Cleaning struggles - application of the DDPG algorithm

Filip Morawiec [1] Wojciech Siostrzonek [1] Witold Beluch [2]

[1]III LO im. A. Mickiewicza w Katowicach [2](supervisor) Politechnika Śląska

Introduction

When cleaning a room, or organizing shipments in a warehouse, we can adopt different strategies. A first approach could be to gather all items at some central point. On the other hand, several such points could be established if the layout of the walls does not clearly indicate any particular central location.

The purpose of this work was to decide whether there is a rule that determines the optimal distribution of such points - in the cleaning context adopted here, we will call them mounds. The size of the room is 35x35 grids.

Problem type

At the very beginning, we established the branch of machine learning in which we will work. Reinforcement learning encompasses problems involving two entities - the agent and the environment.

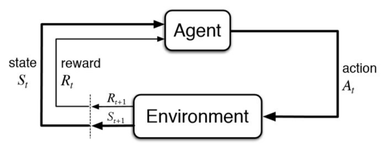

Figure 1. Reinforcement learning scheme

The agent, based on its previous experience and the current state of the environment S_t acts upon it with an action A_t to get feedback on the reward it got R_{t+1} and the following state S_{t+1}. The subscript refers to a moment in time.

Outline of the project

The final program consists of two modules:

- **environment**, which, among other things, returns S_{t+1} and the amount of dirt that fell into the mound R_{t+1} based on the starting and ending coordinates of the broom movement - so $A_t = (x_1, y_1, x_2, y_2)$.
- **cleaning agent**, which learns how to clean optimally by acting in the environment.

The environment was implemented first, followed by cleaning agents in 3 versions corresponding to three different reinforcement learning algorithms.

The project's repository can be seen at The simulation file github.com/gournge/cleaning-optimization. It is worth noting that in the configuration file textttconfig.cfg more important parameters of the simulation, such as the width of the broom or the number of mounds, are set. In addition, more important implementation issues are explained.

Environment

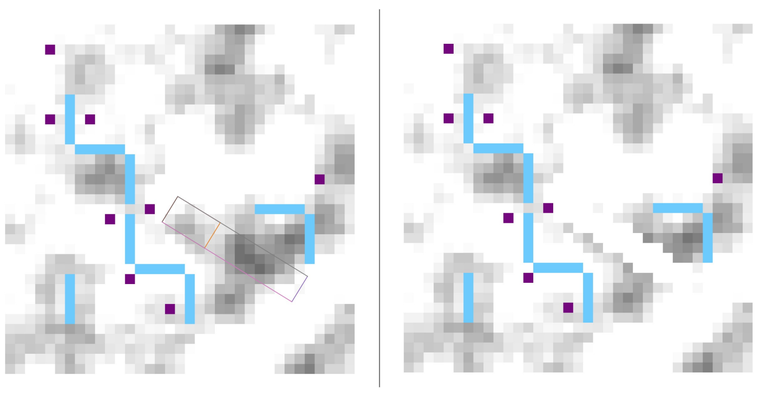

Figure 2. Broom movement without point P. Randomly set purple mound, blue walls.

When the environment receives an action A_t, the following actions are performed:

1. a rectangle that represents the area of the movement of the broom,
2. in the rectangle, we determine the point P where its capacity will be exceeded during the movement of the broom,
3. the dirt is randomly scattered in the rest of the main rectangle:
   1. if P exists, we spread the dirt to the sides and forward,
   2. otherwise - since the dirt has not been spilt - we just move it forward
4. we return how much dirt fell into the mounds and whether the movement of the broom did not interfere with the wall.

Choosing the algorithm

Of the following, only the last one turned out to be correct - this is related to the type of actions returned by the agent. With A2C and DQL, the action A_t is selected from a discrete space (e.g. Atari console buttons), and then DDPG deterministically selects it from a continuous space (e.g. 4 numbers in the interval $(0, 35)$).

Actor-critic (A2C) and Deep Q-learning (DQL)

Auto-critic consists of two main components: the actor and the critic. The actor selects the best actions based on the observed environment, while the critic evaluates these actions and provides feedback on their quality. Deep Q-learning is a deep-neural version of a popular algorithm that is based on finding the best rounding of the Q function. - a function that returns an evaluation of the possible actions A_{t+1}.

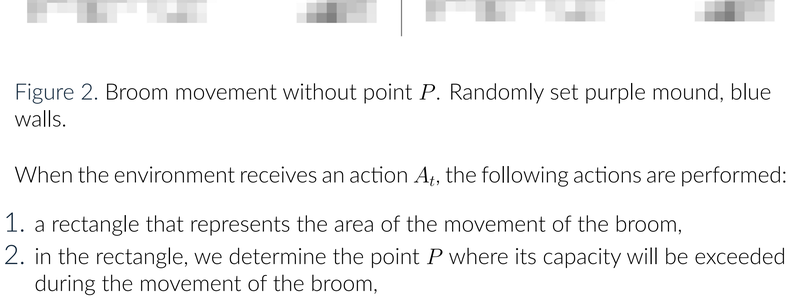

Figure 3. Scheme of A2C and DQL.

