The RL Agent - Documentation

Instructions

- The program to be run is `main.py`
- The average reward and highest reward over 1000 iterations will be displayed in terminal, after which the visualisation will be shown (done using OpenCV library).
- It progresses step-by-step, and you have to press any key for it to proceed.
- The code for the RL algorithm is under `gridsolver.py` and the code for visualisation using OpenCV is under `gridvisualiser.py`.

Process and Methods Used

- I tried using two methods - Q-learning and SARSA. Based on the results obtained over 1000 iterations, I found that Q-learning worked better for this model, so I went ahead with it.
- I made a Q-table of 3 dimensions: first 2 dimensions corresponding to each coordinate in the grid, and the third coordinate corresponding to the action (0,1,2 or 3).
- In other words, it is a set of all $Q(s_t, a_t)$ where s_t corresponds to each state and a_t corresponds to each action.
- The initial Q-values for start and goal positions were given as 10, whereas all the other coordinates had 0.
- This Q-table was then updated over N episodes following this formula:

  $Q(s_t, a_t) = (1 - \alpha) * Q(s_t, a_t) + \alpha * (r_{t+1} + \gamma * max_a Q(s_{t+1}, a_{t+1}))$

  Where:

  ```
  - $s_t$ is the current state
  - $a_t$ is the action taken from that state, which was chosen following $\epsilon$-greedy
  policy.
  - $s_{t+1}$ is the state reached after carrying out $a_t$ from $s_t$

  - $\alpha$ is the step length taken for updation
  - $r_{t+1}$ is the reward obtained for carrying out action $a_t$ from state $s_t$
  - $\gamma$ is the discount factor for future rewards
  - ${max}_{a}Q(s_{t+1},a_{t+1})$ is the maximum reward that can be obtained from state
  $s_{t+1}$
  ```

- The values for each constant I used were:
  - $N = 10000$
  - $\gamma = 1$
  - $\alpha = 0.9$
  - $\epsilon = 0.1$ for first 1000 iterations, then 0.01
- Once the Q-table was filled, the values were used to find a path from start to goal, following a greedy policy wherein the action corresponding to maximum Q-value was chosen for each state.
- The path was then stored and then visualised using OpenCV library.

Results

- Over 1000 iterations, an average reward of around -25 was obtained and the highest reward obtained was -23.

References Used

- The Complete Reinforcement Learning Dictionary
- What is a Policy in Reinforcement Learning?
- Reinforcement Learning and the Markov Decision Process
- Reinforcement Learning: Value Function and Policy
- Bellman Equation Basics for Reinforcement Learning (Video)
- Bellman Equation
- Temporal Difference Learning Wiki
- Simple Reinforcement Learning: Temporal Difference Learning
- Temporal-Difference (TD) Learning
- SARSA Reinforcement Learning
- Q-Learning vs. SARSA