Reinforcement Learning: Reward Design (ESE 3990)

Instructor:
Antonio Loquercio

Fall 2026

University of Pennsylvania

Rewards: The Unsung Hero of RL

Where is the reward?

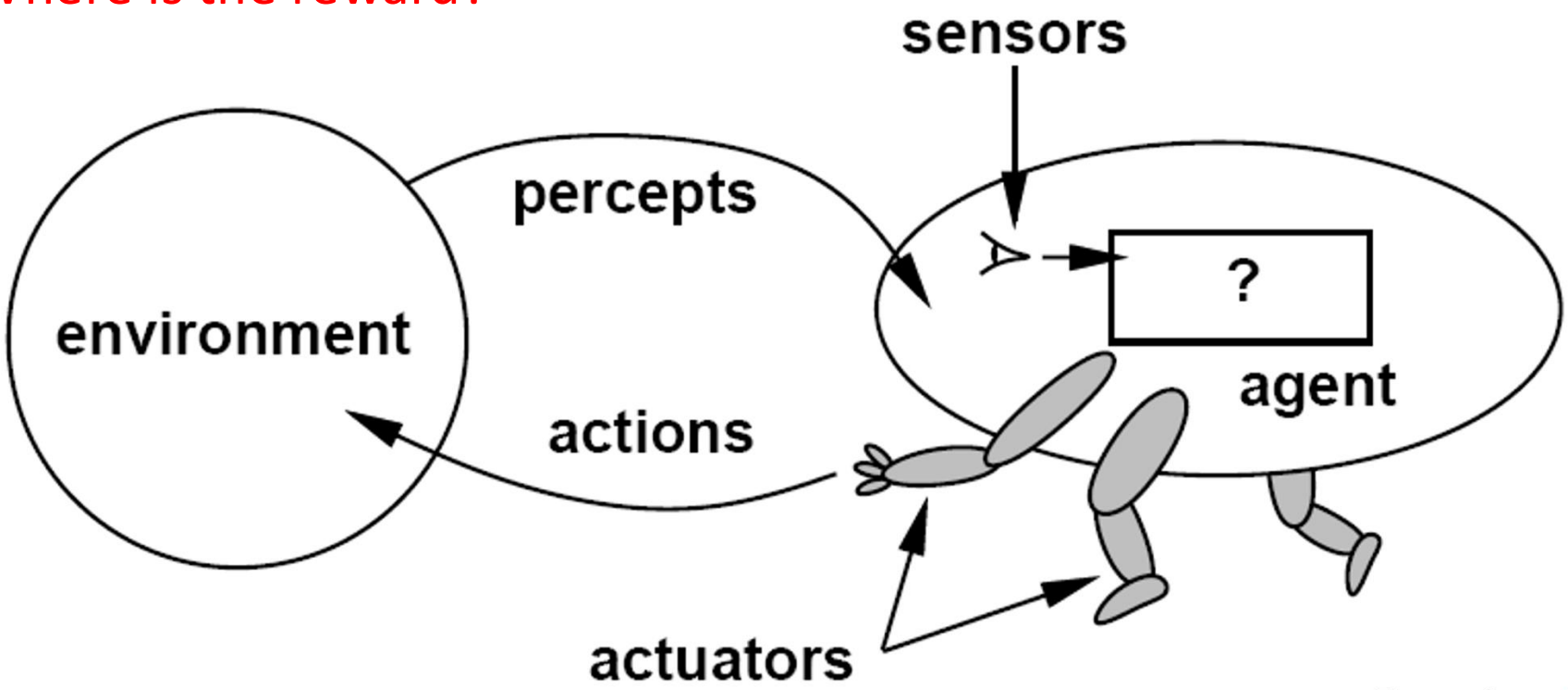


Figure from Russell & Norvig

\begin{Hypothesis}

Reward is enough

David Silver *, Satinder Singh, Doina Precup, Richard S. Sutton



ARTICLE INFO

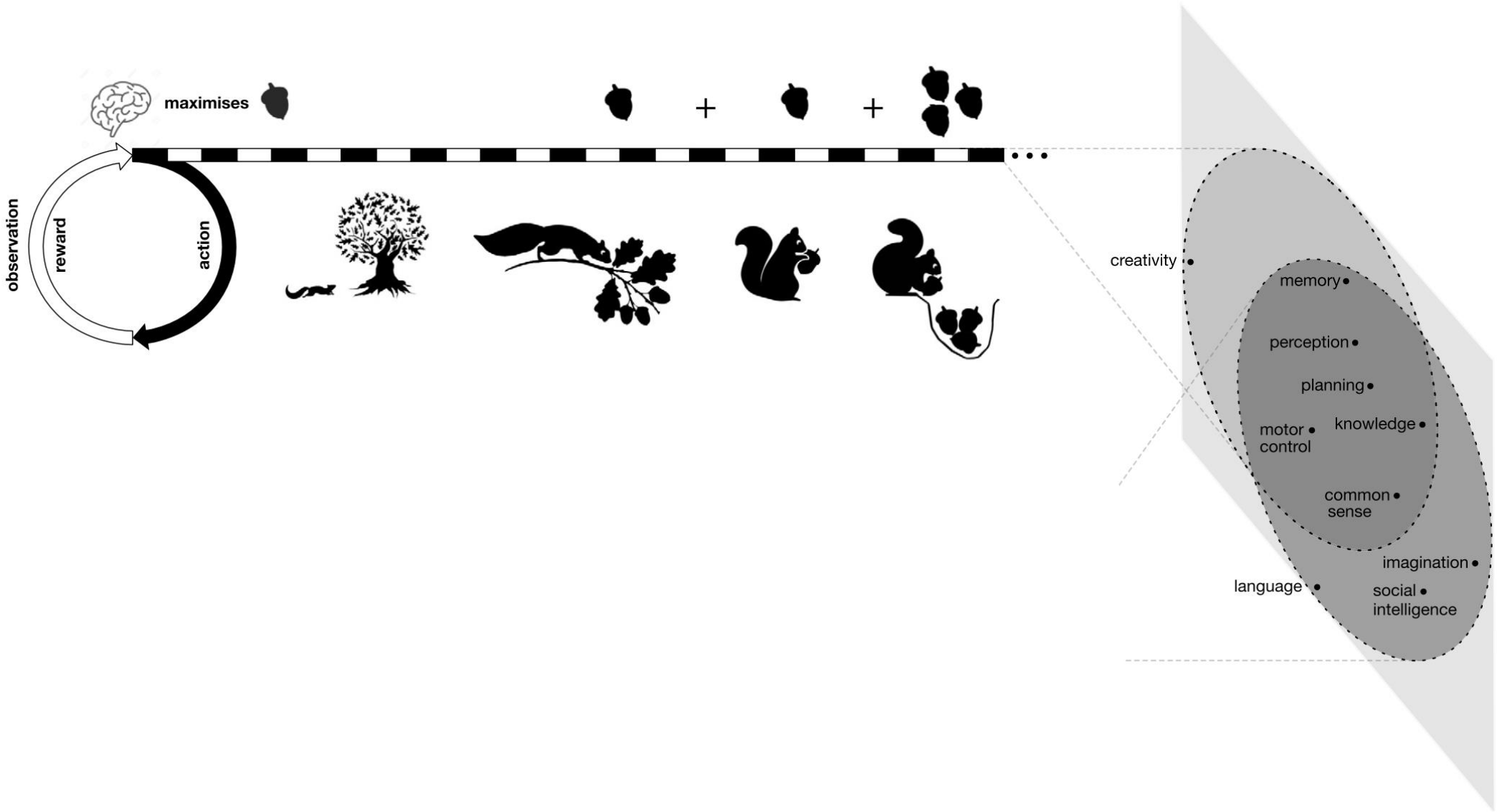
Article history:
Received 12 November 2020
Received in revised form 28 April 2021
Accepted 12 May 2021
Available online 24 May 2021

ABSTRACT

In this article we hypothesise that intelligence, and its associated abilities, can be understood as subserving the maximisation of reward. Accordingly, reward is enough to drive behaviour that exhibits abilities studied in natural and artificial intelligence, including knowledge, learning, perception, social intelligence, language, generalisation and imitation. This is in contrast to the view that specialised problem formulations are needed for each

Reward is Enough: Hypothesis

“Intelligence, and its associated abilities, can be understood as serving the maximization of reward by an agent acting in its environment”.

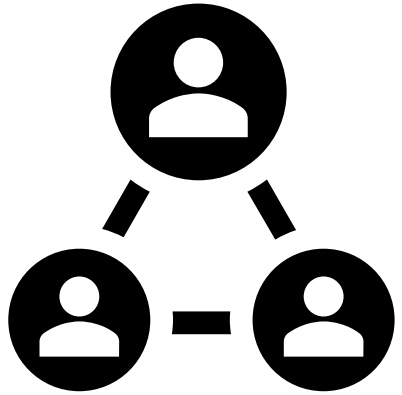

maximises
observation
reward
action
creativity
memory
perception
planning
motor control
knowledge
common sense
imagination
language
social intelligence

Reward is Enough to Understand Intelligence

- Rich environments demand a variety of abilities in the service of maximizing (potentially simple) rewards

- The expressions of intelligence found in nature may be understood as an instantiation of reward maximization
