Bayesian Inverse Reinforcement Learning

Deepak Ramachandran
Computer Science Dept.
University of Illinois at Urbana-Champaign
Urbana, IL 61801

Eyal Amir
Computer Science Dept.
University of Illinois at Urbana-Champaign
Urbana, IL 61801

Abstract

Inverse Reinforcement Learning (IRL) is the problem of learning the reward function underlying a Markov Decision Process given the dynamics of the system and the behaviour of an expert. IRL is motivated by situations where knowledge of the rewards is a goal by itself (as in preference elicitation) and by the task of apprenticeship learning (learning policies from an expert). In this paper we show how to combine prior knowledge and evidence from the expert's actions to derive a probability distribution over the space of reward functions. We present efficient algorithms that find solutions for the reward learning and apprenticeship learning tasks that generalize well over these distributions. Experimental results show strong improvement for our methods over previous heuristic-based approaches.

1 Introduction

The Inverse Reinforcement Learning (IRL) problem is defined in [Russell, 1998] as follows: **Determine** The reward function that an agent is optimizing. **Given** 1) Measurement of the agent's behaviour over time, in a variety of circumstances 2) Measurements of the sensory inputs to that agent; 3) a model of the environment. In the context of Markov Decision Processes, this translates into determining the reward function of the agent from knowledge of the policy it executes and the dynamics of the state-space.

There are two tasks that IRL accomplishes. The first, *reward learning*, is estimating the unknown reward function as accurately as possible. It is useful in situations where the reward function is of interest by itself, for example when constructing models of animal and human learning or modelling opponent in competitive games. Pokerbots can improve performance against suboptimal human opponents by learning reward functions that account for the utility of money, preferences for certain hands or situations and other idiosyncrasies [Billings *et al.*, 1998]. There are also connections to various preference elicitation problems in economics [Sargent, 1994].

The second task is *apprenticeship learning* - using observations of an expert's actions to decide one's own behaviour. It is possible in this situation to directly learn the policy from the expert [Atkeson and Schaal, 1997]. However the reward function is generally the most succint, robust and transferable representation of the task, and completely determines the optimal policy (or set of policies). In addition, knowledge of the reward function allows the agent to generalize better i.e. a new policy can be computed when the environment changes. IRL is thus likely to be the most effective method here.

In this paper we model the IRL problem from a Bayesian perspective. We consider the actions of the expert as *evidence* that we use to update a prior on reward functions. We solve reward learning and apprenticeship learning using this posterior. We perform inference for these tasks using a modified Markov Chain Monte Carlo (MCMC) algorithm. We show that the Markov Chain for our distribution with a uniform prior mixes rapidly, and that the algorithm converges to the correct answer in polynomial time. We also show that the original IRL is a special case of Bayesian IRL (BIRL) with a Laplacian prior.

There are a number of advantages of our technique over previous work: We do not need a completely specified optimal policy as input to the IRL agent, nor do we need to assume that the expert is infallible. Also, we can incorporate external information about specific IRL problems into the prior of the model, or use evidence from multiple experts.

IRL was first studied in the machine learning setting by [Ng and Russell, 2000] who described algorithms that found optimal rewards for MDPs having both finite and infinite states. Experimental results show improved performance by our techniques in the finite case.

The rest of this paper is organised as follows: In section 2 we define our terms and notation. Section 3 presents our Bayesian model of the IRL process. Section 4 discusses how to use this model to do reward learning and apprenticeship learning while section 5 discusses the sampling procedure. Sections 6, 7 and 8 then present experimental results, related work and our conclusions respectively.

2 Preliminaries

We recall some basic definitions and theorems relating to Markov Decision Processes and Reinforcement Learning.

A (finite) *Markov Decision Problem* is a tuple (S, A, T, γ, R) where

- S is a finite set of N *states*.
- $A = \{a_1, \ldots, a_k\}$ is a set of k *actions*.

- $T : S \times A \times S \mapsto [0, 1]$ is a *transition probability function*.
- $\gamma \in [0, 1)$ is the *discount factor*.
- $R : S \mapsto \mathbb{R}$ is a *reward function*, with absolute value bounded by R_{max}.

The rewards are functions of state alone because IRL problems typically have limited information about the value of action and we want to avoid overfitting.

A *Markov Decision Process* (MDP) is a tuple (S, A, T, γ), with the terms defined as before but without a reward function. To avoid confusion we will use the abbreviation MDP only for Markov Decision Processes and not Problems.

We adopt the following compact notation from [Ng and Russell, 2000] for finite MDPs : Fix an enumeration $s_1 \dots s_N$ of the finite state space S. The reward function (or any other function on the state-space) can then be represented as an N-dimensional vector $\boldsymbol{R}$, whose ith element is $R(s_i)$.

