Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence

Optimizing Objective Function Parameters for Strength in Computer Game-Playing

Yoshikuni Sato
Graduate School of Systems and Information Engineering, University of Tsukuba
Advanced Research Technology Laboratories, Panasonic Corporation

Makoto Miwa
NaCTeM and School of Computer Science, University of Manchester

Shogo Takeuchi
JST ERATO Minato Discrete Structure Manipulation System Project

Daisuke Takahashi
Faculty of Engineering, Information and Systems, University of Tsukuba

Abstract

The learning of evaluation functions from game records has been widely studied in the field of computer game-playing. Conventional learning methods optimize the evaluation function parameters by using the game records of expert players in order to imitate their plays. Such conventional methods utilize objective functions to increase the agreement between the moves selected by game-playing programs and the moves in the records of actual games. The methods, however, have a problem in that increasing the agreement does not always improve the strength of a program. Indeed, it is not clear how this agreement relates to the strength of a trained program. To address this problem, this paper presents a learning method to optimize objective function parameters for strength in game-playing. The proposed method employs an evolutionary learning algorithm with the strengths (Elo ratings) of programs as their fitness scores. Experimental results show that the proposed method is effective since programs using the objective function produced by the proposed method are superior to those using conventional objective functions.

Introduction

Learning from expert demonstrations has been widely studied in artificial intelligence (Abbeel and Ng 2004; Argall et al. 2009). In the field of computer game-playing, learning from experts' play (game records) has achieved great success in various two-player games, including Othello (Buro 2002), Chess (Tesauro 2001), Shogi (Japanese Chess) (Hoki and Kaneko 2011; Tsuruoka, Yokoyama, and Chikayama 2002), and Go (Coulom 2007). In particular, the automatic tuning of evaluation functions has been one of the most important topics, since evaluation functions are one of the crucial factors in determining the strength of computer game-playing programs.

Comparison training has been successful for tuning the evaluation functions for the games of Chess (Tesauro 2001) and Shogi (Hoki and Kaneko 2011; Kaneko and Hoki 2011). In order to imitate the moves of expert players, the training method attempts to maximize the objective functions that represent the difference between the value of the expert's move and the values of other legal moves. IBM DEEP BLUE incorporated the method for optimizing the king-safety parameters and defeated a world champion in Chess (Tesauro 2001; Campbell, Hoane Jr, and Hsu 2002). The strength of state-of-the-art Shogi programs that use this method is now close to that of professional players (Hoki and Kaneko 2011).

The training method, however, has a drawback in that it is difficult to define an objective function that generates a strong evaluation function. Existing objective functions used in comparison training aim to imitate expert play, so the relationship between the objective functions and the strength of the generated programs is unclear. In addition, the objective functions do not consider the contribution of each move to the outcome (win/lose/draw) of the game. Thus, maximizing the degree to which the moves imitate experts' moves does not always mean better playing (Hsu et al. 1990; Tesauro 2001).

In this paper, we propose a learning method that optimizes the parameters of the objective function in order to maximize the strength of the generated programs. Our method introduces an additional optimization level, optimizing the objective function itself for strength, and trains an evaluation function using the optimized objective function. We optimize the objective function parameters by an evolutionary algorithm, continuous population-based incremental learning (PBILc) (Sebag and Ducoulombier 1998). We utilize the strength (Elo ratings) of each program, as estimated by many games (Coulom 2007), as its fitness score in PBILc. A strong program then learns by using the optimized objective function.

The rest of this paper is organized as follows. The next section describes related work on evaluation functions learning from game records. We then introduce a novel method of optimizing the objective function parameters for the strength of the programs, and then present experimental results. Finally, we present our conclusions and discuss areas of future work.

Related Work

There are several studies of learning from demonstrations (imitation learning) in which agents are allowed to learn from experts' behavior. Several methods have been proposed in various fields of AI, such as robotics (Abbeel and Ng 2004), natural language processing (Neu and Szepesvári 2009), computer vision (He, Daumé III, and Eisner 2012), and games (Ross and Bagnell 2010).

