Assignment 4: Signaling Policy

1 Introduction

Previously, we inferred another agent's goal as an outsider by observing the trajectory of states they passed across time. Here, instead of observing a state trajectory, we get to approach the problem from the agent's point of view by choosing which actions we want to take. For this assignment, we assume a cooperative agent that wants to reach its goal while helping an observer understand what it is. This means our chosen actions should try to signal our true goal while ruling out other goals as quickly and unambiguously as possible.

An agent's policy should change to reflect the reduction of ambiguity between goals. However, we do not directly control the policy. Instead, we perform value iteration to find the relative values of states and actions and use these to derive our policy. The environment dynamics and reward function define what makes a state valuable. However, an agent is not in control of the dynamics of the environment, so we change the reward function to reflect our desire to send cooperative signals.

Recall that reward is defined: $R(s', a, s)$, so we have a reward for each state, action, next state set. Before, the components of this reward function were the large reward or cost of special states (goal or trap) as well as the small cost of moving. Now, we want to add another term to our reward: a measure of how much ambiguity we are reducing by taking a particular action. In the previous assignment on goal inference, we found $P(s_{t+1}|s_t, g)$, the probability of observing the s_{t+1} given the current state s_t and the goal g.

$$P(s_{t+1}|s_t, g) = \sum_{a_t \in A_{s_t}} P(s_{t+1}|s_t, a_t)\pi(a_t|s_t, g) \tag{1}$$

There we used it to evaluate with a known s_{t+1}, which goal was most likely. Now we can take the ratio of:

$$r_{info} = \frac{P(s_{t+1}|s_t, g_{true})}{\sum_{g \in G} P(s_{t+1}|s_t, g)} \tag{2}$$

This is the likelihood ratio of the next state given the actual goal compared to all goals. Thus, it should be more rewarding to try to get to next states that are likely for the true goal and unlikely for the other goals.

For each state-action-next state combination, you should take the given reward and update it to reflect this additional r_{info}.

$$r_{new} = r_{original} + \alpha r_{info} \tag{3}$$

Where α affects how much preference is given to the information added. From this, you can create a new reward function for each state-action-next state, and using that new reward function, create a new policy that takes intention to signal into account.

2 Setup

Similar to the previous assignments, you will receive environmental dynamics, initial reward function for each goal, and other parameters of the environment. You should fill in the appropriate functions/class

provided and perform the additional steps necessary to get a policy for a cooperative agent with a particular goal.

The GetLikelihoodReward class should take in the environment dynamics and policies for all goals. The class callable takes in the agent's true goal and original reward function corresponding to that goal. To make this possible, the goal policies attribute must contain all of the goals. You specify which is the agent's actual goal using the true goal variable. You should return a new reward function corresponding to equation 3 in the same format as the original reward (nested dictionary of form state : action : next state : scalar reward)

3 Requirements

In addition to filling in the GetLikelihoodReward class, you should also write the appropriate additional code to get a policy that reflects signaling to an observer for each possible true goal in the environment. You should visualize the policy as before by providing graphs of the value table and policy (the same helper functions are in the attached visualization file). Write a brief commentary on what has changed between the original goal policies and the new signaling policies.

You should upload at least 2 files: the completed python file containing the GetLikelihoodReward class and a PDF containing the plots and commentary. Your additional implementation code can be a main function in the original file, an additional python file, or a Jupyter notebook.