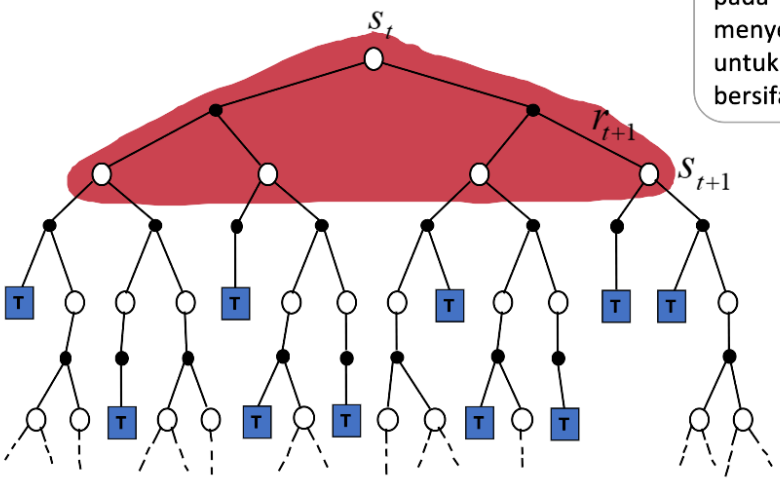
5.6.1 Robotics-based Reinforcement Learning

Tags

Recap

Dynamic Programming

$$V(S_t) \leftarrow \mathbb{E}_\pi[R_{t+1} + \gamma V(S_{t+1})]$$

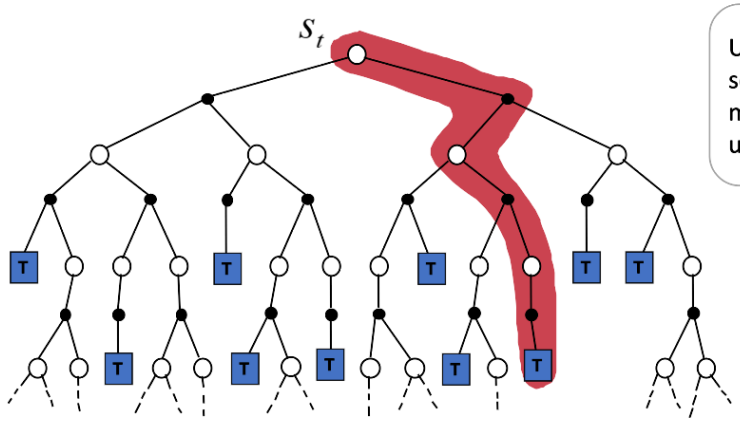


Update value state $V(S_t)$ per step. Jadi pada DP algoritma **tidak harus** menyelesaikan 1 episode secara utuh untuk meng-update value state V(s). Dan bersifat bootstrapping.

Simple Monte Carlo

$$V(S_t) \leftarrow V(S_t) + \alpha\big(G_t - V(S_t)\big)$$

Dimana G_t adalah *actual return* terhadap *state* S_t

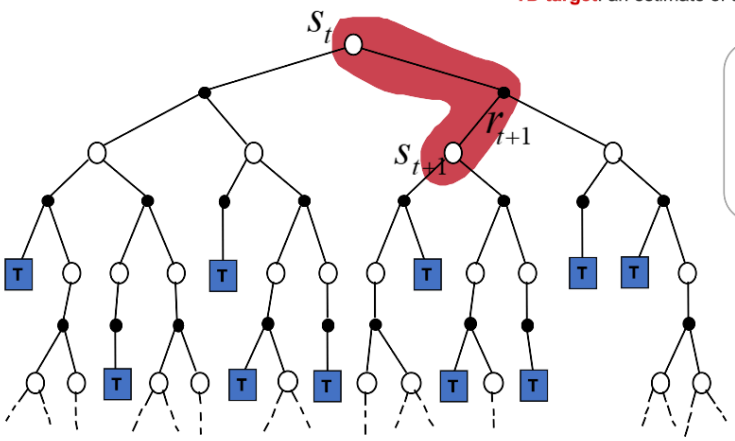


Update value state $V(S_t)$ setelah episode selesai. Jadi pada MC algoritma **harus** menyelesaikan dahulu 1 episode secara utuh untuk meng-update value state V(s).