In the field of computer game-playing, learning the parameters from the game records of expert players has been successful in several two-player games such as Othello (Buro 2002), Chess (Tesauro 2001), Shogi (Hoki and Kaneko 2011; Kaneko and Hoki 2011; Tsuruoka, Yokoyama, and Chikayama 2002), and Go (Coulom 2007). Most supervised learning methods utilize the moves in the game records or the outcome of the games (win/draw/lose).

Most learning algorithms in game-playing methods generate appropriate training positions by using a game-tree search (Baxter, Tridgell, and Weaver 1998; Tesauro 2001; Takeuchi et al. 2007; Silver and Tesauro 2009; Hoki and Kaneko 2011).

We introduce two supervised learning methods that use experts' game records for the evaluation functions; our method is based on these. The first is a learning method that compares moves, and the second is a method that learns from evaluation of the position and the outcome of the recorded games.

Learning the Evaluation Functions by Comparing Moves

Tesauro (2001) proposed comparison training for learning the parameters of the evaluation functions. For supervision, this method uses the moves in the recorded games of expert players, and it incorporates a game-tree search in the learning process. Hoki and Kaneko (2011) extended the objective function and applied this learning method to Shogi; Hoki's program BONANZA, which employed this method, won the 16th World Computer Shogi Championship (2006). This method is widely employed in recent strong Shogi programs.

In the training process in BONANZA, in order to imitate the move selections of expert players, the evaluation function parameters are optimized by minimizing the following objective function:

$$J_1(P, \boldsymbol{v}) = \sum_{p \in P} \sum_{m=2}^{M_p} T\big(\xi(p_m, \boldsymbol{v}) - \xi(p_1, \boldsymbol{v})\big), \qquad (1)$$

where P is the set of all positions in the training records, M_p is the number of legal moves from the position p, p_m is the position that results from a move m in p, the move $m = 1$ denotes an expert player's move in p, $\xi(p_m, \boldsymbol{v})$ is the evaluation value that results from a game-tree search of position p_m, and $\boldsymbol{v}$ contains the features of the position being evaluated, i.e., the leaf position reached by the best play from both sides. $T(x)$ is a sigmoid function, $T(x) = \frac{1}{1+\exp(-a \cdot x)}$, and the parameter $a > 0$ controls the slope of the sigmoid function.

Adjustment of Evaluation Functions by Positional Evaluation and the Probability of Winning

Takeuchi et al. (2007) proposed a method to adjust the evaluation functions by using the relationship between the positional evaluation values and the probability of winning. The probability is computed from the training records. This method uses a logistic regression model to predict the probability of winning for a position from its evaluation value. Let $g(x)$ be a sigmoid function: $g(x) = \frac{1}{1+(\exp(-b \cdot x))}$. Here, x is the evaluation value of the training position, and b is a constant parameter that normalizes the evaluation function. The likelihood of a training position p is defined as

$$likelihood(p, y_p) = g\big(\xi(p_1, \boldsymbol{v})\big)^{y_p} \big(1 - g(\xi(p_1, \boldsymbol{v}))\big)^{(1-y_p)},$$

where y_p is the outcome of the training position p, whose value is 1 (0) if black (white) wins.

The method adjusts the evaluation function parameters by minimizing the sum of the negative log likelihood of all training positions:

$$J_2(P, \boldsymbol{v}) = - \sum_{p \in P} \Big\{ y_p \cdot \log\big(g(\xi(p_1, \boldsymbol{v}))\big) + (1 - y_p) \cdot \log\big(1 - g(\xi(p_1, \boldsymbol{v}))\big) \Big\}. \qquad (2)$$

This adjustment enables the evaluation function to predict the probability of winning for the positions being evaluated.

The proposed method is able to adjust evaluation function parameters that have already been optimized to some extent (e.g., by comparison training), but it is not able to optimize the parameters from scratch, since each game record has only one outcome and it is difficult to obtain enough training data. Methods that use only the outcomes of games have produced strong programs for games such as Othello (Buro 2002), but not for more complicated games such as Chess or Shogi.

