Autonomous Robotics

Reinforcement Learning II

Josiah Hanna
University of Wisconsin — Madison

Announcements

Homework 4 due on Tuesday.

Midterm Evaluations

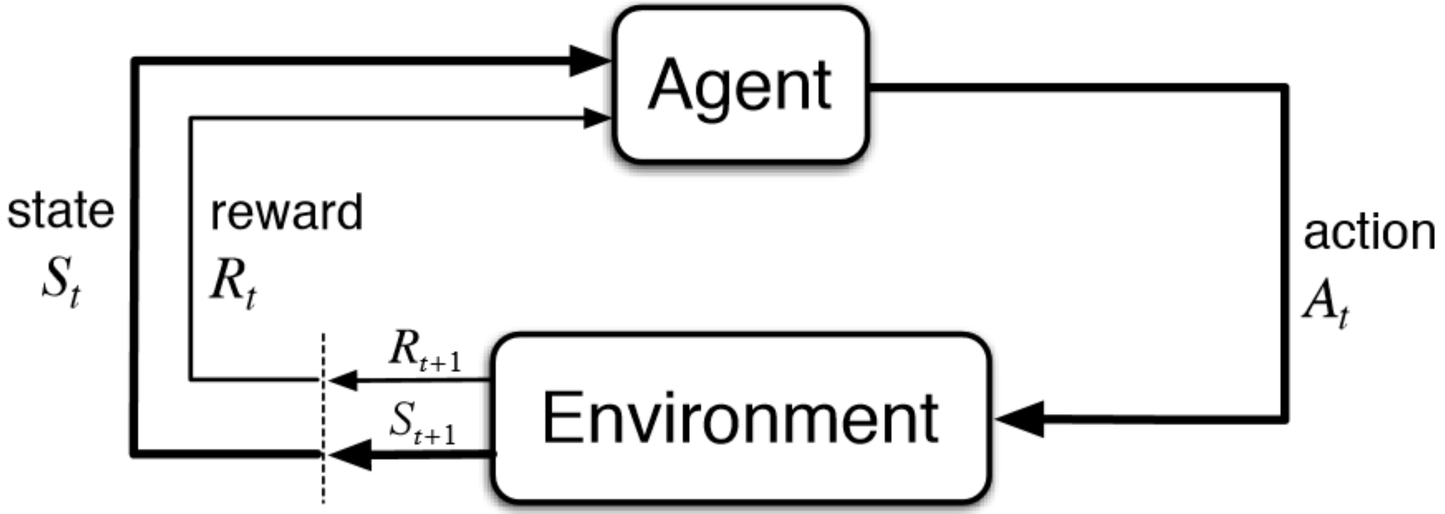
- Things to improve on:
    - Homework: Homework open-endedness / difficulty / fun
    - Communication: Piazza response time
    - Lecture: More detail on slides, increase pace, math can be a challenge
- Liked so far:
    - Lecture: complementing readings, improves understanding
    - Homework: improved understanding of topics
    - Communication: Professor availability, able to find information on the website

Learning Outcomes

After today's lecture, you will:

- Be able to identify key classes of RL methods for robot control problems.
- Be able to identify key challenges of RL in robotics and describe solutions to these challenges.

General Reinforcement Learning



$$\ldots S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}, \ldots$$

$$S_{t+1}, R_{t+1} \sim p(\cdot \mid S_t, A_t)$$

$$A_{t+1} \leftarrow \pi(S_{t+1})$$

Q-Learning

- Q-learning: an alternative to SARSA:

