Assignment 6: Final Programming Project

Advanced Machine Learning Proseminar, Winter Semester 2019-20

Total points: 50
Deadline: 18.02.2020

Sayantan Auddy
sayantan.auddy@uibk.ac.at

1 Introduction

Your task is to implement a policy gradient algorithm for solving the gym lunar lander environment. You will need to use the continuous version of this environment which is named `LunarLanderContinuous-v2`. The objective is to control the two thrusters of the lander (left-right, up-down) in such a way that the lander is able to land in the designated landing zone (between the flags in Fig. 1).

Figure 1: Lunar Lander environment

For using the lunar lander environment, you may first need to install the gym box2d environments as shown below. Using a Python virtual environment[1] is recommended. For more information about how to use gym environments, refer to the code in the course git repository or take a look at `https://gym.openai.com/docs/`.

```
pip install gym[box2d]
```

2 Environment Details

This section briefly describes the environment (observations, actions and rewards). For details, please consult the source code of the environment at `https://github.com/openai/gym/blob/master/gym/envs/box2d/lunar_lander.py` or refer to `https://gym.openai.com/envs/LunarLanderContinuous-v2/`.

1. **Observations**: At each step, the environment will emit an observation which consists of an array of 8 numbers. These numbers are usually in the range -1...+1 but some times bigger and smaller values can occur.

2. **Actions**: The action space is two-dimensional (action should consist of 2 floats). The first action is used to control the main engine and is responsible for the up-down motion of the lander. The range of values for this action and their corresponding meanings are: -1..0 off, 0..+1 throttle from 50% to 100% power. The engine can't work with less than 50% power. The second action controls the left-right motion of the lander. The range for this is: -1.0..-0.5 fire left engine, -0.5..0.5 off, +0.5..+1.0 fire right engine.

3. **Rewards**: Reward for moving from the top of the screen to landing pad with zero speed is about 100..140 points. An episode finishes if the lander crashes or comes to rest, receiving additional -100 or +100 points. Each leg ground contact is +10. Firing main engine is -0.3 points in each frame. The agent receives 200 points if the task is solved.

Running the code segment below, will give you information about the environment, particularly about the possible actions and the observations.

[1] `https://docs.python.org/3/tutorial/venv.html`

```
import gym

# Initialize the environment
# Check the env source at
    https://github.com/openai/gym/blob/master/gym/envs/box2d/lunar_lander.py
env_name = 'LunarLanderContinuous-v2'
env = gym.make(env_name)

# Information about observations
# Observation is an array of 8 numbers
# These numbers are usually in the range of -1 .. +1, but spikes can be higher
print('Shape of observations: {}'.format(env.observation_space.shape))
print('A few sample observations:')
for i in range(5):
   print(env.observation_space.sample())


# Information about actions
# Action is two floats [main engine, left-right engines].
# Main engine: -1..0 off, 0..+1 throttle from 50% to 100% power. Engine can't work with
    less than 50% power.
# Left-right: -1.0..-0.5 fire left engine, +0.5..+1.0 fire right engine, -0.5..0.5 off
print(env.action_space.shape)
print('A few sample actions:')
for i in range(5):
   print(env.action_space.sample())
```

3 Tasks

You will solve the task using policy gradients in 2 ways:

1. Using a linear function approximator as your policy function. For this, you will need to convert the observation of the state $\mathbf{s} \in \mathbb{R}^8$ into a polynomial feature vector $\Phi \in \mathbb{R}^N$ of degree n, and then combine the feature vector linearly with weights to produce the action $\mathbf{a} \in \mathbb{R}^2$. Since each of the actions should be in the range $-1 \cdots +1$, you will need to use the tanh function to compute the final action $\mathbf{a} = [a_0, a_1]$ as shown in Fig. 2. You will need to choose an appropriate value for the polynomial degree n.

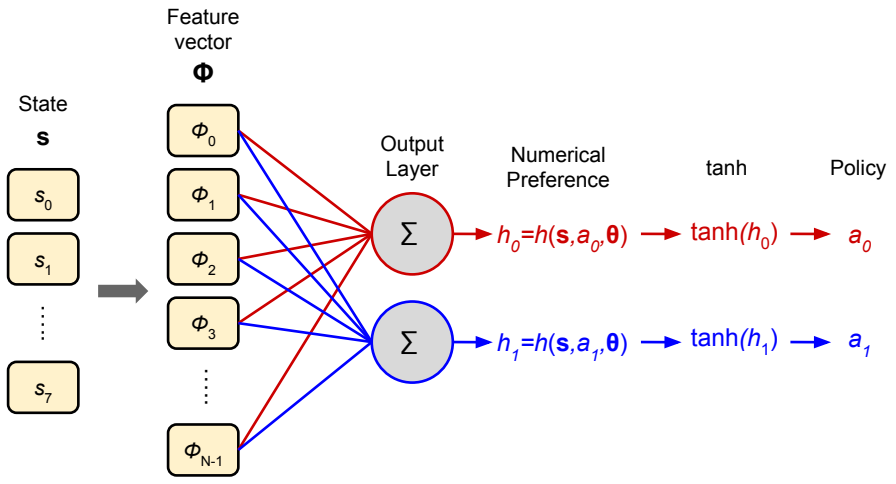

Figure 2: Policy function using linear function approximation

2. Using a neural network as your policy function. For this, you will use a fully connected network which takes the observation $\mathbf{s} \in \mathbb{R}^8$ as input and produces the action $\mathbf{a} \in \mathbb{R}^2$. You should use tanh as the activation function for each of the output units because each action should be in the

