RobôCIn Team Description Paper 2019

Cristiano Santos de Oliveira, Lucas Vinicius da Costa Santana, Mateus Gonçalces Machado, Marvson Allan Pontes de Assis, Walber de Macedo Rodrigues, Edna Natividade da Silva Barros, Tsang Ing Ren, and Paulo Salgado Gomes de Mattos Neto

robocin@cin.ufpe.br

February 17, 2019

Abstract. RobôCIn Soccer Simulation 2D team started in 2018 at the Universidade Federal de Pernambuco. Our first competition was at João Pessoa, Paraíba, Brazil in Latin American Robotics Competition (LARC) 2018 where we obtained the 4th place against teams from Latin America. In this paper we describe some of the approaches that we are currently developing for our second year of research in the category, focusing our studies on the use of Deep Reinforcement Learning and Statistical Methods for decision making in the agents' chain action.

1 Introduction

RobôCIn is a robotic research team from the Centro de Informática, Universidade Federal de Pernambuco, created in 2015 to participate in competitions and research subjects related to robotics.We are currently working in four categories: Very Small Size (VSS) since 2015, Soccer Simulation 2D (SS2D), @Home and Small Size categories since 2018. On our first time competing on SS2D, we obtained the 4th place on Latin American Robotics Competition (LARC), developing a strong and competitive agent in Brazil and Latin America. We based our code on agent2d 3.1.1 [1], since it is a well-structured base and from it, we could develop our studies and approaches more quickly.

2 Deep Reinforcement Learning Strategies to Intercept the Ball

We based our strategies on similar approaches of other teams. Similarly to FRA-UNIted2017 [2], we decided to use a Reinforcement Learning approach but using to improve the accuracy of the ball interception. CYRUS2018 [3] did a similar procedure predicting players positions and then intercepting it or marking. FRA-UNIted2018 [4] and CYRUS2018 both used Neural Networks approaches. We also used C++ Neural Networks frameworks and training results based on their decisions.

2.1 Prioritized Experience Replay

Prioritized Experience Replay (PER) [5] is a strategy which assumes that some transitions are more important than others in terms of learning. We cannot infer from the usual Memory that, for example, walking is more important than running when we are children. Our human memory works more likely as PER. The sampling of a PER memory is done with a biased (priority) probabilistic model. Doing so, our model prefer to learn from states that do not fit well the next values, because these states are the ones to learn the main objective.

2.2 Dueling Double Deep Q Networks

The main idea of Dueling Double Deep Q Networks (DDDQNs) [6] is to decompose the Deep Q Network's Q-value (DQN)[7] to "how good is to be on that state" and "how good is to take that action at that state", resulting on Eq. 1. V(s) represents the value of the state and A(s, a) is the value of the action at that state.

Decomposing the predicts, the network can learn which states are valuable or not without predicting the impact of an action on each state. However, if only considering Eq. 1, the model will fall into the issue of identifiability, that is, given Q(s,a) it cannot predict A(s,a) and V(s). Ziyu Wang et al. proposed to subtract the advantage of action with the average advantage of all possible actions given a state, resulting on Eq. 2, where θ is the common network parameters, α is the advantage stream parameters and β is the value stream parameters.

$$Q(s,a) = A(s,a) + V(s) \tag{1}$$

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + (A(s,a;\theta,\alpha) - A^{-1} * \sum_{a'} A(s,a';\theta,\alpha)) \tag{2}$$

3 Ball Possession Algorithm (BPA)

We developed a ball possession algorithm that acts among the agents in order to maintain possession of the ball with our team. Following the teams [8], [9], and [10], which changed the way how the agents evaluate the decisions to score a goal, we change the evaluation to increase our ball possession. To achieve this, we evaluated the chain of actions in order to select the actions that have a lower probability of losing the possession of the ball. To create such algorithm, it is necessary to have a dataset with information about the player's surroundings, the actions that the player performed with the ball and the outcome of these actions.

3.1 Generating the Data Base

The approach used was modifying the agent code to extract the surroundings, the actions followed and the outcome. This is done by adding code to our Standard agent and playing three times with other teams. We selected the

first and second places from Robocup of 2018, 2017 and the agent2d itself. An action has 2 possible outcomes, we still have the ball, counting 70 points, or we do not have the ball, counting -100, evaluating after ten cycles if the player does not have the ball. The player surroundings have information about how many opponents are near, how many teammates and which area the player is in, following the division described in [11].

3.2 Statistical Method

To increases the ball possession, we developed an algorithm that statistically finds the best action. Composed of a set of Path Trees, represented in Figure 1, each path in the tree have an expected outcome, statistically learned from the dataset.