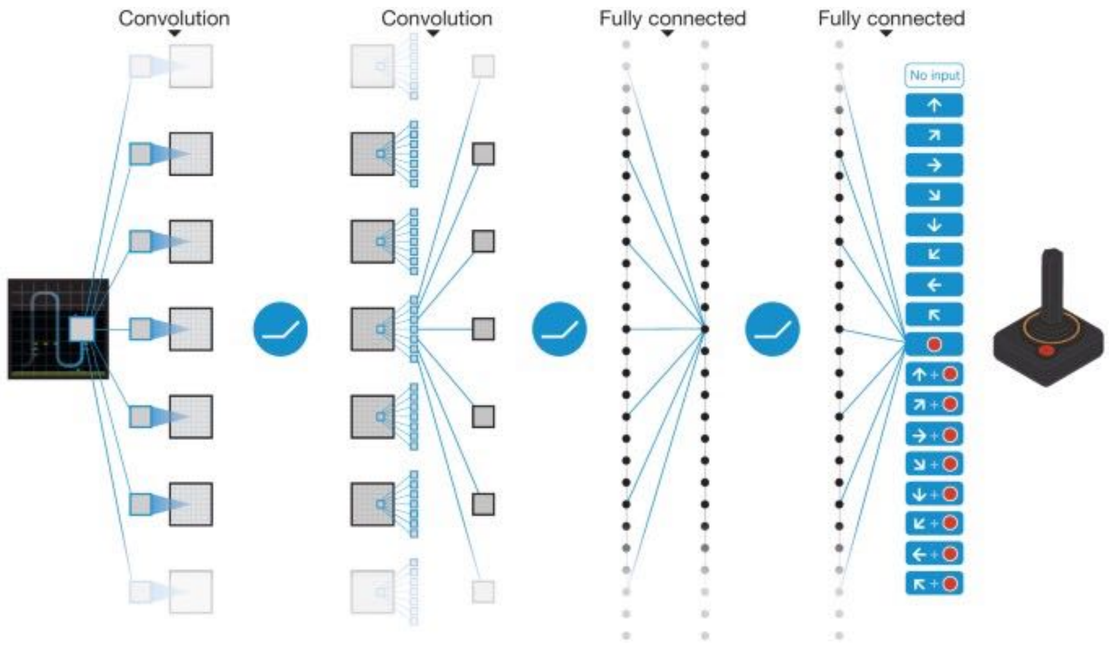
$$q_{k+1}(S_t, A_t) \leftarrow q_k(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_{a'} q_k(S_{t+1}, a') - q_k(S_t, A_t)]$$

- Converges to $q^\star$ if data comes from any sufficiently exploratory exploration policy.
- On-policy vs off-policy?
- The underlying algorithm for Deep Q-networks, which was a landmark result in the history of RL.

Derivative algorithms: DQN, Rainbow, BBF, QT-Opt

Deep Q-learning

- Traditionally, RL algorithms are developed with a tabular representation in mind.
  - Value functions and policies are represented as look-up tables.
- Need function approximation to scale.
- Modern choice: deep neural networks.



Mnih et al, "Human-level control through deep reinforcement learning"

Q-Learning for Continuous Actions

$$q_{k+1}(S_t, A_t) \leftarrow q_k(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_{a'} q_k(S_{t+1}, a') - q_k(S_t, A_t)]$$

- The max over the action space is difficult to compute when actions take on real-values. Why?
- One idea: use optimization to find the best action.
  - Examples: cross-entropy method, gradient ascent
  - But can be slow.
- Another idea: discretize the action space.
  - Sometimes works but loses precision in control.

Deterministic Actor-Critic

$$q_{k+1}(S_t, A_t) \leftarrow q_k(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_{a'} q_k(S_{t+1}, a') - q_k(S_t, A_t)]$$

- Final idea: learn a *parameterized* policy that outputs the maximizing action.
  - $\mu_\theta(s) \rightarrow a$, e.g., a neural network.
- Learn θ such that $q_k(s, \mu_\theta(s)) \approx \max_a q_k(s, a)$.
  - $\theta_{t+1} \leftarrow \theta_t + \alpha \nabla_\theta q_k(s, \mu_\theta(s))$ $\quad \nabla_\theta q_k(s, \mu_\theta(s)) = \nabla_a q_k(s, a) \nabla_\theta \mu_\theta(s) \mid_{a=\mu_\theta(s)}$
- Actor: the policy μ_θ.
- Critic: the action-value function, trained with SARSA to estimate q_{μ_θ}.

