**LAPORAN TUGAS 3**

**PEMBELAJARAN MESIN**

***Q-Learning***



|  |  |
| --- | --- |
| Sarah Fauziah Lestari | 1301154552 |

**Teknik Informatika**

**Fakultas Informatika**

**Telkom University**

**2018**

1. **Analisis Masalah**

Analisis masalah yang didapat adalah menentukan *action* mana saja yang dilakukan agent agar menuju goals dan mendapatkan *reward* yang optimum/maksimal.

1. **Pembangunan Model**

Dalam pembangunan model ada beberapa tahap yang dilakukan antara lain adalah sebagai berikut :

1. Input Data

Pada input data ini program akan membaca file txt yang berisikan matriks reward untuk setiap state pada *grid world*.

1. Mengubah data kedalam bentuk list

Data yang tadinya berbentuk matriks di ubah menjadi bentuk list artinya setiap state diberi nama dari 1 sampai 100 karena jumlah state semuanya adalah 10 x 10 = 100. Contohnya untuk state (1,1) diberi nama state 1, state (1,2) state 2 dan begitupun selanjutnya samapi di state (10,10) menjadi state 100. Setiap state akan memiliki reward masing masing, untuk itu dibuatlah list untuk setiap state dan rewardnya masing-masing.

1. *Weighting Action*

Terdapat empat *action*  yaitu North , East , West, dan South. Setiap *action* diberi nilai untuk perpindahan state. Karena dalam program ini berbentuk indeks maka yang digunakan adalah *state + action*.

North (N) = 10

East (E) = 1

West (W) = -1

South (S) = -10

1. *Initial State*

Untuk *state* awal yang digunakan adalah state 1

1. *Action Rules*

*Action Rules* ini adalah aturan dimana setiap state mempunya langkah-langkah yang diperbolehkan karena menyesuaikan dengan matriks pada soal / *grid world. Rules* yang dimaksud adalah :

* 1. Jika 1 < state < 10 maka langkah yang diperbolehkan hanya N, E, W
  2. Jika state adalah 11 , 21, 31, 41, 51, 61, 71, 81 maka langkah yang diperbolehkan adalah N, W, S
  3. Jika state adalah 90, 80, 70, 60, 50, 40, 30, 20 maka langkah yang diperbolehkan adalah N, W, S
  4. Jika 90 < state < 100 maka langkah yang diperbolehkan adalah W, E, S
  5. Jika state = 1 langkah yang diperbolehkan adalah N, dan E
  6. Jika state = 91 langkah yang diperbolehkan adalah S, E
  7. Jika state = 10 langkah yang diperbolehkan adalah N , W

1. Membuat fungsi maksimum

Fungsi maksimum digunakan untuk menemukan Q pada *action* mana yang mempunyai nilai maksimum

1. Membuat fungsi perhitungan

Fungsi perhitungan untuk mendapatkan Q yang baru dari sebuah *state, rumusnya adalah :*

|  |
| --- |
| Q(s,a) = Q(s,a) + alfa \* (r + gamma \* qMax(state+action) – Q(s,a) |

1. Menentukan parameter alpha dan gamma

Nilai alpha dan gamma berada pada sekitaran nilai : 0 < = 1. Jika bernilai 0 maka tidak penting perhitungan table Q. Oleh karena itu alfa dan gamma di beri nilai 1 masing-masing.

1. Menentukan per-*episode*

Membuat *looping* dengan kondisi state != 100, program akan terus me-*looping* sampai state = *goal state*. Proses tersebut disimpan menjadi satu episode untuk satu table Q. Maka untuk mengetahui yang optimum perlu di *looping* sampai table Q terisi semua ( semua *state* telah belajar / *learning)*. Episode yang digunakan berjumlah 10 .

1. Mencoba dengan jumlah episode yang berbeda

Episode yang dicoba (n) adalah 20 dan 2000 dengan beberapa kali *running* program untuk mendapatkan *total reward*.

1. Menentukan total *reward*

Total *reward* didapat dari table Q terakhir dari *learning* semua episode.

1. **Eksperimen**

*Goal*  dari program ini adalah setiap *state* belajar untuk mengetahui *action* mana yang terbaik untuk *state* tersebut. Dari *state* 1 sampai ke *state* 100. Episode yang diambil adalah 20 dan 2000 untuk melihat perbandingan total reward yang didapat.

