Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2007)

Example-based Estimation of Hand's Strength in the Game of Bridge with or without Using Explicit Human Knowledge

Jacek Mańdziuk and Krzysztof Mossakowski
Faculty of Mathematics and Information Science
Warsaw University of Technology
Plac Politechniki 1, 00-661 Warsaw
POLAND
Emails: {mandziuk, mossakow}@mini.pw.edu.pl

Abstract— **The paper presents results of experiments of estimating the number of tricks to be taken by one pair of bridge players in so-called Double Dummy Bridge Problem, using artificial neural networks. In addition to deals presented to neural network's inputs, also some human methods of estimating strength of a hand were applied. Influence of human knowledge on neural network's results depends on the way of coding a deal.**

One of deal representations tested in the paper definitely outperformed all the other choices even when the remaining representations were additionally supported by human knowledge implemented by several estimators frequently used in professional play. This superior representation output a perfect answer in 53.11% of test deals and only in 3.52% of them was mistaken by more than one trick.

I. Introduction

The game of bridge is one of the best known card games, and, as such worldwide game, has attracted attention of several Artificial Intelligence and Computational Intelligence researchers, e.g. [1], [2], [3], [4], [5]. There are many interesting aspects of the game of bridge, one of them is estimation of hand's strength.

In this paper description and results of experiments of using artificial neural networks as estimators of the number of tricks to be taken by one pair of players in a deal are presented. In some of the experiments human methods of estimating hand's strength are also used.

The underlying issue considered in the paper is analysis of the problem of data representation in neural network learning and exploration of the network's capabilities of discovering the rules and nuances in efficient hand's strength estimation based solely on examples. The results of example-based learning are compared with the situation in which some additional expert knowledge, besides example deals, is added in the input layer. It is experimentally proven that with adequate choice of network's architecture and data representation in the input layer the neural net trained without explicit human knowledge is as efficient as the one that observes additional input data representing well known human estimators of hands' strength.

The paper is organized as follows. The data used in experiments and some human methods of estimating hand's strength are described in section II. Section III contains information about artificial neural networks' architectures used in experiments and description of the ways of coding a deal in the network's input layer. Section IV presents results of experiments with emphasis put on comparison between no trump and suit contracts as well as analysis of an influence of changing the hand which makes defender's lead on the quality of results. Section V discusses 4 sample deals, that illustrate capabilities of trained neural networks and reveal some interesting nuances of the play. Conclusions are placed in the last section.

It is worth to underline that although the focus of this paper is not on playing the game of bridge *per se*, the problem of effective hand's strength estimation is essential in the development of strong bridge playing agent. This issue is further discussed in the next section.

II. Problem Definition

Estimating hand's strength is a crucial aspect of the bidding phase of the game of bridge, since contract bridge is a game with incomplete information and during the bidding phase each player can see only his/her cards and has to make several assumptions about placement of other cards[1]. This incompleteness of information forces 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.

Assuming any particular variant of cards' location is equivalent to the case of having all four hands revealed. Under these circumstances the question to be answered is "How many tricks are to be taken by one pair of players assuming perfect play of all four sides?". The above situation is called Double Dummy Bridge Problem (DDBP) [5], [9].

There is an important difference between solving DDBP and the real bridge playing, since in the latter the exact placement

[1]For the sake of clarity and due to space limits the introduction to the game of bridge is omitted in the paper. The interested reader can learn the basic rules of the game from anyone of numerous sources available online or in printed form, e.g. [6], [7] or [8].

TABLE I
HUMAN POINT COUNT METHODS

Method	A	K	Q	J	10
Work Point Count	4	3	2	1	0
Bamberger Point Count	7	5	3	1	0
Collet Point Count	4	3	2	0.5	0.5
Four Aces points	3	2	1	0.5	0
Polish points	7	4	3	0	0
Reith Point Count	6	4	3	2	1
Robertson Point Count	7	5	3	2	1
Vernes Point Count	4	3.1	1.9	0.9	0
AKQ points	4	3	2	0	0

of most of the cards is unknown. Consequently, in a real play, the player has to calculate probabilities of cards' distributions and choose a strategy with the highest expected outcome. In DDBP there is no hidden data and the best strategy can be pointed out.

A. The GIB Library

The data used in solving DDBP was taken from the GIB Library [10]. This library was created by Ginsberg's Intelligent Bridgeplayer [5] computer program, which is considered to be one of the best bridge playing programs.

The GIB Library includes $717,102$ deals with all hands revealed. Additionally, for each deal the library provides the numbers of tricks to be taken by the pair NS for each combination of the trump suit (including no trump contract) and the hand which makes defender's lead. Together there are 20 numbers for each deal (5 trump suits by 4 sides). All these numbers were calculated by GIB program under assumption of a perfect play of all players.

In most experiments reported in this paper, $100,000$ deals from the library (with numbers from 1 to $100,000$) were used during training and another $100,000$ ones (numbered from $600,001$ to $700,000$) were used for testing.

