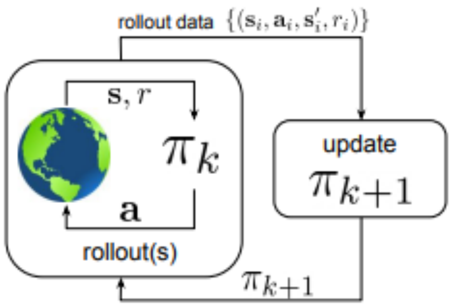
Inverse Reinforcement Learning. Learning Decisions from Preferences

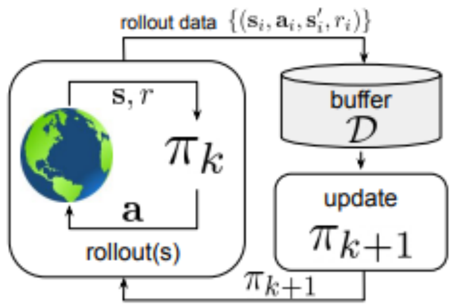
With thanks to Pieter Abbeel, Wen Sun, J. Colaco-Carr, P. Panangaden, R. Munos, B. Piot, M. Valko, D. Calandrielo

Recall: Batch/offline RL

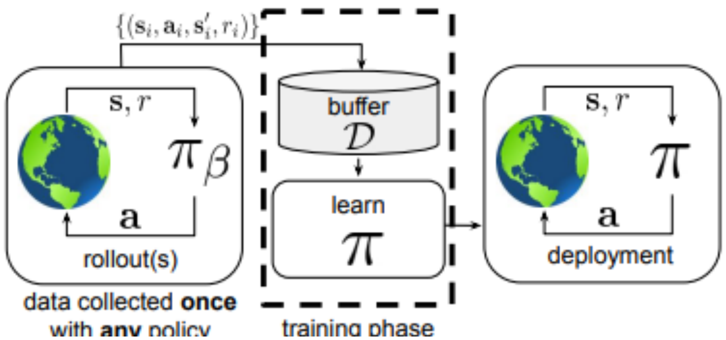
on-policy RL



off-policy RL



offline reinforcement learning



Formally:

$$\mathcal{D} = \{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i, r_i)\}$$

$\mathbf{s} \sim d^{\pi_\beta}(\mathbf{s})$

$\mathbf{a} \sim \pi_\beta(\mathbf{a}|\mathbf{s})$ ← generally **not** known

$\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$

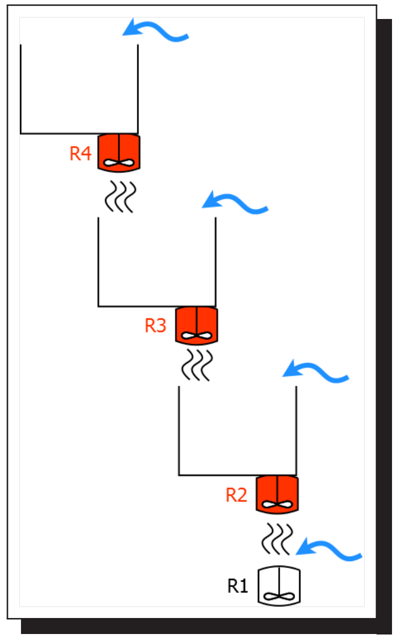
$r \leftarrow r(\mathbf{s}, \mathbf{a})$

RL objective: $\max_\pi \sum_{t=0}^{T} E_{\mathbf{s}_t \sim d^\pi(\mathbf{s}), \mathbf{a}_t \sim \pi(\mathbf{a}|\mathbf{s})}[\gamma^t r(\mathbf{s}_t, \mathbf{a}_t)]$

Recall: Batch RL classes of algorithms

1. Behavior cloning (no rewards required)
2. Learn a model, use it for model-based RL (LSTD, LSPI)
3. Pessimistic algorithms (require rewards)
4. Inverse RL (learn reward function from data, use it for RL agent) - today!

