Dolphin Hands

Brian Beckman

[2019-03-25 Mon]

Contents

1 Control Shadow Hands with Dolphin

1.1 Dimensions and Degrees of Freedom

1.1.1 One Hand

The following code gets the original, old environment to tell us the dimensions.

```
import gym
import numpy as np
# env = gym.make("TwoHandsManipulateBlocks-v0")
env = gym.make("HandManipulateBlock-v0")
_= env.reset()
action = env.action_space.sample() # your agent here
observation, reward, done, info = env.step(action)
result = list({"shape(observation)": np.shape(observation[’observation’]),
               "shape(achived goal)": np.shape(observation[’achieved_goal’]),
               "shape(desired goal)": np.shape(observation[’desired_goal’]),
               "shape(action)": np.shape(action)# ,
               # "done": done,
               # "info": info
               }.items())
return result
```

In the following paragraphs, we account for all the numbers in the table above.

1. Goals

   The `achieved_goal` and `desired_goal` are 7, concatenating an (x, y, z) 3-vector position and a 4-component (w_q, x_q, y_q, z_q) quaternion. The quaternion is normalized, so the goals aren’t *mathematically* of seven dimensions, only six. However, the code keeps track of the seven parameters as if they were independent. Evidently, that practice is not a source of trouble.

2. Actions

   Action for one hand is a row vector of length twenty. These represent the twenty tendons available on the robot. The code clips actions to the interval $[-1, +1]$.

3. Joints in Hand

   There are twenty-four joints in each hand:

   (a) two wrist joints, `WRJ1` and `WRJ0` (in reverse order, intentionally)

   (b) four first-finger joints, `FFJ3` through `FFJ0` (reverse order)

   (c) four middle-finger joints, `MFJ3` through `MFJ0` (reverse order)

   (d) four ring-finger joints, `RFJ3` through `RFJ0` (reverse order)

   (e) five little-finger joints, `LFJ4` through `LFJ0` (reverse order)

(f) five thumb joints, `THJ4` through `THJ0` (reverse order)

The following symbols in the code build up the state:

- twenty-four `robot_qpos`, generalized coordinates, in the order above
- twenty-four `robot_qvel`, generalized velocities

`qpos` appears to be a generic word in the code for a 7\=/vector (three Cartesian position coordinates concatenated to four components of a normalized quaternion *versor*, in pos-quat order despite the name), but `qpos` is abused, in this case, to contain twenty-four generalized coordinates encoding joint configuration.

Likewise, `qvel` appears to mean concatenation of velocity and angular velocity, in that order despite the name, but it's abused hear to contain twenty-four generalized velocities.

Total is forty-eight.

4. Object == Cube

   object usually means "cube on the left."

   object_right means "cube on the right."

   Each cube has thirteen dimensions:

   (a) seven elements for `object_qpos` the position and normalized quaternion (in that order, despite the name); really only six degrees of freedom because of normalization.

   (b) six elements for `object_qvel`, surmised to be $\{\dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}\}$, where ϕ, θ, ψ are *roll*, *pitch*, and *yaw*, respectively.

5. Total: Sixty-One Dimensions

   $\boxed{48 + 13 = 61}$

1.1.2 Two Hands

The state of the left hand plus cube has a total of sixty-one elements. A linear regulator for one hand plus cube has 61 columns and 20 rows, for $1,220$ elements. We may guess at a linear regulator by constructing a multivariate normal in $1,220$ dimensions and then reshaping.

$$\boxed{\text{dimensions of linear regulator for one hand and one cube} = 1,220}$$

1.1.3 Sequenced Linear Controllers

We will need one new linear-control matrix (LCM) every few steps, say every ten. In $1,000$ steps, we will need 100 LCMs, or $122,000$ numbers. This is an overestimate because the LCMs will probably vary slowly with time. It will likely take fewer judgments to learn $\text{LCM}(t+1)$ than it took to learn $\text{LCM}(t)$, perhaps many fewer.

1.2 Learn a Neural Network

Alternatively, we can have Dolphin learn the Dactyl neural network directly.

