Single Agent Renforcement Learning resource finding Innopolis.RL.2022.Team6

Alaa Aldin Hajjar
Innopolis University
Innopolis, Russia
a.hajjar@innopolis.university

Lionel Randall Kharkrang
Innopolis University
Innopolis, Russia
l.kharkrang@innopolis.university

1 Introduction:

With the increase usage of autonomous robots in industry such as healthcare and agriculture. An important challenge for industry is location of finding an optimal path for a robot to navigate to obtain require resource such as oil,minerals etc. The challenge is exploring enough of the space , while also managing resources like amount of oil used for exploring and navigating. In this project we aim address this challenge by creating a single agent Reinforcement learning system that can search for materials in given peace of land, so for achieving this we built a model with some kind of simulation to test its performance

2 Literature Review:

Numerous Reinforcement Learning algorithms have been used in robotics, according to [4]. Furthermore, a specialized use of Deep Reinforcement Learning in Robot navigation has been documented [5]. For a mobile robot operating in a dynamic environment, a path planning technique based on the improved Q-learning (IQL) algorithm plus a few heuristic searching strategies is suggested in [2]. In IQL, a novel exploration approach is utilized that blends Boltzmann exploration and *epsilon*-greedy exploration. [3], In contrast to the conventional method, deep reinforcement learning was presented in this study as an end-to-end method for mobile robot navigation in an uncharted environment. In this study, a dueling architecture based double deep q network (D3QN) is evolved from dueling network architectures for deep reinforcement learning (Dueling DQN) and deep reinforcement learning with double q learning (Double DQN). Via the D3QN algorithm, a mobile robot may progressively understand the surroundings through its amazement and learn to navigate to the goal location independently using only an RGB-D camera. In [1], A new concept for the states space was used to limit the number of states in the Q-learning algorithm in order to solve the challenge of mobile robot navigation in dynamic environments. This has the effect of lowering the Q-size table's and speeding up the navigation algorithm as a result. We further adopted the strategy similar to what is described in [1], due to the short training period and ease of execution. In order to take this into account, we divided our target region into a two-dimensional grid search space, as will be covered below.

3 Methodology:

3.1 Environment:

The environment would in real world scenario involved a convex complicated surface area, represented in 3D. However, in our case , we are tackling the 2D scenario, and limiting our search space in a grid search. The grid will have the start state, points where it deposits the oil, and states where the actual oil field is located. The oil field or multiple oil fields are spread over in a a square of side length l, for simplification. The reward was manually designed. The agent i.e our robot, will receive or incur negative reward for each move every time it digs, and a positive reward when it discovers oil and transports it to the oil points it needs to deposit. The grid also has randomly marked locations where the robot can dig but continues no oil and is rather more harmful to dig , than regular cells.

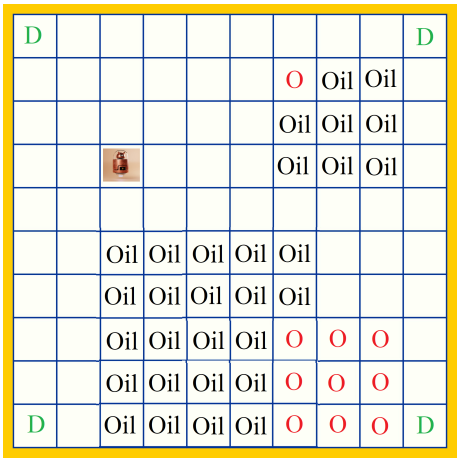

Figure 1. representation of the environment

3.2 Agent:

The agent is a robot that can move and Dig or not dig. There are associated rewards of the robot moving in the environment, reward for digging in the right place and not the right place. The robot has to explore , find patches where oil is located, decide based on its own evaluation, whether to transport the oil to certain drop-off locations or continue exploring for oil. The robot also has a maximum oil load capacity

3.3 Exploration

For exploration we use primarily ϵ-greedy. We also took into account softmax policy in the approach, with varying values of temperature

3.4 Action:

The agent can do one of the actions moveUP, moveDown, moveLeft, moveRight, dig, transportOil.
and each time the robot dig in oil field the oil field value will change and the side length of the square will change.

