HOME WORK 2 - CS 687

DANIEL SAM PETE THIYAGU

Part 1

Question 1. Answer:
I wrote a program for the below : https://github.com/danielsamfdo/RL_CS687/blob/master/Value%20Iteration.ipynb

It is also present as a code snippet below the answer.

V1:

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	8.0
0.0	0.0	-2.0	8.0	0.0

V2:

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	6.4
0.0	0.0	0.0	6.8	9.2
0.0	0.0	4.0	9.2	0.0

V3:

0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	5.12
0.0	0.0	0.0	5.76	8.32
0.0	0.0	0.0	8.84	9.72
0.0	0.0	6.16	9.72	0.0

V4:

0.0	0.0	0.0	0.0	4.096
0.0	0.0	0.0	4.864	7.424
0.0	0.0	0.0	8.352	9.312
0.0	0.0	0.0	9.588	9.9
0.0	0.0	7.008	9.9	0.0

V5:

0.0	0.0	0.0	4.096	6.554
0.0	0.0	3.891	7.539	8.806
0.0	0.0	0.0	9.389	9.734
0.0	0.0	0.0	9.853	9.964
0.0	0.0	7.322	9.964	0.0

V6:

```
0.0    0.0    3.471  6.768  8.233
0.0    3.113  6.615  8.9    9.485
0.0    0.0    0.0    9.777  9.901
0.0    0.0    0.0    9.947  9.987
0.0    0.0    7.436  9.987  0.0
```

V7:

```
0.0    2.932  6.266  8.382  9.161
2.49   5.603  8.286  9.517  9.789
0.0    2.49   0.0    9.919  9.964
0.0    0.0    0.0    9.981  9.995
0.0    0.0    7.477  9.995  0.0
```

V8:

```
2.47   5.733  8.06   9.223  9.624
4.731  7.46   9.17   9.791  9.915
2.117  4.856  0.0    9.971  9.987
0.0    1.992  0.0    9.993  9.998
0.0    0.0    7.491  9.998  0.0
```

V9:

```
5.193  7.681  9.046  9.639  9.837
6.67   8.611  9.611  9.91   9.966
4.345  6.802  0.0    9.989  9.995
1.793  4.184  0.0    9.997  9.999
0.0    1.468  7.497  9.999  0.0
```

V10:

```
7.257  8.82   9.549  9.836  9.93
8.033  9.274  9.822  9.961  9.986
6.328  8.126  0.0    9.996  9.998
3.954  6.159  0.0    9.999  10.0
1.508  3.369  7.499  10.0   0.0
```

