Inverse Reinforcement Learning

Shaz Nazar Karumarot

Overview

What is Reinforcement Learning?

Inverse Reinforcement Learning

Algorithms and Experiments

IRL in Finite Spaces

- 5x5 Grid World

IRL in Large Spaces

- Mountain Car

IRL from Sampled Trajectories

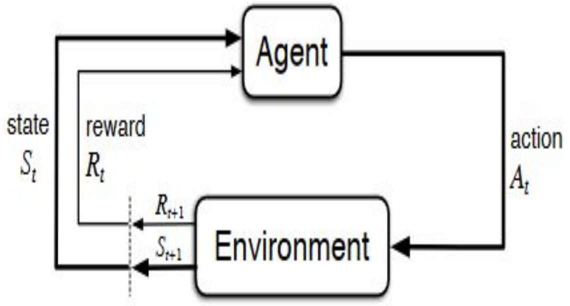
- Continuous Grid World

Conclusion

What is Reinforcement Learning?

Reinforcement Learning

- Computational approach to understanding and automating goal-directed learning and decision making.
- Learning by an agent from direct interaction with its environment
- We use Markov Decision Processes (MDPs) to formulate the problem of learning.



What is Reinforcement Learning?

Challenge in Reinforcement Learning

- Given states, actions, sometimes a model of the environment and a reward function, the objective is to learn an optimal policy
- It is often easier to specify the reward function than to directly specify the value function (and/or optimal policy)
- However, in some problems even the reward function is frequently difficult to specify manually.
- An example is the task of ‘driving well’.

What is Inverse Reinforcement Learning?

- The reverse problem of Reinforcement Learning.
- Reconstruct the reward function by observing an agent act out under an optimal policy.
- Relevant due to the fact that reward function has to be manually tweaked multiple times in a RL problem and determining the exact reward function is not trivial.

Inverse Reinforcement Learning

Definition

- The problem of deriving a reward function from observed behavior is referred to as inverse reinforcement learning (Ng & Russell, 2000).

Given 1) measurements of an agent's behavior over time, in a variety of circumstances, 2) if needed, measurements of the sensory inputs to that agent; 3) if available, a model of the environment.

Determine the reward function being optimized.

Inverse Reinforcement Learning

RL

given:

states $\mathbf{s} \in \mathcal{S}$, actions $\mathbf{a} \in \mathcal{A}$

(sometimes) transitions $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$

reward function $r(\mathbf{s}, \mathbf{a})$

learn $\pi^\star(\mathbf{a}|\mathbf{s})$

IRL

given:

states $\mathbf{s} \in \mathcal{S}$, actions $\mathbf{a} \in \mathcal{A}$

(sometimes) transitions $p(\mathbf{s}'|\mathbf{s}, \mathbf{a})$

samples $\{\tau_i\}$ sampled from $\pi^\star(\tau)$

learn $r_\psi(\mathbf{s}, \mathbf{a})$

reward parameters

...and then use it to learn $\pi^\star(\mathbf{a}|\mathbf{s})$

Motivation for Inverse RL

Better Reward Functions

- Often, a predefined reward function is not given, and determining an appropriate one is challenging without relying on intuition.
- This requirement to specify a reward function in advance restricts the potential applications of Reinforcement Learning (RL).
- Inverse Reinforcement Learning (IRL) addresses this limitation by offering a method to obtain a suitable numerical reward function.
- While access to an optimal policy is necessary, these demonstrations are typically easily obtainable in practical scenarios from experts.

Motivation for Inverse RL

Better Transferability

- A reward function serves as a reflection of an agent's preferences and is transferable to another agent.
- When the specifications of a second agent only slightly differ from the first one, the learned reward function becomes a foundation for the second agent.
- However, this advantageous transferability does not extend to optimal policies. For instance, if the second agent has a larger state space while maintaining the same dynamics as the first one, the optimal policy for the first agent does not provide insights into what the optimal policy for the second agent would be.