A (stationary) *policy* is a map $\pi : S \mapsto A$ and the (discounted, infinite-horizon) *value* of a policy π for reward function $\boldsymbol{R}$ at state $s \in S$,denoted $V^\pi(s, \boldsymbol{R})$ is given by:

$V^\pi(s_{t_1}, \boldsymbol{R}) = R(s_{t_1}) + E_{s_{t_2}, s_{t_2}, \dots}[\gamma R(s_{t_2}) + \gamma^2 R(s_{t_3}) + \dots | \pi]$

where $Pr(s_{t_{i+1}} | s_{t_i}, \pi) = T(s_{t_i}, \pi(s_{t_i}), s_{t_{i+1}})$. The goal of standard Reinforcement Learning is to find an *optimal policy* π^* such that $V^\pi(s, \boldsymbol{R})$ is maximized for all $s \in S$ by $\pi = \pi^*$. Indeed, it can be shown (see for example [Sutton and Barto, 1998]) that at least one such policy always exists for ergodic MDPs. For the solution of Markov Decision Problems, it is useful to define the following auxilliary *Q-function*:

$$Q^\pi(s, a, \boldsymbol{R}) = R(s) + \gamma E_{s' \sim T(s,a,\cdot)}[V^\pi(s', \boldsymbol{R})]$$

We also define the optimal Q-function $Q^*(\cdot, \cdot, \boldsymbol{R})$ as the Q-function of the optimal policy π^* for reward function $\boldsymbol{R}$.

Finally, we state the following result concerning Markov Decision Problems (see [Sutton and Barto, 1998]) :

Theorem 1 (Bellman Equations). *Let a Markov Decision Problem $M = (S, A, T, \gamma, R)$ and a policy $\pi : S \mapsto A$ be given. Then,*

1. *For all $s \in S, a \in A, V^\pi$ and Q^π satisfy*

$$V^\pi(s) = R(s) + \gamma \sum_{s'} T(s, \pi(s), s') V^\pi(s') \quad (1)$$

$$Q^\pi(s, a) = R(s) + \gamma \sum_{s'} T(s, a, s') V^\pi(s')$$

2. *π is an optimal policy for M iff, for all $s \in S$,*

$$\pi(s) \in \operatorname*{argmax}_{a \in A} Q^\pi(s, a) \quad (2)$$

3 Bayesian IRL

IRL is currently viewed as a problem of infering a single reward function that explains an agent's behaviour. However, there is too little information in a typical IRL problem to get only one answer. For example, consider the MDP shown in Figure 1. There are at least three reasonable kinds of reward functions: $R_1(\cdot)$ has high positive value at s_1 (and low values elsewhere) which explains why the policy tries to return to this state, while $R_2(\cdot)$ and $R_3(\cdot)$ have high values at s_2 and s_3 respectively. Thus, a probability distribution is needed to represent the uncertainty.

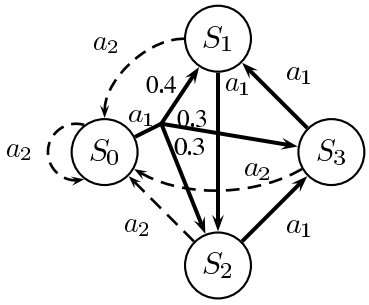

Figure 1: An example IRL problem. Bold lines represent the optimal action a_1 for each state and broken lines represent some other action a_2. Action a_1 in s_1 has probabilities 0.4,0.3 and 0.3 of going to states s_1, s_2, s_3 respectively, and all other actions are deterministic.

3.1 Evidence from the Expert

Now we present the details of our Bayesian IRL model (Fig. 2). We derive a posterior distribution for the rewards from a prior distribution and a probabilistic model of the expert's actions given the reward function.

Consider an MDP $M = (S, A, T, \gamma)$ and an agent $\mathcal{X}$ (the expert) operating in this MDP. We assume that a reward function $\boldsymbol{R}$ for $\mathcal{X}$ is chosen from a (known) prior distribution P_R. The IRL agent receives a series of observations of the expert's behaviour $O_\mathcal{X} = \{(s_1, a_1), (s_2, a_2) \dots (s_k, a_k)\}$ which means that $\mathcal{X}$ was in state s_i and took action a_i at time step i. For generality, we will not specify the algorithm that $\mathcal{X}$ uses to determine his (possibly stochastic) policy, but we make the following assumptions about his behaviour:

1. $\mathcal{X}$ is attempting to maximize the total accumulated reward according to $\boldsymbol{R}$. For example, $\mathcal{X}$ is not using an epsilon greedy policy to explore his environment.
2. $\mathcal{X}$ executes a stationary policy, i.e. it is invariant w.r.t. time and does not change depending on the actions and observations made in previous time steps.