The tested ways of coding a deal in numeric format suitable for artificial neural networks are described in section III-A.

B. Human Methods of Estimation of Hand's Strength

Three groups of experiments were carried out. In the first group only example deals (appropriately coded as neural networks' inputs) were used. In the second group inputs from human hand's strength estimators were added. Finally, for comparison purposes, in the third group of tests only human estimators were used in the neural network training without accompanying presentation of a deal.

Human estimators of hand's strength can be divided into two categories. The first category contains methods of calculating the strength of a hand as a sum of single card strengths [11], [12]. In these methods the value of each card depends only on a rank of the card. The most widely used points counting system is called Work Point Count, which scores 4 points for an *Ace*, 3 points for a *King*, 2 points for a *Queen*, and 1 point for a *Jack*. Table I presents other popular human point count methods, which were also used in experiments described in this paper.

The second category contains so-called distributional points [11], [12]. These methods score patterns which can be found in a set of cards assigned to one hand. The most important patterns are: suits' lengths and groups of figures in one suit. Even novice bridge players know, that a void (lack of cards in a suit) or a singleton (single card in a suit) are very valuable in suit contracts, so it is not surprising that almost all distributional points methods award such shortness. Results obtained in experiments described in the paper confirm that suits' lengths are crucial for suit contracts.

Another very important pattern which is searched in cards of both players from a pair, is a group of honours (i.e. figures and a Ten) in one suit. Having a group of top honours in a suit allows to predict more precisely the number of tricks available in this suit.

Human distributional points methods used in our experiments are listed in Table II. Most of them join Work Point Count scoring with some rewarding for short or long suits.

Zar Points [13] is another method of estimating hand's strength, which combines elements of point count methods with ideas of distributional points. In this method, each hand is scored by adding the following figures:

- points for honours: 6 points for each *Ace*, 4 points for a *King*, 2 points for a *Queen*, and 1 points for a *Jack*,
- the difference between the lengths of the longest and the shortest suits,
- the sum of the lengths of the longest 2 suits.

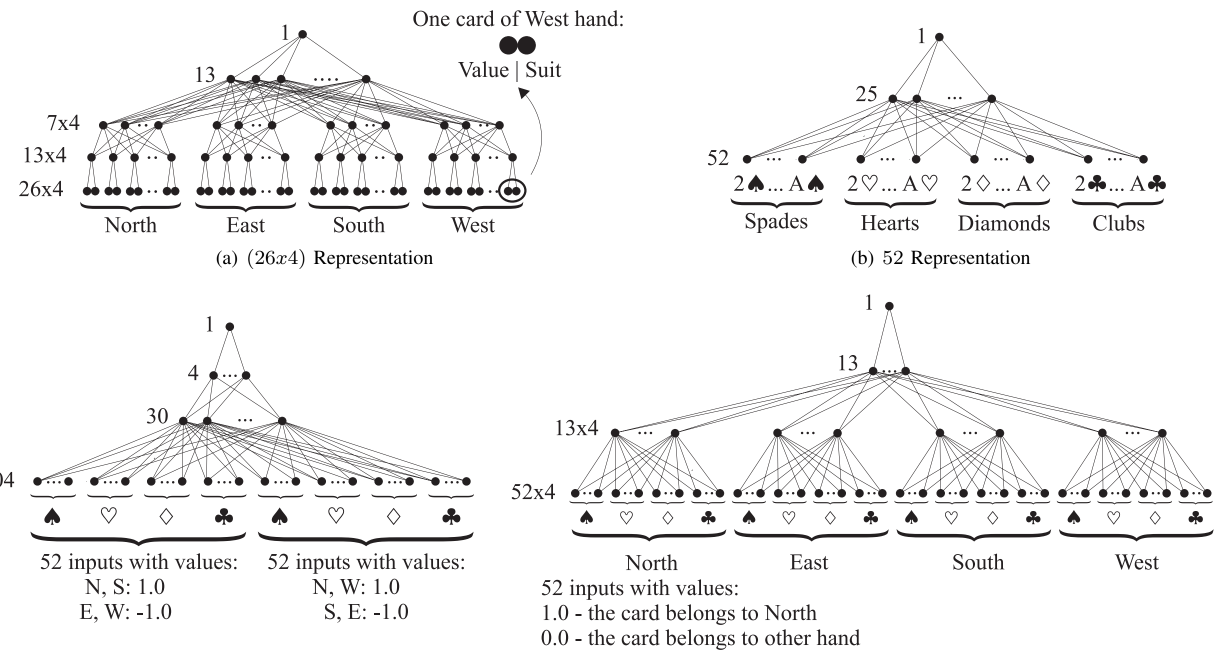
III. EXPERIMENT DESIGN

Artificial neural networks are very suitable tools for estimating the number of tricks, due to their ability to generalize knowledge acquired from training data. Another very important feature of neural nets, when considered as an estimation tool during bidding phase of the game of bridge, is their speed. Calculating the expected number of tricks to be taken using trained network is very fast, and can be efficiently carried out for many potential variants of a deal, which have to be considered due to partly hidden information.

