Improving the Efficiency of Bayesian Inverse Reinforcement Learning

Bernard Michini* and Jonathan P. How**
Aerospace Controls Laboratory
Massachusetts Institute of Technology, Cambridge, MA 02139 USA

***Abstract*— Inverse reinforcement learning (IRL) is the task of learning the reward function of a Markov Decision Process (MDP) given knowledge of the transition function and a set of expert demonstrations. While many IRL algorithms exist, Bayesian IRL [1] provides a general and principled method of reward learning by casting the problem in the Bayesian inference framework. However, the algorithm as originally presented suffers from several inefficiencies that prohibit its use for even moderate problem sizes. This paper proposes modifications to the original Bayesian IRL algorithm to improve its efficiency and tractability in situations where the state space is large and the expert demonstrations span only a small portion of it. The key insight is that the inference task should be focused on states that are similar to those encountered by the expert, as opposed to making the naive assumption that the expert demonstrations contain enough information to accurately infer the reward function over the entire state space. A modified algorithm is presented and experimental results show substantially faster convergence while maintaining the solution quality of the original method.**

I. Introduction

Inverse reinforcement learning (IRL) is the subset of learning from demonstration methods in which the reward function, or equivalently the task description, is learned from a set of expert demonstrations. The IRL problem is formalized using the Markov Decision Process (MDP) framework in the seminal work [2]: Given expert demonstrations in the form of state-action pairs, determine the reward function that the expert is optimizing assuming that the model dynamics (i.e. transition probabilities) are known.

In the larger context of learning from demonstration, most algorithms attempt to directly learn the policy (sometimes in addition to the model dynamics) using the given demonstrations [3]. IRL separates itself from these methods in that it is the *reward function* that is learned, not the policy. The reward function can be viewed as a high-level description of the task, and can thus "explain" the expert's behavior in a richer sense than the policy alone. No information is lost in learning the reward function instead of the policy. Indeed, given the reward function and model dynamics an optimal policy can be solved for. Thus the reward function is also transferable, in that changing the model dynamics *would not* affect the reward function but *would* render a given policy invalid. For these reasons, IRL may be more advantageous than direct policy learning methods in many situations.

The advantages of IRL come at a cost, namely that the IRL problem itself is ill-posed. In general, there is no single reward function that will make the expert's behavior optimal [4–8]. This is true even if the expert's policy is *fully specified* to the IRL algorithm, i.e. many reward functions may map to the same optimal policy. Another challenge in IRL is that in real-world situations the expert may act sub-optimally or inconsistently. Finally, in problems with a large state space there may be a relatively limited amount of expert information.

Several algorithms address these limitations successfully and have shown IRL to be an effective method of learning from demonstration [4–7, 9–11]. A general Bayesian approach is taken in Bayesian inverse reinforcement learning (BIRL) [1]. In BIRL, the reward learning task is cast as a standard Bayesian inference problem. A prior over reward functions is combined with a likelihood function for expert demonstrations (the evidence) to form a posterior over reward functions which is then sampled using Markov chain Monte Carlo (MCMC) techniques. BIRL has several advantages. It does not assume that the expert behaves optimally (and in fact naturally models the degree of sub-optimality). Since a *distribution* over reward functions is solved for, the ambiguity of an inconsistent or uncertain expert is addressed explicitly. External *a priori* information and constraints on the reward function can be encoded naturally through the choice of prior distribution. Perhaps most importantly, the principled Bayesian manner in which the IRL problem is framed allows for the algorithm designer to leverage a wide range of inference solution techniques from the statistics and machine learning literature. Thus BIRL forms a general and powerful foundation for the problem of reward learning.

As will be shown in more detail, the Bayesian IRL algorithm as presented in [1] suffers from several practical limitations. The reward function to be inferred is a *vector* whose length is equal to the number of MDP states. Given the nature of the MCMC method used, a large number of iterations is required for acceptable convergence to the mean of the posterior. The problem stems mainly from the fact that each of these iterations requires re-solving the MDP for the optimal policy, which can be very computationally expensive as the size of the state space increases (the so-called "curse of dimensionality").

