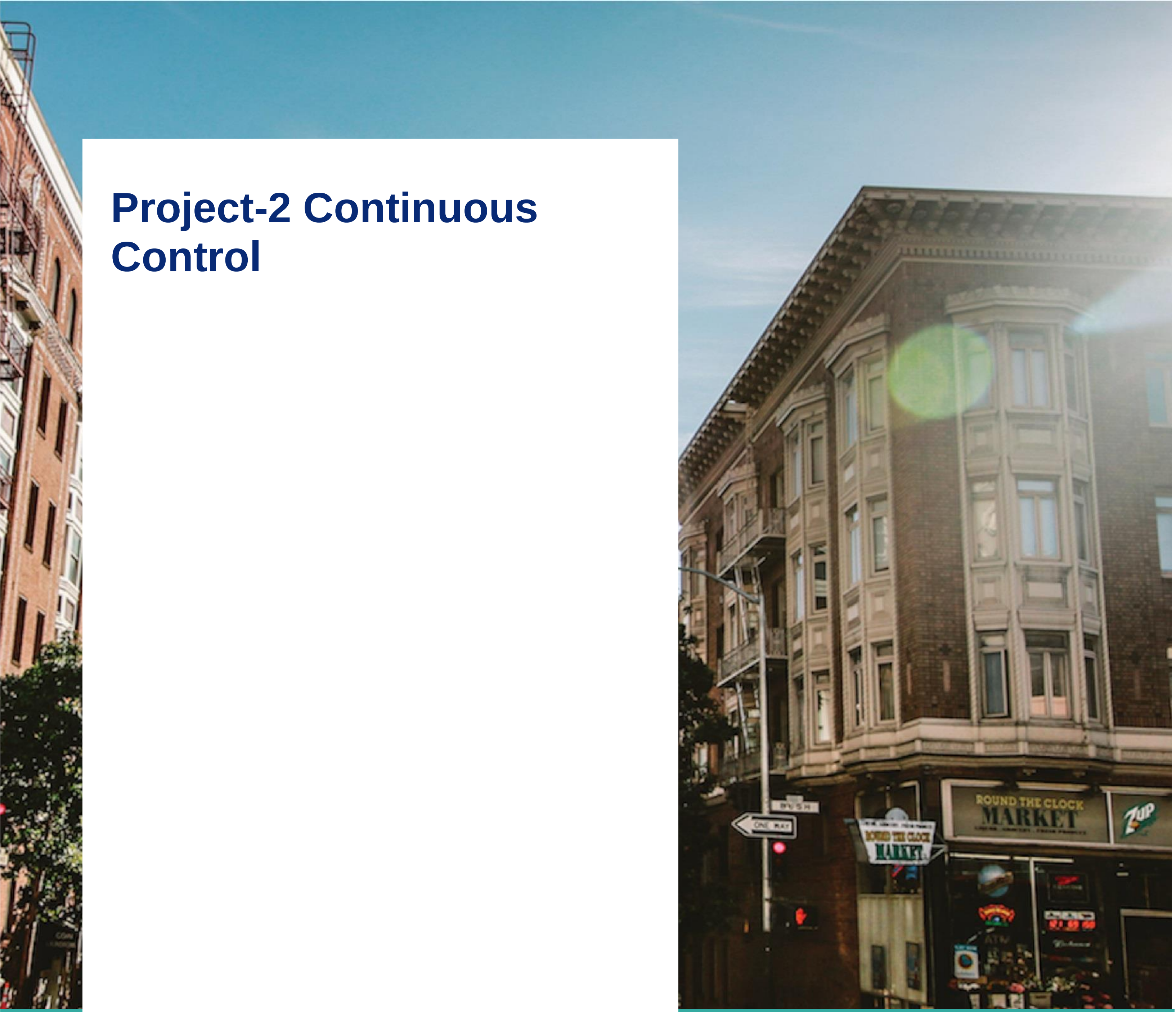


Project-2 Continuous Control

23RD MAR 2020

AMRIT PAUL

Problem Statement

The goal of the project is to demonstrate the abilities of a model-free reinforcement learning algorithm, particularly Deep Deterministic Policy Gradients (DDPG) Algorithm, which consists of two neural networks namely Actor network and Critic network. The project uses Unity environment, a game development framework and Pytorch, a deep learning framework. The algorithm was trained on 20 agents, with each agent consisting of 33 states and 4 actions.

