CS698R: Deep Reinforcement Learning Project

Final Project Report

Team Name: Deep Team
Project Title: Penalty Shot Game Environment Dynamics
Project #: 1
Team Member Names: Roll No
Aditya Gupta : 190061
K Nikita : 180335
Nitesh Kumar : 21111276
Umang Pandey : 180833

1 Introduction

Learning sources by which agents can communicate with complicated environments and individuals is a crucial aim of decision neuroscience and Game theory Myerson [2013]. In most experimental standards, choices are discrete, play is static, and optimal solutions are known. Yet, in real environments, communications between agents typically involve continuous action spaces, continuous dynamics, and no known optimal solution. Here, we overcome this issue by using a "penalty shot" task in which agents compete against each other in a competitive, real-time video game.

This paradigm generates a rich complexity in individuals' behavior that can be succinctly described by individualized, instantaneous policy and value functions, facilitating analysis at multiple timescales of interest. It quantifies complex interactions between various agents in a parsimonious manner and suggests new classes of tractable paradigms for studying human behavior and decision-making. From a game-theoretic and a neuroscientific point of view, it can give us tremendous insights into decision-making. Briefly describe the problem you are trying to solve, why the issue is relevant, motivation for solving the problem, and possible implications.

In literature survey, we found that the algorithms can be classified based on action/state space's type, which is – continuous or discrete. Algorithms like Q-learning and SARSA are helpful when the environment has a discrete space, i.e., There is a measurable collection of actions and states present in discrete space. No finite state and action are found in continuous space, as a range of values defines the continuous space. To deal with continuous spaces as discrete spaces, two popular approaches are used, namely the discretization of space and the second is function approximation. The problem with discretization is sometimes there becomes a need for sizeable discrete space, then only a second approach is applicable. For the continuous state space but the discrete action space, deep q learning seems to work the best. Suppose we have both state and action space continuous, as in our problem statement. In that case, policy-based approaches such as s Proximal Policy Optimization (PPO) and value-based in combination with policy-based actor-critic performs better.

Our primary focus is to solve the problem in continuous action and state spaces; but, we also try discretized action spaces. After discretizing action space, we apply the Deep Q-Network (DQN) algorithm Mnih et al. [2013], Fan et al. [2020], For continuous action space experiments, we have used algorithms such as Proximal Policy Optimization (PPO) Engstrom et al. [2019], Deep Deterministic Policy Gradient (DDPG) Li et al. [2019], and Advantage Actor Critic(A2C) Grondman et al. [2012].

2 Related Work

The authors in McDonald et al. [2019] chose a reinforcement learning approach, and they use Gaussian Processes to model the policy and value functions of members as a function of both game state and opponent identity. They decided that higher-scoring participants timed their last change in the way to moments when the opponent's counter-strategy was weaker. In contrast, lower-scoring participants timed their final moves less precisely. Their method gives a natural set of metrics for promoting analysis at multiple timescales and suggests new classes of experimental models for assessing behavior.

In Canese et al. [2021] and Nguyen et al. [2020] the authors present a review of challenges and solutions with a few approaches and applications for different fields for multiagent reinforcement learning. They point out the main challenges like nonstationarity, scalability, and observability when moving from single agent to multiagent. The solution approaches they mentioned in the category of Value-based, Actor-critic based and Policy-based. These approaches overcome the challenges present in the multiagent environment. Hence, we can use them to solve the challenges for our environment, as our's also use a multiagent setting.

In Schulman et al. [2017] authors propose a novel approach that belongs to the policy gradient method. They introduce a unique objective function that allows various epochs' minibatch updates. They termed their approach as proximal policy optimization (PPO). They also mentioned that their approach is simple for implementation.

The authors in Hong et al. [2017] present an approach called deep policy inference Q-network for targeting multiagent systems that include agents which are collaborative and competitive as well when interacting with each other. They apply policy features learned from the raw observations of agents by gathering their policies. They include the learned policy features within the deep Q-Network (DQN) as a hidden vector and predict the better Q values. They also control the partial observability problem by enhancing their mechanism. They use a soccer game to evaluate their models.For the multiagent cooperative approach, another paper Diallo et al. [2017] is also present. In this, they examine the behavior of agents for the cooperative environment using DQN (deep Q-networks). For investigation, they use the doubles pong game. In the experiments, agents learn together and divide their ability and use their DQN for behavior.

