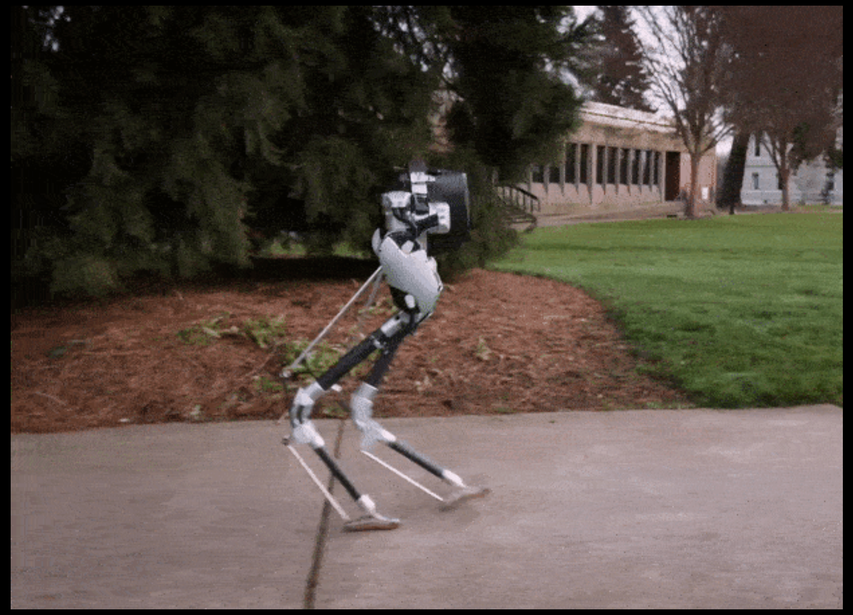
A Brief Introduction to **Reinforcement Learning**

Bryce MacInnis



A "delivery ostrich" named Cassie that learned to walk.

Vocabulary

- Agent
- Policy
- Environment
- Trajectory
- State
- Action

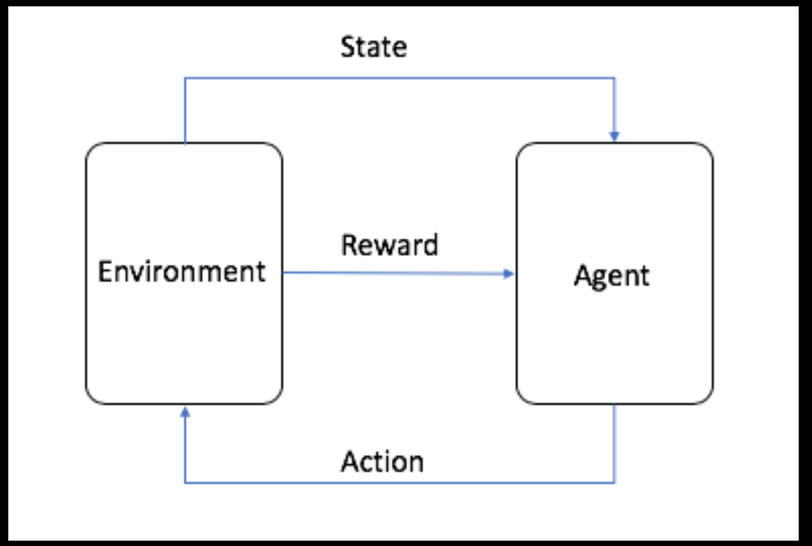
Machine Learning

Computers should be able to solve problems without being explicitly programmed how to do it.

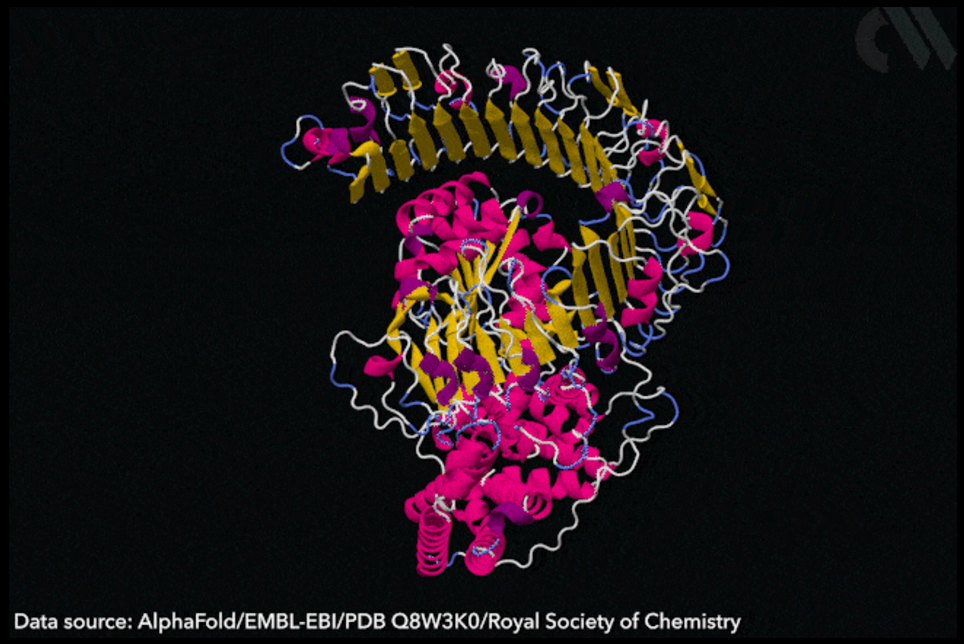
Reinforcement Learning

Computers should be able to learn how to play games, make decisions, how to strategize and how to control things

