یاسمن توکلی: 98522175

توضیح الگوریتم Q-State:

Q-learning یک الگوریتم یادگیری تقویتی (Reinforcement Learning) است که به عامل (Agent) کمک می‌کند تا در یک محیط (Environment) بهینه‌ترین سیاست (Policy) را یاد بگیرد. هدف Q-learning به‌روزرسانی Q-جدول است که مقدار Q هر حالت-عمل (State-Action) را تخمین می‌زند. این مقادیر Q نشان‌دهنده انتظارات پاداش بلندمدت از انجام یک عمل خاص در یک حالت خاص هستند.

فرمول:

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

فرمول محاسبه امتیاز:

```python
# the formula for the reward system
if is_training:
    q[state, action] = q[state, action] + learning_rate_a * (
        reward + discount_factor_g *
        np.max(q[new_state, :]) - q[state, action]
    )
```

`q[state, action]` مقدار Q فعلی برای حالت `state` و عمل `action` است.

`reward` پاداش دریافت‌شده از محیط است.

`discount_factor_g` همان γ\gammaγ است که مقدارش 0.9 تعیین شده است.

`np.max(q[new_state, :])` بیشترین مقدار Q برای حالت جدید `new_state` است.

`learning_rate_a` همان α\alphaα است که مقدارش 0.9 تعیین شده است.

یاسمن توکلی: 98522175

1. نرخ یادگیری (α\alphaα)

نرخ یادگیری مشخص می‌کند که عامل تا چه حد باید دانش جدید را نسبت به دانش قبلی خود اولویت دهد. مقدار α\alphaα بین 0 و 1 قرار دارد:

- اگر α\alphaα نزدیک به 1 باشد، عامل تغییرات جدید را سریع‌تر یاد می‌گیرد.
- اگر α\alphaα نزدیک به 0 باشد، عامل بیشتر به دانش قبلی خود اعتماد می‌کند.

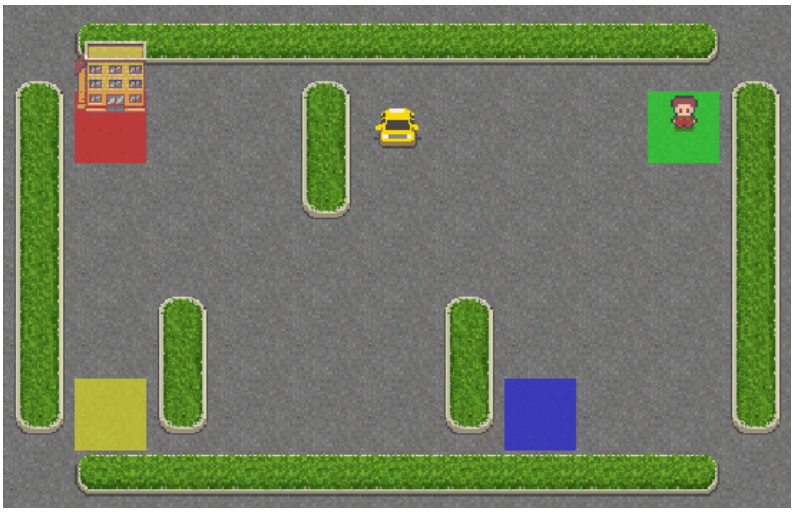
3. نرخ تنزیل (γ\gammaγ)

نرخ تنزیل تعیین می‌کند که چقدر عامل به پاداش‌های آینده اهمیت دهد. مقدار γ\gammaγ نیز بین 0 و 1 قرار دارد:

- اگر γ\gammaγ نزدیک به 1 باشد، عامل بیشتر به پاداش‌های بلندمدت اهمیت می‌دهد.
- اگر γ\gammaγ نزدیک به 0 باشد، عامل بیشتر به پاداش‌های کوتاه‌مدت اهمیت می‌دهد.

