گزارش فاز اول پروژه

مبانی و کاربردهای هوش مصنوعی

دکتر حسین کارشناس

آدرینا ابراهیمی ۹۹۳۶۲۳۰۰۲

کیان مجلسی ۹۹۳۶۱۳۰۵۱

اردیبهشت ۱۴۰۲

نام و علت انتخاب الگوریتم

از الگوریتم تکرار ارزش (Value iteration) استفاده شده است که مطابق با شبه کد زیر بر اساس سودمندی مورد انتظار سیاست بهینه را پیدا می‌کند. به علت سادگی در پیاده‌سازی و همگرا شدن به یک جواب بهینه از این الگوریتم استفاده شده است.

V^*, Q^* = VALUE-ITERATION(*MDP*)
// MDP consist of States, Actions(s), $R(s,a,s')$, $T(s'|s,a)$, and discount factor γ
hyperparameters: *maxIters* //maximum number of steps in the environment

foreach state $s \in$ States **do**
$V^*(s) \leftarrow 0$
foreach action $a \in$ Actions(s) **do**
$Q^*(s,a) \leftarrow 0$
$t = 1$
while $t <$ *maxIters* **and not** (converged **or** terminated) **do**
foreach state $s \in$ States **do** // except terminal states
foreach action $a \in$ Actions(s) **do** // old values buffered
$Q^*(s,a) \leftarrow \sum_{s'} T(s'|s,a)\big(R(s,a,s') + \gamma V^*(s')\big)$
$V^*(s) \leftarrow \max_a Q^*(s,a)$
$t = t + 1$
return V^*, Q^*

گزارش کار الگوریتم

تابع value_iteration:

در این تابع عملیات اصلی الگوریتم ذکر شده انجام شده است. در خطوط ۷ تا ۱۶ دیکشنری‌های v_value (مقدار مورد انتظار سودمندی با شروع از هر حالت) و q_value (مقدار مورد انتظار سومندی با انجام هر کنش در هر حالت) مقداردهی اولیه شده‌اند و خانه‌های چاله مشخص شده‌اند.

در خط ۱۹ مقدار مورد انتظار سودمندی برای خانه هدف برای ۱ در نظر گرفته شده و در خط ۲۰ و ۲۱ این مقدار برای خانه‌های چاله برابر ۱- در نظر گرفته شده است.

سپس در یک حلقه به طول ۱۰۰۰۰ (برای همگرایی سیاست‌های ممکن به یک حالت بهینه)، یک حلقه روی تمام حالت‌های موجود در محیط زده و پس از آن، برای تمام کنش‌های موجود در آن حالت احتمال، پاداش و سودمندی مورد انتظار با شروع از حالت را برای نتیجه‌های ممکن آن کنش بررسی کرده و q_value را برای آن کنش در آن حالت محاسبه می‌کنیم. لازم به ذکر است پاداش خانه‌های چاله برابر ۱- ، پاداش خانه‌ی هدف برابر ۱ و پاداش سایر خانه‌ها با توجه به فاصله‌شان تا خانه هدف و تابع لجستیک مقداری بین ۰ و ۱- در نظر گرفته شده است. در نهایت مقدار مورد انتظار سومندی با شروع از آن حالت (v_value) و سیاستی که باید در پیش گرفته شود را آپدیت کرده و در صورت برقرار شدن شرط همگرایی از حلقه‌ها خارج می‌شویم.

