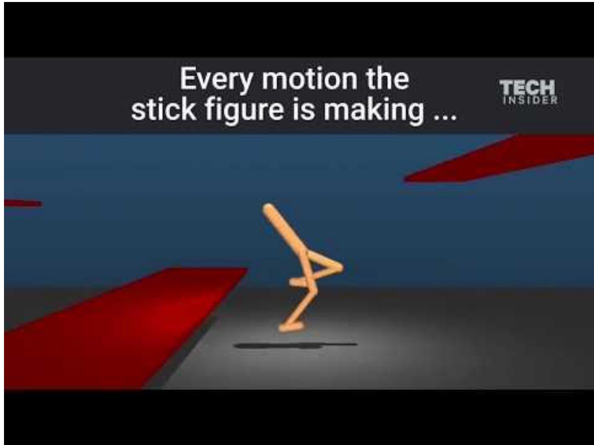
Reinforcement Learning for Robotics

Iretiayo Akinola

Columbia University



Reinforcement Learning for Robotics

Iretiayo Akinola
http://www.cs.columbia.edu/~iakinola/

COLUMBIA UNIVERSITY

Robot Learning

Robotics: see, think, act

Direct programming can be hard for different tasks

- Degree of structure and consistency
- Perception
- Manipulation
- Deformation

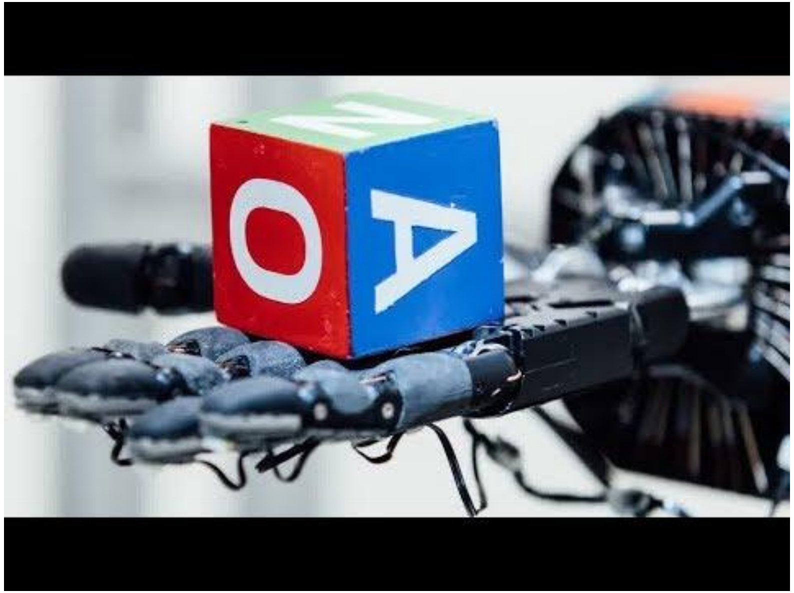
Vacuum robots	Home-cleaning robot
Lawn mowing	Cooking Robot
Pool cleaning	Laundry Robot
Manufacturing robots	Warehouse Robot

Robot Learning

Robotics: see, think, act

Direct programming can be hard for different tasks

- Degree of structure and consistency
- Perception
- Manipulation
- Deformation



Vacuum robots
Lawn mowing
Pool cleaning
Manufacturing robots

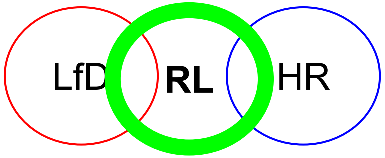
Home-cleaning robot
Cooking Robot
Laundry Robot
Warehouse Robot

Robot Learning

Manufacturing robots

Cooking Robot

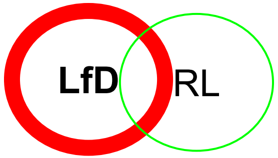
Robot Learning



An alternative to direct programming is learning-based methods:

- Learning from Demonstration
- Reinforcement Learning

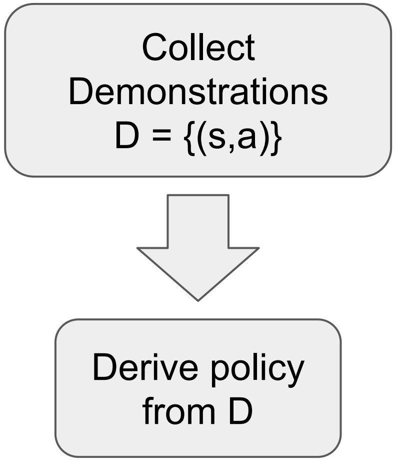
Learning from Demonstration (LfD)



LfD Dataset (D): a set of state-action pairs

Goal: Learn $\pi(a_t|s_t) - \text{policy}$

Assumptions: Human Teacher exists, Demonstration is possible



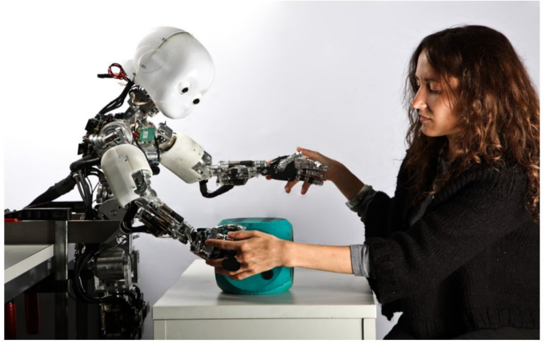
Key Considerations:

- Demonstration mode
- State representation
- Policy Derivation method
    - Supervised learning/Function approximation

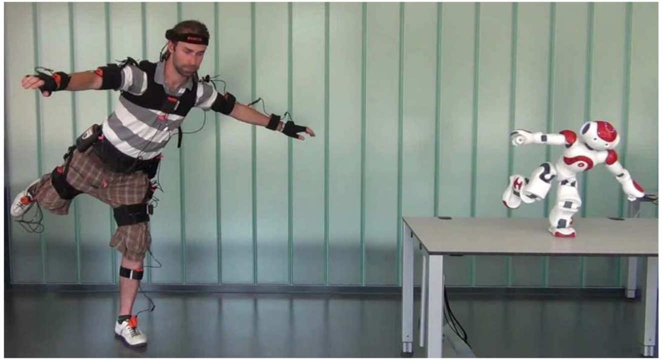
Learning from Demonstration (LfD)

LfD

- Learning from Demonstration data
  - Learns a mapping from state to action
- Demonstration modes
  - Teleoperation
  - Kinesthetic teaching (e.g. in motion trajectory learning)
  - Camera recording a human teacher
  - Robotic teachers

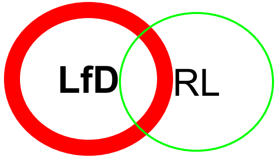


Kinesthetic Teaching



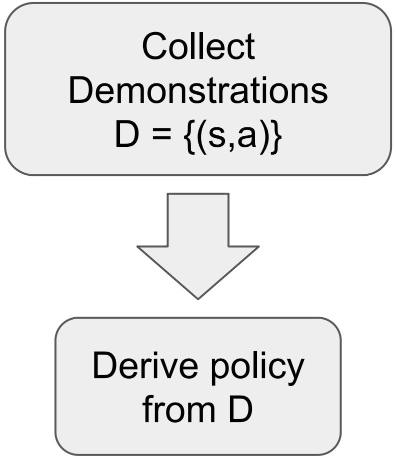
Motion Capture

LfD



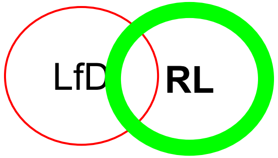
Pros

- Supervised Learning
- No need for manual reward function
- Exploration not an issue



Cons

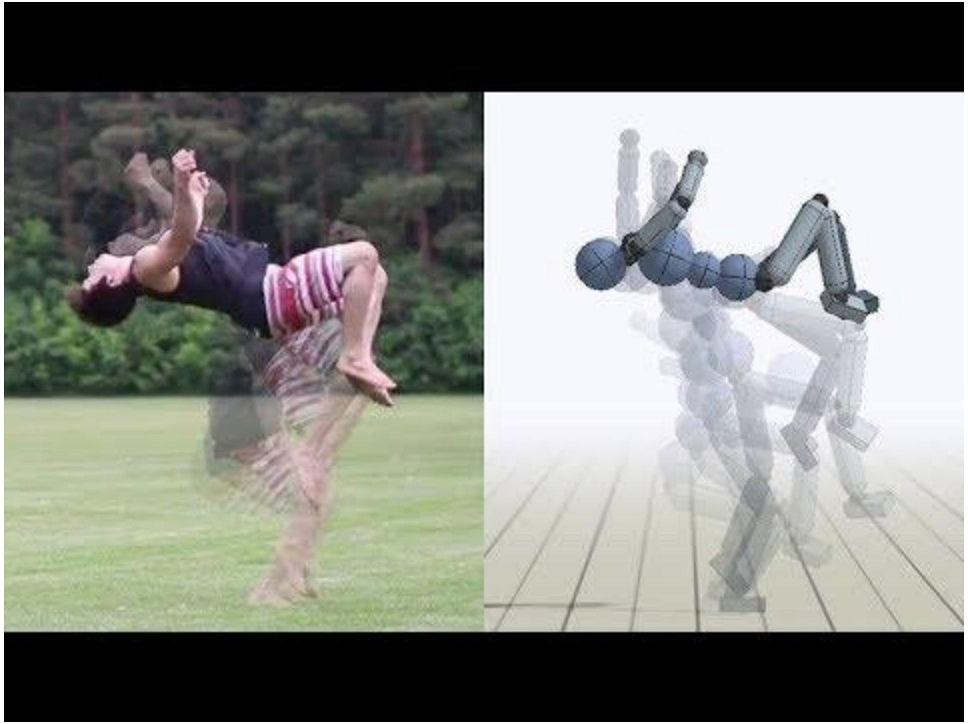
- Might not generalize well- Covariate shift
- Volume of demonstration
- Some tasks are difficult to demonstrate
- Suboptimal Demonstrations. Limited human patience and inconsistent user input
- Performance of the robot can be limited by that of the teacher.



