AE598: Project Report
Inverse Reinforcement Learning over MDPS.

Vijeth Hebbar

May 12, 2022

1 Introduction

At the risk of over simplification, Reinforcement Learning (RL) can be viewed broadly as a method to arrive at the optimal strategy to complete a given task with higher rewards being given for 'good' attempts and low rewards for 'bad' ones. Specifically , 'good quality' actions taken by the learning agents are rewarded well and 'bad quality' actions are penalized. Given the history of rewards obtained for each action[1] the agent then learns to perform actions that lead to larger overall rewards in subsequent attempts of solving the same problem. Inverse Reinforcement Learning (IRL) on the other hand seeks to learn the rewards that an agent must be receiving for every action given the optimal strategy for a particular problem. But this begs the question of why one would wish to learn the rewards if we are already have the optimal strategy.

The first reason to learn the reward functions is to inherently understand the intent of the decision making agent [1]. Learning the reward function for the agent can then allow us to model their behaviour in other situations of a similar context. A simple example is that of recommender systems. An individual (who is an expert on their own preferences) picks a series of songs over Spotify, which is then used by the app to form a preference model of the individual. This model can then be used to design personalized song recommendations catered to the individual. A closely related notion is the theory of revealed preferences in the economics literature where the economic choices made by an individual are used to learn about their utilities [2]. Intent learning also plays a very important role in human-robot interaction, where the robot can infer goals that the human is trying to achieve and adapt its own strategy to achieve the joint goals better.

The second reason to use knowledge of optimal strategies to obtain reward functions is in the context of *apprenticeship learning* i.e. using an expert's knowledge to solve a problem by imitating their actions. While one can simply learn the strategy of the expert at face value, reward functions are arguably a more succinct and robust representation of a expert's goals. Indeed under a change of dynamics, the knowledge of rewards will allow the agent to re-compute their own optimal strategy, while blindly imitating the actions of the agent (performed under different dynamics) may not likely produce optimal rewards. The primary domain of interest for apprenticeship learning based IRL is robotics [3, 4].

While the problem of learning rewards from actions has been a long standing one in economics (as indicated above) and control literature[2], Ng and Russell [6] were among the first to study the problem of IRL in an MDP setting. In their work, they begin by formalizing the IRL problem over a finite-state MDP and obtain sufficient and necessary conditions that the reward function should satisfy for a given optimal policy. They

[1]The reward for the same action may be different at different states for the agent.

[2]As an example, the problem of learning the quadratic cost function for a linear system from the control history and state trajectory was solved recently in [5]

then consider the case of infinite state MDPs, as well as the problem of IRL when trajectory realizations of the optimal policy are provided rather than the policy themselves. In this latter portion of their work they restrict the search for reward functions over linear approximations using finite basis functions. Many extensions have been proposed that do away with such approximations and instead look at distributions over the space of reward functions [7, 8]. A larger survey of IRL problems over MDPs is contained in [9].

In this review, we will begin by taking a close look at the IRL problem over finite state MDPs broadly following the work in [6] in Section 3. In doing so, we will construct a simple example MDP problem (Section 3.3) for which we will learn the reward function given an optimal policy. We will then look more closely at the IRL problem where trajectories generated by the optimal policy are given rather than the policy itself in Section 4. We consider an example(Section 4.4), wherein an agent observes an expert moving from one point to another on a grid with stochastic dynamics. The agent then learns the reward function for the expert using the latter's sampled trajectories. The approach we employ to do so is heuristic and easy to implement but lacks performance guarantees[6].

2 Preliminaries and Notation

As a first step, let us develop a framework to solve the problem of Inverse Reinforcement Learning in finite state space. We denote an MDP as the tuple $\{\mathcal{S}, \mathcal{A}, P, \gamma, R\}$ where

1. $\mathcal{S}$ is the set of N states. We denote the states simply as $\{1, 2, \dots, N\}$.
2. $\mathcal{A} = \{a_1, \dots, a_k\}$ is the set of k actions available at every state.
3. $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the probability transition function with $P(s, a, s')$ denoting the probability of transitioning from state s to s' on taking action a.
4. $\gamma \in [0, 1)$ is the discount factor for rewards.
5. And $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function for the MDP.

First we make

Assumption 1 *We will assume that the reward achieved at any state in independent of the action. So going forward we will denote rewards as $R(s)$ instead of $R(s, a)$.*

This assumption is not particularly restrictive as will be shown later, and assumption itself allows for easier exposition.

For every policy $\pi : \mathcal{S} \to \mathcal{A}$ we can define the value function at any state s_0 as

$$V^\pi(s_0) = \mathbb{E}[R(s_0) + \gamma R(s_1) + \dots | \pi]$$

where $(s_0 s_1 s_2)$ is the sequence of states the MDP attains under the execution of policy π. This can be written concisely as

$$V^\pi(s_0) = R(s_0) + \gamma \sum_{s' \in \mathcal{S}} P(s_0, \pi(s_0), s') V^\pi(s'). \tag{1}$$

We also define the Q-function as

$$Q^\pi(s_0, a) = R(s_0) + \gamma \sum_{s' \in \mathcal{S}} P(s_0, a, s') V^\pi(s'). \tag{2}$$

Note that in addition to the dependence of Q on action a, it also depends on the policy π as the value function $V^\pi(\cdot)$ depends on π. We will state without proof Bellman's principal of optimality as it applies to our MDP.