Reinforcement Learning



Major Successes

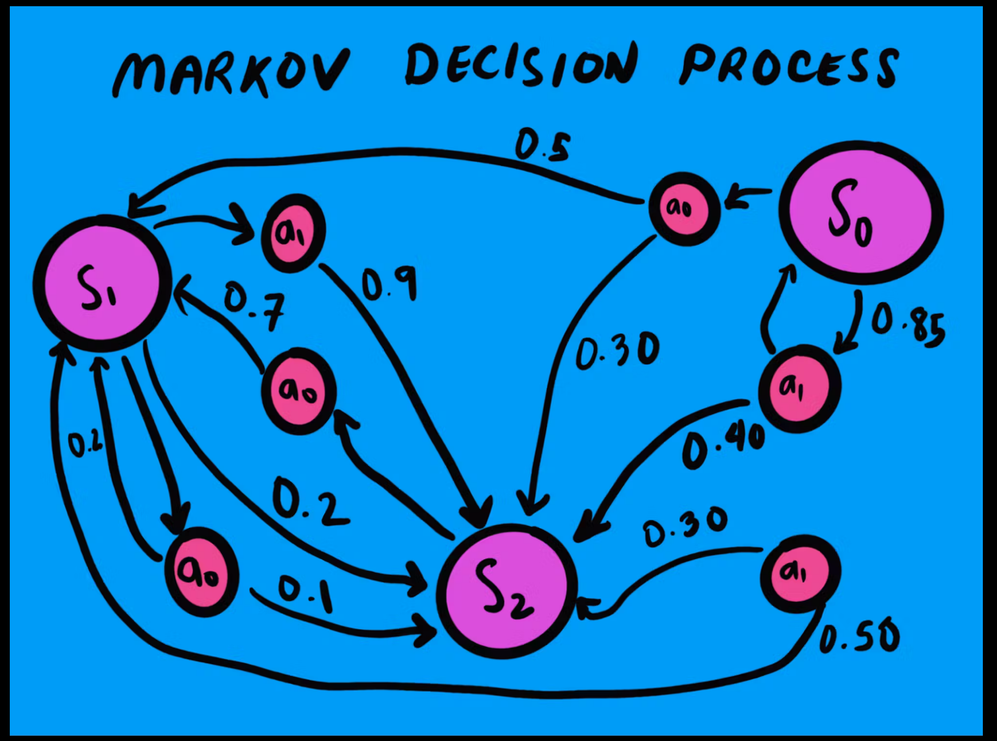


Data source: AlphaFold/EMBL-EBI/PDB Q8W3K0/Royal Society of Chemistry

Markov Decision Processes

Reinforcement Learning are modelled with Markov Decision Processes

Formally, they are 4-tuples (S, A, R, T(S->A))



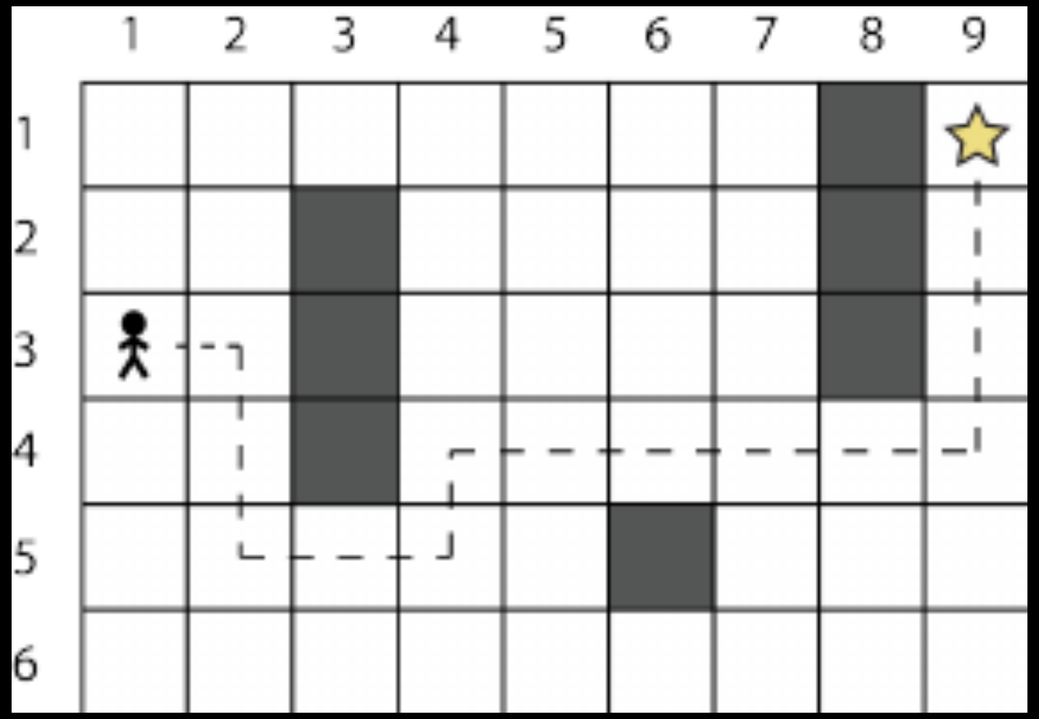
Value Estimation

Goal state is +100 reward

-10 penalty for hitting a wall

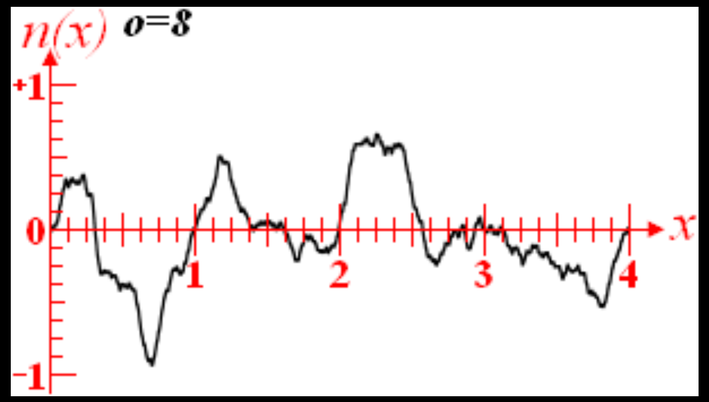
"The Bucket Brigade"