In this paper, a modified Bayesian IRL algorithm is proposed based on the simple observation that the information

*Ph.D. Candidate, Aerospace Controls Laboratory, MIT, bmich[at]mit[dot]edu

**Richard C. Maclaurin Professor of Aeronautics and Astronautics, Aerospace Controls Laboratory (Director), and Laboratory for Information and Decision Systems, MIT, Associate Fellow AIAA, jhow[at]mit[dot]edu

contained in the expert demonstrations may very well *not* apply to the entire state space. As an abstract example, if the IRL agent is given a small set of expert trajectories that reside entirely in one "corner" of the state space, those demonstrations may provide little if any information about the reward function in some opposite "corner", making it naive to perform reward function inference over the entire state space. The proposed method takes as input a kernel function that quantifies similarity between states. The BIRL inference task is then scaled down to include only those states which are similar to the ones encountered by the expert (the degree of "similarity" being a parameter of the algorithm). The resulting algorithm is shown to have much improved computational efficiency while maintaining the quality of the resulting reward function estimate. If the kernel function provided is simply a constant, the original BIRL algorithm from [1] is obtained. Also, a cooling schedule is proposed that is shown to further speed up convergence.

The paper is organized as follows. Section II reviews MDP preliminaries and the standard BIRL method from [1]. In Section III, a counter-example is presented that illustrates the practical limitations of standard BIRL. Section IV proposes a modified BIRL algorithm to address some of these limitations. Experimental results are given in Section V, and conclusions and future work are presented in Section VI.

II. Background

This section briefly summarizes some MDP and Bayesian IRL preliminaries and states any assumptions made in this paper.

A. Markov Decision Processes

A finite-state Markov Decision Process is a tuple $M = (S, A, T, \gamma, R)$ where:

- S is a set of N *states*.
- A is a set of *actions*.
- $T : S \times A \times S \mapsto [0, 1]$ is the function of *transition probabilities*, such that $T(s_1, a, s_2)$ is the probability of being in state s_2 after taking action a from state s_1.
- $R : S \mapsto \mathbb{R}$ is the *reward function*.
- $\gamma \in [0, 1)$ is the *discount factor*.

In this paper it is assumed that T is known, and that R is a function only of the state and bounded in absolute value by $R_{\max}$. The reward function is represented throughout as an N-dimensional vector $\mathbf{R}$ whose ith element is $R(s_i)$.

A *stationary policy* is a function $\pi : S \mapsto A$. It is assumed that the expert executes a stationary (but potentially suboptimal) policy. From [12] we have the following set of definitions and results:

1) The infinite-horizon expected reward for starting in state s and following policy π thereafter is given by the *value function* $V^\pi(s)$:

$$V^\pi(s) = E_\pi \left[\sum_{i=0}^{\infty} \gamma^i R(s_i) \middle| s_0 = s \right] \quad (1)$$

The value function satisfies the following *Bellman Equation* for all $s \in S$:

$$V^\pi(s) = R(s) + \gamma \left[\sum_{s'} T(s, \pi(s), s') V^\pi(s') \right] \quad (2)$$

2) The infinite-horizon expected reward for starting in state s, taking action a, and following policy π thereafter is given by the *action-value function* $Q^\pi(s, a)$:

$$Q^\pi(s, a) = E_\pi \left[\sum_{i=0}^{\infty} \gamma^i R(s_i) \middle| s_0 = s, a_0 = a \right] \quad (3)$$

The action-value function satisfies the following *Bellman Equation* for all $s \in S, a \in A$:

$$Q^\pi(s, a) = R(s) + \gamma \left[\sum_{s'} T(s, a, s') V^\pi(s') \right] \quad (4)$$

3) A policy π is optimal for M iff, for all $s \in S$:

$$\pi(s) = \arg\max_{a \in A} Q^\pi(s, a) \quad (5)$$

An optimal policy is denoted as π^* with corresponding value and action-value functions V^* and Q^*.

