Evolving Explicit Opponent Models in Game Playing

Alan J. Lockett, Charles L. Chen*, and Risto Miikkulainen
Department of Computer Sciences
Department of Electrical and Computer Engineering*
The University of Texas at Austin
alockett@cs.utexas.edu, clchen@ece.utexas.edu, risto@cs.utexas.edu

ABSTRACT
Opponent models are necessary in games where the game state is only partially known to the player, since the player must infer the state of the game based on the opponent's actions. This paper presents an architecture and a process for developing neural network game players that utilize explicit opponent models in order to improve game play against unseen opponents. The model is constructed as a mixture over a set of cardinal opponents, i.e. opponents that represent maximally distinct game strategies. The model is trained to estimate the likelihood that the opponent will make the same move as each of the cardinal opponents would in a given game situation. Experiments were performed in the game of Guess It, a simple game of imperfect information that has no optimal strategy for defeating specific opponents. Opponent modeling is therefore crucial to play this game well. Both opponent modeling and game-playing neural networks were trained using NeuroEvolution of Augmenting Topologies (NEAT). The results demonstrate that game-playing provided with the model outperform networks not provided with the model when played against the same previously unseen opponents. The "cardinal mixture" architecture therefore constitutes a promising approach for general and dynamic opponent modeling in game-playing.

Categories and Subject Descriptors
F.1.1 [**Theory of Computation**]: Models of Computation – *self-modifying machines.* I.2.1 [**Computing Methodologies**]: Artificial Intelligence -- Applications and Expert Systems – *games.* J.m [**Computer Applications**]: Miscellaneous – *card games.*

General Terms
Algorithms, Experimentation, Theory.

Keywords
Opponent modeling, neuroevolution, neural networks, games of imperfect information, AI, artificial intelligence, evolutionary computation.

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

1. INTRODUCTION
Opponent modeling is used in the game-playing literature to refer to the process of training computer players to play against specific opponents. General strategies that can defeat all opponents do not exist, *e.g.*, in games where the state space is too large to model, or where aspects of the current game state are not observable. Role Playing Games (RPGs), RoboCup Soccer, and Go fall into the first category; poker and other games of chance are examples of the second category. From the perspective of an individual player, massively multi-player online RPGs (MMORPGs) and first-person shooter games fall into both categories. Furthermore, even if an optimal strategy may exist in a game, it may be considered unfair for automated players to take advantage of it when playing against humans. Thus a variety of effective, opponent-specific models are useful in many games.

Opponent models can be either *explicit* or *implicit*. Much of the work on opponent modeling to date uses implicit models; that is, the learning algorithm encodes knowledge about opponents within its internal representation or decision-making process as a side effect of the training process. In some sense, any player that is successfully trained against diverse, specific opponents will encode a rudimentary, black-box knowledge of its opponents; this knowledge constitutes its implicit opponent model. The goal with an implicit model is to defeat specific opponents, but it is not clear how or whether this implicit knowledge of specific opponents can generalize to previously unseen opponents without further training.

Explicit modeling, by contrast, means training a functional form (*e.g.*, a neural net) to take information about the opponent as input and produce a representation of the opponent as an output, either in terms of the actions the opponent is to take, the game-playing characteristics the opponent is expected to possess, or the strategies that an opponent is likely to employ. An automated game player can then be trained to utilize the output of the explicit model to determine its own counterstrategy. An *explicit* model is therefore intended to generalize directly to previously unseen opponents without any extra training by enabling recognition of unseen opponents.

A crucial question in designing an explicit opponent model is what information should be included in the model. This issue is intimately connected with the problem of generalization. The most direct approach would be to use the opponent's past actions in order to predict his future actions. In practice, however, predicting future actions directly does not generalize well to previously unseen opponents, as we will see. Another approach is to model previously unseen opponents with reference to a set of known opponents. This approach can be effective only insofar as the set of known opponents fully characterizes all relevant dimensions of player action. In this vein, one can conceive of a *opponent space* defined by these dimensions, and a set of *cardinal opponents* that generate

the opponent space. Any possible opponent can be viewed along these lines as a linear combination of opponents in the *cardinal set*. Thus, a reasonable opponent model would only need to identify appropriate coefficients to describe unseen opponents linearly in terms of the cardinal opponents. We refer to such coefficients as a *mixture*, and to opponents generated from such coefficients as *mixture players*. Opponent actions can be predicted from these mixtures by combining the actions of the cardinal opponents.

To capture the intuition for this approach, consider an analogy with how humans learn to play games. Over time we learn to recognize broad themes in our opponents: whether they have a tendency to be aggressive, cautious, presumptuous, *etc*. By identifying the degree to which a new opponent exemplifies one or more of these basic strategies, we can then play effectively against that opponent using past knowledge. The cardinal opponents described above can play a similar role.

This paper evaluates opponent models in the game of Guess It, a simple game of chance first described by Isaacs [7]. The best mixed strategy in a game-theoretic sense wins about half of all the games played. Further, there are basic strategies that can defeat this strategy with high probability, though each of them is also defeated easily by other basic strategies. Playing Guess It well therefore requires diagnosing the opponent's strategy. Once the opponent's playing style is known, he can be easily defeated. In fact, networks trained to play Guess It using only the identity of the opponent as input play the game better than networks that also take the game state as input. Thus the key to winning in Guess It is the ability to model the opponent. On the other hand, the game is sufficiently simple to allow a clear demonstration that the mixture-based approach to opponent modeling is both feasible and promising.

