Recognizing Probabilistic Opponent Movement Models

Patrick Riley and Manuela Veloso

Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213-3891

Abstract. In multiagent adversarial domains, team agents should adapt to the environment and opponent. We introduce a model representation as part of a planning process for a simulated soccer domain. The planning is centralized, but the plans are executed in a multi-agent environment, with teammate and opponent agents. Further, we present a recognition algorithm where the model which most closely matches the behavior of the opponents can be selected from few observations of the opponent. Empirical results are presented to verify that important information is maintained through the abstraction the models provide.

1 Introduction

Multiagent domains can include team and adversarial agents. Our work is driven by the goal of significantly improving the performance of teams of agents through their adaptation and effective response to different adversarial teams.

We use the concrete simulated robotic soccer platform, which is a rich multiagent environment including fully distributed team agents in two different teams of up to eleven agents. Extensive work has been done on adapting the teammates' behavior to their opponents, mainly at the individual low-level of positioning and interactions between a small number of agents [5]. The procedure for these positioning adaptations does not change throughout a game, limiting the degree to which the team adapts to the opponent team's behaviors.

This paper reports our work on going one additional major step towards the adaptive response of teammates to the opponent, by gathering and responding to the opponents' behaviors throughout the game. We specifically focus on responding effectively after the game is stopped, in what are known as setplay situations. Several preset setplay plans [5] have been introduced which indeed provide great opportunities to position the teammates strategically and have been shown to impact the performance of a team. In this work, we contribute *adaptive* setplays which change and improve throughout a game as a function of and in response to the opponent team's behavior.

We use a "coach" agent with a centralized view of the world, but whose only action is to communicate with its teammates, but only at occasional times. In the robotic soccer domain, these times are when the game is stopped for some reason. This type of scenario is one instance of the *Periodic Synchronization Domains* [6] where team agents can periodically synchronize their team strategies.

Our coach agent compiles the necessary overall view of how the opponent team behaves. The coach communicates to its teammates a team plan which is executed and monitored in a fully distributed manner. The coach agent is equipped with a number of pre-defined opponent models. These models are probabilistic representations of predicted opponents' locations. The models can then be matched to observed movements of the opponent agents. The coach continues to gather observations and when the game is stopped, e.g., due to an out-of-bound call, the coach rapidly takes advantage of the short available time to create a team setplay plan that is a function of the matched modeled opponents' behavior. The plan is generated through a hill-climbing search in plan space using an evaluation function that embeds the predictions of the opponent model perceived to be the most likely during the game. The plan generation, in addition to the recognition of the opponents' model, notably uses the model *to predict* the opponent agents' behaviors. The coach also observes the execution of the plan generated in order to update the selection of an appropriate model for the current opponent. This paper focuses on the structure of an opponent model and the algorithm for recognizing the best-matching model. A more complete overview of the system can be found in [3] and more details on the planning can be found in [4].

2 Opponent Models and Model Selection

Conceptually, we want an opponent model to represent how an opponent plays defense during setplays. We expect that a wide range of decision making systems can be roughly captured by a small set of models. In order to handle the uncertainty in the environment and to allow models to represent more information than just a simple predicted location, we use probabilistic models. Given the recent history of the ball's movement (from the start of the set play for example) and the player's initial location, the model should give, for each player, a probability distribution over locations on the field.

Let p be the number of players on a team. Let F be the set of positions on the field, discretized to 1m. We represent the ball movement as a sequence of locations on the field (an element of F^*). A location for each player is a sequence of positions on the field (an element of F^p).

An opponent model defines a probability distribution for each player over the end locations on the field for that player, given the ball movement $B \in F^*$ and a set of starting positions $S \in F^p$ for all the players. If R is a probability distribution over locations on the field, an opponent model M is a function:

$$M: \underbrace{F^*}_{\text{ball movement}} \times \underbrace{F^p}_{\text{initial pos.}} \rightarrow \underbrace{R^p}_{\text{probability distribution for each player}} \qquad (1)$$

In particular, for an opponent model M, the probability for player i being at end location ℓ_i ($P_i[\ell_i|B,S]$) is calculated by choosing the ith probability distribution output by the model and calculating the probability of the location ℓ_i according to that distribution.

Notice that each player's distribution may depend on the starting positions of *all* the opponent players. In order to avoid recursive modelling, we do not allow the opponents distributions to depend on our own player locations. The ball movement B is the planned ball movement. The starting positions for the players are their current locations. We have a probability distribution for each player, which is the probabilistic obstacle for planning.

Given a set of opponent models, it is still a challenge to decide which model best describes the opponent. We assume that the opponent has chosen one of our models at the beginning of the game and is then independently generating observations from that model. We can then use a naive Bayes classifier.

We maintain a probability distribution over the models. The original distribution (the prior) is set by hand. Then, whenever a planning stage is entered, the model with the highest probability is used. Upon observing a plan execution, we use observations of that execution to update our probability distribution over the models.