Theorem 2.1 *π is an optimal policy for our MDP, if and only if*

$$\pi(s) = \arg\max_{a \in \mathcal{A}} Q^\pi(s,a) \quad \forall s \in \mathcal{S}. \tag{3}$$

Before we move on to formulating the IRL problem, we present some additional notation that will come in handy later. Since we are considering finite state spaces we can denote the value function and the reward function as N dimensional vectors. We define vectors $\mathbf{V}^\pi$ and $\mathbf{R}$ whose i^{th} elements are $V^\pi(i)$ and $R(i)$ respectively. Extending this idea, we can represent the probability transition function as a set of matrices $\{\mathbf{P}_a\}_{a\in\mathcal{A}}$ where

$$[\mathbf{P}_a]_{ij} = P(i,a,j).$$

Finally, we use the notation $\succ$ and $\succeq$ to denote element-wise vector inequalities. Specifically, for two vectors $\mathbf{X} = [x_1\ x_2 \ldots x_N]^T$ and $\mathbf{Y} = [y_1\ y_2 \ldots y_N]^T$, $\mathbf{X} \succeq \mathbf{Y}$ if and only if $x_i \geq y_i$ for every i.

3 Inverse Reinforcement Learning from Optimal Policy

Let π^* denote the optimal policy for some unknown reward vector $\mathbf{R}^*$. Now the inverse reinforcement learning problem can be stated informally as

"obtain an estimate $\hat{\mathbf{R}}$ of the reward vector that generated the optimal policy π^* given the MDP parameters $(\mathcal{S}, \mathcal{A}, P, \gamma)$."

Before we attempt to analytically tackle this problem, we make the following

Assumption 2

$$\pi^*(s) = a_1 \quad \forall s \in \mathcal{S}.$$

Note that this assumption is WLOG as this condition can indeed be achieved for an arbitrary policy by interchanging the index of actions at every state such that the optimal action is indeed a_1.

3.1 Sufficient and Necessary Conditions for the Solution Set.

Let us now try to characterize the set of reward functions that 'satisfy' a certain optimal policy. We present the following

Theorem 3.1 *The policy $\pi^*(s) \equiv a_1$ is optimal for the reward vector* $\mathbf{R}$ ***if and only if***

$$(\mathbf{P}_{a_1} - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_{a_1})^{-1}\mathbf{R} \succeq \mathbf{0} \quad \forall a \in \mathcal{A} \tag{4}$$

where $\mathbf{I}$ *denotes the identity matrix.*

To prove this theorem we will employ without proof the following result about convergence of the Neumann series.

Lemma 3.1 *$(I - M)$ is invertible if $\rho(M) < 1$ where ρ denotes the spectral radius of a matrix and the inverse is then given by the convergent (in matrix norm) series*

$$\sum_{i=0}^{\infty} M^i$$

.

Proof of Theorem 3.1 Using the vector notations introduced earlier we first rewrite (1) for π^* as

$$\begin{aligned} &\mathbf{V}^{\pi^*} = \mathbf{R} + \gamma \mathbf{P}_{a_1} \mathbf{V}^{\pi^*} \\ \implies &\mathbf{V}^{\pi^*} = (\mathbf{I} - \gamma \mathbf{P}_{a_1})^{-1} \mathbf{R} \end{aligned} \tag{5}$$

where the inevitability of $(\mathbf{I} - \gamma \mathbf{P}_{a_1})$ is guaranteed by Lemme 3.1 and the fact that $\rho(\gamma \mathbf{P}_a) = |\gamma| \rho(\mathbf{P}_a) = \gamma < 1$ because the largest eigenvalue of a stochastic matrix is 1. Then for π^* to be the optimal policy using the (3) we obtain the fact that

$$\begin{aligned} &Q^{\pi^*}(s, a_1) \geq Q^{\pi^*}(s, a) \quad \forall s \in \mathcal{S}, \forall a \in \mathcal{A} \\ \iff &\sum_{s' \in \mathcal{S}} P(s_0, a_1, s') V^{\pi}(s') \geq \sum_{s' \in \mathcal{S}} P(s_0, a, s') V^{\pi}(s') \quad \forall s \in \mathcal{S}, \forall a \in \mathcal{A} \quad \text{(Using (2).)} \\ \iff &\mathbf{P}_{a_1} \mathbf{V}^{\pi^*} \succeq \mathbf{P}_a \mathbf{V}^{\pi^*} \quad \forall a \in \mathcal{A} \\ \iff &(\mathbf{P}_{a_1} - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_{a_1})^{-1} \mathbf{R} \succeq \mathbf{0} \quad \forall a \in \mathcal{A} \end{aligned}$$

where the final step above uses (5) to substitute the value of $\mathbf{V}^{\pi^*}$. ■

Note that condition (4) obtained above is a sufficient and necessary condition and thus completely characterizes the solution set of the IRL problem. But this immediately presents the problem that the solution of (4) is not unique. Indeed any constant vector of rewards (and trivially, the zero vector) is admissible to the solution set. This can be informally understood by nothing that if the rewards at every state were the same, the total reward collected at the end of any finite number of steps is always the same irrespective of the path taken and so every policy is optimal. So a constant reward vector must always be admissible in the IRL solution set for any optimal policy. To see this formally we again invoke Lemma 3.1 to rewrite (4) as

$$(\mathbf{P}_{a_1} - \mathbf{P}_a) \sum_{i=0}^{\infty} \gamma^i \mathbf{P}_{a_1}^i \mathbf{R} \succeq \mathbf{0}.$$

Taking $\mathbf{R} = r\mathbf{1}$, where $\mathbf{1}$ is the vector of ones, we have

$$(\mathbf{P}_{a_1} - \mathbf{P}_a) \sum_{i=0}^{\infty} \gamma^i \mathbf{P}_{a_1}^i r\mathbf{1} = (\mathbf{P}_{a_1} - \mathbf{P}_a)\mathbf{1} \frac{r}{1-\gamma} = (\mathbf{1} - \mathbf{1}) \frac{r}{1-\gamma} = \mathbf{0}$$

where $\mathbf{0}$ is the vector of zeros. Above we used the fact that $\mathbf{1}$ is an eigenvector of an stochastic matrix with eigenvalue 1.

