Improved Learning of Dynamics Models for Control

Arun Venkatraman[1], Roberto Capobianco[2], Lerrel Pinto[1], Martial Hebert[1], Daniele Nardi[2], and J. Andrew Bagnell[1]

[1]The Robotics Institute, Carnegie Mellon University, [2]Sapienza University of Rome
{arunvenk,lerrelp,hebert,dbagnell}@cs.cmu.edu
{capobianco,nardi}@dis.uniroma1.it

Abstract. Model-based reinforcement learning (MBRL) plays an important role in developing control strategies for robotic systems. However, when dealing with complex platforms, it is difficult to model systems dynamics with analytic models. While data-driven tools offer an alternative to tackle this problem, collecting data on physical systems is non-trivial. Hence, smart solutions are required to effectively learn dynamics models with small amount of examples. In this paper we present an extension to Data As Demonstrator for handling controlled dynamics in order to improve the multiple-step prediction capabilities of the learned dynamics models. Results show the efficacy of our algorithm in developing LQR, iLQR, and open-loop trajectory-based control strategies on simulated benchmarks as well as physical robot platforms.

Keywords: Reinforcement Learning, Optimal Control, Dynamics Learning, Sequential Prediction

1 Motivation

Learning based approaches for controlling robotic systems, or more generally autonomous agents, fall into two primary categories: model-based [1–3] and model-free [4–7]. In this work, we focus on problems belonging to the former category, where a system transition function – a dynamics model – is used to guide the creation of a control policy. To generate low cost control policies, we need dynamics models that accurately capture the evolution of the true underlying system. However, with the increasing complexity of robotic technologies, it becomes difficult to robustly characterize robot dynamics *a priori* with simple analytic models. To tackle this problem and to scale model-based control techniques to new systems, prior work in dynamics learning has shown promising results in the modeling of dynamics system either by augmenting physics-based models [8] or through non-parametric, black-box learning [9].

Typically, the accuracy of data-driven dynamics models depends on the amount of collected data. However, for many robotic systems, it can be labor intensive and expensive to acquire large data-sets for training models. Hence, it

is often desirable to improve model fidelity by observing fewer example trajectories on the physical system. We propose a model-based reinforcement learning (control) framework that reuses collected data to improve the learned dynamical system models. We leverage Data As Demonstrator (DaD) [10] to generate synthetic training examples for improving the learned dynamics model's multi-step prediction capabilities. However, the original DaD algorithm only handles uncontrolled dynamics. Here, we extend DaD to also work with controlled systems. Our experimental results with this mehtod show it is possible to achieve good control performance with less data.

2 Technical Approach

2.1 Preliminaries

In this work, we consider systems that operate as a Markov Decision Process (MDP). The MDP is defined by states x_t that follow an *unknown* state transition (dynamics) function $f(x_t, u_t) \rightarrow x_{t+1}$, where u_t are controls (actions). We additionally assume a known cost function $c : x_t, u_t \rightarrow \mathbb{R}$. Solving this MDP consists of minimizing the (expected) cumulative cost over a time horizon T, which may be infinite, by finding a control policy $\pi(x_t)$:

$$\pi = \arg\min_{\pi} \sum_{t=0}^{T-1} c(x_t, u_t) \quad \text{s.t. } u_t = \pi(x_t) \text{ and } x_{t+1} = f(x_t, u_t). \tag{1}$$

Model-based reinforcement learning (MBRL) attempts to solve the above in situations where the underlying dynamics and sometimes the cost function are unknown, adding the burden of deriving estimators of both. In this work, we assume knowledge of the cost function and focus solely on system identification in which we fit a function approximator $\hat{f}$ to be used as the dynamics constraint for the policy optimization in Eq. 1. System identification has been studied in the traditional controls literature [11] and in the machine learning community [12, 13]. Some of these approaches [13, 14] can provide performance guarantees in the infinite-data and model-realizable case (i.e. the true underlying model is linear), while others [12, 8] optimize the the single-step predictive criterion

$$\hat{f} = \arg\min \sum_{t=1}^{T-1} \|x_t - \hat{f}(x_{t-1}, u_{t-1})\|_2^2 \tag{2}$$

from a data-set of trajectories $\{(x_0, u_0) \ldots, (x_{T-1}, u_{T-1})\}$ of state-action pairs collected from the system. The downside of optimizing Eq. 2 is that errors can compound (up to exponentially [15, 10]) when using $\hat{f}$ to forward predict multiple time-steps into the future. To solve this problem, we propose an extension to the algorithm presented in [10] for learning multi-step predictive models for system identification in the controlled setting and we experimentally verify that this improves the efficiency of MBRL methods.

