR SOLVING DOUBLE DUMMY BRIDGE PROBLEM

M Dharmalingam¹ and R Amalraj²

¹Assistant Professor, Dept of Computer Science, Nandha Arts and Science College, Erode, Tamil Nadu, India
²Associate Professor, Dept of Computer Science, Sri Vasavi College, Erode, Tamil Nadu, India

ABSTRACT: Contract Bridge is an intelligent game, which enhances the creativity with multiple skills and quest to acquire the intricacies of the game, because no player knows exactly what moves other players are capable of during their turn. The Bridge being a game of imperfect information is to be equally well defined, since the outcome at any intermediate stage is purely based on the decision made on the immediate preceding stage. One among the architectures of Artificial Neural Networks (ANN) is applied by training on sample deals and used to estimate the number of tricks to be taken by one pair of bridge players is the key idea behind Double Dummy Bridge Problem (DDBP) implemented with the neural network paradigm. This study mainly focuses on Cascade-Correlation Neural Networks (CCNN) which is used to solve the Bridge problem by using Resilient Back-Propagation (RBP) Algorithm and Work Point Count System.

Keywords: Artificial Neural Network, Cascade-Correlation Neural Network, Resilient Back-Propagation Algorithm, Sigmoid activation functions, Contract Bridge, Double Dummy Bridge Problem, Work Point Count System.

1. Introduction

The game of bridge is one of the well known card games played worldwide and drawing attention of many of the researchers and Computational Intelligence (CI) methods are normally applied to games and mainly focused on the aspect of learning in the game playing systems [1]. Artificial Neural Networks (ANN) are based on non-linear activation function approximations which make them suitable for most applications, since the outcome of the game could only be foreseen, but can’t be stated earlier at any stage. However, the training is usually a cumbersome process and requires proper tuning of the learning algorithm, and in practice based on the knowledge and expertise acquired so far in the problem domain.

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 in Resilient Back Propagation algorithm. Many FFNN were trained to solve the Double Dummy Bridge Problems (DDBP) in bridge game [2][3][4][5] 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 new heuristics of beta-reduction and iterative biasing were introduced in the general tree search algorithm capable of providing consistent performance in the actual game. The effectiveness of these heuristics, particularly when combined with payoff-reduction mini-maxing results in an iprm-beta algorithm which makes fewer errors than the human experts and it represents the first general search algorithm capable of consistently performing at and above the expert level on a significant aspect of bridge card play [6].

Two different kinds of game trees viz., N-Game trees and N-Game like trees were used to investigate, how forward pruning affects the possibility of choosing the correct move. The results revealed that, mini-maxing with forward pruning perform better than ordinary mini-maxing, in cases where there was a high correlation among the mini-max values of sibling nodes in a game tree. The results recommended that forward pruning may probably be a viable decision-making method in bridge games [7]. 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 [8].

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 architecture (CCNN) is focused in this paper with Resilient Back-Propagation (RBP) algorithm and the Sigmoidal function is used to train and test the data and the results are compared.

2. 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 process, besides acquiring these basic information, several other more sophisticated features need to be developed by the learning system [9][10].

2.1. 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 [11] 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 declarer’s partner is well-known as the dummy shown in Fig. 1.

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 [12].

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 [13].

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 (♠, ♦, ♣, ♦). 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 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♠. Fig. 2. 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 bidding 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

Fig. 1. A Bridge playing table
played without a trump. After three successive passes, the last bid becomes the contract.

2.3 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 [19]. 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 [20].

2.4 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 1 point 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.

3. 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.

4. 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 [21] 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.

5. The Resilient Back-Propagation (RBP) Algorithm

The algorithm RBP is a local adaptive learning scheme, performing supervised batch learning in cascade-correlation neural networks. The basic principle of RBP is to eliminate the harmful influence of the size of the partial derivative on the weight step. As a consequence, only the sign of the derivative is considered to indicate the direction of the weight update.