Dari hasil program yang dilakukan didapatkan **contoh** hasil untuk 10 episode adalah sebagai berikut :

**Episode 10**

|  |  |
| --- | --- |
| **Tabel Q** | **State** |
| [[56.0, 60.0, 0, 0], [63.0, 63.0, 50.0, 0], [64.0, 63.0, 60.0, 0], [65.0, 65.0, 58.0, 0], [69.0, 63.0, 63.0, 0], [66.0, 64.0, 65.0, 0], [-11.0, 69.0, 63.0, 0], [71.0, 70.0, -12.0, 0], [72.0, 69.0, 69.0, 0], [71.0, 0, 70.0, 0], [48.0, 63.0, 0, 51.0], [62.0, 64.0, 61.0, 60.0], [59.0, 65.0, 63.0, 63.0], [65.0, 69.0, 64.0, 63.0], [70.0, -12.0, 57.0, 65.0], [66.0, 66.0, 69.0, 63.0], [68.0, 71.0, 62.0, -12.0], [71.0, 72.0, 66.0, 69.0], [76.0, 71.0, -6.0, 70.0], [72.0, 0, 72.0, 69.0], [44.0, 62.0, 0, 52.0], [64.0, 62.0, 48.0, 63.0], [67.0, 66.0, 61.0, 64.0], [69.0, 70.0, 59.0, 65.0], [71.0, 66.0, 66.0, 69.0], [70.0, 68.0, 70.0, 66.0], [73.0, 71.0, 66.0, 63.0], [75.0, 76.0, 68.0, 71.0], [78.0, 72.0, 71.0, 72.0], [75.0, 0, 76.0, 71.0], [45.0, 64.0, 0, 48.0], [66.0, 67.0, 44.0, 62.0], [67.0, 69.0, 64.0, 62.0], [72.0, 71.0, 67.0, 66.0], [72.0, 69.0, 69.0, 70.0], [73.0, 73.0, 71.0, 66.0], [77.0, 75.0, 70.0, 68.0], [78.0, 78.0, 73.0, 71.0], [79.0, 75.0, 75.0, 76.0], [78.0, 0, 78.0, 72.0], [49.0, 66.0, 0, 44.0], [68.0, 67.0, 45.0, 63.0], [72.0, 72.0, 66.0, 64.0], [75.0, 73.0, 67.0, 69.0], [77.0, 73.0, 72.0, 71.0], [78.0, 78.0, 73.0, 70.0], [79.0, 78.0, 73.0, 73.0], [80.0, 79.0, 78.0, 75.0], [81.0, 78.0, 78.0, 78.0], [82.0, 0, 79.0, 75.0], [53.0, 68.0, 0, 45.0], [64.0, 72.0, 49.0, 66.0], [69.0, 75.0, 68.0, 67.0], [70.0, 77.0, 71.0, 73.0], [76.0, 78.0, 75.0, 73.0], [75.0, 79.0, 77.0, 72.0], [79.0, 80.0, 78.0, 78.0], [84.0, 81.0, 79.0, 78.0], [84.0, 82.0, 80.0, 79.0], [86.0, 0, 81.0, 78.0], [56.0, 64.0, 0, 49.0], [69.0, 69.0, 53.0, 68.0], [71.0, 70.0, 64.0, 71.0], [75.0, 75.0, 69.0, 75.0], [78.0, 75.0, 70.0, 77.0], [76.0, 79.0, 75.0, 78.0], [81.0, 84.0, 75.0, 79.0], [85.0, 84.0, 79.0, 80.0], [87.0, 86.0, 84.0, 81.0], [90.0, 0, 84.0, 82.0], [67.0, 69.0, 0, 53.0], [73.0, 71.0, 64.0, 64.0], [74.0, 75.0, 69.0, 69.0], [75.0, 78.0, 71.0, 70.0], [81.0, 76.0, 75.0, 77.0], [81.0, 81.0, 78.0, 74.0], [86.0, 85.0, 76.0, 79.0], [89.0, 87.0, 81.0, 84.0], [90.0, 90.0, 85.0, 84.0], [95.0, 0, 87.0, 86.0], [69.0, 73.0, 0, 64.0], [70.0, 74.0, 67.0, 69.0], [70.0, 75.0, 73.0, 71.0], [75.0, 79.0, 74.0, 75.0], [83.0, 81.0, 75.0, 78.0], [74.0, 86.0, 79.0, 76.0], [89.0, 89.0, 81.0, 81.0], [94.0, 94.0, 86.0, 85.0], [99.0, 95.0, 89.0, 87.0], [100.0, 0, 94.0, 90.0], [0, 70.0, 0, 67.0], [0, 70.0, 69.0, 73.0], [0, 75.0, 70.0, 74.0], [0, 83.0, 70.0, 75.0], [0, 86.0, 75.0, 81.0], [0, 89.0, 83.0, 81.0], [0, 94.0, 86.0, 86.0], [0, 99.0, 89.0, 89.0], [0, 100.0, 94.0, 94.0], [0, 0, 0, 0]] | Current State 99  Next STATE 100 |