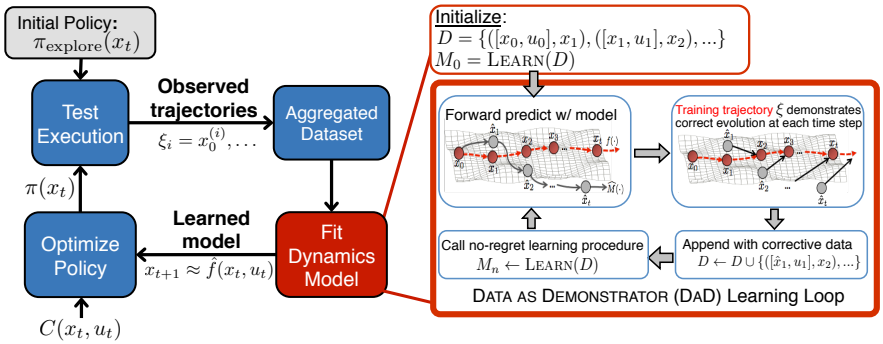


Fig. 1. (Left) DAgger System Identification for Model Learning and Control. (Right) DAD improves the multi-step prediction performance of learned dynamics models.

2.2 System Identification for Control

Simply collecting system trajectories, learning the dynamics, and optimizing the control policy typically results in inaccurate or unstable dynamics models and poorly performing control policies. An iterative process to achieve better performance was formalized in the MBRL literature [16, 17], generating a procedure similar to the one outlined in Fig. 1 (left). By alternating between fitting the dynamics model and collecting new data under the distribution induced by the policy, the model becomes better at capturing the dynamics over the important regions of the state-space while the control policy derived from the dynamics is either improved over that region or erroneously exploits inaccuracies in the dynamics model. Thus in each iteration, a good policy is found or data is collected from the controller's mistakes for improvement at the next iteration.

We specifically refer to the left loop in Fig. 1 as the DAgger (Data-set Aggregation) system identification learning framework [18]. A key difference lies in the aggregation step of the procedure in order to provide model agnostic guarantees. At the beginning of the algorithm, DAgger initializes an empty training data-set and an exploration policy $\pi_{\text{explore}}(x_t)$ that generates an action (control) u_t given a state x_t. This initial policy can either consist of random controls (referred to as a random policy) or be an expert demonstration. Then, DAgger iteratively proceeds by: (1) executing the latest policy to collect a set of new trajectories $\{\xi_i\}_{i=0}^{k-1}$ where $\xi_i = \{(x_t, u_t)\ldots\}_i$ is a time series of state-action pairs; (2) aggregating the trajectories $\{\xi_i\}_{i=0}^{k-1}$ into the training data-set; (3) learning from the data-set a forward dynamics model $\hat{f}(x_t, u_t) \rightarrow x_{t+1}$; (4) optimizing a new control policy π that minimizes a given cost function $c(x_t, u_t)$ over the time horizon T of the control problem; (5) tracking the best policy from all those generated.

During the execution of the first DAgger loop, the state distribution induced by π can greatly differ from the initial π_{explore}; the first generated policies may perform poorly due to inaccuracies in $\hat{f}$. The iterative procedure refines the dynamics model by aggregating data from states induced by running the system with $\pi_1, \ldots, \pi_N$. In particular, Ross et al. [18] provide theoretical guarantees for this algorithm, as long as we also sample states from the exploration distribution

when aggregating data. This can be simply obtained by aggregating additionally a constant percentage of trajectories obtained from the exploration distribution. For example, this can be obtained by sampling from the original dataset or running the system with π_{explore}. This helps prevent the learner from focusing only on the induced distributions from the policies. Finally, the algorithm does not guarantee that the policy gets monotonically better with every iteration. Thus, we must track the performance of the executed policies and return the best one obtained so far.

2.3 Improving the multi-step predictive performance

Despite the use of iterative procedures, MBRL methods can suffer from compounding errors during the policy optimization phase, either during the forward planning with the model or in the back-propagation of the model gradients [19]. The cascading error is due to sequential predictions performed with the learned model. By performing sequential predictions, the model is recursively applied and its previous output is fed as its new input

$$\hat{x}_{t+1} = \hat{f}(\hat{x}_t, u_t). \tag{3}$$

This can result in a significant deviation from the true system. Ideally, we would address this by learning a dynamics model that is optimized for the multiple-step predictive performance (e.g. lagged-error criterion [20])

$$\hat{f} = \arg\min \sum_{t=1}^{T-1} \sum_{j=t}^{T-1} \|x_j - \hat{x}_{j|t}\|_2^2, \tag{4}$$

where $\hat{x}_{j|t}$ is computed by applying Eq. 3 for $j - t$ times to get the roll-out prediction $\hat{x}_{j|t}$ starting from x_t. However, Eq. 4 is difficult to optimize. It is non-convex in the model parameters of our learner f and also differs from standard supervised learning loss functions. For these reasons, much of the dynamics learning literature focuses on optimization of the single-step loss (Eq. 2) utilizing existing supervised learning procedures such as Gaussian Process [8], Kernel Regression [9], and Support-Vector regression [21].

In order to achieve good multi-step predictive performance while using supervised learning methods, we recently introduced Data As Demonstrator (DaD) [10]. DaD is a meta-algorithm that augments the traditional dynamics learning method with an additional iterative procedure. The canonical dynamics learning method uses a learning procedure to find a model $\hat{f}_0$ that minimizes the single-step predictive performance. Conversely, to minimize the cascading error, DaD specifically targets the distribution induced from sequential application of the model. To this end, the algorithm performs "roll-outs" with the learned model (in gray, top-left of the DaD-loop, Fig. 1) along trajectories from the training data (shown in red, top-right of the DaD-loop, Fig. 1). Then, DaD generates synthetic data by creating new input-target pairs and pointing each prediction to the correct time-indexed state along the training trajectory[1].

[1] Trajectories can be sub-sampled shorter than the control problem's time horizon.

Algorithm 1 DATA AS DEMONSTRATOR (DAD) for Control

Input:
▷ Number of iterations N, set $\{\xi_k\}$ of K trajectories of time lengths $\{T_k\}$.
▷ Learning procedure LEARN

Output: Model $\widehat{f}$

1: Initialize data-set $D \leftarrow \{([x_t, u_t], x_{t+1})\}$ of $(T_k - 1)$ input-target pairs from each trajectory ξ_k
2: **for** $n = 1, \ldots, N$ **do**
3: $\quad \hat{f}_n \leftarrow \text{LEARN}(D)$
4: $\quad$ **for** $k = 1, \ldots, K$ **do**
5: $\quad\quad$ Extract $x_0 \leftarrow \xi_k(0)$, $\{u_t\}_{t=0}^{T_k} \leftarrow \xi_k$
6: $\quad\quad (\widehat{x}_1, \ldots, \widehat{x}_T) \leftarrow \text{ROLL-OUT}(\hat{f}_n, x_0, \{u_t\}_{t=0}^{T_k})$
7: $\quad\quad D' \leftarrow \{([\widehat{x}_1, u_1], x_2), \ldots, ([\widehat{x}_{T_k-1}, u_{T_k-1}], x_{T_k})\}$ where $x_t \leftarrow \xi_k(t)$
8: $\quad\quad D \leftarrow D \cup D'$
9: $\quad$ **end for**
10: **end for**
11: **return** $\hat{f}_n$ with lowest error on validation trajectories

