A Machine Learning Approach to Opponent Modeling in General Game Playing

Tyler Hicks-Wright 05142387
Eric Schkufza 05059547

December 15, 2006

Abstract

General Game Playing is an area of research in artificial intelligence that focuses on the development of agents that are capable of playing competent and even competitive matches of arbitrary games that they have never seen before. Successful general game playing focuses largely on the search and expansion of game trees in a limited amount of time. Our goal is to further enhance the efficiency of that search by relaxing the assumption that a player will always select the move that is most detrimental to its opponent (via Minimax) and instead to compute the probabilities with which a player may make its next move. Once those probabilities are determined, an agent can make educated guesses as to which parts of a game tree to invest search time in, and which can be trimmed from that search all together.

1 General Game Playing

General Game Playing is an area of research in artificial intelligence that focuses on the representation of games in terms of a universal, abstract language known as Game Description Language (GDL). The abstract nature of that language allows for the development of intelligent agents that without modification can perform competently on games that they have never seen before, based on exploitable, common features of games described within that framework.

GDL allows for the representation of discrete, multi-player, deterministic, complete-information games. GDL represents games as sequences of states in which each state is represented as a set of facts that are currently true of the world (for instance, a particular state of tic-tac-toe might contain the fact that there is an X in the center cell). GDL represents game dynamics as sets of moves that players can perform. Players are each permitted one legal move per state, where the legality of a move is a function of the facts that hold true in that state (for instance, in tic-tac-toe, a player may place an X in the center square if and only if he is playing the role of X, and that square is currently empty). GDL allows for turn based games by specifying that a player's only legal move when it is not his turn, is to noop.

The Stanford Logic Group maintains a working implementation of a GGP testbed known as GameMaster, at http://games.stanford.edu:4000.

2 Opponent Move Modeling

General game players typically rely on two main techniques for determining which of a host of legal moves is the best one for them to play: Either established search techniques based on game theory, such as Minimax and Alpha-Beta pruning, or because game trees tend to be too deep to fully search, a suite of heuristics to determine the utility of each player's position during a game.

Difficulty often arises because proper application of game theory's search techniques requires an accurate evaluation of each state in a game tree, and unfortunately, there is no single heuristic that provides an accurate state evaluation function for all games. Additionally, the cut-throat assumption that any other player will choose the move most detrimental to its opponents tends to be too strong, as there are a number of games which require cooperation, and can be un-winnable if a player makes the wrong assumption about some other players' intentions.

By forming a conjecture as to the likelihood of a player choosing a certain move based on its previous behavior rather than by way of search, two major advantages can be gained. First, assuming low error rates have been achieved, moves can be searched in order of highest probability, and very low probability moves can be trimmed from search altogether. This simultaneously decreases game tree search time and increases the likelihood that the consequences of a player's chosen move will be evaluated more thoroughly. Additionally, it allows a player to judge the expected utility of each of its moves, based on the likelihood of it's opponents' responses in such a way as to allow it to improve over the results it would obtain with traditional methods. Consider, for example, a situation in chess where a player has a winning move which would be a guaranteed if its opponent didn't have the option of a sacrifice which created a winning position for itself. If that player knows that the probability of its opponent playing that sacrifice is very low, then it can determine that the expected reward obtained by playing that move is worth the risk of its opponent discovering the win.

3 Technique

Given an instance of a general game, our technique is to develop an accurate model of the strategy employed by an opponent while playing that game.

To do so, we simulate matches of that game by replacing every player other than the opponent by a random player (one that selects a non-deterministic move from the set of legal moves available in each state[1]) and record both the sequence of states observed during that game, and the corresponding moves performed by the opponent.

A training set $\mathcal{S} = \{(x^{(i)}, y^{(i)}) : i = 1, 2, \ldots, m\}$ is then formed by creating a pair $(x^{(i)}, y^{(i)})$ for each of the game states observed during simulation, where $x^{(i)}$ represents one of those states, and $y^{(i)}$ represents the move that the opponent performed. The only restriction that is enforced while developing training data is that all instances of the games being simulated are generated from identical initial conditions. For instance, in the case of tic-tac-toe, all games begin with an identical empty board.

From $\mathcal{S}$, a classifier is trained to rank an opponent's legal moves based on the probability that they will be performed, given some arbitrary game state. The resulting predictions are Markovian, insofar as they do not consider move history.

[1]Remark: Because the GGP framework does not support non-determinism, non-determinism can be artificially injected by introducing a random player that's only purpose is to manipulate game dynamics.