In all experiments feed-forward networks were used. Training and testing was carried out with JNNS's [14] assistance. JNNS, Java Neural Network Simulator, is a freely available successor to Stuttgart Neural Network Simulator (SNNS).

In most cases logistic (unipolar sigmoid) activation function was used for all neurons. Only when negative values were presented in the input layer, the hyperbolic tangent (bipolar sigmoid) activation was applied.

All networks were trained using Rprop (Resilient Backpropagation) algorithm [15], with the following choice of method's parameters: initial and maximum values of an update-value factor were equal to 0.1 and 50.0, resp., and weight decay parameter was equal to $1E-4$.



(a) $(26x4)$ Representation

(b) 52 Representation

(c) 104 Representation

(d) $(52x4)$ Representation

Fig. 1. Representations of Deals

TABLE II
HUMAN DISTRIBUTIONAL POINTS METHODS

Method	Short Description
Honour Trick	Potential number of tricks to be taken by honours in one suit (e.g. 2 points for *AK*, 1.5 for *AQ*, etc.)
Playing Trick	Similar to Honour Trick, with additional bonuses for short and long suits.
Losing Trick Count	Potential number of tricks to loose in one suit (e.g 1 point for *Ax*, 2 points for *Qx*).
Asset System	Work Point Count's enhancement with bonuses for short ($+2$ for a void, $+1$ for a singleton) and long ($+1$ for 5 or more cards) suits.
Stayman Point Count	Work Point Count method with rewarding short and long suits and lowering the status of single honours in suits.
Rule of three and four	Work Point Count method with additional $+1$ point for 5th, 6th, etc. card in a trump suit and for 4th, 5th, etc. card in any other suit.
Moins-value	Modification of Work Point Count: -1 for no *Aces*, -0.5 for no *Tens*, -1 when there are too few cards in a suit to make honour(s) potential trick(s), and -0.5 when there are less than 3 honours in a suit, or only one of cards: *Ace*, *King*, and *Queen*.
Plus-value	Modification of Work Point Count: $+0.25$ for each *Ace*, $+0.5$ for a *Ten* with a honour or *Nine*, and $+0.5$ when there are 3 honours or at least two of cards: *Ace*, *King*, and *Queen* in a suit.

A. Representation of a Deal

The number of input neurons depended on the way a deal was represented. Four ways of coding a deal suitable for neural networks learning were tested. One of them (denoted $52x4$) achieved significantly better results than the others. The effectiveness of the remaining representations was on comparable level (see section IV for details). Three ways of coding a deal required similar numbers of training epochs, whereas the fourth coding $(26x4)$ required much longer training - over $100,000$ epochs, compared to a few thousand iterations needed by the other representations.

1) $(26x4)$: The first tested way of deal's representation used 104 input values grouped in 52 pairs. Each pair represented one card. The first input value in a given pair determined the rank of the card (*Ace*, *King*, *Queen* etc.) and the second one represented the suit of the card (*Spades*, *Hearts*, *Diamonds* or *Clubs*). Hence, 26 input neurons (13 pairs) were necessary to fully describe the content of one hand (see Fig. 1(a)).

A few ways of transforming card's rank and suit into real numbers suitable as input values for the network were tested. Finally the rank of the card was transformed using a uniform linear transformation to the range $[0.1, 0.9]$, with biggest values for *Aces* (0.9), *Kings* (0.83) and smallest for *Three spots* (0.17) and *Two spots* (0.1). Some other ranges, e.g. $[0, 1]$ or $[0.2, 0.8]$, were also tested, but no significant difference in results was noticed. Suit of the card was also coded as a real number, usually by the following mapping: 0.3 for *Spades*, 0.5 for *Hearts*, 0.7 for *Diamonds*, and 0.9 for *Clubs*.

