Environment

In this environment, two agents (players) 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.

Solution

The task is episodic, and in order to solve the environment, the agents must get an average score of +0*:*5 (over 100 consecutive episodes, after taking the maximum over both agents).

the environment can be solved with the *DDPG* algorithm after some obvious modifications.

Deep Deterministic Policy Gradient (DDPG)

Deep Deterministic Policy Gradient, is a modelfree off-policy actor-critic algorithm, combining DPG with DQN (Deep QNetwork) stabilizes the learning of the Q-function by using experience replay and a frozen target network.

The actor is used to approximate the optimal policy deterministically.
That means we always want to output the best believed action for any given state.
This is unlike a stochastic policy which learn a probability distribution over the actions.

In DDPG, we want the believed best action every single time we query the actor network, that is a deterministic policy.

The actor is basically learning *argmaxa*
Q(*s; a*) which is the best action

The critic learns to evaluate the optimal action-value function by using the actor's best believed action.
In DDPG, not only we have a regular network for the actor as well for the critic, we do also have a copy of these regular networks.

A major challenge of learning in continuous action spaces is exploration.

- Policy-based methods have(critic) high variance
- value-based methods(actor) have low variance

combine both actor and critic to get a Actor-critic methods where Actor represented as a neural network that update the policy by choice best action to maxmamize the reward and the critic another network that evaluate the policy being learned that been used to train the actor

Hyperparameters

There were many hyperparameters involved in the experiment.

```
# training hyperparameters
lr_actor = 1e-3                    # learning rate of Actor
lr_critic = 1e-3                   # learning rate of Critic
weight_decay = 0.                  # L2 weight decay
gamma = 0.99                       # discount factor
tau = 1e-3                        # soft update parameter
batch_size = 1024                   # batch size to sample from replay buffer
buffer_size = int(1e5)             # max size (capacity) of the replay buffer
n_agents = 2                       # number of agents
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
update_every = 20
update_freq = 10
```

Actor network is FNN with input of 24 and output of 2 , the rest layers (heddin layers are : 512 units , 256 units , 128 units (we use tanh for the output layer to represent values -1,1)
Critic network is FNN with input of 24 and output of 1 , the rest layers (heddin layers are : 512 units , 256 units , 128 units.

idea for improvment

try to use The ***MADDPG (Multi-Agent DDPG)***