A Data-Driven Simulator for Assessing Decision-Making in Soccer

Tiago Mendes-Neves[1,2](✉), João Mendes-Moreira[1,2], and Rosaldo J. F. Rossetti[1,3]

[1] Faculdade de Engenharia, Universidade do Porto, Porto, Portugal
{up201406104,jmoreira,rossetti}@fe.up.pt
[2] LIAAD - INESC TEC, Porto, Portugal
[3] LIACC - Laboratório de Inteligência Artificial e Ciência de Computadores, FEUP, Porto, Portugal

Abstract. Decision-making is one of the crucial factors in soccer (association football). The current focus is on analyzing data sets rather than posing "what if" questions about the game. We propose simulation-based methods that allow us to answer these questions. To avoid simulating complex human physics and ball interactions, we use data to build machine learning models that form the basis of an event-based soccer simulator. This simulator is compatible with the OpenAI GYM API. We introduce tools that allow us to explore and gather insights about soccer, like (1) calculating the risk/reward ratios for sequences of actions, (2) manually defining playing criteria, and (3) discovering strategies through Reinforcement Learning.

Keywords: Soccer simulation · Simulation · Decision-making · Reinforcement learning

1 Introduction

There are multiple approaches to simulate the game of soccer. However, it is not easy to obtain insights into the actual game from these simulators. Here we propose an alternative approach: use event stream data [4] and deep learning to build the simulator. Using data to build the simulator allows us to obtain a different representation of the game that abstracts from simulating complex physics and other details that are not relevant to analyze the game at a high-level. Having a high-level simulator can provide interesting insights to the users, like soccer analysts.

Data-driven soccer simulators have advantages over current state-of-the-art simulators. The most relevant advantage is that the high level of knowledge required to build a simulator is replaceable with data models. This advantage

This work is financed by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, within project UIDB/50014/2020.

shifts the requirement of sufficient knowledge about the system to requiring sufficient data points to build the simulator accurately.

Another advantage of this approach is that it allows us to change the testing environment by altering the data used to build the simulator's model. For example, we can build a simulator that tries to replicate a team's behavior by training the models with data of a specific team. Furthermore, having a simulator allows using Reinforcement Learning (RL) algorithms to discover new patterns in the game.

Soccer is a challenging game to model because human-like physics is hard to simulate. We do not address this problem in this paper. Instead, we abstract from this problem by using high-level event data from soccer games.

Currently, the literature does not address the problem of building a soccer simulator from existing data. Physics-based simulators dominate current approaches, making it challenging to research new strategies or ask "what if" questions. Furthermore, the environments have low flexibility for adaptation, i.e., it is hard and expensive to model a team's behavior in these simulators.

This paper describes an approach to fill this gap by creating a simulator built from data. We describe a new simulator that can simulate the soccer game from a high-level perspective, oriented towards actions. We developed the simulator following the OpenAI GYM API, which facilitates experiments with compatible libraries for RL.

The simulator requires an agent to decide the best action to make according to the current state and then gives feedback in the form of a reward. We present three use cases for the simulator that offer evidence of the proposed framework's utility: comparing playing sequences, comparing playing criterion, and discovering policies using RL.

The main contributions of this work are the following: In Sect. 2 we review the related work, focusing on soccer simulators, use cases for soccer data, and RL. In Sect. 3 we describe the main contribution of the paper, the soccer simulator, presenting the flow diagram, data used, and the resulting models used to produce the simulations. In Sect. 4 we present and discuss the results of three proposed use cases for the simulator. Finally, in Sect. 5 we present our conclusions and future direction.

2 Related Work

Solutions for simulating soccer games have been available since *NASL Soccer* was launched in 1979 by *Intellivision*. In fact, soccer is widely popular as a video game. Recently, this genre is represented by the *FIFA*[1], *PES*[2] and *Football Manager*[3] franchises. Video games have become very realistic from the graphics point of view. However, this has come at the price of complexity. Moreover, behavioral realism is not adequate since entertainment is the main purpose of

[1] ea.com/games/fifa.
[2] konami.com/wepes.
[3] footballmanager.com.

these games, with business decisions affecting how the game is designed. This lack of focus on realism leads to problems in terms of reliability.

