Learning Dynamic Policies from Demonstration

Byron Boots
Computer Science and Engineering
University of Washington
Seattle, WA
bboots@cs.washington.edu

Dieter Fox
Computer Science and Engineering
University of Washington
Seattle, WA
fox@cs.washington.edu

Abstract

We address the problem of learning a policy directly from expert demonstrations. Typically, this problem is solved with a supervised learning method such as regression or classification to learn a *reactive policy*. Unfortunately, reactive policies lack the ability to model long-range dependancies and this omission can result in suboptimal performance. So, we take a different approach. We observe that policies and dynamical systems are mathematical duals, and then use this fact to leverage the rich literature on system identification to learn dynamic policies with state directly from demonstration. Many system identification algorithms have desirable properties like the ability to model long-range dependancies, statistical consistency, and efficient off-the-shelf implementations. We show that by employing system identification algorithms to learning from demonstration problems, all of these properties can be carried over to the learning from demonstration domain. We further show that these properties can be beneficial in practice by applying state-of-the-art system identification algorithms to real-world direct learning from demonstration problems.

1 Introduction

Learning from demonstration (LfD) (Argall et al., 2009), sometimes called programming by demonstration (Billard et al., 2008), and including subareas such as apprenticeship learning (Abbeel & Ng, 2004) and imitation learning (Schaal et al., 2003), is a learning-based method for deriving a policy that specifies how an agent interacts with a dynamical system. By policy we mean a *mapping* that assigns a probability distribution of an agent's action space for each possible state that the agent may encounter.

Traditionally, policies are derived by first completely specifying the state and dynamics of a domain and then optimizing behavior with respect a given cost function. For example, the dynamics may encode the physics of how a robot interacts with its environment, while the cost function is used to direct the robot's behavior. Unfortunately, the process of deriving the dynamics that govern the agent and the system can be laborious, requiring extensive and accurate domain knowledge. Specifying an appropriate cost function can be difficult as well, and optimizing behavior with respect to the cost often leads to an intractable optimization problem. In contrast to this traditional approach, LfD algorithms do not necessarily require access to a model of the domain and do not need a pre-specified cost function. Instead, behavior is *learned* by observing demonstrations provided by an expert.

In this paper we focus on learning a policy *directly* from demonstrations.[1] In direct methods, LfD is typically formulated as a supervised learning problem (Argall et al., 2009). Given a set of train-

[1] In contrast to *indirect* methods which utilize a (partial) model of the system in order to learn a more general policy (including apprenticeship learning (Abbeel & Ng, 2004; Neu & Szepesvári, 2007), maximum margin planning (Ratliff et al., 2006) and inverse optimal control (Ziebart et al., 2008; Ziebart et al., 2010)).

ing examples (demonstrated sequences of actions and observations), LfD attempts to learn a *reactive* policy that maps the previous k observations to an action distribution that mimics the expert's demonstration. Depending on whether the actions are discrete or continuous, LfD may be formulated as a regression or as a classification problem. There have been a number of successful applications of this approach in robotics. For example, ALVINN (Autonomous Land Vehicle in a Neural Network) (Pomerleau, 1988) used artificial neural networks to learn a mapping from features of camera images to car steering angles from human demonstrated behavior. And, Ross et al. (Ross et al., 2012) used linear regression to learn a controller that maps features of images to steering commands for a micro aerial vehicle (MAV).

Despite these successful examples, it is not difficult to see that such reactive policies are limited in their expressivity. In the general case, a policy maintains a posterior distribution of actions a_t conditioned on the entire history of previous observations and actions $h_t = \{o_1, a_1, \dots o_{t-1}, a_{t-1}, o_t\}$ up to time t; e.g. $\mathbb{P}[a_t \mid h_t]$. Reactive policies, based on a truncated length-k history typically miss important long-term dependancies in the data. Dynamic models correct this problem by assuming a *latent state* z_t that captures all of the predictive information contained in h_t. A policy with a latent state does not have to condition on the entire history $\mathbb{P}[a_t \mid h_t]$, but only its state representation $\mathbb{P}[a_t \mid z_t]$. Latent state dynamic models have been used in a wide range of applications including time series forecasting, statistics, economics, robotics, and control.

In this work we consider the problem of learning dynamic policies directly from demonstrations. We *only* assume access to sequences of actions and partial-observations from demonstrated behavior; we do not assume any knowledge of the environment's state or dynamics. The goal is to learn a policy that mimics an expert's actions.

In contrast to previous work, we do not restrict the learning problem to reactive policies, but instead learn dynamic policies with internal state that capture long-range dependancies in the data. In order to learn such policies, we identify LfD with the problem of learning a dynamical model. Specifically, we show that dynamical systems and policies can be viewed as *mathematical duals* with similar latent variable structure; a fact which is greatly beneficial for LfD. The duality means that we can draw on a large number of dynamical system identification algorithms in order to learn dynamic policies. Many of these algorithms have desirable theoretical properties like *unbiasedness* and *statistical consistency*, and have efficient off-the-shelf implementations. We perform several experiments that show that system identification algorithms can provide much greater predictive accuracy compared with regression and classification approaches. To the best of our knowledge, this is the first work to leverage the extensive literature on dynamical system learning for direct LfD.

2 System-Policy Duality

The key insight in this work is that the problem of learning a model of a dynamical system is dual to the problem of learning a dynamic policy from demonstration.

Consider the graphical model depicting the interaction between a policy and dynamical system in Figure 1. A generative dynamical system has a latent state that takes actions as input and generates observations as output at each time step. In such a model the observation distribution at time t is given by $\mathbb{P}[o_t \mid a_1, o_1, \dots, a_{t-1}, o_{t-1}, a_t]$. In this case, we usually assume a latent variable x_t such that $\mathbb{P}[o_t \mid a_1, o_1, \dots, a_{t-1}, o_{t-1}, a_t] = \mathbb{P}[o_t \mid x_t]$. A policy has the same graphical structure with the actions and observations reversing roles: a latent state that takes observations as inputs and generates actions as output at each time step. In such a model the *action* distribution at time t is given by $\mathbb{P}[a_t \mid o_1, a_1, \dots, o_{t-1}, a_{t-1}, o_t]$. The similarity between the system and policy suggests a number of dualities. Simulating observations from a dynamical system is dual to sampling an action from a stochastic policy. Predicting future observations from a dynamical system model is dual to activity forecasting (Kitani et al., 2012) from a policy. Finally, and most importantly in the context of this work, *learning* a policy from demonstration is dual to dynamical system identification. We summarize the dualities here for convenience:

Dynamical System	$\leftrightarrow$	Policy
Simulation	$\leftrightarrow$	Policy Execution
Observation Prediction/Forcasting	$\leftrightarrow$	Activity Forcasting
Dynamical System Identification	$\leftrightarrow$	Learning From Demonstration

To make these dualities more concrete, we consider the problem of linear quadratic Gaussian control.

