The Application of TD(λ) Learning to the Opening Games of 19$\times$19 Go

Byung-Doo Lee
Dept. of Computer Science,
Univ. of Auckland,
Auckland, New Zealand
blee026@cs.auckland.ac.nz

Hans Werner Guesgen
Dept. of Computer Science,
Univ. of Auckland,
Auckland, New Zealand
hans@cs.auckland.ac.nz

Jacky Baltes
Dept. of Computer Science,
Univ. of Manitoba,
Winnipeg, Canada
jacky@cs.umanitoba.ca

Abstract

This paper describes the results of applying Temporal Difference (TD) learning with a network to the opening game problems in Go. The main difference from other research is that this experiment applied TD learning to the full-sized (19$\times$19) game of Go instead of a simple version (e.g., 9$\times$9 game). We discuss and compare TD(λ) learning for predicting an opening game's winning and for finding the best game among the prototypical professional opening games. We also tested the performance of TD(λ)s by playing against each other and against the commercial Go programs. The empirical result for picking the best game is promising, but there is no guarantee that TD(λ) will always pick the identical opening game independent of different λ values. The competition between two TD(λ)s shows that TD(λ) with a higher λ has better performance.

1. Introduction

Go is a board game for two players: black and white. Two players alternately play a single stone onto the 19$\times$19 intersections (i.e. 361 points). The goal in Go is to surround more territory than the opponent by enclosing some area.

Go is academically characterized as a perfect information, deterministic and zero-summed game between two players [2]. In recent years, the advent of new technology and better algorithms has led to increased interest in computer Go. Go is an excellent testbed for artificial intelligence (AI) research because of its vast search space and the importance of strategic and tactical decisions during the opening game.

The traditional AI approach when faced with a computer board game problem is to search the space of all possible moves to find a move sequence that leads to an advantageous position. Searching with α-β tree pruning or other more advanced search algorithms are frequently used. Although its algorithm works in theory, this approach does not work well in Go. That is, the complexity of the domain of Go is too large. In the early stages of the opening game, Go has a branching factor of over 200 [1], and the number of moves in a game is approximately between 250 and 300. This means that any naive search method is doomed to failure. In fact, the search spaces need to be strongly biased so that efficient problem solving strategies can be developed. One such approach is the use of machine learning in learning good sequence for the opening game of Go.

In this paper, we used TD(λ) learning (with a back-propagation neural network) to learn the board evaluation function. Using the board evaluation function, we analyzed several characteristics in the opening games.

2. TD(λ) Learning

The fundamental idea of TD stated by Richard Sutton is that learning is based on the difference between temporally successive predictions. The aim of TD in the game of Go is to evaluate the possible moves and to determine the best next move. In general, researchers randomly generate possible moves using special methods (e.g., Gibb's sampling [2] and a modified tree search [4]). In a real game of Go, the number of possible moves (branching factor) is too vast for these methods. To reduce the branching factor, simplified rules and scaled-down game boards (e.g., 9$\times$9 board) are frequently used by other researchers.

TD learning is a well-known reinforcement learning technique which is used for sequential prediction as well as sequential decision. Using scalar rewards and a supervised neural network, it can tune the function approximator.

In a game of Go, each state represents the current whole board position; i.e. it describes the positions of all stones that have been played so far (excluding captured stones).

When a sequence of states $< S_0, S_1, \cdots, S_t, \cdots, S_f >$ of the board is given, a network visits those states sequentially until the final state S_f is encountered. Here S_0 indicates the state of the initial empty board and f indexes the board situation after the final move. At the final state S_f, a utility value (e.g., reward or punishment) is received as a final reward z. A general approach to derive the general equation, which is a function of the state and the utility value, would be to pair each state with the final reward z to build a training sequence of the form $< (S_0, z), (S_1, z), \cdots, (S_t, z),$

$\cdots$, $(S_f, z) >$. A temporal difference method, by contrast, uses a training sequence such as $< (S_0, P_0)$, (S_1, P_1), $\cdots$, (S_t, P_t), $\cdots$, $(S_f, P_f) >$ where P_t represents the prediction generated by the network at time t.

Weights in a neural network are generally updated by the batch weight update or the intra-sequence weight update. In the batch weight update, the new weight matrix w_n is updated only once for each sequence of moves and thus the network will not change its weights until encountering the final move. After processing a sequence of moves, the weight w_n is updated by the sum of all the moves' increments:

$$w_n = w_o + \Delta w = w_o + \sum_{t=0}^{f} \Delta w_t \tag{1}$$

where w_o is the old weight matrix and f denotes the total number of states in a sequence of moves.

