Project 2: Continuous Control

April 25, 2021

Contents

1 Project Overview

In this project, we implement the Deep Deterministic Policy Gradient reinforcement learning technique of Lillicrap et al (2015) to guide the motions of a double-jointed robotic arm in an adaptation of OpenAI Gym's Reacher Environment to the targeted destinations. The observation space consists of 33 variables corresponding to position, rotation, velocity, and angular momentum of the arm. The actions are represented by 3 number corresponding to torque in the two joints between 1 and -1. The task is episodic in nature. A reward of +0.1 is provided for each step that the agent's hand is in the target position, so our goal is to keep the agent's hand in the target location as much as possible. We consider the environment to be solved when the agent is able to achieve a score of +30 over 100 consecutive episodes.

2 Learning Algorithm

2.1 Network Architecture

2.1.1 Actor

- **Input Layer:** 33 dimensional layer
- **Hidden Layer 1**: 256 dimensional with relu activation
- **Hidden Layer 2:** 128 dimensional with relu activation
- **Output Layer:** 4 dimensional, corresponding to the 4 actions with tanh activation

2.1.2 Critic

- **Input Layer:** 33 dimensional layer
- **Hidden Layer 1:** 256 dimensional with relu activation
- **Hidden Layer 2:** 128 dimensional with relu activation
- **Output Layer:** 1 dimensional, corresponding to the 1 core with no activation, as the output is predicting the score

2.2 Parameters

- **BUFFER_SIZE** R= int(1e6); Number of episodes to retain in replay buffer.
- **BATCH_SIZE** N = 128; minibatch size
- **GAMMA** γ= 0.99; inter-temporal discount factor for future rewards
- **TAU** τ= 1e-3 ; for soft update of target parameters
- **LR_ACTOR** α_μ= 1e-4 ; Rate at which the actor learns during the gradient descent step.
- **LR_CRITIC** α_Q= 3e-4 ; Rate at which the critic learns during the gradient descent step.
- **WEIGHT_DECAY** = 0; L2 weight decay
- **M**=1,000: Maximum number of episodes
- **T**=1,000: Maximum length of any single episode

2.3 Algorithm

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q\left(s,a|\theta^{Q}\right)$ and actor $\mu\left(s|\theta^{\mu}\right)$ 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 of size D
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 $a_t = \mu\left(s_t|\theta^{\mu}\right) + \mathcal{N}_t$ according to the current policy and exploration noise
    - The noise in this case is sample from an Ornstein-Uhlenbeck process (Uhlenbeck & Ornstein, 1930) in order to induce further exploration.
  - 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 $y_i = r_i + \gamma Q'\left(s_{i+1}|\mu'\left(s_{i+1}|\theta^{\mu'}\right)|\theta^{Q'}\right)$
    - This is the famous Bellman error term. There are 3 steps implicit in this representation:
      1. Use the actor network to choose action a_i in state s_i and receive reward r_i.
      2. Use the critic network to evaluate the value of taking action a_i in state s_i.
      3. These $y_i's$ can be thought of as expected values of the critic's action-value function
  - Update critic by gradient descent with learning rate α_Q by minimizing the loss:

$$\mathcal{L} = \frac{1}{N}\Sigma_i\left(y_i - Q\left(s_i, a_i|\theta^{Q}\right)\right)^2$$

    - In this step we update the weights in our critic network to best match the $(y_i's)$ in the last step.
  - Update actor policy with learning rate $\alpha_\mu$using the sampled policy gradient

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\Sigma_i \nabla_a Q\left(s,a|\theta^{Q}\right)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu\left(s|\theta^{\mu}\right)|_{s_i}$$

    - In this step, essentially we observe all of the episodes N in our minibath sampled from the replay buffer R, and simply update the weights in the network to better align the state,action pairs with the ultimate reward of the game.
    - This step is similar to a supervised learning problem where we have "images" (s_i, a_i), predicted rewards or "labels" $Q\left(s_i, a_i|\theta^{Q}\right)$ and actual labels for the reward r_i received at the end of episide i.
  - Perform a soft update of the target networks:

$$\begin{aligned}\theta^{Q'} &\leftarrow \tau\theta^{Q} + (1-\tau)\,\theta^{Q'}\\ \theta^{\mu'} &\leftarrow \tau\theta^{\mu} + (1-\tau)\,\theta^{\mu'}\end{aligned}$$

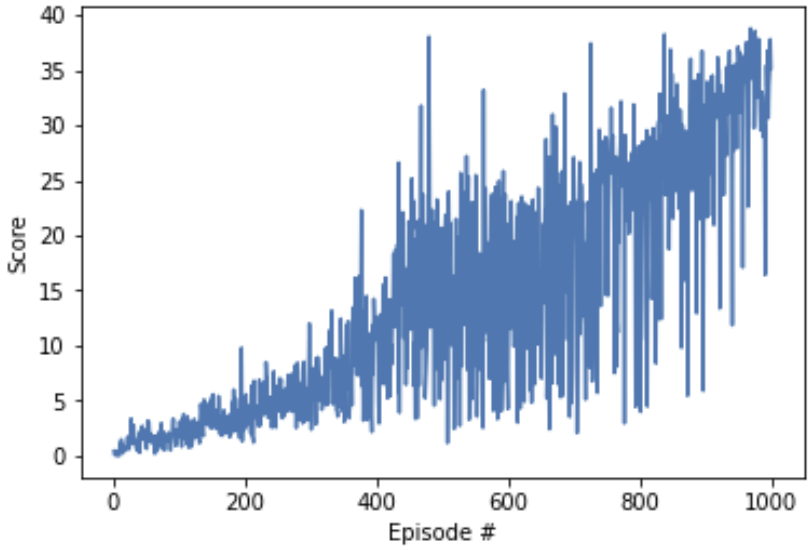
- **end for**

end for

3 Learning Curve

The aforementioned algorithm was able to solve the environment in fewer than 1,000 episodes.

Episode	Average Score
100	1.39
200	2.99
300	5.05
400	8.04
500	14.28
600	14.82
700	16.06
800	21.39
900	25.26
1,000	31.74



4 Future Work and Discussion

Duan et. al (2016) examine a range of reinforcement learning techniques in the context of simple tasks - such as the cart-pole balancing problem, locomotion tasks, partially observable tasks, and hierarchical tasks. While DDPG's performance was good overall, it is sensitive to the scaling of the reward, so in future iterations a re-scaled reward could lead to faster training time. The Trust Region Policy Optimization (Schulman et al., 2015) outperformed the DDPG on nearly all tasks, as it appears to provide a more precise and stable estimate of the expected policy improvement so utilizing this algorithm would likely lead to more efficient convergence. Efficiency is always subject to choice of hyperparameter; there is always a trade-off between exploration and exploitation, so finding a better heuristic, or perhaps performing a grid search to choose the parameters for the noise could hep us to better manage this trade-off. Perhaps it would be prudent to allow the variance of the process to decay with time, as exploration is more important early in training than late and large amounts of noise could sabotage an otherwise productive learning path. Another potential area of improvement would be the utilization of multiple agents with a shared replay buffer.

Bibliography

Duan, Yan, et al. "Benchmarking deep reinforcement learning for continuous control." International conference on machine learning. PMLR, 2016.

Lillicrap, Timothy P., et al. "Continuous control with deep reinforcement learning." arXiv preprint arXiv:1509.02971 (2015).

Schulman, John, et al. "Trust region policy optimization." International conference on machine learning. PMLR, 2015.

Uhlenbeck, George E and Ornstein, Leonard S. On the theory of the brownian motion. Physical review, 36(5):823, 1930.