Multi-Agent Reinforcement Learning: Avoiding Tragedy of the Commons

Based partially on work by Ariel Kwiatkowski – Ben Greenberg – Quinn Dougherty

Talk by Quinn Dougherty

Quinn Dougherty

Logician

Platonic.Systems

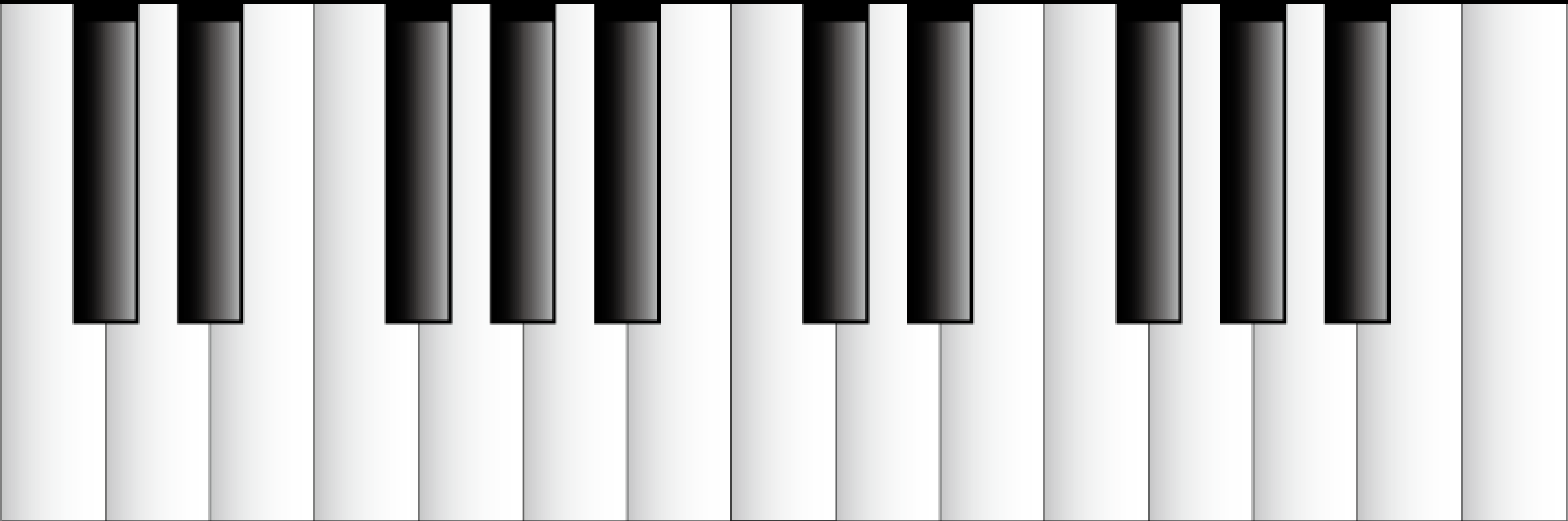
https://calendly.com/quinn-d/

: /in/quinn-dougherty

quinnd@tutanota.com

: quinn#9100

https://quinnd.net



Quinn Dougherty
Logician
Platonic.Systems

Where I’m coming from

- Was a musician and producer. Did scores, ops/logistics, scripts, etc. for films and theater pieces as well as free improv here in philly (that was at least 5 years ago)
- Math major at Community College of Philadelphia (‘16-’18)
- Lambda School’s first data science cohort, later TA (2019)
- Python, cloud ops, security at a startup in 2020
- Participated in AI Safety Camp 5 in 2021, https://aisafety.camp
- Research intern at Stanford Existential Risks Initiative
- Logician (auditor, formal verification engineer) at a consultancy validating a decentralized finance product (Cardano)

Quinn Dougherty
Logician
Platonic.Systems

Where I’m coming from

Motivation:

- Computers ought to be good for civilization
- AI advances could be ***really really bad*** or at least suboptimal
- What leverage does research have in 2021?

