Agnostic System Identification for Model-Based Reinforcement Learning

Stéphane Ross STEPHANEROSS@CMU.EDU
Robotics Institute, Carnegie Mellon University, PA USA

J. Andrew Bagnell DBAGNELL@RI.CMU.EDU
Robotics Institute, Carnegie Mellon University, PA USA

Abstract

A fundamental problem in control is to learn a model of a system from observations that is useful for controller synthesis. To provide good performance guarantees, existing methods must assume that the real system is in the class of models considered during learning. We present an iterative method with strong guarantees even in the agnostic case where the system is not in the class. In particular, we show that any *no-regret online learning* algorithm can be used to obtain a near-optimal policy, provided some model achieves low training error and access to a good exploration distribution. Our approach applies to both discrete and continuous domains. We demonstrate its efficacy and scalability on a challenging helicopter domain from the literature.

1. Introduction

Model-based reinforcement learning (MBRL) and much of control rely on *system identification*: building a model of a system from observations that is useful for controller synthesis. While often treated as a typical statistical learning problem, system identification presents different fundamental challenges as the executed controller and data generating process are inextricably intertwined. Naively attempting to estimate a controlled system can lead to a model that makes small error on a training set, but exhibits poor controller performance. This problem arises as the policy resulting from controller synthesis is often very different from the "exploration" policy used to collect data. While we might expect the model to make good predictions at states frequented by the exploration policy, the learned policy usually induces a different state distribution, where the model may poorly capture system behavior (Fig. 1).

Appearing in *Proceedings of the 29th International Conference on Machine Learning*, Edinburgh, Scotland, UK, 2012. Copyright 2012 by the author(s)/owner(s).

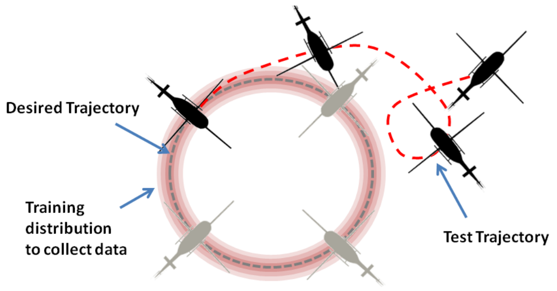

Figure 1. Example train-test mismatch in a helicopter domain. Train: model is fit based on samples near the desired trajectory, e.g. from watching an expert. Test: learned policy ends up in new regions where model is bad, leading to poor control performance.

This problem is fully appreciated in the system identification literature and has been attacked by considering "open loop" identification procedures and "persistent excitation" (Ljung, 1999; Abbeel & Ng, 2005) that attempt to sufficiently "cover" the state-action space. Unfortunately, such methods rely on the strong assumption that the true system lies in the class of models considered: *e.g.*, for continuous systems, they may require the true system to be modeled in a class of linear models. With this assumption, they ensure that eventually the correct model is learned– *e.g.*, by learning about every discrete state-action pair or all modes of the linear system– to provide guarantees.

In this work, we provide algorithms for system identification and controller synthesis (*i.e.* MBRL) that have strong performance guarantees with a weaker agnostic assumption that the system identification achieves statistically good prediction. We adopt a reduction-based analysis (Beygelzimer et al., 2005) that relates the learned policy's performance to prediction error during training. We begin by providing agnostic bounds for a simple generic "batch" algorithm that can represent many learning methods used in practice (*e.g.*, building a model from open loop controls, watching an expert, or running a base policy we want to improve upon). Due to the mismatch in train/test distributions, uniform exploration is often the best option with this approach. Unfortunately, this makes the sample complexity and performance bounds scale with the size of the Markov Decision Process (MDP) (*i.e.* state/action space). Next, we propose a simple iterative approach, closely related to online learning, with stronger guarantees that do not scale with the size of the MDP when given a good exploration distribution. The approach is very simple to implement and iterates between 1) collecting new data about the system by executing a good policy under the current model, as well as by sampling from a given exploration distribution, and 2) updating the model with that new data.

