Weight Patterns in the Networks Trained to Solve Double Dummy Bridge Problem

Krzysztof Mossakowski
Jacek Mańdziuk

Faculty of Mathematics and Information Science,
Warsaw University of Technology,
Plac Politechniki 1, 00-661 Warsaw, Poland
E-mail: {mossakow, mandziuk}@mini.pw.edu.pl

Abstract

This paper presents analysis of connection weights of artificial neural networks trained to solve double dummy bridge problems. The networks were trained using only sample deals, no human knowledge and no rules of the game of bridge were presented. However analysis of connection weights reveals some patterns explainable based on human knowledge of the game.

1 Introduction

The general goal of this research is to create a bridge playing program. The main assumption is to avoid presentation of human knowledge of the game of bridge in any form. In all experiments described here only deals and target number of tricks to be taken by one pair of players were presented. There was no human knowledge of the game and also no rules of the game were presented during training.

The goal of experiments presented in this paper was to verify neural networks' ability to estimate the number of tricks that can be taken by one pair of players in a deal, in assumption of optimal play of all players, when all hands are revealed. This is so-called *double dummy problem*.

2 Neural network architectures

In all experiments feed-forward networks created, trained and tested using JNNS (Java Neural Network Simulator) [1] were used. In most cases logistic (unipolar sigmoid) activation function was used for all neurons except for the case of representation of data using negative numbers, where the hyperbolic tangent (bipolar sigmoid) activation function was applied.

The number of input neurons was specified by the chosen method of deal's representation. The numbers of hidden layers and neurons varied. In most of the experiments the output layer was composed of a single neuron representing the estimated number of tricks taken by a pair NS. All networks were trained using Rprop algorithm [2], 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 $1E-4$.

3 The data

The data used in solving double dummy bridge problems was taken from GIB Library [3], created by Ginsberg's Intelligent Bridgeplayer [4], which is considered to be the best bridge playing program.

The GIB Library includes $717,102$ deals with all hands revealed. Additionally the library provides a number of tricks taken by the pair NS for each deal under the assumption of a perfect play of both parties. In all experiments the attention was fixed on a number of tricks taken by a pair NS for no trump play with player W making defender's lead.

The set of deals used in experiments was divided into three groups. The first $500,000$ deals were assigned to training. Deals numbered from $500,001$ to $600,000$ were assigned to validation, and the rest of deals to testing. The training set contained $10,000$ or $100,000$ deals depending on the complexity of the trained network.

4 Deals representation

The main approach of coding a deal for neural networks was to assign one input neuron for each card, hence networks had 52 input neurons which denoted the hand containing this card. The hands were coded as follows: $N : 1.0$, $S : 0.8$, $W : -1.0$, $E : -0.8$.

In the second approach to coding a deal each card of each hand was represented by two real numbers: the value (*two*, *three*,..., *king*, *ace*) and the suit (*Spades*, *Hearts*, *Diamonds*, *Clubs*). Both real numbers were calculated using a uniform linear transformation to the range $[0.1, 0.9]$. Networks had 104 input neurons (26 inputs for each hand).

Number of tricks was coded as a real value from the range $[0.1, 0.9]$, calculated using linear transformation from integer values: $0, \ldots, 13$.

5 Results

Results obtained for various input codings and neural architectures are summarized in Table 1. The best results were accomplished by networks in which a deal was coded by card assignment, but results achieved for the other coding are only slightly worse. The difference between these two approaches is evident when comparing training times - networks using the first approach (card

Table 1: Results obtained for various coding schemes and network architectures. The results are presented as three numbers: $A \mid B \mid C$, representing the fractions *in percent* of deals for which the network was mistaken by no more than 2 tricks (A%), no more than 1 trick (B%) and was perfectly right (C%).