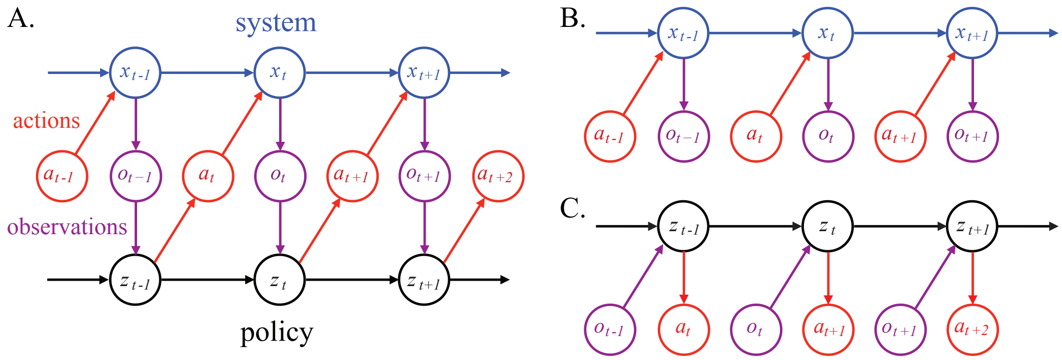

Figure 1: System-Policy Duality. (A) Graphical model depicts the complete interaction between the a policy with a state z_t and a dynamical system with state x_t. Considering the system and the policy independently, we see that they have the same graphical structure (an input-output chain). (B) The system receives actions as inputs and produces observations as outputs. (C) Conversely, the policy receives observations as inputs and produces actions as outputs. The similar graphical structure between the system and policy suggests a number of dualities that can be exploited to design efficient learning algorithms. See text for details.

2.1 Example: Linear Quadratic Gaussian Control

Consider the following simple linear-Gaussian time-invariant dynamical system model, the steady-state Kalman filter with actions a_t, observations o_t and latent state x_t:

$$x_{t+1} = A_s x_t + B_s a_t + K_s e_t \tag{1}$$
$$o_t = C_s x_t \tag{2}$$

Here A_s is the system dynamics matrix, B_s is the system input matrix, C_s is the system output matrix, K_s is the constant Kalman gain, and $e_t = o_t - Cx_t$ is the innovation.

Given a quadratic input cost function R and the quadratic infinite-horizon state-value function V^*, the optimal control law is given by

$$a_t = -(R + B_s^\top V^* B_s)^{-1} B_s^\top V^* A_s x_t \tag{3}$$
$$= -Lx_t \tag{4}$$

where $L = (R+B_s^\top V^* B_s)^{-1} B_s^\top V^* A_s$, found by solving a discrete time Riccati equation, is called the constant *control gain* (Todorov, 2006). In other words, the optimal control a_t is a linear function $-L$ of the system state x_t.

The Policy as a Dynamic Model If we rewrite the original system as

$$\begin{aligned}
x_{t+1} &= A_s x_t + B_s a_t + K_s e_t \\
&= A_s x_t + B_s a_t + K_s(o_t - Cx_t) \\
&= (A_s - K_s C_s) x_t + B_s a_t + K_s o_t \\
&= ((A_s - K_s C_s) - B_s L)\, x_t + K_s o_t + B_s(a_t + Lx_t)
\end{aligned} \tag{5}$$

and then substitute

$$A_c := (A_s - K_s C_s) - B_s L \quad B_c := K_s \quad C_c := -L \quad K_c := B_s \quad v_t := (a_t - C_c x_t) \tag{6}$$

then we see that

$$x_{t+1} = A_c x_t + B_c o_t + K_c v_t \tag{7}$$
$$a_t = C_c x_t \tag{8}$$

In other words, it becomes clear that the policy can be viewed as a dynamic model dual to the original dynamical system in Eqs. 1 and 2. Here the environment chooses an observation and the controller generates an action. We assume the action produced may be affected by stochasticity in the control, so, when we observe the action that was actually generated, we condition on that action.

The Policy as a Subsystem In many cases, the policy may be much simpler than the dynamics of the system. Intuitively, the dynamical system is a full generative model for what may be quite complex observations (e.g. video from a camera (Soatto et al., 2001; Boots, 2009)). However, expert demonstrations may only depend on some *sub-manifold* of the state in the original dynamical system. Examples of this idea include *belief compression* (Roy et al., 2005) and *value-directed compression* of dynamical systems (Poupart & Boutilier, 2003; Boots & Gordon, 2010).

