Optimizing Goalkeeper Positioning

Mattia Danese and Evan Loconto

1 Abstract

Soccer, also known as football, is the most watched and followed sport in the world. The World Cup draws in billions of viewers from around the world, with an estimated 1.5 billion people tuning in for the 2022 World Cup Final. This popularity gave motivation for pursuing soccer as a topic for modeling in Reinforcement Learning. Our project focused primarily on goalkeeping positioning; our objective was to determine the best location for a goalkeeper given an attacking position somewhere on the field. The reinforcement learning method used for modeling was a Multi-Armed Bandit agent, where on every episode, the agent observed where the attacker was and then chose an action. Our hypothesis followed that the goalkeeper would maximize goal coverage by placing itself between the goal and attacker as much as possible to cut off the angle.

2 Introduction

In the world of professional sports, nothing is more important than beating your opponent. Whether one plays on a team or individually, winning is the be-all end-all. There are many sports played professionally, however no sport is as scrutinized as the game of soccer.

Soccer, also known as football, is a sport played and watched by billions across the world. Soccer is a game involving two 11-player teams, two nets, and a soccer ball. Each team tries to score into their opponent's net more times than the other team. The one caveat however is players cannot use their hands to achieve this goal; they must use other body parts (primarily their feet). One player, however, can use their hands in certain scenarios: the goalkeeper. The goalkeeper is the one position in soccer that is allowed to use their hands in a designated box outside of their net. The objective of the goalkeeper is to defend, or ”save” balls from going into their teams net. Each team is only allowed one goalkeeper on the field at a time, meaning a lot of pressure mounts on this singular position. A popular tactic numerous teams employ is to simply focus on outscoring their opponent; though, many other teams take a more defensive approach (especially if they are stronger defensively rather than offensively) and focus on minimizing the goals of the other team as a means to victory. In either case, the success of the team more often than not hinges on how well their goalkeeper performs.

With the World Cup, one of the largest sporting tournaments worldwide, coming up, there is even more focus on the performance of teams and their individual players. This is because, in the World Cup, the stakes are much higher for the players. Players are not just competing for money and fame, they are competing for national recognition and are representing their nation on a world stage, an opportunity which only comes once every 4 years. As such, anything from in-game tactics, the changing or substitution of players, to even how players train and practice are scrutinized even more by fans, media, and other teams. It goes without saying that, at the time of the World Cup, any movement and decision made by a goalkeeper are at everyone's center of attention.

Along with this, it was found after some preliminary research that there were not many models/projects looking at ideal goalkeeping positioning. Many of the relevant projects found often-times dealt with either attackers passing, positioning, and scoring or defender positioning. Seeing as how vital a goalkeeper is to a soccer team and how the World Cup is coming up, an interesting idea was brought up: where is the ideal position the goalkeeper given certain scenarios? More specifically, how should the goalkeeper be positioned in the scenario where preventing a goal is most difficult (i.e. a 1v1 between goalkeeper and opposing attacker)? Obviously the goalkeeper should stand somewhere in between the ball and goal, but is there an optimal spot? Should the goalkeeper be standing far in front of the goal or closer to the goal line? In order to determine this, the goal of our project is to create a reinforcement learning model that maps the location of a goalkeeper given the location of the opposing attacker.

3 Background Knowledge

In simplest terms, reinforcement learning is when an agent *interacts* with an environment as a means of learning a specific task. This is strictly different from other machine learning techniques because, with reinforcement learning, there is no pre-existing data enabling the agent to "grade" every decision it makes. Instead, a reinforcement learning agent begins with some *policy* (a mapping from states to actions), which initially is usually very wrong, and uses this policy to choose its next *action*. Once the agent performs the action it picked, it will observe a *reward* from the environment and transition to the next state, which is directly determined by the previous state and action taken. The agent learns by updating the approximate value of the state and action pairing (how it does so heavily depends on the chosen reinforcement learning method) which, under the necessary assumptions, will converge to the true value of that state-action pair. As such, with a good reward function, the agent will strive to either minimize its reward (i.e. when the goal is to be completed as quickly as possible) or maximize its reward and, in turn, tweak its policy (i.e. through state-action pair updates) until it has converged to an *optimal* policy.

