Collaboration and Competition

September 6, 2024

By Haya Alhuraib

1. Overview

In this environment, two agents control rackets to bounce a ball over a net. If an agent hits the ball over the net, it receives a reward of +0.1. If an agent lets a ball hit the ground or hits the ball out of bounds, it receives a reward of -0.01. Thus, the goal of each agent is to keep the ball in play. The observation space consists of 8 variables corresponding to the position and velocity of the ball and racket. Each agent receives its own, local observation. Two continuous actions are available, corresponding to movement toward (or away from) the net, and jumping. The task is episodic, and in order to solve the environment, your agents must get an average score of +0.5 (over 100 consecutive episodes, after taking the maximum over both agents).

2. Implementation

The Reacher environment was solved using a deep reinforcement learning agent, with the implementation available in the collab_and_comp directory. The agent.py file contains the RL agent, while model.py contains the neural networks used as estimators. Some code was adapted from the Udacity DDPG-pendulum exercise to suit this problem.

3. Learning Algorithm

The Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm was employed as the learning algorithm for the agent. This actor-critic approach is derived from DDPG and is capable of solving tasks with multiple agents. MADDPG utilizes four neural networks: a local actor, a target actor, a local critic, and a target critic. During each training step, the experience (state, action, action_other_agent, reward, next state, next_state_other_agent) gained by the two agents was stored. The agent then learned from a random sample from the stored experience. The actor estimates the optimal policy using the estimated state-action values from the critic, while the critic estimates the optimal q-value function using a normal q-learning approach. This approach combines the benefits of value-based and policy-based methods.

4. **Hyperparameters**

The following hyperparameters were used:

- Replay buffer size: 1e5
- Max timesteps: 10000
- Minibatch size: 128
- Discount factor: 0.99

- Tau (soft update for target networks factor): 1e-3
- Learning rate: 1e-4 (actor) and 1e-3 (critic)
- Update interval: 1
- Beta start (factor for the noise added to the actions selected by the actor): 1.0
- Beta decay factor: 0.995
- Min beta: 0.01

5. **Neural Networks**

The actor model is a simple feedforward network with batch normalization, consisting of:

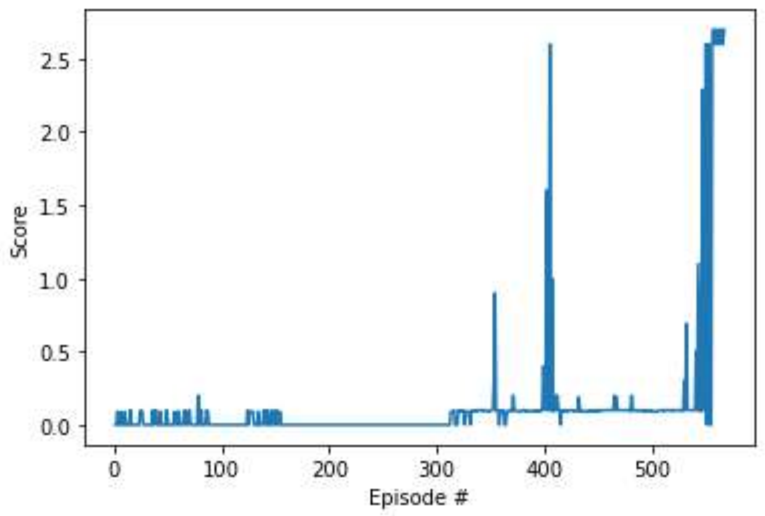
- Input layer: 24 neurons (state size)
- 1st hidden layer: 128 neurons (leaky ReLU)
- 2nd hidden layer: 128 neurons (leaky ReLU)
- Output layer: 2 neurons (1 for each action) (tanh)

The critic model also uses batch normalization and consists of:

- Input layer: 24 neurons (state size)
- 1st hidden layer: 132 neurons (action with 2 * action_size 2 added) (leaky ReLU)
- 2nd hidden layer: 128 neurons (leaky ReLU)
- Output layer: 1 neuron

6. **Results**

The agent successfully solved the environment after 467 episodes, achieving an average maximum score of 0.51 over the last 100 episodes of the training process. The average maximum scores of the 2 agents during the training process are shown in the figure below:



7. **Future Improvements**

The algorithm can be improved in various ways. For example, implementing DQN improvements such as Prioritized Experience Replays could enhance the learning effect gained from the saved experience. Additionally, optimizing the hyperparameters or changing the neural network architectures could lead to further improvements.