ECE7121 Learning-based control – 2025 Fall

Inverse RL and Curriculum learning

INHA UNIVERSITY

Inverse RL

- Offline RL: learning from suboptimal or mixed demonstrations
  - reward labeling: at least, we know the relative goodness
- Imitation learning: copy the good demonstrations
  - no reward labeling

- Inverse RL: a type of learning from good demonstrations
  - similar to IL, no reward available
  - learn a reward function such that

$$\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}_{\pi,p}[r^*(s,a)]$$

  - learn policy given the reward

Inverse RL

> Why don't we directly learn a policy?
  - assumes learning the reward function is statistically easier than directly learning the policy
  - (perhaps) closer to how human learns

Inverse RL

> Why don't we directly learn a policy?
  - more interpretable (we can reason what an expert is trying to do)
  - can generalize or adapt to new environments via RL
  - manually design reward function can be difficult

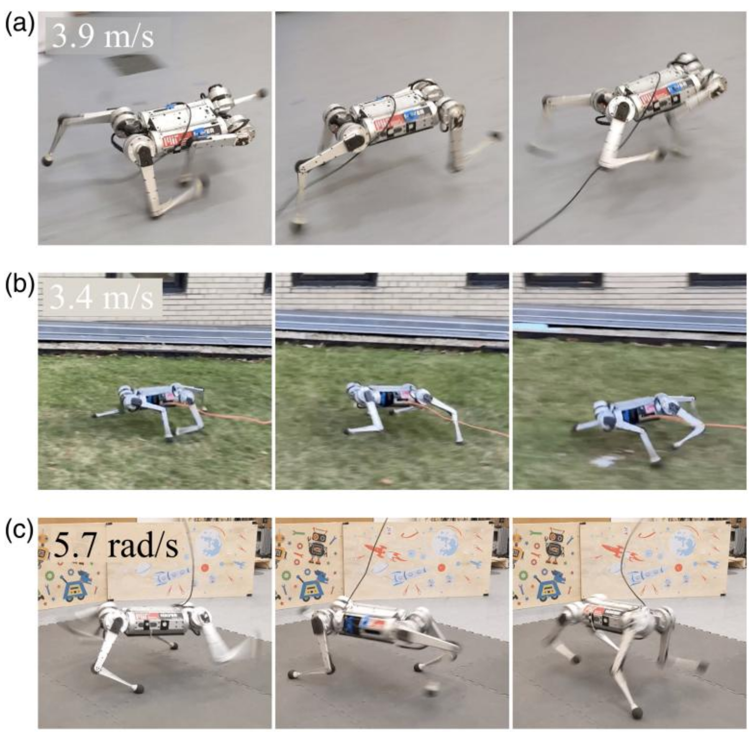


Table 1. Reward terms for task, stability, and smoothness. We adapt the reward structure from Rudin et al. (2021) to our robot and task.

Term	Equation	Weight
$r_{v_{xy}^{\text{cmd}}}$: xy-vel	$\exp\{-\lvert\mathbf{v}_{xy} - \mathbf{v}_{xy}^{\text{cmd}}\rvert^2/\sigma_{vxy}\}$	0.02
$r_{\omega_z^{\text{cmd}}}$: yaw-vel	$\exp\{-(\boldsymbol{\omega}_z - \boldsymbol{\omega}_z^{\text{cmd}})^2/\sigma_{\omega z}\}$	0.01
z vel	$\mathbf{v}_z^2$	-0.04
roll-pitch vel	$\lvert\boldsymbol{\omega}_{xy}\rvert^2$	-0.001
height	$(h - h^0)^2$	-0.6
orientation	$\lvert\mathbf{g}_{xy}^{\text{ori}}\rvert^2$	-0.002
self-collision	$\mathbf{1}_{\text{selfcollision}}$	-0.02
joint limit	$\mathbf{1}_{q_i > q_{max} \lvert\lvert q_i < q_{min}}$	-0.2
joint torques	$\lvert\boldsymbol{\tau}\rvert^2$	-2e-7
joint accel	$\lvert\ddot{\mathbf{q}}\rvert^2$	-5e-9
action rate	$\lvert\mathbf{a}_{t-1} - \mathbf{a}_t\rvert^2$	-2e-4
foot airtime	$\sum t_{air} * \mathbf{1}_{\text{new contact}}$	0.02

Inverse RL

> Why should we worry about learning rewards?

Computer Games

reward

Mnih et al. '15

Real World Scenarios

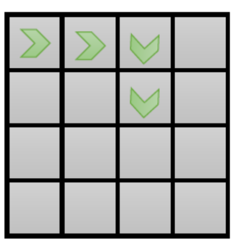
robotics

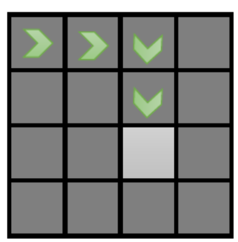
dialog

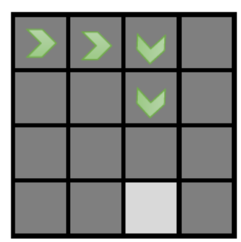
autonomous driving

what is the reward?
often use a proxy

- might take very different actions
- by itself, this is an underspecified problem
  (many reward function can explain the same behavior)







Linear reward formulations

- Commonly used in IRL algorithms
  - easier to optimize
  - θ: unknown weight, ϕ: known features

$$R_\theta(s) = \theta_1 \phi_1(s) + \cdots + \theta_d \phi_d(s) = \boldsymbol{\theta}^\top \phi(s)$$

  - Why linear?
    - value function is linear in expected features
    - $V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r\left(s^{(t)}\right) | s^{(0)} = s\right]$

$$= \boldsymbol{\theta}^\top \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \phi\left(s^{(t)}\right) | s^{(0)} = s\right]$$

$$= \boldsymbol{\theta}^\top \mu^\pi(s)$$

Linear reward formulations

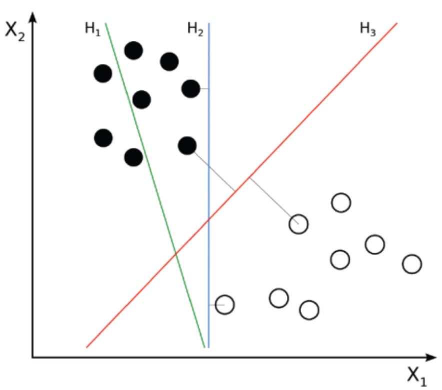
> Key idea: the expert is better than other policies
  - other policies π_n: RL, manually designed, etc
  - Linear program IRL (LP-IRL)

$$\hat{\theta} = \arg\max_{\theta}\{\sum_i \theta^\top(\mu^{\pi^*} - \mu^{\pi_i})\} \;\; \text{s.t.} \;\; |\theta_d| \leq 1 \; \forall d$$

    - this process can be iterative: solve θ, train new π_n, solve new θ, ...