4 Implementation

4.1 GameMaster Test Bed

GameMaster provides a test bed from which it is possible to simulate general games and generate training data. A suite of games of varying difficulty have already been encoded in GDL and are available for bulk simulation. A small set of general game players are also maintained by GameMaster, including a random player, a deterministic legal player, and an iterative deepening MiniMax player.

4.2 Data Representation

Training data is recorded as a matrix x, where each row, $x^{(i)} \in \mathcal{R}^{|A|}$, represents a game state, where A is the "vocabulary" of axioms encountered during training and $x_j^{(i)} \in \{0, 1\}$ indicates whether or not the j^{th} axiom occurred in the i^{th} game state. For the games that we considered, the axiom vocabulary, A, was never so large that the matrix, x, proved unwieldy. However, there may exist some games for which this representation is infeasible. Similarly, each $y^{(i)} \in \mathcal{R}^{|\alpha|}$ is an index into the vocabulary of a game's legal moves, α.

Additionally, although not strictly part of a GDL game state encoding, game states are occasionally augmented to include the set of legal moves that they admit.

4.3 Classification and Move Ranking

Training data is used to tune a Multivariate Naive Bayes classifier, modified to make use of chi-squared feature selection. Opponent move predictions are formed by considering the set of legal moves that a game state allows for and ordering those moves based on their probability, given the current state.

5 Experimental Results

5.1 Experiments

Experiments were performed on three separate domains, each intended to represent qualitatively unique opponent types: the Mummy Maze domain, the Tic-Tac-Toe domain, and the Racetrack domain.

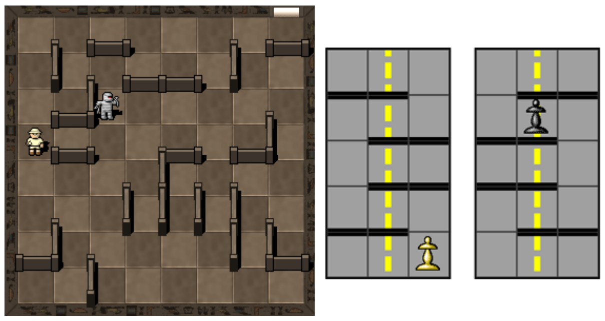

Figure 1: The Mummy Maze and Racetrack domains

5.1.1 Mummy Maze

The Mummy Maze domain is a one-player game, played on an 8x8 grid. The objective is to maneuver an explorer, one step at a time between adjacent cells towards an exit. Additionally, the explorer must avoid a mummy, which is allowed two steps for each of his own, but is constrained to move in a deterministic fashion (first horizontally towards the explorer if possible, and then vertically if possible). Successful play involves capitalizing on the mummy's determinism to trap him safely behind walls.

Training data was obtained by simulating 100 games of Mummy Maze played by a random explorer. The objective of the experiment was to determine whether our opponent modeler could successfully predict a deterministic strategy.

5.1.2 Tic-Tac-Toe

The Tic-Tac-Toe domain is a simulation of the well known two-player game with the advantage that it's relatively small size (the complete game tree contains just over 350 000 states) allows for the application of traditional game theoretic techniques, such as the MiniMax algorithm.

Training data was obtained by simulating 100 games of Tic-Tac-Toe between a random player and a MiniMax player. The objective of the experiment was to determine whether our opponent modeler could successfully predict a perfect strategy.

5.1.3 Race Track

The Racetrack domain is a two-player game in which players attempt to move from one end of a hallway to the other in as little time as possible while at the same time are given the ability to place walls in their opponent's way. It's modest size renders the application of traditional game-theoretic techniques computationally infeasible.

Training data was obtained by simulating 100 games of Racetrack between a random player and a General Game Player designed to augment iterative-deepening MiniMax search with game state utility estimation heuristics. The objective of the experiment was to determine whether our opponent modeler could successfully predict an imperfect strategy.

5.2 Results

5.2.1 Mummy Maze

Figure 2 presents a top-N curve for the Mummy Maze domain obtained by accumulating predictions made over ten test games. It plots the percentage of instances in which the moves performed by the mummy were ranked somewhere between 1 and n.

