Constructing a Reinforcement Learning Agent to Play the Game of Checkers

Mike Morris MMORRIS@SOSU.EDU
P.O. Box 455, Durant, OK 74702

1. Introduction / Abstract

This project is for partial fulfillment of Dr. Amy McGovern's Machine Learning class at the University of Oklahoma. The basic charge is to create a machine learning agent and demonstrate its success or failure in a given task. This particular effort involves an agent that intends to improve its ability to play the age old game of checkers.

The game of checkers seems very simple at first glance and in fact, it does indeed have few rules. However, little is known about winning strategies and complications arise quickly, as there are literally more possible board states and moves than suspected atoms in the universe.

Checkers is particularly well-suited to reinforcement learning because a board state is easily observed and an action is simply the next move. The details of this statement are outlined with more detail further in this document. The problem addressed by this experiment is to test the type of value associated with a given board state and test the effectiveness of the rewards calculations to be given after each move.

The experiment intends to show that reinforcement learning in the form of a single C++ program will work as the theory suggests by demonstrating a series of games with a particular opposing player.

Like any basic research, it is difficult to judge the project's importance to mankind but the hope is always here for helpful insight into more important concepts that further the cause of humanity.

2. Problem Definition and Algorithm

2.1 Task Definition

The game of checkers is meant to be played by a learning agent that is a single C++ program. The opponent may vary but will be consistent when determining the results presented further below.

2.2 Algorithm Definition

Standard reinforcement learning techniques are employed in this project.

The state will consist of the 32 board positions along with the pieces or lack of pieces that populate these spaces. Every space will have a value, calculated by the following value function:

$$V(s) = \gamma w_0 + \alpha_1 w_1 + \alpha_2 w_2 + \alpha_3 w_3 + \alpha_4 w_4$$

where $w_0 = 1$ if a "jump" was made by the agent to arrive at the given state, and 0 otherwise, γ is a discount factor between 0 and 1, w_1 = the number of black pieces left on the board, w_2 = the number of red pieces left on the board, w_3 = the number of black "kings" on the board and w_4 = the number of red "kings" on the board. The α_i are weights suitably small and can be altered by the agent as it sees fit. The value function is tunable on the fly and several were examined.

The action for a given state is simply the next move. Without loss of generality, it is accepted that the agent is always black and the red opponent always gets to move first.

The agent will allow its opponent to request a move. A check will be made to ensure the move is legal. If it is, then the request will be granted and the move executed.

There is only one policy that determines what action, or move, the agent will take. It is described as follows. The agent will analyze the board and first determine if the game is still active. If so, the agent will examine each of the possible legal moves and apply the value function to each resulting state. Note that there might be as many as 8 moves for each of the 12 pieces, or 96 possible moves by the opponent, even more if double jumps are executed. The state with the highest value will be obtained by taking the move associated with it. This move will thus be the action.

If the game is over, the coefficients of the value function will be updated. For example, recalling that the agent is black, if the number of black pieces remaining in a winning state 6 or more (of a total of 12) then that state is most desirable and we add weight, say 0.001, to w_1, the coefficient of x_1, which is the black piece count. i.e., we encourage x_1 to have more weight. On the other hand, if the number of red king pieces in a losing state is 4 or more (of a total of 12) then that state is most undesirable and we subtract weight, say 0.001, to w_4, the coefficient of x_4, which is the red king piece count, i.e., we encourage x_4 to have less weight.

The γ factor is originally arbitrarily set at 1 and changes based on certain statuses of the game. Different starting values have been tested. The agent finds the maximum reward from these numbers and takes the move associated with the maximum reward. The status of the game is monitored after each move and the game ends when one side is out of players or until a pre-determined maximum number of moves is taken, in which case the game is declared a tie. The maximum number of moves seemed to work effectively at 1,000.

3. Experimental Evaluation

3.1 Methodology

The project began by simply coding an agent that followed the basic rules of checkers and asked the operator for a move to be entered at the screen for each of the red player's moves. This worked but it was quickly obvious that playing enough games to give the agent a chance to learn was infeasible by this method. Just over 100 games were played to determine this decision.

A near-complete rewrite of the code changed the program to "play itself", i.e., to let the original agent maintain its code and methods and add additional code that simulated the red opponent. In addition, a batch of any number of games could be played.

Time constraints and other considerations prevented the agent from connecting to an online player such as the one offered by the University of Alberta and others. It was decided that for this project, it was more important to focus on machine learning than foreign interfacing.

The first version had the red agent choose a random move from the set of all legal moves for it at its turn. The agent did the same - without benefit of machine learning or any strategy. The results were predictably disappointing. In fact, the maximum-move criteria was often met and the game would end in a draw. When viewing the board in these early random games, it was quite amusing to see both sides wandering around haplessly with little hope of accomplishing anything. Figure 1 below illustrates the activity by the two evenly-matched players that simply examined all legal moves and took one of them at random. The center line represents a fictitious player that might win exactly 50% of the time and the two players in my program can be seen oscillating around it.

