solution7

March 7, 2021

1 HOMEWORK 7

1.0.1 EE 578B - Winter 2021

1.0.2 Due Date: Wednesday, Mar 3rd, 2021 @ 11:59 PM

Consider the Markov Decision Process with the following graph and action structure. (SEE PDF)

1.1 1. Transition Kernel Constraints

(PTS:0-2) Write down the incidence matrices for the graph.

$$E_i \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{E}|}, \quad E_o \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{E}|}, \quad P \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}, \quad A \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}, \quad W \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{A}|}$$

[14]:
```
import numpy as np
Ei = np.array([[0., 1., 0., 0., 1., 0.],
               [0., 0., 0., 1., 0., 0.],
               [1., 0., 0., 0., 0., 1.],
               [0., 0., 1., 0., 0., 0.]])
Eo = np.array([[1., 0., 0., 0., 0., 0.],
               [0., 1., 0., 0., 0., 0.],
               [0., 0., 1., 1., 0., 0.],
               [0., 0., 0., 0., 1., 1.]])
P  = np.array([[0., 1., 0., 0., 1., 0.5],
               [0., 0., 0., .5, 0., 0.],
               [1., 0., 0., 0., 0., .5],
               [0., 0., 1., .5, 0., 0.]])

A  = np.array([[1., 0., 0., 0., 0., 0.],
               [0., 1., 0., 0., 0., 0.],
               [0., 0., 1., 1., 0., 0.],
               [0., 0., 0., 0., 1., 1.]])

W = np.array([[1., 0., 0., 0., 0., 0.],
              [0., 1., 0., 0., 0., 0.],
              [0., 0., 1., .5, 0., 0.],
```

```
              [0., 0., 0., .5, 0., 0.],
              [0., 0., 0., 0., 1., .5],
              [0., 0., 0., 0., 0., .5]])
print('Ei: ')
print(Ei)
print('Eo: ')
print(Eo)
print('P: ')
print(P)
print('A: ')
print(A)
print('W: ')
print(W)
```

```
Ei:
[[0. 1. 0. 0. 1. 0.]
 [0. 0. 0. 1. 0. 0.]
 [1. 0. 0. 0. 0. 1.]
 [0. 0. 1. 0. 0. 0.]]
Eo:
[[1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0.]
 [0. 0. 1. 1. 0. 0.]
 [0. 0. 0. 0. 1. 1.]]
P:
[[0.  1.  0.  0.  1.  0.5]
 [0.  0.  0.  0.5 0.  0. ]
 [1.  0.  0.  0.  0.  0.5]
 [0.  0.  1.  0.5 0.  0. ]]
A:
[[1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0.]
 [0. 0. 1. 1. 0. 0.]
 [0. 0. 0. 0. 1. 1.]]
W:
[[1.  0.  0.  0.  0.  0. ]
 [0.  1.  0.  0.  0.  0. ]
 [0.  0.  1.  0.5 0.  0. ]
 [0.  0.  0.  0.5 0.  0. ]
 [0.  0.  0.  0.  1.  0.5]
 [0.  0.  0.  0.  0.  0.5]]
```

(PTS:0-2) For the incidence matrices given above show the following identities

$$\begin{aligned}
\mathbf{1}^T E_i &= \mathbf{1}^T E_o = \mathbf{1}^T \\
\mathbf{1}^T A &= \mathbf{1}^T P = \\
\mathbf{1}^T W &= \mathbf{1}^T \\
E_i W &= P, \quad E_o W = A
\end{aligned}$$

where the dimension of each $\mathbf{1}$ is determined by context.

