Reward and Constraint Learning: Foundations for Human-AI Alignment

Scaling to Complicated Reward Spaces, Deep Reward Modeling, Active Reward Learning

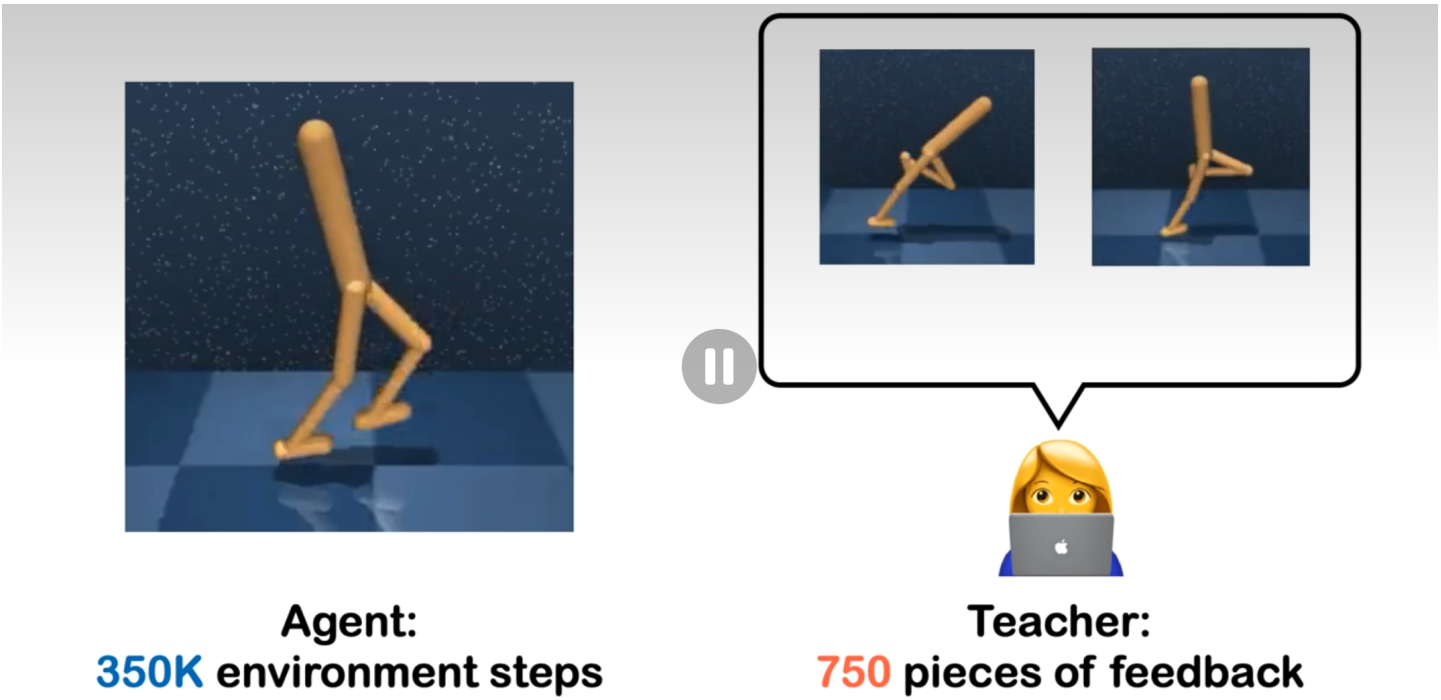
Sebastian Tschiatschek

Session 2, July 2026

European Summer School on AI (ESSAI) 2026

Teaser

Human feedback can enable efficient learning



» show pebble.mp4

Lee, K., Smith, L., and Abbeel, P. (2021). *PEBBLE: Feedback-Efficient Interactive Reinforcement Learning via Relabeling Experience and Unsupervised Pre-training*, *ICML*.

Course overview

Session 1 (Mon)

- RL Basics
- IRL Basics
- Maximum Entropy IRL

Session 2 (Tue)

- Deep Reward Modeling
- Learning from Human Feedback

Session 3 (Wed)

- Constraints
- Importance and Learning

Session 4 (Fri)

- Human-AI Alignment
- Advanced Topics

Lecture overview

Today's session

- Quick recap
- Maximum Margin IRL, Maximum Entropy IRL
- Scaling to non-linear reward functions and large state spaces
  - Maximum entropy deep inverse reinforcement learning
  - Generative Adversarial Imitation Learning (GAIL)
- Deep Reward Modeling
- Reinforcement Learning from Human Preferences
- Wrap-up and Q&A

Recap

Session 1 Recap

- Basic RL Formalization (MDPs)
- Apprenticeship Learning & IRL
- Feature Expectations & Matching

IRL via Gradient Optimization

To find θ, we typically iterate:

1. Policy Update

   Given $\mathcal{R}_\theta$, find optimal policy π_θ (using standard RL)

2. Reward Update

   Adjust θ to make features $\mu(\pi_\theta)$ similar to expert features $\mu(\pi_E)$

3. Gradient

   $\nabla_\theta \mathcal{L} \approx \mu(\pi_E) - \mu(\pi_\theta)$

Maximum Margin & Maximum Entropy IRL

Ambiguity of IRL and a First Approach (Maximum Margin IRL)

Many reward functions $\mathcal{R}$ can induce the same expert policy π_E

Which one is "correct" / which one should we pick?

Ambiguity of IRL and a First Approach (Maximum Margin IRL)

Many reward functions $\mathcal{R}$ can induce the same expert policy π_E
Which one is "correct" / which one should we pick?

Approach 1: Max-Margin IRL. Key idea: choose θ so the expert's feature expectations are better than every other policy by a margin

$$\begin{aligned} \max_{\theta,\gamma} \quad & \gamma \\ \text{s.t.} \quad & \langle \theta, \mu_E - \mu(\pi) \rangle \geq \gamma \quad \forall \pi, \qquad \|\theta\|_2 \leq 1. \end{aligned}$$

Equivalent hinge (structured large-margin) form:

$$\min_{\theta} \quad \frac{\lambda}{2}\|\theta\|_2^2 + \max_{\pi} \Big[\langle \theta, \mu(\pi) \rangle - \langle \theta, \mu_E \rangle \Big]$$

Maximum Margin IRL

Algorithm (cutting-plane / most-violating policy):

- Planning (separation oracle):
  $\hat{\pi} \leftarrow \arg\max_{\pi} \langle \theta, \mu(\pi) \rangle$ (solve MDP with $r_\theta(s,a) = \langle \theta, \phi(s,a) \rangle$)
- Add constraint $\langle \theta, (\mu_E - \mu(\hat{\pi})) \rangle \geq 1$ and re-solve the QP:

$$\min \tfrac{1}{2}\|\theta\|_2^2 \quad \text{s.t.} \quad \langle \theta, (\mu_E - \mu(\pi_i)) \rangle \geq 1, i = 1, \ldots, k$$

- Stop when $\langle \theta, (\mu_E - \mu(\hat{\pi})) \rangle \geq 1 - \varepsilon$

Subgradient for hinge form (using current $\hat{\pi}$):

$$\nabla_\theta \mathcal{L} = \mu(\hat{\pi}) - \mu_E + \lambda\theta$$

Maximum Entropy IRL

- MM-IRL: Arbitrarily picks a "best" reward that maximizes the gap
- Could argue that the above is not satisfactory — need a principled way to choose among the valid reward functions

❓ Alternatives?