Simple TD Method

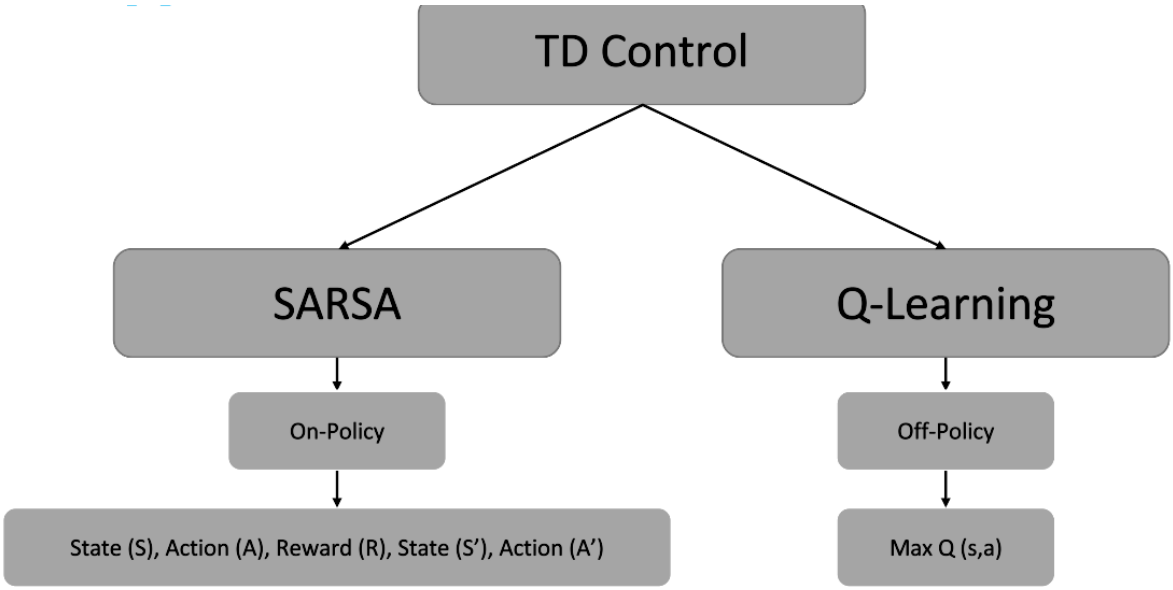
$$V(S_t) \leftarrow V(S_t) + \alpha\big(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\big)$$

TD target: an estimate of the return



Update value state $V(S_t)$ per step seperti dynamic programming. Jadi pada TD algoritma **tidak harus** menyelesaikan 1 episode secara utuh untuk meng-update value state V(s).

