

Laporan Machine Learning Reinforcement Q-Learning

Nama : Kartini Nurfalah

Nim : 1301154577

Kelas : IF-39-01

Problem Statement

Bangunlah sebuah sistem Q-learning untuk menemukan optimum policy sehingga Agent yang berada di posisi Start (1,1) mampu menemukan Goal yang berada di posisi (10,10) dengan mendapatkan Total Reward maksimum pada grid world dalam Figure 1 berikut ini. Data pada Figure 1 dapat dilihat di file DataTugasML3.txt. Pada kasus ini, Agent hanya bisa melakukan empat aksi: N, E, S, dan W yang secara berurutan menyatakan North (ke atas), East (ke kanan), South (ke bawah), dan West (ke kiri). Anda boleh menggunakan skema apapun dalam mengimplementasikan sebuah episode.

10	-1	-3	-5	-1	-3	-3	-5	-5	-1	100
9	-2	-1	-1	-4	-2	-5	-3	-5	-5	-5
8	-3	-4	-4	-1	-3	-5	-5	-4	-3	-5
7	-3	-5	-2	-5	-1	-4	-5	-1	-3	-4
6	-4	-3	-3	-2	-1	-1	-1	-4	-3	-4
5	-4	-2	-5	-2	-4	-5	-1	-2	-2	-4
4	-4	-3	-2	-3	-1	-3	-4	-3	-1	-3
3	-4	-2	-5	-4	-1	-4	-5	-5	-2	-4
2	-2	-1	-1	-4	-1	-3	-5	-1	-4	-1
1	-5	-3	-1	-2	-4	-3	-5	-2	-2	-2
	1	2	3	4	5	6	7	8	9	10

Figure 1: Sebuah *grid world* ukuran 10 x 10, di mana angka-angka dalam kotak menyatakan *reward*. *Agent* berada di posisi *Start* (1,1) dan *Goal* di posisi (10,10)

Q-Learning Algorithm

- ```
1 The Q-Learning algorithm goes as follows:
2 > 1. Set the gamma parameter, and environment rewards in matrix R.
3 > 2. Initialize matrix Q to zero.
4 > 3. For each episode:
5 > Select a random initial state.
6 > Do While the goal state hasn't been reached.
7 - Select one among all possible actions for the current state.
8 - Using this possible action, consider going to the next state.
```

```

9 - Get maximum Q value for this next state based on all possible
actions.
10 - Compute: $Q(\text{state}, \text{action}) = R(\text{state}, \text{action}) + \text{Gamma} * \text{Max}[Q$
(next state, all actions)]
11 - Set the next state as the current state.
12 > End Do
13 > End Fo

```

## Syntax and algorithm

Berikut ini merupakan algoritma dan syntax yang saya bangun untuk bisa mencapai Q-Learning

### Dependencies :

```

1 import numpy as np
2 import random

```

### Load Data Reward :

```
Soal = np.loadtxt('DataTugasML3.txt')
```

### Set Matriks Reward :

```

1 Result = []
2 for i, reward in enumerate(Soal):
3 for j, item in enumerate(reward):
4 UP= 0
5 RIGHT= 0
6 DOWN = 0
7 LEFT= 0
8 if (i != 0): UP = Soal[i-1][j]
9 if (i != len(Soal) - 1): DOWN = Soal[i+1][j]
10 if (j != 0): LEFT = Soal[i][j-1]
11 if (j != len(Soal) - 1): RIGHT = Soal[i][j+1]
12 Result.append([UP, RIGHT, DOWN, LEFT])

```

Pada proses diatas kita akan membuat matriks 100 \* 4 dimana 4 merupakan *possible action* atau *transition* dari Agent. Kombinasi transition yang digunakan yaitu mulai dari UP, RIGHT, DOWN, LEFT atau bisa dibilang searah jarum jam. Jika pada Agent tidak bisa bergerak (*moved*) ke salah satu *possible transition* maka akan di set dengan nilai 0, jika Agent bisa melakukan perpindahan berdasarkan matriks *reward* pada soal maka kita set dengan nilai yang sesuai pada matriks *reward* nya.

