Reinforcement Learning in the Robosoccer Domain with Monte Carlo Methods

Anderson Thomas Collin Duncan Samsara Counts

The George Washington University

Abstract

We describe a reinforcement learning algorithm capable of training pairs of agents to handle the robotic soccer domain in a playoff format using Reinforcement Learning with Monte Carlo methods. We then explore how we adapted reinforcement learning methods for policy evaluation and action selection in this distributed, real-time, partially observable, noisy domain in a playoff format. We adapted that strategy to model, on top of Simha's framework, a series of playoffs between teams composed of pairs of players. We then present a sample of our results to demonstrate that a learned policy can perform comparably to hand-coded policies provided by Dr. Simha.

I. Literature Review

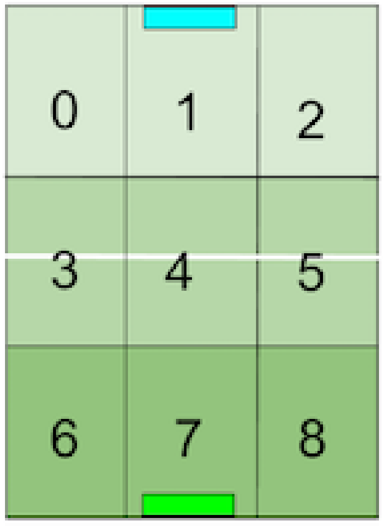
i. Introduction

In 1997, several researchers, among them Sanderson and Kitano, et. al., proposed Robotic soccer as a new domain for exploration in Machine Learning research focused on creating algorithms for gameplay [Kitano et. al., 1997][Sanderson, 1997]. This proposal was a consequence of the enormous growth of the Artificial Intelligence research: in the past 20 years, human experts in complex, finite games such as chess and Go have been defeated by algorithms. Within the same year, the RoboCup competition emerged, a tournament between robotic soccer players who use different Machine Learning algorithms to learn and compete in the game of soccer.

ii. Evolution of Our Methodology

In deciding upon our model for a small instance of Robotic soccer playoffs, we initially considered using the 3 vs. 2 keepaway domain, a subproblem of robotic soccer domain [Stone et. al.] We modeled our problem using Stone's 3-2 keepaway algorithm for offensive playing, with the goal as the third player. The parameters we picked were the Euclidean distance between players on the same team, along with the angles between players. However, we encountered difficulty when when we realized the state space was infinite, due to the nature of distance and theta as continuous values.

Refining our model, we then decreased the size of our state space by discretizing the field with a 3x3 grid. Furthermore, we defined four parameters of each state to take a finite range of values.



Hence the locations of our grid = $\{0, 1, 2, 3, 4, 5, 6, 7, 8\}$.

II. Methods

In our algorithm we adapt Dr. Simha's soccerJar framework to support a playoff between teams of player pairs that learns a strategies with Reinforcement Learning using Monte Carlo methods. Inherently, this represents successive choices of the actions as a Markov Decision Process, where the game is broken down into episodes with a goal scored as our terminal state. Ultimately, we chose to represent the State Space S as a function of four parameters: $S(B_s, T_{HB}, T_L, O_L)$, where:

- $B_s = \{-1, 0, 1\}$ represents a set of states of the ball
- $T_{HB} = \{T, F\}$ is a boolean that records whether our team has possession of the ball
- $T_L = \{(0,0)...(8,8)\}$ where each t_L is a tuple of each player on our team's location on the field
- $O_L = \{(0,0)...(8,8)\}$ where each o_L is a tuple of each player on our opponen's team's location on the field

The cardinality of our state space is $|S| = 3 \cdot 2 \cdot 2^9 \cdot 2^9 = 1572864$ states.

We define our high-level actions $a \in A$ as:

- moveToBall() directs the agent to move towards the ball, whether stationary or moving.
- holdBall() directs the agent to stand still while holding the ball, if on defense, we...
- passBall() pass the ball to player p
- blockPass() where we intercept a pass from opposing players.

Move to Ball, Pass the Ball, and Hold Ball are fairly straightforward.
Block Pass directs the player to move between the opponent with the ball and another opponent that we expect to receive it. The method includes getOpen(), which used to be another one of our actions that we dropped to reduce the size of the state space. Then, we move to a position that is away from opponents towards the goal.
The reward r for a given time step is determined by the previous time step?s reward minus the reward at this time step. We can reward based on:

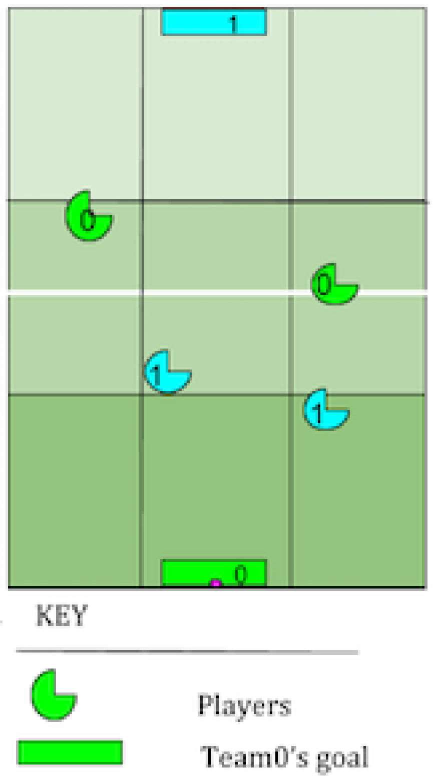
- Our team scored goal and we return $r = 1$

- Other team scored goal and we return $r = -1$
- else we return $r = 0$

We refer to the players on our team as P and the opposing players as O. P_1 is the player with the ball. The following are our key functions:

- $Q_k(s, a$ accepts an action and state and returns the value of taking that action in that state at step k.
- π is a policy, a stochastic strategy that chooses an action $a \in A$ for a given a state $s \in S$.
- $\pi(s, a)$ accepts state, s and returns the action to be taken (with the highest reward)
- $R(s, a)$

Once an episode is over, we reward all of those state-action pairs. In this case the episode ends when the goal is scored. Hence the agent is retroactively learning and acting off of knowledge of previous episodes. Once it completes a new episode, it adds the reward to its State-Action-Value Map ($savMap$) and updates the average of those state actions accordingly. Then, when that state comes around, we pick the best action based off of those updates values. chooseMove() returns an action, then we add this state-action pair to $sacList$.