We can formalize this intuition for linear quadratic Gaussians. Information relevant to the controller lies on a *linear subspace* of the dynamical system when the system state can be exactly projected onto a subspace defined by an orthonormal compression operator $V^\top \in \mathbb{R}^{n \times m}$, where $n \leq m$, and still generate exactly the same output. That is *iff*

$$C_c x_t \equiv C_c V V^\top x_t \tag{9}$$

Given the system in Eqs. 7 and 8, this subsystem will have the form:

$$V^\top x_{t+1} = V^\top A_c V V^\top x_t + V^\top B_c o_t + V^\top K_c v_t \tag{10}$$

$$a_t = C_c V V^\top x_t \tag{11}$$

We can now replace

$$z_t := V^\top x_t \qquad \tilde{A}_c := V^\top A_c V \qquad \tilde{B}_c := V^\top B_c \qquad \tilde{C}_c := C_c V \qquad \tilde{K}_c v_t \tag{12}$$

Substituting Eq. 12 into Eqs. 10 and 11, we get can parameterize the dynamic policy in terms of the policy state directly

$$z_{t+1} = \tilde{A}_c z_t + \tilde{B}_c o_t + \tilde{K}_c v_t \tag{13}$$

$$a_t = \tilde{C}_c z_t \tag{14}$$

Direct Policy Learning Learning from demonstration in the LQG domain normally entails executing a sequence of learning algorithms. First, system identification is used to learn the parameters of the system in Eqs. 1 and 2. Next, IOC can be used to approximate the cost and value functions V^* and R. Finally, these value functions together with Eq. 3 find the optimal control law. To execute the controller we would filter to find the current estimate of the state $\hat{x}_t$ and then calculate $u_t = -L\hat{x}_t$.

Unfortunately, error in *any* of these steps may propagate and compound in the subsequent steps which leads to poor solutions. The problem can be mitigated somewhat by first learning the dynamical system model in Eqs. 1 and 2, and then applying linear regression to learn the control law directly. However, in the event that the policy only lies on a subspace of the original system state space, we are wasting statistical effort learning irrelevant dimensions of the full system.

Given the system-controller duality of Eqs. 1 and 2 and Eqs. 7 and 8, we suggest an alternative approach. We *directly* learn the controller in Eqs. 13 and 14 via system identification *without* first learning the dynamical system or performing IOC to learn the cost and value functions. We use precisely the same algorithm that we would use to learn the system in Eqs. 1 and 2, e.g. N4SID (Van Overschee & De Moor, 1996), and simply switch the role of the actions and observations.

We can use a similar strategy to directly learn dynamic policies for other dynamical system models including POMDPS and predictive state representations. Furthermore, unbiasedness and statistical consistency proofs for system identification will continue to hold for the controller, guaranteeing that the predictive error in the controller decreases as the quantity of training data increases. In fact, controller error bounds, which are dependent on the dimensionality of the controller state, are likely to be tighter for many polcies.

3 Learning From Demonstration as System Identification

Given the dualities between dynamical systems and policies, a large set of tools and theoretical frameworks developed for dynamical system learning can be leveraged for LfD. We list some of these tools and frameworks here:

- Expectation Maximization for learning linear Gaussian systems and Input-Output HMMs (Bengio & Frasconi, 1995; Ghahramani, 1998; Roweis & Ghahramani, 1999)
- Spectral learning (subspace identification) for learning linear Gaussian systems, Input-Output HMMs, and Predictive State Representations (Van Overschee & De Moor, 1996; Katayama, 2005; Hsu et al., 2009; Boots et al., 2010; Boots & Gordon, 2011)
- Nonparametric Kernel Methods for learning very general models of dynamical systems (Gaussian processes (Ko & Fox, 2009), Hilbert space embeddings (Song et al., 2010; Boots & Gordon, 2012; Boots et al., 2013))

From a theoretical standpoint, some of these algorithms are better than others. In particular, several spectral and kernel methods for system identification have nice theoretical properties like statistical consistency and finite sample bounds. For example, subspace identification for linear Gaussian systems is numerically stable and statistically consistent (Van Overschee & De Moor, 1996). Boots et al. (Boots et al., 2010; Boots & Gordon, 2011) has developed several statistically consistent learning algorithms for predictive state representations and POMDPs. And Boots et al. have developed statistically consistent algorithms for nonparametric dynamical system models based on Hilbert space embeddings of predictive distributions (Song et al., 2010; Boots et al., 2013).