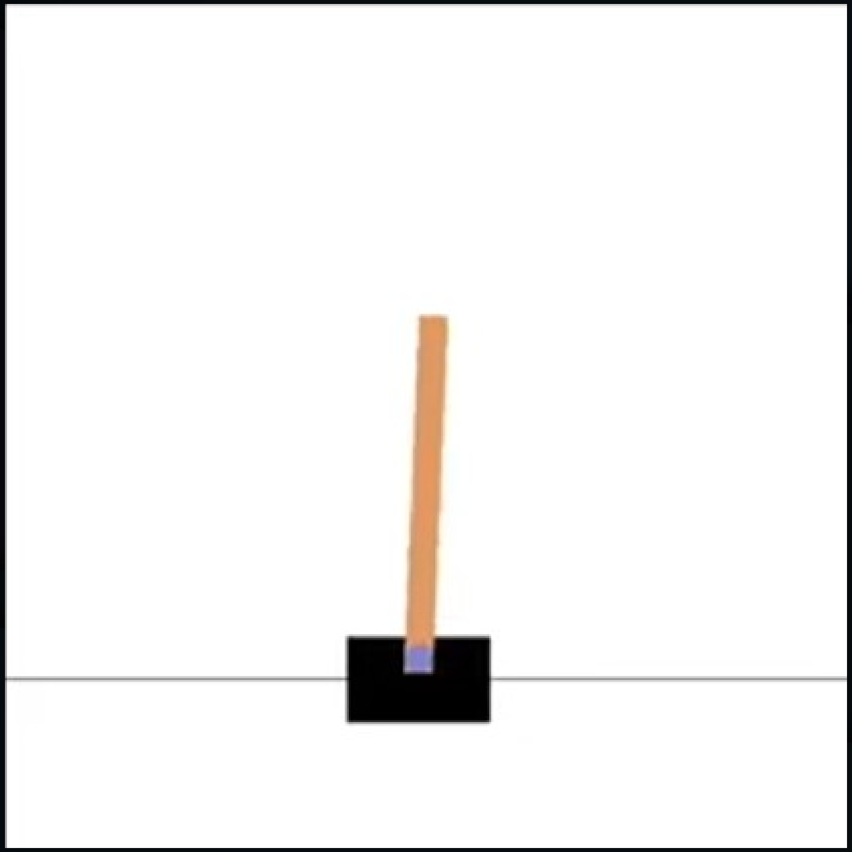
Agenda

- What is reinforcement learning (RL)?
- What is a common-pool resource problem?
- What is multi-agent reinforcement learning (MARL)?
- How easy is it to spin up with Ray and RLLib?
- Code examples tailor made for the talk: https://github.com/quinn-dougherty/marl-cpr-talk see all the hyperlinks I’m going to share in `BIBLIO.md`
- Research code (harder to read): https://github.com/redTachyon/cpr_reputation

Reinforcement learning

- Represent environments as **states**, **actions**, and **rewards**.
- We “solve” these environments by (selecting actions to) **maximize reward**.
- The study of these “solvers” (maximizers) is called *reinforcement learning*.

Reinforcement learning



Reinforcement learning

Cartpole:

- States: angle, horizontal position
- Actions: change in angle, change in horizontal position
- State transition function: physics simulator
- Reward function: large negative reward for pole dipping beneath “ground” line, 0 or small positive reward for pole staying up

Reinforcement learning – the gym paradigm

```
class MyEnv(gym.Env):
    def __init__(self, env_config):
        self.action_space = <gym.Space>
        self.observation_space = <gym.Space>
    def reset(self):
        return <obs>
    def step(self, action):
        return <obs>, <reward: float>, <done: bool>, <info: dict>
```

Reinforcement learning

```python
from typing import Tuple

import gym  # type: ignore
from gym.spaces import Discrete  # type: ignore


class GuessingGame(gym.Env):
    def __init__(self, env_config):
        self.length = self.remaining_rounds = env_config["length"]
        self.n = env_config["n"]
        self.observation_space = Discrete(self.n)
        self.action_space = Discrete(self.n)
        self.previous_obs = None

    def reset(self) -> int:
        self.remaining_rounds = self.length
        self.previous_obs = self.observation_space.sample()
        return self.observation_space.sample()

    def step(self, action: int) -> Tuple[int, float, bool, dict]:
        self.previous_obs = self.observation_space.sample()

        self.remaining_rounds -= 1

        return (
            self.observation_space.sample(),
            float(action == self.previous_obs),
            self.remaining_rounds <= 0,
            {}
        )
```

