

5.1.1 Introduction to Reinforcement Learning 1

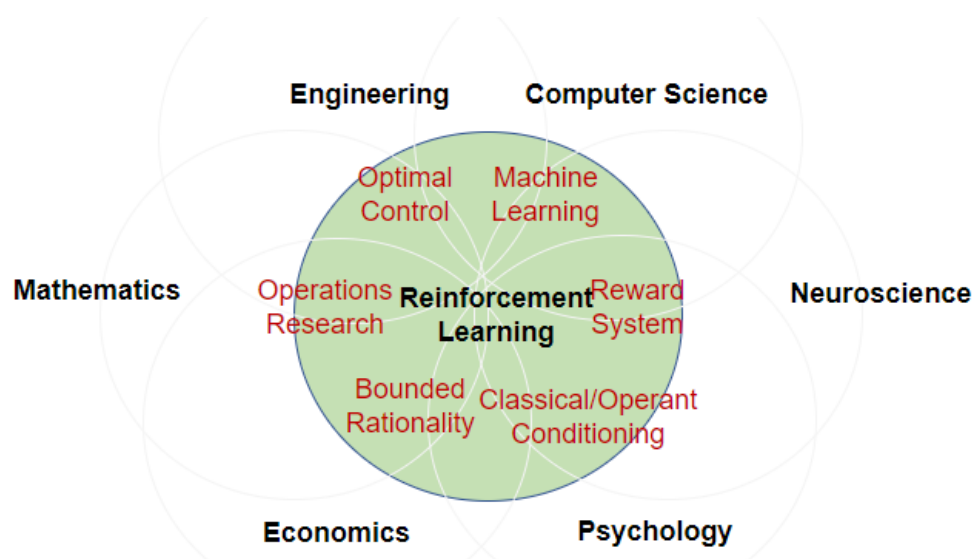
Tags

Learning Objectives

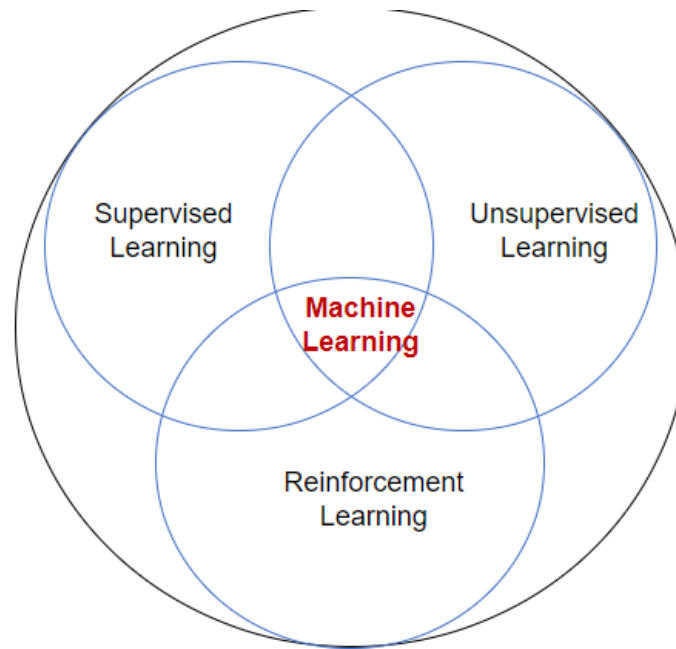
1. Dapat mendeskripsikan definisi dan sejarah reinforcement learning (RL).
2. Dapat mendeskripsikan algoritme RL.
3. Dapat mendeskripsikan elements & environment dari RL.
4. Dapat mendeskripsikan aplikasi dari RL

Definisi & Sejarah RL

RL dalam Berbagai Bidang



Cabang Machine Learning



Stanford Marshmallow Test

- Tahun 1972
- Lebih lanjut lihat di: https://en.wikipedia.org/wiki/Stanford_marshmallow_experiment

Agent, Reward, dan Action?

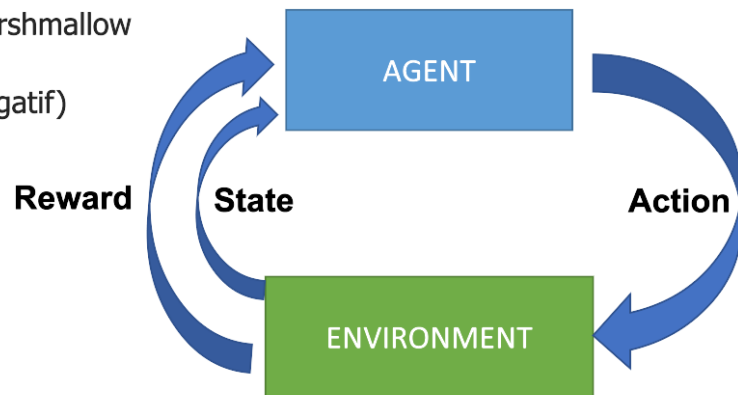
- Agent: Siapakah Agent dalam eksperimen marshmallow?
- Reward: Apakah Karakteristik Reward yang bisa kamu ketahui dari contoh di atas?
- Action: Tindakan apa saja yang tersedia pada Eksperimen Marshmallow?

