3C3-IOS-2-8

A Supervised Learning Method for Chinese Chess Programs

Wen-Jie Tseng[1], Jr-Chang Chen[2*], I-Chen Wu[1], Ching-Hua Kuo[1], Po-Han Lin[23]

[1] Department of Computer Science, National Chiao Tung University
[2] Department of Information & Computer Engineering, Chung Yuan Christian University
[3] Department of Computer Science & Information Engineering, National Central University

The Elo-rating system and the Minorization-Maximization (MM) algorithm have been used for pattern learning in the game of Go. This paper proposes a learning method based on the Elo-rating and MM to help adjust feature weights in Chinese chess programs, enhancing position evaluation accuracy. The learning method automatically adjusts feature weights iteratively according to expert game records. In experiments, we demonstrate the method by adding three new features, including a different way of calculating piece mobility. The experimental results show the playing strength of our Chinese chess program, CHIMO, has improved, yielding a win ratio of 61.7% against the pre-learned version. In addition, we also justify the method by demonstrating the learned feature weights for piece mobility.

1. Introduction

Chinese chess is one of the most popular two-player chess-like games. It has a long history and very important cultural status. It was estimated in [CBCGAC 2013] that about 200 million people play Chinese chess worldwide. In game theory, Chinese chess has higher game-tree and state-space complexities than chess [Herik 2002].

Players in Chinese chess conventionally play as red and black sides, and the red side moves first. Each player has sixteen pieces on the board, including one king (K), two advisors (A), two elephants (E), two rooks (R), two horses (H), two cannons (C), and five pawns (P). In the game, players move alternately and their main goal is to capture the opponent's king.

Chess or chess-like game programs have been an important research topic in artificial intelligence [Allis 1994][Campbell 2002]. Alpha-beta search ($\alpha\ \beta$ search) and quiescence search (QS) [Marsland 1992] are the most common algorithms used in Chinese chess programs. An evaluation function [Knuth 1975], used in search algorithms to evaluate a position by giving it a score, may influence the search tree and even the search result.

In many game-playing programs, when a position p is being evaluated, each feature of p is given a weighted score and eventually the sum of all feature weight scores is returned as the score of p. For example, if one rook is evaluated at 1000 points and one pawn is evaluated at 100 points, a position with one rook and two pawns is evaluated at 1×1000+2×100=1200 points. For accuracy, Chinese chess programs usually try to evaluate a large number of features. For example, our Chinese chess program CHIMO, which has won a silver medal in ICGA Tournaments, uses about 1500 features during evaluation.

However, manually adjusting each feature weight accurately is time-consuming. For this problem, [Cheng 2006] proposed a method to automatically obtain weights for certain features using statistical methods. His method counts the appearance of pieces in different locations in many expert game records and converts the counted value to scores using self-defined formulas. However, the method can only be applied to specific features for evaluating pieces in different locations. [Kao 2009] also proposed a genetic algorithm method to automatically adjust feature weights, for which any feature can be applied to. However, the method requires considerable computation time for self-play testing after each generation in order to choose good feature weights into the next generation.

In this paper, we propose a supervised learning method based on the Minorization-Maximization (MM) algorithm that adjusts feature weights of Chinese chess programs automatically by learning from expert game records. Moreover, when new features are added, the method can adjust new feature weights rapidly and accurately. Most importantly, the method uses QS to evaluate the moves in game records to avoid being affected by the horizon effect.

In our experiments, we applied the method to CHIMO with three newly added features. The new features improved the playing strength of CHIMO, which was verified through self-play games against its original version.

Contact: Jr-Chang Chen, Department of Applied Mathematics, Chung Yuan Christian University, No. 200, Chung Pei Rd., Chung Li, Taoyuan County 32023, Taiwan (R.O.C.), phone: +886-3-2653131, email: jcchen@cycu.edu.tw

2. Related Work

Section 2.1 introduces the main algorithms used in computer Chinese chess programs. Section 2.2 introduces the design of the evaluation function in CHIMO. Section 2.3 explains how we applied the Elo-rating system and Bradley-Terry model to features and describes the MM algorithm used for learning.

