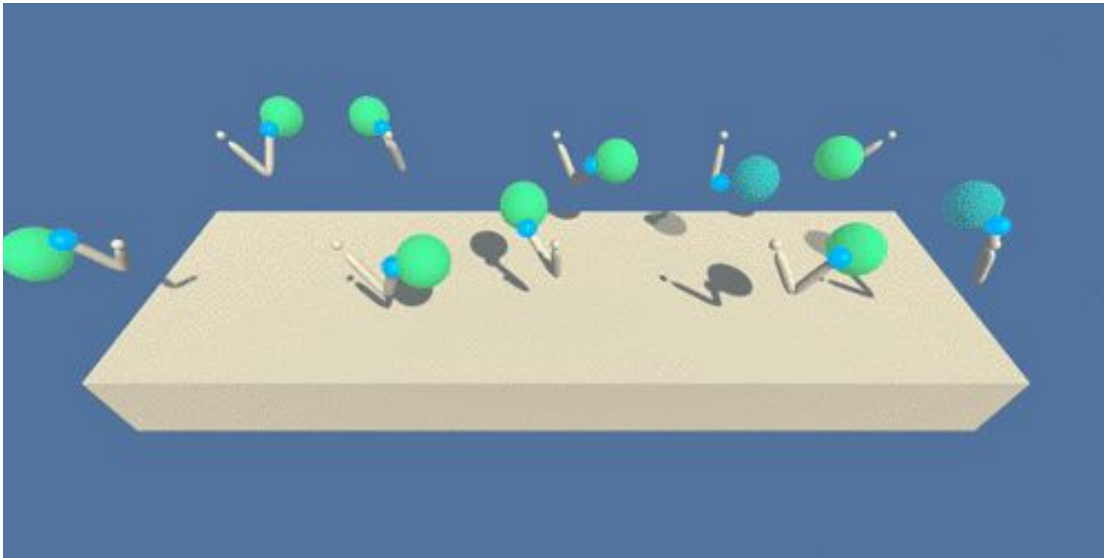


Continuous Control

Kellin Bershinsky



Introduction

This project is intended to provide an opportunity for students to gain more experience using Unity's ML-Agents machine learning toolset and apply their knowledge of Asynchronous Advantage Actor-Critic (A3C), Deep Deterministic Policy Gradient (DDPG), or Proximal Policy Optimization (PPO) algorithms to train a Deep Reinforcement Learning (DRL) agent to perform a task. The task is to train a double-jointed arm to track a target location as it orbits the arm's anchor point. The defined task environment includes 33 variables which include position, rotation, velocity, and angular velocities of the arm. The action performed by the agent is a four element vector corresponding to the torque at each of the two joints. Each element in the vector is a number that ranges from -1 to +1. A reward of +0.1 is given to the agent for every step the agent's hand is touching the target area. There are two versions of the environment which include a single agent or 20 agent distributed training scenario. The task is considered solved when the agent yields an average score of +30 over 100 consecutive episodes.

Learning Algorithm Used - DDPG

DDPG is a type of actor-critic method which can be seen as an approximate Deep Q-Network (DQN). The critic in DDPG is used to predict the maximum Q-value for the next state similar to the DQN algorithms estimate function. The main advantage that DDPG has over DQN is that it can be used in a continuous state and action space.

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ .
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
 Initialize a random process \mathcal{N} for action exploration
 Receive initial observation state s_1
 for t = 1, T **do**
 Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
 Execute action a_t and observe reward r_t and observe new state s_{t+1}
 Store transition (s_t, a_t, r_t, s_{t+1}) in R
 Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
 Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'}))|\theta^{Q'}$
 Update critic by minimizing the loss: $L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
 Update the actor policy using the sampled policy gradient:
$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

 Update the target networks:
$$\theta^{Q'} \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau \theta^\mu + (1 - \tau) \theta^{\mu'}$$

 end for
end for

Figure 1: DDPG Algorithm¹

Improvements Added:

A mean neural network was placed between the first and second hidden layers of the neural network.

¹ "Continuous control with deep reinforcement learning."
<https://arxiv.org/abs/1509.02971>. Accessed 24 Jul. 2019.