Karakteristik Reward dan Action

- Reward:
Sistem Reward dalam sebuah Lingkungan mungkin tidak langsung diberikan. (delayed reward).
Reward bisa dilihat sebagai umpan balik terhadap Agent, karena beberapa reward mengalami penundaan maka umpan balik seringkali menjadi tertunda. Ada yang namanya "Delayed Gratification" dan seringkali memberikan total reward yang lebih besar.
- Action:
Pada contoh di atas Semua Tindakan yang mungkin telah diketahui.

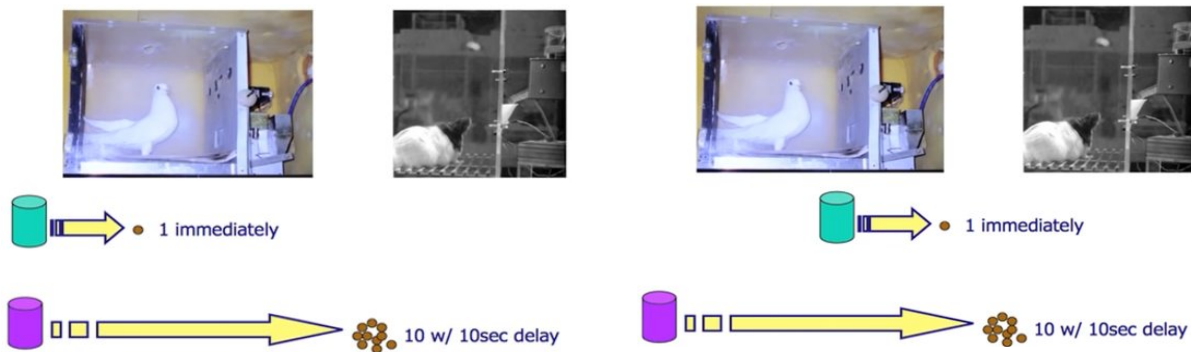
RL Secara Formal?

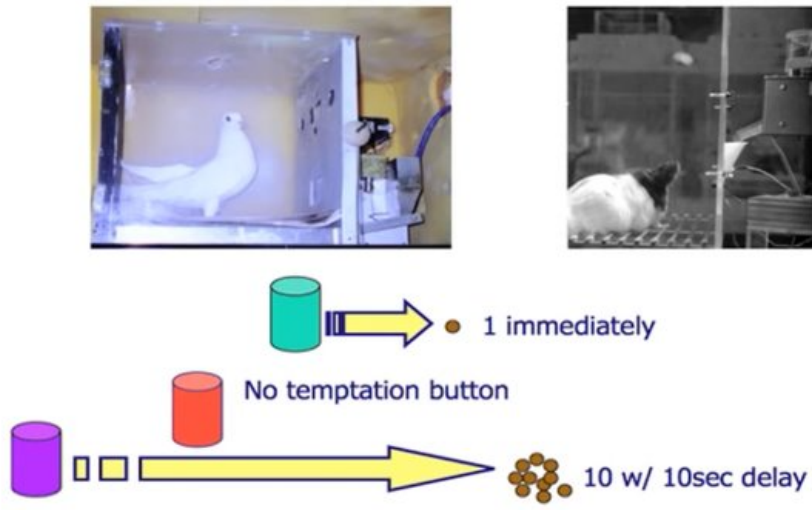
- **Reward (R_t)**
 - Bahagia langsung makan Marshmallow
 - Marshmallow Tambahan
 - Sabar Menunggu (*reward* negatif)
- **State (S_t)**
 - Menunggu
 - Makan
- **Action (A_t)**
 - Langsung makan
 - Menunggu dengan bengong
 - Menunggu sambil Fitness
 - Menunggu sambil Tidur
 - Menunggu 5 menit lalu makan



Trial & Error System 1

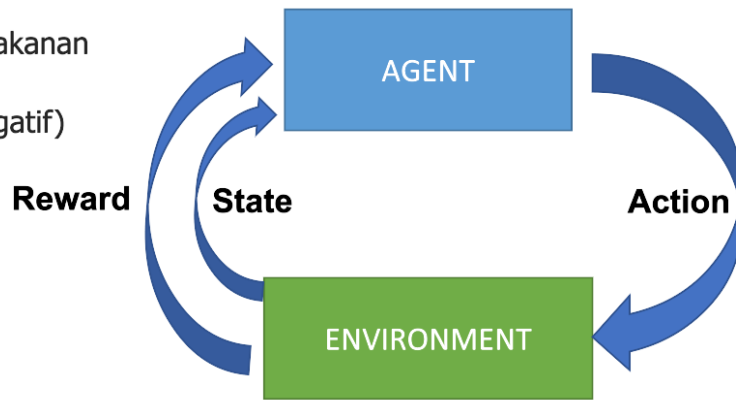
- Ainslie & Herrnstein 1974





RL Secara Formal

- **Reward (R_t)**
 - Bahagia langsung dapat 1 makanan
 - Dapat 10 makanan
 - Sabar Menunggu (*reward* negatif)
- **State (S_t)**
 - Tombol Hijau ditekan
 - Tombol Ungu ditekan
 - Tombol Merah ditekan
- **Action (A_t)**
 - Menekan Tombol Hijau
 - Menekan Tombol Ungu
 - Menekan Tombol Merah



Environment (Lingkungan)?