Motivating example: Power Plant Control



- 3 turbines to control (continuous variables), one per reservoir
- turbine R1 is controlled by the water flow
- (stochastic) ground water inflows
- weekly time steps
- objective: maximize average annual power production while satisfying constraints (see below)

Cf. Grinberg et al, 2014; collaboration with Hydro Quebec

- Major: sufficient flow needs to be maintained to allow easy passage for fish
- Major: stable turbine speed throughout weeks 43-45 to allow fish spawning
- Minor: amount of water in second reservoir should be above a minimum

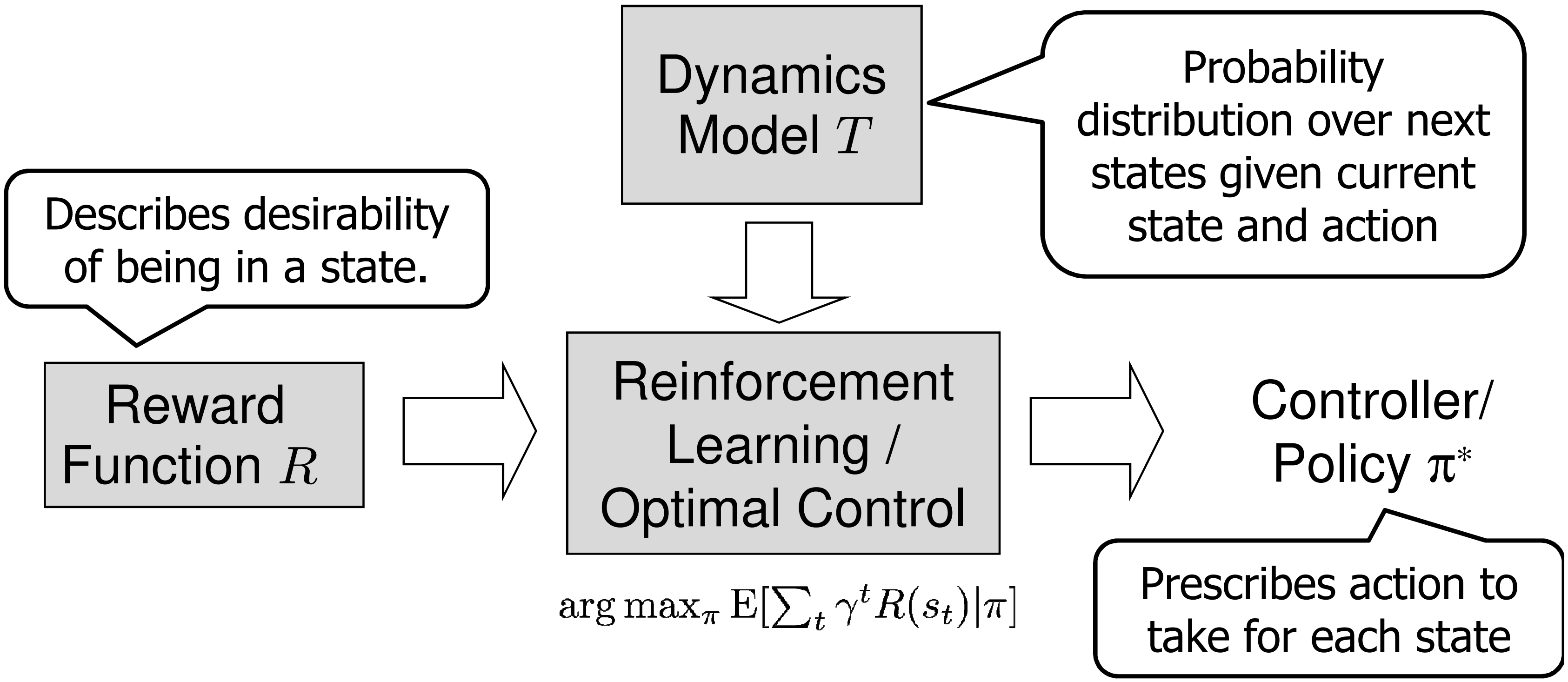
Reward function can be quite hard to formulate!

How to Solve Power Plant Control?

- Spent a lot of time trying to craft a reward function that captures the objective
- *Reward hacking is a major issue*
- Tried various constrained and risk-sensitive optimization (hyper-parameter tuning is no better than fitting rewards)
- Ended up doing *randomized policy search*!

Crafting reward functions is hard in practice!

Inverse RL



Inverse RL:

Given π^* and T, can we recover R?
More generally, given execution traces, can we recover R?

IRL problem formulation

- Input:
  - State space, action space
  - Transition model $P_{sa}(s_{t+1} \mid s_t, a_t)$
  - *No* reward function
  - Teacher's demonstration: $s_0, a_0, s_1, a_1, s_2, a_2, \ldots$
    (= trace of the teacher's policy π^*)
- Inverse RL:
  - Can we recover R ?
- Apprenticeship learning via inverse RL
  - Can we then use this R to find a good policy ?
- Behavioral cloning
  - Can we directly learn the teacher's policy using supervised learning?