While we refer the reader to [10] for theoretical details, DAD (as presented in [10]) only handles uncontrolled dynamics. Here we introduce an extension to this algorithm to enable it to handle controlled systems to be used in the MBRL setting, as shown on the right side of Fig. 1. As detailed in Alg. 1, we learn a forward dynamics by optimizing a supervised learning loss to predict targets x_{t+1} from "features" $[x_t, u_t]$. Also in this case, we rely on a data aggregation procedure on the training data-set. When executing the roll-out of the model (line 6, Alg. 1), we start at the state x_0 taken from the first timestep of the trajectory ξ_k and forward simulate by performing recursive updates (Eq. 3) with the learned model $\hat{f}_n$ and the true sequence of controls $\{u_t\}$ from ξ_k. Then, when augmenting the data-set, we utilize the original control and the estimated state to create an input-target pair $([\hat{x}_t, u_t], x_{t+1})$, (line 7, Alg. 1). Here, differently from the procedure in [15], during the DAD step we do not separate the state transition dynamics from the controls but do a joint optimization of the model.

Intuitively, our algorithm (detailed in Algorithm 1) aims to give the learner synthetic recovery examples to compensate for the compounding error. In practice, as we want to upper-bound the loss for the learner (i.e. make the learning problem easier for finding $\hat{f}_{n+1}$), we discard examples during the aggregation step if the error was too high (significantly higher than the trajectory's magnitude). In the next section, we experimentally verify that this extension allows us to find better control policies with less data than the traditional approach.

3 Experimental Evaluation

We evaluate our algorithm ('DAgger +DAD') both on simulated dynamical systems[2] and real robotic platforms. In particular, we consider two simulated sce-

[2] Simulators, except the helicopter, available at `https://github.com/webrot9/control_simulators` with C++ and Python APIs

narios: the classic cartpole swing-up problem and the challenging helicopter hovering problem. Additionally, we show the applicability of our approach on real systems such as the Videre Erratic mobile base and the Baxter robot. In each described experiment, we learn dynamical models of the form:

$$\Delta_t \leftarrow f(x_t, u_t), \quad \text{where } \Delta_t = x_{t+1} - x_t. \tag{5}$$

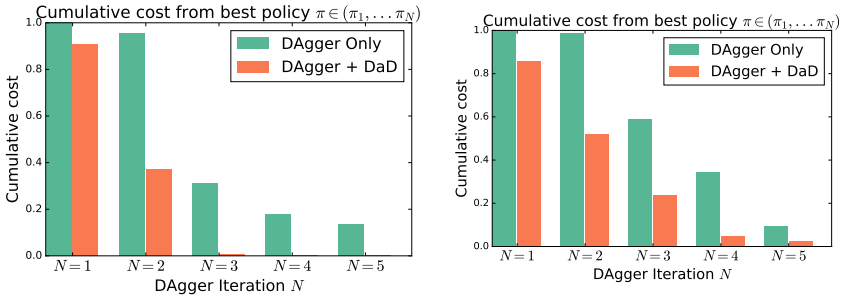
This parametrization is similar to [17], where the previous state is used as the mean prior for predicting the next state. Due to the difficulty of optimizing Eq. 1 under arbitrary dynamics and cost models, for simplicity, we focus on minimizing a sum-of-quadratics cost-to-go function:

$$\sum_t c(x_t, u_t) = \sum_t x_t^T Q_t x_t + u_t^T R_t u_t. \tag{6}$$

By using this form of cost function, along with a linearization of the learned dynamics model, we can formulate the policy synthesis problem as that of a Linear Quadratic Regulator, which allows the policy to be computed in closed-form. In each experiment, we compare 'DAD +DAgger' to 'DAgger Only'. For Cartpole, Erratic, and Baxter the data-set was initialized with a random exploratory policy, while the helicopter problem received both a random and an expert policy (generated form LQR on the true dynamics) roll-out for initialization. The simulated cartpole and helicopter experiments got additional exploratory roll-outs on every iteration of DAgger with the random and expert policies respectively. For the Baxter robot experiment, instead, we achieved exploration through an ϵ-random controller that added random perturbation to the commanded control with ϵ probability. For each method, we report the average cumulative cost at each iteration of DAgger as averaged over ran trials. Three trials were run on the Erratic while five were ran for other benchmarks. The charts that illustrate the obtained results are all normalized to the highest observed cost, since the cost functions are tuned to promote the desired control behavior rather than to have a physical interpretation.

