Computer Engineering and Applications Vol. 2, No. 1, March 2013

On Constructing Static Evaluation Function using Temporal Difference Learning

Samuel Choi Ping Man
The Open University of Hong Kong
email: schoi@ouhk.edu.hk

ABSTRAKSI

Komputer pemrograman untuk bermain permainan papan melawan pemain manusia telah lama digunakan sebagai ukuran untuk pengembangan kecerdasan buatan. Pendekatan standar untuk bermain game komputer adalah untuk mencari langkah terbaik dari bagian permainan yang diberikan dengan menggunakan pencarian minimax dengan fungsi evaluasi statis. Fungsi evaluasi statis penting untuk kinerja memainkan game namun desain sering mengandalkan pemain ahli manusia. Makalah ini membahas bagaimana perbedaan sementara (TD) belajar dapat digunakan untuk membangun sebuah fungsi evaluasi statis melalui bermain sendiri dan mengevaluasi dampak untuk berbagai pengaturan parameter . Permainan Kalah , permainan non - kesempatan kompleksitas moderat, terpilih sebagai tempat pengujian . Hasil empiris menunjukkan bahwa TD belajar sangat menjanjikan untuk membangun fungsi evaluasi yang baik untuk permainan akhir dan secara substansial dapat meningkatkan kinerja secara keseluruhan bermain game dalam mempelajari seluruh permainan .

Kata kunci: temporal perbedaan pembelajaran, pembelajaran penguatan , bermain game komputer , fungsi evaluasi statis , kecerdasan buatan .

ABSTRACT

Programming computers to play board games against human players has long been used as a measure for the development of artificial intelligence. The standard approach for computer game playing is to search for the best move from a given game state by using minimax search with static evaluation function. The static evaluation function is critical to the game playing performance but its design often relies on human expert players. This paper discusses how temporal differences (TD) learning can be used to construct a static evaluation function through self-playing and evaluates the effects for various parameter settings. The game of Kalah, a non-chance game of moderate complexity, is chosen as a testbed. The empirical result shows that TD learning is particularly promising for constructing a good evaluation function for the end games and can substantially improve the overall game playing performance in learning the entire game.

Keywords: Temporal Difference learning, reinforcement learning, computer game playing, static evaluation function, artificial intelligence.

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

1. INTRODUCTION

Programming computers to play games has a history as long as computers. It is not only fun in nature but also a challenge to develop a machine that has the ability to match or even surpass the intelligence of human in planning and reasoning. Computer game playing against human players has also been used as a measure for the development of artificial intelligence. It is a good testbed for machine learning as its performance measures are clear and well-defined. In the past, complex board games such as checkers, chess, Othello, backgammon and Go have been widely studied as a testing ground for various machine learning procedures [1-4].

The standard approach for computer game playing is to search for the best move from a given game state by using minimax search with static evaluation function. The idea is to consider all legal moves as a game proceeds and maximize one's winning chance by assuming the opponent always pick the move worst to the player. However, the search space in most games is of considerable size. It is impractical, if not impossible, to make a move based on the exhaustive search. Heuristic evaluation functions, which estimate one's possibility of winning for a given game configuration, are thus often used for the tip nodes of the partially generated game tree. In order for the program to have a strong level of play, it is important to have a good control strategy on both searching technique and static evaluation function.

Good evaluation functions are difficult to design, and they are often manually sketched by human experts. Expert players need to extract features which may contribute to the winning of the game, such as piece advantage, relative mobility, and center control. It is, however, even more difficult to combine these features together to yield a good approximation of a game state. Linear combination of weights and features is often used but it suffers from three problems. First, as most of the hand craft features are correlated, it is questionable that a good estimation can be obtained by a simple linear combination. Second, the linear combination has very limited computational power on solving real world problems. Third, it is difficult to determine the exact weights of the features even for human experts.

This paper focuses on the construction of good evaluation functions by using TD learning with self-playing.

