Artificial Intelligence Laboratory

Task 8

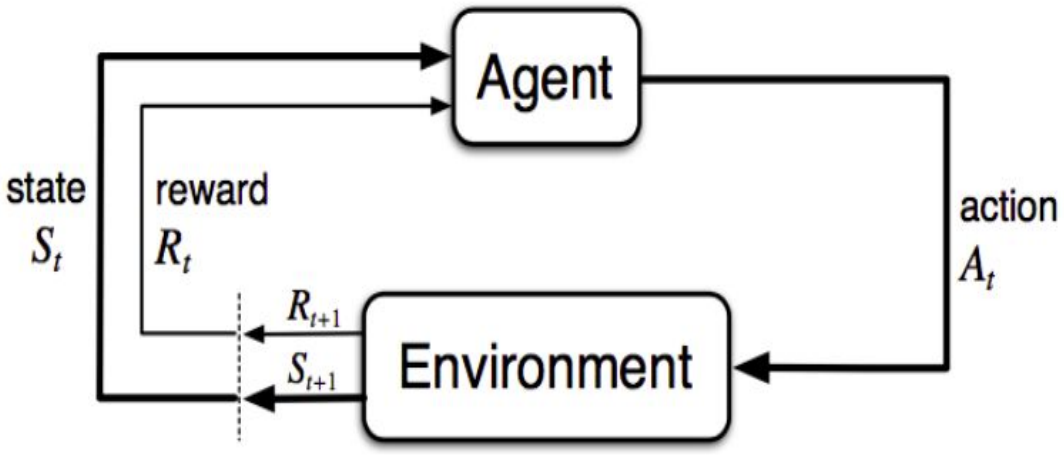
Reinforcement Learning

Kunal Kumar
170010012

---

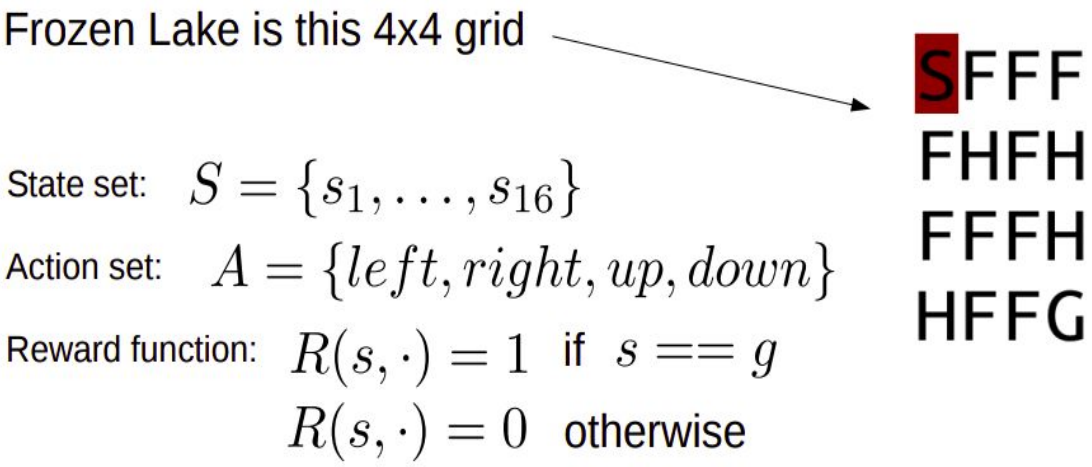
1. MDP Description

In MDP, there is an agent. The agent choose an action a_t at time t and as a consequance the enviorment changes. Here The evniorment is world around the agent. After the action the enviorment state changes to s_{t+1}. A reward might be emitted assciated with what just happened and then this process repeats.



So, there is a feedback cycle in that the next action you take, the next decision you make is in a situation that's the consiquence of what you did before.