Code Snippet:

```
def left(s_i, s_j):
    if(s_j==0):
        i,j =  (s_i, s_j)
    else:
        i,j =  (s_i, s_j-1)
    if((i==2 or i==3 ) and j==2):
        return s_i,s_j
    return i,j

def up(s_i, s_j):

    if(s_i==0):
        i,j = (s_i, s_j)
    else:
        i,j =  (s_i-1,s_j)
    if((i==2 or i==3 ) and j==2):
        return s_i,s_j
    return i,j

def reward(ns_i,ns_j):
    if(ns_i == 4 and ns_j == 2):
        r = -10
    elif(ns_i == 4 and ns_j == 4):
        r = 10
    else:
        r = 0
    return r

def down(s_i,s_j):
    if(s_i==4):
        i,j =   (s_i,s_j)
    else:
        i,j =   (s_i+1,s_j)
    if((i==2 or i==3 ) and j==2):
        return s_i,s_j
    return i,j


def right(s_i,s_j):
    if(s_j==4):
        i,j =   (s_i,s_j)
    else:
        i,j =   (s_i,s_j+1)
    if((i==2 or i==3 ) and j==2):
        return s_i,s_j
    return i,j


def find_max_action(s,a,prev_V,print_debug=False):
    s_i,s_j = s

    for action in a:
        sum_v = 0
        if(action == 0):#UP

            ns_i,ns_j = up(s_i,s_j)
            ls_i,ls_j = left(s_i,s_j)
            rs_i,rs_j = right(s_i,s_j)
            r = reward(ns_i,ns_j)
            sr = reward(s_i,s_j)
            lr = reward(ls_i,ls_j)
            rr = reward(rs_i,rs_j)
            sum_v += 0.8 * (r + prev_V[ns_i][ns_j])
            sum_v += 0.1 * (sr + prev_V[s_i][s_j])
            sum_v += 0.05 * (lr + prev_V[ls_i][ls_j])
            sum_v += 0.05 * (rr + prev_V[rs_i][rs_j])

            max_action = sum_v;
            if(print_debug):
                print "CAME IN TAKING UP ACTION"
                print r,sr,lr,rr
                print 0.8 * (r + prev_V[ns_i][ns_j]) , 0.1 * (sr + prev_V[s_i][s_j]), 0.05

                print sum_v
        elif(action == 1):#LEFT

            ns_i,ns_j = left(s_i,s_j)
            ls_i,ls_j = up(s_i,s_j)
            rs_i,rs_j = down(s_i,s_j)
            r = reward(ns_i,ns_j)
            sr = reward(s_i,s_j)
            lr = reward(ls_i,ls_j)
            rr = reward(rs_i,rs_j)

            sum_v += 0.8 * (r + prev_V[ns_i][ns_j])
            sum_v += 0.1 * (sr + prev_V[s_i][s_j])
            sum_v += 0.05 * (lr + prev_V[ls_i][ls_j])
            sum_v += 0.05 * (rr + prev_V[rs_i][rs_j])
            max_action = max(sum_v,max_action);
            if(print_debug):
                print "CAME IN TAKING LEFT ACTION"
                print r,sr,lr,rr
                print 0.8 * (r + prev_V[ns_i][ns_j]) , 0.1 * (sr + prev_V[s_i][s_j]), 0.05

                print sum_v
        elif(action == 2):#DOWN

            ns_i,ns_j = down(s_i,s_j)
            ls_i,ls_j = left(s_i,s_j)
            rs_i,rs_j = right(s_i,s_j)
            r = reward(ns_i,ns_j)
            sr = reward(s_i,s_j)
            lr = reward(ls_i,ls_j)
            rr = reward(rs_i,rs_j)
            sum_v += (0.8 * (r + prev_V[ns_i][ns_j]) )
            sum_v += (0.1 * (sr + prev_V[s_i][s_j]))
            sum_v += (0.05 * (lr + prev_V[ls_i][ls_j]))
            sum_v += (0.05 * (rr + prev_V[rs_i][rs_j])  )
            max_action = max(sum_v,max_action);
            if(print_debug):
                print "CAME IN TAKING DOWN ACTION"
                print r,sr,lr,rr
                print 0.8 * (r + prev_V[ns_i][ns_j]) , 0.1 * (sr + prev_V[s_i][s_j]), 0.05
```

```
                print sum_v
        elif(action == 3):#RIGHT

            ns_i,ns_j = right(s_i,s_j)
            ls_i,ls_j = up(s_i,s_j)
            rs_i,rs_j = down(s_i,s_j)
            r = reward(ns_i,ns_j)
            sr = reward(s_i,s_j)
            lr = reward(ls_i,ls_j)
            rr = reward(rs_i,rs_j)
            sum_v += 0.8 * (r + prev_V[ns_i][ns_j])
            sum_v += 0.1 * (sr + prev_V[s_i][s_j])
            sum_v += 0.05 * (lr + prev_V[ls_i][ls_j])
            sum_v += 0.05 * (rr + prev_V[rs_i][rs_j])
            max_action = max(sum_v,max_action);
            if(print_debug):
                print "CAME IN TAKING RIGHT ACTION"
                print r,sr,lr,rr
                print 0.8 * (r + prev_V[ns_i][ns_j]) , 0.1 * (sr + prev_V[s_i][s_j]), 0.05
                print sum_v
    return max_action



def Value_Iteration(prev_V,deb=False):
    V = np.zeros(prev_V.shape)
    r,c = V.shape
    for i in range(r):
        for j in range(c):
            if((i==2 and j==2) or (i==3 and j==2) or (i==4 and j==4)):
                continue;
            else:
                if(deb):
                    print "STATE IS " + str(i)+" , "+ str(j)
                s = (i,j)
                a = [0,1,2,3]
                V[i][j] = find_max_action(s,a,prev_V,deb)
    return V
```

Question 2. Answer:

We know that for an optimal policy, π^*, it has to satisfy $\forall \pi^i \in \pi V^{\pi^*}(s) \geq V^{\pi^i}(s)$

