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

Лабораторная работа №7  
по дисциплине  
«Методы машинного обучения»  
на тему

# «Алгоритмы Actor-Critic»

Выполнил:  
студент группы ИУ5-23М  
Чжэн Сяохуэй

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.

**3. текст программы**

import gym

import torch

import torch.nn as nn

import torch.optim as optim

import numpy as np

import matplotlib.pyplot as plt

# Actor-Critic网络

class ActorCritic(nn.Module):

def \_\_init\_\_(self, input\_dim, action\_dim):

super(ActorCritic, self).\_\_init\_\_()

self.fc = nn.Linear(input\_dim, 128)

self.policy = nn.Linear(128, action\_dim)

self.value = nn.Linear(128, 1)

def forward(self, x):

x = torch.relu(self.fc(x))

policy\_dist = torch.softmax(self.policy(x), dim=-1)

value = self.value(x)

return policy\_dist, value

# Hyperparameters

env = gym.make('CartPole-v1')

input\_dim = env.observation\_space.shape[0]

action\_dim = env.action\_space.n

lr = 0.01

gamma = 0.99

episodes = 1000

# 初始化网络和优化器

model = ActorCritic(input\_dim, action\_dim)

optimizer = optim.Adam(model.parameters(), lr=lr)

# 训练过程

def train():

episode\_rewards = []

for episode in range(episodes):

state, info = env.reset()

state = torch.tensor(state, dtype=torch.float32).unsqueeze(0)

done = False

total\_reward = 0

while not done:

policy\_dist, value = model(state)

action = np.random.choice(action\_dim, p=policy\_dist.detach().numpy().flatten())

next\_state, reward, done, \_, \_ = env.step(action)

next\_state = torch.tensor(next\_state, dtype=torch.float32)

\_, next\_value = model(next\_state)

# 计算优势

advantage = reward + (1 - done) \* gamma \* next\_value - value

# 损失函数

policy\_loss = -torch.log(policy\_dist.squeeze(0)[action]) \* advantage

value\_loss = advantage \*\* 2

loss = policy\_loss + value\_loss

# 反向传播

optimizer.zero\_grad()

loss.backward()

optimizer.step()

state = next\_state

total\_reward += reward

episode\_rewards.append(total\_reward)

if (episode + 1) % 100 == 0:

print(f'Episode {episode + 1}, Total Reward: {total\_reward}')

return episode\_rewards

# 运行训练并绘制结果

rewards = train()

plt.plot(rewards)

plt.xlabel('Episode')

plt.ylabel('Total Reward')

plt.title('Training Performance')

plt.show()

def play\_agent():

env2 = gym.make('CartPole-v1', render\_mode='human')

state, info = env2.reset()

state = torch.tensor(state, dtype=torch.float32).unsqueeze(0)

done = False

while not done:

policy\_dist, \_ = model(state)

action = torch.multinomial(policy\_dist, 1).item()

next\_state, reward, done, \_, \_ = env2.step(action)

env2.render()

state = torch.tensor(next\_state, dtype=torch.float32)

if \_\_name\_\_ == '\_\_main\_\_':

# 训练代理

# train()

# 使用训练好的代理进行演示

play\_agent()

4. экранные формы с примерами выполнения программы.

Episode 100, Total Reward: 56.0

Episode 200, Total Reward: 94.0

Episode 300, Total Reward: 103.0

Episode 400, Total Reward: 139.0

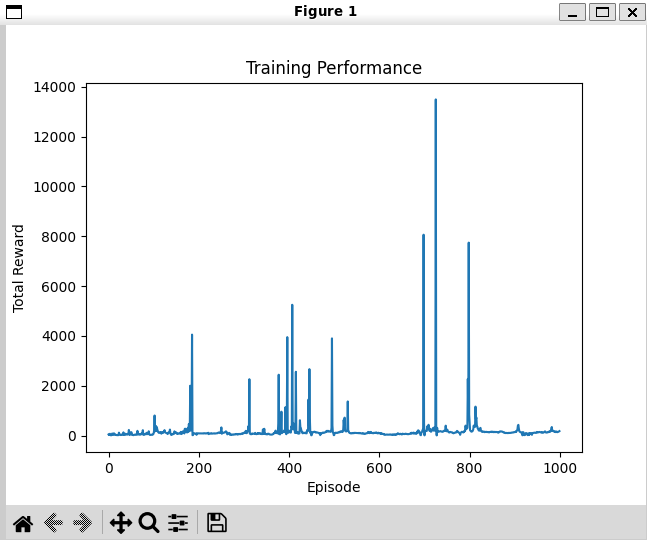
Episode 500, Total Reward: 20.0

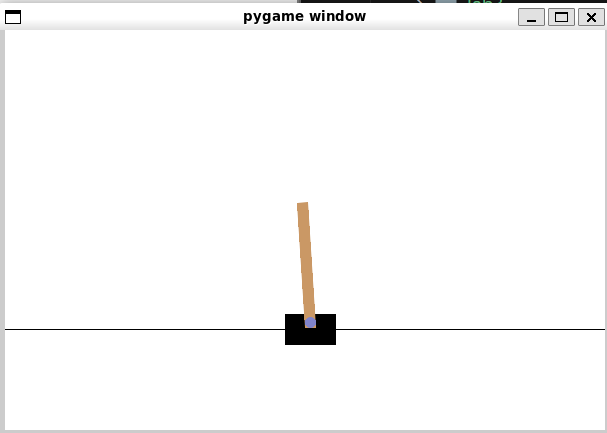
Episode 600, Total Reward: 94.0

Episode 700, Total Reward: 497.0

Episode 800, Total Reward: 843.0

Episode 900, Total Reward: 143.0

Episode 1000, Total Reward: 184.0



**Список литературы**

[1] Гапанюк Ю. Е. COURSE\_MMO\_SPRING\_2024// GitHub. –– 2024. –– Режим доступа: https://github.com/ugapanyuk/courses\_current/wiki/COURSE\_MMO\_SPRING\_2024

[2] <https://www.kaggle.com/datasets>