flow Documentation

Release 0.1

Cathy Wu

Aug 27, 2018

Contents:

Flow is a computational framework for deep RL and control experiments for traffic microsimulation. Visit our website for more information.

Flow is a work in progress - input is welcome. Available documentation is limited for now. Tutorials are available in iPython notebook format.

If you are looking for Akvo FLOW, their documentation can be found at http://flowsupport.akvo.org.

CHAPTER 1

Introduction

1.1 The Team

Flow was built by Professor Alexandre Bayen's lab at UC Berkeley. Contributing members are Cathy Wu, Eugene Vinitsky, Aboudy Kreidieh, Kanaad Parvate, Nishant Kheterpal, Kathy Jang, and Ethan Hu. Alumni contributors include Leah Dickstein, Nathan Mandi, Ananth Kuchibhotla, and Arjun Sridhar.

CHAPTER 2

Setup Instructions

To get Flow running, you need three things: Flow, SUMO, and (optionally) a reinforcement learning library (RL-lib/rllab). If you choose not to install a reinforcement learning library, you will still be able to build and run SUMO-only traffic tasks, but will not be able to run experiments which require learning agents. Once each component is installed successfully, you might get some missing module bugs from Python. Just install the missing module using your OS-specific package manager / installation tool. Follow the shell commands below to get started.

2.1 Dependencies

We begin by installing dependencies needed by the four repositories mentioned above. **It is highly recommended that users install 'Anaconda <https://www.anaconda.com/download>'_ or 'Miniconda <https://conda.io/miniconda.html>'_ for Python and the setup instructions will assume that you are doing so.**

For Ubuntu 16.04:

```
sudo apt-get update && sudo apt-get upgrade
sudo apt-get install cmake swig libgtest-dev python-pygame python-scipy autoconf␣
↪libtool pkg-config libgdal-dev libxerces-c-dev libproj-dev libfox-1.6-dev libxml2-
↪dev libxslt1-dev build-essential curl unzip flex bison python python-dev python3-dev
sudo pip3 install cmake cython
```

For OSX (feel free to ignore the rllab dependencies if you don't wish to install it):

```
# rllab dependencies
brew install swig sdl sdl_image sdl_mixer sdl_ttf portmidi
# sumo dependencies
brew install Caskroom/cask/xquartz autoconf automake pkg-config libtool gdal proj␣
↪xerces-c fox
```

2.2 sumo

Next, we install SUMO, an open source traffic microsimulator which will be used the update the states of vehicles, traffic lights, and other RL and human-driven agents during the simulation process.

```
cd ~
git clone https://github.com/eclipse/sumo.git
cd sumo
git checkout 1d4338ab80
make -f Makefile.cvs
```

If you have OSX, run the following commands

```
export CPPFLAGS=-I/opt/X11/include
export LDFLAGS=-L/opt/X11/lib
./configure CXX=clang++ CXXFLAGS="-stdlib=libc++ -std=gnu++11" --with-xerces=/usr/
↪local --with-proj-gdal=/usr/local
make -j$nproc
echo 'export SUMO_HOME="$HOME/sumo"' >> ~/.bash_profile
echo 'export PATH="$HOME/sumo/bin:$PATH"' >> ~/.bash_profile
echo 'export PYTHONPATH="$HOME/sumo/tools:$PYTHONPATH"' >> ~/.bash_profile
source ~/.bash_profile
```

If you have Ubuntu 14.04+, run the following command

```
./configure
make -j$nproc
echo 'export SUMO_HOME="$HOME/sumo"' >> ~/.bashrc
echo 'export PATH="$HOME/sumo/bin:$PATH"' >> ~/.bashrc
echo 'export PYTHONPATH="$HOME/sumo/tools:$PYTHONPATH"' >> ~/.bashrc
source ~/.bashrc
```

Finally, test your sumo install and version by running the following commands

```
which sumo
sumo --version
sumo-gui
```

2.3 Flow

Once sumo and the various dependencies are in place, we are ready to install a functional version of Flow. With this, we can begin to simulate traffic within sumo using OpenAI gym-compatible environments. Note that separate RL algorithms will be needed to train autonomous agents within the simulation to improve various traffic flow properties (see the sections on rllab-multiagent and Ray/RLlib for more).

```
cd ~
git clone https://github.com/flow-project/flow.git
cd flow
conda env create -f environment.yml
source activate flow
python3 setup.py develop
```

For linux run

```
echo 'export PYTHONPATH="$HOME/flow:$PYTHONPATH"' >> ~/.bashrc
source ~/.bashrc
```

For mac run

```
echo 'export PYTHONPATH="$HOME/flow:$PYTHONPATH"' >> ~/.bash_profile
source ~/.bash_profile
```

2.4 Testing the Installation

Once the above modules have been successfully installed, we can test the installation by running a few examples.

Let's see some traffic action:

```
python examples/sumo/sugiyama.py
```

Running the following should result in the loading of the SUMO GUI. Click the run button and you should see unstable traffic form after a few seconds, a la (Sugiyama et al, 2008). This means that you have Flow properly configured with SUMO.

Optionally, run the unit tests:

```
nose2 -s tests/fast_tests
```

Congratulations, you now have successfully set up Flow!