In all this we have shown that the inverse reinforcement problem as we stated above is ill-posed and has a non-unique solution. This indicates that to obtain a reward vector we may need to employ some heuristics to pick one of the reward vectors from the admissible set satisfying (4). One work around would be to replace the inequality in 4 with a strict inequality. Here we can state the corollary.

Corollary 3.1 *The policy $\pi^*(s) \equiv a_1$ is the **unique** optimal policy for the reward vector $\mathbf{R}$ **if and only if***

$$(\mathbf{P}_{a_1} - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_{a_1})^{-1} \mathbf{R} \succ \mathbf{0} \quad \forall a \in \mathcal{A} \tag{6}$$

where $\mathbf{I}$ denotes the identity matrix.

But this again does not guarantee uniqueness of the reward vector solution as for every solution $\mathbf{R}$ to (6), $a\mathbf{R} + b\mathbf{1}$ is also a solution for all $a > 0, b$.

As a solution to this ill-posedness we will provide some heuristics that allows us to pick a unique reward vector amongst this solution set stemming from (4) in the following section. But before we go there, we end this section by commenting on the ramifications of Assumption 1. Suppose instead that we considered reward function of the form $R(s,a)$ and defined the set of reward vectors $\{\mathbf{R}_{a_1}, \mathbf{R}_{a_2}, \dots, \mathbf{R}_{a_k}\}$, where $\mathbf{R}_{a_1}$ is an N-dimensional vector containing the rewards obtained at each state on taking action a_1. Then much the same way we obtained Theorem 3.1, we can obtain

Corollary 3.2 *The policy $\pi^*(s) \equiv a_1$ is optimal for the set of reward vectors $\{\mathbf{R}_{a_1}, \mathbf{R}_{a_2}, \dots, \mathbf{R}_{a_k}\}$* ***if and only if***

$$(\mathbf{R}_{a_1} - \mathbf{R}_a) + (\mathbf{P}_{a_1} - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_{a_1})^{-1}\mathbf{R}_{a_1} \succeq \mathbf{0} \quad \forall a \in \mathcal{A} \tag{7}$$

where $\mathbf{I}$ denotes the identity matrix.

For a solution to (7), we can first take any reward vector that solves (4) and set it as the value of $\mathbf{R}_{a_1}$. Then $\mathbf{R}_a$ for any $a \in \mathcal{A} \setminus \{a_1\}$ can be chosen satisfying $\mathbf{R}_{a_1} \succeq \mathbf{R}_a$. Here we are trying to obtain rewards for actions that do not form a part of optimal policy. Realistically speaking, we cannot know anything about the rewards of actions that are never played (apart from maybe the fact that they must be dominated by the rewards from the corresponding optimal actions). So we see that we can do no better by doing away with Assumption 1.

3.2 Heuristic Method to Obtain Reward Vector

Let us consider the set of reward vectors that comprise solutions to (4). Clearly some of these rewards would be more 'meaningful' than others (say, compared the to zero vector, for instance). To pick such a 'meaningful' vector we employ heuristics.

1. We may want to pick the reward vector such that any other policy is strongly suboptimal compared to the given optimal policy. Heuristically this could be done by picking the $\mathbf{R}$ such that

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

is maximized. This could be rewritten as

$$\max \sum_{1}^{N} \min_{a \in \mathcal{A} \setminus a_1} (\mathbf{P}_{a_1}(i) - \mathbf{P}_a(i)) \mathbf{V}^{\pi^*} \tag{8}$$

where $\mathbf{P}_a(i)$ denotes the i^{th} row of $\mathbf{P}_a$.

2. The second heuristic we may employ is to want a 'simple' enough reward. We may model a 'simple' reward as the one having a small overall magnitude and can look for rewards with a small $||\mathbf{R}||_1$. We can do so by adding a $-\lambda||\mathbf{R}||$ regularization term to the maximization in (8). But here we must be careful as we know that $\mathbf{R} = 0$ is always a solution (4). So taking too large a value of λ will force a solution of $\mathbf{R} = \mathbf{0}$.

We can put together the two heuristics and provide the following optimization problem for obtaining the rewards vector.

$$\max_{\mathbf{R}} \quad \sum_{1}^{N} \min_{a \in \mathcal{A} \setminus a_1} \left\{ (\mathbf{P}_{a_1} - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_{a_1})^{-1} \mathbf{R} \right\} - \lambda ||\mathbf{R}||_1 \tag{9}$$

$$S.T \quad (\mathbf{P}_{a_1} - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_{a_1})^{-1} \mathbf{R} \quad \forall a \in \mathcal{A} \setminus a_1 \tag{10}$$

$$|R_i| \le R_{max} \quad \forall i \in \{1, \dots, N\} \tag{11}$$

where the first term in the objective is obtained by substituting (5) in (8) and the last line of constraints is enforces compactness of the constraint set. The summands in the first term in the objective are point-wise minimums of linear functions, and thus, concave. The concavity of the second term is apparent from the triangle inequality property of norms. Noting that summation preserves concavity, it is easy to see that the objective function is concave. Moreover the constraints sets are compact and convex. The optimization problem can be easily solved using any convex optimization solver[3].