For example, $\mathcal{X}$ could be an agent that learned a policy for $(M, \boldsymbol{R})$ using a reinforcement learning algorithm. Because the expert's policy is stationary, we can make the following independence assumption:

$$Pr_\mathcal{X}(O_\mathcal{X} | \boldsymbol{R}) = Pr_\mathcal{X}((s_1, a_1) | \boldsymbol{R}) Pr_\mathcal{X}((s_2, a_2) | \boldsymbol{R}) \dots Pr_\mathcal{X}((s_k, a_k) | \boldsymbol{R})$$

The expert's goal of maximizing accumulated reward is equivalent to finding the action for which the Q^* value at each state is maximum. Therefore the larger $Q^*(s, a)$ is, the more likely it is that $\mathcal{X}$ would choose action a at state s. This likelihood increases the more confident we are in $\mathcal{X}$'s ability to select a good action. We model this by an exponential distribution for the likelihood of (s_i, a_i), with Q^* as a potential function:

$$Pr_\mathcal{X}((s_i, a_i) | \boldsymbol{R}) = \frac{1}{Z_i} e^{\alpha_\mathcal{X} Q^*(s_i, a_i, \boldsymbol{R})}$$

where $\alpha_\mathcal{X}$ is a parameter[1] representing the degree of confidence we have in $\mathcal{X}$'s ability to choose actions with high

[1]Note that the probabilities of the evidence should be conditioned

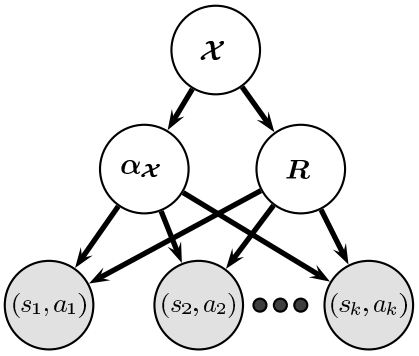

Figure 2: The BIRL model

value. This distribution satisfies our assumptions and is easy to reason with. The likelihood of the entire evidence is :

$$Pr_{\mathcal{X}}(O_{\mathcal{X}}|\boldsymbol{R}) = \frac{1}{Z} e^{\alpha_{\mathcal{X}} E(O_{\mathcal{X}}, \boldsymbol{R})}$$

where $E(O_{\mathcal{X}}, \boldsymbol{R}) = \sum_i Q^*(s_i, a_i, \boldsymbol{R})$ and Z is the appropriate normalizing constant. We can think of this likelihood function as a Boltzmann-type distribution with energy $E(O_{\mathcal{X}}, \boldsymbol{R})$ and temperature $\frac{1}{\alpha_{\mathcal{X}}}$.

Now, we compute the posterior probability of reward function $\boldsymbol{R}$ by applying Bayes theorem,

$$\begin{aligned} Pr_{\mathcal{X}}(\boldsymbol{R}|O_{\mathcal{X}}) &= \frac{Pr_{\mathcal{X}}(O_{\mathcal{X}}|\boldsymbol{R}) P_R(\boldsymbol{R})}{Pr(O_{\mathcal{X}})} \\ &= \frac{1}{Z'} e^{\alpha_{\mathcal{X}} E(O_{\mathcal{X}}, \boldsymbol{R})} P_R(\boldsymbol{R}) \qquad (3) \end{aligned}$$

Computing the normalizing constant Z' is hard. However the sampling algorithms we will use for inference only need the ratios of the densities at two points, so this is not a problem.

3.2 Priors

When no other information is given, we may assume that the rewards are independently identically distributed (i.i.d.) by the principle of maximum entropy. Most of the prior functions considered in this paper will be of this form. The exact prior to use however, depends on the characteristics of the problem:

1. If we are completely agnostic about the prior, we can use the uniform distribution over the space $-R_{max} \leq R(s) \leq R_{max}$ for each $s \in S$. If we do not want to specify any R_{max} we can try the improper prior $P(\boldsymbol{R}) = 1$ for all $\boldsymbol{R} \in \mathbb{R}^n$.
2. Many real world Markov decision problems have parsimonious reward structures, with most states having negligible rewards. In such situations, it would be better to assume a Gaussian or Laplacian prior:

$$P_{Gaussian}(\boldsymbol{R}(s) = r) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{r^2}{2\sigma^2}}, \forall s \in S$$

$$P_{Laplace}(\boldsymbol{R}(s) = r) = \frac{1}{2\sigma} e^{-\frac{|r|}{2\sigma}}, \forall s \in S$$

