CS294-112 Deep Reinforcement Learning HW3: Q-Learning on Atari
due March 8th, 11:59 pm

1 Introduction

This assignment requires you to implement and evaluate Q-Learning with convolutional neural networks for playing Atari games. The Q-learning algorithm was covered in lecture, and you will be provided with starter code. You may modify the code to use any automatic differentiation package you want, though the default code uses TensorFlow, and you may find it easier to use that. You may run the code either on GPU or CPU. A GPU machine will be faster, but you should be able to get good results with about 20 hours of compute on a modern CPU.

Please start early! The questions will require you to perform multiple runs of Q-learning, each of which can take quite a long time. Furthermore, depending on your implementation, you may find it necessary to tweak some of the parameters, such as learning rates or exploration schedules, which can also be very time consuming. The actual coding for this assignment will involve about 50 lines of code, but the evaluation may take a very long time.

2 Installation

Obtain the code from: `https://github.com/berkeleydeeprlcourse/homework`
To run the code, go into the hw3 directory and simply execute:

```
python run_dqn_atari.py
```

It will not work however until you finish implementing the algorithm in dqn.py
You will also need to install the dependencies, which are OpenAI gym, TensorFlow, and OpenCV (which is used to resize the images). Remember to also follow the instructions for installing the Atari environments for OpenAI gym, which can be found on the OpenAI gym github page. To install OpenCV, run "pip install opencv-python"
There are also some slight differences between different versions of TensorFlow in regard to initialization. If you get an error inside dqn_utils.py related to variable initialization, check the comment inside initialize_interdependent_variables,

it explains how to modify the code to be compatible with older versions of TensorFlow.

3 Implementation

The first phase of the assignment is to implement a working version of Q-learning. The default code will run the Pong game with reasonable hyperparameter settings. The starter code already provides you with a working replay buffer, all you have to do is fill in parts of dqn.py, by searching for "YOUR CODE HERE." The comments in the code describe what should be implemented in each section. You may find it useful to look inside dqn_utils.py to understand how the replay buffer works, but you should not need to modify it. You may also look inside run_dqn_atari.py to change the hyperparameters or the particular choice of Atari game. Once you implement Q-learning, answering some of the questions may require changing hyperparameters, neural network architectures, and the game, which should be done by editing run_dqn_atari.py

To determine if your implementation of Q-learning is performing well, you should run it with the default hyperparameters on the Pong game. Our reference solution gets a reward of around -20 to -15 after 500k steps, -15 to -10 after 1m steps, -10 to -5 after 1.5m steps, and around +10 after 2m steps on Pong. The maximum score of around +20 is reached after about 4-5m steps. However, there is considerable variation between runs.

To accelerate debugging, you may also check out run_dqn_ram.py, which runs the game Pong but using the state of the emulator RAM instead of images as observations. This version will run faster, especially if you're not using a GPU, but takes more iterations and probably won't converge to as good of a solution. You may use this version for debugging your algorithm, though you are not required to report results for it. Our reference solutions with the default hyperparameters gets around -19.9 after 500k steps, -19.2 after 1m steps, -15.0 after 1.5m steps, and -13.2 after 2m steps.

4 Evaluation

Once you have a working implementation of Q-learning, you should prepare a report. The report should consist of one figure for each question below, similarly to homework 1. You should turn in the report as one pdf and a zip file with your code. If your code requires special instructions or dependencies to run, please include these in a file called README inside the zip file. For all the questions below, it is your choice how long to run for. Although running for 2-4m steps is ideal for a solid evaluation, especially when running on CPU, this may be difficult. We strongly recommend running at least 1m steps, and including at least one run of 4m steps for Question 1. If you have severe computational constraints and find that you are unable to run image-based Atari fast enough for evaluating the design choices in Q2, you may use the RAM version for Q2

only, but Q1 results *must* use images. If you use RAM state for Q2, please specify this in the caption.

Question 1: basic Q-learning performance. Include a learning curve plot showing the performance of your implementation on the game Pong. The x-axis should correspond to number of time steps (consider using scientific notation) and the y-axis should show the mean 100-episode reward as well as the best mean reward. These quantities are already computed and printed in the starter code. Be sure to label the y-axis, since we need to verify that your implementation achieves similar reward as ours. If you needed to modify the default hyperparameters to obtain good performance, include the hyperparameters in the caption. You only need to list hyperparameters that were modified from the defaults.

Question 2: experimenting with hyperparameters. Now let's analyze the sensitivity of Q-learning to hyperparameters. Choose one hyperparameter of your choice and run at least three other settings of this hyperparameter, in addition to the one used in Question 1, and plot all four values on the same graph. Your choice what you experiment with, but you should explain why you chose this hyperparameter in the caption. Examples include: learning rates, neural network architecture, exploration schedule or exploration rule (e.g. you may implement an alternative to ϵ-greedy), etc. Discuss the effect of this hyperparameter on performance in the caption. You should find a hyperparameter that makes a nontrivial difference on performance. Note: you might consider performing a hyperparameter sweep for getting good results in Question 1, in which case it's fine to just include the results of this sweep for Question 2 as well, while plotting only the best hyperparameter setting in Question 1.

Note: for Question 2, you may run on other games or multiple games, but please submit results for Question 1 using only the game Pong. Running on multiple games may require considerable time or computing power, so it is not required, but encouraged. If you have any other interesting experiments you wish to report on, you may include those after your answer to Question 2. Interesting additional experiments or extensions will be considered for bonus points.