2. RELATED WORK

This work builds directly on that of DiPietro *et al*. [4], who showed that opponent modeling could be used to learn to play effectively against basic opponents in Guess It. The opponent models were implicit, in the sense that no identifying information about the opponent was provided to the learners. They compared evolutionary, hill-climbing, and particle swarm learning algorithms, each of which was trained to find a mixed strategy for Guess It by playing against one basic opponent at a time. Separate learners were trained for each game state and then combined into a team to play the game. Both the evolutionary and the particle swarm algorithms successfully learned to defeat their opponents. Evolutionary algorithms adapted fastest, but particle swarm generated more optimal solutions. Unfortunately, the solutions learned did not generalize across opponents. A change in opponents required retraining the algorithms completely, with an accompanying temporary drop in performance. It took 100 generations for the learners to return to peak performance against the new player. In contrast, in the approach taken in this paper, neural networks are trained to play the game, with an opponent model as input. Using such the game state and a functional form makes it possible to avoid training a player for each game state. Though this approach results in a game player that is more complex and correspondingly more difficult to train, it makes it possible to construct explicit opponent models that generalize well to unseen opponents.

A significant body of work on opponent modeling has been done in poker, much of which includes explicit opponent modeling [9, 12]. Billings *et al*. [1] used statistical methods to estimate the probability of the opponent's hand given his history of calling, raising, or folding. They also developed a predictor to determine what decision a specific opponent would make when holding a given hand. Although the original predictor was only 51% accurate, Davidson *et al*. [2, 3] used a neural network trained with backpropagation to increase its accuracy to 81%. These data are especially interesting since their poker player, Loki, gathered statistics on actual human players by playing in online poker games. However, the approach required a significant history of data for training and therefore could not be used online. In contrast, the mixture-based approach does not require additional training in order to generalize to new players, though it could benefit from adding knowledge of new opponents to the system offline.

Considerable work on opponent modeling has also been performed in the domain of RoboCup Soccer. For instance, Riley and Veloso [11] used explicit models of opponent actions in order to build a coach for a four-legged soccer team, with whom players would 'confer' at breaks in order to adjust their strategy. The probable locations of the opponent team members were modeled. Further, Whiteson *et al.* [15] worked on RoboCup Keepaway, where three players must keep the ball away from a fourth player. Implicit opponent models are needed in order to avoid the fourth player. They trained a group of neural networks to interact competitively on the task, with strong results.

Opponent modeling has proven useful in video games as well. In particular, Spronck *et al.* [13] and Ponsen *et al*. [10] implemented opponent modeling in Warcraft, a popular strategy game. In this domain, a form of reinforcement learning called dynamic scripting allows automated players to learn online. A number of uses for opponent modeling were discussed, including developing automated sidekicks that can better anticipate their teammates' actions, and autonomous players that can mimic their owner's action while the owner is temporarily away from the game.

The approach taken in this paper is based on NeuroEvolution of Augmenting Topologies (NEAT), developed by Stanley and Miikkulainen [14]. In this approach, only inputs and outputs are specified for the neural network. The appropriate internal topology is discovered through a search using a genetic algorithm. Connections and hidden nodes are added and changed with a given probability, and are retained in the population if they improve the performance of the network against a fitness function. In theory, the capability of the algorithm to iteratively add structure (or *complexify*) allows it to adjust to new situations without losing old capabilities. The implementation used in this paper, SharpNEAT [5], makes a minor adjustment to NEAT by adding a pruning stage once learning stagnates, whereby excess structure can be removed. It is important to note that the opponent modeling architecture proposed in this paper is independent of the particular neuroevolution algorithm. However, since NEAT has been used effectively in various game-playing approaches in the past, it was a natural choice for the opponent modeling approach as well. Supervised training algorithms such as backpropagation could not be used in Guess It, due to the lack of supervised targets for the bluff and call probabilities on the task of playing against mixture players.

3. GUESS IT

The game of Guess It consists of a fixed set of cards, 13 in our experiments. Six cards are distributed to each player, and neither

player may see the other player's cards. The goal is to identify the 13th card, which is hidden from both players. The game proceeds in turns. On their turn, each player has one of three options: (1) ask for a card they do not have; (2) bluff by asking for a card they do have; or (3) attempt to name the center card. If the player asks for a card in the opponent's possession, the opponent loses it and must reveal that card. If the player bluffs, then he must reveal the bluff card after the opponent's next turn. If the player correctly names the center card, he wins, but if he is incorrect, he loses. When the opponent asks for a card not held by the player, then either the opponent is bluffing, or the card requested is the center card. What a player expects the opponent to do is therefore central. If the opponent does not often bluff, then the player should attempt to call the bluff, *i.e.*, name the requested card as the center card. If the opponent is given to bluffing, however, then the request should be ignored. Conversely, the player must decide whether or not to bluff himself. If the opponent is expected to call the bluff (and therefore lose), then a bluff should be attempted. If the opponent will ignore the bluff, then the player cannot afford to lose a card, because that lowers his chance of winning later.