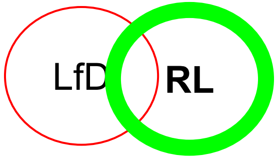
Robot Learning

An alternative to direct programming is learning-based methods:

- Learning from Demonstration
- Reinforcement Learning
- **Hybrid**

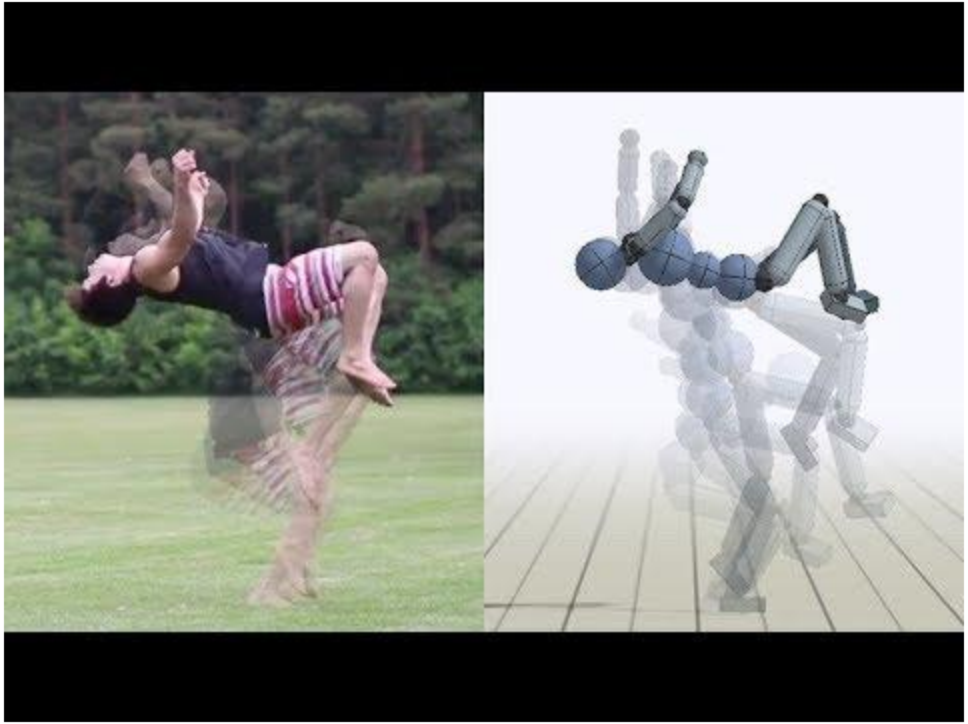


Robot Learning

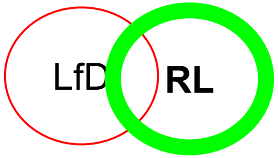


An alternative to direct programming is learning-based methods:

- Learning from Demonstration
- Reinforcement Learning
- **Hybrid**



Robot Learning

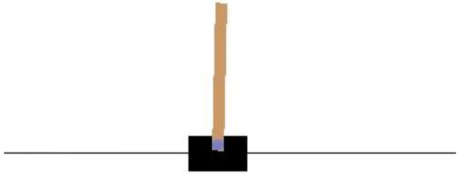


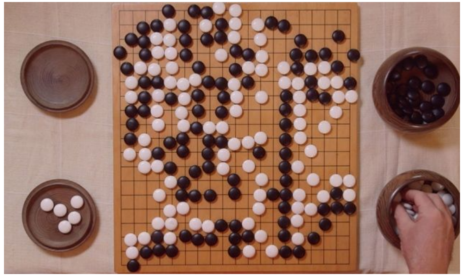
An alternative to direct programming is learning-based methods:

- Learning from Demonstration
- **Reinforcement Learning**

Reinforcement Learning

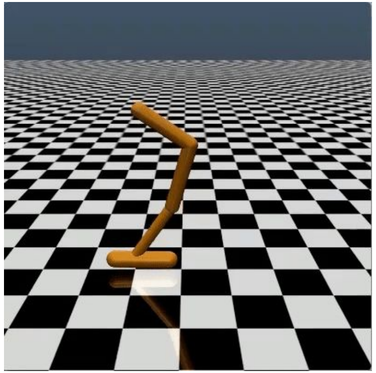
- Learning by trial and Error
- Maximize cumulative rewards
- Learn a policy





Reinforcement Learning Progress

- Classical Control
- Games: Atari, Go
- Robotics
  - continuous space, complex transition dynamics, complex rewards

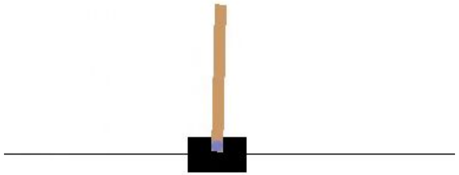


Reinforcement Learning

- Learning by trial and Error
- Maximize cumulative rewards
- Learn a policy

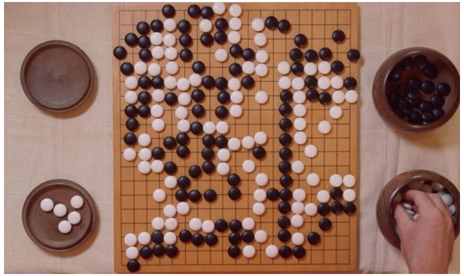
Reinforcement Learning Progress

- [Classical Control]
- Games: Atari, Go
- Robotics
    - continuous space, complex transition dynamics, complex rewards



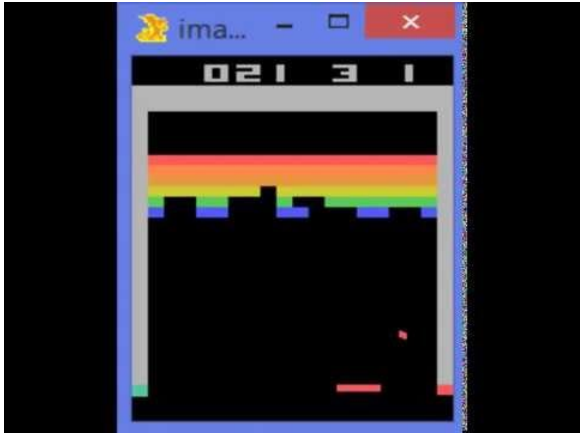
Reinforcement Learning

- Learning by trial and Error
- Maximize cumulative rewards
- Learn a policy



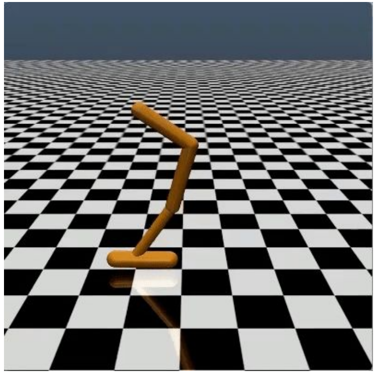
Reinforcement Learning Progress

- Classical Control
- Games: Atari, Go
- Robotics
    - continuous space, complex transition dynamics, complex rewards



Reinforcement Learning

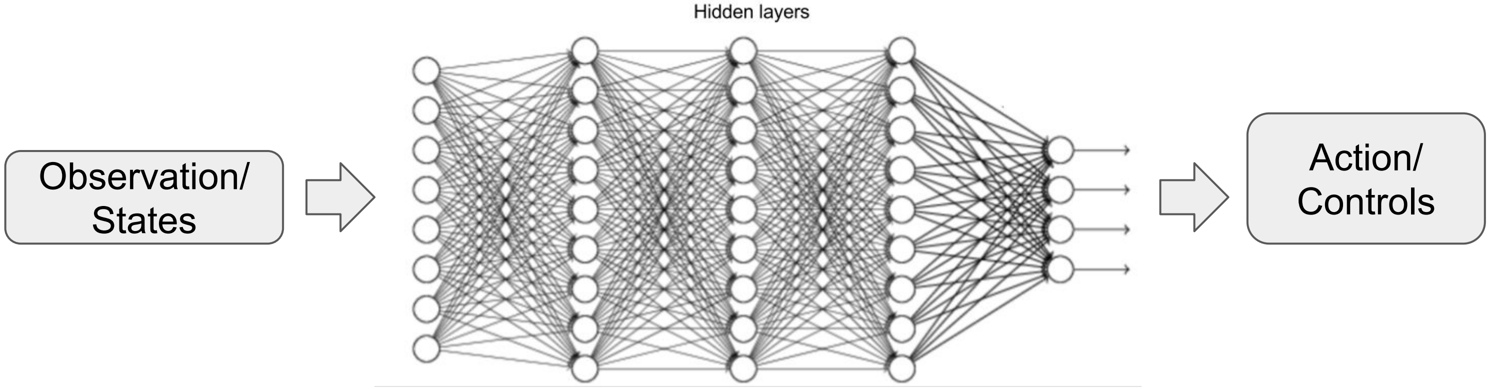
- Robotics
  - continuous space
  - complex transition dynamics
  - complex rewards



Reinforcement Learning in Robotics

Key Elements of recent Success

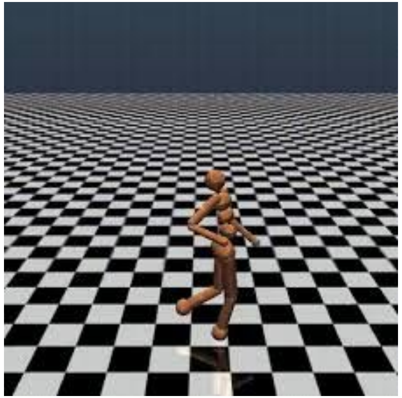
- **Deep Learning**
- Simulators (mujoco, bullet, roboschool, dart, gazebo, carsim)



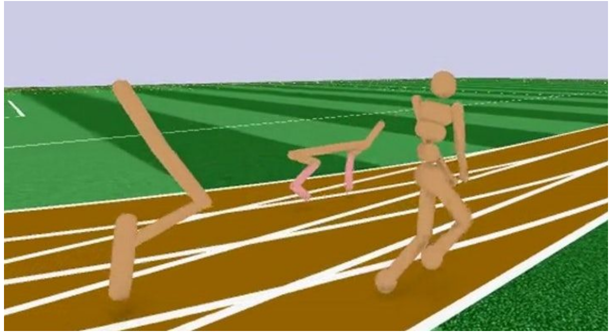
Reinforcement Learning in Robotics

Key Elements of recent Success

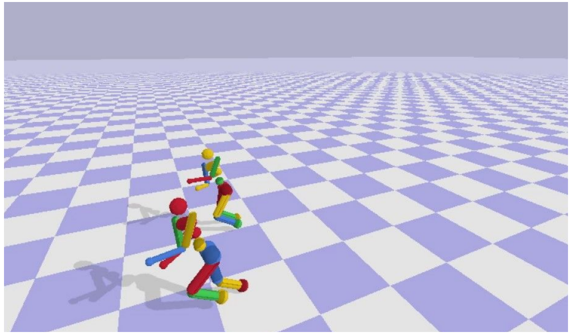
- Deep Learning
- **Simulators (mujoco, pybullet, roboschool, gazebo)**



mujoco



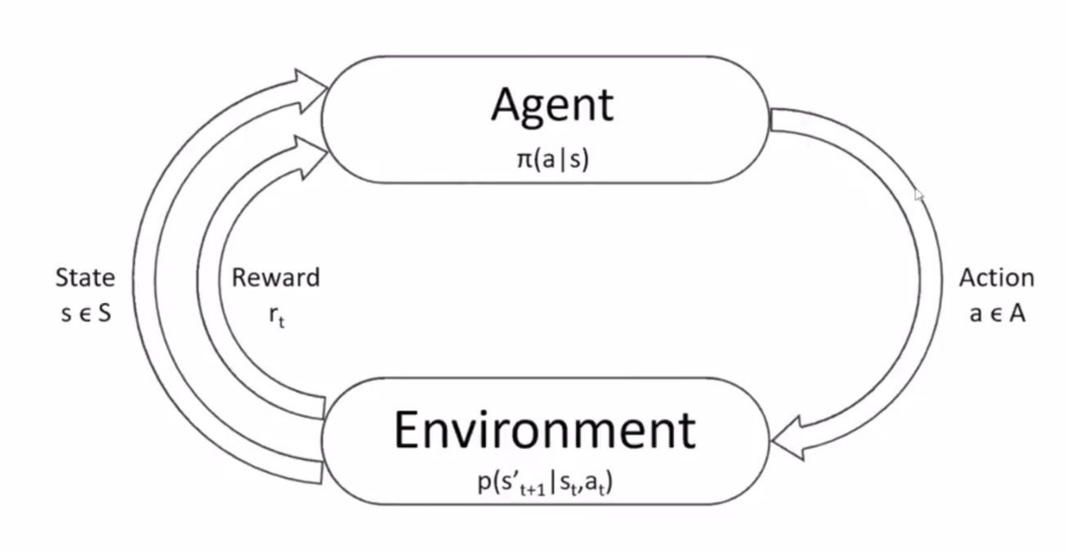
roboschool



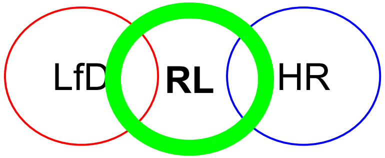
pybullet

Reinforcement Learning Formulation

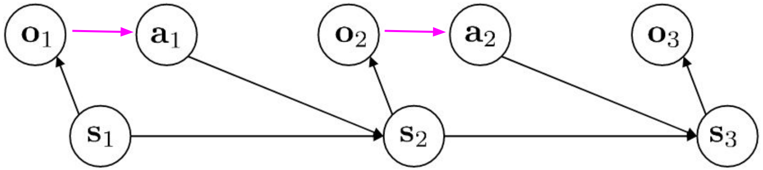
Markov Decision Process



Reinforcement Learning Formulation



The goal of RL is to get **a policy**: $s_t \xrightarrow{\pi} a_t$



$$\underbrace{p_\theta(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T)}_{\pi_\theta(\tau)} = p(\mathbf{s}_1) \prod_{t=1}^{T} \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$

