Inverse Optimal Control with Linearly Solvable MDPs

Krishnamurthy Dvijotham  Emanuel Todorov
Department of Computer Science and Engineering, University of Washington

Abstract

We present novel efficient algorithms for inverse reinforcement learning, that is, recovering the (unknown) cost function of an optimal agent from its trajectories. Ng et al (Ng & Russell, 2000) showed that the problem in the general MDP setting is ill-posed, that is, there are infinitely many such cost functions and heuristics were proposed for selecting one of them. We show that for linearly solvable MDPs (Todorov, 2009), the problem is well-posed and it is possible to recover a unique cost function from given trajectories of the agent using an unconstrained convex program. Our algorithms don’t require access to an MDP solver, unlike the common apprenticeship learning algorithms (Abbeel & Ng, 2004). Experiments show that even with large state spaces, the algorithm learns good estimates of the value function/cost function/policy beating other algorithms (Abbeel & Ng, 2004)(Schapire & Syed, 2008)(Ng & Russell, 2000) in speed with comparable or better accuracy.

1 Introduction

The problem of inverse optimal control is to compute the cost function with respect to which the behavior of a given control system (i.e. the "expert") is optimal. Inverse optimal control is useful in the context of imitation or apprenticeship learning (Abbeel & Ng, 2004). It is also useful as a data analysis tool in neuroscience: there are good reasons to believe that biological movements are near-optimal, but the exact cost function being optimized in a particular task is not always clear (Todorov, 2004; Wolpert, 2004). Inferring goals from behavior is of interest in cognitive science as well (Baker et al., 2007).

In the machine learning community, Ng et al (Ng & Russell, 2000) were the first to consider the problem of inferring the cost function given the trajectories of an expert who is assumed to be optimally solving an MDP. The authors showed that the problem is ill-posed, that is, there are infinitely many cost functions that make a given policy optimal and proposed some heuristics for choosing one of them. A related problem is that of apprenticeship learning (Abbeel & Ng, 2004)(Schapire & Syed, 2008), where the goal is to simply learn a policy that performs nearly as well as (or perhaps better than) the observed expert, even though the algorithms are unable to accurately recover the value function.

In this paper, we consider a special, yet fairly general class of MDPs (Todorov, 2009)(Todorov, 2007), for which we can show that the problem is well-posed and we can uniquely recover a cost function from trajectories of an optimal agent by solving an unconstrained convex optimization problem. In this special case, there is a one-to-one mapping between cost and value functions so that learning a good policy is equivalent to learning the cost function well. We show that it is possible to recover either the cost or the value function interchangeably and the optimization remains convex under linear parameterizations of either. Our algorithms are all formulated as a single unconstrained optimization problem and do not need repeated access to an MDP solver like common apprenticeship learning algorithms. (Abbeel & Ng, 2004), which makes them much faster.

Maximum Entropy IRL (Ziebart et al., 2008) is another recent algorithm that tries to resolve the ill-posedness of the general IRL problem by using maximum entropy heuristic. We show that it is a special case of our framework obtained by choosing a specific way of linearly parametrizing a combination of the cost function and transition dynamics: this gives a clearer
interpretation of the maximum entropy heuristic as actually solving the inverse problem for our special class of MDPs. Thus, our algorithm can be interpreted either as solving the inverse problem exactly for a special class of MDPs, or using a maximum entropy/min KL-divergence criterion to resolve the ambiguity that exists in the general MDP setting. However, our algorithms exploit the structure of the special class of MDPs and are hence more efficient that those presented in (Ziebart et al., 2008) although they are solving the same problem.

2 Background

We will be able to develop efficient inverse methods for stochastic optimal control problems within a certain class (Todorov, 2009) (Kappen, 2005). This class of problems is briefly summarized in the present section. Despite its special properties giving rise to efficient algorithms, the class is very general. It includes infinite horizon (discounted and average cost), finite horizon and first-exit formulations, in either continuous or discrete space and time. The state costs and stochastic dynamics can be arbitrary. The only restrictions are on the form of the control signal and the associated control cost. In this paper we focus on discrete first-exit problems, although towards the end we show applications to continuous problems using discretization.

