ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 12, December 2013

ARTIFICIAL NEURAL NETWORK ARCHITECTURE FOR SOLVING THE DOUBLE DUMMY BRIDGE PROBLEM IN CONTRACT BRIDGE

M Dharmalingam[1] and R Amalraj[2]

Ph.D Research Scholar, Department of Computer Science, Bharathiar University, Coimabatore, India[1]

Associate Professor Department of Computer Science, Bharathiar University, Coimabatore, India[2]

Abstract: Card games are interesting for many reasons besides their connection with gambling. Bridge is being a game of imperfect information, it is a well defined, decision making game. The estimation of the number of tricks to be taken by one pair of bridge players is called Double Dummy Bridge Problem (DDBP). Artificial Neural Networks are Non – Linear mapping structures based on the function of the human brain. Feed Forward Neural Network is used to solve the DDBP in contract bridge. The learning methodology, supervised learning was used in Back – Propagation Network (BPN) for training and testing the bridge sample deal. In our study we compared back – Propagation algorithm and obtained that Resilient Back – Propagation algorithms by using Hyperbolic Tangent function and Resilient Back – Propagation algorithm produced better result than the other. Among various neural network architectures, in this study we used four network architectures viz., 26x4, 52, 104 and 52x4 for solving DDBP in contract bridge.

Key words: BPN, Contract Bridge, Back – Propagation Algorithm, Resilient Back – Propagation Algorithm, Hyperbolic Tangent function.

I. INTRODUCTION

In the game playing domain the most popular Computational Intelligence (CI) disciplines are Neural Networks (NN), Evolutionary Methods (EM), and Supervised Learning (SL) [1]. Neural Networks are computational structure capable of processing information in order to finish a given task. A Neural Network is composed of many simple neurons each of which receives input from selected other neurons, and performs basic operations on this input information and send its response out to other neurons in the network. Stimulation for the above way of processing information is biological anxious system and in particular biological brain. NN models can therefore be regarded as very rough simplification and abstraction of biological networks. NN have been successfully applied to various recognition, classification problems [2] and games [3] [36][37][38][39].

The Card game is skillful and knowledgeable which it increases the creativity of the human mind and there are extremely powerful Artificial Neural Network (ANN) approaches in which playing agents are equipped with carefully designed evaluation functions. Artificial Neural Networks (ANNs) are non – linear mapping structures based on the function of the human brain. Neural networks are type of artificial intelligence that attempts to imitate the way a human brain works rather than using a digital model [31]. The Feed-Forward Neural Networks (FFNN) are one of the most common types of neural network in use and these are often trained by the help of supervised learning supported by Back-propagation algorithm [5]. Many of the feed-forward neural networks were trained to solve the Double Dummy Bridge Problems (DDBP) in bridge game [9][21][22][40]. Among the various neural networks, in this paper we mainly focus Back-propagation Network (BPN) for training and testing the data. Back - Propagation Algorithm and Resilient - Back Propagation Algorithms were used in BPN network to train the data for solving Double Dummy Bridge Problems in Contract Bridge.

II. PROBLEM DESCRIPTION

In bridge games, basic representation include 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 course learning, besides acquiring this basic information several other more sophisticated game features need to be developed by the learning system [8][9].

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 12, December 2013

A. The Game of Contract Bridge

Contract bridge, usually known simply as bridge, is a trick - taking card game. There are four players in two fixed partnerships (Pairs). Partners sit facing each other. It is traditional to refer to the players 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*. Example shown in Fig 1.

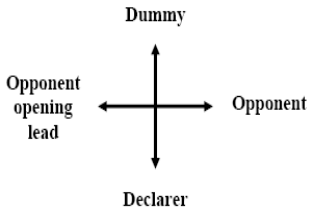

Fig 1 Game disposition

A standard 52 card pack is used. The cards in each suit rank from highest to lowest: *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. Next level bid to decide who will be the declarer takes place. A bid specifies a number of tricks and a trump suit (or that there will be no trumps). 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 consecutive passes, the last bid becomes the contract. The team who made the final bid will now try to make the contract. The first player of this team who mentioned the denomination (suit or *no-trump*) of the contract becomes the declarer. The declarer's partner is known as the dummy shown in Fig 2.

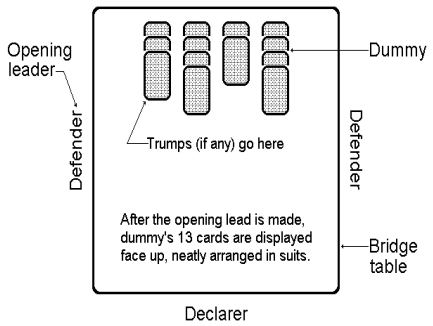

Fig 2 Bridge Table

The player to the left of the declarer leads to the first trick. Immediately after this opening lead, the dummy's cards are exposed. The play proceeds clockwise. Each player must, if possible, play a card of the suit led. A player with no card of the suit led may play any card. A trick consists of four cards, and is won by the highest trump in it, or if no trumps were played by the highest card of the suit led. The winner of a trick leads to the next. The aim of the declarer is to take at least the number of tricks announced during the bidding phase. The players of the opposite pair try to prevent him from doing it [6][7]. In bridge, special focus in game representation is on the fact that players cooperate in pairs, thus sharing potentials of their hands.

