LAPORAN TUGAS PROGRAM

MACHINE LEARNING

Q-LEARNING

Nama : I Putu Indra Aristya

NIM : 1301154219

Kelas : IF39-09

Mata Kuliah : Pembelajaran Mesin

Universitas Telkom

BANDUNG

2018

I. Deskripsi Masalah

Dalam tugas program ini, mahasiswa diminta untuk menyelesaikan permasalahan pada *grid world* dengan menggunakan metode *q-learning*. Permasalahan yang harus dipecahkan adalah menemukan nilai matriks Q paling optimal sehingga agen dapat berpindah dari titik awal hingga menemukan *goal* sebagai titik akhir dengan mendapatkan *reward* maksimal. Setiap berpindah dari satu titik ke titik lain, dihitung *reward* yang didapatkan sesuai *reward* pada titik tersebut. Gambar 1 adalah *grid* ukuran 10x10 sebagai lintasan dari agen.

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

Gambar 1 Lintasan *Grid World* berukuran 10x10

Nilai-nilai di dalam area suatu titik merupakan nilai *reward* yang dihitung saat titik tersebut dilewati oleh agen. Agen hanya dapat berpindah ke atas, bawah, kiri dan kanan yang di representasikan dengan arah mata angin utara (*north*), selatan (*south*), kiri (*east*) dan kanan (*west*). Posisi awal dari agen adalah titik (1,1) yang ditandai dengan warna kuning dan titik akhir adalah (10,10) yang ditandai dengan warna hijau pada Gambar 1.

II. Landasan Teori dan Desain

Q-Learning adalah salah satu metode dari *reinforcement learning*. Pada *q-learning* terdapat 2 matriks utama yang digunakan, yaitu matriks R yang berisi *reward* dan lintasan yang ingin dilalui serta matriks Q yang berisi nilai perhitungan dari rumus untuk mengubah nilai pada setiap titik Q. Berikut adalah algoritma dari *q-learning*.

Algoritma *Q-Learning*

1. Diinisialisasikan jumlah episode dan matriks Q yang digunakan sesuai representasi
2. Inisialisasi titik awal
3. Lakukan perulangan hingga batas episode
    - Lakukan perulangan hingga mencapai titik akhir
        - Pilih aksi yang dilakukan oleh titik
        - Update nilai Q(titik,aksi) dengan rumus 1.
        - Berpindah ke titik selanjutnya sesuai aksi yang diberikan
4. Didapatkan nilai matriks Q
5. Gunakan nilai matriks Q untuk menelusuri langkah dari titik awal ke titik akhir dengan memilih aksi pada setiap titik dengan melihat arah yang memiliki nilai maksimal pada Q(titik,:)

$$q[state][arah] = q[state][arah] + alpha * (r[newRow][newCol] + (gamma * (max(check\_max) - q[state][arah]))) \quad (1)$$

Pada tugas ini, matriks R adalah lintasan dari *grid world* yang diberikan berukuran 10x10. Pada representasinya, matriks R digambarkan dengan *list* 2 dimensi seperti Gambar 2.

index	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

Gambar 2 Matriks R pada desain Tugas Q-Learning

Matriks Q direpresentasikan dengan ukuran 4x100 seperti pada Gambar 3, dimana terdiri atas 4 baris sebagai representasi arah dan 100 kolom sebagai representasi titik yang diubah menjadi *state*.

	0	1	2	3	4	5	6	7	8	9	10	...	96	97	98	99
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Gambar 3 Representasi matriks Q pada desain Tugas Q-Learning

Pada matrix Q, *state* dibuat menjadi 100 dimana *state* adalah bentuk representasi dari baris dan kolom. Baris dan kolom pada matriks R bernilai sama dengan kolom ke – ((baris x 10) + kolom) pada matriks Q. Jadi misalnya R(9,0) maka pada matriks Q adalah Q(<0/1/2/3>,90). Karena pada program ini matriks Q dibuat dengan *dataframe pandas*, maka saat pemanggilan format diubah menjadi Q[90][<0/1/2/3>] atau Q.iloc[<0/1/2/3>][90]. Baris pada matriks Q merepresentasikan arah, dimana 0 adalah atas/utara, 1 adalah bawah/selatan, 2 adalah kiri/barat dan 3 adalah kanan/timur.