Bias & DAgger Unfortunately, direct LfD methods, including the policy learning methods proposed here, do have a drawback. Samples of demonstrations used to learn policies can be biased by the expert's own behavior. This problem has been recognized in previous direct policy learning applications. For example, both the autonomous car policy parameterized by ALVINN (Pomerleau, 1988) and reactive policy for the micro aerial vehicle in Ross et al. are imperfect and occasionally move the vehicles closer to obstacles than in any of the training demonstrations (Ross et al., 2012). Because the vehicle enters into a state not observed in the training data, the learned policy does not have the ability to properly recover.

Ross et al. has suggested a simple iterative training procedure called data aggregation (DAgger) for this scenario (Ross et al., 2011). DAgger initially trains a policy that mimics the expert's behavior on a training dataset. The policy is then executed, and, while the policy executes, the expert demonstrates correct actions in newly encountered states. Once the new data is collected, the policy is relearned to incorporate this new information. This allows the learner to learn the proper recovery behavior in new situations. The process is repeated until the policy demonstrates good behavior. After a sufficient number of iterations, DAgger is guaranteed to find a policy that mimics the expert at least as well as a policy learned on the aggregate dataset of all training examples (Ross et al., 2011).

Although DAgger was originally developed for reactive policy learning, DAgger can also be applied to system identification (Ross & Bagnell, 2012), and, therefore to any system identification method that we use to learn a dynamic policy.

4 Empirical Evaluations

We compared a number of different approaches to direct policy learning in two different experimental domains. In the first experiment we looked at the problem of learning a policy for controlling a slotcar based on noisy readings from an inertial measurement unit. In the second experiment we looked at the problem of learning a policy that describes the navigation behavior of the Manduca Sexta, a type of hawkmoth.

4.1 Slotcar Control

We investigate the problem of learning a slotcar control policy from demonstration. The demonstrated trajectories consisted of continuous actions corresponding to the velocity of the car and continuous measurements from an inertial measurement unit (IMU) on the car's hood, collected as the car raced around the track. Figure 2(A) shows the car and attached 6-axis IMU (an Intel Inertiadot), as well as the 14m track. An optimal policy was demonstrated by a robotic system using an overhead camera to track the position of the car on the track and then using the estimated position to modulate the speed of the car. Our goal was to learn a policy similar to the optimal policy, but based only on information from previous actions and IMU observations.

