Project: Markov Decision Process

ITCS 6150/8150

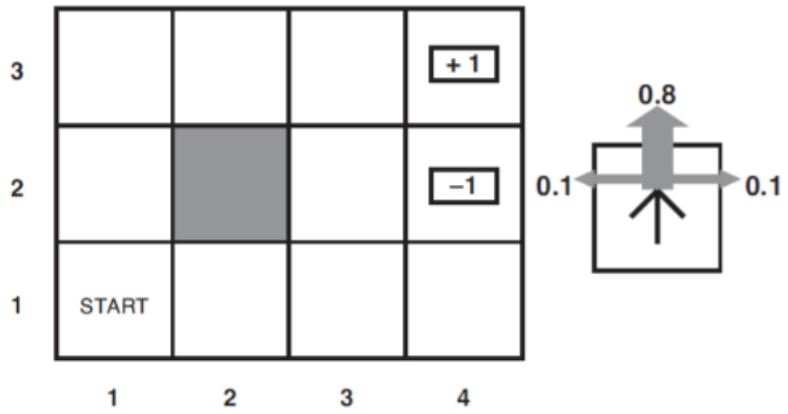
Student Name: Rucha Pramod Visal
Student ID: 801257824

Intelligent Systems Project 3

Value Iteration algorithm to solve a Markov Decision Process (MDP)

Problem Formulation

A Markov decision process (MDP) is a sequential choice problem with a Markovian transition model and additive rewards in a fully observable, stochastic environment. A set of states, a set of actions, a transition model, and a reward function make up the system. Here's an example:



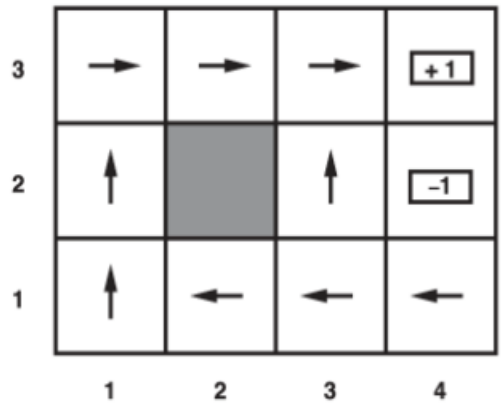
This is a simple 4 x 3 environment, and each block represents a state. The agent can move left, right, up, or down from a state. The "intended" outcome occurs with probability 0.8, but with probability 0.2 the agent moves at right angles to the intended direction. A collision with the wall or boundary results in no movement. The two terminal states have reward +1 and -1, respectively, and all other states have a constant reward (e.g., of -0.01).

Also, a MDP usually has a discount factor γ, a number between 0 and 1, that describes the preference of an agent for current rewards over future rewards.

1. **Agent:** An RL agent is the entity which we are training to make correct decisions (for e.g.: a robot that is being trained to move around a house without crashing).
2. **Environment:** The environment is the surrounding with which the agent interacts (for eg: the house where the Robot moves). The agent cannot manipulate the environment; it can only control its own actions (for eg: the Robot cannot control where a table is kept in the house, but it can walk around it to avoid crashing).
3. **State:** The state defines the current situation of the agent (for eg: it can be the exact position of the Robot in the house, or the alignment of its two legs, or its current posture; it depends on how you address the problem).
4. **Action:** The choice that the agent makes at the current time step (for eg: it can move its right or left leg, or raise its arm, or lift an object, turn right, or left, etc.). We know the set of actions (decisions) that the agent can perform in advance.
5. **Policy:** A policy is the thought process behind picking an action. In practice, it is a probability distribution assigned to the set of actions. Highly rewarding actions will have a high probability and vice versa.

Policy

A solution to an MDP is called a policy π(s). It specifies an action for each state. In an MDP, we aim to find the optimal policy that yields the highest expected utility. Here's an example of a policy.



There are many iterative solutions to obtain the optimal solution for the MDP. For e.g.:

- Value Iteration
- Policy Iteration
- SARSA
- Q-Learning

We are using Value Iteration algorithm to perform and solve Markov Decision Process (MDP).

Value Iteration

Value iteration is an algorithm that gives an optimal policy for an MDP. It calculates the utility of each state, which is defined as the expected sum of discounted rewards from that state onward.

$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U(s')$$

This is called the Bellman equation. For example, the utility of the state (1, 1) in the MDP example shown above is:

$$\begin{aligned} U(1,1) = -0.04 + \gamma \max[\; & 0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1), && (Up) \\ & 0.9U(1,1) + 0.1U(1,2), && (Left) \\ & 0.9U(1,1) + 0.1U(2,1), && (Down) \\ & 0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1) \;]. && (Right) \end{aligned}$$

For n states, there are n Bellman equations with n unknowns (the utilities of states). To solve this system of equations, value iteration uses an iterative approach that repeatedly updates the utility of each state (starting from zero) until an equilibrium is reached (converge). The iteration step, called a Bellman update, looks like this:

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U_i(s')$$

Then, after the utilities of states are calculated, we can use them to select an optimal action for each state.

$$\pi^*(s) = \operatorname*{argmax}_{a \in A(s)} \sum_{s'} P(s' \mid s, a) U(s')$$

Program Structure

I have implemented the Value Iteration algorithm to solve a Markov Decision Process (MDP) using **Python**.

Global Variables:

Variable Name	Description
Board	Board structure for policy calculation
Cell	Cell structure for each index

Functions and Procedures:

Function Name	Description
getTransitionSum	This function generates transition sum of each cell and each action.
output	This function is used to print the output i.e., the utility array and policy matrix.
solve	This function is the main function. This will take the discount and reward values to calculate the utility and policy.

Input/Output:

Case 1:

Discount : 0.99 Reward : -0.04

```
============================
[
0.6507  0.5927  0.5601  0.3380  0.7166  0.6413  -1.0000 0.7762  0.8439  0.9051  1.0000  ]
RIGHT   RIGHT   RIGHT   T
UP      0       UP      T
UP      LEFT    UP      LEFT
============================
```

Case 2:

Discount : 0.5 Reward : -0.04

```
============================
[
-0.0620 -0.0533 -0.0199 -0.0745 -0.0406 0.0663  -1.0000 0.0086  0.1255  0.3824  1.0000  ]
RIGHT   RIGHT   RIGHT   T
UP      0       UP      T
UP      RIGHT   UP      DOWN
============================
```

Case 3:

Discount : 0.5 Reward : -0.25

```
============================
[
-0.4745 -0.4551 -0.3989 -0.4838 -0.4451 -0.2550 -1.0000 -0.3765 -0.2136 0.1445  1.0000  ]
RIGHT   RIGHT   RIGHT   T
UP      0       UP      T
UP      RIGHT   UP      LEFT
============================
```

Case 4:

Discount : 0.99 Reward : -0.25

```
============================
[
-0.6590 -0.5482 -0.2395 -0.5978 -0.3656 0.1566  -1.0000 -0.0545 0.2993  0.6188  1.0000  ]
RIGHT   RIGHT   RIGHT   T
UP      0       UP      T
UP      RIGHT   UP      LEFT
============================
```