Network type	Results for training deals (in %)
(26x4)-(13x4)-1	94.77 | 77.45 | 31.91
(26x4)-(13x4)-(7x4)-13-1	96.02 | 80.14 | 33.57
(26x4)-(13x4)-(13x4)-26-13-1	97.29 | 82.02 | 34.99
52-1	94.17 | 76.22 | 31.06
52-4-1	94.52 | 77.13 | 31.80
52-8-1	95.42 | 78.77 | 32.92
52-25-1	96.27 | 81.02 | 34.60
52-52-1	96.79 | 82.23 | 35.45

Network type	Results for testing deals (in %)
(26x4)-(13x4)-1	94.77 | 77.50 | 32.05
(26x4)-(13x4)-(7x4)-13-1	93.87 | 75.70 | 31.04
(26x4)-(13x4)-(13x4)-26-13-1	90.09 | 69.17 | 26.87
52-1	94.15 | 76.15 | 31.29
52-4-1	94.44 | 77.05 | 32.13
52-8-1	95.24 | 78.53 | 32.88
52-25-1	95.81 | 79.95 | 34.02
52-52-1	95.66 | 79.46 | 33.64

assignment) needed only several hundred iterations while networks with the other coding, i.e. $26x4$ - a few tens of thousands.

The best neural network had 25 hidden neurons ($52 - 25 - 1$), networks with bigger number of hidden neurons accomplished better results for training set, but worse for testing deals. On the other hand networks with fewer hidden neurons yielded worse results, but this degradation was relatively small, including the case of a network without hidden layer ($52 - 1$).

For more results and their deeper analysis the reader is referred to [5].

6 Analysis of trained networks

In this section weights of connections of several trained neural networks are discussed. All figures present the networks with 52 input neurons and one hidden layer, with 4, 8 or 25 hidden neurons resp. All of these networks received input values according to the first approach to coding a deal (see Section 4). Figure 1 presents the way of visualization of neural network's connections. Each circle represents the weight of one connection. If the circle is placed in the leftmost column, it represents the weight of connection from hidden to output neuron, otherwise - from input to hidden neuron.

The radius of the circle represents the absolute value of the connection's

Table 2: Weights of connections of trained network without hidden neurons $(52-1)$. Each value represents a weight of connection from the input neuron assigned to the given card to the output neuron.

Card's value	Spades	Hearts	Diamonds	Clubs
2	0.342	0.327	0.329	0.342
3	0.340	0.334	0.328	0.353
4	0.347	0.314	0.351	0.345
5	0.341	0.332	0.341	0.344
6	0.356	0.349	0.339	0.329
7	0.380	0.331	0.354	0.356
8	0.358	0.361	0.375	0.400
9	0.496	0.469	0.461	0.473
10	0.660	0.663	0.671	0.684
J	1.047	1.032	1.056	1.030
Q	1.676	1.688	1.675	1.656
K	2.643	2.643	2.677	2.655
A	3.975	3.971	3.966	3.989

weight and is calculated as linear transformation from the range $[0, 1]$. All weights with absolute values bigger or equal 1 are represented by circles with the same radius. The color of the circle denotes the sign of weight's value: black for negative and white for positive ones.

6.1 Network without hidden neurons $(52-1)$

The simplest trained network had no hidden neurons, so it contained 52 connections, which weights are presented in Table 2.

These weights of connections are very similar to Work point count - the human way of estimating the strength of cards (*ace* - 4 points, *king* - 3, *queen* - 2, *jack* - 1). This very simple network however achieved much better results than naive estimator of the number of tricks based only on the Work point count [5].

6.2 Networks with 4 hidden neurons $(52-4-1)$

Figure 2 presents weights of connections of 4 neural networks with 4 hidden neurons $(52-4-1)$. All these networks were trained independently using the same data. The results achieved by them were of similar quality. It can be noticed that most of weights with biggest absolute values are assigned to connections from input neurons representing *aces* and *kings.* These feature is quite natural - these cards are the most important in the play of bridge, especially in no trump play.

Some "special" hidden neurons, which fix their attention only on one suit, can also be pointed (e.g. the second one of the first network). More such neurons will appear in the networks with bigger number of hidden neurons.

