UNIVERSITY OF PADOVA

Homework 3 Report

Subject: INP9086459 - NEURAL NETWORKS AND DEEP LEARNING 21/22
Professor: ALBERTO TESTOLIN
Student: Menglu Tao N°: 2041389

---

1. Introduction

In this homework, the goal is to implement and test neural network for solving reinforcement learning problems.The fisrt task is to **implement some extensions** to the code seen in the lab7 (**CartPole-v1**) and tune the model a bit to perform **better learning convergence**.

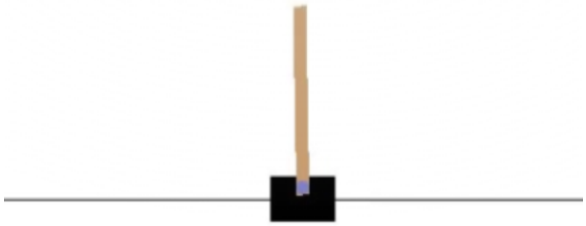
The second task performed in the homework is to **train another deep RL agent** on a different Gym environ- ment, here I chose **MountainCar-v0**, and try to find the hyperparameters and strategy that can make car drive up the mountain on the right, and if the car reaches the right top that means winning, otherwise if after 500 epoches, the car still can't reach the top, means game failure.

2. Improve Code of CartPole-v1

In lab 7 the code is about CartPole-v1, and it is a pole that is attached by an un-actuated joint to a cart, which moves along a frictionless track. The system is controlled by applying a force of +1 or -1 to the cart. The pendulum starts upright, and the goal is to prevent it from falling over. A reward of +1 is provided for every timestep that the pole remains upright. The episode ends when the pole is more than 12 degrees from vertical, or the cart moves more
than 2.4 units from the center.

State space vector:

0: Cart Position ∈[-4.8,4.8]
1: Cart Velocity ∈[-Inf,Inf]
2: Pole Angle ∈[-24°,24°]
3: Pole Angular Velocity ∈[-Inf,Inf]



Actions space vector:

0: Push cart to the left

1: Push cart to the right

Reward: every step is given a reward of +1 is every timestep the pole remains upright.
pos_weight =1, reward = reward - pos_weight * np.abs(state[0])

Initial state: All observations are assigned a uniformly random value in (-0.05, 0.05)

Terminate state:

1. Pole Angle is greater than ±12°
2. Cart Position is greater than ±2.4 (center of the cart reaches the edge of the display)
3. Episode length is greater than 500 (200 for v0)

The network here is using **DeepQNetwork** network , the structure is 3 linear layers and first layer has state_space_dims number of neurons, second and third one has 128 neurons. Both followed by Tanh activation method.

Parameters:

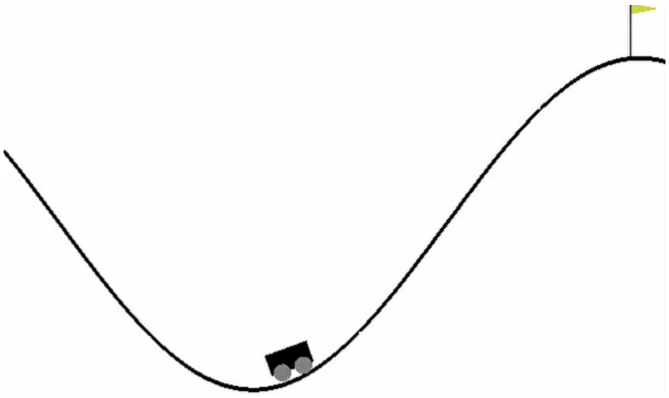
- **gamma**: 0.97
- **replay_memory_capacity**: 10000
- **min_samples_for_training**: 1000
- **lr**: 1e-2
- **target_net_update_steps**: 10
- **batch_size**: 128
- **bad_state_penalty**: 0
- **exploration_profile__initial_temperature_value**: 5
- **exploration_profile__num_iterations**: 1000
- **reward_function**: reward - pos_weight * np.abs(state[0])
-

The whole process is taking around **830 epochs** to reach score 500. Then we tuned the parameters from Lab 7 to make it finish faster. The new parameter combination is as below:

- gamma: 0.97
- replay_memory_capacity: 10000
- min_samples_for_training: 1000
- **lr**: 2e-2
- **target_net_update_steps**: 5
- **batch_size**: 64
- bad_state_penalty: 0
- **exploration_profile__initial_temperature_value**: 7
- **exploration_profile__num_iterations**: 400
- **reward_function**: reward = reward - (np.abs(state[0]) + np.abs(state[2])*2)

Compared with the old parameters, we increased the **learning rate, decreased batch size**, and the greatest change is redefined the **reward function**. In this time we consider the state[0] and state[2] together, which means the reward is highly related to cart position and cart angle. **The further the cartpole is away from the center and the more angle it has, the less reward would get**. Using the tuned parameters, the running time only takes only about **300 epochs** to reach 500, greatly increase the convergence speed.

3. MountainCar-v0



Description : The car starts in between two hills. The goal is for the car to reach the top of the hill on the right. The car does not have enough engine power to reach the top of the hill by just accelerating forward. To win, the car must build momentum by swinging itself backwards and forwards until it has enough speed to reach the flag.(position = 0.5)

State space vector:

0: Cart Position ∈[-1.2, 0.6]

1: Cart Velocity ∈[-0.07,0.07]

Actions space vector:

0: Push to the left

1: No push

2: Push to the right

Reward: every step is given a reward of +1 is every timestep the pole remains upright, pos_weight =1, reward = reward - pos_weight * np.abs(state[0])

Initial state:

- The position of the car is assigned a uniform random value in [-0.6 , -0.4].
- The starting velocity of the car is always assigned to 0.

Terminate state:

The position of the car is greater than or equal to 0.5 (the goal position on top of the right hill)
The length of the episode is 200.

Parameters:

- gamma: 0.97
- replay_memory_capacity: 10000
- min_samples_for_training: 1000
- lr: 2e-2

- target_net_update_steps: 15
- batch_size: 64
- bad_state_penalty: 0
- exploration_profile__initial_temperature_value: 7
- exploration_profile__num_iterations: 500
- Reward_function:lambda reward, state: (state[0] if state[0] < 0 else (state[0] * 20)) + 100 * abs(state[1])

The reward function indicating that **if current position is positive, a high reward will be given, multiplying its value by 20, summing the result of velocity multiplying by 100.** In other words, a high reward will be given for positive position and high velocity values. After finishing 500 iterations we pick the one with highest score which will be the best one. And the best one is episode 409 with a highest score of 779.36 which means it takes the shortest time to reach the goal state..

4. Reference

- Cart pole <https://github.com/openai/gym/blob/master/gym/envs/classic_control/cartpole.py >
- Mountain Car <https://www.gymlibrary.ml/environments/classic_control/mountain_car >