  - Max margin IRL (apprenticeship learning)
    - adopt the max margin philosophy from SVM

$$\min_{\theta} \lambda\|\theta\|_2 + \sum_i \xi_i \;\; \text{s.t.} \;\; \theta^\top(\mu^{\pi^*} - \mu^{\pi_i}) \geq 1 - \xi_i$$



Linear reward formulations

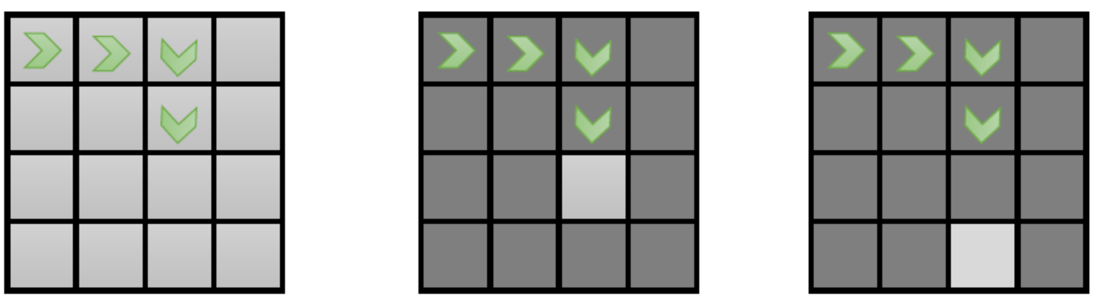
- Max margin IRL (apprenticeship learning)
  - applications in autonomous helicopter aerobatics (2010, IJRR)
    - first, obtain reward function from expert demonstrations
    - obtain policy using Gauss-Newton LQR under real helicopter dynamics
    - shows in-place flips, in-place rolls, loops and hurricanes, auto-rotation landings, chaos and tic-tocs

Linear reward formulations

- Max entropy IRL (MaxEnt IRL)
  - IRL is an ill-posed problem
    - many reward functions correspond to the same policy
    - many stochastic mixtures of policies correspond to the same feature expectation
  - Maximum entropy principle
    - The probability distribution which best represents the current state of knowledge is the one with largest entropy
    - The distribution with largest policy (uncertainty) makes the least assumptions about the true distribution
    - $\max_p \{-\sum p(\tau) \log p(\tau)\}$

Linear reward formulations

> Max entropy IRL (MaxEnt IRL)
  - Given known constraints (prior data), choose the distribution that satisfies them while maximizing entropy
    - unknown parts are random → most random is the largest entropy
    - as random as possible while matching features
    - ex) suppose the probability of rain tomorrow is 30%, and we know nothing else
    - ex) IRL example: maximize the entropy of trajectory distribution $p(\tau)$



  - Given the reward function, obtain the stochastic (soft-optimal) policy via the soft Bellman updates, then match feature expectations

Linear reward formulations

- Max entropy IRL (MaxEnt IRL)
  - Maximizing the entropy → maximize the likelihood of the observed data under the maximum entropy distribution
  - from trajectories to states: successful imitation boils down to learning a policy that matches the state (or state/action) visitation distribution
  - Algorithm (with known dynamics, linear costs)

0. Initialize θ, gather demonstrations $\mathcal{D}$
1. Solve for optimal policy $\pi(a|s)$ w.r.t. c_θ with value iteration
2. Solve for state visitation frequencies $p(s \mid \theta, T)$
3. Compute gradient $\nabla_\theta \mathcal{L} = \frac{1}{M} \sum_{\tau_d \in \mathcal{D}} \mathbf{f}_{\tau_d} - \sum_s p(s \mid \theta, T) \mathbf{f}_s$
4. Update θ with one gradient step using $\nabla_\theta \mathcal{L}$

Linear reward formulations

> MaxEnt IRL requires
  - solving for (soft) optimal policy in the inner loop
  - enumerating all state-action tuples for visitation frequency and gradient
  - reward was assumed linear over features ϕ

    → general function approximations for the reward

> To apply this in practical problem settings, we need to handle
  - large and continuous state and action spaces
  - states obtained via sampling only
  - unknown dynamics

    → sampled based approximations for the partition function

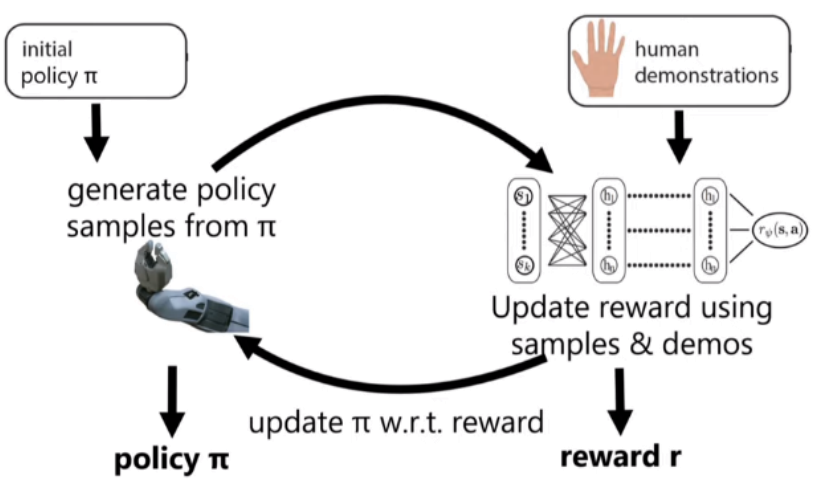
GAN framework

> IRL framework

Expert demo → learn reward → optimize policy (RL)

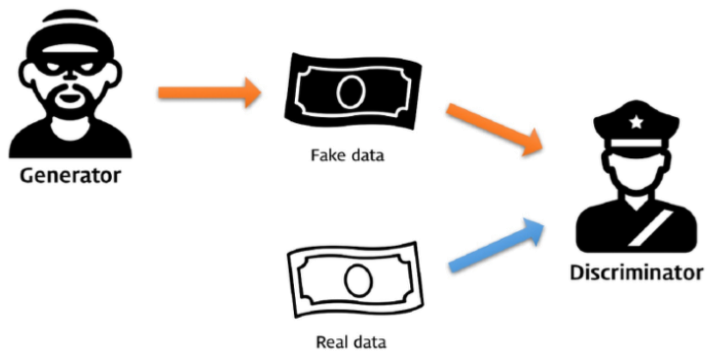
compare policy → guide reward (cost) learning

> Guided cost learning (2016 ICML)



GAN framework

- Generative adversarial network (GAN)
  - train generator and discriminator iteratively



- Inverse RL as a GAN
  - policy changed to make it harder to distinguish from demos

learn reward → optimize policy (RL) → Discriminator

Expert demo → Discriminator

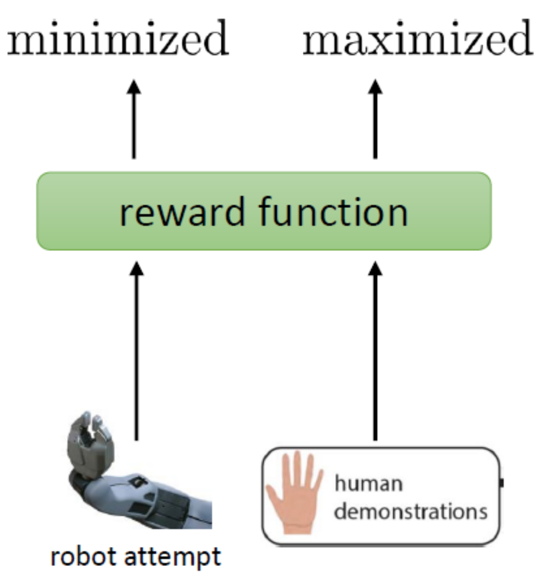
compare policy

GAN framework

> Generative adversarial imitation learning (GAIL, 2016)
  - discriminator learns to tell expert state action pairs from the agent's
  - end-to-end imitation without hand-crafted rewards
  - often simpler to set up optimization

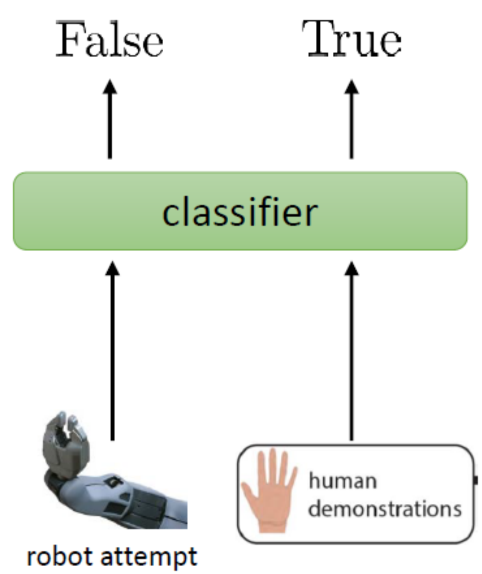
Guided Cost Learning

Finn et al., ICML 2016



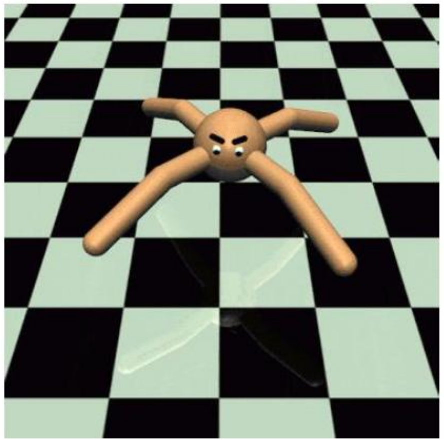
Generative Adversarial Imitation Learning

Ho & Ermon, NIPS 2016

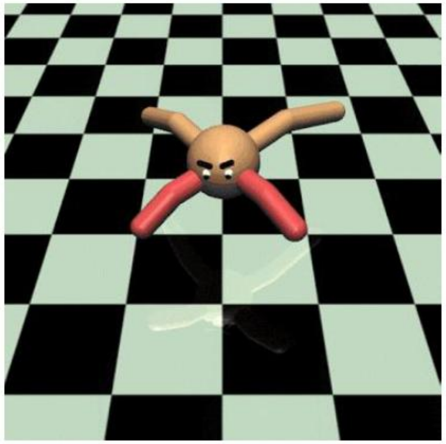


Generalization via inverse RL

> What can we learn from the demo to enable better transfer?
  - need to decouple the goal from the dynamics
  - policy = reward + dynamics



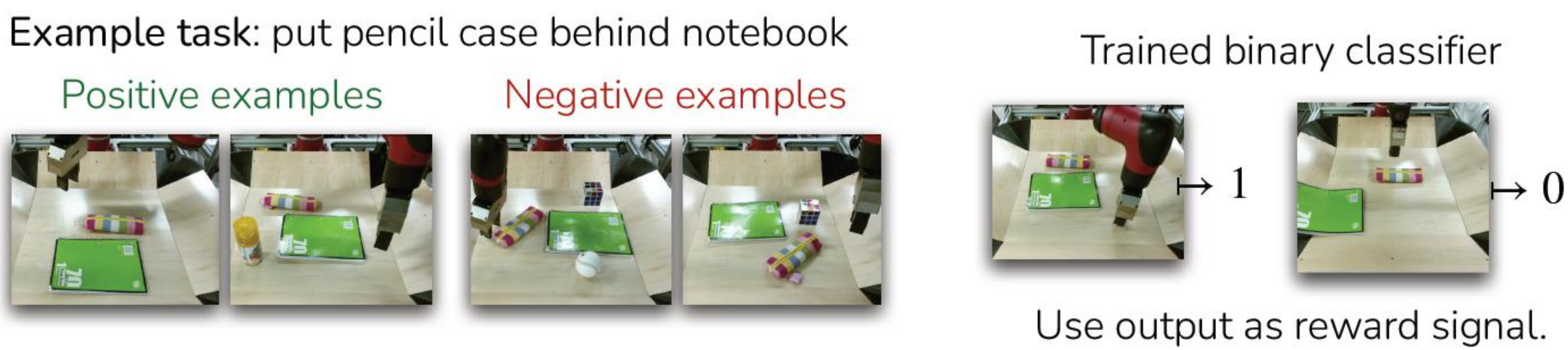
demos



reproduce behavior
under different conditions

Goal classifier

> Reward can be explicitly trained if we know the objective
  - learn to discern goal states from other states

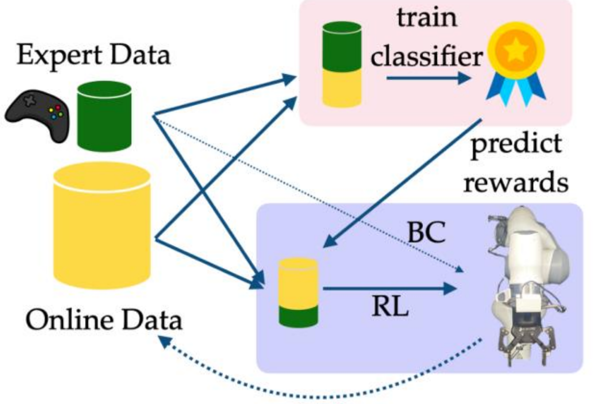


  - collect examples of successful / unsuccessful states
  - train binary classifier
  - run RL with classifier as reward

Goal classifier

> Reward can be explicitly trained if we know the objective
  - RL algorithm will seek out states that the classifier thinks are good
  - It may simply find states that the classifier wasn't trained on
    - exploiting the classifiers weakness

  - Add states that RL visits as negative examples for the classifier
    - classifier can't be exploited (update the classifier during RL)
    - what if some of the visited states are successful?
    - important to regularize the classifier (or balanced training datasets)



Goal classifier

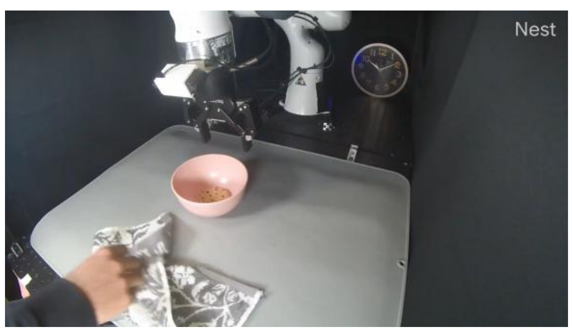
> Goal classifiers for robotic RL example (2023 CoRL)
  - collect 50 demos, use final states as success state examples
  - initialize RL replay buffer with demos

BC (26/12% success rate)

MEDAL++ (62/46% success rate)

Cloth hanging

Bowl cloth cover



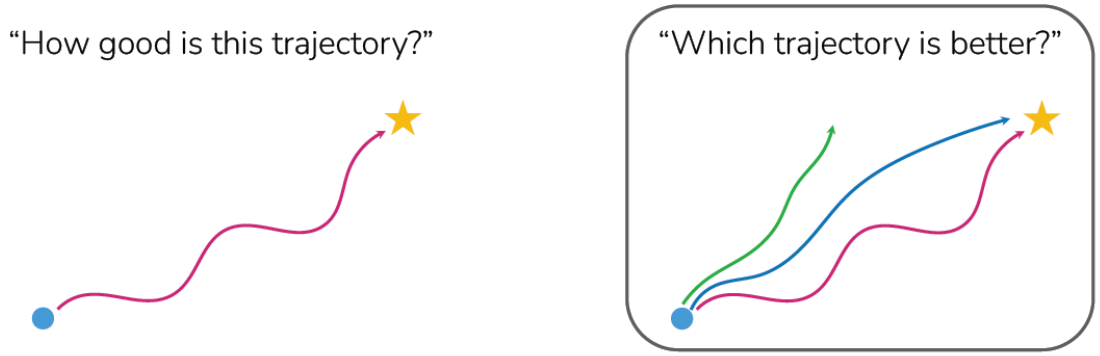
Goal classifier

- Pros and cons
  - practical framework for task specification
  - adversarial training can be unstable
    (though variety of regularization tricks from GAN literature)
  - requires examples of desired behavior or outcomes

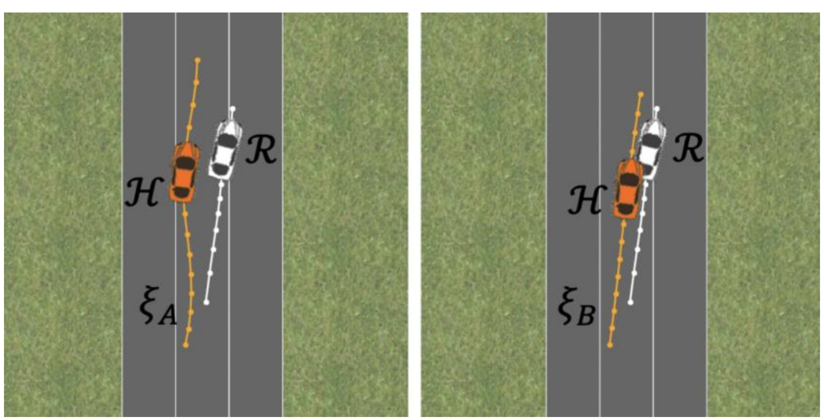
RLHF

> Can humans provide feedback on policy roll-outs?
  - instead of requiring demos or example goals

A couple options

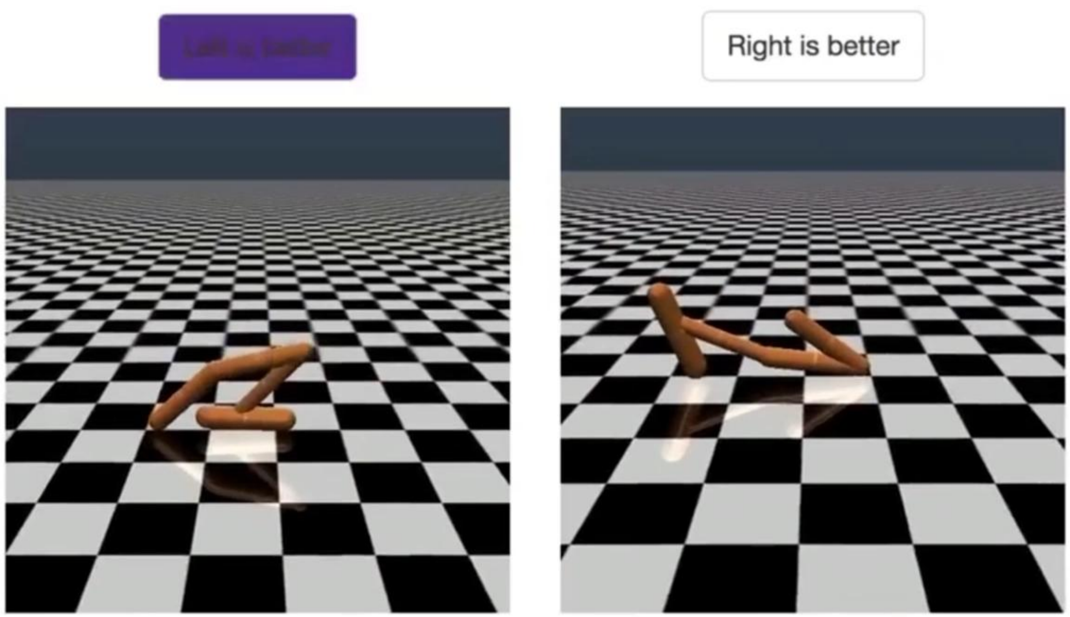


Relative **preferences** are easier to provide!