3. If the underlying MDP represented a planning-type problem, we expect most states to have low (or negative) rewards but a few states to have high rewards (corresponding to the goal); this can be modeled by a *Beta* distribution for the reward at each state, which has modes at high and low ends of the reward space:

$$P_{Beta}(\boldsymbol{R}(s) = r) = \frac{1}{(\frac{r}{R_{max}})^{\frac{1}{2}} (1 - \frac{r}{R_{max}})^{\frac{1}{2}}}, \forall s \in S$$

In section 6.1, we give an example of how more informative priors can be constructed for particular IRL problems.

4 Inference

We now use the model of section 3 to carry out the two tasks described in the introduction: reward learning and apprenticeship learning. Our general procedure is to derive minimal solutions for appropriate loss functions over the posterior (Eq. 3). Some proofs are omitted for lack of space.

4.1 Reward Learning

Reward learning is an estimation task. The most common loss functions for estimation problems are the linear and squared error loss functions:

$$\begin{aligned} L_{linear}(\boldsymbol{R}, \hat{\boldsymbol{R}}) &= \| \boldsymbol{R} - \hat{\boldsymbol{R}} \|_1 \\ L_{SE}(\boldsymbol{R}, \hat{\boldsymbol{R}}) &= \| \boldsymbol{R} - \hat{\boldsymbol{R}} \|_2 \end{aligned}$$

where $\boldsymbol{R}$ and $\hat{\boldsymbol{R}}$ are the actual and estimated rewards, respectively. If $\boldsymbol{R}$ is drawn from the posterior distribution (3), it can be shown that the expected value of $L_{SE}(\boldsymbol{R}, \hat{\boldsymbol{R}})$ is minimized by setting $\hat{\boldsymbol{R}}$ to the mean of the posterior (see [Berger, 1993]). Similarily, the expected linear loss is minimized by setting $\hat{\boldsymbol{R}}$ to the median of the distribution. We discuss how to compute these statistics for our posterior in section 5.

It is also common in Bayesian estimation problems to use the maximum a posteriori (MAP) value as the estimator. In fact we have the following result:

Theorem 2. *When the expert's policy is optimal and fully specified, the IRL algorithm of [Ng and Russell, 2000] is equivalent to returning the MAP estimator for the model of (3) with a Laplacian prior.*

However in IRL problems where the posterior distribution is typically multimodal, a MAP estimator will not be as representative as measures of central tendency like the mean.

4.2 Apprenticeship Learning

For the apprenticeship learning task, the situation is more interesting. Since we are attempting to learn a policy π, we can formally define the following class of *policy loss functions*:

$$L^p_{policy}(\boldsymbol{R}, \pi) = \| \boldsymbol{V}^*(\boldsymbol{R}) - \boldsymbol{V}^{\pi}(\boldsymbol{R}) \|_p$$

where $\boldsymbol{V}^*(\boldsymbol{R})$ is the vector of optimal values for each state acheived by the optimal policy for $\boldsymbol{R}$ and p is some norm. We wish to find the π that minimizes the expected policy loss over the posterior distribution for $\boldsymbol{R}$. The following theorem accomplishes this:

on α_X as well (Fig 2). But it will be simpler to treat α_X as just a parameter of the distribution.

Theorem 3. *Given a distribution $P(\boldsymbol{R})$ over reward functions $\boldsymbol{R}$ for an MDP (S, A, T, γ), the loss function $L^p_{policy}(\boldsymbol{R}, \pi)$ is minimized for all p by π^*_M, the optimal policy for the Markov Decision Problem $M = (S, A, T, \gamma, E_P[\boldsymbol{R}])$.*

Proof. From the Bellman equations (1) we can derive the following:

$$\boldsymbol{V}^\pi(\boldsymbol{R}) = (\boldsymbol{I} - \gamma \boldsymbol{T}^\pi)^{-1} \boldsymbol{R} \tag{4}$$

where T^π is the $|S| \times |S|$ transition matrix for policy π. Thus, for a state $s \in S$ and fixed π, the value function is a linear function of the rewards:

$$V^\pi(s, \boldsymbol{R}) = \boldsymbol{w}(s, \pi) \cdot \boldsymbol{R}$$

where $\boldsymbol{w}(s, \pi)$ is the s'th row of the coefficient matrix $(\boldsymbol{I} - \gamma \boldsymbol{T}^\pi)^{-1}$ in (4). Suppose we wish to maximize $E[V^\pi(s, \boldsymbol{R})]$ alone. Then,

$$\max_\pi E[V^\pi(s, \boldsymbol{R})] = \max_\pi E[\mathbf{w(s, \pi)} \cdot \boldsymbol{R}] = \max_\pi \boldsymbol{w}(s, \pi) \cdot E[\boldsymbol{R}]$$

By definition this is equal to $V^*_M(s)$, the optimum value function for M, and the maximizing policy π is π^*_M, the optimal policy for M. Thus for all states $s \in S$, $E[V^\pi(s, \boldsymbol{R})]$ is maximum at $\pi = \pi^*_M$.