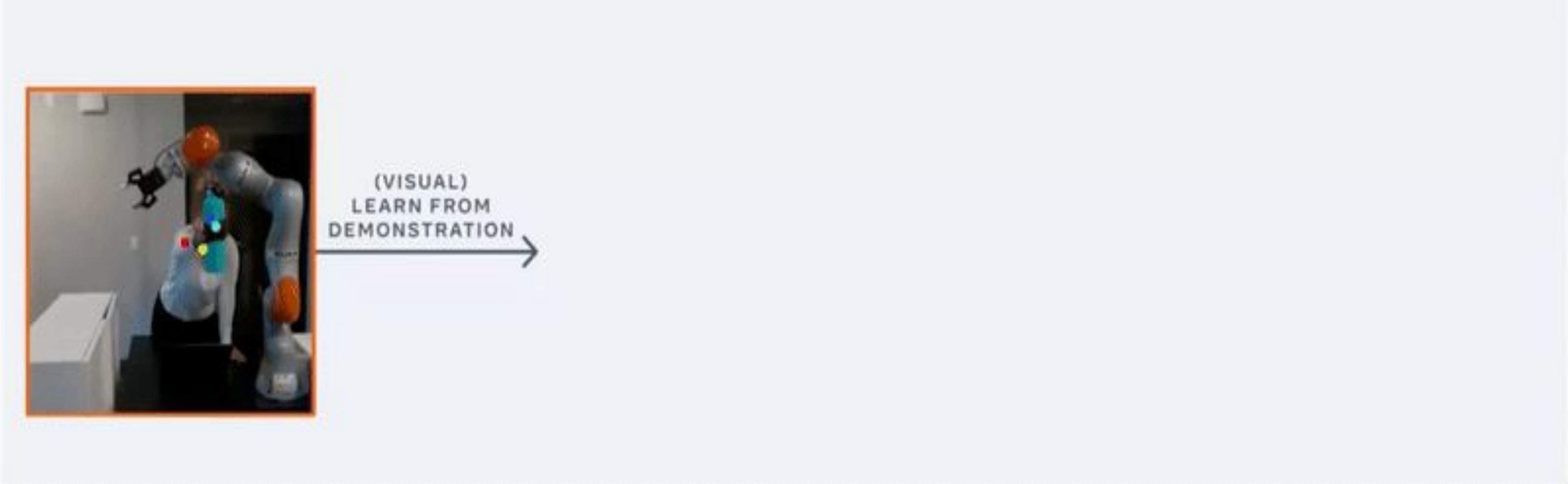
Motivation for Inverse RL

Objectives of Inverse RL

- Application of reinforcement learning and related methods as computational models for animal and human learning - we must consider the reward function as an unknown to be ascertained through empirical investigation.
- Learn from ‘expert’ agents (like humans) - recover the reward function instead of directly learning the policy.

  “The reward function, rather than the policy, is the most succinct, robust, and transferable definition of the task”

Motivation for Inverse RL



Source: https://ai.facebook.com/blog/teaching-ai-to-manipulate-objects-using-visual-demos/

Inverse Reinforcement Learning

Methods

- To obtain the unique solution in IRL, many studies have proposed additional objective functions to be optimized, such as **margin between the optimal policy and others** [Ng and Russell, 2000, Abbeel and Ng, 2004, Ratliff et al., 2006b,a, 2009, Silver et al., 2010] and to **maximize the entropy** [Ziebart et al., 2008, Ziebart, 2010, Kitani et al., 2012, Shiarlis et al., 2016]
- Other IRL methods include **Bayesian IRL**, **Guided Cost Learning**, **Generative Adversarial Imitation Learning** (GAIL) etc..

Algorithms
and
Experiments
(Ng & Russel, 2000)

IRL in Finite Spaces

I. IRL in Finite Spaces

- The state space is finite, the model is known, and the complete policy is observed.
- Components:
  - **Finite State Space (S)**: The set of possible states is finite.
  - **Actions (A)**: A set of k actions, denoted as A={a1,...,ak}A={a1,...,ak}.
  - **Transition Probabilities (P_{sa})**: The probabilities of transitioning from one state to another given an action.
  - **Discount Factor (γ)**: A discount factor influencing the importance of future rewards.
  - **Policy (π)**: An observed policy for decision-making.
- Objective:

  The goal is to find a **set of possible reward functions (*R*)** such that the given policy π is an optimal policy within the MDP (S,A,{P_{sa}},γ,R).

I. IRL in Finite Spaces

- The characterization of the set of all reward functions for which a given policy is optimal in the context of Inverse Reinforcement Learning (IRL) in finite state spaces is given by

Theorem 3 *Let a finite state space S, a set of actions $A = \{a_1, \ldots, a_k\}$, transition probability matrices $\{\boldsymbol{P}_a\}$, and a 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 $a = a_2, \ldots, a_k$, the reward $\boldsymbol{R}$ satisfies*

$$(\boldsymbol{P}_{a_1} - \boldsymbol{P}_a)(\boldsymbol{I} - \gamma \boldsymbol{P}_{a_1})^{-1} \boldsymbol{R} \succeq 0 \qquad (4)$$

I. IRL in Finite Spaces

Theorem 3 *Let a finite state space S, a set of actions $A = \{a_1, \ldots, a_k\}$, transition probability matrices $\{\boldsymbol{P}_a\}$, and a 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 $a = a_2, \ldots, a_k$, the reward $\boldsymbol{R}$ satisfies*

$$(\boldsymbol{P}_{a_1} - \boldsymbol{P}_a)(\boldsymbol{I} - \gamma \boldsymbol{P}_{a_1})^{-1} \boldsymbol{R} \succeq 0 \qquad (4)$$

- This theorem provides an optimality condition for a policy to be optimal in terms of the reward function.
- The term $(P_{a1} - P_a)$ reflects the difference in the transition probabilities when taking action a_1 compared to the other actions a.
- $(I - \gamma P_{a1})^{-1}$: This part represents the inverse of the matrix resulting from the discounted transition probabilities when taking action a1.
- The inequality indicates that the resulting vector RR should be a non-negative vector.

I. IRL in Finite Spaces

- It is derived from the Bellman (1,2) and Bellman Optimality (3) equations.

$$V^\pi(s) = R(s) + \gamma \sum_{s'} P_{s\pi(s)}(s') V^\pi(s') \quad (1)$$

$$Q^\pi(s,a) = R(s) + \gamma \sum_{s'} P_{sa}(s') V^\pi(s') \quad (2)$$

$$\pi(s) \in \arg\max_{a \in A} Q^\pi(s,a) \quad (3)$$

Proof. Since $\pi(s) \equiv a_1$, Equation (1) may be written $\boldsymbol{V}^\pi = \boldsymbol{R} + \gamma \boldsymbol{P}_{a_1} \boldsymbol{V}^\pi$. Thus,[1]

$$\boldsymbol{V}^\pi = (\boldsymbol{I} - \gamma \boldsymbol{P}_{a_1})^{-1} \boldsymbol{R} \quad (5)$$