3.3 Simple Example

Let us apply the the theory developed thus far to a very simple problem. Consider the MDP illustrated in Figure 1.

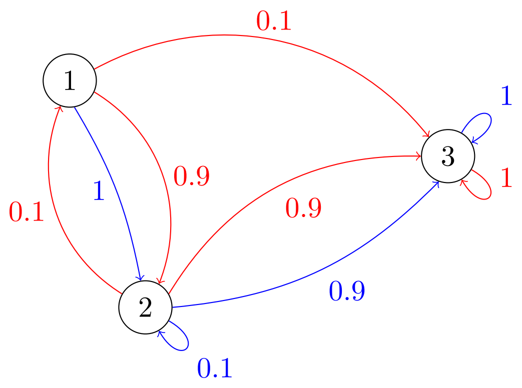


Figure 1: **A simple** 3 **state,** 2 **action MDP.** The true reward vector $\mathbf{R}^*$ is $[-1, -1, 0]$. We pick $\gamma = 0.9$.

The state and action space of the problem above is given by

$$\mathcal{S} = \{1, 2, 3\}, \quad \mathcal{A} = \{R, B\}.$$

We can define the state transition function using the matrices

$$\mathbf{P}_B = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0.1 & 0.9 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{P}_R = \begin{bmatrix} 0 & 0.9 & 0.1 \\ 0.1 & 0 & 0.9 \\ 0 & 0 & 1 \end{bmatrix}.$$

And let us suppose the true rewards at each state are given by $R(1) = R(2) = -1$ and $R(3) = 0$. It can be easily shown than for $\gamma = 0.9$ the optimal policy is given by,

$$\pi^*(1) = R, \pi^*(2) = B, \pi^*(3) = R.^4$$

Let us now set out to obtain the reward function from this optimal policy by optimizing (9) subject to (10) and (11). But before we do so to fit in with the modelling in our work thus far, we need to re-order the actions in the optimal policy to make them all have the same index. In the context of this example, this means that we need to interchange the roles of the R and B actions at state 2 to make action R optimal at all states. This amounts to replacing the second row in $\mathbf{P}_R$ from $\mathbf{P}_B$ giving us

$$\mathbf{P}'_B = \begin{bmatrix} 0 & 1 & 0 \\ 0.1 & 0 & 0.9 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{P}'_R = \begin{bmatrix} 0 & 0.9 & 0.1 \\ 0 & 0.1 & 0.9 \\ 0 & 0 & 1 \end{bmatrix}.$$

[3]This is an erratum in [6], where they claim that this optimization is a linear program.

[4]Clearly, both actions are optimal. We pick R WLOG.

Using (4) we can obtain the necessary and sufficient condition our reward vector $\mathbf{R} = [r_1\ r_2\ r_3]^T$ must satisfy as

$$(\mathbf{P}'_{R} - \mathbf{P}'_{B})(\mathbf{I} - \gamma \mathbf{P}'_{R})^{-1}\mathbf{R} \succeq \mathbf{0}.$$

This gives us the following constraint equations:

$$\begin{aligned} 0.11(r_3 - r_2) &\geq 0 \\ 0.07r_2 + 0.03r_3 - 0.1r_1 &\geq 0 \end{aligned}$$

As a quick sanity check we can check that the true reward function does indeed satisfy these constraints. We now write out the optimization problem specified as (9-11) for our problem as,

$$\begin{aligned} \max \quad & 0.11(r_3 - r_2) + 0.07r_2 + 0.03r_3 - 0.1r_1 - \lambda(|r_1| + |r_2| + |r_3|) \\ S.T \quad & 0 \leq r_3 - r_2 \\ & 0 \leq 0.7r_2 + 0.3r_3 - r_1 \\ & |r_i| \leq 1. \end{aligned}$$

This was solved using CVXPY, a basic convex optimization package in Python, to yield a solution of

$$\mathbf{R} = [0, 0, 0.38]^T$$

for a λ value of 0.13. Larger values of λ return the zero reward vector.

Unsurprisingly the heuristic optimization approach was not effective in obtaining the the true reward function, but this is purely because the true reward $\mathbf{R}^*$ did not in-fact agree with our second heuristic that the reward should have a small magnitude. But nonetheless, our simple example illustrates that despite not being able to obtain the true reward we picked out the qualitative features of the reward that capture the intent of the expert agent. The reward vector we picked was characteristic of the problem of reaching the end goal in minimum expected time. The reward vector we obtained from the optimization had the same feature of 'get to state 3 in minimum time to get the most reward($\because \gamma < 1$)'. Moreover, it can be easily seen that the optimal policy from this and the true reward are identical.

3.4 IRL problem for Infinite State Space MDPs.

Here we will not dive deep into the details of an extension of the solution approach above into infinite state space MDPs, but will address the primary challenges involved in doing so. We also highlight some simple heuristics to address these challenges. We will now do away the vector notation for $\mathbf{V}^\pi, \mathbf{R}$ and $\mathbf{P}_a$ that we introduced earlier and consider a MDP where the state space $\mathcal{S}$ is an infinite (possibly, uncountable) set. For convenience, let us assume $\mathcal{S} = \mathbb{R}^n$. Again our problem statement is -

"obtain an estimate $R(s)$ of the reward function that generated the optimal policy π^* given the MDP parameters $(\mathcal{S}, \mathcal{A}, P, \gamma)$."

The true reward function is now a mapping from $\mathbb{R}^n \to \mathbb{R}$, and so any approach similar to the one discussed for finite state IRL problem would involve optimizing over the space of functions. To avoid the difficulties of optimizing in infinite dimensional spaces we will perform the standard trick of restricting search to a finite dimensional subspace of the function space. This will be the first heuristic we apply.