There are alternative solutions to video games. Robotic soccer simulation has a broad number of solutions available. RoboCup [16] has competitive leagues focused on simulation in both 2D and 3D, and there is a large community developing strategies for these simulators [1]. Still, robotic soccer does not represent the actual soccer game. Even with the field rapidly advancing, we are still decades away from robotic soccer replicating human soccer to a high-fidelity level.

Google Research Football [11] currently represents the state-of-the-art of soccer simulation. The authors present a simulator based on a physics engine used in competitions sponsored by soccer teams [9]. Due to a focus on RL tasks, this simulator represents an improvement over video game simulators in computational performance. The same problem of robotic soccer simulators is still present. This simulator cannot correctly model human body behavior and interaction with the ball, introducing errors in the simulation.

We believe that our proposal will be able to introduce novel ways to analyze the game of soccer. Currently, soccer data analytics has been experiencing a very steep growth. Data tracking systems such as GPS, optical tracking, and event annotation are being applied at the club level, increasing the quantity and quality of data available in the field.

Current use cases for soccer data focus on evaluating player performance. For example, Plus-Minus rating [10] that represent the weighted sum of the contributions of playings to goals scored/conceded in a game, and VAEP [4] that evaluates player actions by calculating the probability of an action leading to a goal in the short term.

From tactical analysis [26] to injury forecast [19], there are many use cases for soccer data. Some of these works are made in partnership with soccer teams, proving that this is a research field with high applicability.

In one of the use cases presented for our simulator, we explore RL algorithms to find policies to help decision-making in soccer. RL aims to train an agent in an environment to maximize a reward function, allowing for a model that can learn from experience without previous information about the goal or the environment.

RL is central to some of the most celebrated recent successes in machine learning. Examples of this success are the high-human ability achieved in computer games such as StarCraft II [24,25], Dota 2 [3] where they have beaten current world champions, and the Atari game library [13,20]. Outside of computer games, we have Chess, Shogi [13,22] and Go [23], with wins over world champions, and Poker [8].

One of the most common comments from the players that faced these agents is that they are innovative. These agents can learn and develop unique and previously unseen strategies to play the games [17]. Novel strategies are valuable in many fields, including soccer. If correctly implemented, these strategies can give an edge to the team, becoming an advantage on the pitch.

In this work, we explore several RL algorithms: Advantage Actor-Critic (A2C) [15], Deep Deterministic Policy Gradient (DDPG) [12], Proximal Policy

Optimization (PPO) [21], Soft Actor-Critic (SAC) [7], and Twin Delayed Deep Deterministic Policy Gradient (TD3) [6]. The main reason for testing these algorithms was that they could handle continuous state and action spaces, which is a requirement of our simulator.

The DDPG algorithm produced the best results. DDPG is built on the foundational work of previous successes in the RL world by DeepMind. This algorithm aims to provide Deep Q Networks' [14] learning environment and extend it to continuous observation and action spaces using a target policy network.

Deep Q Networks aims to approximate the action-value table, known as Q values, using a non-linear approximation. The preferred approximation technique is Deep Learning. It would be possible to use Deep Q Networks in our approach if we discretized the observation and action space. However, due to the curse of dimensionality, this approach would be time-consuming. Using an Actor-Critic approach, DDPG avoids the dimensionality problem.

3 Simulator

The simulator is available in *Python*. All the code necessary to reproduce this work is publicly available[4], alongside the code used in the use cases presented in Sect. 4.

Figure 1 shows the simulator's flow diagram. The simulator has a state that indicates the ball's current coordinates *(X, Y)*. The player chooses an action with a probability of success that depends on the current state and player intention. If the action is successful, the simulator gets into a new state. If it fails, the sequence ends. The shot action always ends the sequence and returns a reward equivalent to the probability of scoring from the current state.

Using the probability of scoring might not be the best approach to find the required solutions. For example, there are situations where it is important to minimize the chances of conceding a goal. While this work does not address it, the framework allows the user to set up their reward function as he wants.

3.1 Data

We used event data from the 2016/17 season to the 2018/19 season from the Portuguese First Division. Event data is composed of the time and location of predefined events, like passes and shots, and describes an action in detail [4]. The dataset contains 15 719 shots and 681 989 passes.

