Inverse RL and RL from Human Feedback

Instructor: Daniel Brown

[Some slides adapted from Sergey Levine (CS 285) and Alina Vereshchaka (CSE4/510)]
Course feedback is open

• Extra credit if class response rate is 70% or higher
  • Sliding scale if we reach 70%:
    • Extra credit points = response rate percentage / 10
Reward Learning
(Inverse Reinforcement Learning)

Why? What is the human’s reward function?
Why not just imitate behavior? (Behavioral Cloning)

What would the human do?

[Diagram showing a robot observing a human, with the action and observation processes represented by arrows and a policy \( \pi \).]
Human Intent Inference
Inverse Reinforcement Learning

- Given
  - MDP without a reward function
  - Demonstrations from an optimal policy $\pi^*$

- Recover the reward function $R$ that makes $\pi^*$ optimal
Imitation Learning

Behavioral Cloning

• Answers the “How?” question
• Mimic the demonstrator
• Learn mapping from states to actions
• Computationally efficient
• Compounding errors

\[ \Rightarrow \pi \]

Inverse Reinforcement Learning

• Answers the “Why?” question
• Explain the demonstrator’s behavior
• Learn a reward function capturing the demonstrator’s intent
• Can require lots of data and compute
• Better generalization. Can recover from arbitrary states

\[ \Rightarrow R \Rightarrow \pi \]
IRL Example: Teaching a robot to navigate through demonstrations
Toy version
What is the reward?

R( ) = ?

R( ) = ?

R( ) = ?

R( ) = ?

R( ) = ?
What is the reward?

\[ R(\boxed{\text{blue}}) = ? \]
\[ R(\boxed{\text{yellow}}) = ? \]
\[ R(\boxed{\text{green}}) = ? \]
\[ R(\boxed{\text{red}}) = ? \]
\[ R(\boxed{\text{white}}) = ? \]
What is the reward?

\[ R(\square) = ? \]
\[ R(\square) = ? \]
\[ R(\square) = ? \]
\[ R(\square) = ? \]
\[ R(\square) = ? \]
What is the reward?

$R(\square\text{blue}) = +1$

$R(\square\text{yellow}) = 0$

$R(\square\text{green}) = 0$

$R(\square\text{red}) = -1$

$R(\square\text{white}) = 0$
What is the reward?

R( ) = +10
R( ) = 0
R( ) = 0
R( ) = -10
R( ) = 0
What is the reward?

- $R(\square) = +10$
- $R(\square) = -1$
- $R(\square) = -1$
- $R(\square) = -1$
- $R(\square) = -10$
- $R(\square) = -1$
What is the reward?

\[
R(\quad ) = 0
\]

\[
R(\quad ) = 0
\]

\[
R(\quad ) = 0
\]

\[
R(\quad ) = 0
\]

\[
R(\quad ) = 0
\]
What is the reward?

\[ R(\quad) = c \]

\[ R(\quad) = c \]

\[ R(\quad) = c \]

\[ R(\quad) = c \]

\[ R(\quad) = c \]

\[ R(\quad) = c \]
Inverse Reinforcement Learning

- **Given**
  - MDP without a reward function
  - Demonstrations from an optimal policy $\pi^*$

- **Recover the reward function $R$ that makes $\pi^*$ optimal**

- **Ill-Posed Problem**
  - Infinite number of reward functions that can make $\pi^*$ optimal
    - Trivial all zero reward
    - Constant reward
    - $aR + c$ (positive scaling $a > 0$, and affine shifts)
Basic IRL Algorithm

- Start with demonstrations, $D$
- Guess initial reward function $R_0$
- $\hat{R} = R_0$
- Loop:
  - Solve for optimal policy $\pi^*_\hat{R}$
  - Compare $D$ and $\pi^*_\hat{R}$
  - Update $\hat{R}$ to try and make $D$ and $\pi^*_\hat{R}$ more similar
Feature count matching

• Assume the reward function is a linear combination of features:

\[ R(s) = w^T \phi(s) = \sum_{i=1}^{K} \omega_i \phi_i(s) \]

• Value function becomes linear combination of (discounted) feature expectations:

\[ V_{R}^{\pi} = \mathbb{E}_{\pi} \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t) \right] \]

Feature count matching

• Assume the reward function is a linear combination of features:

\[ R(s) = w^T \phi(s) \]

• Value function becomes linear combination of (discounted) feature expectations:

\[ V^\pi_R = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t w^t \phi(s_t) \right] \]