Fire Bucket Brigade by Monroe Historical Society

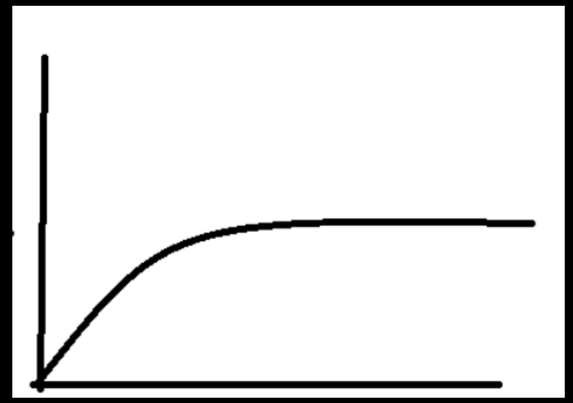


Temporal-Difference

Or how to bootstrap a Value Function

- 

VS



A good value function is a **consistent** value function.

The Bellman Equation

$$V(s) = \max_a \left[R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V(s') \right]$$

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s'} P(s' \mid s,a) \max_{a'} Q^*(s',a')$$

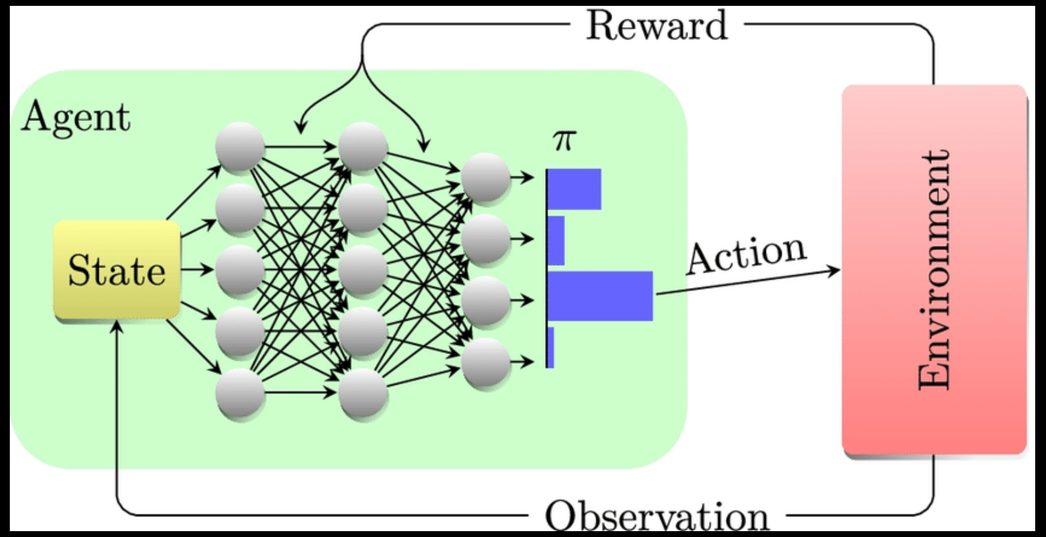
Exploration vs Exploitation Dilemma

Should the agent exploit what it knows gives good reward, or explore actions it hasn’t tried that might be even better?

What about non-stationary environments?

Deep Q-Learning

Using Neural Networks trained through Gradient Descent



Genetic Programming

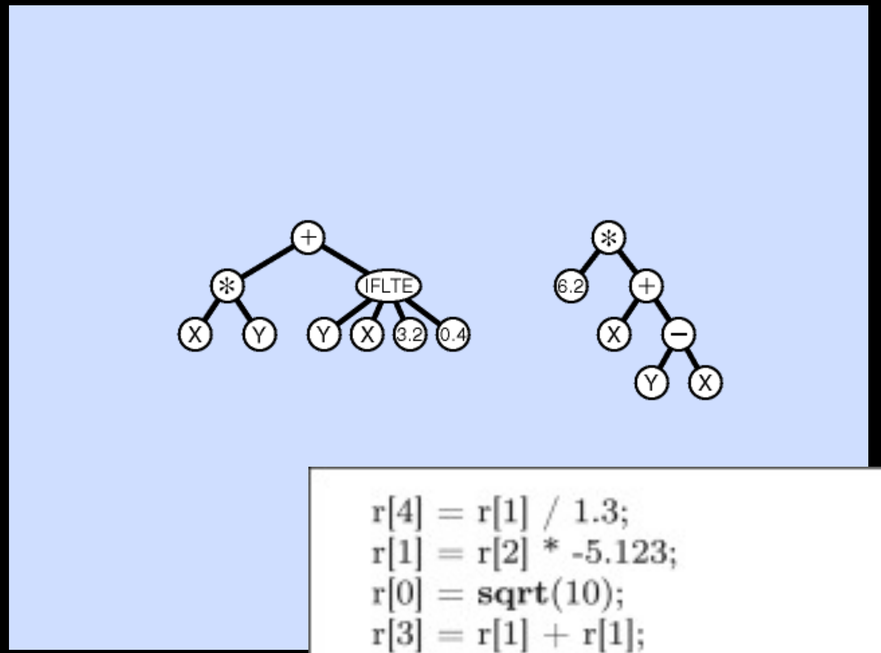
Start with a population of candidate solutions

Decide which subset of the population gets to remain by measuring fitness

Apply variation operators (Mutation and Crossover) to the remaining population.