Deep Deterministic Policy Gradient (DDPG)

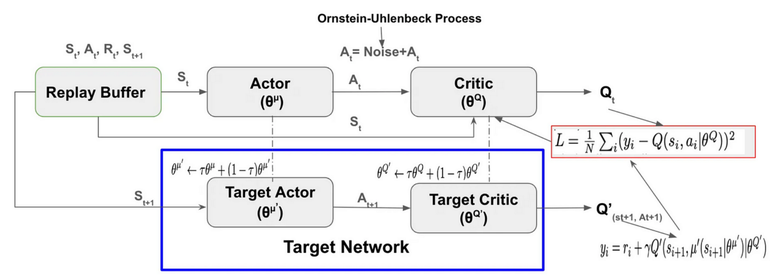

Figure 4. DDPG scheme

- **memory replay**; during training, elements are randomly recorded and selected to vary the process of updating the weights of neural networks,
- **target networks**; separate copies of the actor's and critic's networks - without them, the goal that the networks are aiming at changes with each attempt to bring the networks closer to it,

Network architecture

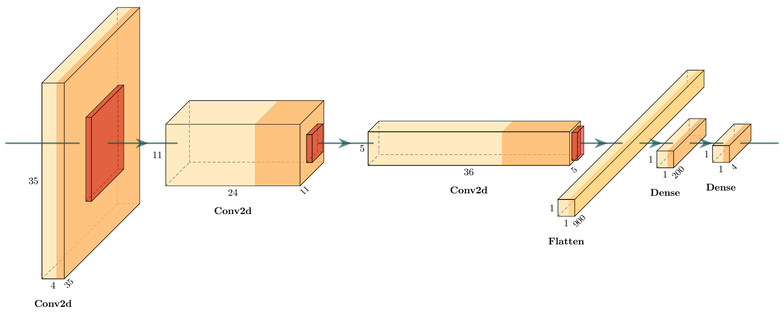

Figure 5. Actor network architecture (the shape is based on [1]). Input size of 3x35x35, the size of the output 4.

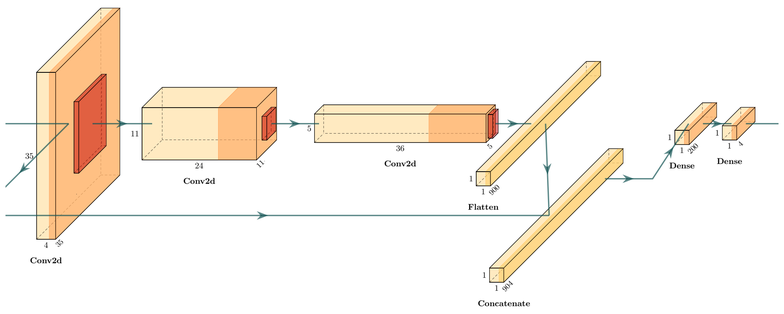

Figure 6. Critic network architecture. Input size of 3x35x35 and 4, the size of the output 4.

The actor makes decisions based on the observation and takes the environment as a tensor with dimensions of 3x35x35. These are the 3 layers of a square room, from which:

- the first is responsible for the placement of the walls,
- the second one is responsible for the placement of the mounds,
- the third one is responsible for the density of the dirt.

The critic takes the action (4 numbers) returned by the agent as an argument (hence the 1x1x904 data size in the penultimate layer). Both types of these networks also return 4 numbers - approximated by their optimal A_t.

It is worth mentioning that the choice of the layer right before the last one as the input for A_t in the critic's network is justified by the difficulty of combining the shape of A_t with the previous convolutional layers.

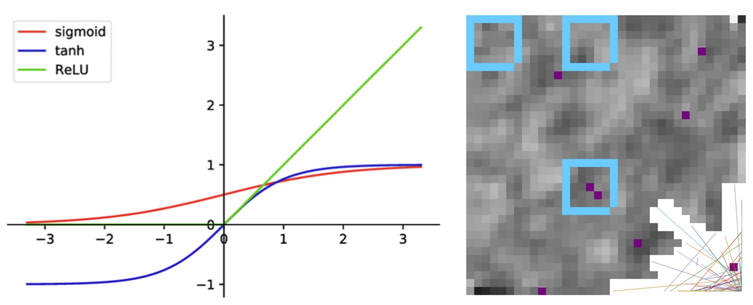

Figure 7. (Left) graphs of the considered activation functions (Right) the moves returned by a trained agent with ReLU function

Suspecting an asymmetrical (therefore potentially favouring the lower right corner) shape of the graph of the ReLU function, experiments were conducted only with the sigmoid and tanh functions. The latter was tested the most times, due to the fact that tanh is an odd function, i.e. somewhat centred in the middle of the room.