As discussed by DiPietro *et al.* [4], there are certain heuristics that every player should follow to maximize their chance of winning. If they know the center card, they should name it. If they have no cards remaining, then they should try to guess the center card, since their opponent will be able to name correctly on the next turn. Also, for computer players, the exact choice of card to play is generally irrelevant, so cards can be picked at random when bluffing or asking. Beyond these circumstances, there are two decisions a player must make: (1) whether to call a potential bluff; and (2) whether to ask or bluff otherwise. Following DiPietro, a set of basic opponents can be defined as:

Always Ask: Always asks for an unknown card; never calls, never bluffs.

Always Bluff: Always bluffs with its own card; never calls.

Call Then Ask: Calls every potential bluff; otherwise asks for an unknown card and never bluffs.

Call Then Bluff: Calls every potential bluff; otherwise bluffs with its own card and never asks.

These four opponents can be considered as the cardinal points in a two-dimensional opponent space defined by the probabilities of calling and bluffing. They are used as a cardinal set in the mixture-based approach. Each of the players is easily defeated, and all except AlwaysAsk can be defeated outright in every game. AlwaysBluff defeats CallThenBluff, CallThenBluff wins against CallThenAsk, CallThenAsk usually beats Always Ask, and AlwaysAsk defeats AlwaysBluff. Thus each one can be defeated by adopting one of the other strategies, and a player with a sufficient model of these opponents should do well against them.

4. VALIDATING THE APPROACH

To test what information about the opponent would be most useful to a player of Guess It, players were evolved given various types of information about the game state and the opponent.

Three types of information were tested, including 13 inputs representing the card status, seven inputs representing the move history, and four inputs identifying the members of the cardinal set. Card status indicates the state of each of the 13 cards, one per input, each with three possible values: held by the player, known to the player, or unknown. Move history has six inputs for each of the six possible moves made by an opponent in the game, with three possible values: ask, bluff, or call. Move history also has a seventh input indicating whether the current opponent has ever called in any past game. The card state and the move history together comprise the game state. The four input identifiers represent four basis vectors over the player space, *i.e.*, 0001, 0010, 0100, and 1000, and were used to inform the network whether it was playing against AlwaysAsk, AlwaysBluff, CallThenAsk, or CallThenBluff. Various combinations of these three input groups were tested against the four cardinal players. Note that it is possible to play optimally against each of the cardinal players if their identity is known, so the identifier should correlate most highly with success, even without the game state.

Table 1. Best fitness for players with different input information after 100 generations, averaged over 10 runs. All comparisons are pairwise statistically significant ($p < 0.05$) except Cards + IDs (Row 2) vs. IDs + Moves (Row 5) and Cards + IDs (Row 2) vs. Cards + IDs + Moves (Row 4). IDs alone provide the most useful information for the player, followed by IDs + Moves; more information beyond the IDs slows down learning, probably due to the increased complexity of training a larger network.

Inputs Used	Avg. Best Fitness, 100 Gens.
Cards	327.9 (Gen 94)
Cards + IDs	377 (Gen 97)
Cards + Moves	357 (Gen 70)
Cards + IDs + Moves	369.7 (Gen 92)
IDs + Moves	382.3 (Gen 94)
ID Only	395.8 (Gen 35)

Six populations of networks were trained with NEAT, corresponding to six different combinations of the three information sets. Each population of network players was allowed 100 generations in which to learn the task. In each generation, each player played 100 games against each of the four cardinal opponents. The networks were awarded one point for each game won and no points for each game lost; thus, a perfect fitness score would be 400 points. After 100 generations, the best fitness achieved was taken to represent the speed with which the network had learned and thus the utility of the information with which it was provided. Results were averaged over 10 trials, and a *t*-test for statistical significance was performed. All results are pairwise statistically significant with the exception of Cards + IDs vs. IDs + Moves and Cards + IDs + Moves vs. Cards + IDs. The results are given in Table 1.

These results suggest that the identifiers do indeed provide a greater advantage to the player than the move history alone. All input combinations using the IDs are better than any without it. As expected, the identifier alone without any other information performs near optimally (its fitness is rightly below 400 because it is not always possible to defeat AlwaysAsk). When other information is included as well, the networks become larger and more difficult to optimize, and thus the fitness decreases. So identifiers are the piece of information about the opponent that contributes most to the

success of the player, where such IDs are known. This result validates the general approach of using mixtures of the cardinal opponents to describe new opponents (that cannot be described directly with IDs). In Guess It, the player only needs to know the identity of its opponent in order to win.

In the above analysis, Guess It players are characterized by two parameters, call probability and bluff probability. All other aspects are played automatically to avoid illegal actions and to ensure victory when victory is immediately possible. It is worth asking whether these two dimensions adequately characterize the game. This question was tested by evolving a set of networks with total control over all aspects of the game, deciding not only when to call, ask, or bluff, but also which cards should be chosen for bluffing, asking, and guessing. Such networks could make illegal moves, but they also had the capability to recognize and employ card-specific strategies. Surprisingly, these networks could not learn to defeat the four cardinal opponents reliably, even after 10,000 generations. Compared with fitness values over 350+ in Table 1, the networks with total control attained values between 150 and 200 after 100 generations and between 300-350 after 10,000 generations. Furthermore, such networks do not play well against opponents that use card-specific strategies; the six networks from Table 1 more readily defeated a sample opponent designed to only bluff on even cards, and only ask for odd cards. These tests confirmed that it is better to use the two-parameter model of Guess It players as opposed to a more complex representation.