Replace the missing individuals with mutated individuals

Loops for many generations until performance is good or converges




```
r[4] = r[1] / 1.3;
r[1] = r[2] * -5.123;
r[0] = sqrt(10);
r[3] = r[1] + r[1];
r[1] = log(r[3]);
r[1] = r[3] >= r[0];
r[0] = abs(r[2]);
r[0] = if r[1] then r[0] else r[3];
```

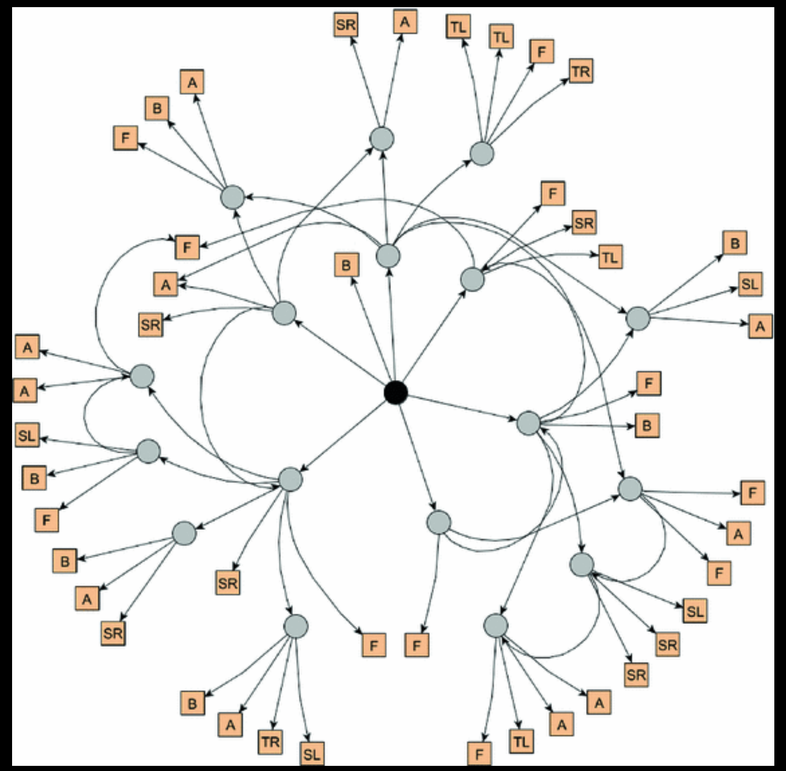
Tangled Program Graphs

Extension of Genetic Programming

Similar to how the brain is developed (has specialized regions)

Network of programs are dynamically learned

Competitive with Deep Q-Learning (DQN) at the Atari Learning Environment



CMA-ES

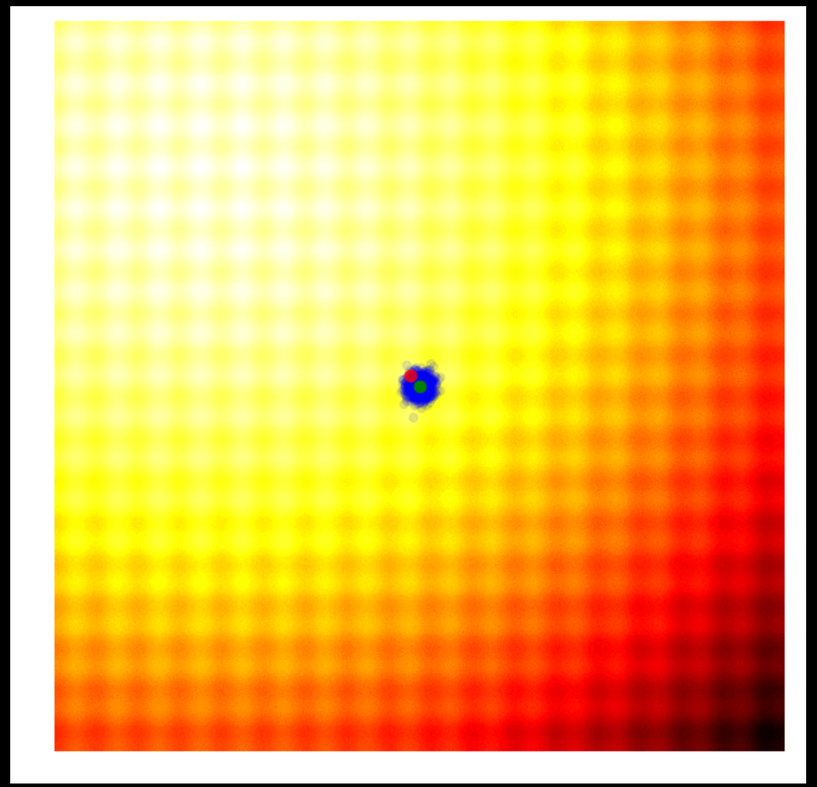
Neuroevolution through CMA (Covariance Matrix Augmentation)

Uses the standard Neural Network architecture: Multilayer Perceptron

Uses CMA to adjust weights instead of Gradient Descent

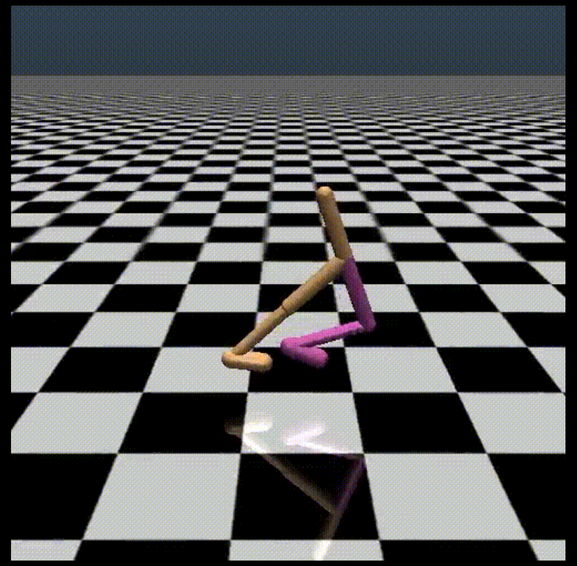
Less likely to get stuck in local optima.