In Li et al. [2019] authors proposed an approach for continuous actions and multiagent learning settings. Their focus was to generalize the trained agent even if the competitors' policies change. To overcome this issue, they propose an algorithm named MiniMax Multi-agent Deep Deterministic Policy Gradient (M3DDPG). They add the minimax addition to the existing approach called as multiagent deep deterministic policy gradient algorithm (MADDPG).

3 Problem Statement

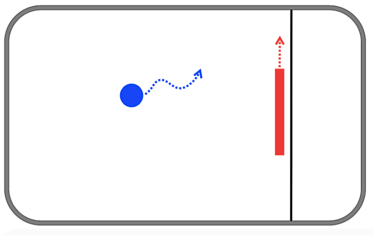

Figure 1: Trajectory of the puck during a penalty shot game trial

Figure 1 shows the game environment. The problem statement here is:

- To find the optimal policy where all agents (participants) get their own choice of action given the dynamic of the problem is fully competitive where return of all the agents sums to zero.
- But the model of MDP does not allow multi agents to work in the same environment. Hence Markov Games is the extension of MDP model for the multi-agent case.

In this game, we have two players,

- The Kicker - who controls the puck.
- The Keeper - who controls the bar.

The kicker tries to get the puck across the goal line, and the keeper tries to prevent him by obstructing him with a bar. If the keeper manages to prevent the bar from crossing across the goal line, he wins, else, the kicker wins.

Initial normalised coordinates at the start of each trial:

- Puck : It is represented as a coloured circle (of diameter 1/64 of the screen width).
$$x_p, y_p = -0.75, 0 \tag{1}$$
- Goal line : It is fixed on the screen during the whole trial.
$$x_g = 0.77 \tag{2}$$
- Bar: It is positioned before the goal line.
$$x_b, y_b = 0.75, 0 \tag{3}$$

Movements during the trial:

- For puck: The puck was constrained to remain onscreen at all times.
  - Puck moved with constant horizontal velocity v_p and its x-coordinate was updated as follows:
$$x_{p,t+1} = x_{p,t} + v_p \tag{4}$$
  - Its vertical velocity is, $v_p * u_{p,t}$, where $u_{p,t} \in [-1, 1]$ is the vertical joystick input (controlled by the participant) at time t. Its y-coordinates were updated as follows:
$$y_{p,t+1} = y_{p,t} + v_p \cdot u_{p,t} \tag{5}$$
- For bar:
  - The bar could only move up or down. But, if the opponent maintained direction at near-maximal input ($|u_{b,t}| \in [0.8, 1]$) for three consecutive time steps, the bar's maximal velocity began to increase on the third step.
  - That is, at each time step, the y-coordinates of the bar are updated as follows:
$$v_\omega \leftarrow \frac{2}{3} \cdot \theta \cdot v_p \tag{6}$$
$$\theta \leftarrow \begin{Bmatrix} \theta + 0.85, & if\ accelerating \\ 1 & otherwise \end{Bmatrix} \tag{7}$$
$$y_{b,t+1} = y_{b,t} + v_\omega \cdot u_{b,t} \tag{8}$$

3.1 Problem Description

The agent controlling the puck begins each trial at the left of the screen and moves rightward at a constant horizontal speed v_p. The agent's objective is to score by crossing the goal line at the right end behind the opponent's screen. The opponent's task is to block the puck from reaching the goal line using the bar. Each agent moves their avatar using a joystick. Both players can only control the vertical velocities of their respective avatars. The puck and bar have distinct game physics, as described above. If the agent handling the puck can get to the finish line, it receives a (+1) reward, and the agent controlling the bar gets a (−1) reward. Similarly, if the agent controlling the bar manages to stop the puck, it gets a (+1) reward, and the other agent gets a (−1) reward.

4 Environment Details and Implementation

4.1 MDP

- In general the Markov Games for n agent is a tuple $< N, S, (A_i)_{i=1}^n, (R_i)_{i=1}^n, T >$, where N is the set if agents, S is the state space, A_i is the action space of agent i (i = 1,2,..,n). Now Let $A = A_1 \times ... \times A_n$ be the joint action space, then Reward function R be like $R_i : S \times A \longrightarrow R$ for agent i and $T : S \times A \times S \longrightarrow [0, 1]$ is the transition function.
- In our problem $n = 2$ (puck and bar), state space for bar is $Box(-1+radius, -1+radius)$ to $(1-radius, 1-radius)$ and state space for puck is continuous value from range of $(-1+lenbar/2)$ to $1-lenbar/2$ and action space for both agent is [-1,1].

