Inverse Optimal Control with Linearly-Solvable MDPs

Krishnamurthy Dvijotham DVIJ@CS.WASHINGTON.EDU
Emanuel Todorov TODOROV@CS.WASHINGTON.EDU
Computer Science & Engineering and Applied Mathematics, University of Washington, Seattle - 98105, USA

Abstract

We present new algorithms for inverse optimal control (or inverse reinforcement learning, IRL) within the framework of linearly-solvable MDPs (LMDPs). Unlike most prior IRL algorithms which recover only the control policy of the expert, we recover the policy, the value function and the cost function. This is possible because here the cost and value functions are uniquely defined given the policy. Despite these special properties, we can handle a wide variety of problems such as the grid worlds popular in RL and most of the nonlinear problems arising in robotics and control engineering. Direct comparisons to prior IRL algorithms show that our new algorithms provide more information and are orders of magnitude faster. Indeed our fastest algorithm is the first inverse algorithm which does not require solving the forward problem; instead it performs unconstrained optimization of a convex and easy-to-compute log-likelihood. Our work also sheds light on the recent Maximum Entropy (MaxEntIRL) algorithm, which was defined in terms of density estimation and the corresponding forward problem was left unspecified. We show that MaxEntIRL is inverting an LMDP, using the less efficient of the algorithms derived here. Unlike all prior IRL algorithms which assume pre-existing features, we study feature adaptation and show that such adaptation is essential in continuous state spaces.

1. Introduction

Inverse optimality has attracted considerable attention in both control engineering and machine learning. Unlike the forward problem of optimal control which is well-defined, the inverse problem can be posed in multiple ways serving different purposes.

Inverse optimality was first studied for control-theoretic purposes in relation to stability (Kalman, 1964). This idea later inspired a constructive approach (e.g. (Deng & Krstic, 1997)) where one designs a control-Lyapunov function, treats it as an optimal value function (and derives the corresponding control law) and finds the cost for which this value function is optimal. Apart from the guesswork involved in designing control-Lyapunov functions, this is easier than solving the forward problem because for many nonlinear systems (see Section 3) the Hamilton-Jacobi-Bellman (HJB) equation gives an explicit formula for the cost once the value function is known. The LMDPs we will be working with (Todorov, 2007; 2009b) also have this property, and it will play a key role here.

It is notable that the above control-theoretic approach does not actually use data. In contrast, IRL methods in machine learning rely on data in the form of state transitions (and possibly actions) obtained from an expert performing some task. In general there are two things that one could do with such data: infer the costs/values of the expert, or build a controller which mimics the expert. The former is relevant to cognitive and neural science, where researches are interested in "theories of mind" (Baker et al., 2007) as well as in identifying the cost functions being optimized by the sensorimotor system (Todorov, 2004; Körding & Wolpert, 2004). While many existing IRL algorithms (Ng & Russell, 2000; Abbeel & Ng, 2004; Syed et al., 2008) use cost features and infer weights for those features, they do not actually aim to recover the cost or value function but only the control law. Indeed in generic MDPs there is a continuum of cost and value functions for which a given control law is optimal (Ng & Russell, 2000). This ill-posedness is removed in the LMDP framework, making our algorithms much more applicable to cognitive and neural science.

Now consider the task of building a control law from

Appearing in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010.

data – which is what the above IRL methods do. One reason to use data (instead of solving the forward problem directly) is that an appropriate cost function which captures the control objectives may be hard to design. But we believe this difficulty is negligible compared to the second reason – which is that we lack algorithms capable of solving forward optimal control problems for complex systems. Take for example the control domain where data has been used most extensively, namely locomotion and other full-body movements seen in movies and games. A sensible cost function for locomotion is not hard to design: it should require the center of mass to remain a certain distance above ground (to prevent falling), move at a certain speed towards the goal, and at the same time conserve energy. Indeed open-loop optimization of similar costs for simplified models can predict various features of human walking and running (Srinivasan & Ruina, 2005). If we could find feedback control laws which optimize such costs for realistic systems, this would constitute a major breakthrough both in animation and in robotics. Unfortunately this is not yet possible, and thus many researchers are exploring ways to build controllers using motion capture data (e.g. (Treuille et al., 2007)). We emphasize this point here because all prior IRL methods we are aware of, including the MaxEntIRL method discussed later (Ziebart et al., 2008), end up solving the forward problem repeatedly in an inner loop. While one can construct problems with moderate numbers of discrete states where such an approach is feasible, scaling it to control problems that involve interesting physical systems is unlikely. A case in point is the elegant work of Abbeel and colleagues on aerobatic helicopter flight (Abbeel et al., 2007). After trying to apply their apprenticeship learning framework (Abbeel & Ng, 2004) to this problem, they eventually gave up and simply recorded reference trajectories from human radio-pilots. If future IRL algorithms are to avoid this fate, they should avoid solving the forward problem. Here we develop the first inverse method which avoids solving the forward problem. This is done by parameterizing and inferring the value function rather than the cost function, and then computing the cost function using an explicit formula.

