Inverse Reinforcement Learning Using Policy Gradient Minimization

Nirvan Singhania, Gulshan Kumar, Aman Bansal

November 26, 2019

1 Basics of inverse reinforcement learning

1.1 RL vs IRL

Inverse reinforcement learning (IRL) is the field of learning an agent's objectives, values, or rewards by observing its behavior.

In RL, our agent is provided with a reward function which, whenever it executes an action in some state, provides feedback about the agent's performance. This reward function is used to obtain an optimal policy, one where the expected future reward (discounted by how far away it will occur) is maximal.

In IRL, the setting is (as the name suggests) inverse. We are now given some agent's policy or a history of behavior and we try to find a reward function that explains the given behavior. Under the assumption that our agent acted optimally, i.e. always picks the best possible action for its reward function, we try to estimate a reward function that could have led to this behavior.

	IRL	**RL**
given	policy π or history sampled from that policy	(partially observed) reward function $\mathcal{R}$
searching	reward function $\mathcal{R}$ for which given behavior is optimal	optimal policy π for given reward

Figure 1.1: RL vs IRL

1.2 Why IRL

In most reinforcement learning tasks there is no natural source for the reward signal. Instead, it has to be hand-crafted and carefully designed to accurately represent the task. Often, engineers manually tweak the rewards of the RL agent until desired behavior is observed. A better way of finding a well fitting reward function for some objective might be to observe a (human) expert performing the task in order to then automatically extract the respective rewards from these observations.

In particular, the reward is a more succinct and powerful information than the policy itself. The reward function can be used to generalize the expert′s behavior to state space regions not

covered by the samples or to new situations. For instance, the transition model can change and, as a consequence, the optimal policy may change as well.

2 Main approach in paper

The authors observe the behavior of an expert that follows a policy π^E that is optimal w.r.t. some reward function R^E. They assume that R^E can be represented through a linear or non-linear function $R(s,a;\omega)$, where $\omega \epsilon R^q$

$$J_D(\pi) = \int_S d_\mu^\pi(s) \int_A \pi(a;s) R(s,a;\omega) da ds \tag{2.1}$$

2.1 Aim of Gradient Inverse Reinfocement Learning

Given a parametric (linear or non linear) reward function $R(s,a;\omega)$,the associate policy gradient can be computed-

$$\Delta_\theta J(\pi_\theta^E, \omega) = \int_S d_\mu^\pi(s) \int_A \Delta_\theta \pi^E(a;s;\theta) Q^{\pi^E}(s;a;\omega) da ds \tag{2.2}$$

If the policy performance $J(\pi,\omega)$ is differentiable w.r.t. the policy parameters θ and the expert π_θ^E is optimal w.r.t. a parametrization $R(s,a,\omega^E)$, the associated policy gradient is identically equal to zero. Clearly, the expert′s policy π_θ^E is a stationary point for $J(\pi,\omega^E)$. The Gradient IRL (GIRL) algorithm aims at finding a stationary point of $J(\pi_\theta^E,\omega)$ w.r.t. the reward parameter ω, that is, for any $x, y \geq 1$

$$\omega^A = argmin_w C_x^y(\pi_\theta^E,\omega) = argmin_w \frac{1}{y} \left\| \Delta_\theta J(\pi_\theta^E,\omega) \right\|_x^y \tag{2.3}$$

Key property of GIRL is its **property of convexity**, which is state by the following lemma

Given a convex representation of the reward function $R(s,a;\omega)$ w.r.t. the reward parameters, the objective function C_x^y, with $x, y \geq 1$, is convex.

3 Pareto Opimality

3.1 Definition

Consider the case with $N = 2$ agents, indexed by $i = 1,2$. Most of what we consider here is generalizable for larger N.. Let agent $1's$ utility depends on his own action $a1$("action" is defined very broadly here) as well as agent $2's$ action, so we can write $U_1(a1,a2)$, and similarly for agent 2 $U_2(a1,a2)$.

The set of feasible actions (a_1^P, a_2^P) is Pareto optimal if there does not exist another set of feasible actions $(a1^{'}, a2^{'})$ such that $U_1(a1^{'}, a2^{'}) \geq U_1(a_1^P, a_2^P) and U_1(a1^{'}, a2^{'}) \geq U_1(a_1^P, a_2^P)$ with at least on above inequality strict.

3.2 FRONTIER

One can imagine the set of all pairs of utility (U_1, U_2) given by all of the different actions a_1 and a_2.The *utility possibility set* is that collection $U = \{(U_1, U_2) : U_1 = U_1(a_1, a_2), U_2 = U_2(a_1, a_2)\}$ which can usually be represented by a graph with U_1 on the $x - axis$ and U_2 on the $y - axis$. By its very nature, a Pareto optimum should be on the very edge of that set - that is its "frontier ". Hence if all points are plotted on the graph and on the basis of its coordinates a point is divided into 4 halves, there will be no point in the upper right half.