2.1 MDPs with linear Bellman equations

The MDP is defined by specifying a state cost \( q(x) \), and passive dynamics \( x' \sim P(\cdot|x) \) characterizing the behavior of the system in the absence of controls. The controller can impose any dynamics \( x' \sim u(\cdot|x) \) it wishes, however it pays a price (control cost) which is the KL divergence between \( u \) and \( P \). We further require that \( u(x'|x) = 0 \) whenever \( P(x'|x) = 0 \) so that KL divergence is well-defined. Thus the discrete-time problem is specified by the dynamics and cost function:

\[
x' \sim u(\cdot|x)
\]

\[
\ell(x, u(\cdot|x)) = q(x) + KL(u(\cdot|x) || P(\cdot|x))
\]

Let \( \mathcal{N} \) denote the set of non-terminal states and \( \mathcal{T} \) the set of boundary states, and let \( q(x) \geq 0 \), \( x \in \mathcal{T} \) be a final cost. Let \( v(x) \) denote the optimal cost-to-go, and define the desirability function:

\[
z(x) = \exp(-v(x))
\]

Let \( \mathcal{G} \) denote the linear operator which computes expectation under the passive dynamics:

\[
\mathcal{G}[z](x) = E_{x' \sim P(\cdot|x)} z(x')
\]

For \( x \in \mathcal{N} \) it can be shown that the optimal control law \( u^*(\cdot|x) \) and the desirability \( z(x) \) satisfy:

\[
u^*(x'|x) = \frac{p(x'|x) z(x')}{\mathcal{G}[z](x)}
\]

\[
\exp(q(x)) z(x) = \mathcal{G}[z](x) \] (1)

At the terminal states \( x \in \mathcal{T} \) we have \( z(x) = \exp(-q(x)) \). For the discrete space case, the linear Bellman equation can be written as

\[
\exp(q_N) \ast z_N = P_{N,T} z_N + P_{N,T} \exp(-q_T) \] (2)

Thus, in the discrete-space case, for the first-exit formulation, the optimal control law is

\[
u^*(j|i) = \frac{P(j|i) z(j)}{\sum_k P(k|i) z(k)} \] (3)

The desirability function \( z \) also has a path-integral representation:

\[
z(x_0) = \sum_{t=1}^{\infty} \sum_{x_{t-1} \in \mathcal{N}, x_t \in \mathcal{T}} \prod_{i=0}^{t-1} P(x_{t+1}|x_t) \exp(-q_t) \]

that is, \( z(x_0) \) is the \( E[\exp(-q(Trajectory))] \) where the expectation is over trajectories sampled from the passive dynamics starting at \( x_0 \) and ending when they first hit a terminal state.

Let us now compare this formulation to traditional MDPs which involve discrete actions \( a \), action-specific transition probabilities \( u_a(\cdot|x) \) and costs \( \ell(x,a) \). Our cost function is more constrained, however it is constrained in a natural way – penalizing the actions for deviating away from the passive dynamics. Our action model on the other hand is less constrained – we allow the controller to choose from a continuum of transition probability distributions. All these distributions have the same set of possible next states, which is the same as the set of possible next state under the passive dynamics. Thus, if we want to prevent the agent from making a transition from \( x_i \) to \( x_j \), we simply set \( P(x_j|x_i) = 0 \). In a grid world for example, we can make \( P \) be uniform over the neighbors of each state and zero everywhere else. In (Todorov, 2009) it is shown how one can embed arbitrary traditional MDPs in this problem class, and conversely, how one can construct traditional MDPs which have the same optimal solution as a problem in this class. The latter construction will be used later in the paper to compare to previous algorithms which assume traditional MDPs. It works as follows: find the optimal \( u^*(\cdot|x) \) for a problem in this class, and define a traditional MDP such that one of the actions (say \( a = 1 \)) has \( u_1(\cdot|x) = u^*(\cdot|x) \) and \( \ell(x,1) = \ell(x,u^*(\cdot|x)) \). Then define other actions \( a \) whose transition probabilities \( u_a(\cdot|x) \) are reshuffled versions of \( u^*(\cdot|x) \) but
can reach the same next states, and whose costs are \( \ell(x,a) = \ell(x,u_a(.|x)) \). For a grid world, one can define a symbolic action for each neighbor and make sure that the reshuffled transition probabilities peak at that neighbor.