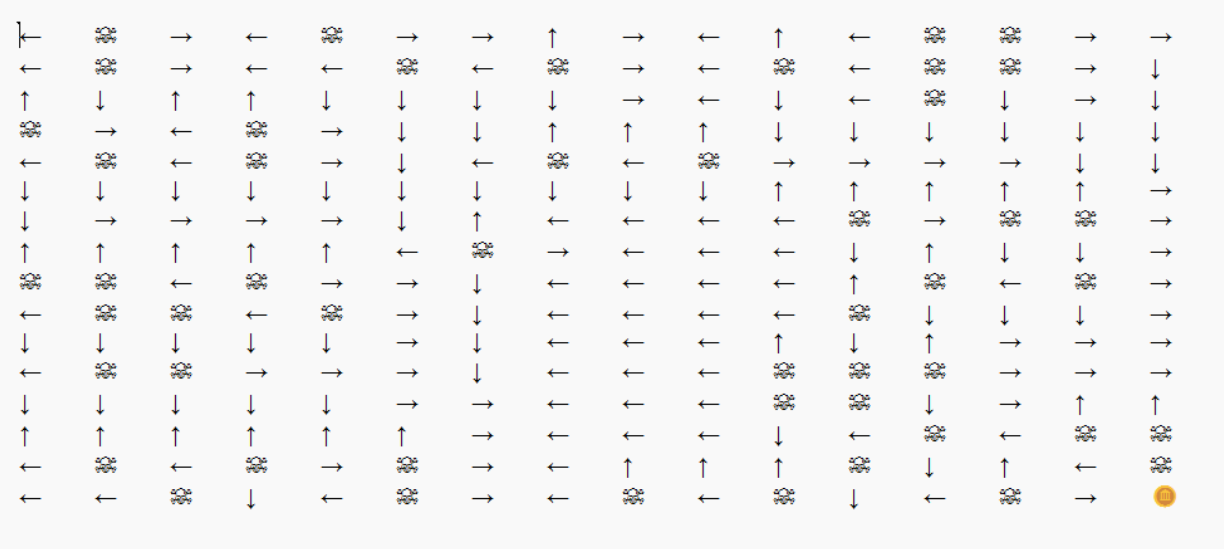
```python
def value_iteration():
    # Initialize
    v_values, q_values = {}, {}
    convergenceTrack = [0]

    # Initialize the value of each state to 0 and store the terminal states
    for state in env.P:
        if state == goal_state:
            continue
        v_values[state] = 0
        q_values[state] = {}
        for act in env.P[state]:
            q_values[state][act] = 0
            for probability, nextState, reward, isTerminalState in env.P[state][act]:
                if (reward == 0) and isTerminalState:
                    terminal_states.add(nextState)

    # Set the value of the goal state to 1 and the terminal states to -1
    v_values[goal_state] = 1
    for ts in terminal_states:
        v_values[ts] = -1

    for i in range(10000):
        # Check states in the environment
        for state in env.P:
            if (state not in terminal_states) and (state != goal_state):
                # Check actions in that state
                for act in env.P[state]:
                    s = 0
                    # Calculate every result of the action
                    for probability, nextState, reward, isTerminalState in env.P[state][act]:
                        # Calculate the reward of each actions
                        if (reward == 0) and isTerminalState:
                            reward = -1
                        elif (reward == 1) and isTerminalState:
                            reward = 1
                        else:
                            # Calculate the distance to the goal
                            x, y = nextState // map_size, nextState % map_size
                            dist_to_goal = np.sqrt(
                                np.power(x - (map_size - 1), 2) + np.power(y - (map_size - 1), 2))
                            reward = -0.1 / (1 + np.exp(-dist_to_goal))
                        s += probability * \
                            (reward + (discount_factor * v_values[nextState]))
                    # Update the q_value of state and action
                    q_values[state][act] = s
                # Update the v_value of state and policy
                v_values[state] = max(q_values[state].values())
                # Check convergence
                convergenceTrack.append(
                    np.linalg.norm(list(v_values.values())))
                if (i > 1000) and np.isclose(convergenceTrack[-1], convergenceTrack[-2]):
                    print('Values Converged')
                    return v_values, q_values
    return v_values, q_values
```

پس از محاسبه سیاست بهینه یک فایل که حاوی نقشه سیاست بهینه است ایجاد می‌شود که محتویات آن به شکل زیر می‌باشد.



منابع ایده

از منابع زیر برای آشنایی با نحوه پیاده‌سازی و الگو گرفتن اولیه استفاده شده است.

Stuart J. Russell, Peter Norvig - Artificial Intelligence_ A Modern Approach, Global Edition-Pearson (2021)

Frozen Lake - Gymnasium Documentation (farama.org)