Let us consider a finite set of known, bounded basis functions $\{\phi_1, \ldots, \phi_d\}$ mapping from $\mathcal{S} \to \mathbb{R}$. We now search for an reward function that can be approximated as

$$R(s) = \sum_{1}^{d} \alpha_i \phi_i(s). \tag{12}$$

Thus we have reduced our search space for the reward function from the function space to the finite set of parameters $\{\alpha_i\}_{i=1}^d$. Let us now define the 'basis' value function V_i^π for policy π as the value function obtained when the rewards function is simply $R(s) = \phi_i(s)$. Then for any reward function represented as (12), by linearity of expectation, the value of any policy π is given by

$$V^\pi = \sum_1^d \alpha_i V_i^\pi. \tag{13}$$

We state without claim a sufficient and necessary condition for the optimality of a policy π as

$$\mathbb{E}_{s' \sim P(s,a_1,\cdot)}[V^\pi(s')] \geq \mathbb{E}_{s' \sim P(s,a,\cdot)}[V^\pi(s')] \tag{14}$$

for every $s \in \mathcal{S}$ and every $a \in \mathcal{A} \setminus a_1$. The process to obtain this condition is similar to the one outlined in Proof of Theorem 3.1.

There are multiple challenges here in evaluating (14).

C-I: Firstly, because we want (14) to hold for every state s, we need to check infinitely many constraints. A heuristic would be to check (14) at only a finite set $\mathcal{S}_0$ of s states. Another possible heuristic would be to consider a distribution $\mathcal{D}$ over the set of states $\mathcal{S}$. Then we evaluate the single constraint,

$$\mathbb{E}_{s \sim \mathcal{D}}\Big[\mathbb{E}_{s' \sim P(s,a_1,\cdot)}[V^\pi(s')]\Big] \geq \mathbb{E}_{s \sim \mathcal{D}}\Big[\mathbb{E}_{s' \sim P(s,a,\cdot)}[V^\pi(s')]\Big]. \tag{15}$$

A distribution $\mathcal{D}$ that places larger weights on some of the states than others encourages the search for a reward function such that these 'high weight' states satisfy (14).

C-II: Secondly, to evaluate the expectations in (14) we need to have functional forms of $V^\pi(s)$. Since we will evaluate this function as a linear approximation using (13), we in turn need to have a functional form for each of the V_i^π's. This is simply a problem of evaluating the value of a policy, which is made complicated in infinite state space. One way to get around this is to approximate the value function, say by using Monte Carlo simulation runs using policy π and averaging the discounted rewards collected over multiple runs.

C-III: The third problem is that our basis functions may not be chosen well and so no reward function that results in policy π^* may even be representable as a linear combination of these functions. In such a case, our heuristic would be to allow some constraint violation by paying a penalty.

C-IV: Finally, even if we were to able to evaluate the feasibility of a set of rewards that satisfy (14) (under the heuristics employed above) we need to come up with a way to pick a 'good' reward function from amongst these, similar to the optimization in (9-11).

[6] provides one heuristic approach to address all these concerns but we will not go into the specifics of that. Instead in the following section, we will employ some of the heuristics discussed here to solve the problem of estimating rewards from sampled trajectories resulting from an optimal policy rather than from the policy itself.

4 Inverse Reinforcement Learning from Sampled Trajectories.

In this section, we will continue to allow infinite state spaces but make a simplifying assumption to avoid some of the issue we discussed in the previous section.

Assumption 3 *We will assume that the initial state for our IRL problem is fixed and that the expert agent takes all trajectories beginning from a fixed state $s_0 = \bar{s}$.*

Let us suppose we are given M trajectories taken by the expert agent who is implementing some (unknown) policy π^*. Note that by a trajectory, here we mean the time history of a states $\{(s_t)\}_{t=0}^{T}$ where s_t denotes the state at time t and T is the time horizon for the trajectory. Our goal then is as follows:

"obtain an estimate $R(s)$ of the reward function that generated the M trajectories (as the result of an optimal policy) given the MDP parameters $(\mathcal{S}, \mathcal{A}, P, \gamma)$."

To solve this problem we will consider a similar necessary[5] condition that our policy must satisfy.

The Big Idea

Consider a set of policies $\{\pi^1\}_1^k$. Then for π^* to be an optimal policy for our problem our reward function must be such that

$$V^{\pi^*}(\bar{s}) \geq V^{\pi}(\bar{s}) \quad \forall \pi \in \{\pi^1\}_1^k. \tag{16}$$

A primary challenge that needs to be addressed here is that of computing these $V^{\pi^*}(\bar{s})$ when the policy itself is unknown. But before we address this challenge let us step back and present some additional notation to help us out.