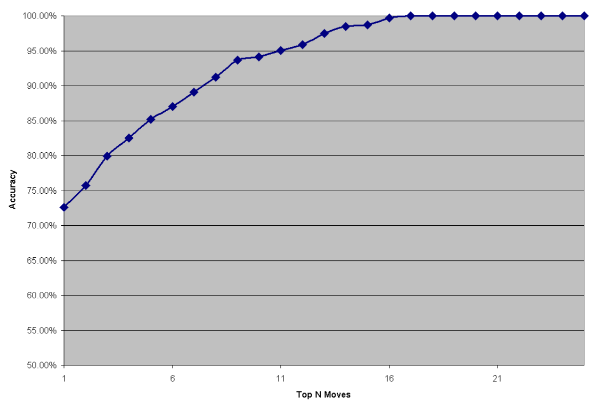

Figure 2: Top-N curve for Mummy Maze

The results are encouraging; for $n \geq 17$, there was a 100% chance that the mummy's moves would be ranked somewhere between 1 and n.

Being able to outright discount eight of the twenty-five legal moves available to the mummy for each turn amounts to an enormous saving in number of the number of nodes that need to be explored to reach a given search depth in the mummy maze game tree. Figures 3 and 4 explore this fact further; their contours represent the number of nodes that need to be explored to reach a given search depth while remaining p% confident that at each level of that search, the moves actually performed by the mummy will be considered.

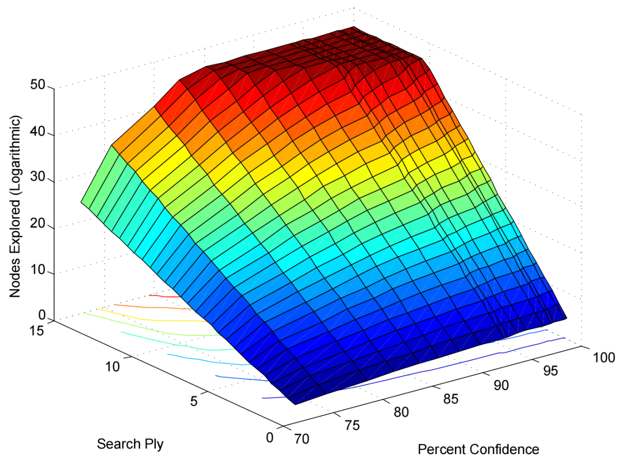

Figure 3: Confidence manifold for Mummy Maze

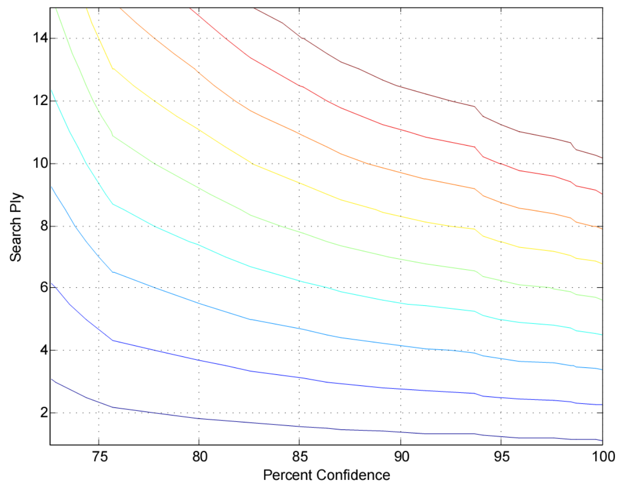

Figure 4: Confidence contours for Mummy Maze

The results suggest that a player with fixed computational resources would be able to search down several additional levels than it would have been able to otherwise, if it were willing to make only a modest sacrifice in the certainty that it considered the mummy's actual moves at each level.

5.2.2 Tic-Tac-Toe

Figure 5 presents top-N curves for the Tic-Tac-Toe domain based on ten test games. Tic-Tac-Toe differs from Mummy Maze in a significant way: the set of legal moves available to a player varies as the game progresses (in particular, a legal move can only be played once).

The blue curve represents the results obtained by training solely on game state data and assuming every move to be legal in every state. Although a poorer result than that obtained for Mummy Maze, it suggests that even in the absence of legal move information, some intelligent predictions can still be made through application of our technique. The pink curve, which represents a modest improvement, was obtained by training on game state data, augmented with the set of legal moves available to each player at each state. Finally, the maroon curve represents the results obtained when only the legal moves available to a player were considered for each state. A much stronger result, it suggests behavior more typical of what might be expected if our technique were to be employed during a live game.

