Computer Science Department
Neuro-Cognitive Modeling

Summer Term 2022

Recurrent and Generative Neural Networks

Exercise Sheet 06

Release: July 1, 2022 Deadline: July 14, 2022 (23:59 pm)

General remarks:

- Download the file `exercisesheet06.zip` from the lecture site (ILIAS). This archive contains files (python modules), which are required for the exercises.
- **Add a brief documentation (pdf) to your submission**. The documentation should contain protocols of your experiments, parameter choices, and a discussion of the results.
- When questions arise start a forum discussion in ILIAS or contact us via email:
  Fedor Scholz (`fedor.scholz@uni-tuebingen.de`)
  Christian Gumbsch (`christian.gumbsch@uni-tuebingen.de`)
  Sebastian Otte (`sebastian.otte@uni-tuebingen.de`)

Introduction

In this exercise sheet we will perform *model predictive control* (MPC) with the *cross-entropy method* (CEM) [1, 2]. The goal is to safely control and land a space ship in a simulation. For this to work, you need to `pip install torch gym[box2d] matplotlib pyglet`.

Excercise 1 – Cross-entropy method [30 points]

CEM Implementation [30 points]

Implement the CEM in `planners.py`. Your class should inherit the `Planner` class. See the `RandomPlanner` for an example of a planner. We provide reasonable values for the hyperparameters as can be seen in `run.py`. The `policy_handler` should be applied to each sampled action in order for the actions to stay in a suitable range as defined by the environment.

Apply a planner to the `SimpleControlGym` environment in `simple_control_env.py` by executing `python run.py --render`. Add the option `--cem` to use your implementation of CEM instead of a random planner. This environment comes with a corresponding ground truth transition model `SimpleControlModel` in `models.py`, which is directly derived from the environment itself. This way you can test your implemented CEM while making sure that the transition model is correct. Otherwise, in case of bad performance, it can be hard to detect whether this stems from a bad transition model or an incorrectly implemented CEM. If you see the agent running towards the orange goal area, you likely implemented the CEM successfully. Note that it is normal for the agent to still run around a bit once it reached the goal area.

Excercise 2 – Model predictive control [20 points]

Now that you are sure your implementation of the CEM is correct, we want to apply it to the `LunarLander` environment. Please read the documentation of the environment. [1] In particular, have a look at how the observations and actions look like. In our code, we drop the information about leg contacts since it is not predictable from position and velocity information alone.

Use the following argument when executing `python run.py` to adjust its behavior:

- Use `--lunarlander` to switch to the `LunarLander` environment.
- Omit `--render` in order to disable rendering and speed up the process during training.
- Use `--record` if you want to save images of the environment to your hard disk. This way you can let your model run and later on inspect what the agent did by looking at the images.
- Use `--seed n` to use seed `n`.

Adjust the variable `state_path` in the code to a corresponding checkpoint path in order to continue training or use a saved model for control. Models are saved after each epoch to `models/{model_id}/{epoch}.pth`.

(a) Transition Model Implementation [10 points]

Implement a neural network that can learn a suitable transition model `ComplexControlModel` in `models.py`. Think about the necessary inputs and outputs and their standardization. Neural networks prefer balanced input values that are approximately standard normally distributed. You do not need a recurrent neural network.

(b) Transition Model Training and MPC-CEM [10 points]

Train the implemented transition model on the `LunarLander` environment with `run.py` by adding the option `--train`. For this, find suitable hyperparameters and define an appropriate optimizer. As labels or targets during training, we use the observations directly coming from the environment. Feel free to implement a dataset and dataloader if you want to train on batches, this however, is not necessary. Make use of the heuristic provided by the environment for suitable actions during training.

Apply your CEM with your trained transition model to the `LunarLander` environment with `run.py`. See if you can make the rocket land more or less reliably in right range between the flags. If required, adapt the criterion used by the CEM. A perfect landing accuracy is not required! Having more than 5 out of 10 is already a good achievement.

[1]https://www.gymlibrary.ml/environments/box2d/lunar_lander/

References

[1] R. Rubinstein, “The cross-entropy method for combinatorial and continuous optimization,” *Methodology and computing in applied probability*, vol. 1, no. 2, pp. 127–190, 1999.

[2] C. Pinneri, S. Sawant, S. Blaes, J. Achterhold, J. Stueckler, M. Rolinek, and G. Martius, “Sample-efficient cross-entropy method for real-time planning,” *arXiv preprint arXiv:2008.06389*, 2020.