**Комп’ютерний практикум №3**

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

**ПІБ: Мєшков Андрій Ігорович, Ткач Владислав Анатолійович**

**Група: ІП-15**

**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: 23**

**Завдання для варіанту:** 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 Value iteration. Виконати дослідження реалізованого методу - вплив коефіцієнта швидкості навчання α.

**Середовище:**

* **Стохастичність**: Лабіринт може мати стохастичність, якщо є можливість деякої випадковості у рухах агента
* **Стани**: Стани представлені координатами клітинок сітки, де кожна клітина може бути порожньою, містити стіну, агента, бомбу або золото. Агент може займати будь-яку клітину, яка не є стіною.
* **Дії**: Агент може рухатися ВГОРУ, ВНИЗ, ВЛІВО або ВПРАВО, якщо він не натрапить на стіну, в іншому випадку він залишається в тій самій клітині.
* **Переходи**: Переходи між станами залежать від вибраної дії. Якщо дія приводить агента до клітини зі стіною, він залишається в тій самій клітині. В іншому випадку він переміщується в сусідню клітину відповідно до вибраної дії.
* **Винагороди**: Винагороди попередньо визначені для певних станів:
  + -1 за звичайні порожні клітини.
  + 100 за клітину зі золотом.
  + -100 за клітину з бомбою.

Це середовище налаштоване для задач навчання з підсиленням, зокрема для ітерації вартості.

**Метод вирішення задачі:** Метод вирішення задачі Value Iteration є одним із найпоширеніших методів для навчання з підсиленням. Основна ідея полягає в тому, щоб ітеративно оновлювати оцінки значень кожного стану у середовищі, використовуючи принцип оптимальності Беллмана.

Ось характеристики методу Value Iteration:

- Офлайн стратегія: Value Iteration є офлайн стратегією, оскільки вона не вимагає взаємодії з середовищем під час навчання. Агент використовує інформацію про середовище, щоб оцінити оптимальні значення станів, не потребуючи активного взаємодії з ним.

- Дослідження середовища: Дослідження середовища необхідне для Value Iteration, але воно відбувається в офлайн режимі. Під час дослідження агент аналізує властивості середовища, такі як можливі переходи між станами і винагороди, для того щоб оцінити оптимальні значення станів.

- Реалізація дослідження середовища: У Value Iteration дослідження середовища зазвичай реалізується через ітерації алгоритму, де агент оновлює свої оцінки значень станів на кожному кроці. Цей процес включає аналіз доступних дій в кожному стані, переходів між станами, а також винагород, отриманих за ці переходи. Після дослідження агент використовує оновлені оцінки для вибору оптимальних дій у майбутньому.

**Реалізація методу:**

def iterate\_values(env, v, gamma, theta, p\_stoch):

converged = False

k = 0

sp = p\_stoch

p = {'UP': [sp + (1 - sp) / 4, (1 - sp) / 4, (1 - sp) / 4, (1 - sp) / 4],

'DOWN': [(1 - sp) / 4, sp + (1 - sp) / 4, (1 - sp) / 4, (1 - sp) / 4],

'LEFT': [(1 - sp) / 4, (1 - sp) / 4, sp + (1 - sp) / 4, (1 - sp) / 4],

'RIGHT': [(1 - sp) / 4, (1 - sp) / 4, (1 - sp) / 4, sp + (1 - sp) / 4]}

*while* not converged:

DELTA = 0

*for* i *in* range(0, env.height):

*for* j *in* range(0, env.width):

k += 1

*if* env.check\_state((i,j)) or (i,j) in env.walls:

v[i,j] = 0

*else*:

old\_v = v[i,j]

new\_v = []

*for* action *in* env.actions:

new\_v\_p = []

*for* idx, action\_p *in* enumerate(env.actions):

((n\_i, n\_j), reward) = env.P.get(((i,j), action\_p))

new\_v\_p.append(p.get(action)[idx] \* (reward + (gamma \* v[n\_i, n\_j])))

new\_v.append(sum(new\_v\_p))

v[i,j] = max(new\_v)

DELTA = max(DELTA, np.abs(old\_v - v[i,j]))

converged = True *if* DELTA < theta *else* False

print(k, 'iterations of state space')

*return* v

**Результати застосування розробленого методу:**

**Оцінка результатів:**

**Задача дослідження впливу параметра алгоритму:**