Reinforcement Learning
Set-up and Submission Instructions

September 3, 2020

1 Introduction

These instructions contain **guidelines** that **must be followed on the submissions** and **hints for submitting high quality exercise reports**. Furthermore, we include instructions to set up software and libraries that will be used throughout the course.

2 Setting things up

In the exercises, we will be using Python 3 along with some libraries: OpenAI Gym, NumPy and PyTorch. Let's start off by setting everything up.

OpenAI Gym is officially supported only on Linux and MacOS/OS X. If you don't have any of them on your machine, you can either set up a virtual machine (using VirtualBox or VMWare) or use one of the Aalto computers (for example in the Panikki room in CS building; they are also available for remote connections via SSH).

We received some reports of people successfully installing OpenAI Gym on Windows - for some of them it worked flawlessly, and some suffered from minor glitches (but were still able to complete the exercises). The general instructions for installing on Windows are also presented, but keep in mind that this procedure is experimental.

2.1 Python 3

On virtually all Linux systems, Python 3 is either installed by default or can be fetched directly from the official repositories. You can check if Python 3 is installed (together with its exact version) by running `python3 --version`

If it's not installed, the exact commands needed to set it up depend on your Linux distro:

Aalto University
School of Electrical
Engineering

Reinforcement Learning course staff
Intelligent Robotics Group
aalto.fi, irobotics.aalto.fi

1. Debian, Ubuntu and their derivatives:
   Open the terminal and run
   ```
   sudo apt-get install python3 python3-pip
   ```
2. Arch (and derivatives)
   Open the terminal and run
   ```
   sudo pacman -S python3
   ```

On Windows, Python can be downloaded from `https://www.python.org/downloads/`

2.2 OpenAI Gym, NumPy, Matplotlib...

If you've installed pip, installing OpenAI gym can be installed with the following command:
```
sudo pip3 install gym numpy matplotlib seaborn pandas
```

If you don't have root access on your machine (for example on university computers) or if you're using a distro that insists on installing pip packages locally on your user account (such as Gentoo), you can run pip as follows:
```
pip3 install --user gym numpy matplotlib seaborn pandas
```

On Windows, the command should be: `pip3 install gym numpy matplotlib seaborn pandas`

For more details, see `https://gym.openai.com/docs/`, `https://www.scipy.org/scipylib/download.html` and `https://matplotlib.org/users/installing.html`.

2.3 PyTorch

Installing PyTorch on Linux should narrow down to running
`sudo pip3 install torchvision` or `pip3 install --user torchvision`.

On Windows, you can try the following:
```
pip3 install torch===1.2.0 torchvision===0.4.0 -f https://download.pytorch.org/whl/torch_stable.html
```

For installation instructions on other platforms (or with tools like anaconda), refer to `https://pytorch.org`

2.4 Working remotely

Aalto is providing computers in the CS and Maari buildings with GPUs where you can develop and run your code or train models if your computer is not capable of doing so within a reasonable time. We recommend that you use these possibilities to reduce the time you are waiting for models to train and especially for the project work.

To set things up, you should follow the standard Linux procedures (bare in mind that — as a student — you don't have `sudo` rights on those machines, so pip packages have to be installed with the `--user` switch).

You can later work remotely by using SSH to connect to these machines. First, you should open a SSH connection to `kosh.aalto.fi` or `lyta.aalto.fi`, log in using your Aalto username and password, and — from Kosh/-Lyta — connect to one of the classroom computers. The full list is available at https://www.aalto.fi/en/services/linux-computer-names-in-it-classrooms.

2.4.1 Connecting from Windows

To use SSH on Windows, the easiest way is to download and install PuTTY. If you want to observe the progress of your training, you will additionally have to install an X server for Windows (Xming is a good option) and enable X11 forwarding when connecting from PuTTY.