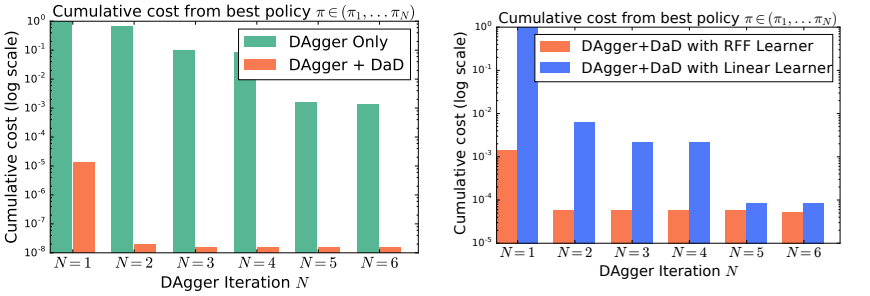
3.1 Simulation Experiments

Cartpole swing-up: The cartpole swing-up is a classic controls and MBRL benchmark where the goal is to swing-up a pendulum by only applying a linear force on the translatable base. We learn a linear dynamics model in the form of Eq. 5 using Ridge Regression (regularized linear regression). We then use an iterative Linear Quadratic Regulator [22] (iLQR) controller about a nominal swing-up trajectory in state-space with an initial control trajectory of zeros. The iLQR optimization procedure finds a sequence of states and controls feasible under the learned dynamics model to minimize the cost. The simulated system has system-transition noise and we compare our algorithim's performance both with and without control noise to simulate the effects of noisy actuation on a real-robot. We show results in Fig. 2 of the evaluated trajectory costs accumulated over the problem's time horizon.



(a) Cartpole Swing-up (b) Swing-up with Control Noise

Fig. 2. Controlling a simulated cartpole for swing-up behavior.



(a) Helicopter Hover (b) Using Different Learning Algorithms

Fig. 3. Controlling a simulated helicopter to hover. Note the log-scale on cost.

Helicopter simulator: Helicopter hovering is a difficult problem due to the instability of the dynamical system, especially under noise. We utilize the helicopter simulator from [16] with additive white noise and follow a problem setup similar to [18]. We make the problem more difficult by initializing the helicopter at states up to 10 meters away from the nominal hover configuration. As the dynamics are highly nonlinear, we show the advantage of using Random Fourier Features (RFF) regression [23] to learn a dynamics model in a 21-dimensional state space. We find a steady-state linear quadratic regulator (LQR) policy to map the helicopter's state to the 4-D control input. The results in Fig. 3 show that DAD dramatically improves performance over only DAgger.

3.2 Real-Robot Experiments

Videre Erratic: In this experiment, we control the velocity of a Videre Erratic mobile base. The goal is to drive the robot to a given position specified in the robot's reference frame. The 3-D state vector includes the robot position and orientation while the 2-D control vector is the robot velocity. The dynamics model is learned using Ridge Regression. Unlike the other experiments, we use a trajectory-control policy that finds a sequence of controls $u_1, \ldots, u_T$ to

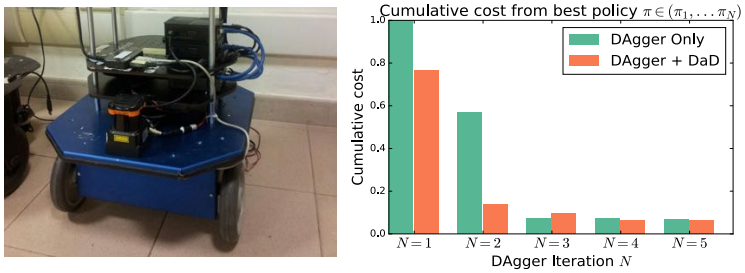


Fig. 4. Results for controlling a Videre Erratic differential-drive mobile robot.

