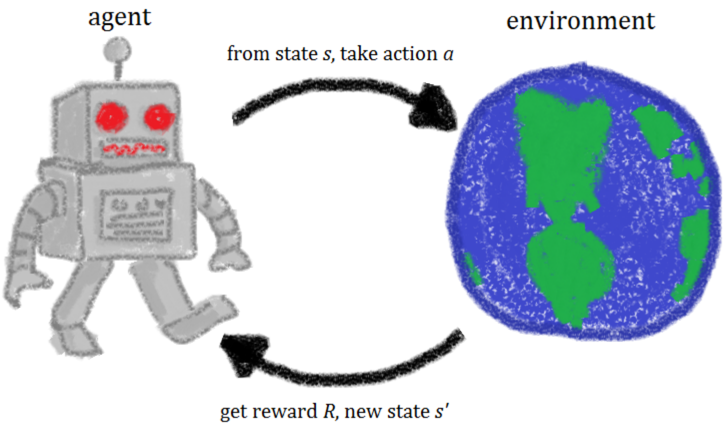
Tutorial: Reinforcement Learning with OpenAI Gym

EMAT31530/Nov 2020/Xiaoyang Wang

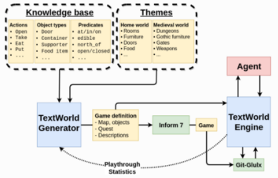


An *environment* provides the agent with state s, new state s', and the reward R. It also defines the action space.

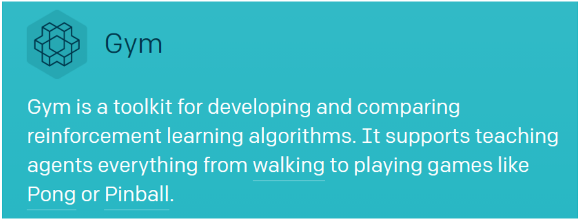
RL Environments

- Google Research Football Environment
- ViZDoom
- TextWorld

  ...



https://research-football.dev/
http://vizdoom.cs.put.edu.pl/
https://textworld.readthedocs.io/en/latest/notes/framework.html

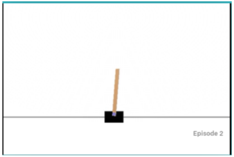


- Open source interface to reinforcement learning tasks
- Gym library is a collection of test problems — environments, with shared interfaces
- Compatible with existing numerical computation libraries and deep learning frameworks
- Customized environments!

https://gym.openai.com/

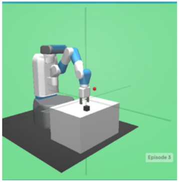
Environments

Atari 2600



Classic control

MuJoCo



Robotics

OpenAI Gym

Requirements

Python 3.5+

Installation:
`pip install gym`

Running example: interaction with an env

```
import gym
env = gym.make('CartPole-v0')
env.reset()
for _ in range(1000):
    env.render()
    env.step(env.action_space.sample()) # take a random action
env.close()
```

https://gym.openai.com/docs/

OpenAI Gym

- Environment versions
- Environment horizons - episodes
- `env.step()` vs $P(s'|s, a)$

Q: Can we record a video of the rendered environment?

Implementation: Q-learning

Algorithm: Q-learning

Parameters: step size $\alpha \in (0, 1]$, $\epsilon > 0$ for exploration

1. Initialise $Q(s, a)$ arbitrarily, except $Q(\text{terminal}, \cdot) = 0$
2. Choose actions using Q, e.g., $\epsilon -$ greedy.
3. On each time step

$$Q^{new}(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(R_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$

4. Repeat step 2 and step 3

If desired, reduce the step-size parameter α over time

Algorithm: ϵ-greedy

Parameters: small $\epsilon \in (0, 1)$

$$a = \begin{cases} \arg\max_a Q(s, a), \text{with probability } 1 - \epsilon \\ \text{a random action, with probability } \epsilon \end{cases}$$

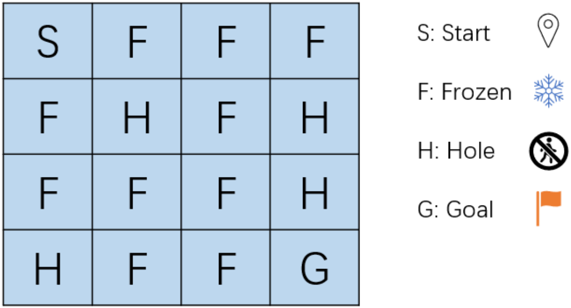
Q table

actions

states

	a_0	a_1	$\cdots$
s_0	$Q(s_0, a_0)$	$Q(s_0, a_1)$	$\cdots$
s_1	$Q(s_1, a_0)$	$Q(s_1, a_1)$	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Env: FrozenLake-v0



`is_slippery=True` by default

Other Environments

FrozenLake: discrete state space, discrete action space

Can we use Q-learning to solve other environments, e.g., CartPole?

Discretization