Opposing red needed at least some help so it was given the strategy that if a move existed that included a jump of a black player then it would be taken. To test this, several batches of varying lengths were run and the red-to-black win count was almost unanimous at about 99 to 1. This is illustrated in figure 2.

The decision was then made to leave the red player with this effective ability and to give the black agent the same

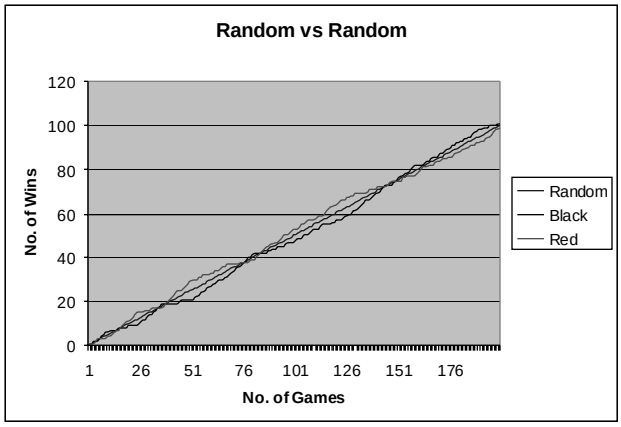

Figure 1.

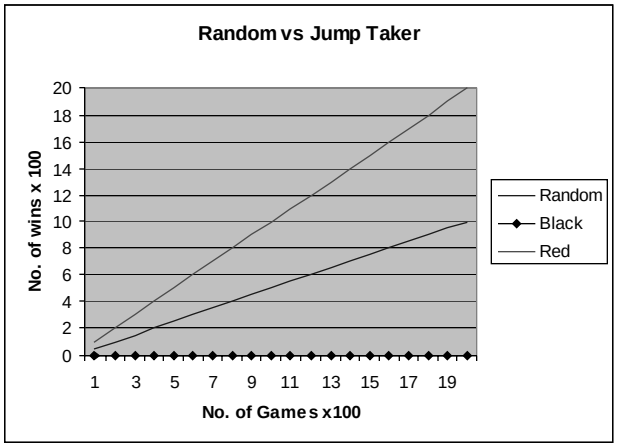

Figure 2.

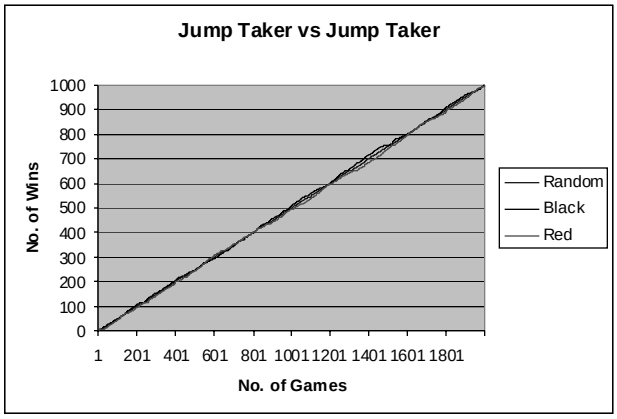

Figure 3.

ability to take an available jump. The results were very similar to figure 1 and a more-game figure 1 is shown in figure 3.

At this stage the two players were declared to be roughly equal in winning ability and the next task was to attempt to inject reinforcement learning into the black agent as described above.

3.2 Results

Before any attempt was made at utilizing reinforcement learning, when any size batch of games was played, the results would generally be within 5% of a 50/50 win/loss count. This gave the experimenter relative assurance that the agents were abiding by standard rules of play.

The methodology was used as described above. In the value and reward calculations, emphasis is placed on jumping an opponent when possible, and the agent appears to be able to learn this after a considerable amount of playing.

The actual results have been somewhat varied and some are actually unexplainable so further work will be required before any concrete claims can be made. However, some batches appear to demonstrate that the agent is learning as predicted and does improve its ability to play.

The following graph, figure 4, depicts a limited successful execution of 14,000 games.

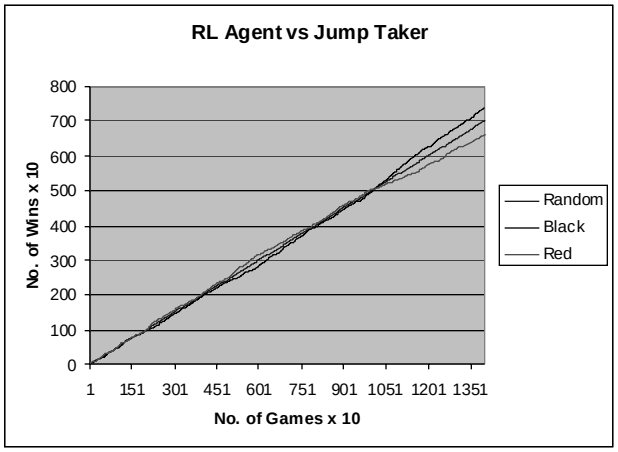

Figure 4.

