11-695: Competitive Engineerng
Assignment 3: Pong with Deep Reinforcement Learning

Spring 2018

Abstract

In this assignment, you will implement a reinforcement learning algorithm that plays the game of Pong. You will use the environment from Arcarde Learning Environment from OpenAI gym [Machado et al., 2017]. This environment allows you to simulate trajectories from the game by sampling an action given any state of the game. Your job is to implement a reinforcement learning algorithm using TensorFlow. The algorithm should take as input the current screen frame of the game and return a distribution over the possible actions, based on which you can sample the action.

This assignment is due April 27th, at 11:59pm. Submission via email.

1 Starter Code

Install TensorFlow. If you have not already, please navigate to TensorFlow's site and follow their instructions to install the framework. The instructions are at

https://www.tensorflow.org/install/

Their instructions should be sufficient to install TensorFlow and all of its dependencies on your system. It is possible to complete this assignment without using any GPU, so you do not need to install CUDA or anything related to GPU programming. However, if you wish to, you are more than encouraged to install TensorFlow GPU, which will make your implementation much faster.

Install OpenAI Gym. You will be working with the game environments from OpenAI Gym [Machado et al., 2017], which is a popular collection of benchmarks to develop reinforcement learning algorithms. Installing OpenAI Gym is as simple as the following command:

```
$ sudo pip install gym
```

More detailed instructions on OpenAI gym can be found at

https://gym.openai.com/docs/

Starter code. The starter code for your project is available at

http://www.cs.cmu.edu/afs/cs/user/hieup/www/11695/starter_code_3.zip

Unlike the previous assignments, there will be no data. Instead, your "data" is the OpenAI Gym. We have implemented the REINFORCE algorithm [Williams, 1992] with a moving average baseline to play the game `Cart-Pole`. You can invoke a training session of this algorithm by:

```
$ ./scripts/cart_pole.sh
```

Both the algorithm and the game are very simple, so you can inspect this code as an example on how to interact with the game.

Using OpenAI Gym. The first step in working with games from OpenAI Gym is to understand how to interact with the game environments in Python. The starter code in `src/cart_pole.py` demonstrates how to create an *environment, i.e.* the game, and how to interact with the environment to sample game trajectories and to learn from them.

2 Your Work and Gradings

If you run the starter code using `./scripts/cart_pole.sh`, you will see the following output:

```
eps=50   loss=513.41     |g|=70.59      len=31  bl=7.44
eps=100  loss=761.43     |g|=170.17     len=40  bl=13.16
eps=150  loss=-20.78     |g|=2.63       len=15  bl=17.01
eps=200  loss=-27.36     |g|=2.25       len=16  bl=18.47
eps=250  loss=1.51       |g|=0.36       len=21  bl=20.90
eps=300  loss=70.35      |g|=10.95      len=27  bl=23.26
eps=350  loss=339.54     |g|=32.78      len=45  bl=33.23
eps=400  loss=-300.09    |g|=138.37     len=22  bl=40.44
eps=450  loss=-195.27    |g|=9.43       len=31  bl=40.95
eps=500  loss=82.14      |g|=9.26       len=48  bl=45.23
eps=550  loss=-396.76    |g|=31.58      len=28  bl=51.39
eps=600  loss=-494.79    |g|=60.75      len=37  bl=56.90
eps=650  loss=-500.48    |g|=87.53      len=38  bl=60.58
eps=700  loss=-588.82    |g|=231.85     len=18  bl=63.49
eps=750  loss=-545.81    |g|=51.46      len=59  bl=73.73
eps=800  loss=530.76     |g|=78.26      len=92  bl=83.05
eps=850  loss=-275.47    |g|=46.32      len=88  bl=93.00
eps=900  loss=12157.97   |g|=696.96     len=200 bl=98.66
eps=950  loss=9534.32    |g|=1129.25    len=200 bl=115.28
eps=1000 loss=4064.63    |g|=2088.58    len=171 bl=131.39
eps=1050 loss=5080.41    |g|=415.32     len=186 bl=141.54
eps=1100 loss=-469.72    |g|=91.83      len=145 bl=150.33
eps=1150 loss=4401.13    |g|=690.02     len=200 bl=161.79
eps=1200 loss=-3869.12   |g|=1179.75    len=113 bl=166.80
eps=1250 loss=3909.06    |g|=485.27     len=200 bl=168.69
eps=1300 loss=2634.79    |g|=183.58     len=200 bl=176.28
eps=1350 loss=2386.50    |g|=180.02     len=200 bl=178.57
eps=1400 loss=1938.32    |g|=236.84     len=200 bl=181.70
eps=1450 loss=1876.57    |g|=227.59     len=200 bl=183.56
eps=1500 loss=1184.29    |g|=82.08      len=200 bl=188.84
```

The purpose of the game Cart-Pole is to be able to stay in the game for as many steps as possible. Therefore, when we print out the logs on how the model learns to play Cart-Pole, we are concerned with `len`, which is the number of steps the agent can keep playing. In the beginning, the game could only last for 11 steps. However, after 1500 training episodes, the agent can consistently keep the game running for 200 steps. We say that a game is solved when the agent can consistently achieve a certain performance.

When you implement your algorithm for Pong, which involves controlling the controller's pad to move up or down to play a tennis-similar game, we consider the game solve when your agent can consistently beat the pre-written AI by 20 points.

You can use any algorithm that you prefer, and you can train for as many time steps as you want. The only requirement is that your code should produce the output similar to the log above. We will compute the total rewards *in the last 10 outputs* of your code and average the score. Your score for

the assignment is the percentage of your score compared to the requirement of 20 points. For example, if your average score is 18.1, then your grade is 90.5%.

Academic Integrity. As normal, you are encouraged to discuss with your friends and the instructors. Anything they tell you, you can use. However, looking at other's code should not happen at all cost.

References

Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew J. Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Arxiv, 1709.06009*, 2017.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 1992.