Maximum Entropy IRL

- MM-IRL: Arbitrarily picks a "best" reward that maximizes the gap
- Could argue that the above is not satisfactory — need a principled way to choose among the valid reward functions

❓ Alternatives?

Maximum Entropy Philosophy

"Make the least number of assumptions about the distribution of policies consistent with the expert's demonstrations."

- MaxEnt IRL posits that the expert is not deterministic, but rather chooses trajectories with probability proportional to their exponentiated reward

Ziebart, B. D. et al. (2008). *Maximum entropy inverse reinforcement learning. AAAI.*

Maximum Entropy IRL

$$\max_{P(\tau)} \quad H(P(\tau)) \qquad \text{entropy over trajectories}$$

$$\text{s.t.} \quad \mu_E = \mathbb{E}_{P(\tau)}[\mu(\tau)]$$

The unique solution is a Boltzmann Distribution:

$$P(\tau \mid \theta) = \frac{1}{Z(\theta)} P(s_0) \prod_t P(s_{t+1} \mid s_t, a_t) \exp\left(\langle \theta, \mu(\tau) \rangle\right)$$

where $\mu(\tau) = \sum_t \phi(s_t, a_t)$ is the sum of features along τ

- $Z(\theta)$ is the partition function
- θ are the Lagrange multipliers (reward weights)
- Learning: Maximize log-likelihood of expert trajectories
- Gradient of the negative log-likelihood:

$$\nabla_\theta \mathcal{L}_{\text{MaxEnt-IRL}} = \mu(\pi_\theta) - \mu_E$$

with $\mu(\pi_\theta)$ being the expected feature counts under the MaxEnt trajectory distribution (not the greedy distribution)

Practical Implementation: MaxEnt IRL

```
def train_irl(trajs, feature_map, lr, n_iter=100):
    expert_features = compute_expert_features(trajs,
        feature_map) # average over tau

    for _ in range(n_iter):
        # 1) Reward r(s,a)
        reward = feature_map @ theta  # shape [S, A]

        # 2) Soft (entropy-regularized) planning
        policy = soft_value_iteration(reward, P)  # pi(a|s)
            \propto exp(Q-V)

        # 3) Expected feature counts under the MaxEnt
            distribution
        learner_features = expected_features(policy,
            feature_map)  # via visitation frequencies

        # 4) Gradient update (feature matching)
        theta += lr * (expert_features - learner_features)

return theta
```

Practical Implementation: MaxEnt IRL

```
# Soft value iteration (backward pass)
V = np.zeros(S)
for _ in range(soft_vi_iters):
    Q = reward + gamma * (P @ V)          # Q[s,a] = r + gamma
        \sum_{s'} P[s,a,s'] V[s']
    V = logsumexp(Q, axis=1)              # V[s] = log \sum_a
        exp(Q[s,a])
```

Practical Implementation: MaxEnt IRL

```
# Forward pass: visitation frequencies d_theta(s,a)
d_s = rho0.copy()                                  # state visitation
    at t=0
d_sa = np.zeros((S, A))
for t in range(T):
    d_sa += d_s[:, None] * pi                      # accumulate
        discounted visitation
    d_s_next = np.zeros(S)
    for s in range(S):
        for a in range(A):
            d_s_next += d_s[s] * pi[s, a] * P[s, a]
    d_s = gamma * d_s_next

learner_features = (d_sa[..., None] * feature_map).sum(axis
    =(0, 1)) # Learner feature expectations
```

Imitation Learning in Action

Frozen Lake

Rewards:

- Reach goal: $+1$
- Reach hole: 0
- Reach frozen: 0

» `maxent-irl.ipynb`

Challenges in Complex State Spaces

Challenges: Complicated State Spaces

Traditional IRL methods rely on tabular states or hand-crafted linear features

Challenges: Complicated State Spaces

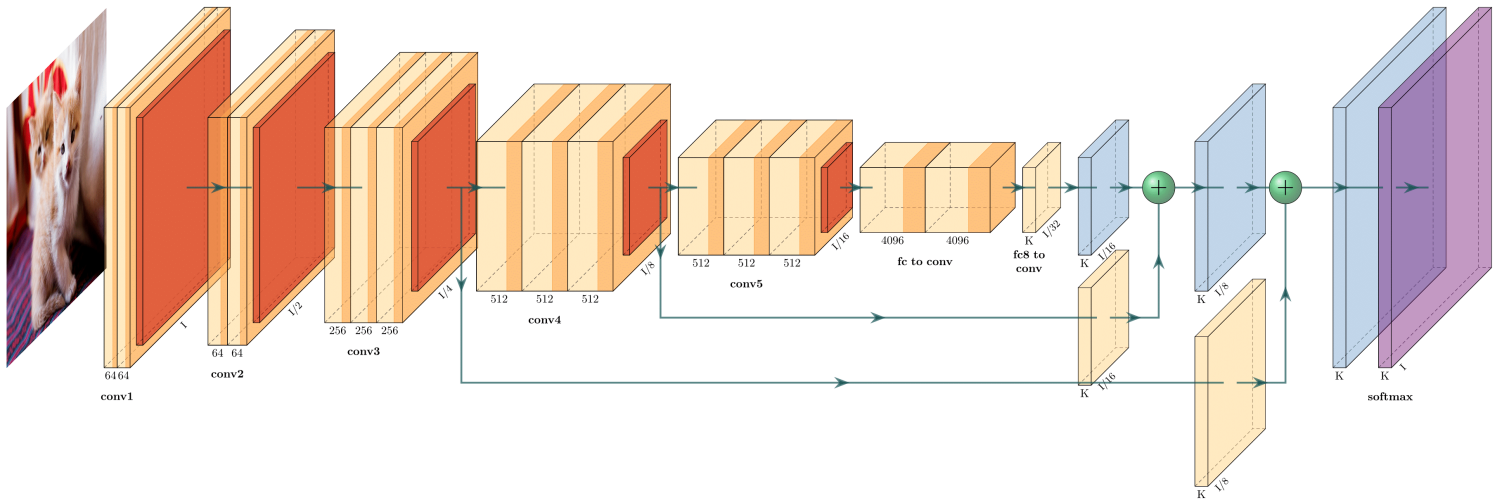
Traditional IRL methods rely on tabular states or hand-crafted linear features

But... Key challenges in modern scenarios:

- Non-linearity in transition dynamics and reward functions
- High-dimensional, raw input data (e.g., images)
- Sample inefficiency of traditional IRL solvers

Beyond Simple (Discrete) State Spaces

Deep Neural Networks enable us to approximate rewards and policies in complex, high-dimensional manifolds



[https://github.com/HarisIqbal88/PlotNeuralNet]

Maximum entropy deep inverse reinforcement learning

Limitations of Linear IRL

Standard MaxEnt IRL assumes a linear reward function:

$$\mathcal{R}(s, a) = \theta^T \phi(s, a)$$

- Restricted to linear reward functions and known features
  - Fails in high-dimensional or unknown feature spaces
- Deep IRL Solution. Replace linear model with a non-linear neural network:

$$\mathcal{R}_\theta(s, a) = g_\theta(\phi(s), a)$$

Deep MaxEnt IRL Formulation

Objective. As before, maximize log-likelihood of expert demonstrations:

$$\mathcal{L}(\theta) = \mathbb{E}_{\tau \sim \mathcal{D}}[\log P(\tau|\theta)] + \log P(\theta)$$

- Neural network g_θ maps state-action pairs to rewards
- Global normalization over behavior distribution via Boltzmann distribution
- Backpropagation used to compute gradients via matched expectations