IRL vs. Imitation learning: which one is better?

- It depends if the policy or the reward is more complicated!
- If the policy is simple learning it is easy supervised learning
- If the reward is simpler, IRL is better
- IRL also allows you to optimize if eg the transition function changes (eg autonomous driving, complex map navigation)

Main idea

- Assume the trajectories we have come from an optimal expert
- Therefore, the expert must have a reward function R^* such that:

$$E\left[\sum_{t=0}^{\infty} \gamma^t R^*(s_t)|\pi^*\right] \geq E\left[\sum_{t=0}^{\infty} \gamma^t R^*(s_t)|\pi\right], \forall \pi$$

  Now solve this type of equation for R^*!
- Problem: reward function ambiguity!
  - Many reward functions satisfy this equation, including eg $R^*(s) = 0 \forall s$
  - We only have some traces, not all of π^*
  - The expert has to be optimal

Feature-based reward functions

- Let $R(s) = w^\top \phi(s)$, where $w \in \Re^n$, and $\phi : S \to \Re^n$.

$$
\begin{aligned}
\mathrm{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi] &= \mathrm{E}[\sum_{t=0}^{\infty} \gamma^t w^\top \phi(s_t)|\pi] \\
&= w^\top \mathrm{E}[\sum_{t=0}^{\infty} \gamma^t \phi(s_t)|\pi] \\
&= w^\top \underbrace{\mu(\pi)}
\end{aligned}
$$

Expected cumulative discounted sum of feature values or “feature expectations”

- Subbing into $\mathrm{E}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t)|\pi^*] \geq \mathrm{E}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t)|\pi] \quad \forall \pi$

  gives us:

  $$\text{Find } w^* \text{ such that } w^{*\top} \mu(\pi^*) \geq w^{*\top} \mu(\pi) \quad \forall \pi$$

Notice that μ is the successor feature, which can be learned from data

Feature matching

- Abbeel and Ng (2004): for a policy π to perform almost as well as the optimal policy π^*, it suffices that the successor feature expectations match:
  $$||\mu(\pi) - \mu(\pi^*)||_1 \leq \epsilon$$
  where $0 \leq \epsilon < 1$
- If so, $\forall w$ with norm less than 1:
  $$|w^T \mu(\pi) - w^T \mu(\pi^*)| \leq \epsilon$$
- Optimization problem can be solved in complexity that depends on $1/\epsilon^2$ and on the number of features in the reward function, NOT depending on complexity of π^* or the number of states
- Approximation property even if the true reward cannot be represented through linear combination of features

Apprenticeship learning (Abbeel and Ng, 2004)

- Assume $R_w(s) = w^\top \phi(s)$ for a feature map $\phi : S \rightarrow \Re^n$.
- Initialize: pick some controller π_0.
- Iterate for i = 1, 2, … :
  - **"Guess" the reward function**:

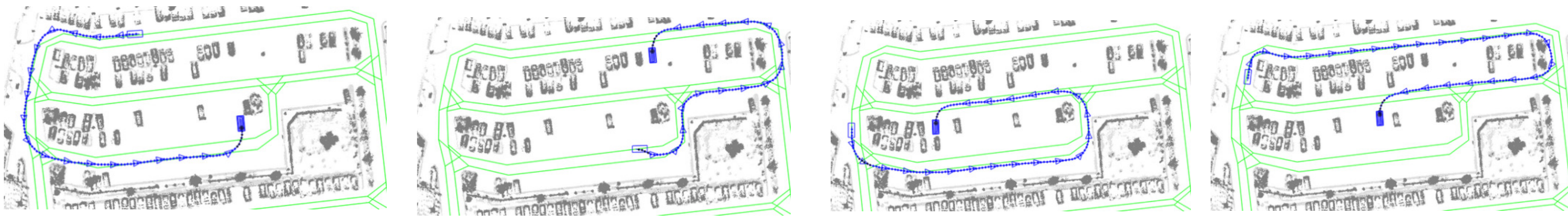
    Find a reward function such that the teacher maximally outperforms all previously found controllers.

    $$\max_{\gamma, w: \|w\|_2 \leq 1} \gamma$$
    $$s.t. \quad w^\top \mu(\pi^*) \geq w^\top \mu(\pi) + \gamma \quad \forall \pi \in \{\pi_0, \pi_1, \ldots, \pi_{i-1}\}$$

  - **Find optimal control policy** π_i for the current guess of the reward function R_w.
  - If $\gamma \leq \varepsilon/2$ exit the algorithm.

