Московский государственный технический университет им. Н.Э. Баумана
Факультет «Головной учебно-исследовательский и методический центр
профессиональной реабилитации лиц с ограниченными возможностями здоровья (инвалидов)»
Кафедра «Системы обработки информации и управления»

Лабораторная работа 4

по дисциплине «Методы машинного обучения в АСОИУ»

" Реализация алгоритма Policy Iteration "

СТУДЕНТ:
студент группы ИУ5Ц-21М
Москалик А.А.

ПРЕПОДАВАТЕЛЬ:
Гапанюк Ю.Е.

Москва, 2024

Цель лабораторной работы: ознакомление с базовыми методами обучения с подкреплением.

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).

Ход работы

Импорт необходимых библиотек и определение функции policy_evaluation

Функция policy_evaluation — это часть алгоритма итерации политики (Policy Iteration), которая используется для оценки текущей политики. Цель этой функции — вычислить ценности состояний (value function) для данной политики, что позволяет понять, насколько хороша текущая политика в терминах ожидаемой награды. Эта информация затем используется для улучшения политики.

```
import gym
import numpy as np

# Функция для оценки текущей политики
def policy_evaluation(policy, env, discount_factor=0.95, theta=0.0001):
    # Инициализация оценок состояний нулями
    V = np.zeros(env.observation_space.n)
    while True:
        delta = 0
        # Проходим по всем состояниям среды
        for s in range(env.observation_space.n):
            v = 0
            # Проходим по всем действиям в текущей политике
            for a, action_prob in enumerate(policy[s]):
                # Проходим по всем возможным переходам из состояния s при действии a
                for prob, next_state, reward, done in env.P[s][a]:
                    v += action_prob * prob * (reward + discount_factor * V[next_state])
            delta = max(delta, np.abs(v - V[s]))
            V[s] = v
        # Условие завершения итераций, если изменения в V меньше порога theta
        if delta < theta:
            break
    return V
```

Определение функции policy_improvement

Функция policy_improvement — это часть алгоритма итерации политики, которая используется для улучшения текущей политики на основе оценок состояний. Основная идея этой функции заключается в выборе наилучших действий для каждого состояния, которые максимизируют ожидаемую награду.

```
# Функция для улучшения текущей политики на основе оценок состояний
def policy_improvement(V, env, discount_factor=0.95):
    # Инициализация новой политики
    policy = np.zeros([env.observation_space.n, env.action_space.n])
    for s in range(env.observation_space.n):
        A = np.zeros(env.action_space.n)
        # Проходим по всем действиям для состояния s
        for a in range(env.action_space.n):
            # Проходим по всем возможным переходам из состояния s при действии a
            for prob, next_state, reward, done in env.P[s][a]:
                A[a] += prob * (reward + discount_factor * V[next_state])
        best_action = np.argmax(A)
        policy[s, best_action] = 1.0
    return policy
```

Определение функции policy_iteration

Функция policy_iteration — это ключевая часть алгоритма итерации политики (Policy Iteration) в обучении с подкреплением. Этот алгоритм используется для нахождения оптимальной политики в детерминированных и стохастических средах. Алгоритм состоит из двух основных шагов: оценка текущей политики (policy evaluation) и улучшение политики (policy improvement). Эти шаги повторяются до тех пор, пока политика не перестанет изменяться, что свидетельствует о достижении оптимальной политики.

```
# Функция для выполнения итераций политики до сходимости
def policy_iteration(env, discount_factor=0.95):
    # Инициализация случайной политики
    policy = np.ones([env.observation_space.n, env.action_space.n]) / env.action_space.n
    while True:
        # Оценка текущей политики
        V = policy_evaluation(policy, env, discount_factor)
        # Улучшение политики на основе оценок состояний
        new_policy = policy_improvement(V, env, discount_factor)
        # Проверка на сходимость политики
        if (new_policy == policy).all():
            return policy, V
        policy = new_policy
```

Инициализация среды, запуск алгоритма и вывод результатов

На этом шаге мы инициализируем среду Taxi-v3 из библиотеки Gym. Среда представляет собой задачу, в которой агент (такси) должен забрать пассажира в одном месте и отвезти его в другое. Среда включает дискретное пространство состояний и действий.

Далее вызывается функция policy_iteration, которая реализует алгоритм итерации политики. Этот алгоритм проходит через циклы оценки политики и улучшения политики до тех пор, пока не будет достигнута оптимальная политика. В результате мы получаем:

- final_policy: оптимальная политика, представленная в виде матрицы, где каждая строка соответствует состоянию, а столбцы — действиям. Значение 1.0 в столбце означает, что это действие является оптимальным для данного состояния.

