The Implementation of Machine Learning in Checkers with Temporal Difference Learning

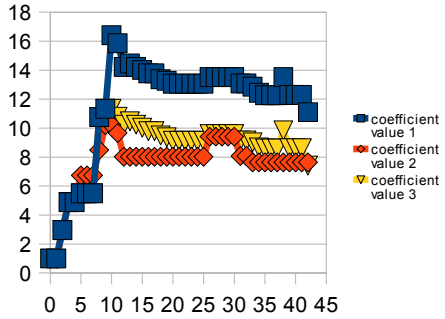
Billy Melicher Computer Systems 08-09

Abstract

The project that I am writing will learn and improve its strategy checkers as it plays, by modifying itself after every game played. The AI will review its performance, and if it played well it will play that way more often. If it did not it will avoid that way of playing.

Background

One method of creating a learning program is to have the program learn by rote. An experience database stores all of the boards encountered along with the moves that had been tried off of these boards. While this is similar to creating all of the possible board combinations and simply searching them exhaustively, this does not require as much memory and is used in conjunction with a static evaluation function. A different method of creating a learning program is by the program modifing the heuristic function each time it plays. The program starts with a heuristic function that has all of the features weighted equally. After the program plays a game, it will change the values of the weights of the heuristic function based on the outcome of the game.



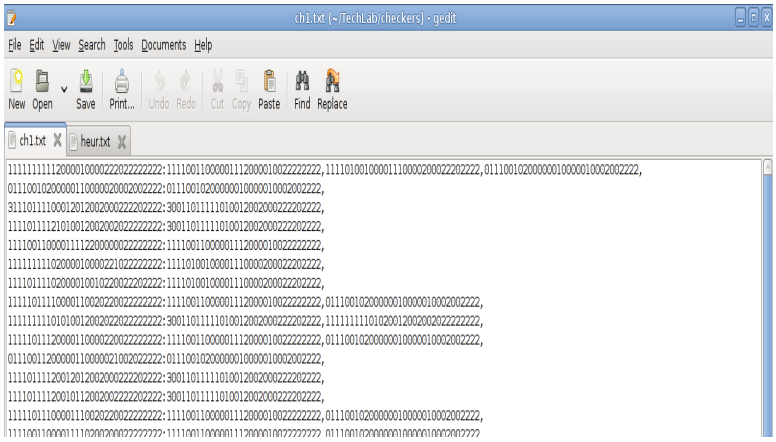
Results

This graph shows the value of the weights of the heuristic over the course of a game. In the beginning, the AI adjusted the weights of the heuristic by a lot but as it went on, they were changed less and less and went the their equillibrium values.

Development

For the learning by generalization algorithm I use a temporal difference learner. Temporal difference learning adjusts the evaluation function during play based on the difference between the evaluation function at one point and the evaluation function at a different time. This will bring the evaluation function into a state of equilibrium toward the ideal evaluation function. Every time another move is made, the heuristic is changed based on the previous heuristic. The equation for this is $U(s) \leftarrow U(s) + \alpha(R(s) + \gamma U(s') - U(s))$. s is the current state of a system, U is the evaluation function of state s, α is the learning coefficient and decreases as the number of times state s has been encountered increases and R is the reward that you will get for state s.

The learning by rote system that I am using is made so that I can decrease α by the number of times that I have seen a different board. My program stores each board that was visited and the boards that can be reached from that board. It does this by converting each board state into a number. The number is a 32 digit number of base 5. There are 32 digits for each space on the board that a piece can be in, and base 5 for each piece that can be there.



This is a copy of the output of the learning by rote file. On the left is a board represented by the string of numbers. On the right of the colon is the list of boards that can be reached from the original board.

Conclusion

Temporal difference learning is a good way to achieve equilibrium value for the weights of the heuristic. However sometimes the temporal difference learning requires the programmer's intervention on getting the program away from a false minimum in the function. Temporal difference learning does not require large amounts of memory like learning by rote. Also substantial learning could be achieved by having relatively few iterations of temporal difference learning. Learning did not require that the program have knowledge of game strategies, however it does require that the game move toward a win and be able to recognize a win.