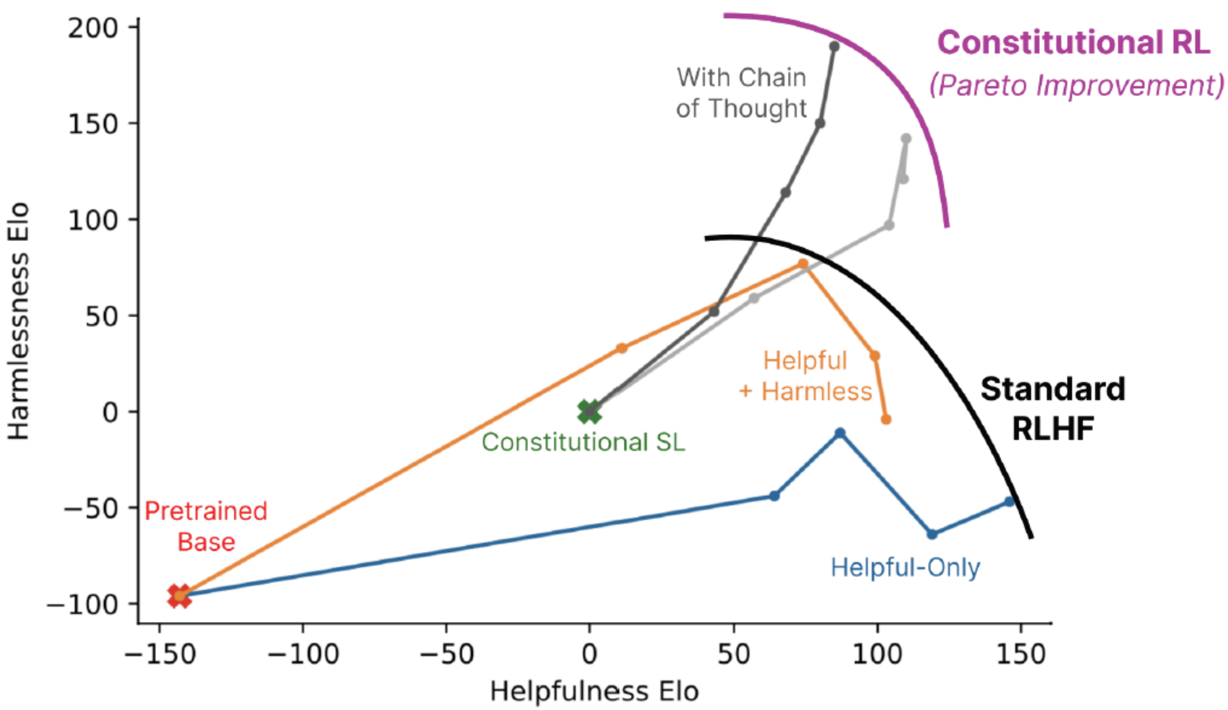
RLHF

> RL from human feedback (RLHF)
  - often easier for people to make than handwriting a reward function
  - often easier than providing scalar reward
    (how much do you like this ad?)
  - needed 900 bits of feedback from a human evaluator to learn to backflip



RLHF

> RL with AI feedback (RLAIF)
  - ask another language model ‘which of these responses is less harmful?’

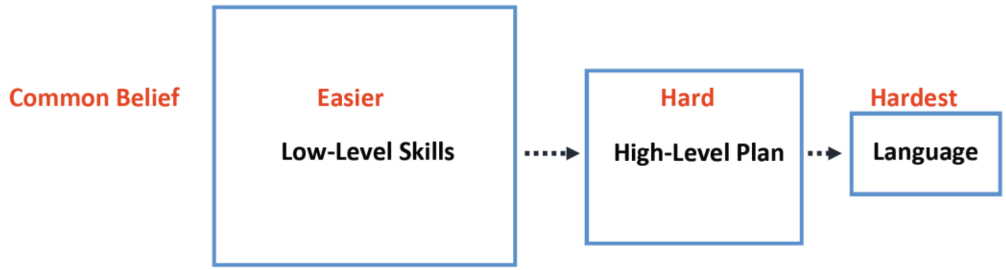


RLHF

> Why human feedback matters in control
  - Unmodeled objectives: comfort (low jerk), noise, safety margins, aesthetics—hard to hand-code in reward functions
    - social navigation
  - Sim-to-real gaps: policies that track in sim can feel unsafe/annoying in reality
  - Contextual trade-offs: users/environments change the weighting of goals; human preferences adapt the objective online.

Curriculum learning

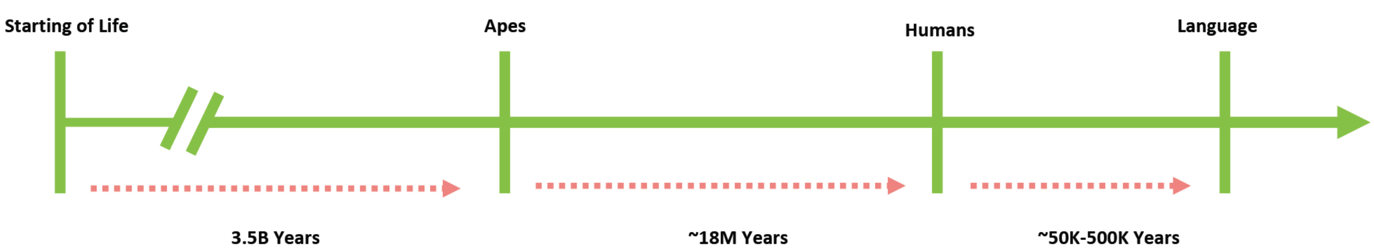
> Training an agent with tasks of gradually increasing difficulty
  - simpler skills are mastered first and then built upon to solve complex tasks



  - ex) learn how to walk (safe and easy env. → dangerous env.)