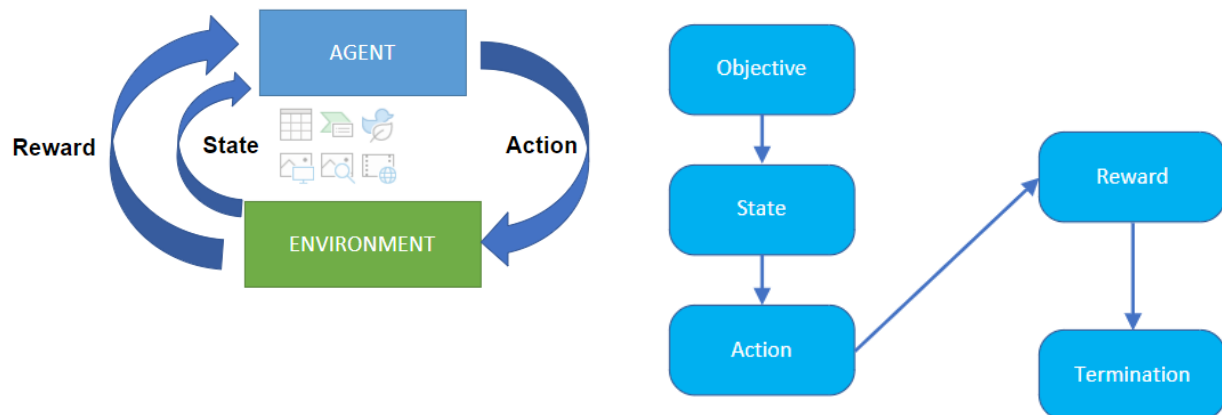
- Apa yang membedakan lingkungan antara percobaan Marshmallow dan Merpati, Tikus?
- Aturan Reward dalam suatu Lingkungan Kadang dideskripsikan dengan jelas.
- Ada Env yang perlu dipelajari karena tidak terdapat aturan yang jelas. (Trial and Error / Eksplorasi).
- Dalam Dunia nyata, Improvisasi masih bisa terjadi meski sudah ada aturan.



Karakteristik dari RL

- Tidak ada “kebenaran” pada proses pembelajaran (training) yang ada hanya hadiah (atau hukuman)
- Data yang diperoleh (State dan Reward) tergantung dari Action yang diberikan oleh Agent.
- Variabel waktu dan urutan state yang dipilih oleh Agent melalui action memiliki pengaruh yang penting.