Derivative algorithms: DDPG, TD3, Soft actor-critic

Stochastic Policy Gradient RL

- So far we have only considered deterministic policies ($\pi(s) = a$).
- Policy gradient methods use a differentiable and stochastic policy (e.g., a neural network) and learn policy parameters with gradient ascent.
- $\pi_\theta(a \mid s) = \Pr(A_t = a \mid S_t = s; \theta)$
- $J(\theta) = v_{\pi_\theta}(s_0)$ for some special start state s_0.
- $\theta_{k+1} \leftarrow \theta_k + \alpha \nabla_\theta J(\theta_k)$

Stochastic Policy Gradient Theorem

- $J(\theta) := v_{\pi_\theta}(s_0) = \sum_a \pi_\theta(a \mid s_0) \sum_{s',r} p(s', r \mid s_0, a)[r + \gamma v_{\pi_\theta}(s')]$
- $\nabla_\theta J(\theta) \propto \sum_s \sum_a \mu_\theta(s) q_{\pi_\theta}(s, a) \nabla_\theta \pi_\theta(a \mid s)$
- The direction in which an infinitesimally small change to θ produces the maximum increase in $J(\theta)$.
- $\nabla_\theta J(\theta)$ does not depend on any gradients of the state transition function, p.

REINFORCE

- $\nabla_\theta J(\theta)$ can only be estimated.
- $\nabla_\theta J(\theta) \propto \mathbf{E}[\nabla_\theta \log \pi_\theta(A_t \mid S_t) q_\pi(S_t, A_t)] \approx q_\pi(S_t, A_t) \nabla_\theta \log \pi_\theta(A_t \mid S_t)$
- Finally, replace $q_\pi(S_t, A_t)$ with the return that follows G_t.
- $\theta_{k+1} \leftarrow \theta_k + \alpha G_t \nabla_\theta \log \pi_\theta(A_t \mid S_t)$

Derivative algorithms: TRPO, PPO, GRPO

REINFORCE

REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
  Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ (G_t)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$

Usually dropped in practice

Is the policy gradient a gradient? Nota and Thomas. 2020.
Bias in Natural Actor-Critic Algorithms. Thomas. 2014.

Actor-Critic Methods

- REINFORCE uses the return following an action to determine which actions are reinforced.
- Actor-critic methods use learned value functions to drive policy changes.
  - Actor: the policy.
  - Critic: value function (trained to minimize prediction error).
  - $\theta_{t+1} \leftarrow \theta_t + \alpha \delta_t \nabla_\theta \log \pi(A_t \mid S_t)$ $\quad\quad$ $\delta_t \leftarrow R_{t+1} + \gamma \hat{v}(S_{t+1}) - \hat{v}(S_t)$