2.1 Alpha-beta search and quiescence search

$\alpha\ \beta$ search is the main algorithm used in Chinese chess programs to look for the best move of a position. It is a kind of depth-first search and always with a limited depth because the game tree is too huge to traverse in a limit time. When the search reaches a leaf node of the limited depth, it will evaluate the node and assign a score.

QS is a common technique used to evaluate leaf nodes for avoiding the horizon effect caused by depth limitation. Evaluating the leaf nodes only in the limited depth is very

dangerous because they may have *noisy* moves which may be able to cause a big change in deeper depths, such as check or capture moves. For example, a position that is evaluated as earning a pawn with the given depth may in fact lose a rook in the next move. In this situation, the evaluation function may inaccurately evaluate a position due to the depth limitation. Unlike α β search, QS looks for noisy moves and evaluates only quiescent (quiet) positions which have no noisy moves.

2.2 The evaluation function and features in CHIMO

The evaluation function is one of the important factors that highly influence the playing strength of Chinese chess programs. [Yen 2004] mentioned five kinds of approximate features considered in the evaluation component of Chinese chess programs, including (1) strength of piece, (2) location of piece, (3) mobility of piece, (4) threats and protection between pieces, and (5) dynamic adjustment according to the situation. CHIMO implemented (1) to (4), and king safety.

Strength of piece is the most common feature in evaluation. In general, the side that has more powerful pieces is more advantageous. Each piece is evaluated and given a score according to its contribution to the game. A rough estimate can be obtained by just looking up each side's pieces.

The location of each piece is evaluated according to its influence. Some locations are easier to attack from or defend against opponents, while others are harder, depending on the particular piece's rules for movement. Thus, two pieces of the same kind which occupy different locations need to be given different scores.

Mobility of piece is about how flexible a piece is. The more mobile a piece is, the more means of attack or defense it has. Thus a piece that has more choices when moving is more powerful generally. In practice, CHIMO only considers the mobility of rooks and horses. The mobility in CHIMO is evaluated linearly with respect to *mobility quantity*, the number of movable locations without being threatened.

Both attacking and attacked pieces are considered when evaluating threats and protection between pieces. Attacking pieces are under risk of being recaptured, so less powerful attacking pieces are evaluated higher, since this is essentially a trade between the attacking piece and the attacked piece. For the attacked pieces, it is obvious that more valuable pieces are evaluated higher.

Finally, king safety is also important because the main way to win the game is to capture the opponent's king. CHIMO not only considers the threatened locations around the king but also whether attacking pieces can check in the next ply.

2.3 The Bradley-Terry model and the MM algorithm

The Elo-rating system [Coulom 2007] is a rating method that calculates the relative strength between players and has been used for chess and other games. In the system, each player is evaluated and assigned a rating value which indicates his/her strength, and a player with stronger skill is evaluated with a higher rating value.

The Bradley-Terry model [Hunter 2004] is used to analyze the probabilities of paired competition results among individuals. One of its generalized versions can deal with competitions among teams of players with an additional assumption that the strength of a team is evaluated as the product of its members' strengths.

We analyze features using the generalized Bradley-Terry model. First, a feature is seen as an individual and multiple features in a single position are seen as a team. Thus, a feature weight is viewed as the strength of the individual, while position strength is viewed as the strength of the multiple features associated with the position. Second, choosing a move from a position is seen as a competition among all moves, and the move chosen by experts is the winner.

Based on the model, we can compute the likelihood that a move would also be chosen by experts, and the objective is to maximize this likelihood. The procedure of maximization can be seen as the program learning evaluation from expert game records. An MM algorithm [Coulom 2007], which adjusts each feature's strength iteratively with a minorizing function, is used to find maximum likelihood estimates in the Bradley–Terry model.

3. Learning Method

This section introduces the new features for learning and the evaluation method for candidate moves. Section 3.1 introduces new features added into CHIMO. In section 3.2, we apply QS to move evaluation to avoid the horizon effect.

3.1 New features and learning

Our objective is to adjust feature weights to improve the accuracy of the evaluation function. However, current feature weights in CHIMO are accurate enough due to years of manual adjustment. In order to distinguish from the original features, we design three extra features for experimentation, for which the weights are unknown. Finally, self-play between different versions of CHIMO is performed for comparison.