But $V^*(s, \boldsymbol{R}) \geq V^\pi(s, \boldsymbol{R})$ for all $s \in S$, reward functions $\boldsymbol{R}$, and policies π. Therefore

$$E[L^p_{policy}(\pi)] = E(\| \boldsymbol{V}^*(\boldsymbol{R}) - \boldsymbol{V}^\pi(\boldsymbol{R}) \|_p)$$

is minimized for all p by $\pi = \pi^*_M$. □

So, instead of trying a difficult direct minimization of the expected policy loss, we can find the optimal policy for the mean reward function, which gives the same answer.

5 Sampling and Rapid Convergence

We have seen that both reward learning and apprenticeship learning require computing the mean of the posterior distribution. However the posterior is complex and analytical derivation of the mean is hard, even for the simplest case of the uniform prior. Instead, we generate samples from these distributions and then return the sample mean as our estimate of the true mean of the distribution. The sampling technique we use is an MCMC algorithm `GridWalk` (see [Vempala, 2005]) that generates a Markov chain on the intersection points of a grid of length δ in the region $\mathbb{R}^{|S|}$ (denoted $\mathbb{R}^{|S|}/\delta$).

However, computing the posterior distribution at a particular point $\boldsymbol{R}$ requires calculation of the optimal Q-function for $\boldsymbol{R}$, an expensive operation. Therefore, we use a modified version of `GridWalk` called `PolicyWalk` (Figure 3) that is more efficient: While moving along a Markov chain, the sampler also keeps track of the optimal policy π for the current reward vector $\boldsymbol{R}$. Observe that when π is known, the Q function can be reduced to a linear function of the reward variables, similar to equation 4. Thus step 3b can be performed efficiently. A change in the optimal policy can easily be detected when moving to the next reward vector in the chain $\tilde{\boldsymbol{R}}$, because then for some $(s, a) \in (S, A)$, $Q^\pi(s, \pi(s), \tilde{\boldsymbol{R}}) < Q^\pi(s, a, \tilde{\boldsymbol{R}})$ by Theorem 1. When this happens, the new optimal policy is usually only slightly different from the old one and can be computed by just a few steps of policy iteration (see [Sutton and Barto, 1998]) starting from the old policy π. Hence, `PolicyWalk` is a correct and efficient sampling procedure. Note that the asymptotic memory complexity is the same as for `GridWalk`.

Algorithm `PolicyWalk`(Distribution P, MDP M, Step Size δ)

1. Pick a random reward vector $\boldsymbol{R} \in \mathbb{R}^{|S|}/\delta$.
2. $\pi :=$ `PolicyIteration`$(M, \boldsymbol{R})$
3. Repeat
   (a) Pick a reward vector $\tilde{\boldsymbol{R}}$ uniformly at random from the neighbours of $\boldsymbol{R}$ in $\mathbb{R}^{|S|}/\delta$.
   (b) Compute $Q^\pi(s, a, \tilde{\boldsymbol{R}})$ for all $(s, a) \in S, A$.
   (c) If $\exists (s, a) \in (S, A), Q^\pi(s, \pi(s), \tilde{\boldsymbol{R}}) < Q^\pi(s, a, \tilde{\boldsymbol{R}})$
      i. $\tilde{\pi} :=$ `PolicyIteration`$(M, \tilde{\boldsymbol{R}}, \pi)$
      ii. Set $\boldsymbol{R} := \tilde{\boldsymbol{R}}$ and $\pi := \tilde{\pi}$ with probability $\min\{1, \frac{P(\tilde{\boldsymbol{R}}, \tilde{\pi})}{P(\boldsymbol{R}, \pi)}\}$
   Else
      i. Set $\boldsymbol{R} := \tilde{\boldsymbol{R}}$ with probability $\min\{1, \frac{P(\tilde{\boldsymbol{R}}, \pi)}{P(\boldsymbol{R}, \pi)}\}$
4. Return $\boldsymbol{R}$

Figure 3: PolicyWalk Sampling Algorithm

The second concern for the MCMC algorithm is the speed of convergence of the Markov chain to the equilibrium distribution. The ideal Markov chain is *rapidly mixing* (i.e. the number of steps taken to reach equilibrium is polynomially bounded), but theoretical proofs of rapid mixing are rare. We will show that in the special case of the uniform prior, the Markov chain for our posterior (3) is rapidly mixing using the following result from [Applegate and Kannan, 1993] that bounds the mixing time of Markov chains for pseudo-log-concave functions.