### 2.2 Controlled diffusions with linear HJB equations

Consider the control-affine diffusion

\[
dx = a(x) \, dt + B(x) \, (udt + \sigma d\omega) \tag{4}
\]

Here \( a(x) \) is the drift in the passive dynamics (including gravity, Corious and centripetal forces, springs and dampers etc), \( B(x) \) is the effect of the control signal which is now a more traditional vector instead of a probability distribution), and \( \omega(t) \) is a Brownian motion process. The cost function is in the form

\[
\ell(x,u) = q(x) + \frac{1}{2\sigma^2} \| u \|^2 \tag{5}
\]

The function \( a(x), B(x), q(x) \) can be arbitrary. It can be shown that the HJB equation for such problems reduces to a 2nd-order linear PDE when expressed in terms of the desirability function \( z(x) = \exp(-v(x)) \), just like the Bellman equation (1) is linear in \( z \). This similarity suggests that the continuous and discrete problems are somehow related. Indeed it was shown in (Todorov, 2009) that the continuous problem can be obtained from the above MDP by taking a certain limit. The passive dynamics \( P(x'|x) \) of the corresponding discrete-time continuous-state MDP are Gaussian, with mean \( \bar{x} + h a(x) + h B(x) u \) and covariance \( h a^2 B(x) B(x)^T \), where \( h \) is the discretization time step. One can further discretize the state space. The quadratic control cost in the continuous problem turns out to be the limit of the KL divergence control cost in the MDP.

Thus stochastic optimal control problems in the form (4, 5) can be discretized with MDPs in the above problem class. They can also be discretized with traditional MDPs. Both discretization converge to the same solution in the limit \( h \rightarrow 0 \), and in practice seem to be equally accurate away from the limit, but one is much faster to compute than the other (Todorov, 2009). Later in this paper we apply our inverse algorithms to MDPs that are obtained by discretizing continuous problems.

### 3 Efficient Algorithms for Inverse Optimal Control

In this section, we present our algorithms for efficiently solving the inverse optimal control problem. We can learn either the cost or the value function and compute the other using the linear Bellman equation (2). Learning the value function directly is more efficient, although one might need to learn the cost function in some cases, as discussed in section 3.4. In all the algorithms described below, the input to the algorithm is a set of state transitions \( D = \{ x_n, x_n' \}_{n=1,\ldots,N} \) where \( x_n' \) sampled from the optimal control law given \( x_n \):

\[
x_n' \sim u^* (.|x_n)
\]

and the passive dynamics \( P \) and the output is an estimate of the cost function \( q \). It turns out that our algorithms actually only require certain “sufficient statistics” from this input: counts of the number of times the initial state/successor state was \( i \)

\[
a(i) = \sum_n I[x_n = i], b(i) = \sum_i I[x_n = i]
\]

#### 3.1 Likelihood of a set of Transitions

In this section, we write down expressions for the likelihood of a given set of transitions in different ways that will be useful while describe our algorithms for inverse optimal control. We can think of \( u^* (.|\cdot) \) as being parameterized by \( z (\cdot) \) as in (3). Since the transitions are independent,

\[
p(D|z(\cdot)) = \prod_{i} u^*(x_n'|x_n, z(\cdot))
\]

Up to an additive constant, the log-likelihood is

\[
L(z(\cdot)) = \sum_n \log z(x_n') - \log \sum_k p(k|x_n) z(k)
\]

The constant \( \sum_n \log p(x_n'|x_n) \) is omitted because it does not affect maximization w.r.t. \( z \).