III. *Code* Program

Berikut ditampilkan kode program dari *q-learning*.

```
#I Putu Indra Aristya - 1301154219

import random
from numpy import genfromtxt
import numpy as np
import pandas as pd

#Fungsi ini adalah menentukan arah yang bisa dilalui oleh suatu titik. Sehingga pada
saat random, hasil random adalah
#memang arah yang bisa dilalui
```

```python
def check_possible_direction(row,col):
    direction = ['N','S','W','E']
    if (row == 9):
        if (col == 0):
            direction = ['N','E']
        elif (col == 9):
            direction = ['N', 'W']
        else:
            direction = ['N','E','W']
    if (row == 0):
        if (col == 0):
            direction = ['S','E']
        elif (col == 9):
            direction = ['S','W']
        else:
            direction = ['S', 'E', 'W']
    if (col == 0) and (row != 0) and (row != 9):
        direction = ['N', 'S', 'E']
    if (col == 9) and (row != 0) and (row != 9):
        direction = ['N', 'S', 'W']
    return direction

#Fungsi ini adalah fungsi untuk menentukan nilai next_row atau titik selanjutnya
#Akan dikembalikan row masukkan jika arah tidak sesuai
def next_state_row(direction,row):
    newRow = row
    if (direction == 'N'):
        newRow = row - 1
    elif (direction == 'S'):
        newRow = row + 1
    return newRow

#Fungsi ini adalah fungsi untuk menentukan nilai next_col (next colomn) atau titik
selanjutnya
#Akan dikembalikan col (column) masukkan jika arah tidak sesuai
def next_state_col(direction,col):
    newCol = col
    if (direction == 'W'):
        newCol = col - 1
    elif (direction == 'E'):
        newCol = col + 1
    return newCol

#Fungsi merubah arah menjadi state dalam tabel Q
def convert_to_state(x,y):
    x = x*10
    return x+y

#Fungsi ini merubah direction dalam bentuk huruf menjadi angka
def direction_to_state(direction):
    if (direction == 'N'):
        return 0
    elif (direction == 'S'):
        return 1
    elif (direction == 'W'):
        return 2
    elif (direction == 'E'):
        return 3

#Fungsi ini merubah direction dalam bentuk angka menjadi huruf
def index_to_direction(idx):
    direc = ''
    if (idx == 0):
        direc = 'N'
    elif (idx == 1):
        direc = 'S'
    elif (idx == 2):
        direc = 'W'
    elif (idx == 3):
        direc = 'E'
    return direc


# Matrix r bernilai 10x10 sesuai dengan data yang telah diberikan
```

```
r = genfromtxt('DataTugasML3.txt', delimiter='\t')

# Matrix q bernilai 4x100, dimana 4 adalah arah dan 100 adalah state
# Saat ingin menggunakan matrix q pada index tertentu, formatnya adalah
q.iloc[baris][kolom] atau q[kolom][baris]
q = pd.DataFrame(np.zeros((4, 100)))

gamma = 0.7
alpha = 0.25
eps = 150  # jumlah episode
list_reward = []  # list untuk menyimpan list reward di setiap episode training
list_step = []  # list untuk menyimpan list step di setiap episode training

for i in range(0, eps):
    row = 9
    col = 0
    reward = 0
    step = []
    j = 0
    while (r[row, col] != 100):
        newRow = row
        newCol = col

        # variabel direct akan berisikan arah-arah yang bisa dilalui oleh suatu titik
dilihat dari nilai row dan column
        direct = check_possible_direction(row, col)

        # variabel x akan berisikan hasil random dari list variabel direct
        x = random.choice(direct)

        # ditentukan titik selanjutnya berdasar arah yang terpilih dan titik lama.
        # disimpan ke dalam variabel newRow dan newCol
        newRow = next_state_row(x, row)
        newCol = next_state_col(x, col)

        # arah ditambahkan ke dalam list step
        step.append(x)

        # variable arah akan berisi nilai 0-3 sebagai index representasi dari arah x
        arah = direction_to_state(x)

        # variabel state akan merubah row dan column ke dalam bentuk state untuk tabel
q.
        # variabel next_state juga akan merubah new row dan new column ke dalam bentuk
state untuk tabel q.
        state = convert_to_state(row, col)
        next_state = convert_to_state(newRow, newCol)

        # nilai dari tabel q pada kolom ke-<state> akan disimpan ke variabel check_max
        check_max = [q[next_state][0], q[next_state][1], q[next_state][2],
q[next_state][3]]

        # dilakukan update terhadap nilai q
        q[state][arah] = q[state][arah] + alpha * (r[newRow][newCol] + (gamma *
(max(check_max) - q[state][arah])))
        reward = reward + r[newRow][newCol]

        # row dan column selanjutnya tadi aktan menjadi current row dan column
        row = newRow
        col = newCol
        j += 1

    list_step.append(step)
    list_reward.append(reward)
# print("Reward Episode ",i+1," = ",reward)
print("Best Reward = ", max(list_reward))
print("Direction = ", list_step[list_reward.index(max(list_reward))])

# pengujian dari tabel Q yang didapat

# dibuat variabel road yang berukuran 10x10 bernilai 0
road = r * 0
row = 9
col = 0
```

```
reward = 0

# nilai road[9][0] diberi nilai 1 sebagai tanda telah dilewati
road[row][col] = 1
step = []

while (r[row][col] != 100):
    possible_dir = []
    # variabel state akan merubah row dan column ke dalam bentuk state untuk tabel q.
    state = convert_to_state(row, col)

    # akan disimpan nilai dari tabel q dengan column ke-<state> ke dalam variabel idx
    idx = [q[state][0], q[state][1], q[state][2], q[state][3]]

    # ditentukan index tempat nilai terbesar yang ada dalam list idx
    max_q = idx.index(max(idx))

    # mengubah index tersebut menjadi representasi arah dalam huruf
    arah = index_to_direction(max_q)
    step.append(arah)

    # ditentukan next row dan column
    newRow = next_state_row(arah, row)
    newCol = next_state_col(arah, col)

    # nilai road[next row][next column] diubah menjadi 1 sebagai tanda telah dilewati
    road[newRow][newCol] = 1

    # dihitung reward yang didapatkan dengan melewati titik tersebut
    reward = reward + r[newRow][newCol]
    row = newRow
    col = newCol
print("Reward: ", reward)
print(road)
print(step)
```

