Московский государственный технический университет им. Н.Э. Баумана
Факультет «Информатика, искусственный интеллект и системы управления»
Кафедра «Системы обработки информации и управления»

Отчет

Лабораторная работа №7

По дисциплине «Методы машинного обучения»

«Алгоритмы Actor-Critic»

ИСПОЛНИТЕЛЬ:

студент ИУ5-21М
Базанова А.Г.

Москва, 2023

Цель работы: ознакомление с базовыми методами обучения с подкреплением на основе алгоритмов Actor-Critic.

Задание: Реализуйте любой алгоритм семейства Actor-Critic для произвольной среды.

Код программы:

```python
import gym
import torch
import torch.nn as nn
import numpy as np
import torch.optim as optim
from torch.distributions import Categorical
import pygame

class Policy(nn.Module):
    def __init__(self):
        super(Policy, self).__init__()
        self.fc1 = nn.Linear(4, 128)
        self.fc2 = nn.Linear(128, 1)

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

def update_policy(optimizer, policy, rewards, log_probs, values, gamma=0.99):
    R = 0
    returns = []
    for r in rewards[::-1]:
        R = r + gamma * R
        returns.insert(0, R)
    returns = torch.tensor(returns)
    returns = (returns - returns.mean()) / (returns.std() + 1e-8)
    log_probs = torch.stack(log_probs)
    values = torch.stack(values).squeeze()
    advantages = returns - values
    policy_loss = (-log_probs * advantages.detach()).mean()
    value_loss = advantages.pow(2).mean()
    entropy_loss = Categorical(torch.exp(log_probs)).entropy().mean()
    loss = policy_loss + 0.5 * value_loss - 0.01 * entropy_loss
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()


env = gym.make("CartPole-v1", render_mode="human")
env.reset()
env.render()
policy = Policy()
optimizer = optim.Adam(policy.parameters(), lr=1e-3)
for i_episode in range(1000):
    rewards = []
    log_probs = []
    values = []
    state,_ = env.reset()

    for t in range(10000):
        state = torch.from_numpy(np.array(state)).float()
        action_logits = policy(state)
        action_dist = Categorical(torch.softmax(action_logits, dim=-1))
        action = action_dist.sample()
        log_prob = action_dist.log_prob(action)
```

```
            value = policy(state).detach().squeeze()
            next_state, reward, done, _, _ = env.step(action.item())
            rewards.append(reward)
            log_probs.append(log_prob)
            values.append(value)
            state = next_state
            if done:
                break
        update_policy(optimizer, policy, rewards, log_probs, values)
        if i_episode % 10 == 0:
            print('Episode {}tLast length: {:5d}'.format(i_episode, t))
```

Пример:

```
Episode 0tLast length:     9
Episode 10tLast length:     8
Episode 20tLast length:     9
Episode 30tLast length:     8
Episode 40tLast length:     8
Episode 50tLast length:     9
Episode 60tLast length:     9
Episode 70tLast length:     8
Episode 80tLast length:     8
Episode 90tLast length:     8
Episode 100tLast length:     8
Episode 110tLast length:     7
Episode 120tLast length:     8
Episode 130tLast length:     8
Episode 140tLast length:     8
Episode 150tLast length:     9
Episode 160tLast length:     9
Episode 170tLast length:    10
Episode 180tLast length:     9
Episode 190tLast length:     8
Episode 200tLast length:     8
Episode 210tLast length:     8
Episode 220tLast length:     9
Episode 230tLast length:     9
Episode 240tLast length:     9
Episode 250tLast length:    8
Episode 260tLast length:    8
Episode 270tLast length:    9
Episode 280tLast length:    9
Episode 290tLast length:    8
Episode 300tLast length:    8
Episode 310tLast length:    7
Episode 320tLast length:    8
Episode 330tLast length:    7
Episode 340tLast length:    9
```

Экранная форма: