ЦЕНТР МОЛОДЁЖНОЙ
РОБОТОТЕХНИКИ
МГТУ ИМ. Н.Э. БАУМАНА

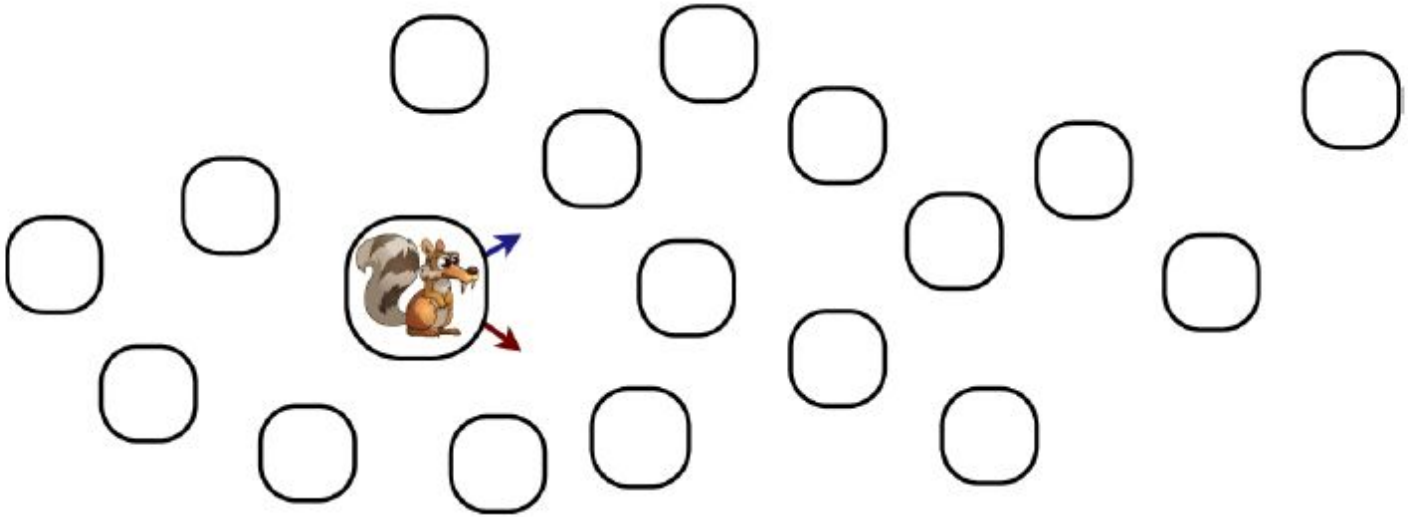
RL environment

Еще раз про среду

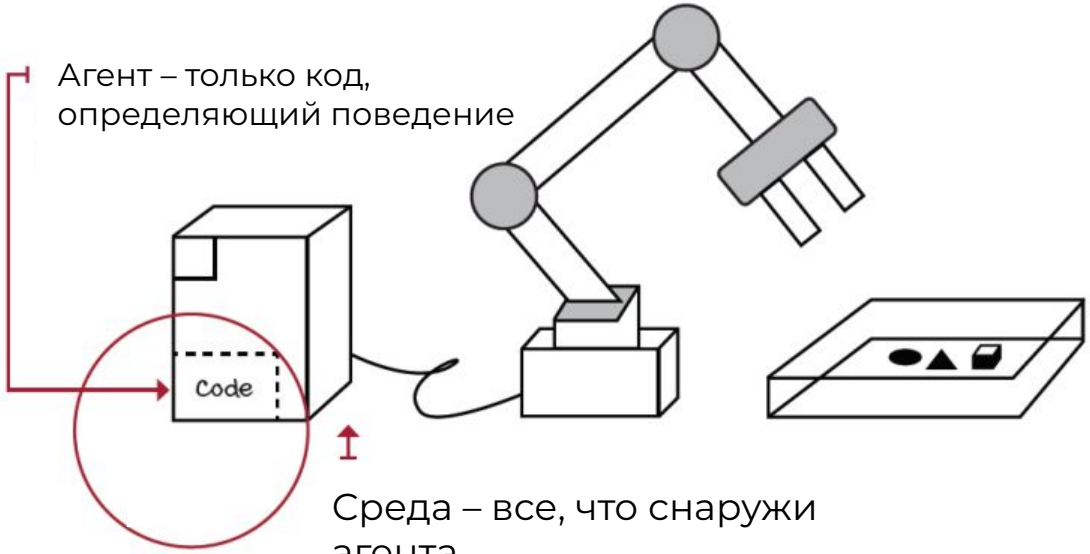
Среда – совокупность **(S,A,P,R)**, где

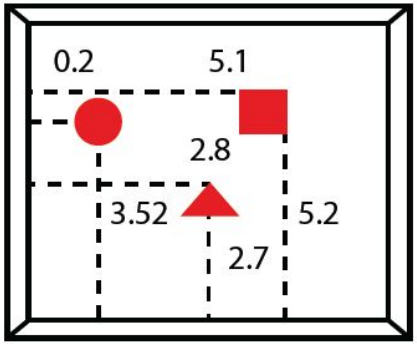
- ***S*** – пространство состояний
- ***A*** – пространство действий
- ***P*** – функция вероятности переходов $p(s' \mid s, a)$
- ***R*** – функция вознаграждения

Необходимо найти

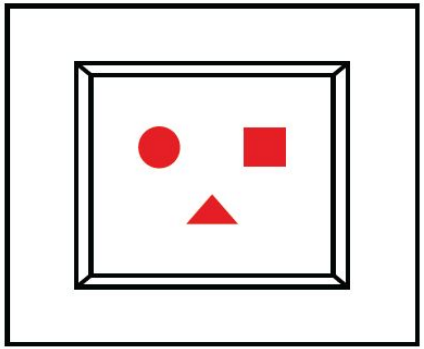


Состояние и наблюдение





Состояние



Наблюдение

РЛ для прохождения видеоигр

- Наблюдение: изображение(я)
- Действие: движение, стрельба, поворот

- Обратная связь: счёт/здоровье/прогресс/…

