Penalty Shot Game Environment Dynamics

Group 8:
Aditya Gupta (Roll No.: 190061)
K Nikita (Roll No.: 180335)
Nitesh Kumar (Roll No.: 21111276)
Umang Pandey (Roll No.: 180833)

Mentor and TA: Avisha and Aishwarya
Special Thanks: Gagesh
Course: CS698R (Deep Reinforcement Learning)

November 28, 2021

Outline

- Problem Statement
- MDP
- Literature Review
- Environment details
- Proposed Solution
- Experiments and Results
- Future Directions and Conclusions
- Contribution

Problem Statement

Problem Statement

Penalty Shot Game Environment Dynamics

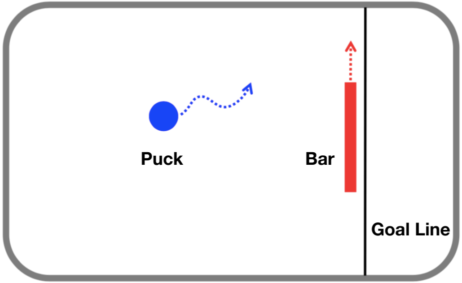


Figure 1: Trajectory of the puck during a penalty shot game trial

Problem Statement

The problem statement here is:

1. 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.
2. 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.

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.
- 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.
- NOTE: Both players can only control the vertical velocities of their respective avatars.

MDP

MDP

1. 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.
2. 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].

Literature Review

Literature Review

1. In [1] and [3] the authors present a review of challenges and solutions with a few approaches and applications for different fields for multi-agent reinforcement learning. They point out the main challenges like **nonstationarity, scalability, and observability** when moving from single agent to multi-agent. 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 multi-agent environment. Hence we can use them to solve the challenges for our environment as our's also use a multi-agent setting.
2. In [4] 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. They have performed tests on multiple simulations and benchmark operations in their fields. Their experimental results outperform other available online policy methods.

Literature Review

1. In [2] authors proposed an approach for continuous actions and multi-agent 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 multi-agent deep deterministic policy gradient algorithm (MADDPG). It helps them to strengthen policy learning. They test their algorithm in various mixed multi-agent environments and register the improved results outperforming the existing ones.

Environment details

Environment details

- 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:

Environment details

- 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:

$$v_\omega \leftarrow \frac{2}{3} \cdot \theta \cdot v_p \tag{6}$$

$$\theta \leftarrow \begin{Bmatrix} \theta + 0.85, & \mathit{if\ accelerating} \\ 1 & \mathit{otherwise} \end{Bmatrix} \tag{7}$$

Environment details

- We initialize the game as per the conditions described 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 and at the end of every episode, the states are reset.

Proposed Solution

Proposed Solution

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 in this work, we have tried experimenting with both approaches and compare the results. The proposed solution is as follows:

- We implement the penalty shot multi-agent environment.
- We create two instances of the environment, first with the discrete action space and second using continuous action spaces.
- For discrete space setting, we apply **DQN and double DQN (DDQN)** approaches and check the performance.
- For continuous space setting, we use **PPO, DDPG, and A2C** approaches and review the results.

Experiments and Results

Experiments

- Our experiments were mainly divided into 2 categories based on the type of action space. 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.

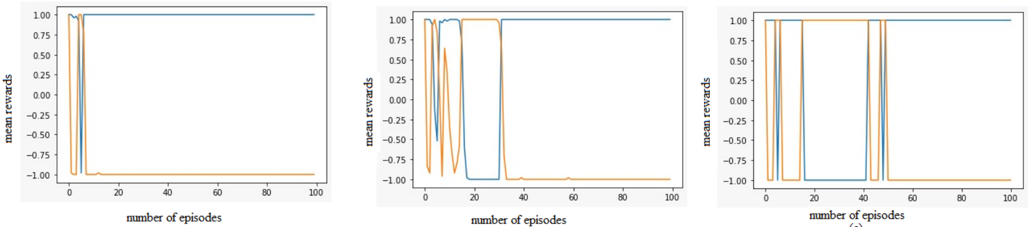
Results

- 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

Results

- 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.

Results



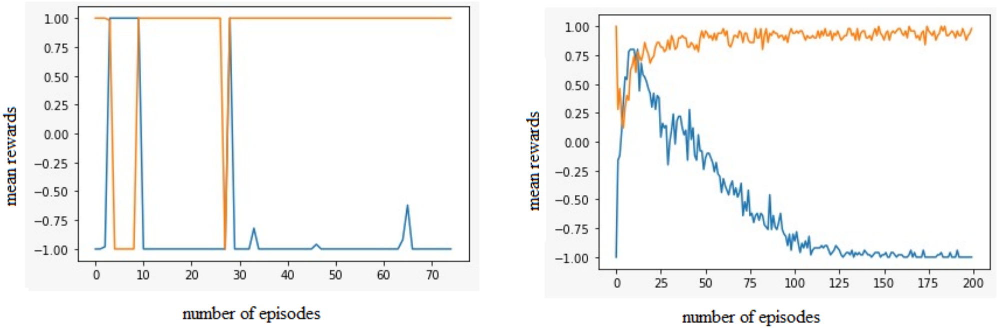
(a)

(b)

(c)

Figure: 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

Results



(a)

(b)

Figure: 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

Future Directions and Conclusions

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 Directions

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.

Contribution

Table: 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

CANESE, L., CARDARILLI, G. C., DI NUNZIO, L., FAZZOLARI, R., GIARDINO, D., RE, M., AND SPANÒ, S.
Multi-agent reinforcement learning: A review of challenges and applications.
Applied Sciences 11, 11 (2021), 4948.

LI, S., WU, Y., CUI, X., DONG, H., FANG, F., AND RUSSELL, S.
Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient.
In *Proceedings of the AAAI Conference on Artificial Intelligence* (2019), vol. 33, pp. 4213–4220.

NGUYEN, T. T., NGUYEN, N. D., AND NAHAVANDI, S.
Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications.
IEEE transactions on cybernetics 50, 9 (2020), 3826–3839.

SCHULMAN, J., WOLSKI, F., DHARIWAL, P., RADFORD, A., AND KLIMOV, O.
Proximal policy optimization algorithms.

arXiv preprint arXiv:1707.06347 (2017).

Thank you