- V: оценки состояний, представляющие собой массив, где каждое значение отражает ожидаемую суммарную награду, начиная из соответствующего состояния и следуя оптимальной политике.

```
# Инициализация среды
env = gym.make('Taxi-v3')

# Применение алгоритма Policy Iteration
final_policy, V = policy_iteration(env)

# Вывод результатов
print("Оптимальная политика (вероятность выбора каждого действия в каждом состоянии):")
print(final_policy)
print("\nОценки состояний:")
print(V)
```

```
Оптимальная политика (вероятность выбора каждого действия в каждом состоянии):
[[0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1. 0.]
 ...
 [0. 1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 1. 0. 0.]]
```

```
Оценки состояний:
[184.61476998 115.74527209 146.659755   122.88967428  70.05607295
 115.74527209  70.05593297  85.03680766 115.74541207  90.56523255
 146.659755    96.3843695   79.78518878  90.56523255  79.7850488
 122.88967428 195.38403148 122.88983755 155.4313985  130.41027379
 174.38403148 108.95800849 138.32676725 115.74519057  74.79592934
 122.88983755  74.79579636  90.56515103 108.95814147  85.03697092
 138.32676725  90.56515103  85.0371039   96.38453276  85.03697092
 130.41027379 184.6148299  130.41042889 146.65982857 138.3266898
 130.41055522  79.78512238 102.51010806  85.03689348 102.51023439
 164.66470357 102.51010806 122.8897601  102.51023439  79.78512238
 130.41042889  85.03689348  90.56543246 102.51010806  90.56530613
 138.3266898  138.32695716 174.3839684  138.32683715 146.659755
 122.89002745  74.79586626  96.38460266  79.7850488  108.9581984
 174.3839684  108.95807838 130.41035531  96.38472267  74.79586626
 122.88990744  79.7850488   96.38472267 108.95807838  96.38460266
 146.659755   130.4106093  184.61476998 130.41049529 155.4313985
 115.74552608  70.05607295  90.56537253  74.79579636 115.74552608
 184.61476998 115.74541207 138.32676725  90.56548654  70.05607295
 115.74541207  74.79579636  90.56548654 102.51017447  90.56537253
 138.32676725 122.89007883 195.38403148 122.88997052 146.65982857
 174.38403148 108.95800849 138.32676725 115.74519057  74.79592934
 122.88983755  74.79579636  90.56515103 122.88997052  96.38453276
 155.4313985  102.50995297  85.0371039   96.38453276  85.03697092
 130.41027379 184.6148299  130.41042889 164.66470357 138.3266898
 164.6648299  102.51010806 130.41042889 108.95793104  79.78524871
 130.41042889  79.78512238  96.38445532 115.745472    90.56530613
 146.65982857  96.38445532  90.56543246 102.51010806  90.56530613
 138.3266898  174.38408841 138.32683715 155.4314684  146.659755
 138.32695716  85.03704082 108.95807838  90.56523255  96.38472267
 155.4314684   96.38460266 115.74527209 108.9581984   85.03704082
 138.32683715  90.56523255  96.38472267 108.95807838  96.38460266
 146.659755   146.66000899 164.66476998 146.65989498 155.4313985
 130.4106093   79.78518878 102.51017447  85.03697092 102.51028848
 164.66476998 102.51017447 122.88983755 102.51028848  79.78518878
 130.41049529  85.03697092 102.51028848 115.74541207 102.51017447
 155.4313985  138.32700854 174.38403148 138.32690023 164.66470357
 122.89007883  74.79592934  96.38466574  79.78512238 108.95824978
 174.38403148 108.95814147 130.41042889  96.38477405  74.79592934
 122.88997052  79.78512238  96.38477405 108.95814147  96.38466574
 146.65982857 130.41065811 184.6148299  130.41055522 155.4314684
 164.6648299  102.51010806 130.41042889 108.95793104  79.78524871
 130.41042889  79.78512238  96.38445532 130.41055522 102.51010806
 164.66470357 108.95793104  90.56543246 102.51010806  90.56530613
 138.3266898  174.38408841 138.32683715 174.3839684  146.659755
 155.43158841  96.38460266 122.88990744 102.51003449  85.03716083
 138.32683715  85.03704082 102.51003449 122.89002745  96.38460266
 155.4314684  102.51003449  96.38472267 108.95807838  96.38460266
 146.659755   164.66488399 146.65989498 164.66476998 155.4313985
 146.66000899  90.56537253 115.74541207  96.38453276  90.56548654
 146.65989498  90.56537253 108.95800849 115.74552608  90.56537253
 146.65989498  96.38453276 102.51028848 115.74541207 102.51017447