The algorithm acts on each weight separately. For each weight, if there was a sign change of the partial derivative of the total error function compared to the last iteration, the update value for that weight is multiplied by a factor \( \eta^- \), where \( 0 < \eta^- < 1 \). If the last iteration produces the same sign, the update value is multiplied by a factor of \( \eta^+ \), where \( \eta^+ > 1 \). The update values are calculated for each weight in the above manner, and finally each weight is changed by its own update value, in the opposite direction of that weight's partial derivative. This is to minimize the total error function. \( \eta^+ \) is empirically set to 1.2 and \( \eta^- \) to 0.5.

To elaborate the above description mathematically we can start by introducing for each weight \( w_{ij} \) its individual update value \( \Delta_{ij}(t) \), which exclusively determines the magnitude of the weight-update. This update value can be expressed mathematically according to the learning rule for each case based on the observed behavior of the partial derivative during two successive weight-steps by the following formula:

\[
\Delta_{ij}(t) = \left\{ \begin{array}{ll}
\eta^- \Delta_{ij}(t-1), & \text{if } \frac{\partial E}{\partial w_{ij}(t)} \frac{\partial E}{\partial w_{ij}(t-1)} > 0 \\
\eta^+ \Delta_{ij}(t-1), & \text{if } \frac{\partial E}{\partial w_{ij}(t)} \frac{\partial E}{\partial w_{ij}(t-1)} < 0 \\
\Delta_{ij}(t-1), & \text{else}
\end{array} \right.
\]

Where \( 0 < \eta^- < 1 < \eta^+ \).

A clarification of the adaptation rule based on the above formula can be stated. It is evident that whenever the partial derivative of the equivalent weight \( w_{ij} \) varies its sign, which indicates that the last update was large in magnitude and the algorithm has skipped over a local minima, the update value \( \Delta_{ij}(t) \) is decreased by the factor \( \eta^- \). If the derivative holds its sign, the update value will to some extent increase in order to speed up the convergence in shallow areas. When the update-value for each weight is settled in, the Weight-update itself tracks a very simple rule. That is if the derivative is positive, the weight is decreased by its update value, if the derivative is negative, the update-value is added.

\[
\Delta_{ij}(t) = \left\{ \begin{array}{ll}
-\Delta_{ij}(t), & \text{if } \frac{\partial E}{\partial w_{ij}} > 0 \\
\Delta_{ij}(t), & \text{if } \frac{\partial E}{\partial w_{ij}} < 0 \\
0, & \text{else}
\end{array} \right.
\]

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.

\[
S = \sum_{i=1}^{O} \sum_{n=1}^{P} \sum_{p=1}^{V_{np}} (E_n - P_n) (E_{np} - E_o)
\]  

(1)

where \( O \) is the number of network outputs, \( P \) is the number of training patterns, \( V_{np} \) is output on the new hidden neuron and \( E_{np} \) 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

\[
\Delta_{new} = \sum_{i=1}^{O} \sum_{n=1}^{P} \sum_{p=1}^{V_{np}} (E_{np} - E_o) f'_{np} \eta_{np}
\]

(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.
However, there is one exception. If the partial derivative changes sign that is the previous step was too large and the minimum was missed, the previous weight-update is reverted

\[
\Delta w_y (t-1) = - \delta_y (t-1),
\]

Due to that ‘backtracking’ weight-step, the derivative is assumed to change its sign once again in the following step. In order to avoid a double penalty of the update-value, there should be no adaptation of the update-value in the succeeding step. In practice this can be done by setting \( \frac{\delta_y}{\delta w_y} (t -1) = 0 \) in the \( \Delta w_y \) update-rule above.

The partial derivative of the total error is given by the following formula:

\[
\frac{\delta_y}{\delta w_y} (t) = \frac{1}{2} \sum_{j=1}^{n} u_j (t),
\]

Hence, the partial derivatives of the errors must be accumulated for all training patterns Fig.4. This indicates that the weights are updated only after the presentation of all of the training patterns [22]. It is noticed that resilient back-propagation is much faster than the standard steepest descent algorithm.