This approach is inspired by a recent reduction of imitation learning to no-regret online learning (Ross et al., 2011) that addresses mismatch between train/test distributions. Our results can be interpreted as a *reduction* of MBRL to no-regret online learning and optimal control, and show that any no-regret algorithm can be used in such a way to learn a policy with strong agnostic guarantees. This enables MBRL methods to match the strongest existing agnostic guarantees of model-free RL methods (Kakade & Langford, 2002; Bagnell et al., 2003).

We first introduce notation and related work. Then we present the batch method and our online learning approach with their agnostic guarantees (proofs are deferred to the supplementary material). Finally we demonstrate the efficacy of our approach on a challenging domain from the literature: learning to perform aerobatic maneuvers with a simulated helicopter (Abbeel & Ng, 2005).

2. Background and Notation

We assume the real system behaves according to some unknown MDP, represented by a set of states S and actions A (both potentially infinite and continuous), a transition function T, where T_{sa} denotes the next state distribution if we do action a in state s, and the initial state distribution μ at time 1. We assume the cost function $C : S \times A \to \mathbb{R}$ is known and seek to minimize the expected sum of discounted costs over an infinite horizon with discount γ.

For any policy π, let π_s be the action distribution performed by π in state s; $D^t_{\omega,\pi}$ the state-action distribution at time t if we started in state distribution ω at time 1 and followed π; $D_{\omega,\pi} = (1-\gamma)\sum_{t=1}^{\infty}\gamma^{t-1}D^t_{\omega,\pi}$ the state-action distribution over the infinite horizon if we follow π, starting in ω at time 1; $V_\pi(s) = \mathbb{E}_{a\sim\pi_s,s'\sim T_{sa}}[C(s,a)+\gamma V_\pi(s')]$ the value function of π (the expected sum of discounted costs of following π starting in state s); $Q_\pi(s,a) = C(s,a)+\gamma\mathbb{E}_{s'\sim T_{sa}}[V_\pi(s')]$ the action-value function of π (the expected sum of discounted costs of following π after starting in s and performing action a); and $J_\omega(\pi) = \mathbb{E}_{s\sim\omega}[V_\pi(s)] = \frac{1}{1-\gamma}\mathbb{E}_{(s,a)\sim D_{\omega,\pi}}[C(s,a)]$ the expected sum of discounted costs of following π starting in ω.

Our goal is to obtain a policy π with small regret, *i.e.* for any policy π', $J_\mu(\pi) - J_\mu(\pi')$ is small. This is achieved indirectly by learning a model $\hat{T}$ of the system and solving for a (near-)optimal policy (under $\hat{T}$); *e.g.*, using dynamic programming (Puterman, 1994) or approximate methods (Szepesvári, 2005; Williams, 1992). For continuous systems, an important special case is linear models with quadratic cost functions, and potentially additive Gaussian noise, known as Linear Quadratic Regulators (LQR)[1] which can be solved exactly and efficiently. Non-linear systems with non-quadratic cost functions can also be solved approximately (local optima) using efficient iterative linearization techniques such as iLQR(Li & Todorov, 2004).

Related Work: In contrast with "textbook" system identification methods, in practice control engineers often proceed iteratively to build good models for controller synthesis. A first batch of data is collected to fit a model and obtain a controller, which is then tested in the real system. If performance is unsatisfactory, data collection is repeated with different sampling distributions to improve the model where needed, until control performance is satisfactory. By doing so, engineers can use feedback of the policies found during training to decide how to collect data and improve performance. Such methods are commonly used in practice and have demonstrated good performance in the work of Atkeson & Schaal (1997); Abbeel & Ng (2005). In both works, the authors proceed by fitting a first model from state transitions observed during expert demonstrations of the task, and at following iterations, using the optimal policy under the current model to collect more data and fit a new model with all data seen so far. Abbeel & Ng (2005) show this approach has good guarantees in non-agnostic settings (for finite MDPs or LQRs), in that it must find a policy that performs as well as the expert providing the initial demonstrations. Our method can be seen as making algorithmic this engineering practice, extending and generalizing the previous methods of Atkeson & Schaal (1997); Abbeel & Ng (2005), and suggesting slight modifications that provide good guarantees even in *agnostic* settings.