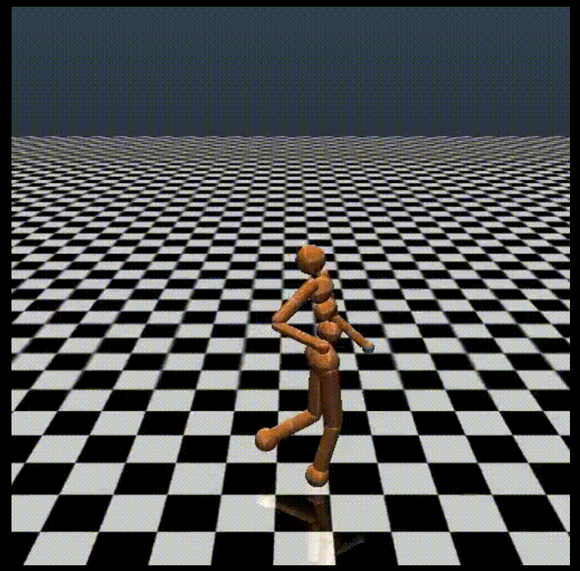
Does not assume differentiability.



Policy Gradients

- What about continuous environments?
- Requires an additional neural network to predict the action (Actor network)
- Same as before, another neural network says how good the action is (Critic network)
- State-of-the-art: TD3 and SAC (Soft-Actor Critic)





Off-policy vs On-policy

- All about data efficiency
- On-policy throws away the data after the agent has made the action
- Off-policy tries to conserve the data by storing it in a buffer.

Model Safety

- Sometimes doing the wrong action can be disastrous.
- One method to mitigate this is Trust-Region Policy Optimization
- (A.k.a. don't do things that you aren't confident about doing).
- Most famous RL algorithm is PPO.

Algorithm Review

- DQN: Earliest attempt to use computer vision + neural networks to solve Reinforcement Learning tasks in high-dimensional environments.
- Genetic Programming / TPG: Evolutionary Algorithms: alternatives to neural networks that emphasize interpretability
- Neuroevolution: Neural networks trained through optimization methods other than Gradient Descent
- TD3 and SAC: Off-policy methods that are considered state-of-the-art in RL.
- PPO: On-policy method that guarantees models improve with time and mitigates dangerous actions.

Challenges

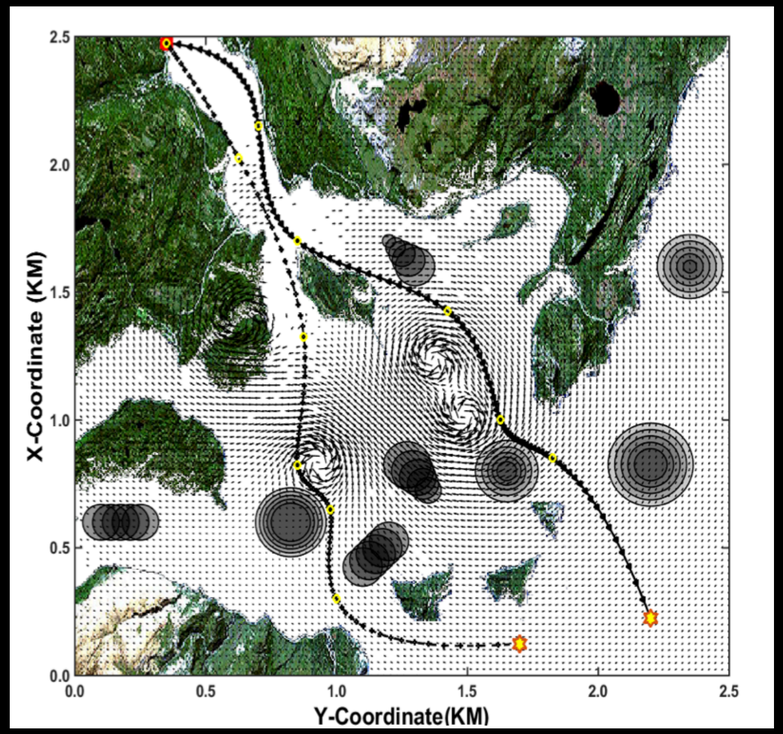
- Interpretable solutions (Deep Reinforcement Learning is a black-box method)
- Sample efficiency
- Model safety
- Difficult to design reward functions
- Difficult to simulate
- Sparse rewards

Hot Areas of Research

- Model-based Reinforcement Learning
- Offline Reinforcement Learning
- Multi-agent Reinforcement Learning
- Multi-task Reinforcement Learning
- Interpretable Reinforcement Learning
- Multi-objective Reinforcement Learning