2. Problem Description



Transition model: only one third chance of going in specified direction
– one third chance of moving +90deg
– one third change of moving -90deg

Gamma = 0.99

The Frozen Lake environment is a 4×4 grid which contains four possible areas — Safe (S), Frozen (F), Hole (H) and Goal (G). The agent in the environment has four possible moves — Up, Down, Left and Right. This environment will allow the agent to move accordingly. The agent moves around the grid until it reaches the goal or the hole. If it falls into the hole, it has to start from the beginning.The agent receives a reward of 1 if he reaches the goal, and zero otherwise. The process continues until it learns from every mistake and reaches the goal eventually. There could be a random action happening once every few episodes. Let's say the agent is slipping in different directions because it is hard to walk on a frozen surface. Considering this situation, we need to allow some random movement at first, but eventually try to reduce its probability.
Assume the grid has N possible blocks where the agent will be at a given time. At the current state, the agent will have four possibilities of deciding the next state. From the current state, the agent can move in four directions as mentioned which gives us a total of N×4 possibilities. These are our weights and we will update them according to the movement of our agent.

The Agent will start moving from ***Safe*** state and will try to reach ***Goal*** state.

- Actions: $\mathcal{A} = \{0, 1, 2, 3\}$
  a. LEFT: 0
  b. DOWN = 1
  c. RIGHT = 2
  d. UP = 3
- Whole lake is a 4 x 4 grid world, $\mathcal{S} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$

Transition function:

For every state we have State value associated with that state(initially Zero). This value gets updated with time. For each state we have probability P. There is a ⅓ chance of moving in a specified direction(given action). As the ice is slippery, the agent won't always move in the direction he intends.

To move from One state to another state (given action) we have transition function as:

Transition function = probability *(reward + (gamma * (next_state_Value))

Value Iteration

Initialize V arbitrarily, e.g., $V(s) = 0$, for all $s \in \mathcal{S}^+$

Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, π, such that
  $\pi(s) = \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$

Policy Iteration

- Policy evaluation for current policy π_k :
  - Iterate until convergence

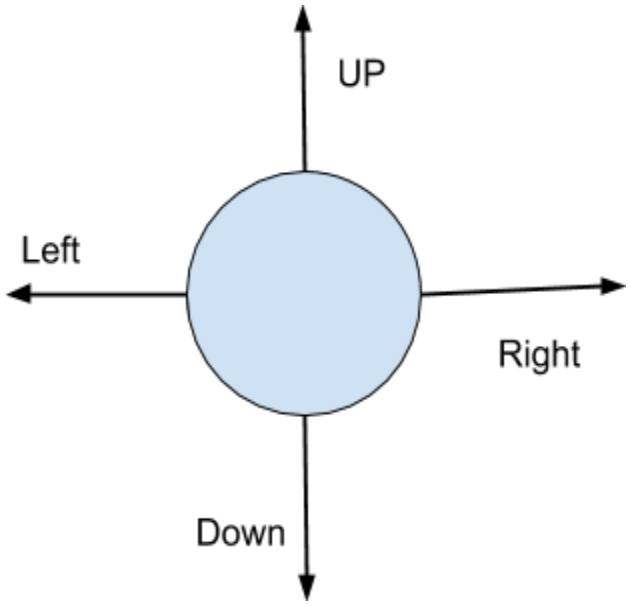
$$V_{i+1}^{\pi_k}(s) \leftarrow \sum_{s'} P(s'|s, \pi_k(s)) \left[R(s, \pi(s), s') + \gamma V_i^{\pi_k}(s')\right]$$

- Policy improvement: find the best action according to one-step look-ahead

$$\pi_{k+1}(s) \leftarrow \arg\max_a \sum_{s'} P(s'|s, a) \left[R(s, a, s') + \gamma V^{\pi_k}(s')\right]$$

