Project Proposal

Egger, Maximilian — Pelat, Guillaume — Pilger, Laura

April 28, 2021

Abstract

In this proposal, we examine the necessary steps to train a Reinforcement Learning agent to play a simplified version of Haxball. Starting with a baseline policy obtained from Dynamic Programming, we designed a feature-based approximation of the state space and potential reward functions to be used with Temporal Difference Learning.

1 Problem description

The goal of our project is to train an agent to play a modified version of Haxball, which is a physical-based simulation of football. In our scenario, there is only one agent on the field and the objective is to send the ball into either of the goal areas. Ball and player are initialized with random positions, but at each score, the ball is repositioned in the center of the field.

This task presents two main challenges. We deal with a continuous state space and need to find a good way to represent it. Furthermore, there are a lot of bad actions the agent can take at each step, hence the chances of discovering a good policy are very low without a carefully designed reward function.

2 Motivation

Although the environment is completely deterministic and simplified, estimating future difficulties and the expected progress for applying Reinforcement Learning (RL) to this setting is still quite challenging. Especially forecasting the learning curves of the algorithms and the required hyperparameter tuning for this particular problem requires some experience. This project enables us to try out very different approaches, ranging from designing a feature space to implementing and validating different algorithms and their practicability. Predetermining the best actions for all possible states would be very expensive and inflexible with regard to a changing environment. This legitimates the approach of using Reinforcement Learning for this task.

3 Milestones

The final objective is to enable the agent to score as many goals as possible in a given amount of time, but therefore, it is essential to achieve some smaller milestones on the way. As a first milestone, it might be the easiest to start by teaching our agent to move to a certain position. Next, the agent should be directed towards the ball and subsequently induced to shoot. With the following milestone, the correct alignment of the shot towards the goal should be achieved. Even though the future modifications are out of scope for this initial report, it should always be kept in mind that we might also need to defeat a goalkeeper or another agent in a later version of this project.

4 Ingredients

Throughout this section, necessary building blocks are investigated. In particular, we focus on obtaining a feature space by abstraction of the state space, describing the possible actions of the agent and designing a reward that is capable of leading the agent towards reaching the milestones defined in section 3.

4.1 Features

In the current version, the states describing the dynamic changes of the environment consist of three two-dimensional continuous variables. The agent and the ball position are each represented by two variables (x, y) describing the respective positions on the field, where $x \in [-4, 4]$ and $y \in [-2, 2]$. The ball velocity is described by two orthogonal variables (v_x, v_y), each bounded by $\pm 3 \cdot 0.075$. Stationary environment descriptions contain the goal positions and the size of the player, the ball and the field.

Purely discretizing this continuous state space requires a proper precision management and results in a huge amount of low-level feature combinations which in turn complicate the agent's learning and decision process. In order to condense the available information and thereby reduce the state space to a level being sufficient for the agent's decision making, we introduce the following encoded high-level features:

- A boolean describing the ability to shoot, resulting from the player and the ball position combined with their sizes.
- Two two-dimensional ternary variable describing the direction from the player to the ball in terms of compass directions (horizontal: W, 0, E; vertical: S, 0, N), i.e., $x, y \in \{-1, 0, 1\}$ for both the current and an estimation of the future ball position. This requires both position vectors and the ball velocity.
- Two integers describing the discretized angles from the player towards the goals, e.g., in steps of $10°$.

These features transform the external view on the environment to a player's view while discretizing the information of interest. In the above case, this results in seven features and $2 \cdot 3^4 \cdot 18^2 = 52488$ possible combinations. Future findings might give rise to adapt the set of features or the discretization to either increase the mutual information between the true environment and the agent or to further reduce the feature space. However, this should be a reasonable starting point for an initial try.

4.2 Actions

While we drastically decreased the sample space, the action space did not need further improvements. Hence, the possible actions are designed as a three-dimensional vector, with:

- The first entry representing the horizontal moving direction along the x-axis, entailing the options west (-1), no movement (0) and east (1).
- The second entry representing the vertical moving direction along the y-axis, entailing the options south (-1), no movement (0) and north (1).
- The last entry representing the action of shooting with the options to shoot (1) or not to shoot (0).

The first two components are then multiplied with the maximum speed of the player to obtain the player's velocity in both the x and y direction, whereas the third is used as a boolean and only realized, if the ball is in shooting distance of the player. This design leads to a total of 18 possible actions.

4.3 Rewards

Since the development objective is to enable an agent to score goals, it seems quite straightforward to design a reward function that rewards the agent every time it succeeds in this task. However, given the continuous state space and the set of different actions, it is unlikely that the agent actually achieves this by chance, as it has no idea or guidance on how to perform the preliminary steps for scoring a goal. Therefore, our reward function must also give incentives that will drive the agent to move closer to the ball and shoot in the right direction in order to enable learning. While designing a multi-part reward function, we have two options that we intend to compare:

1. The sequential approach that uses a reward function dependent on the training stage. At first, we would reward the agent for moving towards the ball, then change it to give a reward for each shot in the right direction, and later on for a goal.