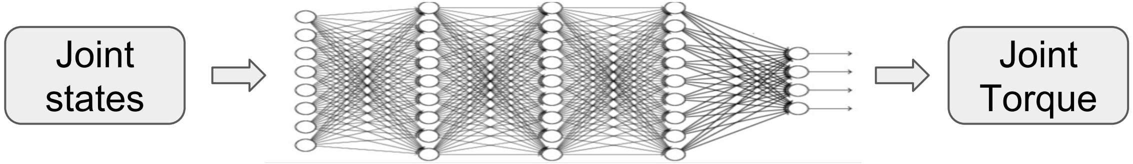
$$\text{reward function} \quad r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$$

$$\theta^\star = \arg\max_\theta E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

Reinforcement Learning Formulation

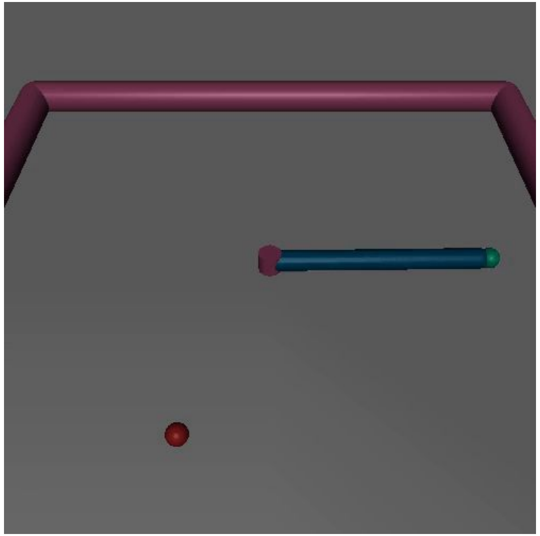
RL

The goal of RL is to get **a policy**: $s_t \xrightarrow{\pi} a_t$



Reward function:

$$r := \text{- dist(goal, end-effector) - } \alpha \text{ magnitude(torque)}$$



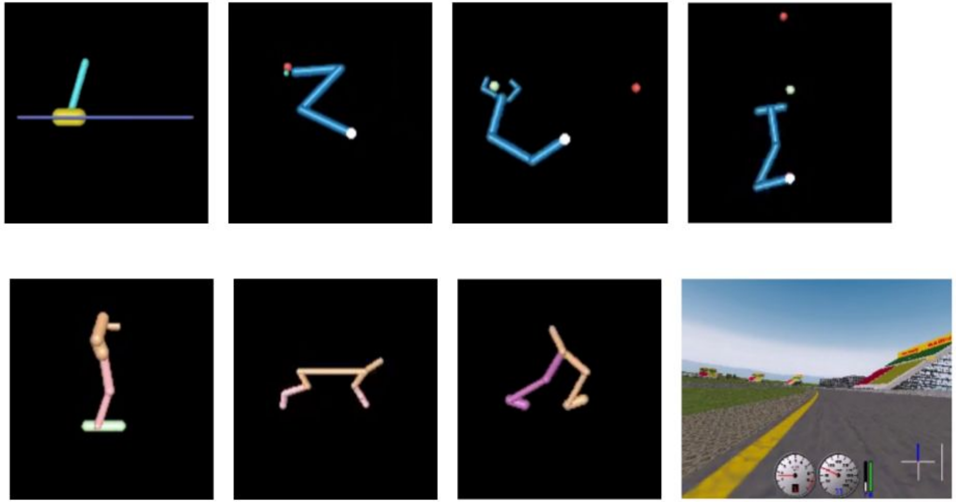
$$\theta^{\star} = \arg\max_{\theta} E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

reward function

Reinforcement Learning in Robotics

RL

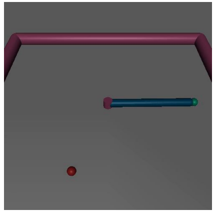
- Deterministic Policy Gradient Algorithms (David Silver et al. 2014)
  - DPG Algorithm
  - Experiments: continuous bandit, **pendulum**, mountain car, 2D puddle world and **Octopus Arm**
- Continuous Control with Deep Reinforcement Learning (Lillicrap et al. 2016)
  - DDPG

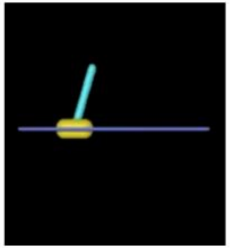


Quiz 1: Write down a reward function for each

RL

Example: $r :=$ - dist(goal, end-effector)



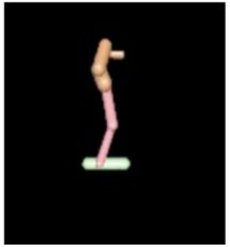


a

b

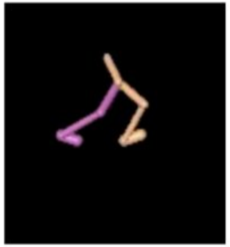
c

d

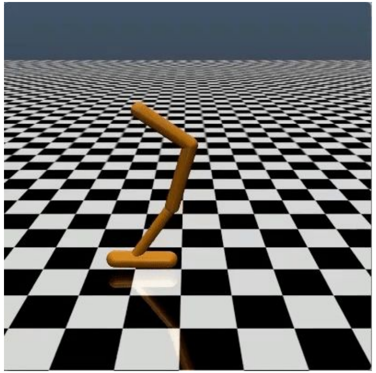


e

f



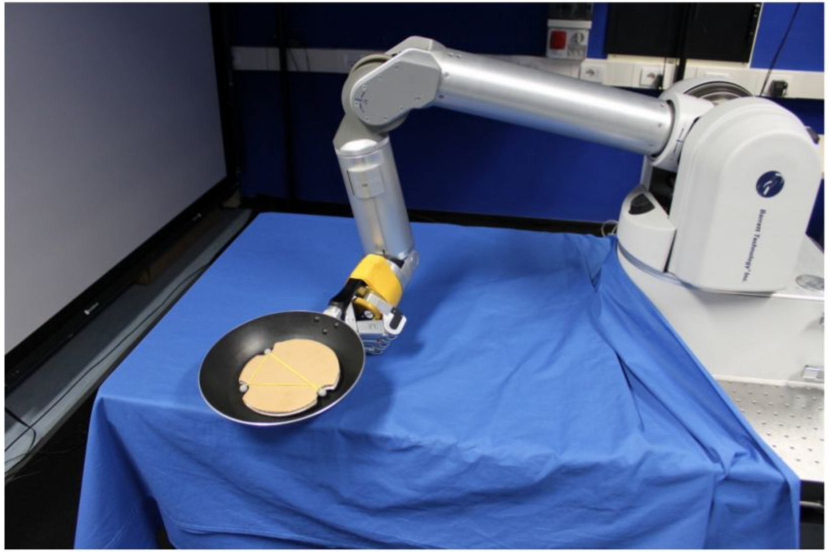
g

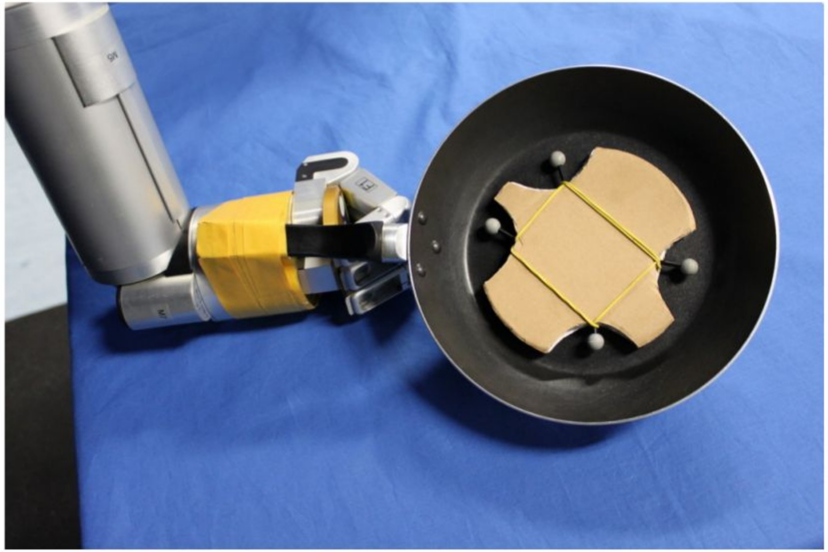


RL

Reinforcement Learning in Robotics

Pancake Flipping Task





Reinforcement Learning in Robotics

RL

Pancake Flipping Task

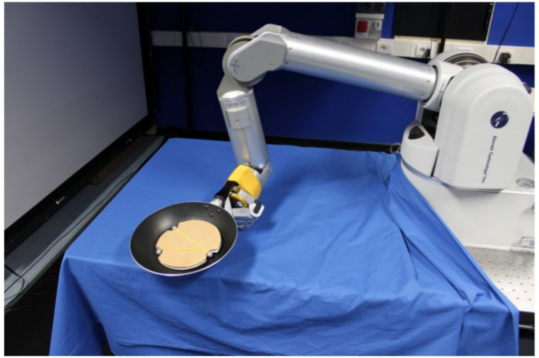
Reinforcement Learning in Robotics

RL

Pancake Flipping Task

$$R(\tau) = w_1 \left[\frac{\arccos(v_0 . v_{t_f})}{\pi} \right] + w_2 e^{-||x^p - x^F||} + w_3 x_3^M$$