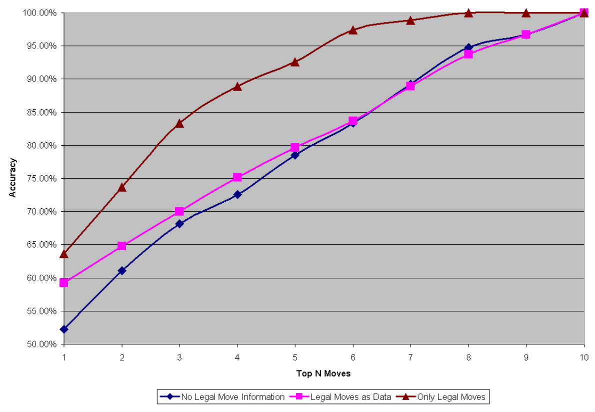

Figure 5: Top-N curves for Tic-Tac-Toe

5.2.3 Race Track

Figure 6 presents a top-N curve for the Racetrack domain based on ten test games; it is encouraging to note its similarity to the top-N curve obtained for Mummy Maze. Even when confronted by an opponent that makes heavy use of non-deterministic heuristics and incomplete information, our opponent modeling technique appears to be able to to safely eliminate the need to explore every legal move available to that opponent.

5.3 Error Analysis

5.3.1 Mummy Maze

The primary source of error in the Mummy Maze domain was the abundance of instances observed during training where the move performed by the mummy

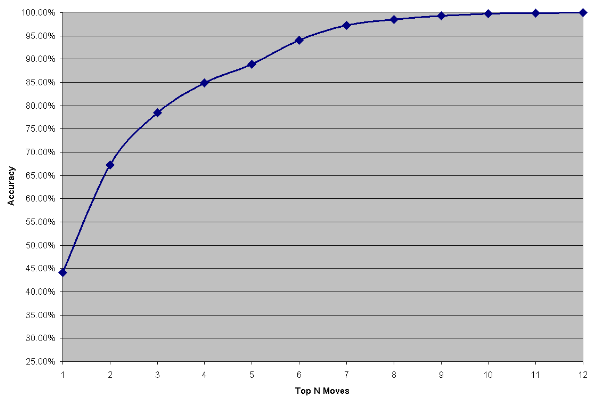

Figure 6: Top-N curve for Racetrack

was `(move2 nowhere nowhere)`. This occurred every other move (when it was the explorer's turn to move) and whenever the mummy was stuck behind a wall (which happened frequently). The resulting bias in the training data presumably caused the classifier to over-fit itself to classifying `(move2 nowhere nowhere)`, making it less likely that the mummy's actual move could be identified.

Training only on instances when it is a player's turn to move (for turn-based games) and narrowing the number of legal moves available to a player in a state by disallowing moves that would require a player to move through a wall might help significantly.

5.3.2 Race Track

Classification accuracy suffered greatly in the Racetrack domain due to the fact that its legal moves are specified absolutely `(move white a 3 b 3)` as opposed to relatively `(move down)`. This fact forced the classifier to tune itself to 35 different moves, which greatly reduced the amount of training data available to each. A re-axiomization of the rules of Racetrack to make use of relative moves would greatly increase the accuracy of the classifier and effectiveness of our modeling technique.

6 Future Work

6.1 Deployment on GameMaster

The underlying motivation beneath this paper was standardization and proofing of an opponent modeling technique to be used as a General Game Playing Heuristic. To the extend that that technique has proven itself useful and computationally feasible, the next step in its development is its deployment as a standard feature to be used by the General Game Players maintained by GameMaster.

Once successfully integrated into GameMaster, additional work might focus on the implementation of on-line learning techniques and more sophisticated feature selection algorithms.

6.2 Application to Transfer Learning

Transfer learning is a field of Artificial Intelligence that attempts to formalize the mechanisms by which an agent trained on a particular game can apply that knowledge to games of varying degrees of similarity. As an attempt to apply our work to transfer learning, we will consider the implementation of opponent modeling techniques that through source game experience can develop opponent models that can be applied to target games that differ slightly in their axiomitization (in the case of tic-tac-toe for instance, to games that begin with non-empty boards).

7 Conclusion

The technique presented in this paper for probabilistic opponent modeling produced strong results. Wherever applied, it afforded the possibility of ignoring some subset of an opponent's legal moves when performing game tree search.

In the future, in combination with existing heuristics, the rankings that it generates for each of an opponent's legal moves might be used to better estimate state utility. In combination with existing game theoretic techniques, such as Alpha-Beta pruning, its rankings might be used to promote earlier pruning and gains in computational efficiency.