Substituting Equation (2) into (3) from Theorem 2, we see that $\pi \equiv a_1$ is optimal if and only if

$$\begin{aligned}
a_1 \equiv \pi(s) \in \arg\max_{a \in A} \sum_{s'} P_{sa}(s') V^\pi(s') \quad & \forall s \in S \\
\Leftrightarrow \sum_{s'} P_{sa_1}(s') V^\pi(s') & \\
\geq \sum_{s'} P_{sa}(s') V^\pi(s') \quad & \forall s \in S, a \in A \\
\Leftrightarrow \boldsymbol{P}_{a_1} \boldsymbol{V}^\pi \succeq \boldsymbol{P}_a \boldsymbol{V}^\pi \quad & \forall a \in A \setminus a_1 \\
\Leftrightarrow \boldsymbol{P}_{a_1} (\boldsymbol{I} - \gamma \boldsymbol{P}_{a_1})^{-1} \boldsymbol{R} & \\
\succeq \boldsymbol{P}_a (\boldsymbol{I} - \gamma \boldsymbol{P}_{a_1})^{-1} \boldsymbol{R} \quad & \forall a \in A \setminus a_1
\end{aligned}$$

where the last implication in this derivation used Equation (5). This completes the proof. $\square$

I. IRL in Finite Spaces

- However, R=0 is a solution and there could be multiple solutions that satisfy (4)
- Linear Programming can be used to find feasible point of constraints
- Not all solutions are equally meaningful, and there is a desire to choose between solutions satisfying the given optimality condition.
- The goal is to choose a reward function R that not only makes the given policy π=a1 optimal but also favors solutions where any single-step deviation from π is as costly as possible.
- We aim to maximize the sum of differences between the quality of the optimal action and the quality of the next-best action (maximum margin).

$$\sum_{s \in S} \left(Q^{\pi}(s, a_1) - \max_{a \in A \setminus a_1} Q^{\pi}(s, a) \right) \qquad (6)$$

I. IRL in Finite Spaces

- A penalty term is introduced in the objective function, represented as **−λ||R||** to favor solutions with mainly small rewards, assuming that simpler solutions are preferable.
- The adjustable penalty coefficient λ balances the goals of having small reinforcements and maximizing the sum of differences.
- Putting it all together, the optimization problem is formulated as:

$$\begin{aligned}
\text{maximize} \quad & \textstyle\sum_{i=1}^{N} \min_{a \in \{a_2,\ldots,a_k\}} \{(\boldsymbol{P}_{a_1}(i) - \boldsymbol{P}_a(i)) \\
& \qquad (\boldsymbol{I} - \gamma \boldsymbol{P}_{a_1})^{-1} \boldsymbol{R}\} - \lambda ||\boldsymbol{R}||_1 \\
\text{s.t.} \quad & (\boldsymbol{P}_{a_1} - \boldsymbol{P}_a)(\boldsymbol{I} - \gamma \boldsymbol{P}_{a_1})^{-1} \boldsymbol{R} \succeq 0 \\
& \qquad \forall a \in A \setminus a_1 \\
& |\boldsymbol{R}_i| \leq R_{\max}, \quad i = 1, \ldots, N
\end{aligned}$$

- This can be solved efficiently as a linear program.

5x5 Grid World

Experiment 1: 5x5 Grid World

- 5x5 grid world where the agent starts from the lower-left grid square and aims to reach the upper-right grid square, receiving a reward of 1 upon reaching it.
- Actions correspond to moving in the four compass directions but are noisy, having a 30% chance of moving in a random direction instead.
- An optimal policy is known
- Objective:
  - The inverse reinforcement problem involves recovering the reward structure given the known policy and problem dynamics.
- Result:
  - Obtained a reward function which closely approximated the true reward

Experiment 1: 5x5 Grid World

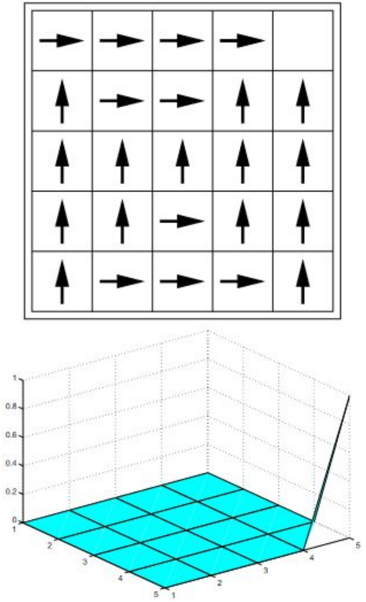


Figure 1. Top: 5x5 grid world with optimal policy. *Bottom*: True reward function.

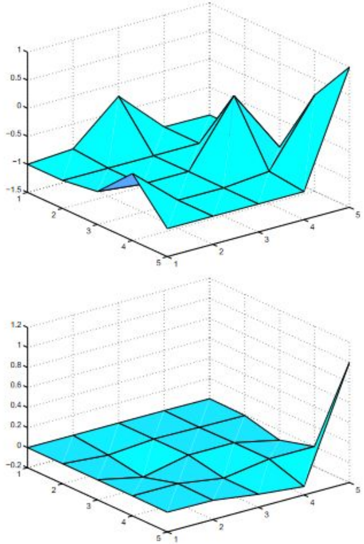
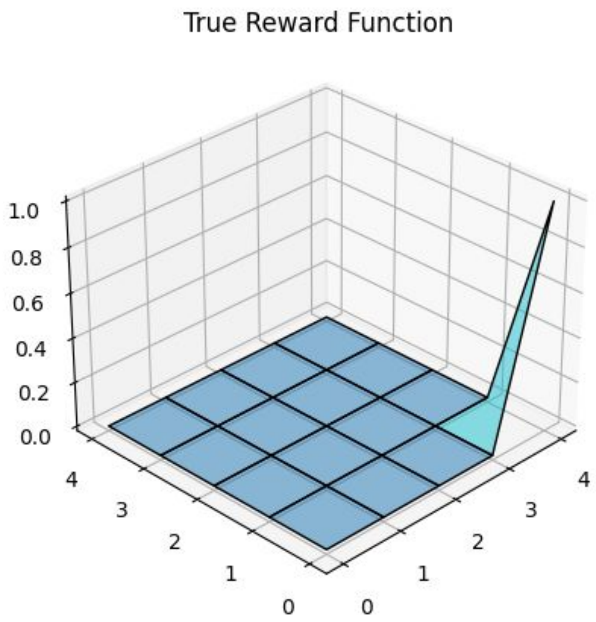