5.1. RBP Algorithm Architecture

The minimum (maximum) operator is supposed to deliver the minimum (maximum) of two numbers; the sign operator returns +1, if the argument is positive, -1, if the argument is negative and 0 otherwise. Repeat

Compute Gradient \( \frac{\delta_y}{\delta w_y} (t) \)

For all weights and biases {

1. if \( (\frac{\delta_y}{\delta w_y} (t -1) + \frac{\delta_y}{\delta w_y} (t) \geq 0) \) then { 
   \[ \Delta w_y (t) = \min \left( \Delta w_y (t -1) + \eta \phi_y (t), \Delta w_y_{\text{max}} \right) \]
   \[ \Delta w_y (t) = - \text{sign} \left( \frac{\delta_y}{\delta w_y} (t -1) + \frac{\delta_y}{\delta w_y} (t) \right) \Delta w_y_{\text{max}} \]
   \[ \Delta w_y (t) = w_y (t -1) + \Delta w_y (t) \]
 }

2. else if \( (\frac{\delta_y}{\delta w_y} (t -1) + \frac{\delta_y}{\delta w_y} (t) < 0) \) then {
   \[ \Delta w_y (t) = \max \left( \Delta w_y (t -1) + \eta \phi_y (t), \Delta w_y_{\text{max}} \right) \]
   \[ \Delta w_y (t) = \frac{\delta_y}{\delta w_y} (t -1) \]
 }

3. else if \( (\frac{\delta_y}{\delta w_y} (t -1) + \frac{\delta_y}{\delta w_y} (t) = 0) \) then {
   \[ \Delta w_y (t) = \text{sign} \left( \frac{\delta_y}{\delta w_y} (t -1) + \frac{\delta_y}{\delta w_y} (t) \right) \Delta w_y_{\text{max}} \]
   \[ \Delta w_y (t) = \frac{\delta_y}{\delta w_y} (t -1) = \frac{\delta_y}{\delta w_y} (t) \]
 }

Until(converged)

In our research we used Resilient Back-Propagation (RBP) algorithm to train and test the data in MATLAB 2008a.

6. Implementation

In this paper 20 sample data were used for training in MATLAB 2008a. Only one output neuron was used and in order to get the result, decision boundaries were defined the range of 0.1 to 0.9 denoting particular number of tricks. The rank of the card was transformed using a uniform linear transformation to the range from 0.1 to 0.9 with biggest values to lowest values. The decision boundaries were defined a priori and target ranges for all possible number of tricks from 0 to 13 were pair wise equal length. Gradient descent training function was used to train the data and gradient descent weight/bias learning function was used for learning the data. For training and learning the data, two transfer functions viz., Log Sigmoid transfer function and Hyperbolic Tangent Sigmoid functions were used. The results produced are represented in Table 1 and Table 2 respectively.

