Tackling Degeneracy in Inverse Reinforcement Learning: Evaluation and a Novel Combination

Evgeny Maksakov *maksakov@cs.ubc.ca*
Sancho McCann *sanchom@cs.ubc.ca*
Marcus Rohrbach *rohrbach@cs.ubc.ca*

Abstract

Inverse reinforcement learning (IRL) is under-constrained: many reward functions may lead to the same optimal behaviour. An observer of an expert's actions must assume a bias in order to estimate this reward function. In this paper we review three approaches to this problem — two of them in depth. Apprenticeship learning via IRL tackles this problem with the goal of learning to act like the expert by assuming that the reward function being optimized is a linear combination of sensory input. Bayesian inverse reinforcement learning (BIRL) assumes only a preference based on prior knowledge regarding the probability of certain reward functions. Lastly there is work from the field of econometrics focused on structural estimation of Markov decision processes that is yet to be applied in this domain. We present an overview of these methods and discuss implementations and comparisons of the first two. We also discuss the possibility of a novel combination of apprenticeship learning and the BIRL approaches.

1 Introduction

Often, an agent is expected to learn to act optimally in an environment where the reward function can not be observed. Instead, the agent may be able to observe an expert that chooses actions optimally with respect to a reward function (which does not necessarily need to be known explicitly). The goal of inverse reinforcement learning (IRL) is to infer this underlying reward function.

As an example, consider a robot learning to walk based on a human's demonstration. The robot observer is provided with some of the relevant sensory input that the human receives, such as incline, surface texture, and its centre of gravity. Along with each sensory input, the robot is also shown the path of the human's limbs. From these inputs, the robot is to learn the reward function that the human is optimizing while walking.

The difficulty of IRL is degeneracy, i.e. in non-trivial cases there are a large number of reward functions leading to the same optimal policy guiding the expert [2]. However, the expert's actions are all that the agent can observe.

One approach, when we are only interested in the policy and not the actual reward function, is presented by Abbeel & Ng [1]. This task of learning is normally referred to as *apprenticeship learning*. We discuss the ideas and assumptions of this approach (IRL apprenticeship learning) in section 2.1.

Another approach is to learn a distribution over the reward function rather than a single reward vector. In section 2.2 we take a look at the properties of this Bayesian reinforcement learning (BIRL) approach suggested by Ramachandran & Amir [3].

If the available number of expert trajectories (examples of the expert moving through the world) is very small, then BIRL can not learn the reward function distribution. In section 2.3 we explore one idea that attempts to combine IRL apprenticeship learning with BIRL to overcome this problem.

The previous approaches assume that the agent can fully observe the state space, but this is often not the case. In section 2.4 we will describe how Rust [4] addresses states that are not seen by the observer.

After the discussion of these approaches we evaluate IRL apprenticeship learning, BIRL, and our combined approach on a grid world in section 3. We evaluate each individually for different settings and compare them to each other. In the comparison we explore how the algorithm can handle deviations from their original assumptions. We conclude with directions for future work.

2 Algorithms and Implementation Details

Following the description from Ramachandran and Amir [3] we define a *Markov Decision Problem* as tuple (S, A, T, γ, R) where $S = \{s_1, \ldots, s_N\}$ is a finite set of N states; $A = \{a_1, \ldots, a_k\}$ is a set of k actions; $T : S \times A \times S \to [0,1]$ is a transition probability function; $\gamma \in [0, 1)$ is the discount factor; $R : S \to \mathbb{R}$ is a reward function, with absolute value bounded by R_{max}.