Lemma 1. *Let $F(\cdot)$ be a positive real valued function defined on the cube $\{x \in \mathbb{R}^n | -d \leq x_i \leq d\}$ for some positive d, satisfying for all $\lambda \in [0, 1]$ and some α, β*

$$|f(x) - f(y)| \leq \alpha \| x - y \|_\infty$$

and

$$f(\lambda x + (1 - \lambda) y) \geq \lambda f(x) + (1 - \lambda) f(y) - \beta$$

where $f(x) = \log F(x)$. Then the Markov chain induced by `GridWalk` (and hence `PolicyWalk`) on F rapidly mixes to within ϵ of F in $O(n^2 d^2 \alpha^2 e^{2\beta} \log \frac{1}{\epsilon})$ steps.

Proof. See [Applegate and Kannan, 1993]. □

Theorem 4. *Given an MDP $M = (S, A, T, \gamma)$ with $|S| = N$, and a distribution over rewards $P(\boldsymbol{R}) = Pr_{\mathcal{X}}(\boldsymbol{R}|O_{\mathcal{X}})$ defined by (3) with uniform prior P_R over $C = \{\boldsymbol{R} \in \mathbb{R}^n | -R_{max} \leq R_i \leq R_{max}\}$. If $R_{max} = O(1/N)$ then P can be efficiently sampled (within error ϵ) in $O(N^2 \log 1/\epsilon)$ steps by algorithm `PolicyWalk`.*

Proof. Since the uniform prior is the same for all points $\boldsymbol{R}$, we can ignore it for sampling purposes along with the normalizing constant. Therefore, let $f(\boldsymbol{R}) = \alpha_{\mathcal{X}} E(O_{\mathcal{X}}, \boldsymbol{R})$. Now choose some arbitrary policy π and let

$$f_\pi(\boldsymbol{R}) = \alpha_{\mathcal{X}} \sum_i Q^\pi(s, a_i, \boldsymbol{R})$$

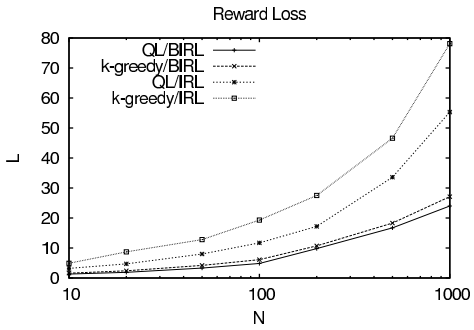

Figure 4: Reward Loss.

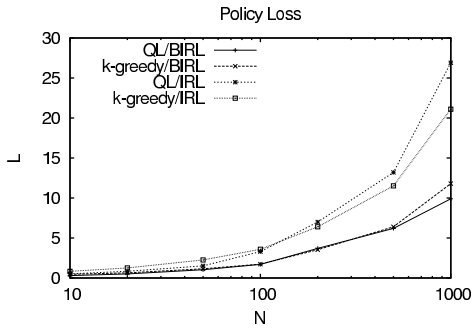

Figure 5: Policy Loss.

Note that f_π is a linear function of $\boldsymbol{R}$ and $f(\boldsymbol{R}) \geq f_\pi(\boldsymbol{R})$, for all $\boldsymbol{R} \in C$. Also we have,

$$\max_{s,a} Q^*(s,a) = \max_{s,a,\pi} Q^\pi(s,a) = \max_{s,\pi} V^\pi_{max}(s) \leq \frac{R_{max}}{1-\gamma}$$

Similarly, $\min_{s,a} Q^*(s,a) \geq -\frac{R_{max}}{1-\gamma}$. Therefore, $f(\boldsymbol{R}) \leq \frac{\alpha_{\mathcal{X}} N R_{max}}{1-\gamma}$ and $f_\pi(\boldsymbol{R}) \geq -\frac{\alpha_{\mathcal{X}} N R_{max}}{1-\gamma}$ and hence

$$f_\pi(\boldsymbol{R}) \geq f(\boldsymbol{R}) - \frac{2\alpha_{\mathcal{X}} N R_{max}}{1-\gamma}$$

