Trained Neural Networks Play Chess Endgames

Jie Si
IEEE Member, ACM member
Comsearch
Reston, VA 20191

Rilun Tang
Department of Computer Science
University of Nevada, Reno
Reno, NV 89557

Abstract

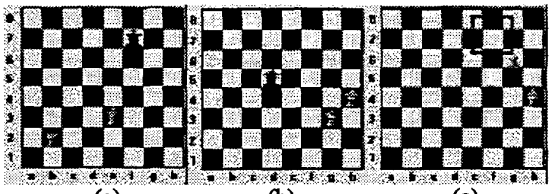
In this paper, three types of chess endgames were studied and three layer feedforward neural networks were applied to learn the hidden rules in chess endgames. The purpose of this paper is to convert the symbolic rules of chess endgames into numerical information that neural networks can learn. The neural networks have been proved efficient in learning and playing some simple cases of chess endgames.

I. Introduction

It has been a trend since the 1980s to develop some robots or computers to play games with human being. Among all the games, chess got most enthusiasm. This is not only because chess is one of the most popular games all over the world, but also due to the reason that it is a game requiring high intelligence. Neural networks may be applied well in various practical areas, e.g. pattern recognition, signal processing, etc [1] - [3]. It is appropriate to use neural networks to solve some intelligent problems because neural networks are parallel, distributed systems with high fault-tolerance. In this paper, the authors tried to use feedforward neural networks to learn the patterns in chess games. However, it is almost impossible to train a neural network to remember all changes and traps in a chess game since chess as a whole is too complex even for human's brain. Therefore, only some simplest cases of the end games, whose complete knowledge is available, were considered. The aim of the paper is to transform the end games information into neural network structure.

II. Examples of end games

Fig. 1 shows some examples of chess end games. As it can be seen from Fig. 1, a chessboard consists of 8 vertical files and 8 horizontal ranks. The ranks are labeled from '1' to '8' while the files are labeled from 'a' to 'h'. Just like a point in Cartesian geometry, the location and moves of pieces in a chessboard can be totally expressed in terms of ranks and files. For example, chessboard condition shown in Fig. 1(a) can be expressed by 'WK e3, WR b2, and BK f7', where 'W', 'B', 'K', 'R' represent white, black, king and rook respectively. The expression 'R b2-b3' means to move the white rook one square upwards. Some abbreviations used in this paper are 'Q' for queen, 'P' for pawn, 'K' for king, 'R' for rook, 'W' for white and 'B' for black. Three types of end games, i.e., single rook vs. lone king, single queen vs. lone king, and single pawn vs. lone king, depicted as Fig. 1(a), (b), and (c) respectively, were studied in this paper. In each type, there are always three pieces on the chessboard. For simplification, white side is set to be stronger than the black side.



(a) (b) (c)

Fig. 1 Examples of chess end games

In a chess game, players select the next move according to the current condition of the chessboard. Generally, there are multiple choices for each move. Expressed in mathematical ways, if the current condition and the next move are considered to be input and output of a model, this model actually completes a one-to-multiple mapping process. It makes chess game very hard to learn, either by human or by computer. Fortunately, for some simplest end games, there exist the definite solutions which can simplify the playing to be a series of one-to-one mappings. It has been proved that for finite games like chess, an optimal strategy can be constructed based on the set theoretic considerations [4], [5]. In the 1970s, these ideas resulted in the construction of comprehensive End game DataBase (EDB) which provides the supervisor for the training of neural networks.

0-7803-5529-6/99/$10.00 ©1999 IEEE

III. Neural networks used for chess end games

1. Neural network for rook end games

A. Neural Network structure

The neural network used for rook end games is shown in Fig. 2. It is an all connected feedforward neural network with one hidden layer. The number of neurons in the hidden layer, which are not drawn to make the plot neat, is 34. There are 18 nodes in the input layer, divided into three groups, each of which corresponds to the position of a particular piece. In the output layer there are 6 neurons. The white neuron in the output layer in Fig. 2 is used as a king/rook index. If this index is 1, the output of the other five neurons indicates a rook's move. Otherwise, the output of the other five neurons indicates a king's move. The black neurons show the direction in which a rook needs to move if the K/R index is 1. If the K/R index is 0, the output of these two neurons is always 01. The gray neurons have two functions. If the K/R index is 1, i.e., a rook's move, the output of these three neurons gives the squares a rook needs to move in the direction indicated by the black neurons. If the K/R index is 0, these three neurons show the direction in which the king moves.