Finally, all IRL methods including ours use linear combinations of features to represent the costs (or values in our case). However, previous work has left the choice of features to manual design. This is arguably one of the biggest unsolved problems not only in IRL but in AI and machine learning in general. Here we consider automatic methods for initializing the parameterized features and methods for adapting their parameters. When the number of features is small relative to the size of the state space (which is always the case in high dimensional problems), feature adaptation turns out to be essential. While the problem of feature adaptation in IRL is far from being solved in its generality, the present work is an important first step in this direction.

2. Discrete problems

We consider problems with discrete state space in this section, and problems with continuous state space in the next section. In both cases we derive IRL algorithms from the recently-developed framework of linearly-solvable stochastic optimal control (Todorov, 2007; 2009b; Kappen, 2005). Below we first summarize this framework from the viewpoint of the forward problem, and then present our results regarding the inverse problem.

2.1. Linearly-solvable MDPs

The MDP is defined by a state cost $q(x) \geq 0$, 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 $x' \sim \pi(\cdot|x)$ it wishes, however it pays a price (control cost) which is the KL divergence between π and p. We further require that $\pi(x'|x) = 0$ whenever $p(x'|x) = 0$ so that KL divergence is well-defined. Thus the cost function is

$$\ell(x, \pi(\cdot|x)) = q(x) + KL(\pi(\cdot|x) \,||\, p(\cdot|x)) \tag{1}$$

Define the *desirability* function $z(x) = \exp(-v(x))$ where $v(x)$ is the optimal value function (or the differential value function in average-cost settings). It can now be shown that the optimal control law is

$$\pi^*(x'|x) = \frac{p(x'|x)\, z(x')}{\mathcal{G}[z(\cdot)](x)} \tag{2}$$

where the normalizing term $\mathcal{G}$ is a linear operator which computes next-state expectations under p:

$$\mathcal{G}[z(\cdot)](x) = \sum_{x'} p(x'|x)\, z(x') \tag{3}$$

The minimized Bellman equation is linear in z:

$$\lambda z(x) = \exp(-q(x))\, \mathcal{G}[z(\cdot)](x) \tag{4}$$

In infinite-horizon average-cost problems the solution corresponds to the principal eigen-pair, and the average cost is $-\log(\lambda)$. In first-exit problems we have $\lambda = 1$ and (4) becomes a linear algebraic equation, whose solution in vector notation is

$$\mathbf{z}_{\mathcal{N}} = \left(\mathrm{diag}\left(\exp(\mathbf{q}_{\mathcal{N}})\right) - P_{\mathcal{N}\mathcal{N}}\right)^{-1} P_{\mathcal{N}\mathcal{T}} \exp(-\mathbf{q}_{\mathcal{T}}) \tag{5}$$

Here $\mathcal{N}$ and $\mathcal{T}$ are the sets of non-terminal and terminal states. Infinite-horizon discounted-cost problems (with discount factor α) can also be handled by defining $z(x) = \exp(-\alpha v(x))$. In that case (2, 3) remain the same while (4) becomes nonlinear, namely $z(x)^{-\alpha} = \exp(-q(x))\,\mathcal{G}[z(\cdot)](x)$.

In first-exit problems, the probability that the passive dynamics generate trajectory $\varsigma = (x_1, x_2, \cdots, x_T \in \mathcal{T})$ given initial state x_0 is $p(\varsigma|x_0) = \prod_{t=1}^{T} P(x_t|x_{t-1})$. The probability that the same trajectory is generated by the optimal control law is

$$p^*(\varsigma|x_0) = \frac{p(\varsigma|x_0)\exp\left(-\sum_{t=0}^{T} q(x_t)\right)}{z(x_0)} \tag{6}$$

This formulation is different from traditional MDPs, and a critic might argue that IRL algorithms based on it do not solve the problem one wants to solve. We have two responses to this criticism. First, this formulation should not be considered less natural just because it is more recent. Indeed MDPs are often used to model the physical world which has continuous actions and non-trivial passive dynamics. Second, traditional MDPs can be embedded in this problem class (Todorov, 2007; 2009b) in a way reminiscent of linear programming relaxation in integer programming: it is not guaranteed to yield the same result but often yields very similar results much faster.

