Reinforcement Learning (EL2805)

Computer Lab 2 – Deep Q Learning

December 1, 2019

Division of Decision and Control Systems
School of Electrical Engineering
KTH The Royal Institute of Technology

Instructions (read carefully):

- Solve Problem 1. Refer to Installation for help with getting started.
- Work in groups of 2 persons.
- Write a joint report where you answer the questions and include relevant figures.
  - **Include both persons' names and personal numbers in the report.**
  - Name the file as follows:

    `LASTNAME1-FIRSTNAME1-LASTNAME2-FIRSTNAME2-Lab1.pdf` for the report,

    where `FIRSTNAME1` is the firstname of Student 1 in the group (etc.).
  - Preferably, use the NIPS template for the report: https://nips.cc/Conferences/2018/PaperInformation/StyleFiles
  - You must attach your code in the report as part of the appendix.
  - Hand-written solutions will not be corrected.
- **Both** students in the group should upload the report as a .pdf-file to Canvas before December 14, 23:59. The deadline is strict.

Good luck!

Problem 1:
Deep Reinforcement Learning for Cartpole

For this lab, we will apply deep reinforcement learning to solve a popular benchmark problem in classic control theory. Consider a cart (Figure 1) confined on a frictionless track with a pole attached on an unactuated revolute joint. The system is initialized with the pole standing upright and the goal is to prevent it from falling over. At every time instant, we can observe both the cart position x (w.r.t. the centre) and pole angle θ as well as their derivatives. Furthermore, a force of magnitude $1N$ can be applied on either side of the cart.

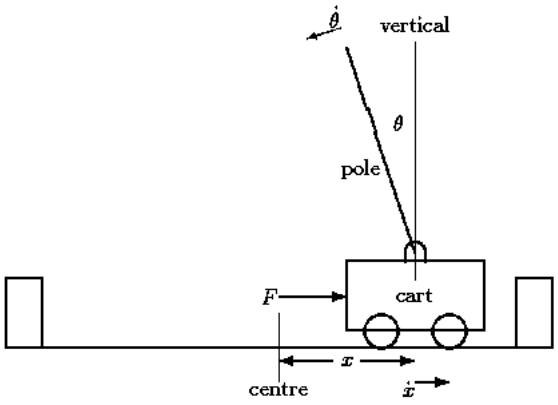

Figure 1: The cart and pole system.

We wish to train an agent to find the optimal policy that achieves this goal. To this end, we set up an episodic Reinforcement Learning (RL) problem where the agent receives a reward $r_t = 1$ for every time step that θ is kept within 12.5 degrees from the vertical position and the magnitude of x is less than $2.4m$. The agent receives $r_t = 0$ otherwise. The episode terminates after 200 time steps or once the agent receives 0 reward for the first time.

(a) Formulate the RL problem (state-space, action-space, etc.). Can you use standard RL methods to solve it? Why would you consider deep RL?

As our training environment, we will use the CartPole-v0 simulator which is part of the gym library; a reinforcement learning research platform provided by OpenAI. We also provide you with the file *cartpole_dqn.py* which contains an incomplete implementation of DQN.

(b) Familiarize yourself with the code. Provide a brief description outlining the steps taken in *main* as well as the behavior of each function.

(c) Write pseudo-code for DQN and explain it line by line. Associate your pseudo-code line numbers with corresponding (and potentially missing) parts of the code provided to you.

(d) Consult the Keras documentation and briefly explain the layout of the given neural network model. Note that we use a network which takes only the state as input and outputs the Q values for each action at that state.

(e) Fill your ϵ-greedy policy code in the *get_action* function.

(f) Complete the *train_model* function by filling in the computation of the target value.

At this point your code should be complete and able to run error-free. With the default hyperparameters and neural network model, the agent will outperform the random policy but will be very inconsistent.

(g) Assess the impact of the model on training. Try increasing the number of neurons (nodes) and document their impact on the agent's performance. Is there a benefit to increasing the number of hidden layers? Note that you may have to adjust some of the hyperparameters referred to below in order to observe significant differences.

(h) Once you are satisfied with your model, investigate the effect of the discount factor, learning rate, and memory size on learning performance. Document your findings with three representative simulations per parameter.

(i) The variable target_update_frequency controls how often the target network is updated (measured in episodes). Investigate its impact on training and document it with three representative simulations.

(j) Based on the above, choose an appropriate set of hyperparameters and model. In the code, set the variable self.check_solve to True and verify that you indeed solve the problem. The problem is considered solved when the average reward over 100 consecutive episodes is greater than or equal to 195.

Installation

The provided code has the following dependencies:

- NumPy: Fundamental package for scientific computing in Python.
- Tensorflow: Software library for training neural networks.
- Keras: High level neural network API for Python which calls Tensorflow.
- Matplotlib: 2D Matlab-like plotting library.

Below are step by step installation instructions, which have been tested with Windows, Linux, and Mac operating systems. If you have already followed the installation instructions of lab 0, you may disregard these.

Windows

For Windows we recommend that you first install the Conda package manager. This gives you access to a Linux-like terminal which allows you to simplify the installation procedure, as well as create virtual environments to avoid conflict with other Python packages you might have installed. You can either install Anaconda or Miniconda; we recommend the later because it is minimalistic (does not come with a bloat of software and packages you won't use).

1. Download Anaconda with Python 3.7 at `https://conda.io/miniconda.html`.
2. Install Anaconda (Add Anaconda to your PATH variable when prompted).
3. Run Anaconda Prompt.
4. Create a virtual environment **tensorflow** using Python 3.7:

   ```
   $ conda create -n tensorflow pip python=3.6
   ```

5. Activate the virtual environment:

   ```
   $ conda activate tensorflow
   ```

6. Change directory to the cartpole folder:

   ```
   (tensorflow) $ cd /path/to/cartpole
   ```

7. Install the required packages(notice that it is only installed in the virtual environment):

   ```
   (tensorflow) $ pip install -r requirements.txt
   ```

Linux

For Linux, the installation process is done entirely through the terminal:

1. Open a terminal.
2. Install Python 3.7:

   ```
   $ sudo apt-get install python3.7
   ```

3. Install Python virtualenv using pip:

```
$ sudo pip install virtualenv
```

4. Create a virtual environment:

```
$ virtualenv -p /usr/bin/python3.7 tensorflow
```

5. Activate the virtual environment **tensorflow** using Python 3.7::

```
$ source ~/tensorflow/bin/activate
```

6. Change directory to the cartpole folder:

```
(tensorflow) $ cd /path/to/cartpole
```

7. Install the required packages(notice that it is only installed in the virtual environment):

```
(tensorflow) $ pip install -r requirements.txt
```

Mac

For Mac we recommend that you first install Homebrew which adds the ability to install and update packages from the terminal, similarly to Linux. Afterwards, follow these instructions:

1. Open a terminal.
2. Install Python 3.7

```
$ brew install python
```

3. Install Python virtualenv using pip:

```
$ sudo pip install virtualenv
```

4. Create a virtual environment:

```
$ virtualenv -p /usr/bin/python3.7 tensorflow
```

5. Activate the virtual environment **tensorflow** using Python 3.7::

```
$ source ~/tensorflow/bin/activate
```

6. Change directory to the cartpole folder:

```
(tensorflow) $ cd /path/to/cartpole
```

7. Install the required packages(notice that it is only installed in the virtual environment):

```
(tensorflow) $ pip install -r requirements.txt
```