Description

Deep Deterministic Policy Gradient (DDPG) is an algorithm which concurrently learns a Q-function and a policy. It uses off-policy data and the Bellman equation to learn the Q-function and uses the Q-function to learn the policy.

The approach is closely connected to Q-learning and is motivated the same way: if you know the optimal action-value function Q*(s,a), then in any given state, the optimal action a*(s) can be found by solving :-

$$a^*(s) = \arg\max_a Q^*(s,a).$$

DDPG interleaves learning an approximator to Q*(s,a) with learning an approximator to a*(s) and it does so in a way which is specifically adapter for environments with continuous action spaces. The actor produces a deterministic policy instead of the usual stochastic policy and the critic evaluates the deterministic policy. The critic is updated using the TD-error and the actor is trained using the deterministic policy gradient algorithm.

$$\begin{aligned}\nabla_{\theta^\mu} J &\approx \mathbb{E}_{s_t \sim \rho^\beta}\left[\nabla_{\theta^\mu} Q(s,a|\theta^Q)|_{s=s_t,a=\mu(s_t|\theta^\mu)}\right]\\ &= \mathbb{E}_{s_t \sim \rho^\beta}\left[\nabla_a Q(s,a|\theta^Q)|_{s=s_t,a=\mu(s_t)} \nabla_{\theta_\mu}\mu(s|\theta^\mu)|_{s=s_t}\right]\end{aligned}$$

Pseudocode

Algorithm 1 Deep Deterministic Policy Gradient

Input: initial policy parameters θ, Q-function parameters ϕ, empty replay buffer $\mathcal{D}$

Set target parameters equal to main parameters $\theta_{\text{targ}} \leftarrow \theta$, $\phi_{\text{targ}} \leftarrow \phi$

repeat

  Observe state s and select action $a = \text{clip}(\mu_\theta(s) + \epsilon, a_{Low}, a_{High})$, where $\epsilon \sim \mathcal{N}$

  Execute a in the environment

  Observe next state s', reward r, and done signal d to indicate whether s' is terminal

  Store (s, a, r, s', d) in replay buffer $\mathcal{D}$

  If s' is terminal, reset environment state.

  if it's time to update **then**

    for however many updates **do**

      Randomly sample a batch of transitions, $B = \{(s, a, r, s', d)\}$ from $\mathcal{D}$

      Compute targets

$$y(r, s', d) = r + \gamma(1 - d) Q_{\phi_{\text{targ}}}(s', \mu_{\theta_{\text{targ}}}(s'))$$

      Update Q-function by one step of gradient descent using

$$\nabla_\phi \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} \left(Q_\phi(s, a) - y(r, s', d)\right)^2$$

      Update policy by one step of gradient ascent using

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} Q_\phi(s, \mu_\theta(s))$$

      Update target networks with

$$\begin{aligned} \phi_{\text{targ}} &\leftarrow \rho \phi_{\text{targ}} + (1 - \rho)\phi \\ \theta_{\text{targ}} &\leftarrow \rho \theta_{\text{targ}} + (1 - \rho)\theta \end{aligned}$$