Further we have a *policy* $\pi : S \to A$ and define the *value* V of a policy π by

$$V^{\pi}(s, R) = R(s) + \gamma \sum_{s_{t_{i+1}}} P(s_{t_{i+1}} | s_{t_i}, \pi(s_{t_i})) V^{\pi}(s_{t_{i+1}})$$

where $p(s_{t_{i+1}} | s_{t_i}, a) = T(s_{t_i}, a, s_{t_{i+1}})$; and the $Q\text{-}function$:

$$Q^{\pi}(s, a, R) = R(s) + \gamma \sum_{s_{t_{i+1}}} P(s_{t_{i+1}} | s_{t_i}, a) V^{\pi}(s_{t_{i+1}}).$$

These are the standard Bellman Equations relating R, T, π, and γ to the valuation of states and state-action pairs [2].

2.1 Apprenticeship Learning via IRL

The idea of apprenticeship learning is that in many cases the actual reward function is not needed and it is sufficient to learn the policy. To learn the policy from observing the expert's trajectories Abbeel and Ng [1] suggest to use IRL.

To simplify the general IRL problem, they assume that the true reward function is linearly dependent on a set of known features: $R^*(s) = w^* \cdot \phi(s)$ where $w^* \in \mathbb{R}^l$ and $\phi : S \to [0, 1]^l$. For example, the reward function for walking might only be dependent on how quickly we move and if we stay in upright position.

Abbeel and Ng define *feature expectations* for a policy π as:

$$\mu(\pi) = E[\Sigma_{t=0}^{\infty} \gamma^t \phi(s_t) | \pi] \in \mathbb{R}^l$$

For the expert's feature expectation $\mu_E = \mu(\pi_E)$ this can be estimated using the given set of m trajectories over states. Our goal is then to find a policy $\tilde{\pi}$ which performs similarly to the unknown expert's optimal policy π^* on the *unknown* reward function. This can be solved by finding $\tilde{\pi}$ such that $||(\mu(\tilde{\pi}) - \mu_E)||_2 \leq \epsilon$ [1].

Abbeel and Ng's apprenticeship learning algorithm initializes this $\tilde{\pi}$ with a random π_0, $\mu^{(0)} = \mu(\pi_0)$, and $i = 0$. We iterate until the optimizer $t^{(i)} \leq \epsilon$:

The optimizer can be found with support vector machines. We use a simpler *projection method*, described in the same paper: we make a projection orthogonal to the last update direction, towards the expert's feature expectation μ_E. The weight $w^{(i)}$ is then set to the difference of μ_E and the previous projection point. We set $t^{(i)} = ||w^{(i)}||_2$.

Then we compute the current optimal policy $\pi^{(i)}$ with $R = (w^{(i)})^T \phi$ using forward reinforcement learning. After updating $\mu^{(i)} = \mu(\pi^{(i)})$ and $i = i + 1$, we continue the iteration.

After the algorithm terminates, we have learned a policy $\tilde{\pi}$ which is guaranteed to match the feature expectation of the expert with a maximum error of ϵ.

To calculate the feature expectation $\mu(\pi)$, we can truncate the sum after a finite number of steps because after a large number of steps the next summands are negligible due to the discount factor $\gamma < 1$ [1].

2.2 Bayesian Inverse Reinforcement Learning

Bayesian inverse reinforcement learning (BIRL) avoids degeneracy and ill-posedness by using prior knowledge about the reward function and observations of the expert's actions to learn a distribution over the rewards. Whereas the previously discussed algorithm enforces a *model* bias by requiring the implicit reward function guiding the policy to be a linear function of observed features, BIRL provides a *preference* bias. This allows less restricted learning of a variety of reward functions. As mentioned above, the input to this algorithm is the observation O_χ of the expert's behaviour in the form of a set of trajectories of state-action pairs.

We now present the Bayesian formulation of Ramachandran and Amir [3]. The posterior distribution is given by $P(R|O_\chi) \propto P(O_\chi|R)P(R)$. The likelihood of the set of observed actions given a particular reward estimate is assumed to be the product of the likelihoods of the individual observed state-action pairs: $P(O_\chi|R) = \prod_{i=1}^{k} P(s_i, a_i|R)$. They note that the larger $Q(s, a)$ is, the higher the state-action likelihood and model this as an exponential distribution: $P(s_i, a_i|R) \propto e^{\alpha_\chi Q(s_i, a_i, R)}$.