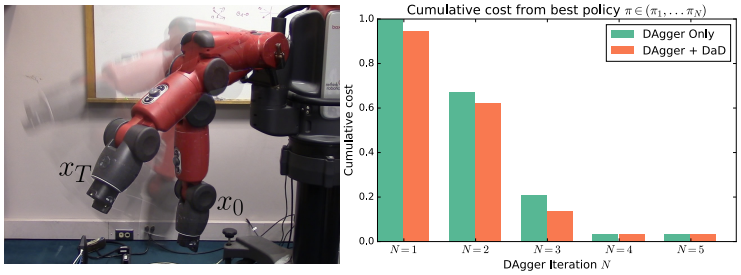


Fig. 5. Results on controlling a Baxter robot. We learn a dynamics model and compute a control policy to move the robot manipulator from state x_0 to x_T.

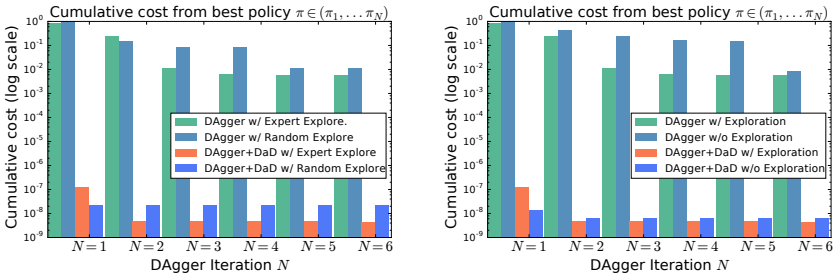
apply open-loop at run time on the robot. We compute the control sequence by simulating the learned dynamics model $\hat{f}$ with a simple proportional controller. Results are shown in Fig. 4.

Baxter robot: We use the 'DAgger +DaD' approach to control a 7-degree-of-freedom manipulator to a target joint configuration. We command the robot arm in torque control mode with *suppression* of the inbuilt gravity compensation. The 14-dimensional state vector consists of the joint angles and their velocities. We learn the dynamics model using Ridge Regression and compute a steady-state LQR control policy, obtaining the results in Fig. 5.

4 Discussion

In our simulation experiments we compared the performance obtained by applying 'DAgger +DaD' on a cartpole with and without control noise. Results show that the improvement of our method over 'DAgger Only' decreases in presence of actuation noise. This can be explained by the fact that, over the same generated nominal controls, the state trajectories obtained during each roll-out are slightly different and represent a limitation on the efficacy of the learner over the same number of iterations – i.e. there is a higher baseline error in the dynamics model.

In the case of the helicopter, we additionally compared the results obtained by using two different learning algorithms and by applying different exploration policies. For the former, we compared the non-linear RFF [23] regression against linear regression. As shown in Fig. 3(b), the nonlinear learner performs much



(a) Random vs. Expert Explore Policy (b) Explore vs. No Exploration Policy

Fig. 6. Comparison of Exploration policies. Cost values are not normalized across plots.

better as it better captures the heavy nonlinearity of the helicopter dynamics. The DAgger method [18] requires drawing state-transition samples at every iteration from some exploration distribution. In Fig. 6(a), we compare using an expert exploration policy (LQR controller using the true dynamics) versus a random-control exploration policy. With DAgger +DaD, the learned dynamics and policy yield a stable behavior for both types of exploration, with some improvement using the expert policy. The DAgger Only baseline often is unable to learn a stable policy using the random exploration policy. We believe that DAgger +DaD learns a more stable multi-step predictive dynamics model – an important aspect for the Bellman backup during policy optimization. An interesting observation is that DAgger +DaD without the exploration policy does not lead to a significant performance difference (Fig. 6(b)) compared to the 'DAgger Only' baselines. This comparison shows the difference between [20] (no exploration) and [18] (constant fraction exploration). Note that to keep the amount of data constant in the trials without the exploration trajectories, the learners were given the difference as test trials under the current optimized policy.

The real-robot evaluations show the applicability of our method on real systems and complex platforms. In particular, the Erratic experiments show that by using DaD, we are indeed able to get a better dynamics model for forward-prediction. This model can be used for trajectory generation and optimization as described in Section 3.2, where the sequence of obtained controls has been directly applied to the Erratic in an open-loop as a control trajectory. While the application of 'DAgger +DaD' on the Baxter robot results in a limited performance improvement, this confirms our hypothesis that, in robotic platforms characterized by high actuation noise (e.g. Baxter's chain of noisy actuators), only smaller improvements over 'DAgger Only' can be achieved (consistent with the simulated noisy-actuation result in Fig. 2(b)). Additionally, the considered problem on the Baxter is relatively simple with control authority at every joint. In these settings, DAgger seemingly can still efficiently capture the dynamics of the system with only a minor benefit from the additional DaD loop.