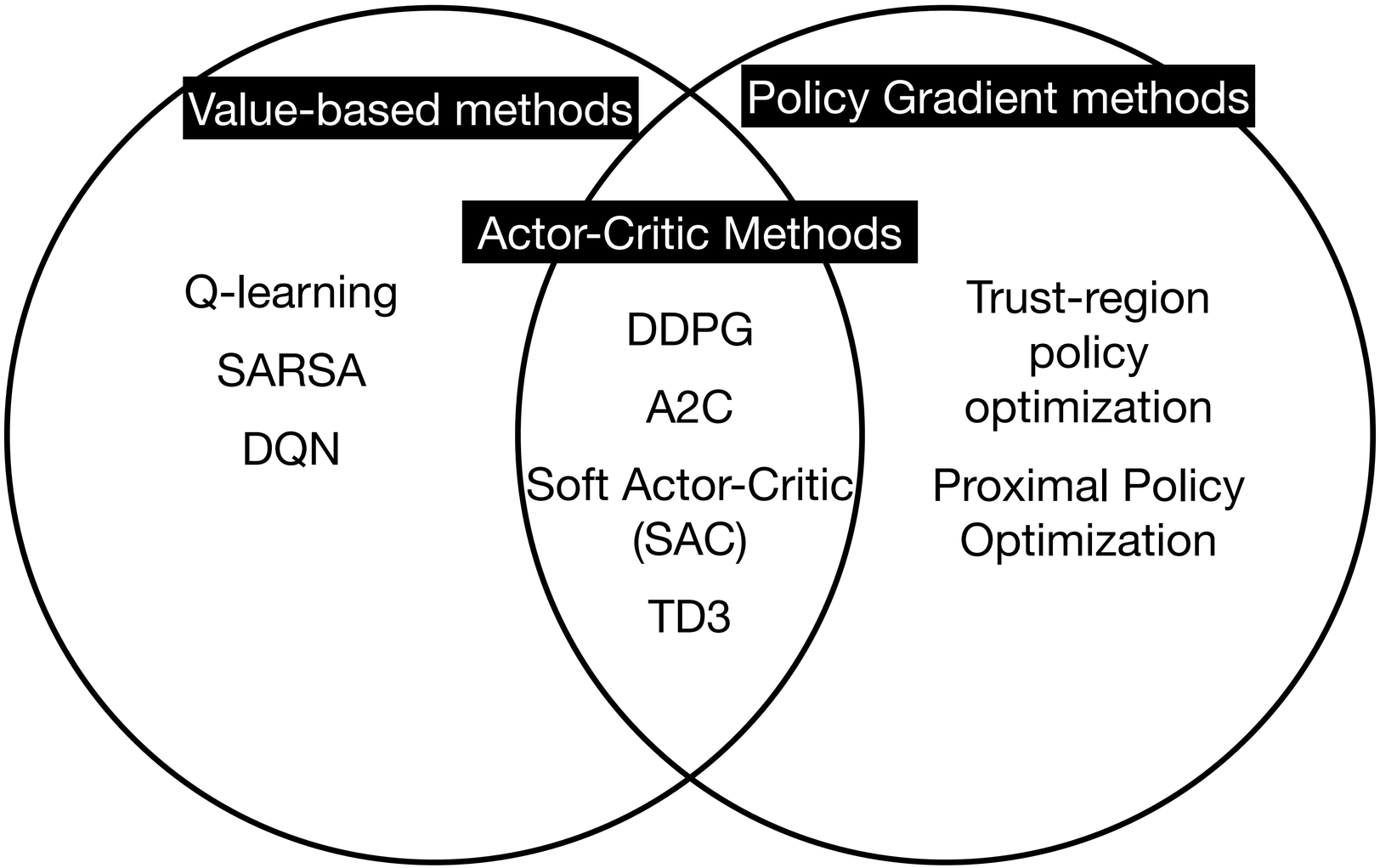
Derivative algorithms: A2C, soft actor-critic, PPO

Actor-Critic Methods

One-step Actor–Critic (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$ (e.g., to $\mathbf{0}$)
Loop forever (for each episode):
  Initialize S (first state of episode)
  $I \leftarrow 1$
  Loop while S is not terminal (for each time step):
    $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
    Take action A, observe S', R
    $\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla \ln \pi(A|S,\boldsymbol{\theta})$
    ~~$I \leftarrow \gamma I$~~
    $S \leftarrow S'$

Model-Free RL



Challenges in RL for Robotics

Problem: where do rewards come from?

Need to balance defining the objective with learnability (may require “shaping” rewards).

Solution:

- Manually tuning a reward by hand.

- Inverse reinforcement learning: infer a reward from demonstrations.

- Potential-based shaping to preserve optimal policy.