2.2. Inverse optimal control (OptV)

We now turn to the inverse problem. Unlike prior IRL algorithms which require trajectory data, our algorithms work with any dataset of transitions $\{x_n, x'_n\}_{n=1\cdots N}$ sampled from the optimal control law:

$$x'_n \sim \pi^*(\cdot|x_n) \tag{7}$$

We are also given the passive dynamics p. Our objective is to estimate the cost q, the desirability function z, the optimal value function v and the optimal control law π^*. Conveniently we have explicit formulas relating these quantities, thus it is sufficient to infer one of them. For reasons explained below it is most efficient to infer v. Once we have an estimate $\widehat{v}$, we can obtain $\widehat{z} = \exp(-\widehat{v})$, $\widehat{\pi}^*$ from (2), and $\widehat{q}$ from (4).

The inference method is maximum likelihood. Think of the optimal control law $\pi^*(\cdot|\cdot)$ as being parameterized by the desirability function $z(\cdot)$ as given by (2). Then the negative log-likelihood is

$$L[z(\cdot)] = -\sum_n \log z(x'_n) + \sum_n \log \sum_{x'} p(x'|x_n)\, z(x') \tag{8}$$

We have omitted the term $\sum_n \log p(x'_n|x_n)$ because it does not depend on z, although this term could be used in future work attempting to learn p under some regularizing assumptions. Now L could be minimized w.r.t. z, however it is not a convex function of z. We have experimented with such minimization and found it to be slower as well as prone to local minima.

If however we write L in terms of v it becomes convex – because it is a positive sum of log-sum-exp functions plus a linear function. One additional improvement, which enables us to compute L faster when the number of data points exceeds the number of states, is to write L in terms of the visitation counts $a(x')$ and $b(x)$ defined as the number of times $x'_n = x'$ and $x_n = x$ respectively. It is interesting that the likelihood depends only on these counts and not on the specific pairings of states in the dataset. We now have

$$L[v(\cdot)] = \sum_{x'} a(x')\, v(x') + \sum_x b(x) \log \sum_{x'} p(x'|x) \exp(-v(x')) \tag{9}$$

Thus inverse optimal control in the linearly-solvable MDP framework reduces to unconstrained convex optimization of an easily-computed function. We will call the resulting algorithm **OptV**. In our current implementation we compute the gradient and Hessian of (9) analytically and apply Newton's method with backtracking linesearch.

We did not distinguish between first-exit, average-cost and discounted problems because the algorithm is the same in all three cases; the only differences are in how the data are sampled and how $\widehat{q}$ is subsequently computed from $\widehat{v}$. This is an advantage over other IRL methods which are usually derived for a single problem formulation.

Finally, the above discussion implied lookup-table representations, however it is easy to use features as well. Consider a linear function approximator in v-space:

$$v(x) = \sum_i w_i f_i(x) \tag{10}$$

where $f_i(x)$ are given features and w_i are unknown weights. Then $L(\mathbf{w})$ is again convex and can be optimized efficiently. Later in the paper we consider methods for initializing and adapting the features automatically when the state space is continuous.

2.3. Learning the cost directly (OptQ)

We can also express L as a function of q and infer q directly (algorithm **OptQ**). When using lookup-table representations the two algorithms yield identical results, however the results are generally different when

using features. This is because the transformation between v and q given by (4) is nonlinear, thus a linear function approximator in v-space does not correspond to a linear function approximator in q-space. A second reason to explore direct inference of q is because this turns out to reveal an interesting relationship to the MaxEntIRL algorithm (Ziebart et al., 2008).

For simplicity we focus on first-exit problems where we have the explicit formula (5) relating z and q. This formula enables us to express L as a function of q and compute the gradient analytically – which is cumbersome due to the matrix inverse, but doable. Computing the Hessian however is too cumbersome, so we use a BFGS method which approximates the Hessian. L turns out to be convex in q (see Appendix). Nevertheless the OptQ algorithm is much slower than the OptV algorithm. This is because computing $L\left[q\left(\cdot\right)\right]$ requires solving the forward problem at every step of the minimization. Therefore learning q directly is not a good idea. If one wants to use features in q-space, it may be better to do the learning in v-space (perhaps with a different set of features) and then fit the function approximator for q using linear regression.

The function $L\left[q\left(\cdot\right)\right]$ can be written in an alternative form using the trajectory probabilities (6). Suppose the transitions are sampled along trajectories $\varsigma^{(k)}$ with lengths $T\left(k\right)$, and let $x_t^{(k)}$ denote the state at time t along trajectory k. Using (6) and omitting the p-dependent term which does not involve q, we have

$$L\left[q\left(\cdot\right)\right] = \sum_k \left(\log z\left(x_0^{(k)}\right) + \sum_{t=0}^{T(k)} q\left(x_t^{(k)}\right) \right) \quad (11)$$

Again we see that computing $L\left[q\left(\cdot\right)\right]$ requires $z\left(\cdot\right)$.

