Обучение с подкреплением

Обучение с подкреплением

Обучение человека, как правило, осуществляется путем проб и ошибок: человек предпринимает попытки решить задачу, взаимодействуя с окружающим миром, и как-то улучшая своё поведение на основе полученного в ходе этого взаимодействия опыта.

Обучение с подкреплением (reinforcement learning, RL) моделирует обучение методом проб и ошибок. Вместо обучающей выборки рассматривается взаимодействие с некоторой средой (environment) или окружающим миром, а в роли разметки выступает награда (reward) — скалярная величина, которая выдаётся после каждого шага взаимодействия со средой и показывает, насколько хорошо алгоритм справляется с поставленной ему задачей.

Награда: не показывает, что и как нужно делать, а показывает, на сколько тот или иной результат хороший/плохой; может быть отложенной во времени, разреженной.

Субъект, взаимодействующий со средой и влияющий на неё, называется агентом (agent).

Основные характеристики ОП

- вознаграждение может быть плотным, разреженным или приходить с большим запаздыванием. Во многих случаях вознаграждение становится известно только в конце задания (например, в шахматах);
- задача последовательная и зависящая от времени – выбранные действия могут влиять на следующие действия, которые, в свою очередь, влияют на возможные вознаграждения и состояния;
- агент должен совершать действия, которые с высокой вероятностью приведут к достижению цели (использование), но в то же время должен пробовать иные действия, чтобы другие части окружающей среды не остались неисследованными (исследование). Эту двойственность называют дилеммой (или компромиссом) **исследования–использования**, она призвана решить трудную проблему поиска баланса между исследованием и использованием окружающей среды. Она важна также и потому, что, в отличие от обучения с учителем, агент ОП может влиять на окружающую среду, т. к. вправе собирать новые данные, коль скоро считает это полезным;
- окружающая среда стохастическая и недетерминированная, и агент должен учитывать это в процессе обучения и для предсказания следующего действия. Многие компоненты ОП можно спроектировать так, что они будут выдавать либо только одно детерминированное значение, либо диапазон значений, каждому из которых сопоставлена вероятность.

Постановка задачи

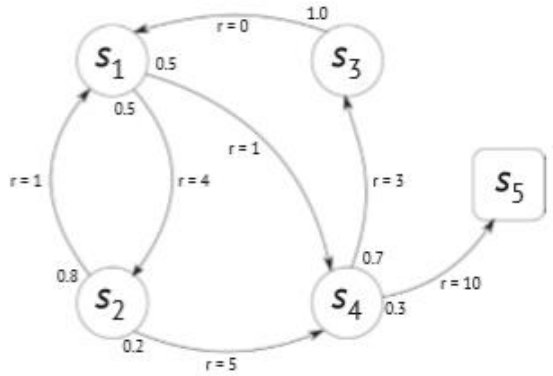
Задача обучения с подкреплением задаётся Марковским процессом принятия решений (MDP) - это четвёрка (S, A, P, r), где:

S — пространство состояний (state space), множество состояний, в которых в каждый момент времени может находится среда.