We know the definition of

$$V^{\pi}(s) = E[\sum_{k=0}^{\inf} \gamma^k R_{t+k} | S_t = s, \pi]$$

$$V^{\pi}(s) = \sum_a \pi(s,a) \sum_{s'} P(s,a,s^{'})(R(s,a,s^{'}) + \gamma * V^{\pi}(s^{'})))$$

$$V^{\pi}(s) = \sum_a \pi(s,a) \sum_{s'} P(s,a,s^{'}) R(s,a,s^{'}) + \sum_a \pi(s,a) \sum_{s'} P(s,a,s^{'}) * \gamma * V^{\pi}(s^{'}))$$

Similarly unrolling,

$$V^{\pi}(s) = \sum_a \pi(s,a) \sum_{s'} P(s,a,s^{'}) R(s,a,s^{'}) + \gamma \sum_a \pi(s,a) \sum_{s'} P(s,a,s^{'}) \sum_{a'} \pi(s^{'},a^{'}) \sum_{s''} P(s^{'},a,s^{''})(R(s^{'},a,s^{''}) + \gamma * V^{\pi}(s^{''})))$$

$$V^{\pi}(s) = \sum_a \pi(s,a) \sum_{s'} P(s,a,s^{'}) R(s,a,s^{'}) + \gamma \sum_a \pi(s,a) \sum_{s'} P(s,a,s^{'}) \sum_{a'} \pi(s^{'},a^{'}) \sum_{s''} P(s^{'},a,s^{''}) R(s^{'},a,s^{''}) +$$
$$\gamma^2 \sum_a \pi(s,a) \sum_{s'} P(s,a,s^{'}) \sum_{a'} \pi(s^{'},a^{'}) \sum_{s''} P(s^{'},a,s^{''}) \sum_{a''} \pi(s^{''},a^{''}) \sum_{s'''} P(s^{''},a,s^{'''}) R(s^{''},a,s^{'''}) + ...$$

Now if we were to multiply all the rewards of any Policy by a positive constant c, we know that $E(c{*}R_t | S_t = s, A_t = a, S_{t+1} = s') = c * R(s,a,s')$

$$V^{\pi}(s) = c \sum_a \pi(s,a) \sum_{s'} P(s,a,s^{'}) R(s,a,s^{'}) + c\gamma \sum_a \pi(s,a) \sum_{s'} P(s,a,s^{'}) \sum_{a'} \pi(s^{'},a^{'}) \sum_{s''} P(s^{'},a,s^{''}) R(s^{'},a,s^{''}) +$$
$$c\gamma^2 \sum_a \pi(s,a) \sum_{s'} P(s,a,s^{'}) \sum_{a'} \pi(s^{'},a^{'}) \sum_{s''} P(s^{'},a,s^{''}) \sum_{a''} \pi(s^{''},a^{''}) \sum_{s'''} P(s^{''},a,s^{'''}) R(s^{''},a,s^{'''}) + ...$$

$$V^{\pi}(s) = cV^{\pi^{old}}(s)$$

The new Policy value statue function for each state would be c times the values for the previous/old policy.

It therefore follows that if we had found an optimal policy , if we multiply Reward with a positive constant c, The optimal policy doesn't change for the new environment with the different rewards function.

Since it satisfies $\forall \pi^i \in \pi,$ we see that $\text{c} * V^{\pi^*}(s) \geq cV^{\pi^i}(s)$

Question 3. Answer:
In the above question, we already derived,

$$V^{\pi}(s) = \sum_a \pi(s,a) \sum_{s'} P(s,a,s^{'})R(s,a,s^{'}) + \gamma \sum_a \pi(s,a) \sum_{s'} P(s,a,s^{'}) \sum_{a'} \pi(s^{'},a^{'}) \sum_{s''} P(s^{'},a,s^{''})R(s^{'},a,s^{''}) +$$
$$\gamma^2 \sum_a \pi(s,a) \sum_{s'} P(s,a,s^{'}) \sum_{a'} \pi(s^{'},a^{'}) \sum_{s''} P(s^{'},a,s^{''}) \sum_{a''} \pi(s^{''},a^{''}) \sum_{s'''} P(s^{''},a,s^{'''})R(s^{''},a,s^{'''}) + ...$$

Let us call the above derivation, V_{old}.