Figure 2. Inverse RL on the 5×5 grid. *Top*: $\lambda = 0$. *Bottom*: $\lambda = 1.05$.

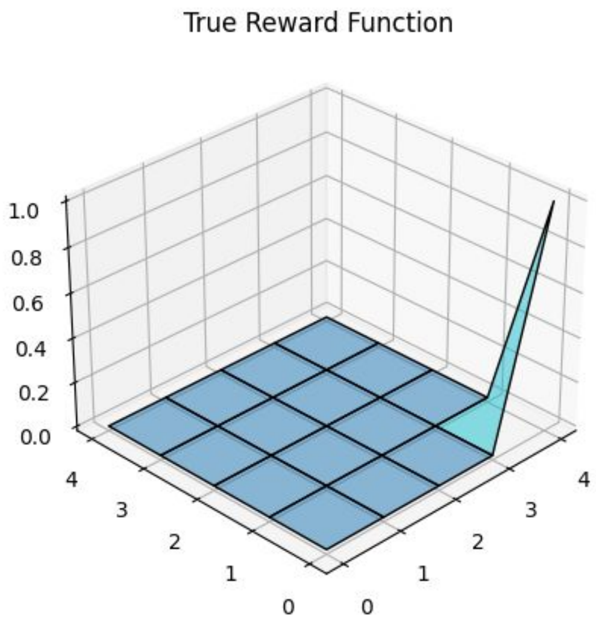
Ng, A. Y., & Russell, S. (2000). Algorithms for inverse reinforcement learning. Proc. ICML