РЛ для управления вертолётами

- Наблюдение: акселерометр, гироскоп, данные двигателя
- Действие: изменение скорости вращения, угла

- Обратная связь: успешность прохождения трассы / близость к цели и т. д. (зависит от задачи)

РЛ для прохождения гоночных трасс дроном

Autonomous Drone Racing with Deep Reinforcement Learning
https://arxiv.org/abs/2103.08624 (2021)

Champion-level drone racing using deep reinforcement learning
https://www.nature.com/articles/s41586-023-06419-4 (2023)

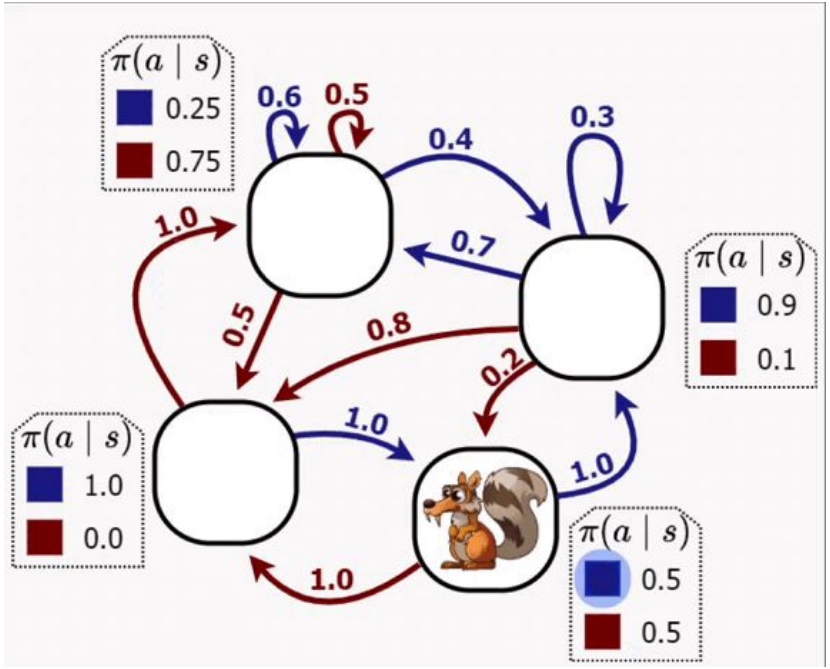
- Полностью наблюдаемая среда (MDP)

Выполняется марковское свойство: будущее зависит только от текущего состояния.

