НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
УКРАЇНИ «Київський політехнічний інститут імені
Ігоря Сікорського»

Комп’ютерний практикум №2

з дисципліни «Технології штучного інтелекту»
Тема: «Моделі машинного навчання»

Виконав:
студент групи ЗПІ-зп01
Дишкант Л. Л.

Перевірив:
доцент кафедри ІСТ
Олійник В.В.

Київ 2023

Комп’ютерний практикум №2

Моделі пошуку в стохастичному і невідомому середовищі

ПІБ: Дишкант Лариса Леонідівна
Група: ЗПІ-зп01

Мета роботи: ознайомитись з алгоритмами пошуку в умовах невідомості та навчання з підкріпленнями; дослідити їх використання для інтелектуального агента в типовому середовищі.

Завдання: середовище моделювання Gridword та задача дослідження вплив коефіцієнта знецінення, що містить агента, який може бути навчений методом «з підкріпленнями». В обраному середовищі вирішити задачу знаходження найкращої стратегії поведінки, реалізувавши Value iteration. Виконати дослідження реалізованого методу.

Номер варіанту: 3.
Завдання для варіанту: задача дослідження вплив коефіцієнта знецінення, Value iteration.

Середовище: двовимірне середовище розміром 20*20, з можливими чотирма діями: вгору, вниз, вліво та вправо. Карта з стінами та вузькими проходами, агент, який у вигляді червоного квадрата та винагорода - +100 жовтий квадрат , рух в заборонені квадрати штрафується - 1. ВІзуалізація середовища на рисунку 1

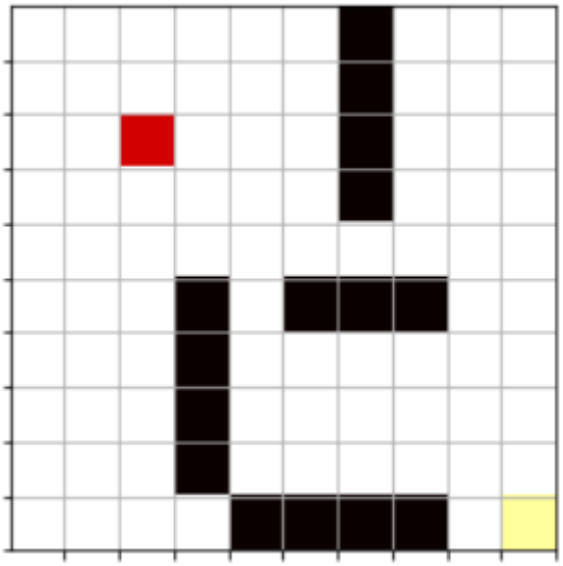

Рисунок 1 - середовище де винагорода жовтий квадрат та агент червоний

Метод вирішення задачі: метод Value iteration - метод знаходження оптимальної вартісної функції шляхом вирішення Рівняння Беллмана ітераційно. Він використовує концепцію динамічного програмування для підтримки функції значень, яка апроксимує функцію оптимального значення , ітераційно вдосконалюючись, поки вона не зійдеться до розв’язку, або заздалегідь визначену кількість ітерацій .

Реалізація методу:

```
def Value_iteration(qmaze, gamma=0.9, delta_provided=0.1,
max_iterations = 80000):
    # initialize V(s) for all states
    nrows, ncols = qmaze.maze.shape
    V = np.zeros((nrows, ncols))
    iterations = 0
    while True:
        delta = 0
        # update value of each state
        for r in range(nrows):
            for c in range(ncols):
                state = (r, c)
                if state == qmaze.target:
                    continue
                old_value = V[r, c]
                new_value = -1e6
```

```
                # calculate the expected value for all possible actions
                for action in qmaze.valid_actions(state):
                    qmaze.reset(state)
                    envstate, reward, _ = qmaze.act(action)
                    nrow, ncol, _ = envstate
                    value = reward + gamma * V[nrow, ncol]
                    if value > new_value:
                        new_value = value
                # Add the expected value of false action
                false_action_value = gamma * V[r, c]
                new_value = (qmaze.false_action_prob * (-1.0) + (1 -
qmaze.false_action_prob) * new_value) + qmaze.false_action_prob *
false_action_value
                V[r, c] = new_value
                delta = max(delta, np.abs(old_value - new_value))
                iterations = iterations + 1

        # check if the value function has converged or max iterations
reached
        if delta < delta_provided or iterations > max_iterations:
            break

    return V
```