  - Such expressions arise from differences in the environment and reward, not design objectives

Example: Social Intelligence



- The ability to interact effectively with other agents

- Often formalized using game theory
  - The equilibrium solution of a multi-agent game
  - Best possible value against worst-case opponents

- With the reward-is-enough hypothesis, we instead understand it as
  - Maximizing the reward of an agent if the environment *contains* other agents
  - Social intelligence will be there *only* if the environment *and* the reward require it.
  - Better than Nash equilibria in game theory if one can capitalize on (predictable) suboptimal behavior of other agents

Reward is Enough: Implications

- Shift of focus in understanding intelligence from
  - *what* (is this particular expression of intelligence doing)
  - *To why* (is this particular expression of intelligence present)

- Broader understanding of abilities that are otherwise hard to formalize:
  - Such as “irrational agents” in economic theory

- Enough complexity in the environment is necessary for interesting abilities to arise
  - But complexity in reward signals *might not be necessary*

Reward is Enough for Artificial Agents

- Any “robot” optimizes a set of reward signals

- For example, a kitchen robot optimizes:
  - The health eating index of its user
  - Positive feedback from the user
  - Their gastronomic endorphins

- The combination of such rewards yields abilities such as:
  - Perception
  - Locomotion and Manipulation
  - …

Reward is Enough for Artificial Agents

- **What about designing such reward signals? It seems challenging…**

- The path to intelligence might be orthogonal to the specific reward signal.

