D3_solution

Son Min Sang

2021-02-03

차 례

Exercise 1 2

Exercise 2 2

Exercise 3 3

Exercise 4 3

Exercise 5 4

Exercise 1

Problem

How would you genalize this game with arbitraty value of m_1 (minimum increment),m_2 (maximum increment), and N (the winning number)?

Solution

- $State = S = \{1, 2, \ldots, N\}$
- $Action = A = \{a_{m1}, a_{m1} + 1, \ldots, a_{m2}\}$
- a_m means the action of incrementing the previous number by m
- $Reward = R(N - m_1, m_1) = R(N - m_2, m_2) = 1$ and other $R(s, a) = 0$.

When $State = N - a_{m1}$, optimal action $= a_{m1}$

When $State = N - a_{m2}$, optimal action $= a_{m2}$

When $State = N - a_{m1} - l - k * (a_{m1} + a_{m2})$ $(l \leq a_{m2} - a_{m1})$ $(k : integer)$,

optimal action $= a_{m1} + l$

Exercise 2

Problem

Two players are to play a game. The two players take turns to call out integers. The rules are as follows. Describe A's winning strategy.

- A must call out an integer between 4 and 8, inclusive.
- B must call out a number by adding A's last number and an integer between 5 and 9, inclusive.
- A must call out a number by adding B's last number and an integer between 2 and 6, inclusive.
- Keep playing until the number larger than or equal to 100 is called by the winner of this game.

Solution

- $S = \{1, 2, \ldots, 100\}$
- $A_{start} = \{4, 5, 6, 7, 8\}$
- $A_B = \{5, 6, 7, 8, 9\}$
- $A_A = \{2, 3, 4, 5, 6\}$
- $Reward = R(100 - A_A, A_A) = R(N - A_B, A_B) = 1$ and other $R(S, A) = 0$.

There is no A's winning strategy

Exercise 3

Problem

There is only finite number of deterministic stationary policiy. How many is it?

Solution

If we say number of states as S, and number of possible actions as A

$|\prod| = |A|^{|S|}$

Exercise 4

Problem

Formulate the first example in this lecture note using the terminology including state,action, reward, policy, transition. Describe the optimal policy using the terminology as well.

Solution

State :

$S = \{1, 2, \dots, 31\}$

Action :

$A = \{a_1, a_2\}$

a_m means the action of incrementing the previous number by m

Reward :

$R(30, a_1) = R(29, a_2) = 1$ otherwise $R(s, a)$ = 0

Transition :

$P^a_{ss'} = P(S_{t+1} = S'|S_t = s, A_t = a) = 1$

$s' = s + 1, if(a = a_1)$

$s' = s + 2, if(a = a_2)$

otherwise 0.

Optimal Policy :

$\pi^* = argmax_\pi V_t(s)^\pi$

$S(3n - 1) : a_2$

$S(3n) : a_1$

Exercise 5

Problem

From the first example,

- Assume that your opponent increments by 1 with prob. 0.5 and by 2 with prob. 0.5.
- Assume that the winning number is 10 instead of 31.
- Your opponent played first and she called out 1.
- Your current a policy π_0 is that
- If the current state $s \leq 5$ then increment by 2.
- If the current state $s > 5$ then increment by 1.

Evaluate $V^{\pi_0}(1)$.

Solution

```
import numpy as np
R=np.array([[0],[0],[0],[0],[0],[0],[0],[0],[0],[1]])
P=np.array([[0,0,0,0.5,0,0.5,0,0,0,0],
           [0,0,0,0,0.5,0.5,0,0,0,0],
           [0,0,0,0,0,0.5,0.5,0,0,0],
           [0,0,0,0,0,0,0.5,0.5,0,0],
           [0,0,0,0,0,0,0,0.5,0.5,0],
           [0,0,0,0,0,0,0,0.5,0.5,0],
           [0,0,0,0,0,0,0,0,0.5,0.5],
           [0,0,0,0,0,0,0,0,0,1],
           [0,0,0,0,0,0,0,0,0,1],
           [0,0,0,0,0,0,0,0,0,1]])
states=[1,2,3,4,5,6,7,8,9,10]
gamma=0.9
epsilon=10**(-8)

v_old = np.array([[0],[0],[0],[0],[0],[0],[0],[0],[0],[0]])
v_new = R+gamma*np.dot(P,v_old)
results = np.array(v_old) # to save
results= np.append(results,v_new,axis=0) # to save
while np.max(np.abs(v_new-v_old))>epsilon:
    v_old=v_new
    v_new=R+gamma*np.dot(P,v_old)
    results= np.append(results,v_new) # to save
```

```
import pandas as pd
results = pd.DataFrame(np.array(results).reshape(len(results)//10,10),columns = states)
```

```
results.head()
```

```
##          1      2       3       4      5      6      7      8      9     10
## 0  0.000000  0.000  0.0000  0.0000  0.000  0.000  0.000  0.000  0.000  0.000
## 1  0.000000  0.000  0.0000  0.0000  0.000  0.000  0.000  0.000  0.000  1.000
## 2  0.000000  0.000  0.0000  0.0000  0.000  0.000  0.450  0.900  0.900  1.900
## 3  0.000000  0.000  0.2025  0.6075  0.810  0.810  1.260  1.710  1.710  2.710
## 4  0.637875  0.729  0.9315  1.3365  1.539  1.539  1.989  2.439  2.439  3.439
```

```
results.tail()
```

```
##            1     2       3       4    5    6     7    8    9    10
## 172  7.198875  7.29  7.4925  7.8975  8.1  8.1  8.55  9.0  9.0  10.0
## 173  7.198875  7.29  7.4925  7.8975  8.1  8.1  8.55  9.0  9.0  10.0
## 174  7.198875  7.29  7.4925  7.8975  8.1  8.1  8.55  9.0  9.0  10.0
## 175  7.198875  7.29  7.4925  7.8975  8.1  8.1  8.55  9.0  9.0  10.0
## 176  7.198875  7.29  7.4925  7.8975  8.1  8.1  8.55  9.0  9.0  10.0
```