What this means is that we start with a probability distribution over the set of models $\{M_1,\ldots,M_m\}$ (known as the *prior*) and then we get an observation. An observation is a triple of starting locations for all the players $S \in F^p$, a ball movement $B \in F^*$, and ending locations for all of the players $E \in F^p$. We want to use that observation to calculate a new probability distribution, the *posterior*. That distribution then becomes the prior for the next observation update.

We make the following assumptions, in order to simplify our equations:

1. The players movements are independent. That is, the model may generate a probability distribution for player x based on everyone's starting locations. However, what the actual observation is for player x (assumed to be sampled from this probability distribution) is independent from the actual observations of the other players.
2. The probability of a particular starting position and ball movement are independent of the opponent model. This assumption is questionable since the ball movement (i.e. plan) generated depends on the opponent model.

Consider one updating cycle of getting an observation $\omega = (S,B,E)$. We want $P[M_i|\omega]$.

$$\begin{aligned}
P[M_i|\omega] =& \frac{P[\omega|M_i]P[M_i]}{P[\omega]} = \frac{P[S,B,E|M_i]}{P[\omega]}P[M_i] = \frac{P[E|S,B,M_i]P[S,B|M_i]}{P[\omega]}P[M_i] \\
=& P[E|S,B,M_i]\frac{P[S,B]}{P[\omega]}P[M_i] \quad \text{(assump. 2)} \\
=& \underbrace{P[\ell_1|S,B,M_i]P[\ell_2|S,B,M_i]\ldots P[\ell_p|S,B,M_i]}_{\text{what opponent model calculates}} \; \underbrace{\frac{P[S,B]}{P[\omega]}}_{\text{norm. constant}} \; \underbrace{P[M_i]}_{\text{prior}} \quad \text{(assump. 1)}
\end{aligned}$$

The term labeled "norm. constant" is a normalization constant, i.e. it does not depend on which model is being updated, so it does not need to be explicitly calculated. We calculate the remaining terms and normalize the result to a probability distribution.

3 Empirical Evaluation

In order to verify that the abstracted form of models given here are still sufficient to reliably capture variation, we implemented several different movement recipes for the defense. We then implemented models to capture each of these effects. This is a nontrivial task, both because of the great deal of uncertainty in the environment, and the abstraction which the models provide over the basic actions. An empirical evaluation was necessary to verify that our model implementations match the effects of the players and that the naive Bayes update can quickly recognize models of this form.

Besides the choice of the models, a decision must be made about how to generate observations from the stream of data being received about the opponent positions. Clearly, if an observation triple is generated every cycle we will be violating the independence assumption of the naive Bayes update, as well as giving the models little information (in terms of the ball movement) with which to work. For this testing we decided to create an observation for processing every time the agent who is controlling the ball changes. An agent is considered to be controlling the ball if (i) the agent is the closest player to the ball and (ii) the player can kick the ball. A given observation can cover anywhere from approximately 5 to 50 cycles.

In all of the models used, the distribution of each players final position is represented by a 2-dimensional Gaussian with equal variance in all directions. The standard deviation is an affine function of time (since the beginning of the setplay); we tested with several different slopes. The mean is computed as discussed below. This is likely not the optimal distributional representation, but it suffices for recognition task here.

We used five models for the empirical evaluation. Naturally, the mean of each player's final distribution is computed relative to the initial position as follows:

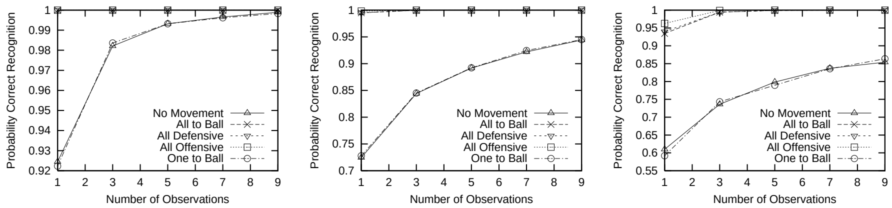
No Movement At the initial position of the player
All to Ball Moved towards the ball at a constant speed
All Defensive Moved towards the defensive side of the field at a constant speed
All Offensive Moved towards the offensive end of the field at a constant speed
One to Ball This model is slightly different from the others. The ball's movement is broken down into cycles. At each cycle, whichever player is closest to the ball is moved slightly closer. Note that since the ball can move faster than the players, which player is closest to the ball can change several times during the ball's movement. The final positions of the players are the means of the distributions.

Before looking at how well each model can be recognized out of this set of models, it is important to understand how "separable" the models are. That is, if two models make similar predictions in most cases, it will be difficult to recognize one model over the other. Clearly, getting more observations should improve the recognition accuracy.

We will develop the concept of separability in three stages. First we will consider the separability of two distributions, then the separability of multiple *sets* of distributions, and finally define empirically the separability of a set of models.

Start with two distributions A and B over two dimensions. The question we are interested in is if we are seeing an observation probabilistically generated from A, what is the probability that the naive Bayes update (starting with a uniform prior) will return with A being the most likely distribution? This is equivalent to the probability mass in A of the area where the probability distribution function (i.e. p.d.f.) of A if greater than the p.d.f. of B. Of course, we are also interested in the case where B is the distribution generating observations, and in general these probabilities can be different.

