Maximum Entropy IRL

Inverse Reinforcement Learning

Inverse Reinforcement Learning (IRL) seeks to infer the underlying reward function R that an expert agent is optimizing, based on observed behavior. Formally, consider a Markov Decision Process (MDP) defined by the tuple (S, A, T, γ, p_0), where:

- S is the finite set of states,
- A is the finite set of actions,
- $T(s' \mid s, a)$ is the transition probability from state s to state s' given action a,
- $\gamma \in [0, 1]$ is the discount factor,
- $p_0(s)$ is the initial state distribution.

In the context of IRL, the reward function $R : S \to \mathbb{R}$ is unknown and needs to be inferred from expert demonstrations $\mathcal{D} = \{\tau_i\}_{i=1}^{N}$, where N is the number of expert trajectories, and each trajectory τ_i is a sequence of state-action pairs $\tau_i = \{(s_1^i, a_1^i), (s_2^i, a_2^i), \ldots, (s_{T_i}^i, a_{T_i}^i)\}$, with T_i being the length of trajectory τ_i.

Feature Expectation Matching

To facilitate computation and generalization, we define a feature mapping $\phi : S \to \mathbb{R}^d$ that assigns a d-dimensional feature vector to each state. We assume that the reward function R is linearly parameterized by these features:

$$R(s) = \theta^\top \phi(s),$$

where $\theta \in \mathbb{R}^d$ is the parameter vector we aim to learn.

The expected discounted accumulated feature vector under a policy π is defined as:

$$\mu(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t) \right],$$

where the expectation is over trajectories starting from the initial state distribution p_0 and following policy π. The value of a policy π under the reward function $R(s)$ can then be expressed as:

$$V^\pi = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \right] = \theta^\top \mu(\pi).$$

This shows that the expected reward depends linearly on the expected feature counts.

Our objective is to find a policy π whose feature expectations match those of the expert:

$$\mu(\pi) = \hat{\mu},$$

where $\hat{\mu}$ is the empirical feature expectation computed from the expert demonstrations:

$$\hat{\mu} = \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{T_i} \gamma^t \phi(s_t^i).$$

Matching feature expectations ensures that the learned policy behaves similarly to the expert in terms of the features that are important for the task.

However, this feature expectation matching condition alone is insufficient to uniquely determine the reward function R, as multiple reward functions can lead to the same feature expectations.

Maximum Entropy Objective

To address this ambiguity, the Principle of Maximum Entropy is employed. This principle selects the distribution over trajectories $p(\tau)$ that maximizes entropy while satisfying the feature expectation constraints. Maximizing entropy ensures that we make the least additional assumptions beyond the observed data, leading to the most unbiased distribution possible. The entropy $H(p)$ of the distribution $p(\tau)$ is defined as:

$$H(p) = -\sum_{\tau} p(\tau) \log p(\tau).$$

The optimization problem can thus be formulated as:

$$\begin{aligned} \max_{p} \quad & H(p) \\ \text{subject to} \quad & \sum_{\tau} p(\tau)\phi(\tau) = \hat{\mu}, \\ & \sum_{\tau} p(\tau) = 1, \\ & p(\tau) \geq 0, \quad \forall \tau, \end{aligned}$$

where $\phi(\tau) = \sum_{t=0}^{T} \gamma^t \phi(s_t)$.

Deriving the Probability Distribution

Using the method of Lagrange multipliers, we introduce multipliers $\theta \in \mathbb{R}^d$ for the feature matching constraints and $\eta \in \mathbb{R}$ for the normalization constraint. The Lagrangian $\mathcal{L}$ is:

$$\mathcal{L} = -\sum_{\tau} p(\tau) \log p(\tau) + \theta^\top \left(\sum_{\tau} p(\tau)\phi(\tau) - \hat{\mu} \right) + \eta \left(\sum_{\tau} p(\tau) - 1 \right).$$

