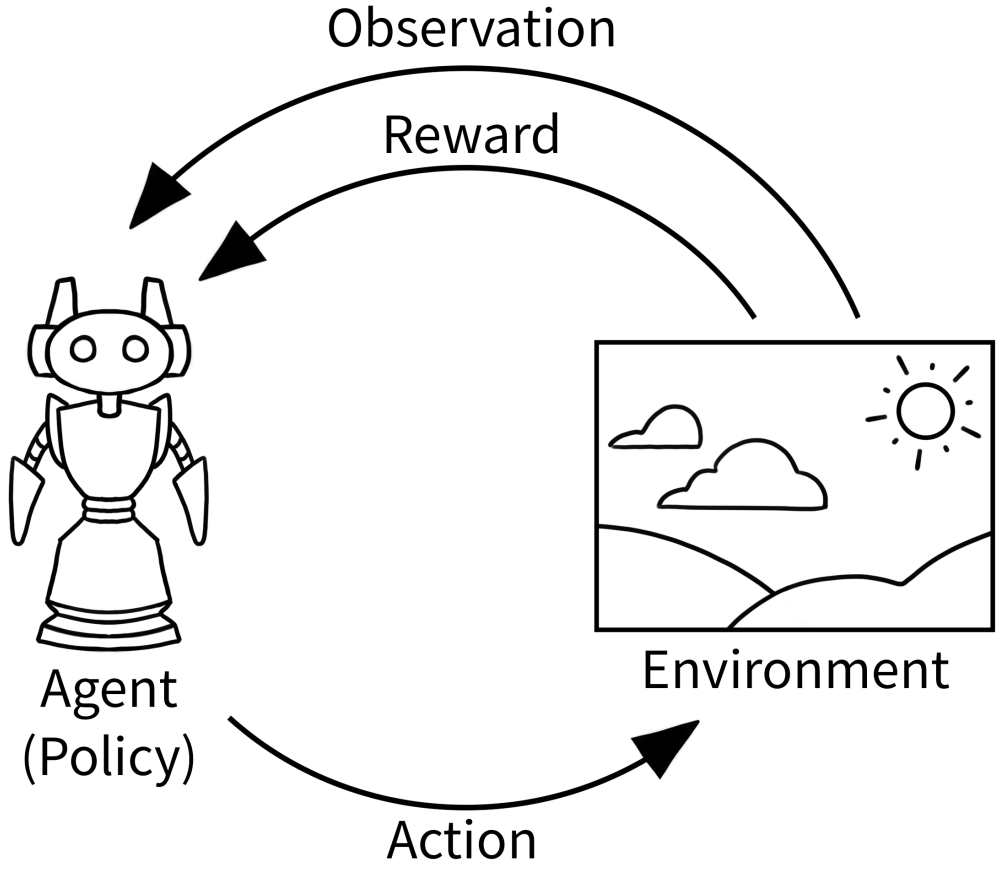
2024-01 INF-2600 AI Methods & Applications

Assignment 2: Reinforcement Learning

Reinforcement Learning – Learning from interaction

- Reinforcement Learning ~ Science of decision making
- There is no supervisor, only a *reward* signal
- In RL an agent learns from the experiences it gains by interacting with the environment.
- The goal is to maximize an accumulated reward given by the environment.
- An agent interacts with the environment via states, actions and rewards.



RL: A subfield of Machine Learning

(from Machine Learning course, 2011, Marc Toussaint)

- *Supervised* learning: learn from "labelled" data $\{(x_i, y_i)\}_{i=1}^{N}$
  Unsupervised learning: learn from "unlabelled" data $\{x_i\}_{i=0}^{N}$ only
  Semi-supervised learning: many unlabelled data, few labelled data