It is notable that \( L \) depends on the collection of \( x_n' \)'s and \( x_n \)'s but not on the pairing between them. This enables us to simplify the above expression as follows:

\[
L(D|z) = \sum_i a(i) \log z(i) - \sum_j b(j) \log \sum_k p(k|j) z(k)
\]

One can also write down the likelihood directly in terms of the \( q, v \) leading to algorithms that enable cost approximation for \( q \). To do this, we need to use the linear Bellman equation relationship \( z(i) \exp(q(i)) = \sum_j P(j|i) z(j) \forall j \in \mathcal{N} \), where \( \mathcal{N} \) is the set of non-terminal states in the first-exit formulation, in order to get:

\[
L = \sum_{i \in \mathcal{N}} (b(i) - a(i)) v(i) - \sum_{j \in \mathcal{N}} b(j) q(j) + \sum_{g \in T} a(g) q(g)
\]

(7)
3.2 Learning the Value Function

In this section, we consider learning the value \( v \) or the desirability function \( z = \exp(-v) \) by maximizing the likelihood (6) of the set of observed transitions. Note that it is sufficient to compute \( z \) up to a multiplicative constant as scaling \( z \) (or equivalently adding a constant to each component of the value function) doesn’t change the problem. Written in terms of the value function \( v(i) = -\log(z(i)) \), the likelihood becomes

\[
L = -\sum_i a(i)v(i) - \sum_j b(j)\log\left(\sum_j p(j|i)\exp(-v(i))\right)
\]

(8)

The partial derivatives are

\[
\frac{\partial L(v(\cdot))}{\partial v(i)} = -a(i) + \sum_j b(j)p(i|j)\exp(-v(i))\delta_i^j
\]

and the Hessian is

\[
\frac{\partial L(v(\cdot))}{\partial v(i)}\frac{\partial L(v(\cdot))}{\partial v(s)} = -\sum_j b(j)p(i|j)\sum_k p(k|j)\exp(-v(k))\delta_i^j\delta_s^k
\]

\[
+ \sum_j b(j)p(i|j)p(s|j)\exp(-v(i))\exp(-v(s))\frac{1}{\sum_k p(k|j)\exp(-v(k))}^2
\]

Now, the first term is linear in \( v \) and the second term is concave (sum of negative multiples of the convex log-sum-exp function). Thus, we have a convex optimization problem in terms of \( v \):

**Algorithm optv** Optimize \( L \) by iterating \( v \leftarrow Tv \)

This formulation remains convex even in the setting where we do linear function approximation for the \( v(x) = w^T f(x) \) where \( f(x) \) is a real-valued feature vector describing the state space and \( w \) are the weights assigned to the different features. Once the optimal value function is recovered, the cost function can be recovered from the linear Bellman equation.

3.3 Fixed-Point Iteration Scheme

One can maximize the likelihood (8) wrt \( v \) by standard algorithms like Newton’s method/Conjugate Gradients/Levenberg-Marquardt etc, but in this section we describe an alternative fixed-point iteration scheme for which the per-iteration complexity is much lower. Setting the gradient (9) to 0 and rearranging, we get

\[
\exp(-v^*(i)) = \frac{a(i)}{\sum_j b_j\frac{P(j|i)\exp(-v^*(j))}{\sum_k P(k|i)\exp(-v^*(k))}}
\]

so that

\[
v^*(i) = \log\left(\sum_j b_j\frac{P(j|i)\exp(-v^*(j))}{\sum_k P(k|i)\exp(-v^*(k))}\right) - \log(a(i))
\]

Let \( T : \mathbb{R}^n \rightarrow \mathbb{R}^n \) denote the operator such that

\[
Tv(i) = \log\left(\sum_j b_j\frac{P(j|i)\exp(-v^*(j))}{\sum_k P(k|i)\exp(-v^*(k))}\right) - \log(a(i))
\]

Then \( v^* \) is a fixed point of \( T \). Also, any fixed point of \( T \) is a solution to the maximum likelihood estimation problem. Thus, we have the following algorithm:

**Algorithm optv** Optimize \( L \) by iterating \( v \leftarrow Tv \)

We know show that this is a non-expansive mapping so that the estimates cannot diverge.

\[
Tv_2 - Tv_1 = \left[\frac{\partial Tv}{\partial v}\right]_{v=v_c}(v_2 - v_1)
\]

by the mean value theorem for some \( v_c \).

