Assignment 3: Playing Pong with Deep Q-Network

ECS 271 Machine Learning

April 10, 2022

1 Environment

If you want a challenge, rather than train to learn Pong, you can learn to play Pacman. Just change the starter code so the environment id (PongNoSkipFramev4 to MsPacManNoSkipFramv4). Note, our pre-trained/seeded model is only for Pong and only work on Pacman if you have time.

In this assignment, you will utilize deep Q-Learning to play the Atari game: `PongNoFrameskip-v4` from OpenAI gym. You should have read the "A Brief Survey of Deep Reinforcement Learning" article on Canvas as well as the Mitchell chapter on reinforcement learning.

The typical RL and DRL covered in class and given in the starter code is insufficient to learn Pong so you will explore variations. In Part 1, you need to answer some questions and implement some key additions for DQN. In Part 2, you will visualize and analyze the trained DQN in Part 1. The starter code has been provided (based on `PyTorch` and `gym` environments).

If you have available GPU resources, you may work locally. Alternatively, you can use the Google Cloud platform (refer to the distributed Google Cloud Tutorial). If you are using Google Cloud, here are some suggestions to avoid some pitfalls:

a. When launching Deep Learning VM, I suggest you to choose the `TensorFlow 1.13` framework instead of `PyTorch 1.1` because `gym` is not installed by default in the latter one.

b. After launching the `TensorFlow 1.13` virtual machine, open its command line and do the following installation:

```
sudo apt install cmake libz-dev
pip install torch torchvision
pip isntall gym[atari]
```

c. I suggest you to run the codes in `screen` so that you don't have to keep the command line window open all the same. If you are not familiar with this, you might refer to this site.

2 Part 1: Extend the Deep Q-learner (60 points)

2. (**coding**) Implement the "randomly sampling" function of replay memory/buffer (see line 89 of `dqn.py` for TODO and hints). Submit necessary code as sampling.zip.

3. (**written**) Is randomly sampling strategy good? Should we treat each frame in the buffer equally?

4. (**coding**) Given a state, write code to calculate the Q value and the corresponding chosen action based on neural network (see lines 50 $\sim$ 55 in `dqn.py`). Submit necessary code as action.zip.

5. (**written**) Given a state, what is the goal of line 48 and line 57 of `dqn.py`? Aren't we calculating the Q value and the corresponding chosen state from the neural network?

```
if random.random() > epsilon:
    ...
else:
    action = random.randrange(self.env.action_space.n)
```

6. (**written and coding**) Implement the appropriate objective function of DQN (see lines 69 $\sim$ 73 of `dqn.py`), which is described in the Mitchell Q-learning text on the website. Write which update function did you use and why?

   Submit necessary code as TD.zip.

7. (**written**) After you make these changes, train DQN by running `run_dqn_pong.py`. To achieve relatively good performance, you need to train more than 1000000 frames. It takes $\sim$ 10 hours on a Google Cloud Virtual Machine with 2 vCPUs and 1 NVIDIA Tesla K80 GPU.

   Explain what parameter tuning you performed such as:

   a. You might tune the hyper-parameters like γ and initial size and the size of replay buffer to gain a better performance.

   b. Modify `run_dqn_pong.py` to track and plot how loss and reward changes during the training process. Attach these figures in your report. Your grade will be based on the ranking of final reward.

3 Part 2: Improving Convergence (40 points)

Our starter code with your additions was able to learn to play the game, but at a slower than ideal convergence. In this part of the assignment you will try to aim for faster convergence.
Part a) of this question is to conjecture (and back up with experimental results from the first part) a reason for slow convergence. Part b) of this question is to implement a method to speed up convergence and verify it indeed does improve results over part 1.
Examples (but not limited) of ideas can be:

- Changing the DL architecture
- Changing the Exploration vs Exploitation Strategy
- Seeding the Q-learner in some principled manner.

Be sure to cite any papers you use.

4 Requirements

- Report.pdf - pdf report for the above questions marked 'written'; no handwritten responses.
- model.pth - your best performing model saved to a pth file. This does not have to be your most trained model, but is instead your best performing model. Make sure test dqn_pong.py can load your saved model.
- run_dqn_pong.py - with your relevant changes.
- dqn.py - with your relevant changes.
- any additional programming files you used for bonus points questions.