[15]:
```
print(np.ones(4)@Ei)
print(np.ones(4)@Eo)
print(np.ones(4)@A)
print(np.ones(4)@P)
print(np.ones(6)@W)
print(Ei@W-P)
print(Eo@W-A)
```

```
[1. 1. 1. 1. 1. 1.]
[1. 1. 1. 1. 1. 1.]
[1. 1. 1. 1. 1. 1.]
[1. 1. 1. 1. 1. 1.]
[1. 1. 1. 1. 1. 1.]
[[0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]]
[[0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0.]]
```

(PTS:0-2) Consider two policies with the following actions chosen from each state

Policy 1:	State 1: Action 1,	State 2: Action 2,
	State 3: Action 4,	State 4: Action 6
Policy 2:	State 1: Action 1,	State 2: Action 2,
	State 3: 50% Action 3 50% Action 4'	State 4: 50% Action 5 50% Action 6

Write each policy in matrix form $\Pi \in \mathbb{R}^{6\times 4}$. Compute the corresponding Markov matrix $M = P\Pi$. Also show that $A\Pi = I$ for each policy.

[16]:
```
Pi1 = np.array([[1., 0., 0., 0.],
                [0., 1., 0., 0.],
                [0., 0., 0., 0.],
                [0., 0., 1., 0.],
                [0., 0., 0., 0.],
                [0., 0., 0., 1.]])
```

```python
Pi2 = np.array([[1., 0., 0., 0.],
                [0., 1., 0., 0.],
                [0., 0., 0.5, 0.],
                [0., 0., 0.5, 0.],
                [0., 0., 0., 0.5],
                [0., 0., 0., 0.5]])
print('Pi1: ')
print(Pi1)
print('Pi2: ')
print(Pi2)
M1 = P@Pi1
M2 = P@Pi2
print('M1: ')
print(M1)
print('M2: ')
print(M2)
```

```
Pi1:
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 1.]]
Pi2:
[[1.  0.  0.  0. ]
 [0.  1.  0.  0. ]
 [0.  0.  0.5 0. ]
 [0.  0.  0.5 0. ]
 [0.  0.  0.  0.5]
 [0.  0.  0.  0.5]]
M1:
[[0.  1.  0.  0.5]
 [0.  0.  0.5 0. ]
 [1.  0.  0.  0.5]
 [0.  0.  0.5 0. ]]
M2:
[[0.   1.   0.   0.75]
 [0.   0.   0.25 0.  ]
 [1.   0.   0.   0.25]
 [0.   0.   0.75 0.  ]]
```

(PTS:0-4) The stationary (state) distribution associated with each Markov chain is the solution to the equation $\rho = M\rho$. Compute this stationary distribution by finding the eigenvector with eigenvalue 1. (You can use the function eig in Matlab or numpy.linalg.eig in Python.). Make sure to scale the eigenvector so that it is an appropriate probability distribution that sums to 1 and has all positive values. Compute the corresponding action distribution y as $y = \Pi\rho$.

[17]:
```
import numpy.linalg as mat
eigs1,V1 = mat.eig(M1);
rho1 = V1[:,np.where(np.abs(eigs1)>=0.99)[0][0]];
rho1 = np.real(rho1); rho1 = (1./np.sum(rho1))*rho1;
y1 = Pi1@rho1;
eigs2,V2 = mat.eig(M2);
rho2 = V2[:,np.where(np.abs(eigs2)>=0.99)[0][0]];
rho2 = np.real(rho2); rho2 = (1./np.sum(rho2))*rho2;
y2 = Pi2@rho2
print('rho1: ',np.round(rho1,2))
print('rho2: ',np.round(rho2,2))
print('y1: ',np.round(y1,2))
print('y2: ',np.round(y2,2))
```

```
rho1:  [0.27 0.18 0.36 0.18]
rho2:  [0.29 0.09 0.36 0.27]
y1:  [0.27 0.18 0.   0.36 0.   0.18]
y2:  [0.29 0.09 0.18 0.18 0.13 0.13]
```

(PTS:0-2) Show that each y from the previous part satisfies $Py = Ay$ and $\mathbf{1}^T y = 1$. Compute the corresponding edge mass vector for each $x = Wy$. Show that x is in the nullspace of $E = E_i - E_o$.

