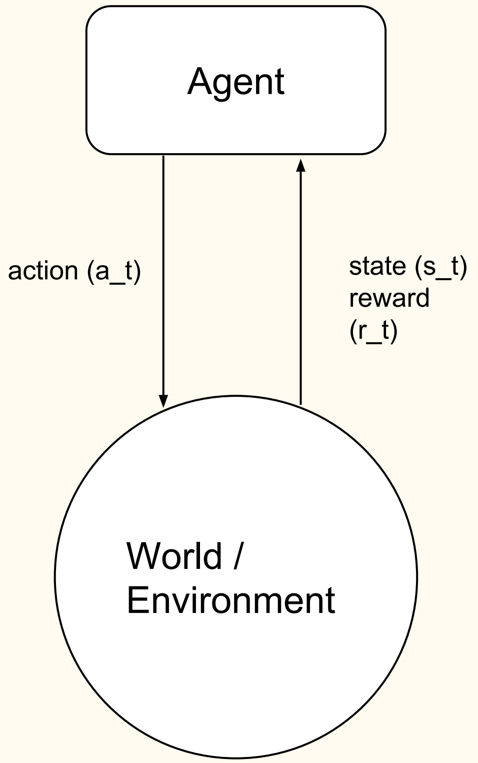
Robotic Reinforcement Learning

Bryce Wiley and Cannon Lewis

Reinforcement Learning

An Abstraction

Markov Decision Process =



More Precisely

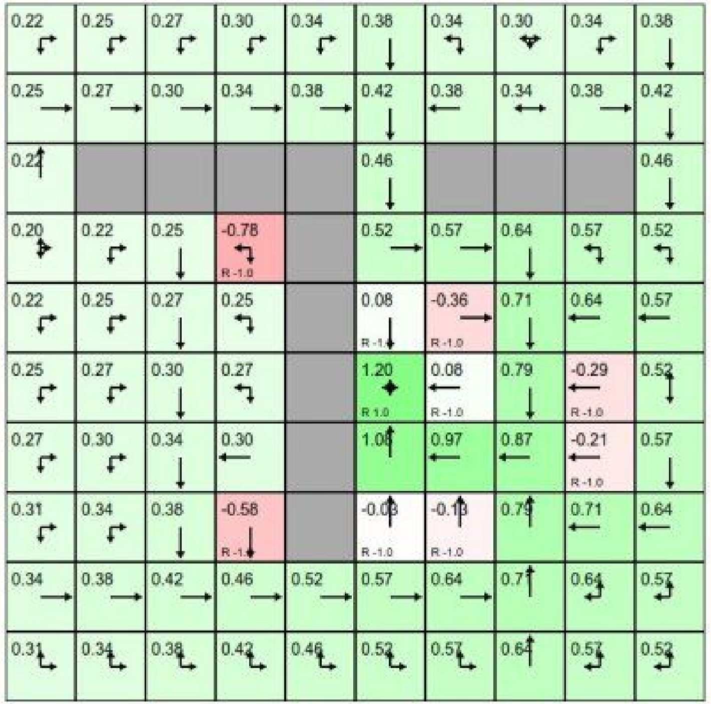
Markov Decision Processes (MDPs) are the fundamental theoretical underpinning of reinforcement learning. They consist of:

- A set of states S
- A set of actions A
- A stochastic transition function $P_a(s, s')$
- A reward function $R_a(s, s')$

The Goal

Find a policy $\pi(s, a)$ specifying the optimal action to take in a given state.

- Easier in discrete spaces, where exact solutions are tractable.
- Policies can be deterministic or stochastic.
- If using function approximation, a policy is often represented by its parameters, θ.



http://cs.stanford.edu/people/karpathy/reinforcejs/index.html

Value Functions

$$V^{\pi}(s) = E_{\pi}\left\{R_t | s_t = s\right\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right\}$$

$$Q^{\pi}(s,a) = E_{\pi}\left\{R_t | s_t = s, a_t = a\right\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\right\}$$

Bellman Equations

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s'} P_a(s,s') \left[R_a(s,s') + \gamma V^{\pi}(s')\right]$$

$$Q^{\pi}(s,a) = \sum_{s'} P_a(s,s') \left[R_a(s,s') + \gamma V^{\pi}(s')\right]$$

Types of Algorithms

Algorithms are largely divided up by what they attempt to optimize.

- Model-free methods: focus on V^π or Q^π.
- Model-based methods: develop a model of the world, optimize π via planning.
- Policy gradient methods: directly optimize π by evaluating the current policy and making rough steps.