Optimization and Gradients

Gradient of the log-likelihood:

$$\nabla_\theta \mathcal{L}(\theta) = (\mu_\mathcal{D} - \mathbb{E}_{\pi_\theta}[\mu]) \frac{\partial}{\partial \theta} \mathcal{R}_\theta$$

- Match expert feature counts with feature expectations for learned policy
- Allows for arbitrary non-linear feature representations
- Generalizes across complex, high-dimensional environments

Deep MaxEnt IRL: Algorithm

Deep MaxEnt IRL

1: **Input:** Expert demos $\mathcal{D}$, environment, learning rate α
2: Initialize reward network parameters θ
3: **while** not converged **do**
4: Sample trajectories $\tau \sim \pi_\theta$ (using current reward $\mathcal{R}_\theta$)
5: Compute gradient $\nabla_\theta \mathcal{L}(\theta)$:
$\nabla_\theta \mathcal{L} \leftarrow \mathbb{E}_{\tau \sim \mathcal{D}}[\sum \nabla_\theta \mathcal{R}_\theta(s, a)] - \mathbb{E}_{\tau \sim \pi_\theta}[\sum \nabla_\theta \mathcal{R}_\theta(s, a)] - \lambda\theta$
6: Update $\theta \leftarrow \theta + \alpha \nabla_\theta \mathcal{L}$
7: Update policy π_θ using RL on $\mathcal{R}_\theta$ (MaxEnt RL, e.g., soft value iteration)
8: **end while**
9: **Return:** Trained reward network $\mathcal{R}_\theta$

Evaluation: Expected Value Difference (EVD)

Expected Value Difference (EVD)

Measures the performance gap between the expert policy and the learned policy under the ground-truth reward function $\mathcal{R}^*$:

$$\text{EVD} = \mathbb{E}_{\tau \sim \pi_E} \left[\sum_t \mathcal{R}^*(s_t, a_t) \right] - \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_t \mathcal{R}^*(s_t, a_t) \right]$$

- A lower EVD indicates the learned policy π_θ achieves cumulative rewards closer to the expert performance
- Computing the EVD requires access to the hidden environment reward function $\mathcal{R}^*$
- Unlike behavioral cloning, this metric confirms that the agent has recovered the underlying task *intent* rather than merely mimicking trajectories

Empirical Results

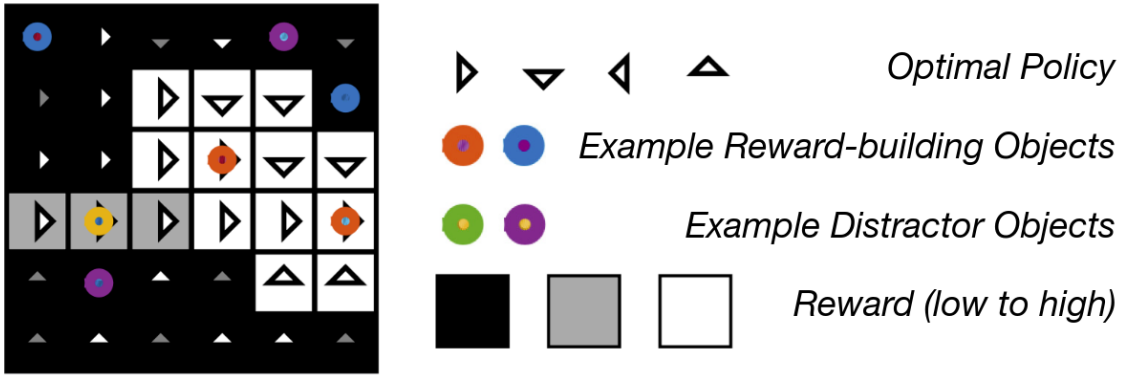


Figure 3: Objectworld benchmark. The true reward is displayed by the brightness of each cell and based on the surrounding object configuration. Only a subset of colors influences the reward, while the others serve as distracting features.

Wulfmeier, M., Ondruska, P., and Posner, I. (2015). *"Maximum entropy deep inverse reinforcement learning"*.

Empirical Results

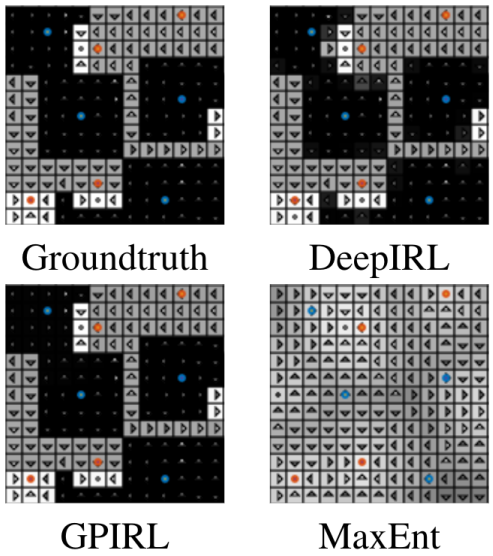


Figure 4: Reward reconstruction sample in Objectworld benchmark provided $N = 64$ examples and $C = 2$ colours with continuous features. White - high reward; black - low reward.

Wulfmeier, M., Ondruska, P., and Posner, I. (2015). *"Maximum entropy deep inverse reinforcement learning"*.

Empirical Results

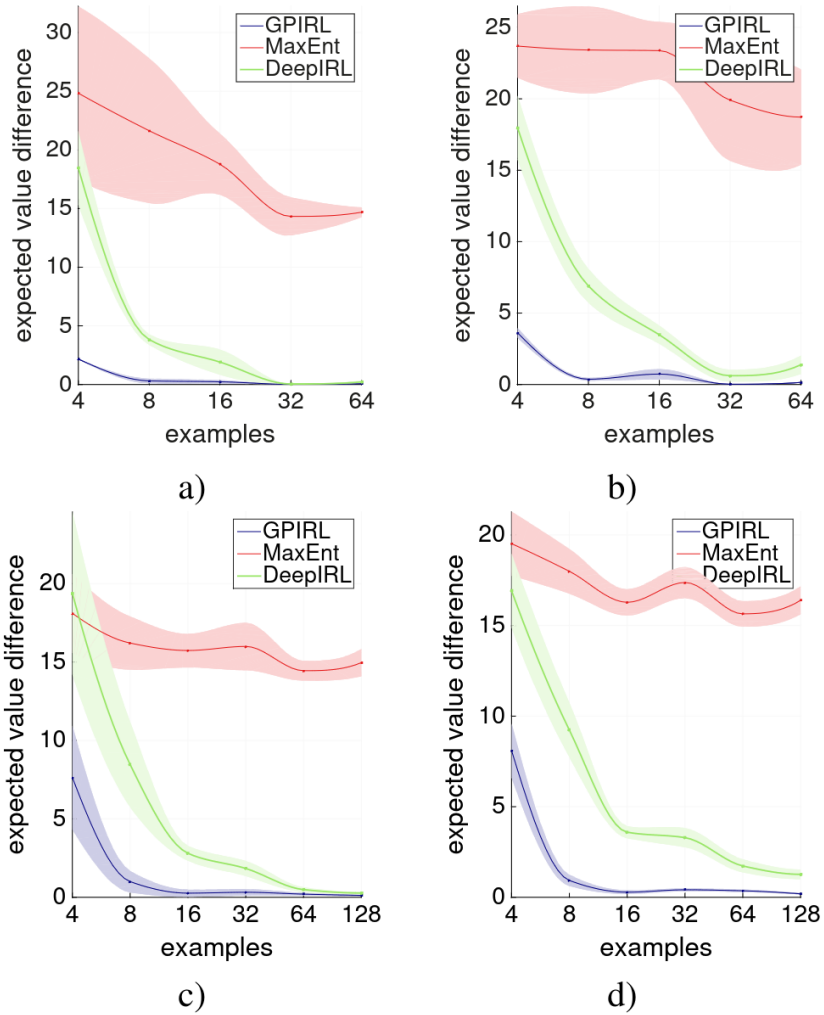


Figure 5: Objectworld benchmark. From top left to bottom right: expected value difference (EVD) with $C = 2$ colours and varying number of demonstrations N for training a) and transfer case b) with continuous and subsequently with discrete features in c) & d) ; As the number of demonstrations grows DeepIRL is able to quickly match performance of GPIRL on the task.

