Muhammad Andhika Ramadhan

AlphaZero

Perbandingan Q Learning dan SARSA dalam Penyelesaian Labirin (Berdasarkan Waktu)

Ujian Praktik Reinforcement Learning – Orbit Future Academy

Latar Belakang, Rumusan dan Tujuan

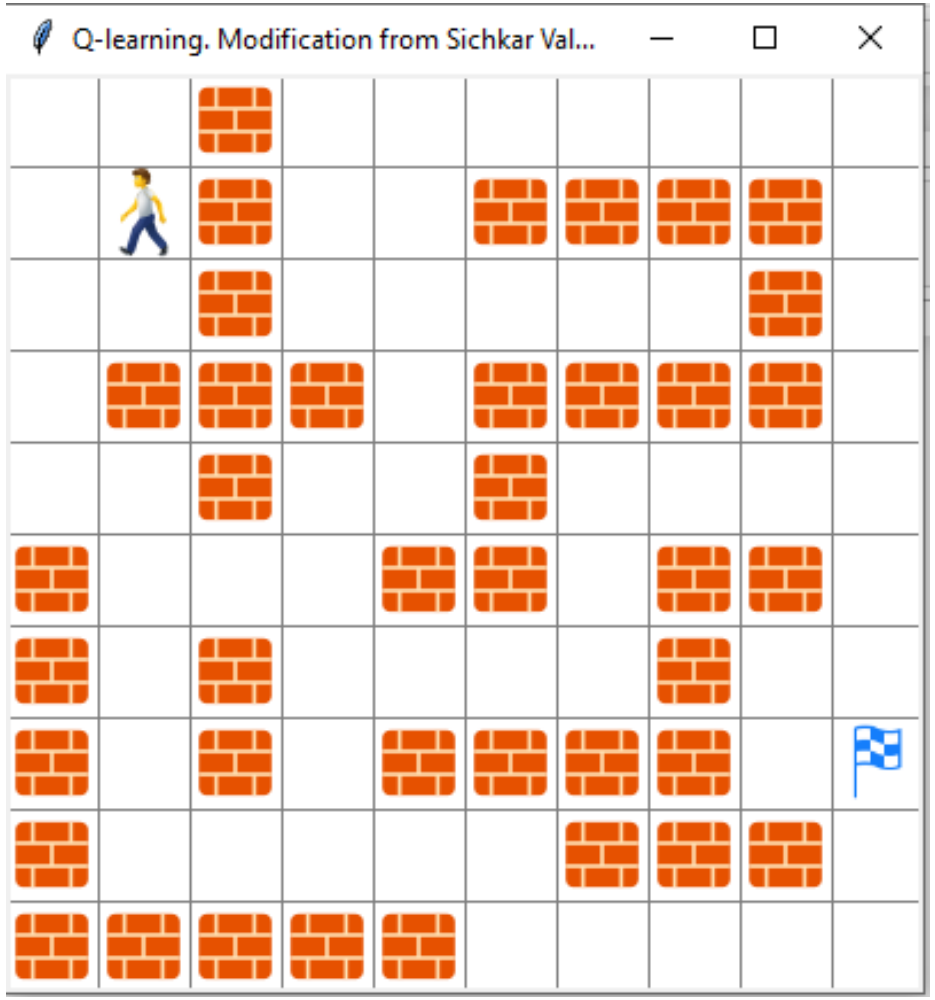
- **Latar Belakang**
  - Q Learning dan SARSA merupakan 2 contoh algoritma dalam Reinforcement Learning.
  - Labirin merupakan sebuah sistem jalur yang rumit, berliku-liku, serta memiliki banyak jalan buntu.
  - Menggunakan Reinforcement Learning untuk menyelesaikan Labirin