2 Get the Hands Going

2.1 Start Here (Ubuntu)

```
https://github.com/rebcabin/baselines
```

The instructions for setting up the Python environment are pretty good. Here is what I ended up with:

```
env PYTHONPATH=/usr/lib/tensorflow/lib/python3.6:$PYTHONPATH pytest \
    ./baselines/common/tests/test_serialization.py -k test_serialization
```

That test *FAILS* [TODO]. The immediate goal, however, is to get the hands doing *something rather than nothing*. Obsess on the unit tests later. For now, we need to get mujoco, graphics, CUDA, Tensorflow going.

To get the hands going on Linux, you will need the following.

2.2 Graphics Display (Ubuntu)

If graphics don't work for you, you may have to do some things in this section. I broke graphics by doing `sudo apt install libglew-dev`. To fix it, I had to chase down `glfw`, which doesn't have an obvious name. This took time to figure out: you don't want to discover all this on your own.

```
https://github.com/glfw/glfw/issues/808
https://github.com/openai/mujoco-py/issues/268
https://www.reddit.com/r/learnprogramming/comments/51u1bg/
    how_to_install_glew_on_ubuntu/
```

At one point, I had to add an untrusted `.deb` repository to `apt`, but that step no longer appears necessary. If the following doesn't work for you,

```
sudo apt update
sudo apt install libglfw3-dev
sudo apt install libglfw3
```

Then *temporarily* add the following line to `/etc/apt/sources.list` using `sudo.nano` (that's the easiest way to add the line).

```
deb http://ppa.launchpad.net/keithw/glfw3/ubuntu trusty main
```

and try again.

Then comment out that line in `sources.list` because that `.deb` repo is not digitally signed and your automatic update software will stall on it. You may need to uncomment and recomment it later to reinstall `glfw`, however, so don't remove the line from the file; leave it as a reminder of what to install.

2.3 Mujoco

```
https://www.roboti.us/index.html
```

Install mujoco 150 and 200 (I leave that to you — there is a license involved, but everything goes in a directory named `~/.mujoco`.

2.4 Environment Variables

It's difficult to get the versions of mujoco, CUDA, Tensorflow, glfw, glew that will work with multiple applications. If you get all tied in knots, go here:

```
https://docs.nvidia.com/deeplearning/sdk/cudnn-install/
  index.html#ubuntu-network-installation
```

Add these lines to your `.bashrc` or `.zshrc`

```
export PATH=/usr/local/cuda-9.0/bin\
  ${PATH:+:${PATH}}
export LD_LIBRARY_PATH=/usr/local/cuda/lib64\
  ${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
export LD_LIBRARY_PATH=~/.mujoco/mjpro150/bin\
  ${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
export LD_LIBRARY_PATH=~/.mujoco/mujoco200/bin\
  ${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
export PYTHONPATH=/usr/lib/tensorflow/lib/python3.6\
  ${PYTHONPATH:+:${PYTHONPATH}}
export LD_PRELOAD=/usr/lib/x86_64-linux-gnu/libGLEW.so
```

Add them to PyCharm under `Run->Edit Configuration->Environment Variables`.

Add some, if not all, to the section on environment variables in the `Run->EditConfiguration` dialog box of PyCharm.

Use PyCharm. It's worth the trouble to set it up because its debugger is — worth the trouble.

2.5 Python

Make sure your Python is working (it must be Python 3.6, at least if you're following along with me):

```
python --version
```

