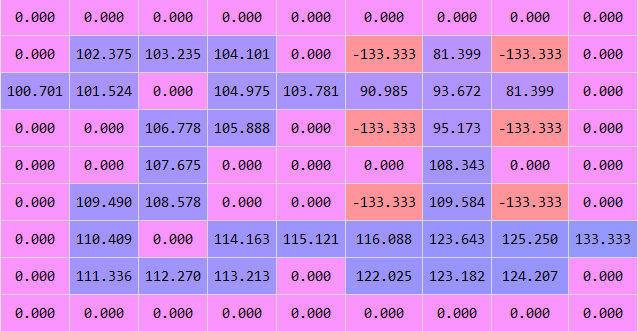
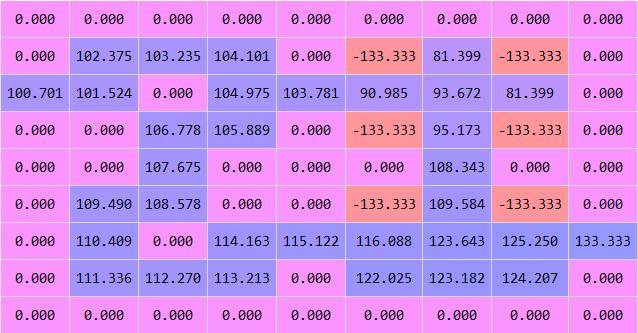
**>>>** V\_Policy**,** V\_Value**,** V\_t **=** GetOptimumStateValues**(**1000000**,** **True,** 0.000001**)**

Grid’s State Values using Value iteration. (epsilon = .000001, Initial Value = 0, Converged after 1836 iteration)



**>>>** P\_Policy**,** P\_Value**,** P\_t **=** GetOptimumPolicy**(**10000**,** 0 **,True)**

Grid’s State Values using Policy iteration (epsilon = .000001, Initial Policy = 0 i.e. ‘WEST’ for every state, Converged after 5 iteration)



The State Values computed in policy iteration and value iterations converged to same value for each block.

c) i) Starting with the initial policy that points EAST in every state, it takes 5 iterations for convergence. (epsilon = 0.000001)

ii) Starting with the initial policy that points SOUTH in every state, it takes 11 iterations for convergence. (epsilon = 0.000001)

#Source

"""

Created on Sat Nov 19 21:47:01 2016

@author: gopal

"""

**import** numpy **as** np

#P(S'={1..81} |S={1..81},a={1,2,3,4})

states **=** 81**;** actions **=** 4**;** gamma **=** 0.9925

#P[action][PreviousState][NextState]

**def** LoadInput**():**

P **=** np**.**zeros**(**shape**=(**actions**,** states**,** states**))**

**for** i **in** range**(**actions**):**

Pa\_raw **=** np**.**loadtxt**(**'prob\_a' **+** str**(**i**+**1**)** **+** '.txt'**)**

**for** item **in** Pa\_raw**:**

P**[**i**][**item**[**0**]-**1**][**item**[**1**]-**1**]** **=** item**[**2**]**

R **=** np**.**loadtxt**(**'rewards.txt'**)**

**return** P**,** R

P**,** R **=** LoadInput**()**

#StateValue(S) = R(S) + gamma\* SumOverS\_dash(P[policy(S)][S][S\_dash] \* StateValue(S\_dash))

**def** EvaluatePolicy**(**Policy**):**

I **=** np**.**identity**(**states**,** dtype**=**int**)**

P\_policy **=** np**.**zeros**(**shape **=** **(**states**,** states**));**

**for** S **in** range**(**states**):**

**for** S\_dash **in** range**(**states**):**

P\_policy**[**S**][**S\_dash**]** **=** P**[**Policy**[**S**]][**S**][**S\_dash**]**

**return** np**.**linalg**.**solve**(**I **-** gamma**\***P\_policy**,** R**)**

**def** GetBetterPolicy**(**Policy**):**

V **=** EvaluatePolicy**(**Policy**);**

Policy\_better **=** np**.**argmax**([**np**.**inner**(**P**[**i**][:][:],** V**)** **for** i **in** range**(**actions**)],** axis**=**0**)**

**return** Policy\_better

**def** GetOptimumPolicy**(**T**,** startPolicy **=** 0**,** stopAtConvergence **=** **False,** epsilon**=**0**):**

Policy **=** np**.**zeros**(**states**)** **+** startPolicy

**for** t **in** range**(**T**):**

betterPolicy **=** GetBetterPolicy**(**Policy**)**

**if** stopAtConvergence **and** np**.**count\_nonzero**(**betterPolicy **-** Policy**)** **==** epsilon**:**

Policy **=** betterPolicy

**break;**

Policy **=** betterPolicy

**return** Policy**,** EvaluatePolicy**(**Policy**),** t

**def** GetBetterStateValues**(**V**):**

V\_better **=** R **+** gamma**\***np**.**max**([**np**.**inner**(**P**[**i**][:][:],** V**)** **for** i **in** range**(**actions**)],** axis**=**0**)**

**return** V\_better

**def** GetOptimumStateValues**(**T**,** stopAtConvergence **=** **False,** epsilon**=**0**):**

Value **=** np**.**zeros**(**states**)**

**for** t **in** range**(**T**):**

betterValue **=** GetBetterStateValues**(**Value**)**

**if** stopAtConvergence **and** np**.**max**(**np**.**abs**(**betterValue **-** Value**))** **<=** epsilon**:**

Value **=** betterValue

**break;**

Value **=** betterValue

Policy\_star **=** np**.**argmax**([**np**.**inner**(**P**[**i**][:][:],** Value**)** **for** i **in** range**(**actions**)],** axis**=**0**)**

**return** Policy\_star**,** Value**,** t