2. The scaling approach that puts everything in the same reward function and scales the different parts accordingly. With this approach, the agent could learn faster, but we need to tune the hyperparameters of the scaling to avoid counter-productive behaviors. As an example, keep moving around the ball instead of shooting could turn out to be more profitable for the agent, if we chose the reward for moving towards the ball as too large.

The different parts of the reward function can easily be evaluated by utilizing the current and the subsequent state space, i.e., considering the euclidean distance between the agent and the ball, the movement of the ball w.r.t. the goal and the ball position in regard to the goal areas.

5 Algorithms

Out of all the algorithms seen in class, we have selected two that seem suitable for this task.

5.1 Dynamic Programming

Since we have a continuous state space, applying Dynamic Programming (DP) to this task is not possible as it is. However, the environment here is completely deterministic, which means that we could use it somehow. Hence, we plan to discretize the field in a 200*100 grid and apply DP on this approximated state space.

Since the loss in precision compared to the original state space is big, we do not expect to obtain an optimal policy that way. However, this will surely provide us with a policy that is at least "not bad", especially compared to other techniques relying too heavily on a good exploration in order to converge. The policy obtained this way will then be used as a baseline and compared against our future results.

5.2 Value Function Approximation

We designed our features with Temporal Difference Learning in mind, using it with Linear Value Function Approximation. In this algorithm, a good exploration strategy is the key: if we reward the agent for scoring, but it has no idea on how to hit the ball, no learning will be achieved. As already discussed in subsection 4.3, we plan on implementing and evaluating two different reward function approaches in order to tackle this problem.

In addition, it is obvious that most actions will only see their reward multiple steps later – especially in the sequential approach, in which we will give no further reward for moving towards the ball in later training steps. Hence, the use of an eligibility trace for our weights seems unavoidable.

6 Time Plan

To ensure the success of this project, we decided to have weekly meetings where we evaluate our progress with regard to our time plan, which is depicted in Figure 1. The milestones described in section 3 are not directly reflected by this time plan since we aim to achieve all milestones until the intermediate report submission in order to use the remaining time for further improvements or modifications.

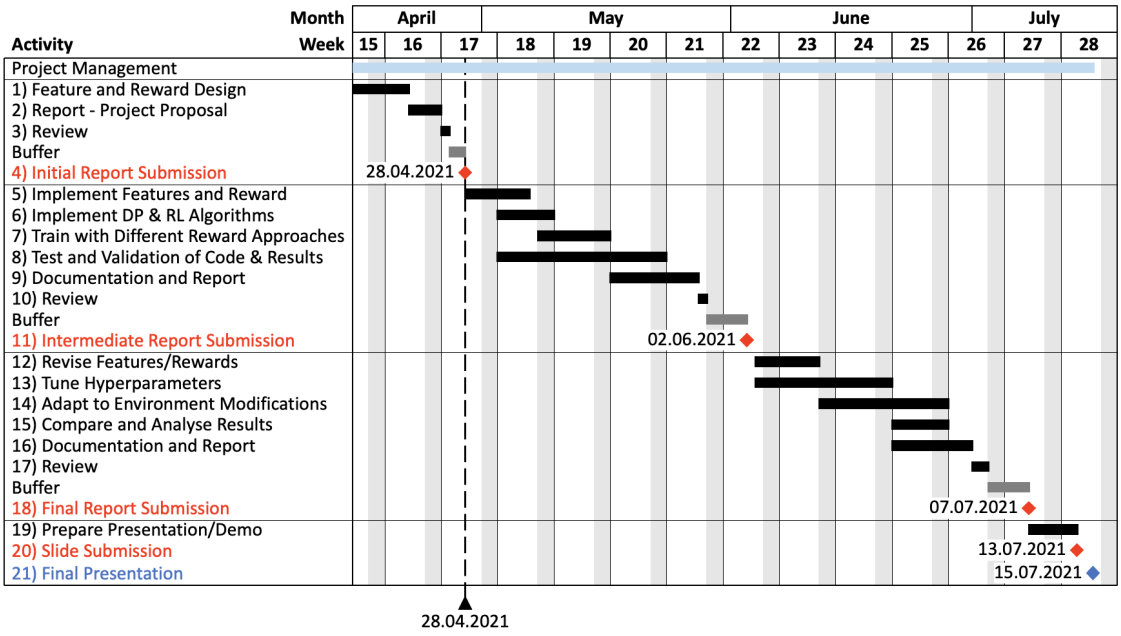

Figure 1: Time Plan

The preliminary task assignment, which is illustrated in Table 1, should provide an overview of the estimated workload for each team member. Throughout the project, however, the distribution of tasks may change depending on the actual progress.

Table 1: Preliminary Task Assignments

Team Member	Tasks
Egger, Maximilian	1, 2, 5, 8, 9, 10, 11, 12, 14, 16, 19, 21
Pelat, Guillaume	1, 2, 6, 7, 9, 13, 15, 16, 17, 18, 19, 21
Pilger, Laura	1, 2, 3, 4, 6, 7, 9, 13, 14, 16, 19, 20, 21