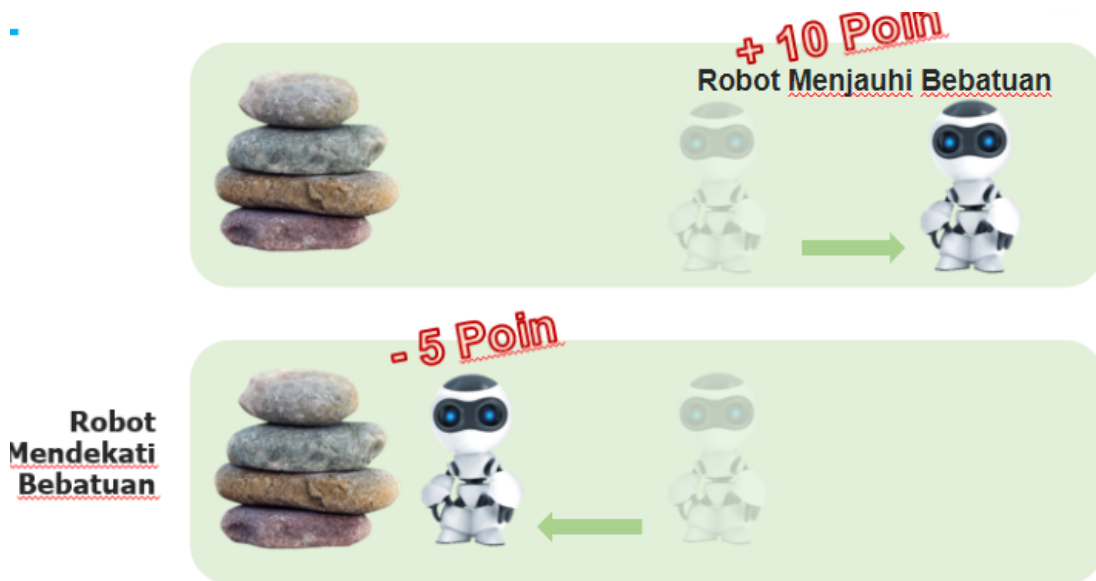
Algoritma RL



- Algoritme RL bekerja berdasarkan feedback environment yang diterima agent dari environment yang melibatkan variabel-variabel state-action-reward (st, at, rt). Proses ini dinamakan sequential decision-making process, dan (st, at, rt) dinamakan experience.
- RL menghitung jumlah dari reward yang diterima oleh agent. Tujuan (objective) dari agent adalah memaksimalkan total reward.
- RL belajar (learn) dari interaksi agent dengan environment menggunakan proses trial & error, model-based optimization (reward), atau keduanya yang diperoleh agent, untuk memperkuat (reinforce) aksi positif.
- Agent berinteraksi terhadap environment dengan melakukan sebuah aksi dari satu kondisi ke kondisi yang lain.
- Agent akan menerima reward berdasarkan aksinya.

- Berdasarkan reward tersebut, agent akan memahami apakah aksinya baik atau buruk.
- Jika aksinya baik, agent akan menerima reward positif, sehingga agent tersebut akan cenderung lebih memilih melakukan aksi yang serupa dengan aksi tersebut (exploitation), atau melakukan aksi lain yang mungkin menghasilkan reward positif atau negatif (exploration).

Ilustrasi Reward pada pergerakan agen RL



- Environment pada RL tidak mengajari agent untuk melakukan ini dan itu.
- Tetapi, agent melakukan aksi/pergerakan sendiri sebagai respon dari perubahan kondisi environment (state) yang diterimanya, lalu agent diberikan reward terhadap performa aksinya tersebut. Disinilah proses belajar dalam RL terjadi.
- Mekanisme reward tersebut, secara tidak langsung, akan mendorong agent untuk mencari reward positif sebanyak-banyaknya.
- Reward dapat diberikan setiap langkah, atau dapat juga diberikan setelah beberapa langkah.
- Environment pada RL tidak mengajari agent untuk melakukan ini dan itu.
- Tetapi, agent melakukan aksi/pergerakan sendiri sebagai respon dari perubahan kondisi environment (state) yang diterimanya, lalu agent diberikan reward terhadap performa aksinya tersebut. Disinilah proses belajar dalam RL terjadi.
- Mekanisme reward tersebut, secara tidak langsung, akan mendorong agent untuk mencari reward positif sebanyak-banyaknya.
- Reward dapat diberikan setiap langkah, atau dapat juga diberikan setelah beberapa langkah.

Implikasi Reward pada Algoritme RL

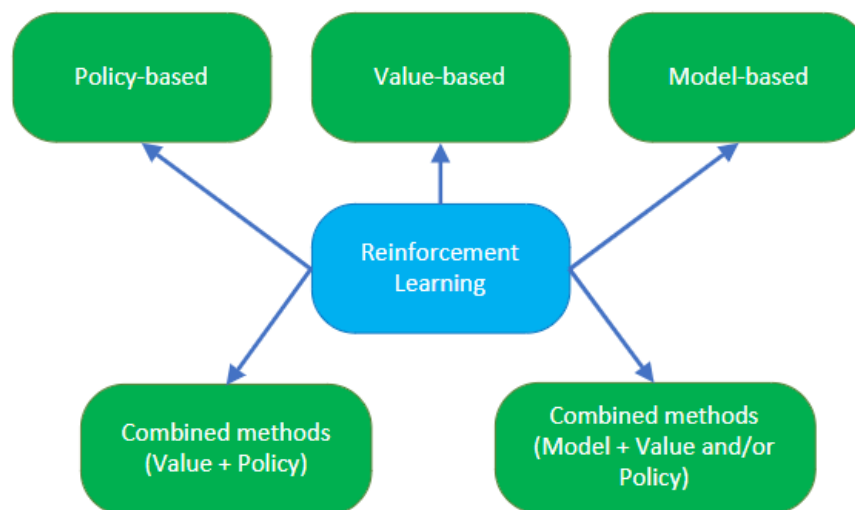
- Penentuan besarnya nilai reward akan menjadi acuan bagaimana algoritme RL melakukan eksploitasi dan eksplorasi.

- Eksploitasi adalah aksi algoritma RL dalam menggunakan aksi serupa sebelumnya yang mendapat reward positif.
- Eksplorasi adalah aksi algoritma RL dalam melakukan aksi berbeda untuk mencari reward positif yang lain.
- Ada pitfall (jebakan) untuk ketidaktepatan dalam menentukan reward, menentukan hasil keluaran.
- Algoritma RL yang lebih dominan melakukan eksploitasi, akan mengakibatkan kemungkinan mendapatkan peluang solusi baik lain akan terlewatkan
- Algoritma RL yang lebih dominan eksplorasi, ada kemungkinan agent RL tersebut akan mendapatkan banyak peluang solusi buruk.
- Solusi: Penerapan metode optimasi.