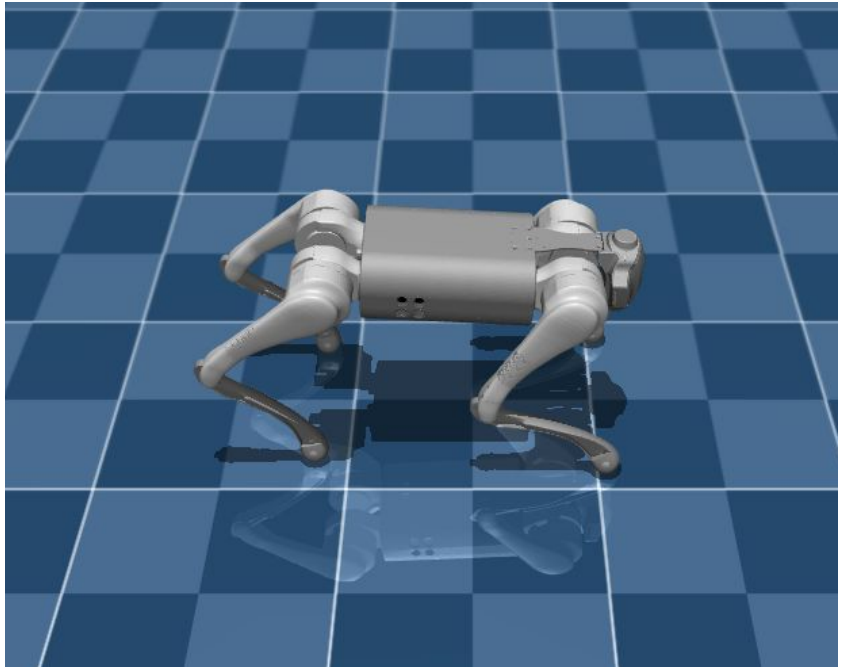
- Частично наблюдаемая среда (PoMDP)

Марковское свойство нарушено → оптимальная политика зависит от истории.

- Дискретное пространство



- Непрерывное пространство



Выбор алгоритма обучения

- Дискретные действия

  DQN

- Дискретные действия + параллельность

  PPO или A2C

- Непрерывные действия

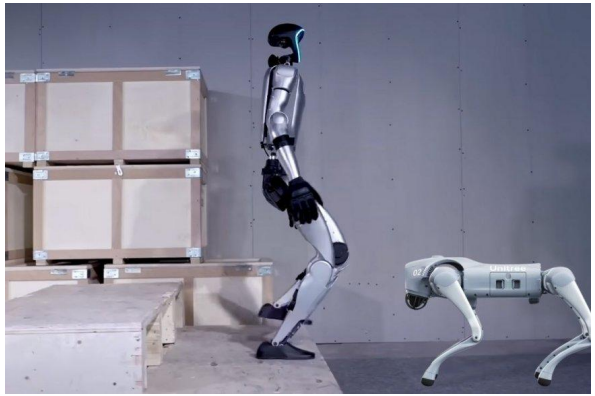
  SAC, TD3, и TQC

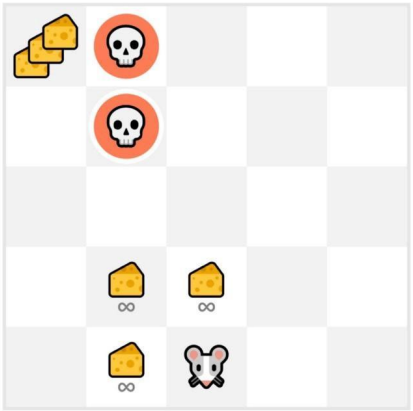
- Непрерывные действия + параллельность

  SAC, TD3, CrossQ или TQC

Более подробно про выбор алгоритма: https://stable-baselines3.readthedocs.io/en/master/guide/algos.html