Example: Learning to park (Abbeel and Ng, 2004)

- Demonstrate parking lot navigation on “train parking lots.”



- Run our apprenticeship learning algorithm to find the reward function.
- Receive “test parking lot” map + starting point and destination.
- Find the trajectory that maximizes the *learned reward function* for navigating the test parking lot.

Learned policies reflect the data!

- Training on nice data (left) vs sloppy data (middle) vs reverse as well (right)

Helping with disambiguation: Maximum-entropy IRL

- IRL is essentially trying to match the feature distribution in expert demonstrations
- But we need to add further constraints to make this well conditioned (previously discussed work uses a margin)
- Principle of maximum entropy: Given prior information about a distribution, the best approximation is the distribution matching the data with the largest uncertainty
- Because this makes the fewest assumptions about the true distribution!
- Another interpretation: we want to avoid overfitting the data

Helping with disambiguation: Maximum-entropy IRL (Ziebart et al, 2008)

- Main idea: match feature distributions but otherwise maximize the entropy of trajectory distributions
- Recall the definition of entropy: $H(P) = -\sum_x P(x) \log P(x)$
- Recall the probability of a trajectory $\tau = s_0 a_0 \ldots s_T$:

$$P^\pi(\tau) = p_0(s_0) \prod_{t=0}^{T-1} \pi(a_t|s_t) P(s_{t+1}|s_t, a_t)$$

- We want to maximize the entropy of P^π while matching feature expectations:

$$\max_\pi H(P^\pi) \text{ s.t. } \mu(\pi) = \mu(\pi^*)$$

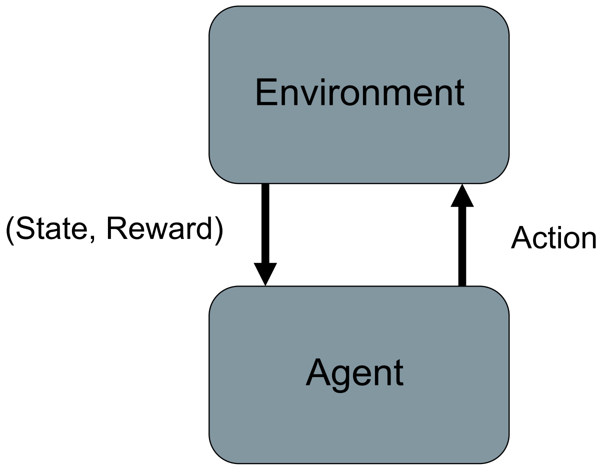
- Can be re-written as: $\min_\pi E_{s,a \sim \mu^\pi} [\log(\pi(a|s))]$
- Can be solved using a Lagrangian and gradients (various flavors exist)