Using the delta rule the general equation of TD(λ) is given by:

$$\Delta w_t = \alpha(P_{t+1} - P_t) \sum_{k=0}^{t} \left[\lambda^{t-k} \cdot \frac{\partial P_k}{\partial w} \right] \tag{2}$$

where α is the learning rate and P_t is the prediction generated by the network at time t.

In TD(λ), the predictions of observation vectors, which occur k steps in the past, are weighted by λ^k where $\lambda \in$ [0,1]. The discount factor λ determines the amount of sensitivity in successive predictions. We tried four values for λ: λ = 0.0, 0.3, 0.7 and 1.0. Note that TD(1.0) is the prototypical supervised learning.

Finally the batch weight update is represented as:

$$w_n = w_o + \sum_{t=0}^{f} \left[\alpha(P_{t+1} - P_t) \sum_{k=0}^{t} \left[\lambda^{t-k} \cdot \frac{\partial P_k}{\partial w} \right] \right] \tag{3}$$

where P_{f+1} is the final reward z.

3. Implementation of TD(λ)

For training the network, the encoded board positions of each opening game were fed into a single-layered neural network (i.e. the 362-1 network as in Figure 1) with the tangent hyperbolic function as an activation function. The input data is a sequence of the form $< S_0, S_1, \cdots, S_{f-1}, S_f >$ for an opening game. The final reward z was put into the (f+1)-th column in the output layer to conduct as the (f+1)-th prediction P_{f+1}.

In Figure 1, the input data S_t at time t is composed of a sequence of moves for black and white: $< B_1, W_2, B_3, W_4, \cdots, W_{t-1}, B_t >$. X_n in the (n+1)-th column holds one value: 0 if there is no stone, +1 for a black stone and -1 for a white stone on the game board.

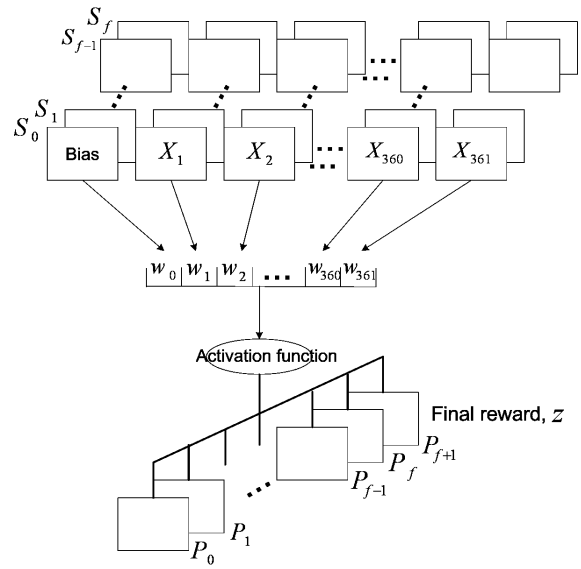

Figure 1: Neural network structure for TD learning.

Since there were no prototypical professional opening games, each of which has a final reward, we created a database of 213 professional opening games manually. The games were selected from [3]. We then generated 1,704 games by using the symmetries (i.e. reflection and rotation) of each game.

To analyze how TD(λ) learning works varying with the opening styles, we categorized 1,704 opening games as five styles in Table 1. We assigned final reward z, which was judged by professional players, as +1 for a game which is favorable to black, 0 for an even game, and -1 for a game which is favorable to white.

Opening styles	Favorable to			Total
	Black	White	Both	
Two star	144	72	136	352 (20.7%)
Three star	152	184	120	456 (26.8%)
Chinese	104	104	312	520 (30.5%)
Shusaku	24	72	152	248 (14.5%)
Misc.	16	40	72	128 (7.5%)
Total	440	472	792	1,704 (100%)

Table 1: Distribution of the training opening games.

4. Experimental Results

4.1. Generalizing the Network

To generalize TD(λ) learning with a network, we split the 1,704 opening games randomly into 1,136 for training data (66.6%) and 568 for validation data (33.3%).

The early stopping method using cross validation was used to avoid overfitting to the training data. The number of epochs was limited to 100 epochs. After applying the early stopping method, we obtained the number of epochs: 100 epochs for $\lambda = 1.0$ and $\lambda = 0.7$, and 25 epochs for $\lambda = 0.3$ and $\lambda = 0.0$.