2. LITERATURE REVIEW

Samuel Arthur was the first one who proposed a learning algorithm to adjust the weights of features for his checker program [1]. Through self-play, the algorithm learned the best weights which minimized the error of the linear combination of features. A few years later, he improved his algorithm by employing signature table, which adds non-linearity to the feature combination [2]. Though his checker program is one of the pioneers in machine learning as well as the state-of-art at that time, his algorithm is highly domain-dependent and thus difficult to reuse in other game playing program.

As oppose to the linear combination of features, Lee and Mahajan proposed a Bayesian learning method to optimize the process [3]. It belongs to the pattern matching approach which assumed multivariate normal distribution of data and the

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

known priori probability of each class. Lee and Mahajan also empirically showed that the gain of this function was about equivalent to two extra plies of search. While this approach sounds appealing, it requires considerable amount of training data labeled by human experts, and is in fact a variation of supervised learning.

In the past two decades, more and more people investigated learning methods on constructing evaluator based on supervised learning method, such as neurogammon [4]. It was designed to examine whether back-propagation might be useful in higher-level tasks which are currently tackled by expert systems and knowledge engineering approaches. Despite the success of these programs, supervised learning approach faces one serious problem – it requires massive labelled training data. Sometimes training data can be obtained from recorded games played by experts – not only it is tedious, but also suffers the problem of human error on imprecise score assignment for a given game configuration.

Tesauro applied TD method for the game of backgammon and obtained surprisingly good result [5]. With zero knowledge built in at the start of learning, i.e. given only a raw description of the board state, the network learned to play at a strong intermediate level through self-play. The program selected a move by generating all legal moves in a given position and picking the one with the highest value (it is equivalent to one-ply searching). With hand craft features added on, it is able to surpass neurogammon [4]. With 2-ply searching and 1,500,000 games being trained, it becomes competitive to the best human player in the world [4]. More recent advances in backgammon can be found in [].

Nicol Schraudolph et. al. [7] also tried to apply TD method to a more complex game - 9x9 Go. Their approach is to base the training on network architectures that reflect the spatial organization of both input and reinforcement signals on the Go board, and training protocols that provide exposure to competent play. They found these yield far better performance than undifferentiated networks trained by self-play alone. With domain-specific constraint like reflection, color reversal properties of the game, feature extractions are trained instead of extracted by human experts. By one-ply search, their program is able to defeat a commercial Go program at low-playing level. More recent advances in Go can be found in [8-9].

3. TEMPORAL DIFFERENCE LEARNING

Training game playing program is difficult and belongs to the class of delayed reinforcement learning paradigm, in which reward or penalty is delayed until the end of the game. As such, it is difficult to know which move in a sequence of moves makes a mistake and thus the temporal credit assignment problem must be solved so as to train the network properly. These methods are studied systematically by Sutton [10] and named Temporal Difference (TD) method. TD training method is a class of reinforcement learning algorithm which uses past experience with an incompletely known system to predict its future and is well suit for the game playing problems.

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

Specifically, a network is set up to observe a sequence of board position, x_1, x_2, …, x_t, resulting in a final reward signal z. In the simplest case the reward signal is z=1 if network wins and z=0 if the opponent wins. In this case the network's output Pt is an estimate of its probability of winning from board position x_t. The move selected at each time step is the moves that maximize P_t when the network to play.

TD training is based on the prediction difference between two temporally successive predictions and solves the credit assignment problem by using the following equation:

$$\Delta w_t = \alpha(P_{t+1} - P_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_k$$

where P_t is the network's output upon observation of input pattern x_t at time t, w is the vector of weights that parameterizes the network, and $\nabla_w P_k$ is the gradient of network output with respect to weights. This equation allows the weights to be updated incrementally. When λ = 1, TD learning is equivalent to supervised learning by the pairing of each input with the final reward. For the case of λ = 0, Sutton proved that TD always converges to the optimal predictions for a linear network and a linearly independent set of input patterns.

TD method overcomes the temporal credit assignment problem and makes the reinforcement training feasible. One potential advantage of TD method over supervised learning is that training data set is not fixed, and thus might be able to avoid the problem of overtraining and over fitting. In other words, the performance always improves with increasing training time, provided that the neural network can learn to represent the changes as instructed by TD learning. Though lack of theoretical support of convergence for non-linear network, Tesauro shows empirically the convergence of TD method in his backgammon program. It is interesting to see if the same result holds for other non-chance board games with similar complexity level.

4. GAME OF KALAH

The game of Kalah is a game with great ancient. Variously known as Mancala, Wari, or Awari, Kalah is a board game originally played in Africa, with stones placed in small pits dug in the ground. Though little written analysis exists, Kalah has been played at the drop of a stone by people of many cultures for centuries. Kalah's simple equipment and rules make it a natural for computer play. Details of the game rules are described in http://en.wikipedia.org/wiki/Kalah. The game of Kalah is chosen because of its simplicity of rules and the moderate complex level of search space (about 10^{12}), which is similar to the game of backgammon. Starting from 1989, Kalah competition is included in the International Computer Olympiad and the winner of the year is MACRO, which was built into substantial strategical knowledge with shallow searches. However, in the next year, it was being defeated by Lithidion, a program implemented with standard game playing techniques (α-β search) and omniscient endgame database, in the score of 5-0 [11]. It is believed that omniscient endgame database is the key for winning the game. It is interesting to see if TD learning can train a good evaluation function comparable to these programs.

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

5. IMPLEMENTATION

The complete implementation of the TD network program comprises about 1500 lines of C code, and is designed to be a tool to study TD learning in practice. Besides the network training capability, it also contains several functions which allow users to set different parameters and train the network. The basic functions includes loading a game board configuration, setting up parameters for both training and testing, conducting head-to-head competition against human or computer, plotting the testing result, etc. The program also saves the training network weight automatically for every 1000 epochs. The entire program can be divided roughly into two models: minimax model and Multi-Layer Perceptron (MLP) model, where they were developed and tested separately before they are combined together.

The minimax search model is used as a baseline in the experiment. Since Kalah is a game with small number of fan-outs, this sort of game playing problems is usually tackled by the traditional game tree searching approach with fairly good result. Using the minimax model, the performance of the TD network can be evaluated more objectively and accurately. In this paper, the model is built using standard minimax search with piece-advantage as the evaluator. Even with such a simple set up, this program can defeat an intermediate level human player with 7-ply look ahead search.

The MLP model is first built instead of TD model because it can be changed to TD model later with some modifications. This model is implemented with the momentum term and increasing the learning rate. The TD network being built has a feed-forward network structure with full connectivity from one layer to the next. Both single layer with no hidden units and multi-layer with a single hidden layer of varying number of units are also examined.

The representation of input and output data in multi-layer networks is a crucial factor in the success of both supervised learning and TD learning procedures. The network trained in this project contains 14 units to encode the game board configuration: 12 units to encode the number of stones contained in each pit on both side, 2 units to encode the number of stones captured in both Kalah. Before the data are fed into the network, the signal will be first normalized by dividing the number by total number of stones in the game. This representation ensures that the input range is between 0 and 1, which is more suitable for the network learning. However, since the number of stones in a pit is usually small, the input is often biased towards 0. One remedy is to multiply this value by 2, and leave the possibility of having an input value greater than 1.

It is also known that features relevant to good play always gives significantly better performance in training compared to raw board description, therefore hand craft features are also added to the network as input for the purpose of comparison. Five selected hand craft features are Kalah difference (Analog -1 to 1, 1 unit), vulnerable pits (Binary 0/1, 6 units on each side), attackable pits (Binary 0/1, 6 units

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

on each side), total stone difference (Analog -1 to 1, 1 units) and total number of stones left on both side (Analog 0 to 1, 1 unit).

Sigmoid functions are used for both hidden and output units in order to add in the non-linearity computation to the network. As with back-propagation, a certain amount of parameter tuning was required in order to obtain good results with the TD algorithm. For example, initial weights of the network should be small, parameters like λ and learning rate α should be adjusted heuristically for a given network and task. It is known that low learning rate improves the learning result, whereas high learning rate degrades performance. In this paper, α varies from 0.05 to 0.2, λ from 0 to 1, and initial weights from -0.333 to 0.333 are examined in order to select the best combination for the learning.