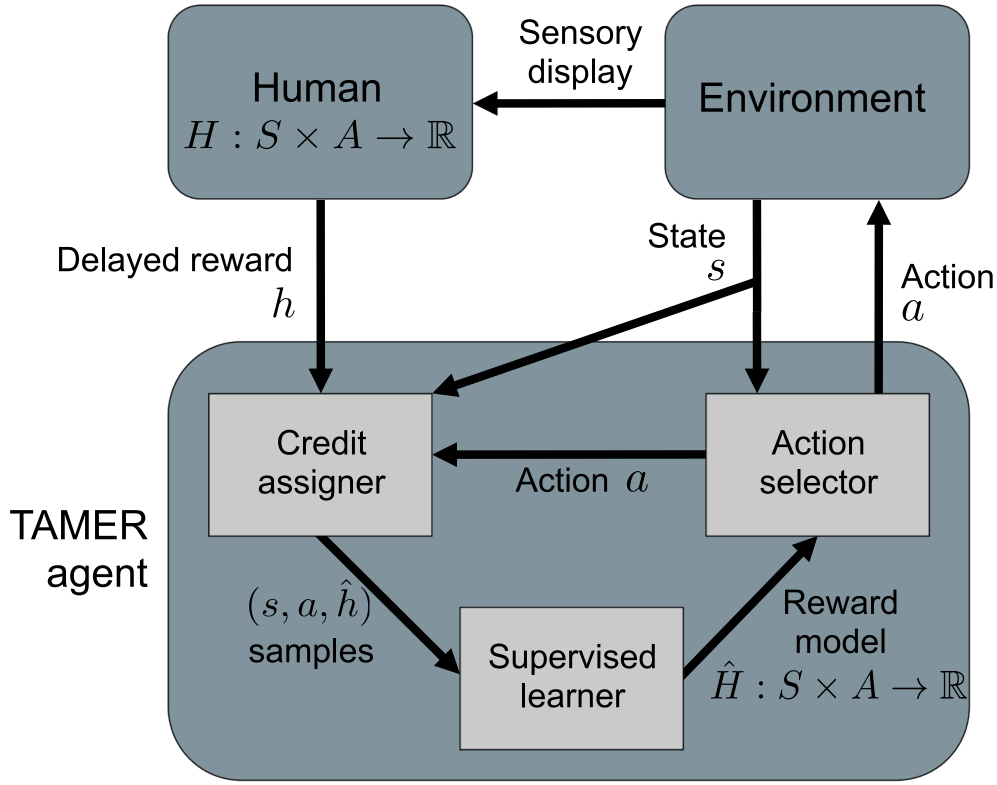
Learnign from data vs learnign from feedback

- Batch RL: have trajectories (with or without rewards), learn to copy them (by generating a policy or a reward function that is then optimized)
- One flavor that we did not discuss: data is only states, actions have to be inferred

  Eg learning from video demonstrations
- But what if we don't have trajectories? How can we still get a reward function?
- Crux of modern LLM stuff!

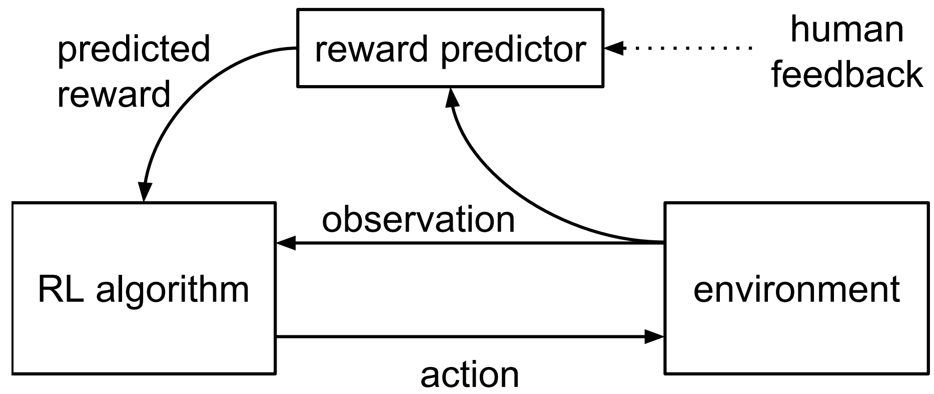
Learning from Human Feedback (Knox, 2012)





- Numerical reward is a high-variance signal even when learned

Deep RL from Human Feedback (Christiano et al, 2017)



- People provide a *preference* among two choices
- Assuming there is a latent variable explaining the choice, reward is fit using maximum likelihood (Bradley-Terry model)
- Cf. https://arxiv.org/pdf/1706.03741.pdf