Decroos [4] defined the SPADL language for describing player actions. The goal was to standardize soccer event data from multiple providers. We used this framework to prepare our data for the simulator. Our feature set contains the following features: *type of event*, *player*, *team*, *success*, *(x, y, endX, endY)* coordinates, and *isGoal*. The only difference compared to the SPADL language is that the *BodyPart* attribute was not available on our data set.

[4] github.com/nvsclub/SoccerActionsSim.

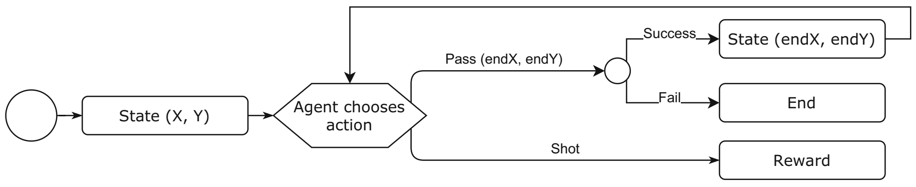

Fig. 1. Diagram showing the flow of the simulator. From a state, we can perform an action that will change our current state. In case the agent chooses the shot action, the sequence ends, and the agent receives a reward equivalent to the probability of scoring from the current state. If the sequence ends with a failed pass, the agent receives no reward.

We were unable to address a limitation in data. For passes, we need to assume that end coordinates represent the player's intention. This assumption is reasonable for successful passes, but for blocked passes, the *(endX, endY)* coordinates will represent the position where the pass was blocked, rather than player intention. This factor will introduce some errors in our simulations.

3.2 Models

To build the models, we used the MLPClassifier, from the scikit-learn Python package. We required fast prediction times, and Deep Learning algorithms allowed predictions to be quick without losing quality in the model. To define the hyperparameters, we used Optuna TPE optimizer [2]. Optuna helps us find which hyperparameter combination minimized the error of our models.

The goal of the models is to learn a representation of the underlying data. Therefore, after finding the optimal parameters on Optuna using a 50–50% train-test dataset, we use the whole dataset in the training procedure.

To adapt the environment for simulating a specific team, the user needs to change the data used to train the models, increasing the team's patterns in data. We do not want the environment to have models that generalize to all teams. Therefore we incentive a certain level of memorization on the models, i.e., models with larger architectures than what is required.

Shots. The concept of expected goals [18] inspires the model that simulates shots. The success of a shot is dependent on the state of the agent *(x, y)*. This approach is used to build other analytic frameworks in soccer such as Decroos [4], and Fernández [5]. To train the network, we used the *isGoal* feature as the target variable. *isGoal* is a boolean variable that is true when the shot leads to a goal. Figure 2 (on the left) shows a visualization of the probability matrix. The model converged to log-loss of 0.27.

Passes. To simulate passes, we followed a similar approach. The probability of success is dependent on current coordinates *(x, y)* and target coordinates *(endX,* *endY)*. In the training process, the target feature was the *success* variable, a boolean that is true when the action was successful.

When controlling the simulator, the agent has to decide about the (*endX, endY*) variables. These coordinates indicate to the simulator where the agent intends to pass the ball. Figure 2 (on the right) shows a visualization of the probability matrix. The model converged to log-loss of 0.34.

