Reinforcement Learning
Project work

November 12, 2018

1 Introduction

The project work for the course is about implementing a reinforcement learning agent that can play the game of Pong (see Figure 1). In this environment, the agent controls one paddle and can take one of two actions: moving up or down.

The assignment is supposed to be done in groups of 2 students. If you need to find a partner for the project, please join the `project-groups` channel on Slack and advertise yourself :). Each group can give a name to their agent (anything you like, keep it civil).

To get familiar with Pong and reinforcement learning from pixels, you can read the blog-post by Andrej Karpathy: `http://karpathy.github.io/2016/05/31/rl/`.

You are allowed to use get inspiration from other sources, **as long as you reference them and clearly state which parts you used, why, and how they work**.

2 Report

Your report should be structured as follows:

1. Introduction
2. Review of external sources you used
3. Approach and method
4. Results and performance analysis
5. Conclusions

The report should be submitted in a PDF file. For more details, refer to section 4.

3 Technical requirements

Your agent **has to be trained using some kind of reinforcement learning method**. Hardcoded algorithms or other machine learning techniques (such as supervised learning) will not be accepted.

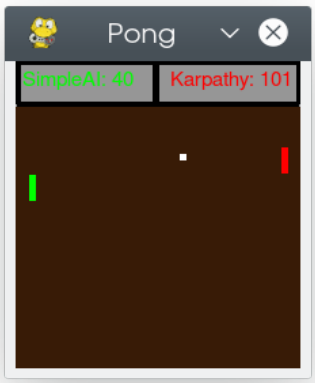

Figure 1: Pong

3.1 The pong game

Components:

- `pong.py`: Contains the implementation of the pong game. It provides the following public methods: `step(actions)`, `reset()`, `render()`, `set_names(name player 1, name player2)`. An example implementation of how these functions are used can be found in `test_pong_ai.py`, which demonstrates an implementation for 2 simple hard-coded AI players.
- `simple_ai.py`: A simple hard-coded pong agent
- `test_pong_ai.py`: This file contains an exemplary implementation of two simple AI agents playing against each other.

3.2 Interfacing with the pong game

Your agent should be contained in a separate file called `agent.py`, and defined as a separate class called `Agent`. The `Agent` class must implement the following interface:

- `load_model(filename:str):void` - a method that takes a model file name (`string`) as input and loads the saved model,
- `reset():void` - a function that takes no arguments and resets the agent's state after an episode is finished,
- `get_action(frame:np.array):int` - a function that takes a `raw` frame from Pong (as a numpy array), and return an action (integer),
- `get_name():str` - a function that returns the name of the groups agent (max 16 characters, ASCII only).

3.3 Changes to the pong game

Please be informed that during the first 1-2 weeks it could be possible that we have to make minor changes to the interface or the game. (Fixing bugs or adjusting things for the competition) We will inform you on mycourses and slack should this be the case. Stay tuned!

4 Grading

The grading will be based on the following criteria:

1. Approach, **together with a thorough explanation of the reasoning behind all design choices** - 20%
2. Analysis of the whole system and its performance - 20%
3. Performance against the baseline AI included in OpenAI Gym (based on the winrate) - 20%
4. Performance in the competitive phase - 30%
5. Report quality - 10%

During the competitive phase, the agents submitted by all groups will be evaluated against each other. The final score for this part will be based on the number of encounters won by each group.

The best 4 agents will fight against each other and compete for extra bonus points in the final stage, which will take place after the deadline. The exact time and place will be announced in early December.

5 Hints

Several hints that may help you to keep the ball bouncing :)

5.1 Computational power for training

You can train your models on the computers in the Maari building. In order to connect remotely, first use SSH to log in to `kosh.aalto.fi` or `lyta.aalto.fi`, and from there use the `ssh` command to connect to one of the CS computers.

