Multiple AI Competition in Self Developed Game

Term 2 Report

ESTR4998/4999

April 13, 2021

Partners:	XIAO Tianyi
	LUO Lu
Instructor:	Prof. Andrej Bogdanov

Abstract

(some abstract)

1 Background

1.1 Previous Progress

Write something.

2

2 Implementation

2.1 Single Player Mode

In the beginning stage of our project in this term, we found that it's hard to train AI to play our game with DQN. So to simplify the issue and solve our prolem better, we first implement a single player mode.

Then with the experience from single player mode, we are finally able to train DQN AI that could play 1v1 mode with logic.

In this mode, we randomize the initialization of our player and ball. They will be put on the soccer field randomly when each episode begins.

```python
def reset_random(ball, player):
    ball.rect.centerx = screen_rect.centerx + (np_random.rand() - 0.5) * conf.width * 0.65
    ball.rect.centery = screen_rect.centery + (np_random.rand() - 0.5) * conf.height * 0.65
    ball.if_caught = False
    ball.catcher = -1
    ball.v.x = 0
    ball.v.y = 0
    player.rect.centerx = screen_rect.centerx + (np_random.rand() - 0.5) * conf.width * 0.65
    player.rect.centery = screen_rect.centery + (np_random.rand() - 0.5) * conf.height * 0.65
    player.v.x = 0
    player.v.y = 0
```

Figure 1: implementation of reset_random

2.2 Agent State

The state of agent represent the features sent to neural network. We add one more item into the agent state, which shows if the player is catching the ball or not. In the soccer game, there is cool down time for each player to prevent it to get the ball back as soon as it just shoots the ball. Therefore the coincidence of positions of player and ball doesn't mean that player could catch and shoot the ball. So one more item in agent state could help AI better understand its situation.

2.3 Reward Function

Reward function is crucial in reinforcement learning. As a soccer game, when a team get a goal, it will get huge reward. Besides, when a player catch a ball, it will also recive reward. Besides, to encourage player to keep the ball longer, once the player shoot the ball, it will get punishment.
However, only these basic rewards are not enough for training. Below are our modification for other part of reward function.

2.3.1 Original Reward Function

In our original reward function, we give reward to the agent if it's closer to the ball than other player. Otherwise, the agent would get punishment. Besides, if the ball get closer to one door, the team attacking this door would get reward, and other team would get punishment.

Algorithm 1 Original Reward($Agent1, Agent2, Ball$)

Closer_Agent(Agent1, Agent2, Ball).reward $+=$ 200
Further_Agent(Agent1, Agent2, Ball).reward $-=$ 200
if Ball Moves right **then**
    Agent1.reward $+=$ 600
    Agent2.reward $-=$ 600
if Ball Moves left **then**
    Agent2.reward $+=$ 600
    Agent1.reward $-=$ 600
if Ball goes into right door **then**
    Agent1.reward $+=$ 100000
    Agent2.reward $-=$ 10000
if Ball goes into left door **then**
    Agent2.reward $+=$ 100000
    Agent1.reward $-=$ 10000
if Agent shoot the ball **then**
    Agent-shoot-ball.reward $-=$ 1000

2.3.2 Reward Function For Single Mode

With Single Player Mode, we need a new reward function. This one is simple. If the player is moving closer to the ball, it will get reward, otherwise it will get punishment. Also, if the player shoot the ball to the direction to the right door, it will get reward. But if it shoots to wrong direction, it will get punishment. Then to encourage the player to keep the ball, it will also have reward when it keeps the ball.

Algorithm 2 Single Mode Reward($Agent$, $Ball$)

if Closer(Agent, Ball) or Keep(Agent, Ball) **then**
    Agent.reward $+=$ 200
else
    Agent.reward $-=$ 200
if Ball goes into right door **then**
    Agent.reward $+=$ 100000
if Ball goes into left door **then**
    Agent.reward $-=$ 10000
if Agent shoot the ball **then**
    team-of-the-agent.reward $-=$ 1000

2.3.3 New Reward Function

Then, with experience from single player mode, we modify our original reward function. We cancel the comparison of distance to ball between two agents, instead we only care if the agent is closer to the ball. Also, we cancel some punishment to avoid agent playing to negatively. And we adjust the amount of reward during our test.

Algorithm 3 Original Reward($Agent1, Agent2, Ball$)

```
if Closer(Agent1, Ball) then
    Agent1.reward + = 100
else
    Agent1.reward − = 100
if Closer(Agent2, Ball) then
    Agent2.reward + = 100
else
    Agent2.reward − = 100
if Ball Moves right then
    Agent1.reward + = 100
if Ball Moves left then
    Agent2.reward + = 100
if Ball goes into right door then
    Agent1.reward + = 100000
    Agent2.reward − = 10000
if Ball goes into left door then
    Agent2.reward + = 100000
    Agent1.reward − = 10000
if Agent shoot the ball then
    Agent-shoot-ball.reward − = 1000
```

2.4 Neural Network

2.5 Measurement of Performance

3 Result

3.1 Single Player Mode

3.2 1v1 Mode

4 Discussion

[2] Mnih, Volodymyr, et al. ”Playing atari with deep reinforcement learning.” arXiv preprint arXiv:1312.5602 (2013).

[3] Berner, Christopher, et al. ”Dota 2 with large scale deep reinforcement learning.” arXiv preprint arXiv:1912.06680 (2019).

[4] Zoph, Barret, and Quoc V. Le. ”Neural architecture search with reinforcement learning.” arXiv preprint arXiv:1611.01578 (2016).

[5] Mnih, Volodymyr, et al. ”Human-level control through deep reinforcement learning.” nature 518.7540 (2015): 529-533.