Learning Opponent Strategies through First Order Induction

Katie Genter
Dept. of Computer Science
The Univ. of Texas at Austin
Austin, TX 78712 USA
katie@cs.utexas.edu

Santiago Ontañón
IIIA - CSIC
Bellaterra, Spain
santi@iiia.csic.es

Ashwin Ram
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332 USA
ashwin@cc.gatech.edu

Abstract

In a competitive game it is important to identify the opponent's strategy as quickly and accurately as possible so that an effective response can be planned. In this vein, this paper summarizes our work in exploring using first order inductive learning to learn rules for representing opponent strategies. Specifically, we use these learned rules to perform plan recognition and classify an opponent strategy as one of multiple learned strategies. Our experiments validate this novel approach in a simple real-time strategy game.

Introduction

Most competitive games, ranging from professional soccer to StarCraft, require participants to have a solid strategy to be successful. An effective way to defeat an opponent is often to identify their strategy, as then their actions can be predicted and an effective response can be organized.

One way to identity an opponent's strategy is through plan recognition (Schmidt, Sridharan, and Goodson 1978). Plan recognition is the process of inferring the plan of an intelligent agent from observations of the agent's actions or the effects of those actions. Plans can be represented in various forms. In this paper we focus on learning opponent strategies that are composed of collections of rules. Many different algorithms could be used to learn these rules (Fürnkranz 1999), but we chose the first order inductive learning (FOIL) algorithm (Quinlan 1990) because of its ability to take in predicates and produce a list of rules ordered by their Laplace accuracy.

There is previous work in both rule learning and plan recognition in the context of games (e.g. Kabanza et al. 2010, Molineaux, Aha, and Sukthankar 2009). However, to the best of our knowledge, ours is the first to present an inductive logic programming approach to plan recognition. In particular, the FOIL algorithm takes gameplay traces of a strategy as input and generates a set of rules representing that strategy. After learning multiple strategies, we can predict which learned strategy an opponent is following by comparing the rules for the opponent's trace against the learned rules for previously seen strategies.

Learning Strategies through FOIL

A *trace* is created every time a complete game is played. Specifically, a trace consists of a list of triples, where each triple contains a time stamp, game state, and a set of actions. The list of triples represents the evolution of the game and the actions executed during different time intervals.

Traces are difficult to input and use by FOIL in their original form, so we translate the relevant information from each trace into a more usable form. In particular, we define a set of attributes $F = \{f_1, ..., f_n\}$ to be extracted from each entry as either true or false. In this way, each entry S_i is translated to a set of identifiers $F(S_i)$ that correspond to all of the attributes in S_i. We can then easily input these sets of identifiers into the FOIL algorithm as a modified trace $T' = [\langle t_1, F(S_1)\rangle, ..., \langle t_n, F(S_n)\rangle]$.

Using Rules to Represent Strategies

After gathering traces that demonstrate specific opponent strategies, we use FOIL to learn rules to represent each specific strategy. For each strategy, we consider each attribute f_i separately. Given an attribute f_i and a modified trace T', a training example can be generated for each instant t_j (except for t_1). In particular, if f_j is true in $F(S_j)$, then we will create a *positive* example $e_j = \langle F(S_{j-1}), +\rangle$. Likewise, if f_j is false, then we create a *negative* example.

Using the set of training examples collected for an attribute f_i, FOIL produces a set of rules which predict whether attribute f_i will be true in the next cycle given the current game state. FOIL is applied to each attribute f_i independently to generate a rule set for each attribute. Together, these rule sets represent a specific opponent strategy.

Identifying Opponent Strategies

We can use learned strategies to predict what strategy an unknown opponent is utilizing given a trace of its actions in the world. Given a trace in which we want to identify the opponent's strategy, we gather positively correlated rules that characterize the opponent's movements and actions in the world as described in the previous section. We then compare these rules to the positively correlated rules of each learned strategy and determine which learned strategy is closest to the opponent's behavior. Specifically, we assign a score to represent the similarity between rules representing a known strategy and rules representing the unknown opponent's behavior. The strategy that earns the highest score represents

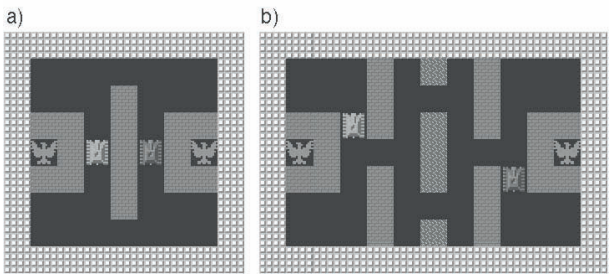

Figure 1: The two BattleCity maps used in our evaluation.

the most likely strategy of the unknown opponent.

