MDPmaze

June 4, 2018

1 MDPmaze

In [1]:
```
import numpy as np
import MDP
import sys

''' Construct a simple maze MDP

  Grid world layout:

  ---------------------
  |  0 |  1 |  2 |  3 |
  ---------------------
  |  4 |  5 |  6 |  7 |
  ---------------------
  |  8 |  9 | 10 | 11 |
  ---------------------
  | 12 | 13 | 14 | 15 |
  ---------------------

  Goal state: 15
  Bad state: 9
  End state: 16

  The end state is an absorbing state that the agent transitions
  to after visiting the goal state.

  There are 17 states in total (including the end state)
  and 4 actions (up, down, left, right).'''

# Transition function: |A| x |S| x |S'| array
T = np.zeros([4,17,17])
a = 0.8;  # intended move
b = 0.1;  # lateral move

# up (a = 0)
```

```
T[0,0,0] = a+b;
T[0,0,1] = b;

T[0,1,0] = b;
T[0,1,1] = a;
T[0,1,2] = b;

T[0,2,1] = b;
T[0,2,2] = a;
T[0,2,3] = b;

T[0,3,2] = b;
T[0,3,3] = a+b;

T[0,4,4] = b;
T[0,4,0] = a;
T[0,4,5] = b;

T[0,5,4] = b;
T[0,5,1] = a;
T[0,5,6] = b;

T[0,6,5] = b;
T[0,6,2] = a;
T[0,6,7] = b;

T[0,7,6] = b;
T[0,7,3] = a;
T[0,7,7] = b;

T[0,8,8] = b;
T[0,8,4] = a;
T[0,8,9] = b;

T[0,9,8] = b;
T[0,9,5] = a;
T[0,9,10] = b;

T[0,10,9] = b;
T[0,10,6] = a;
T[0,10,11] = b;

T[0,11,10] = b;
T[0,11,7] = a;
T[0,11,11] = b;

T[0,12,12] = b;
T[0,12,8] = a;
```

```
T[0,12,13] = b;

T[0,13,12] = b;
T[0,13,9] = a;
T[0,13,14] = b;

T[0,14,13] = b;
T[0,14,10] = a;
T[0,14,15] = b;

T[0,15,16] = 1;
T[0,16,16] = 1;

# down (a = 1)

T[1,0,0] = b;
T[1,0,4] = a;
T[1,0,1] = b;

T[1,1,0] = b;
T[1,1,5] = a;
T[1,1,2] = b;

T[1,2,1] = b;
T[1,2,6] = a;
T[1,2,3] = b;

T[1,3,2] = b;
T[1,3,7] = a;
T[1,3,3] = b;

T[1,4,4] = b;
T[1,4,8] = a;
T[1,4,5] = b;

T[1,5,4] = b;
T[1,5,9] = a;
T[1,5,6] = b;

T[1,6,5] = b;
T[1,6,10] = a;
T[1,6,7] = b;

T[1,7,6] = b;
T[1,7,11] = a;
T[1,7,7] = b;

T[1,8,8] = b;
```

```
T[1,8,12] = a;
T[1,8,9] = b;

T[1,9,8] = b;
T[1,9,13] = a;
T[1,9,10] = b;

T[1,10,9] = b;
T[1,10,14] = a;
T[1,10,11] = b;

T[1,11,10] = b;
T[1,11,15] = a;
T[1,11,11] = b;

T[1,12,12] = a+b;
T[1,12,13] = b;

T[1,13,12] = b;