B. Double Dummy Bridge Problem

To estimate the number of tricks to be taken by one pair of bridge players is called Double Dummy Bridge Problem (DDBP). A bridge problem presented for entertainment, in which the solver is presented with all four hands and is asked to determine the course of play that will achieve or defeat a particular contract. The partners of the declarer, whose cards are placed face up on the table and played by declarer. Dummy has few rights and may not participate in choices concerning the play of the hand [9].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 and during the bidding phase. This incompleteness of information force considering many variants of a deal cards distributions. The player should take into account all these variants and quickly estimate the expected number of tricks to be taken in each case [10] [21][22]

C. The Bidding phase

The bidding phase is a conversation between two cooperating team members against an opposing partnership. It 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. Each player has knowledge of his own hand and any previous bids only. A very interesting aspect of the bidding phase is cooperation of players in a North with South and West with East. In each, player is modeled as an independent, active agent that takes part in the communication process. The agent-based algorithm to use of achieve in appropriate learning, a bidding ability close to that of a human expert [11] [12] [13] [14] [15].

D. The Play Phase

In the game, the play phase seems to be much less interesting than the bidding phase. Artificial Intelligence (AI) approaches tried to imitate human strategy of the play by using some "tactics". The new system was able to find a strategy of play and additionally a "human" explanation of it [16] [17].The player to the left of the declarer leads to the first trick and may play any card. Immediately after this opening lead, the dummy's cards are exposed. Play proceeds clockwise. Each of the other three players in turn must, if possible, play a card of the same suit that the leader played. A player with no card of the suit led may play any card. A trick consists of four cards, one from each player, and is won by the highest trump in it, or if no trumps were played by the highest card of the suit led. The winner of a trick leads to the next and may lead any card. Dummy takes no active part in the play of the hand and is

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 12, December 2013

not permitted to offer any advice or comment on the play. Whenever it is dummy's turn to play, the declarer must say which of dummy's cards is to be played, and dummy plays the card as instructed. Finally, the scoring depends on the number of tricks taken by the declarer team and the contract [18].

III. THE DATA REPRESENTATION OF GIB LIBARARY

The data used in this game of DDBP was taken from the Ginsberg's Intelligent Bridge (GIB) Library. The data created by Ginsberg's Intelligent Bridge player [19][35]. In our research for implementing GIB library data we used MATLAB 2008a.The GIB library includes 7,17,102 deals and for each of them 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. Together there are 20 numbers of each deal i.e. 5 trump suits by 4 sides. Here 5 trump suits are No-trumps, spades, Hearts, Diamonds and Clubs. No-trump which is the term for contracts played without trump. Four sides are West, North, East and South. So North and South are partners playing against East and West [20].

IV. SOFT COMPUTING

Nowadays the on-going development of computer technology, soft computing will considerably enhance traditional computation methods. The machine-intelligent behavior is determined by the flexibility of the architecture, the ability to recognize machine incorporations of human expertise, laws of inference procedure and the high speed of learning. All these titles are the main constituents of the research area named Soft Computing and it is a practical alternative for solving mathematically complex problems [4]. Soft computing involves partnership of several fields, the most important being Artificial Neural Networks (ANN), Fuzzy Logic (FL), Genetic Algorithm (GA) and Evolutionary Computations (EC) [5]. Among the above fields, Artificial Neural Networks used to solve the Double Dummy Bridge Problem in Contract Bridge.

V. ARTIFICIAL NEURAL NETWORKS (ANN)

Artificial Neural Network consists of several processing units which are interconnected according to some topology to accomplish a pattern classification task. An Artificial Neural Network is configured for a specific application, such as pattern recognition or data classification through learning process. ANNs are Non-linear information processing devices, which are built from interconnected elementary processing devices called neurons [23][24][25].

In Artificial Neural Networks following the supervised learning, each input vector requires a corresponding target vector, which represents the desired output. The input vector along with the target vector is called *training pair*. In supervised learning, a supervisor is required for error minimization. Hence the network trained by this method is said to be using supervised learning methodology. In supervised learning, it is assumed that the correct target output values are known for each input pattern [26][27][28][29][30][31][32].

A. Activation Functions

The activation function is used to determine the output response of a neuron. The sum of the weighted input signal is applied with an activation to obtain the response. For neurons in same layer, same activation functions are used. There may be linear as well as Non - Linear activation functions. The Non–Linear activation functions are used in a Multilayer Neural Network. There are several activation functions namely as follows Identity function, Binary step function, bipolar step function, Sigmoidal functions and Ramp functions.

Among the above activation functions, the Sigmoidal functions are widely used in Back-propagation networks. Because of the relationship between the value of the functions at a point and the value of derivative at that point which reduce the computational burden during training. There are two types of sigmoidal functions viz., binary sigmoidal function and bipopular sigmoidal functions [27]. In this paper we have focused only Binary Sigmoidal function. If the network uses a binary it is better to convert it to bipolar form and use the bipolar sigmoid function activation function or hyperbolic tangent function shown in the Fig 4.

