Reinforcement Learning Assignment 3

Athanasios Agrafiotis
a.agrafiotis@umail.leidenuniv.nl

Ricardo Blikman
ricardo@blikman.com

ABSTRACT
For Assignment 3 we present reinforcement learning approaches that can be used to solve problems in a gaming environment using control policy. We implement a Deep deterministic policy gradient method and a Deep-Q learning approach in an effort to solve these problems as best as we can and compare the results. The performance of the algorithms is based on the reward after each episode. We used OpenAI Gym as an environment for our experiments and run our code on a game known as "Sokoban" and "Highway-env" a driving environment.
KEYWORDS
Sokoban, Highway-road, Deep Deterministic policy gradient, Deep q learning

1. INTRODUCTION

The goal of this assignment is to solve a problem by implementing reinforcement learning techniques using an environment of choice. We have selected a video game called Sokoban and Highway environment. Reinforcement learning approaches have been popular of solving complex problems and navigate through environments such as:

Sokoban (倉庫番) is a puzzle video game that was created in 1981 by Hiroyuki Imabayashi. Sokoban is Japanese for warehouse keeper. The game is played on a board of squares where each square represents a floor or wall. Floor spaces contain boxes, storage locations or are empty. The player must move on the floor tiles and push all the boxes onto storage location squares in order to solve the puzzle and win the game.

Highway-env[1] is a collection of environments for autonomous driving and tactical decision-making tasks. One of these tasks is the ego-vehicle is driving on a multi lane highway populated with other vehicles. The objective is to reach a high speed while avoiding collisions with neighbouring vehicles. Driving on the right side of the road is also rewarded.

The paper is organized as follows. In Section 2 we introduce some previous work of reinforcement learning. In Section 3 We will describe the main research questions for our research, in Section 4 We explain the theory and algorithms, Section will explain our implementation, in Section 5 we will show the results of our experiment, In Section 6 we share our conclusions and finally in Section 7 we discuss possible improvements that can be made.

2. RELATED WORK

Sokoban is not a new problem to solve in the field of reinforcement learning. To implement and experiment with Sokoban, gym-Sokoban[2] is made available as an environment to use that implements the game Sokoban based on the rules presented in the paper: Imagination Augmented Agents for Deep Reinforcement Learning[3]. The room generation is random and therefore, will allow to train Deep Neural Networks without overfitting on a set of predefined rooms.
For the highway environment which has been trained using a reinforcement learning technique. It is an environment which simulates driving and testing which deployed to the OpenAi in the year of 2018. A neural network approach was introduced et al. Tamas Becsi. [4] The training process aims to minimize the loss function using a accumulation of gradients. The reward values have been discounted and normalized per episode.

3. PROBLEM DESCRIPTION

The Sokoban video game is a challenging problem. The main reason is that when one is trying to solve the puzzles there is the possibility of making irreversible mistakes. This is a great challenge for Reinforcement learning algorithms, which mostly lack the ability to think ahead.

In order to solve this problem we will need to answer the following questions:

1. Can we implement a reinforcement learning technique that can efficiently solve Sokoban puzzles within the limited time given for this assignment?
The second environment presented in this paper provides a road traffic scenario. The traffic generation starts with an empty road section. For each lane vehicles are generated. The The agents moves through the environment and the restrictions of the agent is the move between the lanes while avoids to crash other cars. Environment must consists of:

-The car model
-The highway model
-Surrounding vehicles
The reward function increases as the car picks the correct lane choice-maintain speed, avoid collisions.

2. Can we drive the car in the environment in a safe and efficient way within the limited time given for this assignment?

4. METHODS

Deep-Q learning
DQN, also known as Deep-Q network, approximates a state-value function in a Q-Learning framework with a neural network.

Algorithm 1 Experience replay pseudocode

```
function DQN-EXPERIENCE REPLAY()
    s=env.reset()
    while not done do
        a = argmax(Q[s, :])
        s', r, done = env.step(a)
        replay_buffer.add((s, a, r, s', done))
        inputs, targets = sample data from replay_buffer
        params = params - lr * grad((targets - predictions)^2, params)
        s = s'
    end while
```

The DQN selects the action with the highest policy by taking the max Q.

$$Q(s,a) = r + \gamma * max_{a'} Q(s', a') \tag{1}$$

Deep Deterministic Policy Gradient
Deep Deterministic Policy Gradient[5] (DDPG) combines both Q-learning and Policy gradients. It is a model-free off-policy actor-critic algorithm. The actor is a policy network. It takes the state as input and outputs and the the exact action (continuous) unlike a probability distribution over actions.

The critic is a Q-value network that takes in state and action as input and outputs the Q-value. DDPG is an "off"-policy method and it is used in the continuous action setting, it is "deterministic" because the actor computes the action directly it is not a probability distribution over actions.

Q Learning Q learning (quality learning) is a model-free reinforcement learning algorithm. The main objective of Q Learning is to learn the quality of actions, it does so by exploring the actions and store the values a table, called a Q Table. it will then select the action an agent must taken under which circumstance. It is a simple and intuitive This approach.