Keterangan :

Untuk satu indeks dalam list terdapat 3 atribut yaitu :

*List[index][0] = action North*

*List[index][1] = action East*

*List[index][2] = action West*

*List[index][3] = action South*

Berikut adalah **contoh** semua state dengan *action* terbaik hasil *learning.* Dengan cara memilih nilai max dari action untuk setiap state nya.

|  |
| --- |
| [1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 2, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 2, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0] |

Keterangan :

North = 0

East = 1

West = 2

South = 3

1. **Hasil**

Berikut diambil hasil dengan *reward* optimum dari beberapa running dengan 2000 episode.

|  |
| --- |
| Tabel Q [[65.0, 64.0, 0, 0], [67.0, 67.0, 60.0, 0], [68.0, 67.0, 64.0, 0], [69.0, 69.0, 67.0, 0], [73.0, 67.0, 67.0, 0], [70.0, 68.0, 69.0, 0], [70.0, 73.0, 67.0, 0], [75.0, 74.0, 68.0, 0], [76.0, 73.0, 73.0, 0], [75.0, 0, 74.0, 0], [61.0, 67.0, 0, 60.0], [66.0, 68.0, 65.0, 64.0], [66.0, 69.0, 67.0, 67.0], [70.0, 73.0, 68.0, 67.0], [74.0, 70.0, 69.0, 69.0], [70.0, 70.0, 73.0, 67.0], [72.0, 75.0, 70.0, 68.0], [75.0, 76.0, 70.0, 73.0], [80.0, 75.0, 75.0, 74.0], [76.0, 0, 76.0, 73.0], [57.0, 66.0, 0, 65.0], [68.0, 66.0, 61.0, 67.0], [71.0, 70.0, 66.0, 68.0], [73.0, 74.0, 66.0, 69.0], [75.0, 70.0, 70.0, 73.0], [74.0, 72.0, 74.0, 70.0], [77.0, 75.0, 70.0, 70.0], [79.0, 80.0, 72.0, 75.0], [82.0, 76.0, 75.0, 76.0], [79.0, 0, 80.0, 75.0], [57.0, 68.0, 0, 61.0], [70.0, 71.0, 57.0, 66.0], [71.0, 73.0, 68.0, 66.0], [76.0, 75.0, 71.0, 70.0], [76.0, 74.0, 73.0, 74.0], [76.0, 77.0, 75.0, 70.0], [81.0, 79.0, 74.0, 72.0], [81.0, 82.0, 77.0, 75.0], [83.0, 79.0, 79.0, 80.0], [79.0, 0, 82.0, 76.0], [61.0, 70.0, 0, 57.0], [72.0, 71.0, 57.0, 68.0], [75.0, 76.0, 70.0, 71.0], [78.0, 76.0, 71.0, 73.0], [80.0, 76.0, 76.0, 75.0], [81.0, 81.0, 76.0, 74.0], [82.0, 81.0, 76.0, 77.0], [83.0, 83.0, 81.0, 79.0], [85.0, 79.0, 81.0, 82.0], [82.0, 0, 83.0, 79.0], [65.0, 72.0, 0, 57.0], [68.0, 75.0, 61.0, 70.0], [73.0, 78.0, 72.0, 71.0], [74.0, 80.0, 75.0, 76.0], [79.0, 81.0, 78.0, 76.0], [78.0, 82.0, 80.0, 76.0], [82.0, 83.0, 81.0, 81.0], [87.0, 85.0, 82.0, 81.0], [88.0, 82.0, 83.0, 83.0], [86.0, 0, 85.0, 79.0], [68.0, 68.0, 0, 61.0], [72.0, 73.0, 65.0, 72.0], [73.0, 74.0, 68.0, 75.0], [77.0, 79.0, 73.0, 78.0], [78.0, 78.0, 74.0, 80.0], [77.0, 82.0, 79.0, 81.0], [82.0, 87.0, 78.0, 82.0], [87.0, 88.0, 82.0, 83.0], [91.0, 86.0, 87.0, 85.0], [90.0, 0, 88.0, 82.0], [71.0, 72.0, 0, 65.0], [76.0, 73.0, 68.0, 68.0], [77.0, 77.0, 