Temporal Difference Learning

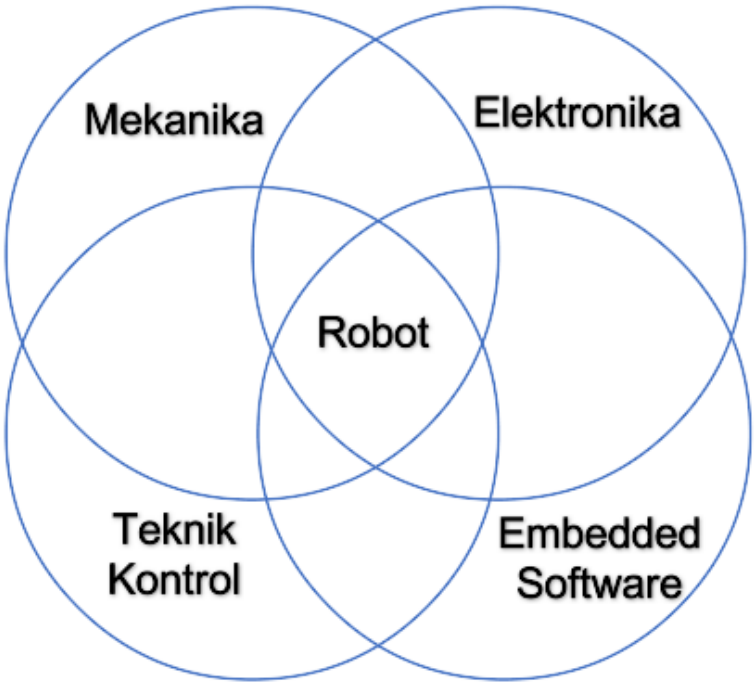


Introduction to Robotics

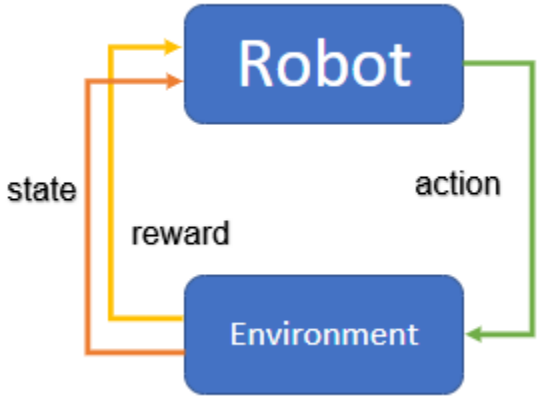
Robotics adalah suatu disiplin ilmu yang mempelajari tentang konsep suatu robot.

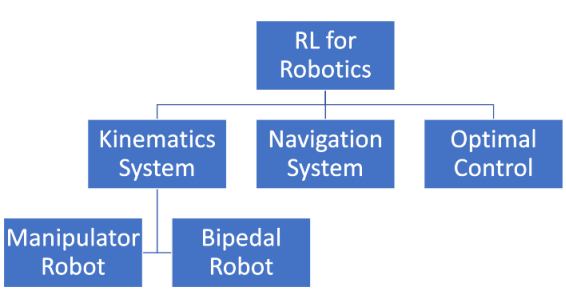
Sedangkan Robot merupalan mesin yang beroperasikan secara otomatis yang menggantikan usaha manusia, meskipun mungkin tidak menyerupai manusia dalam penampilan atau melakukan fungsi dengan cara yang mirip manusia.

Interdisiplin Robotika :



- Apakah RL bisa diterapkan ke dalam Robotics?
- Kira kira apa saja penerapan RL ke dalam Robotics?





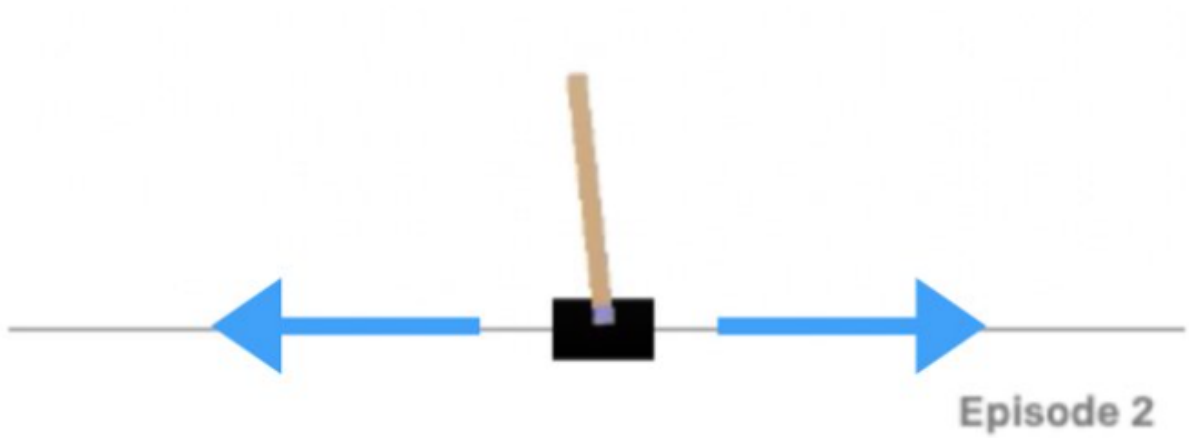
Kinematics System

- State: Total joints or degrees of freedom of the robot on radian.
- Action: X, Y, Z certasian position of each legs (6 action)

