Laporan Q Learning – Machine Learning

Nama : Fazrian Ramadlan Sumarna

Kelas : IF 39-11

NIM : 1301154221

Masalah

Pembelajaran mesin metode reinforcement learning menjadi suatu pilihan dalam penentuan pengendalian robot. Metode ini mengasumsikan bahwa lingkungan terdefinisi sebagai himpunan keadaan (states) S dengan agen (robot) memiliki pilihan aksi A dengan jumlah tertentu. Untuk setiap langkah, yang didefinisikan sebagai pembagian waktu secara diskrit, agen melakukan pengamatan terhadap keadaan lingkungan, st ,dan memberikan keluaran berupa aksi, at.

Agen mendapatkan suatu reward, R yang menunjukkan kualitas aksi yang diberikan agen berdasarkan ekspektasi pemrogram. Agen kemudian melakukan observasi ulang terhadap lingkungannya, . Keadaan yang dituju dari metode pembelajaran ini ialah mendapatkan experience tuples (st, , ,), dan mendapatkan pembelajaran atas suatu pemetaan keadaan-keadaan untuk mengukur nilai jangka panjang pada keadaan tersebut. Pemetaan tersebut didefinisikan sebagai optimal value function.

Salah satu algoritma reinforcement learning yang dapat digunakan adalah Q-Learning. Fungsi tersebut merepresentasikan nilai reward akibat agent mengambil aksi a dari keadaan s yang mengakibatkan perpindahan keadaan menjadi s’. Parameter merupakan discount factor sebagai ukuran terhadap reward yang pada proses berikutnya. Setelah mendapatkan Q-function yang optimal, terdapat pertimbangan optimasi p*(s) yang merupakan nilai maksimum dari suatu keadaan.

Nilai Q-function disimpan dalam suatu struktur tabel dalam indeks yang mengacu pada state dan action. Untuk setiap waktu robot menghasilkan aksi, experience tuple dihasilkan dan tabel untuk keadaan s dan aksi a.

```
Episode terbaik =  181839
Reward =  55
Step =  18
```

Desain

Pada tugas ini, algoritma Q learning digunakan untuk mencari jalu agar mencapai final state yang sudah ditentukan, dari firstState yang sudah ditentukan. Ukuran table nya yatu 10x10 sehingga total state yang ada itu 100 kemungkinan state yang mana table tersebut saya balik posisinya sehingga initial statenya ada diatas pada kolom pertama dan final state nya ada di bawah kolom 10. Untuk pergerakanya, N (atas) S(bawah) E(kanan) W(kiri) adalah 4 kemungkinan pergerakan yang valid dan ditentukan secara random. setelah mendapatkan pergeraknya kemana (atas,bawah,kiri,kanan) secara acak, lalu kita menghitung nilai Q nya yang akan dimasukan kedalam table Q dengan rumus seperti dibawah ini:

$$Q(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

Gambar 1 rumus q

Setelah itu masukan nilai hasil perhitungan kedalaam table Q yang berukuran 4x100(4 adalah total kemungkinan pergerakan yang valid dan 100 adalah totalState). Setelah itu lakukan step diatas hingga menemukan final state.

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

Gambar 2 table R (setelah di balaik posisi)

```
Table Q

N -6.2   -3.7   -1.2   -2.5   -5   -3.7   -6.2   -2.5   -2.5   -2.5   -2.5   -11.9   -11.3   -39.5   -8.8   -20.9   -39.5   -7.8
S 0   0   0   0   0   0   0   0   0   -2.5   -11.9   -11.1   -40.2   -8.7   -21.1   -37.9   -8   -31.2   -1.2   -5   -20.7   -4
W -6.2   -3.7   -1.2   -2.5   -5   -3.7   -6.2   -2.5   -2.5   0   -2.5   -12   -10.8   -38.7   -8.5   -21.8   -39   -7.9   -31.1
E 0   -3.7   -1.2   -2.5   -5   -3.7   -6.2   -2.5   -2.5   -2.5   0   -11.6   -11.3   -40.1   -8.5   -21.4   -40.6   -7.5   -31.7
```

Gambar 3 table Q (tidak semua masuk kedalam gambar karena tidak muat)