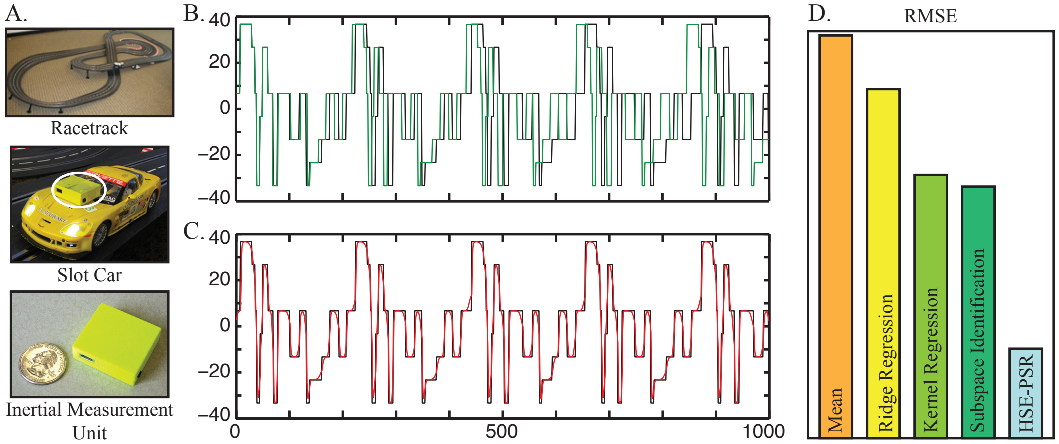

Figure 2: The slotcar experiment. (A) The platform: the experimental setup consisted of a 14m racetrack and a slotcar with an IMU attached to the hood. (B) Even though the track is topologically circular and the policy is repetitive, the car experiences significant imprecision in its movement, especially when traveling at high speeds. This means that simple replay cannot be used to control the car. The black line indicates the current control signal to the car. The green line indicates a replayed control signal. The error in car movement causes the two signals to become increasingly decoupled over time. (C) The predicted action from the best learned policy (the HSE-PSR model). The red line indicates the learned policy and the black line indicates the true policy. (D) Comparison of several different direct LFD approaches on the slotcar dataset. Mean is a baseline consisting of the average action. Ridge regression and kernel regression were used to learn reactive policies. Subspace identification and the nonparametric HSE-PSR method produce dynamic policies.

We collected the estimated angular velocity of the car (observations) from the IMU as well as the control input (a number between 0 and 255) at 10Hz for 5000 steps. The first 2500 data points were used for training and the second 2500 data points were held out for testing. We compared several different methods for learning a policy. First, we learned two reactive policies that map the preceding action and observation to the next action: a linear policy via ridge regression and a nonlinear policy via kernel regression. We then learned two dynamic policies using system identification. We learned a linear Gaussian system by subspace identification (Van Overschee & De Moor, 1996) and a nonparametric dynamical system via Hilbert space embeddings of predictive state representations (HSE-PSRs) (Boots et al., 2013).

The four methods were compared on the test dataset by evaluating the expected action from the learned policies to the actual action executed by the optimal policy. As mentioned above, the reactive methods predicted the next action based solely on the previous action and observation. The two dynamic policies, however, had a latent state and were updated at each time step by filtering the previous action and observation. A comparison of the four approaches is given in Figure 2(D). As expected, the two policies learned by system identification are able to produce better predictions than their reactive counterparts due the ability to model long-range dependancies in the data. Additionally, we note that the nonparametric HSE-PSR policy performed much better than the linear policy learned by subspace identification.

4.2 Hawkmoth Behavior Modeling

In the second experiment, we investigate the problem of learning a control policy that mimics the behavior of the hawkmoth Manduca sexta. M. sexta is large moth that is active in low-light environments, making it especially well suited to laboratory vision experiments. M. sexta is also amenable to tethering and has been extensively used in previous tethered flight experiments (e.g. (Gra, 2002), (Dyhr et al., 2012)). In our navigation experiments, we used a virtual flight arena similar to that described in (Sponberg & Daniel, 2012), in which a M. sexta hawkmoth is attached by means of a rigid tether to a single-axis optical torquemeter (Figure 3). The torquemeter measures the left-right turning torque exerted by the flying moth in response to a visual stimulus projected