TABLE III

COMPARISON OF RESULTS OBTAINED FOR NO TRUMP AND SUIT CONTRACTS

The Network	Inputs	Results for No Trump Contracts	Results for Spades Contracts
(26x4)-(13x4)-(7x4)-13-1	Deals in $(26x4)$ representation	93.87 | 75.70 | 31.04	97.67 | 84.24 | 36.82
52-25-1	Deals in 52 representation (with input values: 1.0 for N, 0.8 for S, -1.0 for W, -0.8 for E)	95.81 | 79.95 | 34.02	98.47 | 86.83 | 39.16
52-25-1	Deals in 52 representation (with input values: 1.0 for N, 0.8 for S, -1.0 for W and E)	95.97 | 80.46 | 34.35	98.52 | 87.09 | 39.47
52-25-1	Deals in 52 representation (with input values: 1.0 for N and S, -1.0 for W and E)	96.07 | 80.88 | 34.66	98.77 | 88.00 | 40.13
104-30-4-1	Deals in 104 representation	95.64 | 79.63 | 33.74	98.61 | 87.17 | 39.21
(52x4)-(13x4)-13-1	Deals in $(52x4)$ representation	97.34 | 84.31 | 37.80	99.78 | 95.00 | 50.03
1-1	Sum of Work Point Count values for the NS pair of players	93.73 | 76.41 | 31.37	76.22 | 49.55 | 16.93
4-1	Work Point Count values for each hand	93.73 | 76.34 | 31.31	76.22 | 49.64 | 16.91
20-1	Work Point Count values for hands (4 inputs) and suit lengths (16 inputs)	93.73 | 76.34 | 31.32	97.00 | 82.21 | 35.29
20-10-5-1	Work Point Count values for hands (4 inputs) and suit lengths (16 inputs)	94.24 | 77.78 | 32.78	98.75 | 88.21 | 40.30
36-25-1	9 human point count methods, 4 inputs per method (see Table I)	94.87 | 78.30 | 32.39	76.83 | 49.77 | 16.75
(52+36)-25-1	Deals in 52 representation and 9 point count human methods	96.33 | 81.39 | 35.01	98.72 | 87.90 | 40.07
32-25-1	8 human distributional points methods, 4 inputs per method (see Table II)	94.94 | 77.71 | 32.50	98.56 | 88.07 | 39.94
(52+32)-25-1	Deals in 52 representation and 8 distributional points human methods	96.86 | 83.02 | 36.67	99.68 | 94.28 | 48.63
68-25-1	9 point count methods and 8 distributional points methods (4 inputs for each method)	96.03 | 81.34 | 35.41	98.78 | 89.24 | 41.50
(52+102)-77-38-19-1	Deals in 52 representation, 9 point count human methods, and 8 distributional points human methods (6 inputs per method - 4 for hands and 2 for pairs of players)	96.06 | 81.21 | 35.15	99.49 | 92.22 | 45.33

In order to allow the network gather full information about cards' placements, some special groups of neurons were created in subsequent layers. For example, the network $(26x4)-(13x4)-(7x4)-13-1$ was composed of 5 layers of neurons arranged in a way depicted in Fig. 1(a). The first hidden layer was responsible for collecting information about individual cards. Four groups of neurons in the second hidden layer gathered information about respective hands. The last hidden layer combined the whole information about a deal and was connected to the output neuron.

2) 52: The second way of coding a deal implemented different point of view - input values were not assigned to hands, but to cards from the deck. 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: *Two of Spades*, *Three of Spades*, ..., *King of Spades*, *Ace of Spades*, *Two of Hearts*, ..., *Ace of Hearts*, *Two of Diamonds*, ..., *Ace of Diamonds*, *Two of Clubs*, ..., *Ace of Clubs* (see Fig. 1(b)).

A value presented in the input neuron denoted the hand to which a given card belonged, i.e. 1.0 for *North*, 0.8 for *South*, -1.0 for *West*, and -0.8 for *East*. Interestingly, as came out from further experiments, using the same input value (-1.0) for *West* and *East* hands improved the results. Additionally, hiding the information about exact cards' assignment in the pair *NS*, i.e. using an input value equal to 1.0 for both *North* and *South* hands, yielded another slight improvement (see Table III).

In this coding there were no dedicated groups of neurons in hidden layers. Layers were fully connected, e.g. in $52-25-1$ network all 52 input neurons where connected to all 25 hidden neurons, and all hidden neurons were connected to 1 output neuron.

3) 104: The next way of coding a deal was an extension of the previous one, with 104 inputs. The first 52 input values represented assignment to a pair (value 1.0 represented NS and -1.0 - WE), and the other 52 ones exactly pointed out the hand (value 1.0 for *North* or *West* and -1.0 for *South* or *East*). In both groups positions of cards were fixed in the same way as in coding 52 (see Fig. 1(c)).

Networks using this coding were fully connected, and usually contained two layers of hidden neurons, e.g. $104-30-4-1$.

4) $(52x4)$: The last way of a deal coding used the biggest number of inputs - 208. Input values were divided into 4 groups, one group per hand. The first group represented cards of the *North* player, the second - the *East* one, the third - the *South* one, and the fourth - the *West* player. In this representation 4 input neurons were assigned to each card from a deck. Exactly one of these four input neurons was equal to 1.0, three remaining ones received value 0.0 as the input. This way, a hand to which the card was assigned in a deal was unambiguously defined.