B. Bayesian IRL

The following summarizes the Bayesian inverse reinforcement learning framework from [1]. The reader is referred to the original work for details and proofs. The basic premise of BIRL is to infer a posterior distribution for the reward vector $\mathbf{R}$ from a prior distribution and a likelihood function for the evidence (the expert's actions). The evidence O takes the form of observed state-action pairs, so that $O = \{(s_1, a_1), (s_2, a_2), ..., (s_k, a_k)\}$. Applying Bayes Theorem, the posterior can be written as:

$$Pr(\mathbf{R}|O) \;=\; \frac{Pr(O|\mathbf{R}) Pr(\mathbf{R})}{Pr(O)} \quad (6)$$

where each term is explained below:

- $Pr(\mathbf{R}|O)$: The posterior distribution of the reward vector given the observed actions of the expert. This is the target distribution whose mean will be estimated.
- $Pr(O|\mathbf{R})$: The likelihood of the evidence (observed expert state-action pairs) given a particular reward vector $\mathbf{R}$. A perfect expert would always choose optimal actions, and thus state-action pairs with large $Q^*(s_i, a_i, \mathbf{R})$ would be more likely. However, the expert is assumed to be imperfect, so the likelihood of each state-action pair is given by an exponential distribution:

$$Pr(a_i|s_i, \mathbf{R}) = \frac{e^{\alpha Q^*(s_i, a_i, \mathbf{R})}}{\sum_{b \in A} e^{\alpha Q^*(s_i, b, \mathbf{R})}} \quad (7)$$

where α is a parameter representing our confidence that the expert chooses actions with high value (the lower the value of α the more "imperfect" the expert is expected to be). The likelihood of the entire evidence is thus:

$$Pr(O|\mathbf{R}) = \frac{e^{\alpha \sum_i Q^*(s_i, a_i, \mathbf{R})}}{\sum_{b \in A} e^{\alpha \sum_i Q^*(s_i, b, \mathbf{R})}} \quad (8)$$

- $Pr(\mathbf{R})$: Prior distribution representing how likely a given reward vector is based *only* on prior knowledge. This is where constraints and *a priori* knowledge of the rewards can be injected.
- $Pr(O)$: The probability of O over the entire space of reward vectors $\mathbf{R}$. This is very difficult to calculate but won't be needed for the MCMC methods used.

For the reward learning task, we wish to estimate the expert's reward vector $\mathbf{R}$. One common way to determine the accuracy of an estimate is the *squared loss function*:

$$L_{SE}(\mathbf{R}, \hat{\mathbf{R}}) = ||\mathbf{R} - \hat{\mathbf{R}}||_2 \tag{9}$$

where $\mathbf{R}$ and $\hat{\mathbf{R}}$ are the actual and estimated expert reward vectors, respectively. It is shown in [1] that the *mean* of the posterior distribution (6) minimizes (9).

The posterior distribution of $\mathbf{R}$ must also be used to find a policy that is close to the expert's. Given some reward vector $\mathbf{R}$, a sensible measure of the closeness of policy π to the optimal policy obtained using $\mathbf{R}$ is a *policy loss function*:

$$L^p_{\text{policy}}(\mathbf{R}, \pi) = ||V^*(\mathbf{R}) - V^\pi(\mathbf{R})||_p \tag{10}$$

where p is a norm. It is shown in [1] that the policy which minimizes (10) is the optimal policy obtained using the *mean* of the posterior (6).

Thus, for both the reward estimation and policy learning tasks, inference of the mean of the posterior (6) is required. Markov chain Monte Carlo (MCMC) techniques are appropriate for this task [13]. The method proposed in [1], termed `PolicyWalk`, iterates as follows. Given a current reward vector $\mathbf{R}$, sample a new proposal $\tilde{\mathbf{R}}$ randomly from the neighbors of $\mathbf{R}$ on a grid of length δ, i.e. $\mathbf{R} = \tilde{\mathbf{R}}$ except for one randomly chosen $s \in S$:

$$\tilde{\mathbf{R}}(s) = \mathbf{R}(s) \pm \delta \tag{11}$$

The proposal is accepted ($\mathbf{R} := \tilde{\mathbf{R}}$) with probability $\min\left\{1, \frac{Pr(\tilde{\mathbf{R}}|O)}{Pr(\mathbf{R}|O)}\right\}$, where the posteriors are given by (6) so that:

$$\frac{Pr(\tilde{\mathbf{R}}|O)}{Pr(\mathbf{R}|O)} = \frac{Pr(O|\tilde{\mathbf{R}})Pr(\tilde{\mathbf{R}})}{Pr(O)} \cdot \frac{Pr(O)}{Pr(O|\mathbf{R})Pr(\mathbf{R})} = \frac{Pr(O|\tilde{\mathbf{R}})Pr(\tilde{\mathbf{R}})}{Pr(O|\mathbf{R})Pr(\mathbf{R})} \tag{12}$$

The mean of the posterior is thus approximated by the empirical mean of $\mathbf{R}$ over all of the iterations. Note that none of the normalizing constants are needed and thus the likelihood and prior only need to be known to a constant. Here it is also noted that finding Q^* for the likelihood calculation requires the MDP to be solved using $\tilde{\mathbf{R}}$, and this must be done at *every* MCMC iteration. Solving the MDP each iteration is typical among IRL algorithms, and highlights the need to reduce the number of iterations to the extent possible.

III. Limitations of Standard Bayesian IRL

In this section, a simple example is presented that illustrates some practical limitations of the original Bayesian IRL algorithm from [1].

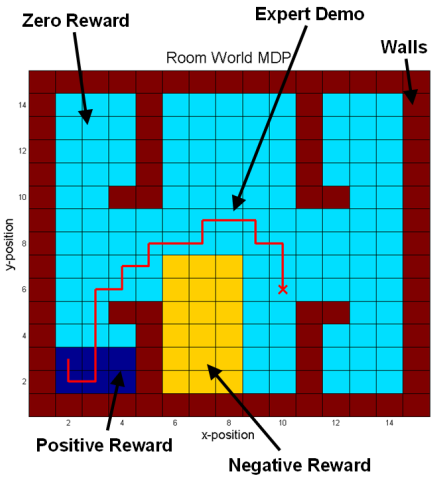

Fig. 1. Room World MDP, showing the walls (maroon), zero reward (cyan), positive reward (dark blue), negative reward (yellow), and example expert demonstration (red).

A. Room World MDP

"Room World", shown in Figure 1, is a $15x15$ grid with walls that form five rooms. The true reward function consists of a block of negative reward in the center room, positive reward in the lower-left room, and zero reward elsewhere. The agent can choose from four actions (up, down, left, right). If "up" is chosen, the agent moves up with probability 0.75, left or right each with probability 0.1, and stays in the same cell with probability 0.05 (and similarly for the other actions). The agent is not allowed to enter wall states. The discount factor is 0.93 and the magnitude of rewards is 0.01.

The "expert" executes the optimal policy found using the true reward function. The simulate an imperfect expert, the optimal action is chosen with probability 0.95, and a random action is chosen otherwise. The expert always starts in the cell $(x, y) = (10, 6)$. An example expert demonstration is shown in Figure 1 (in red).

B. Applying Bayesian IRL

The Bayesian IRL algorithm presented in [1] is applied to attempt to learn the reward function for the Room World MDP given a set of 100 expert demonstrations shown in Figure 4. The reward vector is assumed to be composed of independently identically distributed (i.i.d.) components, each with prior distribution:

$$Pr(R) = 0.4e^{-0.001(R-R_{\text{max}}).^2} + 0.4e^{-0.001(R-R_{\text{min}})^2} + e^{-0.001(R)^2} \tag{13}$$

as shown in Figure 2 (recall that the prior only needs to be known to a constant). This prior reflects external knowledge that for any given state the reward is most likely zero, and if not than will likely take the minimum or maximum reward value. [1]

[1]It is noted that the ability of Bayesian IRL to impose a prior such as this effectively reduces the ambiguity and ill-posedness of the IRL reward estimation problem by intuitively limiting the space of possible reward functions.

