Deep Learning Lab Course
Third Exercise: CarRacing

Maria Hügle

November 13, 2018

The goal of this exercise is to experiment with imitation learning. You will implement a behavioral cloning agent and evaluate its performance on the CarRacing control task from the OpenAI Gym benchmark suite. Therefore, you'll have to collect data by driving on the track as an expert. Add your driving score and the score of your agents in our google doc to take part in a small competition.

At the end, send us a report (1 or 2 pages) with your experiments and results, including tables and plots. Submit this exercise until the 04.12.2018 via mail to hueglem@cs.uni-freiburg.de and eitel@cs.uni-freiburg.de.

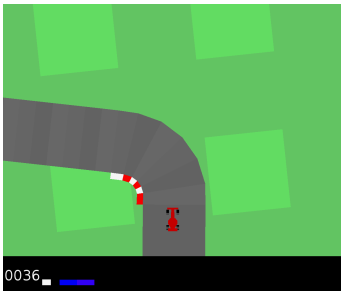

Figure 1: CarRacing

1 Getting Set Up

The code can be found at https://github.com/aisrobots/dl-lab-2018. Besides TensorFlow, install OpenAI Gym[1] and Tensorboard[2].

```
git clone https://github.com/openai/gym.git
cd gym
pip install -e '.[box2d]'
```

Install Tensorboard with `pip install tensorboard`. In the command-line, execute `tensorboard --logdir=path/to/log-directory --port=6006` and watch the progress of your training in your web browser under `localhost:6006`.

2 Behavioral Cloning

1. Get familiar with the CarRacing environment by running `drive_manually.py` and get used to the car control (keys up/down/left/right). Note that this does not work via ssh, so you need to sit in front of a pool computer.

2. Collect driving data with `drive_manually.py --collect_data`. The data will be stored every 5000 steps in the `data` folder. It will take a while to collect all samples, but the more you collect the better you can train your agent[3]. The script will create the folders `./data` and `./results` with your expert score stored in a json file. You will need these results for your report and for the competition (see below).

[1] https://gym.openai.com/doc
[2] https://www.tensorflow.org/guide/summaries_and_tensorboard
[3] collect at least 10000 samples

Have a look into the source code of `drive_manually.py` and get familiar to the interface to gym.

3. Use your collected data to train a behavioral cloning agent: Implement a convolutional neural network in `model.py` and train it in `train_agent.py`. Use Tensorboard to watch and to create plots of the training progress for the report. Evaluate the performance of your agent in `test_agent.py` over 10 episodes. Check the code further instructions/hints.

4. Improve your agent:

   - Check the distribution of actions in the dataset. You will see that it is imbalanced because most of the time you drive straight. This can be a problem for the training. Data augmentation methods like sampling the actions uniformly or other oversampling methods can help here.
   - Find a different architecture and hyperparameter configuration that improves the evaluation results of your first agent. Compare them in your report and write down why you think it works better.
   - Analyze the impact of one hyperparameter on the final performance, e.g. number of demonstrations, number of training epochs, score of expert policy or a hyperparameter that you find interesting.
   - In your first experiments you used only the current image as input feature for your network. Append the history of the last N images to your input. Check the performance for different values for N, e.g. $N \in \{1, 3, 5\}$. What happens if you increase the history length? Show the results in your report.

3 Bonus: LSTMs

(Optional) In the last exercise you experimented with different history lengths to improve your agent. You can also use recurrency to let the agent learn the best history length by itself. Implement a CNN-LSTM architecture. It involves CNNs for feature extraction and LSTMs to support sequence prediction.

4 Competition

(Optional) Choose a name and enter your final scores HERE. The winner will receive a price!