72.0, 73.0], [78.0, 78.0, 73.0, 74.0], [81.0, 77.0, 77.0, 79.0], [81.0, 82.0, 78.0, 78.0], [86.0, 87.0, 77.0, 82.0], [89.0, 91.0, 82.0, 87.0], [94.0, 90.0, 87.0, 88.0], [95.0, 0, 91.0, 86.0], [73.0, 76.0, 0, 68.0], [74.0, 77.0, 71.0, 72.0], [77.0, 78.0, 76.0, 73.0], [82.0, 81.0, 77.0, 77.0], [83.0, 81.0, 78.0, 78.0], [86.0, 86.0, 81.0, 77.0], [89.0, 89.0, 81.0, 82.0], [94.0, 94.0, 86.0, 87.0], [99.0, 95.0, 89.0, 91.0], [100.0, 0, 94.0, 90.0], [0, 74.0, 0, 71.0], [0, 77.0, 73.0, 76.0], [0, 82.0, 74.0, 77.0], [0, 83.0, 77.0, 78.0], [0, 86.0, 82.0, 81.0], [0, 89.0, 83.0, 81.0], [0, 94.0, 86.0, 86.0], [0, 99.0, 89.0, 89.0], [0, 100.0, 94.0, 94.0], [0, 0, 0, 0]]  [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 2, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 2, 1, 1, 1, 0, 0, 1, 0, 1, 0, 2, 1, 0, 1, 0, 0, 0, 0, 0, 0, 2, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 3, 1, 3, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0]  Current State 1  aksi 10  Next State 11  Current State 11  aksi 1  Next State 12  Current State 12  aksi 1  Next State 13  Current State 13  aksi 1  Next State 14  Current State 14  aksi 1  Next State 15  Current State 15  aksi 10  Next State 25  Current State 25  aksi 10  Next State 35  Current State 35  aksi 10  Next State 45  Current State 45  aksi 10  Next State 55  Current State 55  aksi 1  Next State 56  Current State 56  aksi 1  Next State 57  Current State 57  aksi 1  Next State 58  Current State 58  aksi 10  Next State 68  Current State 68  aksi 1  Next State 69  Current State 69  aksi 10  Next State 79  Current State 79  aksi 10  Next State 89  Current State 89  aksi 10  Next State 99  Current State 99  aksi 1  Next State 100  65.0 |

1. **Evaluasi Hasil Eksperimen**

Dari hasil yang didapatkan dengan percobaan 20 episode total *reward* yang didapatkan ada dalam rentan 50 sampai 65 maksimal tetapi dengan total reward berbeda-beda tiap *running-*nya dengan rata-rata 60-an.

Sedangkan dengan menggunakan 2000 episode *reward* yang didapat dari hasil *running* beberapa kali rata-rata *reward* adalah 65.

Kesimpulannya adalah dengan memperbanyak *learning* maka *agent* akan semakin pintar untuk menuju *goals* dengan *reward* yang maksimum.

Untuk mendapatkankan *reward* optimum/maksimal dari hasil *learning* *agent* ke *goals* diperlukan episode yang banyak, ketika *reward* yang didapat tidak berubah meskipun jumlah episode > 2000 dan sudah di *running* beberapa kali maka *agent* sudah belajar dan memang itu adalah jalur yang terbaik/benar.