Method

As stated above, the learning of evaluation functions from the records of expert players has been successful in several games.

However, such methods aim to imitate expert players but ignore the relationship between this imitation and the strength of the resulting program. Specifically, the shape of the sigmoid function in Equation (1) is arbitrary, and it is not known what that shape contributes to the strength of the program. These kinds of parameters in objective functions are still optimized manually, and this manual optimization is one of the factors that determine the strength of the programs.

Moreover, different training positions have different characteristics, such as the number of legal moves, the difficulty of determining the correct moves, or the impact of selecting the correct move on the outcome. In order to learn the parameters of the objective function, it is necessary to take account of these characteristics of the training positions.

To resolve these problems, we propose a novel method to optimize the parameters of the objective function. We selected Shogi as the target game. We first designed a new

objective function by extending and combining two different objective functions (Equations (1) and (2)), then optimized the parameters of the new objective function. Finally, we trained the evaluation function by using the optimized objective function. We note that our method is generally applicable to other two-player games, and also that the new objective function can be extended by incorporating other objective functions. We will, however, leave these extensions for future work.

Designing a New Parameterized Objective Function

We designed a new objective function by combining two objective functions (Equations (1) and (2)) and parameterizing it by categories of the training positions.

We first extended the objective function (Equation (1)) with parameters. Equation (1) is the most common objective function for learning evaluation functions in Shogi. We added three parameters to the sigmoid function in Equation (1) to make it possible to express the relationship between the objective function and the strength of the generated programs:

$$T(x) = \frac{W_{3,p}}{1 + \exp(-W_{1,p} \cdot (x + W_{2,p}))}, \tag{3}$$

where $W_{1,p}$, $W_{2,p}$, and $W_{3,p}$ are parameters that control the sigmoid function based on the category to which the training position p belongs. $W_{1,p}$ controls the slope of the sigmoid function, $W_{2,p}$ controls the margin between the correct move and the other legal moves, and $W_{3,p}$ is the weight of each training position.

We then combine two objective functions that have different characteristics. Equation (1) increases the agreement between the moves selected by programs and the moves (by expert players) in the game records, and Equation (2) adjusts the positional evaluation values for representing the probability of winning. Both are known to be useful for optimizing the evaluation function parameters. We thus decided to build a new objective function by combining Equation (1) (with Equation (3)) and Equation (2).

In order to combine the two objective functions, and since we do not know how the functions contribute to the strength of the program, we added a weight parameter to Equation (2), as follows:

$$J_2'(P, \boldsymbol{v}) = -\sum_{p \in P} W_{4,p} \Big\{ y_p \cdot \log\big(g(\xi(p_1, \boldsymbol{v}))\big) + (1 - y_p) \cdot \log\big(1 - g(\xi(p_1, \boldsymbol{v}))\big) \Big\}, \tag{4}$$

where $W_{4,p}$ is a parameter that controls the weighting of each training position.

The new objective function formed from the combination of the two previous objective functions is as follows:

$$J(P, \boldsymbol{v}) = J_1(P, \boldsymbol{v}) + J_2'(P, \boldsymbol{v}) + w_0 \cdot |\boldsymbol{v}|, \tag{5}$$

where w_0 is the weight of L1 regularization.

Our method optimizes $W_{1,p}$, $W_{2,p}$, $W_{3,p}$, $W_{4,p}$, and w_0 in Equation (5) in order to determine which learning parameters result in the greatest strength.

The parameters $W_{i,p}$ $(i = 1, 2, 3, 4)$ are calculated by the product of the weights $w_{i,j}$ of the categories to which each training position belongs:

$$W_{i,p} = c_i \prod_{j \in C_p} w_{i,j}, \tag{6}$$

where C_p is the set of categories to which training position p belongs, and $w_{i,j}$ is the parameter value of category j. c_i is determined by a combination of previous work (Hoki and Kaneko 2011) and a rough grid search. ($c_1 = 0.0237$, $c_2 = 64$, $c_3 = 1$ and $c_4 = 16$).