Experimental Evaluation

To evaluate our technique, we use a two-player real-time strategy game called BattleCity (see Figure 1). In this game, each player navigates their tank around a maze while trying to simultaneously destroy the opponent's tank or base and defend their own tank and base. Bases are static, but tanks can move in four directions and fire bullets in whichever direction the tank last moved. The environment is fully observable and the only sources of non-determinism in the game are the actions of the opponent. The terrain consists of land, water (which can be shot across, but not traversed), indestructible walls, and walls that can be shot through.

We use two different maps in our evaluation: a simple map (Figure 1.a) and a more complex map (Figure 1.b). For each map, we recorded traces demonstrating three distinct strategies. We recorded 14 winning traces for each strategy, making a total of 42 traces per map. In each trace, a human tester controlled the tank protecting the left base, while the other player was controlled by the built-in artificial intelligence of the game. The human tester attempted to exhibit a particular playing strategy in each trace. The three strategies considered in the simple map (Figure 1.a) were:

'Direct' : turn right from the start and fire through the walls towards the opponent base until it is captured.

'SouthernRoute' : travel down and around the center wall, approach the opponent base from the bottom, and then fire through the wall to capture the base.

'NorthernRoute' : is similar to strategy two. However, in this case travel up and around the wall, approaching the base from the top.

The three strategies considered in the complex map (Figure 1.b) were:

'Direct' : position the tank across from the opponent base and fire across the water and through the wall until the base is captured.

'SouthernBridge' : travel down until aligned with the lower bridge. Then fire through walls and travel across the bridge before approaching the base from the bottom. Then fire through the wall and capture the base.

'NorthernBridge' : is similar to our second strategy, except now take the upper path and approach the opponent base from the top.

Map	Strategy Exhibited	Percent Correct
Simple	Direct	10/10
Simple	SouthernRoute	10/10
Simple	NorthernRoute	9/10
Complex	Direct	10/10
Complex	SouthernBridge	10/10
Complex	NorthernBridge	8/10

Table 1: The number of times each strategy was correctly identified on the simple map (Figure 1.a) and the complex map (Figure 1.b).

For each map, we divided the 42 full-game traces into two sets: 12 traces (4 of each strategy) constituted the training set and 30 traces (10 of each strategy) constituted the test set. We learned rule sets off-line for each strategy using the traces in the training set. We then used these rules to classify each of the 30 unseen traces as one of the three learned strategies. Results can be found in Table 1.

It is important to note that although the human tester tried to demonstrate the same strategy in each trace recorded for a particular strategy, traces exhibiting the same strategy are often rather different due to uncertainty introduced by the opponent. For example, the enemy might block the path the tester is trying to move through, causing a delay or slight deviation from the prescribed strategy. Lastly, it is important to note that the attributes used in our evaluation were not fine tuned. In fact, the same set of attributes were used in a previous paper (Ontañón et al. 2009).

Future Work

One future step is to evaluate our approach in domains that are more complicated than BattleCity. Additionally, other approaches for identifying opponent strategies should be considered. It could also be beneficial to attempt to learn opponent strategies from losing traces. Finally, learned strategies could be used in an on-line fashion to predict the next move that will be made by the adversary in real-time. Strategies learned from partial traces could potentially be used for this since predictions would be made mid-game.

References

Fürnkranz, J. 1999. Separate-and-conquer rule learning. *Artificial Intelligence Review* 13:3–54.

Kabanza, F.; Bellefeuille, P.; Bisson, F.; Benaskeur, A. R.; and Irandoust, H. 2010. Opponent behavior recognition for real-time strategy games. In *AAAI Workshop on Plan, Activity, and Intent Recognition*.

Molineaux, M.; Aha, D.; and Sukthankar, G. 2009. Beating the defense: Using plan recognition to inform learning agents. In *Proceedings of Florida Artifical Intelligence Research Society*.

Ontañón, S.; Bonnette, K.; Mahindrakar, P.; Gómez-Martín, M.; Long, K.; Radhakrishnan, J.; Shah, R.; and Ram, A. 2009. Learning from human demonstrations for real-time case-based planning. In *IJCAI Workshop on Learning Structural Knowledge From Observations*.

Quinlan, J. R. 1990. Learning logical definitions from relations. *Machine Learning* 5:239–266.

Schmidt, C.; Sridharan, N.; and Goodson, J. 1978. The plan recognition problem: An intersection of psychology and artificial intelligence. *Artificial Intelligence* 11(1-2):45–83.