Wulfmeier, M., Ondruska, P., and Posner, I. (2015). *"Maximum entropy deep inverse reinforcement learning"*.

Empirical Results

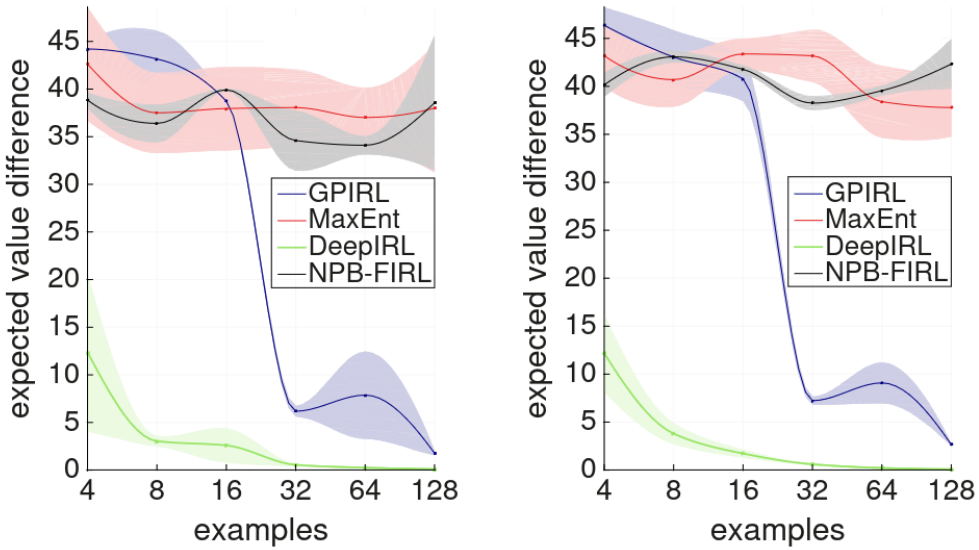


Figure 6: Value differences observed in the Binaryworld benchmark for GPIRL, MaxEnt and DeepIRL for the training scenario (left) and the transfer task (right).

Wulfmeier, M., Ondruska, P., and Posner, I. (2015). *"Maximum entropy deep inverse reinforcement learning"*.

Empirical Results

- Deep MaxEnt IRL outperforms linear methods on various tasks
- Faster convergence to high-reward regions
- Robustness to noise in demonstration data

Generative Adversarial Imitation Learning

From MaxEnt IRL to GAIL

MaxEnt IRL is powerful, but requires:

- Multiple inner-loop RL iterations
- Explicit calculation of feature expectations

From MaxEnt IRL to GAIL

MaxEnt IRL is powerful, but requires:

- Multiple inner-loop RL iterations
- Explicit calculation of feature expectations

Can we bypass explicit reward recovery and optimize the policy directly?

GAIL: Connection to Generative Adversarial Networks

Generative Adversarial Imitation Learning (GAIL) bridges imitation learning and Generative Adversarial Networkds (GANs):

$$\min_{\pi} \max_{D} \mathbb{E}_{\pi}[\log D(s, a)] + \mathbb{E}_{\pi_E}[\log(1 - D(s, a))]$$

GAIL: Connection to Generative Adversarial Networks

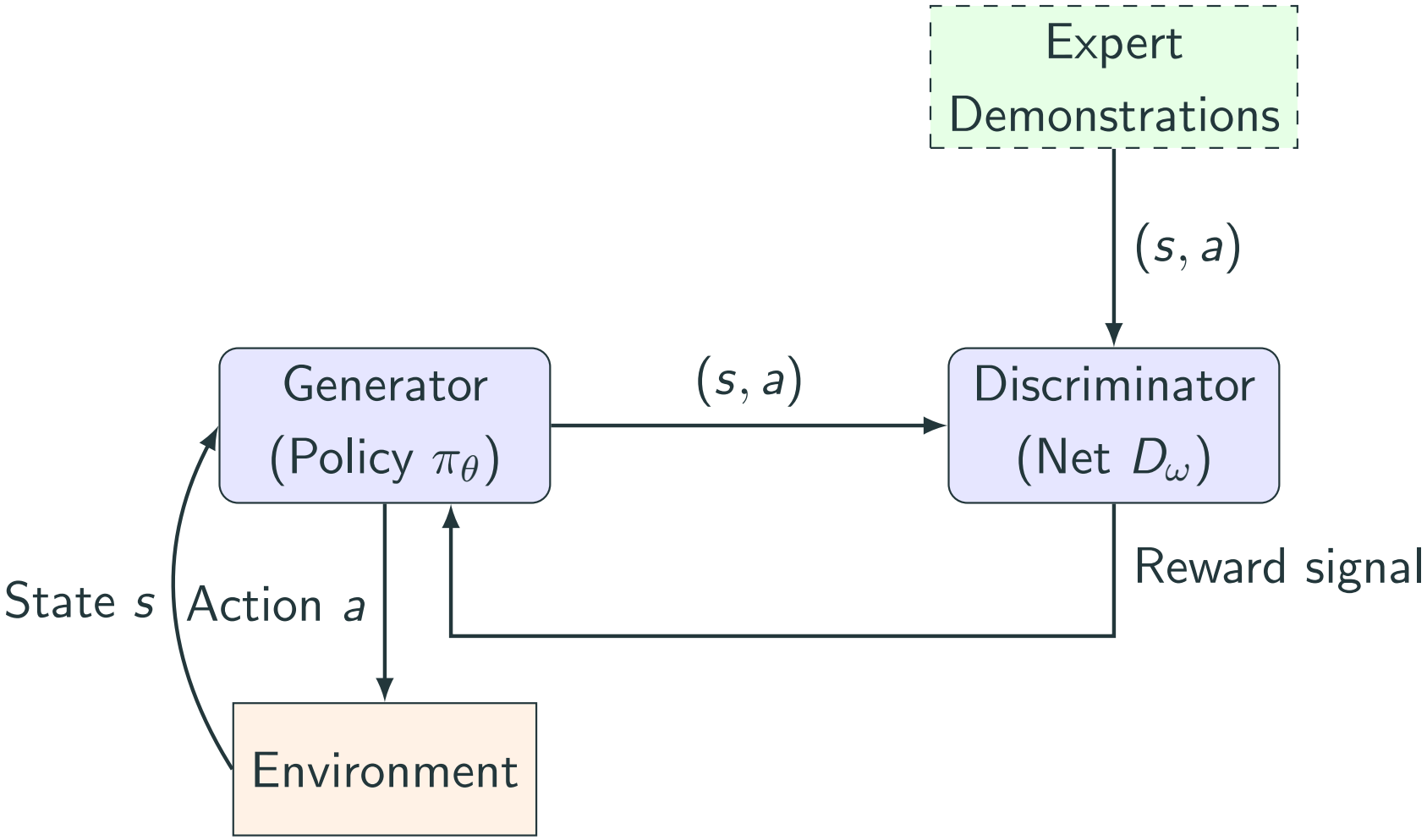
Generative Adversarial Imitation Learning (GAIL) bridges imitation learning and Generative Adversarial Networkds (GANs):

$$\min_{\pi} \max_{D} \mathbb{E}_{\pi}[\log D(s,a)] + \mathbb{E}_{\pi_E}[\log(1 - D(s,a))]$$

Training. Alternates between improving the policy (TRPO) and improving the discriminator

GANs. Goodfellow, I. J. et al. (2014). *"Generative adversarial nets"*.
GAIL. Ho, J. and Ermon, S. (2016). *"Generative adversarial imitation learning"*.