In terms of our project, we will be using an off-policy Monte Carlo method. The basis of Monte Carlo methods is running many simulations of the agent in the environment with the purpose of accumulating a lot of data. With this observed *real* data, the agent can then estimate (and update its estimates of) the values of states and actions in the environment and formulate (and improve) some policy. Monte Carlo methods can be broken down into *On-Policy* and *Off-Policy* methods. The key difference between the two is that On-Policy Monte Carlo methods use the policy they are trying to learn for interacting with the environment while Off-Policy Monte Carlo methods have a *behavior* policy (used to determine actions) and a potentially different *target* policy (the policy being updating and improved). We chose to employ an Off-Policy, rather the On-Policy, Monte Carlo method because of the added benefit that the agent can still sample all possible actions while strictly improving the *target* policy. Additionally, Monte Carlo methods are a type of model-free reinforcement learning (the agent does not have a complete probability distribution of states and actions) which we deemed appropriate since, from the goalkeeper's perspective, the attacker could theoretically shoot anywhere regardless of its position or the position of the goalkeeper.

4 Related Work

Our project is certainly not the first that applies reinforcement learning to the game of soccer, or skills related to the game of soccer. There have been numerous projects and published papers regarding reinforcement learning agents in some *soccer* environment, though not many cover the problem of optimizing goalkeeper position. Instead, the vast majority of published papers take the perspective of the attacker and have the agent learn to kick the ball, shoot and score, or optimize goal-scoring scenarios. Interestingly, there are also some papers that revolve around optimizing strategy and tactics or evaluating how likely it is for a particular team to score or concede a goal given some state (i.e. snapshot) of a soccer game.

One paper, published just a couple of months ago by Ji et al., regards a Quadrupedal Robot learning how to accurately kick a soccer ball and hit random targets. Seeing as the agent and environment were not simulated and instead the experiment was conducted in the physical world, where aspects like friction and the deformity of the ball are hard to quantify, Ji et al. took the approach of developing a *model-free* RL framework. In order to aid the agent's learning, Ji et al. also decided to deconstruct the goal of hitting a random target into two subproblems: motion control when kicking the soccer ball with one leg and balancing on the other three and motion planning in order to find the optimal way to kick the soccer ball [2]. Even though our experiment deals with the position of the agent and does not consider teaching movement to a physical robot, the work of Ji et al. is still relevant to us. Our project will also deal with randomness (the position of the attacker) and following a similar methodology of decomposing the agent's goal into subproblems (such as learning where the goalkeeper should be and learning whether the goalkeeper should dive or not) could prove beneficial.

A second paper worth mentioning, which is more comparable to our experiment, is *Refinement of Soccer Agents' Positions Using Reinforcement Learning.* In this paper, the author, Tomohito Andou, configures an 11-agent soccer team that will compete in the simulated RoboCup tournament. The specifics of the tournament are elaborated on in their paper. The problem Andou tackles in their paper is learning the optimal *position* for each agent on his RoboCup team. As is explained in the paper, the basis of learning the optimal position for an agent is answering the following question: what is the position where I (the agent) can kick or get the ball most frequently given some state of the game? This overarching question is akin to our experiment because our goalkeeper agent will also have to answer a similar question: what is the position where I (the agent) can save or get the ball most frequently given some state of the game? The experiment undertaken by Andou is clearly more complex than ours, since it takes into account a complete snapshot of the entire field and multiple agents, though a few aspects of their methodology can still be applied to our experiment. For example, the distance to the ball and to the goal are deemed integral by Andou in order to determine agent position. In our experiment, we will make the same assumptions and base the goalkeeper's position on where the attacker is relative to the goal (how far back and how far left or right). Additionally, Andou utilizes stochastic gradient ascent (SGA) as the learning method for the agents' positions. Though not completely necessary for our experiment given the relatively simple and discrete state space, this is a method we could explore in the future if we choose to expand our experiment and allow attacker and goalkeeper to continuously move before the attacker kicks the soccer ball. [1]

Finally, several environments and simulations were looked at for our model. It was originally recommended to us to use an environment for our solution so as to reduce the complexity of implementing some actions on our own. We looked at Google research football, and Robocup, two well known soccer environment simulators. We first looked at Google research football, a 3D soccer simulator compatible with several helpful APIs such as OpenAI Gym. However we considered several recommendations and deemed this as our second preferred option. Along with this, we did attempt to set up and run the environment, however our team was unsuccessful in doing so, resulting in us feeling less inclined to work with Google research football. The main environment looked at was Robocup's 2D environment. Setting this environment up was also challenging for us. We found the process to be not only tedious, but partially unsuccessful; due to technical limitations, only one of us could get the working environment set up on their computer. This resulted in lots of wasted time, along with inefficiencies when developing and testing possible code. Despite this however, we were able to make progress with setting up the server, client, and overall environment. Unfortunately, the documentation provided was not overly helpful for us, resulting in relying on other useful done within this environment for learning purposes and as a base for our project. Out of all the projects, the most useful and the one we looked at the most was a half-field offense simulation. This simulation involved attackers and defenders (both where you could change their actions), alongside a default goalkeeper. We tried utilizing this environment, however we ran into numerous issues. For one, the fact only one of us could use the environment at a time really limited how much work we could do at a time. Not having the ability for both of us to work outside of our meetings and find solutions resulted in needing to spend a lot of time together working, which could not always happen due to our already busy schedules. Another issue, which is the biggest one, was the complexity of the environment. Seeing as the half-field offense focused on simulating different offensive and defensive behaviors, we found it extremely complicated and tedious to understand the base code, and find a way to implement our desired actions for the goalkeeper. The poor documentation also did not help in furthering our progress. After several weeks, we decided to use the simulation as inspiration and move to working within our own environment. This allowed it so we could work on the project individually and build solid processes that we could understand.