IV. Hasil Keluaran Program

Berikut adalah hasil penggunaan matriks Q yang sudah didapat saat proses pembelajaran (*training*) dan didapatkan *reward* adalah 65 dengan urutan arah dapat dilihat pada Gambar 4.

```
Reward:  65.0
[[-0. -0. -0. -0. -0. -0. -0. -0.  1.  1.]
 [-0. -0. -0. -0. -0. -0. -0. -0.  1. -0.]
 [-0. -0. -0. -0. -0. -0. -0. -0.  1. -0.]
 [-0. -0. -0. -0. -0. -0. -0.  1.  1. -0.]
 [-0. -0. -0. -0.  1.  1.  1.  1. -0. -0.]
 [-0. -0. -0. -0.  1. -0. -0. -0. -0. -0.]
 [-0. -0. -0. -0.  1. -0. -0. -0. -0. -0.]
 [-0. -0. -0. -0.  1. -0. -0. -0. -0. -0.]
 [ 1.  1.  1.  1.  1. -0. -0. -0. -0. -0.]
 [ 1. -0. -0. -0. -0. -0. -0. -0. -0. -0.]]
['N', 'E', 'E', 'E', 'E', 'N', 'N', 'N', 'N', 'E', 'E', 'E', 'N', 'E', 'N', 'N', 'N', 'E']
```

Gambar 4 Nilai *reward* dan urutan arah yang dilalui

Pada Gambar 4 juga terdapat matriks yang bernilai 0. Matriks tersebut memiliki nilai 1 yang merupakan langkah yang dilalui oleh agen sehingga mendapatkan *reward* 65.

V. Kesimpulan

Pembelajaran menggunakan *q-learning* dapat dipengaruhi oleh nilai gamma dan juga jumlah episode yang digunakan. *Q-learning* menentukan nilai maksimum dari aksi-aksi yang dapat dilakukan sehingga pemilihan aksi tersebut berdasarkan nilai terbesar dari suatu *state* di tabel Q sesuai arahnya. Gamma berfungsi untuk menentukan nilai *reward* di aksi selanjutnya, sehingga semakin besar nilai gamma maka akan dilihat nilai *reward* masa depan dan semakin kecil nilai gamma maka akan dilihat nilai *reward* dekat. Learning rate (alpha) berguna untuk menentukan perubahan nilai lama yang akan diupdate (*decay value*). Pada hasil yang saya dapat, saya menggunakan episode sebesar 150, gamma 0.7 dan alpha 0.25 dan didapatkan *reward* maksimum yang dapat diperoleh adalah 65.

Referensi

BURLAP. (t.thn.). *Tutorial: Creating a Planning and Learning Algorithm*. Dipetik April 24, 2018, dari BURLAP: http://burlap.cs.brown.edu/tutorials/cpl/p3.html

Q-Learning Tutorial. (t.thn.). Diambil kembali dari Mnemosyne Studio: http://mnemstudio.org/path-finding-q-learning-tutorial.htm