Introduction to Reinforcement Learning

Recitation: OpenAI Gym, Bandits & More

Ayoub Ajarra

20 de noviembre de 2024

About me

- Second year PhD student at Inria - Scool team.
- Working on learning theory and sequential decision making, Trustworthy ML, Fairness, Privacy ...

You can reach me via email at ayoub.ajarra@inria.fr

Course Logistics

- Grading:
  - 2 Homework assignments: implementation and extra grade questions - (60 %, will be specified later)
  - 3 quizzes - (40 %, will be specified later)
- Resources: will be uploaded on the website: "https://debabrota-basu.github.io/course_bandit_rl.html"

About Programming Assignments

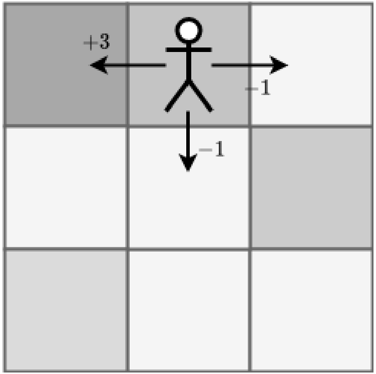
- Projects will be in Python (knowledge of programming in Python will be considered a prerequisite).
- Roughly a short programming exercise, with some technical questions, but it may be adjusted.
- Implement core algorithms in RL (See Richard Sutton Book).

You're a reinforcement learner

Imagine you represent an agent that interacts with an environment (A room):

- Actions: left, right, up, down.
- Observations: Colors of tiles after you step in.
- Goal: Find an optimal policy (by selecting actions that get you the highest reward).

What does the environment look like ?



What is your policy?

Formalism

- Known to the agent:
  - Observations $\mathcal{O} = \{o_1, o_2, \cdots\}$
  - Actions $\mathcal{A} = \{a_1, a_2, \cdots\}$
  - Rewards after taking actions

$o_0, a_0, r_0, o_1, a_1, r_1, o_2 \cdots$

- Unknown to the agent:
  - Environment: $\mathcal{S} = 3 \times 3$ grid.
  - Reward function: $\mathcal{R} : \mathcal{A} \times \mathcal{O} \to \mathbb{R}$
  - State transition function (model): $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{O}$

$s_0, o_0, a_0, r_0, s_1, o_1, a_1, r_1, s_2, o_2 \cdots$

$$r_i = \mathcal{R}(a_i, o_i)$$

$$o_i = \mathcal{T}(s_i, a_i)$$

The Big Picture

Learning can be:

- Supervised: Learn from labeled examples.
- Unsupervised: Cluster unlabeled examples.
- Reinforced: Learn by interaction.

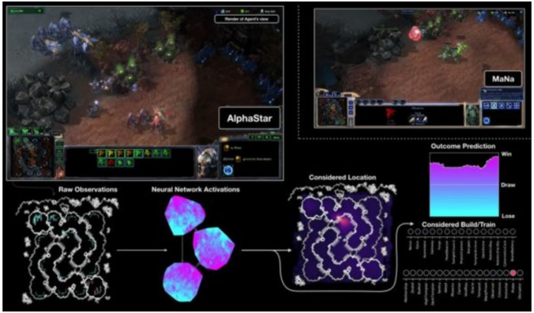
Trial and Error Learning

Setting & Success of Reinforcement Learning

What can RL do ?

AlphaGo (Silver et al. 2016)

More Games



AlphaStar (Vinyals et al. 2019)

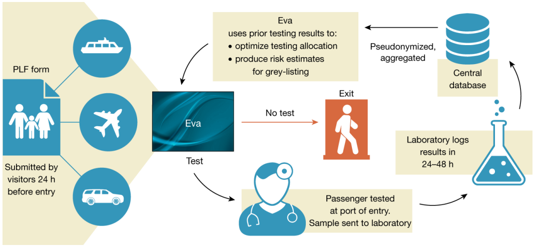
More Games

OpenAI Five (OpenAI et al. 2019)

More Games

Atari games (Mnih et al. 2015)

Beyond Games



Covid-19 testing allocation (Bastani et al., 2021)