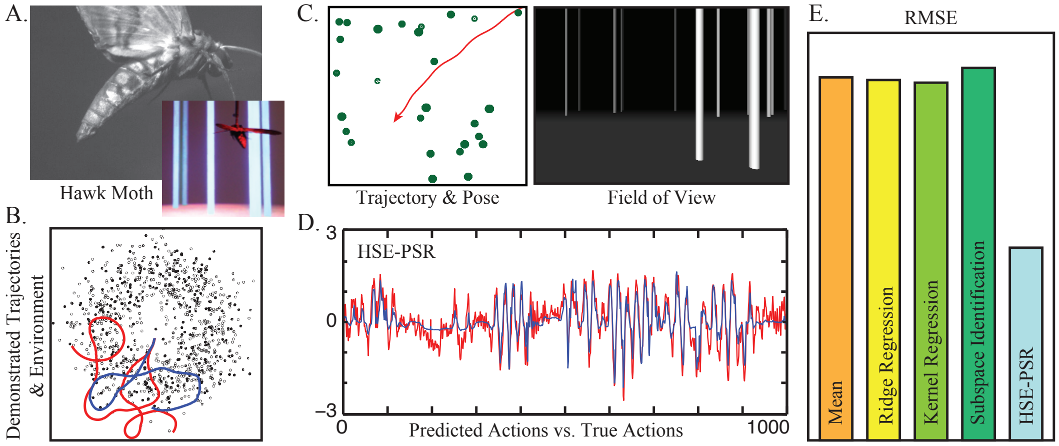

Figure 3: Hawkmoth experiment. (A) The hawkmoth suspended in the simulator. (B) Overhead view of the paths taken by the hawkmoth through the simulated forest. The two red paths were used as training data. The blue path was holdout as test data. (C) The position of the moth in the environment and the corresponding moth field of view. (D) The predicted action from the best learned policy (the HSE-PSR model). The blue line indicates the action predicted by the learned policy and the red line indicates the true actions taken by the moth. (E) Comparison of several different direct LFD approaches on the hawkmoth dataset. The nonparametric HSE-PSR model is the only approach that is able to learn a reasonably good policy.

onto a curved screen placed in front of it. The left-right turning torque was used to control the visual stimulus in closed-loop.

Each visual stimulus consisted of a position-dependent rendered view of a virtual 3-D world meant to look abstractly like a forest. In the simulated world, the moths began each trial in the center of a ring-shaped field of cylindrical obstacles. For every trial, the moth flew for at least 30s at a fixed flight speed and visibility range. The rendered images presented to the moth and the resulting actions (the rotation of the moth, computed from the torque) were recorded. Our goal was to learn a policy that mimicked the moth behavior, taking images as input and producing rotation as output.

We collected 3 trials of M. sexta flight (Figure 3(B)). The first two trials were used for training and the last trial was held out for testing. As in the slot car experiment, we evaluated four different methods for learning a policy: ridge regression, kernel regression, subspace identification, and HSE-PSRs. The four methods were compared on the test dataset by evaluating the expected action from the learned policies to the actual action executed by the optimal policy. A comparison of the four approaches is given in Figure 3(E). The M. sexta policy, which is nonlinear, non-Gaussian, and has an internal state, was much harder to model than the slotcar control policy. In this case, the nonparametric HSE-PSR policy performed much better than any of the competing approaches.

References

(2002). A method for recording behavior and multineuronal {CNS} activity from tethered insects flying in virtual space. *Journal of Neuroscience Methods*, *120*, 211 – 223.

Abbeel, P., & Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. *ICML*.

Argall, B. D., Chernova, S., Veloso, M., & Browning, B. (2009). A survey of robot learning from demonstration. *Robot. Auton. Syst.*, *57*, 469–483.

Bengio, Y., & Frasconi, P. (1995). An Input Output HMM Architecture. *Advances in Neural Information Processing Systems*.

Billard, A., Calinon, S., Dillmann, R., & Schaal, S. (2008). Survey: Robot programming by demonstration. Handbook of Robotics, . chapter 59, 2008.

Boots, B. (2009). *Learning Stable Linear Dynamical Systems*Data Analysis Project). Carnegie Mellon University.

Boots, B., & Gordon, G. (2010). Predictive state temporal difference learning. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel and A. Culotta (Eds.), *Advances in neural information processing systems 23*, 271–279.

Boots, B., & Gordon, G. (2011). An online spectral learning algorithm for partially observable nonlinear dynamical systems. *Proc. of the 25th National Conference on Artificial Intelligence (AAAI-2011)*.

Boots, B., & Gordon, G. (2012). Two-manifold problems with applications to nonlinear system identification. *Proc. 29th Intl. Conf. on Machine Learning (ICML)*.