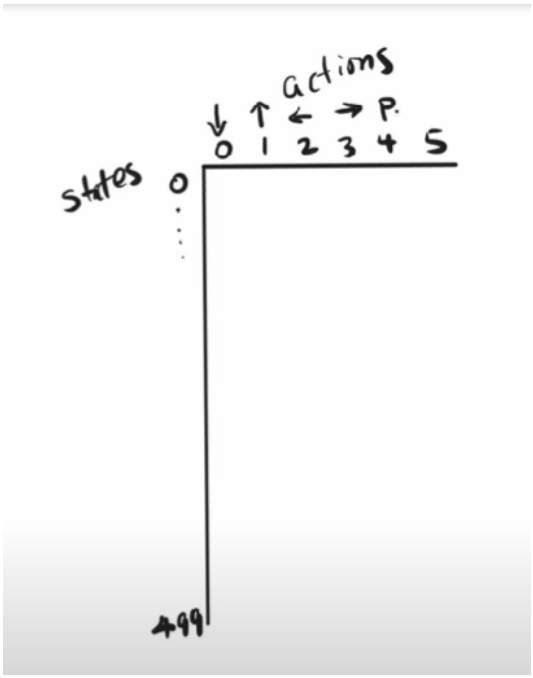
4. بیشینه مقدار

این مقدار نشان‌دهنده بالاترین ارزش Q برای حالت جدید s′s's′ و تمام اعمال ممکن a′a'a′ است. این بخش از فرمول به عامل کمک می‌کند تا ارزش بلندمدت حالت جدید را در نظر بگیرد.



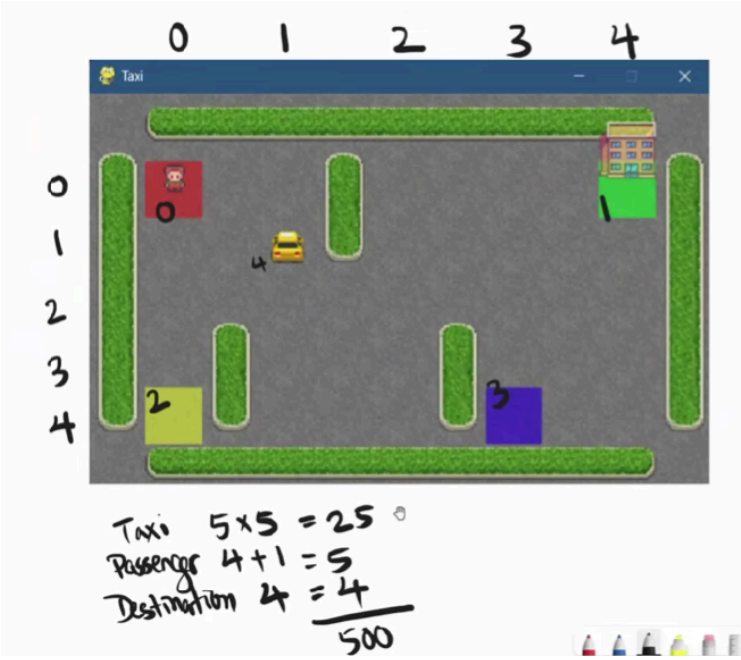
ماتریس ما ترکیبی از فضای حالت observation و action هست. که هرکدام طبق زیر میباشند.

Action Space	`Discrete(6)`
Observation Space	`Discrete(500)`
import	`gymnasium.make("Taxi-v3")`



فرمول محاسبه هر state به عنوان شماره ردیف در ارایه:

((taxi_row * 5 + taxi_col) * 5 + passenger_location) * 4 + destination



Action Space

The action shape is (1,) in the range {0, 5} indicating which direction to move the taxi or to pickup/drop off passengers.

- 0: Move south (down)
- 1: Move north (up)
- 2: Move east (right)
- 3: Move west (left)
- 4: Pickup passenger
- 5: Drop off passenger

Observation Space

There are 500 discrete states since there are 25 taxi positions, 5 possible locations of the passenger (including the case when the passenger is in the taxi), and 4 destination locations.