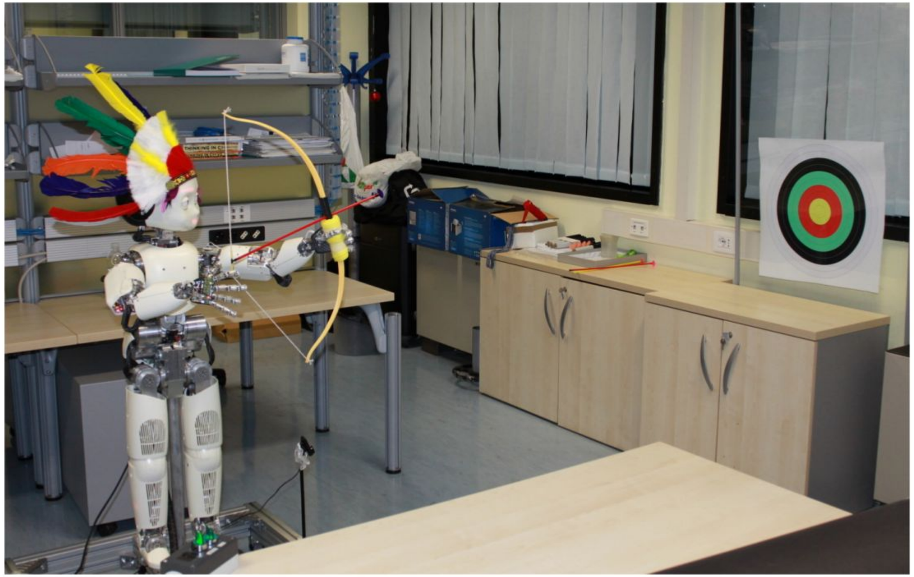
- Reward
  - Positional reward
  - Orientational reward



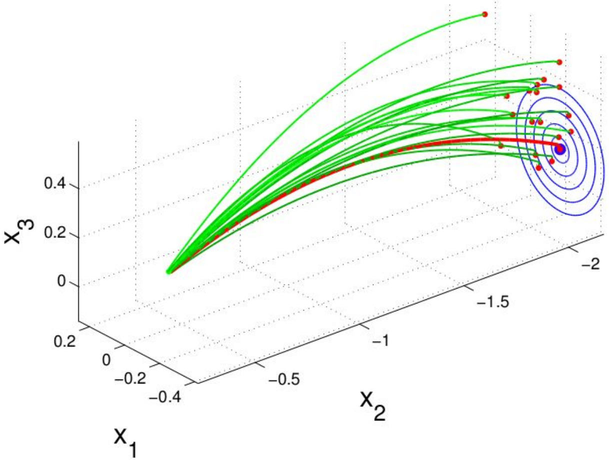
Reinforcement Learning in Robotics

RL

Archery Task



$$R(\tau) = e^{-||\hat{r}_T - \hat{r}_A||}$$



Reinforcement Learning

RL

Pros

- Fully autonomous
- Skills not explicit coded
- No demonstration needed

Cons

- Reward definition: Where does the R come from?
- Curse of Dimensionality: Exploration cost increases exponentially with dimension
- Generalization issues: Simulation to Real??
- Convergence

RL Class of Methods

RL

- Value Iteration methods (Q-Learning, SARSA)
- Policy Gradient Methods (REINFORCE)
- Actor-Critic Methods (DDPG, TRPO, PPO, A3C)
- Model-based RL (Guided Policy Search, Dyna)

RL Class of Methods

RL

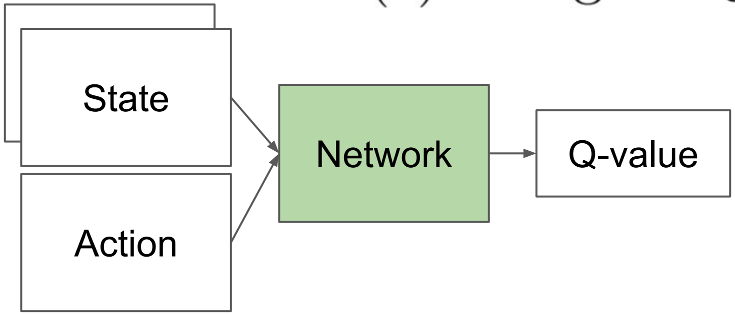
- Value Iteration methods (**Q-Learning**, SARSA)

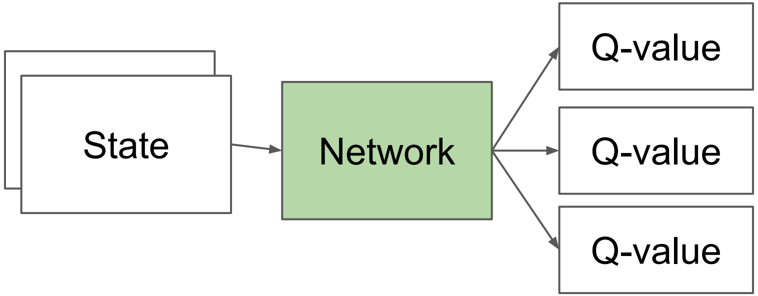
$$V^{\pi}(\mathbf{s}_t) = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'})|\mathbf{s}_t\right]$$

$$\pi(s) = \arg\max_a (r + V(s'_{s,a}))$$

$$Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'})|\mathbf{s}_t, \mathbf{a}_t\right]$$

$$\pi(\mathbf{s}) = \arg\max_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a})$$



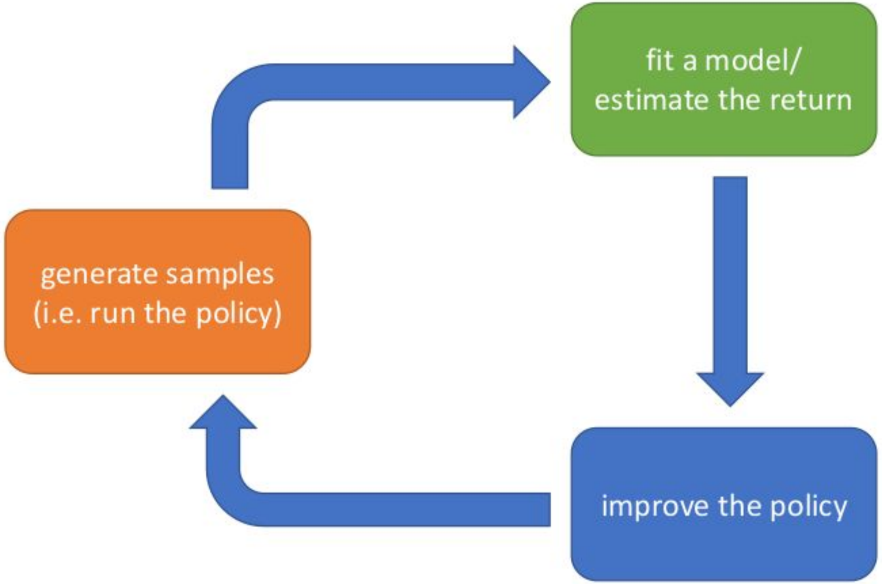


Playing Atari with Deep Reinforcement Learning (Mnih etal 2015)

RL Class of Methods

RL

- Value Iteration methods (**Q-Learning**, SARSA)



fit $V(\mathbf{s})$ or $Q(\mathbf{s}, \mathbf{a})$

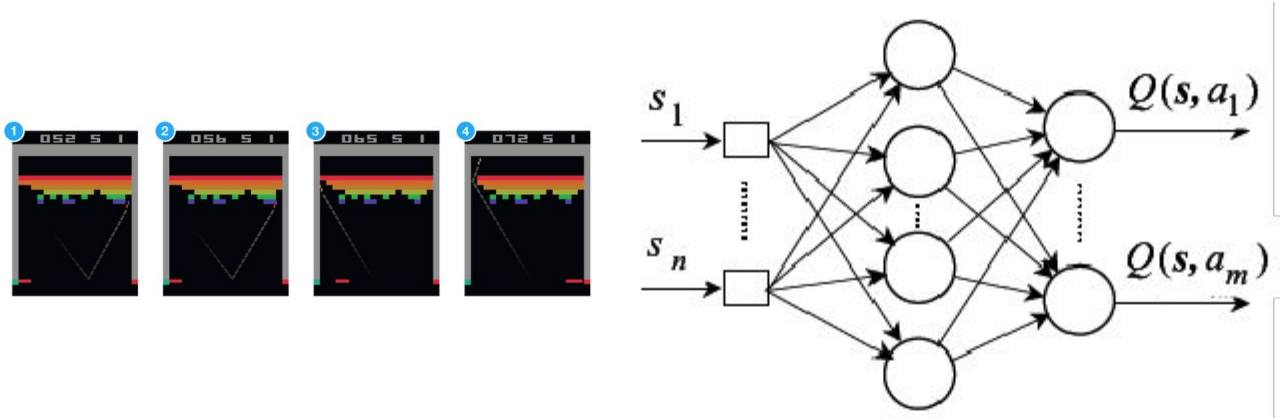
set $\pi(\mathbf{s}) = \arg\max_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a})$

Image from Levine's RL class

RL Class of Methods

RL

- Value Iteration methods (**Q-Learning**, SARSA)
- DQN: Playing Atari with Deep Reinforcement Learning (Mnih etal 2015)



Move Left

Move Right

Works well with Games- choose between discrete actions but robotics need continuous actions

RL Class of Methods

RL

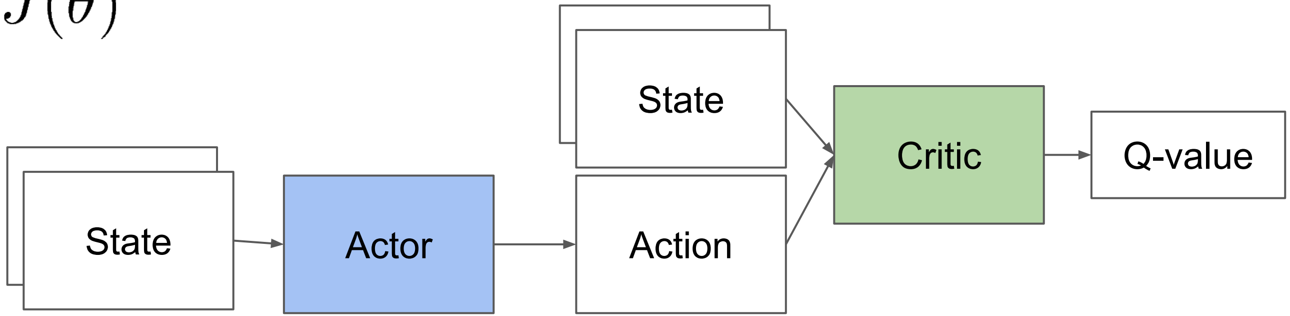
- Actor-Critic Methods (**DDPG**, A3C, TRPO, PPO)

$$Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = \sum_{t'=t}^{T} E_{\pi_\theta}\left[r(\mathbf{s}_{t'}, \mathbf{a}_{t'}) | \mathbf{s}_t, \mathbf{a}_t\right]$$

$$\pi_\theta(a|s) = \mathbb{P}[a|s, \theta]$$

$$J(\theta) = \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(s, a) Q(s, a)$$

$$\theta := \theta + \alpha \nabla_\theta J(\theta)$$

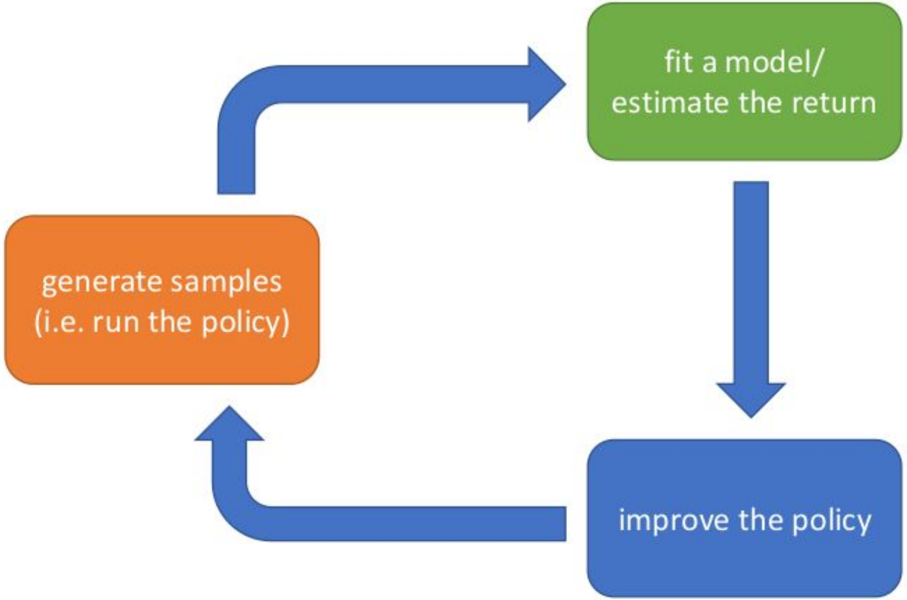


Continuous control with deep reinforcement learning (Lillicrap etal Deepmind 2016)

RL Class of Methods

RL

- Actor-Critic Methods (**DDPG**, TRPO, PPO, A3C)



fit $V(\mathbf{s})$ or $Q(\mathbf{s}, \mathbf{a})$

evaluate returns using V or Q!

