Memory-Based Approaches To Learning To Play Games

Christopher G. Atkeson
Department of Brain and Cognitive Sciences and the Artificial Intelligence Laboratory
Massachusetts Institute of Technology
NE43-771, 545 Technology Square, Cambridge, MA 02139, USA.
617-253-0788, cga@ai.mit.edu

Abstract

This paper explores memory-based approaches to learning games. The learning element stores evaluated positions for future use. A deterministic game (checkers) and a stochastic game (backgammon) are being used as test cases for these approaches.

1 Introduction

Inspired by the success of TD-Gammon (Tesauro 1993), we are trying to understand the range of learning approaches that will work on difficult planning problems such as playing backgammon and checkers. One issue we are focusing on is understanding the spectrum of global and local representations. TD-Gammon uses a relatively global representation. The most extreme form of global representation would use a single model for all positions. TD-Gammon uses a small number of models (represented using neural networks) for different phases of the game, so it is not quite on the edge of the global-local representation spectrum. By global we mean the following: a small number of models are used, the neural networks (models) each have a small number of adjustable parameters, each piece of new information can affect all of the parameters for the corresponding network or model, and each prediction of the value of a position can depend on all of the parameters for a particular network or model as well.

Endgame databases and transposition tables are at the local end of the global-local representation spectrum. Individual positions and their values are stored in a database. The only generalization that is performed is due to the expansion of the evaluation tree. Stored positions affect the value of positions being evaluated only if those stored positions can be reached by a series of moves from the positions being evaluated. Endgame databases are currently often formed by exhaustively evaluating certain classes of positions, such as KPK in chess (Decker et al 1990) or six men left (checkers and/or kings) in checkers (Schaeffer et al 1992). Transposition tables are currently often used during the evaluation of a single position, and after the evaluation of that position is terminated the transposition table is typically discarded.

This paper explores local approaches to learning to play games. One motivation for this research is the feeling that global representations cannot represent the full detail of the true value function for a complex game such as checkers, backgammon, or chess. Tesauro states "The only thing which prevents TD-Gammon from genuinely equaling world-class human play is that it still makes minor, practically inconsequential technical errors in its endgame play" (Tesauro 1993). Clearly, an endgame database would eliminate this problem. Perhaps local representations could also improve TD-Gammon's play throughout the game.

2 Related Work

Previous attempts to utilize local learning date back to Samuel's rote learning method (Samuel 1959) in which positions and the corresponding correct moves were stored. De Jong and Schultz (1988) provide a more recent example of a local learning approach (applied to Othello). Constructive solutions of games are usually based on a complete exploration of the evaluation tree, resulting in a database of positions and their corresponding values. Tic-tac-toe, Connect 4, and double dummy bridge have been solved in this way (Levy and Beal 1989).

Memory-based modeling has a long history. Approaches which represent previous experiences directly and use a similar experience or similar experiences to form a local model are often referred to as nearest neighbor or k-nearest neighbor approaches. Local models (often polynomials) have been used for many years to smooth time series (Sheppard 1912, Sherriff 1920, Whittaker and Robinson 1924, Macauley 1931) and interpolate and extrapolate from limited data. Eubank (1988) surveys the use of nearest neighbor estimators in nonparametric regression. Lancaster and Šalkauskas (1986) refer to nearest neighbor approaches as "moving least squares" and survey their use in fitting surfaces to data. Farmer and Sidorowich (1988a, 1988b) survey the use of nearest neighbor and local model approaches in modeling chaotic dynamic systems. Cleveland, Devlin and Grosse (1988) analyze the statistical properties of local model approaches and Cleveland and Devlin (1988) show examples of their use.

An early use of direct storage of experience was in pat-

tern recognition (surveyed by Dasarathy (1991)). Fix and Hodges (1951, 1952) suggested that a new pattern could be classified by searching for similar patterns among a set of stored patterns, and using the categories of the similar patterns to classify the new pattern. Steinbuch and Taylor proposed a neural network implementation of the direct storage of experience and nearest-neighbor search process for pattern recognition (Steinbuch 1961, Taylor 1959), and pointed out that this approach could be used for control (Steinbuch and Piske 1963). Stanfill and Waltz (1986) coined the term *memory-based* and proposed using directly stored experience to learn pronunciation, using a Connection Machine and parallel search to find relevant experience. They have also applied their approach to medical diagnosis (Waltz 1987) and protein structure prediction. Aha et al (1991) describe related approaches.