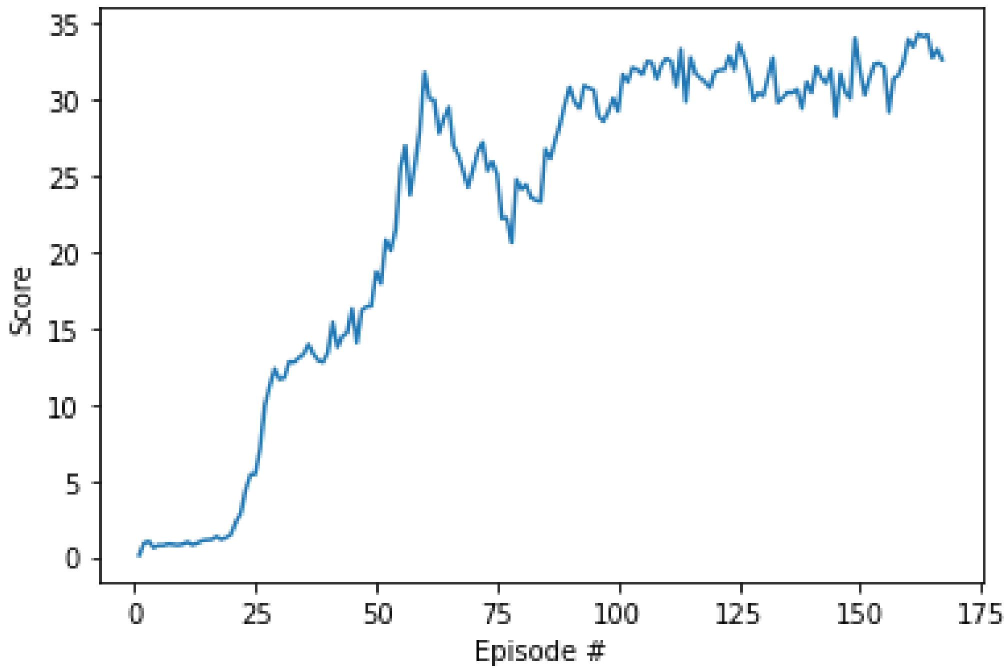
    end for

  end if

until convergence

Results

The following diagram shows the results after the DDPG algorithm is trained-



Episode: 10 Average Score: 0.75 Current Score: 0.84

Episode: 20 Average Score: 0.94 Current Score: 1.50

Episode: 30 Average Score: 3.05 Current Score: 11.66

Episode: 40 Average Score: 5.54 Current Score: 13.41

Episode: 50 Average Score: 7.56 Current Score: 18.70

Episode: 60 Average Score: 10.33 Current Score: 31.77

Episode: 70 Average Score: 12.77 Current Score: 25.41

Episode: 80 Average Score: 14.23 Current Score: 24.11

Episode: 90 Average Score: 15.58 Current Score: 30.80

Episode: 100 Average Score: 16.99 Current Score: 29.24

Episode: 110 Average Score: 20.12 Current Score: 32.69

Episode: 120 Average Score: 23.16 Current Score: 31.80

Episode: 130 Average Score: 25.61 Current Score: 30.23

Episode: 140 Average Score: 27.37 Current Score: 30.45

Episode: 150 Average Score: 28.95 Current Score: 31.97

Episode: 160 Average Score: 29.70 Current Score: 33.94

Episode: 167 Average Score: 30.05 Current Score: 32.61

Environment solved in 67 episodes!Average Score: 30.05

The hyperparameters used in this model are given below-

Hyperparameter	Value
Number of Actions	4
Number of States	33
Number of Episodes	2000
Max time steps per episode	1000
Replay Buffer Size	1e6
Batch Size	1024
Soft update of target parameters	1e-3
Gamma	0.99
Actor learning rate	1e-3
Critic learning rate	1e-3
Leaky ReLU leakiness	0.01
D4PGActivation	Leaky ReLU
Update after every step	10

Improvements

- Other algorithms such as TRPO, PPO, A3C, D4PG could potentially lead to better results
- Using prioritized experience replay can improve the performance of the model
- Using a combination of on-policy and off-policy algorithm could potentially lead to better results. One such algorithm is Q-prop
- Using convoluted neural network architectures with complex activation functions could potentially give better results!