3.5 Reward:

moveUP, moveDown, moveLeft, moveRight: here the reward is -2 for moving and spending energy.
dig: if the agent dig in a place that contains no oil,the robot will get -1 reward. If the robot digs in a place that contain solid rocks (harming the robot) the robot will get -5 even if there is oil. If the robot digs in a place that contains oil the robot will get +5 reward from taking oil from the ground, but if his load is full the robot will get -1 reward instead.
transportOil: in this action the robot will get robot Load*10 - Manhattan distance to the nearest delivery cell, and if the robot try to do the action from a delivery cell (he is inside the delivery cell) the robot will get -1 reward to force him to go out of the cell.

3.6 Q_table

the state is made from the following:
the load of the robot.
the x value of the robot on the grid.
the y value of the robot on the grid.
followed with n fields, and each field is the square side length of an oil field. In our training example we have two fields, so the number of possible states is number of possible robot load values * possible robot x values on the grid * possible robot y values on the grid * the possible side square length for each oil field. and this is at the end the number of rows in the Q_{table} and the number of columns is the number of possible actions.

3.7 Algorithms

: We followed a mix of value-action algorithm approaches. We use Q-Learning 2, SARSA 1, and a mix of Q-learning and SARSA, where we applied Q-learning for half the time step and SARSA for the remaining number of time steps.

Algorithm 1 SARSA

procedure SARSA($\mathcal{X}$, A, R, T, α, γ)
  $Q : \mathcal{X} \times \mathcal{A} \rightarrow \mathbb{R}$ randomly
  while The table didn't converge: **do**
    Begin with $s \in \mathcal{X}$
    Choose an action a, $a \leftarrow \pi(s)$
    while s is not terminal **do**
      $r \leftarrow R(s, a)$
      $s' \leftarrow T(s, a)$
      Choose a' from s' , $a \leftarrow \pi(s')$
      $Q(s, a) \leftarrow Q(s, a) + \alpha \cdot (r + \gamma \cdot Q(s', a') - Q(s, a)$
      $s \leftarrow s'$
      $a \leftarrow a'$
  return Q

Algorithm 2 Q-learning

procedure Q-Learning($\mathcal{X}$, A, R, T, α, γ)
  Initialize $Q : \mathcal{X} \times \mathcal{A} \rightarrow \mathbb{R}$ arbitrarily
  while Q is not converged **do**
    Starting in state $s \in \mathcal{X}$
    while s is not terminal **do**
      $a \leftarrow \pi(s)$
      $r \leftarrow R(s, a)$
      $s' \leftarrow T(s, a)$
      $Q(s, a) \leftarrow Q(s, a) + \alpha \cdot (r + \gamma \cdot \max_{a'} Q(s', a) - Q(s, a)$
      $s \leftarrow s'$
  return Q

4 Results:

The following consists of graphs of mean rewards of various algorithms implemented over a successive number of episodes. For all algorithms using greedy epsilon policy, we implemented a decay rate approach. i.e for successive episodes, we exponential decrease the ϵ value , to explore in the initial episodes and exploit in the further episodes, a value of 0.005, showed optimal results. For the softmax policy, we took a Temperature value of 5 for SARSA and 0.85 for Q-Learning.

Results				
Alg	rw	ep	α	mS
Q_L	159.0	6000	0.05	300
SR	231.00	2000	0.05	300
Q_L&SR	231.00	4000	0.05	300
SM_SR	-691	6000	0.001	400
SM_Q_L	-474	6000	0.001	400

Now by looking to the results that we got from our models we can see that each one of them acted differently so let us plot some graphs for each one of them to show the training procedure.