2.5 rllab-multiagent (optional)

Flow has been tested on a variety of RL libraries, the installation of which is optional but may be of use when trying to execute some of the examples files located in Flow. rllab-multiagent is one of these such libraries. In order to install the *rllab-multiagent* library, follow the below instructions

```
cd ~
git clone https://github.com/cathywu/rllab-multiagent.git
cd rllab-multiagent
conda env create -f environment.yml
python3 setup.py develop
```

For linux run

```
echo 'export PYTHONPATH="$HOME/rllab-multiagent:$PYTHONPATH"' >> ~/.bashrc
source ~/.bashrc
```

For mac run

```
echo 'export PYTHONPATH="$HOME/rllab-multiagent:$PYTHONPATH"' >> ~/.bash_profile
source ~/.bash_profile
```

2.6 Ray/RLlib (optional)

RLlib is another RL library that has been extensively tested on the Flow repository. First visit <http://ray.readthedocs.io/en/latest/installation.html> and install the required packages. The installation process for this library is as follows:

```
cd ~
git clone https://github.com/eugenevinitsky/ray.git
pushd ray/python
sudo python3 setup.py develop
popd
```

If missing libraries cause errors, please also install additional required libraries as specified at <http://ray.readthedocs.io/en/latest/installation.html> and then follow the setup instructions.

2.7 Getting started (rllab-multiagent)

To run any of the RL examples, make sure to run

```
source activate flow
```

In order to test run an Flow experiment in rllab-multiagent, try the following command:

```
python examples/rllab/stabilizing_the_ring.py
```

If it does not fail, this means that you have Flow properly configured with rllab-multiagent.

2.8 Getting started (Ray/RLlib)

See getting started with RLlib for sample commands.

To run any of the RL examples, make sure to run

```
source activate flow
```

In order to test run an Flow experiment in RLlib, try the following command:

```
python examples/rllib/stabilizing_the_ring.py
```

If it does not fail, this means that you have Flow properly configured with RLlib.

To visualize the training progress:

```
tensorboard --logdir=~/ray_results
```

For information on how to deploy a cluster, refer to the Ray instructions. The basic workflow is running the following locally, ssh-ing into the host machine, and starting jobs from there.

```
ray create_or_update scripts/ray_autoscale.yaml
ray teardown scripts/ray_autoscale.yaml
```

2.9 Custom configuration

You may define user-specific config parameters as follows

```
cp flow/core/config.template.py flow/core/config.py  # Create template for users␣
↪using pycharm
```

CHAPTER 3

Visualization

Flow supports visualization of both rllab and RLlib experiments.

3.1 rllab

Call the rllab visualizer with

```
python ./visualizer_rllab.py /result_dir/itr_XXX.pkl
```

The rllab visualizer also takes some inputs:

- `--num_rollouts`
- `--plotname`
- `--use_sumogui`
- `--run_long`
- `--emission_to_csv`

The `params.pkl` file can be used as well.

3.2 RLlib

Call the RLlib visualizer with

```
python ./visualizer_rllib.py /ray_results/result_dir 1
# OR
python ./visualizer_rllib.py /ray_results/result_dir 1 --run PPO
# OR
python ./visualizer_rllib.py /ray_results/result_dir 1 --run PPO \
    --module cooperative_merge --flowenv TwoLoopsMergePOEnv \
    --exp_tag cooperative_merge_example
```

The first command-line argument corresponds to the directory containing experiment results (usually within RLlib's `ray_results`). The second is the checkpoint number, corresponding to the iteration number you wish to visualize. The `--run` input is optional; the default algorithm used is PPO. If the experiment module, Flow environment name, and experiment tag have not been stored automatically (see section below), then those parameters can be passed in using the flags `--module`, `--flowenv`, and `--exp_tag`.

3.2.1 Parameter storage

RLlib doesn't automatically store all parameters needed for restoring the state of a Flow experiment upon visualization. As such, Flow experiments in RLlib include code to store relevant parameters. Include the following code snippet in RLlib experiments you will need to visualize

```
# Logging out flow_params to ray's experiment result folder
json_out_file = alg.logdir + '/flow_params.json'
with open(json_out_file, 'w') as outfile:
    json.dump(flow_params, outfile, cls=NameEncoder, sort_keys=True, indent=4)
```

These lines should be placed after initialization of the PPOAgent RL algorithm as it relies on `alg.logdir`. Store parameters before training, though, so partially-trained experiments can be visualized.

Another thing to keep in mind is that Flow parameters in RLlib experiments should be defined **outside** of the `make_create_env` function. This allows that environment creator function to use other experiment parameters later, upon visualization.

CHAPTER 4

Code Documentation

4.1 flow package

4.1.1 Subpackages

flow.benchmarks package

Submodules

flow.benchmarks.bottleneck0 module

flow.benchmarks.bottleneck1 module

flow.benchmarks.bottleneck2 module

flow.benchmarks.figureeight0 module

flow.benchmarks.figureeight1 module

flow.benchmarks.figureeight2 module

flow.benchmarks.grid0 module

flow.benchmarks.grid1 module

flow.benchmarks.merge0 module

flow.benchmarks.merge1 module

flow.benchmarks.merge2 module

Module contents

flow.controllers package

CHAPTER 5

Indices and tables

- genindex
- modindex
- search