We categorized the training positions by each of the following three classes, which denote the characteristics of the positions.

- Progress Ratio

  The progress ratio is often used in classifying positions in game-playing (e.g., (Buro 2002)). We define the progress ratio as the ratio of the number of moves divided by the length of the game; eight categories are used for this class.

- Best-move Change

  The best-move change denotes how often the best (highest score) move changes as the depth of the search is incrementally increased. This best-move change is the same as the *best change* proposed by Heinz (1998). A position for which the best-move change is large is considered to be complex. This value is calculated by searching to the third level of depth, and so we have three categories (0, 1, and 2 exchanges) for this class.

- Correct Depth (Depth of search required in order to determine the correct move)

  A shallow search is used in the learning phase because it is infeasible to search deeply for all legal moves of all training positions. Thus, the selection of an incorrect move can be caused by an inaccurate evaluation as well as an insufficiently deep search. The former case is considered important for learning evaluation functions. Each training position is searched to measure the depth to select the correct move based on the classification of the position. To determine the correct depth for each training position, we conducted an alpha-beta search with three depths and thus have four categories (depth: 1, 2, 3, >3) for this class.

Optimization of Objective Functions

For optimizing the objective function parameters, we employed PBILc (Sebag and Ducoulombier 1998), one of the estimation of distribution algorithms (EDAs). PBILc generates the parameters of the objective functions for each individual computer game program, based on a Gaussian distribution $\mathcal{N}(X, \sigma^2)$. The center of the distribution X and the variance σ^2 are updated as follows:

$$X^{t+1} = (1 - \alpha) \cdot X^t + \alpha \cdot (X^{best,1} + X^{best,2} - X^{worst}) \tag{7}$$

$$\sigma_i^{t+1} = (1 - \alpha) \cdot \sigma_i^t + \alpha \cdot \sqrt{\frac{\sum_{j=1}^{K} (X_i^j - \overline{X}_i)^2}{K}}, \tag{8}$$

where $X^{best,1}$, $X^{best,2}$, and X^{worst} are the best, second best, and worst individuals, respectively, and α $(0 < \alpha < 1)$

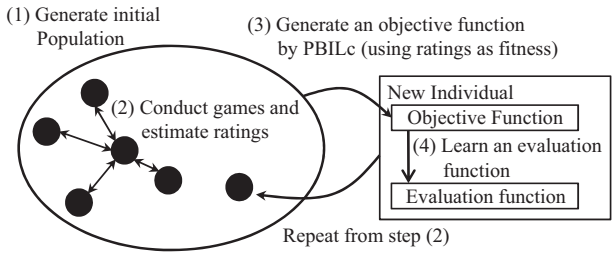

Figure 1: Procedure of Objective Function Parameters Optimization

is the learning rate. The parameters are updated quickly when α is large.

All individuals are ranked by their Elo ratings, which are estimated based on a series of competitions among the individuals (Coulom 2007). The Elo ratings denote the relative strengths of individuals and are employed as fitness scores in PBILc; this enables the optimization of the objective function parameters for generating a strong program.

Figure 1 shows the procedure for the optimization of objective function parameters with PBILc.

First, an initial population is generated using the parameters $w_{i,j}$ determined by a Gaussian random number $\mathcal{N}(1, 0.3)$. The evaluation function of each individual is learned by using a generated objective function. N individuals are generated by repeating this operation N times.

Second, the rating of each individual is estimated by playing games. A number of games are required to accurately estimate the strength of each individual. In our study, each individual played 4,000 games.

Third, a new individual is generated as follows. Objective function parameters are generated by PBILc, which uses Elo ratings as the fitness, and the evaluation function of a new individual is learned by using its objective function with the generated parameters. The weights of the evaluation function are trained by the gradient descent method (Hoki and Kaneko 2011).

Fourth, the new individual is exchanged with the worst individual in the population.

Steps (2) to (4) are repeated until the specified number of iterations is reached.

