Московский государственный технический университет им. Н.Э. Баумана
Кафедра «Системы обработки информации и управления»

Лабораторная работа №6
по дисциплине
«Методы машинного обучения»
на тему

«RL_PG»

Выполнил:
студент группы ИУ5и-24М
Аунг Пьио Нанда

Москва — 2024 г.

1. Цель лабораторной работы

Ознакомление с базовыми методами обучения с подкреплением на основе алгоритмов Actor-Critic.

2. Задание

Реализуйте любой алгоритм семейства Actor-Critic для произвольной среды.

Ход выполнения работы

Установка необходимых библиотек

```
[3] !pip install gym
    !pip install numpy
    !pip install torch
```

```
Requirement already satisfied: gym in /usr/local/lib/python3.10/dist-packages (0.25.2)
Requirement already satisfied: numpy>=1.18.0 in /usr/local/lib/python3.10/dist-packages (from gym) (1.25.2)
Requirement already satisfied: cloudpickle>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from gym) (2.2.1)
Requirement already satisfied: gym-notices>=0.0.4 in /usr/local/lib/python3.10/dist-packages (from gym) (0.0.8)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (1.25.2)
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (2.3.0+cu121)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch) (3.14.0)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.10/dist-packages (from torch) (4.12.0)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch) (1.12.1)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch) (3.3)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch) (3.1.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch) (2023.6.0)
Collecting nvidia-cuda-nvrtc-cu12==12.1.105 (from torch)
  Using cached nvidia_cuda_nvrtc_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (23.7 MB)
Collecting nvidia-cuda-runtime-cu12==12.1.105 (from torch)
  Using cached nvidia_cuda_runtime_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (823 kB)
Collecting nvidia-cuda-cupti-cu12==12.1.105 (from torch)
  Using cached nvidia_cuda_cupti_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (14.1 MB)
Collecting nvidia-cudnn-cu12==8.9.2.26 (from torch)
  Using cached nvidia_cudnn_cu12-8.9.2.26-py3-none-manylinux1_x86_64.whl (731.7 MB)
Collecting nvidia-cublas-cu12==12.1.3.1 (from torch)
  Using cached nvidia_cublas_cu12-12.1.3.1-py3-none-manylinux1_x86_64.whl (410.6 MB)
Collecting nvidia-cufft-cu12==11.0.2.54 (from torch)
  Using cached nvidia_cufft_cu12-11.0.2.54-py3-none-manylinux1_x86_64.whl (121.6 MB)
```

Реализация агента A2C

Импортируем необходимые модули

```
[4] import gym
    import numpy as np
    import torch
    import torch.nn as nn
    import torch.optim as optim
    from torch.distributions import Categorical
```

Определим нейронные сети для актора и критика

```python
class Actor(nn.Module):
    def __init__(self, input_dim, output_dim):
        super(Actor, self).__init__()
        self.fc1 = nn.Linear(input_dim, 128)
        self.fc2 = nn.Linear(128, output_dim)

    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = torch.softmax(self.fc2(x), dim=-1)
        return x

class Critic(nn.Module):
    def __init__(self, input_dim):
        super(Critic, self).__init__()
        self.fc1 = nn.Linear(input_dim, 128)
        self.fc2 = nn.Linear(128, 1)

    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

Создадим функцию для обучения агента

```python
def train(env, actor, critic, actor_optimizer, critic_optimizer, num_episodes=1000, gamma=0.99):
    for episode in range(num_episodes):
        state = env.reset()
        log_probs = []
        values = []
        rewards = []
        done = False

        while not done:
            state = torch.FloatTensor(state).unsqueeze(0)
            dist = actor(state)
            value = critic(state)

            dist = Categorical(dist)
            action = dist.sample()

            next_state, reward, done, _ = env.step(action.item())

            log_prob = dist.log_prob(action)
            log_probs.append(log_prob)
            values.append(value)
            rewards.append(reward)

