Assignment #2

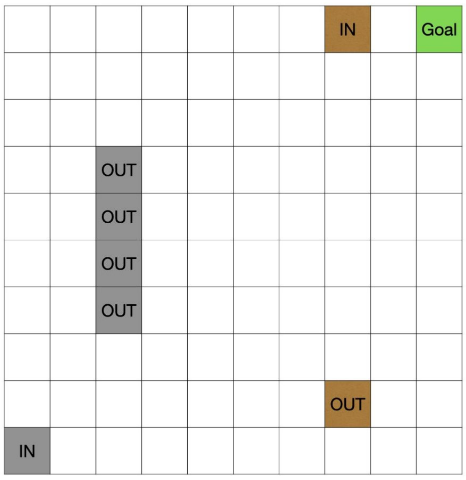
Course: *Reinforcement Learning (CS6700)*

Instructor: *Prashanth L.A.*

Due date: *April 20th, 2021*

Question 2. Consider a 10×10 gridworld (GridWorld-1) as shown in Figure 1.

Figure 1: GridWorld-1



- **State space**: Gridworld has 100 distinct states. There is a special absorbing state called "Goal". There are two wormholes labeled as "IN" in Grey and Brown, any action taken in those states will teleport you to state labeled "OUT" in Grey and Brown respectively. In case of Grey wormhole you can teleport to any one of the states labeled OUT with equal probability (i.e. $1/4$). States labeled OUT is just a normal state.

- **Actions:** In each non-terminal state, you can take 4 actions $\mathcal{A} = \{$North, East, South, West$\}$, which moves you one cell in the corresponding direction.

- **Transition model:** Gridworld is stochastic because the actions can be unreliable. In this model, an action $X \in \{$North, East, South, West$\}$ moves you one cell in the X direction of your current position with probability 0.8, but with probabilities $0.2/3$ it will transition to one of the other neighbouring cells. For example, if the selected action is North, it will transition you one cell to the North of your current position with probability 0.8, one cell to the East, West, or South of your current position with probability $0.2/3$.
  Transitions that take you off the grid will not result in any change. There are no transitions available from the "Goal" state.

- **Rewards:** You will receive a reward of 0 for all transitions except the transition to the "Goal" state. Any transition to the "Goal" state gives you a reward of $+10$.

Implement the following. $(2 + 2 + 2 + 1$ marks)

2.1: Find optimal values using **value iteration** (Algorithm 1)). Start with $J_0(s) = 0$ and $\pi_0(s) =$ North, $\forall s$. Let the discount factor $\gamma = 0.7$.

2.2: Find optimal policy for each state using **policy iteration** (Algorithm 3). Start with $J_0(s) = 0$ and $\pi_0(s) =$ North, $\forall s$. Let the discount factor $\gamma = 0.7$.

2.3: Implement **TD(λ)** (Algorithm 5). Choose the "Grey IN " state as the start state. Let the discount factor $\gamma = 0.7$, and step-size $\alpha = 0.5$. Let the maximum number of episodes $N = 1000$. Consider the policy π as given in Algorithm 6.

2.4: Run TD(λ) for different values of λ ranging from 0 to 1 with intervals of 0.05 and display the results.

Answer the following questions. $(2 + 2 + 2 + 2$ marks)

2.a Compare **value iteration** from **2.1** and **policy iteration** from **2.2** by plotting $J(s)$ vs iterations (the outer loop iterations) for states "Brown IN", "Brown OUT", and "Grey IN". Which algorithm converges faster? Why?

2.b How is different values of λ affecting the TD(λ) from **2.4**? Explain your findings.

Let $\forall s \in \mathcal{S}, J_v(s)$ be the optimal values (final values) obtained using **value iteration** from **2.1**. Then the root mean squared (RMS) error of J with respect to J_v averaged over states can be calulated as

$$error(J) = \sqrt{\frac{\sum_{s\in\mathcal{S}} \left(J(s) - J_v(s)\right)^2}{|\mathcal{S}|}} \tag{1}$$

2.c Plot graph of $error(J^i)$ from **policy iteration** from **2.2** vs iterations i. Here J^i is the values obtained after i^{th} iteration of the outer loop of the algorithm.

2.d Plot graph of $error(J^i)$ from **TD(λ)** from **2.3** vs iterations i for $\lambda \in \{0, 0.25, 0.5, 0.75, 1\}$. Here J^i is the values obtained after i^{th} iteration of the outer loop the algorithm.

