**Комп’ютерний практикум №3**

**Марківські процеси прийняття рішень. Навчання з підкріпленням.**

**ПІБ: Головня Олександр Ростиславович**

**Група: ІП-11**

**Мета роботи:** ознайомитись з методами пошуку в умовах невідомості та навчання з підкріпленням в моделях на основі станів; дослідити їх використання для інтелектуального агента в типовому середовищі.

***З*авдання:** обрати середовище моделювання та задачу, що містить агента, який може бути навчений методом «з підкріпленнями». В обраному середовищі вирішити задачу знаходження найкращої стратегії поведінки, реалізувавши один з методів. Виконати дослідження реалізованого методу.

**Номер варіанту: 15**

**Завдання для варіанту:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер студента/бригади** | **Форма карти** | **Задача дослідження** | **Алгоритм** |
| 15 | приміщення з окремими перешкодами | вплив коефіцієнта швидкості навчання α | Q-learning |

**Середовище:**

Обране середовище - це (GridWorld), який складається з клітинок розміром 10х10. Середовище є детермінованим,.

Стани в середовищі - це координати, які визначають місцезнаходження агента в сітці (self.current\_location).

Дії агента - це переміщення вгору, вниз, вліво або вправо. Переходи між станами відбуваються згідно з вибраною дією.

Винагороди в середовищі -1 для кожної клітинки, крім позначених як перешкоди, бомба та золото. Для перешкод винагорода -10, для бомби -100, для золота 100. Мета агента - навчитися обирати оптимальні дії для досягнення золота, уникаючи перешкод і бомб.

**Метод вирішення задачі:** Обраний метод вирішення задачі - Q-learning. Це метод навчання з підсиленням, який дозволяє агентові вчитися оптимальній стратегії дій в невідомому середовищі, спираючись лише на винагороди, отримані за свої дії. Q-learning є онлайн стратегією, оскільки агент навчається в режимі реального часу, і кожна взаємодія з середовищем допомагає йому вдосконалювати свою стратегію.

Дослідження середовища для Q-learning допомагає агентові навчитися оптимальній стратегії без знання моделі середовища. У цьому випадку дослідження включає в себе взаємодію агента з середовищем, отримання винагород та оновлення Q-значень для кожної пари (стан, дія). В результаті тренування агент навчається виробляти оптимальні дії, щоб максимізувати очікувану суму винагород за тривалість гри.

У випадку Q-learning для цього використовується метод play, де агент взаємодіє з середовищем, обираючи дії на основі поточного стану та оновлюючи свої Q-значення за допомогою функції навчання. Тренування проводиться протягом кількох епізодів гри, під час яких агент навчається оптимальній стратегії.

**Реалізація методу:**

class Q\_Agent():

    # Intialise

    def \_\_init\_\_(self, environment, epsilon=0.05, alpha=0.1, gamma=1):

        self.environment = environment

        self.q\_table = dict() # Store all Q-values in dictionary of dictionaries

        for x in range(environment.height): # Loop through all possible grid spaces, create sub-dictionary for each

            for y in range(environment.width):

                self.q\_table[(x,y)] = {'UP':0, 'DOWN':0, 'LEFT':0, 'RIGHT':0} # Populate sub-dictionary with zero values for possible moves

        self.epsilon = epsilon

        self.alpha = alpha

        self.gamma = gamma

    def choose\_action(self, available\_actions):

        """Returns the optimal action from Q-Value table. If multiple optimal actions, chooses random choice.

        Will make an exploratory random action dependent on epsilon."""

        if np.random.uniform(0,1) < self.epsilon:

            action = available\_actions[np.random.randint(0, len(available\_actions))]

        else:

            q\_values\_of\_state = self.q\_table[self.environment.current\_location]

            maxValue = max(q\_values\_of\_state.values())

            action = np.random.choice([k for k, v in q\_values\_of\_state.items() if v == maxValue])

        return action

    def learn(self, old\_state, reward, new\_state, action):

        """Updates the Q-value table using Q-learning"""

        q\_values\_of\_state = self.q\_table[new\_state]

        max\_q\_value\_in\_new\_state = max(q\_values\_of\_state.values())

        current\_q\_value = self.q\_table[old\_state][action]

        self.q\_table[old\_state][action] = (1 - self.alpha) \* current\_q\_value + self.alpha \* (reward + self.gamma \* max\_q\_value\_in\_new\_state)

**Результати застосування розробленого методу:** результати роботи програми,отримані показники корисності тощо. Бажано візуально відобразити динаміку навчання, наприклад, у вигляді залежності корисності від кількості епізодів.