Destination on the map are represented with the first letter of the color.

Passenger locations:

- 0: Red
- 1: Green
- 2: Yellow
- 3: Blue
- 4: In taxi

Destinations:

- 0: Red
- 1: Green
- 2: Yellow
- 3: Blue

جایزه ها:

Rewards

- -1 per step unless other reward is triggered.
- +20 delivering passenger.

- -10 executing “pickup” and “drop-off” actions illegally.

An action that results a noop, like moving into a wall, will incur the time step penalty. Noops can be avoided by sampling the action_mask returned in info.

کد:

در تابع run، محیط بازی را اجرا و عامل را آموزش داده یا تست می‌شود.

مقداردهی اولیه Q-جدول:

```
if(is_training):
    q = np.zeros((env.observation_space.n, env.action_space.n)) # init a 500 x 6 array
else:
    f = open('taxi.pkl', 'rb')
    q = pickle.load(f)
    f.close()
```

اگر در حالت آموزش باشد، Q-جدول با صفر مقداردهی اولیه می‌شود.

اگر در حالت تست باشد، Q-جدول از فایل taxi.pkl بارگذاری می‌شود.

```
learning_rate_a = 0.9 # alpha or learning rate
discount_factor_g = 0.9 # gamma or discount rate. Near 0: more weight/reward placed
epsilon = 1         # 1 = 100% random actions
epsilon_decay_rate = 0.0001        # epsilon decay rate. 1/0.0001 = 10,000
rng = np.random.default_rng()   # random number generator
```

learning_rate_a: نرخ یادگیری.

discount_factor_g: نرخ تنزیل برای پاداش‌های آینده.

epsilon: مقدار اولیه برای سیاست اکتشافی.

epsilon_decay_rate: نرخ کاهش epsilon.

rng: تولید کننده اعداد تصادفی.

یاسمن توکلی: 98522175

```python
for i in range(episodes):

    state = env.reset()[0]
    terminated = False      # True when reached goal
    truncated = False       # True when actions > 400

    rewards = 0
    while (not terminated and not truncated):
        if is_training and rng.random() < epsilon:
            action = env.action_space.sample()
        else:
            action = np.argmax(q[state, :])

        new_state, reward, terminated, truncated, _ = env.step(action)

        rewards += reward

        # the formula for the reward system
        if is_training:
            q[state, action] = q[state, action] + learning_rate_a * (
                reward + discount_factor_g *
                np.max(q[new_state, :]) - q[state, action]
            )

        state = new_state
```

rewards_per_episode: آرایه‌ای برای ذخیره پاداش‌های هر قسمت.

برای هر قسمت:

- محیط بازی بازنشانی می‌شود.
- وضعیت اولیه و متغیرهای خاتمه و قطع شدگی تنظیم می‌شوند.
- در حالی که قسمت تمام نشده است:
  - اگر در حالت آموزش و با احتمال epsilon، عمل تصادفی انتخاب می‌شود.
  - در غیر این صورت، بهترین عمل بر اساس Q-جدول انتخاب می‌شود.
  - عمل در محیط انجام می‌شود و وضعیت جدید و پاداش دریافت می‌شود.
  - اگر در حالت آموزش باشد، Q-جدول با استفاده از قانون به‌روزرسانی Q یادگیری به‌روز می‌شود.
  - وضعیت به وضعیت جدید به‌روز می‌شود.