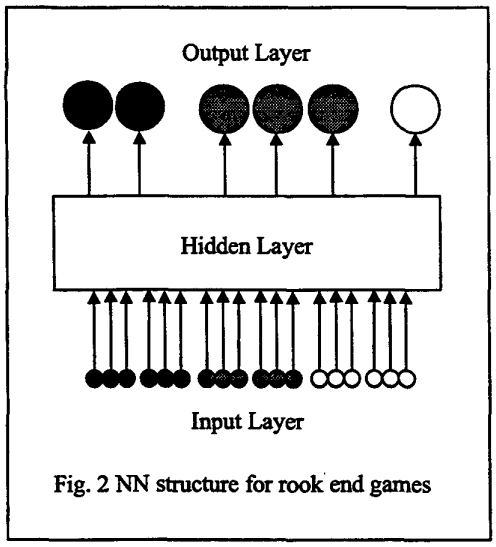


Fig. 2 NN structure for rook end games

B. Construction of training data

As it is known, the location of each piece can be totally described by its rank and file, both range from 1 to 8. Therefore, three digits ranging from 000 to 111 can be used to show the rank/file of a piece. The location of each piece needs only 6 digits to be totally expressed. On the output side, if the next move is the rook's move, three digits, changing from 001 to 111, are needed to show the squares a rook needs to move, which is a number between 1 and 7. Another two digits are needed to show the direction in which the rook wants to move. The conversion between these two digits and the direction is shown in Fig. 3(a). If the next move is moving the king, only three digits are needed to show the moving direction of the king, since a king can only move one square at a time in any direction. The conversion between these three digits and the moving directions of the king is shown in Fig. 3(b). The two digits that are used for indicating rook's moving direction are set to be 01 if a king is moving. Besides these digits, an extra digit is needed to indicate if the output is king's move or rook's move.

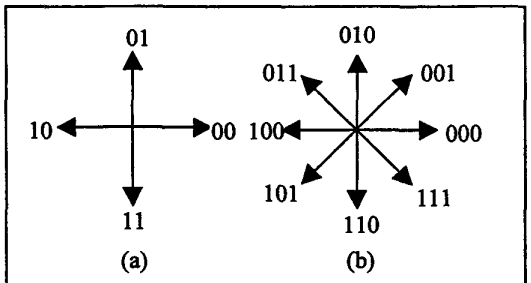


Fig. 3 Conversion between directions and binary codes

C. Training algorithm

In this paper, a hybrid algorithm which combines both BP and random search was used to train the neural network to guarantee the global minimum of the error function [3].

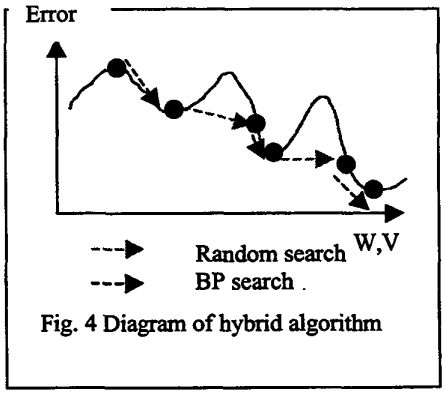


Fig. 4 Diagram of hybrid algorithm

Fig. 4 shows a simple diagram of this algorithm. The basic idea of this algorithm is to use BP to find a local minimum, then in a range around this local minimum, randomly change the weight matrices until a smaller error function value is found. Based on these new weights, run BP algorithm again to find another local minimum. Continue this process until the global minimum is found. The random search range should be reduced during the search.