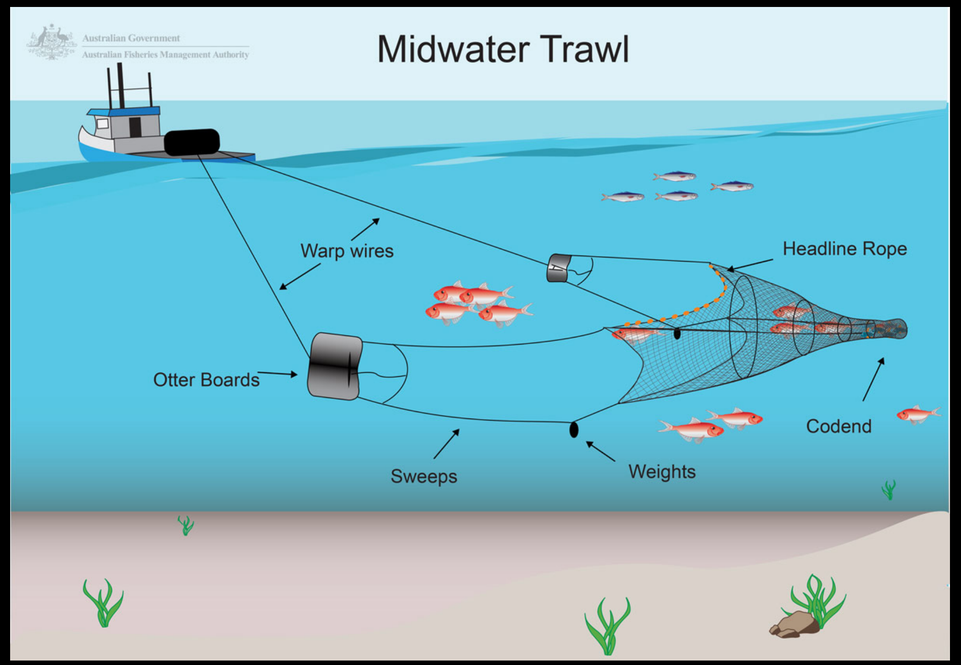
Autonomous Underwater Vehicles (AUVs)

- Model-based Reinforcement Learning
- Objective: Learn a complete model of the environment
- Reward: % Area covered, Penalty for proximity to hazards



Learning to control a trawling net

- Smarter trawling systems
- Reduce bycatch
- Increase desired catch
- Reward: +1 per positive catch, -1 for bycatch.
- Domains: Online Reinforcement Learning, Offline Reinforcement Learning

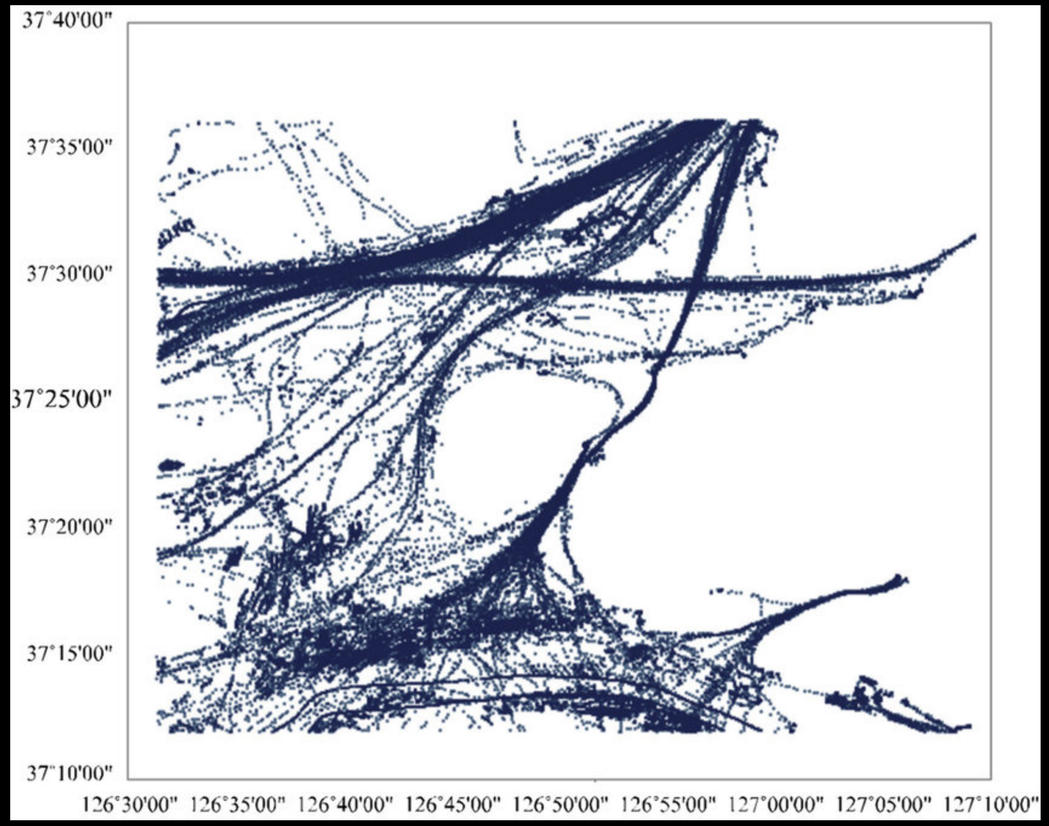


Wave and Tidal Energy

- Underwater currents are a great source of power but they are also dynamic, complicated environments.
- Turbine blades need to be adjusted (rotated) to the correct orientation.
- Tradeoff between generating enough power and not damaging the turbine.

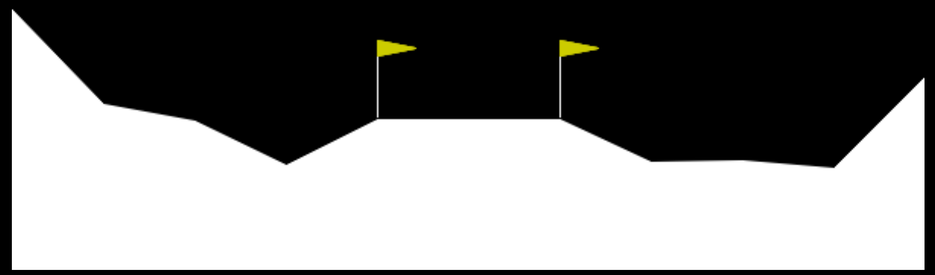
Illegal Fishing Monitoring

- Plan routes for patrol ships to detect illegal fishing
- Reward for detecting suspicious activity
- Penalty for amount of fuel used, distance of trajectory
- Multi-objective: Reduce the amount of travel, increase the amount of coverage.