Similarly, the Dataset Aggregation (DAgger) algorithm of Ross et al. (2011) uses a similar data aggregation procedure over iterations to obtain policies that mimic an expert well in imitation learning. The authors show that such

[1]LQR is defined by 4 matrices A,B,Q,R s.t. $x_{t+1} = Ax_t + Bu_t + \xi_t$, for x_t and u_t the state and action at time t, and $\xi_t \sim N(0,\Sigma)$ is (optional) Gaussian white noise, and the cost $C(x,u) = x^\top Qx + u^\top Ru$ ($Q \succeq 0$, $R \succ 0$). The optimal policy is linear ($u = Kx$) and the value function is quadratic ($x^\top Vx$). LQR can be solved by dynamic programming on V and K.

a procedure can be interpreted as an online learning algorithm (Hazan et al., 2006; Kakade & Shalev-Shwartz, 2008), more specifically, Follow-the-(Regularized)-Leader (Hazan et al., 2006), and that using any no-regret online algorithm ensures good performance. Our approach can be seen as an extension of DAgger to MBRL settings.

Our approach leverages the way agnostic model-free RL algorithms perform exploration. Methods such as Conservative Policy Iteration (CPI) (Kakade & Langford, 2002) and Policy-Search by Dynamic Programming (PSDP) (Bagnell et al., 2003) learn a policy directly by updating policy parameters iteratively. For exploration, they assume access to a state exploration distribution ν that they can restart the system from and can guarantee finding a policy performing nearly as well as any policies inducing a state distribution (over a whole trajectory) close to ν. Similarly, our approach uses a state-action exploration distribution to sample transitions and allows us to guarantee small regret against any policy with a state-action distribution close to this exploration distribution. If the exploration distribution is close to that of a near-optimal policy, then our approach guarantees near-optimal performance, provided a good model of data exists. This allows our model-based method to match the strongest agnostic guarantees of existing model-free methods. Good exploration distributions can often be obtained in practice; *e.g.*, from human expert demonstrations, domain knowledge, or from a desired trajectory we would like the system to follow. Additionally, if we have a base policy we want to improve, it can be used to generate the exploration distribution – with potentially additional random exploration in the actions.

3. A Simple Batch Algorithm

We now describe a simple algorithm, refered to as *Batch*, that can be used to analyze many common approaches from the literature, *e.g.*, learning from a generative model[2], open loop excitation or by watching an expert (Ljung, 1999).

Let $\mathcal{T}$ denote the class of transition models considered, and ν a state-action exploration distribution we can sample the system from. *Batch* first executes in the real system m state-action pairs sampled i.i.d. from ν to obtain m sampled transitions. Then it finds the best model $\hat{T} \in \mathcal{T}$ of observed transitions, and solves (potentially approximately) the optimal control (OC) problem with $\hat{T}$ and known cost function C to return a policy $\hat{\pi}$ for test execution.

3.1. Analysis

Our reduction analysis seeks to answer the following question: if *Batch* learns a model $\hat{T}$ with small error on training data, and solves the OC problem well, what guarantees does it provide on control performance of $\hat{\pi}$? Our results illustrate the drawbacks of a purely batch method due to the mismatch in train-test distribution.