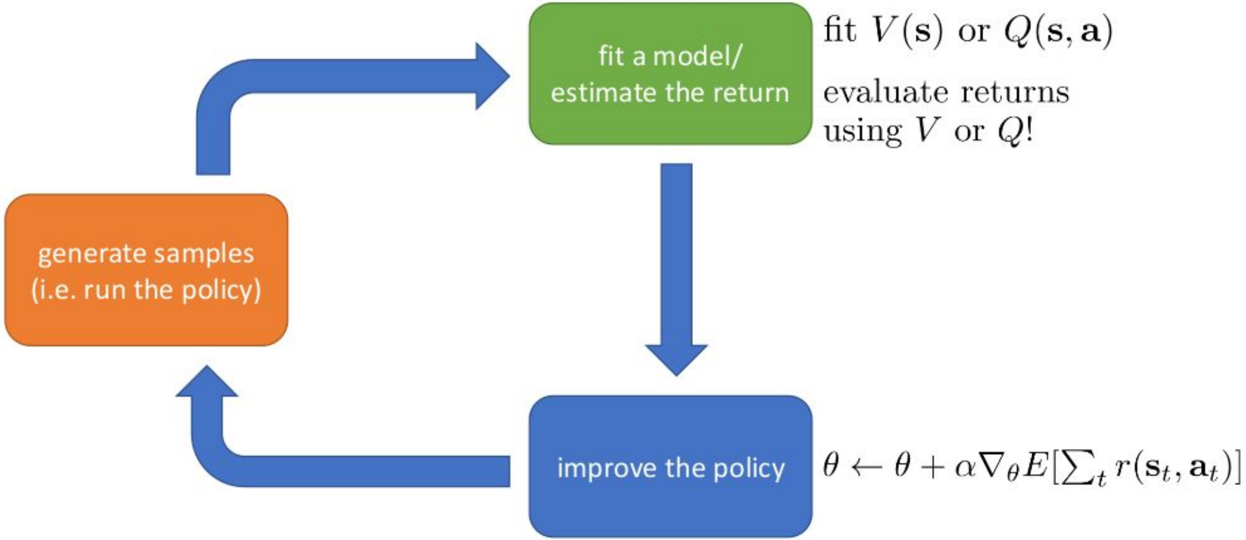
$\theta \leftarrow \theta + \alpha \nabla_\theta E[\sum_t r(\mathbf{s}_t, \mathbf{a}_t)]$

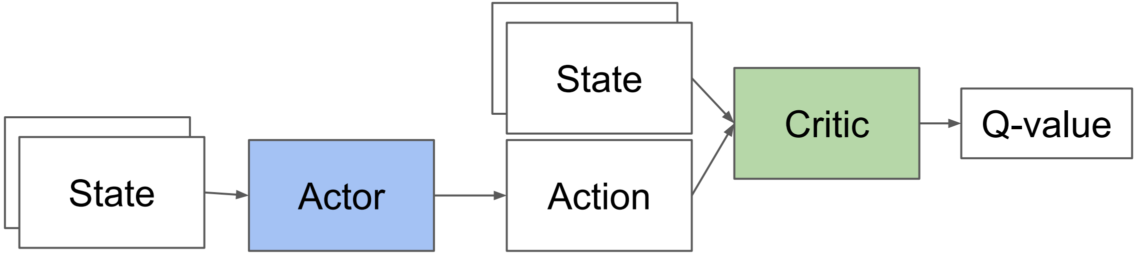
Image from Levine's RL class

RL Class of Methods (Actor/Critic)

RL

- A

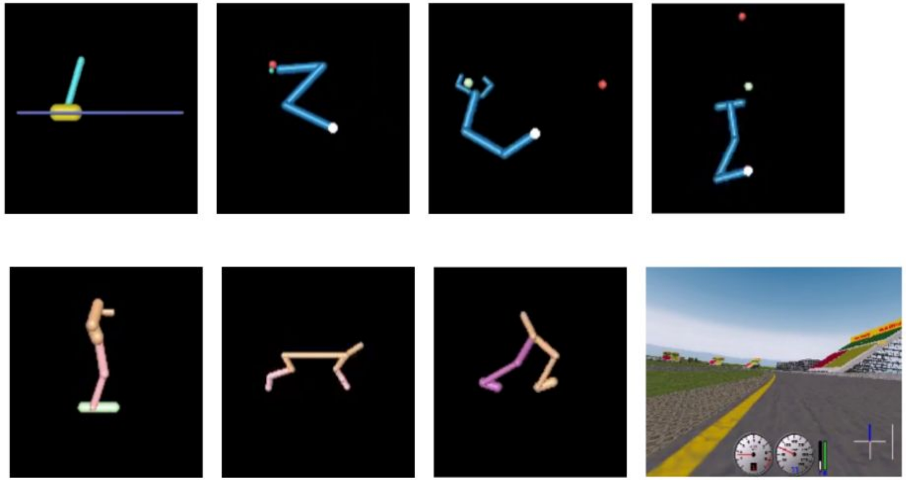




RL

RL Class of Methods

- Actor-Critic Methods (**DDPG**)
- Deterministic Policy Gradient Algorithms (Silver etal 2014)
  - Critic: linear function, Actor: Gaussian policy
- Continuous control with deep reinforcement learning (Lillicrap etal 2016)
  - Critic: NN, Actor: NN

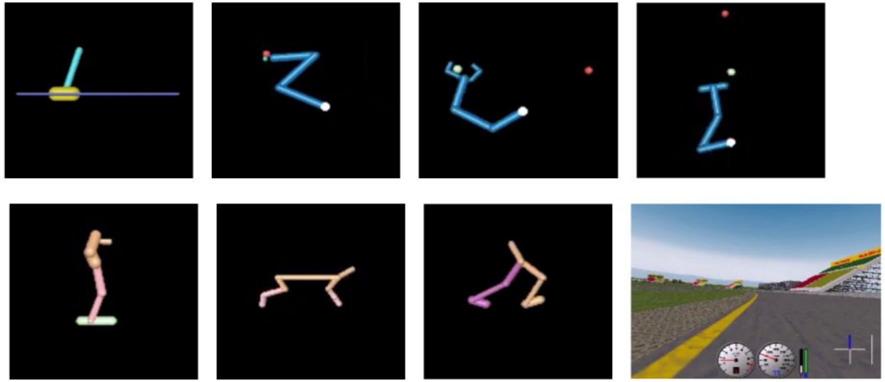


Case-Study: Actor-Critic Methods (DDPG)

RL

CONTINUOUS CONTROL WITH DEEP REINFORCEMENT LEARNING

Timothy P. Lillicrap*, Jonathan J. Hunt*, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver & Daan Wierstra
Google Deepmind
London, UK
{countzero, jjhunt, apritzel, heess, etom, tassa, davidsilver, wierstra} @ google.com



1. Initialize Actor and Critic networks
2. Generate samples from actor policy
3. Fit Critic model based on samples
4. Calculate actor gradients
5. Update actor

RL Class of Methods

RL

- Policy Gradient Methods (REINFORCE)

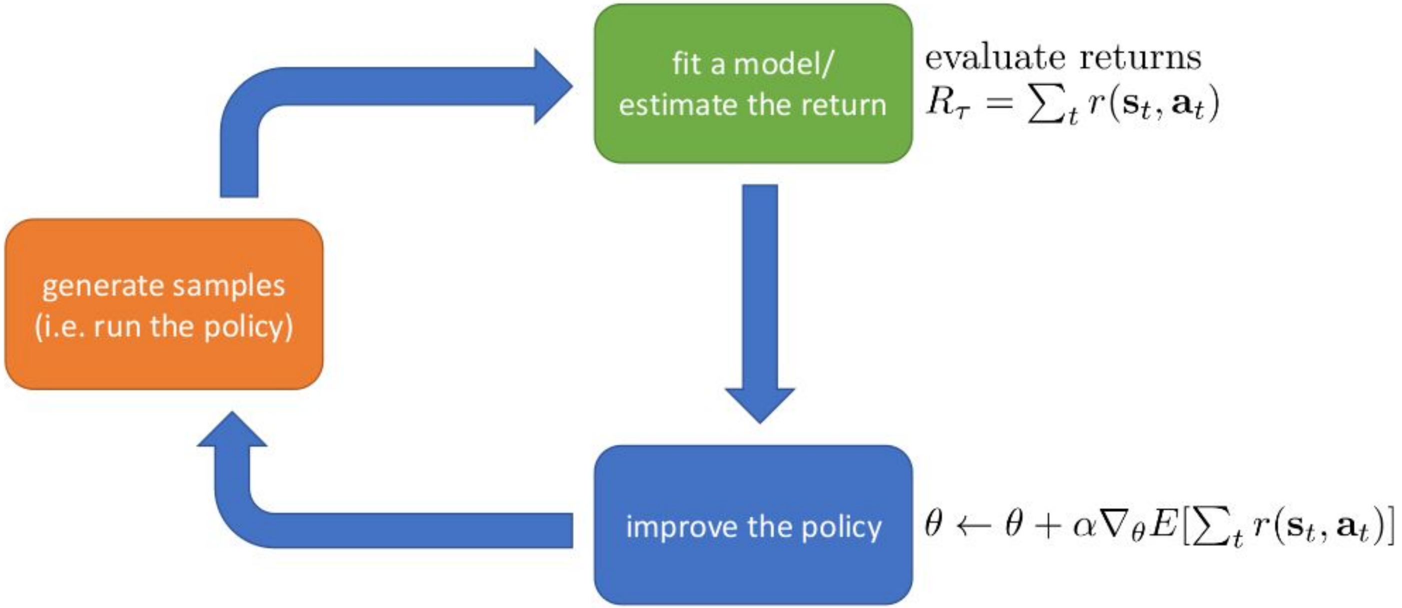
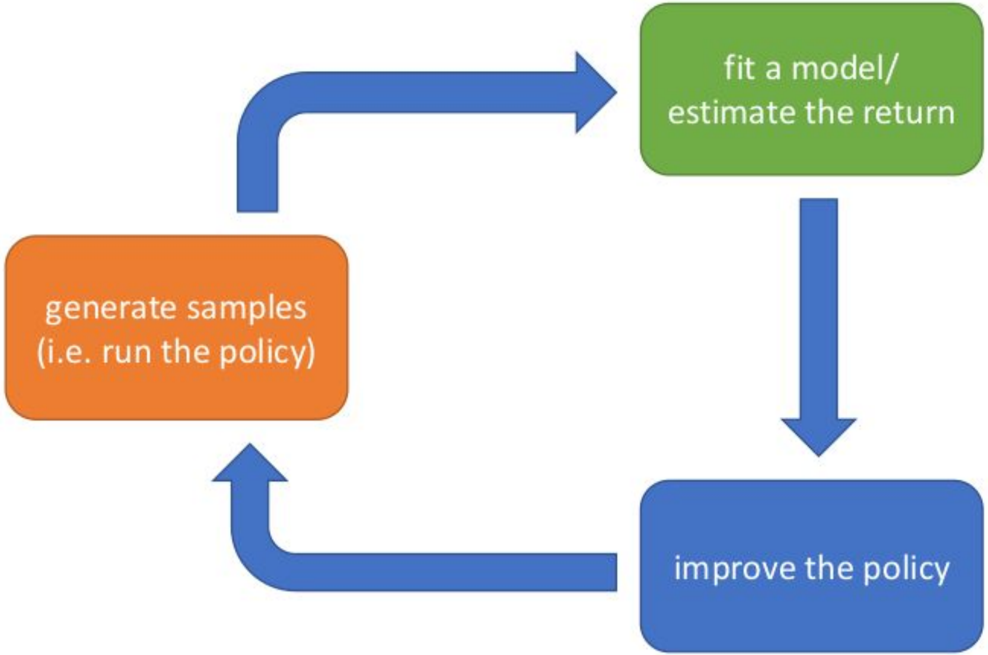