Now if we were to add a constant "c" to the reward, we know that $E(c + R_t|S_t = s, A_t = a, S_{t+1} = s') = E(c|S_t = s, A_t = a, S_{t+1} = s') + R(s,a,s') = c + R(s,a,s')$

$$V^{\pi}(s) = \sum_a \pi(s,a) \sum_{s'} P(s,a,s^{'})(R(s,a,s^{'})+c) + \gamma \sum_a \pi(s,a) \sum_{s'} P(s,a,s^{'}) \sum_{a'} \pi(s^{'},a^{'}) \sum_{s''} P(s^{'},a,s^{''})(R(s^{'},a,s^{''})+c) +$$
$$\gamma^2 \sum_a \pi(s,a) \sum_{s'} P(s,a,s^{'}) \sum_{a'} \pi(s^{'},a^{'}) \sum_{s''} P(s^{'},a,s^{''}) \sum_{a''} \pi(s^{''},a^{''}) \sum_{s'''} P(s^{''},a,s^{'''})(R(s^{''},a,s^{'''}) + c) + ...$$

$$V^{\pi}(s) = \sum_a \pi(s,a) \sum_{s'} P(s,a,s^{'})R(s,a,s^{'}) + \gamma \sum_a \pi(s,a) \sum_{s'} P(s,a,s^{'}) \sum_{a'} \pi(s^{'},a^{'}) \sum_{s''} P(s^{'},a,s^{''})R(s^{'},a,s^{''}) +$$
$$\gamma^2 \sum_a \pi(s,a) \sum_{s'} P(s,a,s^{'}) \sum_{a'} \pi(s^{'},a^{'}) \sum_{s''} P(s^{'},a,s^{''}) \sum_{a''} \pi(s^{''},a^{''}) \sum_{s'''} P(s^{''},a,s^{'''})R(s^{''},a,s^{'''}) + ...$$
$$+ \sum_a \pi(s,a) \sum_{s'} P(s,a,s^{'})c + \gamma \sum_a \pi(s,a) \sum_{s'} P(s,a,s^{'}) \sum_{a'} \pi(s^{'},a^{'}) \sum_{s''} P(s^{'},a,s^{''})c +$$
$$\gamma^2 \sum_a \pi(s,a) \sum_{s'} P(s,a,s^{'}) \sum_{a'} \pi(s^{'},a^{'}) \sum_{s''} P(s^{'},a,s^{''}) \sum_{a''} \pi(s^{''},a^{''}) \sum_{s'''} P(s^{''},a,s^{'''})c + ...$$

$$V^{\pi}(s) = V_{old} + \sum_a \pi(s,a) \sum_{s'} P(s,a,s^{'})c + \gamma \sum_a \pi(s,a) \sum_{s'} P(s,a,s^{'}) \sum_{a'} \pi(s^{'},a^{'}) \sum_{s''} P(s^{'},a,s^{''})c +$$
$$\gamma^2 \sum_a \pi(s,a) \sum_{s'} P(s,a,s^{'}) \sum_{a'} \pi(s^{'},a^{'}) \sum_{s''} P(s^{'},a,s^{''}) \sum_{a''} \pi(s^{''},a^{''}) \sum_{s'''} P(s^{''},a,s^{'''})c + ...$$

We can never guarantee that the optimal policy will remain the same since it depends on the dynamics of the environment as seen in the above equation. It therefore follows that if we had found an optimal policy , if we add Reward with a positive constant c, The optimal policy can change for the new environment with the different rewards function(addition of c).

As an example, consider a grid world of 1 * 2 : ie has 2 states, start and terminal state. If the reward at each state was -1. An agents optimal policy would be to end the game, ie to transition into the terminal state from the start state. Now add a positive constant 2 to the reward, Now an agents optimal policy would be to not end the game, ie. to never transition to the terminal state and gather the positive +1 reward.