Feature count matching

• Assume the reward function is a linear combination of features:

\[ \phi(s) = \begin{bmatrix} \phi_1(s) \\ \phi_2(s) \\ \vdots \\ \phi_K(s) \end{bmatrix}, \quad R(s) = w^T \phi(s) = \sum_{i=1}^{K} w_i \phi_i(s) \]

• Value function becomes linear combination of (discounted) feature expectations:

\[ V_{\pi}^R = w^T \mathbb{E}_{\pi} \left[ \sum_{t=0}^{\infty} \gamma^t \phi(s_t) \right] = w^T \mu_\pi \]

Inverse reinforcement learning: feature matching

(Abbeel and Ng 2004, Syed and Schapire 2007)

• If $||w||_1 \leq 1$, then

$$V_{R}^{\pi^*} - V_{R}^{\pi_{robot}} = w^T (\mu_{\pi^*} - \mu_{\pi_{robot}}) \leq ||w||_1 ||\mu_{\pi^*} - \mu_{\pi_{robot}}||_\infty$$

• If feature expectations match, then expected returns are identical.

• Idea: Can we update the reward guess $\hat{R}$ so the feature counts get closer?

Maximum Entropy IRL
(Ziebart et al. 2008)

• Collect M demonstrations \( D = \{\tau_1, \ldots, \tau_M\} \)
• Initialize reward weights \( w \)
• Loop
  • Solve for (soft) optimal policy \( \pi(a|s) \) via Value Iteration
  • Solve for state visitation frequencies \( p(s|\pi) \)
  • Compute weight update \( w \leftarrow w + \alpha \left( \frac{1}{M} \sum_{\tau \in D} \sum_{s \in t} \phi(s) - \sum_s p(s|\pi) \phi(s) \right) \)
Watch This: Scalable Cost-Function Learning for Path Planning in Urban Environments

Markus Wulfmeier\textsuperscript{1}, Dominic Zeng Wang\textsuperscript{1} and Ingmar Posner\textsuperscript{1}

Fig. 1: Schema for training neural networks in the Maximum Entropy paradigm for IRL.
How to avoid fully solving MDP

\[ P(\tau) = \frac{e^{R_\theta(\tau)}}{Z} \]

\[ Z = \int e^{R_\theta(\tau)} d\tau \]

• Estimate \( Z \) with a finite set of trajectories \( Z_\tau \).

• Loop:
  • Update parameters \( \theta \) so demonstrations have higher reward than trajectories in \( Z_\tau \).

  • Optionally
    • Update \( Z_\tau \)
How to make this more tractable

\[ P(\tau) = \frac{e^{R_\theta(\tau)}}{Z} \]

Uniform sampling to approximate Z.

Noisy perturbations of demonstrations to approximate Z

Use current policy to approximate Z.
Alternate between a few steps of reward updates and a few steps of policy updates.
Finn et al. “Guided Cost Learning.” 2016
GANs (Generative Adversarial Networks)
GAIL (Generative Adversarial Imitation Learning)

Ho and Ermon, 2016
What if we don’t want just a single reward estimate?

- Can we get a samples from the full Bayesian posterior?

\[ P(R|D) \propto P(D|R)P(R) \]
Markov Chain Monte Carlo (MCMC)

Markov chain:

\[
\begin{array}{c}
X_1 \rightarrow X_2 \rightarrow X_3 \rightarrow X_4 \\
\end{array}
\]

\[
P(X_1) \hspace{1cm} P(X_t|X_{t-1})
\]

Stationary Distribution:

\[
P_\infty(X) = P_\infty+1(X) = \sum_x P(X|x)P_\infty(x)
\]

MCMC is a sampling approach for Bayesian inference where we construct a Markov chain such that the stationary distribution is the posterior distribution we care about.
MCMC (Metropolis Hastings Algorithm)

- We want to sample from $P(R|D)$
- Start with random sample $r_0$
- Loop
  - Sample $r' \sim q(R_{t+1}|r_t)$
  - With probability $\min \left\{ 1, \frac{P(r'|D)}{P(r_t|D)} \right\}$ set $x_{t+1} = r'$
  - Else set $r_{t+1} = r_t$

Assume $q$ is symmetric. For example, a Gaussian distribution with mean $x_t$ and standard deviation $\sigma$

Normalizing constant cancels in the ratio!