Image from Levine's RL class

RL Class of Methods

- Guided Policy Search

RL Class of Methods



compute $\hat{Q} = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$ (MC policy gradient)

fit $Q_\phi(\mathbf{s}, \mathbf{a})$ (actor-critic, Q-learning)

estimate $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ (model-based)

$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$ (policy gradient)

$\pi(\mathbf{s}) = \arg\max Q_\phi(\mathbf{s}, \mathbf{a})$ (Q-learning)

optimize $\pi_\theta(\mathbf{a}|\mathbf{s})$ (model-based)

Image from Levine's RL class

Reinforcement Learning

Pros

- Fully autonomous
- No explicit coding needed
- No demonstration needed

Cons

- Reward definition: Where does the R come from?
- Curse of Dimensionality: Exploration cost increases exponentially with dimension
- Generalization issues
- Convergence

Reinforcement Learning

Pros

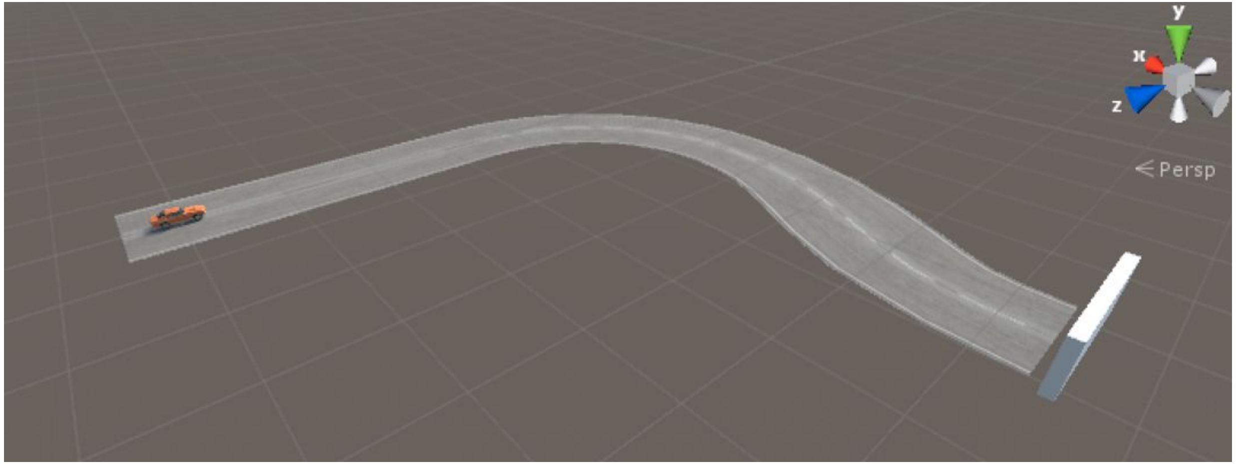
- Fully autonomous
- No explicit coding needed
- No demonstration needed

Cons

- **Reward definition: Where does R come from?**
- **Curse of Dimensionality: Exploration cost increases exponentially with dimension**
- Generalization issues
- Convergence

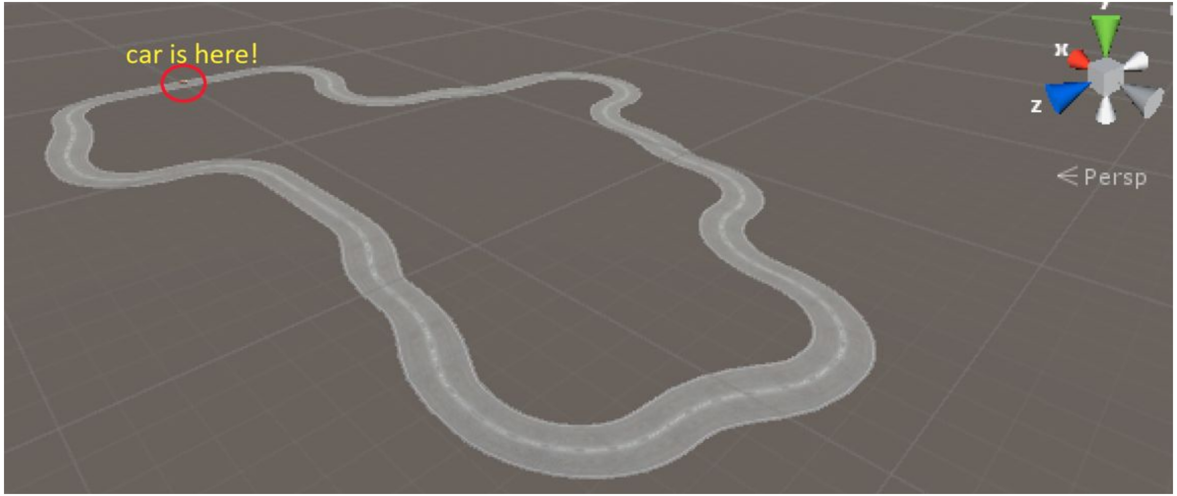
Reinforcement Learning in Robotics

- **Quiz 2: Mobile Robot Example**



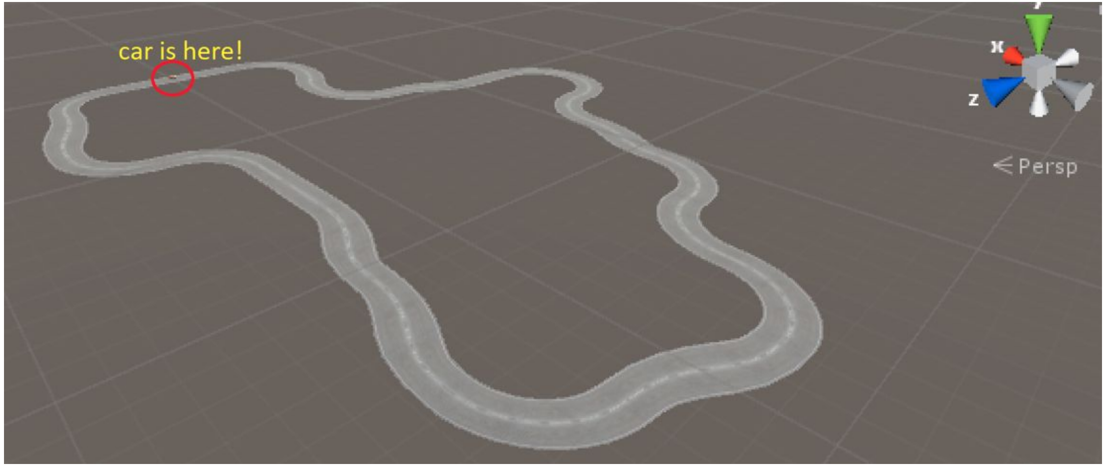
Reinforcement Learning in Robotics

- **Quiz 2: Mobile Robot Example**



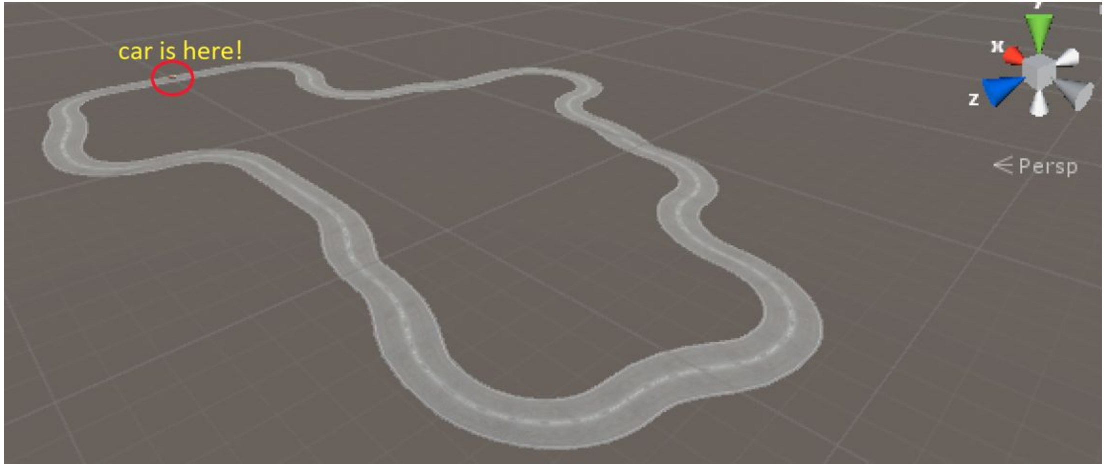
Reinforcement Learning in Robotics

- **Quiz 2: Mobile Robot Example**
  - **State space:**
  - **Action space:**
  - **Reward function:**



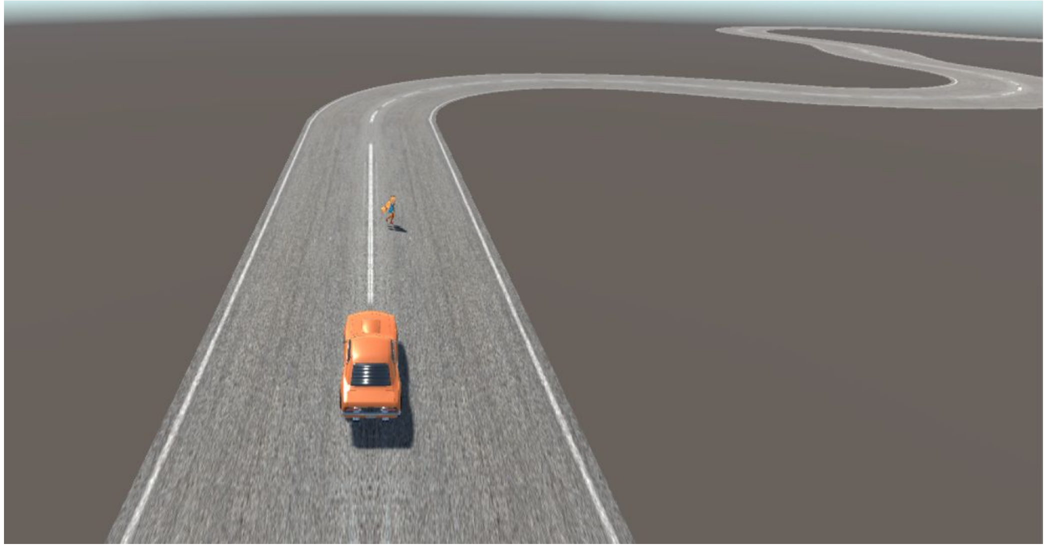
Reinforcement Learning in Robotics

- **Quiz 2: Mobile Robot Example**
  - **State space:** image + car velocity
  - **Action space:** gas, wheel
  - **Reward function:** forward velocity/5, and -500 if fall off