\[
\frac{\partial (Tv)(i)}{\partial v(t)} = \frac{\sum_j b_jU_{ij}U_{it}}{\sum_j b_jU_{ij}}
\]

where \( U_{ij} = \frac{P(j|i)\exp(-v(j))}{\sum_k P(k|i)\exp(-v(k))} \). The sum of any row of this Jacobian matrix is

\[
\sum_i \frac{\sum_j b_jU_{ij}U_{it}}{\sum_j b_jU_{ij}} = \frac{\sum_j b_jU_{ij}\sum_i U_{it}}{\sum_j b_jU_{ij}} = 1
\]

and hence its \( l_\infty \) norm is 1. Thus, it is a non-expansive map

\[
\|Tv_2 - Tv_1\|_\infty \leq \|v_2 - v_1\|_\infty
\]

3.4 Learning the Cost Function Directly

There are certain scenarios where it becomes necessary to optimize the likelihood wrt the cost rather than the value function: For example, if one wants to consider parameterizing the cost function: \( q(x) = w^T f(x) \) where \( f(x) \) is a vector of basis costs for state \( x \), an approach that has commonly been followed in previous work like (Abbeel & Ng, 2004)(Ziebart et al., 2008), we cannot optimize the value function directly since we need to enforce the constraint \( \frac{\partial}{\partial v}\exp(-q)Pz = \frac{\partial}{\partial z}z \). In the un-parameterized case, we can always choose \( q = Pz/z \) to satisfy this constraint, but in the parameterized case, since \( q(t) \) is not chosen independently for every state, this cannot be done.

Another scenario where this arises is when we
have multiple MDPs sharing the same costs, but having different goal states. This arises, for example, in the driving application modeled by (Ziebart et al., 2008). Here, we have data about routes taken by people driving to different destinations, which can be modeled as a first-exit MDP with the destination as the goal state. In this setting, it is reasonable to assume that the cost of a route is independent of the goal state. Thus, we have trajectories from multiple MDPs sharing a state space and a cost function, but with different goal states, and we need to recover the cost function given these trajectories. Even in this case, it becomes necessary to consider optimizing directly for the cost function.

3.4.1 Likelihood as an Explicit Function of q

In this section, we first show that for the first-exit formulation, the negative log-likelihood of a set of trajectories sampled from the optimal control law can be written as a convex function of q.

From (Todorov, 2009), for the case of a first-exit formulation, we get

\( \text{diag}(\exp(q_N)) - P_N \exp(-q_T) \)

where \( P_N \) is the matrix of transition probabilities between the non-terminal states and \( P_T \) is the vector of transition probabilities from each non-terminal state to the goal state \( g \). From (Todorov, 2009), we also know that \( z \) also has a path-integral representation

\[
z(x_0) = \sum_{t=1}^{\infty} \sum_{x_0 \in T, x_t \in N} \prod_{i=0}^{t-1} P(x_{i+1}|x_i) \exp \left( -\sum_{i=0}^{t} q_t \right)
\]

so that \( \log(z(x_0)) \) is a convex function of \( q \). Even if we parameterize the cost function as \( q = Fw \), \( \log(z(x_0)) \) is a convex function of \( w \). Letting \( M = (\text{diag}(\exp(-q_N))P - I) \) and using equation (7), we can write the negative log likelihood as

\[
\sum_{i \in N} (b(i) - a(i)) \log (M \backslash P_N \exp(-q_T)) + b(i)q(i)
- \sum_{g \in T} a(g)q(g)
\]

If we observe a set of trajectories that always end in terminal state, it is easy to check that \( b(i) \geq a(i) \) \( \forall i \in N \) and hence this is a convex function of \( q \) (or \( w \) for the parameterized case). We have

\[
\frac{\partial L(q)}{\partial q_N} = ((b - a)/z)^T M^{-1} T \star z + b
\]

\[
\frac{\partial L(q)}{\partial q_T} = ((b - a)/z)^T M^{-1} P_N^{-1} T \star \exp(-q_T) - a
\]