Another interesting phenomenon concerns big absolute values of weights of all connections from hidden to output neuron. The absolute value of connection's weight determines the importance of the source neuron, hence a conclusion that all hidden neurons are relevant in these networks can be drawn from this feature.

6.3 Networks with 8 hidden neurons $(52-8-1)$

Weights of connections of 2 neural networks with 8 hidden neurons $(52-8-1)$, trained using the same data and achieving similar results, are presented in Fig. 3 The first conclusion which can be drawn from the figure is the presence of many hidden neurons focused on one particular suit. Another observable feature is increasing importance of inputs from *two* to *ace*.

There exist one hidden neuron which weights of input connections are surprising (the first hidden neuron of the second network). This neuron seems to be irrelevant for the network since the weight of its connection to output neuron equals -0.068, whereas all other connections from hidden to output neuron have absolute values bigger than 0.499. The number of such "useless" neurons increases for more complicated networks.

6.4 Network with 25 hidden neurons $(52-25-1)$

Figure 3 presents the network with 25 hidden neurons $(52-25-1)$ which achieved the best results. More complicated networks, with more hidden neurons or more hidden layers, achieved results better for the training set but worse for testing deals, mainly due to overfitting. Rectangles drawn using long-chain lines mark parts of input connections of hidden neurons which are specialized in one suit. This kind of weight pattern is repeatable, i.e. such four hidden neurons, focused on one suit only, could be found in all neural networks with 25 hidden neurons, trained using the same data

Another very interesting feature which appeared in all trained neural networks with 25 hidden neurons, was the presence of four hidden neurons specialized in five cards from one suit: *ten*, *jack*, *queen*, *king* and *ace* (in Fig. 4 marked using the dotted line). In all these groups the most important were *queens* and *kings*, *jacks* were less important, but still much more relevant than *aces* and *tens*. The hypothesis is that these hidden neurons are responsible for very important aspect of the play of bridge - the finesses.

7 Conclusions

The most important conclusion, which can be drawn from analysis of connections of trained neural networks, is the possibility to explain some patterns using human knowledge of the game of bridge.

Estimating strengths of suits, which is fundamental in human analysis of a deal, is performed by trained neural networks by assigning one hidden neuron for each suit. Such neurons consider values of cards - the connection from input neuron representing an *ace* has weight of biggest absolute value and the connection representing *two* - the smallest one.

Another four hidden neurons are specialized in a group of cards from one suit - *king*, *queen* and *jack*. This is also a part of human analysis of a deal, which allows to take into account a possibility of finesse - very important aspect of the play.

It must be emphasized that all described networks were trained only by presenting deals and target numbers of tricks. There was no human knowledge of the game, actually even rules of the game were unknown. In this case results achieved by networks and patterns visible in their connections weights should be considered as interesting and promising.

References

[1] http://www-ra.informatik.uni-tuebingen.de/software/JavaNNS/welcome_e.html

[2] Riedmiller M., Braun H. (1992), *A fast adaptive learning algorithm*, Technical Report, University Karslruhe, Germany.

[3] Ginsberg M.L., http://www.cirl.uoregon.edu/ginsberg/gibresearch.html

[4] Ginsberg M.L. (2001),*GIB: Imperfect Information in a Computationally Challenging Game*, Journal of Artificial Intelligence Research **14**, 303–358.

[5] Mossakowski K, Mańdziuk J. (2004),*Artificial Neural Networks for Solving Double Dummy Bridge Problems*, In: L. Rutkowski, J.H. Siekmann, R. Tadeusiewicz and L.A. Zadeh (Eds.), Artificial Intelligence and Soft Computing - ICAISC 2004, 7th International Conference, Zakopane, Poland, Lecture Notes in Computer Science **3070**, *Springer-Verlag*, 915–921.