4.1 Preliminaries and Notations

We will continue with the linear approximation for the reward function as

$$R(s) = \sum_{1}^{d} \alpha_i \phi_i(s) = \boldsymbol{\alpha} \cdot \boldsymbol{\phi}(s). \tag{17}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^d$ is the weight vector and $\boldsymbol{\phi}(s)$ is the vector function with elements comprising of the d basis functions $\{\phi_i(s)\}_{i=1}^d$. Then given any policy π we can obtain the value function at state $\bar{s}$ using (17) as,

$$\begin{aligned} V^{\pi}(\bar{s}) &= \mathbf{E}[R(s_0 = \bar{s}) + \gamma R(s_1) + \gamma^2 R(s^2) + \ldots | \pi] \\ &= \boldsymbol{\alpha} \cdot \mathbf{E}[\boldsymbol{\phi}(s_0) + \gamma \boldsymbol{\phi}(s^1) + \ldots] \\ &= \boldsymbol{\alpha} \cdot [V_1^{\pi}(\bar{s}) \ \ldots \ V_d^{\pi}(\bar{s})]^T \end{aligned}$$

where $V_i^{\pi}(\bar{s})$ is the value function for the policy π evaluated at state $\bar{s}$ when the true reward function is given by $\phi_i(s)$. We define the vector

$$\boldsymbol{\mu}(\pi) \triangleq [V_1^{\pi}(\bar{s}) \ \ldots \ V_d^{\pi}(\bar{s})]^T$$

giving us the simple representation for $V^{\pi}(\bar{s}) = \boldsymbol{\alpha} \cdot \boldsymbol{\mu}(\pi)$. Then the problem of evaluating (16) can be written as

$$\boldsymbol{\alpha} \cdot (\boldsymbol{\mu}(\pi^*) - \boldsymbol{\mu}(\pi_i)) \geq 0 \forall i \in \{1, \ldots, k\}. \tag{18}$$

As we noted above a challenge we are faced with in evaluating (18) is in the evaluation of value function of each policy to compute the vectors $\boldsymbol{\mu}$. We will approximate these value function using samples.

[5]Note that (16) is not a sufficient condition. For instance, we actually need (16) to hold for every state s and not just for the initial state $\bar{s}$ for π^* to be an optimal policy.

4.2 Approximating the Value Function

The approaches we discuss is be slightly different for evaluating π^* and for any other policy π. Recall that the policy π^* is in fact unknown and all we have are access to M state trajectories of horizon length T. Then a natural approximation of the value function $V_i^{\pi^*(\bar{s})}$ is obtained as

$$\hat{V}_i^{\pi^*}(\bar{s}) = \frac{1}{M}\sum_{j=1}^{M}\sum_{t=0}^{T}\gamma^t\phi_i(s_t) \quad \forall i \in \{1,\dots,d\}.$$

This can then be used to estimate $\boldsymbol{\mu}(\pi^*)$ as $\hat{\boldsymbol{\mu}}(\pi^*) = [\hat{V}_1^{\pi^*}(\bar{s}),\dots,\hat{V}_d^{\pi^*}(\bar{s})]$. To evaluate any other policy π we can simply run Monte-Carlo simulations starting from state $\bar{s}$ in every iteration and by generating actions according to policy π at every state. Let us suppose we run N iterations each with time horizon length τ. Then we estimate the value of a policy π as

$$\hat{V}_i^{\pi}(\bar{s}) = \frac{1}{N}\sum_{j=1}^{N}\sum_{t=0}^{\tau}\gamma^t\phi_i(s_t) \quad \forall i \in \{1,\dots,d\}.$$

Once again we obtain $\hat{\boldsymbol{\mu}}(\pi) = [\hat{V}_1^{\pi}(\bar{s}),\dots,\hat{V}_d^{\pi}(\bar{s})]$.

4.3 Obtaining the Reward Function

We are now equipped to present an algorithmic approach to obtain the reward function that relies on the big idea in (16). The steps involved are as follows-

Step-1: Pick a random policy $\pi^{(0)}$ and estimate $\hat{\boldsymbol{\mu}}(\pi^{(0)})$ as highlighted above. We will also maintain a list of policies initialized $\Pi = \{\pi^{(0)}\}$. Set $i = 1$.

Step-2: Evaluate

$$\boldsymbol{\alpha}^{(i)} = \arg\max_{\alpha} \quad \alpha\sum_{\pi\in\Pi} l\big(\boldsymbol{\alpha}\cdot(\hat{\boldsymbol{\mu}}(\pi^*) - \hat{\boldsymbol{\mu}}(\pi_i))\big) \tag{19}$$

$$S.T \quad |\alpha_j| \le 1 \quad j = 1,\dots,d \tag{20}$$

where $\hat{\boldsymbol{\mu}}(\pi^*)$ and $\hat{\boldsymbol{\mu}}(\pi)$ are estimated as highlighted in Section 4.2.

Step-3: Using any method obtain the optimal policy $\pi^{(i)}$ for our MDP with reward function $R(s) = \boldsymbol{\alpha}^{(i)}\boldsymbol{\phi}(s)$.

Step-4: Add $\pi^{(i)}$ to π. Set $i = i + 1$ and repeat from step 2.

Above we have not indicated the termination criteria but any criteria may be used. One heuristic maybe to check whether $\min_{\pi\in\Pi}\boldsymbol{\alpha}\cdot(\hat{\boldsymbol{\mu}}(\pi^*) - \hat{\boldsymbol{\mu}}(\pi_i)) \le \epsilon$ after **Step 3**. In this case, optimal policy for the true reward function does not perform far worse than the optimal policy for the particular reward function. So it is likely that we found the true reward function itself.

Note that the function $l : \mathbb{R} \to \mathbb{R}$ in (19) is a simple loss function used to penalize constraint deviations according to the heuristic described in C-III in Section 3.4. In the simulations we perform subsequently, we use the following form for the loss function.

$$l(x) = \begin{cases} x & x \ge 0 \\ 2x & x < 0 \end{cases}$$

In the following section we consider a simple finite-state space MDP as an example, and attempt to recover the true reward functions from a set of state trajectories that were obtained on implementing the optimal policy.

4.4 Example

We consider the problem of traversing over a grid as shown in Figure 2.

