Policy-and-Value-Iteration

December 8, 2017

```
In [1]: import numpy as np
        import random
```

```
In [2]: rewards = np.zeros((81, 1), dtype = int)
        i = 0
        for l in open('rewards.txt'):
            rewards[i][0] = int(l)
            i += 1
```

```
In [3]: def init_matrix(fname):
            action = np.zeros((81, 81), dtype = float)
            for l in open(fname):
                l = l.split()
                action[int(l[0]) - 1, int(l[1]) - 1] = float(l[2])
            return action
```

```
In [4]: a1 = init_matrix('prob_a1.txt')
        a2 = init_matrix('prob_a2.txt')
        a3 = init_matrix('prob_a3.txt')
        a4 = init_matrix('prob_a4.txt')
```

```
In [5]: if sum(a1[random.randint(0,len(a1)-1)]) == 1.0 and \
        sum(a2[random.randint(0,len(a2)-1)]) == 1.0 and \
        sum(a3[random.randint(0,len(a3)-1)]) == 1.0 and \
        sum(a4[random.randint(0,len(a4)-1)]) == 1.0:
            print "LOADED CORRECTLY"
```

```
LOADED CORRECTLY
```

```
In [6]: V = [0.0] * len(a1[0])
        a = [a1, a2, a3, a4]
```

```
In [7]: def init_identity(size):
            I = np.zeros((size,size), dtype = float)
            for i in range(size):
                I[i,i] = 1.0
            return I
```

```
        def max_value_action(action, state, v_k):
            max_value = -10000.00
            max_action = -1
            for i in range(4):
                temp = 0.0
                for k in range(81):
                    temp += action[i][state][k] * v_k[k]
                if temp > max_value:
                    max_value = temp
                    max_action = i
            return max_value, max_action

        def get_matrix_policy(action, pi):
            T = np.zeros((81, 81), dtype = float)
            for i in range(81):
                a = pi[i]
                T[i] = action[a][i]
            return T

        def optimize_policy(a, rewards, V):
            pi_ = [0] * 81
            gamma = 0.9925
            I = init_identity(len(V))
            for k in range(30):
                P = get_matrix_policy(a, pi_)
                old_v = np.matrix(I - gamma * P).I * rewards
                for i in range(len(V)):
                    max_value, max_action = max_value_action(a, i, old_v)
                    V[i] = rewards[i][0] + gamma * max_value
                    pi_[i] = max_action
            return V, pi_

        def optimize_value(a, rewards, V):
            pi = [0] * 81
            gamma = 0.9925
            for k in range(30):
                old_v = list(V)
                for i in range(81):
                    max_value, max_action = max_value_action(a, i, old_v)
                    V[i] = rewards[i][0] + gamma * max_value
                    pi[i] = max_action
            return V, pi

In [8]: val, pi_policy_iter = optimize_policy(a, rewards, V)

In [16]: idx = 0
         board = np.zeros((9,9), dtype = float)
         for i in range(9):
             for j in range(9):
                 board[i,j] = round(val[idx].item(0),2)
                 idx += 1
```

```
In [17]: print "VALUES in 9x9 board"
         board = board.T
         for i in range(board.shape[0]):
             print board[i]
```

```
VALUES in 9x9 board
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.]
[   0.     102.38  103.23  104.1     0.    -133.33   81.4  -133.33    0.  ]
[ 100.7    101.52    0.     104.98  103.78   90.99   93.67   81.4     0.  ]
[   0.       0.     106.78  105.89    0.    -133.33   95.17 -133.33    0.  ]
[   0.       0.     107.67    0.      0.       0.    108.34    0.      0.  ]
[   0.     109.49  108.58    0.      0.    -133.33  109.58 -133.33    0.  ]
[   0.     110.41    0.     114.16  115.12  116.09  123.64  125.25  133.33]
[   0.     111.34  112.27  113.21    0.     122.02  123.18  124.21    0.  ]
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.]
```

```
In [25]: print "POLICY AFTER POLICY ITER"
         pi = pi.T
         for i in range(pi.shape[0]):
             print pi[i]
```

```
POLICY AFTER POLICY ITER
[ 0.  0.  2.  0.  0.  0.  0.  0.  0.]
[ 0.  2.  1.  0.  0.  3.  3.  2.  0.]
[ 0.  2.  0.  3.  3.  0.  0.  2.  0.]
[ 0.  3.  3.  0.  0.  0.  2.  1.  0.]
[ 0.  0.  0.  0.  0.  0.  2.  0.  0.]
[ 0.  0.  0.  0.  0.  0.  2.  2.  0.]
[ 0.  3.  3.  3.  3.  3.  2.  2.  0.]
[ 0.  0.  0.  0.  0.  0.  2.  1.  0.]
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.]
```

```
In [26]: val, pi_value_iter = optimize_value(a, rewards, V)
```

```
In [64]: idx = 0
         board = np.zeros((9,9), dtype = float)
         for i in range(9):
             for j in range(9):
                 board[i,j] = round(val[idx].item(0),2)
                 idx += 1
```

```
In [13]: print "VALUES in 9x9 board"
         board = board.T
         for i in range(board.shape[0]):
             print board[i]
```

```
VALUES in 9x9 board
[   0.      0.    100.7     0.      0.      0.      0.      0.      0. ]
[   0.    102.38  101.52    0.      0.    109.49  110.41  111.34    0.  ]
[   0.    103.23    0.    106.78  107.67  108.58    0.    112.27    0.  ]
[   0.    104.1   104.98  105.89    0.      0.    114.16  113.21    0.  ]
[   0.      0.    103.78    0.      0.      0.    115.12    0.      0.  ]
[   0.   -133.33   90.99 -133.33    0.   -133.33  116.09  122.02    0.  ]
[   0.     81.4    93.67   95.17  108.34  109.58  123.64  123.18    0.  ]
[   0.   -133.33   81.4  -133.33    0.   -133.33  125.25  124.21    0.  ]
[   0.      0.      0.      0.      0.      0.    133.33    0.      0.  ]
```

```
In [27]: idx = 0
         pi = np.zeros((9,9), dtype = float)
         for i in range(9):
             for j in range(9):
                 pi[i,j] = pi_value_iter[idx]
                 idx += 1
```

```
In [28]: print "POLICY AFTER VALUE ITER"
         pi = pi.T
         for i in range(pi.shape[0]):
             print pi[i]
```

```
POLICY AFTER VALUE ITER
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.]
[ 0.  2.  2.  3.  0.  0.  3.  0.  0.]
[ 2.  1.  0.  3.  0.  0.  3.  0.  0.]
[ 0.  0.  3.  0.  0.  0.  3.  0.  0.]
[ 0.  0.  3.  0.  0.  0.  3.  0.  0.]
[ 0.  3.  0.  0.  0.  0.  3.  0.  0.]
[ 0.  3.  0.  2.  2.  2.  2.  2.  0.]
[ 0.  2.  2.  1.  0.  2.  2.  1.  0.]
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.]
```