5. EXPERIMENTS AND RESULTS

DQN We did three experiments with our DQN on the Highway environment. We set the experiment on an Interl(R) Celeron(R) N4000 CPU @ 1.10GHz , 1.10GHz on ram 4.00

Algorithm 2 DDPG Algorithm

function DDPG ALGORITHM
    Randomly initialize critic network Q(s,a| θ^q) and actor m(s|θ^μ) with weights θ^q and θ^μ.
    Initialize target network Q and μ with weights $\theta^{q'} \leftarrow \theta^q$, $\theta^{\mu'} \leftarrow \theta^\mu$
    Initialize replay buffer R
    for episode=1, M **do**
        Initialize a random process N for action exploration
        Receive initial observation states_1
        for $t \in en(s)$ **do**
            Select action $a_t = \mu(st|\theta^\mu)$+ N_t according to the current policy and exploration noise
            Execute action a_t and observe reward r_t and observe new state s_{t+1}
            Store transition (s_t, a_t, r_t, s_{t+1}) in R
            Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
            Set $y_i = r_i + \gamma Q(s_{i+1}, m(s_{i+1}|theta^\mu)theta^q)$
            update critic minimizing the loss:

$$L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i|\theta^Q))^2 \tag{2}$$

            update the actor policy using the sampled policy gradient

$$\bigtriangledown_{\theta^\mu} J \approx \frac{1}{N} \sum_i \bigtriangledown_a Q(s, a|\theta^Q)|s = s_i, a = \mu(s_i) \bigtriangledown_{\theta^\mu} \mu(s|\theta^\mu)|s_i \tag{3}$$

            update the target networks:

            $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
            $\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$

        end for
    end for

GB. We set the learning rate to 0.02 and the epsilon greedy to 0.01 for our first experiment and set the number of episodes to 100. As depicted in figure 1, the average reward was 12.16. We conducted several more experiments and our best result was an average reward of 15.95 over 100 episodes as depicted in figure two, which has an learning rate of 0.02 and e-greedy 0.9.

We did a third experiment running on a Intel i7-10870H with 16 threads, 32 GB RAM and a INVIDIA RTX 3060 GPU. We set the learning rate to 0.03 and e-greedy to 0.5 and set the number of episodes to 1,000. The result was an average reward of 15.55.

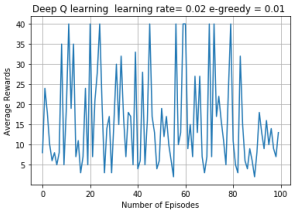

Figure 1: DQN with learning rate=0.02 and e-greedy = 0.01 , Average reward 12.16 of 100 episodes

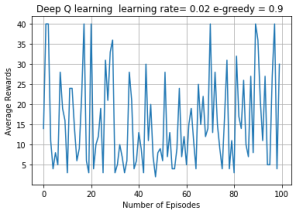

Figure 2: DQN with learning rate=0.2 and e-greedy = 0.9 Average reward 15.95 of 100 episodes

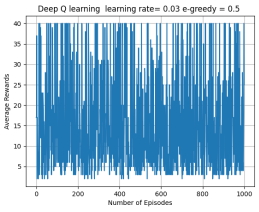

Figure 3: DQN learning rate=0.3 and e-greedy = 0.5, Average reward 15.55 of 100 episodes

When we tried to run our experiment with Sokoban we encountered errors and where not able to fix them.

6. CONCLUSION

We started this assignment trying to answer 2 questions:

1. Can we implement a reinforcement learning technique that can efficiently solve Sokoban puzzles within the limited time given for this assignment? 2. Can we drive the car in the environment in a safe and efficient way within the limited time given for this assignment?

Deep deterministic policy gradient run into some errors of our code and experiment.

As for question two, We manage to build an implementation and we ran our DQN on the Highway environment the initial average reward was low, after conducting several more experiments we managed to get the average reward up to 15.95, which was better than the previous one.

The results of our experiments where not as we expected, It was our goal to implement a reinforcement learning technique that can drive a car in the highway environment efficiently. It does play the game but not as good as we wished for. As you can see from the experiment results the algorithm performance.

7. DISCUSSION

The result of our experiment was not as we expected, yet it is our opinion that better results can be achieved if more time is spend on this project. The topic was study intensive, we ran into some technical difficulties and we have overestimated the complexity of this subject. We could have chosen a different approach or a different game but in the end it was a great learning experience and we had fun doing it.

8. REFERENCES

[1] Edouard Leurent. An environment for autonomous driving decision-making. `https://github.com/eleurent/highway-env`, 2018.

[2] Max-Philipp B. Schrader. gym-sokoban. `https://github.com/mpSchrader/gym-sokoban`, 2018.

[3] Sébastien Racanière, Theophane Weber, David Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adrià Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, Razvan Pascanu, Peter Battaglia, Demis Hassabis, David Silver, and Daan Wierstra. Imagination-augmented agents for deep reinforcement learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[4] Tamás Bécsi, Szilárd Aradi, Árpád Fehér, János Szalay, and Péter Gáspár. Highway environment model for reinforcement learning **the research reported in this paper was supported by the higher education excellence program of the ministry of human capacities in the frame of artificial intelligence research area of budapest university of technology and economics (bme fikpmi/fm).efop-3.6.3-vekop-16-2017-00001: Talent management in au-tonomous vehicle control technologies- the project is supported by the hungarian government and co-financed by the european social fund. *IFAC-PapersOnLine*, 51(22):429–434, 2018. 12th IFAC Symposium on Robot Control SYROCO 2018.

[5] 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.