3 Issues In Memory-Based Learning

One way to envision the issues involved in a local learning approach is to ask what would happen if a transposition table was maintained as a permanent database? How could one guarantee that the stored evaluations were always improved, if the evaluations were based on a limited depth search guided by changing heuristics and approximate evaluation functions? How can we decide what to remember and what to forget, given limited memory resources? How far should one extrapolate values from stored positions to predict values for similar positions? What distance metric should be used to compute similarity? What are the interactions and tradeoffs between generalization based on similarity and generalization based on the evaluation tree?

Given the range of design choices in designing a local learning approach to games, we have decided to explore the global-local spectrum starting at the far edge of the local end. Values for specific positions are remembered, only exact values are stored, and no generalization based on similarity is used. More "globalness" will be incorporated as the local learning algorithms reach performance limits or we run out of memory or computer time. Various forms of similarity based generalization will then be explored.

Another research approach we could have taken would be to start with a successful global approach such as TD-Gammon and add corrections to its representation of the value function using additional local representations. One could "correct" move choices or values of positions not accurately generated by the global representation. We have not pursued this route, although it is a very reasonable approach to take. We also note that a large database of evaluated positions is an excellent addition to the training set for a global learning approach, and can also be used for discovering or selecting features to be used in heuristic evaluation functions.

To dispell fears that a local approach is hopeless due to the number of positions that need to be stored, let us crudely calculate the number of stored evaluated positions needed to force a 1 ply evaluator to follow the optimal game path. Let us initially assume we have a cooperative opponent who always chooses an optimal move and that for each set of possible moves there is a single best move. In that case each player chooses between B moves (B being the branching factor) at each point in a deterministic game. If the game lasts for L moves, the number of moves evaluated and positions looked up in the database is BL. If there are W best moves with the same value at each position then BLW^L positions need to be stored. A player can resolve the choice between its best moves in a fixed way, but it is unlikely the opponent will be that cooperative, so this estimate can be reduced to $BLW^{L/2}$. If there is a random element before each move (such as R possible outcomes of a roll of the dice) then an estimate of the number of positions that need to be stored is BLR^L.

All of these estimates assume the opponent is 1) unaware of the limited memory of the player, and 2) an optimal player. Neither of these assumptions may be true. However, for the opponent to change the path of the game, the opponent must make a non-optimal move (or a sequence of non-optimal moves). The amount that these moves are non-optimal is controlled by the coverage of the database. A value threshold can be chosen to determine which suboptimal game paths are also stored. If we assume a fixed distribution of move values from any position, the threshold will select T moves at each position to be explored, leading to $BLT^{L/2}$ positions looked up in the database. If the opponent succeeds in moving the game outside the expertise of the database, we expect a more global representation such as TD-Gammon to be able to take advantage of the non-optimal opponent moves.

Given the exponential dependence of some of these estimates on the length of the game remaining, it may be the case that local learning will be most useful in midgames and endgames (local representations are already successful in many endgames in checkers and chess). It is also important to keep in mind that a powerful search algorithm similar to what is currently used in chess and checkers will be used in combination with the position database. (Current backgammon programs search 2 ply, our program based on a local representation routinely searches up to 8 ply in endgame positions). This may dramatically reduce the number of positions that need to be stored for expert play (Schaeffer et al 1992 hope to solve checkers by combining an endgame database with powerful search).

Local learning approaches may still be hopeless, however, when one considers the amount of computation needed to find and evaluate the positions along the optimal game paths that will be stored in the final database. Here, we make use of the same trick used in many global learning programs. We only "train" on positions seen in actual games and positions explored during the evaluation of those game positions. This approach allows us to

take advantage of expert play by human or computer, and focus attention and memory resources on positions that occur during expert play. We are no longer representing optimal game paths, but only expert game paths.

We use additional tricks to limit the number of positions that are stored. If a function that bounds the value of an unevaluated position is available, these bounds can be used to eliminate this position from consideration in the database building process. Any heuristic evaluator can be used to guide game tree search to minimize the number of positions stored in a similar way to how the alpha-beta algorithm allows us to ignore portions of a evaluation tree. So far, we have stored all evaluated positions. In the future we plan to store only the positions resulting from best moves (the best of all legal moves from a given position). We hope to use the technology of limited rationality (Russell and Wefald 1991, Baum and Smith 1993) to help decide what is worth computing, and what is worth remembering. Schaeffer et al (1992) point out that it is sometimes cheaper to run length encode a complete database for a class of positions rather than to encode a limited number of positions from the same class, even when the sparse database represents the errors of a parametric evaluation function fit to the complete database. We do not expect this to be generally true, but we will use exhaustive databases for classes of positions which can be compactly represented in this way.