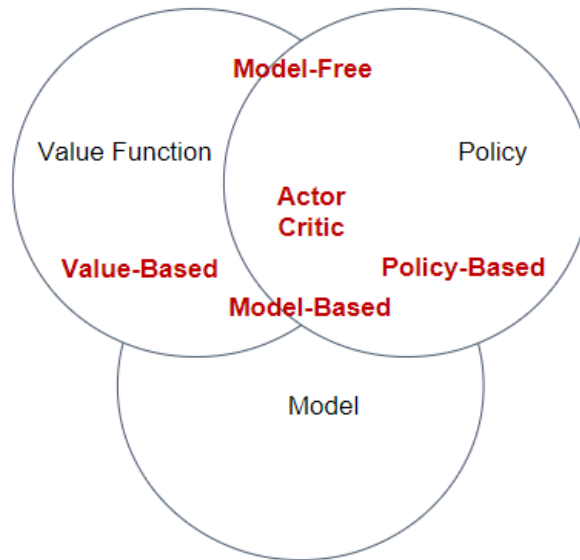
Contoh Algoritme RL

1. **Objective:** Memaksimalkan skor permainan
2. **State:** Gambar RGB dengan resolusi 160x210 px
3. **Action:** Integer dari himpunan {0, 1, 2, 3} yang memetakan pengendalian aksi dari game {no-action, launch the ball, move right, left, up, down}
4. **Reward:** Skor permainan
5. **Termination:** Kalah permainan

Kategori Implementasi Algoritme RL



Taksonomi RL Agent



Elements & Environment RL

Element pada RL

Di luar dari Agent dan Environment yang merupakan sebagai element utama, ada 4 sub elemen sebagai penyusun utama sistem Reinforcement Learning:

Elemen utama:

1. **Agent** : yang dimaksud dengan agent adalah perangkat atau software yang dapat belajar dari environment
2. **Environment** : sedangkan yang dimaksud dengan environment adalah segala sesuatu yang ada di luar agent yang dimana sebagai tempat agent untuk melakukan exploration dan exploitation.



Sub Elemen utama:

1. **Policy** : rules (aturan) atau strategi yang digunakan oleh agent untuk melakukan action (**A**) selanjutnya, berdasarkan state (**S**) saat ini.
2. **Reward (R) signal** : feedback atau umpan baik untuk agent, Reward dapat bernilai positif (berupa hadiah) atau negatif (berupa hukuman) dan juga nol (tidak ada tindakan apapun terhadap agent).

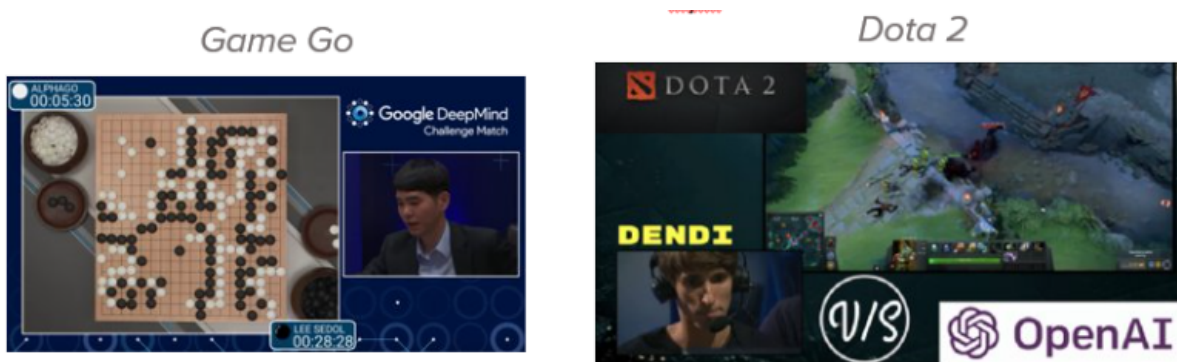
3. **Value function (V) s** : total nilai jangka yang diharapkan (expected long-term return without discount) dari state saat ini di bawah policy, Value ini kebalikan dari short-term reward (**R**).
 - a. Optimal Value Function adalah sebuah fungsi yang memiliki nilai tertinggi untuk semua state dibandingkan dengan fungsi nilai lainnya.
 - b. Optimal Policy adalah policy (kebijakan) yang memiliki fungsi nilai optimal.
4. **Model environment (optional)** : segala sesuatu yang meniru perilaku environment. Atau, secara umumnya, sebuah kesimpulan tentang bagaimana perilaku dari environment.

Model Environment dibagi menjadi dua, yaitu:

- Model Based RL
- Model Free RL

Model Based RL vs Model Free RL

Dapatkah anda melihat perbedaan dari kedua game ini dari sudut pandang model environment-nya?