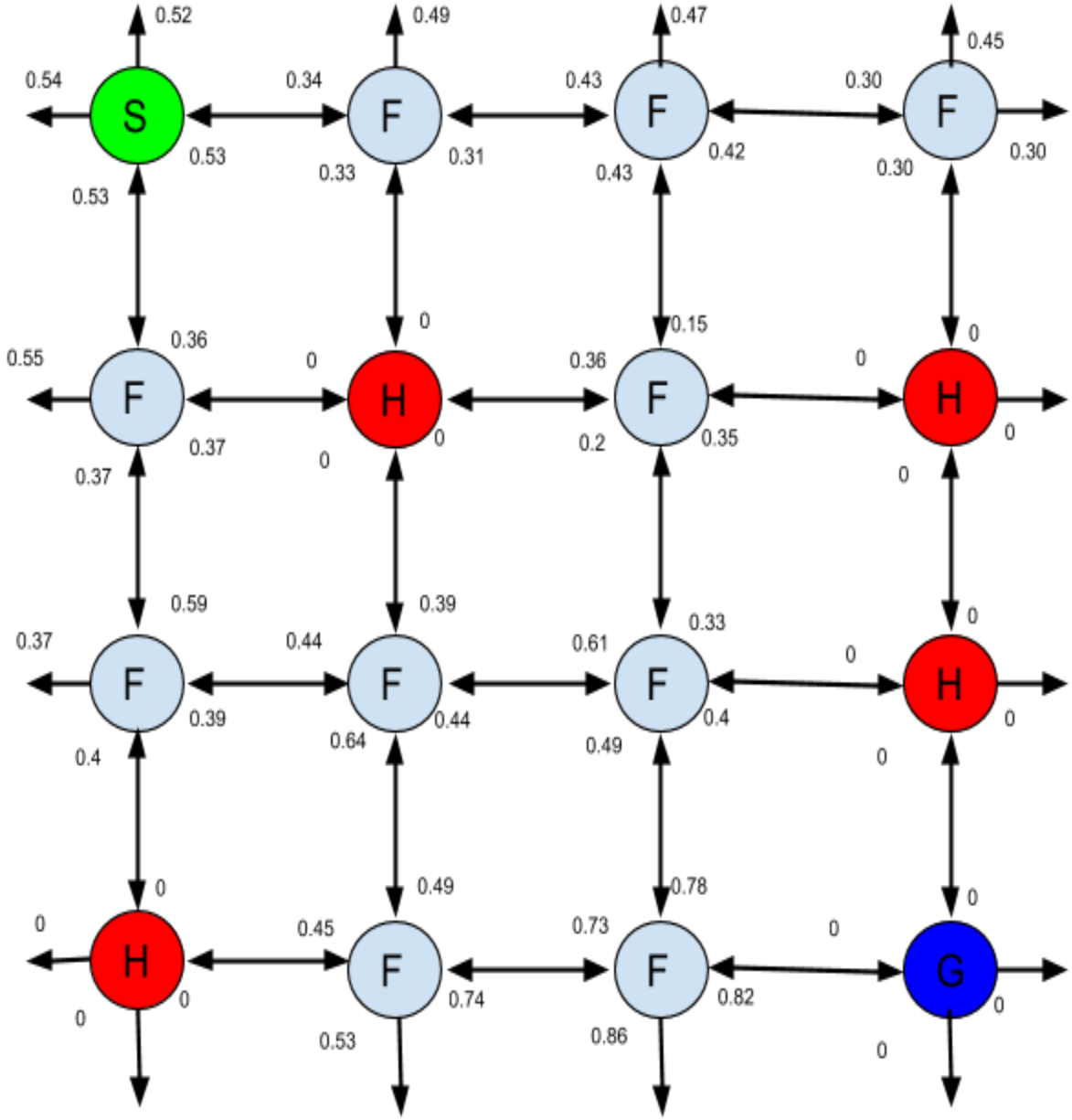
3. State-transition Graph

Below picture show the action for a given state



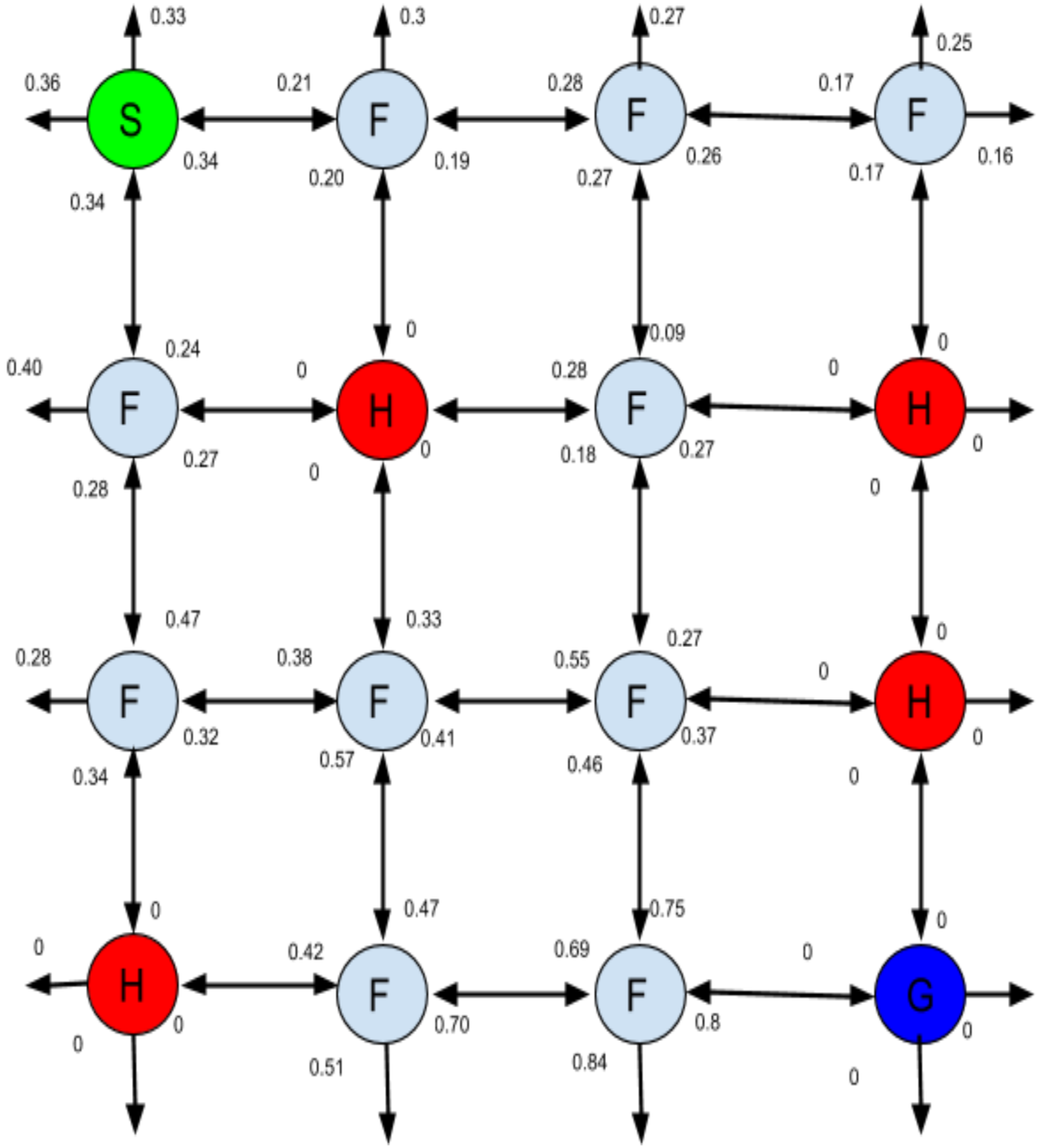
3.1 Value Iteration

Policy : {0,3,3,3,0,0,0,0,3,1,0,0,0,2,1,0}



3.2 Policy Iteration

Policy : {0,3,0,3,0,0,0,0,3,1,0,0,0,2,1,0}



4. Optimal Policy

1. If the policy obtained from both algorithms are the same, then It will be an optimal policy.
2. Find policy from some other method and match with given policy, if matched it will be optimal policy.
3. If a given policy is optimal then it will give the highest winning reward.
4. If slippery is removed from the ice environment then using optimal policy we can get full reward.