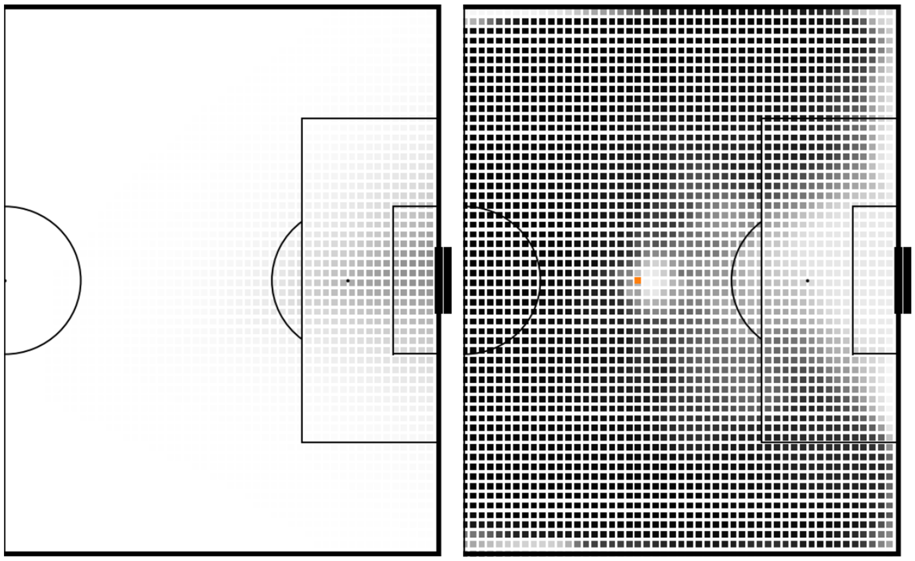

Fig. 2. On the left: a probability matrix given by the shot model for every position in the opposition's half. Darker spots indicate higher probabilities of scoring. On the right: a probability matrix given by the passing model. Predicts the success of passes with origin in the orange dot. (Color figure online)

4 Results

To demonstrate the capabilities of our simulator, we designed three use cases. First, we will present the methodology to calculate and visualize the likelihood of scoring from a specific play sequence. In this use case, we show that it is possible to compare different sequences of play among themselves to assess the efficacy of that sequence of actions.

The second use case illustrates the simulator's usage alongside action maps to assess the best actions in certain parts of the field, referred to as the playing criterion. For this, we divide the soccer field into 38 parts: first, we divide the defensive half into 12 equal-sized squares, and then we divide the rest of the pitch into areas that become smaller when closer to the opponent's goal to capture more detail in the most relevant areas. For each of these parts, we set an action. Then, we systematically evaluate this playing criterion and return information to the user. The information returned allows the user to improve the action map iteratively.

Finally, we take advantage of implementing the simulator according to the OpenAI GYM API to run RL algorithms in our simulator. We will run a set of experiments using the Stable-Baselines3 library, using the algorithms that can handle continuous state and action spaces, such as A2C, DDPG, PPO, SAC, and TD3. The goal is to find an automated way of building the playing criterion on the simulator.

4.1 Simulating Sequences of Play

The first use case that we present is the simulation of sequences of play. With this approach, we can simulate and compare different sequences of play and their outcomes. A sequence of play is a set of actions executed sequentially. By changing this set of actions, we will obtain different sequences.

Figure 3 shows an example of how to compare sequences. We calculate the outcome of three sequences. We will consider the first sequence as the baseline sequence. Then we produce two other suggestions of actions.

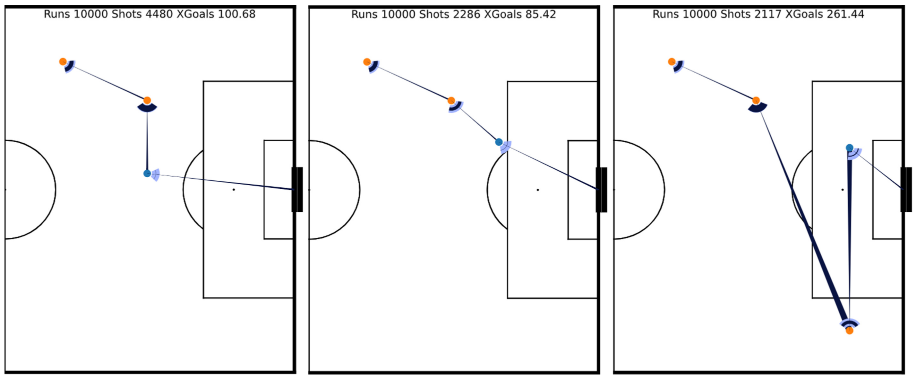
Runs 10000 Shots 4480 XGoals 100.68
Runs 10000 Shots 2286 XGoals 85.42
Runs 10000 Shots 2117 XGoals 261.44

Fig. 3. The outcome of three play sequences. Comparing the first sequence with the others, we can find that the third option was the one that leads to more goals overall. Note that we obtained these results with the simulator trained with data from all teams. For specific teams, the results seen in this figure can be substantially different.