range $-1 \cdots +1$. The choice of how many hidden layers to use, and how many units to put in each hidden layer is up to you.

4 Algorithm

For both the tasks, use the REINFORCE algorithm (discussed in class) for training your agents. You can use any deep learning framework of your choice for the second task.

5 Baseline

The code for implementing an agent which always takes random actions is provided in the file `random_agent.py`. You will need to compare the performance of your agents with that of the random agent.

6 Division of Points

There are 50 points in total, out of which 30 points are for the report and 20 points for the code.

6.1 What to Include in the Report [30 points]

1. For task 1, what is the polynomial degree n that you used and why? What would be the effect of choosing a very high or a very low value for n? [1 point]

2. Train the agent 3 times for task 1 (using the same hyperparameters and number of episodes, but with different random seeds) and plot the performance of the agent (rewards in the episode) for each episode. Each time, also compare this performance with the performance of the agent taking random actions. Thus, you will have 3 figures containing 2 line plots each (one for the linear FA policy and one for the random policy). Include the random seed in the caption of the respective figures. Provide proper labels and legends. [9 points]

3. Analyze the performance of the agent for task 1. State the reasons you feel are responsible for its success or failure. Which are the hyperparameters that have the highest impact on the agent's performance and why? [3 points]

4. For task 2, describe the structure of your policy neural network, and list all the hyperparameters that you have used and their values. Provide reasons for your design choice. [4 points]

5. Train the agent 3 times for task 2 (using the same hyperparameters, number of episodes, and network structure but with different random seeds) and plot the performance of the agent (rewards in the episode) for each episode. Each time also compare this performance with the performance of the agent taking random actions. Thus, you will have 3 figures containing 2 line plots each (one for the neural network policy and one for the random policy). Include the random seed in the caption of the respective figures. Provide proper labels and legends. [9 points]

6. Analyze the performance of the agent for task 2. State the reasons you feel are responsible for its success or failure. How can the performance be improved? [4 points]

6.2 Code [20 points]

Please submit Python code in `.py` files. Your code will be evaluated on its correctness, clarity and readability. Specifically:

1. Include a file named `README.txt` with the names of the authors, dependencies (if any) and instructions for running your code.

2. Comment your code properly so that it is understandable.

3. Provide headers for each function which describe the function's task and its arguments and outputs.

> **Note**: Points are awarded for the way you write your report, the soundness of the reasons behind your decisions, the clarity of your code, and most importantly for the way you analyze your results. If at the end your agent cannot solve the task completely, then do not worry. This will not impact your grades if your work is complete and shows that you have tried different **meaningful** options for solving the task.

7 Submission Instructions

Treat this project as assignment number 6 and make sure that you follow the folder structure shown in Fig. 3. Failure to follow these instructions will lead to bad karma (and possibly some deduction of points)!

1. Maximum number of people per team = 2.
2. The submission should consist of a PDF report (with your c-number in the title) and sub-folders with your code, as shown in Fig. 3.
3. The PDF report **must** contain the names and matriculation numbers of the two team members.
4. Only include the answers to the questions in Sec. 6.1 in your report and use the same numbering for your answers.
5. Write concise answers.
6. Make sure that all your plots are labeled and have captions and legends.
7. Your code must be accompanied by a readme file.
8. Submit in the same way you have submitted previous assignments.

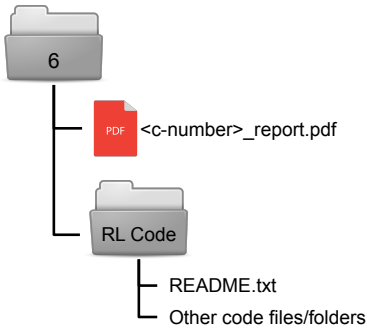

Figure 3: Organization

8 Template for Report

Your PDF report should look like below (template is provided in `report_template.tex`).

Assignment 6: Final Programming Project

Advanced Machine Learning Proseminar, Winter Semester 2019-20

Name/C-number:
Name/C-number:

1 Answers

1. Answer to question 1
2. Answer to question 2 (include 3 figures)
3. Answer to question 3
4. Answer to question 4
5. Answer to question 5 (include 3 figures)
6. Answer to question 6

Figure 4: Template for PDF report.