Challenges in RL for Robotics

Problem: how many actions must be taken before an optimal policy is learned?

Solution:

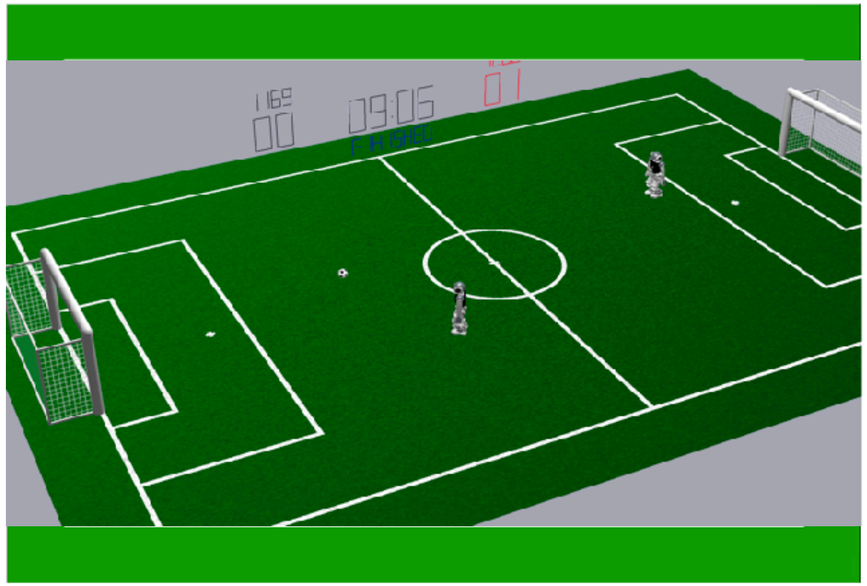
- Learn in a simulator and transfer to the physical robot.

- Initialize the policy well.

- Data augmentation

Multi-fidelity Sim2Real

Generate diverse situations for robust behaviors

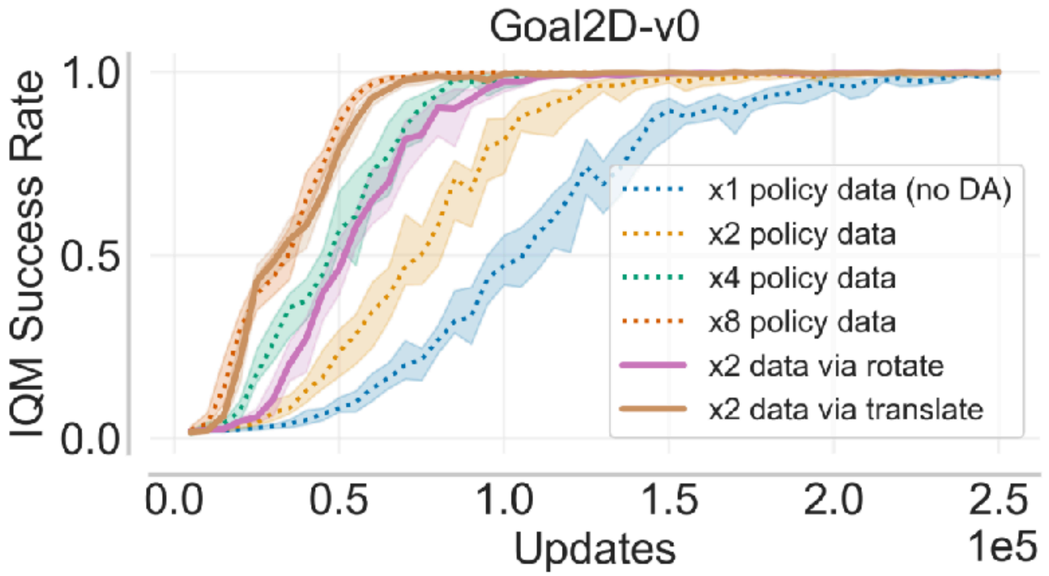


Abstract away low-level details for scalability

Train in simulator, deploy to the real robot.