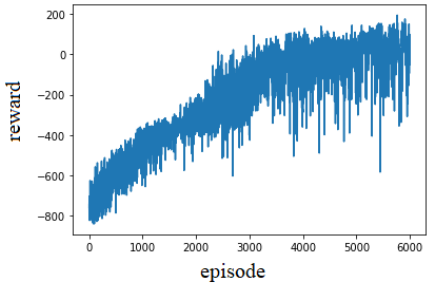

Figure 2. reward per episode while training for Q-learning

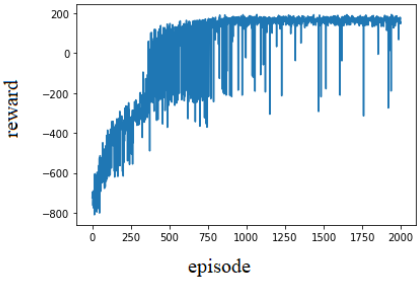

Figure 3. reward per episode while training for SARSA

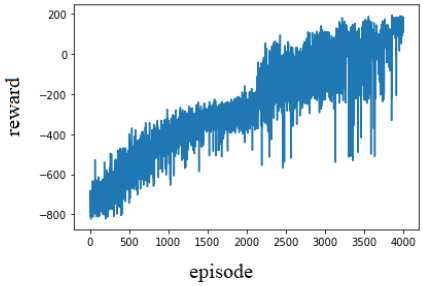

Figure 4. reward per episode while training for mix of Q-learning and SARSA

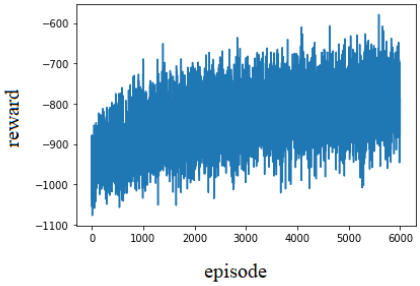

Figure 5. reward per episode while training for SARSA with softmax policy

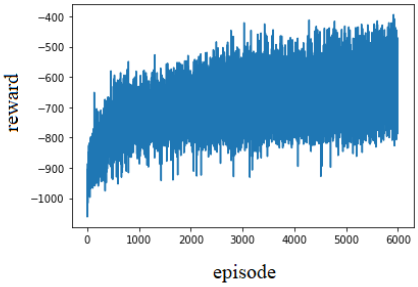

Figure 6. reward per episode while training for Q-learning with softmax policy

5 Discussion:

For Q_Learning we found that it is not stable because it is using a greedy method to choose actions and in this way if the robot find an oil field the robot will try to take all the oil in that field but after this the robot will be stuck near to the field and the robot will not find the other fields.
For SARSA we can clearly see that it was able to give the best results for the lowest number of episodes and all the thanks is for the better exploration because we are not always going to the max value but epsilon greedy and in this way the robot was able to find all the oil fields.
For applying a mix between Q_Learning and SARSA to fix the exploration issue in Q_Learning it worked by running Q_Learning at the first half of the episode and then run SARSA in the next half, but the issue here it took more episodes to train than normal SARSA.
For the last two actions , softmax policy takes the most probable actions. This may not be useful, as we might take the most probable action, for a time step t, but it may not be the one that gives the best return, especially since the terminal state is not a particular location on the grid, but rather it gets updated. for example, most probable, might be cycling back and forth between two coordinates.

6 Conclusion

In this paper we presented various implications, that could help in robot navigation targeted at resource finding. SARSA using ϵ-greedy showed the most promising results compared to other approaches. For the future, we plan to extend this approach to three dimensional, and add more complexity to the environment, like randomly generated obstacles etc. Also , we will explore Policy based methods and model-based methods, and possibly integrate search algorithms like Monte Carlo Search tree etc. Another approach, that can be taken into account, is formulating the problem as a multi objective problem, one where going to oil field and return to drop points can be considered seperately.

References

[1] Mohammad Abdel Kareem Jaradat, Mohammad Al-Rousan, and Lara Quadan. "Reinforcement based mobile robot navigation in dynamic environment". In: *Robotics and Computer-Integrated Manufacturing* 27.1 (2011), pp. 135–149.

[2] Siding Li, Xin Xu, and Lei Zuo. "Dynamic path planning of a mobile robot with improved Q-learning algorithm". In: *2015 IEEE international conference on information and automation*. IEEE. 2015, pp. 409–414.

[3] Xiaogang Ruan et al. "Mobile robot navigation based on deep reinforcement learning". In: *2019 Chinese control and decision conference (CCDC)*. IEEE. 2019, pp. 6174–6178.

[4] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. "Sim-to-real transfer in deep reinforcement learning for robotics: a survey". In: *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE. 2020, pp. 737–744.

[5] Kai Zhu and Tao Zhang. "Deep reinforcement learning based mobile robot navigation: A review". In: *Tsinghua Science and Technology* 26.5 (2021), pp. 674–691.