2.4.2 Connecting from Linux/Mac

On Linux and Mac, connecting basically narrows down to invoking `ssh username@kosh.aalto.fi` or `ssh username@lyta.aalto.fi`, entering your Aalto password, and from there connecting to any of the lab machines by running `ssh machine_name`.

If you wish to visually follow the training process, you should pass the `-X` flag to the `ssh` command. For example, if you're logging in to `kuukivi`, you would first run `ssh -X user@kosh.aalto.fi` followed by `ssh -X kuukivi`. Sometimes you may have to use `-Y` flag instead of `-X` (depends on your system configuration).

You can make the process easier by appropriately setting your SSH configuration in `/.ssh/config`:

```
Host kosh
        HostName kosh.aalto.fi
        User aaltousername
        ForwardX11 yes
        ForwardX11Trusted yes
```

Then, add similar definitions for the machines you want to use (example for `bit`):

```
Host bit
        HostName bit
        User aaltousername
        ProxyCommand ssh -q -A kosh nc -q0 %h %p
        ForwardX11 yes
        ForwardX11Trusted yes
```

After adding these setting, you can login to `bit` simply with `ssh bit`.

You can also copy your own machine's RSA identification, such that you don't have to provide your password every time you log in with `ssh-copy-id kosh` (**don't do it on a public computers**). For security reasons, this requires appropriate permissions on your SSH-related files and directories to work (e.g., your home directory and the `authorized_keys` file cannot be writable by other users).

On Linux, you may also have to enable indirect GLX rendering in your X server configuration; this can be done by adding the following to the `/etc/X11/xorg.conf` file on your local machine (create the file if it doesn't exist):

```
Section "ServerFlags"
        Option "AllowIndirectGLX" "on"
        Option "IndirectGLX" "on"
EndSection
```

You will have to restart your X server (or reboot) afterwards.

3 Exercise Submissions

Exercises **must be done individually**. We will use TurnItIn to verify this. You can work with peers on the solutions but **the solution you provide must be your own work**. It is advised to attend to the exercise sessions as TAs will assist you on the solutions.

The answers to the exercises are given on the first exercise session. This session will be recorded so that it is available to everyone that cannot attend to the first session.

A good exercise submission report should meet the following criterias:

- The latex template is used
- Sections in the latex report match the tasks/questions in the instructions
- Plots have axis labels and titles
- Plots are included in the report and as files in the submission box
- Submission are not in a compressed zip file, including only the required files

We will subtract one point per criteria that is not met, up to 5 points total for each exercise.

3.1 Late submissions

We do not accept any submissions or additional files after the submission system in MyCourses closes. he only exceptions are in well justified cases such as illness (supported by a proper certificate) or military service. If you cannot submit the assignment on time due to university-related reasons, such as attending a conference, please inform the course staff in advance.

3.2 Latex Template

The submissions must be written using the latex template we provide. Refer to the latex template for examples on how to structure the report and include code pieces or formulas.

3.3 Plots

Plots must to be included in the report and as files in the submission box.

3.4 How to answer questions

Tasks: require you to complete a programming assignment. The relevant lines you changed or filled in the provided code must be in the report. Plots generated during the training of a task must be in the report. No argumentation is required in the programming tasks.

Questions: are divided into:

- Yes/no questions, e.g. Can the provided model be trained without further modification?
- Discussing questions, requires you to briefly explain or justify your answer, e.g. why/why not?

It is usually marked which kind of answer we expect from a question. The questions that require you to justify your answer must be properly argued. Please avoid answers that do not provide any reasoning. Some examples of answers that are poorly justified and why they are poorly justified are:

- Why does the reinforcement learning algorithm perform better when adding the parameter γ? Because it takes less time than before. - This answer does not provide any reasoning about how modifying γ affects the learning algorithm.
- How does the reward change over multiple runs of the algorithm? The reward is the same at the end of the training. - This answer might be obvious. What is interesting about how it changed, what could be the reason behind it?