[18]:
```
x1 = W@y1;
x2 = W@y1;

print('x1: ',np.round(x1,2))
print('x2: ',np.round(x2,2))
print('1.T@y1=1: ',np.ones(6)@y1-1)
print('1.T@y2=1: ',np.ones(6)@y2-1)

print('(P-A)y1: ', np.round(P@y1-A@y1,2))
print('(P-A)y2: ', np.round(P@y2-A@y2,2))

print('(Ei-Eo)x1: ', np.round(Ei@x1-Eo@x1,2))
print('(Ei-Eo)x2: ', np.round(Ei@x2-Eo@x2,2))
```

```
x1:  [0.27 0.18 0.18 0.18 0.09 0.09]
x2:  [0.27 0.18 0.18 0.18 0.09 0.09]
1.T@y1=1:  0.0
1.T@y2=1:  0.0
(P-A)y1:  [-0.  0.  0.  0.]
(P-A)y2:  [ 0. -0.  0. -0.]
(Ei-Eo)x1:  [-0.  0.  0.  0.]
(Ei-Eo)x2:  [-0.  0.  0.  0.]
```

1.2 Infinite Horizon, Average Reward

Consider the following optimization problem for finding the optimal steady-state action distribution $y \in \mathbb{R}^{|\mathcal{A}|}$

$$\begin{aligned} \max_{y} \quad & r^T y \\ \text{s.t.} \quad & Py = Ay,\ \mathbf{1}^T y = 1,\ y \geq 0 \end{aligned} \tag{1}$$

for reward vector $r \in \mathbb{R}^{|\mathcal{A}|}$.

(PTS:0-2) Write the dual optimization problem with dual variables $\lambda \in \mathbb{R}$ associated with the constraint $\mathbf{1}^T y = 1$, $v \in \mathbb{R}^{|\mathcal{S}|}$ associated with constraint $Py = Ay$, $\mu \in \mathbb{R}_+^{|\mathcal{A}|}$ associated with the constraint $y \geq 0$.

1.3 Solution

The Lagrangian is given by

$$L(x, \lambda, v, \mu) = r^T y + v^T (P - A) y + \lambda(1 - \mathbf{1}^T y) + \mu^T y$$

The game form is given by

$$\max_{y} \min_{\lambda, v, \mu \geq 0} \quad r^T y + v^T (P - A) y + \lambda(1 - \mathbf{1}^T y) + \mu^T y$$

Switching min and max gives the game form of the dual problem

$$\max_{y} \min_{\lambda, v, \mu \geq 0} L \leq \min_{\lambda, v, \mu \geq 0} \max_{y} \ r^T y + v^T (P - A) y + \lambda(1 - \mathbf{1}^T y) + \mu^T y$$

Solving the inner max problem gives

$$\frac{\partial L}{\partial y} = r^T + v^T (P - A) - \lambda \mathbf{1}^T + \mu^T = 0$$

The dual problem is

$$\begin{aligned} \min_{\lambda, v, \mu} \quad & \lambda \\ \text{s.t.} \quad & \lambda \mathbf{1}^T + v^T A = r^T + v^T P + \mu^T,\ \mu \geq 0 \end{aligned}$$

[ ]:

(PTS:0-2) The KKT conditions at optimum (for either the primal or dual problem) are given by

$$\begin{aligned} r^T - \lambda \mathbf{1}^T + v^T (P - A) + \mu^T = 0, \quad & \mu \geq 0 \\ Py - Ay = 0, \quad \mathbf{1}^T y = 1, \quad & y \geq 0 \\ \mu^T y = 0 & \end{aligned}$$

Use these conditions to show that λ is an upper bound on the primal objective $r^T y$ for any feasible y. What does $\mu^T y$ represent for a specific y? What does the condition $\mu^T y = 0$ imply about the optimal y?