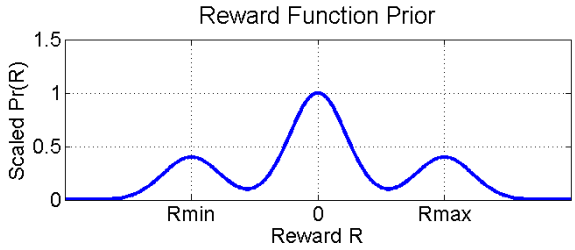

Fig. 2. Reward function prior distribution (to a constant).

For the Room World MDP with 225 states, the policy loss (defined in Section V) does not converge for roughly 600 iterations as seen in Figure 5. Recall that each iteration requires resolving the MDP to find the optimal policy according to the proposed reward function. While it is true that this typically takes few policy iterations since the reward function is only perturbed slightly from the previous, the number of MCMC iterations required for a more realistically-large state space will quickly become prohibitive. There are two main reasons for this inefficiency, each discussed below.

Limited expert information: Foremost, it is clear that the set of expert demonstrations given to the BIRL algorithm contains a limited amount of information relative to the entire state space. Intuitively, it would be difficult to infer much about the reward function in the upper-right room since there are no observed state-action pairs near that area. Even so, standard BIRL will attempt to infer the reward of *every* state. Empirically, the estimates in states far from any expert demonstrations tend to "wander", slowing convergence of the posterior reward distribution as a whole. More concretely, when a new proposal is drawn in which a far-away state is perturbed, the effect on the likelihood of the evidence and the prior as a whole is very small. Thus $Pr(\tilde{\mathbf{R}}|O) \approx Pr(\mathbf{R}|O)$ and the acceptance probability $\frac{Pr(\tilde{\mathbf{R}}|O)}{Pr(\mathbf{R}|O)} \approx 1$, meaning that the new proposal will most likely be accepted. As a result, the reward estimate at far-away states will change frequently and in a way that is not affected by the evidence. The efficiency of the algorithm suffers since it spends much of its time attempting to infer the reward in states for which it has little expert evidence.

Exploration vs. exploitation: The MCMC algorithm must search over a reward function space whose dimension is the number of MDP states. Even for toy problems this can easily grow to over $N = 10^6$, and as mentioned before the number of MCMC iterations needed to approximate the mean of the posterior will become extremely large. Simulated annealing is a method used to focus the sampled distribution around it's maximum by using a modified acceptance probability of $\left(\frac{Pr(\tilde{\mathbf{R}}|O)}{Pr(\mathbf{R}|O)}\right)^{1/T_i}$ where T_i is a decreasing "cooling schedule" [13]. While this method is typically used to estimate the maximum of the posterior (MAP), it provides a "knob" to focus the samples on areas of higher posterior probability essentially trading exploration of the full distribution for exploitation of it's peaks. For high-dimensional reward spaces (i.e. MDPs with large N), this is necessary to reduce the number of samples needed to converge to a high-probability area of the posterior.

IV. Modifications to the BIRL Algorithm

This section describes two modifications to the original Bayesian IRL algorithm to address the aforementioned limitations.

A. Kernel-based Relevance Function

It unlikely that the observed expert demonstrations will span every state of the MDP, or even provide a sparse covering of the entire state space for large problems. Thus it is naive to assume that the reward function over the entire state space can be accurately inferred. Instead, it would make intuitive sense to learn the rewards in states "similar to" to those encountered by the expert. The notion of similarity must be rigorously defined, and for this a kernel function is used. Kernel functions are commonly used in machine learning for exactly this purpose [14], and are defined as the dot product of two *feature vectors*. A feature is a mapping from states to feature space $\mathbf{\Phi} : S \mapsto \mathbb{R}^{k\times 1}$, so that the corresponding kernel function is given by[2]:

$$k(s, s') = \mathbf{\Phi}^T(s) \cdot \mathbf{\Phi}(s') \tag{14}$$

The kernel function is passed in as a parameter to the modified algorithm, and is used to define the *state relevance function* $\rho : S \mapsto [0, 1]$:

$$\rho(s) = \frac{\sum_{s'\in O} k(s, s')}{Z} \tag{15}$$

where O is the set of expert state-action pairs and $Z = \max_{s\in S} \rho(s)$ is a normalizing constant. Intuitively $\rho(s)$ is a normalized measure of how similar state s is to the set of states encountered by the expert.