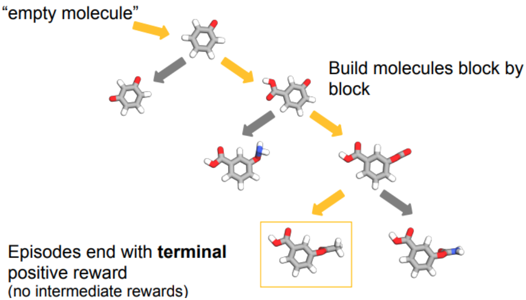
Recent/Future Successes: Exa-Scale Search for Molecules

The goal is to find drugs that bind to protein(s). The search space is enormous, with more than 10^{16} to 10^{20} possible molecules (simplified and for one protein). Most molecules are "bad":

- Not chemically feasible
- Not binders
- Toxic

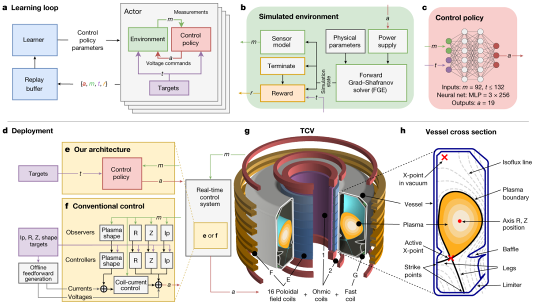
This search is like looking for a needle in a haystack.

Molecule Search as Reinforcement Learning



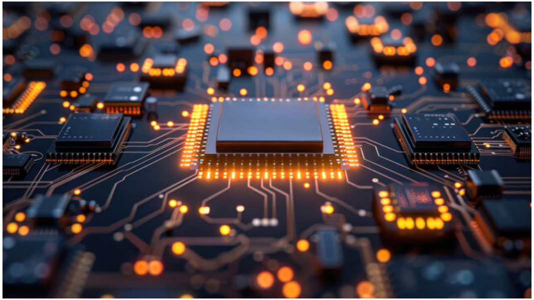
Bengio et al, NeurIPS2021

Beyond Games



Nuclear fusion control (Degrave et al., 2022)

Beyond Games

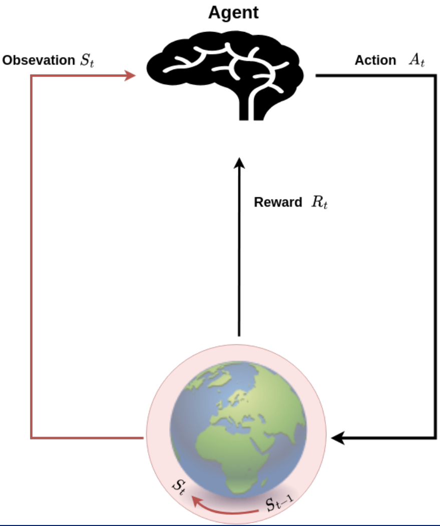


Chip design (Mirhosenini et al., 2021)

Beyond Games

Navigation of stratospheric balloons (Bellemare et al., 2021)

What is Reinforcement Learning ?



Interacting with an environment

Reinforcement $\iff$ Learning from interaction with the environment.

- As interaction occurs, information about cause and effect is obtained.
- Such knowledge should be exploited to achieve the goal faster.

Human-inspired analogous situation: baby learning to walk.

- Tries many actions and observe consequences (rewards).
- Encodes cause and effect for future actions.
- Eventually becomes able to walk by applying successful actions.

Delayed Reward

Most Reinforcement Learning problems involve multiple timesteps: At each time step, the agent takes some action and obtains a reward.
An action that is taken at a given step can influence not only the current reward but also subsequent states and subsequent rewards.
Rewards associated with an action are not always observable, some are delayed.

Return

Goal-seeking behavior of an agent can be formalized as the behavior that seeks maximization of the expected value of the cumulative sum of (potentially time-discounted) rewards, we call it return.

