CS747 - Assignment 2 Report

Shashank Kumar

October 23, 2020

Directory Structure

- *encoder.py* - File to encode maze as transition probabilities and rewards
- *decoder.py* - Decode the policy into directions
- *planner.py* - Main script to call to different algorithms
- *vi.py* - Code to implement value-iteration algorithm
- *hpi.py* - Implementation of Howard's policy iteration algorithm
- *lp.py* - Linear programming implementation
- *helper.py* - A file for additional tasks like reading and parsing instances

Implementation Details

Array Dimensions

Transitions(T) and rewards(R) are numpy arrays of dimensions (S, A, S), Q is defined to be an array of the dimensions (S, A) and V is of dimensions (1, 1, S) where S is the number of states and A is the number of actions

Value Iteration

While $||V - V_{prev}||_\infty > \epsilon$ where $\epsilon = 10^{-10}$, update:

$$V_{prev} = V \tag{1}$$

$$V = max_{a \in A} \sum_{s^{`} \in S} T * (R + \gamma * V_{prev}) \tag{2}$$

Howard's Policy Iteration

While $P\ != \ P_{prev}$ where P is the current policy, update:

$$P_{prev} = P \tag{3}$$

$$P = Policy(V_P) \tag{4}$$

Initialize the policy to take action 0 at all the states.
Compute the value function using the policy.
Update the policy as per the computed value function.

Helper

Assumption: The MDP files have multiple space in the discount line. The helper file, however, requires it to be a single space.
File to parse MDP file to generate transition and reward arrays, generating transitions using grid files and reading value-policy file.

Maze Formulation

1. No transitions are generated from and to the walls
2. No transitions are generated from the final states
3. Transitions from empty tile can have four actions - North, East, South, West
4. Every transition has a probability one
5. Reward of 100 is awarded on reaching the final state
6. Reward of -1 is awarded on moving along the tiles
7. Discount value used is one
8. Goal is to maximize the reward, which forces for the shortest path to be taken and avoid any loops

---

Conclusions

Performance comparision:
Value iteration > Howards policy iteration > Linear programming

---

References

[1] Python documentation available at `https://docs.python.org/3/`

[2] Numpy documentation available at `https://numpy.org/doc/`

[3] PuLP documentation available at `https://coin-or.github.io/pulp/`