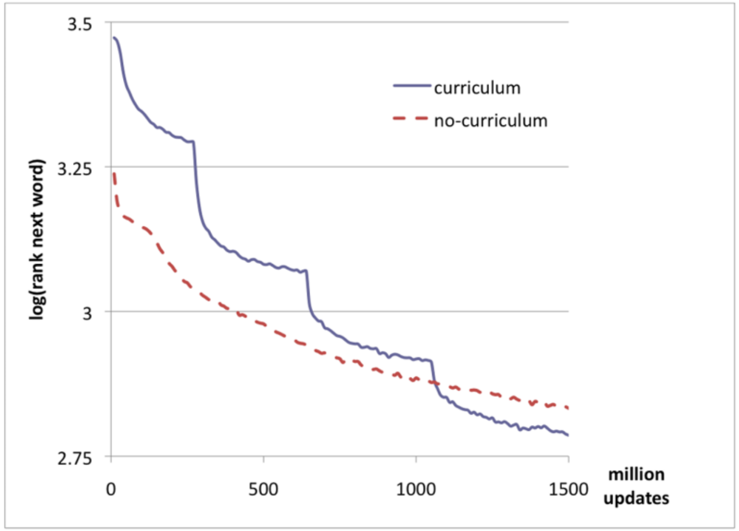
  - ex) evolution of life on earth

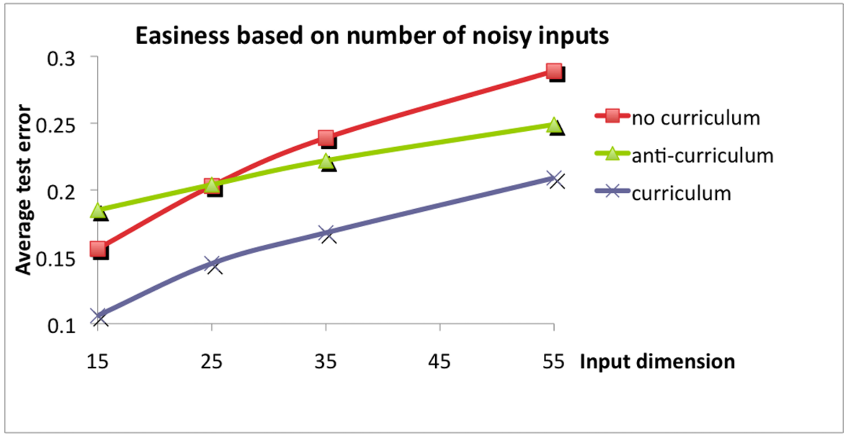


Curriculum learning

> The importance of starting small (1993)
  - Humans display an exceptional capacity to learn
  - Humans are remarkable for the unusually long time it takes to reach maturity
  - Through culture, learning has created the basis for a non-genetically based transmission of behaviors which may accelerate the evolution of our species

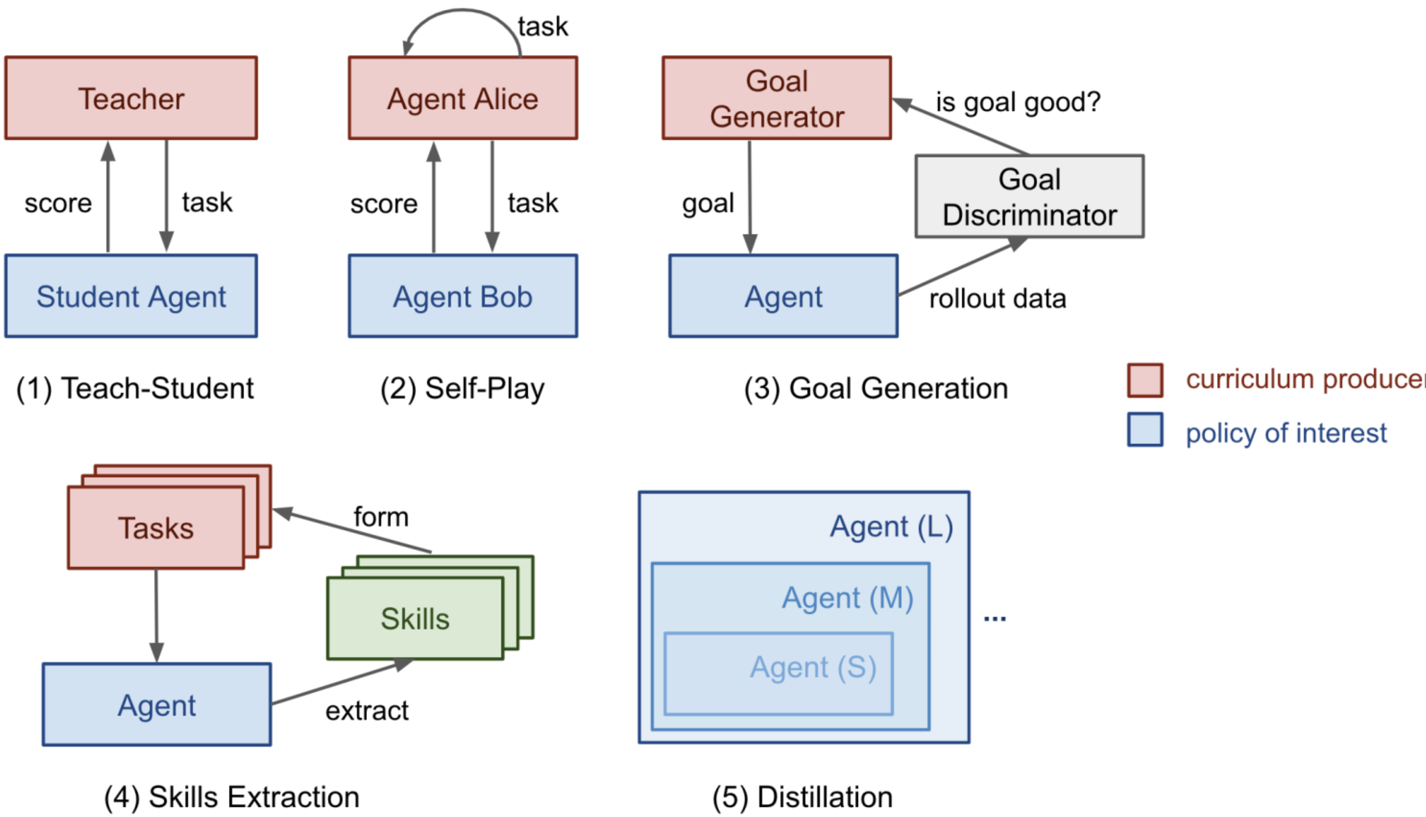
> Curriculum in supervised learning (2009)
  - clarify when and why a curriculum strategy can benefit