Data Augmentation

Corrado et al. 2024 (RLC)



Corrado and Hanna 2024 (ICLR)

Challenges in RL for Robotics

Problem: actions during learning may harm the robot or those around it.

Solution:

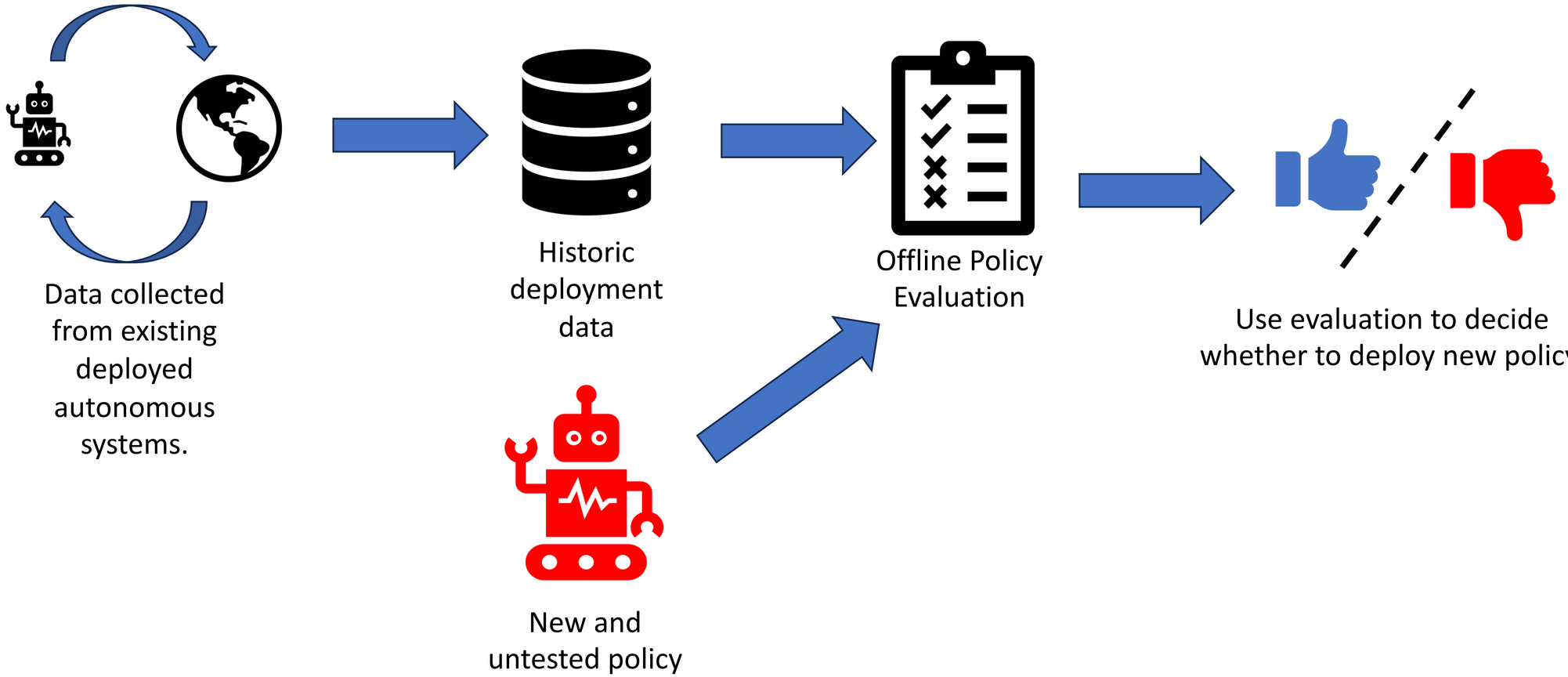
- Learn in a simulator and transfer to the physical robot.