- Different reward signals in different environments may produce similar abilities associated with intelligence.

What about generalization?

- The concept of “generalization” is ill-defined when the agent’s goal is maximizing a reward
- Either the agent does well in the environment (current and future) or not
- Whether or not the environment changes is something the agent might have to account for
  - For environments more complex than the agent’s capacity, it is necessary to “learn to learn”
  - *Learning is an emergent capability*, not something any agent requires

What about offline learning from a dataset?

- Offline learning from a sufficiently large dataset *might* be insufficient for intelligence.
  - Not a good scientific hypothesis because it is unfalsifiable
  - The data necessary to solve the agent's current problem will often have a negligible probability of occurring in a large offline dataset
  - **Example**: A large dataset of squirrels collecting nuts is unlikely to demonstrate all behaviors required to build a nut harvester

- The key is *online interaction*.
  - Specialize to current problems
  - Continually verify and correct the most pressing holes in knowledge

Isn’t a Reward Signal Too Impoverished?

- An effective agent may make use of additional experiential signals to maximize future rewards

- For example, *value approximation* is the process of associating features in the observation with features in the rewards
  - It provides rich secondary signals that drive learning of deeper associations via bootstrapping
  - Observation of other agents in the environment might facilitate learning (more on this later)

Which reward maximization process?

- We are not yet aware of a sample-efficient reinforcement learning process that can maximize rewards in complex, unknown environments.

- The reward could be maximized directly (model-free RL)

- Or indirectly, by decomposing the problem into subgoals
  - Representation learning
  - Value prediction
  - Model-learning and planning
  - …

Summary

- Maximization of total reward may be enough to understand intelligence and its associated abilities

- The singular goal of reward maximization may yield a deeper, broader, and more integrated understanding of abilities than a specialized formulation for each ability.

- An alternative formulation to understanding and constructing artificial intelligence

\end{Hypothesis}

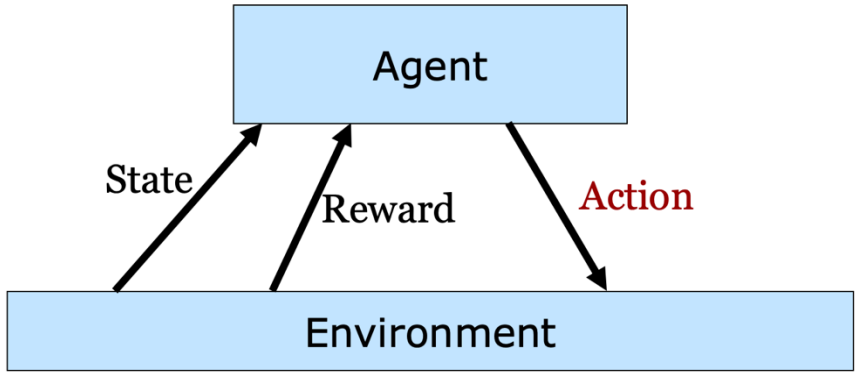
What do you think?

Today’s Agenda

- Inverse Reinforcement Learning: “Observe” and Learn
  - Behavioral Cloning
  - Reward Learning
- Reward Design

Inverse RL: Problem Definition

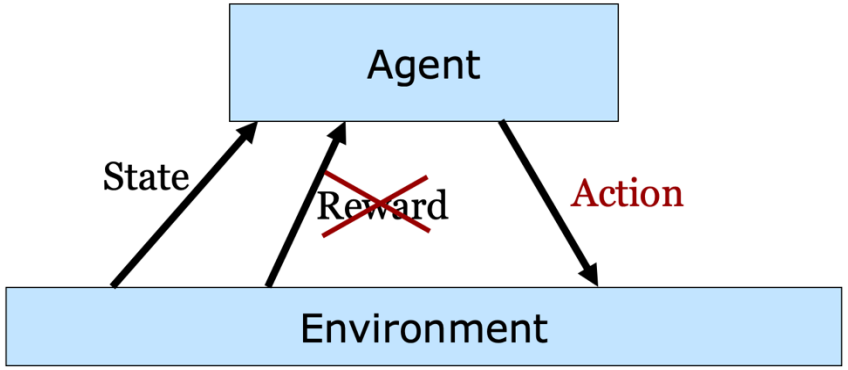
Reinforcement Learning



Goal

Learn to choose actions that maximize rewards

Inverse Reinforcement Learning



Goal

Given demonstrations of an (optimal) policy, learn to choose actions that imitate it

Slide adapted from Pascal Poupart

Inverse RL via Behavioral Cloning



From: Mobile Aloha, Fu et al.