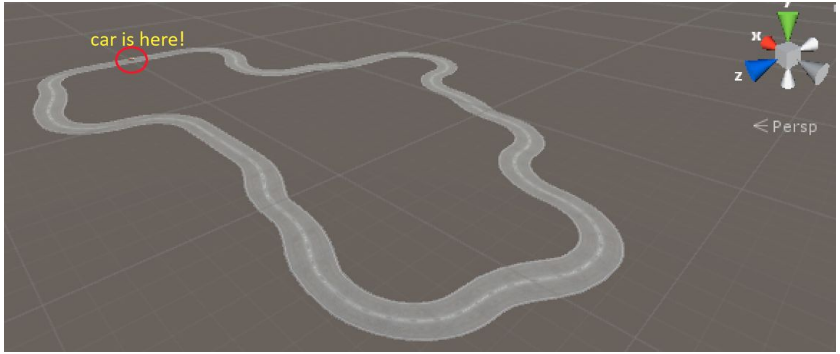
Reinforcement Learning in Robotics

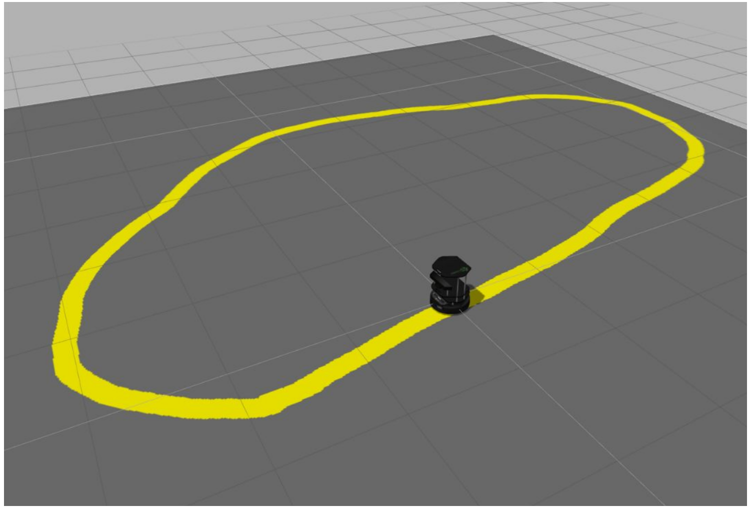
- **Quiz 2: Mobile Robot Example**
  - **State space:** image + car velocity
  - **Action space:** gas, wheel
  - **Reward function:** forward velocity/5, and -500 if fall off
  - **Pedestrians!!! What will you change?**



Reinforcement Learning in Robotics

- **Quiz 2: Mobile Robot Example**
  - **Similar to Homework 5?**





Reinforcement Learning in Robotics

Mobile Robot -> Autonomous Vehicle

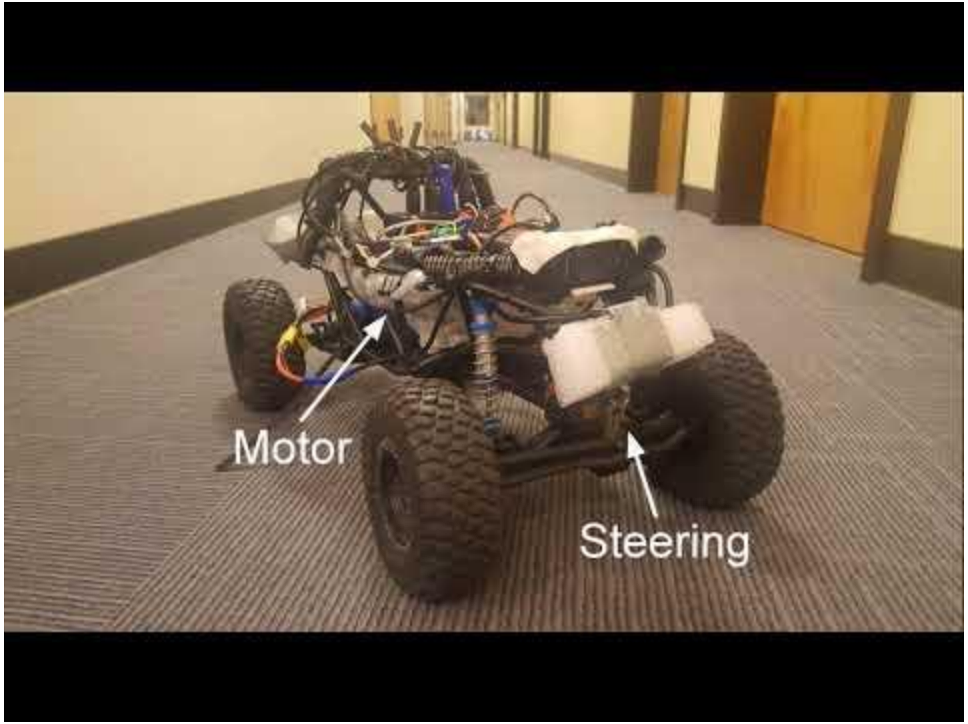
- CARLA: An Open Urban Driving Simulator (Dosovitskiy etal 2017) (GitHub)

Reinforcement Learning in Robotics

CARLA: An Open Urban Driving Simulator (Dosovitskiy etal 2017) (GitHub)

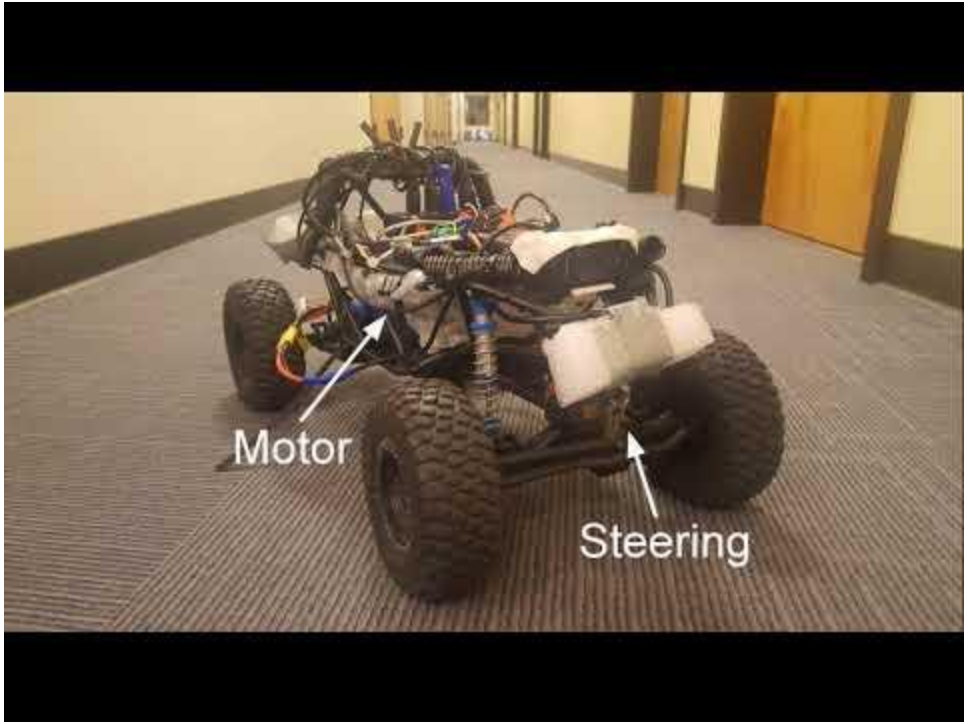
Reinforcement Learning in Robotics

Indoor Navigation



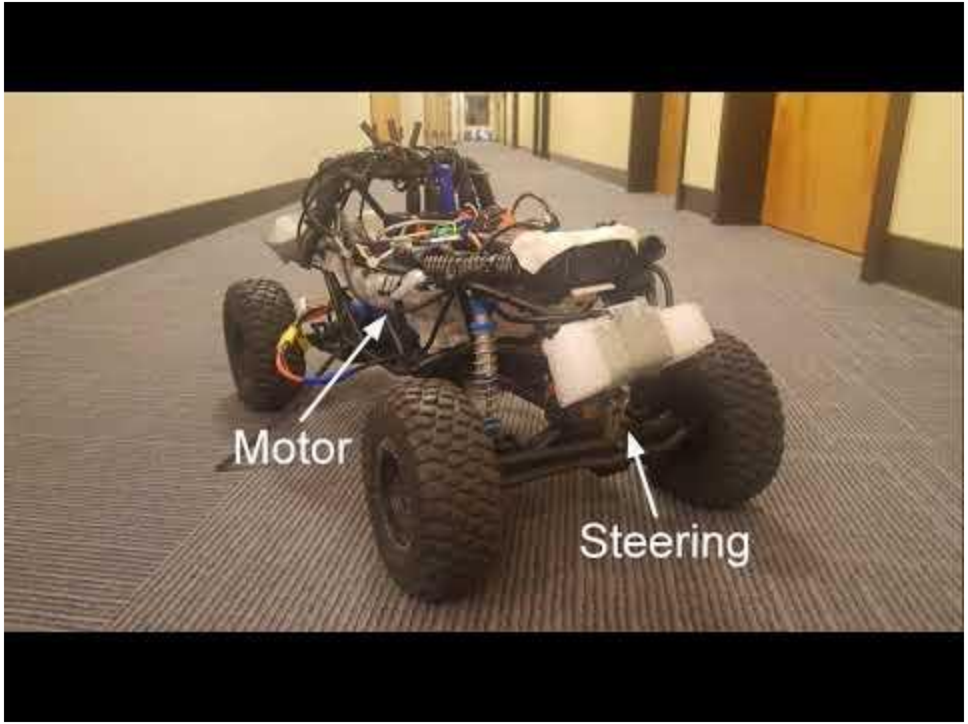
Reinforcement Learning in Robotics

Indoor Navigation



Reinforcement Learning in Robotics

Indoor Navigation



References

- “Robot Learning From Human Teachers”, Sonia Chernova and Andrea L. Thomaz (2014)
- “Deterministic Policy Gradient Algorithms” (David Silver et al. 2014)
- “Continuous control with deep reinforcement learning” (Lillicrap etal Deepmind 2016)
- "CARLA: An open urban driving simulator." Dosovitskiy, Alexey, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. (2017).
- “Self-supervised Deep Reinforcement Learning with Generalized Computation Graphs for Robot Navigation” Kahn, Gregory, Adam Villaflor, Bosen Ding, Pieter Abbeel, and Sergey Levine. (ICRA 2018)