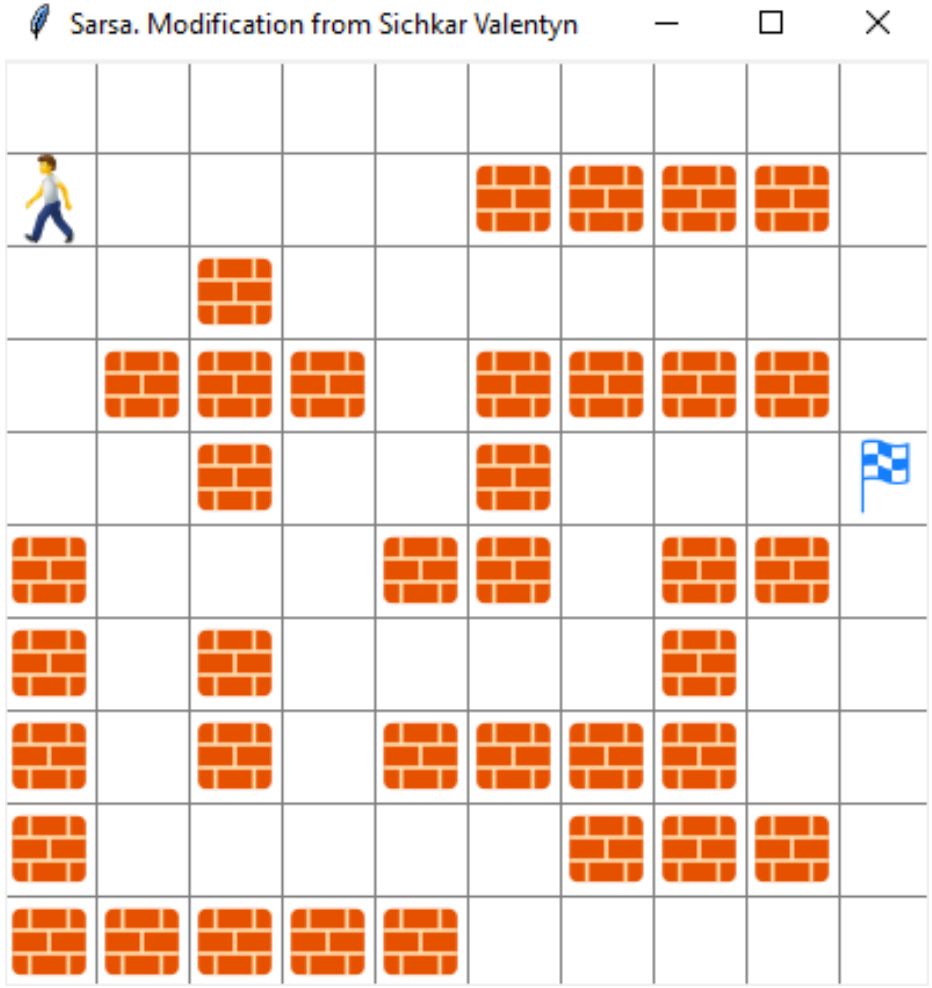
- **Rumusan Masalah**
  - Bagaimana jalan keluar yang paling efektif yang ditemukan?
  - Manakah algoritma yang lebih baik untuk kasus ini?

- **Tujuan**
  - Menemukan jalan keluar yang paling efektif di dalam sebuah labirin.
  - Menemukan Algoritma yang lebih baik diantara keduanya

Algoritma RL yang digunakan

Q-Learning dan SARSA



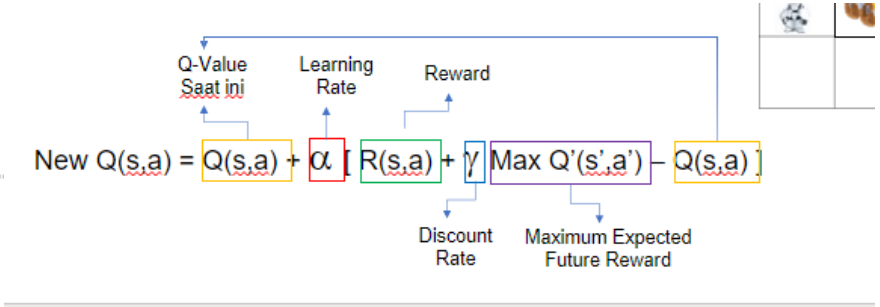


Teori Singkat

- **Q Learning**
  - Bagian dari Temporal Difference Learning
  - Off-policy: Sample policy berbeda dengan learning policy
  - Pengembangan RL yang menggunakan Q-Values (action values), meningkatkan kemampuan agent untuk belajar secara berulang
  - Konsep dasar
    - Terinspirasi dari value iteration
      - Melakukan komputasi mencari optimal state value function secara iterative (berulang)
    - Aksi didasari pada Max Q
  - Mencari policy dan Q Values yang optimal

$$Q(s,a) \leftarrow \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V^*(s')]$$

$$v_*(s) \leftarrow \max_a q_*(s,a)$$



Teori Singkat

- **SARSA**
  - Bagian dari Temporal Difference Learning
  - On-policy: Sample policy sama dengan learning policy
  - Policy Iteration
    - Diawali dengan policy yang arbitrary/random, kemudian dievaluasi dan lakukan improve dari policy-policy sebelumnya
  - Mengganti value function dengan action-value function
  - State dan Action

TD $$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

SARSA $$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S', A') - Q(S_t, A_t)]$$