Дизайн функции награды

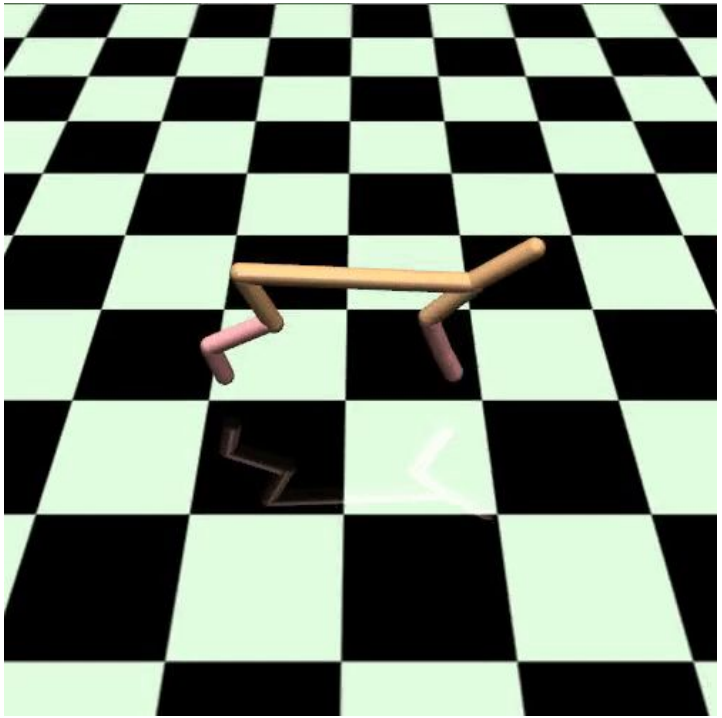




Успешный дизайн награды позволяет выполнять действия, в которых мы не можем полностью формализовать финальный результат, но можем направлять агента

Возможные проблемы при разработке среды

- Неинформативные наблюдения
- Разреженность награды
- ”Взлом” награды



Создание своей среды (environment)

🎯 Какому навыку должен научиться агент?

- Проходить через лабиринт?
- Баланс и управление системой?
- Оптимизировать распределение ресурсов?

👀 Какая информация нужна агенту?

- Положение и скорость?
- Текущее состояние системы?
- Исторические данные?
- Частичная или полная наблюдаемость?

🎮 Какие действия может предпринять агент?

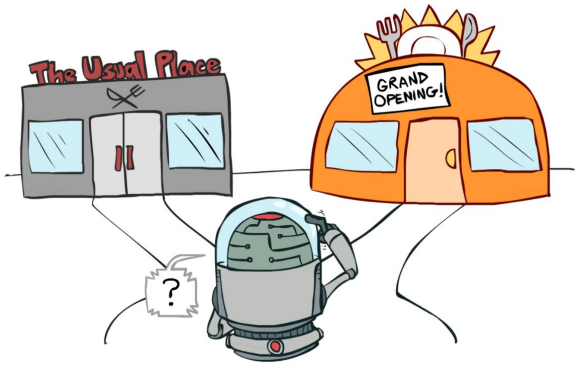
- Дискретный выбор (перемещение вверх/вниз/влево/вправо)?
- Непрерывное управление (угол поворота)?
- Несколько одновременных действий?

🏆 Как мы измеряем успех?

- Достижение конкретной цели?
- Минимизация времени или энергии?
- Максимизация оценки?

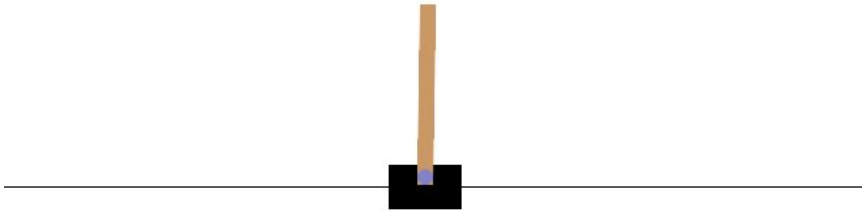
⏰ Когда должны закончиться эпизоды?

- Выполнение задачи (успех/неудача)?
- Ограничения по времени?
- Ограничения безопасности?