Deep Reinforcement Learning

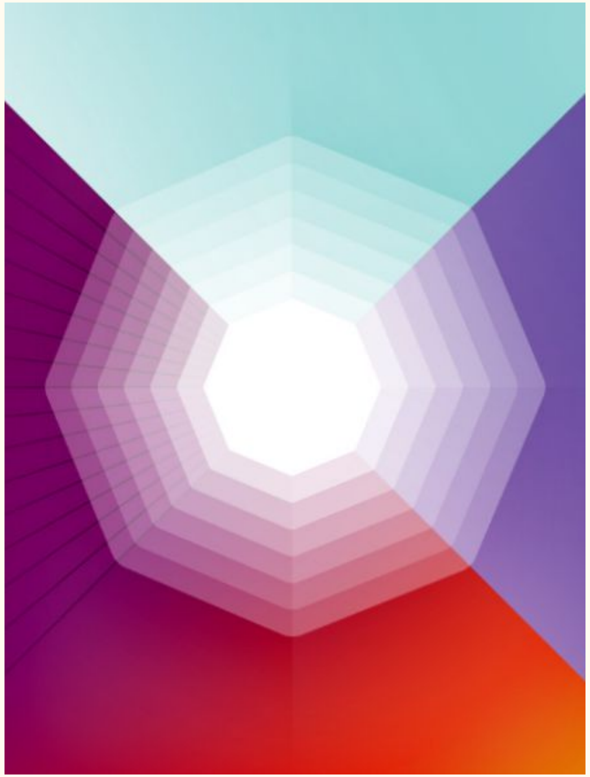
Algorithms (typically value-function based) which use a neural network as a value function approximator.

- Examples are DQN, TRPO, DDPG, A3C, etc.
- Much better than traditional RL approaches at handling continuous state and action spaces.
- Responsible for most recent high-profile advances in RL.

OpenAI Gym & ROS

OpenAI

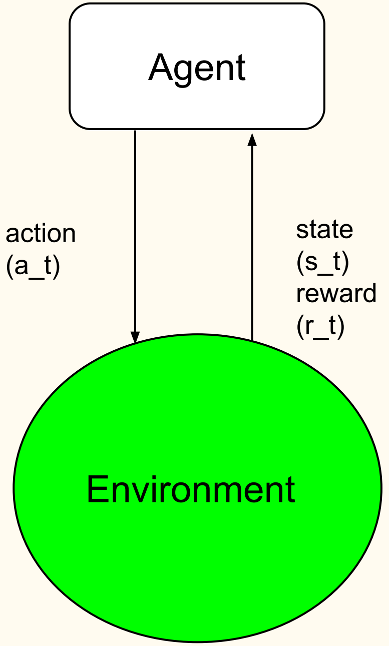
- General AI Research
  - RL and Unsupervised Learning mostly
- 4 Technical Goals
  - Measure progress (Gym)
  - Build a household robot (but probably not)
  - Build an agent with useful natural language understanding
  - Solve a wide variety of games using a single agent
- Work Fast, Shift Priorities?

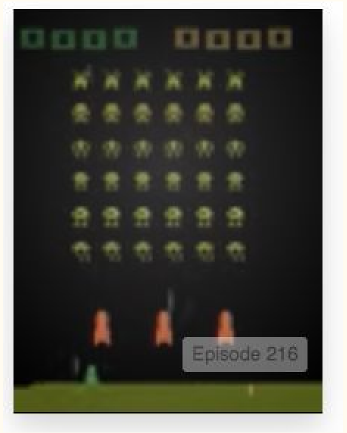


OpenAI Gym

- A set of RL environments used for benchmarks and standardization
- Classic Control Environments (Pendulum)
- Atari Games (Space Invaders)
- MuJoCo Physics Simulations
  - 2D Hopping, Humanoid Running
  - Recently ported to Bullet