<table>
<thead>
<tr>
<th>No</th>
<th>Actual value in GIB</th>
<th>Calculated value in Log Sigmoid transfer function</th>
<th>Calculated value in Hyperbolic Tangent Sigmoid function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.75000</td>
<td>0.50537</td>
<td>0.56336</td>
</tr>
<tr>
<td>2</td>
<td>0.83000</td>
<td>0.76809</td>
<td>0.69161</td>
</tr>
<tr>
<td>3</td>
<td>1.00000</td>
<td>0.83654</td>
<td>0.84209</td>
</tr>
<tr>
<td>4</td>
<td>0.83000</td>
<td>0.62388</td>
<td>0.70788</td>
</tr>
<tr>
<td>5</td>
<td>0.75000</td>
<td>0.54815</td>
<td>0.53020</td>
</tr>
<tr>
<td>6</td>
<td>0.50000</td>
<td>0.50021</td>
<td>0.59114</td>
</tr>
<tr>
<td>7</td>
<td>0.58000</td>
<td>0.50046</td>
<td>0.50565</td>
</tr>
<tr>
<td>8</td>
<td>0.75000</td>
<td>0.54373</td>
<td>0.94029</td>
</tr>
<tr>
<td>9</td>
<td>0.50000</td>
<td>0.50018</td>
<td>0.50012</td>
</tr>
<tr>
<td>10</td>
<td>0.83000</td>
<td>0.84651</td>
<td>0.84685</td>
</tr>
<tr>
<td>11</td>
<td>0.58000</td>
<td>0.50195</td>
<td>0.51703</td>
</tr>
<tr>
<td>12</td>
<td>1.00000</td>
<td>0.83235</td>
<td>0.72176</td>
</tr>
<tr>
<td>13</td>
<td>0.58000</td>
<td>0.61464</td>
<td>0.50233</td>
</tr>
<tr>
<td>14</td>
<td>0.50000</td>
<td>0.52667</td>
<td>0.50134</td>
</tr>
<tr>
<td>15</td>
<td>0.91000</td>
<td>0.58635</td>
<td>0.95803</td>
</tr>
<tr>
<td>16</td>
<td>0.50000</td>
<td>0.78009</td>
<td>0.96317</td>
</tr>
<tr>
<td>17</td>
<td>0.50000</td>
<td>0.50010</td>
<td>0.50110</td>
</tr>
<tr>
<td>18</td>
<td>0.83000</td>
<td>0.50799</td>
<td>0.62475</td>
</tr>
<tr>
<td>19</td>
<td>0.66000</td>
<td>0.50299</td>
<td>0.65817</td>
</tr>
<tr>
<td>20</td>
<td>0.58000</td>
<td>0.53902</td>
<td>0.50061</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No</th>
<th>Actual value in GIB</th>
<th>Calculated value in Log Sigmoid transfer function</th>
<th>Calculated value in Hyperbolic Tangent Sigmoid function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.83000</td>
<td>0.94354</td>
<td>0.99995</td>
</tr>
<tr>
<td>2</td>
<td>0.83000</td>
<td>0.56256</td>
<td>0.99996</td>
</tr>
<tr>
<td>3</td>
<td>0.50000</td>
<td>0.70507</td>
<td>0.99778</td>
</tr>
<tr>
<td>4</td>
<td>0.75000</td>
<td>0.96712</td>
<td>0.77965</td>
</tr>
<tr>
<td>5</td>
<td>0.83000</td>
<td>0.64946</td>
<td>0.99962</td>
</tr>
</tbody>
</table>
7. Results and Discussion
The results presented in the Fig 5, Fig 6, Fig 7 and Fig 8 shown that the comparison of target tricks along with Log Sigmoid transfer function and Hyperbolic Tangent Sigmoid function. While comparing the train and test data along with target data, the result indicated that, train and test data shown significantly better results in both transfer functions, which minimized the total mean square.

The result also revealed that, Log Sigmoid transfer function and Hyperbolic Tangent Sigmoid functions were compared with each other and Hyperbolic Tangent Sigmoid function was given significantly superior results than Log Sigmoid transfer function. 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.

8. 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. In the Work Point Count System, Log Sigmoid transfer function and Hyperbolic Tangent Sigmoid functions were used in Resilient Back Propagation algorithm. Hyperbolic Tangent Sigmoid function produced better result when compared to Log Sigmoid transfer function which is used to bid a final contract. It provides some new ideas to the bridge players and helpful for beginners and semi professional players to improving their bridge skills.

References


AUTHORS

Mr M Dharmalingam received his Under Graduate, Post Graduate and Master of Philosophy degrees from Bharathiyar University, Coimbatore in 2000, 2004 and 2008 respectively. He is a Doctoral Research Scholar of Sri Vasavi College at Erode and his research interest is Soft Computing.

Dr. R. Amalraj is an Associate Professor in Computer Science in the Department of Computer Science, Sri Vasavi College, Erode. He obtained his PhD in Computer Science from PSG College of Technology, affiliated to Bharathiar University, Coimbatore in 2003. He is a Life Member of Computer Society of India. He has been a Principal Investigator for a Minor Project sponsored by University Grants Commission, New Delhi. He has published several research papers in reputed National and International journals. His specific interests include Artificial Intelligence, Image Processing and Soft Computing.