Comparing Training Paradigms for Learning to Play Backgammon

Marco A. Wiering MARCO@CS.UU.NL
Intelligent Systems Group, Institute of Information and Computing Sciences, Utrecht University

1. Introduction

The backgammon learning program TD-Gammon of Tesauro (1995) was probably the greatest demonstration of the impressive ability of machine learning techniques to learn to play games. TD-Gammon used reinforcement learning techniques, in particular temporal difference learning (Sutton, 1988) for learning a backgammon evaluation function from training games generated by letting the program play against itself.

For learning a game evaluation function for mapping positions to moves (which is done by the agent), there are the following three possibilities for obtaining experiences or training examples; (1) Learning from games played by the agent against itself (learning by self-play), (2) Learning by playing against a (good) opponent, (3) Learning from observing other (strong) players play games against each other. The third possibility might be done by letting a strong program play against itself and let a learner program learn the game evaluation function from observing these games or from database games played by human experts.

One advantage of learning from games provided by another expert or a database is that games are immediately played at a high level instead of completely random when the agent would play its own games. Another advantage is that for particular games such as draughts, chess, and Go, usually expensive lookahead searches are necessary to choose a good move. Since lookahead is expensive, learning from a database of played games could save a huge amount of computation time. A disadvantage of learning from databases games or from observing an expert play is that the learning agent is never allowed to try the action which it would prefer. Basically, the exploration is governed by human decisions and there is no exploitation. Therefore, the agent might remain biased to particular moves which the experts would never select and are therefore never punished. We will examine in this paper whether learning from game demonstrations for the game of backgammon is really fruitful compared to other paradigms for generating training games.

2. Experiments with Backgammon

Tesauro's TD-Gammon program learned after about 1,000,000 games to play at human world class level, but already after 300,000 games TD-Gammon turned out to be a good match against the human grand-master Robertie. In our research, first an expert backgammon program was trained so that we have a program against which we can train other learning programs and which can be used for generating games that can be observed by a learning program. Finally, we will evaluate the learning programs by playing test-games against this expert. To make the expert player we used TD-learning combined with learning from self-play using a hierarchical neural network architecture consisting of 9 neural networks of 40 hidden units that evaluate different kinds of strategical positions. This program was trained by playing more than 1 million games against itself during which intermediate tests were held to finally keep the best program.

Experimental setup. We first made a number of simulations in which 200,000 training games were used and after each 5,000 games we played 5,000 test games between the learner and the expert to evaluate the learning program. Because these simulations took a lot of time (several days for one simulation), they were only repeated two times for every setup. For the learning program we made use of different architectures. First of all, we used the same large architecture consisting of 9 neural networks. Furthermore, we used a smaller architecture consisting of three networks; one for the endgame of 20 hidden units, one for the long endgame (racing game) of 20 hidden units, and one for the other board positions with 40 hidden units. We also used a larger network architecture with the same three networks, but with 80 hidden units for the other board positions, and finally we used an architecture with 20, 20, 40 hidden units with a kind of radial basis activation function. These architectures were trained by playing training games against the expert. We also experimented with the small network architecture that learns by self-play or by observing games played by the expert against itself.

Experimental results. Table 1 shows that all architectures and training paradigms, except for the architecture using RBF neurons, obtained an equity higher than 0.5 in at least one of the 80 tests. Testing these found solutions 10 times for 5000 games against the expert indicated that their playing strengths were equal.

Architecture	5000	100,000	175,000	Max eval
Small Network (SN)	0.327	0.483	0.478	0.508
Large architecture	0.290	0.473	0.488	0.506
Network 80 hidden	0.309	0.473	0.485	0.505
Network 40 RBF	0.162	0.419	0.443	0.469
SN Self-play	0.298	0.471	0.477	0.502
SN Observing expert	0.283	0.469	0.469	0.510

Table 1. Results for the different methods as averages of matches of 5,000 games against the expert. The results are smoothed so that the equity for 5000 games is the mean of the 6 tests after 100, 5000, and 10,000 games.

Results of smaller simulations. We performed a number of smaller simulations of 15,000 training games where we tested after each 500 games for 500 test-games. We repeated these simulations 5 times with the small neural network architecture and different paradigms for generating training games. The results displayed in Figure 1 and Table 2 show that observing the expert play and learning from these generated games (expert plays against expert) progresses slower and reaches slightly worse results within 15,000 games.

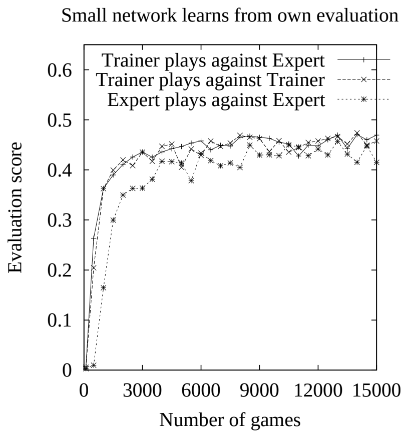

Figure 1. Results for the small architecture when using a particular paradigm for generating games.

Method	100	500	1000	5000	10,000
Self-play	0.006	0.20	0.36	0.41	0.46
Against expert	0.007	0.26	0.36	0.45	0.46
Observing expert	0.003	0.01	0.16	0.41	0.43

Table 2. Results for the three different paradigms for generating training games using the small architecture (SN). The results are averages of 5 simulations.

Effect of λ. Finally, we examine what the effect of different values for λ is when the small architecture learns by playing against the expert. We tried values for λ of 0.0, 0.2, 0.4, 0.6, 0.8, and 1.0. When using $\lambda = 1$ we needed to use a smaller learning-rate, since otherwise initially the weights became too large. Table 3 shows the results. We can see that higher values of λ initially result in faster learning which can be explained by the fact that bootstrapping from the initially random evaluation function does not work too well and therefore larger eligibility traces are profitable. After a while λ values between 0.2 and 0.8 perform similarly.

λ	100	500	1000	5000	10,000
0.0	0.004	0.13	0.31	0.42	0.43
0.2	0.002	0.24	0.34	0.43	0.45
0.4	0.002	0.26	0.35	0.44	0.44
0.6	0.007	0.26	0.36	0.45	0.46
0.8	0.06	0.34	0.39	0.44	0.45
1.0	0.12	0.23	0.31	0.39	0.40

Table 3. Results for different values of λ when the small architecture learns from playing against the expert. The results are averages of 5 simulations.

3. Discussion

Learning a good evaluation function for backgammon with temporal difference learning appears to succeed very well. Already within few thousands of games which can be played in less than one hour a good playing level is learned with an equity of around 0.45 against the expert program. The results show that learning by self-play and by playing against the expert obtain the same performance. Learning by observing an expert play progresses slower than the other methods. In our current experiments the learning program observed another program that still needed to select moves. Therefore there was no computational gain in generating training games. However, if we would have used a database, then in each position also one-step lookahead would not be needed. Since the branching factor for a one-step lookahead search is around 16 for backgammon, we would gain 94% of the computational time for generating and learning from a single game. Therefore learning from database games could still be advantageous compared to learning by self-play or playing against an expert. We intend to study this further on different games such as Go, chess, and 3D action games.

References

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, *3*, 9–44.

Tesauro, G. (1995). Temporal difference learning and TD-Gammon. *Communications of the ACM*, *38*, 58–68.