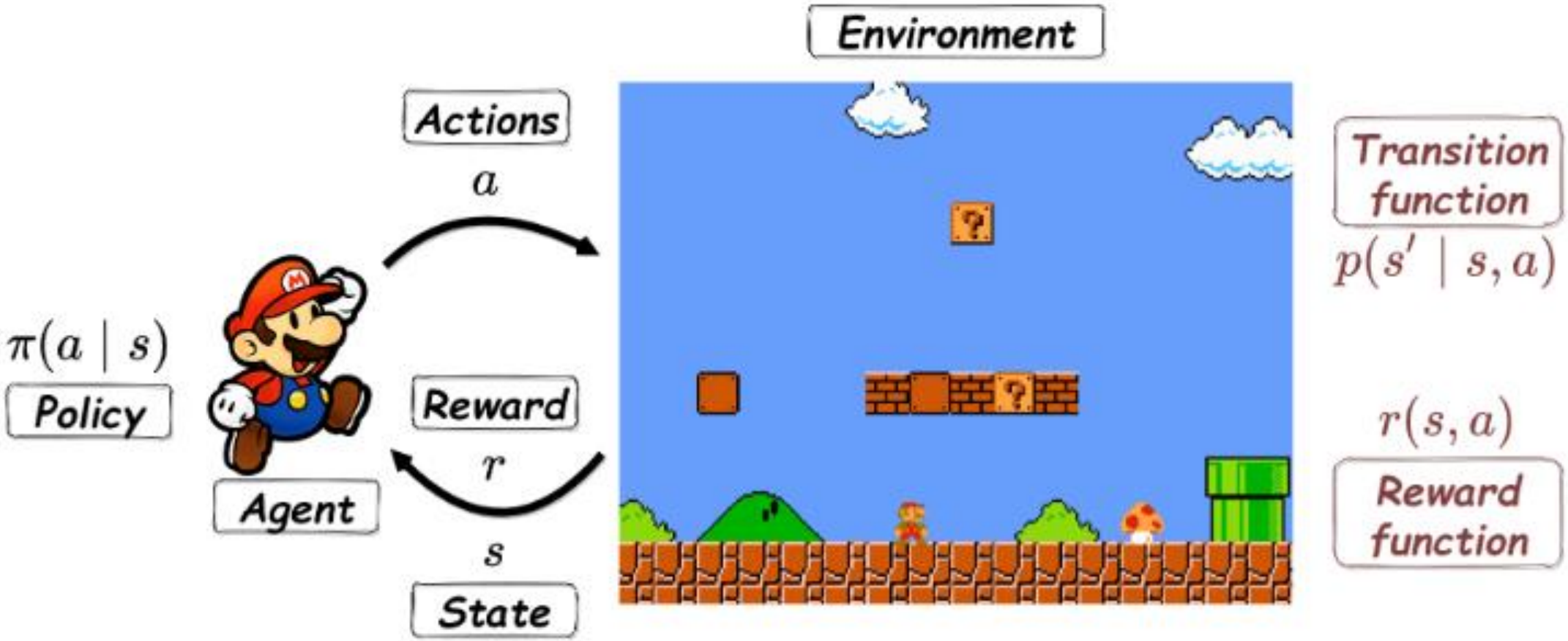
A — пространство действий (action space), множество вариантов, из которых нужно производить выбор на каждом шаге своего взаимодействия со средой.

P — функция переходов (transition function), которая задаёт изменение среды после того, как в состоянии $s \in S$ было выбрано действие $a \in A$. В общем случае функция переходов может быть стохастична, и тогда такая функция переходов моделируется распределением $p(s'|s,a)$: с какой вероятностью в какое состояние перейдёт среда после выбора действия *a* в состоянии *s*.

$r: S \times A \to \mathbb{R}$ — функция награды (reward function), выдающая скалярную величину за выбор действия *a* в состоянии *s*. Это наш «обучающий сигнал».



Взаимодействие агента со средой



$$\mathbb{E}_{\mathcal{T}\sim\pi}\sum_{t\geq 0} r_t \to \max_{\pi}$$

$$\mathbb{E}_{\mathcal{T}\sim\pi}\sum_{t\geq 0} \gamma^t r_t \to \max_{\pi}$$

Связь с динамическим программированием

$$Q^*(s,a) = \max_{\pi} \mathbb{E}_{\mathcal{T}\sim\pi|s_0=s,a_0=a} \sum_{t\geq 0} \gamma^t r_t$$

*Q**(*s*, *a*) - это то, сколько максимально награды можно (в среднем) набрать после выбора действия *a* из состояния *s*. Это оптимальная Q-функция.