The decision to model previously unseen opponents as linear combinations, or mixtures, of cardinal players also needs to be validated. An alternative model would be to attempt to predict the opponent's actions directly. In Guess It, the hidden portion of the game state includes two factors: whether the opponent bluffed or asked on his last turn, and whether the opponent will call a potential bluff. To evaluate this alternative, predictor networks were trained using NEAT. NEAT was used as a training mechanism instead of backpropagation due to the likelihood that an appropriate topology would require recurrent connections, which did indeed develop during evaluation. The predictor took the game state, represented by the thirteen inputs for the card state and the seven inputs for the opponent's move history, as its input, and estimated the probability that the opponent would bluff or call as its output. Since Guess It players are described by these two parameters, the predictors were thus designed to produce a complete opponent model. A rule-based player then used the output of the predictor in order to play optimally. Fitness of the predictor network was determined as number of wins out of 400 games each generation.

Averaged over 10 trials the players with these predictors achieved a fitness of 346 after 100 generations, compared with 357 for the network in Table 1. There are reasons for this result. First, the predictor only looks at the opponent's actions in the current turn; it only implicitly describes the opponent's strategy as a whole. Second, using a rule-based player on top of the predictor meant that inaccuracies in the predictor led to faulty play; it is difficult to estimate the accuracy, or confidence, of the predictors and take it into account in making decisions. For these reasons, the mixture-based model is a more promising alternative, and will be pursued in this paper.

5. MIXTURE-BASED APPROACH

The mixture-based approach recasts opponent modeling as the problem of identifying a mixture of known cardinal strategies based on opponent play. In the case of Guess It, a mixture is a vector of four coefficients describing an opponent in terms of its similarity to each of the cardinal opponents. Players can then take this mixture as an input to influence their move decisions.

In some cases it might be possible to construct a set of rules for optimal play once the mixture is known. However, this assumes that the mixture can be identified accurately, and that the opponent space is fully characterized by the cardinal set. Since these assumptions may not be valid, players that use a mixture-based opponent model should be trained to filter out or adjust to this noise (as was demonstrated in Section 4). A process for building a game player with an explicit opponent model of this form therefore involves three steps:

1. Identify a cardinal set of opponents sufficient to describe anticipated opponents.
2. Train a neural network or some other learning platform to identify the mixture based on the opponent's play.
3. Train a player that takes as input the game state and the mixture given by the network in step 2.

This process is illustrated in Figure 1 for the game of Guess It. The two components of the game player are the Mixture Identification Module and the Decision Module. The Mixture Identification Module takes the move history of the opponent as its input and generates the four coefficients for the mixture. The Decision Module takes the game state and the mixture coefficients as its input and produces game actions as its output. During training, a bank of cardinal opponents is also necessary. The Decision Module must be trained using a sufficient sample of mixture outputs, since it must learn to deal with the noise patterns of the Mixture Identification Module.

For Guess It, the cardinal set consists of the four basic opponents described above. As described above, Guess It opponents can be represented adequately by two probabilistic parameters: the probability of calling a potential bluff and the probability of bluffing. Defined this way, all opponents must fall within the region $[0,1] \times [0,1]$. In a strict sense only two cardinal opponents would be needed to form a basis over this space (*e.g.* AlwaysBluff and CallThenAsk); all four were used in the experiments as they have been previously used with success [4]. This redundancy did not cause problems in practice.

Network inputs and outputs. The Mixture Identifier takes the move history of the opponent as input. There are at most six opponent moves, and each of these corresponds to one input node in the network. Each move is provided to the network as soon as it becomes available. The input value is 1 if the opponent bluffed on that move, –1 if the opponent asked, and 0 if the move is still unknown. A seventh input indicates whether the opponent called in any previous game; it has a value of 1 if he did and 0 if he did not. An important aspect of this input representation is that the opponent's previous move is unknown at the current move unless he asked for a card that belonged to the player. The Mixture Identifier has four outputs, corresponding to the mixture coefficients for each of the four cardinal players.

The Decision Module takes the four outputs of the mixture identifier as its input, as well as the current game state. It has two outputs, corresponding to the bluff probability and call probability. The game state for the Mixture-Based Player was represented by the seven-input move history as described above. This input set (Moves + IDs) was used because it achieved the highest fitness against the cardinal set among candidates from Table 1 that did not use only the explicit opponent IDs. Although the IDs alone performed best, IDs are not available against previously unseen opponents, and strong performance against previously unseen opponents is the goal of this research. The Mixture Identifier provides surrogates for these IDs, but in order to enable a reliably controlled experiment, it is preferable that both the control (which does not see the Mixture Identifier output) and the Mixture-Based Player (Figure 1) have the same network topology. The control needs input from the game state, since it will not have any information in the form of IDs against previously unseen opponents. Therefore both the control and the Mixture-Based Player were given access to the game state through the seven-input move history (see section 6 for experiment details).