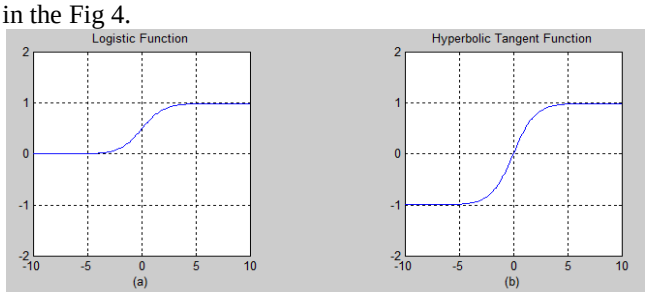

Fig 4.Logistic function and Hyperbolic Tangent function

ii. B. Supervised Learning Methodology

Learning or training is a process by means of which a neural network adopts itself to stimulus by making proper parameter adjustments, resulting in the production of desired response. The Learning in an Artificial Neural Networks can be generally classified into three categories viz., Supervised Learning, Unsupervised Learning and Reinforcement Learning. Among these three categories we mainly focused on Supervised Learning in this paper.

In Artificial Neural Networks following the supervised learning, each input vector requires a corresponding target vector, which represents the desired output. The input vector along with the target vector is called *training pair*.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 12, December 2013

In supervised learning, a supervisor is required for error minimization. Hence the network trained by this method is said to be using supervised learning methodology. In supervised learning, it is assumed that the correct target output values are known for each input pattern [5][26][27][28].

VI. Architecture of Back-propagation Network (BPN)

A Back-propagation network is a multilayer, Feed-Forward Neural Network (FFNN) with an input layer, a hidden layer and an output layer which is shown in the Fig 5. The neuron in the hidden and the output layers has biases which are connections from units whose output is always is 0 to 1.

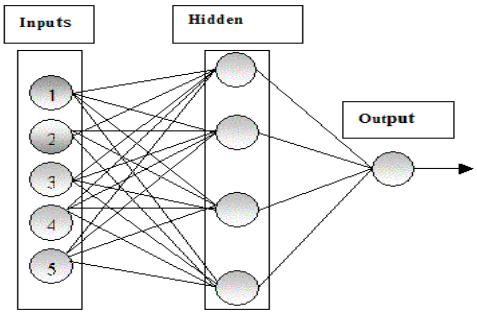

Fig 5 Architecture of BPN

BPN is a multi – layer forward network using extended gradient-descent based delta-learning rule, commonly known as Back-propagation rule. Back-propagation provides a computationally efficient method for changing the weights in a feed-forward network, with differentiable activation function units to learn a training set of input output patterns. The Back-propagation network implements the generalized delta rule. A gradient – descent method, it minimize the total squared error of the output computed by the network. The network is trained by supervised learning method [5][24][26].

i. A. BPN training algorithm

The training algorithm of Back-propagation invokes four stages viz., Initialization of weights, Feed-forward, Back-propagation of errors and Updating of the weights and bias.

The detailed algorithm is shown in Fig 6.

function BACK-PROP-LEARNING(*example, network*)
returns a neural network
inputs: *example*, a set of examples, each with input vector **x** and output vector **y**
network, a multilayer network with L layers, weights $W_{j,i}$, activation function g
repeat
for each *e* **in** *examples* **do**
for each node j in the input layer **do** $a_j \leftarrow x_j[e]$
for $\ell = 2$ **to** M **do**
$in_j \leftarrow \sum_j W_{j,i}\, a_j$
$a_i \leftarrow g(in_j)$
for each node i in the output layer **do**
$\Delta_i \leftarrow g'(in_i) \times (y_i[e] - a_i)$
for $\ell = M - 1$ **to** l **do**
for each node j in layer ℓ **do**
$\Delta_j \leftarrow g'(in_j) \sum_i W_{j,i}\ \Delta_i$
for each node i in layer $\ell + 1$ **do**
$W_{j,i} \leftarrow W_{j,i} + \alpha \times a_j \times \Delta_i$
until some stopping criterion is satisfied
return NEURAL-NET-HYPOTHESIS(*network*)

Fig 6 The back-propagation algorithm for learning in BPN networks

For the mathematically inclined, the back-propagation equations are derived from first principals. The squared error on a single example is defined as,

$$E = \frac{1}{2}\sum_i (y_i - a_i)^2 \qquad (1)$$

where the sum is over the nodes in the output layer. To obtain the gradient with respect to a specific weight $W_{j,i}$ in the output layer, one should need only expand out the activation α_i as all other terms in the summation are unaffected by $W_{j,i}$:

$$\frac{\partial E}{\partial w_{i,j}} = -(y_i - a_i)\frac{\partial a_i}{\partial w_{i,j}} = -(y_i - a_i)\frac{\partial g(in_i)}{\partial w_{i,j}}$$
$$= -(y_i - a_i)g'(in_i)\frac{\partial in_i}{\partial w_{i,j}}$$
$$= -(y_i - a_i)g'(in_i)\frac{\partial}{\partial w_{i,j}}\left(\sum_j w_{i,j}\, a_j\right)$$
$$= -(y_i - a_i)g'(in_i)a_j$$
$$= -a_j \Delta_i, \qquad (2)$$