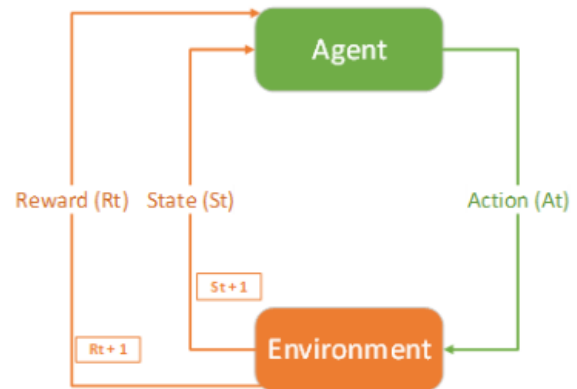


Dalam memecahkan persoalan yang berhubungan dengan Reinforcement Learning, ada dua metode yang dipakai, yaitu model base dan model free, keduanya didasarkan pada bagaimana lingkungannya yang dihadapi oleh agent.

- **Model Based RL**, model ini adalah model dasar atau paling sederhana dan memiliki perencanaan terhadap tindakannya, atau dapat juga dimaknai sebagai agent mengeksplorasi informasi yang dipelajari sebelumnya untuk menyelesaikan tugasnya.
- **Model Free RL**, sedangkan model ini adalah kebalikan dari model Model Based RL, karena agent belajar dari lingkungan melalui metode trial and error untuk memperoleh pengalamannya.

Environment RL

Agent adalah software agents yang melakukan aksi. Pada waktu t kemudian berpindah dari suatu state S_t ke state lainnya S_{t+1} . Sebagai imbalan agent akan mendapatkan reward berupa angka numerik atas setiap tindakannya dari environment.



Tipe-tipe Environment RL:

- **Deterministic environment** : suatu environment dikatakan ketika hasilnya dapat diketahui berdasarkan state saat ini. Misalnya pada permainan catur, kita dapat mengetahui hasilnya setelah pemain (pion, raja, benteng, kuda, dll) mana yang dipindahkan.
- **Stochastic environment** : suatu environment dikatakan stochastic ketika hasilnya tidak dapat diketahui berdasarkan state saat ini, dan environment tipe ini adalah kebalikan dari deterministic, Contohnya pada saat pelemparan dadu, ada kemungkinan sangat besar kita tidak dapat mengetahui angka berapa yang akan muncul.
- **Fully observable environment** : Fully observable environment adalah ketika agent dapat menentukan state dari sistem setiap saat, dengan state dari sistem dapat diamati seluruhnya. Misalnya, dalam permainan catur, posisi pemain kita dapat kapapun dan dimanapun kita pindahkan, sehingga kita dapat menentukan langkah paling optimal pada setiap kali memindahkan pemain.
- **Partially observable environment**: Sedangkan Partially observable, ketika agent tidak dapat menentukan state sistem setiap saat. Misalnya, dalam permainan poker, kita tidak tahu kartu apa yang dimiliki oleh lawan.
- **Discrete environment** : ketika state untuk berpindah dari state ke state lainnya terbatas. Misalnya, dalam permainan catur, dimana hanya memiliki satu set Gerakan terbatas.
- **Continuous environment** : ketika state untuk berpindah dari state ke state lainnya tidak terbatas. Misalnya, dalam kasus menentukan rute jalan menuju suatu tempat dengan jalan yang berbeda-beda.
- **Episodic and non-episodic environment** : Environment episodik disebut juga sebagai lingkungan non-sequensial. Pada environment episodik, tindakan agent saat ini tidak akan mempengaruhi tindakan dimasa depan, sedangkan dalam environment non-episodik tindakan agent saat ini mempengaruhi tindakan di masa depan dan juga disebut sequential environment. Artinya, agent melakukan tugas-tugas independen di lingkungan episodik, sedangkan dalam lingkungan non-episodik semua tindakan agent terkait.
- **Single and multi-agent environment** : Single environment hanya memiliki agent tunggal, sedangkan multi-agent environment memiliki dari 1 agent atau bahkan bisa lebih, multi-agent environment sering digunakan pada saat melakukan tugas-tugas kompleks, Agent dengan environment yang berbeda dapat saling berkomunikasi, dan multi-agent environment sebagian besar dapat berubah menjadi stokastik karena memiliki tingkat ketidakpastian yang lebih besar.