GAIL Architecture

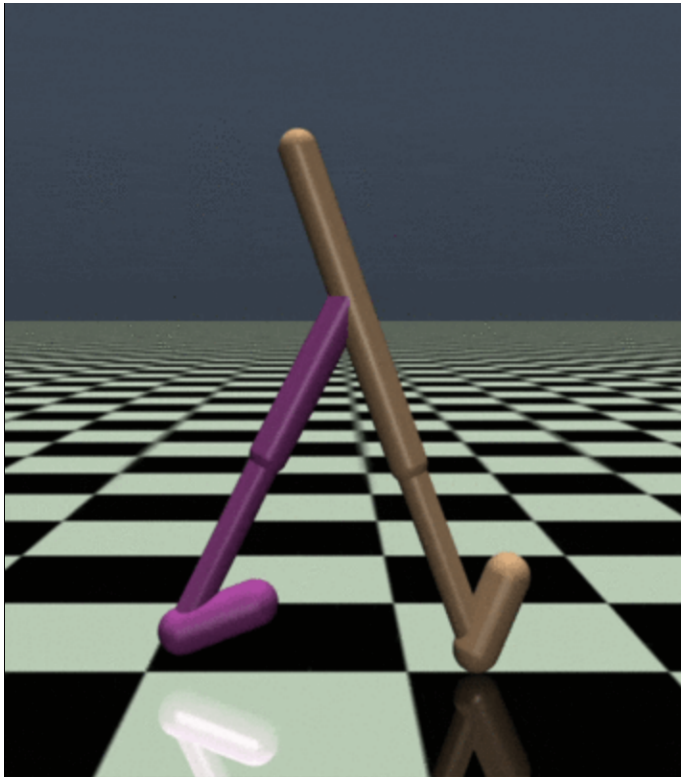


GAIL Algorithm Components

- Generator: Policy π mimicking the expert
- Discriminator: Classifier D distinguishing generated and expert trajectories
- Reward Proxy: $-\log(1 - D(s, a))$: high in regions where the expert visited but the policy has not, effectively acting as an incentive for the policy to explore those "missing" areas.

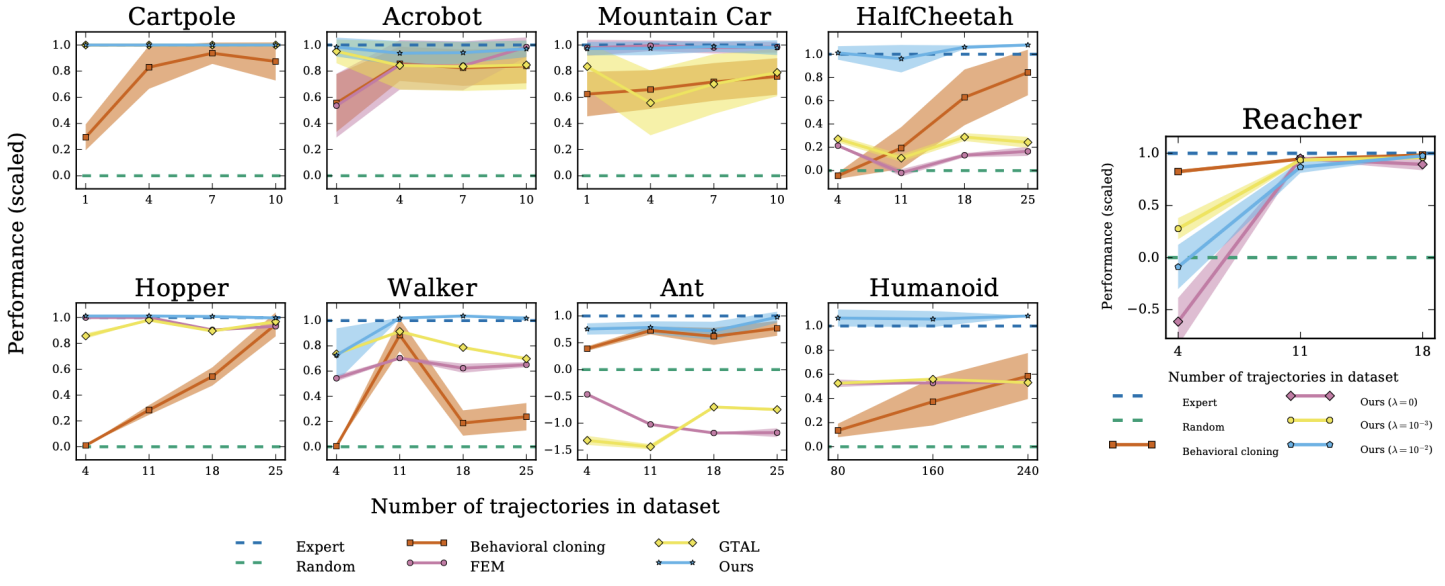
Evaluation

Reults on MuJoCo based environments (physics simulator)



Walker environment

Evaluation



(a) (b)

Figure 1: (a) Performance of learned policies. The y-axis is negative cost, scaled so that the expert achieves 1 and a random policy achieves 0. (b) Causal entropy regularization λ on Reacher.

Ho, J. and Ermon, S. (2016). *"Generative adversarial imitation learning"*.

GAIL Strengths & Weaknesses

- Strengths
  - Directly learns policy from samples
  - Avoids costly inner-loop RL training
  - Scales to high-dimensional state spaces using deep networks
- Weaknesses/Limitations
  - Adversarial training can be unstable
  - Requires a large number of expert trajectories
  - No explicit interpretable reward function

Technical Tools for Large-Scaling Learning in Place
But Where to Get the Data?

Challenges: Complicated State Spaces

Providing (optimal) demonstrations can be hard

E.g., providing demonstration of "optimal" driving / surgery

Challenges: Complicated State Spaces

Providing (optimal) demonstrations can be hard
E.g., providing demonstration of "optimal" driving / surgery

- Can we ease the process of providing information about the reward function?
  - Different types of feedback, e.g., comparisons or corrections
  - Query for feedback where it really helps to improve an agent's performance

Deep Reward Modeling and Modeling Preferences

Deep Reward Modeling (DRM)

- Expert demonstrations might be hard to provide or expensive, e.g., correct response to diverse prompts
- If we want interpretability and safety, we might prefer explicitly modeling preferences and corresponding rewards

Deep Reward Modeling (DRM)

- Expert demonstrations might be hard to provide or expensive, e.g., correct response to diverse prompts
- If we want interpretability and safety, we might prefer explicitly modeling preferences and corresponding rewards

Learn a model of human preferences.
E.g., train a neural network $\mathcal{R}_\theta$ to predict human preferences over trajectories

Deep Reward Modeling (DRM)

- Expert demonstrations might be hard to provide or expensive, e.g., correct response to diverse prompts
- If we want interpretability and safety, we might prefer explicitly modeling preferences and corresponding rewards

Learn a model of human preferences.
E.g., train a neural network $\mathcal{R}_\theta$ to predict human preferences over trajectories

- Human input to comparisons (pairwise ranking) is often cleaner than full trajectories
- Providing such feedback can often be cognitively less demanding and performed faster

Preference Modeling Overview

- How to model preferences?