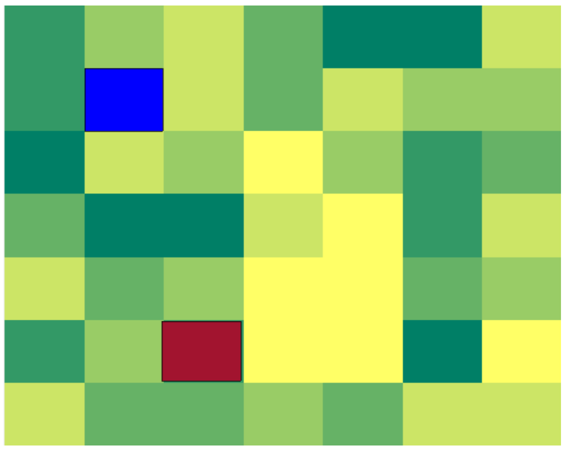


Figure 2: A simple 7×7 grid where the red state is the goal and blue state is the initial state.

The agent's goal is to move from the blue tile to the red tile in as little time as possible. The dynamics are simple: the agent can attempt to move one step in any of the four cardinal directions (except at the border tiles, where the agent cannot attempt to move across the border, so it has fewer directions available). With probability 0.75, the agent will succeed. With probability $0.25/(n-1)$ for each direction, it will instead move in a different cardinal direction, where n is the number of cardinal directions available to the agent (e.g., for agents that are not at the border, $n = 4$). We assume that the red state is absorbing.

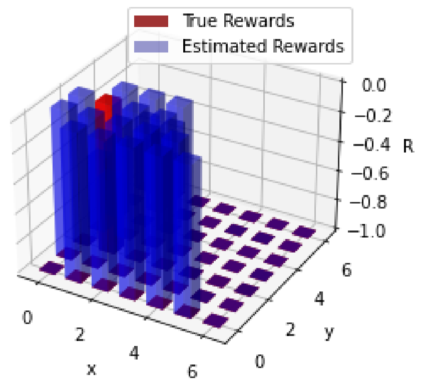
Let us index the 49 states in our system as $\{1, \ldots, 49\}$. We use the natural set of basis functions for our reward function as $\{\phi_i(s)\}_{i=1}^{49}$ with each function defined as

$$\phi_i(s) = \begin{cases} 1 & s = i \\ 0 & \text{otherwise} \end{cases}$$

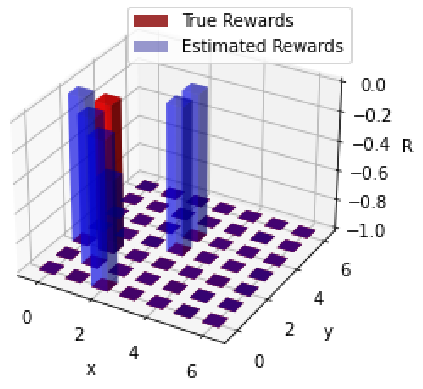
It is easy to see then that the vector $\boldsymbol{\alpha}$ is nothing but a set of rewards at each of the finite states. The true optimal policy π^* can easily be obtained for this problem using a value iteration based approach. We assume access to $M = 50$ trajectories each having time horizon $T = 20$ and use these trajectories to estimate $\hat{\boldsymbol{\mu}}(\pi^*)$. Likewise we run Monte Carlo simulations with $N = 50$ and $\tau = 20$ to estimate $\hat{\boldsymbol{\mu}}(\pi)$ for any policy π.

We run the algorithm described in Section 4.3 over 18 iterations and the results are plotted below. Figure 3 showcases the change in estimated reward function across iterations. A GIF (titled, 'reward_learning.gif') file showing the process of learning the reward function is also attached with this report. We observe that as we get closer to the end of iterations the reward resembles the true reward closely. At the same time, as we had in the example from 3.3 the reward is not learnt perfectly, but the qualitative features of the true reward are captured.

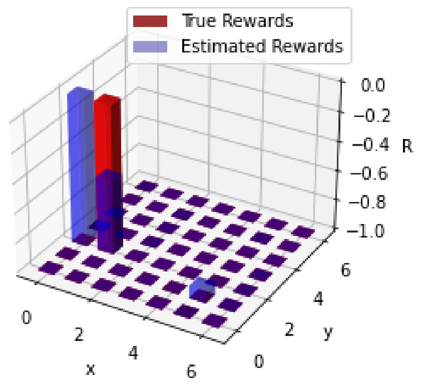
We can also evaluate some metrics on the optimal policies $\pi^{(i)}$'s that are generated during the algorithm presented Section 4.3. Specifically we are interested in the value of the policies evaluated with respect to the true reward function (Figure 4a). We see that as the number of iterations progress the policy starts behaving near optimally despite differing from the true optimal in a few states. This is likely because the algorithm quickly learns about the best action to be taken at the states which occur in the experts trajectories. Given the nature of the problem, it is unlikely that every state will be visited in a path between the blue and the red states (Figure 2). So even if the optimal policy is not perfectly learnt for the some states it is likely that these states are never visited in trying to achieve the goal of maximizing our rewards. Figure 4b tracks the number of states at which the policy $pi^{(i)}$ differs from the true optimal policy across iterations.