Taking the derivative of $\mathcal{L}$ with respect to $p(\tau)$ and setting it to zero yields:

$$\frac{\partial \mathcal{L}}{\partial p(\tau)} = -\log p(\tau) - 1 + \theta^\top \phi(\tau) + \eta = 0.$$

Solving for $p(\tau)$, we obtain:

$$p(\tau) = \exp(\theta^\top \phi(\tau) + \eta - 1).$$

Applying the normalization condition $\sum_\tau p(\tau) = 1$, we recognize that the term $\eta - 1$ acts as a normalization constant, leading us to define the partition function $Z(\theta)$:

$$Z(\theta) = \sum_\tau \exp(\theta^\top \phi(\tau)).$$

Thus, the final form of the trajectory distribution is:

$$p(\tau \mid \theta) = \frac{\exp(\theta^\top \phi(\tau))}{Z(\theta)}.$$

This distribution is the maximum entropy distribution over trajectories that satisfies the feature expectation constraints.

Gradient of the Log-Likelihood

To find the optimal parameters θ, we maximize the log-likelihood of the expert demonstrations under the model:

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \log p(\tau_i \mid \theta).$$

The gradient of the log-likelihood with respect to θ is:

$$\nabla_\theta \mathcal{L}(\theta) = \hat{\mu} - \mathbb{E}_{p(\tau|\theta)}[\phi(\tau)],$$

where $\mathbb{E}_{p(\tau|\theta)}[\phi(\tau)]$ is the expected feature count under the current model. This gradient points in the direction that increases the likelihood of the expert demonstrations by reducing the discrepancy between the empirical and model feature expectations.

Computing the Expected Feature Counts

Direct computation of $\mathbb{E}_{p(\tau|\theta)}[\phi(\tau)]$ is intractable due to the exponential number of possible trajectories. Instead, we compute the expected state visitation frequencies $D(s)$ under the current policy π, which allows us to express the expected feature counts as:

$$\mathbb{E}_{p(\tau|\theta)}[\phi(\tau)] = \sum_{s \in S} D(s)\phi(s).$$

The state visitation frequency $D(s)$ represents the expected discounted number of times state s is visited under policy π.

State Visitation Frequencies

To compute $D(s)$, we perform a forward-backward algorithm:

1. **Backward Pass**: Compute the partition function $Z(s)$ for each state using dynamic programming, starting from terminal states (where $Z(s) = 1$) and recursively applying:

$$Z(s) = \exp(R(s)) \sum_{a \in A} \sum_{s' \in S} T(s' \mid s, a) Z(s').$$

   This step computes the cumulative desirability of reaching future states from state s, weighted by the exponentiated reward.

2. **Forward Pass**: Using the partition functions, compute the policy $\pi(a \mid s)$:

$$\pi(a \mid s) = \frac{\exp(R(s)) \sum_{s' \in S} T(s' \mid s, a) Z(s')}{Z(s)}.$$

   Then, propagate the state visitation frequencies starting from the initial state distribution $D_0(s) = p_0(s)$:

$$D_{t+1}(s') = \gamma \sum_{s \in S} D_t(s) \sum_{a \in A} \pi(a \mid s) T(s' \mid s, a).$$

   The overall visitation frequency is obtained by summing over time steps:

$$D(s) = \sum_{t=0}^{T_{\max}} D_t(s).$$

   This step computes how frequently each state is visited under the current policy, accounting for the discount factor γ.

High-Level Algorithm Overview

1. **Initialization**: Set initial reward parameters θ_0, learning rate α, convergence threshold ϵ, and maximum iterations K.
2. **Compute Empirical Feature Expectations**: Calculate $\hat{\mu}$ from expert demonstrations.
3. **Optimization Loop** (repeat until convergence or maximum iterations):
   (a) Compute the reward function $R(s) = \theta^\top \phi(s)$.
   (b) Perform a backward pass to compute partition functions $Z(s)$.
   (c) Derive the policy $\pi(a \mid s)$ using the partition functions.
   (d) Perform a forward pass to compute state visitation frequencies $D(s)$.
   (e) Compute the gradient $\nabla_\theta = \hat{\mu} - \sum_{s \in S} D(s)\phi(s)$.
   (f) Update the reward parameters $\theta \leftarrow \theta + \alpha \nabla_\theta$.
4. **Output**: The estimated reward function $R(s) = \theta^\top \phi(s)$.

