Neural Connect 4 – A Connectionist Approach to the Game

Marvin Oliver Schneider
moschneider@ig.com.br

João Luís Garcia Rosa
joaol@ii.puc-campinas.br

Mestrado em Sistemas de Computação, Pontifícia Universidade Católica de Campinas,
Rodovia D. Pedro I, km. 136, Caixa Postal 317, CEP 13012-970, Campinas – SP, Brazil.

Abstract

This article presents the system "Neural Connect 4", a program that plays the game Connect Four. This system employs the multilayer perceptron architecture which is learning through the supervised backpropagation algorithm. The required knowledge for training comes from saved games.

After a short introduction to the game itself, the symbolic algorithms used for training and evaluation are described. Comparisons are made within a connectionist approach: effective and ineffective learning techniques are shown and the results are discussed.

"Neural Connect 4" proves that artificial neural networks are completely adequate for learning Connect Four, given that certain principles are observed.

1. Introduction

Connect Four is a popular board game, easy to learn, but difficult to master, since tactical knowledge is required to play well.

The system "Neural Connect 4" is based on the neural network capacity of learning this knowledge adequately.

The main purpose of the project is to prove the abilities of Neural Networks to perform well in non-trivial situations.

Other systems using Neural Networks with board games may be found as references ([1], [2] and [7]).

2. Description of the game

2.1. Basics

Connect Four is a game which is played with a vertical board of 7 x 6 positions (a matrix). Each player has 21 one-color pieces available and, with these pieces, should be able to form vertical, horizontal, or diagonal 4-piece lines. The one who creates the first 4-piece line, wins the game. An interesting feature of Connect Four is the fact that the pieces actually may not be positioned freely on the board, but fall from above into the lowest free position of the column selected, e.g. if there are no pieces already in the chosen column, the piece falls down to the lowest vertical position; in case that there are pieces, it is positioned above the last one. In this way "piles" of pieces are formed. It is important to notice that, because of this fact, there are at most only 7 possibilities to put a piece (see example in Figure 1).

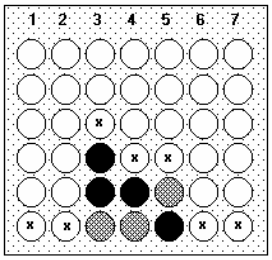

Figure 1. An example of a situation during a game. Possible positions for the next move are marked with an X.

2.2. Tactical knowledge

The game is not as simple as it looks at first sight, because a lot of tactical knowledge is required to become a winner. Imminent lines of 4 are easily detected (and the algorithm that can do this without missing a single one, is quite simple). The real "kick" of the game is to actually force the opponent to permit (or even to contribute to) a line of four. This is done in two ways:

2.2.1. Two open lines. Since each player is able to put only one piece at a time, two lines of three, which are open, i.e. which have free spaces to form lines of four, allow the opponent to close just one of them, and thus

enable the first player to form his line of four pieces on the other side that is still open (see example in Figure 2).

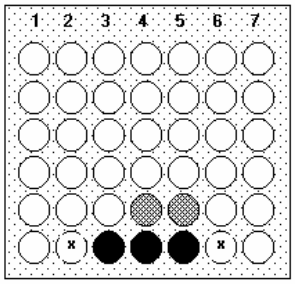

Figure 2. An example of two open lines. White has to play either 2 or 6 in order to close the respective line of Black. However, since White is not able to close both possibilities in one move, Black wins the game.

2.2.2. Forced open lines. In order to prevent losing the game it is necessary to close any open line of three that the opponent may form. If, however, this closure produces another open line of three, there will be a figure of a "forced open line" and the game is won (see example in Figure 3).