Algorithm optq

Optimize \( L \) wrt \( q \) by Conjugate Gradients

In this section, we compare our algorithm to existing algorithms like (Abbeel & Ng, 2004)(Schapire & Syed, 2008)(Ziebart et al., 2008). Since the inverse optimal control problem is ill-posed in the general MDP setting, algorithms like (Abbeel & Ng, 2004) give up on learning the cost function and instead consider the apprenticeship learning problem where one wants to find a policy that is at least as good as that of the observed expert (within limits of finite sampling error). They assume that the cost function is linearly parameterized \( q(x) = w^T f(x) \) using real-valued features \( f(x) \) that summarize the relevant properties of the state for computing the cost. They then compute estimates of these feature expectations from the sampled trajectories of the expert assuming a fixed initial state distribution. Given these estimates, the algorithms iteratively update the weights \( w \) producing a new policy at each time effectively covering the space of all possible \( w \)s and producing a policy that is guaranteed to work for any choice of \( w \). This makes them require repeated access to an MDP solver (Abbeel & Ng, 2004) or further assumptions on the weights (for example that they form a convex combination (Schapire & Syed, 2008)). Also, both these algorithms require specification of an initial state distribution while our algorithm works irrespective of the initial state distribution and computes estimates of the value function for each state separately.

A further computational bottleneck we observed in our implementation of the (Abbeel & Ng, 2004) algorithm is that it needs to compute estimates for the feature expectation under the optimal policy for every set of weights \( w \) produced by the algorithm. Computing this estimate, particularly in the first-exit formulation, is quite slow as one needs to sample repeatedly from the optimal policy until one hits the goal state (as opposed to the discounted case where trajectories of small lengths are enough due to the discount factor diminishing the effect of states further in time).

Our framework is also closely related to the maximum entropy IRL algorithm (Ziebart et al., 2008). In
(Abbeel & Ng, 2004), the authors show that for apprenticeship learning, it is necessary and sufficient to match the expectations of features accumulated along a path with that of the expert. However, this is still an ambiguous notion since there are many policies that achieve this condition. In order to resolve this ambiguity, in (Ziebart et al., 2008), the authors propose considering the MaxEnt distribution subject to those expectation constraints. In this section, we show that this rather ad-hoc approach is actually a special case of our framework.

Consider the linearly solvable MDP framework under the infinite horizon first-exit formulation with non-terminal states $N$ and terminal states $T$. Let $P$ denote the passive dynamics of the system. Under the optimal control, the probability of a trajectory is (Todorov, 2009)

$$u(x_{0:T}|x_0) = \frac{P(x_{0:T}|x_0) \exp \left( - \sum_{k=0}^{T} q(x_k) \right)}{z(x_0)}$$

By the path integral representation for $z$, the denominator is just the sum of $P(x_{0:T}|x_0) \exp \left( - \sum_{k=0}^{T} q(x_k) \right)$ over all possible paths of different lengths starting at $x_0$ and ending at some terminal state. If we parametrize $P(j|i) \exp(-q(i)) = \exp(w^T f(i,j))$, for a path $\zeta = x_{0:T}$, we have

$$u(\zeta|x_0) = \frac{\exp(w^T f(\zeta))}{\sum_{\zeta'} \exp(w^T f(\zeta'))}$$

where $f(\zeta) = \sum_{t=0}^{T-1} f(x_t, x_{t+1})$ which is exactly the MaxEnt IRL distribution.

4 Experiments

4.1 Discretized continuous-state problems

Here we apply our new algorithm to MDPs obtained by discretizing continuous problems, as outlined in Background. We consider two dynamical systems: an inverted pendulum with joint limits, and a car-on-a-hill. For each we define a first-exit problem with terminal cost 0, and $q(x) = \text{const}$ encoding the desire to reach a terminal state as soon as possible. For the car-on-a-hill dynamics we also define an infinite-horizon average-cost problem where the goal is to cycle between two states with 0 cost (all other states incur a positive cost). We force the system to cycle (as opposed to stay in the goal states) by choosing goal states with non-zero velocity. The continuous dynamical systems have 2D dynamics - position and velocity. We discretize this state space with a 100-by-100 grid of 10,000 states. The time axis is discretized with $h = 0.05$ sec.