Unlike backgammon, Kalah is a non-chance game in which deterministic computer programs always give the same response for a given game board configuration. That is, the same set sequence of moves will be used if the network is trained by self-play. This will certainly cause the overfitting problem in the network. In order to make the training possible, controlled randomness are needed in the training so that the repetition of same sequence of moves is avoided. In short, the network is trained by random self-play.

For the performance measure, models are compared according to the result of head-to-head competitions. With the same problem facing in the training, it is necessary to have some methods to ensure different games are played in the head-to-head competitions. One possible solution is to have the games starting with various preset openings, each of which is of equal strength to both sides. In this case, there should be at least a few hundreds of these opening games in order to conduct a fair performance measure. It is, however, difficult to obtain so many opening games. Another alternative is to add randomness to the minimax model. In particular, if the minimax program obtains more than 1 best move with equal score, it will randomly pick one among them. This helps the program to have different games played and we can compare two programs' strength based on head-to-head competitions.

6. EMPIRICAL RESULTS

First, how the parameter setting affects learning is examined. In particular, the learning rate α, the value of λ, and the number of hidden units are studied. Then the best combination of these parameters is chosen for the next experiment.

6.1 THE EFFECT ON THE PARAMETERS

It is well known that the learning rate α has a great impact on the learning. A low learning rate will slow the learning process whereas a high learning rate might cause oscillation in performance, and thus the choice of α must be very careful. A rule of thumb to select the learning rate is to set α inversely proportional to the number of input units, and it is about 0.07 in our case. In the experiment, the network performance is examined for various choices of α after 10,000 epochs. It is found that the learning speed increases as α gets larger and larger, but the network performance begins to oscillate when the learning rate is larger 0.1. This result

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

matches roughly with the rule of thumb and so 0.1 is selected as a good learning rate to this particular problem.

The value of λ controls the learning behavior of TD algorithm. When λ = 0, it is equivalent to Q-learning [11] while when λ = 1, it is same as the supervised-learning. Sutton [10] stated that smaller λ values can learn much faster than supervised-learning methods because "they make more efficient use of their experience, converge more rapidly and make more accurate predictions along the way." However, the result we obtained is rather contradicted, where it seems that higher λ value tends to learn faster. This situation happens because TD assumes the prediction of the successor is more accurate than that of the predecessors, but it is generally not true in the early stage of learning. Tesauro suggested λ be varied as a function of learning time, initially be set to a large value then decrease gradually as the predictions become more accurate. [4]

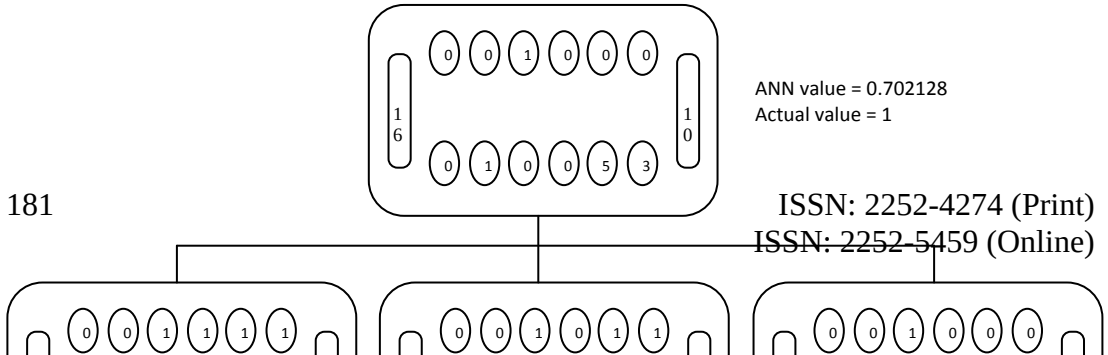
The last parameter to explore is the effect on the number of hidden units in learning. Tesauro [4] stated that due to the randomness of the training data, overfitting and overtraining will not occurred in TD learning, and hence adding more hidden units and longer training time to the network will always improve its performance. This statement is being confirmed in our experiment.

