Министерство науки и высшего образования
Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
"Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)"
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ _ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА ___СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ (ИУ5)_____

Отчет по лабораторной работе №6

«Алгоритмы Actor-Critic»

по курсу «Методы машинного обучения».

ИСПОЛНИТЕЛЬ: Группа ИУ5-24М
Уралова Е.А.
ФИО
_____
подпись

"5" мая 2024 г.

ПРЕПОДАВАТЕЛЬ: Гапанюк Ю.Е.
ФИО
_____
подпись

"__"_____2024 г.

Москва – 2024

Задание:

Реализуйте любой алгоритм семейства Actor-Critic для произвольной среды.

Выполнение:

Создаем актора и критика с помощью нейронных сетей, обучаем их на примере среды CartPole и тестируем обученную модель.

```
[2]  import gym
     import numpy as np
     import tensorflow as tf
     from tensorflow.keras.layers import Dense
     from tensorflow.keras.optimizers import Adam
```

Создаем среду CartPole:

```
[3]  env = gym.make('CartPole-v1')
     state_dim = env.observation_space.shape[0]
     num_actions = env.action_space.n
```

Создаем актора (политику):

```
[4]  actor = tf.keras.models.Sequential([
         Dense(64, activation='relu', input_shape=(state_dim,)),
         Dense(64, activation='relu'),
         Dense(num_actions, activation='softmax')
     ])

     actor.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy')
```

Создаем критика (оценщика):

```
     critic = tf.keras.models.Sequential([
         Dense(64, activation='relu', input_shape=(state_dim,)),
         Dense(64, activation='relu'),
         Dense(1)
     ])

     critic.compile(optimizer=Adam(learning_rate=0.01), loss='mean_squared_error')
```

Обучение актора-критика:

✓ 5 мин.

```
num_episodes = 20
gamma = 0.99

for episode in range(num_episodes):
    state = env.reset()
    done = False
    total_reward = 0

    while not done:
        # Считаем вероятности действий от актора
        action_probs = actor.predict(np.expand_dims(state, axis=0))[0]

        # Выбираем действие на основе вероятностей
        action = np.random.choice(num_actions, p=action_probs)

        # Применяем выбранное действие к среде
        next_state, reward, done, _ = env.step(action)

        total_reward += reward
```

✓ 5 мин.

[11]

```
        # Считаем оценку критика для текущего состояния
        value = critic.predict(np.expand_dims(state, axis=0))[0]

        # Обновляем критика
        next_value = critic.predict(np.expand_dims(next_state, axis=0))[0]
        target = reward + gamma * next_value * (1 - done)
        critic.fit(np.expand_dims(state, axis=0), np.array([target]), verbose=0)

        # Считаем advantage для обновления актора
        advantage = target - value

        # Обновляем актора
        target_action = np.zeros(num_actions)
        target_action[action] = 1  # one-hot encoding
        actor.fit(np.expand_dims(state, axis=0), np.array([target_action * advantage]), verbose=0)

        state = next_state

    print(f'Episode {episode + 1}, Total reward: {total_reward}')
```

```
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 31ms/step
```

✓ 0 сек. выполнено в 12:46

Тестируем обученную модель:

```
[12] total_rewards = 0
     num_test_episodes = 5

     for _ in range(num_test_episodes):
         state = env.reset()
         done = False

         while not done:
             action = np.argmax(actor.predict(np.expand_dims(state, axis=0))[0])
             next_state, reward, done, _ = env.step(action)
             total_rewards += reward
             state = next_state

     average_reward = total_rewards / num_test_episodes
     print(f'Average reward over {num_test_episodes} test episodes: {average_reward}')
```

```
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 28ms/step
```

+ Код + Текст

```
[12] 1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 27ms/step
Average reward over 5 test episodes: 160.6
```

Вывод:

В данной лабораторной работе был реализован алгоритм семейства Actor-Critic для среды CartPole. Обучение проводилось 20 эпизодов, тесты на 5 эпизодах. На тестах обучение дало неплохие результаты: 160,6. Можно улучшать результаты, увеличивая количество эпизодов для обучения.