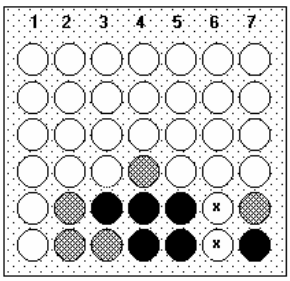

Figure 3. Example of a forced open line. White has to play 6 in order to close Black's line 4,5,6,7. However, this opens the line 3,4,5,6 and Black wins the game playing 6.

3. Description of the system

The system consists of several symbolic procedures as well as five neural network structures. The symbolic algorithms were created in order to supply the necessary knowledge to train and test the neural networks.

Neural Connect 4 provides four different symbolic solutions, which make use of different philosophies and techniques to play the game. None of these algorithms can be considered perfect, however, different levels of efficiency and computability may be clearly observed [1].

3.1. Iterative solutions

Naive 4: Naive 4 is the system's weakest algorithm (see Figure 4). It plays randomly unless it finds an open line of three. If the algorithm created this line, it closes it to a line of four in order to win the game. If the system finds an open line of three of the opponent, it closes the line in order to avoid losing the game. Tactical elements are actually not considered. The algorithm is still instructed to not create accidentally an open line for the opponent.

On an empirical basis of ten beginners, who have played against Naive 4, it can be said that this algorithm represents a challenge for players that are not able to analyze the situation on the board yet – these human players are only seeking to create easily detected open lines of three (especially vertical ones). Advanced players, however, do not find major difficulties in winning against Naive 4.

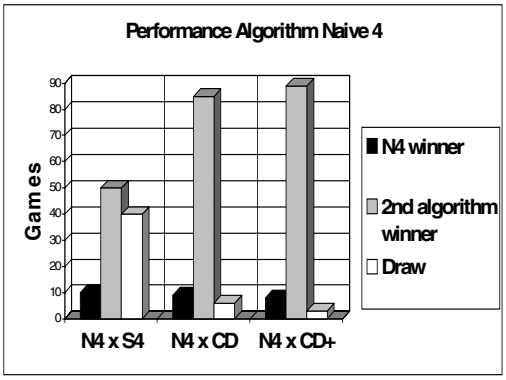

Figure 4. An analysis of performance for Naive 4. 100 games were played in each comparison between Naive 4 and one of the other symbolic algorithms.

Constructor/Destroyer: Constructor/Destroyer is a more sophisticated algorithm. Its assumption is that the game consists of constructing lines and destroying the ones of the opponent. Starting with creation or destruction of eventual lines of four, the algorithm searches for lines of three and lines of two, which, however, should have enough room to create a line of four later – otherwise, the consideration of these possibilities does not make sense, since it would not be linked to an eventual winning situation. The results of the evaluation are stored in an array of points and the best move is chosen. Certainly, the highest (and "unreachable") score is attributed to the construction of an own line of four, and the lowest score to a line of two.

The algorithm plays reasonably well, according to the documented diagram of relative performance (Figure 5). The only problem is the fact that it does not consider all of the tactical knowledge given above.

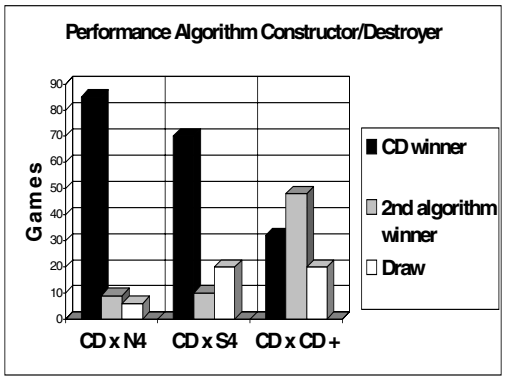

Figure 5. An analysis of performance for the Constructor/Destroyer algorithm.

Constructor/Destroyer+: As the name suggests, the Constructor/Destroyer+ is an advanced version of the Constructor/Destroyer. It contains all elements of the Constructor/Destroyer, but also takes into account situations with two open lines and forced open lines.