Question 4. Answer:

$$V^{\pi}(s) = E[\sum_{k=0}^{\text{inf}} \gamma^k R_{t+k} | S_t = s, \pi]$$

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s^{'}} P(s,a,s^{'})(R(s,a,s^{'}) + \gamma * V^{\pi}(s^{'})))....\ (1)$$

$$Q^{\pi}(s,a) = E[\sum_{k=0}^{\text{inf}} \gamma^k R_{t+k} | S_t = s, A_t = a, \pi]$$

$$Q^{\pi}(s,a) = \sum_{s^{'}} P(s,a,s^{'})(R(s,a,s^{'}) + \gamma * E[\sum_{k=0}^{\text{inf}} \gamma^k R_{t+1+k} | S_{t+1} = s^{'}, \pi])$$

$$Q^{\pi}(s,a) = \sum_{s^{'}} P(s,a,s^{'})(R(s,a,s^{'}) + \gamma * V^{\pi}(s^{'}))....\ (2)$$

Using (2) in (1),

$$V^{\pi}(s) = \sum_{a} \pi(s,a) Q^{\pi}(s,a)$$

Question 5. Answer:

$$V^{\pi}(s) = E[\sum_{k=0}^{\inf} \gamma^k R_{t+k} | S_t = s, \pi]$$

Since $\gamma = 1$

$$V^{\pi}(s) = E[\sum_{k=0}^{\inf} R_{t+k} | S_t = s, \pi]$$

$$V^{\pi}(s) = \sum_a \pi(s,a) \sum_{s'} P(s,a,s^{'})(R(s,a,s^{'}) + V^{\pi}(s^{'})))$$

$$V^{\pi}(s) = \sum_a \pi(s,a) \sum_{s'} P(s,a,s^{'}) R(s,a,s^{'}) + \sum_a \pi(s,a) \sum_{s'} P(s,a,s^{'}) \sum_{a'} \pi(s^{'},a^{'}) \sum_{s''} P(s^{'},a,s^{''}) R(s^{'},a,s^{''}) +$$
$$\sum_a \pi(s,a) \sum_{s'} P(s,a,s^{'}) \sum_{a'} \pi(s^{'},a^{'}) \sum_{s''} P(s^{'},a,s^{''}) \sum_{a''} \pi(s^{''},a^{''}) \sum_{s'''} P(s^{''},a,s^{'''}) R(s^{''},a,s^{'''}) + ...$$

We also know it has a deterministic transition function, which means taking an action in a state can only lead to a unique state. ie, $P(s,a,s^{'}) = \{0,1\}$

$$V^{\pi}(s) = \sum_a \pi(s,a) R(s,a,s^{'}) + \sum_a \pi(s,a) \sum_{a'} \pi(s^{'},a^{'}) R(s^{'},a,s^{''}) +$$
$$\sum_a \pi(s,a) \sum_{a'} \pi(s^{'},a^{'}) \sum_{a''} \pi(s^{''},a^{''}) R(s^{''},a,s^{'''}) + ...$$

We also know it has a deterministic policy, so the probability of taking an action given a state is fixed, ie $\pi(s,a) = \{0,1\}$

$$V^{\pi}(s) = R(s,a,s^{'}) + R(s^{'},a,s^{''}) + R(s^{''},a,s^{'''}) + ...$$

We know that R_t is negative, so all expected reward funtions are negative, ie. $R(s,a,s^{'})$ is negative. R_t is reward obtained at timestep t but i am naming the reward function $R(s,a,s^{'})$ as R_t, just for this question.

$$V^{\pi}(s_t) = R_t + R_{t+1} + R_{t+2} + ... + R_{t+k}$$

where k goes on till $L-1$.

Only since we have a deterministic transition and deterministic policy, we can state that the values keep on increasing, as they can transition to only one unique state and also given that in each state, they can only take on one action.

so $R_t + R_{t+1} + R_{t+2} + ... + R_{t+k} < R_{t+1} + R_{t+2} + ... + R_{t+k} < R_{t+2} + ... + R_{t+k}$ and so on.

In our particular case, where we have our history $H = (S_0, A_0, R_0, S_1, A_2, R_1, \ldots, S_{L-1}, A_{L-1}, R_{L-1})$

we can be sure that from the above statement, that $v^{\pi}(S_0), v^{\pi}(S_1), \ldots, v^{\pi}(S_{L-1})$ is strictly increasing.