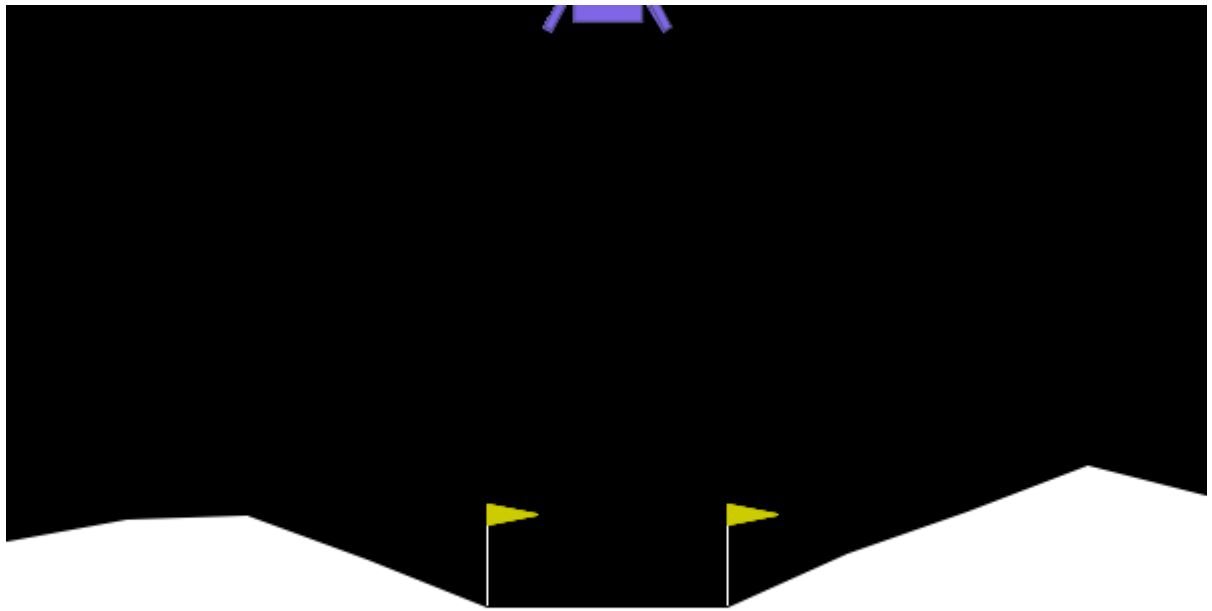
- *Reinforcement* learning: learn from data $\{(s_t, a_t, r_t, s_{t+1})\}$
  – learn a predictive model $(s, a) \rightarrowtail s'$
  – learn to predict reward $(s, a) \rightarrowtail r$
  – learn a behavior $s \rightarrowtail a$ that maximizes the expected total reward

- Some RL Types:
  – Q-Learning
  – DQN
  – SARSA

Gym/Gymnasium

- An API standard for reinforcement learning with a diverse collection of reference environments

```
import gymnasium as gym

env = gym.make("LunarLander-v2", render_mode="human")

observation, info = env.reset()

for _ in (1000):

    action = env.action_space.sample() # agent policy that uses the observation and info

    observation, reward, terminated, truncated, info = env.step(action)

    if terminated or truncated: observation, info = env.reset() env.close()
```



Env

gymnasium.Env

`class gymnasium.Env` [source]

The main Gymnasium class for implementing Reinforcement Learning Agents environments.

The class encapsulates an environment with arbitrary behind-the-scenes dynamics through the `step()` and `reset()` functions. An environment can be partially or fully observed by single agents. For multi-agent environments, see PettingZoo.

The main API methods that users of this class need to know are:

- `step()` - Updates an environment with actions returning the next agent observation, the reward for taking that actions, if the environment has terminated or truncated due to the latest action and information from the environment about the step, i.e. metrics, debug info.
- `reset()` - Resets the environment to an initial state, required before calling step. Returns the first agent observation for an episode and information, i.e. metrics, debug info.
- `render()` - Renders the environments to help visualise what the agent see, examples modes are “human”, “rgb_array”, “ansi” for text.
- `close()` - Closes the environment, important when external software is used, i.e. pygame for rendering, databases

Environments have additional attributes for users to understand the implementation

- `action_space` - The Space object corresponding to valid actions, all valid actions should be contained within the space.
- `observation_space` - The Space object corresponding to valid observations, all valid observations should be contained within the space.
- `reward_range` - A tuple corresponding to the minimum and maximum possible rewards for an agent over an episode. The default reward range is set to $(-\infty, +\infty)$.
- `spec` - An environment spec that contains the information used to initialize the environment from `gymnasium.make()`
- `metadata` - The metadata of the environment, i.e. render modes, render fps
- `np_random` - The random number generator for the environment. This is automatically assigned during `super().reset(seed=seed)` and when assessing `self.np_random`.