Make a nice virtual environment. I called mine "shadow-hand-venv". Make sure it has at least this stuff:

```
pip freeze
```

```
absl-py==0.7.0
astor==0.7.1
atari-py==0.1.7
atomicwrites==1.3.0
attrs==19.1.0
-e git+https://github.com/rebcabin/baselines.git@1b0924#egg=baselines
box2d-py==2.3.8
certifi==2019.3.9
cffi==1.12.2
chardet==3.0.4
Click==7.0
cloudpickle==0.8.0
Cython==0.29.6
dill==0.2.9
filelock==3.0.10
future==0.17.1
gast==0.2.2
glfw==1.7.1
grpcio==1.19.0
-e git+https://github.com/rebcabin/baselines.git@1b0924#egg=gym
h5py==2.9.0
idna==2.8
imageio==2.5.0
joblib==0.13.2
Keras-Applications==1.0.7
Keras-Preprocessing==1.0.9
lockfile==0.12.2
Markdown==3.0.1
mock==2.0.0
more-itertools==6.0.0
mpi4py==3.0.1
mujoco-py==2.0.2.0
numpy==1.16.2
opencv-python==4.0.0.21
pbr==5.1.3
Pillow==5.4.1
pkg-resources==0.0.0
pluggy==0.9.0
progressbar2==3.39.3
protobuf==3.7.0
py==1.8.0
pybullet==2.4.8
pycparser==2.19
pyglet==1.3.2
PyOpenGL==3.1.0
pytest==4.3.1
pytest-forked==1.0.2
python-utils==2.3.0
requests==2.21.0
scipy==1.2.1
six==1.12.0
tensorboard==1.9.0
tensorflow==1.9.0
tensorflow-estimator==1.13.0
termcolor==1.1.0
tqdm==4.31.1
urllib3==1.24.1
Werkzeug==0.14.1
```

Activate your environment:

```
source ./shadow-hand-venv/bin/activate
```

Make sure again that Python 3.6 is working with a nice f-string example (f-strings don’t work in Python 3.5)

```
import time
return f"Hello, today’s date is {time.ctime()}"
```

Run Emacs in the background from a terminal where that environment is active. If you start Emacs without the environment, you won’t be able to run the Python code below. Here is how I do it.

```
$ nohup ~/usr/bin/emacs-26.1 &> /dev/null &
```

2.6 See Hands Run; Run, Hands, Run!

If you use a `:session` header in the following, mujoco will hang.

Give it a go, and best of luck:

```
import gym
env = gym.make("TwoHandsManipulateBlocks-v0")
# env = gym.make("CartPole-v1")
# env = gym.make("Zaxxon-v0")

observation = env.reset() # BOGUS! env.reset returns zoquetes!
for _ in range(25):
    env.render()
    action = env.action_space.sample() # your agent here (this takes random actions)
    observation, reward, done, info = env.step(action)
    if done:
        observation = env.reset() # BOGUS! env.reset returns zoquetes!
env.close()
```

2.7 Action and State Shapes

```
import gym
import numpy as np
env = gym.make("TwoHandsManipulateBlocks-v0")
_= env.reset()
return list({"action space": env.action_space,
             "observation space": env.observation_space}.items())
```

2.8 User Interface

2.8.1 Two Mujoco Windows

[2019-03-28 Thu 09:10] getting Mujoco to show two windows.

Suspending this out of bias-for-action. Turns out to require many changes inside `mujoco_py`. Mujoco assumes it controls one screen, one process. We can implement two mujocos, but it's more work. For now, I will put two hands, two cubes in one mujoco process.

Here is a comment recording my problems with it.

```
def render(self, mode='human', width=DEFAULT_SIZE, height=DEFAULT_SIZE):
    self._render_callback()
    if mode == 'rgb_array':
        self._get_viewer(mode).render(width, height)
        # window size used for old mujoco-py:
        data = self._get_viewer(mode).read_pixels(width, height, depth=False)
        # original image is upside-down, so flip it
        return data[::-1, :, :]
    # [[[ bbeckman --- human mode is ignoring width and height. The ignoring
    # happens way down deep in the mujoco layer. mujoco_py.MjViewer ignores
    # the width and height from here and opens a window full-screen. ]]]
    elif mode == 'human':
        self._get_viewer(mode).render(width, height)
```

2.8.2 Unit Tests; Commit `2df76ec`

This is a record of how to run unit tests. PLEASE do this every time you check code in.

```
$ pushd ~/Documents/baselines
$ . ./shadow-hand-venv/bin/activate
(shadow-hand-venv)
$ pytest shadow-hand-venv/lib/python3.6/site-packages/gym/
==== 224 passed, 1 skipped, 97 warnings in 21.89 seconds =====
```

2.9 Sandbox

```
return list(map(lambda _: [_], [1, 2, 3]))
```

```
return [1, 2, 3]
```

```
return [[1], [2], [3]]
```

2.9.1 Transcript of Some Action Plans

- [March 28, 2019, 7:11 AM] Beckman, Brian: important: you are familiar with piecewise linear approximations (PLA) of a function.