5 Methodology

5.1 Overview of Approach

We approached this problem by implementing the goalkeeper as a Multi-Armed Bandit agent. On every episode, the agent observes the attacker's position and then chooses an arm (i.e. an action) that it thinks would best position it in the goal area. The position of the attacker is randomized between five different locations on the edge of the box. The attacker than calculates where in the goal it will shoot and the goalkeeper calculates if, given its own position, it will be able to save the shot. Based on the outcome (save or goal), the goalkeeper will receive a reward and update its estimate of the arm it chose accordingly. The goal of this experiment is for the agent to essentially learn the shot distributions for each attacking position (this will be elaborated on in Section 5.3) and consistently position itself in the optimal place in the goal area in order to save a shot from a given attacking position.

5.2 Multi-Armed Bandit

As alluded to previously, our agent will be a Multi-Armed Bandit. A Multi-Armed Bandit is a type of learning agent which has n number of arms (i.e. actions) that it can pick from. It chooses which arm to pick based on its stored set of arm-value estimates; the arm with the highest estimate will be chosen. With a small probability ϵ, the agent may also choose a random arm regardless of its arm-value estimates as this allows the agent to explore new actions in hopes of finding the optimal action. After choosing an arm, the agent will receive a reward from the environment which it will use to update its estimate of that arm. There are a couple of different methods a Multi-Armed Bandit can employ in order to update its arm-value estimates: the Sample Average method and the Constant Step-Size method. The Sample Average method scales the error term of the update by the amount of times the currently chosen arm has been picked by the agent in the past while the Constant Step-Size method scales the error term by a consistent value each time, α. In this experiment, we chose to use the Constant Step-Size method for arm-value updating whose equation is defined below:

$$Q_{n+1} = Q_n + \frac{1}{n}[R_n - Q_n]$$

To go more in depth, the agent updates its estimate of some arm, Q_{n+1}, based on the sum of the current estimate of that arm, Q_n, and some error. The error term is calculated by scaling the difference between the reward of the arm, R_n, and the agent's current estimate of the arm, Q_n, by some constant value, α. This makes it so the number of times an arm is chosen has no affect on the arm's new estimate. In other words, all arm rewards are valued equally, no matter *when* they occurred, and thus there is no potential bias, or preference, towards earlier or later rewards (which may occur in the Sample Averages method) [3]. Lastly, the agent will be observing several different states in the environment (i.e. the different attacking positions), therefore, instead of conventionally storing a single list of arm-value estimates (which would imply a single-state environment), the agent will store mappings from states to lists of arm-value estimates.

5.3 Environment Specifics

A real-world "field" and "goal" can be described as almost like two perpendicular sides of a cube: the field being the base of the cube (XY plane) and the goal being a side of the cube (XZ plane). As such, we will represent both the field and the goal as independent 2D arrays. Arrays are efficient and simple data structures and their ability to be indexed allows for the continuous space of the goal and goal area to be easily discretized. As such, the goal area is represented by a 24x18 matrix where each cell is one square-foot of space on the field. It should be noted that the width of the matrix is slightly less than that of a true goal area, though this is due to the matrix being bounded by the two goal posts which ensures the goalkeeper is always between the goal posts. Similarly, the plane of the goal itself is represented by a 24x8 matrix where each cell again represents one square-foot of space. The state-space of our experiment will be the five possible positions the attacker can be in: LEFT, LEFT-MIDDLE, MIDDLE, RIGHT-MIDDLE, and RIGHT. The location in the goal where the attacker will shoot the ball is computed by a probability distribution based on the attacker's position. For example, if the attacker is in the MIDDLE position, then it will have a fairly even shot distribution across the entire goal and a relatively high probability of shooting within the middle two thirds of the goal. If the attacker is in the RIGHT position, however, then its shot distribution is fairly skewed to right side of the goal. We made this design choice based on the tendency of players to aim towards the near post rather than the far post (or middle of the goal) when shooting from an angle. It is important to note that the agent is only given the position of the attacker and chooses an arm solely based on this information (it does not know the location of the attacker's shot nor the shot distribution for each attacking position). The action-space of our experiment is simply the cells of the matrix representing the goal area as each cell represent a location in which the agent can position itself in.