$$P(D) = \sum_r P(r)P(D|r)$$

$$P(r'|D) = \frac{P(D|r')P(r')}{P(D|x_t)P(x_t)}$$

$$P(r_{t+1}|D) = \frac{P(D|x_{t+1})P(x_{t+1})}{P(D|x_t)P(x_t)}$$
Bayesian Inverse Reinforcement Learning
(Ramachandran and Amir 2007)

• Assume demonstrator is Boltzman rational
  • Demonstrator follows a softmax policy with inverse temperature $c$

$$P(D|R) = \prod_{(s,a) \in D} \frac{e^{cQ^*(s,a,R)}}{\sum_{b \in A} e^{cQ^*(s,b,R)}}$$

$Q^*(s, a, R) =$ How much reward will I expect to see if I take action $a$ in state $s$ and act optimally thereafter.
Bayesian Inverse Reinforcement Learning
(Ramachandran and Amir 2007)

• Assume demonstrator is Boltzmann rational
  • Demonstrator follows a softmax policy with inverse temperature $c$

$$P(D|R) = \prod_{(s,a) \in D} \frac{e^{cQ^*(s,a,R)}}{\sum_{b \in A} e^{cQ^*(s,b,R)}}$$

$$P((s,\leftarrow)|R) = \frac{e^{Q^*(s,\leftarrow,R)}}{e^{Q^*(s,\leftarrow,R)} + e^{Q^*(s,\rightarrow,R)}}$$
Bayesian Inverse Reinforcement Learning
(Ramachandran and Amir 2007)

• Assume demonstrator is Boltzman rational
  • Demonstrator follows a softmax policy with inverse temperature $c$

$$P(D|R) = \prod_{(s,a) \in D} \frac{e^{cQ^*(s,a,R)}}{\sum_{b \in A} e^{cQ^*(s,b,R)}}$$

• Perform Bayesian inference (MCMC) to sample from posterior distribution

$$P(R|D) \propto P(D|R)P(R)$$
RL from Human Feedback (RLHF)
RL from Human Preferences
This is similar to how they train GPT models.
Prompts Dataset

$\mathbf{x}: A \text{ dog is...}$

Initial Language Model

Tuned Language Model (RL Policy)

Parameters Frozen*  

RLHF Tuned Text  

Y: a furry mammal  

Y: man's best friend

Reinforcement Learning Update (e.g. PPO)

$\theta \leftarrow \theta + \nabla_{\theta} J(\theta)$

Reward (Preference) Model

$r_{\theta}(y|x)$

$-\lambda_{KL} D_{KL}(\pi_{PPO}(y|x) \parallel \pi_{\text{base}}(y|x))$

KL prediction shift penalty
Why would you want to learn a reward from ranked examples?
Inverse Reinforcement Learning

Prior approaches ...

1. Typically couldn’t do much better than the demonstrator.

2. Were hard to scale to complex problems.

Inverse Reinforcement Learning

Prior approaches ...

1. Typically couldn’t do much better than the demonstrator.
   Find a reward function that explains the ranking, allowing for extrapolation.
2. Were hard to scale to complex problems.

Inverse Reinforcement Learning

Prior approaches ...

1. Typically couldn’t do much better than the demonstrator.
   Find a reward function that explains the ranking, allowing for extrapolation.
2. Were hard to scale to complex problems.
   Reward learning becomes a supervised learning problem.

Trajectory-ranked Reward Extrapolation (T-REX)

< ... <

Pre-ranked demonstrations

Trajectory-ranked Reward Extrapolation (T-REX)

Reward Function

\[ R_\theta : S \rightarrow \mathbb{R} \]

Examples of S:

Current Robot Joint Angles and Velocities

\[ \begin{align*}
\text{Current Robot Joint Angles and Velocities} & \quad \rightarrow 0.5 \\
\end{align*} \]

\[ \begin{align*}
\text{Current Robot Joint Angles and Velocities} & \quad \rightarrow -0.7 \\
\end{align*} \]
Reward Function

\[ R_{\theta} : S \rightarrow \mathbb{R} \]

Examples of \( S \):

Current Robot Joint Angles and Velocities

\[
\begin{align*}
&\rightarrow 0.5 \\
&\rightarrow -0.7
\end{align*}
\]

Short Sequence of Images

\[
\begin{align*}
&\rightarrow 0.9 \\
&\rightarrow -1.2
\end{align*}
\]
Trajectory-ranked Reward Extrapolation (T-REX)

\[
\tau_1 \prec \tau_2 \prec \cdots \prec \tau_T
\]

