Inverse Reinforcement Learning

Professor Vwani Roychowdhury

University of California, Los Angeles

May 3, 2018

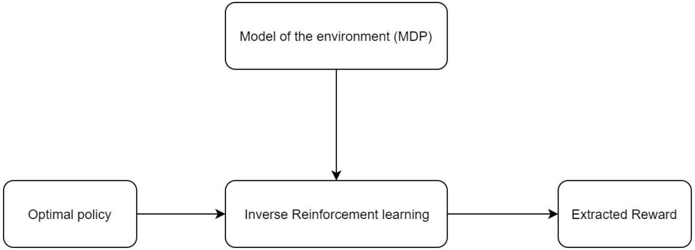
Overview

Introduction

Inverse Reinforcement learning (IRL) algorithms
- IRL Problem setup
- IRL algorithms for finite state spaces
  - LP Feasibility Problem
  - Tackling degeneracy of LP feasibility problem

What is Inverse Reinforcement learning?

Inverse Reinforcement learning is the task of extracting the reward function from the optimal policy.


Model of the environment (MDP)
Optimal policy
Inverse Reinforcement learning
Extracted Reward

Motivation for Inverse Reinforcement learning

- An agent might not know the reward function a priori. Therefore, Reinforcement learning might not be applicable for optimal policy learning.
- Reward function provides a much more parsimonious description of behavior. Therefore, learning the reward function of the agent helps to design more robust agents.
- Reward function, rather than the policy, is the most succinct, robust, and transferable definition of the task.

Applications of Inverse Reinforcement learning

- Aerial imagery based navigation (Ratliff et al, ICML 2006)

- Parking lot navigation (Abbeel et al, IROS 2008)

IRL Problem setup

In a IRL problem, we are given the following quantities:

- $\mathcal{S}$ is a set of **states**
- $\mathcal{A}$ is a set of **actions**
- $\mathcal{P}^a_{ss'}$ is a set of **transition probabilities**, where $\mathcal{P}^a_{ss'}$ is the probability of transitioning from state $s \in \mathcal{S}$ to state $s' \in \mathcal{S}$ after taking action $a \in \mathcal{A}$
  - $\mathcal{P}^a_{ss'} = \mathbb{P}(s_{t+1} = s' | s_t = s, a_t = a)$
- **Optimal deterministic policy** π^*

The goal of the IRL problem is to find the **set of possible reward functions** $\mathcal{R}^a_{ss'}$ **such that** π^* **is an optimal policy in the MDP**.

Some simplifications and notations

In order to simplify the derivation of the IRL algorithm, we will assume that the reward functions are independent of action. To be specific, the immediate reward at a state is same irrespective of the action that lead to this state. We will denote the reward at state s by $R(s)$. For notational simplicity, we define a reward vector $\mathbf{R} \in \mathbb{R}^{|\mathcal{S}|}$ in the following manner:

$$\mathbf{R}(i) = R(s_i)$$

Similarly we represent the transition probabilities as matrices $\mathbf{P}_a \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$, where

$$\mathbf{P}_a(i,j) = \mathbb{P}(s_{t+1} = j | s_t = i, a_t = a)$$

IRL in finite state spaces

In the next few slides, we give a simple characterization of the set of all reward vectors $\mathbf{R}$ for which a given policy is optimal. We then show that the set contains many degenerate solutions and propose a simple heuristic for tackling this degeneracy, resulting in a linear programming (LP) solution to the IRL problem.

Characterization of the solution set

The following theorem characterizes the set of solution reward vectors $\mathbf{R}$:

Theorem

Let a finite state space $\mathcal{S}$, a set of actions $\mathcal{A} = (a_1, a_2, \cdots, a_k)$, transition probability matrices $\{\mathbf{P}_a\}_{a=1}^{k}$, and discount factor $\gamma \in (0, 1)$ be given. Then the policy π given by $\pi(s) \equiv a_1$ is optimal if and only if, for all $a = a_2, \cdots, a_k$, the reward $\mathbf{R}$ satisfies equation 1

$$(\mathbf{P}_{a_1} - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_{a_1})^{-1}\mathbf{R} \succeq 0 \tag{1}$$

We can use the constraints given by equation 1 to find a feasible reward vector $\mathbf{R}$. The problem of finding a feasible $\mathbf{R}$ can be posed as a Linear Feasibility Problem.