There were 4 groups of neurons in the first hidden layer, each of them gathering information from 52 input neurons

TABLE IV

COMPARISON OF RESULTS OBTAINED FOR SUIT CONTRACTS WITH AND WITHOUT CHANGING DEFENDER'S LEAD

The Network	Inputs	Results for Spades Contracts	Results for Spades Contracts with changing defender's lead
(26x4)-(13x4)-(7x4)-13-1	Deals in $(26x4)$ representation	97.67 | 84.24 | 36.82	98.76 | 88.00 | 39.90
52-25-1	Deals in 52 representation	98.77 | 88.00 | 40.13	98.49 | 87.15 | 39.29
104-30-4-1	Deals in 104 representation	98.61 | 87.17 | 39.21	99.09 | 89.79 | 41.92
(52x4)-(13x4)-13-1	Deals in $(52x4)$ representation	99.78 | 95.00 | 50.03	99.79 | 95.49 | 50.62
(52x4)-(26x4)-26-13-1	Deals in $(52x4)$ representation	99.80 | 95.54 | 50.91	99.88 | 96.48 | 53.11
(104+68)-50-10-1	Deals in 104 representation, 9 point count human methods, and 8 distributional points human methods (4 inputs per method)	99.35 | 92.25 | 45.34	99.46 | 92.40 | 45.54
(52x4+84)-(13x4+21)-26-1	Deals in $(52x4)$ representation, 9 point count human methods, 8 distributional points human methods (4 inputs per method), and lengths of all suits from all hands (16 inputs)	99.82 | 95.74 | 51.40	99.84 | 96.12 | 52.47
20-10-5-1	Work Point Count values for hands (4 inputs) and suit lengths (16 inputs)	98.75 | 88.21 | 40.30	98.67 | 87.98 | 40.62

representing one hand. This data was further compressed in another one or two fully connected hidden layers. All neurons from the last hidden layer were connected to a single output neuron. A sample network using this way of coding a deal, $(52x4) - (13x4) - 13 - 1$, is presented in Fig. 1(d). The 3-hidden layer architecture is, for example, represented by the network $(52x4) - (26x4) - 26 - 13 - 1$.

B. Representation of Human Estimators

Values of human estimators of hand's strength (see section II-B) were coded as real numbers from the range [0.1, 0.9]. For each estimator minimum and maximum possible values were determined, and evaluator's value was linearly mapped to the destination range. Usually 4 input neurons were assigned to each estimator, one per hand. In some experiments two additional input neurons were assigned to an estimator with values calculated as sums of estimator's values for pairs of players (NS and WE).

In case of 52 and 104 codings, input neurons representing human estimators were connected to the first hidden layer exactly in the same way as all other input neurons. In this case the first hidden layer collected all information about a deal and estimators' values. In $(52x4)$ representation there existed a special group of neurons in the first hidden layer, devoted to processing this additional input information. For example, in $(52x4 + 84) - (13x4 + 21) - 26 - 1$ network there were 21 neurons belonging to this group. These neurons received values exclusively from input neurons assigned to human estimators (there were 84 such neurons in that case - see Table IV for details). In the second hidden layer all information was mixed together.

IV. RESULTS

In our previous papers the DDBP was examined in the context of NT contracts and two deal codings: 52 and 104 [16], [17]. Despite promising numerical results, the focus of previous research was on exploration and analysis of weight patterns of the networks, after the training process was completed. Several interesting patterns in weights' structures were discovered, including the comparatively big weights' values from inputs representing *aces* and *kings* or detection of the neurons devoted to representation of particular suits (the weights from one suit cards were visibly greater that those from the remaining suits). Also more subtle observations were carried out, e.g. weight patters suitable for *the finesses* - a very important aspect of the game of bridge (see [17] for details).

Our recent paper concerning DDBP [18] was concentrated on suit contracts, where roughly the same weight structures as for NT contracts were discovered, with the choice of deal codings still restricted to 52 and 104 ones.

The underlying research problems considered in this paper include: (a) tests of novel deal codings and more elaborate network architectures, (b) comparison of training *with* and *without* adding human expert knowledge in the input data, (c) comparison between no trump and suit contracts and (d) further enhancement of results by training on examples with changing a defender lead's hand.

Numerical results of experiments are summarized in two tables. Table III compares results achieved for no trump and suit contracts. In Table IV an influence of changing a hand which makes defender's lead on results is presented.

Each result consists of three values, e.g. 93.87 | 75.70 | 31.04. These values denote the fraction *in percent* of test deals for which the network was mistaken, respectively by no more than 2 tricks (93.87), no more than 1 trick (75.70), and was perfectly right (31.04).

A. Comparing Results Obtained for No Trump and Suit Contracts

Analysis of experimental results for no trump contracts reveals, that the difference between the best (and also the most complicated) solution and a simple linear combination of 4 values representing Work Point Count values for each hand, was surprisingly small. The best result achieved by $(52x4) - (13x4) - 13 - 1$ network was 97.34 | 84.31 | 37.80, and a simple mapping of a sum of Work Point Count values for the pair NS, i.e. the network with one input and one

	♠ K J 6 5 4 ♥ 5 ♦ 5 ♣ K 10 9 8 4 2	
♠ 10 9 ♥ A 6 4 2 ♦ 10 2 ♣ Q 7 6 5 3	N W E S	♠ A 8 ♥ K Q 10 8 3 ♦ K Q 8 6 ♣ A J
	♠ Q 7 3 2 ♥ J 9 7 ♦ A J 9 7 4 3 ♣	

Correct number of tricks	10
Networks' estimations	
$(52x4) - (13x4) - 13 - 1$	10
$(52x4) - (26x4) - 26 - 13 - 1$	10
$104 - 30 - 4 - 1$	8
$52 - 25 - 1$	7
$(104 + 68) - 50 - 10 - 1$	10
$(52x4 + 84) - (13x4 + 21) - 26 - 1$	10

Fig. 2. Sample deal and networks' estimation of a number of tricks to be taken by the NS pair in $Spades$ contract with $West$ defender's lead.