Results

Experimental Settings

The environment shown in Table 1 was used in the experiment. 10,000 game records of human expert players were used to learn an evaluation function. We employed an alpha-beta search-based Shogi program as our basis.

The features of the evaluation function were as follows:

- The values of the pieces,
- The positional relationship between each piece and a king,
- Common patterns of the positional relationships of the pieces.

CPU	Memory	Nodes
Xeon E5-2680 $\times$ 2	132GB	64
Core i7-3930K	16GB	7
Xeon X3440	8GB	8
Core2 Quad Q9650	8GB	10

Table 1: Experimental Environment

The total number of features was approximately 400,000. The values of the pieces were initialized to fixed values in order to avoid the problem of local minima (Hoki and Kaneko 2011) and to reduce the cost of the learning time. To calculate the value of $\xi(p_m, \boldsymbol{v})$ in the learning phase, we used a quiescence search (i.e., we searched positions where no winning tactical moves could be made) (Beal 1990).

The parameters of PBILc were set as follows: N (number of individuals) $= 50$, $\alpha = 0.0025$, $K = 10$, and the number of iterations $= 4,000$. Here, the parameter of the learning ratio α was set to this small value in order to avoid premature convergence because the proposed method exchanges the individuals one by one. K was fixed to 10 since PBILc is insensitive to K (Sebag and Ducoulombier 1998).

Evaluation by Play

We optimized an objective function (in Equation (5)) using the proposed method. The optimization took about 10 days using the settings given in the previous section.

To show the effectiveness of the proposed method, games were played between a program with an evaluation function learned by the proposed method and a program with one learned by conventional methods. We used two baseline programs for the games, and these were learned by using two different objective functions, as follows:

- **Comparison**: Equation (1),
- **Combination**: Equation (5).

L1 regularization was used in all cases. Comparison was learned by the conventional method most often used for Shogi as mentioned before, while Combination was learned by the simple combination of two objective functions. W_1, W_2, and W_3 were set to the values used in BONANZA ($W_1 = 0.0237$, $W_2 = 0$, $W_3 = 1$, and $w_0 = 0.00625$). W_4 was determined by a series of experiments ($W_4 = 16$ for both baseline programs).

Table 2 shows the results for a match between our method and the baseline programs; each program was given 100,000 nodes per move. Table 3 shows the results for a match between the two baseline programs, our method, and BONANZA version 6.0; we employed BONANZA since it is one of the strongest Shogi programs and is publicly available. The number of BONANZA's search nodes was set to 12,000 so that the strength was almost equal to that of Comparison (100,000 nodes). These tables show that a program with our method is superior to the two baseline programs, both in direct games (Table 2) and in games against BONANZA (Table 3). The results show that optimizing the parameters based on the categories of the positions is effective for obtaining an objective function that generates a strong pro-

	Win ratio of proposed method (Win-Loss-Draw)
vs Comparison	0.619 (1169-719-112) ***
vs Combination	0.576 (1063-782-155) ***

Table 2: Performance against Baseline Programs (* for p<0.05, ** for p<0.01, and *** for p<0.001.)

	Win ratio against BONANZA (Win-Loss-Draw)
Comparison	0.497 (975-987-38)
Combination	0.531 (1033-913-54) **
Proposed method	0.615 (1205-754-41) ***

Table 3: Performance against BONANZA (12,000 nodes) (* for p<0.05, ** for p<0.01, and *** for p<0.001.)

gram. We note that the results against Bonanza show that the evaluation functions using the optimized objective function do not overfit to the self-play situations where the objective function is optimized.

Objective Function Parameters

In this section, we analyze the weightings of the categories of the progress ratio, number of changes, and correct depth classes that were learned by the proposed method. We do not discuss the result for w_0 since the change was small (0.00625 to 0.00607).

Progress Ratio Figure 2 shows the weights for the categories of the progress ratio.

Parameter $w_{1,i}$ controls the slope of the sigmoid function. The result shows that $w_{1,i}$ becomes small as the progress ratio increases. This is because as the end of the game is approached, there is an increase in the differences between the evaluation values of the different moves. Employing a sigmoid function has the effect of excluding positions for which the differences of the evaluation values from the training data are too large, but simple employment of the function (without parameters) makes the effect the same for all positions. Our results show that in order to control the slope of the sigmoid function, our method assigns different weights to different categories.