Since by substituting $t = 0$ in $R_t + R_{t+1} + R_{t+2} + ... + R_{t+k} < R_{t+1} + R_{t+2} + ... + R_{t+k} < R_{t+2} + ... + R_{t+k}$

we can see that $v^{\pi}(S_0) < v^{\pi}(S_1) < ... < v^{\pi}(S_{L-1})$

Question 6. Answer:

Bellman Operator T is a contraction mapping,
$f < -T, X < -R^{|S|}, d(Q, Q^{'}) = max_{s,a}|Q(s,a) - Q'(s,a)|$
To Prove: $||TQ - TQ'|| \leq \gamma ||Q(s,a) - Q'(s,a)||$

$$||TQ - TQ'|| = max_{s,a}|(TQ - TQ')(s,a)|$$
$$||TQ - TQ'|| = max_{s,a}|TQ(s,a) - TQ'(s,a)|$$
$$||TQ-TQ'|| = max_{s,a}|\sum_{s'} P(s,a,s')(R(s,a,s')+\gamma \max_{a'} Q(s',a')) - \sum_{s'} P(s,a,s')(R(s,a,s')+\gamma \max_{a'} Q'(s',a'))|$$
$$||TQ-TQ'|| = max_{s,a}|\sum_{s'} P(s,a,s')(R(s,a,s')+\gamma \max_{a'} Q(s',a')) - (\sum_{s'} P(s,a,s')(R(s,a,s')+\gamma \max_{a'} Q'(s',a')))|$$
$$||TQ - TQ'|| = max_{s,a}|\sum_{s'} P(s,a,s')\gamma \max_{a'} Q(s',a') - (\sum_{s'} P(s,a,s')\gamma \max_{a'} Q'(s',a'))|$$
$$||TQ - TQ'|| = max_{s,a}|\sum_{s'} P(s,a,s')(\gamma \max_{a'} Q(s',a') - \gamma \max_{a'} Q'(s',a'))|$$
$$||TQ - TQ'|| = \gamma max_{s,a}|\sum_{s'} P(s,a,s')(\max_{a'} Q(s',a') - \max_{a'} Q'(s',a'))|$$

Using $|\max_x f(x) - \max_x g(x)| \leq \max_x |f(x) - g(x)|$

$$||TQ - TQ'|| \leq \gamma \max_{s,a} |\sum_{s'} P(s,a,s')(\max_{a'} |Q(s',a') - Q'(s',a'))||$$

Using $\sum_x p(x) * |f(x)| \leq \max_x |f(x)|$

$$||TQ - TQ'|| \leq \gamma \max_{s,a} \max_{s'} | \max_{a'} |Q(s',a') - Q'(s',a')||$$
$$||TQ - TQ'|| = \gamma \max_{s,a} \max_{s',a'} |Q(s',a') - Q'(s',a')|$$

It doesn't depend on $max_{s,a}$ as s,a are not in the equation.

$$||TQ - TQ'|| = \gamma \max_{s',a'} |Q(s',a') - Q'(s',a')|$$
$$||TQ - TQ'|| = \gamma |Q - Q'|$$

Hence we have proved
$$||TQ - TQ'|| \leq \gamma |Q - Q'|$$

Question 7. Answer:

To Prove:

$$\lim_{n\to\infty} \Pr\left(|\hat{J}_n(\pi, D_n) - J(\pi)| > \epsilon\right) = 0$$

Given:

$$\hat{J}(\pi, H) = \sum_{t=0}^{\infty} \gamma^t \left(R_t - R(S_t, A_t, S_{t+1})\right) + \sum_{t=0}^{\infty} \gamma^t \sum_{s'} P(S_t, A_t, s')R(S_t, A_t, s')$$

$$\hat{J}_n(\pi, D_n) = \frac{1}{n}\sum_{i=1}^{n} \hat{J}(\pi, H_i)$$

$D_n = (H_1, \ldots, H_n)$, each produced by running the policy π

Proof:

In reference to Theorem 8.2 in https://www.dartmouth.edu/~chance/teaching_aids/books_articles/probability_book/Chapter8.pdf Let X1, X2,..., Xn be an independent trials process, with finite expected value $\mu = E(Xj)$ and finite variance $\sigma = V(Xj)$. Let $Sn = X1 + X2 + ... + Xn.$ Then for any $\epsilon > 0$

$$\lim_{n\to\infty} \Pr\left(|\frac{S_n}{n} - \mu| > \epsilon\right) = 0$$

.