Reinforcement Learning for Robotics

Iretiayo Akinola
http://www.cs.columbia.edu/~iakinola/

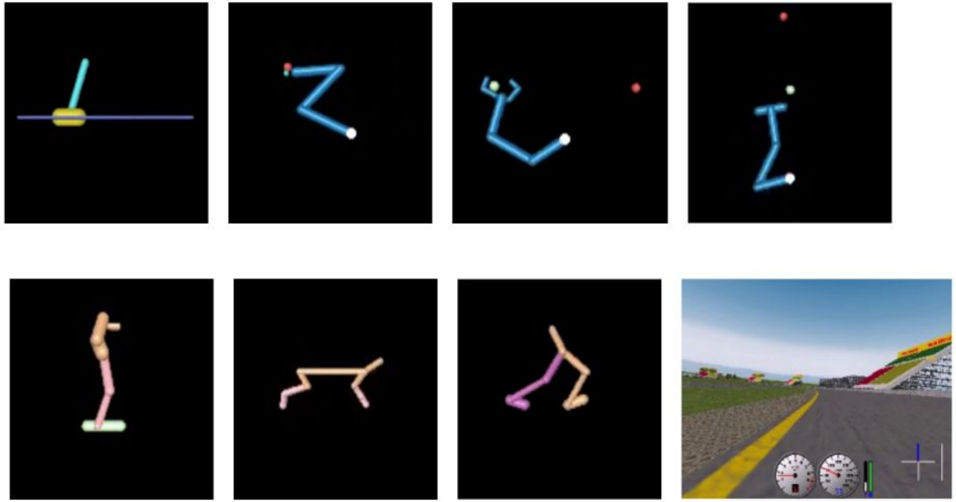
COLUMBIA UNIVERSITY

Thanks

Reinforcement Learning in Robotics

RL

- Deterministic Policy Gradient Algorithms (David Silver et al. 2014)
  - DPG Algorithm
  - Experiments: continuous bandit, **pendulum**, mountain car, 2D puddle world and **Octopus Arm**
- Continuous Control with Deep Reinforcement Learning (Lillicrap et al. 2016)
  - DDPG



DPG (Silver et al. 2014)

RL

- Deterministic Policy Gradient Algorithms (David Silver et al. 2014)

Total discounted reward $r_t^\gamma = \sum_{k=t}^{\infty} \gamma^{k-t} r(s_k, a_k)$ where $0 < \gamma < 1$

Value functions: $V^\pi(s) = \mathbb{E}\left[r_1^\gamma | S_1 = s; \pi\right]$

$$Q^\pi(s,a) = \mathbb{E}\left[r_1^\gamma | S_1 = s, A_1 = a; \pi\right]$$

Performance Objective: $J(\pi) = \mathbb{E}\left[r_1^\gamma | \pi\right]$

$$J(\pi_\theta) = \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \pi_\theta(s,a) r(s,a) \mathrm{d}a \mathrm{d}s \quad = \int_{\mathcal{S}} \int_{\mathcal{A}} \rho^\beta(s) \pi_\theta(a|s) Q^\pi(s,a) \mathrm{d}a \mathrm{d}s$$

$$= \mathbb{E}_{s \sim \rho^\pi, a \sim \pi_\theta}\left[r(s,a)\right]$$

Policy Gradient: $\nabla_\theta J(\pi_\theta)$

$$\theta := \theta + \alpha \nabla_\theta J(\pi_\theta)$$

DPG (Silver et al. 2014)

RL

- Deterministic Policy Gradient Algorithms (David Silver et al. 2014)

Performance Objective:

$$J(\pi_\theta) = \int_{\mathcal{S}} \int_{\mathcal{A}} \rho^\beta(s)\pi_\theta(a|s)Q^\pi(s,a)\mathrm{d}a\mathrm{d}s$$

Policy Gradient:

$$\nabla_\theta J(\pi_\theta)$$

Deterministic Policy Gradient:

$$\nabla_\theta J(\mu_\theta) = \int_{\mathcal{S}} \rho^\mu(s)\nabla_\theta \mu_\theta(s)\ \nabla_a Q^\mu(s,a)|_{a=\mu_\theta(s)}\,\mathrm{d}s$$

$$= \mathbb{E}_{s\sim\rho^\mu}\left[\nabla_\theta \mu_\theta(s)\ \nabla_a Q^\mu(s,a)|_{a=\mu_\theta(s)}\right]$$

Actor

Critic

$$\theta := \theta + \alpha \nabla_\theta J(\mu_\theta)$$

DPG (Silver et al. 2014)

RL

- Deterministic Policy Gradient Algorithms (David Silver et al. 2014)

Actor: $\mu_\theta(s)$ Critic: $Q^w(s_t, a_t)$

DPG Algorithm:

$$\delta_t = r_t + \gamma Q^w(s_{t+1}, \mu_\theta(s_{t+1})) - Q^w(s_t, a_t)$$

$$w_{t+1} = w_t + \alpha_w \delta_t \nabla_w Q^w(s_t, a_t)$$

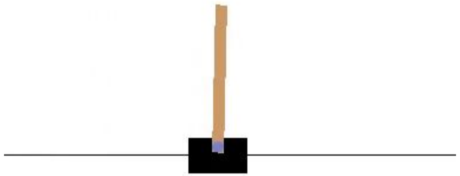
$$\theta_{t+1} = \theta_t + \alpha_\theta \nabla_\theta \mu_\theta(s_t) \left. \nabla_a Q^w(s_t, a_t) \right|_{a=\mu_\theta(s)}$$

RL

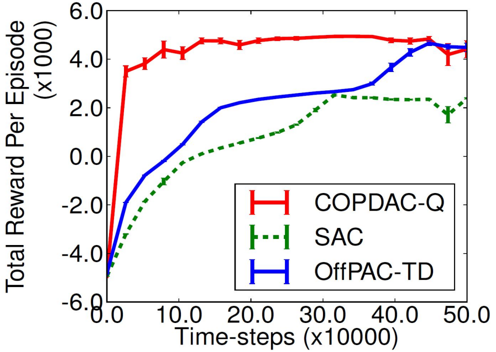
DPG (David Silver et al. 2014)

- Experiments:
  - continuous bandit, **pendulum**, mountain car, 2D puddle world, Octopus Arm

Pendulum:



- State: joint position/velocities
- Action: move left or right
- Reward: +1 for every step while rod is upright
- Critic: $V(s) = v^\top \phi(s)$
- Actor:
  - Target policy: $\mu_\theta(s) = \theta^\top \phi(s)$
  - Behavior policy: $\beta(\cdot|s) \sim \mathcal{N}(\theta^\top \phi(s), \sigma_\beta^2)$



DPG (David Silver et al. 2014)

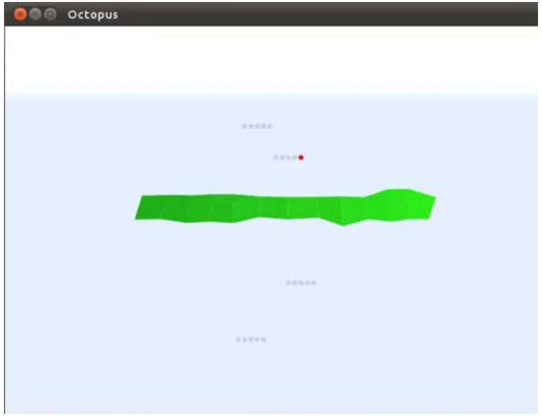
RL

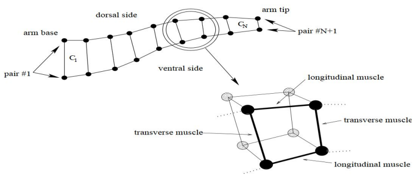
- Experiments:
  - continuous bandit, pendulum, mountain car, 2D puddle world, Octopus Arm

Octopus Arm:

- State: 50 continuous joint state variables
- Action: 20 variables to control muscles
- Reward: change in distance between arm and target
- Critic: NN Multilayer Perceptron (MLP)
- Actor: MLP (8 hidden units)

Octopus Arm

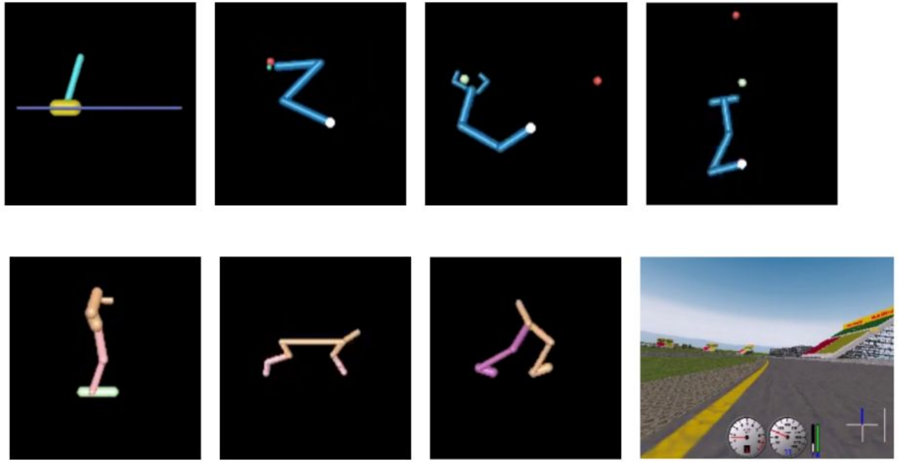




Reinforcement Learning in Robotics

RL

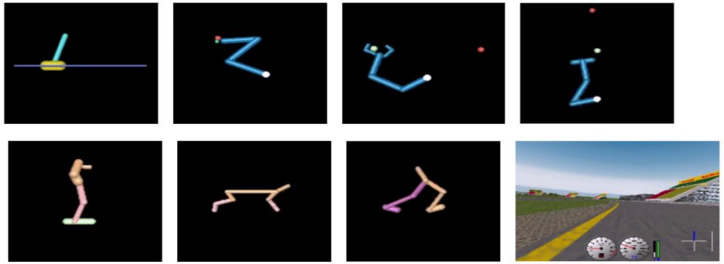
- Deterministic Policy Gradient Algorithms (Silver et al. 2014) **DPG**
  - Critic: linear function, Actor: Gaussian policy
  - Critic: NN(MLP), Actor: NN(MLP)
- Continuous control with deep reinforcement learning (Lillicrap et al. 2016) **DDPG**
  - Critic: DNN, Actor: DNN

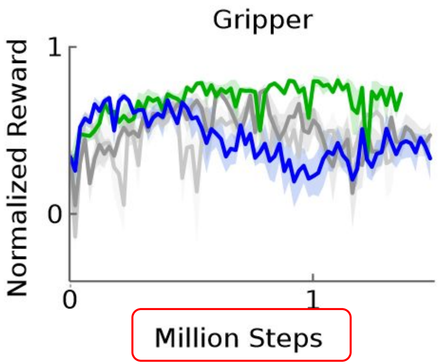


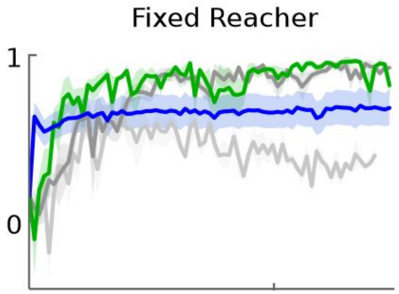
DDPG (Lillicrap 2016)

RL

- Batch normalization
- Target Networks







- original DDPG with batch normalization (light grey),
- with target network (dark grey),
- with target networks and batch norm (green),
- with target networks from pixel-only inputs (blue)

Reinforcement Learning in Robotics

- **Reward definition: Where does R come from?**