1.4 Solution

For any feasible y, we have that

$$\begin{aligned}\lambda \mathbf{1}^T y &= (r^T + v^T(P-A) + \mu^T)y \\ \lambda \mathbf{1}^T y &= r^T y + v^T(P-A)y + \mu^T y \\ \lambda &= r^T y + \mu^T y\end{aligned}$$

$r^T y$ is the reward of distribution y. $\mu^T y$ is the inefficiency of y. Since $\mu \geq 0$ and $y \geq 0$, $\mu^T y \geq 0$. $\lambda = r^T y + \mu^T y$ and thus λ is an upper bound on $r^T y$. The condition $\mu^T y = 0$ implies that the optimal y is completely efficient (ie. doesn't leave any possible reward on the table.)

(PTS:0-4) Use cvx or cvxpy to solve the above optimization problem for the transition kernel given initially and each reward vector

$$\begin{aligned}r^T &= \begin{bmatrix}1 & 2 & 3 & 4 & 5 & 6\end{bmatrix} \\ r^T &= \begin{bmatrix}1 & 1 & 1 & 1 & 1 & 1\end{bmatrix}\end{aligned}$$

What is the optimal joint distribution y in each case? What is the expected average reward $r^T y$ in each case?

[19]:
```
import cvxpy as cp
r1 = np.array([1.,2.,3.,4.,5.,6.]);
y = cp.Variable(6);
obj1 = r1@y
constraints1 = [(P-A)@y==0,1.-np.ones(6)@y==0,y>=0]
primal1 = cp.Problem(cp.Maximize(obj1),constraints1)
primal1.solve()
y1 = y.value;
mu1  =constraints1[2].dual_value;
lam1 =np.abs(constraints1[1].dual_value);

print('Primal Problem 1: ')
print("The optimal value for the primal problem 1 is ", np.round(primal1.
 ↪value,4))
print('Reward vector: ', r1)
print('Optimal y1: ',np.round(y.value,2))
print('Optimal r1.Ty1: ', np.round(r1@y1,2))
print('Optimal lam1: ',  np.round(lam1,2))

import cvxpy as cp
r2 = np.array([1.,1.,1.,1.,1.,1.]);
y = cp.Variable(6);
obj2 = r2@y
constraints2 = [(P-A)@y==0,1.-np.ones(6)@y==0,y>=0]
primal2 = cp.Problem(cp.Maximize(obj2),constraints2)
primal2.solve()
y2 = y.value;
```

```
mu2  =constraints2[2].dual_value;
lam2 =np.abs(constraints2[1].dual_value);
print('')
print('Primal Problem 2: ')
print("The optimal value for the primal problem 2 is ", np.round(primal2.
 ↪value,4))
print('Reward vector: ', r2)
print('Optimal y2: ',np.round(y.value,2))
print('Optimal r2.Ty2: ', np.round(r2@y2,2))
print('Optimal lam2: ', np.round(lam2,2))
```

```
Primal Problem 1:
The optimal value for the primal problem 1 is  3.8
Reward vector:  [1. 2. 3. 4. 5. 6.]
Optimal y1:  [0.2 0.  0.4 0.  0.  0.4]
Optimal r1.Ty1:  3.8
Optimal lam1:  3.8

Primal Problem 2:
The optimal value for the primal problem 2 is  1.0
Reward vector:  [1. 1. 1. 1. 1. 1.]
Optimal y2:  [0.28 0.09 0.17 0.18 0.11 0.16]
Optimal r2.Ty2:  1.0
Optimal lam2:  1.0
```

(PTS:0-2) What is the steady-state state distribution associated with each solution $\rho = Ay$? What is the optimal policy associated with y? Use the formula