- Offline reinforcement learning (no interaction during learning)

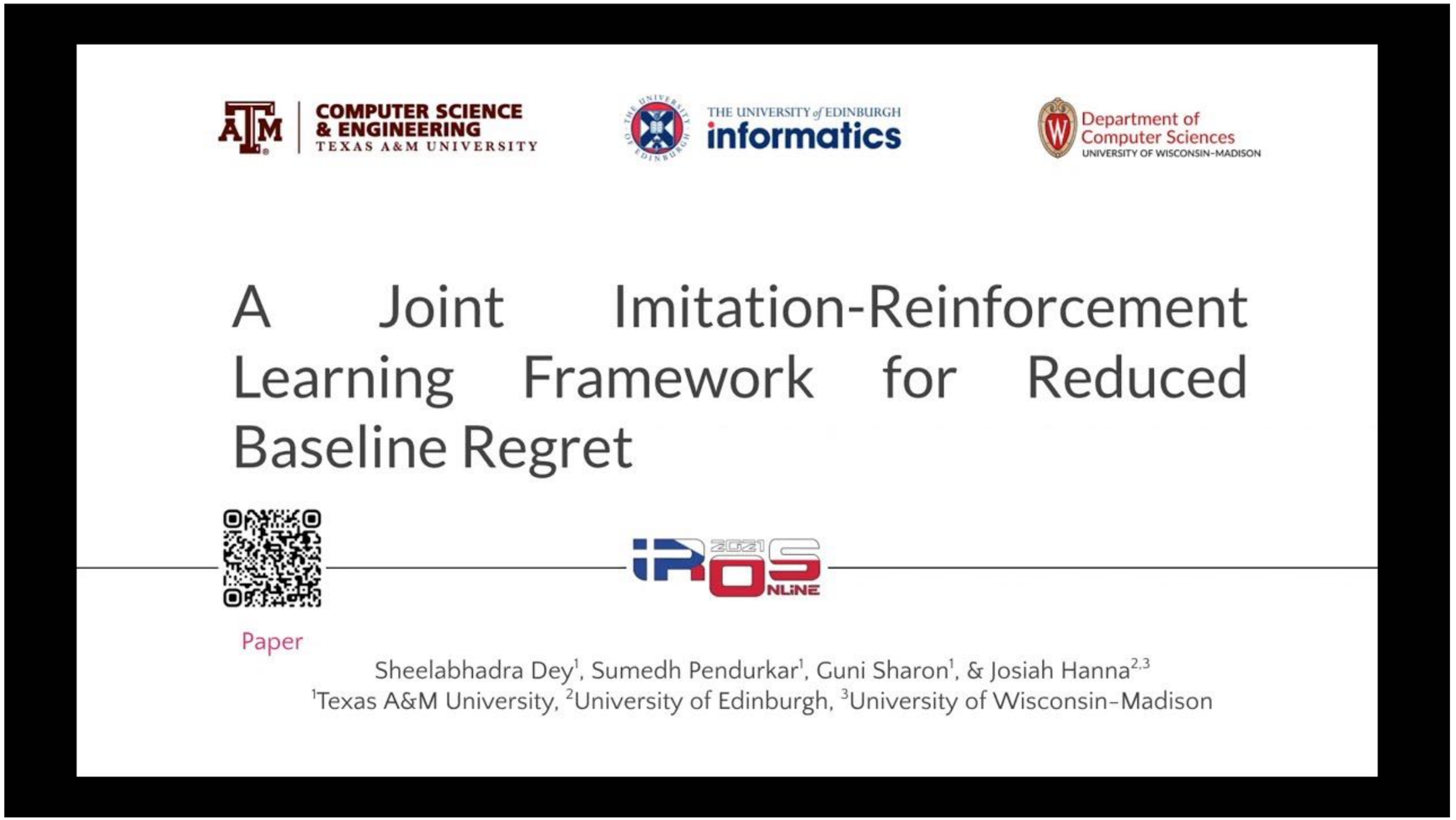
- Off-policy evaluation

- Add safety layers into a robot’s control architecture

Offline Policy Evaluation



Safety

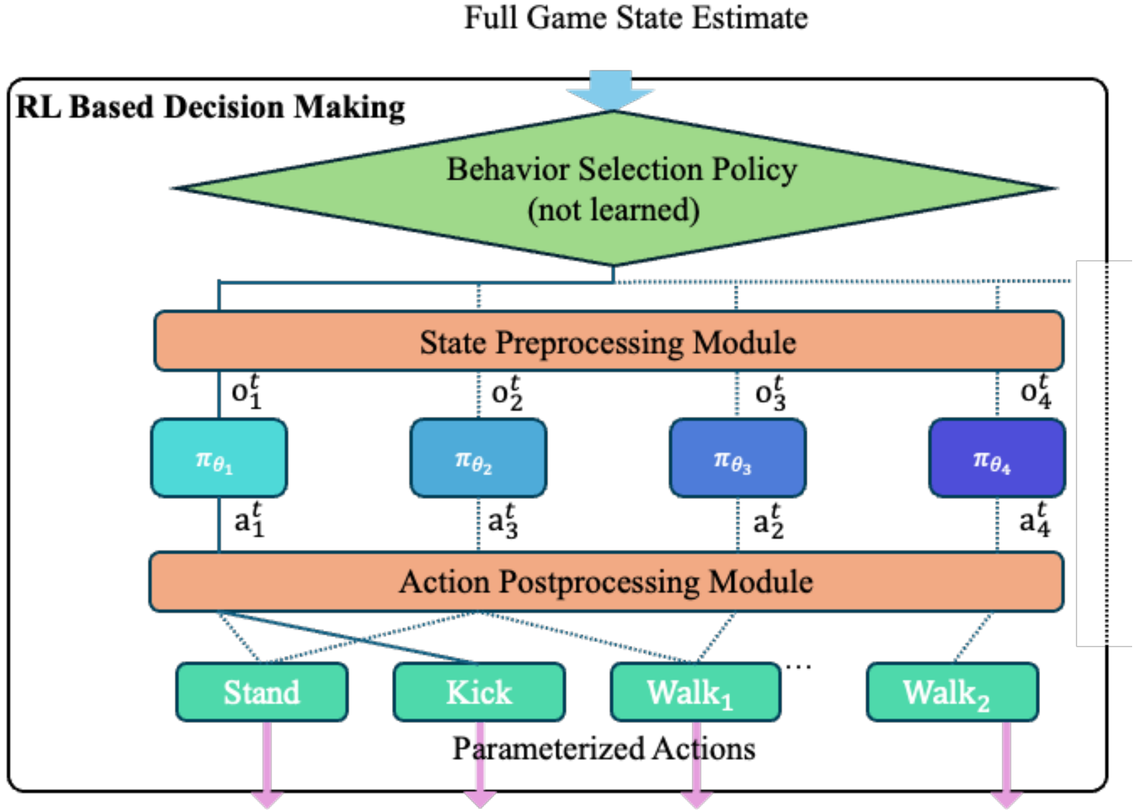


A Joint Imitation-Reinforcement Learning Framework for Reduced Baseline Regret. Dey et al. 2021.

Case Study: Robot Locomotion

A Walk in the Park: Learning to Walk in 20 Minutes With Model-Free Reinforcement Learning. Smith et al. 2022.

Case Study: Robot Soccer



Summary

Today we covered:

1. Basic algorithms for RL in robotics.
2. Discussion of advantages and challenges with using RL.
3. Some case studies of RL in robotics.

Action Items

Complete homework 4

Begin robot learning reading