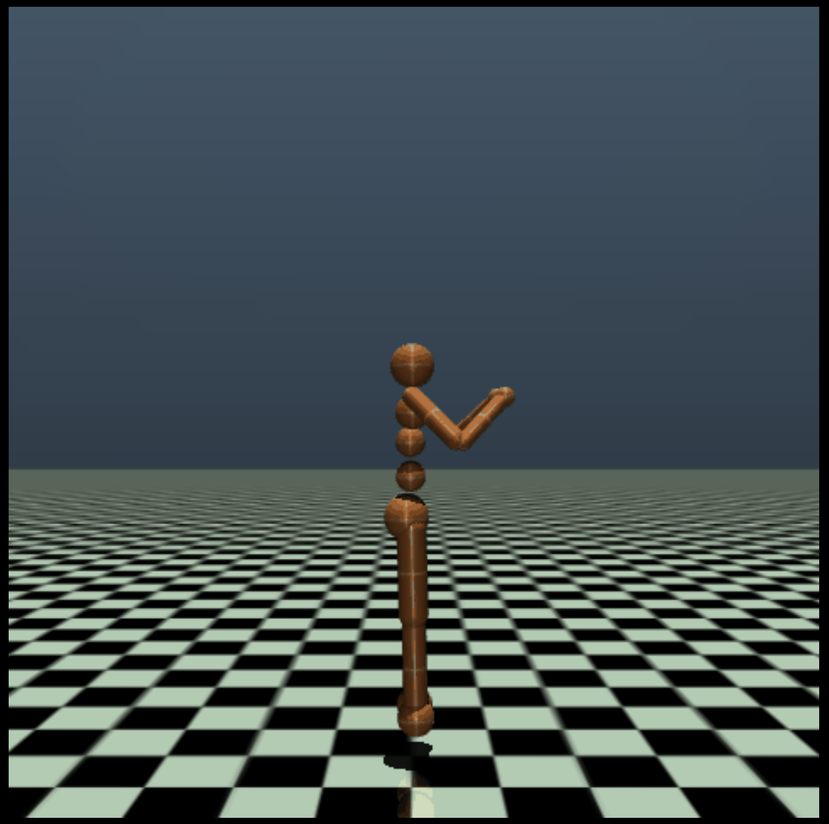
Gymnasium Environments

- Released by OpenAI to encourage Reinforcement Learning development
- Maintained by non-profit Farama Foundation
- Easy-to-use API
- Common benchmarks in RL papers

