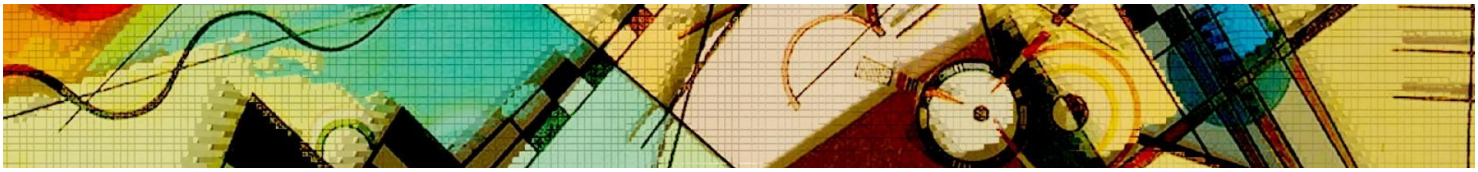
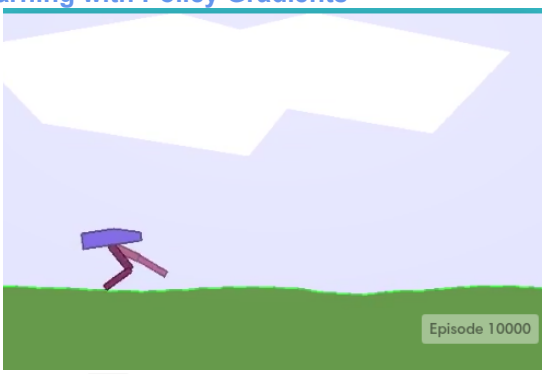


CSC411 Project 4: Reinforcement Learning with Policy Gradients



In this project, you will gain more experience with TensorFlow, and experiment with Policy Gradient methods for a simple Reinforcement Learning problem using the OpenAI Gym framework.

Working with OpenAI Gym

You will need to either install OpenAI Gym on your computer, or work in the Teaching Labs

Installing OpenAI Gym

The easiest thing to try is running the following in the Python shell using

```
>>> pip install gym
```

Alternatively, you can follow the instructions here. (Install `pip` and `git` if necessary. Anaconda comes with pip install already. Note that usually, if you are using `pip` from the command line, it would be `pip3` for Python 3. Caution: in general, using `pip` from the command line might install the library for the wrong Python binady. See here.)

Installing OpenAI Gym on a VM

Especially for installing Box2D (which is only needed to run the Walker demo; it's not strictly necessary), OpenAI Gym could be somewhat hard to deal with. Instructions for installing Linux Mint on a VM are here. Instead of installing everything on the VM yourself, you can also open this image (warning: 6GB) using VirtualBox.

Using OpenAI Gym on teach.cs

In the Teaching Labs, you will need to run a virtual machine. On a teach.cs workstation, type `cvm csc411`.

The username is "student" and the password is "csc411h". The system will force a password change at the first login.

If you want to reset your VM, type `cvm -DESTROY csc411`.

In the VM, OpenAI gym and TensorFlow are installed for for python3.

Part 1 (10 pts)

Handout code for learning in the `BipedalWalker-v2` environment using REINFORCE is provided here. Note that the learning will take *lots* of time – don't expect it to learn anything good anytime soon! The policy function π_θ is implemented as a single-hidden-layer neural network.

Explain precisely why the code corresponds to the pseudocode on p. 271 of Sutton & Barto. Specifically, in your report, explain how all the terms (G_t , π , and the update to θ) are computed, quoting the relevant lines of Python.

Part 2 (20 pts)

Your job is to now write an implementation of REINFORCE that will run for the `CartPole-v0` (source code here) environment.

In the Cart Pole task, two actions are possible – applying a force pushing left, and applying a force pushing right. Each episode stops when the pole inclides at an angle that larger than a threshold, or when the cart moves out of the frame. The at each time-step before the episode stops, the reward is 1.

The policy function should have two outputs – the probability of "left" and the probability of "right.' Implement the policy function as a softmax layer (i.e., a linear layer that is then passed through softmax.) Note that a softmax layer is simply a fully-connected layer with a softmax actication.

This video should be helpful with figuring out the policy paramterization.

In your report, detail all the modifications that you had to make to the handout code, one-by-one, and briefly state why and how you made the modifications. Include new code (up to a few lines per modification) that you wrote in your report.

Hint: if you compute the log-probabilities `log_pi` tensor of dimension `[None, 2]`, you will then want to compute a tensor that includes all the probabilities of the actual actions taken. This can be done using

```
act_pi = tf.matmul(tf.expand_dims(log_pi, 1), tf.one_hot(y, 2, axis=1))
```

(If you use this line, you have to explain why it makes sense and what it means.)

Part 3 (20 pts)

Train CartPole using your implementation of REINFORCE. (Use a very small learning rate, and a discount rate $\gamma = 0.99$.) You do not have to train it until it works perfectly (i.e., the episode doesn't stop until time-step 200), but wait at least until the average number of time-steps per episode is 50. (It's better if you wait longer, although you are not required to do that.)

Part 3(a)

In your report, include a printout that shows how the weights of the policy function changed and how the average number of time-steps per eipsode (over e.g. the 25 last episodes) changed as you trained the agent.

Part 3(b)

Explain why the final weights you obtained make sense. This requires understanding the what each input dimension means (see the source code for CartPole-v0)

What to submit

The project should be implemented using Python 2 or 3, using TensorFlow. Your report should be in PDF format. You should use LaTeX to generate the

report, and submit the .tex file as well. A sample template is on the course website. You will submit at least the following file: `cartpole.py`, as well as the write-up. You may submit more files as well.

Reproducibility counts! We should be able to obtain all the graphs and figures in your report by running your code. The only exception is that you may pre-download the images (what and how you did that, including the code you used to download the images, should be included in your submission.) Submissions that are not reproducible will not receive full marks. If your graphs/reported numbers cannot be reproduced by running the code, you may be docked up to 20%. (Of course, if the code is simply incomplete, you may lose even more.) Suggestion: if you are using randomness anywhere, use `numpy.random.seed()`.

You must use LaTeX to generate the report. LaTeX is the tool used to generate virtually all technical reports and research papers in machine learning, and students report that after they get used to writing reports in LaTeX, they start using LaTeX for all their course reports. In addition, using LaTeX facilitates the production of reproducible results.

Using my code

You are free to use any of the code available from the CSC411 course website.

Readability

Readability counts! If your code isn't readable or your report doesn't make sense, they are not that useful. In addition, the TA can't read them. You will lose marks for those things.

Academic integrity

It is perfectly fine to discuss general ideas with other people, if you acknowledge ideas in your report that are not your own. However, you must not look at other people's code, or show your code to other people, and you must not look at other people's reports and derivations, or show your report and derivations to other people. All of those things are academic offences.