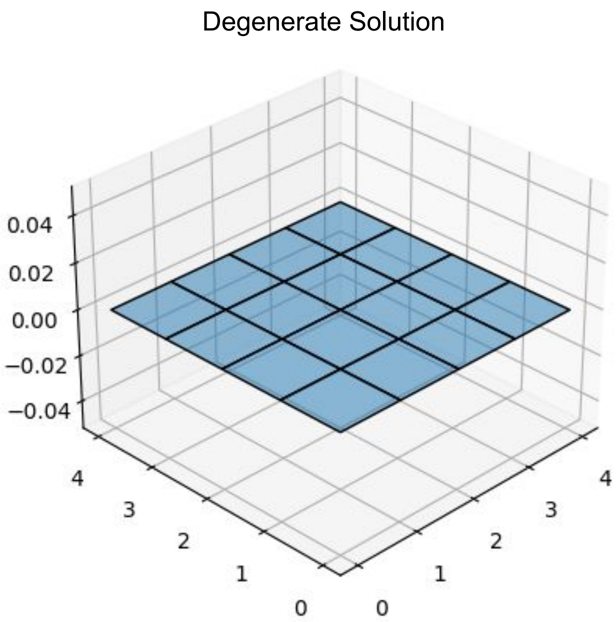
Experiment 1: 5x5 Grid World



https://github.com/Shaz-5/IRL

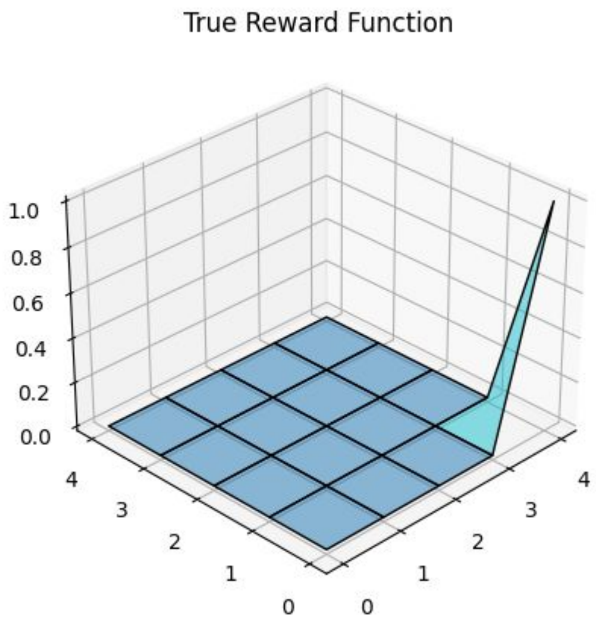
Experiment 1: 5x5 Grid World

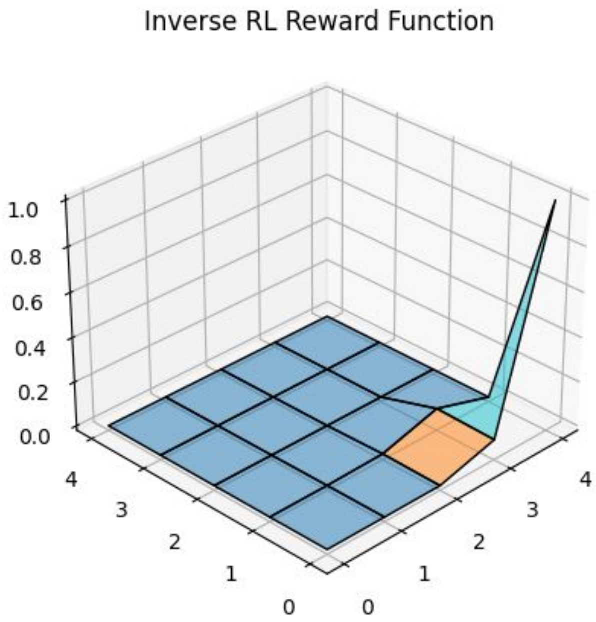




https://github.com/Shaz-5/IRL

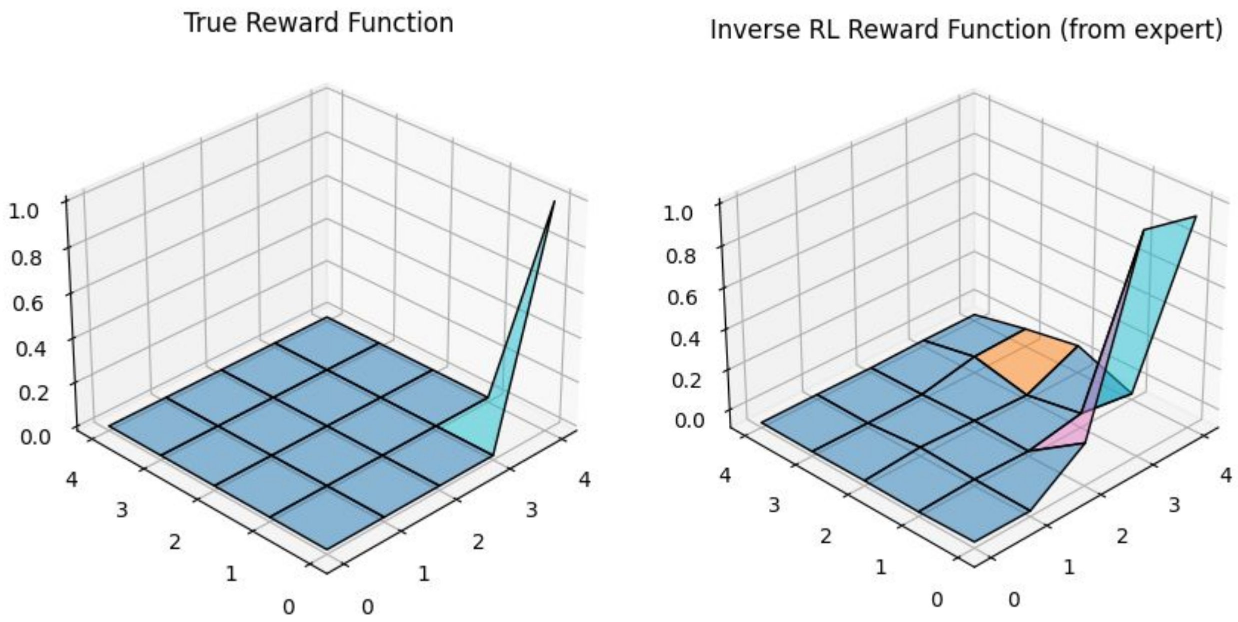
Experiment 1: 5x5 Grid World





https://github.com/Shaz-5/IRL

Experiment 1: 5x5 Grid World



https://github.com/Shaz-5/IRL

IRL in Large State Spaces

II. IRL in Large State Spaces

- Infinite-state Markov Decision Processes (MDPs) are considered, specifically with the state space $S = \mathbb{R}^n$
- The reward function *R* is now a function from *S* to the real numbers.
- **Linear Function Approximation:**
  - The reward function *R(s)* is expressed as a linear combination of fixed, known, and bounded basis functions $\phi_1 \ldots \phi_d$

$$R(s) = \alpha_1\phi_1(s) + \alpha_2\phi_2(s) + \cdots + \alpha_d\phi_d(s) \qquad (8)$$

  - The αs are unknown parameters that need to be 'fit'.
- The value function when *R* is given by the linear combination is expressed as:

$$V^\pi = \alpha_1 V_1^\pi + \cdots + \alpha_d V_d^\pi. \qquad (9)$$

II. IRL in Large State Spaces

- The value function when *R* is given by the linear combination is expressed as:

$$V^\pi = \alpha_1 V_1^\pi + \cdots + \alpha_d V_d^\pi . \qquad (9)$$

- which gives,

$$\mathrm{E}_{s' \sim P_{sa_1}} [V^\pi(s')] \geq \mathrm{E}_{s' \sim P_{sa}} [V^\pi(s')] \qquad (10)$$

The notation $E_{s'}$~$P_{sa}[V^\pi(s')]$ represents the expected value of the value function $V^\pi(s')$ when transitioning from state *s* to state *s'* under action *a*.

II. IRL in Large State Spaces

- **Sampling for Infinite State Spaces:**
  - Due to the **challenge of having infinitely many constraints**, the formulation is adjusted by sampling a **large but finite subset S_0** of states.
  - The linear constraints are then applied only to this subset of states.
- **Penalty for Violation:**
  - A penalty term is introduced to handle the **potential issue of not being able to express any reward function** (other than the trivial R=0) for which π is optimal due to linear function approximation.
  - The penalty term penalizes violations of the constraints (some constraints are relaxed), and the penalty weight is an adjustable parameter.