T[1,13,13] = a;
T[1,13,14] = b;

T[1,14,13] = b;
T[1,14,14] = a;
T[1,14,15] = b;

T[1,15,16] = 1;
T[1,16,16] = 1;

# left (a = 2)

T[2,0,0] = a+b;
T[2,0,4] = b;

T[2,1,1] = b;
T[2,1,0] = a;
T[2,1,5] = b;

T[2,2,2] = b;
T[2,2,1] = a;
T[2,2,6] = b;

T[2,3,3] = b;
T[2,3,2] = a;
T[2,3,7] = b;

T[2,4,0] = b;
T[2,4,4] = a;
```

```
T[2,4,8] = b;

T[2,5,1] = b;
T[2,5,4] = a;
T[2,5,9] = b;

T[2,6,2] = b;
T[2,6,5] = a;
T[2,6,10] = b;

T[2,7,3] = b;
T[2,7,6] = a;
T[2,7,11] = b;

T[2,8,4] = b;
T[2,8,8] = a;
T[2,8,12] = b;

T[2,9,5] = b;
T[2,9,8] = a;
T[2,9,13] = b;

T[2,10,6] = b;
T[2,10,9] = a;
T[2,10,14] = b;

T[2,11,7] = b;
T[2,11,10] = a;
T[2,11,15] = b;

T[2,12,8] = b;
T[2,12,12] = a+b;

T[2,13,9] = b;
T[2,13,12] = a;
T[2,13,13] = b;

T[2,14,10] = b;
T[2,14,13] = a;
T[2,14,14] = b;

T[2,15,16] = 1;
T[2,16,16] = 1;

# right (a = 3)

T[3,0,0] = b;
T[3,0,1] = a;
```

```
T[3,0,4] = b;

T[3,1,1] = b;
T[3,1,2] = a;
T[3,1,5] = b;

T[3,2,2] = b;
T[3,2,3] = a;
T[3,2,6] = b;

T[3,3,3] = a+b;
T[3,3,7] = b;

T[3,4,0] = b;
T[3,4,5] = a;
T[3,4,8] = b;

T[3,5,1] = b;
T[3,5,6] = a;
T[3,5,9] = b;

T[3,6,2] = b;
T[3,6,7] = a;
T[3,6,10] = b;

T[3,7,3] = b;
T[3,7,7] = a;
T[3,7,11] = b;

T[3,8,4] = b;
T[3,8,9] = a;
T[3,8,12] = b;

T[3,9,5] = b;
T[3,9,10] = a;
T[3,9,13] = b;

T[3,10,6] = b;
T[3,10,11] = a;
T[3,10,14] = b;

T[3,11,7] = b;
T[3,11,11] = a;
T[3,11,15] = b;

T[3,12,8] = b;
T[3,12,13] = a;
T[3,12,12] = b;
```

```
        T[3,13,9] = b;
        T[3,13,14] = a;
        T[3,13,13] = b;

        T[3,14,10] = b;
        T[3,14,15] = a;
        T[3,14,14] = b;

        T[3,15,16] = 1;
        T[3,16,16] = 1;

        # Reward function: |A| x |S| array
        R = -1 * np.ones([4,17]);

        # set rewards
        R[:,15] = 100;  # goal state
        R[:,9] = -70;   # bad state
        R[:,16] = 0;    # end state

        # Discount factor: scalar in [0,1)
        discount = 0.95

        # MDP object
        mdp = MDP.MDP(T,R,discount)
```

```
In [2]: [V,nIterations,epsilon] = mdp.valueIteration(initialV=np.zeros(mdp.nStates),tolerance=0.
