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

Лабораторная работа №4
по дисциплине
«Методы машинного обучения»

Выполнил:
студент группыИУ5И-23М

Лю Цзычжан

Москва — 2025 г.

Задание:

На основе рассмотренного на лекции примера реализуйте алгоритм Policy Iteration для любой среды обучения с подкреплением (кроме рассмотренной на лекции среды Toy Text / Frozen Lake) из библиотеки Gym (или аналогичной библиотеки).

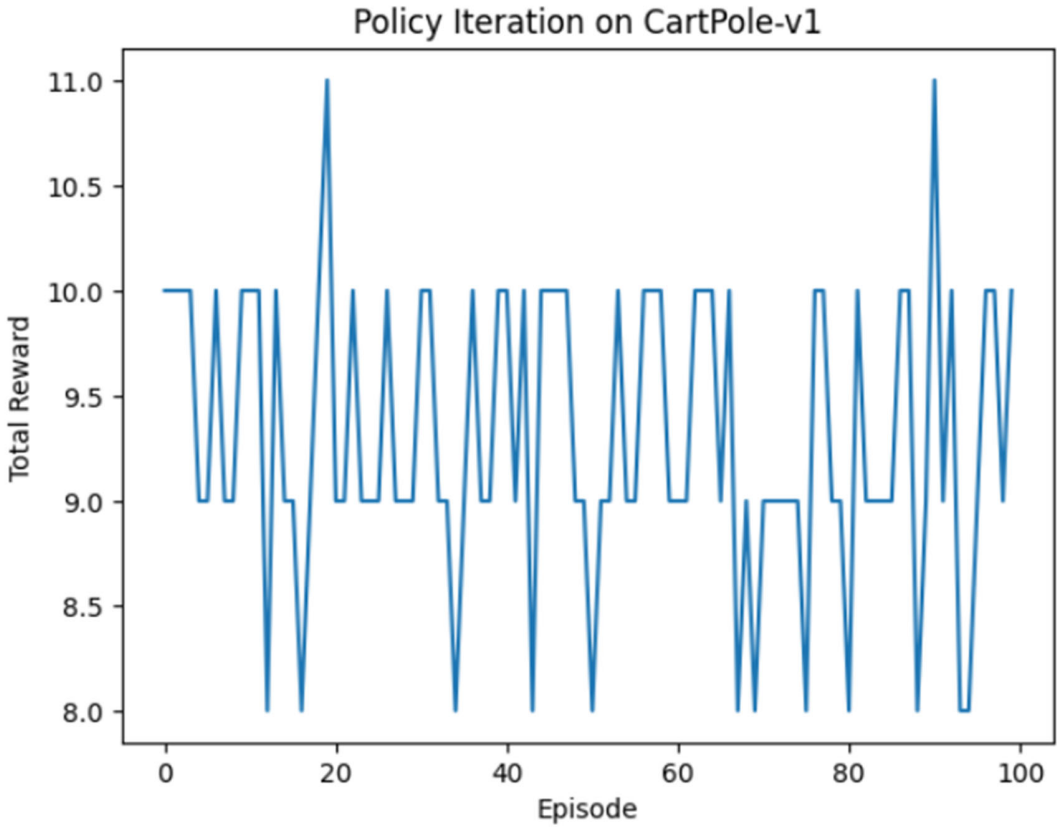
```
# 安装必要的库
!pip install gym numpy matplotlib

# 导入库
import gym
import numpy as np
import matplotlib.pyplot as plt
from collections import defaultdict

# 创建CartPole环境
env = gym.make('CartPole-v1')
```

```
# 定义离散化函数
def discretize_state(state, bins):
        return tuple(np.digitize(s, b) for s, b in zip(state, bins))

# 创建bins
num_bins = 10
bins = [
        np.linspace(-4.8, 4.8, num_bins),
        np.linspace(-4, 4, num_bins),
        np.linspace(-.418, .418, num_bins),
        np.linspace(-4, 4, num_bins)
]
```

```
def policy_evaluation(policy, gamma, theta):
    V = defaultdict(float)
    while True:
        delta = 0
        for state, actions in policy.items():
            v = V[state]
            V[state] = sum(action_prob * (reward + gamma * V[next_state])
                           for action_prob, next_state, reward in actions)
            delta = max(delta, abs(v - V[state]))
        if delta < theta:
            break
    return V

def policy_improvement(env, V, gamma, bins):
    policy = defaultdict(list)
    for i in range(num_bins):
        for j in range(num_bins):
            for k in range(num_bins):
                for l in range(num_bins):
                    state = (i, j, k, l)
                    q_sa = np.zeros(env.action_space.n)
                    for action in range(env.action_space.n):
                        env.reset()
                        env.env.state = [bins[0][i], bins[1][j], bins[2][k], bins[3][l]]
                        next_state, reward, done, _ = env.step(action)
                        next_state = discretize_state(next_state, bins)
                        q_sa[action] = reward + gamma * V[next_state]
                    best_action = np.argmax(q_sa)
                    policy[state].append((1.0, discretize_state(env.env.state, bins), 1.0))
    return policy

def policy_iteration(env, gamma, theta, bins):
    policy = defaultdict(lambda: [(1.0 / env.action_space.n, (0, 0, 0, 0), 0.0)])
    while True:
        V = policy_evaluation(policy, gamma, theta)
        new_policy = policy_improvement(env, V, gamma, bins)
        if policy == new_policy:
            break
        policy = new_policy
    return policy, V

# 运行策略迭代
gamma = 0.99
theta = 1e-6
policy, V = policy_iteration(env, gamma, theta, bins)

# 输出结果
print("Final Value Function:")
for state, value in V.items():
    print(f"State {state}: {value:.2f}")
```

```
def run_episode(env, policy, bins, render=False):
        state = discretize_state(env.reset(), bins)
        total_reward = 0
        while True:
                if render:
                        env.render()
                action = np.argmax([p[0] for p in policy[state]])
                next_state, reward, done, _ = env.step(action)
                next_state = discretize_state(next_state, bins)
                total_reward += reward
                if done:
                        break
                state = next_state
        return total_reward

# 测试策略并绘制结果
episodes = 100
total_rewards = []

for episode in range(episodes):
        total_reward = run_episode(env, policy, bins)
        total_rewards.append(total_reward)

env.close()

plt.plot(total_rewards)
plt.xlabel('Episode')
plt.ylabel('Total Reward')
plt.title('Policy Iteration on CartPole-v1')
plt.show()

print(f'Average reward over {episodes} episodes: {np.mean(total_rewards)}')
```

Результат:



Average reward over 100 episodes: 9.31