We cannot sample directly over the full posterior $P(R|O_\chi) \propto e^{\alpha_\chi \sum_i Q(s_i, a_i, R)} P(R)$, so an MCMC sampling algorithm is needed to determine the mean of this posterior.

The mean of the posterior is the value that minimizes the expected squared error of the reward function. Also, minimizing the expected policy loss over the posterior distribution of R is equivalent to finding the optimal policy on the mean reward function.[3]

An MCMC algorithm *PolicyWalk* generates a Markov chain on the posterior of the reward function, maintaining also the optimal policy for the current reward sample. At each step in the Markov chain, a proposal reward sample $\tilde{R}$ is chosen from within a hyper-sphere neighborhood of the current reward sample $\tilde{R}$. If $\tilde{R}$ would cause a change in the optimal policy π, then $\tilde{R}$ along with the new optimal policy $\tilde{\pi}$ are accepted according to a *Metropolis filter* [5] with probability

$min(1, \frac{P(\tilde{R},\tilde{\pi})}{P(R,\pi)})$. If choosing $\tilde{R}$ would not cause a change in π, it is accepted with probability $min(1, \frac{P(\tilde{R},\pi)}{P(R,\pi)})$. ($P(R,\pi)$ is the posterior evaluated using R and the Q-function that is determined by R and π.)

2.3 Combined Approach

We speculated that a combination of the apprenticeship learning via IRL and BIRL would produce better results than BIRL alone. This combination assumes access to O_χ and the features ϕ observed by the expert at each state. We believed that the observer using BIRL would perform better if it was informed by the policy over the entire environment rather than only information about a close to optimal set of trajectories through the environment.

To give the observer this information, we would first allow the observer to generalize the policy to encompass the entire state space by using IRL apprenticeship learning to learn a linear mapping from features to rewards. These rewards would be constrained only by the linear model and not informed by any preference bias or prior knowledge. Next, the observer would *imagine* how the expert would have hypothetically have acted in the states in which actions were not initially observed. In combination with these imagined observations, the observer using BIRL could apply prior knowledge about the likely reward distribution to estimate the posterior distribution as in the standard BIRL algorithm. Results of this combination are provided in section 3.

2.4 Nested Fixed-Point Maximum Likelihood

All previous described methods assume that the observer is able to see the entire state space of the expert. However, in practice it is often not possible for the observer to perceive the entire state space the expert is operating on. For example, the robot observing only the paths of the expert's limbs cannot take into account the hidden variable of energy efficiency.

Rust [4] discusses this problem of unobserved state variables in the discrete case. R and T are assumed to depend on an unknown vector of parameters θ. This IRL problem (i.e. finding R) can be solved by finding $\widehat{\theta}$ that maximizes the likelihood for the observed trajectories O_χ. Rust presents a *nested fixed-point* algorithm to solve the problem as we do not know how to express it as a function in general. Under several assumptions he shows that $R_{\widehat{\theta}}$ converges to the true reward R_{θ^*} with probability 1 as the number or the length of O_χ approaches infinity. Rust formulates 35 assumptions (A1 - A35) which have to hold.

We describe the first two, which are the most important ones:

We have the parameters $\theta = (\theta_1, \theta_2)$ (unknown to the observer) as well as the disjoint sets of observed states S_O and unobserved states S_U which form the entire state space S (lower case letters represent the elements of the respective sets):

(A1) The reward function R must be separable:

$$R(s_O, s_U, a) = R_O(s_O, a, \theta_1) + R_U(s_U, a)$$

(A2) For the unobserved reward components $\mathcal{R}_U = \{R_U(s_U, a) | a \in A\}$, $\{s_O^{(t)}, \mathcal{R}_U^{(t)}, a^{(t)}\}$ defines a Markov process with the probability density p:

$$p(s_O^{(t+1)}, \mathcal{R}_U^{(t+1)} | s_O^{(t)}, \mathcal{R}_U^{(t)}, a^{(t+1)}, \theta_2)$$