Aplikasi RL

RL for AI in Games

- Atari Games : <https://www.atari.com/atari-games/>
- Open AI five - Dota : <https://openai.com/blog/openai-five/>

Examples of Building AI for snake game using Reinforcement Learning

- Mendefinisikan state yang mungkin terjadi (11 states)

```
state = [  
    # Danger straight  
    (dir_r and game.is_collision(point_r)) or  
    (dir_l and game.is_collision(point_l)) or  
    (dir_u and game.is_collision(point_u)) or  
    (dir_d and game.is_collision(point_d)),  
  
    # Danger right  
    (dir_r and game.is_collision(point_r)) or  
    (dir_l and game.is_collision(point_l)) or  
    (dir_u and game.is_collision(point_u)) or  
    (dir_d and game.is_collision(point_d)),  
  
    # Danger left  
    (dir_r and game.is_collision(point_r)) or  
    (dir_l and game.is_collision(point_l)) or  
    (dir_u and game.is_collision(point_u)) or  
    (dir_d and game.is_collision(point_d)),  
  
    # Move direction  
    dir_l,  
    dir_r,  
    dir_r,  
    dir_d,  
  
    # Food location  
    game.food.x < game.head.x, # food left  
    game.food.x > game.head.x, # food right  
    game.food.y < game.head.y, # food up  
    game.food.y > game.head.y # food down  
]  
  
print(state[0], state[1], state[2], state[3], state[4], state[5], state[6], state[7], state[8], state[9], state[10])
```

- Mendefinisikan action yang ada (3 actions: Straight, Turn right, Turn left)

```
def _move(self, action):  
    """  
    The snake can only move [Straight, right, left]  
    """  
  
    clock_wise = [Direction.RIGHT, Direction.DOWN, Direction.LEFT, Direction.UP]  
    idx = clock_wise.index(self.direction)  
  
    if np.array_equal(action, [1, 0, 0]):  
        new_dir = clock_wise[idx] # no change  
    elif np.array_equal(action, [0, 1, 0]):  
        next_idx = (idx + 1) % 4  
    else: # [0, 0, 1]  
        next_idx = (idx - 1) % 4  
        new_dir = clock_wise[next_idx] # left turn r -> u -> l -> d
```

```

self.direction = new_dir

x = self.head.x
y = self.head.y

if self.direction == Direction.RIGHT:
    x += BLOCK_SIZE
elif self.direction == Direction.LEFT:
    x -= BLOCK_SIZE
elif self.direction == Direction.DOWN:
    y += BLOCK_SIZE
elif self.direction == Direction.UP:
    y -= BLOCK_SIZE

self.head = Point(x, y)

```

- Mendefinisikan nilai gradient policy untuk menentukan kapan akan melakukan explorasi maupun eksploitasi.

```

def get_action(self, state):
    # random moves: tradeoff exploration / exploitation
    self.epsilon = 80 - self.n_games
    final_move = [0, 0, 0]
    if random.randint(0, 200) < self.epsilon:
        move = random.randint(0, 2)
        final_move[move] = 1
    else:
        state0 = torch.tensor(state, dtype=torch.float)
        prediction = self.model(state0)
        move = torch.argmax(prediction).item()
        final_move[move] = 1

    return final_move

```

- Mendefinisikan model Reward yang akan digunakan.

```

target = pred.clone()
for idx in range(len(done)):
    Q_new = reward[idx]
    if not done[idx]:
        # BELLMAN Equation
        Q_new = reward[idx] + self.gamma * torch.max(self.model(next_state[idx]))

    target[idx][torch.argmax(action[idx]).item()] = Q_new

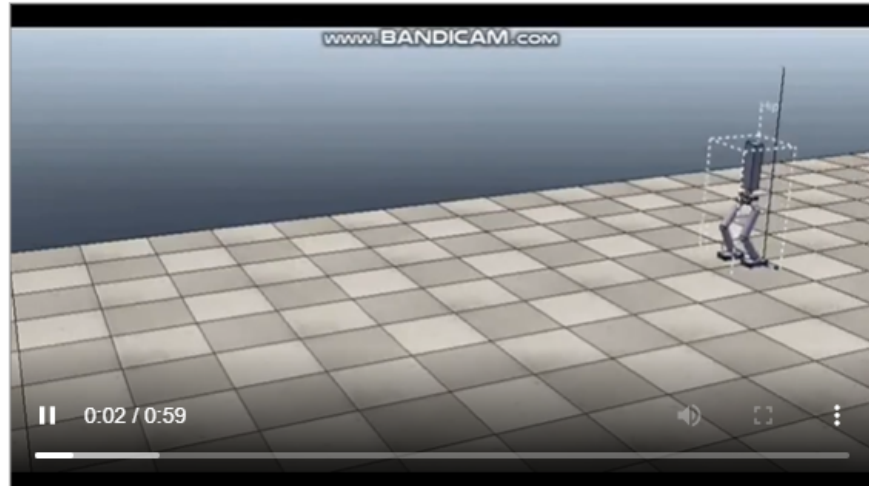
```

RL for Robotics

Examples of building trajectory generation for Humanoid Robot using Reinforcement Learning.

➤ State: Jumlah joint atau derajat kebebasan (DoF) dari suatu robot dalam bentuk sudut radian.

➤ Action: Posisi koordinat kartesian X,Y,Z pada setiap kaki (6 actions).



Examples of Self Driving Car using Reinforcement Learning.