	♠ K 9 ♥ A 9 5 3 2 ♦ 7 ♣ A 7 4 3 2	
♠ Q 10 3 2 ♥ 6 4 ♦ A K J 6 5 4 ♣ 9	N W E S	♠ A J 7 4 ♥ 10 8 7 ♦ 10 9 8 3 2 ♣ 6
	♠ 8 6 5 ♥ K Q J ♦ Q ♣ K Q J 10 8 5	

Correct number of tricks	3
Networks' estimations	
$(52x4) - (13x4) - 13 - 1$	5
$(52x4) - (26x4) - 26 - 13 - 1$	3
$104 - 30 - 4 - 1$	5
$52 - 25 - 1$	5
$(104 + 68) - 50 - 10 - 1$	5
$(52x4 + 84) - (13x4 + 21) - 26 - 1$	4

Fig. 3. Sample deal and networks' estimation of a number of tricks to be taken by the NS pair in $Spades$ contract with $North$ defender's lead.

output neuron, without any hidden neurons, obtained result 93.73 | 76.41 | 31.37.

On the one hand these results suggest, that no trump contracts are very easy - it is enough to compute a sum of weights points for honours to get almost one third chance of a perfect estimation of a number of tricks. On the other hand no trump contracts seem to be more demanding than suit contracts. Even quite complicated networks, with human knowledge applied, were not able to achieve the level of 85% accuracy with 1 trick tolerance.

Preliminary results achieved by $52 - 25 - 1$ network for single suits proved, that there was no significant difference between suits (98.77 | 88.00 | 40.13 for $Spades$, 98.65 | 87.81 | 40.18 for $Hearts$, 98.66 | 87.68 | 39.96 for $Diamonds$, and 98.73 | 87.90 | 40.02 for $Clubs$). Some experiments with all suit contracts to be managed by one network were also performed, and, not surprisingly, achieved results were similar (e.g. 98.68 | 87.88 | 40.11 by $52 - 25 - 1$ network).

Therefore, in order to fix attention, among all suit contracts $Spades$ contracts were chosen for further tests. For these contracts, using only Work Point Count values attained much worse results (76.22 | 49.64 | 16.91) than for no trump contracts (93.73 | 76.34 | 31.31). However, adding 16 input values representing lengths of all suits from all hands allowed to improve results to quite a good level - 98.75 | 88.21 | 40.30 by $20 - 10 - 5 - 1$ network. None of the following ways of representing a deal: $(26x4)$, 52, and 104, even with quite complicated architectures, was able to beat this result. This fact may undermine other neural networks achievements, but it should be emphasized that there is a lot of human knowledge of the game of bridge in such choice of inputs (Work Point Count values and lengths of suits). The network $68 - 25 - 1$, which used all human point count and distributional points methods (see Tables I and II), achieved only slightly better result - 98.78 | 89.24 | 41.50.

It is interesting to compare results achieved by networks $(52+36)-25-1$ and $(52+32)-25-1$. Both these networks used 52 way of coding a deal with additional inputs from human methods of estimating hand's strength. Results of these networks for no trump contracts were comparable, but for $Spades$ contracts using distributional points methods turned out to be definitely more effective (99.68 | 94.28 | 48.63) than using point count methods (98.72 | 87.90 | 40.07).

For both no trump and suit contracts the best results were achieved by the networks using $(52x4)$ deal coding.

B. Changing Defender Lead's Hand

For Spades contracts, in 7% of deals contained in the GIB Library, the number of tricks to be taken by the pair NS depends on the hand which makes defender's lead, i.e. the number of tricks after defender's lead from $West$ side differs from the respective number after $East$ defender's lead. Enlarging both training and testing sets by duplicating deals and changing positions of hands (to make sure, that always the same set of input neurons represents a hand which make defender's lead), improved results, but the benefit was smaller than 7%.