Training the Network. Both modules were trained against a variety of mixture opponents. These opponents were created by sampling the uniform distribution on [0,1] once for each of the cardinal players, then normalizing the result to sum to 1, creating a probability distribution over the cardinal set. During game play, the mixture opponents sampled this distribution at each move, and then played the move suggested by the cardinal opponent thus chosen. Note that these mixture players are complete over the player space $[0,1] \times [0,1]$ only because all four basic opponents were included in the cardinal set. If a probability distribution over just two opponents had been used instead, it would not have been possible to produce players close to (0,0), and the region close to (1,1) would have been inadequately modeled.

The Mixture Identifier was trained with NEAT. Like the prediction task in Section 4, mixture identification is essentially a supervised task. However, since it involved a sequence of inputs and outputs, recurrent connections are typically useful; they allow the network to maintain a state, resulting in more stable output across moves [6]. Recurrency would also be necessary in other games where it is impractical to provide the entire move history as input to the Mixture Identifier. Since NEAT allows discovering the necessary recurrency automatically, it was used to construct the Mixture Identifier as well.

The population was evaluated against 400 mixture opponents in each generation. After each game, the mixtures suggested by the network at each move were averaged, and the Euclidean distance d between the average mixture and the actual distribution defining the opponent was calculated. The network fitness was then updated by $1 - d$. The maximum award was therefore 1, and the maximum possible fitness in each generation was 400. The Mixture Identifier was trained to achieve a fitness of at least 302/400, which it was able to do successfully in each of 10 trials, reaching a mean fitness of 302.48 in 466.6 generations on average.

After the Mixture Identifier was trained, it was given to the Decision Module. During each generation, 400 games were played against different mixture opponents and 50 games against each of the cardinal opponents, for a total of 600 games. For each game, the network was awarded one fitness point for each game won, so that its fitness in each generation ranged from 0 to 600. When playing against a cardinal opponent, either during training or during evaluation, the Mixture-Based Player was given the exact mixture corresponding to the cardinal opponents (*e.g.* 0001, 0010, 0100, 1000), following the assumption that the cardinal opponents represent known opponents. During training, the mixture opponents represent previously unknown opponents. The mixture-based approach was evaluated in a computational experiment similar to the validation experiments, as will be described next.

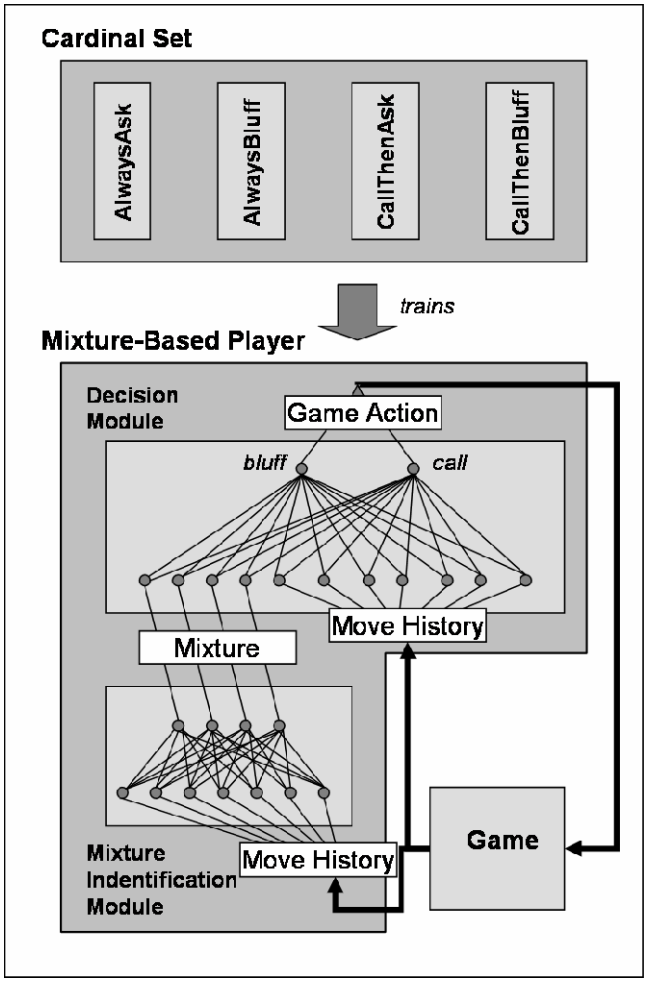

Figure 1. Mixture-Based Architecture for Opponent Modeling in Guess It. Randomly chosen mixtures of opponents in the *cardinal set* are used to train the *Mixture Based Player,* which consists of two modules. The *Mixture Identification Module* takes the history of the opponent's moves and produces an estimate of how close the opponent is to each opponent in the cardinal set. The *Decision Module* takes these mixture coefficients and the game state and generates game actions as its output. The Mixture-Based Player is only consulted when it is not possible to make an optimal move based on a heuristic; in other situations an optimal rule is invoked. The two output nodes of the Decision Module specify whether to call if possible and whether to bluff if possible; the default is to ask.