2.4. Relationship with MaxEntIRL

The MaxEntIRL algorithm (Ziebart et al., 2008) is derived using features (which can also be done in OptV and OptQ) but for simplicity we discuss the lookup-table case with one delta function "feature" per state. MaxEntIRL is a density estimation algorithm: it looks for the maximum-entropy distribution consistent with the observed state visitation counts (or feature counts more generally). It is known that the maximum-entropy distribution under moment-matching constraints is in the exponential family. Thus MaxEntIRL comes down to finding $q\left(\cdot\right)$ which maximizes the probability of the observed trajectories within the family

$$p_{\text{MaxEnt}}\left(\varsigma|x_0\right) \propto \exp\left(-\sum_{t=0}^{T} q\left(x_t\right)\right) \quad (12)$$

The bottleneck is in computing the partition function at each step of the optimization, which is done using a recursive procedure.

Intuitively MaxEntIRL resembles an IRL method. However until now it was unclear what forward optimal control problem is being inverted by MaxEntIRL, and whether such a problem exists in the first place. We can now answer these questions. Comparing (12) to (6), we see that the trajectory probabilities are identical when the passive dynamics are uniform. Therefore MaxEntIRL is an inverse method for LMDPs with uniform passive dynamics. Indeed the recursion used in (Ziebart et al., 2008) to compute the partition function is very similar to the iterative method for computing the desirability function in (Todorov, 2007; 2009b). Both recursions are computationally equivalent to solving the forward problem. As a result both MaxEntIRL and OptQ are slower than OptV, and furthermore MaxEntIRL is a special case of OptQ. MaxEntIRL's restriction to uniform passive dynamics is particularly problematic in modeling physical systems, which often have interesting passive dynamics that can be exploited for control purposes (Collins et al., 2005).

2.5. Embedding Arbitrary IRL Problems

In this section we show how an IRL problem for a traditional MDP can be embedded in the LMDP framework. This is almost the same as the embedding described in (Todorov, 2009b), except that here we do not know the cost function during the embedding, thus we need some additional assumptions. We assume that the MDP cost is in the form $l(x)+r(a)$ where $r(a)$ is a known action cost while $l(x)$ is an unknown state cost. Let $p(x'|x,a)$ be the (known) transition probabilities in the MDP, and assume that the number of actions per state equals the number of possible next states. Let $q(x)$ and $p(x'|x)$ be the unknown state cost and passive dynamics in the corresponding LMDP. The embedding (Todorov, 2009b) comes down to matching the costs for all x, a:

$$l(x) + r(a) = q(x) + \sum_{x'} p(x'|x,a) \log\left(\frac{p(x'|x,a)}{p(x'|x)}\right)$$

These equations are linear in $\log(p(x'|x))$. Let us fix x and suppose k states are reachable from x in one step. Then $p(x'|x)$ has at most k non-zeros (to ensure finite KL divergence). Let the non-zeros be stacked into the vector p_x. Thus we have k linear equations in $k+1$ variables $\log(p_x), l(x)-q(x)$. The additional degree of freedom is removed using $1^T p_x = 1$. We can then solve the LMDP IRL problem, and use the solution as an approximation to the MDP IRL problem. There are no guarantees on the quality of the recovered solution, but we observe that it gives good results experimentally in section 2.6.

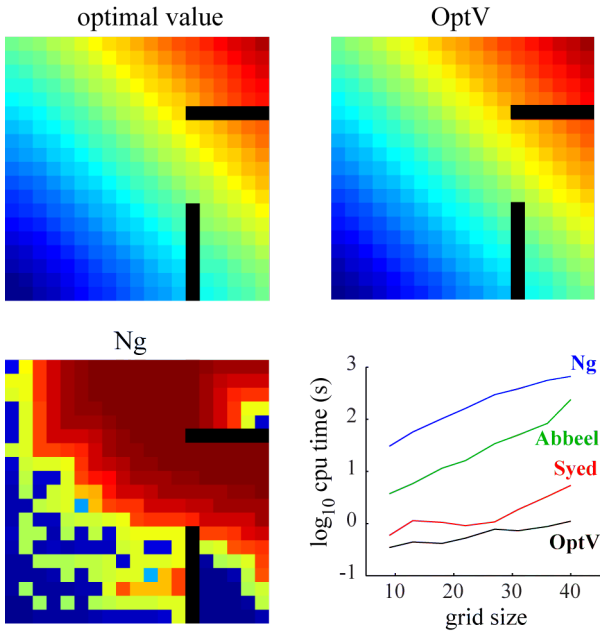

Figure 1. Comparison of OptV and prior IRL algorithms on a grid-world problem. Black rectangles are obstacles.

2.6. Numerical results

We compared OptV to three prior IRL algorithms labeled in Figure 1 according to the name of their first author: Syed (Syed et al., 2008), Abbeel (Abbeel & Ng, 2004), and Ng (Ng & Russell, 2000). The forward problem is a traditional MDP: a grid world with obstacles (black rectangles), a state-action cost which only depends on the state, and discrete actions causing transitions to the immediate neighbors (including diagonals). There is one action per neighbor and it causes a transition to that neighbor with probability 0.9. The rest of the probability mass is divided equally among the remaining neighbors. The problem is in a discounted-cost setting with discount factor 0.5.