- [March 28, 2019, 7:12 AM] Beckman, Brian: We may need multiple 61x20 matrices, one every 10 or 20 time steps.

- [March 28, 2019, 7:12 AM] Beckman, Brian: That amounts to PLA to policy function, in 1220 dimensions
- [March 28, 2019, 7:13 AM] Beckman, Brian: alternative: directly learn the LSTM + etc NN in the paper, not learn reward and then do RL, directly learn the params in the NN
- [March 28, 2019, 7:14 AM] Beckman, Brian: if we have a bunch of 61x20 = 1220 matrices, say 1000, that's 1,220,000 params in the PLA
- [March 28, 2019, 7:15 AM] Beckman, Brian: the NN LSTM + whatever probably has 1,000,000 params
- [March 28, 2019, 7:15 AM] Beckman, Brian: so the amount of information in the two approaches (PLA vs NN) is about the same
- [March 28, 2019, 7:15 AM] Beckman, Brian: PLUS :::: SIDD has EXPLICITLY DEMANDED that we directly learn the NN with HPL
- [March 28, 2019, 7:17 AM] Beckman, Brian: so we have two ways of approx'ing the policy function: NN LSTM+whatever about 1,000,000 params, and PLA, i.e., time series of 61x20 matrices, about 1,220,000 params
- [March 28, 2019, 7:17 AM] Beckman, Brian: we have to try both
- [March 28, 2019, 7:20 AM] Beckman, Brian: Use A/B instead of RL to learn all the params in the big NN in the paper
- [March 28, 2019, 7:20 AM] Beckman, Brian: the policy NN
- [March 28, 2019, 7:21 AM] Beckman, Brian: LSTM + a whole bunch of other stuff
- [March 28, 2019, 7:21 AM] Beckman, Brian: a big freaking NN, Yushan looked into it
- [March 28, 2019, 7:21 AM] Beckman, Brian: it has about 1,000,000 params, maybe 300,000 because it's not fully connected
- [March 28, 2019, 7:22 AM] Beckman, Brian: so we can learn that NN, or we can learn a time-series of matrices
- [March 28, 2019, 7:22 AM] Pham, Thai: Estimate reward function or not?
- [March 28, 2019, 7:23 AM] Beckman, Brian: no
- [March 28, 2019, 7:23 AM] Beckman, Brian: no RL
- [March 28, 2019, 7:23 AM] Beckman, Brian: directly estimate the params in the NN
- [March 28, 2019, 7:23 AM] Pham, Thai: Ok
- [March 28, 2019, 7:23 AM] Beckman, Brian: using A/B even if it means 1,000,000 TRON trips
- [March 28, 2019, 7:23 AM] Pham, Thai: Using NN to model policy is fine
- [March 28, 2019, 7:23 AM] Beckman, Brian: yup