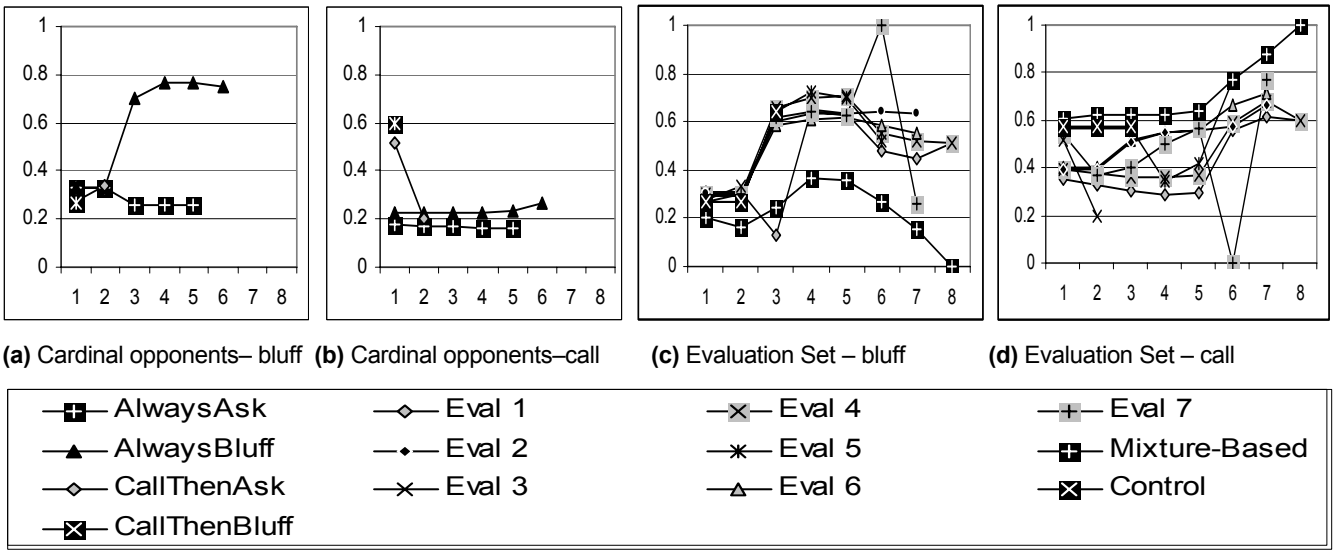

Figure 2. Opponent behavior given by the bluff and call probabilities, derived from the output of the Mixture Identifier when played as first player against the Mixture-Based Player. Horizontal axis is the sequence in which the Mixture Identifier was queried; vertical axis is the probability of either bluffing (a, c) or calling (b, d). (a) Bluff probability for the four cardinal opponents, starting out at a bias point in the Mixture Identifier and moving to their static location by round 2-3. (b) Call probability for the four cardinal opponents. Note that the Mixture-Based Player defeats CallThenAsk and CallThenBluff within two rounds, so that the bluff and call probability for these players do not have time to stabilize away from the initial bias point. (c) Bluff probability for the nine networks in the evaluation set, including the Mixture-Based Player and the Control Player. (d) Call probability for the evaluation set. Members of the evaluation set change during the game depending on the opponent's actions. Their behaviors therefore form trajectories across the opponent space, visualized above. These trajectories cover much of the space, suggesting that the set proffers a comprehensive dynamic challenge for the Mixture-Based Player.

6. MIXTURE-BASED EXPERIMENT

How well does the mixture-based approach generalize to new opponents? In order to answer this question, Mixture-Based Players were evolved in 10 independent trials, and their performance was measured and averaged. In each trial, a Mixture Identifier was first evolved, and then used to evolve the Decision Module. Each Mixture Identifier was trained to a fitness of greater than 302/400, which indicates that its mixture output is on average within a Euclidean distance of 0.25 from the actual probability distributions encountered during training. On 10 trials, the highest fitness for the Mixture Identifier was 302.92 and the lowest was 302.08; the Mixture Identifier never reached a fitness higher than 305 in unbounded training. Such imperfect performance is realistic given that opponent strategies can be generally described only approximately in most games. Each Decision Module was trained for 100 generations according to the setup described in the last section. At that point, the player averaged a best fitness of 543.1/600 (90.51% win rate, variance 0.8%) over 10 trials.

In addition to the Mixture-Based Player, another player was trained, called the "Control Player", that was in all respects identical to the Mixture-Based Player, except that it was not allowed to see the output of the Mixture Identifier, either during training or during evaluation. The training regimen for the Control was the same as for the Mixture-Based Player, consisting of 400 games against mixture opponents and 50 games against each of the cardinal opponents. Since each mixture opponent is drawn randomly, the chance of encountering exactly the same mixture twice during the same generation is low. The Control was trained for 100 generations, at which point it achieved a fitness of 562.5/600 (93.8% win rate, variance 0.7%) averaged over 10 trials. Note that the Control Player performed better in training than the Mixture-Based Player (the difference is statistically significant, $p < 0.01$). However, this measure only represents performance against the opponents encountered in training, which follow the same static mixture strategy throughout the game. It does not reflect how well the player would perform against more realistic, dynamic opponents that may vary their strategy according to their opponent. With static opponents, the move history contains sufficient information to play well; the mixture inputs require the Decision Module to learn a more complex, higher-dimensional function, which it does less accurately.