➤ Motion planning, trajectory optimization, dynamic pathing, controller optimization, dan scenario-based learning policies merupakan salah satu bagian dari aplikasi dari self-driving car dengan menggunakan reinforcement learning.

➤ Places, Driveable zones, Obstacle avoidance, Velocity, dan Direction merupakan beberapa variable yang bisa didefinisikan sebagai state dan action pada reinforcement learning model untuk self-driving car.

Source:

<https://wayve.ai/blog/learning-to-drive-in-a-day-with-reinforcement-learning/>



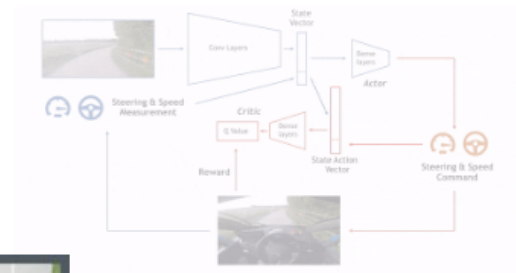
Deep deterministic policy gradients, DDPG



Simulation Result



Self Driving Algorithm



Source:

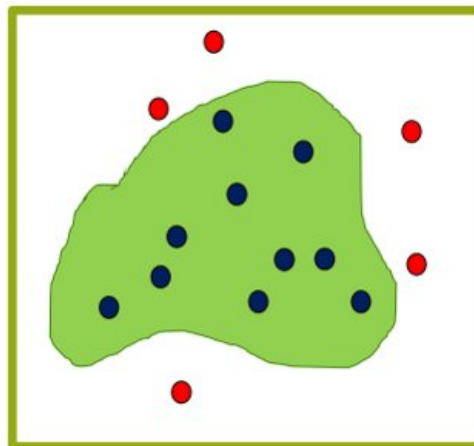
<https://wayve.ai/blog/learning-to-drive-in-a-day-with-reinforcement-learning/>

RL for Optimization

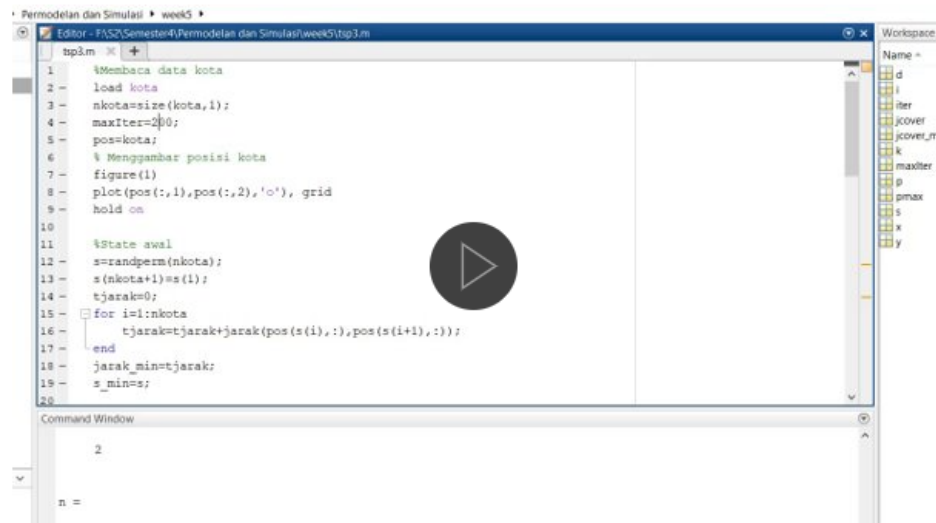
Berikut beberapa contoh dari penerapan Reinforcement Learning untuk Optimasi:

1. Menentukan luasan suatu area dengan pendekatan optimasi monte carlo prediction.
2. Menentukan model suatu fungsi matematika dengan pendekatan optimasi monte carlo prediction
3. Membuat solusi dari permasalahan kasus Traveling Salesman Problem.
4. Membuat suatu sistem Simultaneous Localization and Mapping (SLAM) untuk mengetahui posisi object terhadap lingkungan dalam koordinat lokal maupun koordinat global.

Determine the value of an area using monte carlo prediction



Monte Carlo Prediction for solving Traveling Salesman Problem



```
tp3.m
1 %Membaca data kota
2 load kota
3 nkota=size(kota,1);
4 maxIter=200;
5 pos=kota;
6 % Menggambar posisi kota
7 figure(1)
8 plot(pos(:,1),pos(:,2),'o'), grid
9 hold on
10
11 %State awal
12 s=randperm(nkota);
13 s(nkota+1)=s(1);
14 tjarak=0;
15 for i=1:nkota
16     tjarak=tjarak+jarak(pos(s(i),:),pos(s(i+1),:));
17 end
18 jarak_min=tjarak;
19 s_min=s;
20
```

Command Window

```
2
n =
```

Workspace

- d
- i
- iter
- jcover
- jcover_r
- k
- maxIter
- p
- pmax
- s
- x
- y