ContinuousControl-D4PG

Train a double-jointed arm to reach target locations using D4PG continuous control Deep Reinforcement Learning. Watch this YouTube video to see how some researchers were able to train a similar task on a real robot! The accompanying research paper can be found here.

I`ve implement the in parallel D4PG algorithm to train the agent.

In the environment, there are 20 identical copies of the agent. It has been shown that having multiple copies of the same agent sharing experience can accelerate learning, as explained in this google AI blog post. You can use the second version to implement algorithms like PPO, A3C and D4PG that use multiple (non-interacting, parallel) copies of the same agent to distribute the task of gathering experience.

Rewards

A reward of +0.1 is provided for each step that the agent's hand is in the goal location. Thus, the goal of your agent is to maintain its position at the target location for as many time steps as possible.

States

The observation space consists of 33 variables corresponding to position, rotation, velocity, and angular velocities of the arm. Each action is a vector with four numbers, corresponding to torque applicable to two joints. Every entry in the action vector should be a number between -1 and 1.

Solving the environment

20 identical copies of the agent. The barrier for solving the environment takes into account the presence of many agents. In particular, your agents must get an average score of +30 (over 100 consecutive episodes, and over all agents). Specifically,

- After each episode, we add up the rewards that each agent received (without discounting), to get a score for each agent. This yields 20 (potentially different) scores. We then take the average of these 20 scores.
- This yields an average score for each episode (where the average is over all 20 agents).

The environment is considered solved, when the average (over 100 episodes) of those average scores is at least +30.

Requirements

First of all, you need python 3 and conda. We suggest to use the Anaconda distribution, although other options are available. The project has the following dependencies: pytorch, matplotlib, numpy, and collections. You need to install these dependencies on a conda environment and create a kernel to run this notebook. Follow the instructions of the udacity DRL nano degree at github repository drlnd to create an environment for the project, install dependencies (pytorch, matplotlib, numpy, and collections) and create a kernel. You will not need to install Unity, because Udacity provides the Unity environment:

- It contains 20 identical agents, each with its own copy of the environment.

Unity Machine Learning Agents (ML-Agents) is an open-source Unity plugin that enables games and simulations to serve as environments for training intelligent agents.

For game developers, these trained agents can be used for multiple purposes, including controlling NPC behavior (in a variety of settings such as multi-agent and adversarial), automated testing of game builds and evaluating different game design decisions pre-release.

Twenty (20) Agents

- Linux: click here
- Mac OSX: click here
- Windows (32-bit): click here
- Windows (64-bit): click here

Your own Unity environment

If you are interested in building your own Unity environments after completing the project, you can follow the instructions here, which walk you through all of the details of building an environment from a Unity scene.

For game developers, these trained agents can be used for multiple purposes, including controlling NPC behavior (in a variety of settings such as multi-agent and adversarial), automated testing of game builds and evaluating different game design decisions pre-release.

How it works

Open the file jupyter notebook D4PG.ipynb and execute it reading carefully the instructions. Notice that when creating an environemnt for the game we forced the option no_graphics=True. You can change it to False to see a graphic representation of the game. Please change path to saving files and eracher enviroment!!