Optimizing Parameter Learning using Temporal Differences

James F. Swafford II

Department of Computer Science
East Carolina University
Greenville, NC 27858
jfs0301@mail.ecu.edu

Abstract

Temporal difference algorithms are useful when attempting to predict outcome based on some pattern, such as a vector of evaluation parameters applied to the leaf nodes of a state space search. As time progresses, the vector begins to converge towards an optimal state, in which program performance peaks. Temporal difference algorithms continually modify the weights of a differentiable, continuous evaluation function. As pointed out by De Jong and Schultz, expert systems that rely on experience-based learning mechanisms are more useful in the field than systems that rely on growing knowledge bases (De Jong and Schultz 1988). This research focuses on the application of the TDLeaf algorithm to the domain of computer chess. In this poster I present empirical data showing the evolution of a vector of evaluation weights and the associated performance ratings under a variety of conditions.

The playing strength of modern chess playing programs is really a function of the quality of its search and its evaluation of leaf nodes. The search defines the shape of the search space, and the manner in which the program navigates through that space. The evaluation function attempts to quantitatively measure the value of a chess position, and therefore assign values to the search tree's leaf nodes. Given that computer chess programs must assign values to positions, it becomes necessary to break a chess position down into tiny parts, giving points for some attributes, and penalizing for others. The more evaluation terms included in the evaluation function, the more capable the evaluator is to distinguish between positions. This precision comes at a cost, and that cost is complexity. The more terms in the evaluator, the more difficult it becomes to properly tune a new weight relative to existing weights.

$$w := w + \alpha \sum_{i=1}^{N-1} \nabla r(x_i^l, w) \left[\sum_{j=i}^{N-1} \lambda^{j-i} d_i \right]$$

Figure 1: The TDLeaf Algorithm

Traditional (non-learning) methods of tuning evaluation parameters become increasingly impractical as the number of parameters increases. Consequently, a great deal of research has been done to find methods for the self tuning of evaluation parameters, particularly in the domain of a state space search. One such algorithm is TDLeaf (Figure 1), first introduced by Beal (Beal 1997), and applied to chess by Baxter et al. with "Knightcap" (Baxter, Tridgell, and Weaver 2000). While conventional prediction learning methods are driven by the error between predicted and actual outcomes, TD methods are driven by the error between temporally successive predictions (Sutton 1998). One advantage to this approach is that learning is applied incrementally, once per searched move. The learning is applied to the leaf node of the principal variation, adjusting the evaluation vector at that node "towards" the vector of another principal variation leaf node occurring later in the game. The algorithm allows for some tailoring with the λ value (a decay rate parameter) and α (a scaling factor). Setting λ close to one tends to adjust the vector towards the final position's vector. This could be useful if the evaluator can not be trusted. Conversely, a λ close to zero causes the vector to be adjusted towards the next position's vector.

Temporal difference algorithms have had some success in game playing. Despite the overall success of temporal difference algorithms, none of the top ranked computer chess programs utilize them, suggesting they are still unable to produce a set of evaluation parameters superior to a set of carefully hand tuned parameters. Though Schaeffer reports promising results with "Chinook" (a world class checkers program), it should be noted that Chinook contains relatively few evaluation parameters compared to a competitive chess program. The most promising results reported of temporal differences applied to chess are those of "Knightcap", which is far below the grandmaster level in playing strength. (Schaeffer, Hlynka, and Jussila 2001). Perhaps by better understanding the conditions under which temporal difference algorithms are

able to converge the parameter vector to an optimal state, we will be able to use temporal
difference algorithms more effectively and in a wider variety of applications.

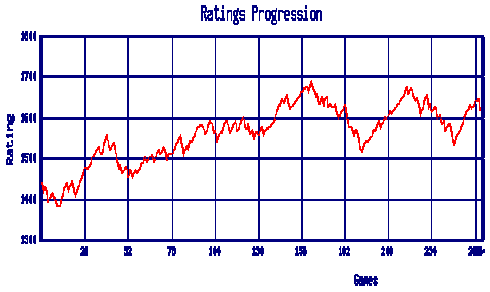

Figure 2: Ratings Progression Beginning with a Material only Vector

Initial data, though showing TDLeaf to be effective, suggest limits to the algorithm's ability to adjust the evaluation vector to an optimal state. Figure 2 plots the ratings progression beginning from a vector in which only the values of the pieces are nonzero. To get a baseline, the engine first played online (against human opponents) with the material-only vector without learning. After 340 games the engine's mean rating was 1504. A separate run of 265 games with learning on (shown in Figure 2) ended with a mean rating of 1558 – substantially higher than the previous non-learning run. The results of a subsequent experiment were, unfortunately, not as promising. Figure 3 illustrates the progression of a separate run of games that was started using a hand-tuned vector of evaluation parameters. After another 265 games the engine had achieved a mean rating of 1715. This is substantially lower than the 1775 mean rating from a previous run of a non-learning engine using the same hand-tuned vector. In this case, the learning was not only ineffective, but was harmful. The vector was moving *away* from optimal.

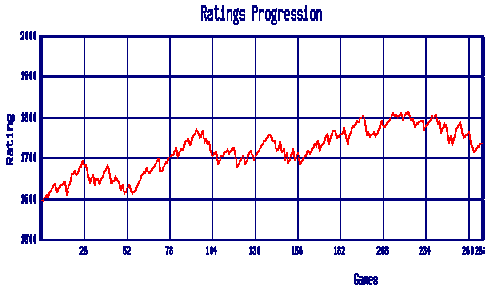

Figure 3: Ratings Progression Beginning with a Hand-Tuned Vector

There are a number of conditions that may delay or even prevent convergence. Shallow searches, for example, leave the program vulnerable to the horizon effect, eventually causing the program to lose for tactical rather than positional reasons. In these cases it is likely the TD algorithm is adjusting the vector *to predict tactical blunders,* and consequently limiting the effectiveness of the program's ability to predict outcome based strictly on positional features. The idea that search depth adversely affects the algorithm's ability to converge to optimality can be tested by running several series of training sessions, each session beginning with an engine that plays at a higher caliber than the last, using the hand tuned vector. Evaluation complexity, distance from the optimal vector, quality of opponents, or length of training games may also delay convergence. Tesauro reported continually improving performance with TD-Gammon after hundreds of thousands of games (Tesauro 1995). Other dynamic or nondeterministic aspects of the program may also confound the algorithm's ability to converge to optimality. Examples include null-move forward pruning, aspiration windows, and principal variation search.

Acknowledgements

The author wishes to acknowledge Dr. Ronnie Smith and Dr. Mike Spurr of East Carolina University for their guidance and helpful insight, and his wife Amy for her support and unending patience.

References

Beal, D. 1997. Learning Piece Values Using Temporal Differences. *International Computer Chess Association Journal,* 20:147-151

Baxter, J., Tridgell, A., and Weaver, L. 2000. Learning to Play Chess using Temporal Differences, in *Machine Learning,* 40:243-263.

Sutton, R. 1998. *Learning to Predict by the Methods of Temporal Differences,* Boston, Kluwer Academic Publishers

Schaeffer J., Hlynka M., and Jussila V. 2001. Temporal Difference Learning Applied to a High-Performance Game-Playing Program, in *Proceedings of the 2001 International Joint Conference on Artificial Intelligence (IJCAI-2001),* 529-534.

De Jong, K. and Schultz, Alan C. 1988. Using Experience-Based Learning in Game Playing, in *Proceedings of the Fifth International Machine Learning Conference,* 284-290.

Tesauro G. 1995. Temporal Difference Learning and TD-Gammon, in *Communications of the ACM,* 38:58-68.