### Define Possible Movement :

```
1 PossibleMovement = []
2 for i, x in enumerate(Result):
3 tmp = []
4 for j, z in enumerate(x):
5 if (z != 0):
6 tmp.append(j)
7 PossibleMovement.append(tmp)
```

Pada tahap ini kita akan mendefinisikan semua kemungkina action untuk Agent dapat berpindah (*moved*) .

### Set Initial Q value to ZERO (0)

```
1 Q = []
2 for i, reward in enumerate(Soal):
3 for j, item in enumerate(reward):
4 Q.append([0,0,0,0])
```

Set nilai matrix Q menjadi 0. Besar matriks Q harus sama dengan matriks reward yang kita definisikan sendiri, bukan pada soal. Dalam hal ini matriks nya akan sebesar 100\*4.

### Function untuk mengetahui next state akan kemana

```
1 def nextState(CurrentState, Action):
2 row = 0
3 col = 0
4 while (CurrentState > 9):
5 row = row + 1
6 CurrentState = CurrentState - 10
7
8 col = CurrentState
9 if (Action == 0): row = row - 1
10 if (Action == 1): col = col + 1
11 if (Action == 2): row = row + 1
12 if (Action == 3): col = col - 1
13
14 result = col
15 while (row != 0):
16 result += 10
17 row = row - 1
18 return result
```

Keterangan Action :

0 == UP (N = North = Ke atas)

1 == RIGHT (E = East = Ke kanan)

2 == DOWN (S = South = Ke bawah)

3 == LEFT (W = West = Ke kiri)

### Q-Learning Algorithm

```
1 log = []
2 GAMMA = 1
3 episode = 1000
4 goalState = 9
5
6 for i in range(episode):
7 state = random.randint(0,99)
8 while (state != goalState):
9 index = random.randint(0, len(PossibleMovement[state]) - 1)
10 actionIndex = PossibleMovement[state][index]
11 actionValue = Result[state][actionIndex]
12
13 next_state = nextState(state, actionIndex)
14 notZeroArray = [x for x in Q[next_state] if x != 0 == 0]
15
16 value_next_state = notZeroArray
17
18 if (value_next_state == []): value_next_state = 0
19
20 QValue = actionValue + (GAMMA * np.amax(value_next_state))
21 Q[state][actionIndex] = QValue
22
23 state = next_state
24 log.append([state, actionIndex])
```

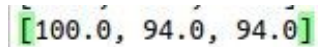
### Keterangan :

- 1 Rentang parameter yaitu  $0 \leq \text{GAMMA} < 1$ , Semakin mendekati ZERO(0) maka AGENT akan bergantung pada immediate reward, Jika mendekati 1 maka akan bergantung pada future reward atau Q.
- 2 Dalam satu episode akan dilakukan proses learning akan dilakukan sampai mencapai *goal-state*. Banyak nya episode yang saya lakukan untuk learning yaitu 1000 Episode.
- 3 `state = random.randint(0,99)` --> dilakukan random untuk initial state
- 4 `notZeroArray = [x for x in Q[next_state] if x != 0 == 0]`--> Jika didalam QValue terdapat nilai 0 yang artinya tidak bisa bergerak pada transition tersebut maka tidak akan di simpan value 0 nya. Figur 2.1 merupakan nilai Q, sedangkan Figure 2.2 hasil setelah di filter sehingga nilai 0 tidak disimpan



[100.0, 0, 90.0, 94.0]

Figure 2.1 Q value



[100.0, 94.0, 94.0]

Figure 2.2 notZeroArray  
dari Q

```
if (value_next_state == []): value_next_state = 0 --> Akan di jalan kan
pada saat nilai Q Value masih ZERO(0) agar program tidak mengalami erro
r
```