Tests have shown that the number of executed procedures makes it difficult to not end up in conflicts between one element and the other. The distribution of the points is the real challenge of this algorithm.

This way, its somewhat disappointing performance when playing against the Constructor/Destroyer can be explained (Figure 6) – however, the real difference may be seen in a match with a human player. In this case Constructor/Destroyer+ may perform much better than the other algorithms, since it perceives possible tactical tricks, part of any advanced player's knowledge.

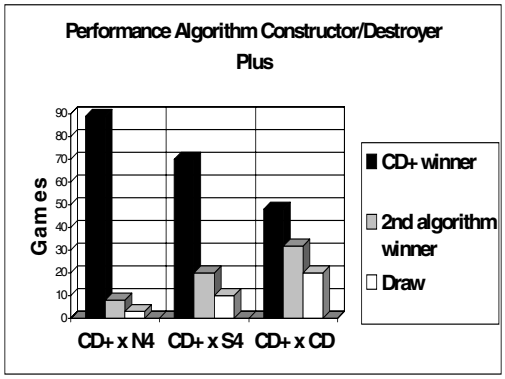

Figure 6. An analysis of performance for Constructor/Destroyer+. Notice that it is clearly better than all the other symbolic solutions.

3.2. Recursive solutions

Search 4: Search 4 was created inspired in a classical approach [1] [4] [7]. This algorithm recursively searches for lines of four pieces (either own or of the opponent). After encountering a line of four, all branches of recursion starting at this point are cut off. Points are distributed according to the number of lines and final scores are written to a 7-position array of points.

The algorithm has surprisingly proven not to be very efficient (Figure 7), considering its quality of game playing, as well as the necessary processing time, which can be up to 50 times greater than that of the iterative solutions. This way, Search 4 is definitely not worthwhile in the context of this game.

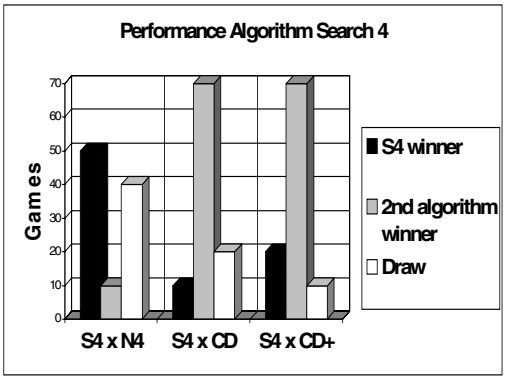

Figure 7. An analysis of performance for Search 4. It can be noticed that this algorithm is clearly weaker than Constructor/Destroyer and Constructor/Destroyer+.

4. Neural networks

Since the focus of this project is the application of artificial neural networks, the symbolic algorithms given above were mainly developed in order to supply training data for the network and to validate its learning success, not only by measuring its errors in the training situations, but also in real game situations playing against algorithms that have proven to be efficient.

As a general network structure, a multilayer perceptron architecture was chosen with one hidden layer. The learning is reached using the backpropagation algorithm [5], and the sigmoid as its activation function [3].

4.1.Topologies

Altogether five different network architectures were created. Their differences lie only in the number of neurons in the hidden layer, which provokes a different number of synapses in layers 1 and 2 (more details in Figure 8). The purpose was to find out, how especially the number of hidden neurons would influence the learning results.