2. *Neural network for queen end games*

A. Neural network structure

The structure of neural network for queen end games is more complicated compared to the one used for rook end games. It is so because a queen can move in 8 directions instead of 4 in which a rook can move. As Fig. 5 shows, the input layer has 18 input nodes, corresponding to the current location of kings and white queen. The output layer has 7 neurons. The white neuron is used as K/Q index, three gray neurons indicate the squares a queen needs to move, and three black neurons show the direction in which white queen or king moves. The number of neurons in the hidden layer is 40.

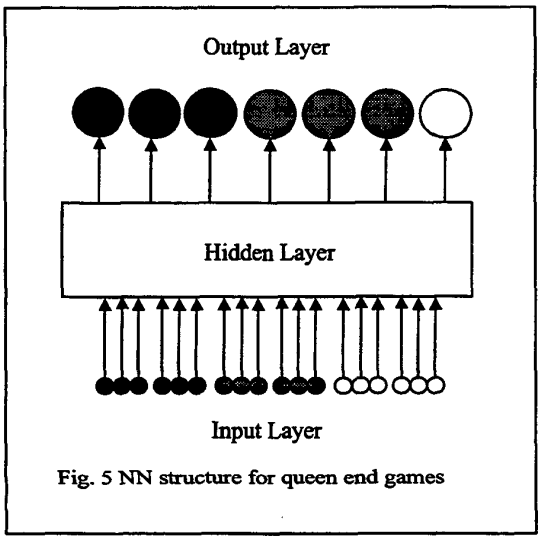


Fig. 5 NN structure for queen end games

B. Construction of training data

The conversion of input data vectors is same as that in the rook end games. In the output data vectors, three digits are needed to indicate the direction of the queen/king's move. Both queen and king can move in 8 directions, and are encoded in a way following Fig. 3(b). The other three digits are used to show the squares the piece moves. The moving squares could range from 001 (1) to 111 (7) if the queen is moving, and is always 001 (1) if the king is moving. Another digit is used as King/Queen index which can be 1 (queen's move) or 0 (king's move).

C. Training algorithm

The hybrid algorithm described in section III1C was used to train this neural network.

3. *Neural network for pawn end games*

A. Neural network structure

The pawn end games are the most complicated one among all three types of end games discussed in this paper [5]. The neural network structure for pawn end games, however, is easier than the other two. It is so because in a chess game, both pawn and king can only move one square at a time, and a pawn can only move in the forward direction. The neural network structure for pawn end games is shown in Fig. 6. The input layer is same as that in Fig. 2 and Fig. 5. The output layer has only four neurons, indicating the direction of white king's next move. There are 8 directions a king can move, resulting in nine choices for the position of the king on the chessboard in the next instance, i.e., 8 directions plus staying. If the output shows the king should stay, it amounts to that the pawn moves one square forward. The number of neurons in the hidden layer is 34.

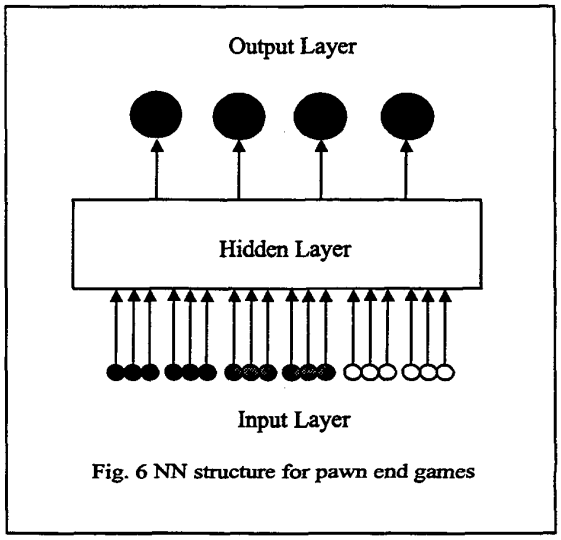


Fig. 6 NN structure for pawn end games

B. Construction of training data

The input data vectors are same as those in rook end games and queen end games. Corresponding to the neural network structure described above, the target output data vectors have four digits. 8 possible directions are indicated by numbers from 0000 to 0111, whose last three digits follow the rule shown in Fig. 3(b), number 1111 indicates the king to stay, i.e., the pawn should move one square forward.