Cartpole

- 🎯 Навык: сохранить маятник на тележке в равновесии
- 👀 Информация: наклон маятника и смещение тележки
- 🎮 Действия: движение тележки вправо или влево
- 🏆 Успех: Сохранение равновесия как можно дольше
- ⏰ Окончание: падение маятника или достижение максимального времени



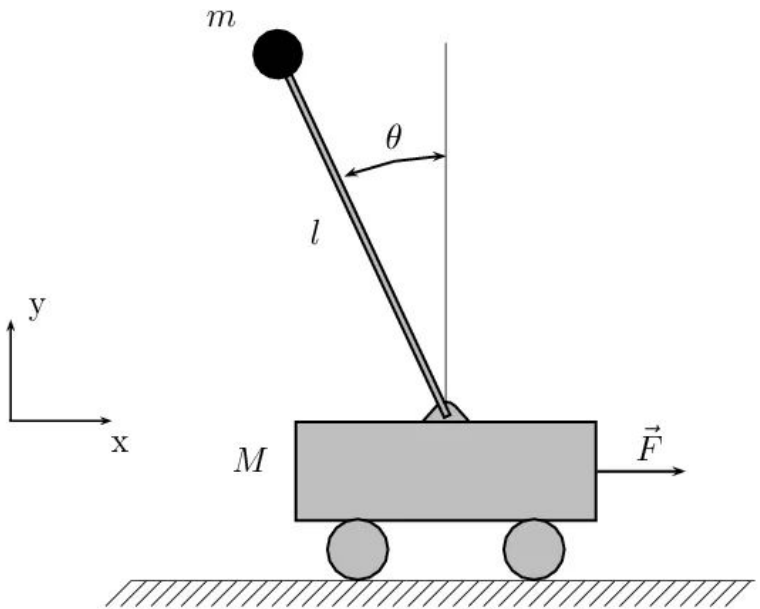
Cartpole

Пространство наблюдений:

- Позиция тележки
- Скорость тележки
- Угол отклонения шеста
- Угловая скорость шеста

Пространство действий:

- 0: Подтолкните тележку влево
- 1: Подтолкните тележку вправо



Создание среды

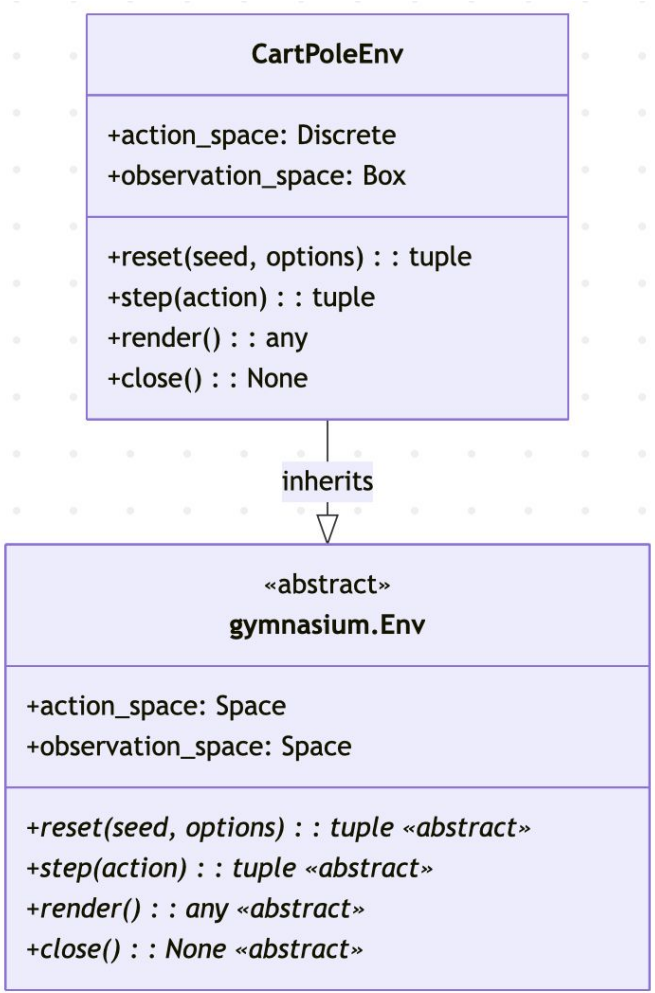
Наша пользовательская среда будет наследоваться от gymnasium.Env.

