MarsRover

Tae Hyeon Kwon, undergrad(ITM)

2021-01-12

monte-carlo simulation

```
import numpy as np

list = ['S1','S2','S3','S4','S5','S6','S7']

def forward(this_state):
    next_state = list[list.index(this_state)+1]

    return next_state
print(forward('S4'))
```

S5

```
def backward(this_state):
    next_state = list[list.index(this_state)-1]

    return next_state

print(backward('S4'))
```

S3

```
def mars_simul(this_state):
    n = np.random.uniform()
    next_state=''
    if this_state =='S1':
        if n<=0.6:
            next_state=this_state
        else:
            next_state=forward(this_state)

    if this_state in ['S2','S3','S4','S5','S6']:
        if n<=0.4:
            next_state=forward(this_state)
        elif 0.4<=n<=0.6:
            next_state=this_state

        else:
            next_state=forward(this_state)
```

```
    if this_state== 'S7':

        if n<=0.6:
            next_state=this_state
        else:
            next_state=backward(this_state)

    return next_state

def cost_eval(path):
    cost_path = path.count('S1')*1+path.count('S7')*10
    return cost_path

def MC_t(start_state):
    time_horizon = 10
    num_episode = 1000
    episode_i = 0
    cum_sum_G_i = 0

    while(episode_i<num_episode):
        path = start_state
        for n in range(time_horizon-1):
            this_state=path[-2:]
            next_state=mars_simul(this_state)
            path+=next_state
            G_j=cost_eval(path)
            cum_sum_G_i+=G_j
            episode_i+=1

    t = cum_sum_G_i/num_episode
    return t

print(MC_t('S1'))
```

```
## 4.98
```

```
print(MC_t('S2'))
```

```
## 8.45
```

```
print(MC_t('S3'))
```

```
## 11.78
```

```
print(MC_t('S4'))
```

```
## 19.88
```

```
print(MC_t('S5'))
```

```
## 26.88
```

```
print(MC_t('S6'))
```

```
## 34.31
```

```
print(MC_t('S7'))
```

```
## 44.3
```

Iterative solution

```
import numpy as np
p =np.array([[0.6,0.4,0,0,0,0,0],
             [0.4,0.2,0.4,0,0,0,0],
             [0,0.4,0.2,0.4,0,0,0],
             [0,0,0.4,0.2,0.4,0,0],
             [0,0,0,0.4,0.2,0.4,0],
             [0,0,0,0,0.4,0.2,0.4],
             [0,0,0,0,0,0.4,0.6]])


r =np.array([1,0,0,0,0,0,10])[:,None]
h = 10
v_t1 =np.array([0,0,0,0,0,0,0])[:,None]

print('p',p)
```

```
## p [[0.6 0.4 0.  0.  0.  0.  0. ]
##  [0.4 0.2 0.4 0.  0.  0.  0. ]
##  [0.  0.4 0.2 0.4 0.  0.  0. ]
##  [0.  0.  0.4 0.2 0.4 0.  0. ]
##  [0.  0.  0.  0.4 0.2 0.4 0. ]
##  [0.  0.  0.  0.  0.4 0.2 0.4]
##  [0.  0.  0.  0.  0.  0.4 0.6]]
```

```
print('r',r)
```

```
## r [[ 1]
##  [ 0]
##  [ 0]
##  [ 0]
##  [ 0]
##  [ 0]
##  [10]]
```

```
print('v_t1',v_t1)
```

```
## v_t1 [[0]
##  [0]
##  [0]
##  [0]
##  [0]
##  [0]
##  [0]]
```

```
t = h-1
while(t>=0):
    v_t2 = np.dot(p,v_t1)
    t = t-1
    v_t1 = v_t2
print(v_t2)
```

```
## [[0.]
##  [0.]
##  [0.]
##  [0.]
##  [0.]
##  [0.]
##  [0.]]
```