With Δ_i defined as before. To obtain the gradient with respect to the $W_{k,j}$ weights connecting the input layer to the hidden layer, to keep the entire summation over i because each output value a_i may be affected by changes in $W_{k,j}$. Activations of a_j also be expanded and the derivative operator propagates back through the network

$$\frac{\partial E}{\partial w_{i,j}} = -\sum_i (y_i - a_i)\frac{\partial a_i}{\partial w_{i,j}} = -\sum_i (y_i - a_i)\frac{\partial g(in_i)}{\partial w_{k,j}}$$

$$= -\sum_i (y_i - a_i)g'(in_i)\frac{\partial in_i}{\partial w_{k,j}} = -\sum_i \Delta_i \frac{\partial}{\partial w_{i,j}}\left(\sum_j w_{i,j}\, a_j\right)$$

$$= -\sum_i \Delta_i\, w_{k,j}\, \frac{\partial a_i}{\partial w_{k,j}} = -\sum_i \Delta_i\, w_{j,i}\frac{\partial g(in_j)}{\partial w_{k,j}}$$

$$= -\sum_i \Delta_i\, w_{k,j}\, g'(in_j)\frac{\partial in_i}{\partial w_{k,j}}$$

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 12, December 2013

$$= -\sum_i \Delta_i \, w_{k,j} \, g'(in_j) \frac{\partial}{\partial w_{k,j}} \left(\sum_k w_{k,j} \, a_k \right)$$

$$= -\sum_i \Delta_i \, w_{k,j} \, g'(in_j) a_k = -a_k \Delta_j, \quad (3)$$

where Δ_j is defined as before. The update rules were obtained earlier from intuitive considerations. It is also clear that the process can be continued for networks with more than one hidden layer, which justifies the above general algorithm [28-29][32-33].

B. The Resilient Back-Propagation (RBP) Algorithm

The algorithm RBP is a local adaptive learning scheme, performing supervised batch learning in feed – forward 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 η−, where 0 <η− < 1. If the last iteration produces the same sign, the update value is multiplied by a factor of η +, where η+ > 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. η+ is empirically set to 1.2 and η− to 0.5.

To elaborate the above description mathematically we can start by introducing for each weight w_{ij} its individual update value Δ_{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:

$$\Delta_{ij}(t) = \begin{cases} \eta^+ . \Delta_{ij}(t-1), & if \ \frac{\partial E}{\partial w_{ij}}(t) . \frac{\partial E}{\partial w_{ij}}(t-1) > 0 \\ \eta^- . \Delta_{ij}(t-1), & if \ \frac{\partial E}{\partial w_{ij}}(t) . \frac{\partial E}{\partial w_{ij}}(t-1) < 0 \\ \Delta_{ij}(t-1), & else \end{cases} \quad (4)$$

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 Δ_{ij} (t) is decreased by the factor η−. 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.

$$\Delta w_{ij}(t) = \begin{cases} -\Delta_{ij}(t), & if \ \frac{\partial E}{\partial w_{ij}}(t) > 0 \\ \Delta_{ij}(t), & if \ \frac{\partial E}{\partial w_{ij}}(t) < 0 \\ 0, & else \end{cases} \quad (5)$$

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t) \quad (6)$$

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

$$\Delta w_{ij}(t) = -w_{ij}(t-1), \quad if \ \frac{\partial E}{\partial w_{ij}}(t) . \frac{\partial E}{\partial w_{ij}}(t-1) < 0 \quad (7)$$

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 $\frac{\partial E}{\partial w_{ij}}(t-1) = 0$ in the Δ_{ij} update-rule above.

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

$$\frac{\partial E}{\partial w_{ij}}(t) = \frac{1}{2} \sum_{p=1}^{p} \frac{\partial E_p}{\partial w_{ij}}(t) \quad (8)$$

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, [34]. It is noticed that resilient back-propagation is much faster than the standard steepest descent algorithm.

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

C. 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**

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 12, December 2013

Compute Gradient $\frac{\partial E}{\partial w}(t)$
For all weights and biases {
if $\left(\frac{\partial E}{\partial w_{ij}}(t-1) * \frac{\partial E}{\partial w_{ij}}(t) > 0\right)$ **then** {
$\Delta_{ij}(t) =$ **minimum** $(\Delta_{ij}(t-1) * \eta^{+}, \Delta_{\max})$
$\Delta w_{ij}(t) = -$**sign** $\left(\frac{\partial E}{\partial w_{ij}}(t)\right) * \Delta_{ij}(t)$
$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$
$\frac{\partial E}{\partial w_{ij}}(t-1) = \frac{\partial E}{\partial w_{ij}}(t)$
}
else if $\left(\frac{\partial E}{\partial w_{ij}}(t-1) * \frac{\partial E}{\partial w_{ij}}(t) < 0\right)$ **then {**
$\Delta_{ij}(t) =$ **maximum** $(\Delta_{ij}(t-1) * \eta^{-}, \Delta_{\min})$
$\frac{\partial E}{\partial w_{ij}}(t-1) = 0$
}
else if $\left(\frac{\partial E}{\partial w_{ij}}(t-1) * \frac{\partial E}{\partial w_{ij}}(t) = 0\right)$ **then {**
$\Delta w_{ij}(t) = -$**sign** $\left(\frac{\partial E}{\partial w_{ij}}(t)\right) * \Delta_{ij}(t)$
$\frac{\partial E}{\partial w_{ij}}(t-1) = \frac{\partial E}{\partial w_{ij}}(t)$
$\frac{\partial E}{\partial w_{ij}}(t-1) = \frac{\partial E}{\partial w_{ij}}(t)$
}
}
Until(converged)

Fig 7. The Resilient Back-Propagation Algorithm for learning in BPN networks

VII. NEURAL NETWORKS IN DOUBLE DUMMY BRIDGE PROBLEM (DDBP)

There are several Neural Network architectures have been used to solving the Double Dummy Bridge Problem. In this paper we focus four neural network architectures viz., 26x4, 52, 104 and 52x4 for solving the DDBP in contract bridge.