Navigation System

- Motion planning, trajectory optimization, dynamic pathing, controller optimization, and scenario-based learning policies are the application of self driving car using reinforcement learning.
- Places, Driveable zones, Obstacle avoidance, Velocity, and Direction can be a varible for defining the state and action on reinforcement learning model for self driving car.
- https://wayve.ai/blog/learning-to-drive-in-a-day-with-reinforcement-learning/

Optimal Control

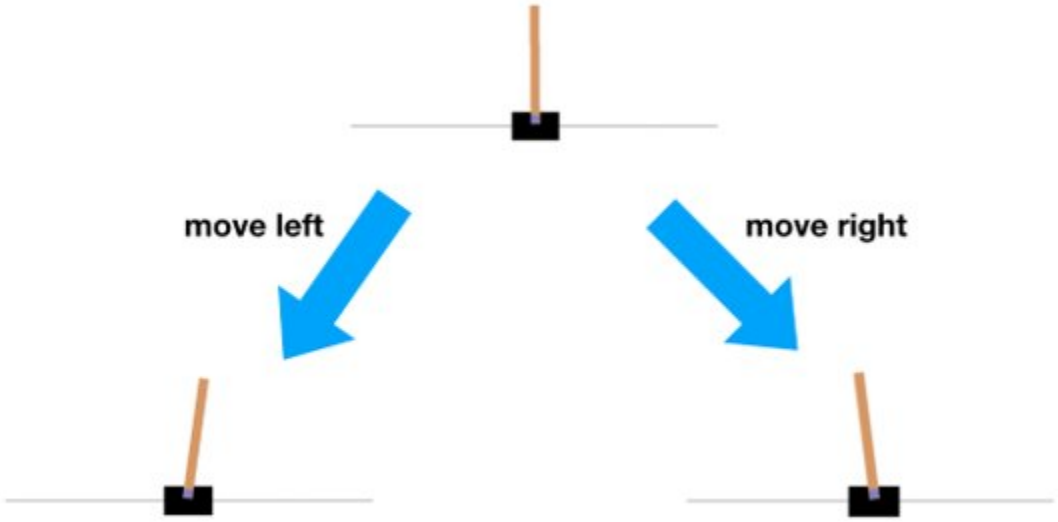


State yang tersedia dari env

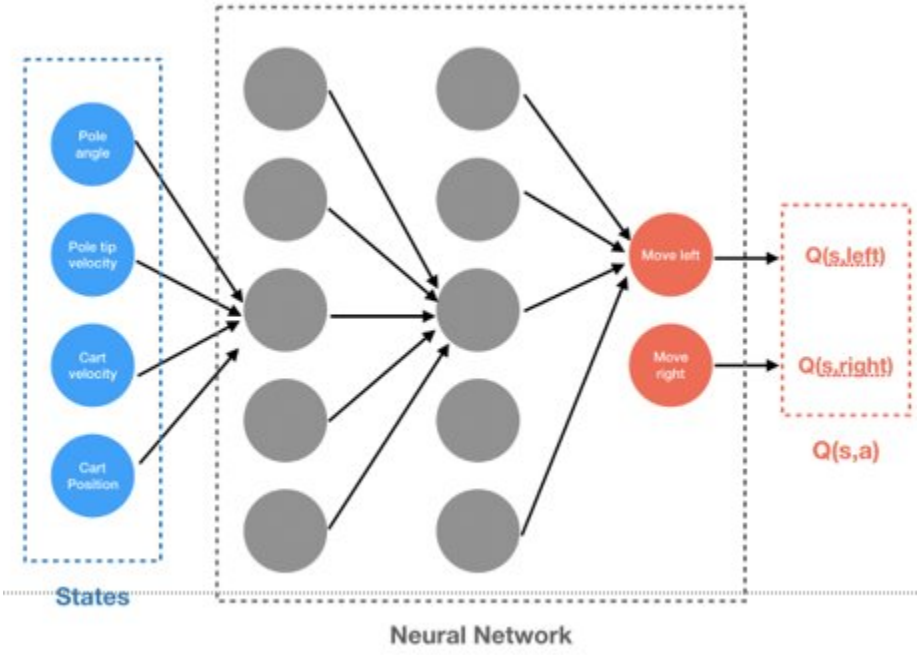
- Untuk Posisi Cart yang dapat diamati meskipun tersedia rentang -4.8~4.8 namun game akan berakhir saat posisi keluar dari rentang -2.4~2.4
- Meskipun Sudut Pendulum yang dapat diamati tersedia rentang -24o~-24o namun game akan berakhir saat posisi keluar dari rentang -12o~12o