Нужно переопределить атрибуты:

- `self.observation_space` — наблюдения должны находиться в пределах этого пространства;
- `self.action_space` — действия должны находиться в пределах этого пространства;

И переопределить методы:

- `reset(self)` — сбрасывает среду в начальное состояние, требуется перед вызовом первого шага;
- `step(self, action)` — обновляет среду действием action, возвращая следующее наблюдение агента, награду и флаги окончания эпизода.



Пространства наблюдений и действий в Gymnasium

Тип пространства	Описание	Применение
Box	Непрерывное пространство произвольной размерности с заданными нижними и верхними границами.	Непрерывные действия (сила, угол), векторы состояния (позиция, скорость), пиксели изображения.
Discrete	Дискретное множество целых чисел {0, 1, …, n−1} .	Нажатие кнопок (влево/вправо), выбор пунктов меню, дискретные команды.
Dict	Словарь, в котором каждое значение — отдельное пространство (Box, Discrete и т.д.).	Структурированные состояния: {"position": Box, "inventory": Discrete, "status": MultiBinary}.
…		

```
# Для CartPoleEnv
action_space = spaces.Discrete(2)
observation_space = spaces.Box(-high, high, dtype=np.float32)
```

Шаг среды

`step(self, action)` → `(observation, reward, terminated, truncated, info)`

В методе step:

- Рассчитывается новое состояние среды (например, напрямую на основе физических уравнений, или с помощью симулятора)
- Проверяются условия окончания эпизода (терминальное состояние)
- Рассчитывается награда