We measure the quality of the OC problem's solution as follows. For any policy π', let $\epsilon_{\text{oc}}^{\pi'} = \mathbb{E}_{s\sim\mu}[\hat{V}^{\hat{\pi}}(s) - \hat{V}^{\pi'}(s)]$ denote how much better π' is compared to $\hat{\pi}$ on model $\hat{T}$ ($\hat{V}^{\hat{\pi}}$ and $\hat{V}^{\pi'}$ are the value functions of $\hat{\pi}$ and π' under learned model $\hat{T}$ respectively). If $\hat{\pi}$ is an ϵ-optimal policy on $\hat{T}$ within some class of policies Π, then $\epsilon_{\text{oc}}^{\pi'} \leq \epsilon$ for all $\pi' \in \Pi$. A natural measure of model error that arises from our analysis is in terms of L_1 distance between the predicted and true next state's distributions. That is, we define $\epsilon_{\text{prd}}^{\text{L1}} = \mathbb{E}_{(s,a)\sim\nu}[||T_{sa} - \hat{T}_{sa}||_1]$ the predictive error of $\hat{T}$, measured in L_1 distance, under the training distribution ν. However, the L_1 distance cannot be evaluated or optimized from sampled transitions during training (we observe samples from T_{sa} but not the distribution). Therefore we also provide our bounds in terms of other losses we can minimize from samples. This directly relates control performance to the model's training loss. A convenient loss is the KL divergence between T_{sa} and $\hat{T}_{sa}$: $\epsilon_{\text{prd}}^{\text{KL}} = \mathbb{E}_{(s,a)\sim\nu,s'\sim T_{sa}}[\log(T_{sa}(s')) - \log(\hat{T}_{sa}(s'))]$. Minimizing KL corresponds to maximizing the log likelihood of the sampled transitions. This is convenient for common model classes, such as linear models (as in LQR), where it amounts to linear regression. For particular cases where $\mathcal{T}$ is a set of deterministic models and the real system has finitely many states, the predictive error can be measured via a classification loss at predicting the next state: $\epsilon_{\text{prd}}^{\text{cls}} = \mathbb{E}_{(s,a)\sim\nu,s'\sim T_{sa}}[\ell(\hat{T}, s, a, s')]$, for ℓ the 0-1 loss of whether $\hat{T}$ predicts s' for (s,a), or any upper bound on the 0-1 loss, *e.g.*, the multi-class hinge loss if $\mathcal{T}$ is a set of SVMs. In this case, model fitting is a supervised classification problem and the guarantee is directly related to the training classification loss. These are related as follows:

Lemma 3.1. *$\epsilon_{prd}^{L1} \leq \sqrt{2\epsilon_{prd}^{KL}}$ and $\epsilon_{prd}^{L1} \leq 2\epsilon_{prd}^{cls}$. The latter holds with equality if ℓ is the 0-1 loss.*

In general, we can use any loss minimizable from samples that upper bounds $\epsilon_{\text{prd}}^{\text{L1}}$ for models in the class. Our bounds are also related to the mismatch between the exploration distribution ν and distribution induced by executing another policy π starting in μ, denoted $c_\nu^\pi = \sup_{s,a} \frac{D_{\mu,\pi}(s,a)}{\nu(s,a)}$. We assume the costs $C(s,a) \in [C_{\min}, C_{\max}]\ \forall(s,a)$. Let $C_{\text{rng}} = C_{\max} - C_{\min}$ and $H = \frac{\gamma C_{\text{rng}}}{(1-\gamma)^2}$. H is a scaling factor that relates model error to error in total cost predictions.

Theorem 3.1. *The policy $\hat{\pi}$ is s.t. for any policy π':*

$$J_\mu(\hat{\pi}) \leq J_\mu(\pi') + \epsilon_{oc}^{\pi'} + \frac{c_\nu^{\hat{\pi}} + c_\nu^{\pi'}}{2} H \epsilon_{prd}^{L1}$$

This also holds as a function of ϵ_{prd}^{KL} or ϵ_{prd}^{cls} using Lem. 3.1.

[2]With a generative model, we can set the system to any state, perform any action to obtain a sample transition.