The new features are listed as follows:

- New mobility (NM): We change the way mobility of pieces are evaluated. The original approach where the evaluated score is linearly dependent on the mobility quantity (mentioned in 2.2) may not be accurate in reality. The new way to evaluate mobility is to look up mobility tables for different mobility quantities.
- Connected advisors/elephants (CAE): The advisor and the elephant are important defending pieces. A common strategy is to let advisors (or elephants) protect each other, referred to as connected advisors (or connected elephants). Because these two strategies share the same mechanism, we can refer to them collectively as sets of connected advisors/elephants (CAE). This feature is often used to prevent the king from being attacked directly by the opponent's attacking pieces.
- Protected river-crossed pawns (PRP): Pawns are allowed to move differently once they have crossed the river, similar to pawn promotion in chess. Therefore, river-crossed pawns (RP) are more powerful than those pawns that have not yet crossed and PRPs can create threats for more opponent pieces. However, unprotected RPs are more likely to be captured.

For each newly designed feature, we use the MM algorithm to update its weight. Every iteration of learning updates all feature weights once. Learning ends when all feature weights converge to its respective value. The adjustment allows the evaluation function to work more like experts within a collection of expert game records.

3.2 QS for learning

For reasons similar to the horizon effect, it is inaccurate for evaluation to take positions which are only one-depth away from the root into consideration because these positions may have noisy moves. For accuracy, we apply QS to evaluate candidate moves in our learning method.

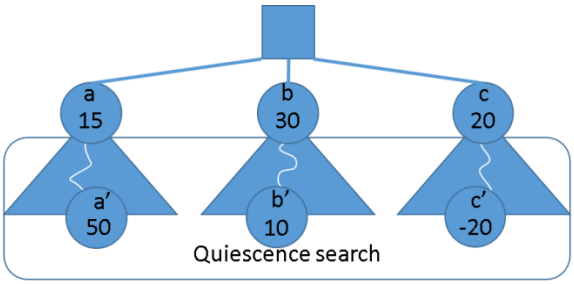

Figure 1. The evaluation of positions in game records.

When using QS to evaluate a move, we use the features of the leaf positions returned by QS for learning. As illustrated in Figure 1, a', b', and c' are the search results from QS. Therein, the leftmost move is chosen by experts. It is more reasonable that the competition (choosing a move) should be among a', b', and c' instead of a, b, and c.

4. Experiments and Results

We performed experiments for all the new features mentioned above. This section describes them and their results. Section 4.1 explains the experimental environment, including learning data, self-play, and some positions inappropriate for learning. Section 4.2 describes the experimental results in different learning schemes, including single, sequentially, and simultaneously. Section 4.3 is the comparison between using and not using QS for evaluating candidate moves.

4.1 Experimental environment

Expert game records are collected as learning data from many websites, including Xiangqi Arena [Movesky 2013]. The records are played by players with Elo-rating higher than 2350.

In our implementation, the iterations end when the update is less than 5%. After learning from game records, self-play between the learned version and the original version is conducted for comparison. The only difference between these two versions is the evaluation function. We chose 107 familiar testing positions from our opening database to initiate self-play. Each version of CHIMO played each position twice, taking red and black sides once respectively for a total of 214 played games. The game results are recorded and the win ratio is defined as follows.

$$\frac{\text{win}+\text{draw}\times 0.5}{\text{total}=(\text{win}+\text{loss}+\text{draw})}$$

Some positions of game records are inappropriate for learning because they are too unstable to be precisely evaluated by the evaluation function. In such positions, some moves can cause a win or loss and the evaluation function cannot account for that. The inappropriate positions which may cause noise are removed from learning.

For example, checking or checked positions are removed because particular moves are forced regardless of the evaluation function. For checking positions, it is obvious that the most prioritized moves are those that capture the opponent's king. On the other hand, in checked positions, the most prioritized moves are those that prevent the king from being captured.

4.2 Experimental results

In our experiments, features can be added sequentially or simultaneously. First, we added each feature individually. The self-play results are listed in Table 1 and the W-L-D column presents the game results against the original version for win, loss, and draw respectively. The enhancement of the win ratio of NM is more apparent than others.