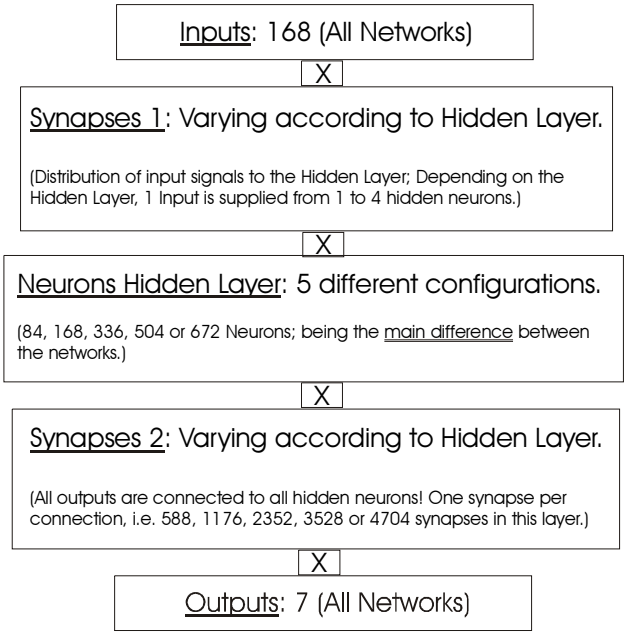

Figure 8. Parameters of the 5 networks.

It is important to notice that the number of inputs is always the same: each field on the board corresponds to 4 inputs (42 x 4 = 168). Inputs are binary: 1 means that a certain feature applies and 0 that it does not apply. Entries by number mean:

1 = empty
2 = current color
3 = opponent's color
4 = (void – shall be used in future versions)

Since the rate of synapses of layer 1 and hidden neurons in network 1 is 2:1, the network makes use of a contraction method. In network 2, they are the same number and in network 3, the relation is the contrary, which means that the network is actually detailing the entries, thus becoming more powerful, but also much slower.

Synapses in layer 2 connect all hidden neurons to output neurons, which means that it is actually a layer for analysis.

The 7 outputs correspond to the 7 possible moves at each turn. The neurons supply an output value between 1 and 0; the move with the highest value attributed will be chosen.

4.2. Learning

The learning algorithm used is classic backpropagation with a learning rate of 0.35.

The learning system is based on saved games [6]. The algorithm assumes that – in order to win – the network must follow the steps of the game's winner, meaning that it shall play the same way in the same situation.

It may be deduced that the quality of the saved games is crucial, as they are the knowledge from which the neural network learns.

Several tests were made concerning the quality of the learning process – some including learning for many days – and the respective results are given in the following item.

5. Results

5.1. Learning behavior

5.1.1. Quality of learning. Quality of learning, i.e., the ratio of successes for a certain number of cases, depends on:

- Number of cases trained. Although it seems that a huge amount of data contains a lot of knowledge, and the result of learning thousands of positions for a long time would result in a very powerful network, practical tests have shown the opposite in "Neural Connect 4".

As shown in Figure 9, the network is simply unable to assimilate huge game archives.

- Available time. The available time for learning converts directly into the number of times that a game is presented to the net, making a great difference to its learning. The graph has a somewhat hyperbolic tendency (Figure 10), i.e. in the beginning - taken that the game archive is adequately chosen - the network generally jumps from high error-levels to much lower rates (about 1-2% at most).

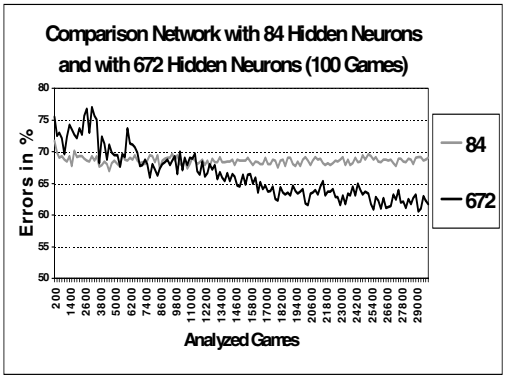

Figure 9. Analysis of learning with a 100 random games archive. The network with 672 hidden neurons is very slow in adjusting itself, whereas the network with 84 neurons is completely unable of learning this form of data.

