Watchdogs 2D Soccer Simulation

Team Description Paper 2014

Devanshu Jain
DA-IICT
Gandhinagar, Gujarat
201201133@daiict.ac.in

Miten Shah
DA-IICT
Gandhinagar, Gujarat
201201219@daiict.ac.in

Bharat Kumar Garg
DA-IICT
Gandhinagar, Gujarat
201201147@daiict.ac.in

ABSTRACT
The paper is an effort to come up with a strategy to control the attack behaviour of a team of simulated soccer playing robots. RoboCup 2D is a multi-agent adverserial game in a stochastic environment. We model the agent's decision making by q-learning and use attraction and repulsion factors, present in the field to control the movement of the players.

Categories and Subject Descriptors
I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems, Intelligent agents*

Keywords
artificial intelligence, robocup, q-learning, dynamic positioning

1. INTRODUCTION
RoboCup 2D is an international robotics competition aimed at providing a competitive platform for researchers worldwide to create an intelligence robot soccer team. It poses several challenges such as noisy environment, adversary in the form of an opposition team and stochastic action. Our job was to modify the agent2D, a HELIOS[1] based Robosoccer bot, using Artificial Intelligence techniques.

2. RELATED WORK
Many computer scientists have tried to model the problem through the means of Markov Decision Process(MDP). The major problem that arises in such a strategy is that its a kind of offline learning strategy. The factors, on the basis of which the decisions are made, are computed offline. No adjustments are made during the actual play. We too started out this project by modelling through Markov Decision Process [4]. But after much research decline of Kalsruhe Brainstormers [5] and discussion on Reinforcement Learning [2] vs. MDP, we decided to adopt, the learning counterpart.

In order to maintain formation of the players, we have come up with a heuristic based approach with attraction and repulsion vectors, which takes into consideration the spread of the players and prevent them from converging to the same point. This has slightly been inspired by the SPAR[6] technique used by CMUnited98, 1998 RoboCup world champions.

3. PROBLEM MODELLING
To apply the q-learning, we have to model the problem. We have to specify the actions and states that the player is allowed to take[3, 8].

3.1 States
First step towards modelling is the discretization of the states. Here we use the positions of the playing entities as description of the states. However, the number of such states become extremely large. To prevent this, we divide the soccer playing field into a grid. So, the player position is no more the x and y coordinate as specified by its world model, but it is the cell coordinates of the grid, in which it is positioned.
The states of the problem for a particular player, p_1 can be specified as follows:

1. P_{p_i} : Position of team player p_i
2. P_{o_i} : Position of opposition team player o_i, where $i = 1, 2..., 11$
3. P_b : Position of ball

3.2 Actions
Now, we specify the actions, available to the player as follows:

1. Pass(i): Pass the ball to i^{th} team player
2. Hold: Hold the ball
3. Dribble: Dribble the ball
4. Move: Move to a position which is favourable for passing and shooting
5. Intercept: Intercept the ball from the player
6. Goal: Kick the ball at the opposition goal

Amongst the actions, all except "Move" is straightforward to implement. To move the player so that he is in an optimised position to receive a pass and shoot the ball is absolutely critical to the team performance. So, for moving the player, we make use of attraction and repulsion factors as follows:

1. Team players repel the agent - This is important to prevent the players from coming together at a single position
2. Opposition players repel the agent
3. Ball attracts the agent
4. Goal attracts the agent

The algorithm for "moving" is as follows:

Algorithm 1 Move

```
1: procedure MOVE(Player p)
2:     count ← the number of player entities
3:     Vector v ← 0
4:     for player entity p_i around p in a radius of 10mt. do
5:         Vector u ← vector(p, p_i)
6:         v ← v − u/|u|^2
7:     v ← v/count
8:     Vector u ← vector(p, ball)
9:     v ← v + u/|u|^2
10:    Vector u ← vector(p, goal)
11:    v ← v + u/|u|^2
```

Here, we only take those player entities that are around the player, p within a radius of 10m. The count variable is important as we distribute the repulsion among all the players so that the high quantity of repulsive elements don't overwhelm the attractive elements. The vector(i, j) method returns a vector (with distance: dist and direction: dir) from i to j. Here, we make sure that repulsive entities that are far away don't affect as much as those entities that are near.