Features	W-L-D	Win ratio
NM	68-41-105	56.3%
CAE	44-42-128	50.5%
PRP	54-41-119	53.0%

Table 1. Self-play results of learning single features.

Next, we tried adding features sequentially, as listed in Table 2. For example, for the learning sequence <NM, CAE, PRP>, the feature NM is learned first, then CAE, and finally PRP.

Feature <NM, CAE> has the best result within Table 2 and outperforms even <NM, CAE, PRP>. The result shows that adding new features one by one does not necessarily improve the win ratio. This may be due to interaction between features. For example, a piece aiming to protect other pieces may need to sacrifice mobility to do so.

Features	W-L-D	Win ratio
<NM, CAE>	72-34-108	58.8%
<NM, CAE, PRP>	63-42-109	54.9%
<NM, PRP>	61-46-107	53.5%
<NM, PRP, CAE>	59-36-119	55.4%
<CAE, NM>	55-44-114	52.3%
<CAE, NM, PRP>	65-40-109	55.8%
<CAE, PRP>	52-58-104	48.6%
<CAE, PRP, NM>	70-39-105	57.2%
<PRP, NM>	68-49-97	54.4%
<PRP, NM, CAE>	55-42-117	53.0%
<PRP, CAE>	59-38-117	54.9%
<PRP, CAE, NM>	74-52-88	55.1%

Table 2. Self-play results of learning features sequentially.

Finally, we tried learning combinations of two or three new features simultaneously. The self-play results are listed in Table 3. The combination of {NM, PRP} outperforms other combinations of two and the combination of all three achieved the best result than other experiments in this paper. This implies that simultaneous learning of feature weights is superior.

Features	W-L-D	Win ratio
{NM, CAE}	67-51-96	53.7%
{NM, PRP}	72-41-101	57.2%
{CAE, PRP}	53-57-104	49.1%
{NM, CAE, PRP}	89-32-93	63.3%

Table 3. Self-play results of learning features simultaneously.

4.3 Experiments for QS

In order to validate that QS is better when evaluating candidate moves, we tried two versions of learning. One used QS and another used only the evaluation function. The results in Table 4

show that using QS is better. Moreover, the result of using just the evaluation function is even worse than the original version in some cases.

Features	QS		evaluation function	
	W-L-D	Win ratio	W-L-D	Win ratio
{NM}	68-41-105	56.3%	41-123-50	30.8%
{CAE}	44-42-128	50.5%	43-59-112	46.3%
{PRP}	54-41-119	53.0%	52-57-105	48.8%
{NM, CAE, PRP}	89-32-93	61.7%	57-43-114	53.3%

Table 4. Self-play results of using QS.

5. Discussion

The experiment for the feature NM obtained, in general, higher win ratios. Intuitively, higher mobility results in a higher score, and higher mobility should always be better than lower mobility. Due to the marginal effect, a piece that already has enough mobility does not need more mobility.

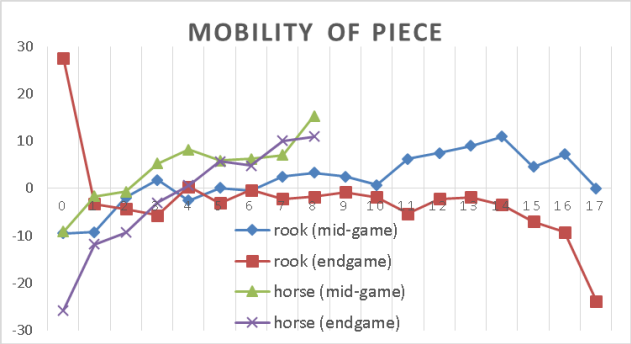

Figure 2. Mobility of piece in mid-game and endgame.

Figure 2 shows the learned weights (Y-axis) for NM in different mobility quantities (X-axis, at most 17 and 8 for rook and horse respectively). The case where the mobility of rook is zero in endgame is unusual because the situation rarely occurs in game records (too few to learn). This exception aside, the other weights (especially horse mobility) fit the tendency mentioned above roughly. However, surprisingly, the rook, with the highest mobility, earns relatively lower score. We think the reason is that the rook is the most powerful piece in Chinese chess and human players usually use it to attack or defend instead of just keeping it flexible with high mobility.