Bradley-Terry pairwise ranking loss

\[
\mathcal{L}(\theta) = - \sum_{\tau_i \prec \tau_j} \frac{\exp \sum_{s \in \tau_i} R_{\theta}(s)}{\exp \sum_{s \in \tau_i} R_{\theta}(s) + \exp \sum_{s \in \tau_j} R_{\theta}(s)}
\]
Trajectory-ranked Reward Extrapolation (T-REX)

\[ \tau_1 < \tau_2 < \cdots < \tau_T \]

Minimize cross-entropy loss

\[ \mathcal{L}(\theta) = - \sum_{\tau_i < \tau_j} \frac{\exp \sum_{s \in \tau_i} R_\theta(s)}{\exp \sum_{s \in \tau_i} R_\theta(s) + \exp \sum_{s \in \tau_j} R_\theta(s)} \]
Trajectory-ranked Reward Extrapolation (T-REX)

Given pre-ranked demos, reward learning can be formulated as a standard supervised learning task.

Minimize cross-entropy loss

\[ \mathcal{L}(\theta) = - \sum_{\tau_i < \tau_j} \frac{\exp \sum_{s \in \tau_i} R_\theta(s)}{\exp \sum_{s \in \tau_j} R_\theta(s)} \]
“Autonomous Driving” in Atari

Best demo (Score = 84)  T-REX (Score = 520)

Uses only 12 ranked demonstrations
AI systems can efficiently infer human intent from suboptimal demonstrations.
T-REX only learns a maximum likelihood estimate of the reward function.

\[ R \leftrightarrow \pi \]
Reward Hacking

- Overfit to spurious correlations
- No consideration of alternative hypotheses
\[ P(R|D) \] \[ \pi \]
Bayesian Reward Inference

\[ P(R|D) \propto P(D|R)P(R) \]

Reward Function
Human Data
Bayesian Reward Inference

\[ P(R|D) \propto P(D|R)P(R) \]

Likelihood of the human producing data D when trying to optimize R.

Demonstrator’s action  \( S \)  Alternative actions
Bayesian Reward Inference

\[ P(R|D) \propto P(D|R)P(R) \]

Likelihood of the human producing data D when trying to optimize R.

Requires reasoning about what a reinforcement learner would do when optimizing R...
Idea: Fast Bayesian Inference

Low-Dimensional Latent Space

Preference-Based Likelihood Function

Bayesian Reward Extrapolation (Bayesian REX)

Feature pre-training

How do we train this embedding?

Bayesian Reward Extrapolation (Bayesian REX)

Feature pre-training

Input

Feature Embedding

CNN

Ranked demos

\([\tau_1, \tau_2, \tau_3, \tau_4, \tau_N]\)

Self-supervised task losses

Feature pre-training

Bayesian Reward Extrapolation (Bayesian REX)

1. Variational Autoencoder
2. Temporal Distance
3. Inverse Dynamics
4. Forward Dynamics

+ T-REX ranking loss

Bayesian Reward Extrapolation (Bayesian REX)

Feature pre-training

Input

Feature Embedding

CNN

Ranked demos

$\tau_1$
$\tau_2$
$\tau_3$
$\tau_4$
$\tau_N$

Self-supervised task losses

Bayesian Reward Extrapolation (Bayesian REX)

Feature pre-training

Input

Feature Embedding

Freeze Weights

CNN

Self-supervised task losses

Ranked demos

Bayesian Reward Extrapolation (Bayesian REX)

Feature pre-training

Input $\tau_1, \tau_2, \tau_3, \tau_4, \tau_N$

Ranked demos

Feature Embedding

Freeze Weights

CNN

$\phi(s)$

Self-supervised task losses

Linear Reward Function

$$R_\theta(s) = \theta^T \phi(s)$$

Bayesian Reward Extrapolation (Bayesian REX)

Feature pre-training

Input ➔ CNN ➔ Feature Embedding

Ranked demos ➔ Self-supervised task losses

\[ \Phi_\tau = \sum_{s \in \tau} \phi(s) \]

Embed demonstrations in latent feature space.
Bayesian Reward Extrapolation (Bayesian REX)

Feature pre-training

Input → CNN → Feature Embedding

Ranked demos → Self-supervised task losses

\[ R_\theta(s) = \theta^T \phi(s) \]

\[ R_\theta(\tau) = \sum_{s \in \tau} R_\theta(s) \]

\[ = \theta^T \sum_{s \in \tau} \phi(s) \]

\[ = \theta^T \Phi_\tau \]

Bayesian Reward Extrapolation (Bayesian REX)

