Project proposal

Arrived on the moon with the Lunar Lander, it is now time to explore. For this purpose, the Lunar Explorer was developed. Its objective is to get to a specific location. On the way there, there are potential interesting ores that could be a source of yet unknown metals (green cell). The Lunar Explorer aims to collect as many other precious metals as possible (blue cell). However, the Lunar Explorer has a limited drilling capacity (e.g. 3 drillings). It will therefore not be able to drill all the available resources but will have to select the most profitable ones. Furthermore, its landing point has been precisely defined (black cell). Overall, the Lunar Explorer should explore and discover the optimal exploitation of the current map.

However, no gas station is yet available on the moon. The Lunar Explorer can only use the initially available fuel and avoid useless trips - translating to a penalty of “” for each iteration. The lunar field is also not very friendly. This is why, depending on the field, the Lunar Explorer can suffer from unexpected behaviours and has to adapt its movements.

The goal of this environment is to have an agent balance between the two main mechanics – minerals and terrain type – in order to exploit the map.

Example of a standard map, no movement tile is reflected:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **1** | **…** | **…** | **…** | **…** | **…** | **…** | **…** | **…** | **…** | **n** |
| **1** |  |  |  |  |  |  |  |  |  |  |  |
| **…** |  |  |  |  |  |  |  |  |  |  |  |
| **…** |  |  |  |  |  |  |  |  |  |  |  |
| **…** |  |  |  |  |  |  |  |  |  |  |  |
| **…** |  |  |  |  |  |  |  |  |  |  |  |
| **…** |  |  |  |  |  |  |  | -1 |  |  |  |
| **…** |  |  |  |  |  |  |  |  |  |  |  |
| **…** |  |  |  |  |  |  |  |  |  |  |  |
| **…** |  |  |  |  |  |  |  |  |  |  |  |
| **n** |  |  |  |  |  |  |  |  |  |  |  |

TABLE 1 Example of the lunar explorer environment

*Legend*

* • Start location
* • Potentially interesting mineral
* • End location, stops simulation

Environment

State space: , with

* being the horizontal position
* being the vertical position
* being the horizontal speed,
* being the vertical speed,

|S| = map dimensions \* possible speeds \* possible types = [x \* y] \* [3 \* 3]

* For a 30\*30 map, this represents 8100 possible states.

Action space: , |A| = 6

* On the 30\*30 map, this represents a Qtable of size 48600.

Movements

The agent can move around the map by going in either direction or stopping. The speed can be seen as a 2D vector of two integer values (Vx, Vy). In discrete mode, the speed can be either “moving” (-1, 1) or “not moving” (0). Only one speed component can be activated at a time (no diagonal movements). The speed behaves as follow:

* The speed encodes the last movement performed. Going right (or left) sets to 1 (resp -1) the horizontal speed, down / up set to 1 (resp -1) the vertical speed for the next state.
* Doing nothing means the explorer stops and resets the speed to 0.
* No cost in turning: When going from horizontal to vertical or opposite, the absolute value of the speed is carried, only the orientation changes. This is required to keep the speed consistent across all manhattan paths and avoid shortcurts on slow tiles.
  + This means that going right then up results in the Vx, Vy speed to go from (0,0) -> (1, 0) -> (0, -1)
* Going into an opposite direction resets the speed to 0 (left to right, top to bottom)

The Next state is relative to the absolute speed in the actual state, the current and next tile type, and the chosen action:

1. Speed type “slow”: This terrain is frail and can break under the explorer’s weight: advance carefully.
   * If the speed is 0, the safely lands on the tile. If the speed is not 0, then the rover has a chance to fall and the simulation stops. This means that the explorer has to stop before every slow tile it wants to go on.
2. Speed type “random”: This terrain is covered in craters, don’t get lost ! There is a change to be moved to adjacent tiles when arriving upon it.
3. Speed type “normal”: Regular moon terrain. The agent will always move to the juxtaposed state, according to the action.
4. Speed type “fast”: This road-like terrain is particularly adapted to the explorer’s locomotion. The agent has probability to move to the juxtaposed state and probability to move two states away from the current state, according to the action.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Current state with speed type** | | **Abs speed** | **Action** | **New state** | **Probability** |
|  |  | 0 |  |  |  |
|  |  | 1 |  |  |  |
|  |  | 1 |  |  |  |
|  |  | 0, 1 |  |  |  |
|  |  | 0, 1 |  |  |  |
|  |  | 0, 1 |  |  |  |
|  |  | 0, 1 |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Solution proposal

* Which algos do we want to try and why?

I would say we test the algorithms that we have developed by adapting them to the ‘framework’/setup at our disposal. On my side, I have:

* Qlearning
* Sarsa
* DynaQ
* DQN (Simple, dual network, MSE)

If available, we could try eligibility traces (I think you had them ?)

We have the DQN developed in today’s lab

My approach would be: We settle for an epsilon and a decaying methodology, and we explore different algorithms, see how they react on the same map, etc.

From discrete to continuous

If time allows it, the ‘position’ and ‘speed’ observations can be moves to continuous. In this case, similar to the lunar lender, the actions increment the speed by a set amount of acceleration each time they are called to stay with discrete actions. The changes will be:

* Some tiles can slightly change behavior (ex: speed tile ‘propulsing’ you by adding speed, only activated above a set amount of speed; craters having moving probabilities of being lost depending on speed value, etc…). The speed can stay limiter to the bound [-1; 1] in order to have a single tile detection in the board management.

State space: , with

* ‘Activation grid’ of player position
* Fov: tile type of close tiles
* being the horizontal speed,
* being the vertical speed,