All four IRL methods were implemented in Matlab in the most efficient way we could think of. Rather than sampling data from the optimal control policy, we gave them access to the true visitation frequencies under the optimal policy of the traditional MDP (equivalent to infinite sample size). Using the embedding from section 2.5, we get an embedded LMDP IRL problem with passive dynamics that is uniform over possible next states and run OptV on this.

As expected, OptV was substantially faster than all other algorithms for all grid sizes we tested. Even though the forward problem which generated the data is a traditional MDP while OptV is trying to invert an LMDP, it infers a value function very similar to the solution to the forward problem. Since the passive dynamics here are uniform, MaxEntIRL/OptQ produce the same result but about 20 times slower. Although such close similarity is not guaranteed, it is common in our experience. (Ng & Russell, 2000) proposes a heuristic to select a cost function – which we then translated into a value function by solving the forward problem, while the other algorithms only recover a policy, not a cost function. As shown in the figure, the result is quite different from the correct value function.

Two of the prior IRL algorithms (Syed and Abbeel) are guaranteed to recover the control policy given the true visitation counts, and indeed they do. Since OptV is solving a different problem it does not recover the control policy exactly (which it would if the forward problem was an LMDP). Nevertheless the result is very close, and actually improves when the grid size increases. The expected cost achieved by the inferred policy was 6% above optimal for the 9-size grid, and only 0.3% above optimal for the 40-size grid. Thus we pay a small penalty in terms of performance of the inferred policy, but we recover costs/values and do so faster than any other algorithm.

3. Continuous problems

We now focus on optimal control problems in continuous space and time. Such problems lead to PDEs which in our experience are difficult to handle numerically. Therefore the new IRL method we derive below (OptVA) uses time-discretization, along with adaptive bases to handle the continuous state space. We also consider state discretization as a way of obtaining large MDPs on which we can further test the algorithms from the previous section (see Figure 2A below).

3.1. Linearly-solvable controlled diffusions

Consider the control-affine Ito diffusion

$$d\mathbf{x} = \mathbf{a}(\mathbf{x})\,dt + B(\mathbf{x})\,(\mathbf{u}dt + \sigma d\omega) \tag{13}$$

where $\mathbf{a}(\mathbf{x})$ is the drift in the passive dynamics (including gravity, Coriolis and centripetal forces, springs and dampers etc), $B(\mathbf{x})\,\mathbf{u}$ 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

$$\ell(\mathbf{x},\mathbf{u}) = q(\mathbf{x}) + \frac{1}{2\sigma^2}\|\mathbf{u}\|^2 \tag{14}$$

The relationship between the noise magnitude and the control cost is unusual but can be absorbed by scaling q. The only restriction compared to the usual control-

affine diffusions studied in the literature is that the noise and controls must act in the same space.

It can be shown (Kappen, 2005; Todorov, 2009b) that the HJB equation for such problems reduces to a 2nd-order linear PDE when expressed in terms of the desirability z, just like the Bellman equation (4) is linear in z. This similarity suggests that the above problem and the linearly-solvable MDPs are somehow related. Indeed it was shown in (Todorov, 2009b) that problem (13, 14) can be obtained from a discrete-time continuous-state LMDP by taking a certain limit. The passive dynamics for this MDP are constructed using explicit Euler discretization of the time axis: $p(\mathbf{x}'|\mathbf{x})$ is Gaussian with mean $\mathbf{x} + h\mathbf{a}(\mathbf{x}) + hB(\mathbf{x})\mathbf{u}$ and covariance $h\sigma^2 B(\mathbf{x}) B(\mathbf{x})^{\mathsf{T}}$, where h is the time step. The state cost in the MDP is $hq(\mathbf{x})$. It can be shown that the quadratic control cost in (14) is the limit of the KL divergence control cost in (1) when $h \to 0$.

Thus the continuous optimal control problem (13, 14) is approximated by the LMDP described above, and IRL methods for this LMDP approximate the continuous inverse problem. The approximation error vanishes when $h \to 0$. However, time-discretization allows us to use larger h, which usually leads to better performance for a given number of samples and bases.