3.3 Rewards

Reward Dependencies

- State-Action Success
- State-Action Failure
- Scoreline
- Time

Mathematical Analysis

1. Goal:
   (a) Goal Scored: This is the most desired state-action. It is given a constant reward of $+1$ independent of scoreline or time.
   (b) Goal Awarded: This is the most undesirable state. It is similarly given a constant reward of -1 independent of scoreline or time.
2. Pass: This is the fundamental action in a game. The reward will depend upon the simplicity and effectiveness of the pass.

Situation of Game	Scoreline ≥ 0	Scoreline < 0
First Half	less	moderate
Second Half	very less	high

Reward on forward pass

$$\mathrm{F}_1(scoreline, time) = \begin{cases} \frac{Total\ time - time}{Total\ time} \times \frac{|scoreline|}{20}, & \text{if } scoreline < 0 \\ \frac{time}{Total\ time} \times \frac{1}{2} + \frac{|scoreline|}{20}, & \text{if scoreline} \geq 0 \end{cases}$$

(a) Successful Pass:

$$\mathrm{R(s,a,s')} = \mathrm{F}_1(sc, t) \times distanceFromOwnGoal(Player_2)$$

(b) Successful Through Pass:

$$\mathrm{R(s,a,s')} = \mathrm{F}_1(sc, t) \times distanceFromOwnGoal(Player_2)$$

3. Possession: This much very much dependent on time and scoreline.

Situation of Game	Scoreline ≥ 0	Scoreline < 0
First Half	very less	moderate
Second Half	less	high

Reward/Penalty on change of possession

$$\mathrm{F}_2(scoreline, time) = \begin{cases} \frac{time}{Total\ time} \times \frac{|scoreline|}{20}, & \text{if } scoreline < 0 \\ \frac{time}{Total\ time} \times \frac{1}{2} + \frac{|scoreline|}{20}, & \text{if scoreline} \geq 0 \end{cases}$$

(a) Interception/Tackle:

$$\mathrm{R(s,a,s')} = 1 \times F_2(sc, t) \times distanceFromOpponentGoal(ball)$$

(b) Ball Given Away:

$$\mathrm{R(s,a,s')} = -1 \times F_2(sc, t) \times distanceFromOpponentGoal(ball)$$

3.4 Algorithm

Now, the algorithm, which is used for policy building is described [7]:

Algorithm 2 Q-learning

```
1: procedure Q-LEARNING(Player p)
2:     i ← 1
3:     Initialise Q_i(S, A) to be 0
4:     i = i + 1
5:     while Game_is_on do
6:         S ← current state of player p
7:         random ← generate random no. between 0 and 1
8:         if random < 0.3 then
9:             A ← some random action
10:        else
11:            A ← get action according to the policy π
12:        Execute action A
13:        S' ← state of player p after executing action A
14:        r ← evaluate state S' for reward
15:        Q_i(S, A) ← (1 − α)Q_{i−1}(S, A) + α(r + max_a Q_{i−1}(S', a))
16:        i = i + 1
```

Here, we initialise Q values of every state-action pair to be zero. Then, during the game, these values are updated. First, we get the current state of the player p. Then, in order to keep balance between exploration and exploitation, we choose the cut-off factor as 0.3, i.e. 30% of the time, the agent will choose a random action and for the rest of time, he will exploit the already learnt policy from Q values. Then, the agent executes the action and gets reward for it. According to the reward, Q value of the previous state is updated.

4. ACTION TECHNIQUES

In this section, we introduce the algorithm for some of the primitive functionalities of the agent, such as passing, through passing and dribbling.

4.1 Passing

Currently we have implemented a simple passing implementation wherein we obtain the coordinates of the player to be passed to and then implement the kick() function in that direction with maximum power.

There are three main factors affecting the decision of pass to a teammate during normal play.

1. The distance between the teammates
2. The forwardness of the receiver with respect to other teammates
3. How clear the pass is; which depends on how many opponent players are marking the teammates and how many are able to intercept the pass

When a team member is in possession of the ball and there is no possibility to dribble or shoot, he has the option to pass. There are 10 teammates to which the player can pass to. To make the best decision, we iterate over the situation of passing the ball to each teammate. In each such iteration we check the forwardness of the teammate and how clear is it to pass it. The best receiver should have a balance between forwardness and clarity.