- **But nothing for actual robots**





ROS Gym

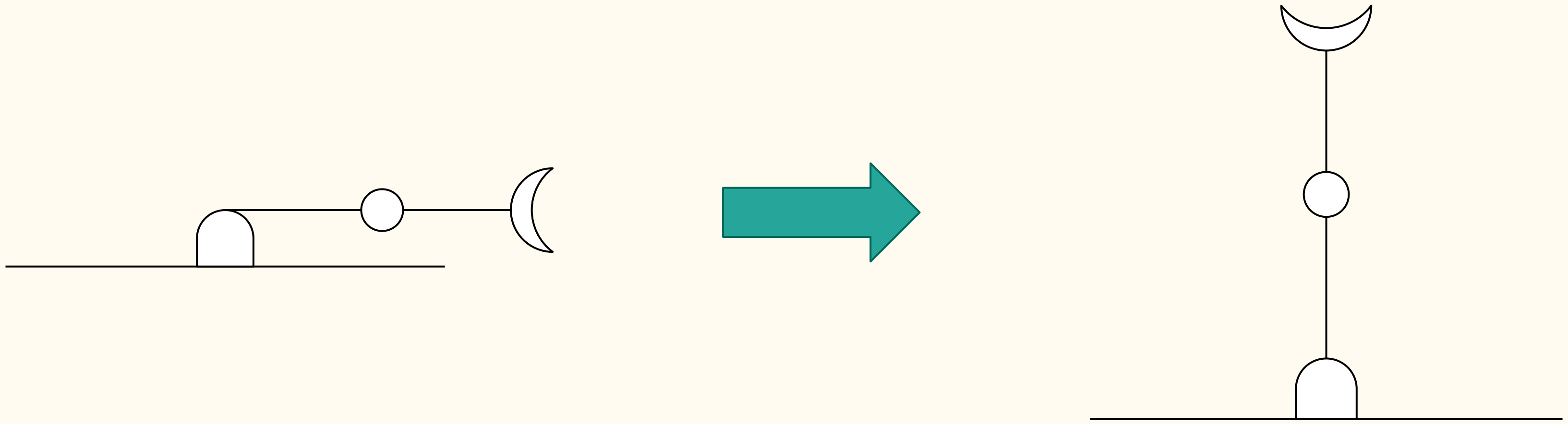
- https://github.com/KavrakiLab/rosgym
- Make ROS accessible to agents using the OpenAI Gym API
- Actions and States are ROS nodes
  - Can run learning in the Gazebo simulator or on a real robot (not very safe)
- Two parts:
  - Robot Configuration:
    - Joints to control (action space)
    - Sensors + joint states (state space)
  - Reward Function: depends on the task you want the robot to accomplish