This bound indicates that if *Batch* solves the OC problem well and $\hat{T}$ has small enough error under the training distribution ν, then it must find a good policy. Importantly, this bound is tight: *i.e.* we can construct examples where it holds with equality (see supplementary material). More interestingly is what happens as we collect more data. If the fitting procedure is consistent (i.e. picks a model with minimal loss in the class asymptotically), then we can relate this guarantee to the capacity of the model class to achieve low error under the training distribution ν. We denote the modeling error, measured in L_1 distance, as $\epsilon_{\text{mdl}}^{\text{L1}} = \inf_{T' \in \mathcal{T}} \mathbb{E}_{(s,a)\sim\nu}[||T_{sa} - T'_{sa}||_1]$. Similarly, define $\epsilon_{\text{mdl}}^{\text{KL}} = \inf_{T' \in \mathcal{T}} \mathbb{E}_{(s,a)\sim\nu, s' \sim T_{sa}}[\log(T_{sa}(s')) - \log(T'_{sa}(s'))]$ and $\epsilon_{\text{mdl}}^{\text{cls}} = \inf_{T' \in \mathcal{T}} \mathbb{E}_{(s,a)\sim\nu, s'\sim T_{sa}}[\ell(T', s, a, s')]$. These are all 0 in realizable settings, but generally non-zero in agnostic settings. After sampling m transitions, the generalization error $\epsilon_{\text{gen}}^{\text{L1}}(m, \delta)$ bounds with high probability $1-\delta$ the quantity $\epsilon_{\text{prd}}^{\text{L1}} - \epsilon_{\text{mdl}}^{\text{L1}}$. Similarly, $\epsilon_{\text{gen}}^{\text{KL}}(m,\delta)$ and $\epsilon_{\text{gen}}^{\text{cls}}(m,\delta)$ denote the generalization error for the KL and classification loss respectively. $\epsilon_{\text{gen}}^{\text{cls}}(m,\delta)$ can be related to the VC dimension (or multi-class equivalent) in finite MDPs.

Corollary 3.1. *After observing m transitions, with probability at least $1-\delta$, for any policy π':*

$$J_\mu(\hat{\pi}) \leq J_\mu(\pi') + \epsilon_{oc}^{\pi'} + \frac{c_\nu^{\hat{\pi}} + c_\nu^{\pi'}}{2} H[\epsilon_{mdl}^{L1} + \epsilon_{gen}^{L1}(m,\delta)].$$

This also holds as a function of $\epsilon_{mdl}^{KL} + \epsilon_{gen}^{KL}(m,\delta)$ (or $\epsilon_{mdl}^{cls} + \epsilon_{gen}^{cls}(m,\delta)$) using Lem. 3.1. In addition, if the fitting procedure is consistent in terms of L_1 distance (or KL, classification loss), then $\epsilon_{gen}^{L1}(m,\delta) \to 0$ (or $\epsilon_{gen}^{KL}(m,\delta) \to 0$, $\epsilon_{gen}^{cls}(m,\delta) \to 0$) as $m \to \infty$ for any $\delta > 0$.

The generalization error typically scales with the complexity of the class $\mathcal{T}$ and goes to 0 at a rate of $O(\frac{1}{\sqrt{m}})$ ($\tilde{O}(\frac{1}{m})$ in ideal conditions). Given enough samples, the dominating factor limiting performance becomes the modeling error: *i.e.* the term $\frac{c_\nu^{\hat{\pi}} + c_\nu^{\pi'}}{2} H \epsilon_{\text{mdl}}^{\text{L1}}$ (or equivalently $\frac{c_\nu^{\hat{\pi}} + c_\nu^{\pi'}}{2} H \sqrt{2\epsilon_{\text{mdl}}^{\text{KL}}}$ and $(c_\nu^{\hat{\pi}} + c_\nu^{\pi'}) H \epsilon_{\text{mdl}}^{\text{cls}}$) quantifies how performance degrades for agnostic settings.

Drawback of Batch: The two factors $c_\nu^{\hat{\pi}}$ and $c_\nu^{\pi'}$ are qualitatively different. $c_\nu^{\pi'}$ measures how well ν explores state-actions visited by the policy π' we compare to. This factor is inevitable: we cannot hope to compete against policies that spend most of their time where we rarely explore. $c_\nu^{\hat{\pi}}$ measures the mismatch in train-test distribution. Its presence is the major drawback of *Batch*. As $\hat{\pi}$ cannot be known in advance, we can only bound $c_\nu^{\hat{\pi}}$ by considering all policies we could learn: $\sup_{\pi \in \Pi} c_\nu^{\pi}$. This worst case is likely to be realized in practice: if ν rarely explores some state-action regions, the model could be bad for these and significantly underestimate their cost. The learned policy is thus encouraged to visit these low-cost regions where few data were collected. To minimize $\sup_{\pi\in\Pi} c_\nu^{\pi}$, the best ν for *Batch* is often a uniform distribution, when possible. This introduces a dependency on the number of states and actions (or state-action space volume) (*i.e.* $c_\nu^{\hat{\pi}} + c_\nu^{\pi'}$ is $O(|S||A|)$) multiplying the modeling error. Sampling from a uniform distribution often requires access to a generative model. If we only have access to a reset model[3] and a base policy π_0 inducing ν when executed in the system, then $c_\nu^{\hat{\pi}}$ could be arbitrarily large (*e.g.*, if $\hat{\pi}$ goes to 0 probability states under π_0), and $\hat{\pi}$ arbitrarily worse than π_0.