The first suggestion is to move the last pass closer to the goal, increasing the probability of scoring from the shot. However, this proposal does not increase the sequence's probability of scoring since the pass became riskier than the previous. We were able to find that an action that looked better did not result in a better outcome since it reduced the number of expected goals in 10 000 iterations from 100 to 85.

A second proposal is to move the ball wide for a cross. This approach has a better risk-reward ratio than the original sequence, scoring 261 goals in 10 000 iterations, more than doubling the baseline. With this information, we can conclude that this was the best sequence to use.

This framework allows us to judge whether an action that happened was the best option available. The reasoning behind this example that we presented applies to many other scenarios.

4.2 Building Playing Criterion

Figure 4 presents a framework that allows the user to define a playing criterion and obtain numerical feedback for the decisions made. For each marked area, the user chooses an action and, in the case of a pass, the length and direction.

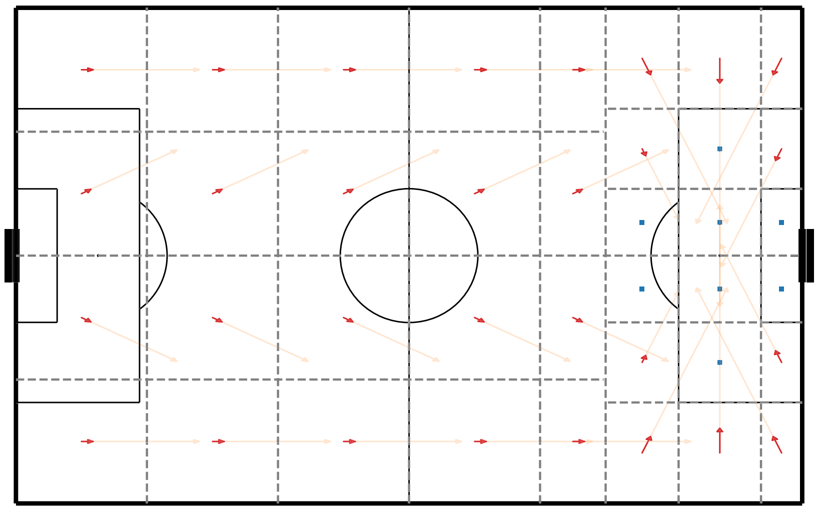

Fig. 4. Manually defined playing criterion. The orange arrows indicate that the action chosen in that area is the passing action, with the direction of the arrow corresponding to the pass's direction. Red arrows are 1/10 of the length of the actual pass length, represented in orange. Blue squares represent areas where the chosen option is to have a shot at the goal. This criterion achieves a reward of 0.0149 (for comparison: the average of 20 randomized criteria is 0.0084). (Color figure online)

This framework can quickly iterate through different criteria and test which types of actions improve or degrade the scoring probability. For example, if we replace a shot with a pass, what is the gain obtained in 10 000 simulations?

This framework allows the user to implement optimization algorithms, e.g., Genetic Algorithms, to optimize the actions. We hope to address this in future work.

4.3 Reinforcement Learning

Our approach to automatically optimize the actions taken in the simulator was to use RL. For this, we used the Stable-Baselines3 Python package. Since we used the OpenAI GYM API, it is easy to use RL libraries in our simulator.

We tested the algorithms compatible with continuous state and action spaces, namely A2C, DDPG, PPO, SAC, and TD3. We present the results of the tests for all algorithms using the default parameters from the library. For the DDPG and TD3 algorithms, we tested two types of action noise (Ornstein Uhlenbeck and Normal noise). Furthermore, since the DDPG presented a higher ceiling in preliminary tests, we also present the larger network results for both the critic and actor networks. Table 1 shows the results obtained.

Table 1. Results from the RL algorithms on the developed simulator. OU - Ornstein Uhlenbeck.