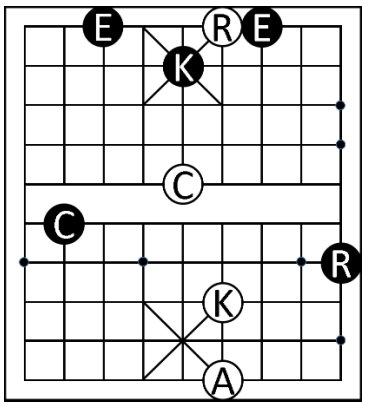

Figure 3. An example for rook mobility.

Figure 3 shows an example in endgame. The black rook can move to locations (dot marks) to get highest mobility. However, such moves gain no advantage and waste a valuable ply. In this case, most experts will not choose these moves, and consequently, the feature weights for high rook mobility quantities are decreased. This example also shows that rooks during the endgame have more opportunities to obtain highest mobility. Moreover, the red rook with a lower mobility is actually more valuable than the black one with higher mobility because it threatens to capture the two black elephants.

6. Conclusion

This paper applied the Bradley-Terry model to expert game records and used the Elo-rating system to calculate feature weights. Based on the MM algorithm, this paper proposed a supervised learning method to automatically adjust feature weights for Chinese chess programs. To solve the horizon effect, we use QS to avoid inaccuracy of candidate move evaluation.

We designed and implemented new features for learning and adjusted feature weights using the learning method proposed in this paper. For each new feature, CHIMO achieved different degrees of win ratio improvement, verified by self-play against the original version.

Through experiments, we discovered that learning features simultaneously is superior to sequentially. We also discovered that the best rook location is not the ones with the highest mobility, which may seem counter-intuitive at first. A comparison was made between using QS and using just the evaluation function. The learning result is more accurate when using positions returned from QS than those in one-depth.

After learning all new features, CHIMO achieved a win ratio of 61.7% against the original version. This implies the learning method can improve the playing strength of our Chinese chess program.

References

[Allis 1994] Allis, L.V., Searching for Solutions in Games and Artificial Intelligence, Ph.D. Thesis, University of Limburg, Maastricht, The Netherlands, 1994.

[Campbell 2002] Campbell, M., Hoane, Jr., A.J., and Hsu, F.-H., Deep Blue, Artificial Intelligence, 134:47-83, 2002.

[Cheng 2006] Cheng, M.-C., The Chess Location Evaluation Tables of Chinese Chess Program, Master Thesis, Department of Engineering Science, National Cheng Kung University, 2006.

[CBCGAC 2013] China Board and Card Games Administrative Center (CBCGAC), Xiangqi – the 1st world mind sports games, available at http://2008wmsg.chinaqiyuan.com/en/others/2008-09-27/1642019.html, March 2013.

[Coulom 2007] Coulom, R., Computing Elo Ratings of Move Patterns in the Game of Go, ICGA Journal, 30(4):198–208, 2007.

[Herik 2002] Herik, H.J. van den, Uiterwijk, J.W.H.M., and Rijswijck, J. van, Games solved: Now and in the future, Artificial Intelligence, 134:277–311, 2002.

[Hunter 2004] Hunter, D.R., MM algorithms for generalized Bradley-Terry models, The Annals of Statistics, 32(1):384–406, 2004.

[Kao 2009] Kao W.-L., The Automatically Tuning System of Evaluation Function for Computer Chinese Chess, Master Thesis, Department of Computer Science, National Chiao Tung University, 2009.

[Knuth 1975] Knuth, D.E., and Moore, R.W., An Analysis of Alpha-Beta Pruning, Artificial Intelligence, 6:293-326, 1975.

[Marsland 1992] Marsland, T.A., Computer Chess and Search. S. Shapiro (Ed.), Encyclopedia of Artificial Intelligence (2nd ed.), Wiley, New York, pp. 224–241, 1992.

[Movesky 2013] Movesky Inc., Xiangqi Arena (弈天棋缘), available at http://www.chesssky.net/, March 2013.

[Yen 2004] Yen, S.-J., Chen, J.-C, Yang, T.-N., and Hsu, S.-C.. Computer Chinese Chess, ICGA Journal, 27(1):3–18, 2004.