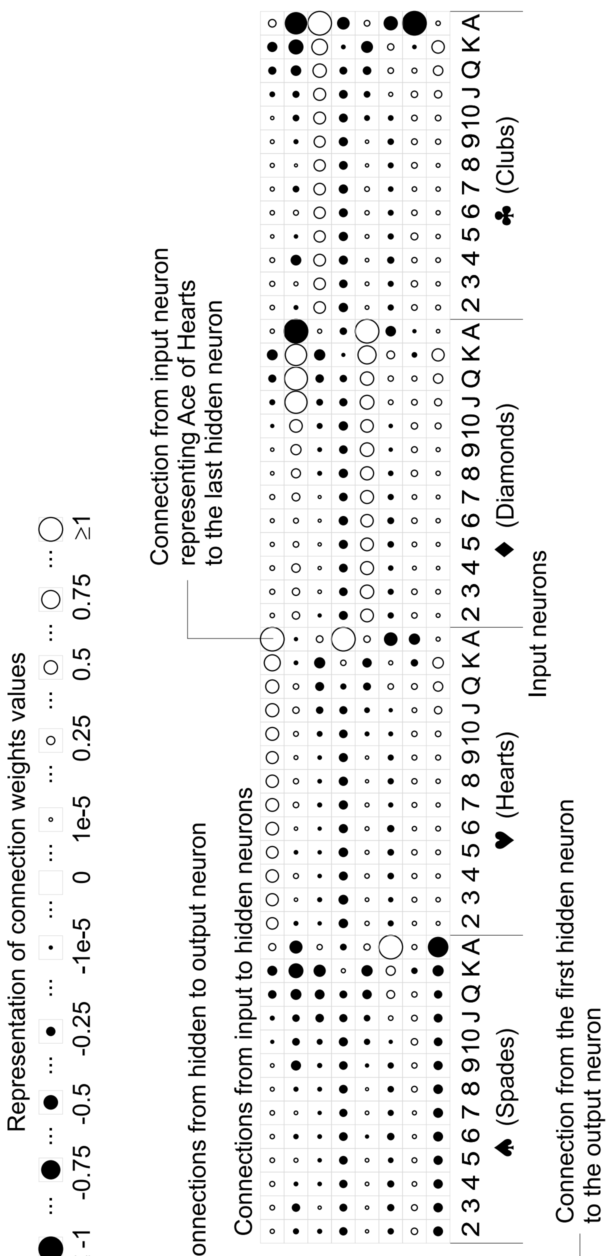
Representation of connection weights values
≤ -1
-0.75
-0.5
-0.25
-1e-5
0
1e-5
0.25
0.5
0.75
≥1
Connection from input neuron representing Ace of Hearts to the last hidden neuron
Connections from hidden to output neuron
Connections from input to hidden neurons
Hidden neurons
2 3 4 5 6 7 8 9 10 J Q K A
♠ (Spades)
♥ (Hearts)
♦ (Diamonds)
♣ (Clubs)
Input neurons
Connection from the first hidden neuron to the output neuron

Figure 1: Visualization of trained neural network connection weights values.