Hyperparameters Used

`BUFFER_SIZE = int(1e6)` # replay buffer size

`BATCH_SIZE = 1024` # minibatch size

`GAMMA = 0.99` # discount factor

`TAU = 1e-3` # for soft update of target parameters

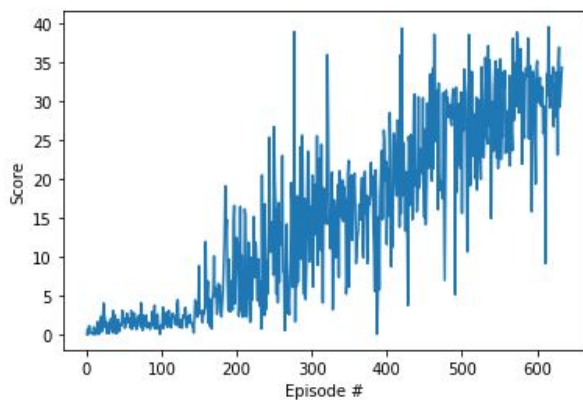
`LR_ACTOR = 2e-4` # learning rate of the actor

`LR_CRITIC = 3e-4` # learning rate of the critic

`WEIGHT_DECAY = 0` # L2 weight decay

Results - Plot of Rewards

Environment Solved! It took 633 episodes!
Average Score: 30.01



Episode 100	Average Score: 1.32
Episode 200	Average Score: 4.05
Episode 300	Average Score: 10.72
Episode 400	Average Score: 15.92
Episode 500	Average Score: 23.09
Episode 600	Average Score: 28.62
Episode 633	Average Score: 30.01

Environment Solved! It took 633 episodes!
Average Score: 30.01

The figure above shows the agents average score throughout the training process which included 633 episodes. The table to the right of the plot shows the average score over 100 hundred episodes for every 100 episodes throughout the training session. Based on these results, it appears the environment was solved in 633 episodes.

Ideas for Future Work

Trust Region Policy Optimization (TRPO)

TRPO is similar to natural policy gradient methods and is effective for optimizing large nonlinear policies such as neural networks.²

Algorithm 1 Policy iteration algorithm guaranteeing non-decreasing expected return η

Initialize π_0 .

for $i = 0, 1, 2, \dots$ until convergence **do**

 Compute all advantage values $A_{\pi_i}(s, a)$.

 Solve the constrained optimization problem

$$\pi_{i+1} = \arg \max_{\pi} [L_{\pi_i}(\pi) - CD_{\text{KL}}^{\max}(\pi_i, \pi)]$$

$$\text{where } C = 4\epsilon\gamma/(1 - \gamma)^2$$

$$\text{and } L_{\pi_i}(\pi) = \eta(\pi_i) + \sum_s \rho_{\pi_i}(s) \sum_a \pi(a|s) A_{\pi_i}(s, a)$$

end for

Figure 2: TRPO Algorithm³

Truncated Natural Policy Gradient (TNPG)

By computing an ascent direction that approximately ensures a small change in the policy distribution TNPG improves upon the REINFORCE algorithm.⁴

Distributed Distributional Deterministic Policy Gradients (D4PG)

D4DG adapts distributional reinforcement learning to the continuous control setting and combines it with distributed off-policy learning combined with improvement including N-step returns and prioritized experience replay.⁵

² "Trust Region Policy Optimization." <https://arxiv.org/abs/1502.05477>. Accessed 24 Jul. 2019.

³ "Trust Region Policy Optimization." <https://arxiv.org/abs/1502.05477>. Accessed 24 Jul. 2019.

⁴ "Benchmarking Deep Reinforcement Learning for Continuous Control." 22 Apr. 2016, <https://arxiv.org/abs/1604.06778>. Accessed 24 Jul. 2019.

⁵ "distributed distributional deterministic policy gradients - OpenReview." <https://openreview.net/pdf?id=SyZipzbCb>. Accessed 24 Jul. 2019.

Works Cited

Barth-Maroon, Gabriel, et al. "Distributed Distributional Deterministic Policy Gradients."

Venues, 15 Feb. 2018, openreview.net/forum?id=SyZipzbCb.

Duan, et al. "Benchmarking Deep Reinforcement Learning for Continuous Control."

ArXiv.org, 27 May 2016, arxiv.org/abs/1604.06778.

P., Timothy, et al. "Continuous Control with Deep Reinforcement Learning." *ArXiv.org*, 5 July

2019, arxiv.org/abs/1509.02971.

Schulman, et al. "Trust Region Policy Optimization." *ArXiv.org*, 20 Apr. 2017,

arxiv.org/abs/1502.05477.