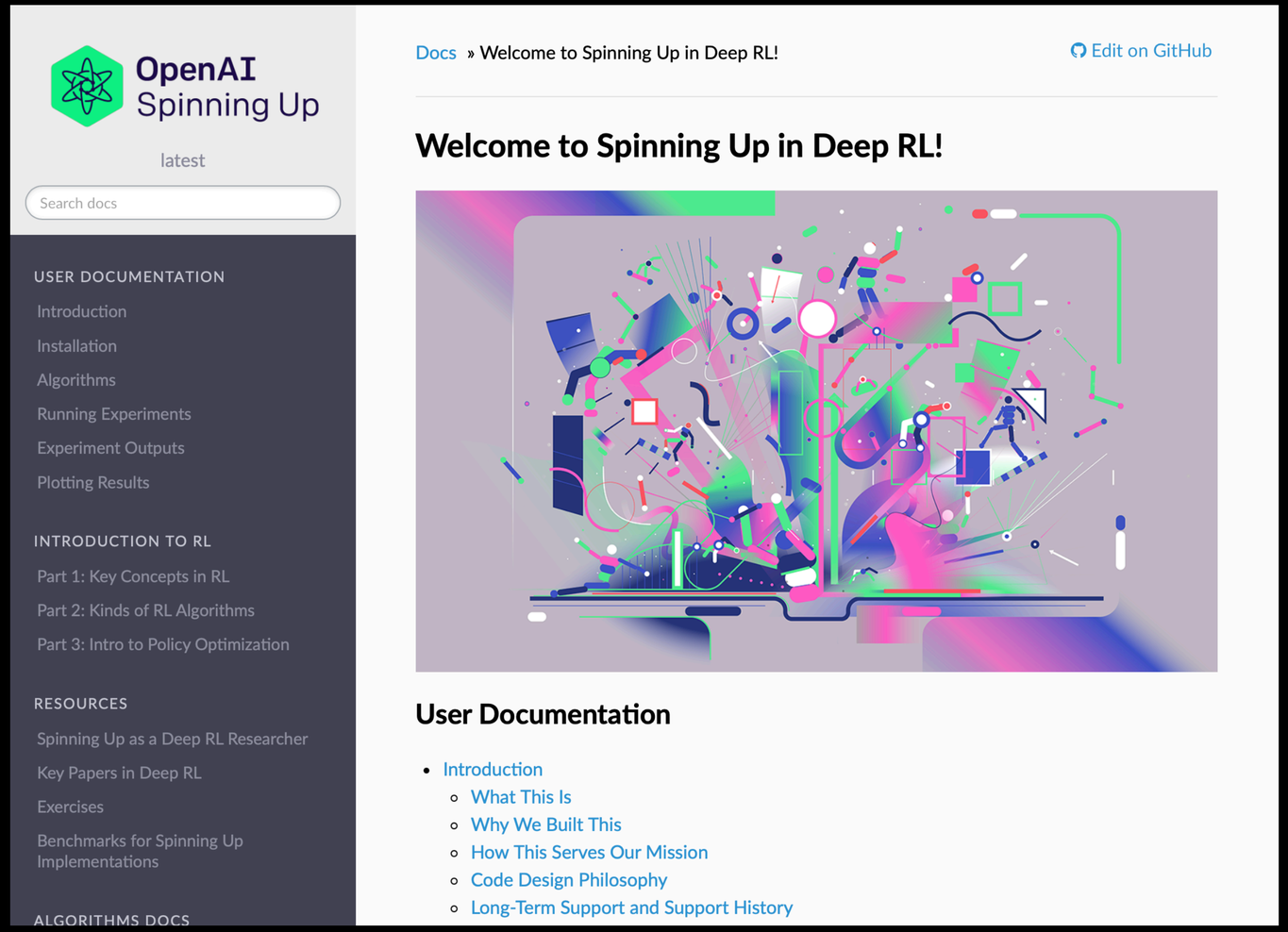


MuJoCo Environments

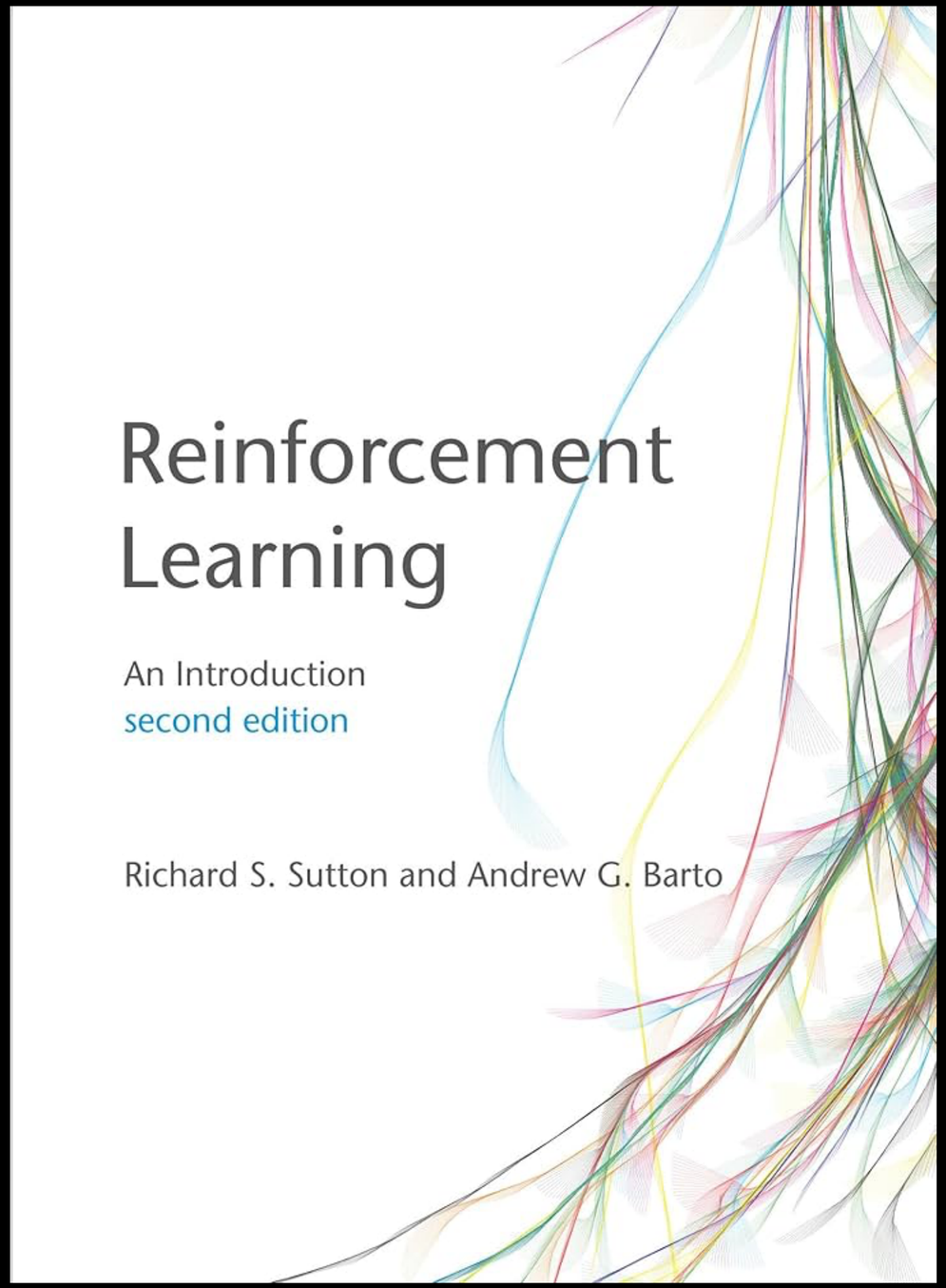
- Locomotion tasks in a physics simulator (MuJoCo)
- Continuous control tasks are some of the most challenging RL problems
- Robotic controllers exist



Next steps


OpenAI
Spinning Up
latest
Search docs
USER DOCUMENTATION
Introduction
Installation
Algorithms
Running Experiments
Experiment Outputs
Plotting Results
INTRODUCTION TO RL
Part 1: Key Concepts in RL
Part 2: Kinds of RL Algorithms
Part 3: Intro to Policy Optimization
RESOURCES
Spinning Up as a Deep RL Researcher
Key Papers in Deep RL
Exercises
Benchmarks for Spinning Up Implementations
ALGORITHMS DOCS
Docs » Welcome to Spinning Up in Deep RL!
Edit on GitHub
Welcome to Spinning Up in Deep RL!
User Documentation
Introduction
What This Is
Why We Built This
How This Serves Our Mission
Code Design Philosophy
Long-Term Support and Support History

Next steps


Reinforcement Learning
An Introduction
second edition
Richard S. Sutton and Andrew G. Barto

Questions

- Any questions?