Preference Modeling Overview

- How to model preferences?
  - Many options making different underlying assumptions
    - Simple pairwise comparisons: Bradley-Terry (BT) model
    - Rankings / top-K: Plackett-Luce model (generalization of BT)
    - Latent utility (stochastic): Thurstone model

Preference Modeling Overview

- How to model preferences?
  - Many options making different underlying assumptions
    - Simple pairwise comparisons: Bradley-Terry (BT) model
    - Rankings / top-K: Plackett-Luce model (generalization of BT)
    - Latent utility (stochastic): Thurstone model

Bradley-Terry Model

Probability model for the outcome of pairwise comparisons between items s.t.

$$P(i \succ j) = \frac{s_i}{s_i + s_j}$$

where s_i is a positive real-valued score of item i

Ties and Noise in Preferences

- Allow a third label (tie/indifferent) to reduce annotator burden and capture indifference (less annotator fatigue and better data quality)
- Model judgment noise via an annotator "temperature" or reliability term

Tie-aware Bradley-Terry (Davidson model)

Let the scores be $s_i = \exp(\mathcal{R}_i)$ and the tie parameter $\nu \geq 0$. For a pair (i, j):

$$P(i \succ j) = \frac{s_i}{s_i + s_j + 2\nu\sqrt{s_i s_j}}$$

$$P(j \succ i) = \frac{s_j}{s_i + s_j + 2\nu\sqrt{s_i s_j}}$$

$$P(i \sim j) = \frac{2\nu\sqrt{s_i s_j}}{s_i + s_j + 2\nu\sqrt{s_i s_j}}$$

Plackett-Luce: Generalizing to Rankings

Plackett-Luce

Assumes K items with scores $s_1, \ldots, s_K$. The probability of a ranking $\pi = (\pi_1, \ldots, \pi_K)$ is

$$P(\pi) = \prod_{m=1}^{K} \frac{s_{\pi_m}}{\sum_{j=m}^{K} s_{\pi_j}}.$$

- Reduces to the BT model for 2 items
- Relies on Luce's choice axiom:
  - Independence of irrelevant alternatives

    Sometimes critiziced: If you prefer Coffee over Tea, adding Hot Chocolate to the menu shouldn't change that ratio, but in the real world, it often does

Thurstone Model: Latent Utility

Thurstone Model (Case V)

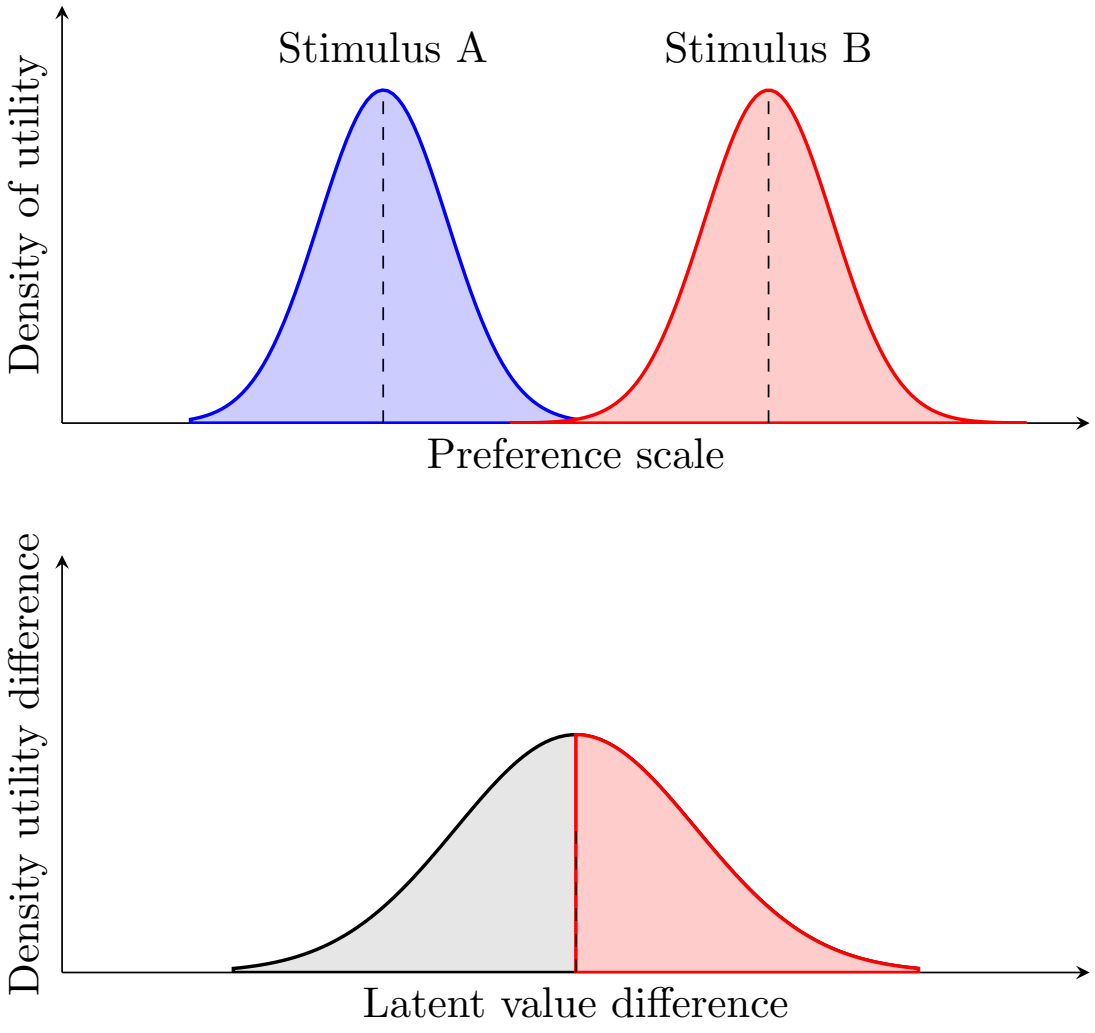
The Thurstone model assumes that the utility of an item i is a random variable $U_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$. An agent prefers item i over item j if $U_i > U_j$.