Exploitation Vs Exploration

One of the oldest and remaining open problems in RL is the exploration vs exploitation trade-off:

- To obtain high rewards, the learner must take actions which were identified successful $\implies$ Exploiting current knowledge of the environment.
- To identify such actions, the agent has to try new actions never taken before $\implies$ Exploring new situations

To be successful (achieve the goal fast), neither exploitation nor exploration can be exclusively pursued. The agent should find a balance.

Key features of RL

- The learner is not told what actions to take, instead it finds out what to do by trial-and-error search
  - Eg. Players trained by playing thousands of simulated games, with no expert input on what are good or bad moves
- The environment is stochastic
- The reward may be delayed, so the learner may need to sacrifice short-term gains for greater long-term gains
  - Eg. Player might get reward only at the end of the game, and needs to assign credit to moves along the way
- The learner has to balance the need to explore its environment and the need to exploit its current knowledge
  - Eg. One has to try new strategies but also to win games

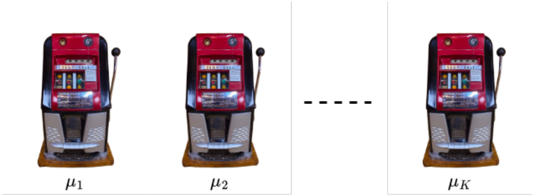
centralelille
ÉCOLE CENTRALE DE LILLE

Multi-Armed Bandits

A Simple Setting For Learning by trial and error

K Armed Bandits

- At each time step t the agent chooses one of the K arms and plays it.
- The k th arm produces reward $r_{k,t}$ when played at timestep t.
- The rewards $r_{k,t}$ are drawn from a probability distribution $\mathcal{D}_k$ with mean μ_k.



The agent does not know the arm rewards distributions or their means.
Agent's objective: Maximize cumulative rewards (over a finite or infinite horizon).

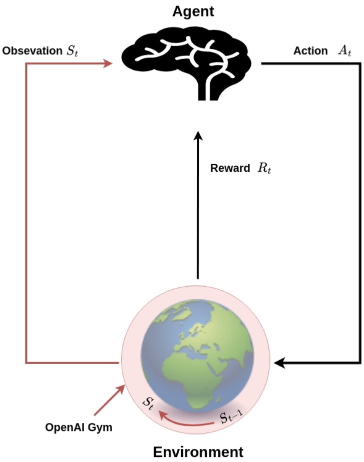
Quiz

Instructions

Join the quiz using the link: ‘https://quizizz.com/join?gc=88087856 ‘

Introduction to OpenAI Gym

RL framework



What is OpenAI Gym?

- A toolkit for testing RL algorithms on simplified examples.
- Provides you with different environments.
- Has a standard API to access these different environments.

Why OpenAI Gym?

- Open-source - large online community, you can modify the source code to fit your needs.
- Intuitive API - easy to get started.
- Widely used in RL research - a common benchmark for RL papers.

Why OpenAI Gym?

To solve an RL problem, one needs the ability to:

- Define the environment
- Generate samples from the environment
- Sample an action from the action space
- Retrieve the next state after taking an action
- Retrieve the reward of taking an action
- Check if the episode has ended
- Reset the episode when the episode ends

OpenAI gym gives you the ability to do all the things.

OpenAI Gym - Basic Concepts

- Define the environment ‘env = gym.make(MountainCar-v0)‘
- Sample an action from the action space ‘action = env.action_space.sample()‘
- Reset the episode when the episode ends ‘state = env.reset()‘
- Retrieve the next state, reward, and the indicator of the episode termination: ‘next_state, reward, done, info = env.step(action)‘

OpenAI Gym - Additional Concepts

- Render the environment ‘env.render()‘
- Record the environment ‘env = gym.wrapper.Monitor(env, ., force=True)‘
- Check out the state space and action space: ‘Print (env.action_space)‘ and ‘Print (env.observation_space)‘

Actions & States/Observation spaces

- OpenAI gym environments have predefined states and actions, but ...
- Key questions to ask when debugging RL in general:
  - Does the state and action space make sense?
  - What is the reward?
  - Are there any constraints on state and action space?
  - Should constraint violations be penalized?
  - When should an episode terminate? Am I handling termination correctly?

Hands-On Session: A simple example

Lets look at a demo.

Next Practical Session

Questions ?