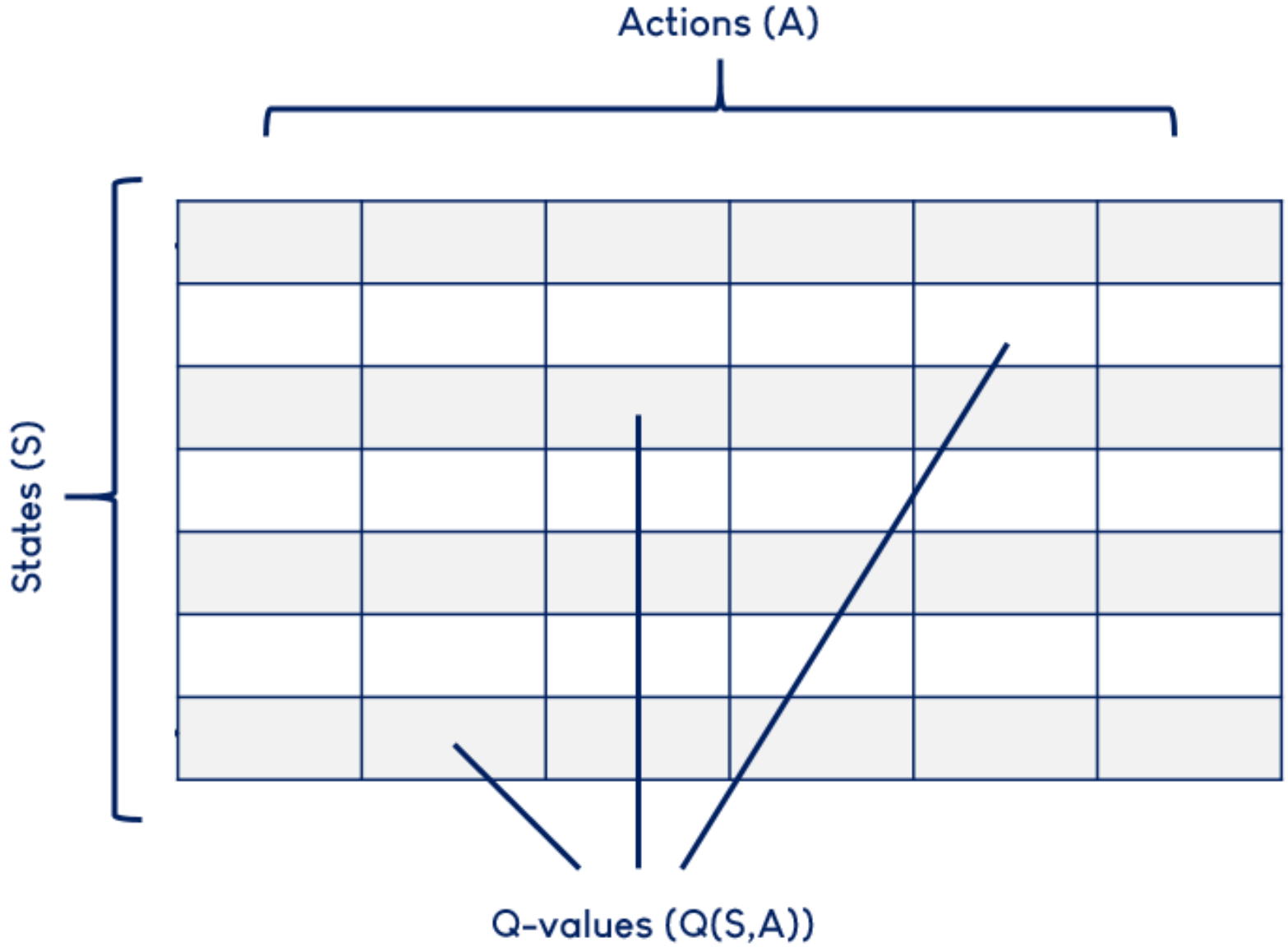
Принцип оптимальности Беллмана используется для выбора *a* по *Q**(*s*, *a*).

Уравнение оптимальности Беллмана для Q-функции:

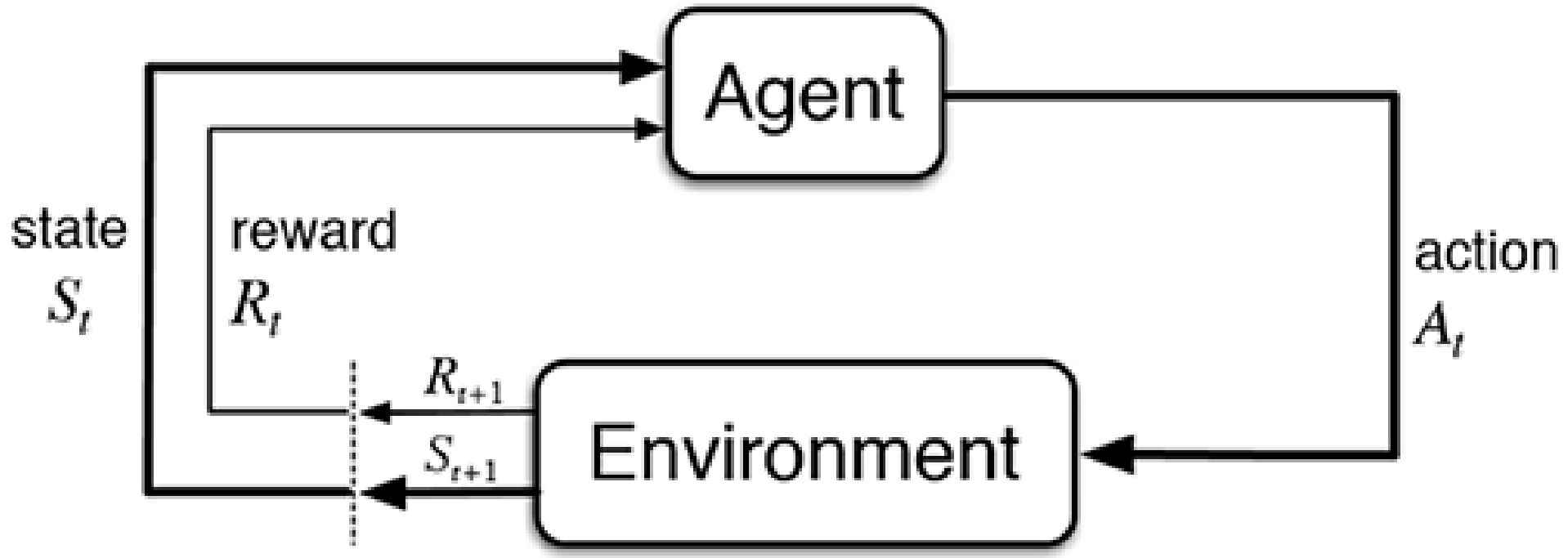
$$Q^*(s,a) = r(s,a) + \gamma \mathbb{E}_{s'\sim p(s'|s,a)} \max_{a'} Q^*(s',a')$$

$$Q^*_{k+1}(s,a) \leftarrow r(s,a) + \gamma \mathbb{E}_{s'\sim p(s'|s,a)} \max_{a'} Q^*_k(s',a')$$

Q-таблицы

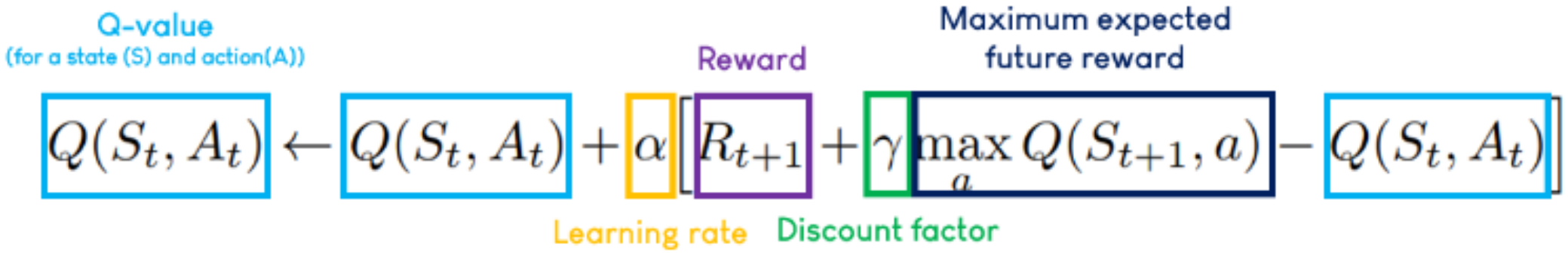


Алгоритм Q-обучения



Алгоритм Q-обучения

Q-learning — это алгоритм обучения с подкреплением, который стремится найти наилучшее возможное следующее действие с учетом его текущего состояния, чтобы максимизировать получаемое вознаграждение («Q» означает качество, т. е. насколько ценно действие).



Уравнение также содержит два гиперпараметра:

Скорость обучения (α): насколько легко агент должен воспринимать новую информацию по сравнению с ранее полученной информацией.

Коэффициент дисконтирования (γ): насколько агент должен учитывать вознаграждения, которые он может получить в будущем, по сравнению с его немедленным вознаграждением.

