Homework 4: Infinite-horizon MDP with Python

Dec. 23. 2017
Juntaek Hong

0. Problem description

There are 3 states in total, and there are one or two possible action in each state as follows:

- Standing(s): slow, fast
- Moving(m): slow, fast
- Fallen(f): slow

When taking slow action in each state,

- transition probability is as follows:

next state ->	s	m	f
s	0	1	0
m	0	1	0
f	0.4	0	0.6

- reward is as follows:

next state ->	s	m	f
s	0	1	0
m	0	1	0
f	1	0	-1

When taking fast action in each state,

- transition probability is as follows:

next state ->	s	m	f
s	0	0.6	0.4
m	0	0.8	0.2
f	0	0	0

- reward is as follows:

next state ->	s	m	f
s	0	2	-1
m	0	2	-1
f	0	0	0

At each state, The robot can choose among availabie options.

1. Policy iteration

The algorithm is as described in lecture note 12.

The algorithm is materialized in policy_iteration.py file, which requires two file inputs for parameter: probability_param.xlsx, reward_param.xlsx.

The policy `{'s': 'slow', 'm': 'slow', 'f': 'slow'}` was used for initial policy.

The result is as follows:

- $\delta = 0.9$:

```
[[ 10.        ]
 [ 10.        ]
 [  7.39130435]]
({'s': 'slow', 'm': 'slow', 'f': 'slow'}, 1)
```

- $\delta = 0.7$:

```
...
[[ 3.53003161]
 [ 3.61433087]
 [ 1.35932561]]
({'s': 'slow', 'm': 'fast', 'f': 'slow'}, 2)
```

When $\delta = 0.9$, optimal policy is [slow action, slow action, slow action] in [standing, moving, fallen] states.
When $\delta = 0.7$, optimal policy is [slow action, fast action, slow action] in [standing, moving, fallen] states.

The matrix means value of each state, and the number right after the policy means number of iterations it took to be finished.

When the value of δ is changed, the change in optimal policy is observed. However, when $\delta \geq 0.9$ or $\delta \leq 0.7$, there is no change in optimal policy.

2. Value iteration

The algorithm is as described in lecture note 12.

The algorithm is materialized in policy_iteration.py file, which requires same file inputs.

The result is as follows:

- $\delta = 0.9, \epsilon = 0.3$:

```
[[ 9.91780629]
 [ 9.91780629]
 [ 7.30911064]] {'s': 'slow', 'm': 'slow', 'f': 'slow'} 45
```

- $\delta = 0.9, \epsilon = 0.2$:

```
[[ 9.94607271]
 [ 9.94607271]
 [ 7.33737705]] {'s': 'slow', 'm': 'slow', 'f': 'slow'} 49
```

- $\delta = 0.9, \epsilon = 0.1$:

```
[[ 9.97134082]
 [ 9.97134082]
 [ 7.36264517]] {'s': 'slow', 'm': 'slow', 'f': 'slow'} 55
```

- $\delta = 0.9, \epsilon = 0.01$:

```
[[ 9.99717773]
 [ 9.99717773]
 [ 7.38848208]] {'s': 'slow', 'm': 'slow', 'f': 'slow'} 77
```

- $\delta = 0.9, \epsilon = 0.001$:

```
[[ 9.99972207]
 [ 9.99972207]
 [ 7.39102642]] {'s': 'slow', 'm': 'slow', 'f': 'slow'} 99
```

- $\delta = 0.7, \epsilon = 0.3$:

```
[[ 3.4469504 ]
 [ 3.53124988]
 [ 1.27624483]] {'s': 'slow', 'm': 'fast', 'f': 'slow'} 10
```

- $\delta = 0.7, \epsilon = 0.2$:

```
[[ 3.48932195]
 [ 3.57362121]
 [ 1.31861593]] {'s': 'slow', 'm': 'fast', 'f': 'slow'} 12
```

- $\delta = 0.7, \epsilon = 0.1$:

```
[[ 3.50153485]
 [ 3.58583411]
 [ 1.33082884]] {'s': 'slow', 'm': 'fast', 'f': 'slow'} 13
```

- $\delta = 0.7, \epsilon = 0.1$:

```
[[ 3.52768478]
 [ 3.61198404]
 [ 1.35697877]] {'s': 'slow', 'm': 'fast', 'f': 'slow'} 20
```

- $\delta = 0.7, \epsilon = 0.001$:

```
[[ 3.52975551]
 [ 3.61405477]
 [ 1.3590495 ]] {'s': 'slow', 'm': 'fast', 'f': 'slow'} 26
```

When $\delta = 0.9$, optimal policy is [slow action, slow action, slow action] in [standing, moving, fallen] states.
When $\delta = 0.7$, optimal policy is [slow action, fast action, slow action] in [standing, moving, fallen] states.

For any ϵ value in {0.001, 0.01, 0.1, 0.2, 0.3}, optimal policy were same for both $\delta = 0.9$ $\delta = 0.7$.

Even when $\epsilon = 0.3$, the number of iterations was bigger than the number of iterations in policy iteration. As the value of ϵ increases, number of iteration increases, and the value of each state approaches to the value of each state calculated by policy iteration.