The middle straight line is simply a fictitious player that wins half of its games. The bottom line represents the wins of the static red player and the top line represents the wins of the dynamic RL-endowed black player. Also notice that this graph is not equal in scale on both axes.

Figure 4 is suggestive of machine learning and the experimenter is pleased with this result, albeit a minor one.

At the presentation, Dr. McGovern suggested a run of 1,000,000 games. That was done and the results were similar to sample runs of ten to fifteen thousand, suggesting that the agent topped out at a certain low level.

I have chosen a different way to graphically illustrate these results. I wrote a quick program to examine the one million win/loss tallies and take a running average, reducing the data points to 1,000. Then I plotted the variance of the RL black agent versus the red player. It's not that exciting, but its upward curve does suggest that its playing skill is slowly increasing. Figure 5 shows this.

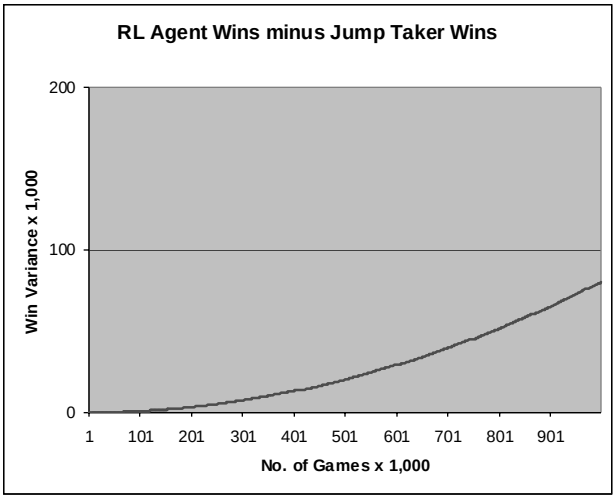

Figure 5.

3.3 Discussion

Confusing results accompanied the most pleasing ones shown above. For instance, some 10,000 game runs show a smaller learning curve, which is unexplainable. A particularly bothering result is the fact that on two separate occasions, a 100,000 game run was made and the learning curve did about the same as the good 14,000 game run shown in figure 4.

The 1,000,000 run was duplicated three times with nearly identical results. Figure 5 simply states that the RL agent won about 540,000 games while the red Jump Taker won about 460,000 games. The curve is non-linear, which was extremely exciting to our fledgling researcher as it indicated improvement in learning.

Better results were initially expected but upon reflection of the dynamics of the project, it is no surprise that these results were somewhat subdued. One suspicion is that part of the problem is the weakness in the evaluation method in the fact that only a one-ply look-ahead is used. This was done in the interest of having a goal that could reasonably be accomplished in the time allotted. This would be one of the first areas addressed in a further development plan. Other checkers agents are known to use a three-ply method.

Another possibility is the debugging of the code. While the utmost efforts were made to write a program that played checkers in a legal fashion, it is indeed a possibility that hidden procedural bugs could warp the findings. Another concern is the fact that the C++ random number generator was used quite a lot, and it does not enjoy a reputation of good randomness.

Last but not least, the researcher confesses that his knowledge of RL has increased dramatically over the duration of the project, but it has not reached full potential at this time.

4. Related and Future Work

The experimenter is only familiar with two other serious projects that address the game of checkers, although no doubt many others exist.

The work done by Dr. A. L. Mitchell back in the late 1950s used three-ply look-aheads and became fairly successful, regularly beating players regarded as "master" players.

Another machine-player is *Chinook*, which is the World Man-Machine Checkers Champion. *Chinook* was developed by a team of researchers led by Dr. Jonathan Schaeffer of the Department of Computing Science at the University of Alberta, Edmonton, Alberta, Canada. It earned this title by competing in human tournaments, winning the right to play for the (human) world championship, and eventually defeating the best players in the world. The Man-Machine title was created to allow there to be both a human world champion and a computer world champion. The man-machine title is bestowed on the victor of a match between the best computer (*Chinook*) and the best human. [This paragraph courtesy and from the University of Alberta web http://www.cs.ualberta.ca/~chinook/about.php.]

The strategies used by *Chinook* are unknown and of utmost power. Owners now claim that their agent can never be beat but only tied. This experimenter's methods do not match either of these agents, but that is not surprising due to the time and resources so far invested.

The experimenter plans to continue work on the project and extend the look-ahead a couple or more plies and hopes to have the agent clearly demonstrating that reinforcement learning is working for this agent and that the learning is clearly evident.

5. Conclusion

Major results have not been accomplished that make the project fit for publication or serious research. However, it appears that the concept of reinforcement learning has been demonstrated to work and this checkers learning agent seems to have learned how to slightly tower over the given opponent.

References:

Samuel, Arthur L., "Some Studies in Machine Learning Using the Game of Checkers", IBM Journal of Research Development, Vol.3. No. 3, 1959

Schaeffer, Lake, Lu, Bryant, Treloar. Chinook. 12 Sep. 2005. University of Alberta, Edmonton, Alberta, Canada <http://www.cs.ualberta.ca/~chinook/>