Assignment #2

Course: *Reinforcement Learning (CS6700)*

Instructor: *Prashanth L.A.*
Due date: *April 20th, 2021*

Question 1. Consider a problem of a taxi driver, who serves three cities A, B and C. The taxi driver can find a new ride by choosing one of the following actions.

1. Cruise the streets looking for a passenger.
2. Go to the nearest taxi stand and wait in line.
3. Wait for a call from the dispatcher (this is not possible in town B because of poor reception).

For a given town and a given action, there is a probability that the next trip will go to each of the towns A, B and C and a corresponding reward in monetary units associated with each such trip. This reward represents the income from the trip after all necessary expenses have been deducted. Please refer Table 1 below for the rewards and transition probabilities. In Table 1 below, p_{ij}^k is the probability of getting a ride to town j, by choosing an action k while the driver was in town i and r_{ij}^k is the immediate reward of getting a ride to town j, by choosing an action k while the driver was in town i.

Town i	**Actions** k	**Probabilities** p_{ij}^k j = A B C	**Rewards** r_{ij}^k j = A B C
A	1 2 3	$\begin{array}{ccc} A & B & C \\ \end{array}$ $\begin{bmatrix} 1/2 & 1/4 & 1/4 \\ 1/16 & 3/4 & 3/16 \\ 1/4 & 1/8 & 5/8 \end{bmatrix}$	$\begin{array}{ccc} A & B & C \\ \end{array}$ $\begin{bmatrix} 10 & 4 & 8 \\ 8 & 2 & 4 \\ 4 & 6 & 4 \end{bmatrix}$
B	1 2	$\begin{array}{ccc} A & B & C \\ \end{array}$ $\begin{bmatrix} 1/2 & 0 & 1/2 \\ 1/16 & 7/8 & 1/16 \end{bmatrix}$	$\begin{array}{ccc} A & B & C \\ \end{array}$ $\begin{bmatrix} 14 & 0 & 18 \\ 8 & 16 & 8 \end{bmatrix}$
C	1 2 3	$\begin{array}{ccc} A & B & C \\ \end{array}$ $\begin{bmatrix} 1/4 & 1/4 & 1/2 \\ 1/8 & 3/4 & 1/8 \\ 3/4 & 1/16 & 3/16 \end{bmatrix}$	$\begin{array}{ccc} A & B & C \\ \end{array}$ $\begin{bmatrix} 10 & 2 & 8 \\ 6 & 4 & 2 \\ 4 & 0 & 8 \end{bmatrix}$

Table 1: Taxi Problem: Probabilities and Rewards

Suppose $1 - \gamma$ is the probability that the taxi will breakdown before the next trip. The driver's goal is to maximize the total reward untill his taxi breakdown.

Implement the following. $(1.5 + 1 + 1.5 + 1.5 + 1.5$ marks)

1.1: Find an optimal policy using **policy iteration**(Algorithm 3) starting with a policy that will always cruise independent of the town, and a zero value vector. Let $\gamma = 0.9$.

1.2: Run **policy iteration** for discount factors γ ranging from 0 to 0.95 with intervals of 0.05 and display the results.

1.3: Find an optimal policy using **modified policy iteration**(Algorithm 4) starting with a policy that will always cruise independent of the town, and a zero value vector. Let $\gamma = 0.9$ and $m = 5$.

1.4: Find optimal values using **value iteration**(Algorithm 1) starting with a zero vector. Let $\gamma = 0.9$.

1.5: Find optimal values using **Gauss-Seidel value iteration**(Algorithm 2) starting with a zero vector. Let $\gamma = 0.9$.

Answer the following questions. $(1 + 1 + 1$ marks)

1.a How is different values of γ affecting the **policy iteration** from **1.2**? Explain your findings.

1.b For **modified policy iteration** from **1.3**, do you find any improvement if you choose $m = 10$? Explain your findings.

1.c Compare and contrast the behavior of **value iteration** from **1.4** and **Gauss-Seidel value iteration** from **1.5**.

The pseudocode for the Algorithms are given below. (courtesy: Sutton & Barto 1998)

Algorithm 1 Value Iteration

Initialize: $J(s) = 0, \forall s \in \mathcal{S}$;
repeat
    $\delta = 0$;
    for each $s \in \mathcal{S}$ **do**
        $H(s) = \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P_{ss'}(a)[r(s, a, s') + \gamma J(s')]$
        $\delta = \max(\delta, |J(s) - H(s)|)$;
    end for
    for each $s \in \mathcal{S}$ **do**
        $J(s) = H(s)$;
    end for
until $(\delta < 1\text{e}{-8})$

Algorithm 2 Gauss-Seidel Value Iteration

```
Initialize: J(s) = 0, ∀s ∈ S;
repeat
    δ = 0;
    for each s ∈ S do
        j = J(s);
        J(s) = max_{a∈A} Σ_{s'∈S} P_{ss'}(a)[r(s, a, s') + γJ(s')]
        δ = max(δ, |j − J(s)|);
    end for
until (δ < 1e−8)
```

Algorithm 3 Policy Iteration

```
Input: π_0(s), ∀s ∈ S;
Initialize: J(s) = 0, π(s) = π_0(s), ∀s ∈ S;
repeat
    repeat
        δ = 0;
        for each s ∈ S do
            j = J(s);
            J(s) = Σ_{s'∈S} P_{ss'}(π(s))[r(s, π(s), s') + γJ(s')]
            δ = max(δ, |j − J(s)|);
        end for
    until (δ < 1e−8)
    done = 1;
    for each s ∈ S do
        b = π(s);
        π(s) = argmax_{a∈A} Σ_{s'∈S} P_{ss'}(a)[r(s, a, s') + γJ(s')];
        if b ≠ π(s) then
            done = 0;
        end if
    end for
until done = 1
```

Algorithm 4 Modified Policy Iteration

Input: $\pi_0(s), \forall s \in \mathcal{S}$, m;
Initialize: $J(s) = 0, \pi(s) = \pi_0(s), \forall s \in \mathcal{S}$;
repeat
  for $k = 0, \cdots, m$ **do**
    for each $s \in \mathcal{S}$ **do**
      $J(s) = \sum_{s' \in \mathcal{S}} P_{ss'}(\pi(s))[r(s, \pi(s), s') + \gamma J(s')]$
    end for
  end for
  $done = 1$;
  for each $s \in \mathcal{S}$ **do**
    $b = \pi(s)$;
    $\pi(s) = \text{argmax}_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P_{ss'}(a)[r(s, a, s') + \gamma J(s')]$;
    if $b \neq \pi(s)$ **then**
      $done = 0$;
    end if
  end for
until $done = 1$