We then retrained the network for the selected λ values with the original 1,706 games and each different epoch. RMSE curves in the dotted box in Figure 2 show that TD(λ) with higher λ has better performance in terms of RMSE.

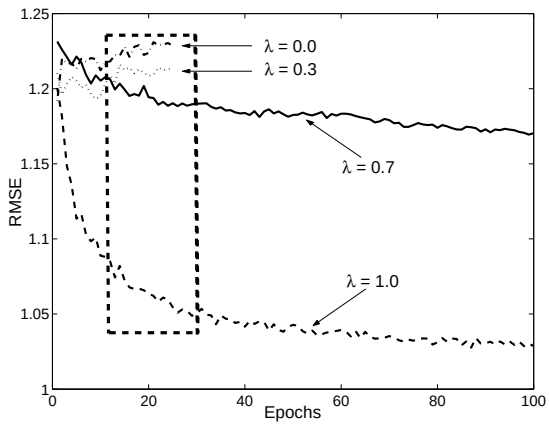

Figure 2: Plot of RMSE for the selected λ values with the corresponding epochs (α = 0.5).

4.2. Predicting an Opening Game's Winning

The network's prediction P_t in Figure 1 can be used as an evaluation function and indicates an estimate of the t-th move's winning probability from the current board position. That is, black is likely to win the game if P_1 is positive. If P_1 is negative, on the other hand, it means that black is likely to lose this game.

Since P_1 of a game is different for each opening style (i.e. the first move position), we calculated the mean of P_1 for the selected λ values from 1,704 games. The resulting set of λ and P_1 values is: $\{$(1.0,+0.98), (0.7,-0.06), (0.3,+0.49), (0.0,-0.01)$\}$.

Two TD(λ)s (with λ = 1.0 and 0.3) regard that the first player (black) has a high probability to win an opening game when the network is trained by these 1,704 opening games. We found that the network's first prediction P_1 differs with the different λ values.

We also analyzed what extent the network can predict each game's winning probability by comparing P_1 and the final reward z. If P_1 is greater than +0.5 and z is +1, we increased the number of the winning count. If P_1 is less than -0.5 and z is -1, we increased the number of the losing count. Meanwhile, if P_1 is between -0.5 and +0.5 and z is 0, we increased the number of the even count.

Table 2 shows that the network predicts 544 games correctly among 1,704 games when λ is 0.7.

Opening styles	Hit count as			Total
	Winning	Losing	Even	
Two star	36/144	36/72	34/136	106/352 (30.1%)
Three star	38/152	92/184	30/120	160/456 (35.1%)
Chinese	26/104	52/104	78/312	156/520 (30.0%)
$Shusaku$	6/24	36/72	38/152	80/248 (32.2%)
Misc.	4/16	20/40	18/72	42/128 (32.8%)
Total	110/440	236/472	198/792	544/1,704 (31.9%)

Table 2: Correctness between the prediction P_1 and the final reward z. (λ = 0.7 and α = 0.5)

We found that TD learning with a neural network is strongly distorted by the composed shapes of the training data rather than the final rewards, and thus the network can not precisely predict each game's winning.

4.3. Finding the Best Opening Game

Human players do not know which game is the best opening game, which is the most favorable to black, among the 1,704 opening games because they can not evaluate all sequences of the games precisely. But we can pick the best opening game by comparing the prediction P_t of the game tree with the simple forward Minimax method. The simple forward Minimax method would select the best opening game which holds the maximum P_t when black plays and the minimum P_t when white plays, at each time step.

Table 3 shows that TD(λ) picks the different best opening game with the different λ values. The prediction P_0 in Table 3 shows that white has a higher probability to win an opening game when the network is trained with TD(0.0) and TD(0.7). On the other hand, black has a high probability to win an opening game when the network is trained with λ = 0.3. When λ is 0.3, we can see that white defends well against black's playing because P_0 and P_1 are positive and z is negative which shows white's winning. Finally, we found that when there are opening games mixed with reward and punishment values, there is no guarantee that TD(λ) will always pick the identical opening game from the training opening games independent of the different λ values.