5.4 Agent and Experiment Specifics

In this experiment, the Multi-Armed Bandit agent has an exploring rate (the percentage of times it will choose an arm at random), ϵ, of 0.1 and a learning step-size (the scaling factor in the update step), α, of 0.1. Additionally, 10 independent agents will run (to help mitigate variance and noise caused by randomness) each for a total of 2,000,000 episodes. The reward for each episode of each agent will be stored and aggregated in order to calculate the running average of rewards across all episodes and agents. One thing to note about our reward function is that the reward for saving a shot is 0, and for letting a goal in being -1. This is due to the fact that our action space is so large (every arm is a position in the savable area). Having no reward for saves allows for the greedy approach to randomly choose the best action, resulting in more actions/arms being explored. These results and the optimal position the agents learned for each attacking position will be analyzed in Section 6.

5.5 High-Level Procedure

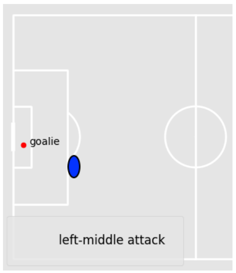
For the sake of conceptualizing the work-flow of our experiment, high-level pseudocode has been provided below:

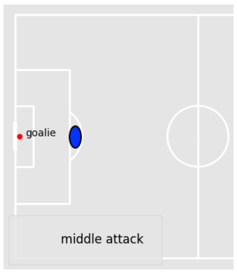
```
1. Initialize attacker S, goalkeeper G, and step-size $\alpha$
    (a) $Q(a, s) \leftarrow 0$ where $a$ is some goalkeeper action and $s$ is some attacking position state
    (b) $\alpha \in (0, 1]$
2. Loop forever:
    (a) P $\leftarrow$ S.newPosition()
    (b) G.newPosition(P)
    (c) L $\leftarrow$ S.newShotLocation()
    (d) R $\leftarrow$ G.saveOrNot(L)
    (e) Q(a, P) $\leftarrow$ Q(a, P) + $\alpha$[R - Q(a, P)]
```

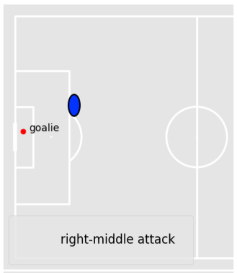
6 Experimental Results

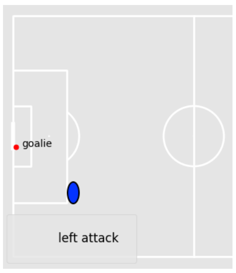
6.1 Optimal Positions

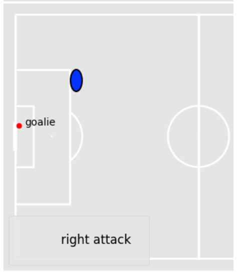
The figures below showcase the optimal goalkeeper position given the position in which the attacker is in. The position of the goalkeeper is denoted with a red dot and the attacker position is denoted with a blue area (seeing as the the specific position of the attacker is never specified). By examining each of the positions the agent deemed optimal, it is clear that the agent did in fact learn the shot distributions of each attacking position. In each scenario, the goalkeeper is correctly positioned in the ”line of sight” of the corresponding attacking position. That is, if a line was drawn from the attacking position to the goal, it would most likely intercept with the goalkeeper position or come close enough to the goalkeeper position that it would intercept with the savable area of the goalkeeper. In either case, this would result in the goalkeeper saving the shot. In simpler terms, the goalkeeper mirrored the attacking position in such a way that the angle of the goal, relative to the attacker, was cut off the most. This supported our hypothesis of where the agent should position itself and mimics the positioning of real life goalkeepers. One thing to note, however, is that the distance away from the goal line varies across the different attacking positions. For example, the goalkeeper is roughly half-way from the goal line and the edge of the goal area when the attacker is in positions LEFT-MIDDLE and RIGHT-MIDDLE, but is then very close to the goal line when the attacker is in positions LEFT, MIDDLE, and RIGHT. We attribute this discrepancy between optimal goalkeeper positions to the noise of our model introduced by different randomness across all the agents and our imperfect shot distributions for each attacking position.