II. IRL in Large State Spaces

- The final linear programming formulation is as follows:

$$
\begin{aligned}
\text{maximize } & \textstyle\sum_{s \in S_0} \min_{a \in \{a_2, \dots, a_k\}} \{ \\
& \quad p(\mathrm{E}_{s' \sim P_{sa_1}}[V^\pi(s')] - \mathrm{E}_{s' \sim P_{sa}}[V^\pi(s')])\} \\
\text{s.t. } & |\alpha_i| \leq 1, \quad i = 1, \dots, d
\end{aligned}
$$

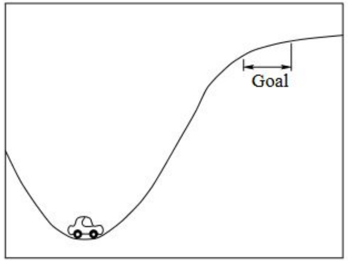
$$
\text{where} \quad p(r) = \begin{cases} r & \text{if } r \geq 0 \\ 2r & \text{otherwise} \end{cases}
$$

- The objective is to maximize a weighted sum over the sampled states S_0 of the minimum violation of the optimality conditions.
- The constraints ensure that the coefficients αiαi are bounded.
- The penalty function p(r) penalizes violations, with the penalty weight 2 being an adjustable parameter.

Mountain Car

Experiment 2: Mountain Car

- Utilized the well-known "mountain-car" task.
- True, undiscounted reward is -1 per step until reaching the goal at the top of the hill.
- The state is defined by the car's position and velocity.
- Due to the continuous state space, the version of the algorithm described in Section 4 (Linear Function Approximation in Large State Spaces) was applied.

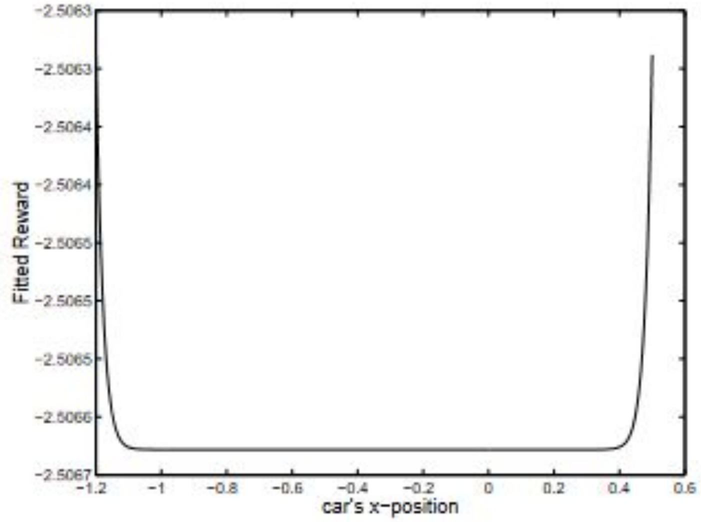


- The function approximator class for the reward was chosen to be functions of the car's position only, consisting of all linear combinations of 26 evenly spaced Gaussian-shaped basis functions.

Experiment 2: Mountain Car

Experiment 2.1: Original Task:

- Provided the optimal policy to the algorithm.
- The solution accurately captures the reward structure.

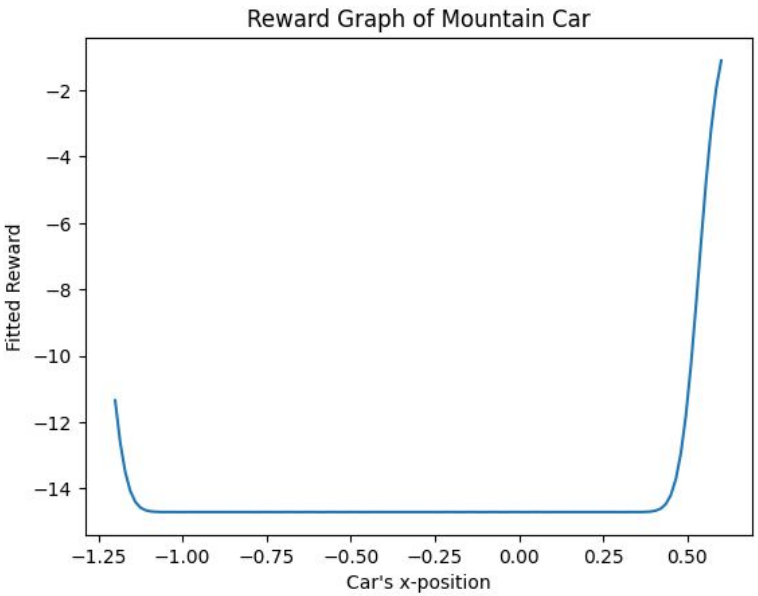


Ng, A. Y., & Russell, S. (2000). Algorithms for inverse reinforcement learning. Proc. ICML

Experiment 2: Mountain Car

Experiment 2.1: Original Task:

- Provided the optimal policy to the algorithm.
- The solution accurately captures the reward structure.

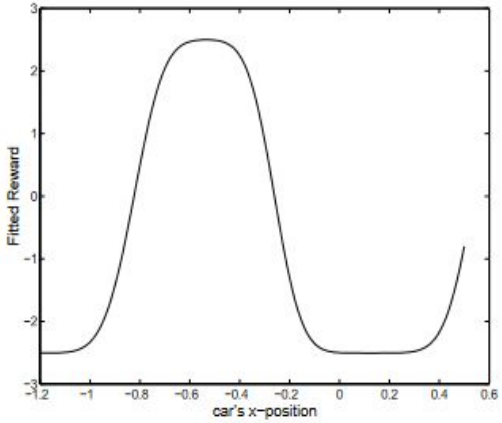


https://github.com/Shaz-5/IRL

Experiment 2: Mountain Car

Experiment 2.2: Modified Task:

- Altered the true reward to be 1 in an interval [-0.72, -0.32] centered around the bottom of the hill and 0 everywhere else.
- Discount factor γ=0.99.
- The optimal policy is to move quickly to the bottom of the hill and park there, if possible.
- The algorithm was run on this modified problem.
- Successfully recovers the main structure of the reward, with an artifact on the right side, potentially from the effect of occasionally "shooting off" the right end.
- Overall, considered a fairly good solution for the problem.