Boots, B., Gretton, A., & Gordon, G. J. (2013). Hilbert Space Embeddings of Predictive State Representations. *Proc. UAI*.

Boots, B., Siddiqi, S. M., & Gordon, G. J. (2010). Closing the learning-planning loop with predictive state representations. *Proceedings of Robotics: Science and Systems VI*.

Dyhr, J. P., Cowan, N. J., Colmenares, D. J., Morgansen, K. A., & Daniel, T. L. (2012). Autostabilizing airframe articulation: Animal inspired air vehicle control. *CDC* (pp. 3715–3720). IEEE.

Ghahramani, Z. (1998). Learning dynamic Bayesian networks. *Lecture Notes in Computer Science*, *1387*, 168–197.

Hsu, D., Kakade, S., & Zhang, T. (2009). A spectral algorithm for learning hidden Markov models. *COLT*.

Katayama, T. (2005). *Subspace methods for system identification: A realization approach*. Springer.

Kitani, K., Ziebart, B. D., Bagnell, J. A. D., & Hebert, M. (2012). Activity forecasting. *European Conference on Computer Vision*. Springer.

Ko, J., & Fox, D. (2009). Learning GP-BayesFilters via Gaussian process latent variable models. *Proc. RSS*.

Neu, G., & Szepesvári, C. (2007). Apprenticeship learning using inverse reinforcement learning and gradient methods. *UAI* (pp. 295–302).

Pomerleau, D. (1988). Alvinn: An autonomous land vehicle in a neural network. *NIPS* (pp. 305–313).

Poupart, P., & Boutilier, C. (2003). Value-directed compression of POMDPs. *Proc. NIPS*.

Ratliff, N. D., Bagnell, J. A., & Zinkevich, M. (2006). Maximum margin planning. *ICML* (pp. 729–736).

Ross, S., & Bagnell, D. (2012). Agnostic system identification for model-based reinforcement learning. *ICML*.

Ross, S., Gordon, G. J., & Bagnell, D. (2011). A reduction of imitation learning and structured prediction to no-regret online learning. *Journal of Machine Learning Research - Proceedings Track*, *15*, 627–635.

Ross, S., Melik-Barkhudarov, N., Shankar, K. S., Wendel, A., Dey, D., Bagnell, J. A., & Hebert, M. (2012). Learning monocular reactive uav control in cluttered natural environments. *CoRR*, *abs/1211.1690*.

Roweis, S., & Ghahramani, Z. (1999). A unified view of linear Gaussian models. *Neural Computation*, *11*, 305–345.

Roy, N., Gordon, G., & Thrun, S. (2005). Finding Approximate POMDP Solutions Through Belief Compression. *JAIR*, *23*, 1–40.

Schaal, S., Ijspeert, A., & Billard, A. (2003). computational approaches to motor learning by imitation. 537–547.

Soatto, S., Doretto, G., & Wu, Y. (2001). Dynamic Textures. *Intl. Conf. on Computer Vision*.

Song, L., Boots, B., Siddiqi, S. M., Gordon, G. J., & Smola, A. J. (2010). Hilbert space embeddings of hidden Markov models. *Proc. 27th Intl. Conf. on Machine Learning (ICML)*.

Sponberg, S., & Daniel, T. L. (2012). Abdicating power for control: a precision timing strategy to modulate function of flight power muscles. *Proceedings of the Royal Society B: Biological Sciences*.

Todorov, E. (2006). Optimal control theory.

Van Overschee, P., & De Moor, B. (1996). *Subspace identification for linear systems: Theory, implementation, applications*. Kluwer.

Ziebart, B. D., Bagnell, J. A., & Dey, A. K. (2010). Modeling interaction via the principle of maximum causal entropy. *ICML* (pp. 1255–1262).

Ziebart, B. D., Maas, A. L., Bagnell, J. A., & Dey, A. K. (2008). Maximum entropy inverse reinforcement learning. *AAAI* (pp. 1433–1438).