Також знаходження найкращої стратегії - policy

```
def policy(qmaze, V, gamma=0.9):



    nrows, ncols = qmaze.maze.shape
    policy = np.zeros((nrows, ncols), dtype=int)

    # calculate the expected values of all possible actions for each
state
    for r in range(nrows):
        for c in range(ncols):
            state = (r, c)
            if state == qmaze.target:
                continue
            best_action = None
            best_value = -1e6
            # calculate the expected value for all possible actions
            for action in qmaze.valid_actions(state):
                qmaze.reset(state)
```

```
                envstate, reward, _ = qmaze.act(action)
                nrow, ncol, _ = envstate
                # calculate the expected value of the next state using
the value function
                value = reward + gamma * V[nrow, ncol]
                if value > best_value:
                    best_value = value
                    best_action = action
            policy[r, c] = best_action

    return policy
```

Результати застосування розробленого методу: результат навчання, тобто гри знаходить оптимальну стратегію рисунок 2 та рисунок 3

```
Policy: [[2 3 2 3 3 3 2 3 3 3]
 [3 3 2 3 2 3 2 3 3 3]
 [2 3 2 3 3 3 2 3 3 3]
 [2 2 3 2 3 3 3 2 3 3]
 [2 2 2 2 2 2 2 2 3 3]
 [2 2 1 2 3 3 3 3 3 3]
 [2 2 1 2 2 3 2 3 3 3]
 [2 2 1 2 2 2 2 3 3 3]
 [1 1 1 2 2 2 2 2 3 3]
 [2 1 1 0 1 1 1 2 2 0]]
Total reward: 95.43
Game status: Win
```

Рисунок 2 - оптимальна стратегія

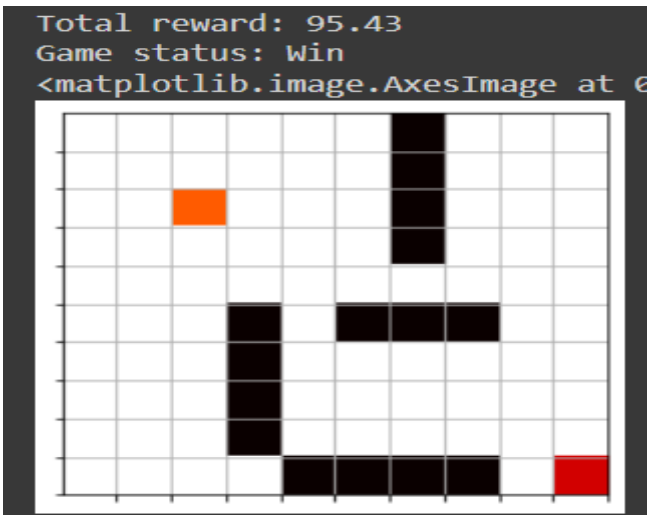

Рисунок 3 - винагорода, яку отримав агент

Оцінка результатів: в невеликих середовищах алгоритм Value-iteration швидко знаходить оптимальний шлях для отримання максимальної винагороди, але в середовищі з великими розмірностями маємо часові втрати.

Задача дослідження впливу параметра алгоритму:вплив коефіцієнту знецінення $gamma$ рисунок 4

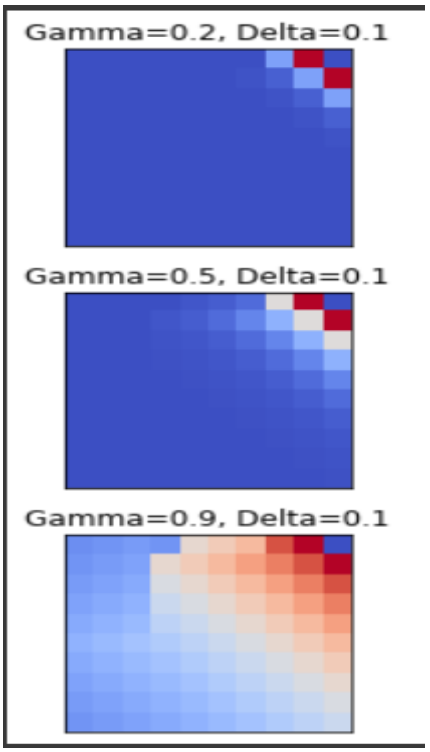

Рисунок 4 - результат методу Value iteration при зміні параметру коефіцієнту знецінення

Чим більша гамма тим більша вага майбутнім винагородам і на графіках ми це бачимо у вигляді зміни кольорів наших квадратів з синього в поступово червоний. При 0.9 - надає вагу максимальній винагороді.

Висновки щодо впливу зміни параметрів:

Чим блище гамма до 1, тим більша вага надається майбутній винагороді, а отже агент буде більш орієнтований на цю винагороду у своїх діях та буде враховувати майбутні наслідки своїх дій. З іншої сторони, якщо гамма близька до нуля - агент буде більш орієнтований на миттєву (поточну)

перспективу, а його дії будуть в більшій мірі зконцентровані на миттєвій винагороді.

Висновок: на практичній роботі ознайомилися з алгоритмом Value iteration пошуку в умовах невідомості та навчання з підкріпленнями; дослідили його використання для інтелектуального агента в типовому середовищі.