λ	P_0	P_1	z	Best game
0.0	-0.49	+1.00	-1	$1,578^{th}$
0.3	+0.90	+1.00	-1	$1,667^{th}$
0.7	-0.39	+1.00	0	902nd
1.0	+0.07	+1.00	0	628th

Table 3: The picked best games. (α = 0.5)

4.4. Performance of TD(λ)s

To test the performance of TD(λ), we firstly let two TD(λ)s play against each other. The number of moves in each game was limited to 50.

As the evaluation of the final board positions was so difficult, we let a professional 9-dan (the highest rank) player assess the game's likely results as: very favorable to black (VB), favorable to black (FB), very favorable to white (VW), favorable to white (FW), and a result which is difficult to be determined by a professional player (ND).

In Figure 3, each of the row entries represent TD(λ) when black plays, and each of the column entries represent TD(λ) for white. There are 16 games in which each game's potential results are shown. Figure 3 shows TD(λ) with higher λ (and white player) mostly wins against a lower λ [2] when λ is not equal to 1.0. Furthermore, better performance is achieved when λ = 0.0 rather than λ = 1.0. Figure 4 is an opening game played between two TD(λ)s.

Play as black \ Play as white	TD(0.0)	TD(0.3)	TD(0.7)	TD(1.0)
TD(1.0)	VW	ND	VW	VW
TD(0.7)	FW	ND	VW	VB
TD(0.3)	VW	VW	FW	ND
TD(0.0)	VW	ND	VW	VB

Figure 3: Results played between two TD(λ)s.

We secondly tested the performance of TD(λ) against the commercial Go programs: Go^{++} and $NeuroGo$. During the competition, TD(λ) without having *a priori* Go knowledge played poorly against the commercial Go programs. We found that *a priori* knowledge is an essential factor to strengthen TD(λ) learning with a network for solving the opening game problems in Go.

5. Conclusions

We firstly analyzed the first move's prediction P_1 which is the estimate of the probability of the first player's winning. We found that the network by TD(λ) learning is strongly distorted by the composed shapes of the training data rather than the final rewards, and thus the network can predict the winner of a game winning only to a very limited extent.

Secondly, we picked the best opening game, which is the most favorable to black, with different λ values among prototypical professional opening games. The empirical result for picking the best game is promising. But there is no guarantee that TD(λ) will always pick the identical best opening game independent of different λ values.

Thirdly, we tested the performance of TD(λ) by playing against each other and against the commercial Go programs. Through a professional player assessing each game, we found that TD(λ) with higher λ (and white player) mostly wins an opening game. From the competition against the commercial Go, we found that the main drawback of TD(λ) learning with a network is generally lack of *a priori* Go knowledge (such as influence, group's safety, connectivity between groups etc), which will be embedded by future work through implementing a neuro-fuzzy controller.

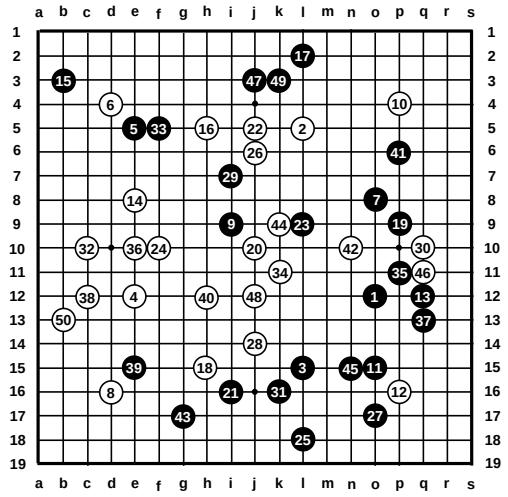

Figure 4: A game record (1-50), which is very favorable to black, played between TD(0.7) (black) and TD(1.0) (white).

References

[1] J. M. Burmeister. *Studies in Human and Computer Go: Assessing the Game of Go as a Research Domain for Cognitive Sciencer*. PhD thesis, Computer Science and Electrical Engineering and School of Psychology, University of Queensland, 2000.

[2] H. W. Chan, I. King, and J. C. S. Lui. Performance Analysis of a New Updating Rule for TD(λ) Learning in Feedforward Networks for Position Evaluation in Go Game. Available online at http://citeseer.nj.nec.com/483789.html, 1996.

[3] D. Hu. *Baduk Exercise Book: Standard Opening Games I and II*. The Korean Baduk Association, 2001.

[4] T. B. Trinh, A. S. Bashi, and N. Deshpande. Temporal Difference Learning in Chinese Chess. In *Proc. IEA/AIE Conf.*, pages 612–618, 1998.