4.2 Implementation

We initialize the game as per the conditions described in section 3 and define two-step functions, one for the puck and the other for the bar. Both the step functions take u_t as input parameter.

After performing each step, the standard four values returned from an Open AI Gym environment-which are observation, reward, done, info. At the time of initialization, the environment takes the inputs, d, v_p, `lenbar`, `wbar` these stand for diameter of puck, horizontal velocity of puck, length of the bar, and width of the bar. The game is played out as described in Part 3 of this report. At the end of every episode, the states are reset. We have done the implementation in two parts; first, the action space is considered continuous, and second, the action space is converted as discrete.

For training the model with stable baselines, we created two environment methods; one takes the policy of bar and action of pucker as input and the other vice versa. So both the environment methods predict the opponent's action using the policy that they take as input. We created two models to predict the action of the puck and bar, respectively. In each episode, we trained both models on initial environments with random policies. After the models were trained, we saved them and then created new instances of environments with the now trained policy of the puck and bar. Then in the next episode, we load the saved models on these new environment instances and then continue as the previous episode.

5 Proposed Solution

In this section, we discuss the proposed solution for the problem explained in section 3. As discussed in the section 1, to deal with continuous action and state-space, either we have to discretize the space, or we can use the algorithms that can handle the continuous space. Therefore,we experiment with both approaches and compare the results. The proposed solution is as follows:

- We implement the penalty shot multiagent environment.
- We create two instances of the environment, first with the discrete action space and second using continuous action spaces.
- Environment setting was :- Diameter of puck = 0.03, Height of bar = 0.7, Width of bar = 0.01. Horizontal velocity of puck was set to 0.3 and 0.01 for different cases.
- For discrete space setting, we apply DQN and double DQN (DDQN) approaches and check the performance.
  - **DQN:** The authors first present the DQN algorithm in Mnih et al. [2013] that uses the concept of neural network. DQN provides the Q values for each possible action (a $\in$ A) as output and uses state (s) as an input. In DQN, prior experiences are collected, and the following action is found with the best output of the Q network. The drawback of DQN is that after every repetition target changes, it causes a non-stationary problem. It also leads to overestimating action values which drive towards unstable training and poor quality policy.
  - **DDQN:** To overcome the problem of DQN, the authors in Hasselt [2010] present the simplest form of DQN, which is DDQN. In this algorithm, two separate Q value estimators are used, which update each other. We unbiased Q-value judgments of the actions chosen utilizing the opposing estimator. We can thus evade maximization bias by freeing our updates from biased judgments.
- For continuous space setting, we use PPO, DDPG, and A2C approaches and review the results.
  - **PPO:** Proximal policy optimization (PPO) Schulman et al. [2017] is one of the numerous successful deep reinforcement learning approaches, producing state-of-the-art achievement over a wide variety of exciting tasks. PPO seems to balance ease of implementation, tuning, unit complexity, attempting to calculate an update at every step that reduces the cost function while assuring the variation from the prior policy is comparatively tiny.
  - **DDPG:** Deep Deterministic Policy Gradient (DDPG) Li et al. [2019] algorithm simultaneously learns a Q-function and a policy. It learns the Q function with the help of the Bellman equation and off-policy data. For policy learning, it uses the Q-function. It is also interleaving an approximator to optimal

action-value function with optimal action. It behaves in a way adapted explicitly for environments having continuous action spaces.

– **A2C** Grondman et al. [2012] method comes under an on-policy algorithm and a policy gradient approach. It contains two networks actor and the critic, and they work mutually to solve a problem. In this advantage function is used, which finds the TD/prediction error. At every timestamp, the actor selects an action and critic network to evaluate the Q-values' quality for an input given. As the critic learns about the state's quality as good or bad, the actor uses this data to tell the agent to try a good state rather than a bad one.