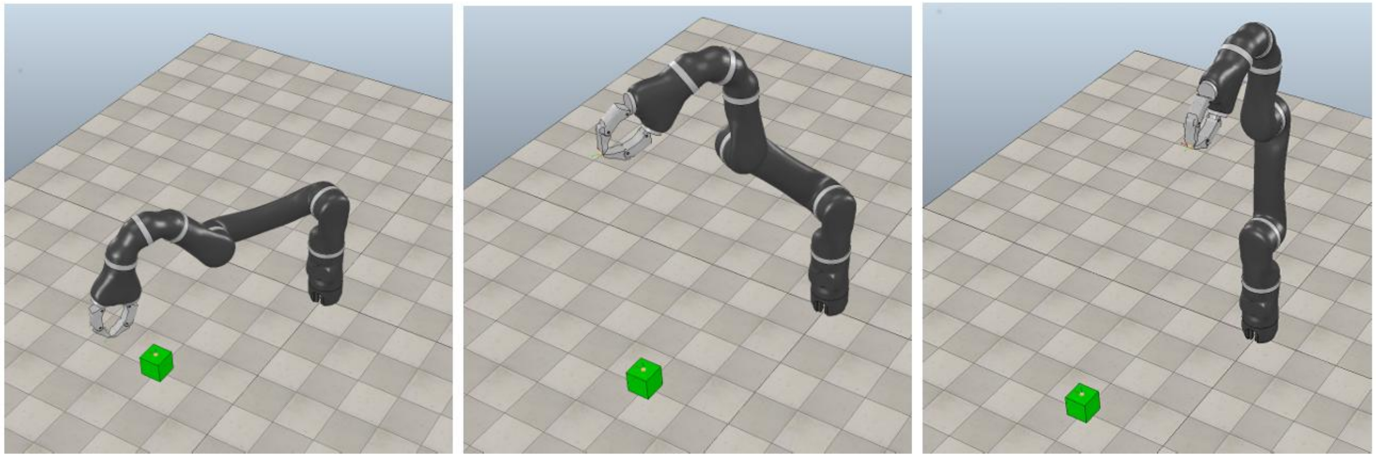
Curriculum learning

> Several types of curriculum learning
  - a special form of transfer learning where the initial tasks are used to guide the learner so that it will perform better on the final task



Curriculum learning

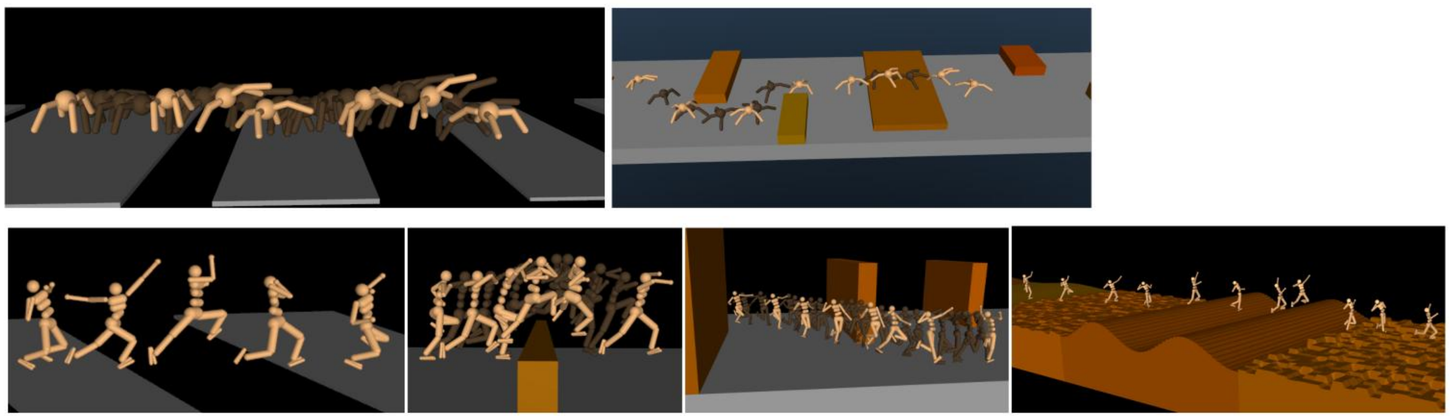
- Curricula for robot tasks
  - exposing the learning agent to a sequence of tasks of increasing difficulty
    - increasing the number of moving joints
    - increasing speeds
    - using easier initial robot configurations
    - change the cost function (increase the coefficient on the torque cost)

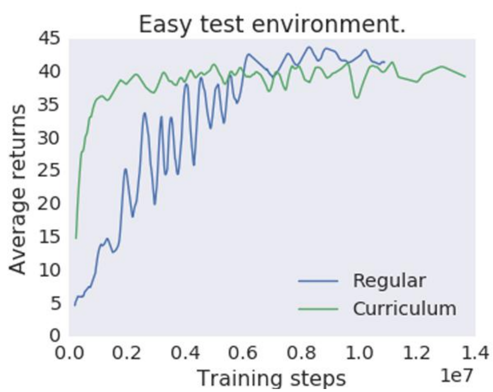


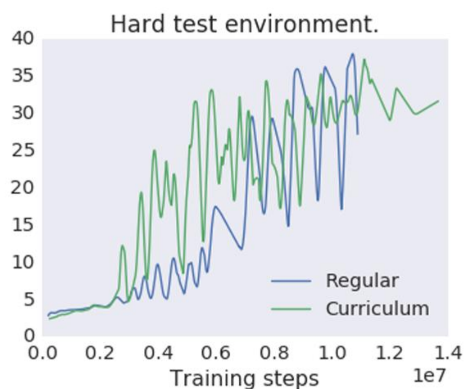
(a) Sub-task 1 **(b)** Sub-task 2 **(c)** Sub-task 3

Curriculum learning

- Curricula for robot tasks
  - change the physics (meta-learning)
  - change the environment