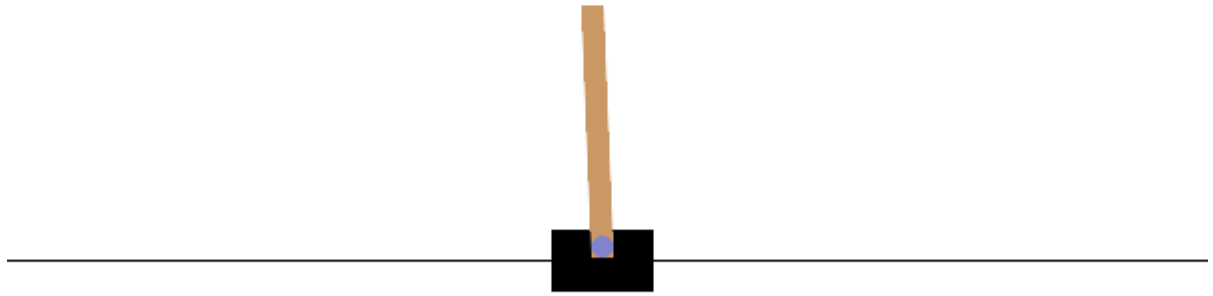
Play with gym Environment

Task 1 - CartPole - V1

A pole is attached by an un-actuated joint to a cart, which moves along a frictionless track.

Goal is to balance the pole by applying forces in the left and right direction on the cart.

Num	Observation	Min	Max
0	Cart Position	-4.8	4.8
1	Cart Velocity	-Inf	Inf
2	Pole Angle	~ -0.418 rad (-24°)	~ 0.418 rad (24°)
3	Pole Angular Velocity	-Inf	Inf

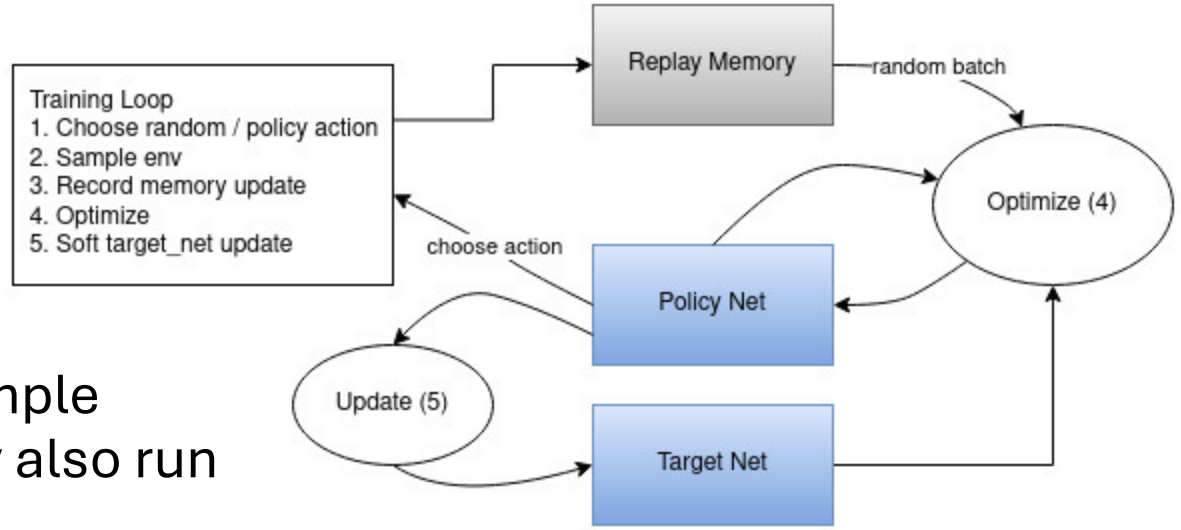


Python Environment Setup

- Tested for python version 3.10.5. Expected to be compatible for higher versions of Python also.
- pip install (package versions for python 3.10.5):
  - `gymnasium==0.27.1`
  - `jupyterlab==3.6.1`
  - `torch==2.2.0`