$$(\pi_s)_a = \frac{y_a}{\rho_s} = \frac{y_a}{\sum_{a \in \mathcal{A}_s} y_a}$$

You could also put the policy in matrix form using the formula

$$\Pi = \operatorname{diag}(y) A^T \operatorname{diag}(\rho)^{-1}$$

[20]:
```
rho1 = A@y1;
rho2 = A@y2;
Pi1 = np.diag(y1)@A.T@mat.inv(np.diag(rho1))
Pi2 = np.diag(y2)@A.T@mat.inv(np.diag(rho2))
print('rho1: ',np.round(rho1,2))
print('rho2: ',np.round(rho2,2))
print('Pi1: ')
print(np.round(Pi1,2))
print('Pi2: '),
print(np.round(Pi2,2))
```

```
rho1:  [0.2 0.  0.4 0.4]
rho2:  [0.28 0.09 0.36 0.27]
```

```
Pi1:
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 1.]]
Pi2:
[[1.   0.   0.   0.  ]
 [0.   1.   0.   0.  ]
 [0.   0.   0.49 0.  ]
 [0.   0.   0.51 0.  ]
 [0.   0.   0.   0.42]
 [0.   0.   0.   0.58]]
```

(PTS:0-2) Now suppose you apply the policy

$$\Pi = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.2 & 0 \\ 0 & 0 & 0.8 & 0 \\ 0 & 0 & 0 & 0.2 \\ 0 & 0 & 0 & 0.8 \end{bmatrix}$$

What reward do you achieve in each case? (Hint: compute ρ such that $\rho = P\Pi\rho$ and then y using $y = \Pi\rho$.) How much does this reward differ from the optimal average reward? How does this difference relate to the quantity $\mu^T y$ where μ is the optimal dual variable?

[21]:
```python
Pi3 = np.array([[1.,0.,0.,0.],
                [0.,1.,0.,0.],
                [0.,0.,0.2,0.],
                [0.,0.,0.8,0.],
                [0.,0.,0.,0.2],
                [0.,0.,0.,0.8]])
M3 = P@Pi3
eigs3,V3 = mat.eig(M3);
rho3 = V3[:,np.where(np.abs(eigs3)>=0.99)[0][0]];
rho3 = np.real(rho3); rho3 = (1./np.sum(rho3))*rho3;
y3 = Pi3@rho3;
print('y3: ',np.round(y3,2))
print('r1.T@y3: ', np.round(r1@y3,2))
print('r2.T@y3: ', np.round(r2@y3,2))
print('lam1 - r1.T@y3: ',np.round(lam1-r1@y3,2))
print('lam2 - r2.T@y3: ',np.round(lam2-r2@y3,2))
print('mu1.T@y3: ',np.round(mu1@y3,2))
print('mu2.T@y3: ',np.round(mu2@y3,2))
```

```
y3:  [0.28 0.14 0.07 0.29 0.04 0.17]
r1.T@y3:  3.2
```

```
r2.T@y3:  1.0
lam1 - r1.T@y3:  0.6
lam2 - r2.T@y3:  0.0
mu1.T@y3:  0.6
mu2.T@y3:  0.0
```

1.5 3. Finite Horizon, Total Reward

Consider the following optimization problem for finding the optimal finite horizon policy.

$$
\begin{aligned}
\max_{y(t),\ t\in\mathcal{T}} \quad & \sum_{t=0}^{T-1} r(t)^T y(t) + g^T A y(T) \\
\text{s.t.} \quad & Ay(0) = \rho(0), \quad y(0) \geq 0 \\
& Ay(t+1) = Py(t), \quad y(t+1) \geq 0, \quad t \in \mathcal{T}
\end{aligned} \tag{2}
$$

where $\mathcal{T} = \{0, \ldots, T-1\}$, $\rho(0) \in \mathbb{R}^{|\mathcal{S}|}$ is a given initial state distribution, and $g \in \mathbb{R}^{|\mathcal{S}|}$ is a final cost on each of the states.

(PTS:0-4) Write the dual optimization problem with dual variables $v(0) \in \mathbb{R}^{|\mathcal{S}|}$ associated with the constraint $Ay(0) = \rho(0)$, $v(t+1) \in \mathbb{R}^{|\mathcal{S}|}$ associated with constraint $Py(t) = Ay(t+1)$, and $\mu(t) \in \mathbb{R}_+^{|\mathcal{A}|}$ associated with the constraint $y(t) \geq 0$.