Feature pre-training

Input

Feature Embedding

CNN

Ranked demos

Self-supervised task losses

\[ \tau_i < \tau_j \]

\[ R_\theta(\tau_i) < R_\theta(\tau_j) \]

\[ \theta^T \Phi_{\tau_i} < \theta^T \Phi_{\tau_j} \]

Bayesian Reward Extrapolation (Bayesian REX)

Feature pre-training

Input

Feature Embedding

CNN

Ranked demos

Self-supervised task losses

\[ \tau_i < \tau_j \]

\[ R_\theta(\tau_i) < R_\theta(\tau_j) \]

\[ \theta^T \Phi_{\tau_i} < \theta^T \Phi_{\tau_j} \]

Bayesian Reward Extrapolation (Bayesian REX)

**Feature pre-training**

- **Input**
- **CNN**
- **Feature Embedding**

**Ranked demos**

- \( \Phi_{\tau_1} \)
- \( \Phi_{\tau_2} \)
- \( \Phi_{\tau_3} \)
- \( \Phi_{\tau_4} \)
- \( \Phi_{\tau_N} \)

- **MCMC step**

\[
P(\mathcal{D} | R_{\theta}) = \prod_{i < j} \frac{\exp(\theta^T \Phi_{\tau_i})}{\exp(\theta^T \Phi_{\tau_i}) + \exp(\theta^T \Phi_{\tau_j})}
\]

**Sampled weight vector**

**Self-supervised task losses**

Bayesian Reward Extrapolation (Bayesian REX)

Feature pre-training

Input

Feature Embedding

CNN

Ranked demos

Self-supervised task losses

\[ P(D \mid R_{\theta}) = \prod_{i \neq j} \frac{\exp(\theta^T \Phi_{\tau_j})}{\exp(\theta^T \Phi_{\tau_i}) + \exp(\theta^T \Phi_{\tau_j})} \]

Repeat N times to sample from posterior

\[ P(R_{\theta} \mid D) \]
Bayesian Reward Extrapolation (Bayesian REX)

First Bayesian reward inference algorithm to scale to visual control tasks.

Self-supervised task losses

Repeat N times to sample from posterior

\[ P(R_{\theta} \mid D) = \frac{1}{\text{MAP}} \prod_{i \neq j} \exp(\theta^T \Phi_{\tau_i}) + \exp(\theta^T \Phi_{\tau_j}) \]
Bayesian Reward Extrapolation (Bayesian REX)

Feature pre-training

- Generates 10,000 samples of likely reward functions in less than a minute!

It would take 5+ hours for prior Bayesian IRL methods to generate one sample.

Self-supervised task losses

\[ P(R_{\theta} \mid D) \]
Best of 12 demos

Behavioral Cloning

GAIL (Ho and Ermon 2016)

T-REX

Bayesian REX
Utilizing a Reward Function Posterior

• High-confidence performance bounds.

• Detect misaligned rewards.

Shameless plug for my class next Fall: CS 5960/6960 Human-AI Alignment

• Imitation Learning
• Inverse RL
• Shared Autonomy
• RLHF
• AI Safety and Robustness
• Existential Risk of AI (should we be worried?)
• AI Alignment (how do we make sure things work out okay)
• Human-Robot Interaction
• Computational Ethics
Look where we’ve been!

Informed Search: A-Star

Adversarial Search: Alpha-Beta Pruning

Expectimax Search

White to move
Black winning
Look where we’ve been!

MDPs: Value Iteration, Policy Iteration

Reinforcement Learning: Q-Learning, Policy Gradients

Environment

Agent

State: s

Reward: r

Actions: a

DQN

AlphaGo
Look where we’ve been!

Bayes’ Nets

- Burglary
- Earthquake
- Alarm
- John calls
- Mary calls

D-Separation

Variable Elimination

Sampling
Look where we’ve been!

Markov Models

Value of Perfect Information

Hidden Markov Models: Particle Filters

- Umbrella
- Weather
- Forecast = bad

X_1 \rightarrow X_2 \rightarrow X_3

E_1 \rightarrow E_2 \rightarrow E_3
Look where we’ve been!

Behavioral Cloning

Inverse RL

What is the reward?

R(□)=?
R(□)=?
R(□)=?
R(□)=?

RL from Human Feedback

Trajectory-ranked Reward Extrapolation (T-REX)

Pre-ranked demonstrations

Browne et al., "Extracting Action-Latent and Action-Dependent Information from Human Demonstrations via RL from Observations.," ICML 2020
We made it!