3.2. Inverse optimal control with adaptive bases (OptVA)

The inverse method developed here is similar to OptV, however it uses a function approximator with adaptive bases. We represent the value function as

$$v(\mathbf{x}; \mathbf{w}, \theta) = \textstyle\sum_i w_i f_i(\mathbf{x}; \theta) = \mathbf{w}^T f(\mathbf{x}; \theta) \tag{15}$$

where $\mathbf{w}$ is a vector of linear weights while θ is a vector of parameters that affect the shape and location of the bases f_i. The bases are normalized Gaussian RBFs:

$$f_i(\mathbf{x}; \theta) = \frac{\exp\left(\theta_i^{\mathsf{T}} \mathbf{s}(\mathbf{x})\right)}{\sum_j \exp\left(\theta_j^{\mathsf{T}} \mathbf{s}(\mathbf{x})\right)} \tag{16}$$

Here θ_i denotes the part of θ specific to f_i, and $\mathbf{s}(\mathbf{x}) = [1; x_k; x_k x_l]$ for all $k \le l$. Thus $\exp\left(\theta_i^{\mathsf{T}} \mathbf{s}(\mathbf{x})\right)$ is Gaussian. In the language of exponential families, θ_i are the natural parameters and $\mathbf{s}(\mathbf{x})$ the sufficient statistics. We chose normalized RBFs because they often produce better results than unnormalized RBFs – which we also found to be the case here.

Similar to the discrete case, the negative log-likelihood of a dataset $\{\mathbf{x}_n, \mathbf{x}'_n\}$ is

$$L(\mathbf{w}, \theta) = \sum_n \mathbf{w}^T f(\mathbf{x}'_{\mathbf{n}}; \theta) + \log \mathcal{G}\left[e^{-\mathbf{w}^T f(\mathbf{x}; \theta)}\right](\mathbf{x}_n)$$

where the linear operator $\mathcal{G}$ is the same as (3), except that the sum becomes an integral.

Thus L is convex in $\mathbf{w}$ and can be minimized efficiently for fixed θ. The optimization of θ, or in other words the basis function adaptation, relies on gradient descent – LBFGS or Conjugate Gradients as implemented in the off-the-shelf optimizer (Schmidt, 2005). We take advantage of the convexity in $\mathbf{w}$ by optimizing $\widetilde{L}(\theta) = \min_{\mathbf{w}} L(\mathbf{w}, \theta)$. Each evaluation of $\widetilde{L}(\theta)$ involves computing the optimal $\mathbf{w}^*(\theta)$ by Conjugate Gradients (which converges very quickly). Then we compute the gradient of $\widetilde{L}$ using

$$\frac{\partial \widetilde{L}(\theta)}{\partial \theta} = \frac{\partial L(\mathbf{w}^*(\theta), \theta)}{\partial \mathbf{w}} \frac{\partial \mathbf{w}^*(\theta)}{\partial \theta} + \frac{\partial L(\mathbf{w}^*(\theta), \theta)}{\partial \theta}$$

The first term on the right vanishes because L has been optimized w.r.t. $\mathbf{w}$. The only complication here is the computation of $\mathcal{G}\left[e^{-\mathbf{w}^T f(\mathbf{x}; \theta)}\right] = \int p(\mathbf{x}'|\mathbf{x}) \exp\left(-\mathbf{w}^T f(\mathbf{x}; \theta)\right) d\mathbf{x}'$. In our current implementation we do this by discretizing the state space around $E_p[\mathbf{x}'|\mathbf{x}]$ and replacing the integral with a sum. In high-dimensional problems such discretization will not be feasible. However the passive dynamics $p(\mathbf{x}'|\mathbf{x})$ are Gaussian, and numerical approximation methods for Gaussian integrals have been studied extensively, resulting in so-called cubature formulas which can be applied here.

The optimization problem is convex in w but non-convex in the basis parameters θ, thus we need good initialization for θ. We developed an automated procedure for this. The intuition is that the optimal controller frequently visits "good" parts of the state space where the function approximator should have the highest resolution. Thus the centers of the Gaussians are initialized using K-means on the data. The function approximator can also benefit from initializing the covariances properly. We do this by finding the nearest Gaussians, computing the covariance of their means, and scaling it by a constant.

We argued earlier that data makes the inverse problem generally easier than the forward problem. Is this still true in the LMDP case given that the forward problem is linear? For fixed features/bases the two computations are comparable, however basis adaptation is much easier in the inverse problem. This is because the data provides good initialization, and good initialization is key when optimizing a non-convex function.