Based on the results of these experiments, the best combination of the parameters seems to be λ = 0.8, α = 0.1 and the number of hidden units = 30. Therefore this parameter setting is used in the following experiments.

6.2 LEARNING THE ENDGAME AND ENTIRE GAME

The next TD experiment is to learn the endgame positions, in which a game is to be finished within 10 moves in optimal play of both players. This is an exceedingly simple situation to consider but useful to verify the correctness of program as well as the capability of the TD learning. Another advantage of having this endgame learning is that the network output can be compared directly with the actual minimax score because the exhaustive search in the endgame becomes possible.

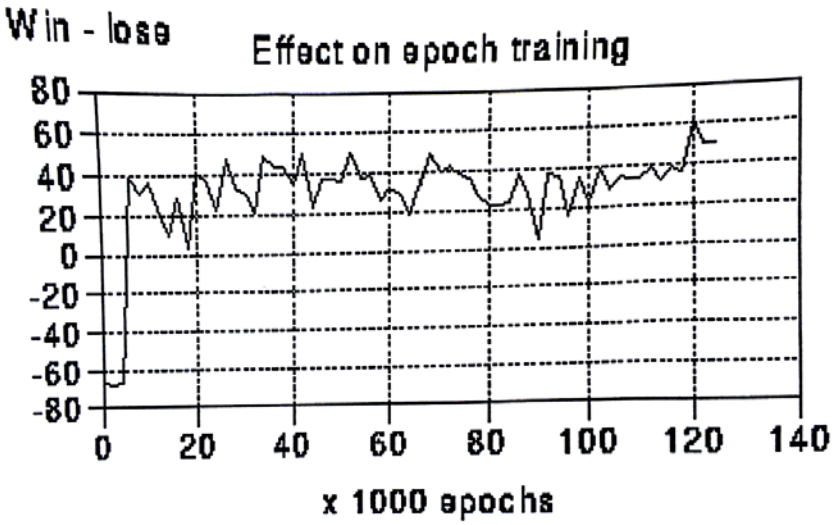
In this set up, south (ANN) should be able to win the game in 3 moves no matter whose turn it is. In traditional minimax search it requires 6-ply or above in order to make the correct sequence of moves. It is found that in TD learning, the network is capable to learn the correct moves in 7000 epoch training by self-play and the performance remains stable on further training. It is worth noting that even the network can play the game perfectly after the training, the value returned from the network is still far from being accurate as shown in the following Figure.



ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

This is because a heuristic evaluation function need not exactly represent the true values of states for correct move selection. Instead, it only needs to have the correct sign for the slope between pairs of points in order to make correct state preference.

The next logical step is to train the network playing the entire game. As one might expect, training for the entire game is far more difficult than for the endgame, since both the number of possible game states and the move sequence to learn are greatly increased. It is expected the number of epochs required will be in the order of hundred thousands. In fact, in the game of backgammon, which has approximate the same complexity as the game of Kalah, requires more than 200,000 games of training in order to reach an intermediate strong level of play.



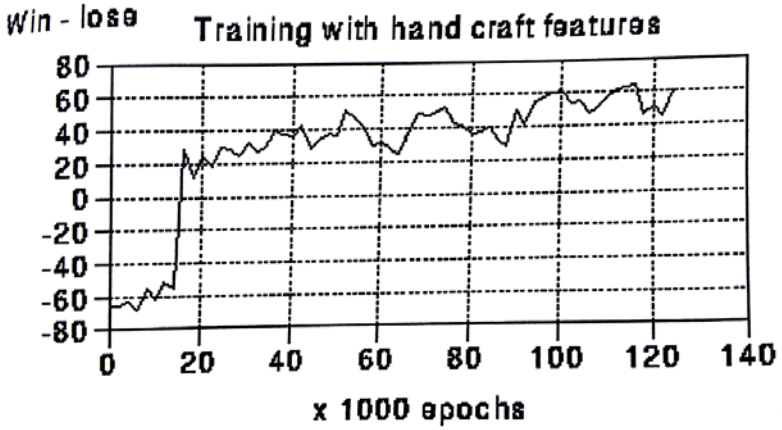
The figure above shows the performance of the network as the learning proceeds. Despite the fluctuation in performance, it can be seen that the learning is actually taken place. The performance of the network after 120,000 epochs of training is about equivalent to two extra ply level of search. The reason of the slow learning might be due to rather different strategies needed in different stages of game. For example, in the early engaged, it is important to build up strong pits to capture opponent's stones whereas near the endgame, it becomes more crucial to maximize