In order to evaluate the performance of both players against unseen, dynamic opponents, both players played 20,000 games against a bank of eleven evaluation opponents, half as first player and half as second. Their performance score was then the win probability against all opponents, that is, the win percentage over all 220,000 games played. The eleven evaluation opponents included the four cardinal opponents as well as the seven networks trained to validate the mixture-based approach, that is, the six networks whose fitness was given in Table 1 plus the network trained to predict opponent actions. Thus, the evaluation opponents represent a diverse set of dynamic strategies, some of which are trained to adapt to their opponent's play (*e.g.* the predictor, as well as the networks that receive the move history as an input).

To illustrate their diversity, the opponents are mapped in terms of their call and bluff probability in Figure 2. Since there is no way to

arrive at estimates of these probabilities directly, the output of the Mixture Identifier during play by the Mixture-Based Player is used as a proxy for these values. Note that the Mixture Identifier does not acquire sufficient information to correctly categorize the opponents in the first two rounds of the game. Figure 2a shows the cardinal players. In the first round, AlwaysAsk and AlwaysBluff appear identical, but in later rounds enough information becomes available to correctly place them in the static locations where they belong. The positions of CallThenAsk and CallThenBluff cannot be correctly ascertained by the Mixture Identifier, since these strategies are defeated by the player in the second and first rounds, respectively. Figure 2b shows an estimate of the trajectories formed by the other members of the evaluation set. Unlike in Figure 2a, the predictions of the Mixture Identifier do not stabilize after a few rounds. Furthermore, the various players move through diverse regions of the player space. These results demonstrate that the evaluation set is indeed distributed broadly throughout opponent space, and thus the evaluation set represents a realistic, dynamic challenge for assessing the generalization performance of the Control and Mixture-Based Player to previously unseen opponents.

Averaging results from 10 trials of such evaluations, each with separately trained players, the Mixture-Based Player won 71.3% of its games, and the Control Player won 57.7%. With the four cardinal opponents removed from consideration, the mixture-based player won 61.5% of its games against the remaining truly unknown opponents, whereas the Control Player won 54.6%. Thus the player with the Mixture Identifier won approximately 44,000 more games than the control, and was 11.3% more likely to win against a previously unseen opponent.

Furthermore, in a separate evaluation over the same 10 trials, the Mixture-Based Player played 20,000 games directly against the Control Player from the same trial. The Mixture-Based Player won 77.6% of these games on average, showing that the Mixture-Based Player has a clear competitive edge over the Control; in fact, in only one trial did the Control Player outplay the Mixture-Based Player. All the above results are statistically significant ($p < 0.05$), demonstrating that the Mixture-Based approach is an effective way to model behavior of novel opponents.

7. DISCUSSION

The results in this paper demonstrate that a cardinal set of opponents can be used to train an explicit opponent model that conveys a material advantage to its bearer. Although Guess It is a simple game, there is no obvious reason why these results would not apply to a more complex game as well. The mixture-based architecture in Figure 1 generalizes readily to almost any game setting and therefore provides a promising starting point for further work in opponent modeling.

Using probability distributions to generate mixture opponents forced a four-dimensional representation for the opponent space, whereas if call and bluff coefficients had been used, only two cardinal opponents would have been needed. As discussed above, this decision was made because the use of these four opponents had been effective in the past. In this work, however, the only effect of this choice in Guess It is that it skews the topology of the opponent space around (0, 0) and (1, 1). Coefficient-based representations should be analyzed in future work, particularly because the dimensionality of the opponent space will be larger for more complex games.

This work proceeds on the assumption that opponents in Guess It can be modeled as points in the control space of $[0,1] \times [0,1]$. Using a linear combination of cardinal opponents to represent different opponents implies that the opponents are uniformly distributed over the space. However, in reality, there are most likely regions of high and low density around points that correspond to effective and ineffective strategies, respectively. Ideally, the mixture identifier should be trained to high accuracy in more dense regions in order to provide the greatest advantage to the Decision Module against most opponents.

Another assumption is that static points in the control space can adequately describe opponents in the opponent space. However, as seen in Figure 2, opponents are more likely to appear as trajectories in this space. To the extent that these trajectories can be described in terms of a parameter set, even very complex opponents could be modeled as static points in an expanded space with both controls and parameters as dimensions. It is uncertain, however, whether such a more complex representation is necessary. In these experiments, the mixture-based players were evaluated against opponent networks with up to 24 input parameters, yet the two-dimensional representation of opponents (as points in $[0,1] \times [0,1]$) still provided a significant advantage. Moreover, the Mixture-Based Player was able to play dynamically against a changing opponent strategy. Against a human opponent, the human shifted strategy midway through the game from AlwaysBluff to AlwaysAsk. As a result, the Mixture-Based Player made a corresponding adjustment to its strategy within two rounds. This change occurred because the Mixture Identifier was trained to recognize its opponents on average. In actual play, an opponent occasionally exhibits behavior that suggests a distribution significantly different from its actual distribution. The more moves the opponent made, however, the less likely such deviations would become. Therefore, Mixture Identifiers that are capable of averaging out such deviations are preferred in evaluation. As a side effect, Mixture Identifiers learned to respond to shifts in opponent play during the game.