The algorithm subdivides the dataset information into clusters using k-means and the elbow method to estimate the number of clusters. We selected 13 clusters to divide the dataset. Each cluster has one specific Path Tree. One Path Tree has a set of actions, represented by Action A and B if followed results in a different state with another set of actions. The player generates candidate paths, and the algorithm returns the expected outcome value to the paths, so the player selects the path with the best outcome.

However, if the agent is in an entirely new situation, there is no information about how it must proceed. This situation is shown in Figure 1 as "X" marks under some actions, showing that the tree does not have information in that direction. We handled this issue by using the basic agent2d field evaluator to find the action, which is not optimal. Another issue is the lack of information about some states, leading to a biased and possibly wrongly decision. To manage this issue, we used supervised learning to classify if an action, in certain conditions, can lead to the loss of ball possession or not.

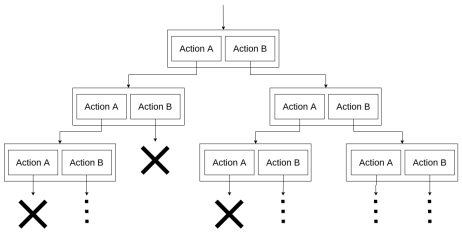


Fig. 1: Path Tree for ball possession algorithm. X indicates no further path when following this action. Each action in the tree have an expected outcome, this helps the agent to keep the ball possession.

4 RoboCup Soccer Simulation Server

Robocup Soccer Simulation Server (RCSSS) is the core of SS2D which process the environment, making all the necessary modifications according to the game

dynamic in real time. Also, it establishes the communication between participating teams, that can receive the environmental information and perform the agent's actions, and the communication with monitors, from which one can watch the game. RCSSS follows a client-server style, which allows teams to develop on any platform, as long as it can perform UDP/IP communication. Each team has the right to build 12 clients that exchange messages with the server, having 11 players (10 fielders + 1 goalie) and the coach, where each runs a separate process and communicate with the server through a specific port.

After initiating the connection with the clients, the server sends the initial parameters of the game. They are the server-params and the players-params (this one is to the teams selected from the random heterogeneous players). Each client performs a function and has related constraints, like the fielders that receive local information captured by the aural sensor and vision sensor and send requisitions of the actions they want to take. The goalie that has actions only he can take (e.g., the catch function), the coach who is a privileged client that can be used to make strategies communicating with the players and game analysis.

An important feature of the server is the discrete division of time (cycle) because the action of each client is executed per cycle that has a specific time duration. The developer should consider this since it can lose cycles if it does not perform well. For the developer there are some server operating parameters that can be used to help during development, e.g., running with server::synch_mode=on speeds up the execution and synchronize with others server parameters of the game which combined with the server::auto_mode=on that executes the kickOff automatically, it could be written a script that executes several games automatically.

5 Experiments and Results

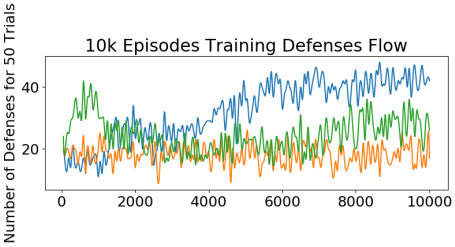
5.1 Deep Reinforcement Learning Strategies to Intercept the Ball

We used the Half Field Offensive (HFO) Environment [12] to test a defensive agent. In this experiment we coded an agent, an Non-Player Character (NPC) goalie using agent2d and an offensive NPC built with HELIOS 2013 [13]. We used the High Level Features set given by the HFO environment to produce our states and actions. As actions, we defined only two: Move (the agent goes to the point given by formation.conf) and Go_To_Ball (the agent intercepts the ball and tries a Tackle).

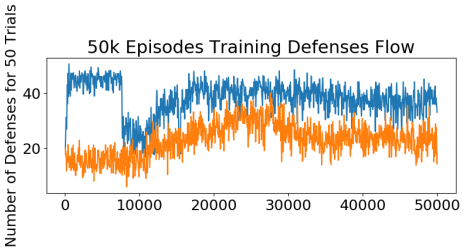
For the agent itself, we modified a code example of [12] to train three neural networks: DQN, DQN with PER and DDDQNs with PER. We used Stacked States only on modules with PER.

When testing, DDQNs obtained 89% of Succeed Defenses while DQN and DQN-PEN obtained 43% and 51% respectively for 100K memory. The larger the memory capacity, more episodes is necessary to converge due to the loss of the initial generated memory, which is the reward itself. So, using a 1Mi Memory we got a DDQNs model with 71% Succeed Defenses. With time, the model that uses 1Mi of memory should obtain a better intercept accuracy than 100K memory model. Compare the training flows, Fig. 2a and Fig. 2b.

Also, we noticed that agent's positioning is very important for the model to converge faster. We trained with the agent2d basic formation, and it was not converging due to the distance to the ball. Then we made a more aggressive formation that fitted better so that the agent tended to push the offensive NPC to the sides of the field, once it is far from the goal.