Inverse RL via Behavioral Cloning

- Collect a dataset of demonstrations with paired observations and actions

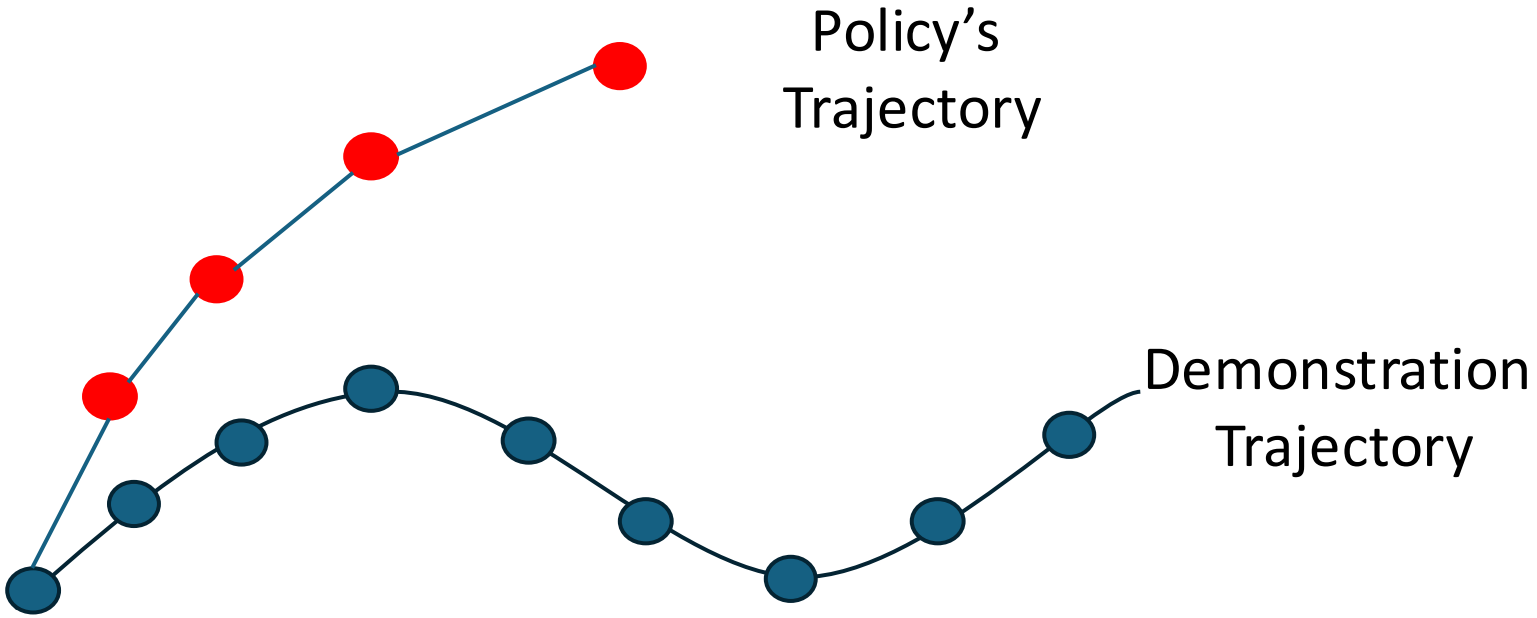
$$\mathcal{D} = \{(o_0, a_0, o_1, a_1, \dots, a_T)\}$$

- Fit a stochastic policy $\pi_\theta(a_t|o_t)$ to the dataset.

- Objective: Maximize the probability of expert actions given obs.

$$\theta^* = argmax_\theta \sum_t \pi_\theta(o_t|a_t)$$

Behavioral Cloning: The problem



Behavioral Cloning: The problem

- What we can optimize (ρ_{π^*} is the trajectory distribution of the expert):

$$\theta^* = \text{argmax}_\theta \sum_{(a_t, o_t) \sim \rho_{\pi^*}} \pi_\theta(a_t^* | o_t)$$

- What we should optimize (ρ_{π_θ} is the trajectory distribution of the policy):

$$\theta^* = \text{argmax}_\theta \sum_{(a_t, o_t) \sim \rho_{\pi_\theta}} \pi_\theta(a_t^* | o_t)$$

What is the cause of this problem?

$$\theta^* = argmax_\theta \sum_t \pi_\theta(o_t|a_t)$$

- This objective function relies on the assumption that *state-action pairs at different timestamps are i.i.d.* (independently and identically distributed)

- Not true for an MDP:
  - A mistake in some states can have a stronger effect than in others

- What we want is to imitate trajectories, not imitate actions!

Aside: Worst-Case Error

- ϵ is the average training error, i.e., $\mathrm{E}_{s_{t\sim\rho_{\pi^*}}}[|\pi(s_t) - \pi^*(s_t)|]$
- T is the episode length