The algorithm to calculate $\widehat{\theta}$ and the associated value function $V_{\widehat{\theta}}$ consists of two parts: an *"inner" contraction fixed-point algorithm* and an *"outer" hill-climbing algorithm*. The fixed point algorithm retrieves V_θ for each θ by running a contraction iteration to get close to V_θ and then the Newton/Kantorovich iteration to converge to the solution. The hill-climbing algorithm searches in an outer loop for $\widehat{\theta}$ by using the BHHH algorithm which is similar to the Gauss-Newton gradient maximization algorithm with an approximation of the Hessian.

For this project, we have translated the main nested-fixed point algorithm from the Gauss Matrix Programming Language into MATLAB code. This translation is not complete and we are still verifying the correctness of our translation.

2.5 Further Implementation Details

To test and compare the algorithms we chose the grid world model with four possible transitions from each state to the neighboring states.

Before starting any learning we need to generate the expert's behavior for the grid walking task. Consequently, we decided to train the expert the optimal policy. Our expert learns it using forward reinforcement learning.

The programming language is MATLAB and we used the Kevin Murphy's *Markov Decision Processes* toolbox[1] to generate the world, run policy iteration, value determination, and Q-function evaluation.

3 Experimental Evaluation

In these evaluations, we compare both policy loss and reward loss. For policy loss, we use the L_2-norm of the difference between the value of the learned policy, and the value of the optimal policy with respect to the true reward function. For reward loss, we use the L_2-norm of the difference between the learned reward function and the true reward function.

3.1 Apprenticeship Learning via IRL

For evaluation of apprenticeship learning, we use a test similar to the one described by Abbeel and Ng in [1]. We use a 32-by-32 grid world with distinctive features. The expert and observer both have the same interaction with the environment: moves in the four cardinal directions, with a success probability of 0.8 (otherwise, the agent moves in a direction perpendicular to the intended direction).

The features provided in this world are vectors indicating which of 64 macrocells the agent is occupying. The macrocells are 4-by-4, non-overlapping regions. We assigned random weights to 10 of the 64 macrocell features to define the reward function (the remainder had zero reward).

In figure 1 we plot the improvement of the feature expectation approximation achieved by the observer over three iterations of the algorithm. We additionally show the evolution of the policy loss associated with the policy implied at each iteration of the algorithm. Oscillation of the policy loss may be explained by two factors. In attempting to match the feature expectation shown by the expert, the observer may try paths that alternately include or exclude macrocells even though they yield significant rewards. These macrocells may not have a significant contribution to the feature expectation displayed by the expert because the expert always

[1] Matlab Toolbox available from www.cs.ubc.ca/ murphyk/Software/MDP/mdp.html

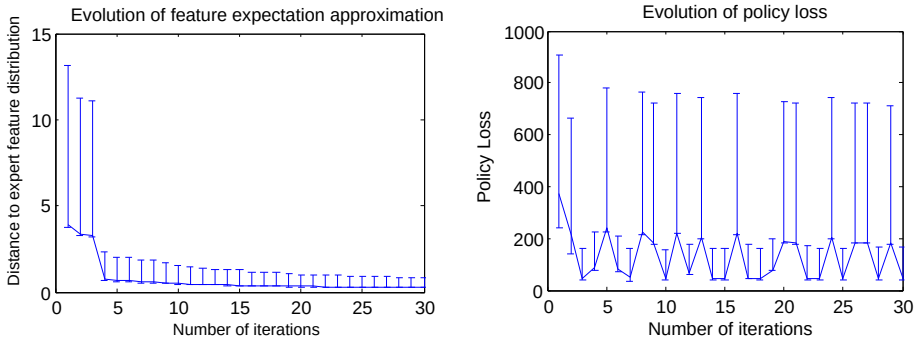

Figure 1: A plot of the evolution of the feature expectation approximation distance and policy loss in IRL apprenticeship learning. The mean, min, and max of three runs are shown.