The state relevance $\rho(s)$ is used in the modified BIRL algorithm shown in Figure 3 as follows. To propose a new reward vector $\tilde{\mathbf{R}}$, a state $\tilde{s} \in S$ is sampled at random. The state is accepted with probability $\rho(\tilde{s})$, and the new reward vector proposal is chosen such that $\tilde{\mathbf{R}} := \mathbf{R}$, except for $\tilde{\mathbf{R}}(\tilde{s}) = \mathbf{R}(\tilde{s}) \pm \delta$. If $\tilde{s}$ is rejected, the process repeats until a state is accepted. This process models the original BIRL algorithm closely, except that now the reward search is focused more heavily on states that are more similar to those encountered by the expert. Note that in the trivial case of a constant kernel $k(s, s') = C$ (i.e. each state s is equally similar to all other states), the original BIRL algorithm `PolicyWalk` from [1] is obtained.

Note in the modified BIRL algorithm that the reward vector $\mathbf{R}$ is initialized to the maximum of the prior. This is because the state relevance modification causes the algorithm to effectively *not* infer the reward vector for states with a low relevance score, and thus the reward in these states

[2]While $k(s, s')$ corresponds to a dot product of feature vectors, this dot product need not be explicated. For instance, the popular radial basis kernel $k(s, s') = e^{-||s-s'||_2/2\sigma^2}$ represents the dot product of an infinitely long feature vector [14].

needs to be initialized to a reasonable value. The relevance function can thus be thought of as a state-by-state measure of how much the expert demonstrations will affect the reward estimate at that state.

ModifiedBIRL(Posterior $Pr(\mathbf{R}|O)$, MDP M, Kernel k, Cooling Sched. T_i, Step Size δ)

1) Initialize reward vector $\mathbf{R}$ to the max of the prior
2) (π^*, Q^*) = PolicyIteration$(M, \mathbf{R})$
3) Repeat
   a) Pick state $\tilde{s} \in S$ uniformly at random, accept $\tilde{s}$ with prob. $\rho(\tilde{s})$ from (15). Otherwise, repeat until an $\tilde{s}$ is accepted.
   b) Set $\tilde{\mathbf{R}} := \mathbf{R}$, except for $\tilde{\mathbf{R}}(\tilde{s}) = \mathbf{R}(\tilde{s}) \pm \delta$
   c) $(\tilde{\pi}^*, \tilde{Q}^*)$ = PolicyIteration$(M, \tilde{\mathbf{R}}, \pi^*)$
   d) Set $\mathbf{R} = \tilde{\mathbf{R}}$ and $\pi^* = \tilde{\pi}^*$ with prob. $\min\left\{1, \left(\frac{Pr(\tilde{\mathbf{R}}|O)}{Pr(\mathbf{R}|O)}\right)^{1/T_i}\right\}$ (i is the iteration number).
4) Return $\mathbf{R}$

Fig. 3. Modified BIRL Algorithm

B. Cooling Schedule

As discussed in Section III, the original BIRL algorithm lacks the ability to trade off exploration for exploitation in order to speed up convergence of the posterior. The degree of exploration is thus entirely reliant on the choice of likelihood and prior distributions. To address this, a small modification to the acceptance probability is made. As in Simulated Annealing, the new acceptance probability is:

$$p_{\text{accept}} = \min\left\{1, \left(\frac{Pr(\tilde{\mathbf{R}}|O)}{Pr(\mathbf{R}|O)}\right)^{1/T_i}\right\} \tag{16}$$

where T_i is a cooling schedule (which is a function of iteration number i) passed into the algorithm. As T_i decreases, the proposals will focus more heavily on areas of large posterior probability (favoring exploitation). Selection of the cooling schedule is left as a parameter, though there are many popular methods in the literature [13]. As will be shown in Section V, the use of a simple decreasing cooling schedule in the modified BIRL algorithm allows the MCMC process to first find areas of high posterior probability then focus the samples towards them.

