Atari Breakout Reinforcement Learning Environment

Haider Sajjad haider.sajjad@mail.utoronto.ca
1004076251

Weiyu Li weiyu.li@mail.utoronto.ca
1003765981

Abstract

Atari Breakout environment implementation and training an agent using multiple algorithms over a generated environment (changing brick layouts)

1. Introduction

Our project is implementing an Atari breakout environment and training an agent to play it across general levels. The project repository can be found here: `https://github.com/duoduocai-dot/csc498-project` To play the original game, after cloning the project, just run this file: `https://github.com/duoduocai-dot/csc498-project/blob/main/run_Breakout.py`.
Our environment is here. We made a made our environment embedded into a pygame class for Breakout, adding environment functions and variables.
We made 6 algorithms which play Breakout, and compare their performance in this report. The algorithms we made are: DQN, double-DQN, policy gradient REINFORCE, tabular Q-Learning, tabular double Q-Learning, and tabular SARSA.

2. Environment

Our environment is here: `https://github.com/duoduocai-dot/csc498-project/blob/main/breakout.py`.

2.1 Rewards

We experimented with various reward mechanisms. Our original reward schema was:

- +10 for ball hitting paddle
- -10 ball missing paddle
- +1 ball hits brick
- -0.2 movement penalty
- +1000 ball destroys all bricks

However, with this model we noticed it was difficult for the agent to actually learn, as it doesn't get direct feedback if making an action is good or not. So we altered the reward schema to be:

- +10 for ball hitting paddle
- -10 ball missing paddle
- Distance between paddle and ball in the x-Axis `https://github.com/duoduocai-dot/csc498-project/blob/main/breakout.py#L179`
- +1000 ball destroys all bricks

2.2 Environment variables, functions

Regarding the rest of the environment, the step function is here, where the paddle moves depending on the action passed in, and updates rewards for that step. It then returns the game state which is [paddle x-location, ball x-location, ball y-location,ball x-speed, ball y-speed, bricks left].
The reset function resets the environment, setting ball, bricks, paddle to their original position.
Render functio, renders the game. Make function creates the brick layout, and main function allows the user to play the game.
Agent actions are to move left (0), move right (1), stay still (2).
There is more about the environment including the brickLayout which we used in testing and comparing the tabular algorithms which I will talk about in that section.

3. Algorithims

3.1 Tabular Algorithms

3.1.1 Tabular Q-Learning

3.1.2 Tabular Double Q-Learning

3.1.3 Tabular Sarsa

3.1.4 Comparing tabular algorithms

3.2 Policy Gradient