First phase of the training

Originally, in order to avoid the habit of hitting the wall with the broom, we programmed that when the broom move was wrong, the environment would return $R_t = K$ <0. Typically, for random moves, the total reward (the total amount of dirt cleaned) was R_t =1.5. Taking $K = -5$, we obtained the following graph:

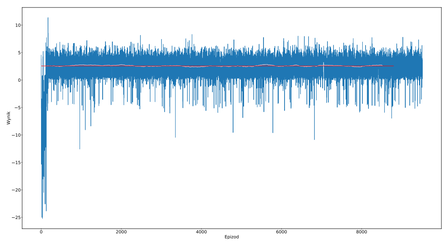

Figure 8. The relation between the results of the cleaning agent and the number of learning episodes (a series of 40 moves in one room)

Anomaly

In an attempt to adjust the parameters of training and the environment, various experiments have been conducted. They are described in the file `experiments.txt`. None of them, unfortunately, led to a sufficiently good level of agent training. Experiment 13 turned out to be important: during training, the variance of the results increased dramatically:

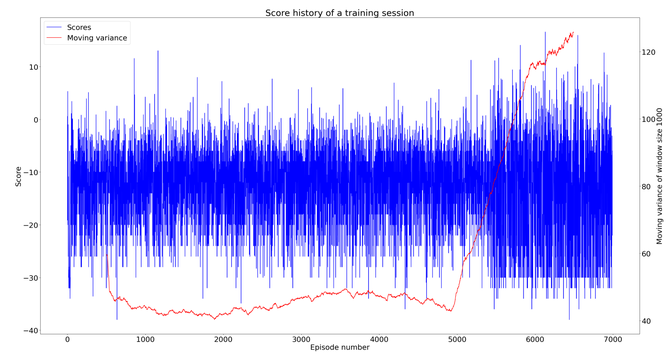

Figure 9. The results of the agent (blue) and its deviation calculated from 1000 closest results (red). Parameters: $K = -2$, `actor-learning-rate = 0.0001`, `replay-memory-size = 1024`, `noise = 0.3`

Visualization of the training

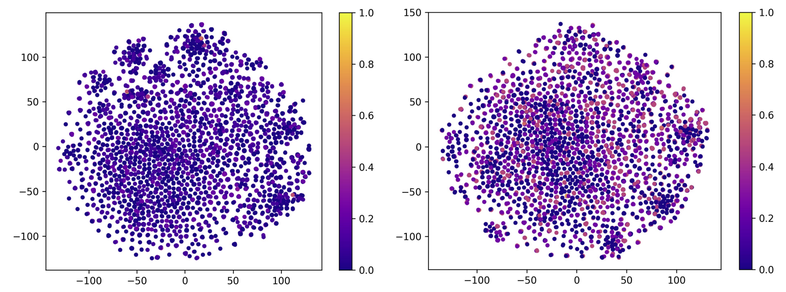

Figure 10. Each point is the environment after passing through the first 4 layers of the network of untrained (left) and trained (right) agent, coloured according to the reward the agent received in this situation

The graphs show only differences in colour variance - the trained agent scores slightly higher. The occurrence of similarly coloured groups of points is due to similar results in cases of environments that deep in its *brain* associates in a similar way. This shows some stability in this algorithm, but it is difficult to draw more complex conclusions because of the *t-SNE* algorithm. [5], which was used to represent multidimensional ($\mathbf{R}^{200}$) data from the penultimate layer on the plane, is to some extent random: regularities in the distribution of points may not be meaningful.

Conclusions

Although the goal of the work - to train optimal cleaning agents and then (potentially) analyze the optimal distribution of mounds - was not achieved, important observations were made that, in addition to pointing to further research directions, provide other insights into the performance of the DDPG algorithm. While it might seem that the observations suggest the inapplicability of DDPG to this particular problem, they are confronted by results from [2], where, after selecting a convolutional network with a suitable shape, it was possible to train a virtual driver effectively making decisions based on a multidimensional image. For this reason, the authors suspect that the chosen network architecture may have been a problem. Instead of attaching the data about the action performed to the first dense layer in the critic's architecture, one might consider attaching it only in the second-to-last layer - it is possible that the information about this action A_t is somehow lost when we combine it with as many as 900 values relating to observations of the environment.

In addition to this, the method of using the t-SNE algorithm to visualize the DDPG's representation of the environment (inspired by [4]) doesn't seem quite right. This may be due to the agent's uneducated nature - if it actually knew how to clean up, the graph would present more information.

Sources

[1] Zhang J. Humaidi A.J. Alzubaidi, L.
Review of deep learning: concepts, cnn architectures, challenges, applications, future directions.
J Big Data 8, 53, 2021.

[2] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A. Al Sallab, Senthil Yogamani, and Patrick Pérez.
Deep reinforcement learning for autonomous driving: A survey.
IEEE Transactions on Intelligent Transportation Systems, 23(6):4909–4926, 2022.

[3] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra.
Continuous control with deep reinforcement learning.
In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2016.

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis.
Human-level control through deep reinforcement learning.
Nature, 518(7540):529–533, February 2015.

[5] Laurens van der Maaten and Geoffrey Hinton.
Viualizing data using t-sne.
Journal of Machine Learning Research, 9:2579–2605, 11 2008.