LP Feasibility Problem

$$\begin{aligned} \underset{\mathbf{R}}{\text{minimize}} \quad & 0 \\ \text{subject to} \quad & (\mathbf{P}_{a_1} - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_{a_1})^{-1}\mathbf{R} \succeq 0, \;\; \forall a \in \mathcal{A} \setminus a_1 \\ & |\mathbf{R}_i| \leq R_{max}, \;\; i = 1, 2, \cdots, |\mathcal{S}| \end{aligned} \tag{2}$$

Although the LP feasibility problem, given by equation 2, will return the set of reward vectors **R** but the solution set suffers from degeneracy.

Degenerate solutions of the LP feasibility problem

The solution set of the LP feasibility problem suffers from two major problems:

- $\mathbf{R} = 0$ is always a solution to the LP feasibility problem
- For most MDP's, there are many choices of $\mathbf{R}$ that will satisfy the constraints of the LP feasibility problem. For example, any constant reward vector $\mathbf{R}$ will be a solution to the LP feasibility problem

In the next few slides, we will describe a natural criterion to address the problem of degeneracy.

Heuristics for tackling degeneracy

There are many heuristics to tackle the issue of degeneracy but we will consider the following one:

- We want to choose **R** such that it makes π optimal (by satisfying the constraints of the LP feasibility problem) and moreover we want to favor reward vectors **R** that make any single step-deviation from optimal policy π^* as costly as possible.
- Among all the **R** satisfying equation 2, we want to choose the **R** that maximizes the following expression

$$\sum_{s \in \mathcal{S}} \left(Q^{\pi^*}(s, a_1) - \max_{a \in \mathcal{A} \setminus a_1} Q^{\pi^*}(s, a) \right) \tag{3}$$

Equation 3 expresses the sum of the differences between the quality of the optimal action and the quality of the next best action. The intuitive meaning of equation 3 is that deviating from the optimal policy should reduce the total reward as much as possible.

Rewriting equation 3

Equation 3 can be rewritten in terms of the reward vector $\mathbf{R}$ and the probability transition matrices $\mathbf{P}_a$ and $\mathbf{P}_{a_1}$

$$\sum_{s\in\mathcal{S}}\left(Q^{\pi}(s,a_1)-\max_{a\in\mathcal{A}\setminus a_1}Q^{\pi}(s,a)\right)$$
$$=\sum_{i=1}^{|\mathcal{S}|}\min_{a\in\mathcal{A}\setminus a_1}[(\mathbf{P}_{a_1}(i)-\mathbf{P}_a(i))(\mathbf{I}-\gamma\mathbf{P}_{a_1})^{-1}\mathbf{R}]$$

In the above expression, $\mathbf{P}_a(i)$ denotes the i^{th} row of $\mathbf{P}_a$. Using the above expression, we can formulate an optimization problem to find non-trivial reward vectors $\mathbf{R}$.

Optimization problem

$$\begin{aligned}
\underset{\mathbf{R}}{\text{maximize}} \quad & \textstyle\sum_{i=1}^{|\mathcal{S}|} \min_{a \in \mathcal{A} \setminus a_1} [(\mathbf{P}_{a_1}(i) - \mathbf{P}_a(i))(\mathbf{I} - \gamma \mathbf{P}_{a_1})^{-1}\mathbf{R}] \\
\text{subject to} \quad & (\mathbf{P}_{a_1} - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_{a_1})^{-1}\mathbf{R} \succeq 0, \;\; \forall a \in \mathcal{A} \setminus a_1 \\
& |\mathbf{R}_i| \leq R_{max}, \;\; i = 1, 2, \cdots, |\mathcal{S}|
\end{aligned} \tag{4}$$

By solving the optimization problem, given by equation 4, we can find non-trivial reward vectors **R**. However, if we want to favor simpler reward vectors then we can add ℓ_1 regularization. In this context, reward vectors with non-zero reward values at very few states are considered to be simpler.

Optimization problem with ℓ_1 regularization

$$\begin{array}{ll}\underset{\mathbf{R}}{\text{maximize}} & \sum_{i=1}^{|\mathcal{S}|} \min_{a \in \mathcal{A} \setminus a_1} [(\mathbf{P}_{a_1}(i) - \mathbf{P}_a(i))(\mathbf{I} - \gamma \mathbf{P}_{a_1})^{-1}\mathbf{R}] - \lambda \|\mathbf{R}\|_1 \\ \text{subject to} & (\mathbf{P}_{a_1} - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_{a_1})^{-1}\mathbf{R} \succeq 0, \ \ \forall a \in \mathcal{A} \setminus a_1 \\ & |\mathbf{R}_i| \leq R_{max}, \ \ i = 1, 2, \cdots, |\mathcal{S}| \end{array} \tag{5}$$

In the optimization problem, given by equation 5, λ is an adjustable penalty coefficient that promotes simpler reward vectors **R**. The optimization problem looks very complex, but it can be formulated as a Linear Program (LP).