A sample diagram of our game model on the grid.

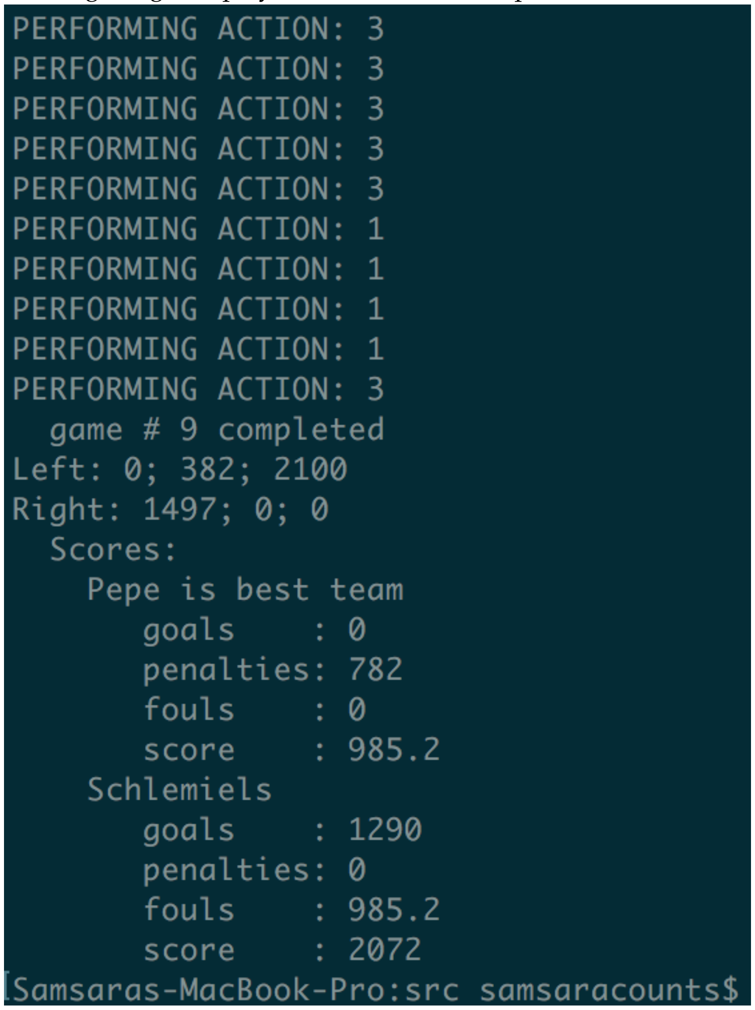
Table 1: *Sample Playoff results*

Team					
Number	Name	Goals	Penalties	Fouls	Score
0	Pepe	0	782	0	985.2
1	Schlemiels	1290	0	985.2	2072

e44e

III. Results

The above are some sample results after a given instance of our team, Pepe is the best team versus the Schlemiels team, featuring policies hand-coded by Dr. Simha. Here is our sample output after running a 9 game-playoff with our trained policies between the aforementioned teams.



```
PERFORMING ACTION: 3
PERFORMING ACTION: 3
PERFORMING ACTION: 3
PERFORMING ACTION: 3
PERFORMING ACTION: 3
PERFORMING ACTION: 1
PERFORMING ACTION: 1
PERFORMING ACTION: 1
PERFORMING ACTION: 1
PERFORMING ACTION: 3
  game # 9 completed
Left: 0; 382; 2100
Right: 1497; 0; 0
  Scores:
    Pepe is best team
       goals    : 0
       penalties: 782
       fouls    : 0
       score    : 985.2
    Schlemiels
       goals    : 1290
       penalties: 0
       fouls    : 985.2
       score    : 2072
[Samsaras-MacBook-Pro:src samsaracounts$
```

Though our team does not yet defeat the opponents with hand-coded policies, we believe there is full potential for our algorithm to outperform them with enough tweaking and refinement of the helper functions of specific actions.

References

[Brown, 2015] Brown, Brandon "Playing Blackjack with Monte Carlo Methods", `http://outlace.com/rlpart2.html`

[Kitano et. al., 1997] Kitano, H. et al. "RoboCup: A challenge Problem for AI" AI Magazine , v. 18, n. 1, p. 73-85, Spring 1997.

[Sanderson, 1997] Sanderson, A. "Micro-Robot World Cup Soccer Tournament (MiroSot)". IEEE Robotics and Automation Magazine, pg.15, December 1997.

[Stone et. al.] Stone, Peter and Sutton, Richard S. and Singh, Satinder P. "Reinforcement Learning for 3 vs. 2 Keepaway", RoboCup, vol. 2019, 2000, `http://dblp.uni-trier.de/db/conf/robocup/robocup2000.html#StoneSS00`, *Springer*

[Sutton and Barto, 1998] Sutton, R.S. and Barto, A.G. (1998). Reinforcement Learning: An Introduction, `https://books.google.com/books?id=CAFR6IBF4xYC`, *Bradford Book*