In the next section, we show that iterative learning methods can leverage feedback of the learned policies to obtain bounds that do **not** depend on $c_\nu^{\hat{\pi}}$. This leads to better guarantees when we have a good exploration distribution ν (*e.g.*, that of a near-optimal policy), or when we can only collect data via a reset model. This also leads to better performance in practice as shown in the experiments.

4. No-Regret Methods for Agnostic MBRL

Our extension of DAgger to the MBRL setting proceeds as follows. Starting from an initial model $\hat{T}^1 \in \mathcal{T}$, solve (approximately) the OC problem with $\hat{T}^1$ to obtain policy π_1. At each iteration n, collect data about the system by sampling state-action pairs from distribution $\rho_n = \frac{1}{2}\nu + \frac{1}{2}D_{\mu,\pi_n}$: *i.e.* w.p. $\frac{1}{2}$, sample a transition occurring from an exploratory state-action pair drawn from ν and add it to dataset $\mathcal{D}$, otherwise, sample a state transition occurring from running the current policy π_n starting in μ, stopping the trajectory w.p. $1-\gamma$ at each step and adding the last transition to $\mathcal{D}$. The dataset $\mathcal{D}$ contains all transitions observed so far over all iterations. Once data is collected, find the best model $\hat{T}^{n+1} \in \mathcal{T}$ that minimizes an appropriate loss (*e.g.* regularized negative log likelihood) on $\mathcal{D}$, and solve (approximately) the OC problem with $\hat{T}^{n+1}$ to obtain the next policy π_{n+1}. This is iterated for N iterations. At test time, we could either find and use the policy with lowest expected total cost in the sequence $\pi_{1:N}$, or use the uniform "mixture" policy[4] over $\pi_{1:N}$. We guarantee good performance for both. The last policy π_N often performs equally well, it has been trained with most data. Our experimental results confirm this intuition. In theory, π_N has good guarantees when the distributions D_{μ,π_i} converge to a small region in the space of distributions as $i \to \infty$, but we do not guarantee this always occurs.

Implementation with Off-the-Shelf Online Learner: DAgger as described can be interpreted as using a *Follow-The-(Regularized)-Leader* (FTRL) online algorithm to pick the sequence of models: at each iteration n we pick the

[3]To sample transitions with a reset model, we can only simulate the system forward in time, or reset to a random initial state.

[4]At start of any trajectory, the mixture policy picks uniformly randomly a policy in $\pi_{1:N}$, and uses it for the whole trajectory.

best (regularized) model $\hat{T}^n$ in hindsight under all samples seen so far. In general, DAgger can also be implemented using any no-regret online algorithm (see Algorithm 1) to provide good guarantees. This is done as follows. When minimizing the negative log likelihood, the loss function of the online learning problem at iteration i is: $L_i^{\text{KL}}(\hat{T}) = \mathbb{E}_{(s,a)\sim\rho_i, s'\sim T_{sa}}[-\log(\hat{T}_{sa}(s'))]$. This can be estimated from sampled state transitions at iteration i, and evaluated for any model $\hat{T}$. The online algorithm is applied on the sequence of loss $L_{1:N}^{\text{KL}}$ to obtain a sequence of models $\hat{T}^{1:N}$ over the iterations. As before, each model $\hat{T}^i$ is solved to obtain the next policy π_i. By doing so, the online algorithm effectively runs over mini-batches of data collected at each iteration to update the model, and each mini-batch comes from a different distribution that changes as we update the policy. Similarly, in a finite MDP with a deterministic model class $\mathcal{T}$, we can minimize the 0-1 loss instead (or any upper bound such as hinge loss) where the loss at iteration i is: $L_i^{\text{cls}}(\hat{T}) = \mathbb{E}_{(s,a)\sim\rho_i, s'\sim T_{sa}}[\ell(\hat{T}, s, a, s')]$, for ℓ the particular classification loss. This corresponds to an online classification problem. For many model classes, the negative log likelihood and convex upper bounds on the 0-1 loss (such as hinge loss) lead to convex online learning problems, for which no-regret algorithms exist (*e.g.*, gradient descent, FTRL). As shown below, if the sequence of models is no-regret, then performance can be related to the minimum KL divergence (or classification loss) achievable with model class $\mathcal{T}$ under the overall training distribution $\overline{\rho} = \frac{1}{N}\sum_{i=1}^N \rho_i$ (*i.e.* a quantity akin to $\epsilon_{\text{mdl}}^{\text{KL}}$ or $\epsilon_{\text{mdl}}^{\text{cls}}$ for *Batch*).