4 LINEAR REWARD SETTING

On linear parametrization of the reward function we $R(s, a, \omega)$, we have the following equation

$$J(\pi, \omega) = \sum_{i=1}^{q} \omega_i J_i(\pi) \tag{4.1}$$

where $J_(\pi) \epsilon R^q$ are the unconditional feature expectations under policy π and basis functions $\phi(s, a) \epsilon R^q$

The above equation is a weighted sum of the objective function. We are interested in searching for solutions which make the Pareto optimal. For this we have the following we have as a solution a non-convex vector α such that the equation holds

$$\alpha \epsilon Null(D_\theta J(\pi)) \tag{4.2}$$

where $(D_\theta J(\pi))$ is the Jacobian for J.

A point is Pareto−stationary if there exists a convex combination of the individual gradients that generates an identically zero vector.

$$(D_\theta J(\pi))\alpha = 0 \tag{4.3}$$

$$\sum_{i=1}^{q} \alpha_i = 1, \alpha_i \geq 0 \tag{4.4}$$

A solution that is Pareto optimal is also Pareto−stationary.

When the solution lies outside the frontier it is possible to identify a set of directions (ascent cone) that simultaneously increase all the objectives. When the solution belongs to the

Pareto frontier, the ascent cone is empty because any change in the parameters will cause the decrease of at least one objective.

The convex combination α represents the scalarization, i.e., the reward parameters. In particular, notice that $\alpha = \omega$ coincides with the solution computed by GIRL. This result follows easily by noticing that

$$||\Delta J(\pi_\theta^E, \omega)||_x = ||D_\theta J(\pi^E,)\omega||_x \tag{4.5}$$

where $J_\pi(s,a)\epsilon R^q$ is the vector of conditional feature expectations given $s_0 = s$ and $a_0 = a$.

5 Geometric Interpretation

We have seen the analytical interpretation of the reward weights. Now, if the expert is optimal w.r.t. some linear combination of the objectives, then she lies on the Pareto frontier. Geometrically, the reward weights are orthogonal to the hyper plane tangent to the Pareto frontier (identified by the individual gradients $\nabla_\theta J_i(\pi^E)$, i = (1, ...q). By exploiting the local information given by the individual gradients w.r.t. the expert parametrization $D_\theta J(\pi^E)$ we can compute the tangent hyperplane and the associated scalarization weights. Such hyperplane can be identified by the q points associated to the Gram matrix of $D_\theta J(\pi^E)$.

$$G = (D_\theta J(\pi^E))^T D_\theta J(\pi^E) \tag{5.1}$$

If G is full rank, the q points univocally identify the tangent hyperplane. Since the expert′s weights are orthogonal to such hyperplane, they are obtained by computing the null space of the matrix $G : \omega\, \epsilon$ Null(G). Given the individual gradients, the complexity of obtaining the weights is $O(q^2 d + q^3)$. This version of it is called PGIRL.

6 Analysis of Linear Reward Setting

6.1 Handling Degeneracy

The GIRL problem is ill-posed under linear reward parametrizations. Possible sources of degeneracy are, for instance, constant reward functions, duplicated features or useless features. The batch nature of the GIRL problem allows to incorporate a phase of feature preprocessing. In that phase we eliminate linearly dependent features− including also the features that are never active under the given policy and constant features.

In PGIRL, this phase is carried out on the gradient matrix. The rank of the Jacobian matrix $D_\theta J(\pi^E)$ plays a fundamental role in the reconstruction of the reward weights. We consider the policy parameters to be LI or more precisely the rows of the Jacobian matrix to be LI. As long as the policy parameters d is greater or equal than the number q of reward parameters

($q \leq d$), any deficiency in the rank is due to linear dependence among the objectives. As a consequence, the Jacobian matrix can be reduced in order to contain only columns that are linearly independent. The weights for the removed columns is set to zero for not affecting the reward partameterization.

When the Gram matrix G is not full rank we do not have a unique solution. In this case the null space operator returns an orthonormal basis Z such that $D_\theta J(\pi^E).Z$ is a null matrix.

6.2 Errors in PGIRL

When the state transition model is unknown, the gradients are estimated through trajectories using some model-free approach. The estimation errors of the gradients imply an error in the expert weights estimated by PGIRL. The following theorem gives an upper bound to the $L2$-norm of the difference between the expert weights ω_E and the weights ω_A estimated by PGIRL, given $L2$-norm upper bounds on the gradient estimation errors.

Let denote with g^i the estimated value of $\nabla_\theta J_i(\pi^E)$. Under the assumption that the reward function maximized by the expert is a linear combination of basis functions $\phi(s,a)$ with weights ω_E and that, for each i,

$||\nabla_\theta J_i(\pi^E)$ - g^i $||_2 \leq \epsilon_i$

$$||\omega_E - \omega_A||_2 \leq \sqrt{(2(1-\sqrt{1-(\epsilon/\rho)^2}} \tag{6.1}$$

where $\epsilon = \max_i \epsilon_i$ and ρ is the radius of the largest $(q-1)$- dimensional ball inscribed in the convex hull of points g^i

6.3 Approximated Expert Policy Parameters

When the expert′s policy is unknown, to apply the same idea presented in the previous section, we need to infer a parametric policy model from a set of trajectories $\{\tau_i\}_{i=0}^N$ of length M. This problem is a standard density estimation problem. Given a parametric policy model $\pi(a;s,\theta)$ a parametric density estimation problem can be defined as a Maximum Likelihood estimation (MLE) problem : $\widehat{\theta}_{MLE} = argmax_\theta \prod_{i=1}^{N.M} \pi(a_i;s_i,\theta)$.