- One can formally prove that

$$\sum_t \mathrm{E}_{o_{t\sim\rho_\pi}}[c(s_t, \pi(o_t))] < O(\epsilon T^2)$$

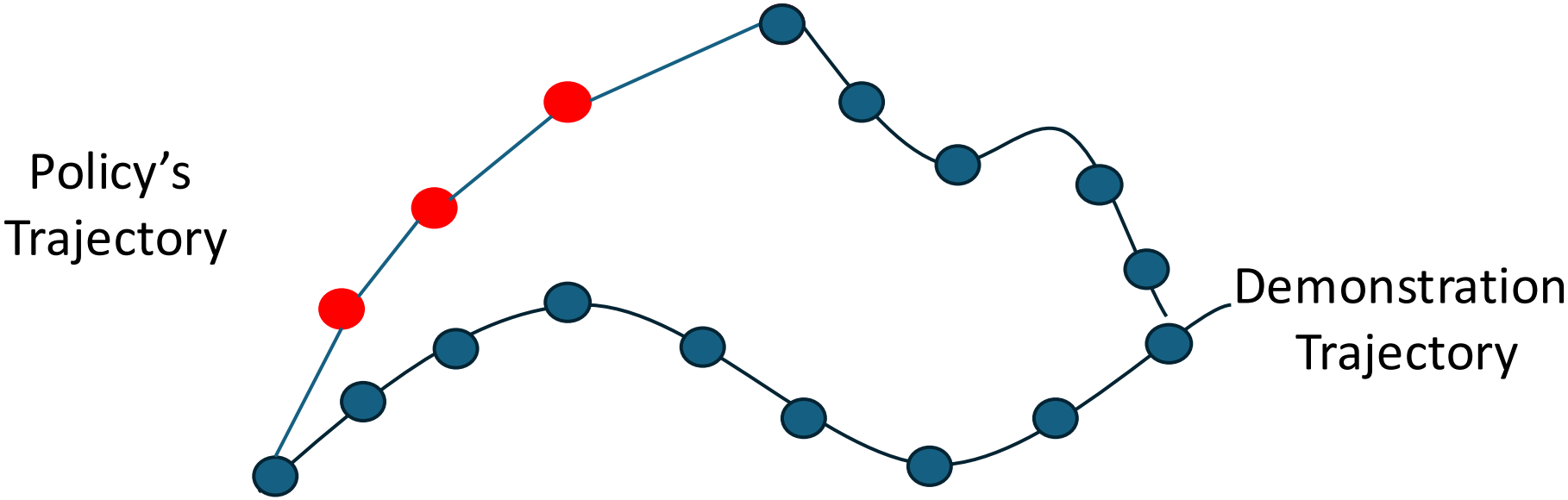
  - Where c is the error function (1 if making an error, 0 otherwise).

- *The number of mistakes increases with the square of episode length!*

Detailed analysis in: A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning

Solution by Brute Force

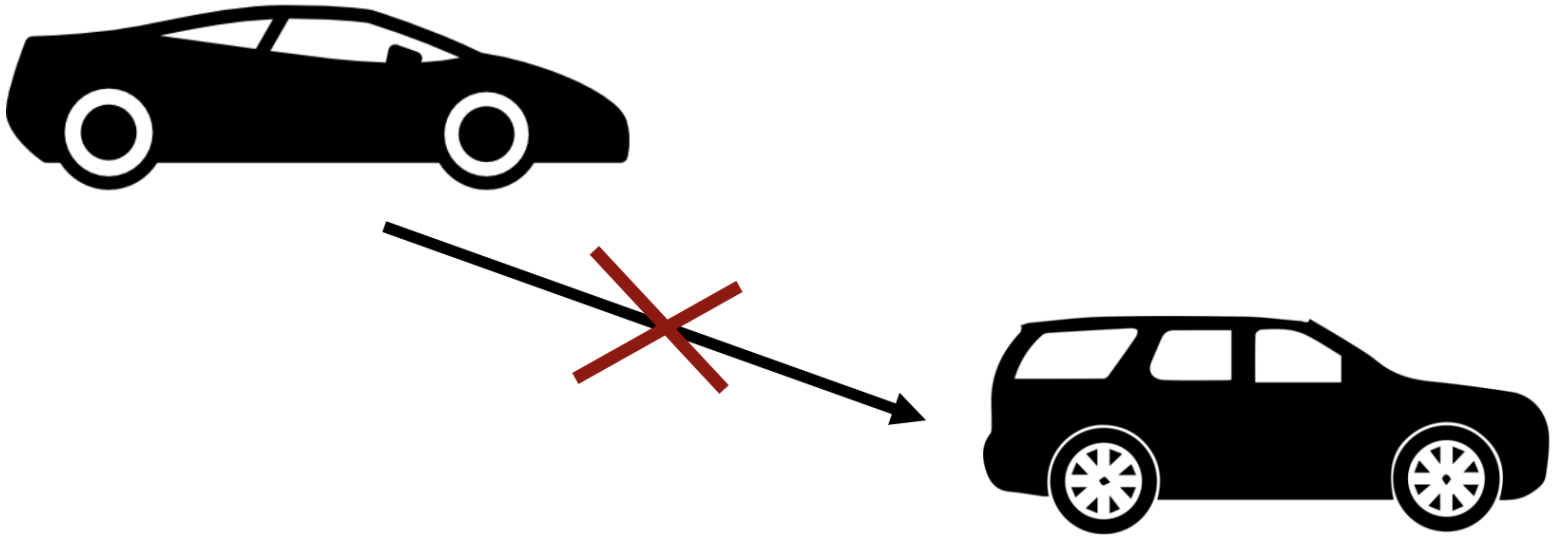
- Collect correction behaviors



- Very popular in industry!

A more saddle problem of Behavioral Cloning

- Once you learn a behavior, it is not easy to transfer to new domains



- Even if finetuning is already a very good start

Slide adapted from Pascal Poupart

Inverse RL by Learning a Reward Function

- Use the expert trajectories to *infer* the reward function
  - Which reward function best explains the data?

- Then use the reward function to learn a policy

- **Advantages**:
  - No assumption that state-action pairs are i.i.d.
  - Transfer the reward function to new domains
- **Disadvantages**:
  - Ill-defined problem
  - Expensive (needs environment samples to infer the reward and policy)
  - Compounding errors (reward inference -> policy learning)

Problem Definition

- Given:
  - State and Action Space S, A
  - Dataset of trajectories by a (near-optimal) policy π^*: $(s_0, a_0, s_1, a_1, \ldots, a_T)$
  - Discount factor γ, Horizon T, (maybe model $\mathcal{M}$)

- **Goal.** Find a reward function such that the trajectories observed in the dataset are optimal.