        print("V = \n",V)
        print('nIterations = ', nIterations)
        print("epsilon = ", epsilon)
```

```
V =
 [  60.62388836   66.03486523   71.80422632   77.09196339   59.81429704
   65.18237783   77.83066489   84.14118981   58.09361039    7.98780239
   84.86704922   91.78159355   69.49584217   76.80962081   91.78159355
  100.            0.        ]
nIterations =  20
epsilon =  0.00807950852154
```

```
In [3]: [policy,V,nIterations] = mdp.policyIteration(np.zeros(mdp.nStates,dtype=int))
        print("V = \n",V)
        print('nIterations = ', nIterations)
        print("Policy : \n", policy)
```

```
V =
 [  60.63246987   66.03893048   71.80621156   77.09294518   59.81939649
   65.18455359   77.83150988   84.14148741   58.09555739    7.98862041
   84.86730311   91.78165027   69.49680342   76.80991341   91.78165027
```

```
  100.            0.        ]
nIterations =  5
Policy :
 [3 3 3 1 3 3 3 1 1 3 3 1 3 3 3 0 0]
```

```
In [4]: [policy,V,nIterations,epsilon] = mdp.modifiedPolicyIteration(np.zeros(mdp.nStates,dtype=
                                                                     tolerance=0.01)
        print(V)
        print('nIterations = ', nIterations)
        print('epsilon = ', epsilon)
        print("Policy : \n", policy)
```

```
[  60.62962384   66.03757326   71.80555343   77.09261722   59.81768682
   65.18383486   77.83122696   84.14138896   58.09491351    7.98834546
   84.86721937   91.78163126   69.49648204   76.80981681   91.78163126
  100.            0.        ]
nIterations =  11
epsilon =  0.00278129145907
Policy :
 [3 3 3 1 3 3 3 1 1 3 3 1 3 3 3 0 0]
```

```
In [5]: [policy,V,nIterations,epsilon] = mdp.modifiedPolicyIteration(np.zeros(mdp.nStates,dtype=
                                                                     tolerance=0.01)
        print(V)
        print('nIterations = ', nIterations)
        print('epsilon = ', epsilon)
        print("Policy : \n", policy)
```

```
[  60.63081999   66.03813751   71.8058303    77.09275346   59.81839226
   65.18413901   77.83134402   84.14143053   58.09518472    7.98845848
   84.8672549    91.78163912   69.49661506   76.80985772   91.78163912
  100.            0.        ]
nIterations =  7
epsilon =  0.00168010988349
Policy :
 [3 3 3 1 3 3 3 1 1 3 3 1 3 3 3 0 0]
```

```
In [6]: [policy,V,nIterations,epsilon] = mdp.modifiedPolicyIteration(np.zeros(mdp.nStates,dtype=
                                                                     tolerance=0.01)
        print(V)
        print('nIterations = ', nIterations)
        print('epsilon = ', epsilon)
        print("Policy : \n", policy)
```

```
[  60.62336865   66.03445601   71.80409785   77.09185985   59.81371806
   65.18228021   77.83056502   84.14117351   58.09362092    7.9876898
```

```
   84.86703929   91.78158649   69.4957765    76.80960823   91.78158649
  100.           0.         ]
nIterations =  5
epsilon =  0.00917847377509
Policy :
 [3 3 3 1 3 3 3 1 1 3 3 1 3 3 3 0 0]
```

```
In [7]: [policy,V,nIterations,epsilon] = mdp.modifiedPolicyIteration(np.zeros(mdp.nStates,dtype=
                                                                    tolerance=0.01)
        print(V)
        print('nIterations = ', nIterations)
        print('epsilon = ', epsilon)
        print("Policy : \n", policy)
```

```
[  60.63000799   66.03776422   71.80564501   77.0926638    59.8179222
   65.18393342   77.83126739   84.14140265   58.09499296    7.9883848
   84.86723093   91.781634     69.49652398   76.80982997   91.781634    100.
    0.         ]
nIterations =  5
epsilon =  0.00242857713537
Policy :
 [3 3 3 1 3 3 3 1 1 3 3 1 3 3 3 0 0]
```

```
In [8]: [policy,V,nIterations,epsilon] = mdp.modifiedPolicyIteration(np.zeros(mdp.nStates,dtype=
                                                                    tolerance=0.01)
        print(V)
        print('nIterations = ', nIterations)
        print('epsilon = ', epsilon)
        print("Policy : \n", policy)
```

```
[  60.63196284   66.0386899    71.80609448   77.09288709   59.81909341
   65.18442545   77.83145984   84.14146988   58.09544119    7.98857186
   84.86728818   91.78164691   69.49674597   76.80989618   91.78164691
  100.           0.         ]
nIterations =  5
epsilon =  0.000564251059252
Policy :
 [3 3 3 1 3 3 3 1 1 3 3 1 3 3 3 0 0]
```

By increasing the partial evaluation iterations, the modifiedPolicyIteration takes fewer interations to converge. Because the more iterations are performed in evaluating the value function, the more precise it is. An accurate value function of the policy will help policy iteration converge faster.