Acknowledgements This material is based upon work supported in part by: National Science Foundation Graduate Research Fellowship Grant No. DGE-1252522, National

Science Foundation NRI Purposeful Prediction Award No. 1227234, and ONR contract N000141512365. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

1. Schaal, S., et al.: Learning from demonstration. NIPS (1997) 1040–1046
2. Bakker, B., Zhumatiy, V., Gruener, G., Schmidhuber, J.: Quasi-online reinforcement learning for robots. ICRA (2006) 2997–3002
3. Hester, T., Quinlan, M., Stone, P.: Rtmba: A real-time model-based reinforcement learning architecture for robot control. ICRA (2012) 85–90
4. Thrun, S.: An approach to learning mobile robot navigation. RAS **15**(4) (1995)
5. Matarić, M.J.: Reinforcement learning in the multi-robot domain. In: Robot colonies. Springer (1997) 73–83
6. Duan, Y., Liu, Q., Xu, X.: Application of reinforcement learning in robot soccer. Engineering Applications of Artificial Intelligence **20**(7) (2007) 936–950
7. Konidaris, G., Kuindersma, S., Grupen, R., Barto, A.: Robot learning from demonstration by constructing skill trees. IJRR (2011) 0278364911428653
8. Ko, J., Klein, D.J., Fox, D., Haehnel, D.: GP-UKF: Unscented kalman filters with Gaussian process prediction and observation models. IROS (2007) 1901–1907
9. Bagnell, J.A., Hneider, J.G.S.: Autonomous helicopter control using reinforcement learning policy search methods. ICRA **2** (2001) 1615–1620
10. Venkatraman, A., Hebert, M., Bagnell, J.A.: Improving multi-step prediction of learned time series models. In: AAAI. (2015) 3024–3030
11. Van Overschee, P., De Moor, B.: N4sid: Subspace algorithms for the identification of combined deterministic-stochastic systems. Automatica **30**(1) (1994) 75–93
12. Ghahramani, Z., Roweis, S.T.: Learning nonlinear dynamical systems using an em algorithm. NIPS (1999) 431–437
13. Siddiqi, S.M., Boots, B., Gordon, G.J.: A constraint generation approach to learning stable linear dynamical systems. NIPS (2007)
14. Van Overschee, P., De Moor, B.: Subspace identification for linear systems: TheoryImplementationApplications. Springer Science & Business Media (2012)
15. Venkatraman, A., Boots, B., Hebert, M., Bagnell, J.A.: Data as demonstrator with applications to system identification. ALR Workshop, NIPS (2014)
16. Abbeel, P., Ng, A.Y.: Exploration and apprenticeship learning in reinforcement learning. In: ICML, ACM (2005) 1–8
17. Deisenroth, M., Rasmussen, C.E.: Pilco: A model-based and data-efficient approach to policy search. ICML (2011) 465–472
18. Ross, S., Bagnell, D.: Agnostic system identification for model-based reinforcement learning. ICML (2012) 1703–1710
19. Heess, N., Wayne, G., Silver, D., Lillicrap, T., Erez, T., Tassa, Y.: Learning continuous control policies by stochastic value gradients. In: NIPS. (2015) 2926–2934
20. Abbeel, P., Ganapathi, V., Ng, A.Y.: Learning vehicular dynamics, with application to modeling helicopters. In: NIPS. (2005) 1–8
21. Müller, K., Smola, A., Rätsch, G.: Predicting time series with support vector machines. ICANN **1327** (1997)
22. Li, W., Todorov, E.: Iterative linear quadratic regulator design for nonlinear biological movement systems. In: ICINCO (1). (2004) 222–229
23. Rahimi, A., Recht, B.: Random features for large-scale kernel machines. NIPS (2007)