ROS Gym Demonstration

First Experiments and Pathologies

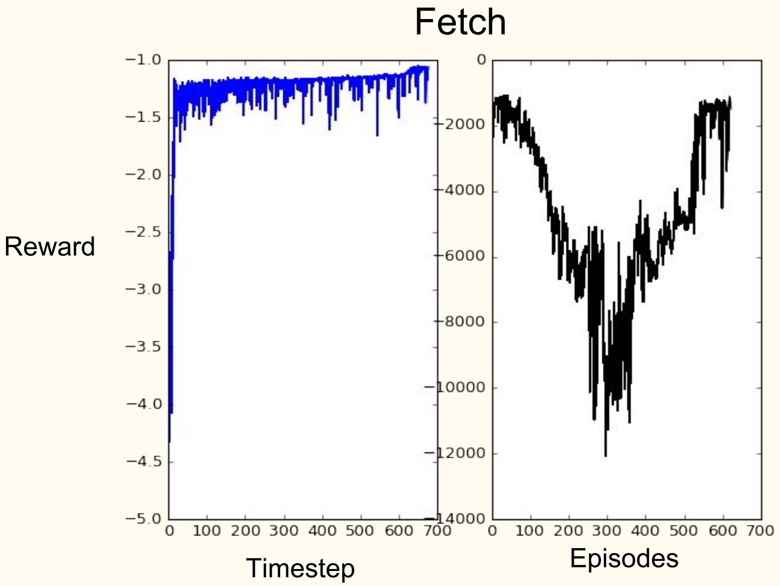
A First Task

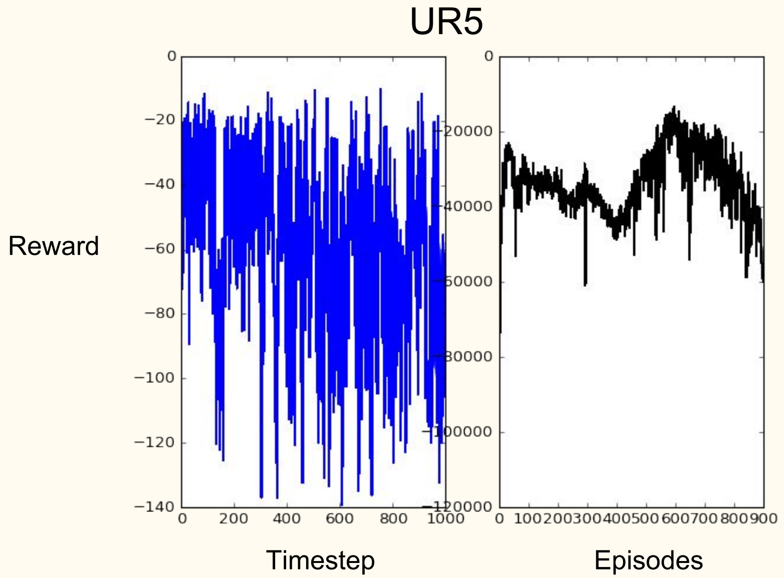
As a simple first task, we wanted to have a reinforcement learning algorithm learn to control from a fixed start position to a fixed end position, using either Cartesian or joint space distance to form a reward function.



Complications

However, it turns out that robotic reinforcement learning is quite difficult, due to the curse of dimensionality, continuity of state and action spaces, and a dearth of research directly applying RL to robots.





Pathologies

As part of the evidence that robotic RL is intractable under naive application, we observe several pathologies in training resulting from local minima in the reward function for our simple task.

Moving Forward: Motion Planning

A Problem

In the pathologies we observed for the initial task, many are connected to exploration.

- Exploration - Exploitation is a classic problem in RL.
- For most RL methods and domains, simply adding Gaussian or time-correlated noise to action selection suffices to induce exploration.
- However, robots present a much more difficult environment, and demand more complex methods.

Some Background

Existing approaches to robotic RL make one or more simplifying assumptions.

- Policies may be initialized via “imitation learning”.
- State or action spaces may be discretized, or reduced to certain “motor primitives”.
- Model-based RL algorithms may be used, essentially incorporating prior knowledge about a robot’s dynamics.

A Hypothesis

The above methods are not desirable, as they do necessarily generalize well. However, sampling-based motion planning is a powerful, general tool in robotics that could aid in robot RL by enabling structured exploration in a theoretically justified way.

OpenAI Gym BETA

ROS

OMPL

A First Task

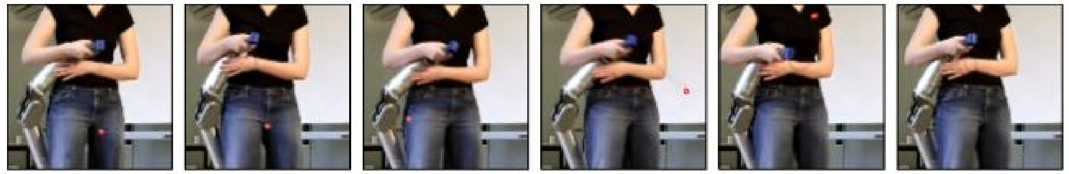
As a first task to demonstrate this framework, we will generalize from the simple, naive initial task to general reaching.

- Attempt to control from an arbitrary initial position to an arbitrary final position.
- If a smooth policy for this task can be learned without imitation learning, it will show that exploration by motion planning is sufficient to overcome previous difficulties in robot RL.
- Gateway to more complex tasks such as manipulation.

Moving Forward: Preference Learning

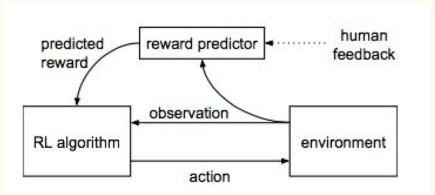
Human Specified Tasks

- Simple Reward: -1 * (dist(pos, goal) $+ \alpha_1$ * vel $+ \alpha_2$ * acc)
- Writing rewards for non-trivial tasks is difficult
  - Unwieldy, complex equations
  - Reward Hacking
  - Wire-heading
- Many approaches
  - Imitation Learning
  - Apprentice Learning
  - Inverse Reinforcement Learning
  - Preference Learning


(b) Kinesthetic teach-in

Preference Learning ⟷ Inverse RL

- Learns a reward function based on preference pairs: trajectory x is better than (or equal to) y
- Sparse information, but easier for non-experts to provide



?

- Middle ground
- More expressive
- Less intensive

- Learns a reward function based on optimal trajectories
- Lots of information, but more difficult to provide for non-anthropomorphic robots

First Task

- Replicate/implement previous preference learners

- Create a platform-specific way to provide more expressive feedback to the reward predictor
    - N-query preference
    - Preference + Trajectory Modification

- Generalize key points of this feedback

Questions?