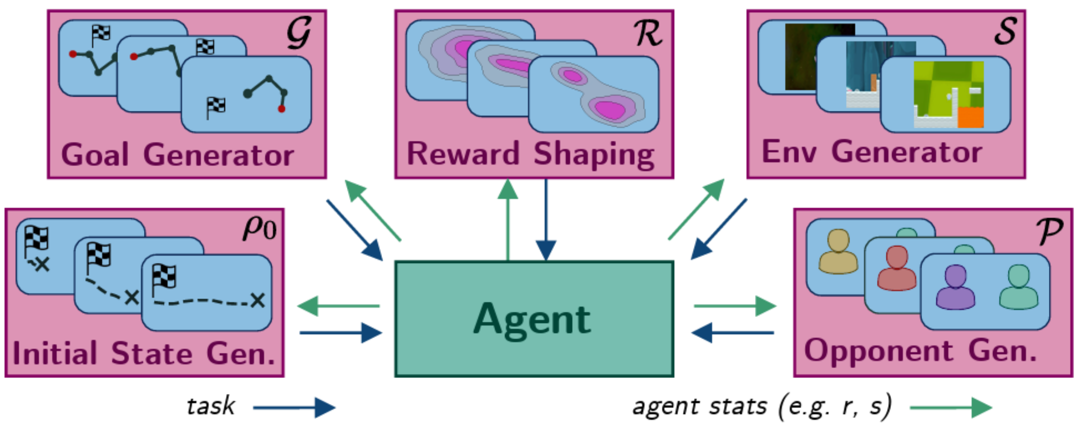






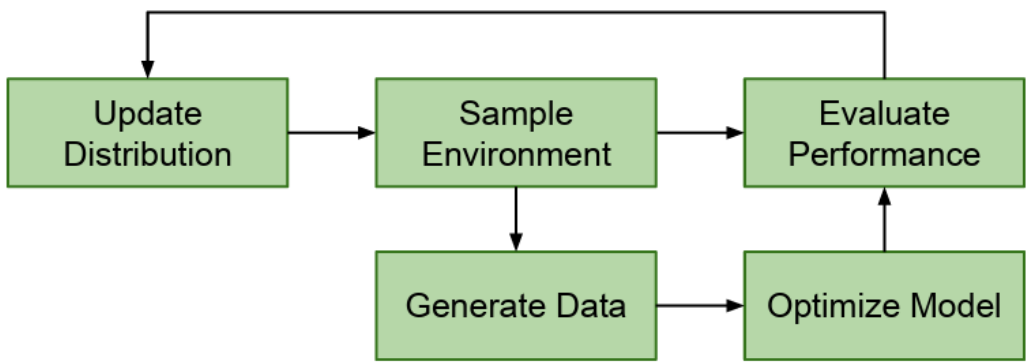
Curriculum learning

- How to automatically generate a curriculum for a given RL agent
  - given metrics of the agent's behavior like performance or visited states, automatic curriculum learning methods generate new tasks
    - initial/target state: start learning from states close to a given target (sparse reward)
    - reward function: curiosity-based approaches, skill discovery
    - environments / opponent (for self-play algorithms)



Curriculum learning

> Ex) Automatic domain randomization (2019)
  - Environment parameters are sampled from independent uniform dist.
  - If agent performance > upper threshold → expand parameter ranges
  - If performance < lower threshold → contract ranges



  - still, there are tuning parameters (update step size)
    - how much to expand/contract the sampling distribution?

Curriculum learning

> Ex) self-paced deep RL (2020)

- ADR relies only on success thresholds
  - need a framework that adapts smoothly and theoretically grounded

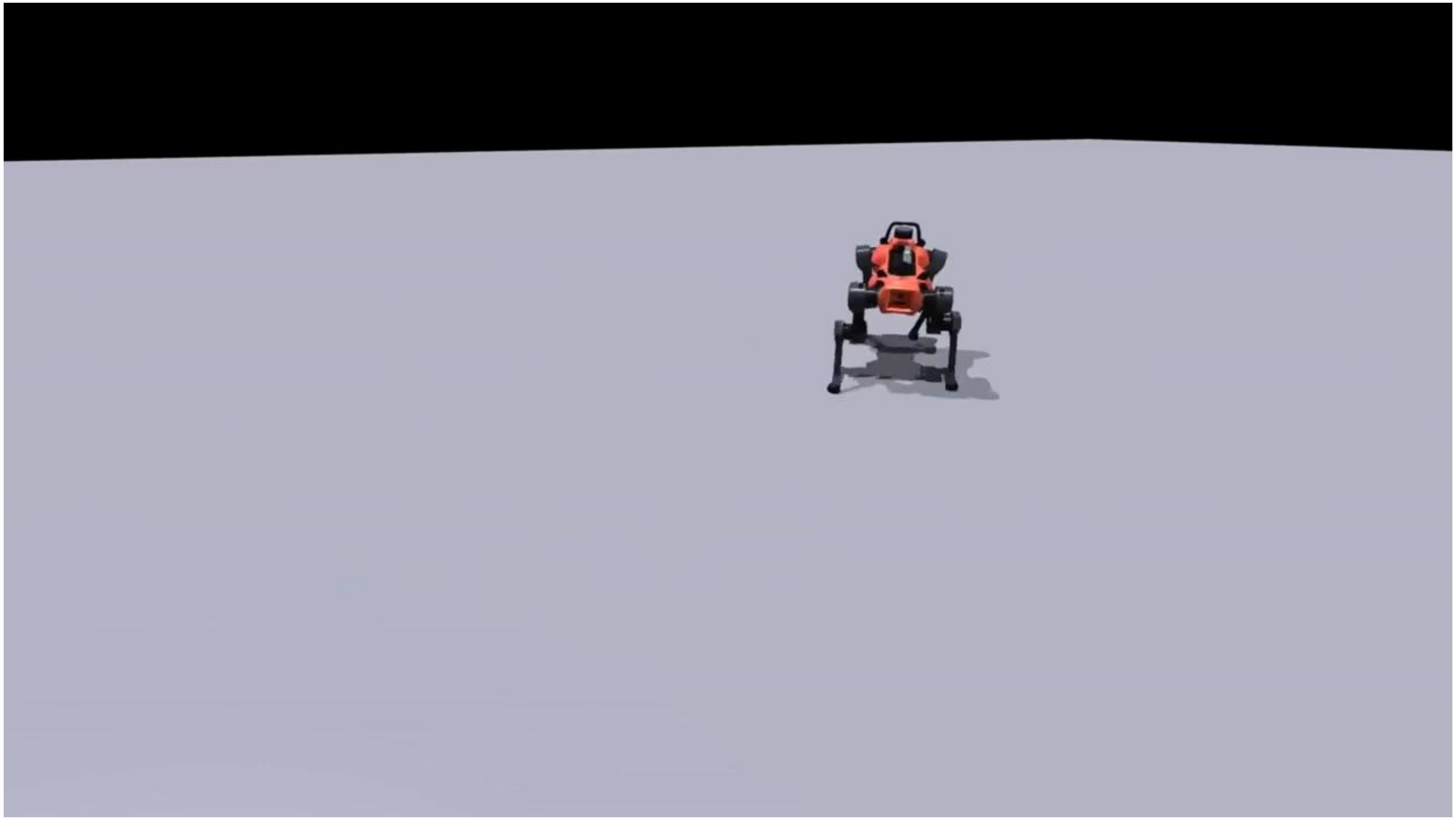
- Key idea: learn a task/context distribution

$$\max_{v,w} J(v,w) - \alpha D_{KL}[p_v(c) \parallel \mu(c)]$$

  $p_v(c)$ is a current task distribution, $\mu(c)$ is a target distribution
- Start with a simple task distribution (easy goals, stable conditions)
- Evaluate the value function → shift distribution toward tasks that improve learning
- Use KL divergence to make the current distribution close to the target distribution gradually

Curriculum learning

> Legged robot walking policy



Reference

- Curriculum learning
  - https://lilianweng.github.io/posts/2020-01-29-curriculum-rl/