8. FUTURE WORK

The results presented in this paper are encouraging, suggesting that explicit modeling of previously unseen opponents is possible using the mixture-based approach. Significantly more work can be done to extend and further verify these results in more complex games. Poker and other card games are an obvious next step, since sophisticated play typically requires recognizing and taking advantage of the playing style of the opponent in order to achieve success. The primary issue, for card games as well as more complex games in general, is how a sufficient cardinal set can be identified.

One potential approach would be to delineate a method for determining a set of control and parameter dimensions that define the opponent space. The bases of this space could then be used to generate a cardinal set directly. It is unclear whether such cardinal opponents would constitute realistic opponents, and it is highly unlikely that the structure of the generated opponent space would reflect the distribution of actual opponents.

In games where real opponents are available for training, a different approach can be used. Poker, for instance, has a vast online community where potential algorithms can be tested against human opponents, as was done in by Billings *et al.*, [1]. In such an environment, it might be possible to use *e.g.* a self-organizing map

[8] to categorize opponents. A SOM learns a topological map of its input space, extracting the most descriptive dimensions and neighborhood structure automatically. A SOM of opponent space could be used for selecting mixture opponents that accurately represent the structure of the opponent space.

On the other hand, the current approach does not store information about the opponent between games; each game is treated independently. An interesting direction for future work would be to extend the architecture to model the opponent over a sequence of games. Such an extension should work well in repeated games such as poker, chess, and *e.g.* sports video games.

9. CONCLUSION

The mixture-based approach to explicit opponent modeling allows identifying and defeating previously unknown opponents by representing them as a mixture over a set of known cardinal opponents. The approach is effective, especially in games where players are unable to see the complete state of the game and therefore must infer their situation based on the opponent's play. Experiments in the game of Guess It show that the mixture-based approach generates automated players that continuously diagnose the strategy of their opponents and dynamically respond to an opponent's changing play. The same process should apply to a variety of interesting games, thus making opponent modeling an integrated part of game-playing AI in the future.

10. ACKNOWLEDGEMENTS

This research was supported in part by NSF under grant EIA-0303609.

11. REFERENCES

[1] Billings, D., Papp, D., Schaeffer, J., and Szafron, D. Opponent Modeling in Poker. *Proceedings of 15th National Conference of the American Association on Artificial Intelligence*. AAAI Press, Madison, WI, 1998, 493-498.

[2] Davidson, A., Billings, D., Schaeffer, J., and Szafron, D. Improved Opponent Modeling in Poker. *Proceedings of the 2000 International Conference on Artificial Intelligence (ICAI'2000)*. 1999, 1467-1473.

[3] Davidson, A. Using Artificial Neural Networks to Model Opponents in Texas Hold 'Em. Unpublished manuscript; *http://spaz.ca/aaron/poker/nnpoker.pdf*. 1999.

[4] DiPietro, A., Barone, L., and While L. Learning In RoboCup Keepaway Using Evolutionary Algorithms. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*. Kaufmann, San Francisco, 2002, 1065-1072.

[5] Green, C. Phased Searching with NEAT: Alternating Between Complexification and Simplification. Unpublished manuscript; *http://sharpneat.sourceforge.net/phasedsearch.html*. 2004

[6] Gomez, F. and Miikkulainen, R. 2-D Pole Balancing with Recurrent Evolutionary Networks. *Proceedings of the International Conference on Artificial Neural Networks* (ICANN-98). Springer, Berlin, 1998, 425-430.

[7] Isaacs, R. A card game with bluffing,. *The American MathematicalMonthly*, vol. 62. 1955, 99-108.

[8] Kohonen, T. The Self-Organizing Map. *Proceedings of the IEEE*, vol. 78. 1990, 1464-1480.

[9] Korb, K., Nicholson, A., and Jitnah, N. Bayesian Poker. *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI-99)*. 1999, 343-350.

[10] Ponsen, M., Munoz-Avila, H., Spronck, P., and Aha, D. Automatically Generating Game Tactics via Evolutionary Learning. *AI Magazine, 27(3)*. AAAI Press, Madison, WI, 2005, 75-84.

[11] Riley, P., and Veloso, A. Planning for Distributed Execution Through Use of Probabilistic Opponent Models. *IJCAI-2001 Workshop PRO-2: Planning under Uncertainty and Incomplete Information*. 2001.

[12] Southey, F., Bowling, M., Larson, B., Piccione, C., Burch, N., and Billings, D. Bayes' Bluff: Opponent Modeling in Poker. *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI-05)*. 2005, 550-558.

[13] Spronck, P., Ponsen, M., Sprinkhuizen-Kuyper, I., and Postma, E. Adaptive Game AI with Dynamic Scripting. *Machine Learning, 63*. Kluwer, Hingham, MA, 2005, 217-248.

[14] Stanley, K. and Miikkulainen, R. Continual Coevolution Through Complexification, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*. Kaufmann, San Francisco, 2002, 113-120.

[15] Whiteson, S., Kohl, N., Miikkulainen, R., and Stone, P. Evolving RoboCup Keepaway Players through Task Decomposition. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2003)*. Kaufmann, San Francisco, 2003, 356-368.