- Formally, find a reward function $r(s, a)$ such that:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\tau \sim \pi} \left[\sum_t \gamma^t r(s_t, a_t) \right]$$

Reward Inference is Ill-defined (Ng&Russel, ‘00)

- **Theorem.** For any dataset of trajectories realizable by a stationary Markovian policy, there *always* exists a reward function according to which the dataset trajectories are optimal.

- Unfortunately, $r(s, a) = 0$ (or any constant c) is a solution to the inverse RL problem.
  - All policies are optimal!

- We need further constraints to guarantee identifiability

Regularization Constraints

- Maximal distance between optimal and non-optimal actions
- Entropy regularization
- Adding structural priors to the reward
  - Sparsity
  - Specific reward class

Maximum Distance Inverse RL

- Assume that the reward function is a linear function of state features:

$$r(s,a) = \sum_i w_i \phi_i(s,a) = \boldsymbol{w^T \phi}(s,a)$$

- Value Function:
- $V^\pi(s) = \sum_t \gamma^t \mathbb{E}_{s_t \sim d_\pi(s_t), a_t \sim \pi(s_t)}[r(s_t,a_t)] =$
- $\sum_t \gamma^t \mathbb{E}_{s_t \sim d_\pi(s_t), a_t \sim \pi(s)}[\boldsymbol{w^T \phi}(s_t,a_t)] =$ (reward def.)
- $\boldsymbol{w^T} \sum_t \gamma^t \mathbb{E}_{s_t \sim d_\pi(s_t), a_t \sim \pi(s)}[\boldsymbol{\phi}(s_t,a_t)]$ (linearity of exp.)
- $\boldsymbol{w^T} \sum_t \gamma^t [\boldsymbol{\phi}_\pi(s_t,a_t)]$ (exp. features at t)
- $\boldsymbol{w^T} \overline{\boldsymbol{\phi}_\pi(s)}$ (exp. Features over τ)

Maximum Distance Inverse RL

- Idea: Due to Bellman optimality, all policies that are optimal share the same value function. Therefore:

$$V^{\pi}(s) = V^{e}(s)$$

  - Where $V^{e}(s)$ is the value function of the expert collecting trajectories

- Assuming that the reward function is linear:

$$\boldsymbol{w}^{\boldsymbol{T}}\overline{\boldsymbol{\phi}_{\pi}(s)} = \boldsymbol{w}^{\boldsymbol{T}}\overline{\boldsymbol{\phi}_{e}(s)}$$

- **Problem**: There are infinitely many $\boldsymbol{w}$ satisfying this equation!

Maximum Distance Inverse RL

- Address the uniqueness problem by finding weights that maximize the margin (value gap) between the expert and non-expert actions.

$$\max_{w,\|w\|<1} \mathbb{E}_{s_t \sim \mathrm{d}_\pi(s_t)} \left[Q^\pi\left(s_t, \pi(s_t)\right) - \max_{a \neq \pi(s)} Q^\pi(s,a) \right]$$

  - Where π is the optimal policy according to the reward function $r(s,a) = w^T \phi(s,a)$

Maximum Distance Inverse RL

- This leads to the following optimization problem

$$w^* = \arg\min w^T(\bar{\phi}^e - \bar{\phi}^\pi)$$
$$s.t. \max_{w,\|w\|<1} \mathbb{E}_{s_t \sim \mathrm{d}_\pi(s_t)} \left[Q^\pi\big(s_t, \pi(s_t)\big) - \max_{a \neq \pi(s)} Q^\pi(s,a) \right]$$

- Why?
  - Rules out $r = 0$ (except if it is the only solution)
  - Guarantees π is unique (for deterministic policies)
  - Bounding the norm $\|w\| < 1$ removes the scaling problem
  - Increasing the margin gives some robustness against estimation errors (safety buffer)
- Hard to implement in practice

Aside: Apprenticeship Learning (Abbeel&Ng. '04)

- The above problem can be reduced to the following optimization problem:

$$w^* = \arg \max_{w, \|w\|<1} \min_{\pi} w^T (\bar{\phi}^e - \bar{\phi}^\pi)$$

- Can be solved effectively via an iterative process.
- But a few important issues remain:
  - Each iteration requires solving for an optimal policy
  - Does not work if the expert policy collecting the demonstrations is stochastic (otherwise, the optimal action is not strictly better than any other)

Regularization Constraints

- Maximal distance between optimal and non-optimal actions

- Entropy regularization

- Adding structural priors to the reward
  - Sparsity
  - Specific reward class

Entropy-Regularized Inverse RL

- The agent maximizes both expected reward and entropy

$$\pi^* = \arg\max_{\pi} \left[\sum_t \gamma^t (r(s,a) + \alpha \mathcal{H}(\pi(\cdot\,|s_t)) \right]$$

- As we've already seen in learning with Q functions, π^* can be written as

$$\pi^* \propto \exp\left(\frac{Q^*(s,a)}{\alpha}\right)$$

Entropy-Regularized Inverse RL

- It is possible to show (but we won't do it in class) that:
  - Given expert demonstrations $D = \{\tau_i\}$
  - And a reward function $r(s, a; w)$
  - If we want to maximize the average (log) probability of expert trajectories

$$\mathcal{L}(w) = \mathbb{E}_{\tau \sim D}[\log p_w(\tau)]$$

  - And the expert is a maximum entropy policy
  - And the dynamics is deterministic

$$\nabla_w \mathcal{L}(w) = \mathbb{E}_{\tau \sim \mathrm{D}}\left[\sum_t \nabla_w r(s_t, a_t; w)\right] - \mathbb{E}_{\tau \sim \pi^*}\left[\sum_t \nabla_w r(s_t, a_t; w)\right]$$

Entropy-Regularized Inverse RL

$$\nabla_w \mathcal{L}(w) = \mathbb{E}_{\tau \sim \mathrm{D}} \left[\sum_t \nabla_w (r(s_t, a_t; w) \right] - \mathbb{E}_{\tau \sim \pi^*} \left[\sum_t \nabla_w (r(s_t, a_t; w) \right]$$

- First term: Maximize the rewards in the observed dataset
- Second term: Minimize the rewards of the optimal policy given the current rewards.