So for all $\boldsymbol{R}_1, \boldsymbol{R}_2 \in C$ and $\lambda \in [0,1]$,

$$\begin{aligned} f(\lambda \boldsymbol{R}_1 + (1-\lambda)\boldsymbol{R}_2) &\geq f_\pi(\lambda \boldsymbol{R}_1 + (1-\lambda)\boldsymbol{R}_2) \\ &\geq \lambda f_\pi(\boldsymbol{R}_1) + (1-\lambda) f_\pi(\boldsymbol{R}_2) \\ &\geq \lambda f(\boldsymbol{R}_1) + (1-\lambda) f(\boldsymbol{R}_2) \\ &\quad - \frac{2\alpha_{\mathcal{X}} N R_{max}}{1-\gamma} \end{aligned}$$

Therefore, f satisfies the conditions of Lemma 1 with $\beta = \frac{2\alpha_{\mathcal{X}} N R_{max}}{1-\gamma} = 2N \cdot \frac{O(\frac{1}{N})}{1-\gamma} = O(1)$ and

$$\alpha = \frac{|f(\boldsymbol{R}_1) - f(\boldsymbol{R}_2)|}{\| \boldsymbol{R}_1 - \boldsymbol{R}_2 \|_\infty} \leq \frac{2\alpha_{\mathcal{X}} N R_{max}}{(1-\gamma) O(\frac{1}{N})} = O(N)$$

Hence the Markov chain induced by the GridWalk algorithm (and the PolicyWalk algorithm) on P mixes rapidly to within ϵ of P in a number of steps equal to $O(N^2 \frac{1}{N^2} N^2 e^{O(1)} \log 1/\epsilon) = O(N^2 \log 1/\epsilon)$. □

Note that having $R_{max} = O(1/N)$ is not really a restriction because we can rescale the rewards by a constant factor k after computing the mean without changing the optimal policy and all the value functions and Q functions get scaled by k as well.

6 Experiments

We compared the performance of our BIRL approach to the IRL algorithm of [Ng and Russell, 2000] experimentally. First, we generated random MDPs with N states (with N varying from 10 to 1000) and rewards drawn from i.i.d. Gaussian priors. Then, we simulated two kinds of agents on these MDPs and used their trajectories as input: The first learned a policy by Q-learning on the MDP + reward function. The learning rate was controlled so that the agent was not allowed to converge to the optimal policy but came reasonably close. The second agent executed a policy that maximized the expected total reward over the next k steps (k was chosen to be slightly below the horizon time).

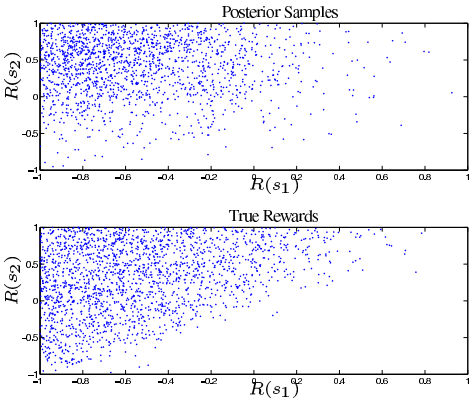

Figure 6: Scatter diagrams of sampled rewards of two arbitrary states for a given MDP and expert trajectory. Our computed posterior is shown to be close to the true distribution.

For BIRL, we used `PolicyWalk` to sample the posterior distribution (3) with a uniform prior. We compared the results of the two methods by their average ℓ_2 distance from the true reward function (Figure 4) and the policy loss with ℓ_1 norm (Figure 5) of the learned policy under the true reward. Both measures show substantial improvement. Note that we have used a logarithmic scale on the x-axis.

We also measured the accuracy of our posterior distribution for small N by comparing it with the true distribution of rewards i.e. the set of generated rewards that gave rise to the same trajectory by the expert. In Figure 6, we show scatter plots of some rewards sampled from the posterior and the true distribution for a 16-state MDP. These figures show that the posterior is very close to the true distribution.

6.1 From Domain Knowledge to Prior

To show how domain knowledge about a problem can be incorporated into the IRL formulation as an informative prior,