(a) DDDQNs-PER, DQN-PER, DQN are respectively blue, green and orange curves.



(b) DDDQNs-PER, DQN-PER are respectively blue and orange curves.

Fig. 2: Training flow for 10K and 50K Episodes models for 100K and 1Mi sized memory respectively

5.2 Ball Possession Algorithm (BPA)

The BPA test setup is simple, and we compared our code used at LARC 2018 (Standard) with and without BPA. The two teams played ten times against the latest agent2d version, 3.1.1. It is important to notice that our evaluation counts only the ball possession ratio, discarding goals, the number of passes and stamina used.

Team	1	2	3	4	5	6	7	8	9	10	Mean (std)
BPA	**67.03**	**59.19**	**56.14**	**61.84**	**46.43**	**86.82**	**79.59**	**92.52**	**86.05**	**87.66**	**72.32 (15.28)**
Standard	58.78	49.10	51.10	49.37	45.60	41.79	58.45	52.76	45.44	51.64	50.40 (5.16)

Table 1: BPA versus Standard team, compared each possession at each game, followed by mean (standard deviation). These distributions have $p = 0.0051$ in Wilcoxon test. In bold, the best result of each.

Table 1 shows the ball possession for each game and the final mean and standard deviation of our tests. Ball possession extracted using [14], then we calculated the mean, standard deviation and used the Wilcoxon test to evaluate the distribution. Using our Standard code against the agent2d, with BPA we increased from 50.40% and 5.16 standard deviation to 72.32% of possession, with 15.28 of standard deviation. The p value of Wilcoxon test gives us 0.0051, confirming that there is a different distribution, considering α of 0.1 and 0.05.

6 Conclusions and Future Work

This paper describes some of the approaches developed by our team in our first year in the SS2D category, focusing the efforts to build ball interception strategies using deep reinforcement learning. We obtained relevant preliminary results, and in decision making for greater ball possession using what we call BPA. In addition, we developed frameworks of data extraction, but we want to build a software that is more comprehensive in this area and give more flexibility to those who need to mine the logs generated by the server, we expected to finish it prior to the competition. Also for the future work, we will try to improve these approaches using deep learning and increase agent action by using supervised learning to choose the best action on the BPA and making predictions of opponent's moves.

References

1. Robocup tools agent2d. `https://pt.osdn.net/projects/rctools/`. Accessed: 2019-01-01.
2. Thomas Gabel, Steffen Breuer, Constantin Roser, Roman Berneburg, and Eicke Godehardt. Fra-united — team description 2017. 2017.
3. Nader Zare, Mohsen Sadeghipour, Ashkan Keshavarzi, Mahtab Sarvmeili, Mohammad Abolnejad, Arad Firouzkoohi, Reza Aghayari, Amin Nikanjam, Amin Akhgari, and Sina Elahimanesh. Cyrus 2d simulation team description paper 2018. 2018.
4. Thomas Gabel, Philipp Kloppner, and Eicke Godehardt. Fra-united — team description 2018. 2017.
5. Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *CoRR*, abs/1511.05952, 2015.
6. Ziyu Wang, Nando de Freitas, and Marc Lanctot. Dueling network architectures for deep reinforcement learning. *CoRR*, abs/1511.06581, 2015.
7. Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.
8. Tomoharu Nakashima, Hidehisa Akiyama, Yudai Suzuki, An Ohori, and Takuya Fukushima. Helios2017: Team description paper. 2017.
9. Tomoharu Nakashima, Hidehisa Akiyama, Yudai Suzuki, An Ohori, and Takuya Fukushima. Helios2018: Team description paper. In *RoboCup 2018 Symposium and Competitions: Team Description Papers, Montreal, Canada*, 2018.
10. Masataka Mizumoto, Tsubasa Fuzimitsu, Takashi Ebara, Seiya Yamamoto, Hiroto Asai, Akira Ishida, Shosaku Inoue, Hiroaki Oe, Yudai Kawakami, Taku Kitamura, et al. Robocup 2017-2d soccer simulation league team.
11. Herinson B. Rodrigues. robocup2d-tutorial. `https://github.com/herodrigues/robocup2d-tutorial/blob/master/sections/Strategy.md`, 2016.
12. Half field offense. `http://www.cs.utexas.edu/~AustinVilla/sim/halffieldoffense/`. Accessed: 2019-01-01.
13. Helios 2013. `https://archive.robocup.info/Soccer/Simulation/2D/binaries/RoboCup/2013/HELIOS_SS2D_RC2013_BIN.tar.gz`. Accessed: 2019-01-01.
14. KN2C Robotics Lab. Rcssanalyzer. `https://github.com/dark-0ne/RcssAnalyzer`, 2018.