A. 26 *x* 4 Representations

In the first way of deal representation, 104 input values were used which were grouped into 52 pairs. Each pair represented one card. The first value in a pair determined the rank of a card (*A*, *K*, *Q*, etc.) and the second one represented the suit of a card (♠, ♥, ♦ or ♣). Hence, 26 input neurons (13 pairs) were necessary to fully describe the content of one hand which is represented in the Fig 8.

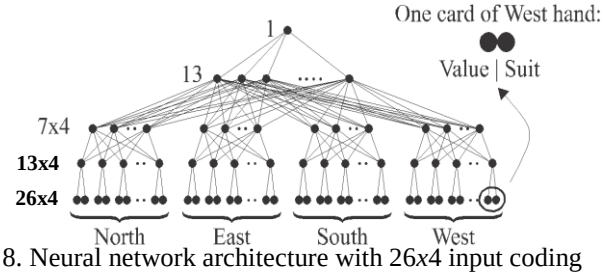

Fig 8. Neural network architecture with 26*x*4 input coding

Neurons in the first two hidden layers are connected selectively. They are responsible for collecting information about individual hands. A few schemes of transforming card's *rank* and *suit* into real numbers suitable as input values for the network were tested. Finally the rank of a card was represented using a uniform linear transformation to the range 0.1 to 0.9 with the biggest values for *Aces* (0.9), *Kings* (0.83) and the smallest for *three spots* (0.17) and *two spots* (0.1). A suit of a card was also coded as a real number, usually by the following mapping: 0.3 for ♠, 0.5 for ♥, 0.7 for ♦, and 0.9 for ♣.

In order to allow the network to gather full information about cards' distribution, special groups of neurons were created in subsequent layers. The Network (26*x*4) − (13*x*4) − (7*x*4) − 13−1 was composed of five layers of neurons arranged in a way depicted in Fig 8. The first hidden layer neurons were responsible for collecting the information about individual cards. Four groups of neurons in the second hidden layer gathered information about the respective hands. The last hidden layer combined the whole information about a deal. This layer was connected to a single output neuron, whose output range is between 0.1 to 0.9. This denotes the particular number of tricks. The decision boundaries were defined a priori and target ranges for all possible number of tricks from 0 to 13 were of pair wise equal length.

B. 52 Representation

In this architecture, 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 9). This way each of the 52 input neurons was assigned to a particular card from a deck and a value presented to this neuron determined the hand to which the respective card (assigned to this input) belonged, i.e. 1.0 for *North*, 0.8 for *South*, −1.0 for *West*, and −0.8 for *East*.

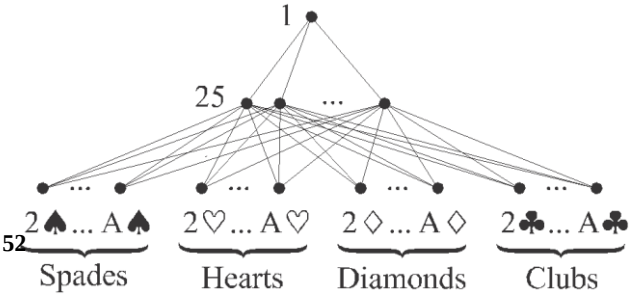

Fig 9. Neural network architecture with 52 input neurons

Layers were fully connected, i.e., in the 52 − 25 − 1 network all 52 input neurons where connected to all 25 hidden ones, and all hidden neurons were connected to a single output neuron, whose role was the same as 26x4 representation.

C. 104 Representation

The next proposed way of coding a deal was a straightforward extension of the 52 representation to the 104 representation. The first 52 input values represented assignments to pairs in a similar way as in the 52

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 12, December 2013

representation, with value 1.0 and -1.0 representing *NS* cards and *WE* cards respectively. The remaining 52 inputs pointed out the exact hand in a pair (value 1.0 for *N* and *W*, and −1.0 for *S* and *E*). In both groups of input neurons positions of cards were fixed according to the same order Fig 10.

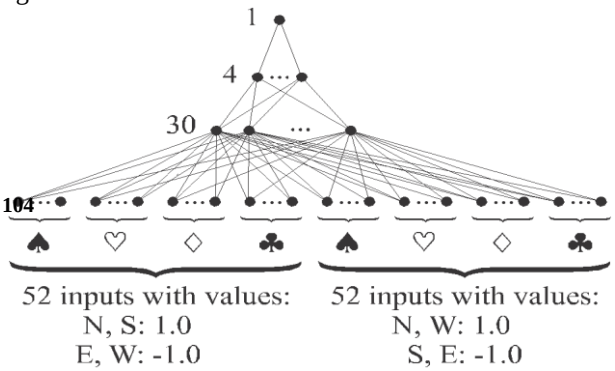

Fig 10. Neural network architecture with 104 input neurons

Networks using this coding were fully connected and usually contained two layers of hidden neurons, e.g. 104 − 30 − 4 − 1. Both two layers of hidden neurons were connected with a single output neuron, whose role was the same as in the previous two cases.