Detailed Algorithm

Algorithm 1 Maximum Entropy Inverse Reinforcement Learning (MaxEnt IRL)

Initialize: Reward parameters $\theta_0 \in \mathbb{R}^d$, Learning rate α, Convergence threshold ϵ, Maximum iterations K

Input: Expert demonstrations $\mathcal{D}$, Transition probabilities $T(s' \mid s, a)$, Feature mapping $\phi : S \to \mathbb{R}^d$, Discount factor γ, Initial state distribution $p_0(s)$

Compute empirical feature expectations $\hat{\mu}$:

$$\hat{\mu} = \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{t=0}^{T_\tau} \gamma^t \phi(s_t)$$

Set $\theta \leftarrow \theta_0$

for $k = 1$ to K **do**

  Compute reward function:

$$R(s) = \theta^\top \phi(s)$$

  Backward Pass: Compute partition functions $Z(s)$ for all $s \in S$

  1. **Initialize** $Z(s) \leftarrow 1$ for terminal states
  2. **For all other states**, compute recursively:

$$Z(s) = \exp(R(s)) \sum_{a \in A} \sum_{s' \in S} T(s' \mid s, a) Z(s')$$

  Compute policy $\pi(a \mid s)$:

$$\pi(a \mid s) = \frac{\exp(R(s)) \sum_{s' \in S} T(s' \mid s, a) Z(s')}{Z(s)}$$

  Forward Pass: Compute state visitation frequencies $D(s)$

  1. **Initialize** $D_0(s) \leftarrow p_0(s)$ for all $s \in S$
  2. **For each time step** $t = 0$ to $T_{\max}$:

$$D_{t+1}(s') = \gamma \sum_{s \in S} D_t(s) \sum_{a \in A} \pi(a \mid s) T(s' \mid s, a)$$

  3. **Compute** $D(s) = \sum_{t=0}^{T_{\max}} D_t(s)$

  Compute gradient ∇_θ:

$$\nabla_\theta = \hat{\mu} - \sum_{s \in S} D(s) \phi(s)$$

  Update reward parameters:

$$\theta \leftarrow \theta + \alpha \nabla_\theta$$

  if $\|\nabla_\theta\| < \epsilon$ **then**

    Terminate optimization

  end if

end for

Output: Estimated reward function $R(s) = \theta^\top \phi(s)$

Alternative Formulation

The Maximum Entropy Inverse Reinforcement Learning (MaxEnt IRL) approach can be formulated as minimizing the Kullback-Leibler (KL) divergence between a distribution $p(\tau)$ over trajectories and a reference distribution $q(\tau)$ induced by the MDP dynamics alone. This objective can be stated as:

$$\min_p D_{KL}(p \parallel q) \quad \text{subject to} \quad \mathbb{E}_{\tau \sim p}[\phi(\tau)] = \hat{\mu}, \quad \sum_\tau p(\tau) = 1, \quad p(\tau) \geq 0, \quad \forall \tau.$$

where $D_{KL}(p \parallel q) = \sum_\tau p(\tau)(\log p(\tau) - \log q(\tau))$ measures the divergence from p to q.

According to the MaxEnt principle, the solution to this constrained optimization problem yields the distribution:

$$p(\tau \mid \theta) = \frac{q(\tau) e^{\theta^\top \phi(\tau)}}{Z(\theta)},$$

where $Z(\theta) = \sum_\tau q(\tau) e^{\theta^\top \phi(\tau)}$ is the partition function, ensuring that $p(\tau)$ sums to one. Thus, MaxEnt IRL is equivalent to taking $q(\tau) = 1$, i.e., a uniform baseline policy.