Ng, A. Y., & Russell, S. (2000). Algorithms for inverse reinforcement learning. Proc. ICML

IRL from Sampled Trajectories

III. IRL from Sampled Trajectories

- This section addresses the IRL problem in a more **realistic scenario** where the policy **π is known only through a set of actual trajectories in the state space**, rather than having access to the explicit model of the Markov Decision Process (MDP). The goal is to find the reward function R that maximizes the expected return under the given trajectories.
- The initial state distribution D is fixed.
- The reward function R is assumed to be expressed using a linear function-approximator class, similar to the approach taken in the previous section for infinite state spaces.
- For each policy π that we will consider (including the optimal one), we will need a way of estimating $V^{\pi}(s_o)$ for any setting of the α_is.

III. IRL from Sampled Trajectories

For each policy π that we will consider (including the optimal one), we will need a way of estimating $V^{\pi}(s_0)$ for any setting of the α_is. To do this, we first execute m Monte Carlo trajectories under π. Then, for each $i = 1, \ldots, d$, define $\hat{V}_i^{\pi}(s_0)$ to be what the average empirical return would have been on these m trajectories if the reward had been $R = \phi_i$. For example, if we take only $m = 1$ trajectories, and if that trajectory visited the sequence of states $(s_0, s_1, \ldots)$, then we have:

$$\hat{V}_i^{\pi}(s_0) = \phi_i(s_0) + \gamma\phi_i(s_1) + \gamma^2\phi_i(s_2) + \cdots$$

In general, $\hat{V}_i^{\pi}(s_0)$ would be the average over the empirical returns of m such trajectories.[2] Then, for any setting of the α_is, a natural estimate of $V^{\pi}(s_0)$ is:

$$\hat{V}^{\pi}(s_0) = \alpha_1\hat{V}_1^{\pi}(s_0) + \cdots + \alpha_d\hat{V}_d^{\pi}(s_0) \qquad (11)$$

III. IRL from Sampled Trajectories

Algorithm:

- Obtain value estimates for the assumed optimal policy π^* and a randomly chosen policy π_1.
- From the set of policies $\{\pi_1,\ldots,\pi_k\}$, the goal is to find a setting of the coefficients α_i in the linear function-approximator class, such that the resulting reward function satisfies

$$V^{\pi^*}(s_0) \geq V^{\pi_i}(s_0), \quad i = 1, \ldots, k \qquad (12)$$

- Modify the objective function to maximize the penalized difference between the value estimates under the assumed optimal policy (π^*) and each policy in the current set.

$$\begin{aligned} \text{maximize} \quad & \textstyle\sum_{i=1}^{k} p\left(\hat{V}^{\pi^*}(s_0) - \hat{V}^{\pi_i}(s_0)\right) \\ \text{s.t.} \quad & |\alpha_i| \leq 1, \quad i = 1, \ldots, d \end{aligned} \qquad \text{where} \quad p(r) = \begin{cases} r & \text{if } r \geq 0 \\ 2r & \text{otherwise} \end{cases}$$

III. IRL from Sampled Trajectories

Algorithm:

- Solve this linear programming problem to obtain a new setting of coefficients α_i, and consequently, a new reward function $R=\alpha_1\phi_1+\alpha_2\phi_2+...\alpha_d\phi_d$
- Find a new policy π_k+1 that maximizes $V^*(s_0)$ under the newly obtained reward function R.
- Add π_k+1 to the current set of policies.
- Repeat the inductive step for a large number of iterations until a satisfactory reward function is found.

Continuous Grid World

Experiment 3: Continuous Grid World

- In this experiment, the continuous version of the 5x5 grid world was considered, with the state space being $[0, 1] \times [0, 1]$.
- The agent could take actions corresponding to the four compass directions, but with the added complexity of noise and truncation to keep the state within the unit square.
- The true reward was set to 1 in a specific square ($[0.8, 1] \times [0.8, 1]$) and 0 everywhere else, with a discount factor (γ) of 0.9.
- The reward function was modeled using a function approximator class consisting of all linear combinations of a 15×15 array of two-dimensional Gaussian basis functions. This choice allowed for a flexible representation of the reward function.
- The initial state distribution D was uniform over the state space, indicating that the agent could start from any point in the $[0, 1] \times [0, 1]$ region.

Experiment 3: Continuous Grid World

- The algorithm was a sample-based approach, and it was run using 5000 trajectories, each consisting of 30 steps, to evaluate each policy. The MDP was solved when needed, based on a 50×50 discretization of the state space.
- The performance of the algorithm was evaluated by comparing the optimal policy derived from the fitted reward function with the true optimal policy. The comparison included calculating the fraction of the state space where their action choices disagreed.
- The algorithm typically produced reasonable solutions after just one iteration. By about 15 iterations, the algorithm had usually settled on fairly good solutions. Discrepancies between the fitted reward's optimal policy and the true optimal policy were found to be between 3% and 10%, which is expected due to the presence of many distinct near-optimal policies.