The left column of figure 1 shows the optimal $v^*(\cdot)$ for each forward problem. We then construct the optimal controller $u^*(\cdot|x)$, sample from it, define the negative log-likelihood as a function of $v(\cdot)$, and minimize it with a Levenberg-Marquard method taking advantage of sparsity. We run the algorithm for 20 iterations which took about 1 minute of CPU time (3 sec per iteration). The right column show the solutions found by the inverse method. The white pixels are state to which the system did not transition, thus inferring $v$ is not possible at those states. The solid curve in each figure shows how the negative log-likelihood decreased over the 20 iterations. The dashed curve shows the mean absolute deviation (over all states) between the current solution and $v^*(\cdot)$. Note that for the infinite horizon problem we had to sample more data in order to achieve reasonable reconstruction. We do not yet know if this is a generic property or something specific to this system.
4.2 GridWorld

Figure 2 shows the true value function of a simple gridworld with a uniform cost on all the states with a zero cost absorbing goal state on the bottom-right corner. We ran our optv algorithm on this with 5000 trajectories sampled from the optimal control law as input and obtained as estimate of the value function plotted in Figure 3. We also ran our algorithms on standard gridworlds and compared it with other algorithms (Ng & Russell, 2000)(Schapire & Syed, 2008)(Abbeel & Ng, 2004). In general, our algorithms beat (Ng & Russell, 2000)(Abbeel & Ng, 2004) by large margins in both running time, accuracy of estimated cost function and performance of the learned controller. Performance of the learned controller was comparable to that of (Schapire & Syed, 2008), but our algorithms are faster. This algorithm does not give an estimate of the cost function.

We also constructed gridworld MDPs such that the optimal solution is always the same as an equivalent problem from our class as explained in section 2.1. Thus, we can use these to compare our algorithms to algorithms that work for traditional MDPs.

Figures 4,5,6 show comparisons of errors in the estimate of the cost function, performance of the learned controller and CPU times for all the algorithms. All plots have the y-axis on a log(base e) scale. Errors in the cost function are measured by relative deviation in the $l_2$ norm. The plots in figure 4,6 are versus varying grid sizes from 3x3 to 7x7 and 5 is versus varying sample size.
4.3 Random MDPs

We also compared the running times of our algorithms to existing algorithms on randomly generated MDPs with 100 states. The results averaged over 30 randomly generated MDPs are plotted in figure 7. We also have plots of how running time and error in the cost function estimates decrease for our algorithm with increasing sample size (Figure 8). We do not present results for the Abbeel and Ng algorithm here because it was too slow to run repeatedly on randomly generated MDPs.

5 Conclusions

In this paper we presented inverse methods for a recently-developed class of stochastic optimal control problems which have the property that the Bellman equation is linear, even though the dynamics and state costs can be arbitrary. This problem class is quite general and make it possible to solve a wide range of forward problems more efficiently than previously possible. Not surprisingly, the corresponding inverse methods also turn out to be more efficient than prior inverse methods for generic MDPs. In particular, we found that solving the inverse problem comes down to optimizing a log-likelihood function which is an unconstrained convex program in the unknown value function. As a result, our algorithms outperformed prior algorithms both in terms of computational efficiency and accuracy. In addition, we are able to recover both the cost function, the value function and the control policy, while most prior algorithms can recover only the policy accurately.

Inverse optimal control has usually been studied in the context of function approximators using features. In this paper the "features" were delta functions centered at every state. Our algorithms can of course benefit from a well-chosen function approximator, like any other algorithm. Indeed we view the choice of algorithm and representation to be largely orthogonal. The possibility of creating good function approximators for certain problems does not obviate the need for having efficient algorithms. This point is particularly important in dealing with continuous systems, where any plausible function approximator is likely to have a large number of bases – perhaps on the order of thousands. Our algorithm illustrated in Fig 1 can solve for 10,000 unknown coefficients in about a minute of CPU time. We find this performance very encouraging as we move towards more complex problems where function approximation will be unavoidable.
References