The machine available for students are:

1. Maari-A: albatrossi, broileri, dodo, drontti, emu, fasaani, flamingo, iibis, kakadu, kalkkuna, karakara, kasuaari, kiuru, kiwi, kolibri, kondori, kookaburra, koskelo, kuukkeli, lunni, moa, pelikaani, pitohui, pulu, ruokki, siira, strutsi, suula, tavi, tukaani, undulaatti
2. Maari-C: akaatti, akvamariini, ametisti, baryytti, berylli, fluoriitti, granaatti, hypersteeni, jade, jaspis, karneoli, korundi, kuukivi, malakiitti, meripihka, opaali, peridootti, rubiini, safiiri, sitriini, smaragdi, spektroliitti, spinelli, timantti, topaasi, turkoosi, turmaliini, vuorikide, zirkoni

3. Paniikki: brainfuck, deadfish, bogo, befunge, bit, smurf, piet, emo, entropy, false, fractran, fugue, glass, haifu, headache, intercal, malbolge, numberwang, haifu, ook, regexpl, remorse, rename, shakespeare, smith, spaghetti, thue, unlambda, wake, whenever, whitespace, zombie

Remember that there might be other people working on these machines - please check the usage with `htop` and `nvidia-smi` before running heavy training.

Also, don't run your programs directly on Kosh or Lyta - those servers were not designed to handle heavy workloads, and are meant to be used as gateways to access other Aalto machines.

5.2 Training over multiple days

In order to leave your processes running in the background after you'v closed your SSH connection or left the machine, you can use the `screen` command. When you start `screen`, it will open a new terminal, in which you can start your lengthy process. When you want to log out of the machine (or close the SSH session), you can detach your `screen` terminal by pressing Ctrl+a and Ctrl+d (one after another). Doing so will close the terminal and leave your processes running in the background.

To reconnect to an existing `screen` session, use `screen -d`.

5.3 Some techniques to try out

A (very) brief overview of machine learning techniques that can help you achieve good results:

- Convolutional neural networks - neural networks with fewer weights, aimed at spatially processing structured data (such as images),
- Variatonal autoencoders - a way of performing unsupervised feature learning (and reducing the dimensionality of your data; example use in RL in [1]),
- Spatial soft-argmax layers - extracting coordinates of points-of-interest from the image [2],

To make your models perform better (or train faster) you can also use one of the "fancier" reinforcement learning algorithms. Things you can check out:

- Deep Q Networks (DQN) - a lot of improvements over the 'basic' Q-learning we had in the exercises [3],
- Trust-region policy optimization (TRPO) - applying additional constraints to policy gradient policy updates [4],
- Proximal policy optimization (PPO) - similar idea to TRPO with simplified mathematical formulation [5],
- Actor-critic with experience replay (ACER) - reuse past experience when doing policy updates [6],

5.4 Testing your model

Later on we will release scripts which will allow you to test your agents against each other, as well as see if your class interface works properly. Stay tuned!

6 Submission

The submission deadline for the project work is the 9th of December, 23:55. The project should be submitted through MyCourses.

Your submission should include the code used for training the agent, the agent itself (in a separate `.py` file, meeting the interface specifications listed in section 3), as well as a trained model and the PDF report (details in section 2).

References

[1] D. Ha and J. Schmidhuber, "World models," *CoRR*, vol. abs/1803.10122, 2018. [Online]. Available: http://arxiv.org/abs/1803.10122

[2] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *CoRR*, vol. abs/1504.00702, 2015. [Online]. Available: http://arxiv.org/abs/1504.00702

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013. [Online]. Available: http://arxiv.org/abs/1312.5602

[4] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," *CoRR*, vol. abs/1502.05477, 2015. [Online]. Available: http://arxiv.org/abs/1502.05477

[5] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[6] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, "Sample efficient actor-critic with experience replay," *CoRR*, vol. abs/1611.01224, 2016. [Online]. Available: http://arxiv.org/abs/1611.01224