Experiment 3: Continuous Grid World

- The quality of the fitted reward's optimal policy was compared with the quality of the true optimal policy, with quality being measured using the true reward function.
- After about 15 iterations, evaluations were unable to detect a statistically significant difference between the value of the true optimal policy and the value of the fitted reward's optimal policy.

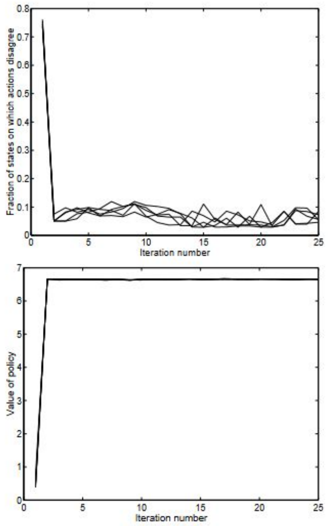
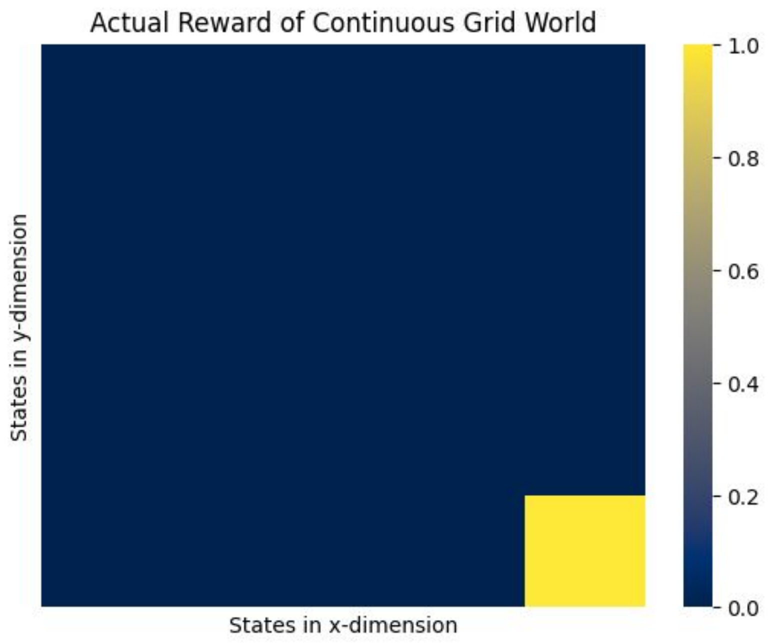


Figure 5. Results on the continuous grid world, for 5 runs. *Top*: Fraction of states on which the fitted reward's optimal policy disagrees with the true optimal policy, plotted against iteration number. *Bottom*: The value of the fitted reward's optimal policy. (Estimates are from 50000 Monte Carlo trials of length 50 each; negligible errorbars).

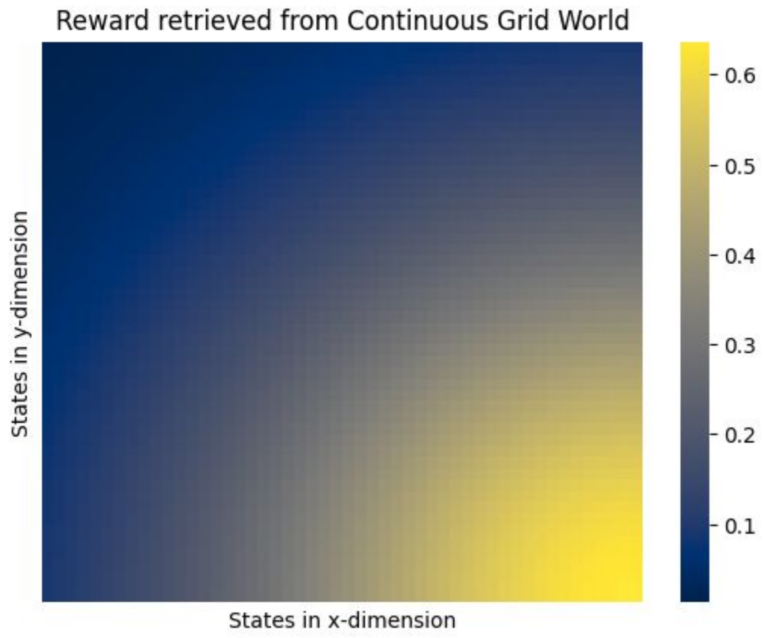
Ng, A. Y., & Russell, S. (2000). Algorithms for inverse reinforcement learning. Proc. ICML

Experiment 3: Continuous Grid World



https://github.com/Shaz-5/IRL

Experiment 3: Continuous Grid World



https://github.com/Shaz-5/IRL

Conclusion

The results indicate that the inverse reinforcement learning (IRL) problem is solvable, particularly for moderate-sized discrete and continuous domains. This implies that IRL algorithms can effectively recover reward functions from observed behavior.

Thank You

References:

1. Ng, A. Y., & Russell, S. (2000). Algorithms for inverse reinforcement learning. Proc. ICML.

2. Abbeel, P., & Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning.

3. Shakya, Ashish & Pillai, Gopinatha & Chakrabarty, Sohom. (2023). Reinforcement Learning Algorithms: A brief survey. Expert Systems with Applications. 231. 120495. 10.1016/j.eswa.2023.120495.

4. ShivinDass. (n.d.). GitHub - ShivinDass/inverse_rl: Implementing the two pioneering IRL papers "Algorithms for Inverse Reinforcement Learning" - (Ng & Russell 2000) and "Maximum Entropy Inverse Reinforcement Learning" - (Ziebart et al. 2008). GitHub. https://github.com/ShivinDass/inverse_rl