Saya mencoba dengan menggunakan beberapa kali episode, sehingga nanti didapatkan reward maksimal pada episode keberapa. Dengan menggunakana alagoritma sebagai berikut :

```java
public void jumEpisode(){
    int episode = 200000; //totalepisode
    int index = 0; //penunjuk episode
    int reward = 0; //reward
    int loop = 0; //step
    for(int i=0;i<episode;i++){
        System.out.println("Episode ke : "+" "+i);
        move();
        if(i == 0){
            index = i;
            reward = rewards.get(i);
            loop = looping.get(i);
        } else{
            if(rewards.get(i) > reward){
                reward = rewards.get(i);
                loop=looping.get(i);
                index=i;

            }
        }


    }
    System.out.println("Episode terbaik = "+" "+index);
    System.out.println("Reward = " +" "+rewards.get(index));
    System.out.println("Step = " +" "+looping.get(index));
}
```

Gambar 4 fungsi meloop episode sebanyak 200000 kali dan menamapilkan pada episode keberapa yang mendapatkanreward maksimal

Lalu hasilnya seperti dibawah ini :

```
Episode ke :  199998
Episode ke :  199999
Episode terbaik =  150574
Reward =  51
Step =  20
```

Gambar 5 hasil output

Fungsi – fungsi pada kelas Qleaarning(controller):

```
public void readData(String file) throws FileNotFoundException{
    tR = new int [10][10];
    int r;

    Scanner in = new Scanner(new File(file));
    in.useDelimiter("\n|\t");

    int i = 9;
    int j = -1;
    while(in.hasNext()){
        j++;

        r = Integer.valueOf(in.next().trim());
        tR[i][j] = r;

        if(j==9){
            i--;
            j=-1;
        }
    }


}
```

Gambar 6 Men-read daata pada file yang sudah disediakan

```
public void jumEpisode(){
    int episode = 200000; //totalepisode
    int index = 0; //penunjuk episode
    int reward = 0; //reward
    int loop = 0; //step
    for(int i=0;i<episode;i++){
        System.out.println("Episode ke : "+ "+i);
        move();
        if(i == 0){
            index = i;
            reward = rewards.get(i);
            loop = looping.get(i);
        } else{
            if(rewards.get(i) > reward){
                reward = rewards.get(i);
                loop=looping.get(i);
                index=i;

            }
        }


    }
    System.out.println("Episode terbaik = "+ "+index);
    System.out.println("Reward = " +" "+rewards.get(index));
    System.out.println("Step = " +" "+looping.get(index));
}
```

Gambar 7 fungsi meloop episode sebanyak 200000 kali dan menamapilkan pada episode keberapa yang mendapatkanreward maksimal

```
public void move(){

    int width=0;
    int height=0;
    int otw=0;
    int reward=0;
    int loop = 0;
    double r = tR[width][height]; //path awal
    double r2 = tR[width][height]; //path setelah terandom

    boolean selesai = false;
    while (selesai == false){
        loop++;

        //mulai otw(gerak)
        boolean valid = false;
        while(valid==false){

            Random a = new Random();

            int otwBaru = a.nextInt(4); //4 kemungkinan erak atas bawah kiri kanan
            int newWidth = width;
            int newHeight = height;
            switch(otwBaru){
                case 0:
                    //N (atas)
                    newWidth -=1;
                    break;
                case 1:
                    //S (bawah)
                    newWidth +=1;
                    break;
                case 2:
                    //E (kiri)
                    newHeight -=1;
                    break;
                case 3:
                    //W (kanan)
                    newHeight +=1;
                    break;
                default:
                    break;
            }
```