1.5.1 Solution

The Lagrangian is given by

$$
\begin{aligned}
L(x, \lambda, v, \mu) = & \sum_{t=0}^{T-1} r(t)^T y(t) + g^T A y(T) + \\
& v(0)^T \Big(\rho(0) - Ay(0)\Big) + \sum_{t=0}^{T-1} v(t+1)^T \Big(Py(t) - Ay(t+1)\Big) + \sum_{t=0}^{T} \mu(t)^T y(t)
\end{aligned}
$$

The game form is given by

$$
\max_{y} \min_{\lambda, v, \mu \geq 0} \quad L(x, \lambda, v, \mu)
$$

Switching min and max gives the game form of the dual problem

$$
\max_{y} \min_{\lambda, v, \mu \geq 0} L \leq \min_{\lambda, v, \mu \geq 0} \max_{y} \; L(x, \lambda, v, \mu)
$$

Solving the inner max problem gives

$$
\begin{aligned}
\tfrac{\partial L}{\partial y(t)} &= r(t)^T + v(t+1)^T P - v(t)^T A + \mu(t)^T = 0, \quad t \in \mathcal{T} \\
\tfrac{\partial L}{\partial y(T)} &= g^T A - v(T)^T A + \mu(T)^T = 0
\end{aligned}
$$

The dual problem is

$$
\begin{aligned}
\min_{\lambda,v,\mu} \quad & v(0)^T\rho(0) \\
\text{s.t.} \quad & v(t)^T A = r(t)^T + v(t+1)^T P + \mu(t)^T, \quad t \in \mathcal{T} \\
& v(T)^T A = g^T A + \mu(T)^T
\end{aligned}
$$

(PTS:0-4) The KKT optimality conditions for the primal and dual optimization problems are given by

$$
\begin{aligned}
g^T A - v(T)A + \mu(T)^T &= 0, \quad \mu(T) \geq 0 \\
r(t)^T + v(t+1)^T P - v(t)^T A + \mu(t)^T &= 0, \quad \mu(t) \geq 0, \quad t \in \mathcal{T} \\
Ay(0) &= \rho(0), \quad y(0) \geq 0 \\
Ay(t+1) &= Py(t), \quad y(t+1) \geq 0, \quad t \in \mathcal{T} \\
\mu(t)^T y(t) &= 0, \quad t \in \mathcal{T},\ t = T
\end{aligned}
$$

Use these conditions to show that $v(0)^T\rho(0)$ is an upper bound on the primal objective $\sum_t r(t)^T y(t) + g^T Ay(T)$ for any feasible $y(t)$ that satisfies the mass flow equations. What does $\sum_t \mu(t)^T y(t)$ represent for a specific mass flow $y(t), t \in \mathcal{T}$.

1.5.2 Solution

A feasible $y(t)$ satisfies...

$$
\begin{aligned}
Ay(0) &= \rho(0), \quad y(0) \geq 0 \\
Ay(t+1) &= Py(t), \quad y(t+1) \geq 0, \quad t \in \mathcal{T}
\end{aligned}
$$

Using the KKT conditions, multiplying by the appropriate $y(t)$ and summing up gives

$$
\begin{aligned}
v(T)Ay(T) &= g^T Ay(T) + \mu(T)^T y(T) \\
v(t)^T Ay(t) &= r(t)^T y(t) + v(t+1)^T Py(t) + \mu(t)^T y(t), \quad t \in \mathcal{T}
\end{aligned}
$$