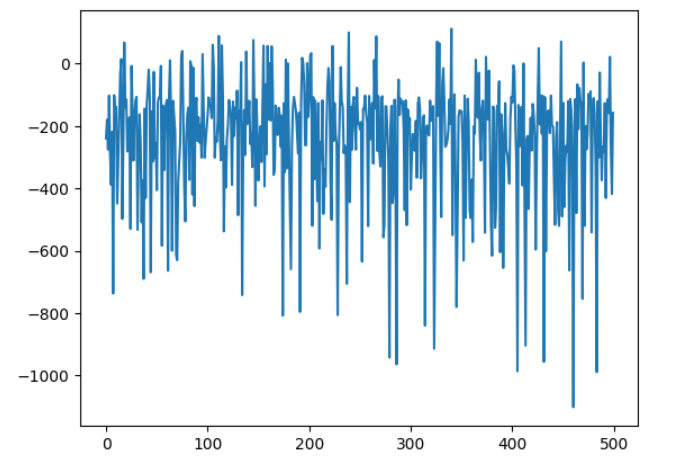
****

Рис.1 – Результат виконання RandomAgent

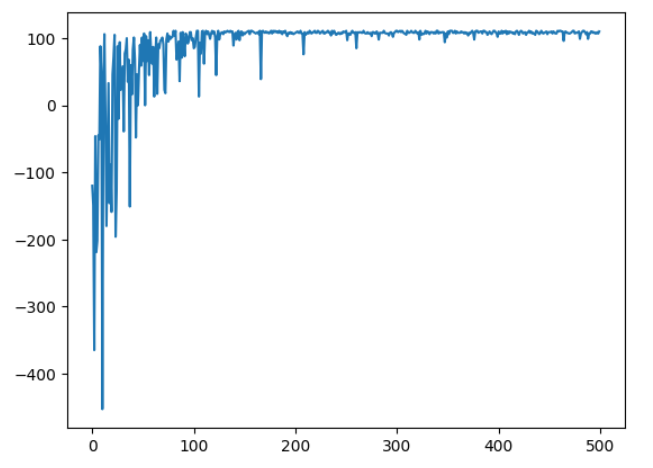
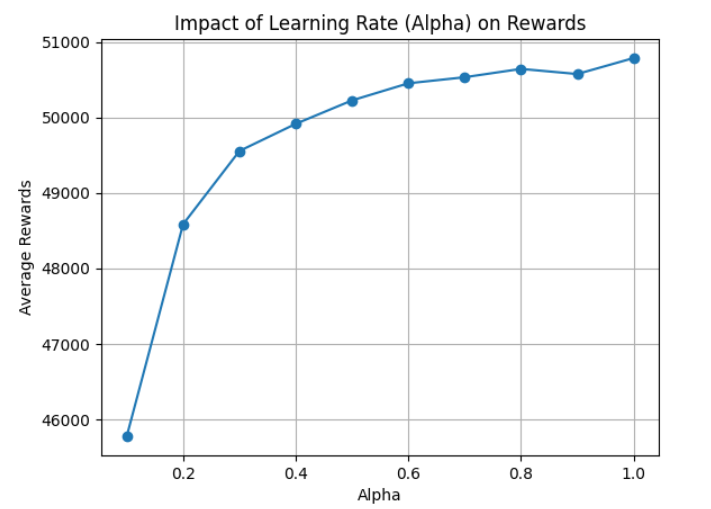
****

Рис.2 – Результат виконання Q\_Agent

**Оцінка результатів:** Отримані результати можуть варіюватися в залежності від ряду факторів, таких як обрана стратегія дослідження, параметри навчання (alpha, gamma, epsilon), кількість тренувальних епізодів, складність середовища тощо. Для кращого розуміння результатів необхідно аналізувати ці аспекти.

**Задача дослідження впливу параметра алгоритму:**



Параметр alpha в Q-learning визначає швидкість навчання, тобто вплив нових винагород на оновлення Q-значень. Якщо alpha дуже велика, агент може занадто агресивно змінювати свої Q-значення, що може призвести до нестабільної або неправильної стратегії.

З іншого боку, якщо alpha дуже мала, агент може повільно вчитися та не досягти оптимальної стратегії. Тому важливо підібрати оптимальне значення alpha, яке забезпечить стабільне та швидке навчання, уникнувши перенавчання або недонавчання.