Model

```
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(1500):
        # Initial Observation
        observation = env.reset()

        # Updating number of Steps for each Episode
        i = 0

        # Updating the cost for each episode
        cost = 0
```

```
# 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)
```

```
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(6000):
        # Initial Observation
        observation = env.reset()

        # Updating number of Steps for each Episode
        i = 0

        # Updating the cost for each episode
        cost = 0

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

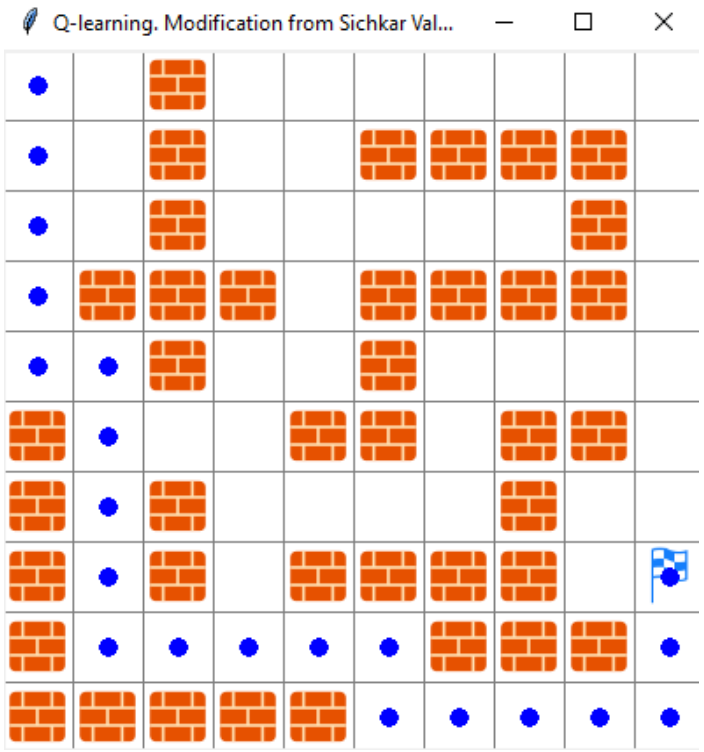
```
# Creating class for the SarsaTable
class SarsaTable:
    def __init__(self, actions, learning_rate=0.01, reward_decay=0.9, e_greedy=0.9):
        # List of actions
        self.actions = actions
        # Learning rate
        self.lr = learning_rate
        # Value of gamma
        self.gamma = 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

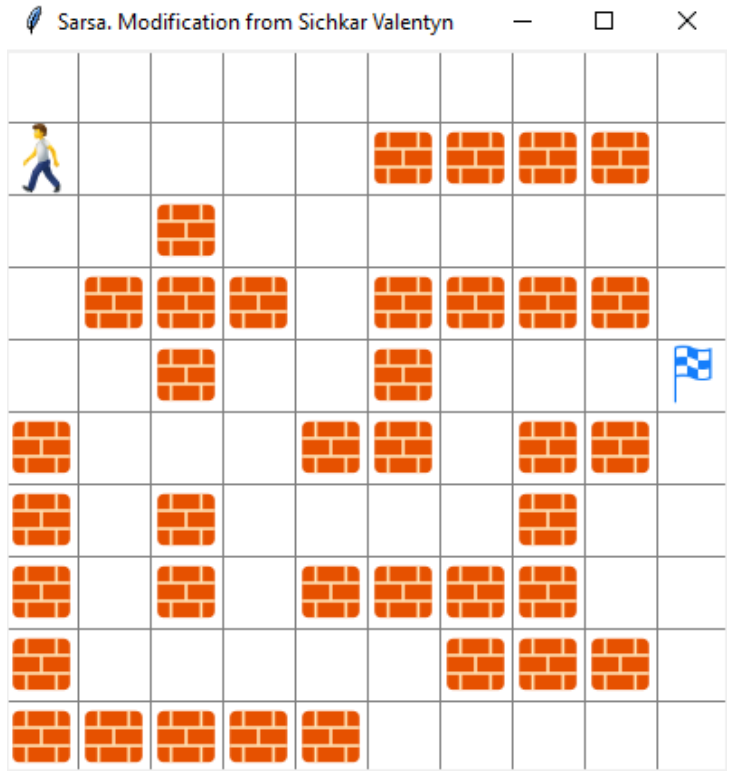
Parameter	Q Learning	SARSA
Episode	1500	6000
Learning Rate	0.01	0.7
Reward Decay	0.9	0.9
Greedy	0.9	0.9
Pixels	40	40
Height	10	10
Width	10	10

Performa

Q-Learning vs SARSA



Wall time: 5min 12s



Unfinished

Analisa dan Kesimpulan

- Q Learning lebih baik dibandingkan dengan SARSA dari sisi waktu.
- Catatan waktu baru ->
- Alasan kenapa lebih baik pada kasus ini?
  - QL belajar dari nilai yang optimal
  - Agent pada QL lebih agresif, SARSA konservatif (eksplorasi)
  - QL ambil jalan terpendek (karena nilai optimal), sementara SARSA jalan yang dinilai aman
- Apakah SARSA akan selalu tidak lebih baik?
  - Tidak juga
  - Kalo misal dalam membuat sebuah robot dan suatu kesalahan bisa menyebabkan kerusakan fatal, SARSA lebih direkomendasikan

https://towardsdatascience.com/intro-to-reinforcement-learning-temporal-difference-learning-sarsa-vs-q-learning-8b4184bb4978#:~:text=QL%20directly%20learns%20the%20optimal,is%20walking%20near%20the%20cliff.

Terima Kasih ☺

Ujian Praktik – Orbit Future Academy