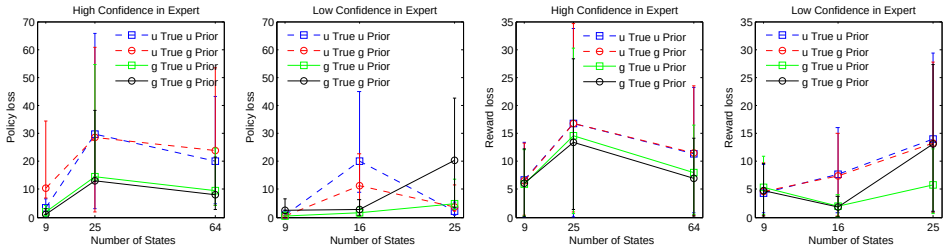

Figure 2: Effects of different combinations of the true prior distribution of rewards and the assumed prior distribution in the Bayesian model (u = uniform; g = Gaussian). Results are included for both high (7, over-confidence) and low confidence (0.8) in the expert. Mean, min and max of three runs are shown.

starts at the same corner of the world, rather than distributing the initial state uniformly over the world. Regardless of the reasons it is clear from this experiment that an improvement in the feature expectation does not necessarily improve the policy loss.

3.2 BIRL

The quality of results obtained from BIRL depends on the mixing of the MCMC walk. It is important to generate enough samples to approximate the posterior, and to not accumulate samples before a sufficient number of steps have been taken to ensure that the samples are being drawn from the posterior. By inspection of the reward sample evolution, we chose to generate 100.000 samples, but to evaluate the mean based on samples 40.000 to 100.000. The 100.000 upper threshold was confirmed by correspondence with the author as the value used in their experiments.

We examined the effect of changing the true distributions of rewards. As can be seen in figure 2, the results are not conclusive enough to make any strong claims, but the general trend seems to indicate that the performance of BIRL is lower when the true rewards are drawn from a uniform prior than when they are drawn from a Gaussian prior.

Further evaluation of this algorithm is presented in the next section where we show results of the individual algorithms and their combination on a simple grid world.

		Chaotic world		Nice world	
		μ	σ	μ	σ
policy loss	IRL-AL	2.20	0.14	0.11	0.08
	BIRL	1.81	0.49	0.15	0.09
	Combo	2.15	0.34	0.14	0.10
reward loss	BIRL	0.43	0.02	0.25	0.02
	Combo	0.52	0.02	0.34	0.02

Table 1: Performance of the three approaches on two different worlds

3.3 Combined Approach and Comparison

For a closer comparison of these algorithms and their combination, we performed experiments on 4-by-4 grid worlds. The feature vectors indicate which of four non-overlapping macrocells, each 2-by-2, the expert or observer is in. We tested the algorithms on two worlds: a chaotic world, and a nice world. In the chaotic world, rewards at each *state* are drawn from a uniform distribution. In the nice world, rewards within a single *quadrant* are identical, drawn from a uniform distribution. In the latter case, the rewards are exactly a linear combination of the features, the assumption made by IRL apprenticeship learning. The input to the tests was the set of trajectories taken by the expert. Our prediction was that in the chaotic world, BIRL would outperform IRL apprenticeship learning (whose assumptions are violated), while in the nice world, IRL apprenticeship learning would perform better. We also hypothesized that the combination of the two as described in section 2.3 would outperform BIRL alone.

Results favour the conclusions that BIRL does perform better than IRL apprenticeship learning when the assumption of linearity from feature to reward is violated (table 3.3,row 1,2) . IRL apprenticeship learning does not display a disadvantage when the linearity assumption is met. Our results do not support our hypothesis that the combination of apprenticeship learning and BIRL would perform better than BIRL alone.