Algorithm	Parameters	Avg score	Max score
A2C	Default	0.0158	0.0169
DDPG	Default + OU noise ($\sigma = 0.3$)	0.0160	0.0272
	HiddenLayers = [400, 300] + OU noise ($\sigma = 0.3$)	0.0124	0.0318
	Default + Normal noise ($\sigma = 0.3$)	0.0108	0.0158
PPO	Default	0.0134	0.0140
SAC	Default	0.0068	0.0079
TD3	Default + OU noise ($\sigma = 0.3$)	0.0115	0.0164
	Default + Normal noise ($\sigma = 0.3$)	0.0126	0.0167

The different action noise types did not show any relevant results. Therefore we opted to maintain the Ornstein Uhlenbeck noise recommended by the authors [12]. In this problem, we are interested in obtaining the highest score possible. Although inconsistent, the DDPG algorithm presents the highest ceiling in terms of performance. Furthermore, the other algorithms have difficulties in learning a better policy than simply shooting in every position. Figure 5 presents the results obtained by running the DDPG algorithm with increased network size (400, 300) and a larger amount of timesteps (100k). The score obtained was 0.0318.

The DDPG algorithm found an interesting policy: it moves the ball towards the right side from the attacking point of view and then crosses it towards the center. In Fig. 5 we can observe a big orange cluster in the zone of the field where the agent crosses from, indicating that RL was able to find an efficient strategy in this particular environment setup.

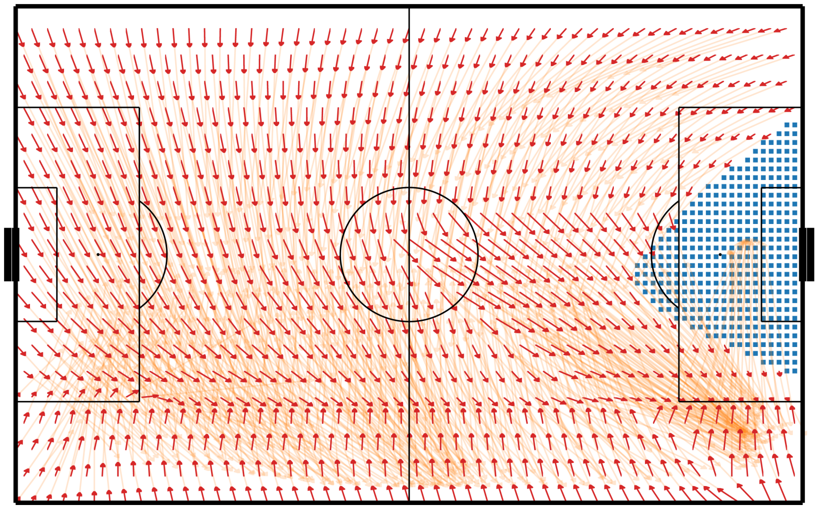

Fig. 5. Visualization of the decision making of the DDPG algorithm. The red arrows indicates that the action chosen specific coordinate is the passing action, with the direction indicating where the pass is directed to. Red arrows are 1/10 of the length of the actual pass length, represented in orange. Blue squares represent areas where the best option is to have a shot at goal. (Color figure online)

5 Conclusion

We built a soccer simulator that models the game from a different perspective. This event-driven approach enabled multiple use cases that we presented in this paper. The work presented in this paper indicates that data-driven soccer simulators can enable new ways of analyzing the game.

The use cases presented demonstrate that we can gather insights from the simulator. In the first use case, we can evaluate how a change an action affects the scenario's outcome, enabling us to improve decision making. The second use case shows how we can use the simulator to test high-level strategies. The last use case automated the discovery of these high-level strategies using RL. The strategy found by the DDPG algorithm provided a clear insight: the agent tried to explore the strategy of directing the ball for a cross from the right side.

We believe that the potential benefits from using this simulator are very relevant for improving decision-making in soccer.

5.1 Future Work

There are two main focuses for further developing this simulator. The first is to use transfer learning or data sampling techniques to adapt the simulator to specific teams' playstyles. This feature will allow us to find insights targeted towards a specific team rather than general policies. The second focus will be to expand the accuracy of the models used to build the simulator by adding positional data. We plan to launch new versions of this simulator in the future.