Now consider seeing multiple observations instead of a single one. Once again, we are interested in the probability that A will have the highest posterior after the naive Bayes update on all of the observations. This is challenging to solve analytically, but Monte Carlo simulation can estimate this probability.



(a) St. Dev. Rate 0.1 (b) St. Dev. Rate 0.3 (c) St. Dev. Rate 0.5

Fig. 1. Separability of the models (Error bars are one standard deviation)

This concept of separability extends naturally to the update over the distributions of all the players and over all of the models. Given a distribution for each player from each model, we can use Monte Carlo simulation to compute the probability that the correct set of distributions will be recognized for any given number of samples.

However, the models are functions from starting positions of the opponents and a movement of the ball to distribution of the opponents, not simply distributions themselves. We use an empirical test to estimate separability of *models*. For each observation $O_1, \ldots O_k$ from real game play, we repeatedly generate a series of artificial observations $(S,B,E_1)\ldots(S,B,E_n)$ (where n is the number of observations for which we want to estimate the probability of correctness). The sequence of ending positions $E_1 \ldots E_n$ are sampled from the distributions output by the correct model (the model for which we want to estimate the probability of the naive Bayes being correct). For each sequence of artificial observations, the update is performed. We can then estimate the probability that the correct model will be the most likely. Averaging over all observations, we can estimate the probability of a model being correctly recognized, given n observations.

Figure 1 shows, for each model, the probability of that model being recognized as a function of the number of observations. The three graphs are for three different choices of the slope of the line controlling the increase in the standard deviation of the distributions over time. One can see that if the models perfectly capture the opponents, after only a few updates, the recognition accuracy should be quite high (more than 80%). Also note that since a lower rate of standard deviation increase means less overlap in the distribution, lower rates give better separability.

The testing of recognition accuracy is much simpler compared to estimating the separability of the models. For each number of observations n, we examine contiguous sets of n observations from real game play (where the opponents are using a recipe which should reflect predictions of the models). For each set, we perform the naive Bayes update and check whether the correct model has the highest posterior. From this sequence of binary answers, we estimate the probability of the model being recognized.

Figure 2 summarizes the recognition accuracy of the models for each of the choices of the standard deviation increase rate. With standard deviation rate of 0.3 or 0.5, an accuracy of 94% is obtained with just four observations. The fact that the recognition is much lower for a standard deviation increase rate of 0.1 suggests that the true standard deviation of the players' positions is higher than predicted by the model.

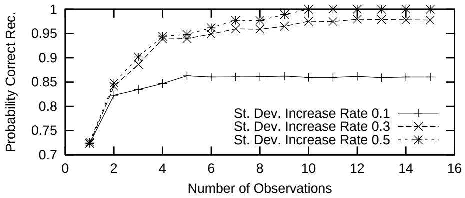


Fig. 2. Summary of Recognition Accuracy

4 Discussion

Similar work is by Intille and Bobick[2], who give a framework for recognizing plays in American football. The plays are given in terms for goals for the agents and limited temporal constraints between actions. Further, similar to Charniak and Goldman [1], they use Bayesian style networks to incorporate observed information.

One important difference between ATAC and other plan and behavior recognition work is that we use the recognized information for purposes for predicting and adapting to *future* behavior. We assume that the opponents will behave similarly to how they performed in the past, and use that information to develop a plan.

We fully implemented this system for RoboCup2000, and it was evident that our team benefited from adaptive setplays. We hope to explore automated learning of the models and providing more flexibility to take advantage of unpredicted opportunities. We plan to continue to explore the area of generating and using opponent models further exhibit adaptive, intelligent responses in our agents.

Acknowledgments: The authors thank Michael Cleary and Deb Dasgupta at Draper Laboratory. This research was sponsored in part by Grants Nos F30602-00-2-0549 and F30602-00-2-0549 and by an NSF Fellowship. The content of this publication reflects only the position of the authors.

References

1. E. Charniak and R. Goldman. A Bayesian model of plan recognition. *Artificial Intelligence*, 64(1):53–79, 1993.
2. S. Intille and A. Bobick. A framework for recognizing multi-agent action from visual evidence. In *AAAI-99*, pages 518–525. AAAI Press, 1999.
3. P. Riley and M. Veloso. Coaching a simulated soccer team by opponent model recognition. In *Agents-2001*, 2001.
4. P. Riley and M. Veloso. Planning for distributed execution through use of probabilistic opponent models. In *IJCAI-2001 Workshop PRO-2: Planning under Uncertainty and Incomplete Information*, 2001.
5. P. Stone, P. Riley, and M. Veloso. The CMUnited-99 champion simulator team. In Veloso, Pagello, and Kitano, eds, *RoboCup-99: Robot Soccer World Cup III*, pages 35–48. Springer, Berlin, 2000.
6. P. Stone and M. Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 110:241–273, 1999.