Table IV contains comparison of results achieved for $Spades$ contracts with and without changing defender's lead. Exactly the same deals were used in training and testing phases, but when the hand which makes defender's lead was

```
                 ♠ J 7
                 ♥ Q J 8 4
                 ♦ K 8 4
                 ♣ J 10 6 5

♠ A 10 9 8 4       N          ♠ 6 5 3 2
♥ 9 2            W   E        ♥ K 6 5 3
♦ 9 6 5 2          S          ♦ A 7
♣ A K                         ♣ Q 3 2

                 ♠ K Q
                 ♥ A 10 7
                 ♦ Q J 10 3
                 ♣ 9 8 7 4
```

	Defender's lead	
	$North$	$South$
Correct number of tricks	4	3
Networks' estimations		
$(52x4) - (13x4) - 13 - 1$	3	4
$(52x4) - (26x4) - 26 - 13 - 1$	4	3
$104 - 30 - 4 - 1$	4	4
$52 - 25 - 1$	4	4
$(104 + 68) - 50 - 10 - 1$	3	4
$(52x4 + 84) - (13x4 + 21) - 26 - 1$	3	4

Fig. 4. Sample deal and networks' estimation of a number of tricks to be taken by the NS pair in $Spades$ contract.

```
                 ♠ Q 9 6
                 ♥ K Q 6 4
                 ♦ 4 3 2
                 ♣ Q 7 4

♠ 5                N          ♠ A K J 10 2
♥ A 3            W   E        ♥ 7 5
♦ A Q J 9 7 5      S          ♦ K 10 8
♣ A 10 8 5                    ♣ 9 3 2

                 ♠ 8 7 4 3
                 ♥ J 10 9 8 2
                 ♦ 6
                 ♣ K J 6
```

Correct number of tricks	0
Networks' estimations	
$(52x4) - (13x4) - 13 - 1$	4
$(52x4) - (26x4) - 26 - 13 - 1$	3
$104 - 30 - 4 - 1$	3
$52 - 25 - 1$	4
$(104 + 68) - 50 - 10 - 1$	3
$(52x4 + 84) - (13x4 + 21) - 26 - 1$	4

Fig. 5. Sample deal and networks' estimation of a number of tricks to be taken by the NS pair in $Spades$ contract with $North$ defender's lead.

changed, all deals were duplicated, so sizes of training and testing sets doubled.

V. Sample Deals

In this section four examples of deals from the test set were chosen in order to illustrate the strong and weak points of deal codings under consideration.

The first sample deal, presented in Fig. 2, shows the advantage of $(52x4)$ network over the networks using 52 and 104 codings. In this deal the NS pair is able to take 10 tricks when playing $Spades$ contract. However, 52 and 104 networks estimated only 7 and 8 tricks, respectively. Both tested networks using $(52x4)$ codings (i.e. $(52x4)-(13x4)-13-1$ and $(52x4)-(26x4)-26-13-1$) were perfectly right. Analysis of this deal shows, that the NS pair of players has only 15 points (Work Point Count) together. There is a void in $Clubs$ suit on $South$ hand and two singletons in $Hearts$ and $Diamonds$ on $North$ hand. These short suits extremely strengthen the pair NS and enable to hold 10 tricks. Adding human knowledge of the game to the network's input by using 9 point count and 8 distributional points methods, and enlarging 104 network's size to $(104+68)-50-10-1$, allows to estimate the correct number of tricks.

In the second deal, presented in Fig. 3, the pair WE can hold 10 tricks in $Spades$ contract, so the pair NS is able to hold only 3 tricks. However, the power of the NS pair promises more. 25 points (Work Point Count) and 2 singletons (unfortunately, in the same suit - $Diamonds$) misled most of the tested neural networks to overestimate the number of tricks for NS. Most of networks claimed 5 tricks, so they were wrong by 2 tricks. The network $(52x4+84)-(13x4+21)-26-1$ overestimate 1 trick, and only $(52x4)-(26x4)-26-13-1$ network answered perfectly.

In the third example (presented in Fig. 4) the number of tricks to be taken by a pair NS depends on which hand makes defender's lead. Only if $North$ makes defender's lead, NS pair is able to hold 4 tricks (1 in $Spades$, 2 in $Hearts$, and 1 in $Diamonds$). When $South$ makes defender's lead, the pair NS can hold only 1 trick in $Hearts$, so 3 in total. $(52x4)-(26x4)-26-13-1$ was the only network which properly estimated the number of tricks in both cases, the rest of networks were wrong either in one case ($104-30-4-1$ and $52-25-1$) or in both cases.

The last presented deal (see Fig. 5) is one of deals for which all neural networks, regardless of applied way of coding, made the biggest error. Perfectly fitted WE hands are able to hold all tricks (the grand slam), thanks to very favorable distribution of $Spades$ suit in NS hands. On the other hand, the strength of the NS pair is noticeable - 14 points (Work Point Count), 7 trumps and a singleton in $Diamonds$ suit. Under these circumstances the networks' estimations seem to be justified (3 tricks by $(52x4)-(26x4)-26-13-1$, $104-30-4-1$ and $(104+68)-50-10-1$ and 4 tricks by $(52x4)-(13x4)-13-1$, $52-25-1$, and $(52x4+84)-(13x4+21)-26-1$).

VI. Conclusions

The first conclusion which can be drawn from results presented in this paper, is the difference between no trump and suit contracts. Results achieved by neural networks for $Spades$ contracts are far better than results for no trump contracts. The Work Point Count method of estimating hand's strength is crucial for no trump contracts and allows to achieve almost one third chance of correct estimation without any additional information. The Work Point Count is also important for suit contracts, however in this case additional information about lengths of suits must be applied to achieve equally good results.