Библиотека gym (gymnasium)

Базовый цикл обучения с подкреплением в gym:

```
import gym
env = gym.make("Name_environment") # создать окружающую среду
env.reset() # привести среду в исходное состояние перед началом (инициализировать)
for i in range(10):  # выполнить 10 ходов
    action = env.action_space.sample() # выбрать случайное действие
    env.step(action) # выполнить случайное действие
    env.render() # нарисовать состояние среды
env.close() # закрыть окружающую среду
```

---

env.observation_space – тип и размерность пространства наблюдений

env.action_space – размерность пространства действий

env.observation_space.low/high – минимальное/максимальное допустимые значения в пространстве наблюдений

Цикл обучения с подкреплением в gym

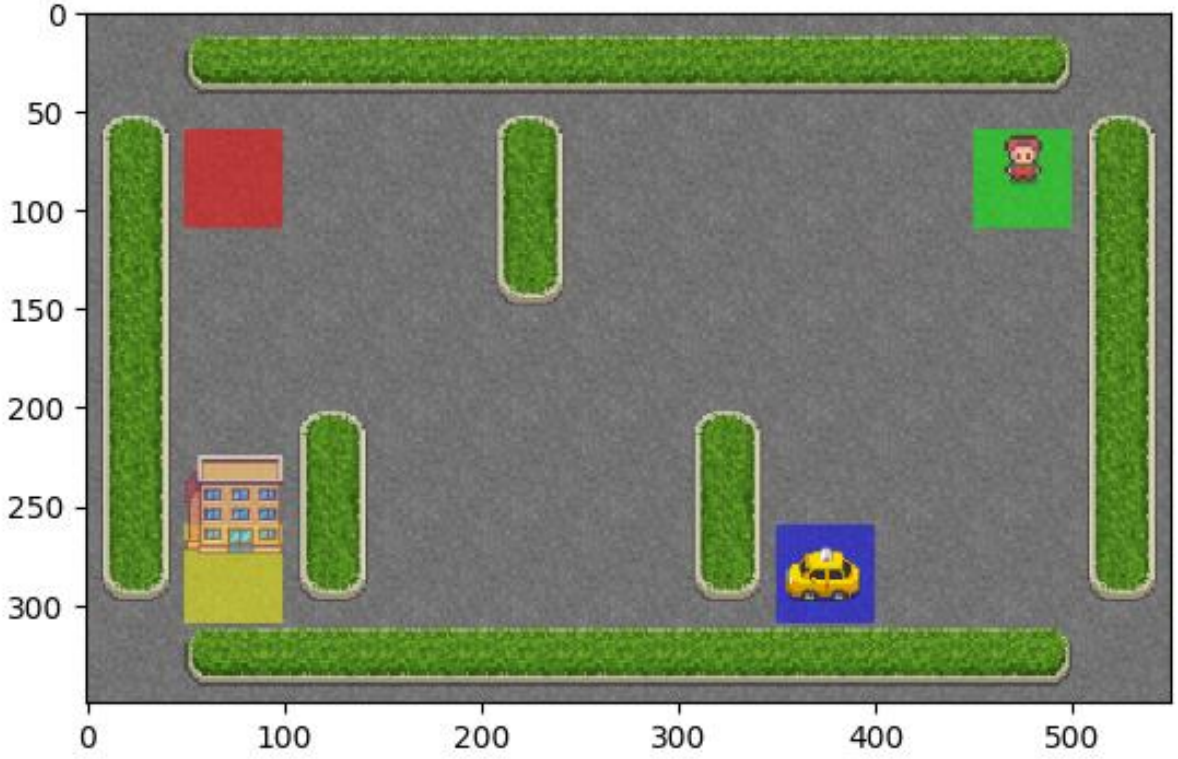
```
import gym
env = gym.make("Name_environment")
env.reset()
for i in range(10): # сыграть 10 игр
    done = False
    game_rew = 0
    while not done:
        action = env.action_space.sample()
        new_obs, rew, done, info = env.step(action) # выполнить один шаг взаимодействия с ОС
        game_rew += rew
        if done: # если завершено, напечатать полное вознаграждение в игре и сбросить среду
            print('Эпизод %d завершен, Вознаграждение: %d' % (i, game_rew))
            env.reset()
```

Визуализация (в colab)

```
!apt-get install python-opengl -y
!apt install xvfb -y
!pip install pyvirtualdisplay
!pip install piglet

import matplotlib.pyplot as plt
%matplotlib inline
from IPython import display
from pyvirtualdisplay import Display
vdisplay = Display(visible=0, size=(400, 300))
vdisplay.start()

prev_screen = env.render(mode='rgb_array')
plt.imshow(prev_screen)
```

```
    # при обновлении данных
    screen = env.render(mode='rgb_array')
    plt.imshow(screen)
    display.display(plt.gcf())
    display.clear_output(wait=True)

display.clear_output(wait=True)
vdisplay.stop()
```

Taxi-v3 в gym



+20 очков за успешную высадку;
-1 очко за каждый шаг времени;
-10 очков за ошибочную посадку и высадку.