Reinforcement learning

GuessingGame:

- **states** = **actions** = {0, 1}
- **State transition function** totally_pseudo_random : {0, 1} → {0, 1}
- **Reward function** : states x actions → {0, 1} such that R(s, a) = 1.0 if s == a else 0.0

Reinforcement learning – learnit in ray

```python
#!/usr/bin/env python3

import ray  # type: ignore
from ray.rllib.agents import ppo  # type: ignore
from dataphilly import GuessingGame, RockPaperScissors

if __name__ == "__main__":
    ray.init()
    trainer = ppo.PPOTrainer(
        env=GuessingGame,
        config={
            "env_config": {"length": 1000, "n": 10},
            "framework": "torch"
        }
    )

    while True:
        print(trainer.train())
```

Reinforcement learning – learnit in ray

(navigate out of slides and open up CLI)

Common pool resource problems (CPRs)

Definition matrix [edit]

	Excludable	Non-excludable
Rivalrous	**Private goods** food, clothing, cars, parking spaces	**Common-pool resources** fish stocks, timber, coal, free public transport
Non-rivalrous	**Club goods** cinemas, private parks, satellite television, public transport	**Public goods** free-to-air television, air, national defense, free and open-source software

Common pool resource problems (CPRs)

- CPRs form an approximate prisoner’s dilemma (PD)
- A PD is type of “game” where each of two players can make one of two moves (per round) and there are payouts (numbers).
- PDs are distinguished by the property that the **most stable tendency is toward the worst outcome for each player**.

Common pool resource problems (CPRs)

Prisoner B / Prisoner A	Prisoner B stays silent (*cooperates*)	Prisoner B betrays (*defects*)
Prisoner A stays silent (*cooperates*)	Each serves 1 year	Prisoner A: 3 years Prisoner B: goes free
Prisoner A betrays (*defects*)	Prisoner A: goes free Prisoner B: 3 years	Each serves 2 years

A “game” can be thought of like a matrix of tuples:
[[(-1, -1), (-3, 0)],
 [(0, -3), (-2, -2)]]

Common pool resource problems (CPRs)

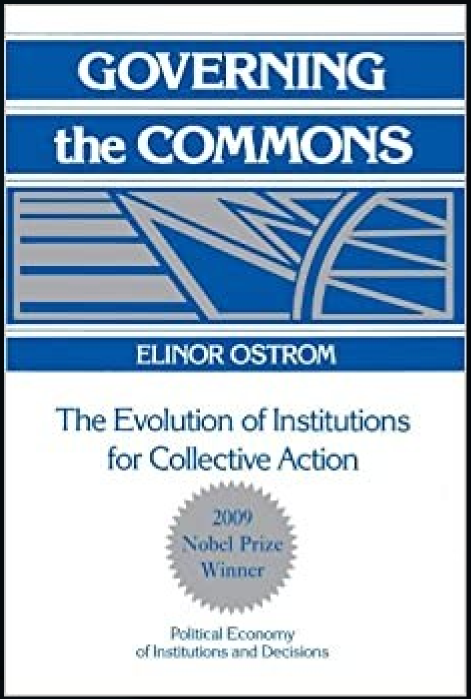
Exercises:

- Convince yourself that (-2, -2) is the most stable outcome even though (-1,-1) has the highest sum reward
- Imagine lake with fish and 10 fishers making a living catching them. In what sense is a PD a useful model?

Common pool resource problems (CPRs)

- The tendency toward PD-like nash equilibria caused economists to believe that CPRs aren’t great, calling this outcome **tragedy of the commons**
- Since we’re living in a post-Elinor Ostrom world, this is considered something of a myth.