D. 52*x*4 Representation

In this deal coding 208 input neurons were divided into 4 groups, one group per hand, respectively for *N*, *E*, *S* and *W* players. Four input neurons (one per hand) were assigned to each card from a deck. The neuron representing a hand to which this card actually belonged received input value equal to 1.0. The other three neurons (representing the remaining hands) were assigned input values equal to 0.0. This way, a hand to which the card was assigned in a deal was explicitly pointed out.

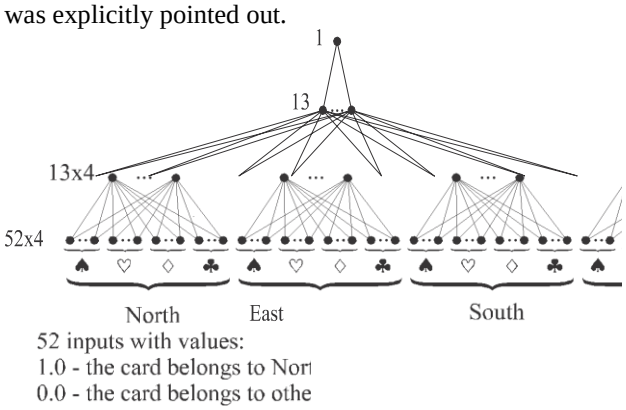

Fig 11. Neural network architecture with 52*x*4 input representation

In this representation each suit on a given hand was represented by 13 input neurons. The number of input values equal to 1.0 among these neurons determined the length of this suit on the hand, so networks using this representation had higher chances to find both long suits and shortnesses (which are very important in bridge), especially voids (no cards in a suit) and singletons (one card in a suit).

There were 4 groups of neurons in the first hidden layer, each of them gathering information from 52 input neurons representing one hand. This data was further compressed in another one or two fully connected hidden layers. A sample network using this way of coding a deal, (52*x*4) − (13*x*4) −13−1, is presented in Fig 11. The 3-hidden layer architecture was realized by the network (52*x*4) − (26*x*4) − 26 − 13 − 1. All neurons from the last hidden layer were connected to a single output neuron.

VIII. REPRESENTATION OF RESULTS

Back - Propagation Algorithm and Resilient – Back - Propagation Algorithm were used for training and testing the data. By using sample data both the algorithms were compared with each other for our study. The four architectures (26 *x* 4 - 25 – 1, 52 - 25 – 1, 104 - 25 – 1 and 52 *x* 4 - 25 – 1) were tested through Back Propagation network by using Back - Propagation Algorithm and Resilient - Back Propagation Algorithm in MATLAB 2008a and the results were produced in Table 1.

Table 1. Comparison of Back - Propagation Algorithm and Resilient–Back Propagation Algorithm in different architectures

S.No	Network Architecture	Back - Propagation Algorithm	Resilient – Back - Propagation Algorithm
1	26 *x* 4 - 25 - 1	15.38	45.86
2	52 - 25 - 1	55.76	65.18
3	104 - 25 - 1	69.23	71.15
4	52 *x* 4 - 25 - 1	71.15	75.96

The result revealed that, the data tested through Resilient – Back - Propagation Algorithm shows better result when compared to Back - Propagation Algorithm. Hence we took Resilient – Back - Propagation Algorithm for our further study.

Gradient descent training function was used to train the data and gradient descent weight/bias learning function was used for learning/training the data. There are two Sigmoidal transfer functions viz., Log Sigmoid Transfer Function and Hyperbolic Tangent Sigmoid Functions were used to training and testing the data. In this experiment we used only Hyperbolic Tangent Sigmoid Function (Bipolar) for testing the data. The above mentioned four architectures were tested through Back Propagation network by using Resilient - Back Propagation Algorithm in MATLAB 2008a.

For our experimental purpose we took 208 deals from GIB library for training and testing through Back Propagation Network. In our study all the 208 deals we fed to BPN with 52 input neurons, 25 hidden layers and one output neuron. The 208 deals were grouped into 4 categories i.e. 26 x 4, 52, 104 and 52 x 4 for training for our

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 12, December 2013

convenience. All the four groups were trained in our MATLAB 2008a and the simulation time was also recorded and the result obtained during training was produced in the Table 2.

Table 2. Training sample deal results

S.No.	Network Architecture	Error (%)	Simulation Time (Min)
1	26 *x* 4 - 25 - 1	80.86	1.39
2	52 - 25 - 1	75.96	1.53
3	104 - 25 - 1	71.15	1.28
4	52 *x* 4 - 25 - 1	65.38	1.90

The training results reported that, 52 *x* 4 - 25 – 1 network architecture minimized the error percentage followed by 104 - 25 - 1 network architecture when compared to other two architectures. Among the four architectures, simulation time taken for training the data was higher in the 52 *x* 4 - 25 – 1 network architecture; hence the error percentage was minimized and has given the highest accuracy when compared to other three architectures.
The trained data was tested through MATLAB 2008a and the result obtained was given in the Table 3.