7 Experiments

7.1 Paper Experiment

Linear Quadratic Regulator: They provide a set of experiments in the well−known Linear Quadratic Regulator (LQR) problem. Experiments are meant to be a proof of concept of algorithm behavior. We consider a linear parametrization of the reward function :

$R(s,a;\omega) = -\sum_{i=1}^{q} \omega_i (s^T Q_i s + a^T R_i a)$

Expert Policy Parameters: In the PGIRL approach the gradient directions are computed using the eNAC algorithm (eNAC−PGIRL) in the 2D LQR domain. As the number of samples increases, the accuracy of the plane identified by the algorithm improves. With 1, 000 trajectories, the plane is almost tangent to the Pareto frontier. The points on the planes are obtained from the Gram matrix (after a translation from the origin).

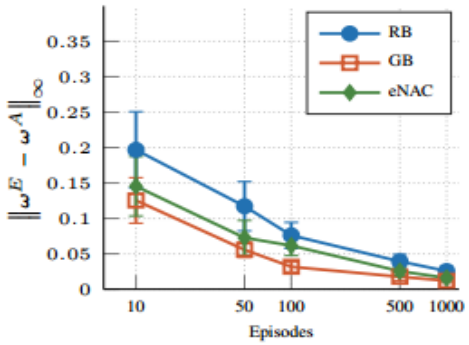

Figure 7.1: The L_∞ norm between the expert′s and agent′s weights using 2D LQR

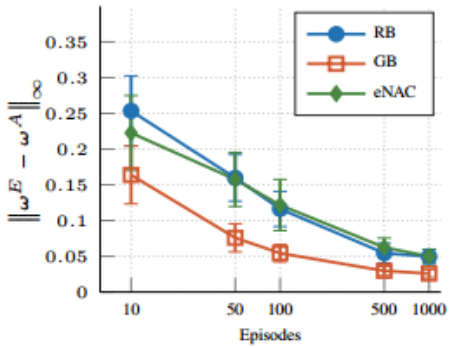

Figure 7.2: The L_∞ norm between the expert′s and agent′s weights using 5D LQR

7.2 Our Results

Due to the complexity in the algorithm we couldn't evaluate this paper on some environment. We have written the code for sampling the expert trajectory for feeding into the network and also write the code for reward parameter estimation.

Apart from this we have simulate the *Maximum Entropy Inverse Reinforcement Learning* paper. We have evaluated the model on **MountainCar-v0**. The model is trained for 30000

episodes with $\gamma = 0.95$, $\eta = 0.05$. The states for the experiment is taken 400 i.e velocity of the mountain car between -20 to 20. The number of action is 3 (left, right, fixed).

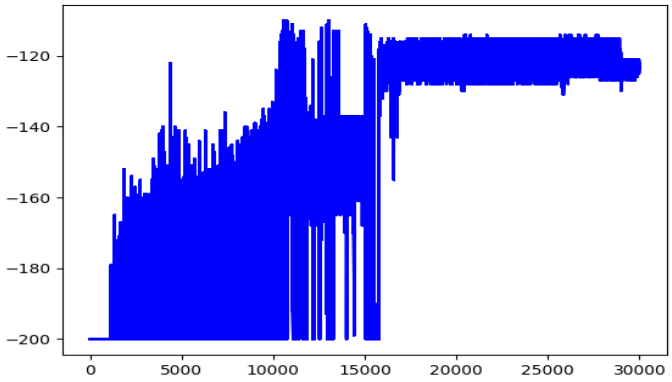

Figure 7.3: Showing the Reward Vs Episode on MounatinCar-v0 using Maximum Entropy Inverse Reinforcement Learning

8 Document Links

- **Presentation Link** : https://rebrand.ly/pgirl
- **Codebase** : https://rebrand.ly/pgirl-codebase
- **Demo**: MountainCar-v0 using Maximum entropy IRL

9 Work Distribution

- Nirvan Singhania
  - Code for Reward parameters in GIRL
  - Results of maximum entropy IRL
  - Report
- Gulshan Kumar
  - Code for Expert Policy
  - Train the maximum entropy IRL
  - Report
- Aman Bansal
  - Code for testing maximum entropy IRL
  - Presentation

10 REFERENCES

[1] Matteo Pirotta, Marcello Restelli,Inverse Reinforcement Learning through Policy Gradient Minimization, Politecnico di Milano, Piazza Leonardo da Vinci, 32 I-20133, Milan, Italy
[2] https://github.com/albertometelli/crirl
[3] Brian D. Ziebart, Andrew Maas, J.Andrew Bagnell, Anind K. Dey, Maximum Entropy Inverse Reinforcement Learning
[4] https://ai.stanford.edu/ ang/papers/icml00-irl.pdf