V. Experiments

To compare the performance of the original Bayesian IRL algorithm to the modified BIRL method proposed in Section IV, the Room World MDP presented in Section III-A is used with the imperfect expert as described providing 100 demonstrations each of length 50 (shown in Figure 4). Four variations are compared:

1) **`PolicyWalk` BIRL:** The algorithm exactly as presented in [1] with likelihood given by (8), prior given by (13), $\alpha = 0.95$, and $\delta = R_{\max}/3$.
2) **`PolicyWalk` BIRL with Cooling:** Same as above, but with the a cooling schedule added as described in Section IV-B. The cooling parameter was set to $1/T_i = 25 + i/50$ where i is the MCMC iteration number.
3) **Modified BIRL with narrow state relevance kernel:** BIRL with cooling as above, but also using the state relevance function from Section IV-A. The relevance kernel is a simple radial basis kernel that uses Euclidean distance as the measure of similarity:

$$k(s, s') = e^{-||s-s'||_2 \ / \ 2\sigma^2} \tag{17}$$

with $\sigma = 0.1$. Figure 4 (left) shows the corresponding state relevance given the expert demonstrations (overlaid).
4) **Modified BIRL with wide state relevance kernel:** Same as above but with a "wider" state relevance kernel defined using $\sigma = 1$. This is shown in Figure 4 (right), and compared to the narrower kernel above it has high value over a wider set of states around the expert demonstrations.

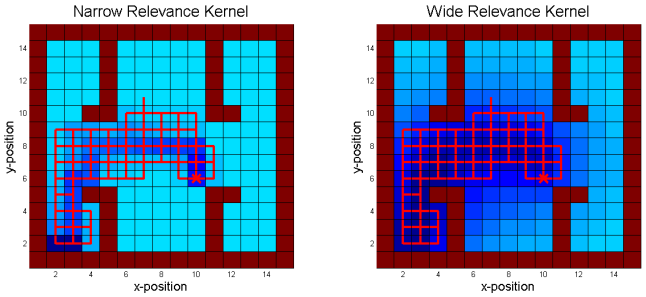

Fig. 4. State relevance scores for a narrow RBF relevance kernel (left) and a wide RBF relevance kernel (right). Cyan corresponds to zero and dark blue corresponds to one. The set of 100 expert demonstrations are overlaid in red.

Figure 5 compares the 0-1 policy loss for each of the four algorithms as a function of the MCMC iteration number, averaged over ten episodes. At each iteration, the current MCMC reward vector $\mathbf{R}$ is used to find the optimal policy π^*, and the 0-1 policy loss is simply the number of expert state-action pairs that do not agree with π^* (i.e. the number of times the expert made the wrong decision according to the current reward function estimate). Policy loss is chosen as the measure of algorithm performance over reward loss given the ill-posedness of the IRL problem to recover the exact true rewards.

Bayesian IRL with a cooling schedule (green triangles) is shown to converge roughly three times faster than standard Bayesian IRL (blue line). Both losses reach the same final value of roughly 500. Intuitively this is explained by the fact that the cooling schedule allows the algorithm to more quickly focus the samples on peaks in the posterior reward distribution.

It is clear that the modified BIRL algorithms which make use of the relevance kernel (cyan crosses and red dashed line) converge much more quickly, about ten times faster than standard BIRL. This stems directly from the fact that the inference is being directed towards states where there is more expert information instead of wasting time in irrelevant states. In addition, the modified BIRL algorithms converge to about half the loss of original BIRL, implying that the solutions not only converge faster but are also more accurate.

It is interesting to note the difference in performance between the two modified IRL algorithms (one using the narrow kernel and one using the wide kernel). The narrow kernel converges faster but to a larger steady-state loss, i.e. inferring the rewards over less states yields faster convergence but restricts the algorithm's ability to accurately explain the evidence. Intuitively this gives the algorithm designer the ability to tradeoff accuracy for lowered computation time by varying the width of the relevance kernel.