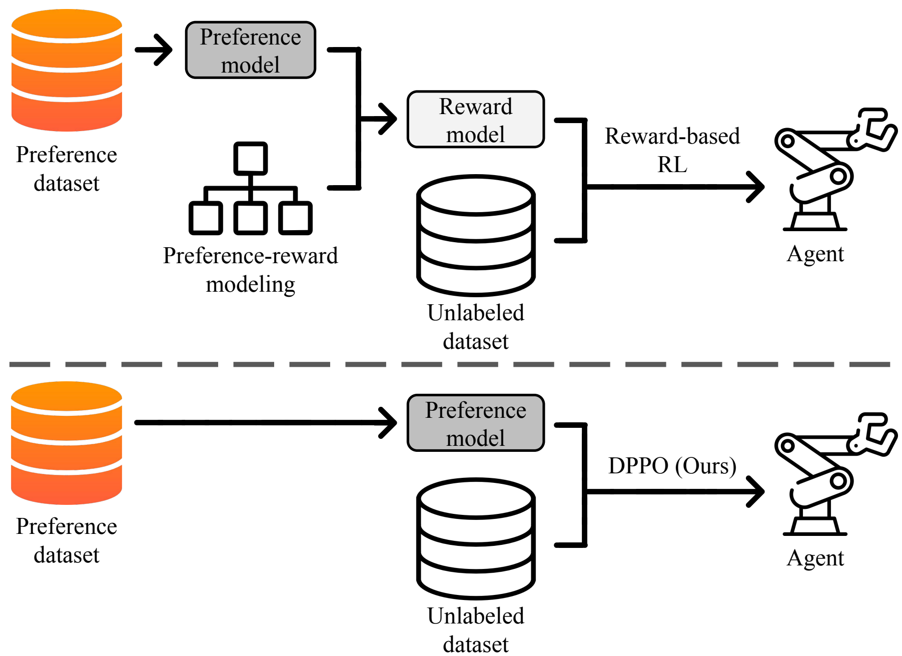
Bradely-Terry reward model

- Collect data from human raters (pairs of y_w, y_l responses to a prompt x)
- Optimize the expected value of:

$$-\log(\sigma(r_\theta(x, y_w) - r_\theta(x, y_l)))$$

  wrt reward parameter vector θ
- Cf. Ouyang et al, InstructGPT
- Corresponds to maximum likelihood fitting of binomial preference function if reward is linear over the variables

Direct Preference Optimization



- Cf. An et al, NeurIPS'2023 (https://arxiv.org/pdf/2301.12842.pdf)
- Direct preference optimization (Rafailov et al, NeurIPS'2023, https://arxiv.org/pdf/2305.18290.pdf)
- Several other almost-concurrent papers in this space

Optimizing Preferences: Setup

- An agent interacting with an environment receives observations for a set $\mathcal{O}$ and performs action from set $\mathcal{A}$
- A *history* h_t is a sequence of observation-action pairs $\langle o_0, a_0, o_1, a_1, \ldots o_t \rangle$
- A *policy* π is a mapping from histories to actions: $\pi : \mathcal{H} \rightarrow \mathcal{A}$
- Consider a *binary relation over trajectory distributions* $\preceq$
- A policy π in an environment e induces a probability distribution over trajectories, D^π
- See Colaco-Carr et al, AISTATS'2024 (https://arxiv.org/abs/2311.01990)

Preference Relations and Their Properties

- We will formalize preference relations through pre-orders
- For trajectory distributions A and B, $A \preceq B$ means is that B is at least as preferred as A
- $\preceq$ is a *pre-order* if it satisfies:
  - *Reflexivity*: $A \preceq A$
  - *Transitivity*:if $A \preceq B$ and $B \preceq C$ the $A \preceq C$
- A pre-order is *total* if for and A, B, $A \preceq B$ and $B \preceq A$

Direct Preference Process

- A *Direct Preference Process* is a tuple $\mathcal{O}, \mathcal{A}, T, e, \preceq$ where:
  - $\mathcal{O}$ is an observation set
  - $\mathcal{A}$ is an action set
  - T is a time horizon
  - e is an environment (transition function from achievable history-action pairs to the next observation)
  - $\preceq$ is a binary (preference) relation over trajectory distributions
- $\preceq$ is *expressible through a reward function* $r : \mathcal{H} \rightarrow \mathbb{R}$ if:

$$\forall A, B, A \preceq B \text{ if and only if } \mathbb{E}_A \left[\sum_{t=0}^{T} r(H_t) \right] \leq \mathbb{E}_B \left[\sum_{t=0}^{T} r(H_t) \right]$$

Preference Relations and Their Properties

- A total pre-order is *consistent* if

$$\forall \alpha \in (0,1), \forall A, B, C, A \preceq B \implies \alpha A + (1-\alpha)C \preceq \alpha B + (1-\alpha)C$$

- A total pre-order is *convex* if

$\forall \alpha \in (0,1), \forall A, B, C, A \preceq B.$ if and only if $\alpha A + (1-\alpha)C \preceq \alpha B + (1-\alpha)C$

- A total pre-order has the *interpolation property* if

$\forall A, B, C, A \preceq B$ and $B \preceq C$ implies $\exists \alpha \in (0,1), \alpha A + (1-\alpha)C \sim B$

- Von Neumann-Morgenstern theorem: if all the above hold, $\preceq$ can be expressed by a utility function

When Are Preferences Representable By Reward Functions?