Let us now take our equation to prove:

$$\hat{J}_n(\pi, D_n) = \frac{1}{n}\sum_{i=1}^{n}\sum_{t=0}^{\infty} \gamma^t \left(R_t - R(S_t, A_t, S_{t+1})\right) + \sum_{t=0}^{\infty} \gamma^t \sum_{s'} P(S_t, A_t, s')R(S_t, A_t, s')$$

Let us take the first part of the equation, Now i would like to prove

$$\frac{1}{n}\sum_{i=1}^{n}\sum_{t=0}^{\infty} \gamma^t \left(R_t - R(S_t, A_t, S_{t+1})\right) = 0 \text{as n tends to inf}$$

For a given timestep t,

$$\frac{1}{n}\gamma^t[(R_t^{H1} + R_t^{H2} + .. + R_t^{Hn}) - (R(S_t, A_t, S_{t+1})^{H1} + R(S_t, A_t, S_{t+1})^{H1} + ... + R(S_t, A_t, S_{t+1})^{Hn})] ...... \quad (7.01)$$

Let us consider a subset of the histories where the State, Action and the Resulting Transition State were (s,a,s')

Let us say the subset was $D_k = \{H_{k1}, ..., H_{kk}\}$

Now we are only considering a timestep $t = t$

$$\frac{1}{k}\gamma^t[(R_t^{H_{k1}} + R_t^{H_{k2}} + .. + R_t^{H_{kk}}) - (R(s, a, s')^{H_{k1}} + R(s, a, s')^{H_{k2}} + ... + R(s, a, s')^{H_{kk}})] = 0 \text{ as k tends to inf}$$

$$\frac{1}{k}\gamma^t[(R_t^{H_{k1}} + R_t^{H_{k2}} + .. + R_t^{H_{kk}}) - (R(s, a, s')^{H_{k1}} + R(s, a, s')^{H_{k2}} + ... + R(s, a, s')^{H_{kk}})]$$

$$\frac{1}{k}\gamma^t[(R_t^{H_{k1}} + R_t^{H_{k2}} + .. + R_t^{H_{kk}}) - (R(s, a, s') * k)]$$

$$\gamma^t[(\frac{R_t^{H_{k1}} + R_t^{H_{k2}} + .. + R_t^{H_{kk}}}{k}) - (R(s, a, s'))]$$

The above follows the specified theorem of Law of Large Numbers, So as n tends to inf, k also tends to inf, and $Sk = R_t^{H_{k1}} + R_t^{H_{k2}} + .. + R_t^{H_{kk}}$, $\mu = R(s, a, s')$

So we are proved it for state,action,resulting state triplet (s,a,s'), we can use the same strategy for all these triplets, and so they equate to 0. Since they all simultaneously converge. We proved it for a given time step, we can extend to say that this follows for all time steps $t = 0, 1, ..., \text{inf}$.

So now our problems boils down to solving,

$$\lim_{n\to\infty} \Pr\left(|(\frac{1}{n}\sum_{i=1}^{n}\sum_{t=0}^{\infty}\gamma^t \sum_{s'} P(S_t, A_t, s')R(S_t, A_t, s')) - J(\pi)| > \epsilon \right) = 0....(7.02)$$

Similar to our method above,

Let us take at time t=t, the ones were state,action were (s,a) Let us say the subset of Histories were state,action were (s,a) was $D_k = \{H_{k1}, ..., H_{kk}\}$

We know that if we expanded $J(\pi)$ as in Question 2 or 3 and took the term where the time step was t it would be the following

$$J(\pi)\text{term at }\gamma^t = \gamma^t \sum_{s0} d_0(s_0) * \sum_{a_0}\pi(s_0,a_0)\sum_{s_1}P(s_0,a_0,s_1)\sum_{a_1}\pi(s_1,a_1)\sum_{s2}P(s_1,a_1,s_2)....$$

$$\sum_{a_{t-1}}\pi(s_{t-1},a_{t-1})\sum_{s}P(s_{t-1},a_{t-1},s)\sum_{a}\pi(s,a)\sum_{s'}P(s,a,s')R(s,a,s')$$