Num	Observation	Min	Max
0	Posisi Cart	-4.8	4.8
1	Kecepatan Cart	-inf	Inf
2	Sudut Pendulum	-0.418 r (-24°)	0.418 r (24°)
3	Kecepatan Sudut Pendulum	-inf	inf

Action yang tersedia pada env



Penerapan DQL pada Cart-Pole

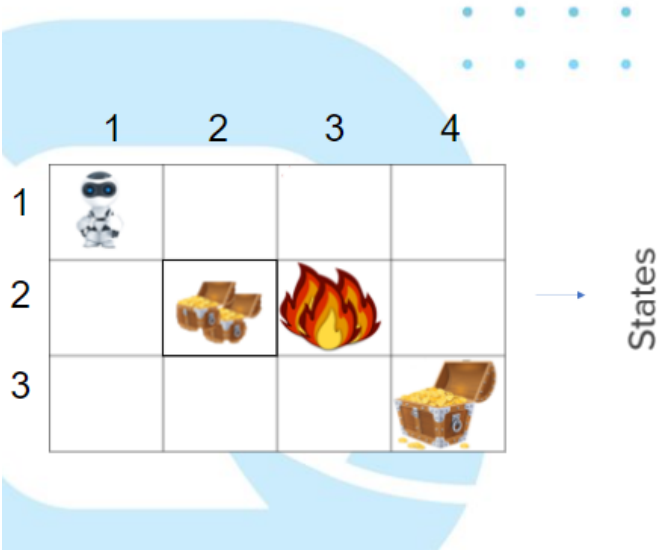


Static Goals Navigation using Reinforcement Learning

Studi Kasus

Studi kasus kali ini kita akan membantu robot untuk menemukan harta karun yang paling banyak dengan step yang sedikit dan menghindari supaya tidak terkena api azab

1. **Definisi Reward dan Q Tabel**



Actions

States

	Kiri	Kanan	Atas	Bawah
(1,1)	0	0	0	0
(1,2)	0	0	0	0
(1,3)	0	0	0	0
(1,4)	0	0	0	0
(2,1)	0	0	0	0
(2,2)	0	0	0	0
(2,3)	0	0	0	0
(2,4)	0	0	0	0
(3,1)	0	0	0	0
(3,2)	0	0	0	0
(3,3)	0	0	0	0
(3,4)	0	0	0	0

2. **Choose Action**

   Pada dasarnya dalam algoritma Q-learning, agnet akan memilih action berdasarkan Q-table. Agent akan memilih action yang mempunyai Q-Value paling besar berdasarkan state saat ini. Tapi hal tersebut akan terdengan aneh untuk langkah pertama, karena pada langkah pertama semua Q-value yang ada pada Q-tabel bernilai 0, jadi gimana cara agent memilih action?

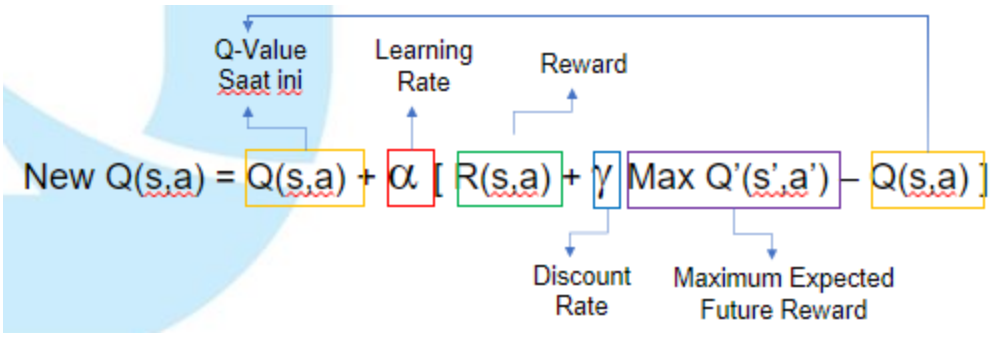
   Nah untuk menjawab itu kita akan coba mengingat kembali tentang Eksplorasi dan Eksploitasi.