$$
\begin{aligned}
0 &= \sum_{t=0}^{T-1} \Big(- v(t)^T Ay(t) + r(t)^T y(t) + v(t+1)^T Py(t) + \mu(t)^T y(t) \Big) - v(T)Ay(T) + g^T Ay(T) + \mu(T)^T y(T) \\
v(0)^T Ay(0) &= \sum_{t=0}^{T-1} r(t)^T y(t) + \sum_{t=0}^{T-1} \Big(- v(t+1)^T Ay(t+1) + v(t+1)^T Py(t) \Big) + \sum_{t=0}^{T} \mu(t)^T y(t) + g^T Ay(T) \\
v(0)^T Ay(0) &= \sum_{t=0}^{T-1} r(t)^T y(t) + \sum_{t=0}^{T-1} v(t+1)^T \Big(- Ay(t+1) + Py(t) \Big) + \sum_{t=0}^{T} \mu(t)^T y(t) + g^T Ay(T) \\
v(0)^T \rho(0) &= \sum_{t=0}^{T-1} r(t)^T y(t) + g^T Ay(T) + \sum_{t=0}^{T} \mu(t)^T y(t)
\end{aligned}
$$

since $\mu(t) \geq 0$ and $y(t) \geq 0$, we have that $\sum_{t=0}^{T} \mu(t)^T y(t) \geq 0$ and $v(0)^T\rho(0)$ is an upper bound on the primal objective. Similarly to the infinite horizon case $\sum_{t=0}^{T} \mu(t)^T y(t)$ is the inefficiency of distribution $y(t)$. For optimal $y(t)$, $\sum_{t=0}^{T} \mu(t)^T y(t) = 0$.

(PTS:0-4) Use cvx or cvxpy to solve the above optimization problem for the MDP given initially with the following rewards

$$r(t)^T = \begin{bmatrix} 2 & 1 & 2 & 1 & 2 & 1 \end{bmatrix} \text{ for } t \in \mathcal{T}, \quad g^T = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$$

for ten time steps $T = 10$ and initial distribution $\rho(0) = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}^T$

What is the optimal action distribution $y(t)$ at each time step? What is the expected total reward $\sum_t r(t)^T y(t)$?

Correction: should read "What is the expected total reward $\sum_{t=0}^{T-1} r(t)^T y(t) + g^T A y(T)$?

[22]:
```
import cvxpy as cp
T = 10; n = 6;
r = np.array([2.,1.,2.,1.,2.,1.]);
g = np.ones(4);
rho0 = 0.25*np.ones(4);

y = cp.Variable([T+1,n]);
obj = 0.
flow=[A@y[0]==rho0];
pos = [y[0]>=0];
for t in range(T):
    obj = obj + r@y[t]
    flow.append(A@y[t+1]==P@y[t]);
    pos.append(y[t+1]>=0);
obj = obj + g@A@y[T];

constraints = flow + pos;
primal = cp.Problem(cp.Maximize(obj),flow+pos)
primal.solve()
# flow = constraints[:T+1];
# pos  = constraints[T+1:];
yopt = y.value;
muopt = np.zeros((T+1,n));
for t in range(T+1): muopt[t] = pos[t].dual_value;

print("The optimal value: ", np.round(primal.value,2))
print('yopt: ')
print(np.round(yopt,2))
print('Note - y(t) stored as row vector')
```

```
The optimal value:  20.75
yopt:
[[0.25 0.25 0.25 0.   0.25 0.  ]
 [0.5  0.   0.25 0.   0.25 0.  ]
 [0.25 0.   0.5  0.   0.25 0.  ]
 [0.25 0.   0.25 0.   0.5  0.  ]
 [0.5  0.   0.25 0.   0.25 0.  ]
```

```
 [0.25 0.   0.5  0.   0.25 0.  ]
 [0.25 0.   0.25 0.   0.5  0.  ]
 [0.5  0.   0.25 0.   0.25 0.  ]
 [0.25 0.   0.5  0.   0.25 0.  ]
 [0.25 0.   0.25 0.   0.5  0.  ]
 [0.5  0.   0.12 0.12 0.12 0.12]]
Note - y(t) stored as row vector
```

(PTS:0-4) What is the policy $\Pi(t)$ chosen at each time step? Use the formula

$$(\pi_s)_a(t) = \frac{y_a(t)}{\rho_s(t)} = \frac{y_a(t}{\sum_{a\in\mathcal{A}_s} y_a(t)}$$

where $\rho(t) = Ay(t)$.