The parameter $w_{2,i}$ controls the margin between the evaluation value of a correct move (one selected by an expert player) and those of the other legal moves. This parameter increases with the progress ratio, which emphasizes the differences between correct moves and the other moves in the endgame. The reason for this is that, in the opening game, there are usually moves other than correct moves that are also good moves. In the end game, however, moves other than the correct moves are often clearly bad (they lead to losing).

According to this, we can conclude that the optimized parameters $w_{1,i}$ and $w_{2,i}$ reasonably reflect the characteristics of the game of Shogi.

The shapes of the sigmoid functions of the progress ratio, which include the parameters $w_{1,i}$ and $w_{2,i}$, are shown

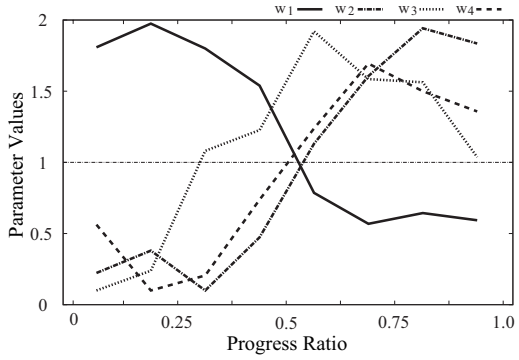

Figure 2: Parameter Values against Progress Ratio

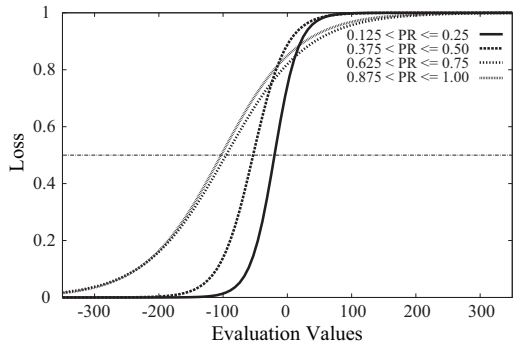

Figure 3: Shapes of the Sigmoid Functions (Progress Ratio: PR)

in Figure 3. We see that as the progress ratio increases, the graph has a shallower slope and its center shifts to the left.

The parameters $w_{3,i}$ and $w_{4,i}$ in Figure 2 are the weights of two basic objective functions, which express the agreement of the moves selected by the programs with those of experts and the accuracy of predicting the outcome by evaluation values, respectively. Both $w_{3,i}$ and $w_{4,i}$ increase with the progress ratio; the training position of the end game is more important than that of the opening game. This is because it is one of the characteristics of the games that the evaluation values in the end game often determine the outcome.

Best-move Change Figure 4 shows the values of the parameters for the categories of the best-move change.

The result shows that $w_{3,i}$ and $w_{4,i}$ decrease with the best-move change. The number of best-move changes is large when positions are not quiescent since such positions contain several important moves (e.g., capture, check, and escape), and the results of the search tend to be changed frequently. The decrease of weights $w_{3,i}$ and $w_{4,i}$ indicate which evaluation positions are the leaf nodes of the search that need to be quiescent in order to get a "strong" evaluation function.

Correct Depth Figure 5 shows the values of the parameters for the categories of correct depth. The result shows two