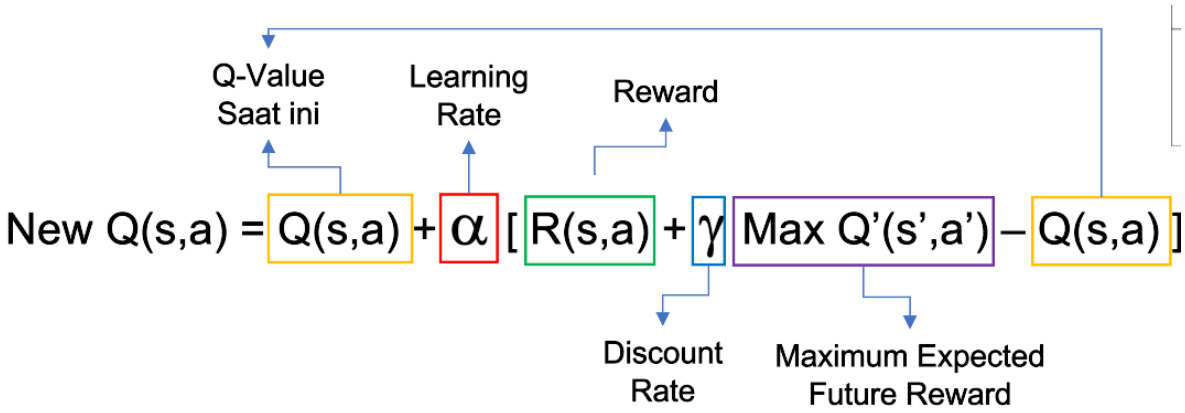
3. **Perform Action and Get Reward**

   Reward = -1

4. **Update Q-Tabel**

   Setelah eksekusi action dan mendapatkan reward, saatnya kita mengupdate Q-table dengan mengganti Q-value yang lama dengan Q-value yang baru, untuk mengupdate Q-tabel dan menghitung Q-value yang baru kita dapat menggunakan formula berikut





Kita akan coba update Q-Tabel untuk agent bergerak ke bawah, dengan kita

$$\text{New } Q(s,a) = Q(s,a) + \alpha[R(s,a) + \gamma MaxQ'(s',a') - Q(s,a)]$$

$$MaxQ'(s',a') = Max(q((2,1),kiri), q((2,1),kanan), q((2,1),atas), q((2,1),bawah))$$

$$MaxQ'(s',a') = Max(0,0,0,0)$$

$$MaxQ'(s',a') = 0$$

	Kiri	Kanan	Atas	Bawah
(1,1)	0	0	0	0
(1,2)	0	0	0	0
(1,3)	0	0	0	0
(1,4)	0	0	0	0
(2,1)	0	0	0	0
(2,2)	0	0	0	0
(2,3)	0	0	0	0
(2,4)	0	0	0	0
(3,1)	0	0	0	0
(3,2)	0	0	0	0
(3,3)	0	0	0	0
(3,4)	0	0	0	0

Kita akan coba update Q-Tabel untuk agent bergerak ke bawah, dengan kita definisikan $\gamma = 0.99$ dan $\alpha = 0.7$

$$MaxQ'(s', a') = 0$$

$$\text{New } Q(s,a) = Q(s,a) + \alpha[\mathbb{R}(s,a) + \gamma MaxQ'(s', a') - Q(s,a)]$$