Formulation as a LP

We can convert the optimization problem, given by equation 5, into a Linear program (LP) by introducing extra variables.

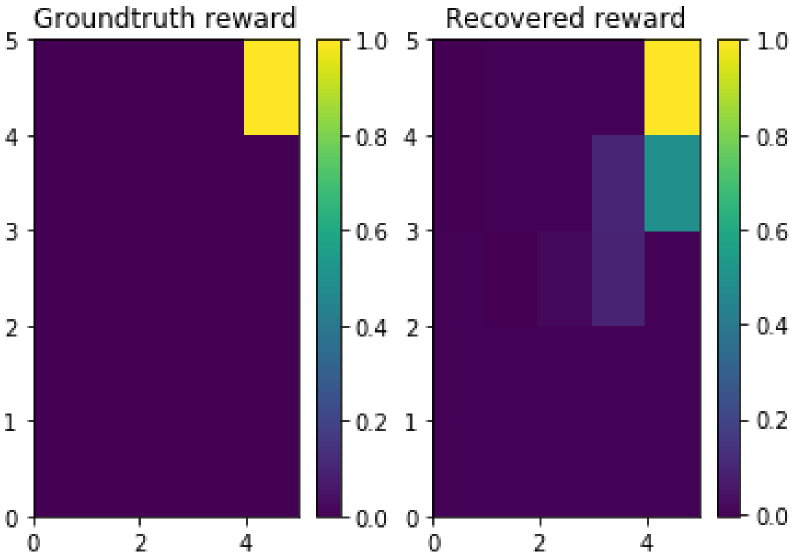
$$\begin{array}{ll} \underset{\mathbf{R}, t_i, u_i}{\text{maximize}} & \sum_{i=1}^{|\mathcal{S}|} (t_i - \lambda u_i) \\ \text{subject to} & [(\mathbf{P}_{a_1}(i) - \mathbf{P}_a(i))(\mathbf{I} - \gamma \mathbf{P}_{a_1})^{-1} \mathbf{R}] \geq t_i, \ \ \forall a \in \mathcal{A} \setminus a_1, \forall i \\ & (\mathbf{P}_{a_1} - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_{a_1})^{-1} \mathbf{R} \succeq 0, \ \ \forall a \in \mathcal{A} \setminus a_1 \\ & -\mathbf{u} \preceq \mathbf{R} \preceq \mathbf{u} \\ & |\mathbf{R}_i| \leq R_{max}, \ \ i = 1, 2, \cdots, |\mathcal{S}| \end{array} \tag{6}$$

In the Linear program, given by equation 6, t_i's and u_i's are the extra variables. We can use the standard LP solvers to solve for the reward vector $\mathbf{R}$.

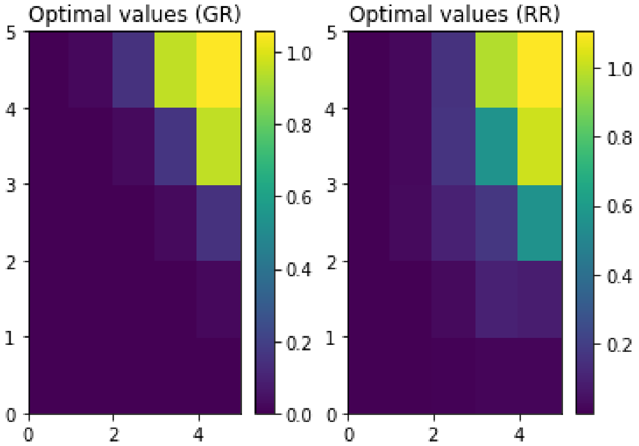
Example: Agent navigating in a 2-D environment

Let's consider an example of an agent navigating in a 2-D square grid of size 5×5. There is some ground-truth reward associated with the environment, which we use to compute the optimal policy of the agent using value iteration algorithm (Reinforcement learning). Then we use the optimal policy of the agent along with the model of the environment to extract the reward function using the LP formulation given by equation 6 (Inverse Reinforcement learning).

Heat map of the ground truth and recovered reward



Heat map of the optimal state values



The heat map on the left displays the optimal state values computed using the ground truth reward while the heat map on the right displays the optimal state values computed using the recovered reward.

Similarity between the optimal state values and the local reward values

From the previous two slides, it can be observed that the optimal value of a state and the local reward at that state is similar in magnitude for this example. We used a discount factor of 0.2 in this example. Due to the small discount factor, the expected discounted future reward and immediate reward of a state is similar in magnitude.