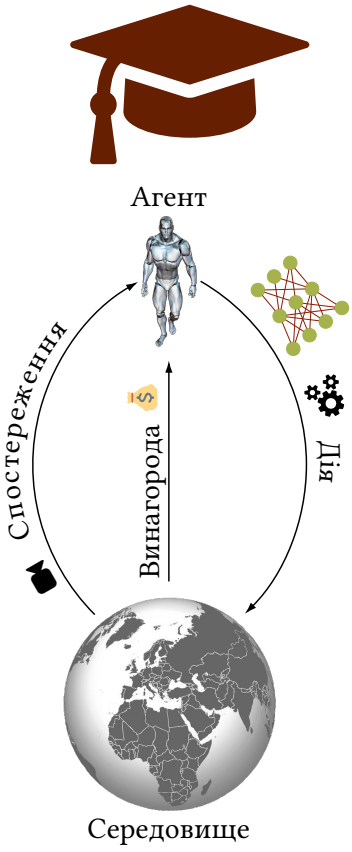
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

# Навчання нейронних мереж з підкріпленням

Методичні вказівки для виконання практичних робіт | Осінній семестр



# Q-навчання

*"Невдачі неодмінно будуть, і те, як ви з ними впораєтеся,*
*буде найважливішим показником того, чи досягнете ви успіху."*

– Джеймі Даймон

## Опис завдання

Агент має знайти вихід з лабіринту, уникаючи перешкод. Для цього розробіть агента, який зможе вивчити оптимальний шлях в лабіринті, використовуючи алгоритм Q-навчання.

Нижче подано приклад простого середовища для лабіринту. Для візуалізації використано текстову графіку та лінійну структуру для руху агента.

**MazeEnv** – це клас, що моделює середовище лабіринту. Лабіринт задається у вигляді матриці, де 0 – вільний простір, 1 – перешкода, 2 – агент, а 3 – мета (вихід). Агент починає з позиції $[0, 0]$, а мета знаходиться в позиції $[size - 1, size - 1]$. Для переміщення по лабіринту агент може використовувати такі дії: up, down, left, right. Метод **step** дозволяє агенту рухатись у вибраному напрямку. Якщо рух призводить до зіткнення з перешкодою, агент залишається на місці і отримує винагороду $-1$. Якщо агент досягає мети, гра завершується і надається винагорода 10. Метод **render** виводить лабіринт у вигляді матриці, де можна бачити, як рухається агент.

## Завдання для виконання

Відкрийте завдання:

https://nbviewer.org/github/YKochura/rl-kpi/blob/main/practice/practice1/Maze.ipynb

Вам потрібно імплементувати кілька функцій, які будуть реалізовувати алгоритм Q-навчання. Використовуючи алгоритм Q-навчання навчіть агента знаходити оптимальний шлях (вихід) з лабіринту. Функції, які потрібно імплементувати позначено у завданні так:

```
# TODO
```

Розміщуйте свою реалізацію між рядками:

```
# BEGIN_YOUR_CODE

# END_YOUR_CODE
```

### 0.0.1 Алгоритм Q-навчання

Алгоритм Q-навчання (Q-learning) є методом навчання з підкріпленням, який дозволяє агенту навчитися знаходити оптимальну стратегію для прийняття рішень у середовищі. Агент використовує Q-таблицю, щоб

оцінити, яку дію йому краще виконати в кожному стані. Цей алгоритм добре підходить лише для середовищ з невеликим дискретним простором станів для яких Q-таблиця є невеликою.

1. **Ініціалізація Q-таблиці.** Це таблиця розміром $n \times m$, де $n$ – кількість станів, а $m$ – кількість можливих дій у заданому стані. Значення таблиці оновлюється на основі винагороди, яку агент отримує за виконану дію в поточному стані. На початку всі значення Q-таблиці зазвичай ініціалізуються нулями або невеликими випадковими значеннями:

$$Q(s, a) = 0 \quad \forall\ s \in S, a \in A,$$

де $s$ – стан у якому знаходиться агент, $a$ – дія, яку виконує агент у цьому стані.

2. **Вибір початкового стану**. Епізод починається з деякого початкового стану $s_0$. Зазвичай агент починає взаємодію з середовищем з деякого фіксованого стану, але у загальному випадку, початковий стан можна обирати випадковим чином на початку кожного епізоду.

3. **Вибір дії за стратегією**. Для вибору дій агент буде використовувати $\varepsilon$-жадібну стратегію. Якщо випадкове число менше за $\varepsilon$, агент обирає випадкову дію (exploration), щоб дослідити середовище, інакше – використовує набуті знання та обирає найкращу дію $a^*$ (exploitation) на основі поточних значень Q-таблиці:

$$a^* = \arg\max_a Q(s, a) \qquad \forall a_i \in A$$

Таким чином, агент балансує між вивченням середовища та використанням раніше набутого досвіду.

4. **Отримання винагороди.** Після виконання кожної дії агент переходить у нових стан та отримує за це винагороду.

5. **Оновлення Q-таблиці.** Після кожного кроку агент оновлює значення Q-таблиці для поточного стану та вибраної дії. Оновлення відбувається за правилом:

$$Q(s, a) \leftarrow Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a') - Q(s, a)),$$

де $\alpha$ – швидкість навчання, який визначає, наскільки сильно буде оновлюватись значення Q-таблиці, $r$ – винагорода за виконану дію, $\gamma$ – коефіцієнт знецінювання, який визначає важливість майбутніх винагород, $\max_{a'} Q(s', a')$ – максимальне значення Q-функції для наступного стану $s'$, яке агент може отримати, якщо виконає дію $a'$.

Приклад: Нехай агент знаходиться у деякому стані $s$, виконує дію $a$, за що отримує від середовища винагороду $r = 5$ і переходить у новий стан $s'$. У новому стані $s'$ агент може виконати як і раніше кілька допустимих дій, але максимальне Q-значення принесе агенту лише одна дія $a'$. Нехай для нового стану $s'$ максимальне Q-значення становить 10. Якщо швидкість навчання $\alpha = 0.1$, а коефіцієнт знецінювання $\gamma = 0.9$, тоді оновлення виглядатиме так:

$$Q(s, a) \leftarrow Q(s, a) + 0.1 \cdot (5 + 0.9 \cdot 10 - Q(s, a))$$

Якщо поточне $Q(s, a) = 2$ , то:

$$Q(s, a) \leftarrow 2 + 0.1 \cdot (5 + 0.9 \cdot 10 - 2) = 3.2$$

6. **Тренування агента.** Агент проходить кілька епізодів (циклів), де намагається досягти мети, навчаючись на власних помилках і оновлюючи свою стратегію. Агент поступово навчиться визначати оптимальні Q-значення для кожної пари стан-дія.

## Оцінювання

Максимальна оцінка за виконання завдання – 10 балів.

## Здача завдання

Відправляйте завдання на перевірку сюди: https://cloud.comsys.kpi.ua/s/yMty3QrYN9Fd9cc

- потрібно надіслати відпрацьований блокнот, який буде містити реалізовані Вами функції:

  `Прізвище Ім'я_група_Maze.ipynb`

**Дедлайн:** 12 жовтня 2025 року о 23:59