Reinforcement Learning

Lecture 2: Environments and bandits

Chris G. Willcocks

Durham University

Lecture Overview



This lecture shows how to set up a modern RL environment ready for training using $\mathbf{Gym} \ \mathbf{C}$, $\mathbf{Brax} \ \mathbf{C}$ and a multi-armed bandits implementation as in chapter 2 of [1].

Gym and Brax

- what are they?
- how do they map to RL concepts?
- gym.Env methods and attributes
- nested spaces example

2 Colab coding

- a random agent
- exploring different environments
- minimal REINFORCE example
- multi-armed bandits

3 Custom environments

- OpenAl gym
- Google brax

Gym and Brax what are they?



OpenAl gym & Google brax

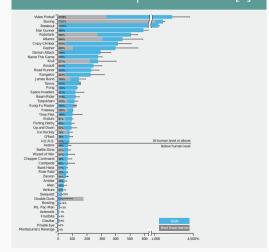
What is it?

- Collection of environments
- Most environments are freely available
- Standardised API makes comparisons and benchmarking easier
- Test same RL algorithms on many different problems

For example:

- Atari 2600 games
- Classic control problems
- Physics simulations

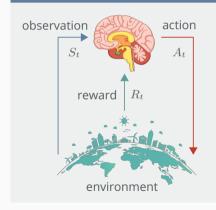
Environments and performance in [2]



Gym and Brax how does it map to RL concepts?



RL Agents



Algorithm: minimal example

```
import gym
env = gvm.make('CartPole-v0')
agent = Agent()
# init the environment to base state
# and get initial observation state
s = env.reset()
# loop until environment says it's done
done = False
while not done:
   a = agent.sample_action(s)
   s, r, done, info = env.step(a)
env.close()
```

OpenAl gym gym.Env methods and attributes



Core methods

In gym/core.py:

- **s = env.reset()** reset the environment and return the initial observation state
- s,r,done,info = env.step(action) takes a single step and returns a tuple
 with the next state, the reward,
 whether the environment is now done,
 and some meta diagnostic info
- env.close() cleanup function
- env.seed(int) seed all the environment randomness
- **env.render()** (optional) render the env state: print text, show an image...

Env attributes

- env.action_space gym.Space object that describes the shape and type of actions
- env.observation_space gym.Space object that describes the shape and type of observations
- env.reward_range tuple of per-step reward range, defaults to $(-\infty, \infty)$

gym.Space in /gym/spaces/space.py:

- Box, Discrete, Binary, ...
- Tuple, Dict can contain nested observation spaces



Example: nested observation space – gym/spaces/dict.py

Colab coding demo



Click this link:

Link to Colab code 🗹

In this demo we will cover:

- A random agent
- Exploring different environments
- Plotting and measuring baselines
- Minimal REINFORCE example
- Multi-armed Bandits chapter two of [1]



Example: gym custom environment

```
class CustomEnv(gym.Env):
    def __init__(self):
    def _seed(self, seed=None):
    def step(self, action):
        return state, reward, done, info
    def reset(self):
        return state
    def render(self):
        pass
```



Example: brax custom environment [3]

```
from brax.envs import env
class Fast(env.Env):
 def __init__(self, **kwargs):
   super().__init__(config='dt: .02', **kwargs)
 def reset(self, rng: jnp.ndarray) -> env.State:
   zero = inp.zeros(1)
   qp = brax.OP(pos=zero, vel=zero, rot=zero, ang=zero)
   obs. reward. done = inp.zeros(2)
    return env.State(qp, obs, reward, done)
  def step(self, state: env.State, action: jnp.ndarray) -> env.State:
   vel = state.gp.vel + (action > 0) * self.svs.config.dt
   pos = state.gp.pos + vel * self.svs.config.dt
   qp = state.qp.replace(pos=pos, vel=vel)
   obs = jnp.array([pos[0], vel[0]])
   reward = pos[0]
    return state.replace(qp=qp, obs=obs, reward=reward)
```

References I



- [1] Richard S Sutton and Andrew G Barto.

 Reinforcement learning: An introduction (second edition). Available online . MIT press, 2018.
- [2] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: nature 518.7540 (2015), pp. 529–533.
- [3] C Daniel Freeman et al. "Brax–A Differentiable Physics Engine for Large Scale Rigid Body Simulation". In: arXiv preprint arXiv:2106.13281 (2021).