3.3. Numerical results

Here we study inverted pendulum dynamics in the form (13, 14), with $\sigma = 1$. The state space is 2D:

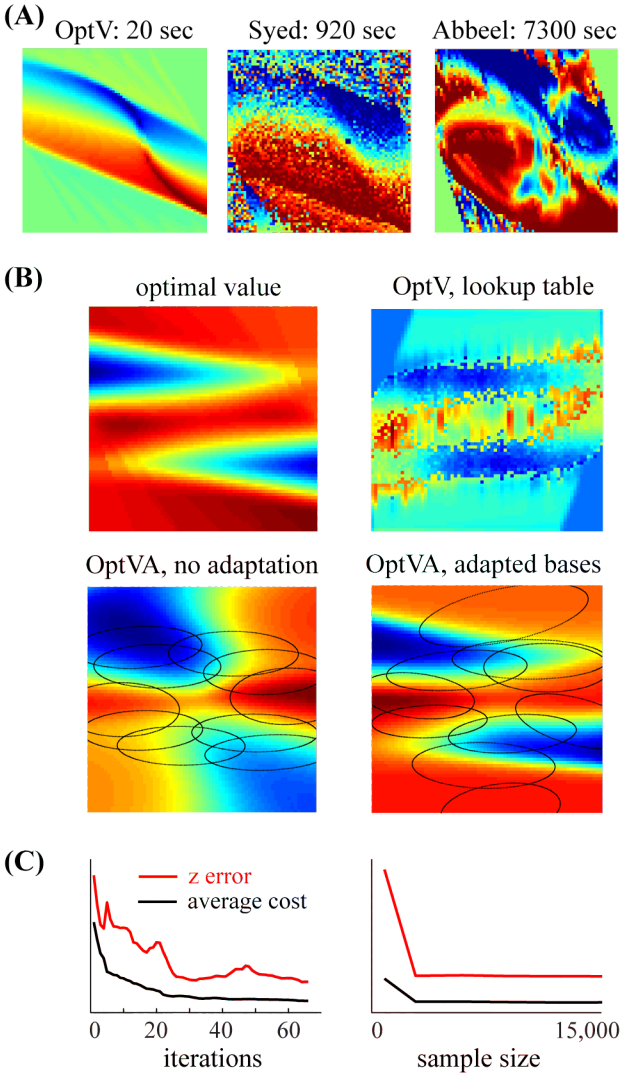

Figure 2. **(A)** Control policies in the first-exit (inverted pendulum) problem. Each subplot shows the CPU time and the policy found given the optimal transition probabilities. The policy found by OptV was indistinguishable from the optimal policy and achieved average cost of 13.06, as compared to 57.21 for Syed and 41.15 for Abbeel. **(B)** Value functions in the infinite-horizon (metronome) problem. Here the algorithms have access to finite data (12,000 transitions) thus the optimal value function can no longer be recovered exactly. OptV with a lookup table representation does quite poorly, indicating the need for smoothing/generalization. The result of OptVA with the initial bases vaguely resembles the correct solution, and is substantially improved after basis adaptation. The ellipses show the location and shape of the Gaussian bases before normalization. **(C)** Performance of OptVA over iterations of basis adaptation for 12,000 samples (left), and as a function of the sample size at the last iteration of basis adaptation (right). We plot the difference between the optimal and inferred z functions (expressed as KL divergence), and the log average cost of the resulting control policy. The curves are scaled and shifted to fit on the same plot.

$\mathbf{x} = [x_p; x_v]$. We consider a first-exit formulation where the goal is to reach a small region around the vertical position with small velocity. We also consider an infinite-horizon average-cost formulation corresponding to a metronome. The cost $q(\mathbf{x})$ only depends on x_v. It is small when $x_v = \pm 2.5$ and increases sigmoidally away from these values. Thus the pendulum is required to move in either direction at constant speed. The system has positional limits; when these limits are hit the velocity drops to zero. The discretization time step is $h = 0.1$. In the first-exit problem the state space is also discretized, on a 70-by-70 grid.

Figure 2A shows further comparison to prior IRL algorithms in a discretized state space using lookup table representaiton (pendulum first exit problem). OptV is faster by orders of magnitude, and recovers the optimal policy almost exactly (relative error $< 10^{-9}$), while prior ILR algorithms recover different policies with significantly worse performance. We used the LP solver Gurobi (Yin, 2009-2010) to implement the Syed-Schapire algorithm. We discretized the actions space with a grid of 70 points for Syed-Schapire and 10 points for Abbeel-Ng (discretization was coarse to limit running time). Figure 2B illustrates the performance of the OptVA algorithm on the infinite horizon metronome problem with finite data. A small number of bases (10) is sufficient to recover the optimal value function quite accurately after basis adaptation. The effects of sample size and iterations of the basis adaptation algorithm are illustrated in Figure 2C.

4. Summary

Here we presented new algorithms for inverse optimal control applicable to LMDPs with discrete and continuous state. They outperform prior IRL algorithms. The new algorithms are solving a restricted class of problems, but this class is broad enough to include or approximate many control problems of interest. It is particularly well suited for modeling the physical systems commonly studied in nonlinear control.

Apart from the benefits arising from the LMDP framework, key to the efficiency of our algorithms is the insight that recovering values is easier than recovering costs because solving the forward problem is avoided. This of course means that we need features over values rather than costs. Cost features are generally easier to design, which may seem like an advantage of prior IRL algorithms. However prior IRL algorithms need to solve the forward problem – therefore they need features over values (or policies, or state-values, depending on what approximation method is used for solving the forward problem) in addition to features

over costs. Thus the feature selection problem in prior IRL work is actually harder.