Table 3. Performance comparison of different Network Architectures

S. No	Network Architecture	Testing output (%)	Error data	Error %
1	26 *x* 4 - 25 - 1	15.38	22	84.61
2	52 - 25 - 1	37.50	34	65.38
3	104 - 25 - 1	55.76	27	25.96
4	52 *x* 4 - 25 - 1	74.03	46	22.11

The results revealed that, while comparing above four architectures, 104 - 25 – 1 and 52 *x* 4 - 25 – 1 architectures produced better results and minimized the error than other two architectures.

Among those four architectures, 52 *x* 4 - 25 – 1 significantly produced better result and reduced the error than other three architectures. For example, while testing 208 deals we got only 22.11% of error from 52 *x* 4 - 25 - 1 and other three architectures 26 *x* 4 - 25 – 1 (26 deals), 52 - 25 – 1 (52 deals) and 104 - 25 – 1(104 deals) produced 84.61%, 65.38% and 25.96% of error respectively which were very much higher than 52 *x* 4 - 25 - 1 architecture.

Performance comparison of 52 *x* 4 - 25 – 1 network architecture and 104 - 25 – 1 network architecture

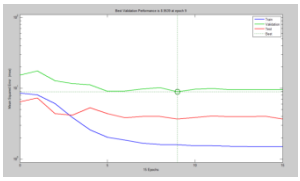

Fig 12a. 52x4-25-1 Network Architecture

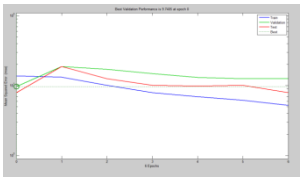

Fig 12b. 104-25-1 Network Architecture

Fig 12a and Fig 12b revealed that, even though both the Network Architectures minimized the mean squared error, 52x4-25-1 Network Architecture significantly produced better result when compared to 104-25-1 Network Architecture**.**

IX. CONCLUSION

Back propagation network was used to minimise the mean squared error in sample data which were used for training and testing in MATLAB 2008a. Back – Propagation Algorithm and Resilient – Back propagation algorithms were used in Back propagation Network. While comparing these two algorithms, Resilient – Back propagation algorithm produce better result which minimize the mean squared error of the output. The best tested algorithms were capable of discovering knowledge concerning the game based exclusively on sample training and testing deals. Among the different neural networks Back - Propagation Network (BPN) appeared to be efficient in solving the Double Dummy Bridge Problem (DDBP) in Contract Bridge and lead to development of some new ideas in human bridge playing and be helpful for beginners and semi professional players in improving their bridge skills.

REFERENCES

[1] Jacek Mandziuk, "*Knowledge-free and Learning – Based methods in Intelligent Game Playing*" Springer, Chapter 5, pp 53 – 70, 2010.
[2] Jacek Mandziuk, "*Computational Intelligence in Mind Games",* In: Studies in Computational Intelligence, vol. 63, Springer, Heidelberg ,pp. 407–442,2007.
[3] I. Frank, D. A. Basin, "A Theoretical and Empirical Investigation of Search in Imperfect Information Game," *Theor.Comput.Sci.*vol.252, no.1-2, pp 217-256, 2001.
[4] B. Yegnanarayana, "*Artificial Neural Networks*" Chapter 4, Printice Hall, 2010.
[5] S.N Sivanandan and S.N Deepa, "*Principles of Soft Computing",* Chapter 3, Wiley India Ltd, First Edition, 2007
[6] H. Francis, A. Truscott, and D. Francis, *The Official Encyclopedia of Bridge*, 5th ed. Memphis, TN: American Contract Bridge League Inc, 1994.
[7] W. H. Root, "*The ABCs of Bridge",* Three Rivers Press, 1998.
[8] Jacek Mandziuk and Krzysztof Mossakowski, "Example – based estimation of hands strength in the game of bridge with or without using explicit human knowledge", *In Proc. IEEE Symp. Comput. Intell. Data mining,* pp.413-420, 2007.
[9] Jacek Mandziuk and Krzysztof Mossakowski, "Neural networks compete with expert human players in solving the double dummy bridge problem*" Proc. of 5th Int. Conf. on Computational Intelligence and games* pp 117-124, 2009.
[10] Krzysztof Mossakowski and Jacek Mandziuk, "Learning without human expertise: A case study of Double Dummy Bridge Problem", *IEEE Transactions on Neural Networks,* vol.20, No.2, pp 278-299, 2009.
[11] W. Jamroga, "Modeling Artificial Intelligence on a case of bridge card play bidding"*In Proce. 8th International Workshop on Intelligent Information System,* Poland, pp 276-277, 1999.
[12] A.Amit and S.Markovitch, "Learning to bid in bridge" *Machine Learning,* vol.63, no.3, pp 287-327, 2006.
[13] T. Ando, Y. Sekiya and T. Uehara, "Partnership bidding for computer bridge," *Systems and Computers in Japan,* vol. 31, No. 2, pp. 72–82, 2000.
[14] T. Ando and T. Uehara, "Reasoning by agents in computer bridge bidding," *in Computers and Games,* vol. 2063, pp. 346–364, 2001.
[15] T. Ando, N. Kobayashi and T. Uehara, "Cooperation and competition of agents in the auction of computer bridge*," Electronics and Communications in Japan,* Part 3, vol. 86, no. 12, pp. 76–86, 2003.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 12, December 2013

