Reinforcement Learning Project
Agent Moving in a Grid
Design Lab : CS69202
March 26, 2025
Submission Deadline : March 31, 2025

Objective

Dirt in a room settles along the edges of walls and areas where there are less footfalls. Model a room with dirt patches, along the lines of a grid-world environment, and the agent (robot) has to clean them. The agent can move around and clean dirt patches. The goal is to find the optimal policy that minimizes the number of movements and cleans the room efficiently. The environment is kept static throughout learning. The agent needs to learn to clean the dirt patches near the walls while avoiding crashing into walls. *That is, the environment is a Grid World Problem wherein there are empty cells, obstacles/walls and a cell where dirt is formed. Your task is to train an agent which cleans **ONE** space composed of dirt. Only one space of dirt is present.*

Your task is to build an agent using RL algorithms to help solve the problem. In particular, you shall be doing the following tasks:

1. Design of Environment Class [50]

Design an Environment Class that models the given specifications. You have to build the state space, action space for the robot and the reward structure following the structure of Gym Environment Classes. The reward structure can however change depending on the requirement of the algorithms. See here and here for reference. In the reference, the environment is a 4x4 Frozen Lake. Create a custom environment using the code and change the grid sizes like 10x10, 100x100, 1000x1000, 10000x10000 etc. More details below.

2. Visualization of Environment Class [45]

Design a visualization tool for rendering how the algorithm controls the robot. The rendering function should preferably be part of the Environment Class. This will help demonstrate the agent's actions. **Text-based visualizations will not be accepted.** Your visualization should

clearly illustrate various features specific to your design environment and show the current state, action and reward received.

3. Q-Learning[35]

Demonstrate the Q Learning algorithm using the Env class designed above.
For Q learning, do the following hyperparameter searches :

1. Learning Rate (α)
2. Discount Factor (γ)
3. Exploration Rate (ε) in epsilon-Greedy Strategy. (0.05,0.1,0.2,0.4,0.5)
4. Step size : Maximum actions taken by an agent per episode. (10,100,1000,10000,100000)

4. Deep Q-Learning (DQN)[55]

Demonstrate the Deep Q Learning algorithm using the Env class designed above. Do the following hyperparameter searches

1. Learning Rate (***α***)
2. Discount Factor (γ)
3. Exploration Rate (ε) in epsilon-Greedy Strategy. (0.05,0.1,0.2,0.4,0.5)
4. Replay Buffer Size : Keeping a history of steps played by an agent (1e7).
5. Step size : Maximum actions taken by an agent per episode. (10,100,1000,10000,100000)
6. Batch Size : Selecting a batch of steps from the replay buffer (32,64,128,256).
7. Update frequency : Update the target neural network based on cumulative steps travelled by the agent. Example : Update the neural network after processing 5 batches of steps from Replay Buffer.

5. Evaluation and Report[65]

Compare the results from the above algorithms, and give insights if any of why the results are better or worse for one algorithm compared to others. Generate Square Grids of size (10,100,1000,10000,1000000,...,1e7) to perform the agent moving using Q-Learning and DQN. Also, change the environment in such a way that there are multiple obstacles with appropriate constraints of putting obstacles . Report the time taken.

For the cells which are in the borders of the grid, consider that any action taken will make the agent stay in the same cell. I.e. if one wants to go left/up from the topmost left cell, the agent stays in the same cell unless it moves right/down.

Note: The program can be written in preferably Python programming language, and should run in Linux.

6. Multiple Cells of Dirt [BONUS : 100]

In addition to the above, try to modify the algorithm (any one of Q Learning or DQN) which tackles the same problem but the agent clears multiple cells containing dirt i.e. there are more than one goal states. Also visualise them.

Note:

1. *Bonus marks will be awarded based on Implementation and appropriate mentioning of configuration applied to achieve this case. Also refrain from using AI help for this part as we shall put a plagiarism check on this.*

2. *In general, we will put plagiarism checks on the actual code also.*

Submission Details

1. ZIPPED Code Distribution
2. A detailed report/manual of your work

Submission Guidelines

1. You should preferably use Python, which should run on a Linux Environment.
2. You should include a README file in the code distribution, which describes how to run the codebase and the important files.
3. Submission will be done in Moodle.
4. There should be 2 Python files (24CS60R01_QL.py, 24CS60R01_DQN.py given roll number 24CS60R01). A bonus Python file shall be made for the bonus part (24CS60R01_Bonus.py).
5. A PDF FIle (24CS60R01_Report.pdf if given roll number is 24CS60R01)
6. For any large files which cannot fit in 20 MB can be shared via Google Drive Link in Readme.
7. Zip file containing all of the above.

You should not use any code available on the Web. Submissions found to be plagiarized will be awarded zero marks.