Problem Statement

In this work, we formalize the ice-cream problem in week 1, 2 and 3's changeling problem and share our results for the questions. Besides, we share our codes for plotting the map, solving the VI, PI and dealing with the multi-agents problem (with function approximation).

Week 1: formalize the problem

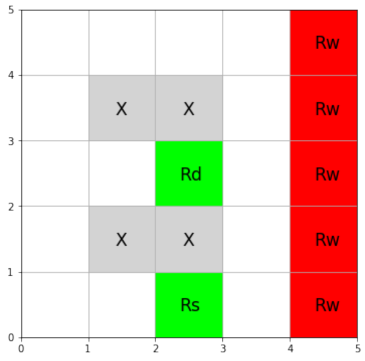
S is all the 5*5 cell. S = (i , j) where $i, j \in 0 \sim 4$.

We define A as an integer from 0,1,2,3,4, here we define 0 is stay still and 1 4 is left, up, right, down. $Action = [0 : (0, 0), 1 : (-1, 0), 2 : (0, 1), 3 : (1, 0), 4 : (0, -1)]$.

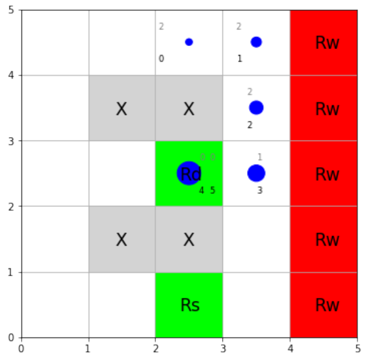
Probability transition is defined as following: $P(s_{t+1} \mid s_t, a_t) = p_e$ if $s_{t+1} = s_t + a_t$ and $P(s_{t+1} \mid s_t, a_t) = (1 - p_e)/4$ if $s_{t+1} = s_t + a_{\setminus t}$ where $a_{\setminus t}$ means all the actions except the desired one. And when $s_t + a_{\setminus t}$ is the wall, we just set it to s_t.

O = $\frac{2}{d_D^{-1} + d_S^{-1}}$ as in the document.

We write codes to draw our map and trajectories. Also, there are codes to move based on a pre-defined strategy (which will be influenced by p_e). Here is some examples. It support any size of the map and any layout of the map.



(a) Put your sub-caption here



(b) Put your sub-caption here

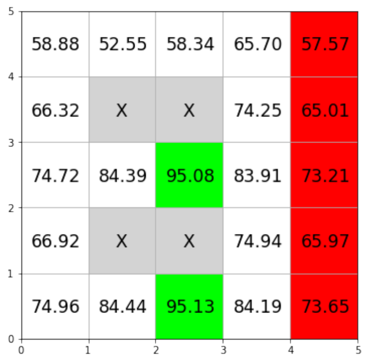
Figure 1: Put your caption here

Week 2

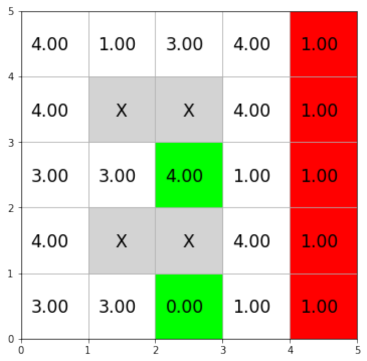
We add codes for VI and PI. We modify the plot function so that we can choose to show the moves, values and actions. It supports different rewards (default means reward for each blank region), horizons and discounts.

We start from the (2,4) point and set discount be 0.9, $p_e = 0.9$ and horizon = infinite. We set the reward for both stores are 10 and the punishment for the red zone is -1.

Here is our best actions in each position and there values:



(a) Put your sub-caption here



(b) Put your sub-caption here

Figure 2: Put your caption here

Answers to the Problems:

1. The trajectory is shown in right of figure 1. With p_e becomes smaller, the algorithm intends to move left instead of right since it becomes more possible to get to R_w.

2. There is only one optimal trajectories from star point to R_D calculated from VI and PI.

3. To make the agent intentionally entering the store, we can get each step a -1 reward (each non-coloered zone). Then we force the agent to go to the target as fast as possible.

4. To formulate a problem that robotic will end whenever it reach either ice-cream shop(R_d or R_s), we just need to set a conditional reward on reward state. Such that When it reach reward state, any other action except from staying at the same state will get a large negative reward. In this way, whenever it reach a reward state, it will end at that state.

5. To make the optimal strategy to visit both ice cream stores in same trajectory, we can still use an conditional reward for actions. Whenever it reaches an reward state, any other action except making the robotic leave that state will get a large negative reward.

Week 3

For multi agent problem, if we have n identical agents, then we will have 25^n states and will have action space of size 5^n.

Will build the following seven basis functions: first is the number of agents in the target; second is the number of agents in negative state (Rw); third is the number of agents in the wall (X region) and the last one the number of agents collide with each other. The sum of distance to the targets (two here), and if there is agent close to Rw. We believe with these seven we can calculate the value of a state.

In the function approximation part, we add two parameters p1 and p2 to decide the if a state will be chosen to be assessed and if an action will be counted in assessment. So, this code will take p1*p2 time compared with the whole algorithm. However, since we are using a learning rate, if it is very small, it will take a lot of time to converge and may use much more time. Here, we use a exponentially decreasing learning rate 0.95^t.

We found that, the features should be able to reflect your progress to the target (if you are close to the target, the features should reflect it). Otherwise it is hard for your algorithm to converge.

We write a code use function approximator to approximate n agents system. It has great flexibility: you can change number of agents, maps, rewards, your features, probability of what percent of states and next states to be counted. And the code can be changed to VI with several modification. However, we are not sure it is 100% correct since the convergence performance is not good.