5. Experimental Results

Gamma	Algorithm	Policy	Reward	Time (ms)	No. of Iteration
0.1	Value Itr.	1 3 2 3 0 0 0 0 3 1 0 0 0 2 1 0	53	2.1	11
	Policy Itr.	1 3 2 3 0 0 0 0 3 1 0 0 0 2 1 0	50	4.19	21
0.2	Value Itr.	1 3 2 3 0 0 0 0 3 1 0 0 0 2 1 0	48	2.0	11
	Policy Itr.	1 3 2 3 0 0 0 0 3 1 0 0 0 2 1 0	49	6.84	41
0.3	Value Itr.	1 3 2 3 0 0 0 0 3 1 0 0 0 2 1 0	41	3.98	21
	Policy Itr.	1 3 2 3 0 0 0 0 3 1 0 0 0 2 1 0	40	5.17	21
0.4	Value Itr.	2 3 2 3 0 0 0 0 3 1 0 0 0 2 1 0	49	4.12	21
	Policy Itr.	1 3 2 3 0 0 0 0 3 1 0 0 0 2 1 0	43	7.4	31
0.5	Value Itr.	1 3 2 3 0 0 0 0 3 1 0 0 0 2 1 0	36	3.89	21
	Policy Itr.	1 3 2 3 0 0 0 0 3 1 0 0 0 2 1 0	49	4.32	21

0.6	Value Itr.	1 3 2 3 0 0 0 0 3 1 0 0 0 2 1 0	38	8.77	31
	Policy Itr.	1 3 2 3 0 0 0 0 3 1 0 0 0 2 1 0	51	3.98	21
0.7	Value Itr.	1 3 2 3 0 0 0 0 3 1 0 0 0 2 1 0	46	9.06	41
	Policy Itr.	1 3 2 3 0 0 0 0 3 1 0 0 0 2 1 0	75	4.58	41
0.8	Value Itr.	2 3 2 3 0 0 0 0 3 1 0 0 0 2 1 0	41	10.4	51
	Policy Itr.	1 3 2 3 0 0 0 0 3 1 0 0 0 2 1 0	40	4.74	21
0.9	Value Itr.	0 3 0 3 0 0 0 0 3 1 0 0 0 2 1 0	71	20.8	81
	Policy Itr.	0 3 0 3 0 0 0 0 3 1 0 0 0 2 1 0	70	9.21	31
0.999	Value Itr.	0 3 3 3 0 0 0 0 3 1 0 0 0 2 1 0	75	66.3	341
	Policy Itr.	0 3 0 3 0 0 0 0 3 1 0 0 0 2 1 0	79	4.83	31

6. Comparison of Policy and Value Iteration

In policy **iteration** algorithms, you start with a random policy, then find the value function of that policy (policy evaluation step), then find a new (improved) policy based on the previous value function, and so on. In this process, each policy is guaranteed to be a strict improvement over the previous one (unless it is already optimal). Given a policy, its value function can be obtained using the *Bellman operator*.

In **value iteration**, you start with a random value function and then find a new (improved) value function in an iterative process, until reaching the optimal value function. Notice that you can derive easily the optimal policy from the optimal value function. This process is based on the *optimality Bellman operator*.

7. Conclusions

Different values of gamma may produce different policies. Lower gamma values will put more weight on short-term gains, whereas higher gamma values will put more weight towards long-term gains. The closer gamma is to 1, the closer the policy will be to one that optimizes the gains over infinite time. On the other hand, value iteration will be slower to converge. Policy iteration does not change much with gamma value less than 0.8. Policy Iteration converges faster for higher value of gamma but converges slower for lower value. Higher values of gamma give a more optimal solution.