- Main result
  - *If convexity and/or interpolation do not hold, $\preceq$ is NOT is expressible through a reward function*
  - *However, total consistent pre-orders have deterministic optimal policy!*
- The latter situation is not exotic or rare!

Examples when Optimal Policies Exist Without Rewards

- *Total consistent convex pre-order not satisfying interpolation: tie-breaking criteria*
  - Use a first criterion, if tied go to a second criterion
  - See not flooding vs water in second reservoir in power plant example
- *Total consistent pre-order that is non-convex: excess risk*
  - If risky event does not occur, linear utility
  - Risky event occurring entails exponential penalty
  - No flooding neighbouring areas in power plant example

How Do We Compute Optimal Policies?

- If $\preceq$ is a total consistent pre-order and a policy π satisfies the following for any attainable history $h_t, t < T$ and any action a_t:

$$D^\pi(h_t \cdot a_t) \preceq D^\pi(h_t)$$

  then π is $\preceq$-optimal
- So we are *justified to do policy search*!
- If $\preceq$ is expressible through a reward function, value iteration is a direct consequence of this result

Discussion

- Nice to know that aproaches such as direct preference optimization are justified
- Our results are currently on distributions - working on sample-based extensions
- If we can fit a reward function, should we?
  - Bias-variance trade-off? Sample complexity considerations?
- *What can we do if other properties of pre-orders are violated?*

Learning with non-transitive preferences: NashLLM

- Objective:find a policy π^* which is preferred over any other policy

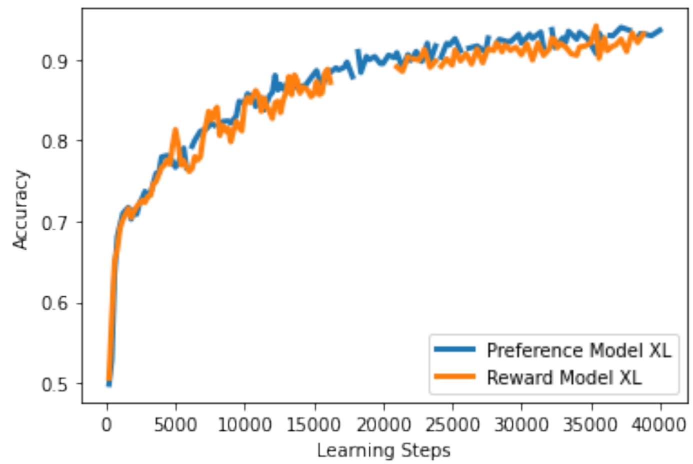
$$\pi^* = \arg\max_{\pi} \min_{\pi'} \mathbb{P}(\pi' \preceq \pi)$$

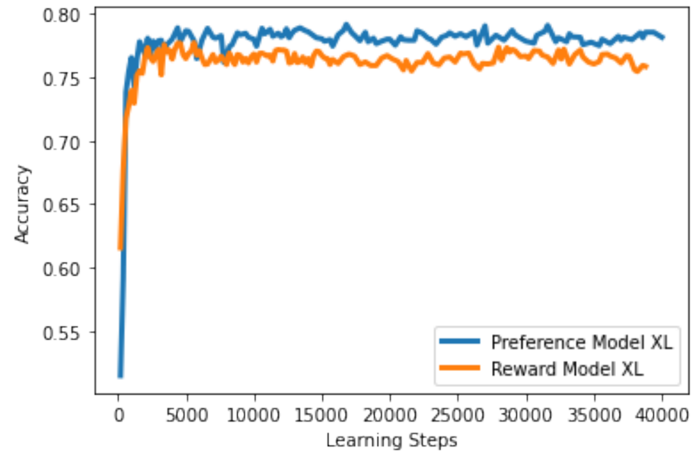
- Think of this as a game: one player picks π the other picks π'
- When both players use π^* this is a *Nash equilibrium* for the game
- For this game an equilibrium exists (even if eg preferences are not transitive)
- Cf. Munos et al, 2024 (https://arxiv.org/pdf/2312.00886.pdf)

NashLLM-style algorithms

- Fit a *two-argument preference function* by supervised learning
- Decide what is the *set of opponent policies*
- Ideally, the max player should play against a mixture of past policies
- *Optimize* using eg online mirror descent, convex-concave optimization...
- A lot of algorithmic variations to explore!

NashLLM results





Using preferences instead of rewards leads to less overfitting