Common pool resource problems (CPRs)



https://quinnd.net/blog/tragedy/

Common pool resource problems (CPRs)

Ostrom’s principles for **sustainable, stable, non-tragedy solutions** to CPRs:

1) Clearly defined boundaries
2) Congruence between appropriation and provision rules and local conditions
3) Collective-choice arrangements allowing for the participation of most of the appropriators in the decision making process
4) Effective monitoring by monitors who are part of or accountable to the appropriators
5) Graduated sanctions for appropriators who do not respect community rules
6) Conflict-resolution mechanisms which are cheap and easy to access
7) Minimal recognition of rights to organize (e.g., by the government)
8) In case of larger CPRs: Organisation in the form of multiple layers of nested enterprises, with small, local CPRs at their bases.

https://en.wikipedia.org/wiki/Common-pool_resource#Common_property_protocols
https://quinnd.net/blog/tragedy/

Common pool resource problems (CPRs)

Insights I got from the book:

- skin-in-the-game principle; do not offload the design process to people who aren’t directly involved
- local knowledge, stuff about the environment an administrator would not know
- **reputation** plays a role

https://en.wikipedia.org/wiki/Common-pool_resource#Common_property_protocols
https://quinnd.net/blog/tragedy/

Multi-agent RL

```python
def __init__(self, env_config):
    self.length = self.remaining_rounds = env_config["length"]
    self.n = env_config["n"]
    self.items = partial(self.Item, modulus=self.n)  # Some atransitive comparator
    self.num_agents = self.n - 1
    self.observation_space = Box(-1, 1, (self.num_agents,), int)
    self.action_space = Discrete(self.n)
    self.previous_obs = None

def reset(self) -> ndarray:
    self.remaining_rounds = self.length
    return array([0] * self.num_agents)

def step(
        self,
        actions_dict: Dict[str, int]
) -> Tuple[Dict[str, ndarray], Dict[str, float], Dict[str, bool], dict]:
    actions = OrderedDict((agent_id, self.items(k=k)) for agent_id, k in actions_dict.items())
    rewards = {
        agent_id: sum(
            actions[agent_id] << actions[other_agent_id]
            for other_agent_id
            in actions.keys()
        )
        for agent_id
        in actions.keys()
    }
    self.remaining_rounds -= 1

    isdone = self.remaining_rounds <= 0
    done = {agent_id: isdone for agent_id in actions_dict.keys()}
    done["__all__"] = isdone

    # each agents' observation is an array of the score it got in each position.
    # so other agents are ordered in a sense.
    observations = {
        agent_id: array([actions[agent_id] << action for action in actions.values()])
        for agent_id
        in actions.keys()
    }

    return (
        observations,
        rewards,
        done,
        {}
    )
```

Multi-agent RL

N-ary RockPaperScissors:

- States: An array of what everyone did in the previous round relative to you
- Actions: rock, paper, or scissors (generalized. i.e. discrete)
- State transition function: each agent’s decision!
- Rewards: {-1, 0, 1} assigned in a zero-sum fashion across agents.