4 A 4x4 checkers game

As a simple initial example we explored a 4x4 checkers game, with each side having two pieces. Considering the possibility that pieces can be promoted to kings, there are 10,836 possible positions. We fully evaluated the starting position, using a minimax value propagation algorithm. Simple pruning algorithms of the following form were used to restrict the search. If a move to a winning position is found, then the value of the initial position is a win, that of the parent position is a loss, and no further evaluation of that parent position or its descendents need be done. The search was terminated when no new positions were generated by any moves from any positions. 3133 positions were evaluated in the complete game tree, 1706 of these positions were unique, and the search reached a depth of 36 ply.

We constructed an optimal player using a local representation. The optimal player used a 1 ply evaluator, and the storage control strategy was that the optimal player used a fixed choice of moves so that only one of the best moves had to be stored, while the opponent could choose from any of its best moves, so that all the best opponent moves had to be stored. In the case where the optimal player moved first, the values of 111 positions had to be stored in the local representation. This number would have been reduced if the value of the game was a win or a loss rather than a draw. In any event, for the opponent to guide play out of the database of 111 positions, the opponent would have had to make a losing move. In this simple example there is almost a square root reduction from the size of an exhaustive database (10,836 positions) to the size of the database needed for an optimal player (on the order of 100 positions). It is an empirical question as to whether this kind of reduction will hold for more complex games.

5 An endgame database for backgammon

We are in the process of generating an endgame database for backgammon. We are currently focusing on positions in which both players are bearing off (2,944,473,169 possible positions). Candidate positions to be included in the database are generated by self play using a linear heuristic evaluation function, Tesauro's pubeval (Tesauro, personal communication). These positions are evaluated using a minimax search with a fixed number of evaluations. Any position whose value is known exactly after that search is stored. Approximate values for positions are used to guide the search, and bounds on the values of the positions are used to provide early cutoff. The approximate and bounding values are propagated along with the known values in the game tree. Approximate values are derived by comparing the probability distributions of two players trying to independently minimize the expected number of rolls until their pieces are off the board. This heuristic was taken from Berliner's BKG program. Bounds on the value of a particular position were taken in the following way. A worst case for a player is if that player follows a suboptimal strategy, while the opponent plays a superoptimal strategy, and an upper bound on the value of a position is provided by calculating the expected outcome of a superoptimal strategy against a suboptimal strategy. The probability distributions of the number of rolls until one player is off the board for each of these strategies were compared to produce a bound on the expected value of a position. The suboptimal strategy used was to minimize the expected number of rolls until all pieces are off the board, independent of the other players actions. The superoptimal strategy used was to maximize the probability of bearing all pieces off, knowing in advance the number of rolls left in the game. This strategy is better than any optimal strategy, but it requires knowledge that is not available during play. These approximations and bounds can be applied to more general race positions, but it is not yet clear how to bound the value of contact positions.

The position database is still being built, and we will provide a crude evaluation of its performance here. Currently the database consists of 28,835,746 evaluated positions. No attempt has been made to prune this database of unnecessary positions. A test set (not used in the construction of the database) of the first mutual bearoff positions encountered in games was generated. The test set contained 327 positions. 127 of these positions could

be evaluated exactly using the bounds or using 1 ply search. Without using the database, and using search trees built using 50,000 position expansions (move generation and evaluation) 14 of the remaining positions were evaluated to an exact value. Successful searches went as deep as 8 ply, and unsuccessful searches went as deep as 17 ply. This measures the usefulness of the approximations and bounds described previously independently of the database. Using the database, and reducing the number of position expansions to 10,000 resulted in 81 positions being evaluated to an exact value, in addition to the 127 immediate evaluations. Successful searches went as deep as 8 ply, and unsuccessful searches went as deep as 14 ply.

6 Conclusions

In some ways the extremely local representational approach we are taking is old, and in some ways it is quite new. Samuel used stored positions in 1959, and endgame databases and transposition tables have been around a long time. On the other hand, our goal is to store evaluated midgame positions on a massive scale. We are currently using 8 bytes to store a position and its value. Using this crude storage scheme 10^9 positions can be stored on a $10,000 disk drive (or less). We intend to combine this with a search engine that is capable of searching 1 million positions/second. This will require the development of new search algorithms, as stored positions will provide early cutoffs of portions of the search, and it may be the case that directing the search towards areas of expertise (islands of knowledge) and away from positions that are typically not in the database (seas of ignorance) may be more useful than a completely forward driven search. It is not clear how to optimally combine a heuristic evaluation function with this search process, or how to optimally generate or learn an evaluation function using the information in the database. A large number of empirical questions lie ahead as to whether this approach will prove to be useful.

Acknowledgements

This paper describes research done at the Department of Brain and Cognitive Sciences, the Center for Biological and Computational Learning and the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support was provided under Air Force Office of Scientific Research grant AFOSR-89-0500, by the Siemens Corporation, and by the ATR Human Information Processing Research Laboratories. Support for CGA was provided by a National Science Foundation Presidential Young Investigator Award.