$$\text{New } Q((1,1), bawah) = 0 + 0.7[-1 + 0.99(0) - 0]$$

$$\text{New } Q((1,1), bawah) = 0 + 0.7[-1]$$

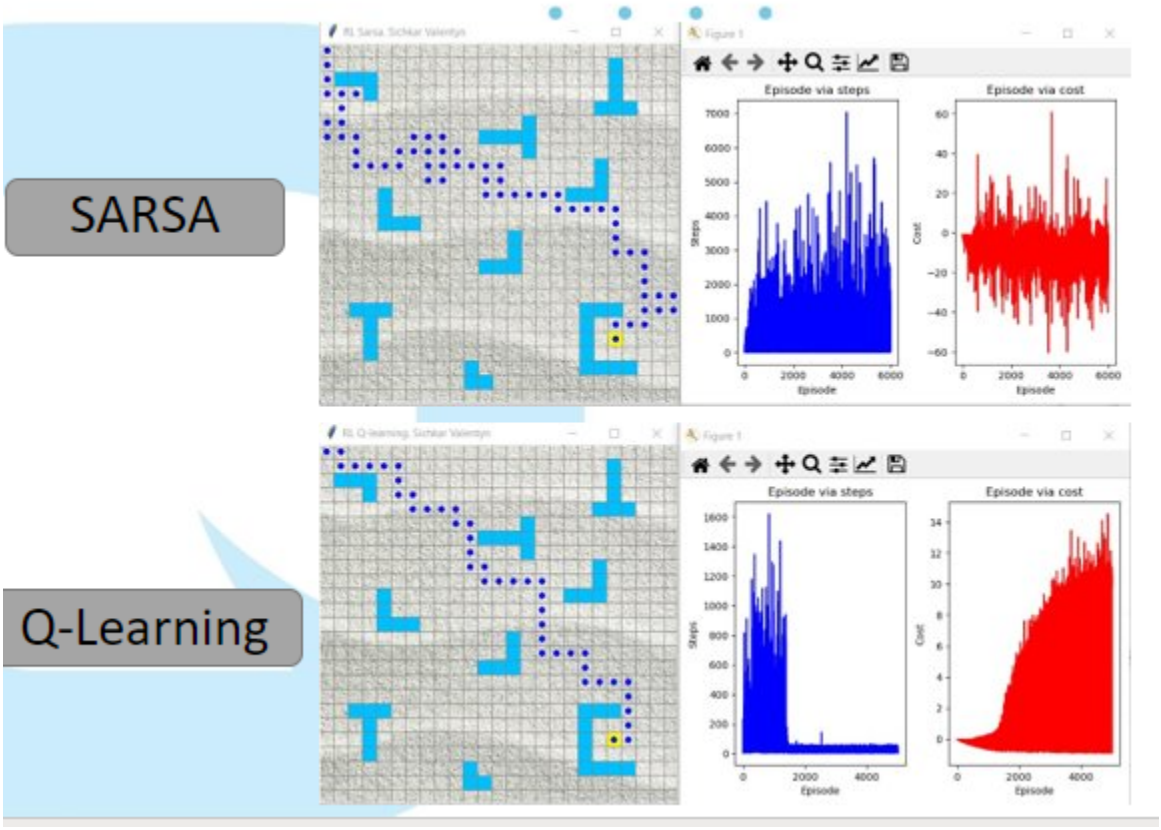
$$\text{New } Q((1,1), bawah) = -0.7$$

Jadi, kita sudah melakukan kalkulasi untuk Q-Value yang baru untuk state (1,1) dan action ke bawah, dan kita juga sudah menyimpan value tersebut ke Q-Tabel.

Kita sudah selesai melakukan proses Q-Learning untuk satu step. Proses ini (langkah 2 sampai 4) akan dilakukan sampai beberapa episode. Sampai Q-tabel yang kita punya konvergen, sehingga kita menemukan policy yang paling optimal.

	Kiri	Kanan	Atas	Bawah
(1,1)	0	0	0	-0.7
(1,2)	0	0	0	0
(1,3)	0	0	0	0
(1,4)	0	0	0	0
(2,1)	0	0	0	0
(2,2)	0	0	0	0
(2,3)	0	0	0	0
(2,4)	0	0	0	0
(3,1)	0	0	0	0
(3,2)	0	0	0	0
(3,3)	0	0	0	0
(3,4)	0	0	0	0

SARSA vs Q-Learning



Deep Q Learning Implementation

💡 Sama seperti materi 5.5