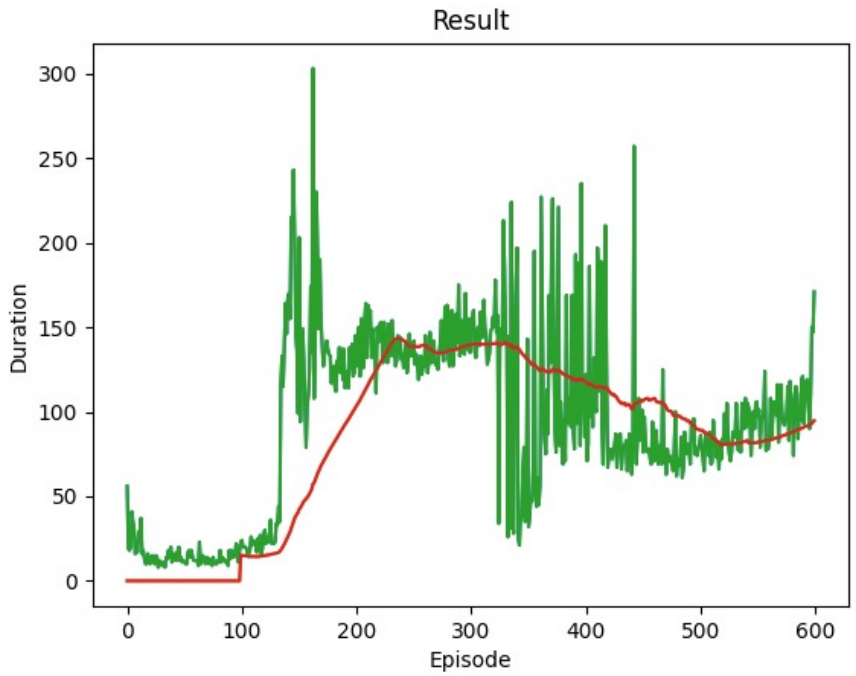
- You may use tensorflow as well. But most keras based RL packages are deprecated with python version > 3.9. So stick to base tensorflow APIs.

Task 1 - CartPole-V1



Read, understand, and run the CartPole-V1 example notebook provided for Deep Q learning. You may also run the code in Google Colab.

1. Why do we need a policy net and target net?
2. When do you think a model converges?
3. Observe the effects of changing the exploration vs exploitation. (Show results in plots or print statements)
4. Vary the number of layers in the model and plot episode vs duration plots. Maybe good to keep an eye on how long!
5. Implement without the replay buffer and observe performance.



Code Setup

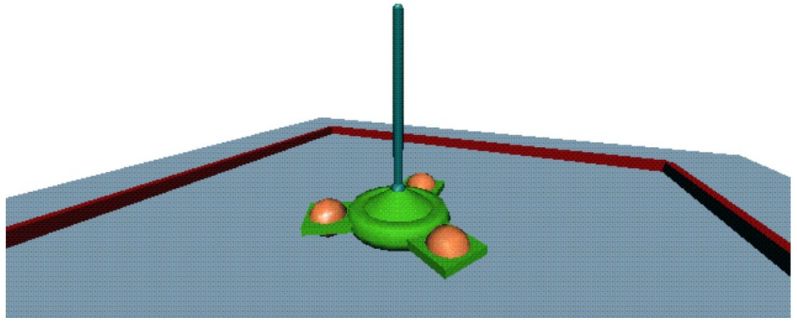
Assignment 2

CartPole 2D

Task 2 - `CartPole2DEnv`

- Use the precode provided in the file “Assignement_2_task_2.py” and repeat the sub-tasks listed for Task 1. Adjust the number of episodes to match your compute needs

- Note, there is no render function provided in the pre-code. See the figure below to get an idea.



TASK 3

- Implement Q-learning and SARSA for the `CartPole-V1` Environment.

Bonus Task

- Implement any other RL algorithm on either of the environments.

Additional Resources

- *David Silver's lecture: http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html*
- *Youtube series https://www.youtube.com/playlist?list=PLqYmG7hTraZBiG_XpjnPrSNw-1XQaM_gB*
- *Richard S. Sutton, Andrew Barto: Reinforcement Learning: An Introduction. The MIT Press Cambridge, Massachusetts London, England, 1998. http://webdocs.cs.ualberta.ca/sutton/book/the-book.html*

Thank you