- The two terms “fight” against one another.
- The equilibrium point is the optimal entropy-regularized reward.
- Nice connection to adversarial learning (see GAIL, Ho and Ermon ‘16)

Entropy-Regularized Inverse RL

- Nice and elegant, with many extensions to address problems of computational and sample complexity
- But still remains a very challenging optimization problem to solve
- Up to now, no algorithm with a general reward function has been shown to be both effective and efficient for complex problems.
  - It could be there; we just have not found it yet

Regularization Constraints

- Maximal distance between optimal and non-optimal policies

- Entropy regularization

- Adding structural priors to the reward
  - Sparsity
  - Specific reward class

Specific Reward Class

- Instead of finding any possible reward class, restrict rewards to a specific type

- A common example is “time-based” rewards

- **General idea**: π should take the same time as the expert to arrive at the final goal, for every state s_t.

General Approach

- Given the dataset of expert trajectories $\mathcal{D} = \{(s_0, a_0, \dots, s_T, a_T)\}$
- Train a feature vector $\phi(s)$ such that:

$$\|\phi(s_t, a_k) - \phi(s_T, a_T)\| > \|\phi(s_{t+k}, a_k) - \phi(s_T, a_T)\|$$

  - The further you're from the goal, the longer it should take.

- After training is done, optimize a policy given the (fixed) Q function:

$$Q(s, a) = \|\phi(s, a) - \phi(s_T, a_T)\|$$

- Possible to prove that this makes sense

Example from robotics

TOWARDS UNIVERSAL VISUAL REWARD AND REPRESENTATION VIA VALUE-IMPLICIT PRE-TRAINING

Yecheng Jason Ma[*2], **Shagun Sodhani**[1], **Dinesh Jayaraman**[2], **Osbert Bastani**[2],
{**Vikash Kumar**[†1], **Amy Zhang**[†1]}

FAIR, Meta AI[1], University of Pennsylvania[2]

https://sites.google.com/view/vip-rl

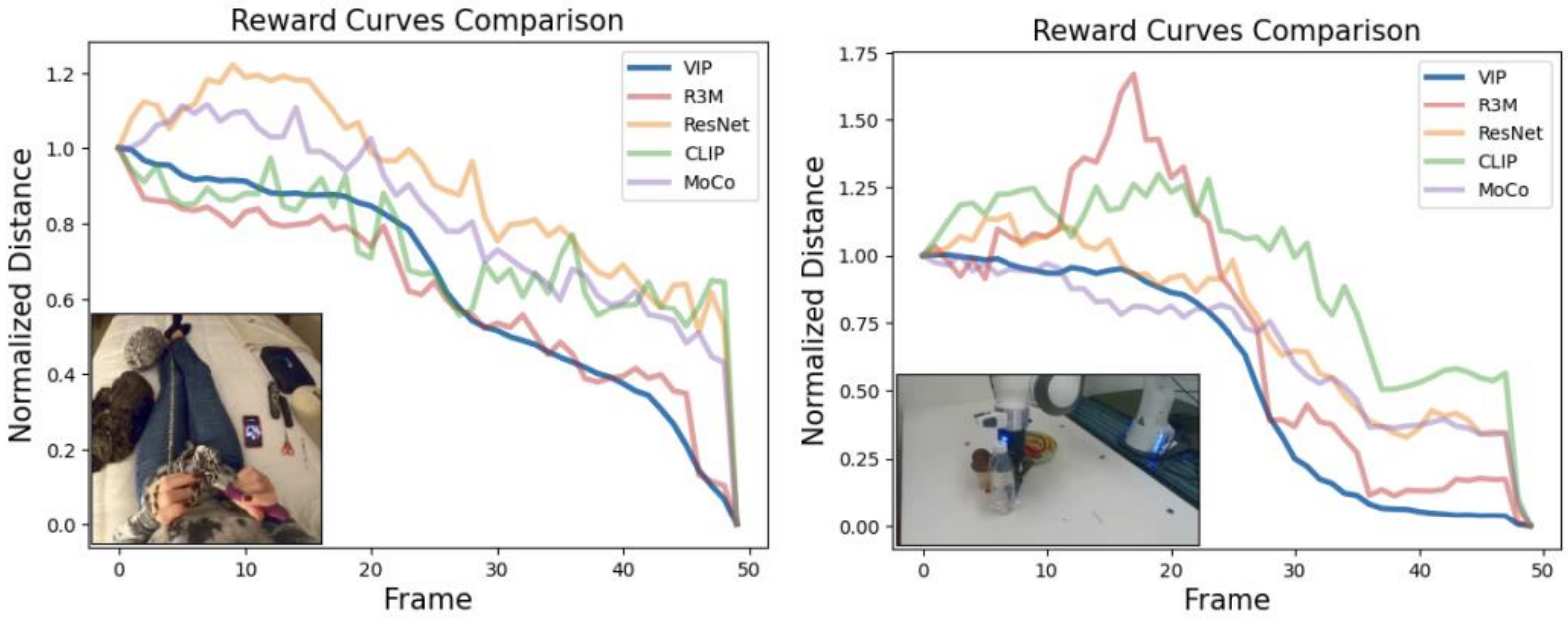


Figure 6: Embedding distance curves on Ego4D (L) and real-robot (R) videos.