6 Experiments

Our experiments were mainly divided into 2 categories based on whether the action space used is discrete or continuous . First, we tried to train the agent by discretizing the environment's continuous action space. Then we trained the agent on the vanilla continuous action space.

These two categories were further divided into two sub-categories each. The horizontal velocity of the puck was high in one sub category, while it was very low in the other category. The purpose of this division was to show the dependence of converging policy and victory on velocity. We hypothesized that the bar agent will win for low velocities, as it would have more time to develop its acceleration. whereas, for high enough velocity, the puck will always win as it has higher base velocity.

7 Results and Analysis

Experiments with discretized action space did not yield satisfactory results.For a small size action space the rewards were oscillating and for a sufficiently large action space the training time was very long. On top of that, the final rewards obtained for the agents was not up to the mark either.

Due to the above-mentioned reason and the fact that one of the main reason the inventors of this game made this game was to test decision-making in a simple continuous environment and discretizing the action space would make the whole existence of this game futile, we decided to stop pursuing any further research and experiments with the discretized action space environment

Experiments with a continuous action space on the other hand gave great results.The training time were not too long, and the rewards returned were high.

The possible reason for this behavior might be that loss of information and dexterity generated by the discretization of the continuous action space was too high to the point that it changes how the underlying rules of the game.

We then looked at the average rewards obtained by the agents for different velocity profiles and we noticed that for low velocities the bar always won and for high velocities the puck always won. Figure 2 and 3 shows the experimental results for two different settings for velocity with various algorithms.

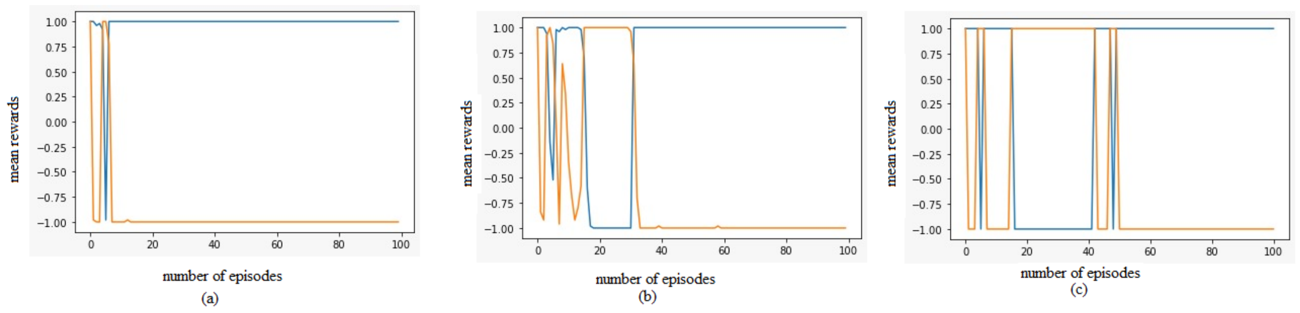

Figure 2: Experimental results for various algorithm with velocity 0.3 mean rewards vs number of episodes (a) A2C, no. timesteps = 1000, no. episodes = 100, (b) PPO, no. timesteps = 1000, no. episodes = 100, and (c) DDPG, no. timesteps = 1000, no. episodes = 100

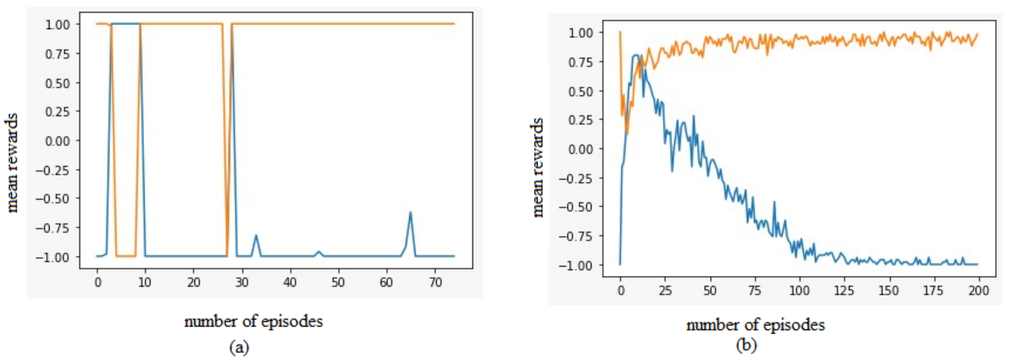

Figure 3: Experimental results for various algorithm with velocity 0.01 mean rewards vs number of episodes (a) A2C, no. timesteps = 1000, no. episodes = 70, (b) PPO, no. timesteps = 1000, no. episodes = 200