 155.4313985  155.43163979 155.43153148 155.43153148 164.66470357
 138.32700854  85.0371039  108.95814147  90.56530613  96.38477405
 155.43153148  96.38466574 115.74534567 108.95824978  85.0371039
```

```
138.32690023  90.56530613 108.95824978 122.88997052 108.95814147
164.66470357 146.6600578  164.6648299  146.6599549  174.3839684
130.41065811  79.78524871 102.51023439  85.03704082 102.51033729
164.6648299  102.51023439 122.88990744 102.51033729  79.78524871
130.41055522  85.03704082 102.51033729 115.745472   102.51023439
155.4314684  138.32705491 174.38408841 138.32695716 164.66476998
155.43158841  96.38460266 122.88990744 102.51003449  74.79598627
122.88990744  74.79586626  90.56523255 138.32695716 108.95807838
174.3839684  115.74527209  85.03716083  96.38460266  85.03704082
130.41035531 164.66488399 130.41049529 184.61476998 138.32676725
146.66000899  90.56537253 115.74541207  96.38453276  79.78530279
130.41049529  79.78518878  96.38453276 115.74552608  90.56537253
146.65989498  96.38453276  90.56548654 102.51017447  90.56537253
138.32676725 155.43163979 138.32690023 155.43153148 146.65982857
138.32700854  85.0371039  108.95814147  90.56530613  85.03721221
138.32690023  85.0371039  102.51010806 108.95824978  85.0371039
138.32690023  90.56530613  96.38477405 108.95814147  96.38466574
146.65982857 146.6600578  146.6599549  146.6599549  155.4314684
130.41065811  79.78524871 102.51023439  85.03704082  90.56553535
146.6599549   90.56543246 108.95807838 102.51033729  79.78524871
130.41055522  85.03704082 115.74557489 130.41055522 115.745472
174.3839684  138.32705491 155.43158841 138.32695716 184.61476998
122.89012521  74.79598627  96.38472267  79.78518878  96.38482042
155.43158841  96.38472267 115.74541207  96.38482042  74.79598627
122.89002745  79.78518878 108.95829615 122.89002745 108.9581984
164.66476998 130.41070216 164.66488399 130.4106093  174.38403148
146.66000899  90.56537253 115.74541207  96.38453276  70.05618696
115.74541207  70.05607295  85.03697092 146.66000899 115.74541207
184.61476998 122.88983755  79.78530279  90.56537253  79.78518878
122.88983755 155.43163979 122.88997052 195.38403148 130.41042889
138.32700854  85.0371039  108.95814147  90.56530613  74.79603765
122.88997052  74.79592934  90.56530613 108.95824978  85.0371039
138.32690023  90.56530613  85.03721221  96.38466574  85.0371039
130.41042889 146.6600578  130.41055522 146.6599549  138.32683715
130.41065811  79.78524871 102.51023439  85.03704082  79.7853516
130.41055522  79.78524871  96.38460266 102.51033729  79.78524871
130.41055522  85.03704082  90.56553535 102.51023439  90.56543246
138.32683715 138.32705491 138.32695716 138.32695716 146.65989498
122.89012521  74.79598627  96.38472267  79.78518878  85.03725858
138.32695716  85.03716083 102.51017447  96.38482042  74.79598627
122.89002745  79.78518878 122.89012521 138.32695716 122.89002745
184.61476998 130.41070216 146.66000899 130.4106093  195.38403148
115.74561895  70.05618696  90.56548654  74.79592934  90.5655794
146.66000899  90.56548654 108.95814147  90.5655794   70.05618696
115.74552608  74.79592934 115.74561895 130.4106093  115.74552608
174.38403148 122.89016705 155.43163979 122.89007883 184.6148299 ]
```

Оптимальная политика: Алгоритм итерации политики успешно определил оптимальные действия для каждого состояния в среде `Taxi-v3`. Эти действия обеспечивают максимизацию ожидаемой награды, что соответствует целям агента.

Оценки состояний: Значения оценок состояний предоставляют полезную информацию о ценности каждого состояния. Это помогает понять, какие состояния являются более предпочтительными для агента с точки зрения получения награды.

Вывод:

Алгоритм Policy Iteration продемонстрировал свою эффективность в решении задачи управления такси в среде `Taxi-v3`. Реализация этого алгоритма позволила глубже понять методы обучения с подкреплением и их применение в задачах принятия решений. Полученные результаты показывают, что алгоритмы обучения с подкреплением могут быть успешно применены для решения сложных задач управления и оптимизации.