Specific Reward Class

- A hot research topic

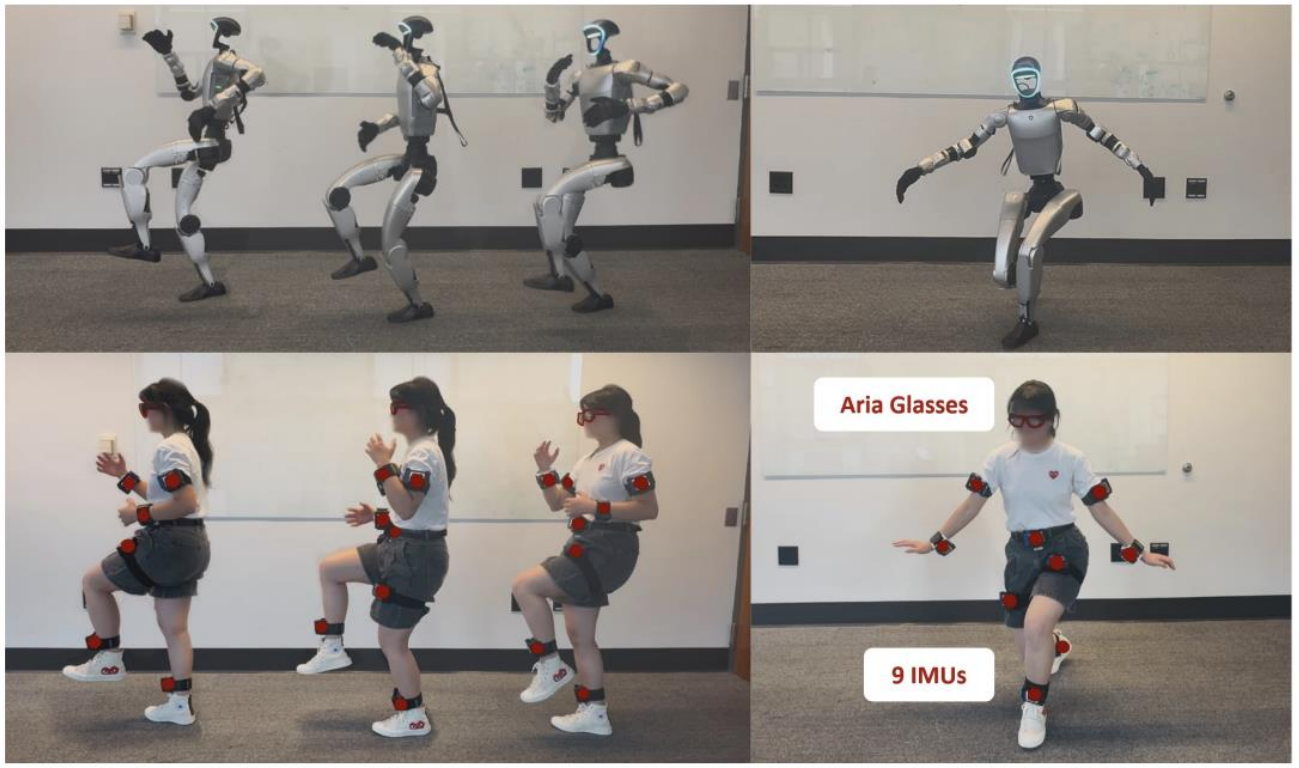
- Other examples include:

  - Matching keypoints in time (human and agent must be at the same position at the same time)
  - Reward sparsity (only receive a reward at an important junction point, such as when a task is completed)

A (very very biased) example

RoSHI: A Versatile Robot-oriented Suit for Human Data In-the-Wild

Wenjing Margaret Mao*, Jefferson Ng*, Luyang Hu*, Daniel Gehrig, Antonio Loquercio



Inverse RL: Summary

- A general family of methods to infer a reward function given demonstrations of an expert
- A very challenging optimization problem, tackled using different kinds of assumptions/heuristics.
- But what if I don’t have demonstrations, just an idea of how the task should be?

Reward Design: Tips and Tricks

- Keep your reward as simple as possible
- The fewer the terms, the easier it is to tune

- Start with very sparse rewards.
- If exploration fails, gradually add denser reward terms.

- Add a cost for action magnitude (you generally want the maximum benefit with minimal effort).

- Keep your rewards mostly positive (or the terminal state becomes attractive)

Reward Design: Tips and Tricks

- In my experience, I've seen the effects of a good reward and reset function far outstrip the benefits of switching between algorithms.
  - Sad story that contradicts the reward-is-enough hypothesis
  - IMO, it tells us more about the immaturity of current algorithms than anything else
- LLMs help, but they tend to generate verbose reward functions that are hard to tune.
- Takes time to learn. Be patient ☺