8 Future Directions and Conclusions

In this work, we have conducted a systematic evaluation of our defined problem using two experiments first is finding the optimal results with discretized action space, and the second is with continuous space. We use baselines for the model training and evaluation. After evaluating the approaches, we found that the experimental results for continuous action space settings are better than the discrete ones. Discrete action space setting experiments show that the policies were using a lot of time to evaluate and did not perform well. In contrast, the continuous action space setting converges very fast. We also perform experiments with multiple hyperparameters during model tuning and with dynamics of the environment, like changing the puck's velocity, the bars length etc.

Future work direction can be concerned with generalizing the approach to overcome as much as possible challenges in the multiagent environment such as non-stationarity, partial observability, etc. Another focus will be to enhance and explore more advanced approaches to solve the problem, such as more customized variants of our algorithms. Another exciting avenue of research could be the use of Gaussian Processes to model agents' policy and value functions as a function of both game state and opponent identity. An exciting and novel research prospect would be to look at the whole game as a perfect information extensive form game or a Bayesian form game. We might solve the problem using techniques such as MTCS (Monte Carlo tree search), etc.

9 Member Contributions

Table 1 shows the contributions from each of the members.

Table 1: Member Contributions

Name	**Contributions**
Aditya Gupta	Model Implementation,Report ,Presentation,Research
K. Nikita	Environment Rendering
Nitesh Kumar	Set up MATLAB and rendering using MATLAB code,Model Implementation,Report , Presentation, Research Literature Review, PPT, Report,
Umang Pandey	Environment implementation,Model Implementation,Report ,Presentation,Research

References

Lorenzo Canese, Gian Carlo Cardarilli, Luca Di Nunzio, Rocco Fazzolari, Daniele Giardino, Marco Re, and Sergio Spanò. Multi-agent reinforcement learning: A review of challenges and applications. *Applied Sciences*, 11(11): 4948, 2021.

Elhadji Amadou Oury Diallo, Ayumi Sugiyama, and Toshiharu Sugawara. Learning to coordinate with deep reinforcement learning in doubles pong game. In *2017 16th IEEE international conference on machine learning and applications (ICMLA)*, pages 14–19. IEEE, 2017.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep rl: A case study on ppo and trpo. In *International conference on learning representations*, 2019.

Jianqing Fan, Zhaoran Wang, Yuchen Xie, and Zhuoran Yang. A theoretical analysis of deep q-learning. In *Learning for Dynamics and Control*, pages 486–489. PMLR, 2020.

Ivo Grondman, Lucian Busoniu, Gabriel AD Lopes, and Robert Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307, 2012.

Hado Hasselt. Double q-learning. *Advances in neural information processing systems*, 23:2613–2621, 2010.

Zhang-Wei Hong, Shih-Yang Su, Tzu-Yun Shann, Yi-Hsiang Chang, and Chun-Yi Lee. A deep policy inference q-network for multi-agent systems. *arXiv preprint arXiv:1712.07893*, 2017.

Shihui Li, Yi Wu, Xinyue Cui, Honghua Dong, Fei Fang, and Stuart Russell. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4213–4220, 2019.

Kelsey R McDonald, William F Broderick, Scott A Huettel, and John M Pearson. Bayesian nonparametric models characterize instantaneous strategies in a competitive dynamic game. *Nature communications*, 10(1):1–12, 2019.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Roger B Myerson. *Game theory*. Harvard university press, 2013.

Thanh Thi Nguyen, Ngoc Duy Nguyen, and Saeid Nahavandi. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE transactions on cybernetics*, 50(9):3826–3839, 2020.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Appendix

A Experimental Details

B Extra Plots

C More Information, etc.

Arguments of environments:-
env1(diameter of puck, horizontal velocity of puck, lenbar, widthbar,policy of bar)
env1.step(action of puck)
env2(diameter of puck, horizontal velocity of puck, lenbar, widthbar,policy of puck)
env2.step(action of bar)