We performed one final test: an evaluation of how performance degrades when transition probabilities of the expert and observer differ. We call the scenarios *clumsy expert* and *clumsy observer*. To model a clumsy observer or expert, we reduce the probability of moving in the direction prescribed by policy to 0.7 while the non-clumsy participant moves correctly with probability 0.98. The results from these tests are shown in figure 3. Unfortunately, they seem too variable to make any conclusions.

4 Future work

Our project gives an overview of the approaches to the inverse reinforcement learning problem. There is still much room for further research. It was not clear from the presentation of Ramachandran and Amir how one should select or learn the confidence parameter in the Bayesian model for IRL. Which of these methods generalizes best to a changing environment? Which can best incorporate states or regions where the true reward *is* known? How can these methods be applied to observing agents with differing transition functions than the expert? Can the approach of feature expectation approximation be applied to a problem where the rewards consist of a *non*-linear combination of the features? How does the structural estimation work from the field of econometrics transfer into this domain? These are just some of the interesting questions that can be explored in future work.

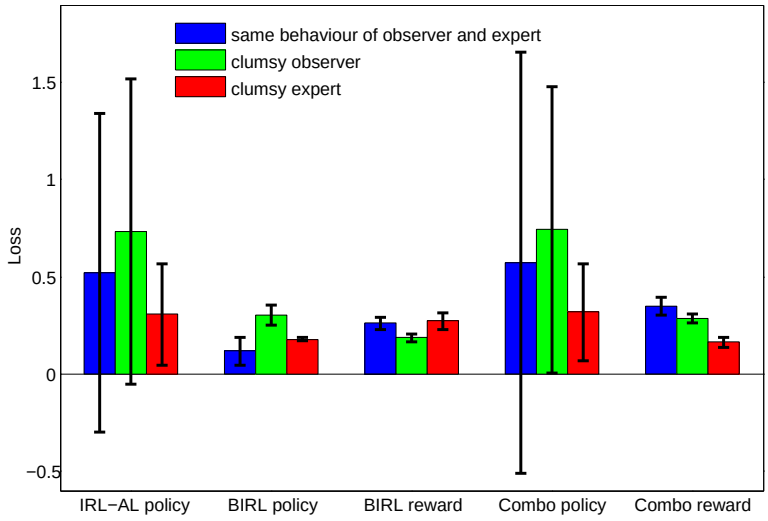

Figure 3: Losses displayed by the three methods when 1) the observer and expert both have the same transition probabilities, 2) when the observer is "clumsy", and 3) when the expert is "clumsy". Mean and σ-error bars for 5 runs are shown.

5 Conclusion

The described algorithms take different approaches to handling the degeneracy of inverse reinforcement learning. The work of Abbeel and Ng on IRL apprenticeship learning enforces a model bias while Ramachandran and Amir's Bayesian IRL enforces a preference bias. When the model assumptions are satisfied, IRL apprenticeship learning displays its advantage, but degrades noticeably when the assumptions its not met. The Bayesian approach can learn a wider variety of reward functions. A combination of the two has been introduced, but has yet to be developed fully.

References

[1] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In Carla E. Brodley, editor, *Machine Learning, Proceedings of the Twenty-first International Conference*. ACM, 2004.

[2] Andrew Y. Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *Proc. 17th International Conf. on Machine Learning*, pages 663–670. Morgan Kaufmann, San Francisco, CA, 2000.

[3] Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In Manuela M. Veloso, editor, *Proceedings of the 20th International Joint Conference on Artificial Intelligencer*, pages 2586–2591, 2007.

[4] John Rust. Maximum likelihood estimation of discrete control processes. *SIAM Journal on Control and Optimization*, 26(5):1006–1024, 1988.

[5] Santosh Vempala. *Geometric Random Walks: A Survey. In Combinatorial and Computational Geometry*, volume 52, pages 573–612. Cambridge University Press, 2005.