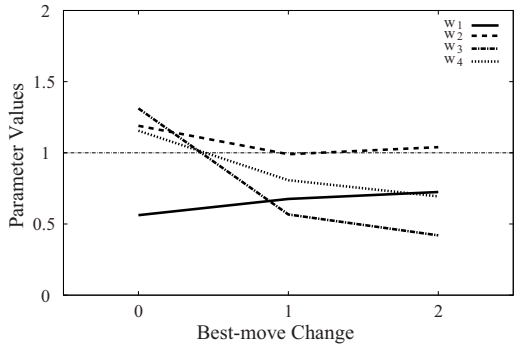

Figure 4: Parameter Values against Best-move Change

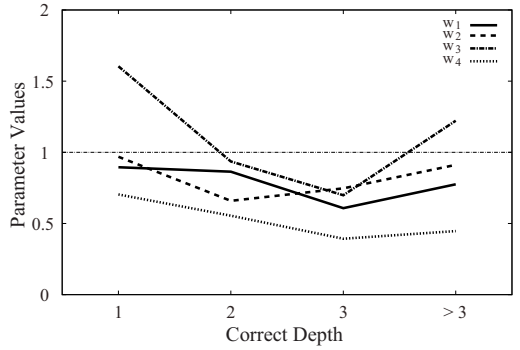

Figure 5: Parameter Values against Correct Depth

characteristics of $w_{3,i}$.

First, the weight is large for positions for which the correct depth is small, and this is because they are quiescent. They are thus appropriate for training. On the other hand, in positions with a higher value of correct depth, rather than optimizing the evaluation function, it is necessary to search more deeply in order to select the correct moves. Therefore, positions that have a small correct depth have a large weight.

Second, the weighting is also large for positions for which the program could not select the correct move. Cases in which programs cannot yield a correct move may be caused by one of two reasons: the evaluation function may be inaccurate, or the search depth may be inadequate. By giving a large weighting to positions for which programs are unable to select a correct move, the training positions with inaccurate evaluations are emphasized in the learning phase.

To summarize, the results for the correct depth show that the parameters learned by the proposed method are related to the reasons why the program cannot yield the correct move in the training position.

Reusability of the Learned Objective Function

The objective function learned by the proposed method is expected to be reusable and effective for different features of the evaluation function and training records, that were unseen in optimizing the objective function.

We conducted another experiment to investigate this reusability of the learned objective function. In this experiment, in which we used different features and different game records from those used in the earlier experiments, an evaluation function was learned by using 10,000 game records. The features of the evaluation function were as follows:

- The values of the pieces,
- The positional relationship of two kings and another piece,
- The positional relationship of a king and two other pieces.

These features were the same as those used in BONANZA.

As was done in the earlier experiments, the program learned the evaluation functions using both the proposed method and the conventional methods (Comparison and Combination) and then compared the results for 2,000 games.

Tables 4 and 5 show the results of the games. The setting of our programs were the same as in the previous experiments (100,000 search nodes). In this experiment, the program with the evaluation function learned by the proposed method was superior to the baseline programs. The results show the reusability of the proposed method; the objective function learned by the proposed method was also effective for different features of an evaluation function with different training game records. The reusability enables us to retrain the evaluation function using the optimized objective function when we modify the game program.

	Win ratio of proposed method (Win-Loss-Draw)
vs Comparison	0.592 (1107-762-131) ***
vs Combination	0.566 (1051-805-144) ***

Table 4: Results using Different Features and Game Records (* for $p<0.05$, ** for $p<0.01$, and *** for $p<0.001$.)

	Win ratio against BONANZA (Win-Loss-Draw)
Comparison	0.506 (989-965-46)
Combination	0.544 (1070-898-32) ***
Proposed method	0.587 (1151-810-39) ***

Table 5: Performance against BONANZA (15,000 nodes) using Different Features and Game Records (* for $p<0.05$, ** for $p<0.01$, and *** for $p<0.001$.)

Conclusions

This paper presented a method to optimize objective function parameters for learning an evaluation function that will generate a strong computer game program.

In conventional methods, there is a problem in that the relationship between the objective function and the strength of the generated program is not clear. To address this problem, we proposed a new method that optimizes the objective function parameters using an evolutionary algorithm, PBILc, so that the objective function can generate a "strong" evaluation function. The proposed method estimates the Elo

ratings of individuals based on the results of a number of games and then uses the ratings as a fitness score. The ratings enable the creation of an objective function that can produce strong computer game programs.