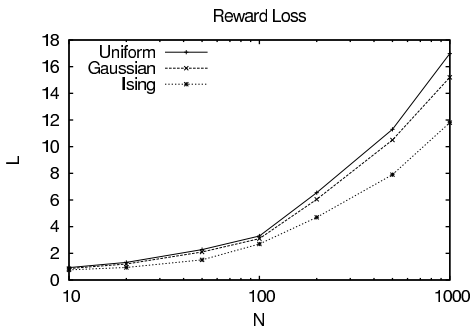

Figure 7: Ising versus Uninformed Priors for Adventure Games

we applied our methods to learning reward functions in adventure games. There, an agent explores a dungeon, seeking to collect various items of treasure and avoid obstacles such as guards or traps. The state space is represented by an m-dimensional binary feature vector indicating the position of the agent and the value of various fluents such as `hasKey` and `doorLocked`. If we view the state-space as an m-dimensional lattice L_S, we see that neighbouring states in L_S are likely to have correlated rewards (e.g. the value of `doorLocked` does not matter when the treasure chest is picked up). To model this, we use an *Ising* prior (see [Cipra, 1987]):

$$P_R(\boldsymbol{R}) = \frac{1}{Z}\exp(-J \sum_{(s',s)\in\mathcal{N}} R(s)R(s') - H\sum_{s} R(s))$$

where $\mathcal{N}$ is the set of neighbouring pairs of states in L_S and J and H are the *coupling* and *magnetization* parameters.

We tested our hypothesis by generating some adventure games (by populating dungeons with objects from a common sense knowledge base) and testing the performance of BIRL with the Ising prior versus the baseline uninformed priors. The results are in figure 7 and show that the Ising prior does significantly better.

7 Related Work

The initial work on IRL was done by [Ng and Russell, 2000] while [Abbeel and Ng, 2004] studied the special case where rewards can be represented as linear combinations of features of the state space and gave a max-margin characterization of the optimal reward function. [Price and Boutilier, 2003] discusses a Bayesian approach to imitating the actions of a mentor during reinforcement learning whereas the traditional literature on apprenticeship learning tries to mimic the behaviour of the expert directly [Atkeson and Schaal, 1997].

Outside of computer science, IRL-related problems have been studied in various guises. In the physical sciences, there is a body of work on *inverse problem theory*, i.e. infering values of model parameters from observations of a physical system [Tarantola, 2005]. In control theory, [Boyd *et al.*, 1994] solved the problem, posed by Kalman, of recovering the objective function for a deterministic linear system with quadratic costs.

8 Conclusions and Future Work

Our work shows that improved solutions can be found for IRL by posing the problem as a Bayesian learning task. We provided a theoretical framework and tractable algorithms for Bayesian IRL and our solutions contain more information about the reward structure than other methods. Our experiments verify that our solutions are close to the true reward functions and yield good policies for apprenticeship learning. There are a few open questions remaining:

1. Are there more informative priors that we can construct for specific IRL problems using background knowledge?
2. How well does IRL generalize? Suppose the transition function of the actor and the learner differed, how robust would the reward function or policy learned from the actor be, w.r.t the learner's state space?

Acknowledgements

The authors would like to thank Gerald Dejong and Alexander Sorokin for useful discussions, and Nitish Korula for help with preparing this paper. This research was supported by NSF grant IIS 05-46663 and CERL grant DACA42-01-D-0004-0014.

References

[Abbeel and Ng, 2004] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *ICML*, 2004.

[Applegate and Kannan, 1993] D. Applegate and R. Kannan. Sampling and integration of near log-concave functions. In *STOC*, 1993.

[Atkeson and Schaal, 1997] C. Atkeson and S. Schaal. Robot learning from demonstration. In *ICML*, 1997.

[Berger, 1993] J. O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer, 2nd edition, 1993.

[Billings *et al.*, 1998] D. Billings, D. Papp, J. Schaeffer, and D. Szafron. Opponent modeling in Poker. In *AAAI*, pages 493–498, Madison, WI, 1998. AAAI Press.

[Boyd *et al.*, 1994] S. Boyd, L. E. Ghaoui, E. Feron, and V. Balakrishnan. Linear matrix inequalities in system and control theory. *SIAM*, 1994.

[Cipra, 1987] B. A. Cipra. An introduction to the Ising model. *Am. Math. Monthly*, 94(10):937–959, 1987.

[Ng and Russell, 2000] A. Y. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *ICML*, pages 663–670. Morgan Kaufmann, San Francisco, CA, 2000.

[Price and Boutilier, 2003] B. Price and C. Boutilier. A Bayesian approach to imitation in reinforcement learning. In *IJCAI*, 2003.

[Russell, 1998] S. Russell. Learning agents for uncertain environments (extended abstract). In *COLT*. ACM Press, 1998.

[Sargent, 1994] T J Sargent. Do people behave according to Bellman's principle of optimality? *JEP*, 1994.

[Sutton and Barto, 1998] R.S. Sutton and A.G. Barto. *Reinforcement Learning*. MIT Press, 1998.

[Tarantola, 2005] A. Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*. SIAM, 2nd edition, 2005.

[Vempala, 2005] S. Vempala. Geometric random walks: A survey. MSRI volume on Combinatorial and Computational Geometry, 2005.