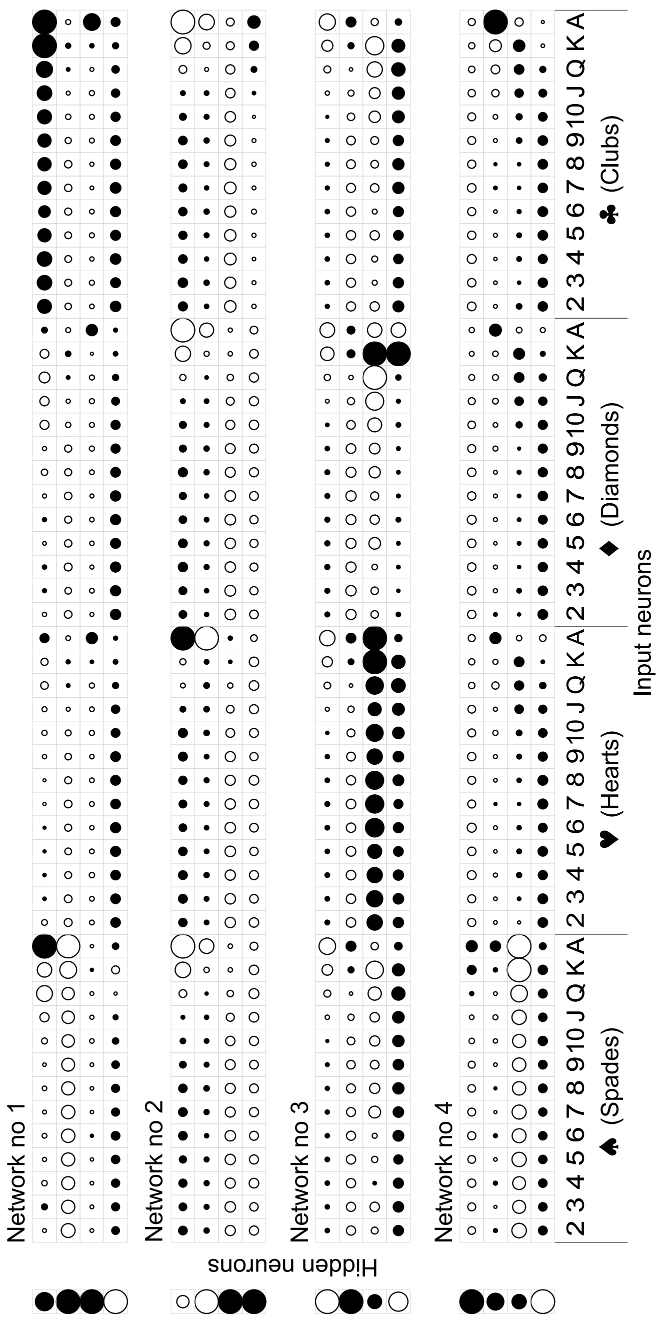

Figure 2: Weights of connections of 4 independently trained neural networks with 4 hidden neurons ($52-4-1$).