[23]:
```
print('Policy stored in tensor of dim 10x6x4...')
Pi = np.zeros((T,n,4));
for t in range(T):
    Pi[t] = np.diag(yopt[t])@A.T@mat.inv(np.diag(A@yopt[t]));


## ALTERNATIVE COMPUTATION ## ALTERNATIVE COMPUTATION
## one liner using np.einsum...
## gives same value as above...
Pi = np.einsum('ij,jk,ik->ijk',yopt,A.T,1./(yopt@A.T))[:-1];
print('easier to display as a vector...')
print('Policy:')
print(np.round(np.sum(Pi,2),2))
```

```
Policy stored in tensor of dim 10x6x4…
easier to display as a vector…
Policy:
[[1. 1. 1. 0. 1. 0.]
 [1. 1. 1. 0. 1. 0.]
 [1. 1. 1. 0. 1. 0.]
 [1. 1. 1. 0. 1. 0.]
 [1. 1. 1. 0. 1. 0.]
 [1. 1. 1. 0. 1. 0.]
 [1. 1. 1. 0. 1. 0.]
 [1. 1. 1. 0. 1. 0.]
 [1. 1. 1. 0. 1. 0.]
 [1. 1. 1. 0. 1. 0.]]
```

(PTS:0-4) Now suppose you apply the policy

$$\Pi(t) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.2 & 0 \\ 0 & 0 & 0.8 & 0 \\ 0 & 0 & 0 & 0.2 \\ 0 & 0 & 0 & 0.8 \end{bmatrix}$$

at each time step. Start by computing $y(0) = \Pi(0)\rho(0)$. $\rho(t)$ is then given by $Py(0) = \rho(1)$. Use $\rho(1)$ to compute $y(1) = \Pi(1)\rho(1)$, etc. What total reward do you achieve? What is the quantity $\sum_t \mu(t)^T y(t)$? How does this relate the total reward to the optimal total reward?

[24]:
```
Pi3 = np.array([[1.,0.,0.,0.],
                [0.,1.,0.,0.],
                [0.,0.,0.2,0.],
                [0.,0.,0.8,0.],
                [0.,0.,0.,0.2],
                [0.,0.,0.,0.8]])

yt      = np.zeros([T+1,n]);
rhot    = np.zeros([T+1,4]);
total_reward = 0.;
total_mu = 0.;
rhot[0] = rho0;
for t in range(T):
    yt[t] = Pi3@rhot[t];
    rhot[t+1] = P@yt[t];
    total_reward = total_reward + r@yt[t];
    total_mu = total_mu + muopt[t]@yt[t];
yt[T] = Pi3@rhot[T];

total_mu = total_mu + muopt[T]@yt[T];
total_reward = total_reward + g@rhot[T];
print('y: ')
print(np.round(yt,2))
print('')
print('Total reward: ')
print(np.round(total_reward,2))
print('Opt. total reward - total reward: ')
print(np.round(primal.value - total_reward,2))
print('Inefficiency sum_t mu(t).T y(t): ')
print(np.round(total_mu,2))
```

```
y:
[[0.25 0.25 0.05 0.2  0.05 0.2 ]
 [0.4  0.1  0.07 0.28 0.03 0.12]
 [0.19 0.14 0.09 0.37 0.04 0.17]
 [0.27 0.18 0.05 0.22 0.06 0.22]
```

```
 [0.35 0.11 0.08 0.3  0.03 0.13]
 [0.21 0.15 0.08 0.33 0.05 0.18]
 [0.29 0.17 0.06 0.24 0.05 0.2 ]
 [0.32 0.12 0.08 0.31 0.04 0.14]
 [0.23 0.15 0.08 0.31 0.05 0.19]
 [0.29 0.15 0.06 0.26 0.05 0.19]
 [0.29 0.13 0.08 0.31 0.04 0.15]]

Total reward:
14.92
Opt. total reward - total reward:
5.83
Inefficiency sum_t mu(t).T y(t):
5.83
```

[ ]:

```
```