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

the number of stones and moves of oneself. It is questionable whether a monolithic network is capable to learn an adaptable strategy for different stages of game.

6.3 TRAIN WITH THE HAND CRAFT FEATURES

Network with hand craft features is also examined. It is found that the hand craft features gives better performance than the raw board data, although the learning becomes much slower due to the increase of the input and hidden units.



When playing against with the trained ANN network, one notices that the network has developed its own strategy of play. The network tends to pick the move on the pits nearest to its Kalah and capture opponent's stones whenever possible. This strategy keeps his moves as many as possible while minimize the opponent's, and keeps as many as stones on its side so that those stones goes into its Kalah when his opponent has no move left.

This is very similar to the strategy employed by human players in the endgame. This result is also consistent with the style of TD learning in which TD network predicts better near the end than the beginning. It also demonstrates the need of several neural networks to evaluate games at different stages.

7. CONCLUSION

This paper demonstrates how temporal differences (TD) learning can be employed to construct a static evaluation function through self-playing in a non-chance game. The game of Kalah, a non-chance game of moderate complexity, is chosen as a testbed. The empirical result shows that TD learning is particularly promising for constructing a good evaluation function for the end games and can substantially improve the overall game playing performance in learning the entire game. Although the network performance is yet to reach an advanced level of playing skill, significant learning has been shown and better result is expected to see for longer training period. The effects for various parameter settings are also

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)

evaluated and it is found that a suitable parameters setting will be $\lambda = 0.8$, $\alpha = 0.1$ with the number of hidden units = 30.

REFERENCES

[1] A. Samuel, “Some studies in machine learning using the game of checkers”, IBM J. of Research and Development, 3, 210-229, 1959.
[2] A. Samuel, “Some studies in machine learning using the game of checkers, II - recent progress”, IBM J. of Research and Development, 11, 601-617, 1967.
[3] K.F. Lee, & S. Mahajan, “A pattern classification approach to evaluation function learning”, Artificial Intelligence, 36, 1-25, 1988.
[4] G. Tesauro, “ Neurogammon: a neural network backgammon program”, IJCNN Proceedings, III, 33-39, 1990.
[5] G. Tesauro, “Practical issues in temporal difference learning”, Machine Learning 8, 257-278, 1992.
[6] M. A. Wiering, “Self-Play and Using an Expert to Learn to Play Backgammon with Temporal Difference Learning”, J. Intelligent Learning Systems & Applications, 2010, 2: 57-68.
[7] N.N.Schraudolph, P. Dayan, T. J. Sejnowski, “Temporal difference learning of position evaluation in the game of Go”, Advances in Neural Information Processing 6, 1994.
[8] S. Gelly and D. Silver, (2011). Monte-Carlo tree search and rapid action value estimation in computer Go. Artificial Intelligence, 175, 1856–1875.
[9] D. Silver, R. S. Sutton, M. Müller, “Temporal-difference search in computer Go”, Machine Learning (2012) 87:183–219.
[10] R.S. Sutton, “Learning to predict by the methods of temporal difference”, Machine Learning, 3, 9-44, 1988.
[11] D.N.L. Levy and D.F. Beal, “Heuristic programming in artificial intelligence 2”, the Second Olympiad, Ellis Horwood, 1990.
[12] C.J.C.H., Watkins, “Learning from Delayed Rewards”, Ph.D. thesis, Cambridge University, 1989.

ISSN: 2252-4274 (Print)
ISSN: 2252-5459 (Online)