Experimental results for the game of Shogi showed that the programs learned by the proposed method were superior to those learned by conventional methods. We also confirmed that the parameters of the optimized objective function were dependent on the positional categories. Furthermore, the objective function learned by the proposed method was reusable for cases in which the features of the evaluation function and the training game records were different. This result shows that we can retrain a slightly modified game program with the same objective function.

As an area of future work, we are planning to apply our proposed method to other areas of learning in game programming. The proposed method is a strength-based learning method; thus, it will be applicable to various types of learning problems. For example, in the Monte Carlo tree search, which is often used in the game of Go (Coulom 2006), it is not known which simulation (playout) policies generate the strongest programs. The proposed method is obviously suitable to this type of problem, in which the objectives for realizing a strong program are unknown.

Acknowledgments

We would like to thank Yoshimasa Tsuruoka and the anonymous reviewers for their valuable comments on this paper. The computation was mainly carried out using the computer facilities at Research Institute for Information Technology, Kyushu University.

References

Abbeel, P., and Ng, A. Y. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-first International Conference on Machine Learning*.

Argall, B.; Chernova, S.; Veloso, M.; and Browning, B. 2009. A survey of robot learning from demonstration. *Robotics and Autonomous Systems* 67:469–483.

Baxter, J.; Tridgell, A.; and Weaver, L. 1998. Knightcap: A chess program that learns by combining td(λ) with game-tree search. In *Proceedings of the 15th International Conference on Machine Learning*, 28–36. Morgan Kaufmann.

Beal, D. F. 1990. A generalised quiescence search algorithm. *Artificial Intelligence* 43(1):85–98.

Buro, M. 2002. Improving heuristic mini-max search by supervised learning. *Artificial Intelligence* 85–99.

Campbell, M.; Hoane Jr, A. J.; and Hsu, F.-h. 2002. Deep blue. *Artificial Intelligence* 134(1-2):57–83.

Coulom, R. 2006. Efficient selectivity and backup operators in monte-carlo tree search. In *5th International Conference on Computer and Games*, 72–83.

Coulom, R. 2007. Computing elo ratings of move patterns in the game of Go. *ICGA Journal* 30(4):198–208.

He, H.; Daumé III, H.; and Eisner, J. 2012. Imitation learning by coaching. In *Advances in Neural Information Processing Systems 25*.

Heinz, E. A. 1998. Darkthought goes deep. *ICCA Journal* 21(4):228–229.

Hoki, K., and Kaneko, T. 2011. The global landscape of objective functions for the optimization of shogi piece values with a game-tree search. In *ACG*, volume 7168 of *Lecture Notes in Computer Science*, 184–195.

Hsu, F.-h.; Anantharaman, T. S.; Campbell, M. S.; and Nowatzyk, A. G. 1990. Deep Thought. In *Computers, Chess and Cognition*, 55–78.

Kaneko, T., and Hoki, K. 2011. Analysis of evaluation-function learning by comparison of sibling nodes. In *ACG*, volume 7168 of *Lecture Notes in Computer Science*, 158–169.

Neu, G., and Szepesvári, C. 2009. Training parsers by inverse reinforcement learning. *Machine Learning* 77:303–337.

Ross, S., and Bagnell, D. 2010. Efficient reductions for imitation learning. *Journal of Machine Learning Research - Proceedings Track* 9:661–668.

Sebag, M., and Ducoulombier, A. 1998. Extending population-based incremental learning to continuous search spaces. In *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, 418–427.

Silver, D., and Tesauro, G. 2009. Monte-carlo simulation balancing. In *Proceedings of the 26th International Conference on Machine Learning*, 945–952.

Takeuchi, S.; Kaneko, T.; Kazunori, Y.; and Kawai, S. 2007. Visualization and adjustment of evaluation functions based on evaluation values and win probability. In *AAAI07*, 858–863.

Tesauro, G. 2001. Comparison training of chess evaluation functions. In *Machines that learn to play games*, 117–130. Nova Science Publishers, Inc.

Tsuruoka, Y.; Yokoyama, D.; and Chikayama, T. 2002. Game-tree search algorithm based on realization probability. *ICGA Journal* 25(3):145–152.