Menentukan Jalur Efektif Dengan Menggunakan Metode Q-Learning

AYU IMAS SUSANTI

FIBONACCI

Latar Belakang Masalah

Reinforcement Learning merupakan algoritma machine learning yang dapat membuat agent software dan mesin bekerja secara otomatis untuk menentukan perilaku yang sempurna sehingga dapat memaksimalkan kinerja algoritmanya.

Pada saat akan melakukan perjalanan, baik sudah mengenal lingkungan maupun belum mengenal lingkungan, akan terdapat sebuah jalur yang lebih efektif atau lebih singkat. Dalam hal ini untuk menentukan jalur tersebut dapat menerapkan sebuah model dari Reinforcement Learning. Salah satu model yang dapat digunakan dalam kasus ini adalah model dari metode Q -Learning. Pada kasus ini Q-Learning menyelesaikan agent untuk mencari jalan tercepat dengan menggunakan pembelajaran secara ekplorasi dan menemukan jalan terbaik dengan eksploitasi.

Rumusan Masalah

Berapakah episode yang diperlukan untuk mencapai konvergen?

Tujuan

Untuk mengetahui episode yang diperlukan untuk mencapai konvergen pada model metode Q Learning.

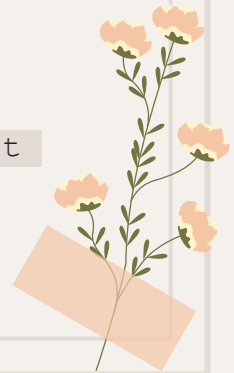
Teori

Q - Learning merupakan pengembangan RL yang menggunakan Q-values (disebut juga action-values) untuk meningkatkan kemampuan agent belajar agent berulang-ulang.

Konsep dasar Q-Learning:

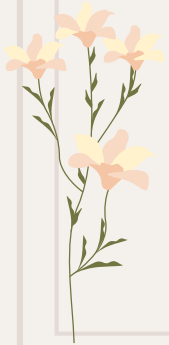
- Terinspirasi dari value iteration
- Contoh action
- Mengamati reward dan state selanjutnya
- Ambil action dengan nilai Q tertiggi (Max Q)

Action dari setiap step dapat dihitung untuk menemukan action terbaik (best action). Untuk keperluan ini digunakan Q-Table.



ALGORITMA Q LEARNING

1.Tentukan current state = initial state.
2. Dari current state, cari dengan nilai Q terbesar.
3. Tentukan current state = next state.
4. Ulang Langkah (2) dan (3) hingga current state - goal state.



Parameter Dan Variabel

```python
# Creating class for the Q-learning table
class QLearningTable:
    def __init__(self, actions, learning_rate=0.01, reward_decay=0.9, e_greedy=0.9):
        # List of actions
        self.actions = actions
        # Learning rate
        self.lr = learning_rate
        # Value of gamma
        self.gamma = reward_decay
        # Value of epsilon
        self.epsilon = e_greedy
        # Creating full Q-table for all cells
        self.q_table = pd.DataFrame(columns=self.actions, dtype=np.float64)
        # Creating Q-table for cells of the final route
        self.q_table_final = pd.DataFrame(columns=self.actions, dtype=np.float64)
```

Parameter dan Variabel yang digunakan pada model Q Learning.

1. Action
2. Learning Rate
3. Gamma
4. Epsilon
5. Q Table
6. Q Table Final

Metode Pada Class Q Learning

```
# Function for choosing the action for the agent
def choose_action(self, observation):

# Function for learning and updating Q-table with new
knowledge
def learn(self, state, action, reward, next_state)

# Adding to the Q-table new states
def check_state_exist(self, state):

# Printing the Q-table with state
def print_q_table(self):

# Plotting the results for the number of step
def plot_results(self, steps, cost):
```

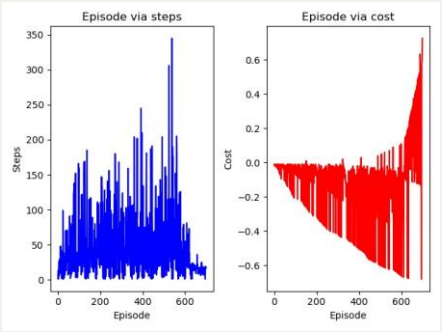
Metode Pada Class Q Learning

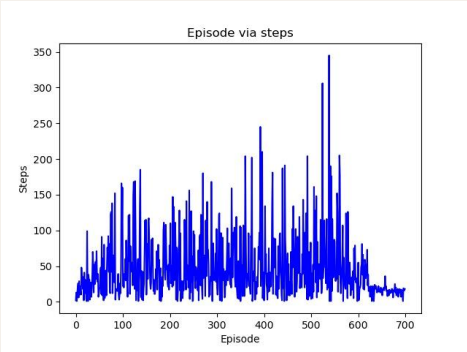
```python
def update():
    # Resulted list for the plotting Episodes via Steps
    steps = []

    # Summed costs for all episodes in resulted list
    all_costs = []

    for episode in range(700):
        # Initial Observation
        observation = env.reset()

        # Updating number of Steps for each Episode
        i = 0

        # Updating the cost for each episode
        cost = 0

        while True:
            # Refreshing environment
            env.render()

            # RL chooses action based on observation
            action = RL.choose_action(str(observation))

            # RL takes an action and get the next observation and reward
            observation_, reward, done = env.step(action)

            # RL learns from this transition and calculating the cost
            cost += RL.learn(str(observation), action, reward, str(observation_))

            # Swapping the observations - current and next
            observation = observation_

            # Calculating number of Steps in the current Episode
            i += 1

            # Break while loop when it is the end of current Episode
            # When agent reached the goal or obstacle
            if done:
                steps += [i]
                all_costs += [cost]
                break
```

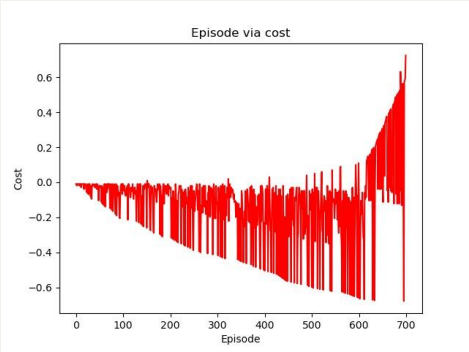
Pemodelan Q-Learning

1. Agent dan Obstacle
2. Goal dan Environment
3. Action dan state
4. Reward dan Termination

Performa Model







Analisa dan Kesimpulan

Berdasarkan gambar plot yang dihasilkan terlihat bahwa dalam 700 episode nilai steps sudah konvergen. Kemudian dapat disimpulkan bahwa model dapat mencapai konvergen pada episode 630.

TERIMA KASIH