Исследование – выполнение случайных действий и наблюдение за полученной наградой. Вычисление Q-значений. Заполнение Q-таблицы полученными Q-значениями.
Использование – получение Q-Значений из Q-таблицы.

Обучение агента.
Тестирование агента.

Характеристики окружающих сред

- Сложность. Спектр окружающих сред широк: от балансирования стержня до манипуляций физическими предметами с помощью роботизированной руки. Чтобы продемонстрировать способность алгоритма справляться с большим пространством состояний, имитирующим сложность реального мира, можно выбрать более сложную окружающую среду. С другой стороны, если нужно продемонстрировать лишь некоторые конкретные качества, то достаточно среды попроще.
- Пространство наблюдений. Пространство наблюдений может варьироваться от полного состояния окружающей среды до частичного наблюдения, доступного системе восприятия, например изображения, состоящего из строк пикселей.
- Пространство действий. Среды с большим непрерывным пространством состояний вынуждают агента иметь дело с вещественными векторами, тогда как в случае дискретных действий обучиться проще, поскольку количество действий ограничено.
- Функция вознаграждения. Среды с большим объемом исследования и отложенным вознаграждением с трудом поддаются решению. Лишь немногие алгоритмы способны достичь уровня человека. Поэтому такие среды используются как испытательный стенд для алгоритмов, предназначенных для решения проблемы исследования.

Некоторые окружающие среды

- Gym Atari (https://gym.openai.com/envs/#atari). Включает игры для Atari 2600, в которых входными данными служат изображения на экране. Полезны для измерения качества алгоритмов ОП на широком спектре игр с одним и тем же пространством наблюдений.
- Gym Classic control (https://gym.openai.com/envs/#classic_control). Классические игры, которые можно использовать для оценки и отладки алгоритма.
- Gym MuJoCo (https://gym.openai.com/envs/#mujoco). Включает непрерывные задачи управления (например, Ant и HalfCheetah), построенные поверх MuJoCo, физического движка с платной лицензией (бесплатная лицензия доступна студентам).
- MalmoEnv (https://github.com/Microsoft/malmo). Среда, построенная поверх Minecraft.
- Pommerman (https://github.com/MultiAgentLearning/playground). Отличная среда для обучен ия многоагентных алгоритмов. Игра Pommerman – вариант знаменитой игры Bomberman.
- Roboschool (https://github.com/openai/roboschool). Среда эмуляции роботов, интегрированная с OpenAI Gym. Включает копию окружающей среды MuJoCo, две интерактивные среды для повышения надежности агента и одну многопользовательскую среду.
- Duckietown (https://github.com/duckietown/gym-duckietown). Эмулятор беспилотного автомобиля с различными картами местности и препятствиями.
- PLE (https://github.com/ntasfi/PyGame-Learning-Environment). Включает много аркадных игр, в т. ч. Monster Kong, FlappyBird и Snake.
- Unity ML-Agents (https://github.com/Unity-Technologies/ml-agents). Среды, построенные поверх Unity с реалистичными физическими законами. ML-agents допускает большую свободу и позволяет создавать собственные окружающие среды с использованием Unity.
- CoinRun (https://github.com/openai/coinrun). Среда, предназначенная для решения проблемы переобучения в ОП. Генерирует различные среды для обуче ния и тестирования.
- DeepMind Lab (https://github.com/deepmind/lab). Комплект трехмерных сред для решения задач навигации и сборки пазлов.
- DeepMind PySC2 (https://github.com/deepmind/pysc2). Среда для обуче ния сложной игре, StarCraft II.

Примеры обучения с подкреплением в gym

- https://www.gocoder.one/blog/rl-tutorial-with-openai-gym/
- https://gym.openai.com/envs/Taxi-v3/
- https://github.com/bmaxdk/OpenAI-Gym-Taxi-v3