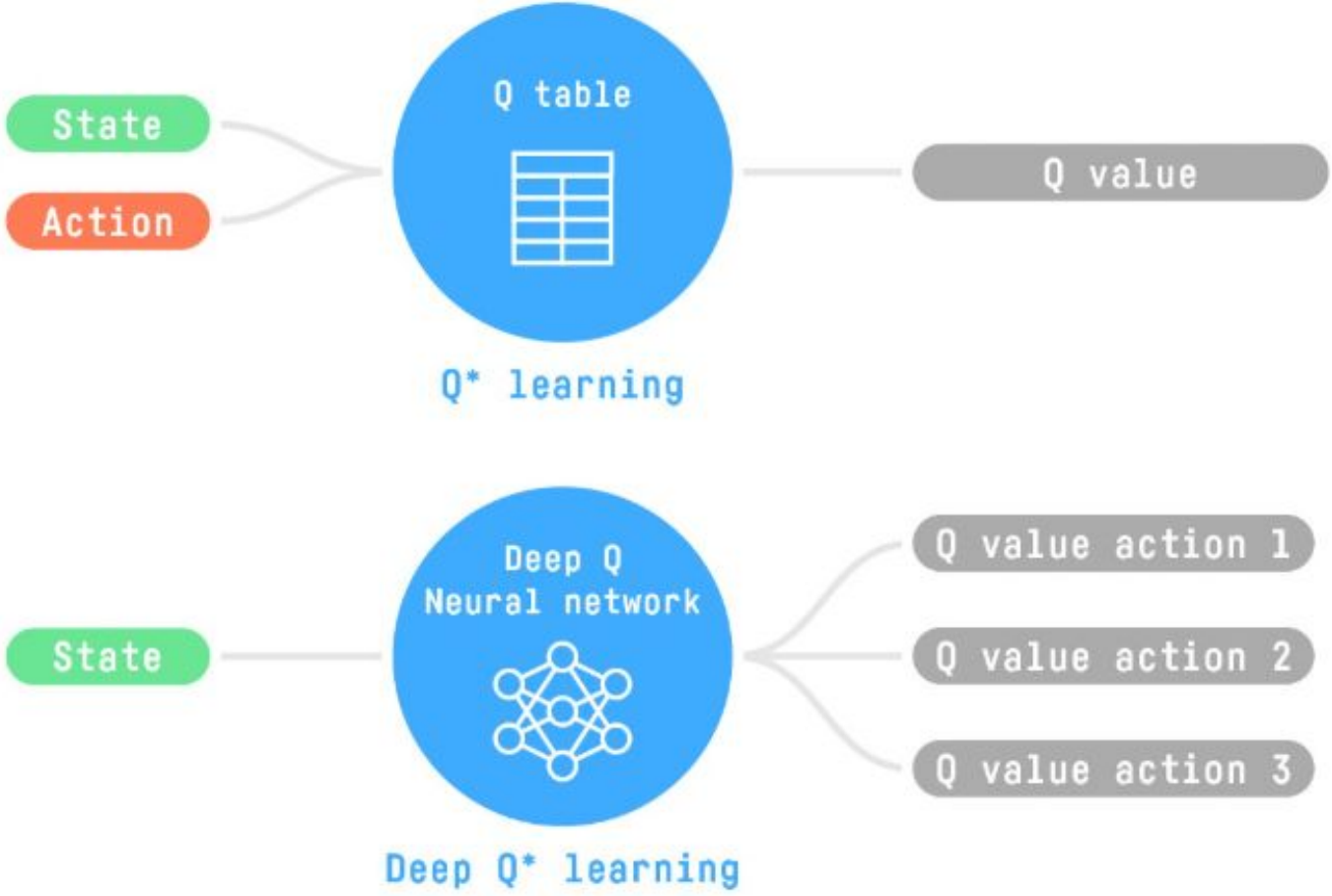
Сброс среды

`reset(self, seed=None, options=None)` → `(observation, info)`

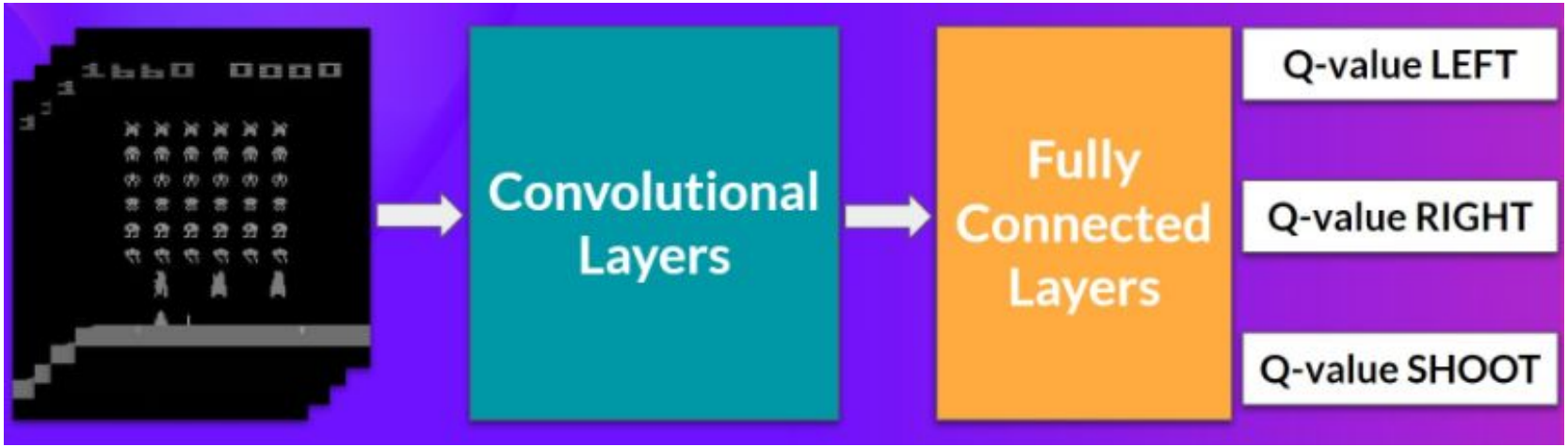
В методе reset:

- Список наблюдений сбрасывается в начальное состояние

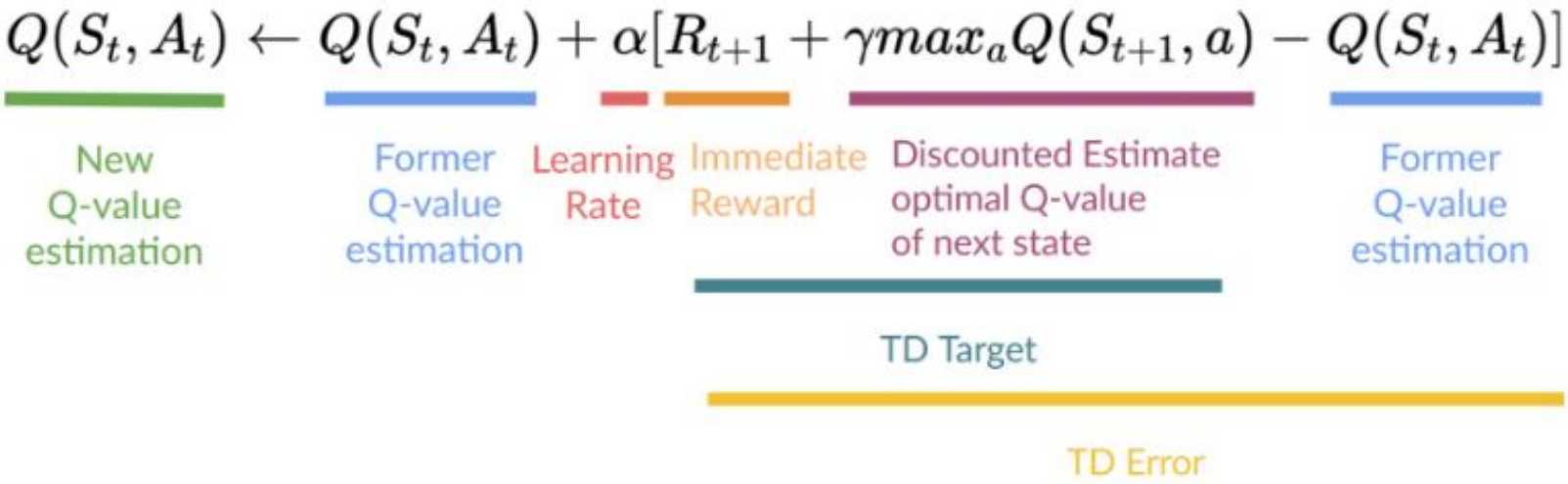
```
    # В данном случае (CartPoleEnv) начальное состояние —
    равномерные случайные значения (-0.05, 0.05)

    self.state = np.random.uniform(low=-0.05, high=0.05, size=(4,))
```


State
Action
Q table
Q value
Q* learning
State
Deep Q
Neural network
Q value action 1
Q value action 2
Q value action 3
Deep Q* learning

The Deep Q-Network (DQN)



Алгоритм обучения нейросети



Нормализация

Зачем нормализовать?
Большинство RL-алгоритмов предполагают гауссово распределение. Нормализация ускоряет обучение и стабилизирует градиенты.

Ситуация	Решение
Известны границы, распределение нормальное	Наблюдение / Max_Значение *Пример: координата тележки в CartPole*
Границы неизвестны, распределение неизвестно	**Z-score Running Norm** (x - running_mean) / running_std *(де-факто стандарт в RL)*
Большие выбросы (типично для ускорений)	Вариант 1: Клиппинг + Z-norm Вариант 2: Нелинейное преобразование (например, tanh)
Очень большие выбросы, содержащие важную информацию (нельзя клипать)	Только эксперименты (tanh не подойдёт, так как сожмёт основную массу данных)

Пространство действий

- **Способ 1:** Используйте `tanh` в качестве финальной функции активации в policy. Это автоматически дает диапазон `[-1, 1]` и корректные градиенты.
- **Способ 2:** Линейно масштабируйте выход сети в `[-1, 1]`. Для ограничения диапазона используйте `clip()`, но не сломайте градиенты.

Нормализация

Т. к. большинство алгоритмов обучения с подкреплением полагаются на гауссово распределение:

- всегда нормализуйте свое пространство наблюдений, если можете, т.е. если вы знаете границы
- нормализуйте свое пространство действий и сделайте его симметричным, если оно непрерывное

$$z = \frac{x - \mu}{\sigma}$$

```
    # метод обёртки (wrapper) NormalizeObservation

       def normalize(self, obs):
            """Нормализует наблюдение, используя скользящее среднее значение
    и дисперсию наблюдений"""
            self.obs_rms.update(obs)
            return (obs - self.obs_rms.mean) / np.sqrt(self.obs_rms.var +
    self.epsilon)
```