Algorithm 3 Passing Decision

procedure PASSTO
  for $i \leftarrow 1, 10$ **do**
    $d \leftarrow distance(self, P_i)$
    $f \leftarrow y_{coordinate}(P_i)$
    $disturb \leftarrow Marking(P_i) + Interceptable(self, P_i)$
    $result = f/disturb$

Algorithm 4 Marking

$threshold \leftarrow value$
procedure MARKING(Player)
  for $i \leftarrow 1, 11$ **do**
    $sum \leftarrow 0$
    if $distance(P, O_i) \leq threshold$ **then**
      $sum \leftarrow sum + 1$
  return sum

Algorithm 5 Interceptability

procedure INTERCEPTABLE(Player1, Player2)
  $diameter \leftarrow distance(Player1, Player2)$
  $radius \leftarrow diameter/2$
  $center \leftarrow (coor_{Player1} + coor_{Player2})/2$
  for $i \leftarrow 1, 11$ **do**
    $sum \leftarrow 0$
    if $distance(center, O_i) \leq radius$ **then**
      $sum \leftarrow sum + 1$
  return sum

4.2 Through Passing

We have implemented a basic through pass where in we kick the ball with maximum power in the direction horizontally ahead of the player to be passed in such a way that the team mate reaches to it first in his movement before any opponent.

There are three main factors affecting the decision to through pass during normal play.

1. No clear pass
2. A teammate running into an empty zone
3. No obstruction
4. Resulting pass should be most forward

Algorithm 6 Through Pass Decision

procedure THROUGHPASSTO
  for $i \leftarrow 1, 10$ **do**
    if $P_i.RUNNING?$ equals $True$ **then**
      $center \leftarrow coor_{P_i}$
      $radius \leftarrow distance(self, center)$
      $sum \leftarrow 0$
      for $j \leftarrow 1, 11$ **do**
        if $distance(O_j, center) \leq radius$ **then**
          $sum \leftarrow sum + 1$

4.3 Dribbling

If player is in the opponent half and the region ahead is empty then hit the ball farther and chase it. If the player is in a slightly vulnerable or congested zone, the player will hit the ball with low power and chase it ahead. Long dribbles will generally be used in the wide areas or flanks where we expect very little opposition.

Algorithm 7 Dribble Decision

$radus \leftarrow value$
procedure DRIBBLE
  for $i \leftarrow 0, 11$ **do**
    if $distance(self, O_i) \geq radius$ **then**
      $LONGDRIBBLE$
    else
      $SHORTDRIBBLE$

5. REFERENCES

[1] H. Akiyama and T. Nakashima. Helios base: An open source package for the robocup soccer 2d simulation. In *RoboCup 2013: Robot World Cup XVII*, pages 528–535. Springer, 2014.

[2] Y. Duan, Q. Liu, and X. Xu. Application of reinforcement learning in robot soccer. *Engineering Applications of Artificial Intelligence*, 20(7):936–950, 2007.

[3] J. Neri, M. Zatelli, C. Farias dos Santos, and J. Fabro. A proposal of qlearning to control the attack of a 2d robot soccer simulation team. In *Robotics Symposium and Latin American Robotics Symposium (SBR-LARS), 2012 Brazilian*, pages 174–178, Oct 2012.

[4] M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*, volume 414. John Wiley & Sons, 2009.

[5] M. Riedmiller, A. Merke, D. Meier, A. Hoffmann, A. Sinner, O. Thate, and R. Ehrmann. Karlsruhe brainstormers-a reinforcement learning approach to robotic soccer. In *RoboCup 2000: Robot Soccer World Cup IV*, pages 367–372. Springer, 2001.

[6] M. Veloso, M. Bowling, and P. Stone. The cmunited-98 champion small-robot team. *Advanced Robotics*, 13(8):753–766, 1998.

[7] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

[8] L. Xiong, C. Wei, G. Jing, Z. Zhenkun, and H. Zekai. A new passing strategy based on q-learning algorithm in robocup. In *Computer Science and Software Engineering, 2008 International Conference on*, volume 1, pages 524–527, Dec 2008.