Adding human knowledge to networks' inputs improves significantly results achieved by 52 and 104 networks. It suggests that networks using the above deal representations are not able to extract this relevant information by themselves. On the contrary, in case of $(52x4)$ networks, no such improvement was observed, which suggests that this way of deal representation allows efficient extraction of human knowledge from raw data. The superiority of $(52x4) - (26x4) - 26 - 13 - 1$ architecture was proven by numerical results and illustrated by a few sample deals presented in section V. Analysis of these deals (and also several other examples, not reported in the paper) suggests, that in case of $52x4$ representation the network (if only large enough) is able to autonomously discover relevant information about hands' points and lengths of suits, as well as appropriately weight this information in the training process leading to highly efficient estimations of possible number of tricks.

Generally speaking, artificial neural networks turned out to be very effective tools for estimating the number of tricks to be taken by one pair of players in Double Dummy Bridge Problem. In several cases it is quite difficult, also for experienced human bridge players, to perfectly answer the question about the number of tricks to be taken by a playing pair, even with all cards revealed. The most efficient neural network $((52x4) - (26x4) - 26 - 13 - 1)$ trained exclusively on examples of deals, without any explicit human knowledge or awareness of nuances of the play (e.g. finesses), and even with no information about the rules of the game, achieved respectful result: it was perfectly right in 53.11% of test deals and mistaken by more than one tricks in only 3.52% out of $100,000$ test cases.

References

[1] M. Kohle and F. Schonbauer, "Experience gained with a neural network that learns to play bridge," in *5th Austrian Artificial Intelligence Meeting*, 1995, pp. 224–229.

[2] 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, 1995, pp. 144–151.

[3] S. Smith, D. Nau, and T. Throop, "Computer bridge: A big win for AI planning." *AI Magazine*, vol. 19, pp. 93–106, 1998.

[4] I. Frank and D. A. Basin, "Search in games with incomplete information: A case study using bridge card play," *Artificial Intelligence*, vol. 100, no. 1-2, pp. 87–123, 1998. [Online]. Available: citeseer.ist.psu.edu/frank98search.html

[5] M. Ginsberg, "GIB: Imperfect information in a computationally challenging game." *Journal of Artificial Intelligence Research*, vol. 14, pp. 303–358, 2001.

[6] Wikipedia. Contract Bridge. [Online]. Available: http://en.wikipedia.org/wiki/Contract_bridge

[7] A. Sheinwold, *Five weeks to winning Bridge*, 1989.

[8] J. Pottage, *The Bridge player's bible: illustrated strategies for staying ahead of the game*, 2006.

[9] R. Wheen, "Brute force programming for solving double dummy bridge problems," in *Heuristic Programming in Artificial Intelligence: the first computer olympiad*, D. Levy and D. Beal, Eds. Ellis Horwood, Chichester, 1989, pp. 88–94.

[10] M. Ginsberg. GIB library. [Online]. Available: http://www.cirl.uoregon.edu/ginsberg/gibresearch.html

[11] H. Francis, A. Truscott, and D. Francis, Eds., *The Official Encyclopedia of Bridge*, 5th ed. Memphis, TN: American Contract Bridge League Inc, 1994.

[12] B. Seifert, Ed., *Encyclopedia of Bridge*. Warsaw: Polish Scientific Publishers PWN, 1996, (in Polish).

[13] Z. Petkov. [Online]. Available: http://www.zarpoints.com

[14] Java neural network simulator. [Online]. Available: http://www-ra.informatik.uni-tuebingen.de/software/JavaNNS/welcome_e.html

[15] M. Riedmiller and H. Braun, "A fast adaptive learning algorithm," University Karslruhe, Germany, Tech. Rep., 1992.

[16] K. Mossakowski and J. Mańdziuk, "Artificial neural networks for solving double dummy bridge problems," in *Artificial Intelligence and Soft Computing - ICAISC 2004*, ser. Lecture Notes in Artificial Intelligence, L. Rutkowski, J. H. Siekmann, R. Tadeusiewicz, and L. A. Zadeh, Eds., vol. 3070. Springer, 2004, pp. 915–921.

[17] J. Mańdziuk and K. Mossakowski, "Looking inside neural networks trained to solve double-dummy bridge problems," in *5th Game-On International Conf. on Computer Games: Artificial Intelligence, Design and Education (CGAIDE 2004)*, Reading, UK, 2004, pp. 182–186.

[18] K. Mossakowski and J. Mańdziuk, "Neural networks and the estimation of hands strength in contract bridge," in *Artificial Intelligence and Soft Computing ICAISC 2006*, ser. Lecture Notes in Artificial Intelligence, L. Rutkowski *et al.*, Eds., vol. 4029. Springer, 2006, pp. 1189–1198.