

# Introduction to Artificial Intelligence

## Exercise 6: Reinforcement Learning

---

Teacher: I Made Wangiyana

15 May 2023

### 1. Exercise details

Create an implementation of the Q-Learning algorithm to solve a toy Reinforcement Learning problem. Use the environment provided from OpenAI gym library. The original gym library is no longer updated, however, there is a continued development on a fork of `gym` called `gymnasium`. Please use `gymnasium` for this exercise.

Use the following environments for each lab variants:

1. Variant 1: `CartPole`. Use `"CartPole-v1"`
2. Variant 2: `MountainCar`. Use `"MountainCar-v0"`
3. Variant 3: `CliffWalking`. Use `"CliffWalking-v0"`
4. Variant 4: `Taxi`. Use `"Taxi-v3"`
5. Variant 5: `FrozenLake`. Use the arguments `"FrozenLake-v1"` and `map_name="8x8"` for `gym.make()`

Read each environment documentation to learn the problem, how to reach the goal, what are the possible actions, what states to observe, rewards and termination conditions.

### 2. Submission

To complete this exercise, students should submit a solution consisting of:

1. Code:
  - Training, more or less contains:
    - Initialization of the Q-table, and function for updating it
    - Training loop: initialize hyperparameters for the Q-Learning algorithm. perform iterative training of taking an action, observing the state, and updating the Q-table
    - Log training metrics (e.g. episode reward) to visualize improvements.
  - Inference. Utilizing the updated Q-table, visualize how the agent solves the environment.
2. Report:
  - Experiment with different initialization values and compare the training metrics (e.g. total reward, convergence rate, episode length, etc)
  - Report interesting findings from your experiments

### 3. Technical details

1. Write your solution using Python.
2. Adhere to clean coding standards and please provide comments to help readability

3. Provide instructions on how to run your code.
4. Do not use Machine Learning or optimization libraries. The `gymnasium` library along with `numpy` and a visualization library is enough to do this exercise
5. Questions related to the exercise and report should be asked in the Lab6 channel
6. If you have trouble running or installing the `gymnasium` library on your local machine, please try to search a solution using the hints provided in the error messages. If you tried your best and still could not solve, then try explaining the problem in the Lab6 channel on MS Teams and I'll try to help.

## 4. Additional information

### 4.1 References

- [Introduction to Q-Learning](#)
- To help with visualization, check out [environment wrappers](#) from gymnasium.

### 4.2 Libraries

In the following code snippet, you can find the minimum libraries needed to perform this exercise. You can [create a python virtual environment using conda](#) by saving the following snippet into `requirements.yml` and running `conda env create -f environment.yml`.

```
name: rl-env
channels:
  - conda-forge
  - defaults
dependencies:
  - gymnasium
  - matplotlib
  - moviepy
  - numpy
  - python=3.9
```

Additionally, it will ask to pip install `pygame` dependencies to render a gym env.

### 4.3 Compatibility: gym vs gymnasium

when calling `env.step()` in `gym` it returns 4 variables:

```
observation (ObsType), reward (float), done (bool), info (dict)
```

in `gymnasium`, the variable `done` was deprecated and replaced by `terminated` and `truncated`. it now returns:

```
observation (ObsType), reward (float), terminated (bool), truncated (bool), info (dict)
```

## 4.4 Starter code

Students can use the following starter code to get familiar with `gymnasium`. It creates one of the environment, the `MountainCar`, assigns a random action until the max steps are reached, and renders one of the episode (saved as `.mp4`).

```
import gymnasium as gym

if __name__ == '__main__':
    # init env
    env = gym.make('MountainCar-v0',
                   render_mode='rgb_array')

    # wrapper to record the video at 3rd episode and saves it to the folder 'vid'
    env = gym.wrappers.RecordVideo(env,
                                   video_folder='vid',
                                   episode_trigger=lambda x: x==3)

    # an episode ends if goal is reached or other game ending factors (e.g.
    # reached max steps)
    n_episodes = 4
    for episode in range(n_episodes):    # iterate episodes
        state, info = env.reset()        # reset the env to an initial state
        done = False                    # boolean to stop an episode

        while not done:                 # iterate steps
            # randomly choose a sample
            action = env.action_space.sample()
            # take the action (step) and observe the state and reward
            next_state, reward, terminated, truncated, info = env.step(action)
            # condition to stop an episode
            done = terminated or truncated

    env.close()
```