References

1. Abreu, M., Rossetti, R.J.F., Reis, L.P.: XSS: a soccer server extension for automated learning of high-level robotic soccer strategies. In: 2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC) (2019). https://doi.org/10.1109/ICARSC.2019.8733635
2. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: a next-generation hyperparameter optimization framework. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. Association for Computing Machinery, New York (2019). https://doi.org/10.1145/3292500.3330701
3. Berner, C., et al.: Dota 2 with large scale deep reinforcement learning. In: arXiv (2019)
4. Decroos, T., Bransen, L., Van Haaren, J., Davis, J.: Actions speak louder than goals: valuing player actions in soccer. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. Association for Computing Machinery, New York (2019). https://doi.org/10.1145/3292500.3330758
5. Fernández, J., Bornn, L., Cervone, D.: A framework for the fine-grained evaluation of the instantaneous expected value of soccer possessions. Machine Learning (2021)
6. Fujimoto, S., van Hoof, H., Meger, D.: Addressing function approximation error in actor-critic methods. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, 10–15 Jul 2018, vol. 80, pp. 1587–1596. PMLR (2018)
7. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International Conference on Machine Learning (ICML) (2018)
8. Heinrich, J., Silver, D.: Deep reinforcement learning from self-play in imperfect-information games. In: arXiv (2016)
9. Kaggle: Google research football with Manchester City f.c. https://www.kaggle.com/c/google-football/overview
10. Kharrat, T., McHale, I.G., Peña, J.L.: Plus–minus player ratings for soccer. Eur. J. Oper. Res. **283**(2), 726–736 (2020). https://doi.org/10.1016/j.ejor.2019.11.026
11. Kurach, K., et al.: Google research football: a novel reinforcement learning environment (2020)
12. Lillicrap, T.P., et al.: Continuous control with deep reinforcement learning. arXiv (2019)
13. Mnih, V., Kavukcuoglu, K., Silver, D.: Human-level control through deep reinforcement learning. Nature (2015). https://doi.org/10.1038/nature14236
14. Mnih, V., et al.: Playing atari with deep reinforcement learning. arXiv (2013)
15. Mnih, V., et al.: Asynchronous methods for deep reinforcement learning. In: Balcan, M.F., Weinberger, K.Q. (eds.) Proceedings of the 33rd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 48, pp. 1928–1937. PMLR, New York (2016). http://proceedings.mlr.press/v48/mniha16.html
16. Noda, I., Suzuki, S., Matsubara, H., Asada, M., Kitano, H.: Robocup-97: the first robot world cup soccer games and conferences. AI Mag. **19**(3), 49 (1998)

17. OpenAI: Openai five. openai.com/projects/five/ Accessed 6 Jan 2021
18. Pollard, R., Ensum, J., Taylor, S.: Estimating the probability of a shot resulting in a goal: the effects of distance, angle and space. Int. J. Soccer Sci. **2**, 50–55 (2004)
19. Rossi, A., Pappalardo, L., Cintia, P., Iaia, F.M., Fernàndez, J., Medina, D.: Effective injury forecasting in soccer with GPS training data and machine learning. PLOS ONE **13**(7), 1–15 (2018). https://doi.org/10.1371/journal.pone.0201264
20. Schrittwieser, J., et al.: Mastering atari, go, chess and shogi by planning with a learned model. In: arXiv (2016)
21. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. CoRR http://arxiv.org/abs/1707.06347 (2017)
22. Silver, D., Hubert, T., Schrittwieser, J.: Mastering chess and shogi by self-play with a general reinforcement learning algorithm. In: Nature (2017)
23. Silver, D., Schrittwieser, J., Simonyan, K.: Mastering the game of go without human knowledge. Nature (2017). https://doi.org/10.1038/nature24270
24. Vinyals, O., et al.: Starcraft ii: A new challenge for reinforcement learning. In: arXiv (2017)
25. Vinyals, O., Babuschkin, I., Czarnecki, W.: Grandmaster level in starcraft ii using multi-agent reinforcement learning. Nature (2019). https://doi.org/10.1038/s41586-019-1724-z
26. Warnakulasuriya, T., Wei, X., Fookes, C., Sridharan, S., Lucey, P.: Discovering methods of scoring in soccer using tracking data. KDD (2015). https://doi.org/10.1038/s41586-019-1724-z