Assignment 3: Goal Inference

1 Introduction

In this assignment, you will combine concepts from the previous assignments to build a model of Goal Inference. You are an observer who sees an agent travel through a series of states and must reason about which state they are likely trying to reach. The environment is similar to the grid-world of week 2's value iteration; however, there are multiple possible goals and obstacles. This is an implementation of Model 1 of a 2007 paper by Baker, Saxe, and Tenenbaum: *Action Understanding as Inverse Planning.* For further background refer to the paper, specifically the appendix.

On the highest level, this is a Bayesian Inference of the form:

$$P(\text{Goal}|\text{Actions, Environment}) \propto P(\text{Actions}|\text{Goal, Environment})P(\text{Goal}|\text{Environment}) \tag{1}$$

However, we do not observe the sequence of actions an agent takes. Instead, we observe the trajectory of their states. Thus, we can rewrite our Bayesian Inference problem in terms of the state sequence:

$$P(g|S_{1:T}) \propto P(S_{2:T}|s_1, g)P(g) \tag{2}$$

Here, s_1 is the initial state and $S_{1:T}$ is a sequence of observed states. We do not need to explicitly worry about the environment because it is incorporated into the reward and transition (e.g. which states have high costs, which states are goal states, and the state space itself). This is now a sequential Bayesian updating problem. Like typical Bayesian Inference, it is broken down into posterior $\propto$ (Likelihood)(Prior), however, this is done at each time point.

Given an initial state and a prior over the goals, we can sequentially update the probability of the goal given this information. As a result $P(S_{2:T}|s_1, g)$ breaks into:

$$P(S_{2:T}|s_1, g) = \prod_{t=1}^{T-1} P(s_{t+1}|s_t, g) \tag{3}$$

Where:

$$P(s_{t+1}|s_t, g) = \sum_{a_t \in A_{s_t}} P(s_{t+1}|s_t, a_t)\pi(a_t|s_t, g) \tag{4}$$

In equation 4, the first term is the transition and the second is the policy. To find the policy, we perform value iteration as in last weeks assignment (see Figure 1) until we get convergence of the state values.

Instead of a deterministic optimal policy, we assume agents have a probability distribution over actions based on the value of taking the action from their current state. This is necessary to infer goals from non-optimal state trajectories. We choose an action according to:

$$\pi(a_t|s_t, g) \propto e^{\beta Q_g^{\pi}(s_t, a_t)} \tag{5}$$

This is a *Boltzmann Policy* where actions are drawn from a 'softmax'. The Q-function is the state-action value function. When it has converged, value iteration gives the value of each state under the (near) optimal policy. The Q-function $Q(s, a)$ is the value of taking action a while in state s. The β parameter

is a measure of how much noise an agent is predicted to have from its optimal path. $\beta = 0$ corresponds to a random walk.

Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $v \leftarrow V(s)$
| $V(s) \leftarrow \max_a \sum_{s',r} p(s', r \,|\, s, a)\big[r + \gamma V(s')\big]$
| $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
$\pi(s) = \arg\max_a \sum_{s',r} p(s', r \,|\, s, a)\big[r + \gamma V(s')\big]$

Figure 1: From Sutton and Barto 2018

2 Setup

Your job is to find the probability of each of the goals from observing a set of sequential states an agent moves through. Like last time, you will be given the environment dynamics in the form of transition table and reward table. However, there will be one reward table for each of the possible goals, allowing you to condition your policy on the goal. Corresponding to this will be a vector of states mapping the path of an agent through its trajectory to a specific goal. At **each** time point in the trajectory, you should evaluate the posterior probability of each goal given the trajectory (observed up to that time point) and the environment from Equation 1.

Note that there are some changes in the environment. Unlike last time, there are multiple trap states which may form a barrier between the agent and the goal. Additionally, the action set includes diagonal moves. The cost of moving is the negative square root of the distance between the current state and next or -1 if the move leads off the board. Your value iteration algorithm from assignment 2 should be general enough to these absorb changes as the format of input is fulfills the specifications from last week.

You will get two environments which share:

- **State Space**
- **Action Space**
- **Transition Table:** $P(s'|s, a)$ The state transition, a nested dictionary of form state : action : next state : probability. The transition only includes state-action-nextstate combinations that have non-zero probability.
- **Beta β:** noise tolerance
- **Gamma γ:** the decay of future rewards
- **Observed State Trajectory List:** a trajectory of ordered states where the first is the initial state s_1 and the last is the goal state.

For each goal in each environment, you will get:

- **Reward Table:** $R(s', s, a)$ The deterministic reward function, a nested dictionary of form state : action : next state : reward. One for each goal.

3 Requirements

You may choose how to initialize your state-values (typically you just set them all to 0) and your convergence tolerance (something sufficiently small).

For this assignment you should include your code as the python file 'GoalInference.py' as well as a PDF write-up that includes:

- A visualization of the value table and policy for each goal (A, B, C) in each environment using the provided functions.
- A graph plotting the posterior probability of each goal at each time point for each given trajectory sequences.

Environment 1 in the code corresponds to the image below where there are three possible goals. There are three reward functions representing this map, one for each of the possible goals. Below are the arguments used to generate the environment. They can be used to help visualize the value and policy tables.

```
goalStates = [(6, 1, "B"), (1, 5, "C"), (6, 4, "A")]
gridWidth = 7
gridHeight = 6
trapStates = [(3,0), (3,1), (3,2), (3,3)]
```

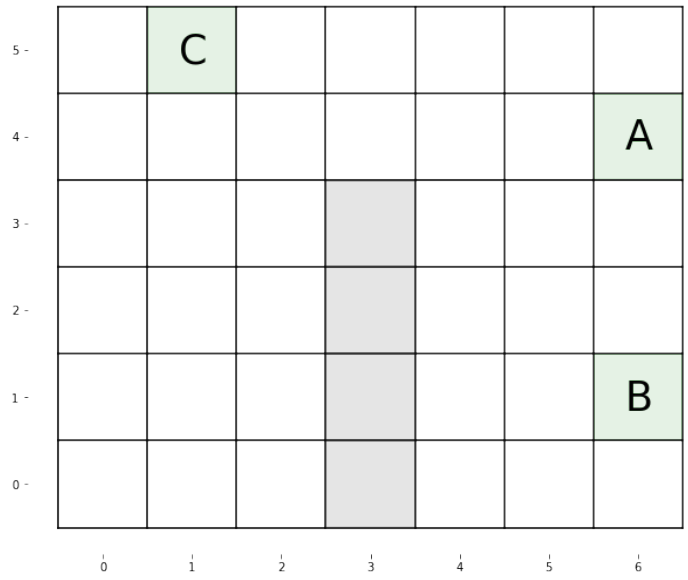

Figure 2: Environment 1 in Example

Environment 2 is similar, with the same goal states and observed trajectories but where the state (3,1) is no longer a trap state.