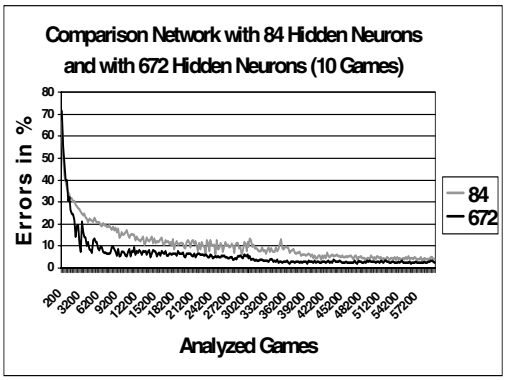

Figure 10. Longtime quality analysis for the smallest and the largest network (84 and 672 hidden neurons).

- Network topology. The network topology is very important to the success of learning. Once chosen a structure (as in the case of "Neural Connect 4" a multilayer perceptron) the main parameter is the number of hidden neurons. Tests with the 5 networks of "Neural Connect 4" have proven that small neural networks tend to adjust very slowly (if at all) and too huge neural networks may start to oscillate more easily (Figure 11). Thus an intermediate structure might be the best choice.

5.1.2.Time for learning. An important issue, which must not be neglected, is the time the respective networks need to learn. A comparison between the five networks of "Neural Connect 4" (Figure 12) shows an almost linear increase of time needed, which makes the larger networks become unrealistic for usage in simple situations.

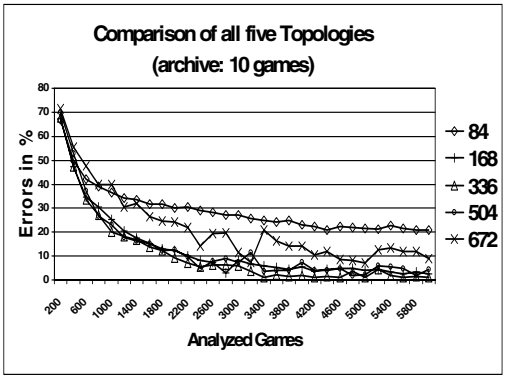

Figure 11. A learning analysis for all the five networks. The network with 336 neurons is the best.

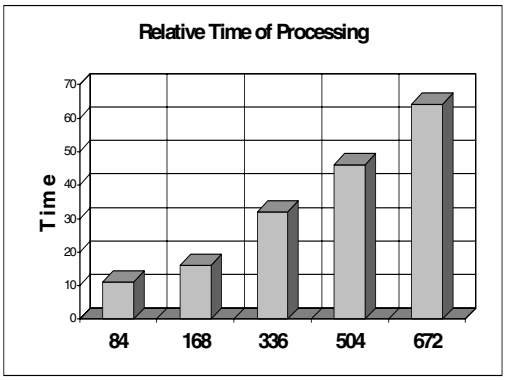

Figure 12. A comparison of time needed for learning considering all 5 networks of "Neural Connect 4".

5.2. System of small changes

In the beginning it was considered that there would be two ways to treat learning in "Neural Connect 4": by the array of points, which is being generated at every move, or by the outcome itself, i.e. the decisions that the networks would take independently of small differences of points.

Tests were made and it was decided that the later procedure was giving better results on a long or medium term (see Figure 13). The system itself is a system of small changes, i.e., adjustment by backpropagation is only executed if the decision of the network is incorrect and only the points of the wrong and the correct decision are modified (e.g. correct decision 5, network plays 7, so position 7 in the array receives 0 and position 5 receives 1, whereas the other points for positions 1,2,3,4 and 6 remain unchanged and finally the set is learned).

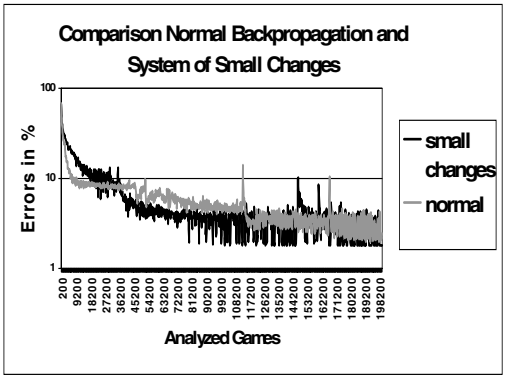

Figure 13. Justification for the system of small changes. Only in the beginning, modification of all points will be more effective.