[16] D. Khemani, "Planning with thematic actions." *in AIPS*, pp. 287–292, 1994.
[17] I. Frank, and D. A. Basin, "Strategies explained" *in Proceeding 5th Game programming Workshop in* Japan. pp 1-8, 1999.
[18] Ali Awada, May Dehayni and Antoun Yaacoub, "An ATMS-Based Tool for Locating Honor Cards in Rubber Bridge" *Journal of Emerging Trends in Computing and Information Sciences*, vol. 2 No.5, 2011.
[19] M.L. Ginsberg, "GIB: Imperfect Information in a Computaination of the trtionally Challenging Game" *Journal of Artificial Intelligence Research*, vol. 14, pp 303-358, 2001.
[20] Krzysztof Mossakowski and Jacek Mandziuk, "Neural networks and the estimation of hands strength in contract bridge," *in Artificial Intelligence and Soft Computing ICAISC* vol. 4029. Springer, pp. 1189-1198, 2006.
[21] Krzysztof Mossakowski, and Jacek Mandziuk, "Artificial neural networks for solving double dummy bridge problems", In: *AI and Soft computing* vol. 3070, Springer, pp. 915–921, 2004.
[22] S. J. J. Smith, D. S. Nau, and T. A. Throop, "Computer bridge - A big win for AI planning," *AI Magazine.* Vol.19, No.2, pp. 93-106, 1998.
[23] M. Sarkar, B. Yegnanarayana, and D. Khemani, "Application of neural network in contract bridge bidding," *in Proc. of National Conf. on Neural Networks and Fuzzy Systems,* Anna University, Madras, pp. 144-151, 1995.
[24] B.Yegnanarayana, D. Khemani, and M. Sarkar, "Neural networks for contract bridge bidding," *Sadhana*, vol. 21, no. 3, pp. 395-413, June 1996.
[25] S.N Sivanandan and M Paulraj, "*Introduction Artificial Neural Networks"*, Vikas Publishing House Private Limited, 2011.
[26] M. Dharmalingam and R.Amalraj, "Supervised Learning in Imperfect Information Game", *International Journal of Advanced Research in Computer Science*, vol. 4, no.2, pp. 195-200, 2013.
[27] Satish Kumar "*Neural Networks A Class Room Approach",* Chapter 6, pp.164-176, Tata McGraw-Hill Education Private Limited, 2011.
[28] J.Stuart , Russell and Peter Norvig " *Artificial Intelligence A-Modern Approach"* Chapter 18, pp. 727-748, Third Edition., USA Printice Hall 2009.
[29] S.N Sivanandam ,S Sumathi, S.N Deepa, *"Introduction to Neural Network Using MATLAB 6.0",* Tata McGraw-Hill Publishing Company Limited, New Delhi, 2008.
[30] Kishan Mehrotra, Chilukuri, Mohan K and Sanjay Ranka "*Elements of artificial Neural Networks",* pp 65-96, 1996.
[31] S.Haykin, "Neural *Networks: A Comprehensive Foundation",* Prentice-Hall,Englewood Cliffs ,1998.
[32] M. Riedmiller. "Advanced supervised learning in multi-layer perceptrons- From backpropagation to adaptive learning algorithms" *Computer Standards and Interfaces*, 16(5), pp.265-278, 1994.
[33] M. Riedmiller and H. Braun. "A direct adaptive method for faster backpropagation learning: The RPROP algorithm" In E. H. Ruspini, editor, *Proceedings of the IEEE International Conference on Neural Networks*, pp 586-591, 1993.
[34] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning internal representations by error backpropagation." *Parallel Distributed Processing: Ex-plorations in the Microstructure of Cognition*, vol. 1, pp 533-536 , 1986.
[35] M.LGinsberg, " GIB: Steps toward an expert-level bridge-playing program", *In: International Joint Conference on Artificial Intelligence* (IJCAI 1999), Stockholm, Sweden, pp. 584–589, 1999.
[36] Jacek Mandziuk, "Incremental learning approach for board game playing agents*",In: Proceedings of the 2000 International Conference on Artificial Intelligence (ICAI 2000),* Las Vegas, vol. 2, pp. 705–711, 2000.
[37] Jacek Mandziuk, "Incremental training in game playing domain", *In: Proceedings of the International ICSC Congress on Intelligent Systems & Applications* Wollongong, Australia, vol. 2, pp. 18–23, 2000.
[38] Jacek Mandziuk, "Some thoughts on using Computational Intelligence methods in classical mind board games", *In: Procedings of the 2008 International Joint Conference on Neural Networks (IJCNN 2008),* Hong Kong, China, pp. 4001–4007, 2008.
[39] Jacek Mandziuk and Krzysztof Mossakowski, "Looking InsideNeuralNetworks Trained to Solve Double-Dummy Bridge Problems," *Int.Proceeding 5th Game – On Computer Games: Artif. Intell.,* U.K. pp. 182-186, 2004.
[40] M. Dharmalingam and R.Amalraj, "Neural Network Architectures for Solving the Double Dummy Bridge Problem in Contract Bridge", *in Proce of the PSG-ACM National Conference on Intelligent Computing,* pp 31-37, 2013.

BIOGRAPHIES

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 Artificial Neural Networks.

Dr. R. Amalraj is an Associated Professor in Computer Science in the Department of Computer Science, Sri Vasavi College, Erode. He obtained his Ph.D 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.