References

Aha, D. W., D. Kibler, and M. K. Albert (1991), "Instance-Based Learning Algorithms, *Machine Learning,* 6:37-66.

Baum, Eric B., and Warren D. Smith (1993), "Best Play for Imperfect Players and Game Tree Search", manuscript.

Cleveland, W.S. and S.J. Devlin (1988), "Locally Weighted Regression: An Approach to Regression Analysis by Local Fitting", *Journal of the American Statistical Association* 83, 596-610.

Cleveland, W.S., S.J. Devlin and E. Grosse (1988), "Regression by Local Fitting: Methods, Properties, and Computational Algorithms", *Journal of Econometrics* 37, 87-114.

B. V. Dasarathy, (1991), *Nearest Neighbor Pattern Classification Techniques,* IEEE Computer Society Press.

Decker, S. T., H. J. van den Herik, and I. S. Hershberg, (1990), "Perfect Knowledge Revisited", *Artificial Intelligence,* 43: 111-123.

De Jong, Kenneth. A. and Alan C. Schultz, (1988), "Using Experience–Based Learning in Game Playing", *Proceedings of the Fifth International Conference on Machine Learning',* Ann Arbor, June 12-14, 1988, published by Morgan Kaufmann Publishers, San Mateo, CA. pp. 284-290.

Eubank, R.L. (1988) *Spline Smoothing and Nonparametric Regression,* Marcel Dekker, New York, pp. 384-387.

Farmer, J.D., and J.J. Sidorowich (1987) "Predicting Chaotic Time Series," *Physical Review Letters,* 59(8): 845-848.

Farmer, J.D., and J.J. Sidorowich (1988a) "Exploiting Chaos to Predict the Future and Reduce Noise." Technical Report LA-UR-88-901, Los Alamos National Laboratory, Los Alamos, New Mexico.

Farmer, J.D., and J.J. Sidorowich (1988b) "Predicting Chaotic Dynamics." in *Dynamic Patterns in Complex Systems,* J.A.S. Kelso, A.J. Mandell, and M.F. Schlesinger, (eds.) World Scientific, New Jersey, pp. 265-292.

Fix, E. and J.L. Hodges, Jr. (1951) "Discriminatory analysis, Nonparametric regression: consistency properties", Project 21-49-004, Report No. 4. USAF School of Aviation Medicine Randolph Field, Texas. Contract AF-41-(128)-31, Feb. 1951

Fix, E. and J.L. Hodges, Jr. (1952) "Discriminatory analysis: small sample performance", Project 21-49-004, Rep. 11 USAF School of Aviation Medicine Randolph Field, Texas. Aug. 1952.

Levy, D. N. L., and D. F. Beal, (1989), *Heuristic Programming in Artificial Intelligence: The First Computer Olympiad,* John Wiley and Sons, New York, NY.

Macauley, F.R. (1931) *The Smoothing of Time Series,* National Bureau of Economic Research, New York.

Russell, S. and E. Wefald (1991), *Do the Right Thing,* MIT Press, Cambridge, MA.

Samuel, Arthur (1959) "Some Studies of Machine Learning Using the Game of Checkers", *IBM Journal of Research and Development,* 3(3):211-229.

Schaeffer, J, J. Culberson, N. Treloar, B. Knight, P. Lu, and D. Szafron, (1992), "A World Championship Caliber Checkers Program (Research Note)", *Artificial Intelligence,* 53: 273-289.

Sheppard, W. F. (1912) "Reduction of Errors by Means of Negligible Differences," *Proceedings of the Fifth International Congress of Mathematicians,* Vol II, pp. 348-384, E. W. Hobson and A. E. H. Love (eds), Cambridge University Press.

Sherriff, C. W. M. (1920) "On a Class of Graduation Formulae," *Proceedings of the Royal Society of Edinburgh,* XL: 112-128.

Stanfill, C., and D. Waltz (1986) "Toward Memory-Based Reasoning." *Communications of the ACM,* vol. 29 no. 12, December, pp. 1213-1228.

Steinbuch, K (1961) "Die lernmatrix," *Kybernetik,* 1:36-45.

Steinbuch, K. and U. A. W. Piske (1963) "Learning Matrices and Their Applications," *IEEE Trans. on Electronic Computers,* EC-12, December, pp. 846-862.

Tesauro, Gerald (1993), "TD-Gammon, A Self–Teaching Backgammon Program, Achieves Master-Level Play", *Neural Computation,* in press.

Waltz, D.L. (1987) "Applications of the Connection Machine", *Computer,* 20(1), 85-97, January.

Whittaker, E., and G. Robinson (1924) *The Calculus of Observations,* Blackie & Son, London.