- epsilon کاهش می‌یابد.
- اگر epsilon به صفر برسد، نرخ یادگیری کاهش می‌یابد.
- پاداش‌های قسمت در rewards_per_episode ذخیره می‌شود.

```
env.close()

sum_rewards = np.zeros(episodes)
for t in range(episodes):
    sum_rewards[t] = np.sum(rewards_per_episode[max(0, t-100):(t+1)])
plt.plot(sum_rewards)
plt.savefig('taxi.png')

if is_training:
    f = open("taxi.pkl","wb")
    pickle.dump(q, f)
    f.close()
```

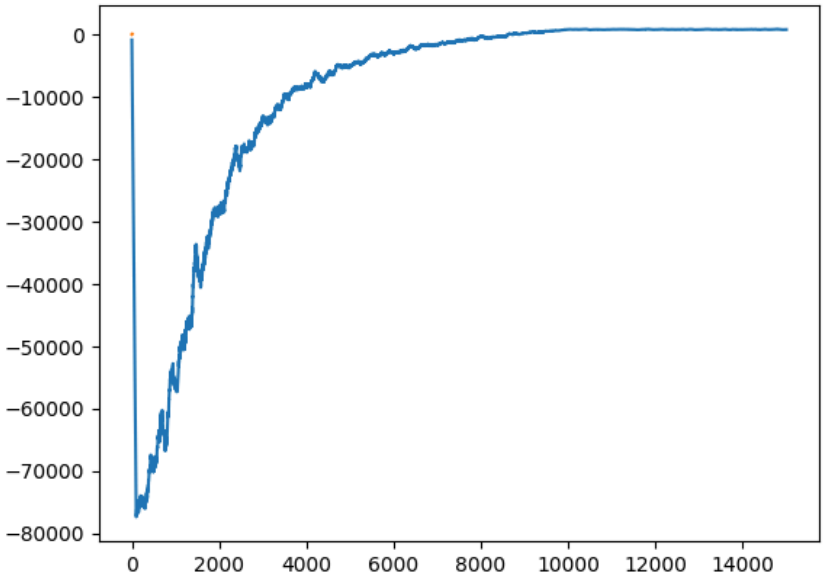
محیط بسته می‌شود و پاداش‌های قسمت‌ها جمع‌آوری و نمودار آن رسم و ذخیره می‌شود.

اگر در حالت آموزش باشد، Q-جدول ذخیره می‌شود.

نتایج:

در ویدیو

نمودار:



محور xxx

- **تعداد اپیزودها**: محور افقی xxx نشان‌دهنده تعداد اپیزودهای یادگیری است. هر اپیزود یک بار شروع و پایان بازی را نشان می‌دهد. همانطور که در کد مشخص شده، تعداد اپیزودها ۱۵۰۰۰ (15000) تعیین شده است.

محور yyy

- **مجموع پاداش‌ها**: محور عمودی yyy نشان‌دهنده مجموع پاداش‌هایی است که عامل در طول هر اپیزود به دست آورده است. پاداش‌ها می‌توانند مثبت یا منفی باشند و نشان‌دهنده موفقیت یا ناکامی عامل در انجام وظایف مختلف هستند.

تحلیل نمودار

- **شروع از مقدار پایین (منفی)**: در ابتدای آموزش، پاداش‌های عامل بسیار پایین هستند (حتی تا -80000). این نشان‌دهنده این است که عامل هنوز به خوبی یاد نگرفته است چگونه به طور بهینه عمل کند و به همین دلیل پاداش‌های زیادی را از دست می‌دهد.

- **افزایش پاداش‌ها در طول زمان**: با گذشت زمان و افزایش تعداد اپیزودها، مجموع پاداش‌ها به طور پیوسته افزایش می‌یابد و به سمت مقادیر مثبت می‌رود. این نشان‌دهنده این است که عامل به تدریج سیاست‌های بهینه‌تری را یاد می‌گیرد و عملکردش بهبود می‌یابد.
- **پایداری در مقادیر بالاتر**: در انتهای نمودار، مجموع پاداش‌ها به یک مقدار ثابت و بالاتر می‌رسد و نوسانات کمتری مشاهده می‌شود. این نشان‌دهنده این است که عامل به یک سیاست پایدار و بهینه دست یافته است و می‌تواند به طور مداوم عملکرد خوبی داشته باشد.