Under the assumption of equal variances and zero correlations, the probability of preferring i over j is

$$P(i \succ j) = \Phi\left(\frac{\mu_i - \mu_j}{\sigma\sqrt{2}}\right)$$

- Models stochastic noise in the human judgment process
- Allows for modeling variances/correlations if we relax the Case V assumptions

Thurstone Model: Latent Utility



Preferences in Reinforcement Learning

In RL, we often learn a reward function $\mathcal{R}_\theta$ consistent with human preferences over trajectory pairs (τ_1, τ_2) according to the BT model:

Preference Probabilities:

$$P(\tau_1 \succ \tau_2) = \frac{\exp(\mathcal{R}_\theta(\tau_1))}{\exp(\mathcal{R}_\theta(\tau_1)) + \exp(\mathcal{R}_\theta(\tau_2))}$$

- Optimization: Minimizing negative log-likelihood of this model is equivalent to Binary Cross-Entropy (BCE) loss (often used in DPO/RLHF — see later)

Number of potential trajectory pairs is $O(N^2)$

Preference Learning: Dataset and Objective

Dataset Construction

- Preference dataset $\mathcal{D}$ containing N human-labeled comparisons:

$$\mathcal{D} = \left\{ (\tau_1^{(i)}, \tau_2^{(i)}, y^{(i)}) \right\}_{i=1}^{N}$$

- $\tau_1^{(i)}, \tau_2^{(i)}$: pairs of trajectories sampled from the environment
- $y^{(i)} \in \{1, 2\}$: label indicating which trajectory the human prefers

Training Objective

- Under the BT model, the probability of the human's preference is:

$$P(y = 1 | \mathcal{R}_\theta) = \frac{\exp(\mathcal{R}_\theta(\tau_1))}{\exp(\mathcal{R}_\theta(\tau_1)) + \exp(\mathcal{R}_\theta(\tau_2))}$$

- Train θ by minimizing the negative log-likelihood (NLL):

Reward Learning Process

Commonly implemented continuous reward learning process:

1. Collect trajectories τ_i using some policy π
2. Get human feedback (rank pairs) on trajectories samples from the trajectory buffer
3. Update $\mathcal{R}_\theta$ by maximizing preference likelihood
4. Update policy π based on $\mathcal{R}_\theta$
5. Iterate

Reward Learning Process

Commonly implemented continuous reward learning process:

1. Collect trajectories τ_i using some policy π
2. Get human feedback (rank pairs) on trajectories samples from the trajectory buffer
3. Update $\mathcal{R}_\theta$ by maximizing preference likelihood
4. Update policy π based on $\mathcal{R}_\theta$
5. Iterate

Moving Targets

- The policy π is updated using the *current* learned reward model r_θ
- As π improves, it generates new, higher-quality trajectories; the human is then queried on these *better* trajectories
- $\rightarrow$ policy and reward model co-evolve

Reward Learning Process: Selecting Pairs of Trajectories

Selecting pairs of trajectories for labeling

1. Not a single reward model is trained, but a whole ensemble (bootstrapped)
2. Used to select "important pairs" based on the uncertainty in the reward function estimator
   - A large number of pairs of trajectory segments of length k is sampled from the latest agent-environment interactions
   - Each reward predictor in the ensemble is used to predict which segment will be preferred from each pair
   - Pairs for which the predictions have the highest variance across ensemble members are selected

Christiano, P. F. et al. (2017). *"Deep reinforcement learning from human preferences"*.

Trajectory Credit Assignment: Segment length k

- Typically compare trajectory segments and not whole trajectories — reduces long-horizon ambiguity
- Define a segment of length k from trajectory $\tau = (s_1, a_1, \ldots, s_T)$ as $\tau_{t:t+k-1}$

  Aggregate score

$$\mathcal{R}_\theta(\tau_{t:t+k-1}) = \sum_{u=t}^{t+k-1} \mathcal{R}_\theta(s_u, a_u) \quad \text{or} \quad \bar{\mathcal{R}}_\theta = \frac{1}{k} \sum_{u=t}^{t+k-1} \mathcal{R}_\theta(s_u, a_u)$$

Segment-level preference model

For a pair of segments $p = \tau^{(1)}_{t_1:t_1+k-1}$ and $q = \tau^{(2)}_{t_2:t_2+k-1}$,

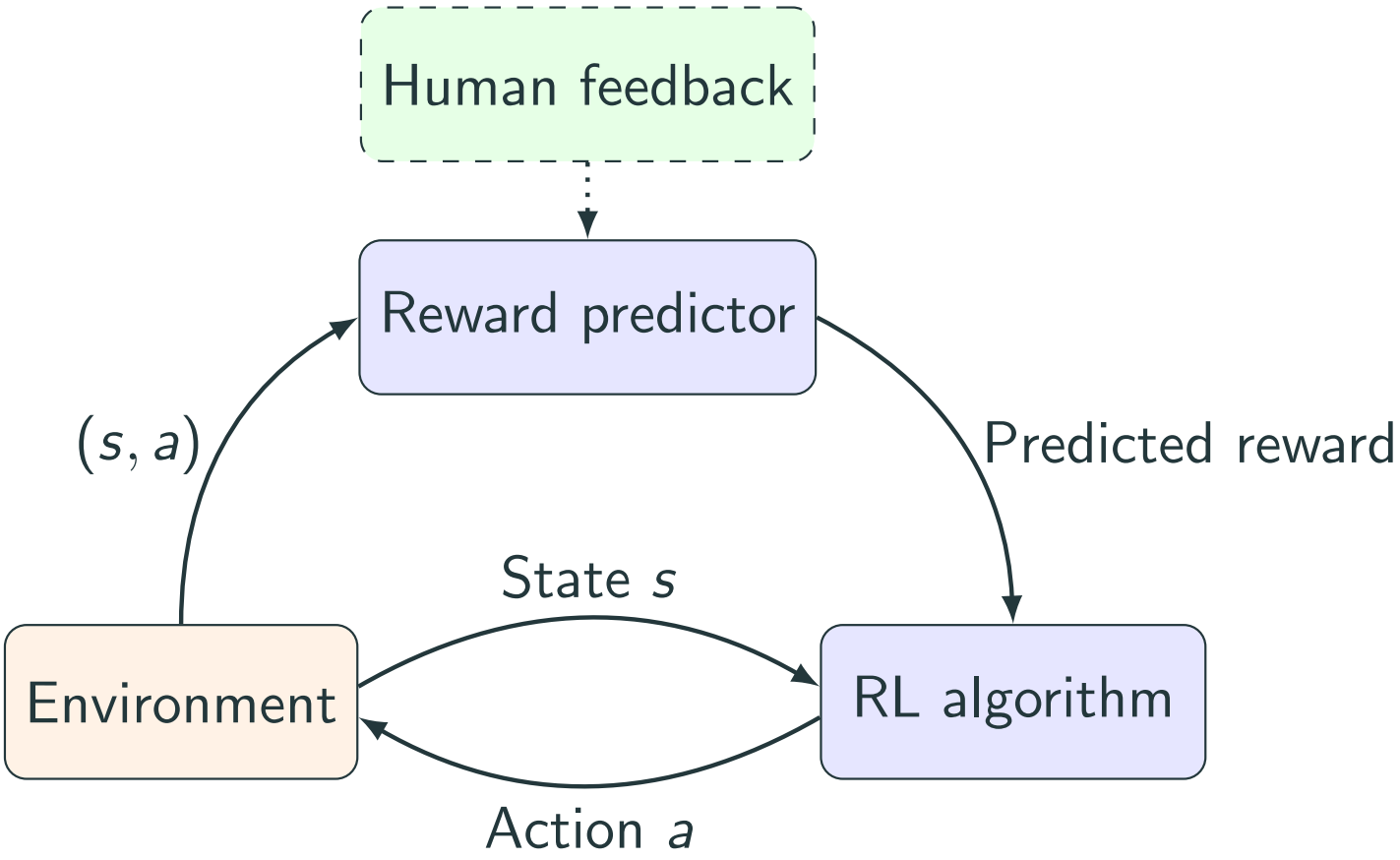
$$P(p \succ q) = \sigma\Big(R_\theta(p) - R_\theta(q)\Big)$$

Trajectory Credit Assignment: Segment length k

Choosing k (trade-offs)

- Small k: cheap, localized signal; may be myopic/noisy for delayed outcomes
- Large k: better captures long-horizon effects; higher label cost and cognitive load
- Practical heuristic: pick the shortest k that contains a complete "atomic outcome"
  - validate with annotator agreement

Reinforcement Learning from Human Feedback (RLHF)



Reproduced from Christian, B. (2020). *The alignment problem: Machine learning and human values*.