5.3. Usage of neural networks

The network has shown to be efficient to learn tactical knowledge from saved games. Even learning only 10 good games, the network turns out to be superior when compared to all the other symbolic algorithms (see Figure 14).

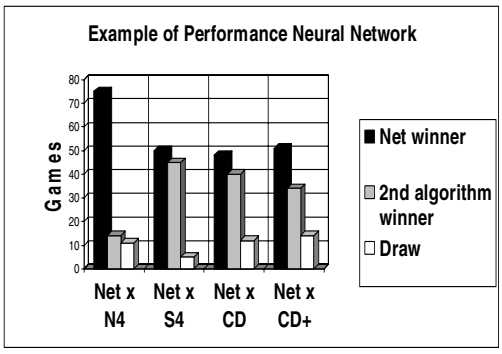

Figure 14. Typical performance of the network. The network used was the one with 332 neurons on the basis of a well chosen 10-game-archive, which learned until reaching 97% of success.

Meanwhile it has to be pointed out that "Neural Connect 4" uses the network for tactical analysis, a task it may perform better than all symbolic algorithms, whereas the routines for closing open lines of three were chosen from the symbolic procedure. This represents a relatively small part of processing, which is not due to any tactical knowledge at all and is easily implemented.

5.4. Problems and improvements

It became clear that small well-chosen game-archives are better than big random archives. Thus an automatic production by letting the symbolic algorithms play one against the other, which was planned at first, has shown to not be a good solution – this may be especially due to the production of eventual contradictions.

The following features are planned for the future:

- Negative learning, i.e. the network shall learn from mistakes and not repeat them
- No learning of irrelevant knowledge
- Usage of recurrent networks.

6. Conclusion

The present article pointed out how to apply neural network architectures in a game situation.

There is no doubt that this field of application of neural networks is in plain development showing the real power of neural networks to solve problems, for which intelligence is needed.

7. References

[1] Buro, M., "*Techniken für die Bewertung von Spielsituationen anhand von Beispielen*", Doctorate Thesis at FB17 Universität-GH-Paderborn, Paderborn, 1994

[2] Freiesleben, B., "*A Neural Network that Learns to Play Five-in-a-Row*", in Proceedings of the 2nd New Zealand Two-Stream International Conference of Artificial Neural Networks and Expert Systems (ANNES '95), Siegen, 1995, pp. 87-90

[3] Helbig, H. et alii, "*Neuronale Netze*", FernUniversität, Hagen, 2000

[4] Mahrenholz, D., "*Strategie des Computers im Vier-Gewinnt-Spiel*", http://ivs.uni-magdeburg.de/~mahrenho/index.html, Magdeburg, 1995

[5] Rumelhart, D. E., Hinton, G. E. and Williams, R. J., "*Learning Internal Representations by Error Propagation*", in D. E. Rumelhart and J. L. McClelland (Eds.), Parallel Distributed Processing – Explorations in the Microstructure of Cognition, Volume 1 - Foundations. A Bradford Book, MIT Press, 1986, pp. 318-362.

[6] Stricker, T., "*Problemlöseverfahren, Algorithmen, Datenstrukturen*", http://www.cs.inf.ethz.ch/edu/37-836/vorl/vorl.html, Informatik II (37-836) ETH Zürich, Zürich, 1998

[7] Thrun, S., "*Learning To Play the Game of Chess*", in G. Tesauro, D. Touretzky, and T. Leen, (Eds.), Advances in Neural Information Processing Systems (NIPS) *7*, Cambridge, MA, MIT Press, 1995.