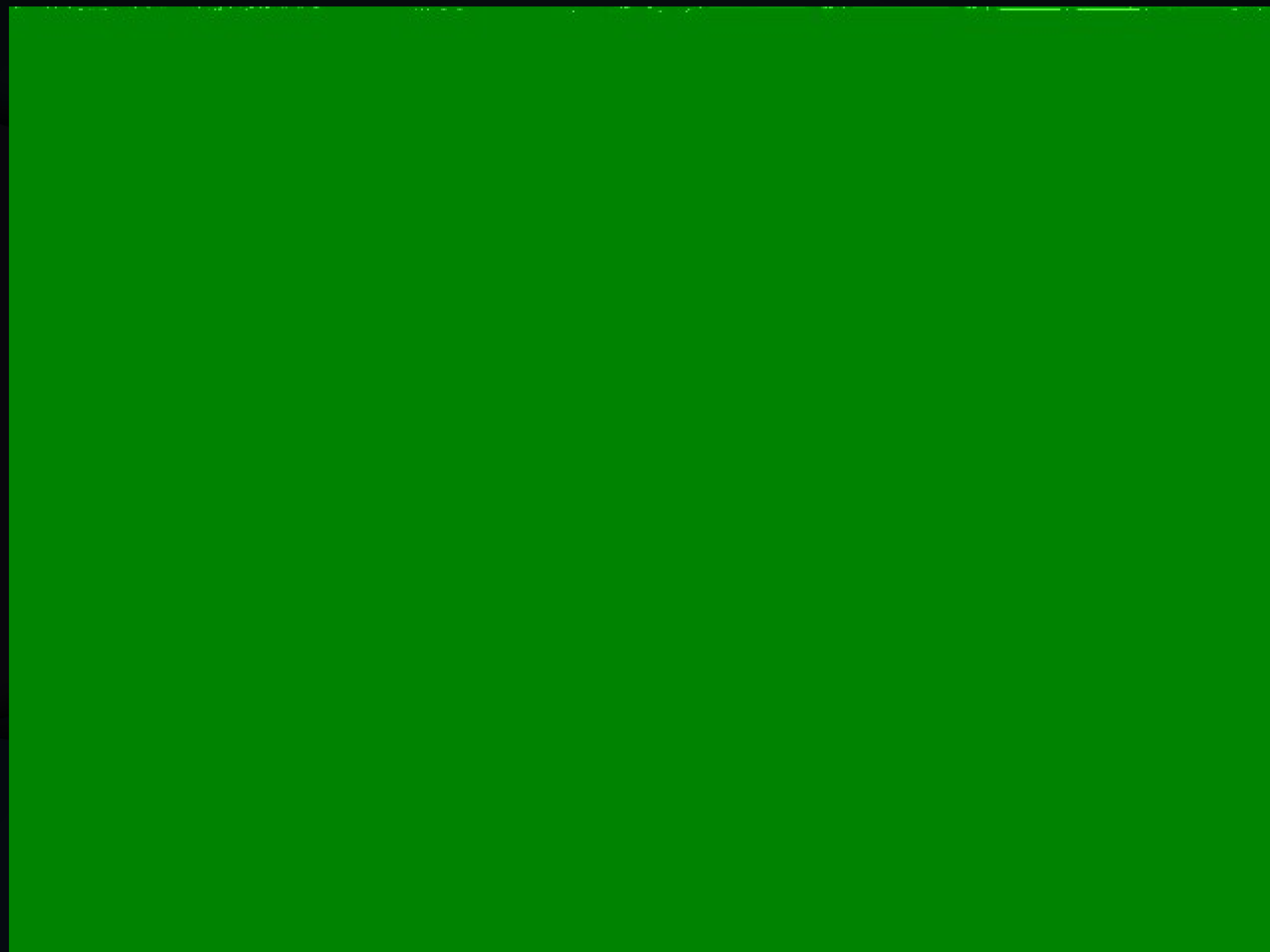
Multi-agent RL

(navigate out of slides and open up CLI)

CPR reputation research

- Joint work with Ariel Kwiatkowski and Ben Greenberg at AI Safety Camp 5
- Research question (derived from my reading of Ostrom): does MARL simulate tragedy of the commons? **Can we design/simulate a mechanism to dodge tragedy of the commons**?
- I won’t show you the code because there’s a lot of it, but it’s on github for you to read later.
- Not the first people to ask the basic CPR, ToC, and MARL question: deepmind has a few papers.
- We implemented a gridworld game about collecting apples which replenish for arbitrary number of agents.
- We had metrics like “sustainability” and “fairness”

CPR reputation research



(open up actual file from term if this doesn’t work)

CPR reputation research

Our findings:

- ...not much interesting!
- Our mechanism didn’t seem to do anything (not even gonna show you before/after vids).
- Did we not represent it adequately for the agents to act on it?
- Was our representation adequate but our hypothesis wrong?

CPR reputation research

Our findings:

- Deepmind paper introduced a “tagging” mechanism, which we replicated.
- Reputation decreased the utilization of tagging mechanism
- Reputation was negligible on sustainability and fairness
- Graphs of metrics against control group in our lesswrong post https://www.lesswrong.com/posts/LBwpubeZSi3ottfjs/aisc5-retrospective-mechanisms-for-avoiding-tragedy-of-the

Conclusions

- Ray is surprisingly easy! You can turn your ideas into experiments in a reasonable time frame
- Science is about not always getting the results you want
- High-impact research is hard

Quinn Dougherty
Logician
Platonic.Systems

Thank you:

To DataPhilly; to FOSS projects libreoffice, python-on-nix, and Ray; to Wikipedia; and to AI Safety Camp 5

Quinn Dougherty
Logician
Platonic.Systems

https://calendly.com/quinn-d/

: /in/quinn-dougherty

: quinn#9100

quinnd@tutanota.com

https://quinnd.net