#### Save the log activity from Agent from start(1,1) = 90 to goal state(9,9) = 9

```
1 currentState = 90 --> Jika pada matriks reward maka akan sama dengan st
 art (1,1)
2 log2 = [] --> menyimpan state, action(transition 0/1/2/3), nilai Q dan
 dikumulatif)
3 score = 0
4 while (currentState != goalState):
5 max = -9999
6 for i, item in enumerate(Q[currentState]):
7 if (item > max and item != 0): max = item
8
9 actionIndex = Q[currentState].index(max)
10 score += Q[currentState][actionIndex]
11 next_state = nextState(currentState, actionIndex)
12 log2.append([currentState, actionIndex, score])
13 currentState = next_state
14 print(currentState, 'You are arrived!')
15
```

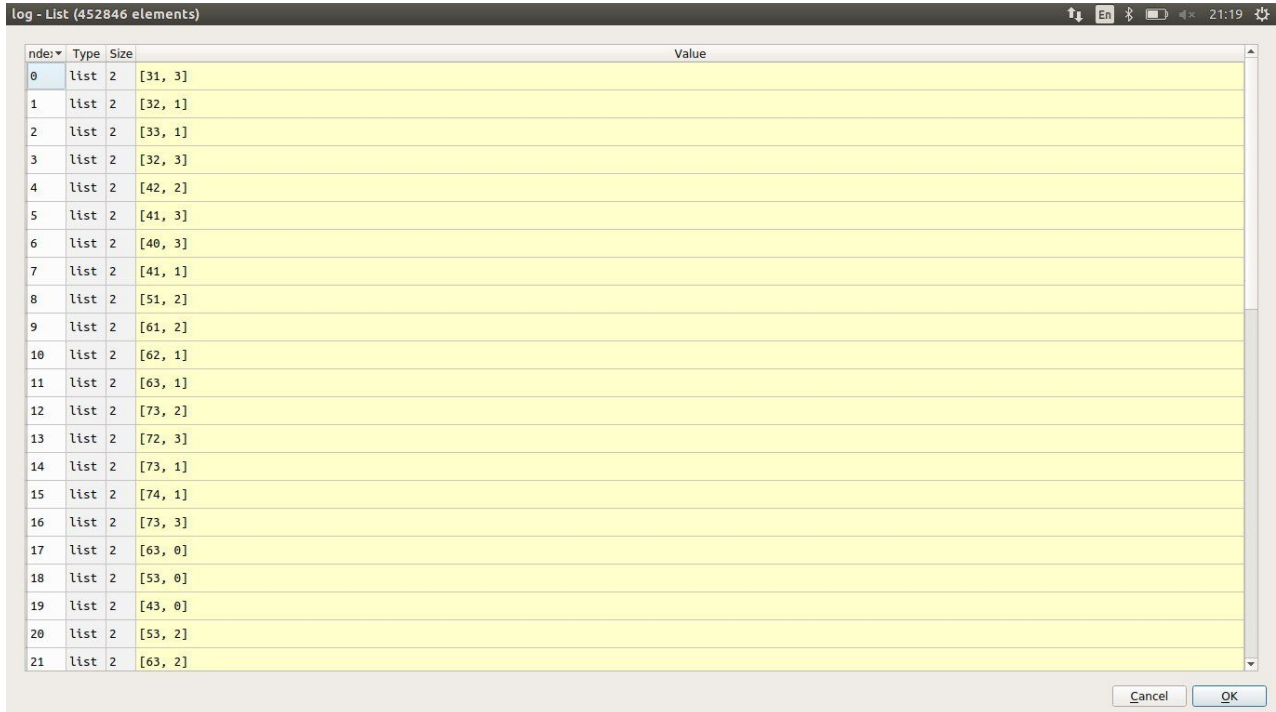
Potongan code diatas merupakan Algorithm to utilize the Q matrix:

1. Set current state = initial state.
2. From current state, find the action with the highest Q value.
3. Set current state = next state.
4. Repeat Steps 2 and 3 until current state = goal state.

The algorithm above will return the sequence of states from the initial state to the goal state.