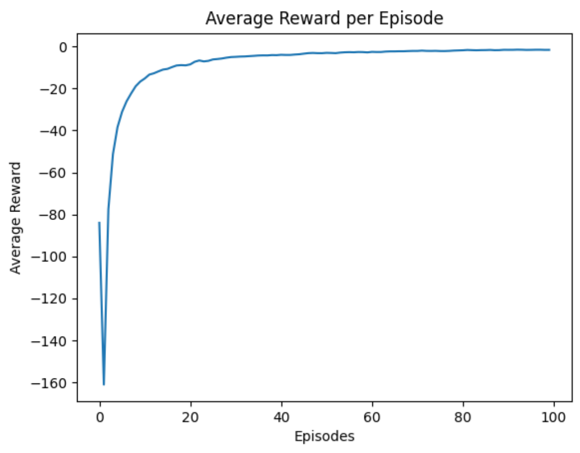






6.2 Rewards Received

The figure below shows the running average reward per episode across 100 bandits. Note that the number of episodes per agent was decreased to 100 for the sake of visualizing the agents converging. In the actual experiment, the bandits simulated the number of episodes specified in Section 4.4. By analyzing the graph, it is clear that the agents underwent a substantial amount of learning. The stark decrease in reward in the initial episodes can be attributed to the agents just not knowing which actions were more and less optimal. Not enough exploring occurred up until this point (since the episodes just started), so the majority of the agents' arm-value estimates were still equal to their initial values. This resulted in the agents picking arms pseudo-randomly, as they broke ties between "equally optimal" arms randomly, and evidently letting the attacker score frequently. Soon after this initial period of large negative reward, the agents garnered a sufficient amount of information about their different arms (whether through exploration or breaking ties randomly) and were able to choose more and more optimal arms as the episodes went on. This can be seen by the swift increase in average reward just after the initial negative spike of reward. Towards the second half of episodes, the average reward graph can be seen to converge at just about a reward of 0, signifying that the agents have found optimal arms for the different attacking positions. The average reward of the agents never went above 0 due to our reward function (see Section 5.3). It should also be noted that the agents converged fairly quickly relative to the number of arms they have. This is due to the fact that the goalkeeper has a savable area that extends beyond its current position. As such, there are almost definitely multiple goalkeeper positions that are sufficient in saving the majority of shots from a given attacking position which is what the agents are converging to. This is ultimately the reason why the agents need to run for 2,000,000 episodes in order to converge to *one* optimal goalkeeper position per attacking position.



7 Conclusion

To conclude, we found our model to be relatively successful. As seen in our results, our model was able to use reinforcement learning to actively improve upon the saving position of our goalkeeper. Along with this, the positioning of the goalkeeper followed our hypothesis in the idea that the goalkeeper would mirror the attacking position so as to maximize the savable area. One thing to note is the vertical distance the goalkeeper is from the goal line. Seeing as it varies slightly for each attacking position, it seems as our model did not fully work out the best vertical position overall in this situation and has some noise. However, given everything else, it seems as our model was successful in determining the optimal positioning for a goalkeeper in the given space.

8 Future Work

In terms of future work, there are many different avenues that could be taken. For one, our model only illustrated one type of scenario: an attacker shooting from specific spots. This concept, while effective for determining positioning, is quite simplistic; it does not reflect real-world situations. In a given game-time situation, there are often times more than one attacker, alongside several defenders. On top of this, each individual player has more complex actions besides shooting, such as dribbling and passing, making the situation a lot more complex. Future work could include incorporating these different actions and complex situations for the goalkeeper, allowing for a more accurate representation of where a goalkeeper should be positioned in a real soccer game. Another avenue of future work could be involving a simulator for our work. Throughout the building of our model, several different simulators were tried in an attempt to better illustrate our problem and the situations our goalkeeper was working with. However, due to time constraints these attempts were unsuccessful, resulting in us not having a more illustrative visual representation of our problem/results. Given the infrastructure we found, incorporating our model a soccer simulator such as RoboCup Simulator would be an excellent way to further this project.

References

[1] Tomohito Andou. *Refinement of Soccer Agents' Positions Using Reinforcement Learning.* Springer-Verlag Berlin Heidelberg, 1998.

[2] Yandong Ji, Zhongyu Li, Yinan Sun, Xue Bin Peng, Sergey Levine, Glen Berseth, and Koushil Sreenath. *Hierarchical Reinforcement Learning for Precise Soccer Shooting Skills using a Quadrupedal Robot.* 2022.

[3] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction (2nd ed.).* The MIT Press, 2018.