            state = next_state
```

```
            if done:
                Qval = critic(torch.FloatTensor(next_state).unsqueeze(0)).item()
                returns = []
                R = Qval
                for reward in reversed(rewards):
                    R = reward + gamma * R
                    returns.insert(0, R)

                returns = torch.FloatTensor(returns)
                log_probs = torch.cat(log_probs)
                values = torch.cat(values)

                advantage = returns - values

                actor_loss = - (log_probs * advantage.detach()).mean()
                critic_loss = advantage.pow(2).mean()

                actor_optimizer.zero_grad()
                critic_optimizer.zero_grad()
                actor_loss.backward()
                critic_loss.backward()
                actor_optimizer.step()
                critic_optimizer.step()

                print(f'Episode {episode}, Actor Loss: {actor_loss.item()}, Critic Loss: {critic_loss.item()}')
                break
```

Создадим основную функцию для запуска обучения

```
[7] def main():
        env = gym.make('CartPole-v1')
        input_dim = env.observation_space.shape[0]
        output_dim = env.action_space.n

        actor = Actor(input_dim, output_dim)
        critic = Critic(input_dim)

        actor_optimizer = optim.Adam(actor.parameters(), lr=1e-3)
        critic_optimizer = optim.Adam(critic.parameters(), lr=1e-3)

        train(env, actor, critic, actor_optimizer, critic_optimizer)

    if __name__ == '__main__':
        main()
```

```
Episode 943, Actor Loss: 0.1003662645816803, Critic Loss: 1020974.9375
Episode 944, Actor Loss: 0.1389334797859192, Critic Loss: 1251826.75
Episode 945, Actor Loss: 0.1466035693883896, Critic Loss: 1240196.75
Episode 946, Actor Loss: 0.11594768613576889, Critic Loss: 845085.8125
Episode 947, Actor Loss: 0.09974848479032516, Critic Loss: 1264375.875
Episode 948, Actor Loss: 0.12682780623435974, Critic Loss: 1253931.125
Episode 949, Actor Loss: 0.1254538595676422, Critic Loss: 1241062.75
Episode 950, Actor Loss: 0.13267351686954498, Critic Loss: 1230978.375
Episode 951, Actor Loss: 0.10688477754592896, Critic Loss: 1077791.625
Episode 952, Actor Loss: 0.10608598589897156, Critic Loss: 1053389.25
Episode 953, Actor Loss: 0.1047956794500351, Critic Loss: 1098285.125
Episode 954, Actor Loss: 0.12186611443758011, Critic Loss: 1295439.875
Episode 955, Actor Loss: 0.09832371026277542, Critic Loss: 1307362.75
Episode 956, Actor Loss: 0.10923359543085098, Critic Loss: 1283497.375
Episode 957, Actor Loss: 0.1007431149482727, Critic Loss: 1110194.125
Episode 958, Actor Loss: 0.12648044526576996, Critic Loss: 1103838.125
Episode 959, Actor Loss: 0.12633495032787323, Critic Loss: 1302061.125
```

Актор (Actor):

Нейронная сеть, которая предсказывает вероятности действий. Мы используем torch.softmax для получения распределения вероятностей действий.

Критик (Critic):

Нейронная сеть, которая оценивает состояние, возвращая его ценность (value).

Обучение:

1) На каждом шаге эпизода агент выбирает действие на основе выходов актора.

2) Критик оценивает текущее состояние.

3) После выполнения действия агент получает награду и переходит в новое состояние.

4) По завершении эпизода вычисляются значения возврата (returns) для каждого шага.

5) Значение возврата используется для вычисления преимущества (advantage).

6) Обучение актера осуществляется с помощью функции потерь, основанной на преимуществах, а критика — на основе квадратов разностей между возвратами и оценками критика.

Список литературы

[1] Гапанюк Ю. Е. Лабораторная работа «Разведочный анализ данных. Исследование и визуализация данных» [Электронный ресурс] // GitHub. — 2024. — Режим доступа: https://github.com/ugapanyuk/courses_current/wiki/LAB_MMO___ RL_PG.