NOw we can split it up as

$$J(\pi)\text{term at }\gamma^t = \gamma^t \sum_{s0} d_0(s_0) * \sum_{a_0}\pi(s_0,a_0)\sum_{s_1}P(s_0,a_0,s_1)\sum_{a_1}\pi(s_1,a_1)\sum_{s2}P(s_1,a_1,s_2)....$$

$$\sum_{a_{t-1}}\pi(s_{t-1},a_{t-1})P(s_{t-1},a_{t-1},s)\pi(s,a)\sum_{s'}P(s,a,s')R(s,a,s')+$$

$$\sum_{s0} d_0(s_0) * \sum_{a_0}\pi(s_0,a_0)\sum_{s_1}P(s_0,a_0,s_1)\sum_{a_1}\pi(s_1,a_1)\sum_{s2}P(s_1,a_1,s_2)....$$

$$\sum_{a_{t-1}}\pi(s_{t-1},a_{t-1})P(s_{t-1},a_{t-1},s*)\pi(s*,a*)\sum_{s'}P(s*,a*,s')R(s*,a*,s')+$$

$$\sum_{s0} d_0(s_0) * \sum_{a_0}\pi(s_0,a_0)\sum_{s_1}P(s_0,a_0,s_1)\sum_{a_1}\pi(s_1,a_1)\sum_{s2}P(s_1,a_1,s_2)....$$

$$\sum_{a_{t-1}}\pi(s_{t-1},a_{t-1})P(s_{t-1},a_{t-1},s**)\pi(s**,a**)\sum_{s'}P(s**,a**,s')R(s,a,s') + \text{...and so on ......(7.03)}$$

I am going to prove the below for a state action pair(s,a) tends to zero, at a fixed timestep $t = t$ as k tends to inf

$$(\frac{1}{k}\sum_{i=1}^{k}\gamma^t\sum_{s'}P(S_t=s,A_t=a,s')^{H_{ki}}R(S_t=s,A_t=a,s')^{H_{ki}}) - \gamma^t\sum_{s0}d_0(s_0)*\sum_{a_0}\pi(s_0,a_0)\sum_{s_1}P(s_0,a_0,s_1)\sum_{a_1}\pi(s_1,a_1)$$

$$\sum_{s2}P(s_1,a_1,s_2)....\sum_{a_{t-1}}\pi(s_{t-1},a_{t-1})P(s_{t-1},a_{t-1},s)\pi(s,a)\sum_{s'}P(s,a,s')R(s,a,s')$$

Let us now carefully look at the below, i term it as Prob(S,A) which is probability of landing in state S and taking action A at a timestep t, Also ExpectedReward(s,a) which is expected reward of being in state S and taking action A at a timestep t.

$$Prob(s,a) = \sum_{s0}d_0(s_0)*\sum_{a_0}\pi(s_0,a_0)\sum_{s_1}P(s_0,a_0,s_1)\sum_{a_1}\pi(s_1,a_1)\sum_{s2}P(s_1,a_1,s_2)....\sum_{a_{t-1}}\pi(s_{t-1},a_{t-1})P(s_{t-1},a_{t-1},s)\pi(s,a)$$

$$ExpectedReward(s,a) = Prob(s,a)\sum_{s'}P(s,a,s')R(s,a,s')$$

Prob(s,a) is similar to the probability of landing at state s at timestep t and taking action a from any start state.

$$Sk = \sum_{i=1}^{k}\sum_{s'}P(S_t=s,A_t=a,s')^{H_{ki}}R(S_t=s,A_t=a,s')^{H_{ki}}$$

$\frac{Sk}{k}$ is similar to the average rewards obtained taking action a from state s at timestep t.

$$(\gamma^t(\frac{Sk}{k} - Prob(s,a)\sum_{s'} P(s,a,s')R(s,a,s'))$$

$$(\gamma^t(\frac{Sk}{k} - ExpectedReward(s,a))$$

Similarly, as $Sk = \sum_{i=1}^{k}\sum_{s'} P(S_t = s, A_t = a, s')^{H_{ki}} R(S_t = s, A_t = a, s')^{H_{ki}}$ and $\mu = ExpectedReward(s,a)$ as n tends to inf, k tends to inf, using law of large numbers they tend to zero.

$$\lim_{k\to\infty}(\gamma^t(\frac{Sk}{k} - \mu) > \epsilon) = 0....(7.04)$$

We have proved the above for a given s,a pair at timestep t. We can extend it to all s,a pairs at time step t. Therefore using (7.03), (7.04) and extending it to all pairs(s,a) we can prove it for time step t. Similarly we can extend it to all time steps and prove (7.02) similarly. They all converge simultaneously. Hence we have proved (7.02).