(a) Rewards at every state after 1 step of the iteration

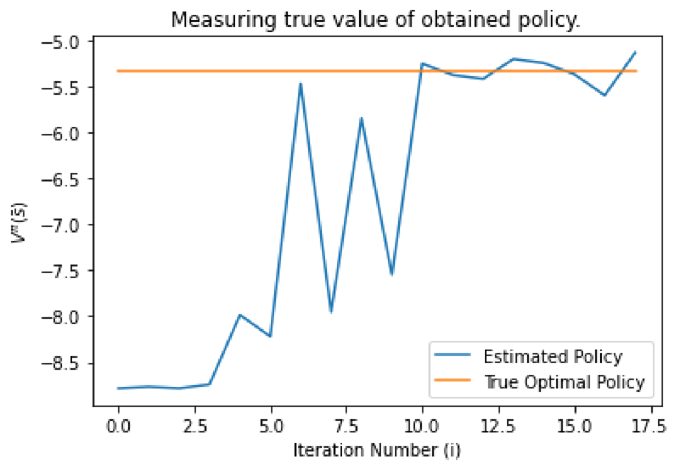


(b) Rewards at every state after 12 step of the iteration

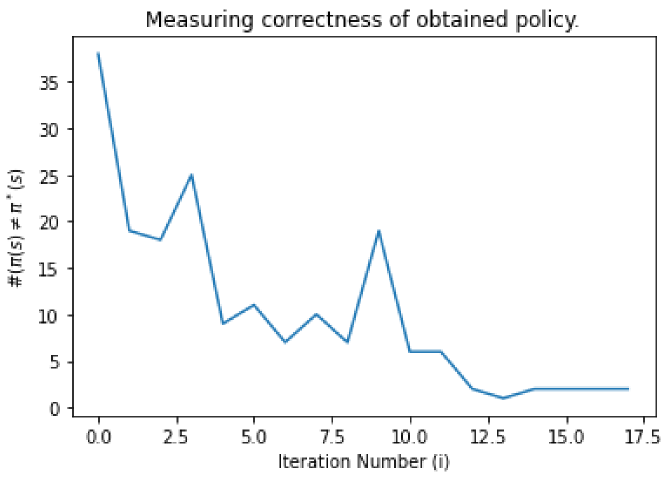


(c) Rewards at every state after 18 step of the iteration

Figure 3: **(Estimated rewards across iterations.)** We see that initially our reward estimates are extremely poor but with iterations of our algorithm, our reward function matches the true reward function very closely.



(a) True value of the policy in every iteration.



(b) Number of states where the policy from an iteration deviates from the optimal policy.

Figure 4: **(Quality of the policy $\pi^{(i)}$ generated over iterations.)**

5 Discussion

Through our first example in Section 3.3 we saw that despite not being able to perfectly recover the reward vector, we obtained a reward that qualitative captured all aspects of the true reward. In particular, both the true and the estimated reward correspond to a minimum time reachability problem. In this sense, it capture our goal of being able to understand the intent of the agent that implemented the original optimal policy. More over one can imagine that the computed reward function will work as well as the true one, if we wished to compute the optimal policy in the event the dynamics of the system changed. Thus, we have been able to accomplish both the goals - intent learning and apprenticeship learning - we desired when we motivated the study of this problem.

While these features maybe be more pronounced in our simple example, the success of our approach in the second example in Section 4.4 seems to further strengthen our claim to fame. Indeed even in the second example, where we learnt the rewards with a good degree of success, we have shown that not only can we perform the task of inverse reinforcement learning in a more complicated problem, but that we can do so with a fairly small number ($M = 50$) of sampled trajectory realization of the optimal policy.

But note that both the examples we worked with assumed a finite state space, and there are still many challenges involved in dealing with infinite state spaces as highlighted in Section 3.4. In addition, we can extend the present work in multiple ways. A non-exhaustive list is presented below.

1. So far we assumed perfect observability but we can also consider the problem of IRL over Partially Observable MDPs (POMDPs). In doing so we would have to work with the uncountable belief space and thus many of the issues pertaining to the infinite state spaces crop up.

2. In considering sampled trajectories, we assumed that the ‘expert’ was perfect in their ability to act according to the optimal policy. But in reality the expert may only be near-optimal in their decision making. Considering such sub-optimality is another direction that can be explored.

3. Finally, in our approach to exploit the sampled trajectories to obtain the reward function, we gave no performance guarantees on how close our estimate was to the true reward, or on how many samples we would need to be able get a close enough estimate.

On the last matter raised above, Abbeel and Ng [3] consider an optimization problem different from the one in (19-20) which converts the problem of finding the vector $\boldsymbol{\alpha}$ into that of finding a separating hyperplane for two convex sets. By doing so, they are also able to provide some of the guarantees on performance discussed in the last point above.

In addition to the issues raised above, an overarching issue in the methods described in this work was outlined in C-III in Section 3.4. Indeed approximating the true reward function using an element from a finite dimensional subspace of the function space may not always be possible. As a solution to this issue, many works [7, 8] seek to work with probability distributions over the space of reward functions. Then, given the sample of optimal (or near-optimal) trajectories, these approaches form posterior distributions on the reward function space. The true reward is then usually estimated as the mean of posterior distribution. A larger survey of IRL problems cane be found in [9].

References

[1] Julian Jara-Ettinger. Theory of mind as inverse reinforcement learning. *Current Opinion in Behavioral Sciences*, 29:105–110, 2019.

[2] Rust John. Maximum likelihood estimation of discrete control processes. *SIAM journal on control and optimization*, 26(5):1006–1024, 1988.

[3] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.

[4] Christopher G. Atkeson and Stefan Schaal. Robot learning from demonstration. *In ICML*, pages 12 – 20, 1997.

[5] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. *Linear matrix inequalities in system and control theory.* SIAM, 1994.

[6] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *ICML*, volume 1, page 2, 2000.

[7] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *IJCAI*, volume 7, pages 2586–2591, 2007.

[8] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

[9] Shao Zhifei and Er Meng Joo. A survey of inverse reinforcement learning techniques. *International Journal of Intelligent Computing and Cybernetics*, 2012.