RLHF Results (Example)

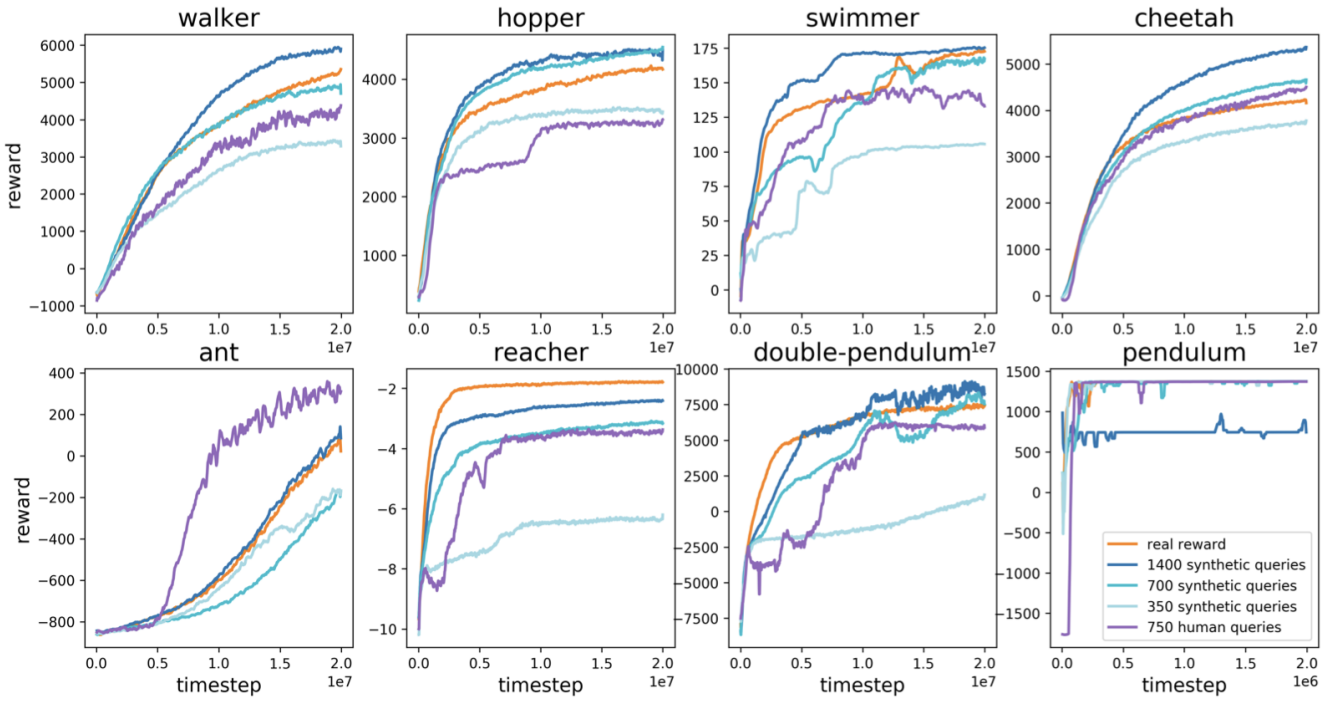


Figure 2: Results on MuJoCo simulated robotics as measured on the tasks' true reward. We compare our method using real human feedback (purple), our method using synthetic feedback provided by an oracle (shades of blue), and reinforcement learning using the true reward function (orange). All curves are the average of 5 runs, except for the real human feedback, which is a single run, and each point is the average reward over five consecutive batches. For Reacher and Cheetah feedback was provided by an author due to time constraints. For all other tasks, feedback was provided by contractors unfamiliar with the environments and with our algorithm. The irregular progress on Hopper is due to one contractor deviating from the typical labeling schedule.

Christiano, P. F. et al. (2017). *"Deep reinforcement learning from human preferences"*.

Additional Key References

- T-REX/D-REX preference learning from ranked suboptimal demonstrations

  Brown, D. et al. (2019). *Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations*, *ICML*.

- Efficient preference-based RL with segment queries

  Lee, K., Smith, L., and Abbeel, P. (2021). *PEBBLE: Feedback-Efficient Interactive Reinforcement Learning via Relabeling Experience and Unsupervised Pre-training*, *ICML*.

- Learning from preferences and demonstrations

  Ibarz, B. et al. (2018). *Reward learning from human preferences and demonstrations in atari*, *NeurIPS*.

Wrap Up and Outlook

Session 2 Summary: The Path to Alignment

What we covered today

- Scaling to non-linear reward fcts & complex state spaces
- Preference modelling
- Reinforcement Learning from Human Feedback (RLHF)

Session 2 Summary: The Path to Alignment

What we covered today

- Scaling to non-linear reward fcts & complex state spaces
- Preference modelling
- Reinforcement Learning from Human Feedback (RLHF)

❓ Are we done?

Session 2 Summary: The Path to Alignment

What we covered today

- Scaling to non-linear reward fcts & complex state spaces
- Preference modelling
- Reinforcement Learning from Human Feedback (RLHF)

❓ Are we done?

Open challenges

- *Minimizing labeling burden:* Active reward learning
- *Constraints:* Rewards only encode what is good/bad but not what should never be done — constraints for generalizable specifications
- *Learning constraints:* How to specify constraints? Can we learn them from data?
- *Aligning constraints with human goals:* Can we infer constraints from human feedback?

Looking Ahead to Session 3

- Active reward learning
- Constraints
- Learning Constraints
- Inference with Constraints

Q&A

Next: Session 3 - Constraint Learning

References & Resources

References

Session 2

- **RLHF.**
  Christiano, P. F. et al. (2017). *"Deep reinforcement learning from human preferences"*.
- **LLM training.**
  Ouyang, L. et al. (2022). *"Training language models to follow instructions with human feedback"*.

References

Session 2 - DPO

- **DPO.**
  Rafailov, R. et al. (2023). *"Direct preference optimization: Your language model is secretly a reward model"*.
- **IPO.**
  Garg, S. et al. (2025). *Ipo: Your language model is secretly a preference classifier*, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.
- **KTO.**
  Ethayarajh, K. et al. (2024). *"Kto: Model alignment as prospect theoretic optimization"*.

References

Session 2 - DPO

- **Fine tuning language models.**
  Ziegler, D. M. et al. (2019). *"Fine-tuning language models from human preferences, 2020"*.
- **SLiC-HF.**
  Zhao, Y. et al. (2023). *"Slic-hf: Sequence likelihood calibration with human feedback"*.

References

Session 2 - RLAIF & Constitutional AI

- **Constitutional AI.**
  Bai, Y. et al. (2022). *"Constitutional ai: Harmlessness from ai feedback"*.
- **Self-critiquing models.**
  Saunders, W. et al. (2022). *"Self-critiquing models for assisting human evaluators"*.
- **Step-by-step verification.**
  Lightman, H. et al. (2024). *Let's verify step by step*, *International Conference on Learning Representations*.