Applying Deep Deterministic Policy Gradient (DDPG) to a Continuous Action Space Problem

aicompetence.org

August 4, 2024

1 Introduction

Deep Deterministic Policy Gradient (DDPG) is a model-free, off-policy actor-critic algorithm that is particularly well-suited for continuous action spaces. In this example, we demonstrate how to apply DDPG to control a pendulum using the `Pendulum-v1` environment from OpenAI Gym.

2 Environment Setup

The `Pendulum-v1` environment is a classic problem where the goal is to swing up a pendulum so that it stays upright. The state space includes the angle and angular velocity of the pendulum, while the action space consists of a continuous force applied to the pendulum.

3 DDPG Implementation

The DDPG algorithm is implemented using the `stable-baselines3` library, which provides a straightforward interface for applying reinforcement learning algorithms.

4 Python Code

The following code demonstrates how to set up and train a DDPG agent to solve the pendulum control problem:

Listing 1: DDPG applied to the Pendulum problem

```
import gym
from stable_baselines3 import DDPG
from stable_baselines3.common.noise import NormalActionNoise
import numpy as np
```

```
# Create the environment
env = gym.make('Pendulum-v1')

# Define the action noise (to encourage exploration)
n_actions = env.action_space.shape[-1]
action_noise = NormalActionNoise(mean=np.zeros(n_actions), sigma=0.1 * np.ones(n

# Create the DDPG model
model = DDPG("MlpPolicy", env, action_noise=action_noise, verbose=1)

# Train the agent
model.learn(total_timesteps=100000)

# Save the model
model.save("ddpg_pendulum")

# Load the model
model = DDPG.load("ddpg_pendulum")

# Test the trained agent
obs = env.reset()
for _ in range(1000):
    action, _states = model.predict(obs, deterministic=True)
    obs, reward, done, info = env.step(action)
    env.render()
    if done:
      obs = env.reset()

env.close()
```

5 Explanation

- **Environment Setup:** The environment is created using the `gym.make` function, which initializes the `Pendulum-v1` environment.
- **Action Noise:** To encourage exploration, Gaussian noise is added to the actions taken by the agent. This noise is parameterized by a mean and standard deviation.
- **Model Creation:** The DDPG model is created using the `MlpPolicy`, which defines a multi-layer perceptron policy. The model is trained by interacting with the environment for 100,000 timesteps.
- **Model Testing:** After training, the model is saved and then reloaded for testing. The agent interacts with the environment, and the pendulum's

behavior is visualized using the `env.render()` function.

6 Conclusion

This example demonstrates how DDPG can be effectively applied to a continuous action space problem. The pendulum problem serves as a fundamental benchmark, and the same approach can be extended to more complex environments with continuous actions.