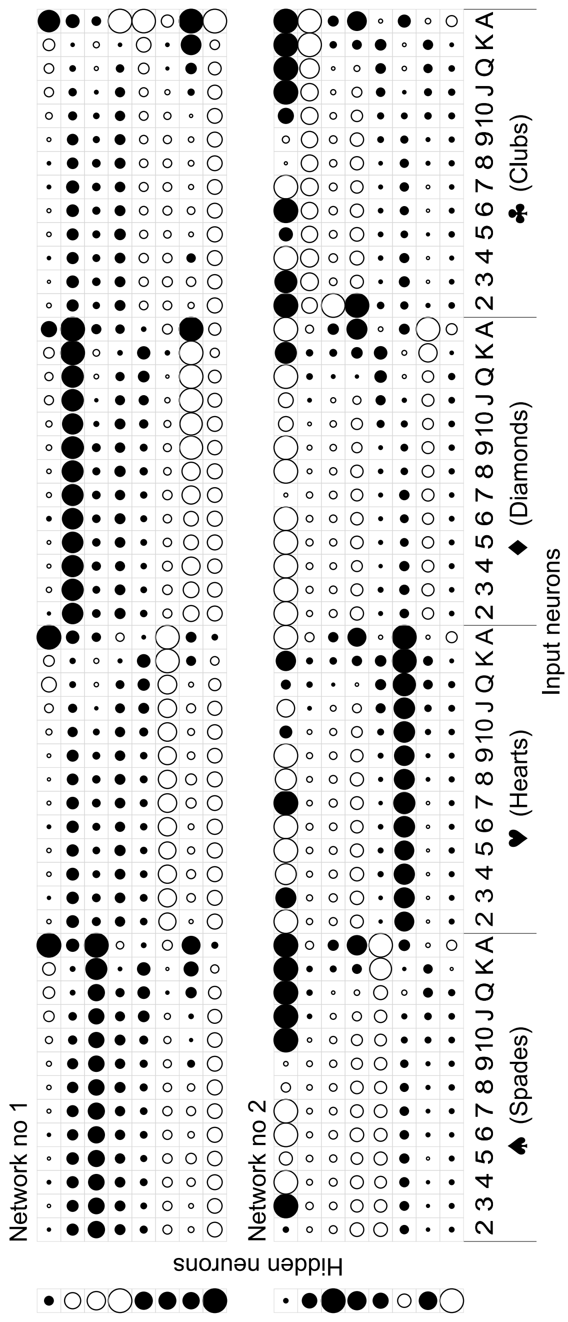

Figure 3: Weights of connections of 2 independently trained neural networks with 8 hidden neurons $(52 - 8 - 1)$.

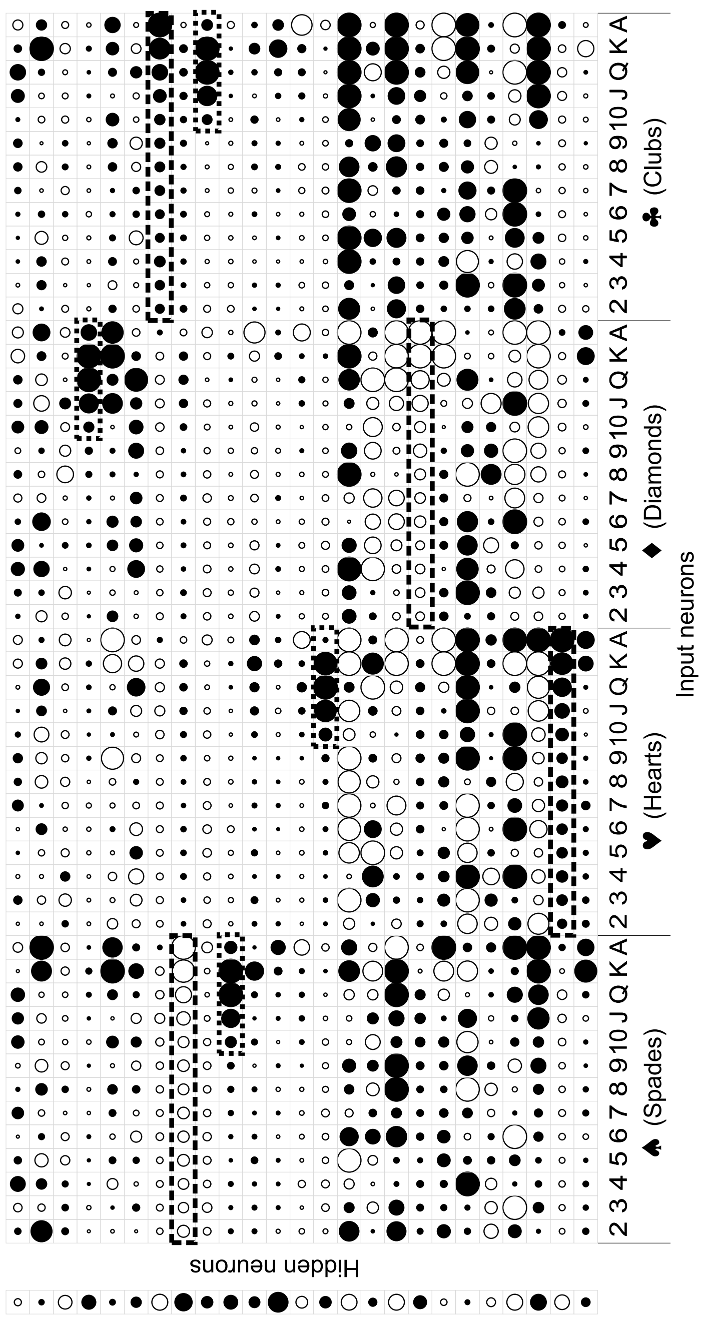

Figure 4: Weights of connections of trained neural network with 25 hidden neurons $(52 - 25 - 1)$.