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

Algorithm 1 Value Iteration

```
 1: Initialize: $J(s) = 0, \forall s \in \mathcal{S}$;
 2: repeat
 3:     $\delta = 0$;
 4:     for each $s \in \mathcal{S}$ do
 5:         $H(s) = \max_{a\in\mathcal{A}} \sum_{s'\in\mathcal{S}} P_{ss'}(a)[r(s,a,s') + \gamma J(s')]$
 6:         $\delta = \max(\delta, |J(s) - H(s)|)$;
 7:     end for
 8:     for each $s \in \mathcal{S}$ do
 9:         $J(s) = H(s)$;
10:     end for
11: until $(\delta < 1e{-}8)$
```

Algorithm 2 Gauss-Seidel Value Iteration

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

Algorithm 3 Policy Iteration

```
1: **Input**: $\pi_0(s), \forall s \in \mathcal{S}$;
2: **Initialize**: $J(s) = 0, \pi(s) = \pi_0(s), \forall s \in \mathcal{S}$;
3: **repeat**
4:     **repeat**
5:         $\delta = 0$;
6:         **for** each $s \in \mathcal{S}$ **do**
7:             $j = J(s)$;
8:             $J(s) = \sum_{s' \in \mathcal{S}} P_{ss'}(\pi(s))[r(s,\pi(s),s') + \gamma J(s')]$
9:             $\delta = \max(\delta, |j - J(s)|)$;
10:        **end for**
11:    **until** $(\delta < 1\text{e}{-8})$
12:    $done = 1$;
13:    **for** each $s \in \mathcal{S}$ **do**
14:        $b = \pi(s)$;
15:        $\pi(s) = \text{argmax}_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P_{ss'}(a)[r(s,a,s') + \gamma J(s')]$;
16:        **if** $b \neq \pi(s)$ **then**
17:            $done = 0$;
18:        **end if**
19:    **end for**
20: **until** $done = 1$
```

Algorithm 4 Modified Policy Iteration

```
 1: Input: $\pi_0(s), \forall s \in \mathcal{S}$, m;
 2: Initialize: $J(s) = 0, \pi(s) = \pi_0(s), \forall s \in \mathcal{S}$;
 3: repeat
 4:     for $k = 0, \cdots, m$ do
 5:         for each $s \in \mathcal{S}$ do
 6:             $J(s) = \sum_{s' \in \mathcal{S}} P_{ss'}(\pi(s))[r(s, \pi(s), s') + \gamma J(s')]$
 7:         end for
 8:     end for
 9:     $done = 1$;
10:     for each $s \in \mathcal{S}$ do
11:         $b = \pi(s)$;
12:         $\pi(s) = \text{argmax}_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P_{ss'}(a)[r(s, a, s') + \gamma J(s')]$;
13:         if $b \neq \pi(s)$ then
14:             $done = 0$;
15:         end if
16:     end for
17: until $done = 1$
```

Algorithm 5 TD(λ)

```
 1: Input: Start state $s_0$, the policy $\pi(s), \forall s \in \mathcal{S}$, maximum number of episodes $N$, step-size
    $\alpha$, and discount factor $\gamma$;
 2: Initialize: $J(s) = e(s) = 0, \forall s \in \mathcal{S}$;
 3: for $i = 1, \cdots, N$ do
 4:     $s = s_0$;
 5:     repeat {each step of episode}
 6:         $a = \pi(s)$;
 7:         Take action $a$, observe reward $r$, and next state $s'$.
 8:         $\delta = r + \gamma J(s') - J(s)$;
 9:         $e(s) = e(s) + 1$;
10:         for each $s \in \mathcal{S}$ do
11:             $J(s) = J(s) + \alpha \delta e(s)$;
12:             $e(s) = \gamma \lambda e(s)$;
13:         end for
14:         $s = s'$;
15:     until $s$ is terminal
16: end for
```

Algorithm 6 policy for TD(λ)

```
1: Input: Current state position $(s_x, s_y)$, Goal state position $(g_x, g_y)$;
2: Output: The direction of the action $d \in \{(1,0), (-1,0), (0,1), (0,-1)\}$;
3: $\delta_x = g_x - s_x$; $\delta_y = g_y - s_y$;
4: if $|\delta_x| \geq |\delta_y|$ then
5:     $d = (sign(\delta_x), 0)$;
6: else
7:     $d = (0, sign(\delta_y))$;
8: end if
9: return $d$;
```