5. Appendix : Convexity of OptQ

The convexity of $L[q]$ follows from the following **Lemma:** Let $x \in \mathrm{R}^m$ and $M(x) \in \mathrm{R}^{n\times n}$ be such that $M(x)_{ij} = \exp(a_{ij}^T x + b_{ij})$. Suppose that $\sum_j \exp(b_{ij}) < 1 \, \forall i$. Then for any $c, d \in \mathrm{R}_+^n$, the function $f(x) = c^T \log((I - M(x))^{-1} d)$ is convex on the domain $\mathcal{X} = \{x : a_{ij}^T x \leq 0 \quad \forall i, j\}$.

Proof: $M(x)$ is a matrix with positive entries and row sums smaller than 1. Thus, the spectral radius of $M(x)$ is smaller than 1. Hence, we use a series expansion of $(I - M(x))^{-1}$ to get $f(x) = c^T \log\left(\sum_{k=0}^{\infty} M(x)^k d\right)$. For $k \geq 1$, letting $l_0 = i, l_{k+1} = j$, we have $[M(x)^k]_{ij} = \sum_{l_1, l_2, \ldots, l_{k-1}} \prod_{p=0}^{k} [M(x)]_{l_p l_{p+1}}$ Since each entry of $M(x)^k$ is a positive linear combination of terms of the kind $\exp(a^T x + b)$, so is $\sum_k M(x)^k d$ (since $d > 0$). Thus, $\log(\sum_k M(x)^k d)$ is a log-sum-exp function of x and is hence convex. Since $c > 0$, $c^T \log(\sum_k M(x)^k d)$ is a positive linear combination of convex functions and is hence convex.

Acknowledgements

This work was supported by the NSF.

References

Abbeel, P. and Ng, A.Y. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1. ACM, 2004.

Abbeel, Pieter, Coates, Adam, Quigley, Morgan, and Ng, Andrew Y. An application of reinforcement learning to aerobatic helicopter flight. In *In Advances in Neural Information Processing Systems 19*. MIT Press, 2007.

Baker, C.L., Tenenbaum, J.B., and Saxe, R.R. Goal inference as inverse planning. In *Proceedings of the 29th annual meeting of the cognitive science society*, 2007.

Collins, S., Ruina, A., Tedrake, R., and Wisse, M. Efficient bipedal robots based on passive-dynamic walkers. *Science*, 307(5712):1082, 2005.

Deng, H. and Krstic, M. Stochastic nonlinear stabilization-I: A backstepping design. *Systems and Control Letters*, 32(3):143–150, 1997.

Kalman, R. When is a linear control system optimal? *Trans AMSE J Basic Eng, Ser D*, pp. 51–60, 1964.

Kappen, H.J. Linear theory for control of nonlinear stochastic systems. *Physical Review Letters*, 95(20): 200201, 2005.

Körding, K.P. and Wolpert, D.M. The loss function of sensorimotor learning. *Proceedings of the National Academy of Sciences of the United States of America*, 101(26):9839, 2004.

Ng, A.Y. and Russell, S. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 663–670. Morgan Kaufmann Publishers Inc., 2000.

Schmidt, M. minfunc., 2005. http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html.

Srinivasan, M. and Ruina, A. Computer optimization of a minimal biped model discovers walking and running. *Nature*, 439(7072):72–75, 2005.

Syed, U., Bowling, M., and Schapire, R.E. Apprenticeship learning using linear programming. In *Proceedings of the 25th international conference on Machine learning*, pp. 1032–1039. ACM, 2008.

Todorov, E. Optimality principles in sensorimotor control. *Nature Neuroscience*, 7(9):907–915, 2004.

Todorov, E. Linearly-solvable Markov decision problems. *Advances in neural information processing systems*, 19:1369, 2007.

Todorov, E. Eigenfunction approximation methods for linearly-solvable optimal control problems. In *IEEE International Symposium on Adaptive Dynamic Programming and Reinforcemenet Learning*, 2009a.

Todorov, E. Efficient computation of optimal actions. *Proceedings of the National Academy of Sciences*, 106(28):11478, 2009b.

Treuille, A., Lee, Y., and Popović, Z. Near-optimal character animation with continuous control. In *ACM SIGGRAPH 2007 papers*, pp. 7. ACM, 2007.

Yin, Wotao. Gurobi mex: A matlab interface for gurobi, 2009-2010. http://www.caam.rice.edu/~wy1/gurobi_mex.

Ziebart, B.D., Maas, A., Bagnell, J.A., and Dey, A.K. Maximum entropy inverse reinforcement learning. In *Proc. AAAI*, pp. 1433–1438, 2008.