## Result

Log (State, action)



log - List (452846 elements)

| Index | Type | Size | Value   |
|-------|------|------|---------|
| 0     | list | 2    | [31, 3] |
| 1     | list | 2    | [32, 1] |
| 2     | list | 2    | [33, 1] |
| 3     | list | 2    | [32, 3] |
| 4     | list | 2    | [42, 2] |
| 5     | list | 2    | [41, 3] |
| 6     | list | 2    | [40, 3] |
| 7     | list | 2    | [41, 1] |
| 8     | list | 2    | [51, 2] |
| 9     | list | 2    | [61, 2] |
| 10    | list | 2    | [62, 1] |
| 11    | list | 2    | [63, 1] |
| 12    | list | 2    | [73, 2] |
| 13    | list | 2    | [72, 3] |
| 14    | list | 2    | [73, 1] |
| 15    | list | 2    | [74, 1] |
| 16    | list | 2    | [73, 3] |
| 17    | list | 2    | [63, 0] |
| 18    | list | 2    | [53, 0] |
| 19    | list | 2    | [43, 0] |
| 20    | list | 2    | [53, 2] |
| 21    | list | 2    | [63, 2] |

Cancel OK

Figure 3.1 Log State dan Action

Log (currentState, action, reward/score)

| log2 - List (18 elements) |      |      |                 |
|---------------------------|------|------|-----------------|
| nde                       | Type | Size | Value           |
| 0                         | list | 3    | [90, 0, 65.0]   |
| 1                         | list | 3    | [80, 1, 132.0]  |
| 2                         | list | 3    | [81, 1, 200.0]  |
| 3                         | list | 3    | [82, 1, 269.0]  |
| 4                         | list | 3    | [83, 1, 342.0]  |
| 5                         | list | 3    | [84, 0, 416.0]  |
| 6                         | list | 3    | [74, 0, 491.0]  |
| 7                         | list | 3    | [64, 0, 567.0]  |
| 8                         | list | 3    | [54, 0, 647.0]  |
| 9                         | list | 3    | [44, 1, 728.0]  |
| 10                        | list | 3    | [45, 1, 810.0]  |
| 11                        | list | 3    | [46, 1, 893.0]  |
| 12                        | list | 3    | [47, 0, 980.0]  |
| 13                        | list | 3    | [37, 1, 1068.0] |
| 14                        | list | 3    | [38, 0, 1159.0] |
| 15                        | list | 3    | [28, 0, 1253.0] |
| 16                        | list | 3    | [18, 0, 1352.0] |
| 17                        | list | 3    | [8, 1, 1452.0]  |

Figure 3.2 Log State, Action, Score/Reward

Dari log diatas maka kita bisa mengetahui bahwa Agent akan bergerak dari start(1,1) hingga mencapai goal-state(9,9) melalui blok dengan warna biru dengan total reward kumulative dari soal yaitu sebesar 60. Jika berdasarkan nilai Q maka total kumulative reward sebesar 1452.

|   | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9   |
|---|----|----|----|----|----|----|----|----|----|-----|
| 0 | -1 | -3 | -5 | -1 | -3 | -3 | -5 | -5 | -1 | 100 |
| 1 | -2 | -1 | -1 | -4 | -2 | -5 | -3 | -5 | -5 | -5  |
| 2 | -3 | -4 | -4 | -1 | -3 | -5 | -5 | -4 | -3 | -5  |
| 3 | -3 | -5 | -2 | -5 | -1 | -4 | -5 | -1 | -3 | -4  |
| 4 | -4 | -3 | -3 | -2 | -1 | -1 | -1 | -4 | -3 | -4  |
| 5 | -4 | -2 | -5 | -2 | -4 | -5 | -1 | -2 | -2 | -4  |
| 6 | -4 | -3 | -2 | -3 | -1 | -3 | -4 | -3 | -1 | -3  |
| 7 | -4 | -2 | -5 | -4 | -1 | -4 | -5 | -5 | -2 | -4  |
| 8 | -2 | -1 | -1 | -4 | -1 | -3 | -5 | -1 | -4 | -1  |
| 9 | -5 | -3 | -1 | -2 | -4 | -3 | -5 | -2 | -2 | -2  |

## Referensi

<http://mnemstudio.org/path-finding-q-learning-tutorial.htm>