C. Training algorithm

The hybrid algorithm was used for pawn end games.

III. Simulation experiment and results

There are two phases in the simulation experiments, training and verification. In the training phase, neural networks and the corresponding training data vectors are constructed in ways described in section III. Hybrid

algorithm was applied to train the neural networks. In verification phase, the current chessboard conditions were sent to the trained neural networks, the output of neural networks shows the next move of pawn, queen, rook or king of the white side.

1. Training

To save the running time, in this paper, only 1000 input-output training vector pairs were chosen to train the neural network for each type of end games.

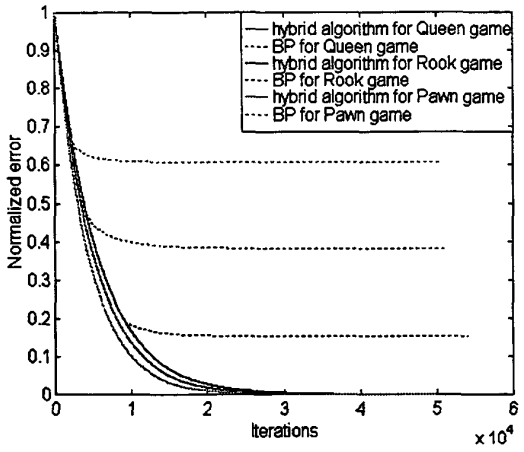


Fig. 7 Training process of chess endgames with BP and hybrid algorithm

Fig. 7 shows the process of the training. The error function has the same definition as in [1]. All curves have been pro-cessed with the fifth order curve fitting. Results of using BP and the hybrid algorithm are drawn in the same plot. From this plot, it can be seen that for each type of end games, after about 35000 iterations, the error dropped below 0.01. The pawn end game has the fastest learning process while the rook end game is the slowest one. After learning, the error levels for all types of games are almost same. BP algorithm, on the other hand, got much worse results. In the training of each end game, the error stopped dropping at higher levels, meaning that the learning stopped at a local minimum. In this figure, the worst results by BP were drawn for purpose of comparison.

2. Verification

After training, neural networks were used to play some selected end games to verify the results of training. Each end game was played in a sequence between one particular neural network and the authors. More specifically, at first neural network indicates the move of the white side based on the current condition of the chessboard, and the authors moved the black king to a new location, then neural network indicated the next move based on the new chessboard condition. Follow this sequence until game over. All instances used for verification have been selected to train the neural networks. Each number of the output vector of neural networks is rounded to 0 or 1. The results proved that after training, neural networks can remember patterns by which they were trained and can give good results for the further applications.

IV. Conclusions

In this paper, the chess end game database was encoded and neural networks were trained to learn the playing patterns. The simulation experiments showed good results. Compared to the previous work in this field, the flexibility and complexity of the chess endgames were expanded in this paper. We proved that using a single feedforward neural network, it is possible to store the complicated information hidden in chess games.

Reference

1. Looney C., *Pattern Recognition Using Neural Networks, Theory and Algorithms for Engineers and Scientists*, Oxford University Press, 1997.
2. Haykin S., *Neural Networks, A Comprehensive Foundation*, Macmillan College Publishing Company, 1994.
3. Baba N. *et al*, "A hybrid algorithm for finding the global minimum of error function of neural networks and its applications", *Neural Networks*, Vol. 7, pp. 1253-1265, 1994.
4. Posthoff C., *et al*, "Neural network learning in a chess endgame", *Proc. IJCNN*, pp. 3420-3425, 1994.
5. Tarrasch S., *The Game of Chess*, pp. 9-76, Dover Publications, Inc., 1987.