- [March 28, 2019, 7:23 AM] Pham, Thai: That's what I plan to use
- [March 28, 2019, 7:24 AM] Beckman, Brian: we will do both (1) use NN to model policy (2) use PLA time-series of 61x20 matrices to model policy
- [March 28, 2019, 7:24 AM] Beckman, Brian: PLA worked for pendulum so I am not sure it's wrong
- [March 28, 2019, 7:25 AM] Beckman, Brian: NN is one way to approximate a function, PLA is just another way to approximate a function
- [March 28, 2019, 7:25 AM] Beckman, Brian: two different equally valid ways to approx functions
- [March 28, 2019, 7:25 AM] Beckman, Brian: I am going to leave the NN part to you and Yushan
- [March 28, 2019, 7:26 AM] Beckman, Brian: you guys will teach me later when you have a demo
- [March 28, 2019, 7:26 AM] Beckman, Brian: I will do the time-series of 61x20 matrices
- [March 28, 2019, 7:26 AM] Pham, Thai: Ok
- [March 28, 2019, 7:27 AM] Beckman, Brian: Neda knows I just told her
- [March 28, 2019, 7:28 AM] Beckman, Brian: there will be lots of talking, later after we have something to show
- [March 28, 2019, 7:28 AM] Beckman, Brian: if anyone comes to you tell them we'll talk after we have some experiments
- [March 28, 2019, 7:29 AM] Beckman, Brian: we need to have some stuff to talk about instead of just abstract ideas
- [March 28, 2019, 7:31 AM] Beckman, Brian: you get some stuff done, you must be my trusted partner
- [March 28, 2019, 7:31 AM] Pham, Thai: No worries
- [March 28, 2019, 7:32 AM] Pham, Thai: I need one full day sitting with Dylon
- [March 28, 2019, 7:32 AM] Beckman, Brian: you can have dylon
- [March 28, 2019, 7:33 AM] Pham, Thai: Yeah I'll wait until he's done with TRON job
- [March 28, 2019, 7:33 AM] Beckman, Brian: ok good
- [March 28, 2019, 7:33 AM] Pham, Thai: I'll take him with me after that
- [March 28, 2019, 7:33 AM] Pham, Thai: For one day
- [March 28, 2019, 7:33 AM] Beckman, Brian: the hand envrt is easy to work with i am going to try to make two windows side-by-side

- [March 28, 2019, 7:33 AM] Pham, Thai: Ok good
- [March 28, 2019, 7:34 AM] Beckman, Brian: i am going solo
- [March 28, 2019, 7:34 AM] Beckman, Brian: no dependencies
- [March 28, 2019, 7:34 AM] Pham, Thai: Ok
- [March 28, 2019, 7:34 AM] Beckman, Brian: it's ok if we dupe work
- [March 28, 2019, 7:34 AM] Beckman, Brian: no cross dependencies
- [March 28, 2019, 7:34 AM] Beckman, Brian: you take Dylon, i will solo
- [March 28, 2019, 7:34 AM] Pham, Thai: Ok sounds good

3 How to

Run Scheme Code

… and how to pass its results to Python.

```
(define (eq-lists? la lb)
  (define (atom? x)
    (or (string? x) (symbol? x) (number? x)))
  (define (eq-elements? a b)
    (cond
     ((null? a) (null? b))
     ((atom? a) (equal? a b))
     (else (eq-lists? a b))))
  (cond
   ((null? la) (null? lb))
   ((atom? la) #f)
   ((list? la)
    (cond
     ((list? lb) (and (eq-elements? (car la) (car lb))
                      (eq-lists? (cdr la) (cdr lb))))
     (else #f)))))

(write (and (eq-lists? '() '())
            (eq-lists? '(a b) '(a b))
            (eq-lists? '((a) b) '((a) b))
            (eq-lists? '((a) (b)) '((a) (b)))
            (eq-lists? '(a (b)) '(a (b)))
            (not (eq-lists? 'a 42))
            (not (eq-lists? '(a) 42))
            (not (eq-lists? '(a) '(b)))
            (not (eq-lists? '((a)) 'b))
            (not (eq-lists? '((a)) '(b)))
```

```
               (not (eq-lists? '((a)) '((b))))
               (not (eq-lists? 'a 'b))
               (not (eq-lists? '(a) '(b)))
               (not (eq-lists? '((a)) '((b))))
              ))
(write 'foo)
```

```
print(f"scheme says: {from_scheme}")
print(f"python says 'bar'")
```

4 Sandbox

1 Aliquam erat volutpat. Nunc eleifend leo vitae magna. In id erat non orci commodo lobortis. Proin neque massa, cursus ut, gravida ut, lobortis eget, lacus. Sed diam.