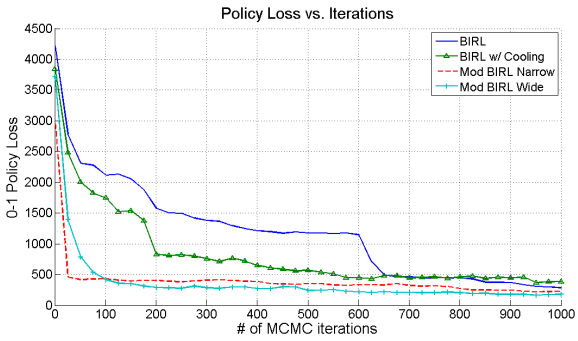

Fig. 5. The 0-1 policy loss versus number of MCMC iterations for the RoomWorld example comparing original BIRL, BIRL with a cooling schedule, modified Bayesian IRL with a narrow relevance kernel, and modified Bayesian IRL with a wide relevance kernel.

VI. Conclusions and Future Work

This paper presents two key modifications to the original Bayesian IRL framework that are shown to reduce convergence time substantially while maintaining solution quality. The proposed methods allow the user to tradeoff computation time for solution accuracy by defining a kernel function that focuses the inference task on states similar to those encountered by the expert.

It should be noted that Active IRL [8] also attempts to improve the efficiency of BIRL by asking the expert for additional demonstrations in states where the policy is most uncertain. While Active IRL is shown to improve performance, there seems to be two main drawbacks. First, Active IRL relies on the fact that the expert can be asked for more information whereas in many situations this is not possible. Second, Active IRL does nothing to improve the tractability of the initial solution (*before* the expert is asked for more demonstrations). Thus, like BIRL, Active IRL remains intractable for large state spaces.

There are several potential avenues for future work. Choosing the prior distribution is (as always) a difficult task since it imposes structure on the learned reward function. Incorporating hierarchical modeling methods into Bayesian IRL to also learn the structure of the reward function could prove to be a powerful and flexible method of behavior learning. Also, the Modified BIRL algorithm presented (Figure 3) only increases the efficiency of the *inference* task, not the *policy solution* task inherent within each iteration. The relevance kernel could just as well be used to improve the efficiency of the policy solution step through the use of sampled-based approximate dynamic programming methods.

Acknowledgements

The authors gratefully acknowledge the Office of Naval Research Science of Autonomy program for supporting this work under contract #N000140910625.

References

[1] D. Ramachandran and E. Amir, "Bayesian inverse reinforcement learning," *IJCAI*, pp. 2586–2591, 2007.

[2] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *Proc. of the 17th International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., 2000, pp. 663–670.

[3] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

[4] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum Entropy Inverse Reinforcement Learning," in *Proc AAAI*. AAAI Press, 2008, pp. 1433–1438.

[5] B. D. Ziebart, A. Maas, and J. A. Bagnell, "Human Behavior Modeling with Maximum Entropy Inverse Optimal Control," *AAAI Spring Symposium on Human Behavior Modeling*, pp. 92–97, 2009.

[6] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum margin planning," *Proc. of the 23rd International Conference on Machine Learning*, pp. 729–736, 2006.

[7] G. Neu and C. Szepesvari, "Apprenticeship learning using inverse reinforcement learning and gradient methods," in *Proc. UAI*, 2007.

[8] M. Lopes, F. Melo, and L. Montesano, "Active Learning for Reward Estimation in Inverse Reinforcement Learning," *Machine Learning and Knowledge Discovery in Databases*, pp. 31–46, 2009.

[9] N. Ratliff, D. Bradley, and J. Bagnell, "Boosting structured prediction for imitation learning," in *Advances in Neural Information Processing Systems 19*, 2007.

[10] U. Syed and R. E. Schapire, "A Game-Theoretic Approach to Apprenticeship Learning," *Advances in Neural Information Processing Systems 20*, vol. 20, pp. 1–8, 2008.

[11] P. Abbeel, "Apprenticeship learning and reinforcement learning with application to robotic control," Ph.D. dissertation, Stanford, 2008.

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

[13] C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan, "An Introduction to MCMC for Machine Learning," *Science*, vol. 50, no. 1, pp. 5–43, 2003.

[14] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, 2001.