Gambar 8 fungsi Move (perpindahan atas bawah kiri kanan)

```
            if((newHeight >=0) && (newHeight <=9) && (newWidth >=0) && (newWidth<=9)){
                r2 = tR[newWidth][newHeight];

                if((r2!=r)||(r2>=r)||(r2==100)){
//                  if(r2!=r){
                    otw = otwBaru;
                    width = newWidth;
                    height = newHeight;
                    valid = true;

                }else{
                    valid = false;
                }
            }

        }
        //itung Q(s,a) = Q(s,) + alpha[r+gamma*max_a'Q(s',a')-Q(s,a)]
        r = tR[width][height];
        boolean x=false;

        double max = fMax(width,height);
        double Q = fQ(width,height,otw);
        double hasil = Q + (alpha*(r+(gamma*(max-Q))));
        setQ(width,height,otw,hasil);
        reward +=r;

        if(width==9){
            if(height==9){
                selesai=true;
            }
        }
//          System.out.println(r);

    }

//      System.out.println("total step : "+loop);
    looping.add(loop);
//      System.out.println("Reward : "+reward+"\n");
    rewards.add(reward);

}
```

Gambar 9 fungsi Move (perpindahan atas bawah kiri kanan)

```
public double fMax(int width, int height){
    int p = ((width)*10)+(height);
    double max = 0;
    for(int i =0;i<4;i++){
        if(i==0){
            max = tQ[i][p];
        }else{
            if(tQ[i][p]>max){
                max = tQ[i][p];
            }
        }
    }

    return max;
}
```

Gambar 10 Fungsi untuk mencari nilai maksimal pada table Q

```
public void setQ(int width, int height, int otw, double hasil){
    int p = ((width)*10)+(height);
    tQ[otw][p]=hasil;
}
```

Gambar 11 funsi untuk menset nilai Q pada table Q

```
public double fQ(int width,int height, int otw){
    int p = ((width)*10)+(height);
    return tQ[otw][p];
}
```

Gambar 12 fungsi untuk mencari nilai q

```
public void createTQ(){
    tQ= new double[4][100];
    for(int i=0;i<4;i++){
        for(int j=0;j<100;j++){
            tQ[i][j]=0;
        }
    }
}
```

Gambar 13 fugnsi untuk membuat table Q

```
public String vTableQ(){
    DecimalFormat df = new DecimalFormat("#.#");
    String s = "Table Q"+"\n";
    s+= "\n";
    for(int i=0;i<4;i++){
        switch(i){
            case 0:
                s+= "N"+" ";
                break;
            case 1:
                s+="S"+" ";
                break;
            case 2:
                s+="W"+" ";
                break;
            case 3:
                s+="E"+" ";
                break;
        }
        for(int j=0;j<100;j++){
            s+= df.format(tQ[i][j])+"    ";
        }
        s+="\n";
    }
    return s;
}
```

Gambar 14 fungsi untuk menampilkan table Q

```
public String vTableR(){
    String s = "Table R"+"\n";
    s+="      ";
    for(int i=1;i<=10;i++){
        s+=(i+"    ");

    }
    s+="\n";
    for(int i=0;i<10;i++){
        if(i<9){
            s+= (i+1)+"    ";
        }else{
            s+= (i+1)+"    ";
        }
        for(int j=0;j<10;j++){
            s+= tR[i][j]+"   ";
        }
        s+= "\n";
    }
    return s;
}
```

Gambar 15 fungsi untuk menampilkan taable R

Fungsi pada kelas driver:

```
 */
public static void main(String[] args) throws FileNotFoundException {
    String file = "D:\\KULIAH\\smt6\\Machine Learning\\TUGAS 3\\DataTugasML3.txt";
    Qlearning q = new Qlearning(file);
    System.out.println(q.vTableR());
    System.out.println(q.vTableQ());



}
```

Evaluasi hasil Eksperimen :

Sebenarnya, algoritma yang dibangun sudah mulai bisa learning, bisa dilihat pada saat mencoba untuk melakukan 200000 episode, reward terbaiknya ada di episode ke 150574, yang memperlihatkan bahwa

semakin banyak episode akan semakin membaik. Tetapi yang harus di evaluasinya adalah setelah episode 150574 tidak ada reward yang lebih baik lagi.

Kesimpulan

Qlearning adalaha suatu metode untuk mencari jalan dengan cara agentnya belajar. Seharusnya semakin lama agent akan semakin belajar dengan harapan mendapat reward terbaik. Dari eksperimen yang sudah dilakukan didapatkan reward terbaiknya 51 dengan jumlah stepnya 20, pada episode ke 150574.