Algorithm 1 DAgger algorithm for Agnostic MBRL.

```
Input: exploration distribution $\nu$, number of iterations $N$, number of samples per iteration $m$, cost function $C$, online learning procedure OnlineLearner, optimal control procedure OCSolver.
Get initial guess of model: $\hat{T}^1 \leftarrow$ OnlineLearner().
$\pi_1 \leftarrow$ OCSolver($\hat{T}^1, C$).
for $n = 2$ to $N$ do
  for $k = 1$ to $m$ do
    With prob. $\frac{1}{2}$ sample $(s,a) \sim D_{\mu,\pi_{n-1}}$ using $\pi_{n-1}$,
    otherwise sample $(s,a) \sim \nu$. Obtain $s' \sim T_{sa}$
    Add $(s,a,s')$ to $\mathcal{D}_{n-1}$.
  end for
  Update model: $\hat{T}^n \leftarrow$ OnlineLearner($\mathcal{D}_{n-1}$).
  $\pi_n \leftarrow$ OCSolver($\hat{T}^n, C$).
end for
Return the sequence of policies $\pi_{1:N}$.
```

4.1. Analysis

Similar to our analysis of *Batch*, we seek to answer the following: if there exists a low error model of training data, and we solve each OC problem well, what guarantees does DAgger provide on control performance? Our results show that by sampling data from the learned policies, DAgger provides guarantees that have no train-test mismatch factor, leading to improved performance.

For any policy π', define $\overline{\epsilon}_{\text{oc}}^{\pi'} = \frac{1}{N}\sum_{i=1}^N \mathbb{E}_{s\sim\mu}[\hat{V}_i(s) - \hat{V}_i^{\pi'}(s)]$, where $\hat{V}_i$ and $\hat{V}_i^{\pi'}$ are respectively the value function of π_i and π' under model $\hat{T}^i$. This measures how well we solved each OC problem on average over the iterations. For instance, if at each iteration i we found an ϵ_i-optimal policy within some class of policies Π on learned model $\hat{T}^i$, then $\overline{\epsilon}_{oc}^{\pi'} \leq \frac{1}{N}\sum_{i=1}^N \epsilon_i$ for all $\pi' \in \Pi$. As in *Batch*, the average predictive error of the models $\hat{T}^{1:N}$ can be measured in terms of the L_1 distance between the predicted and true next state distribution: $\overline{\epsilon}_{\text{prd}}^{\text{L1}} = \frac{1}{N}\sum_{i=1}^N \mathbb{E}_{(s,a)\sim\rho_i}[||\hat{T}_{sa}^i - T_{sa}||_1]$. However, as was discussed, the L_1 distance is not observed from samples which makes it hard to minimize. Instead we can define other measures which upper bounds this L_1 distance and can be minimized from samples, such as the KL divergence or classification loss: *i.e.* $\overline{\epsilon}_{\text{prd}}^{\text{KL}} = \frac{1}{N}\sum_{i=1}^N \mathbb{E}_{(s,a)\sim\rho_i, s'\sim T_{sa}}[\log(T_{sa}(s)) - \log(\hat{T}_{sa}^i(s'))]$ and $\overline{\epsilon}_{\text{prd}}^{\text{cls}} = \frac{1}{N}\sum_{i=1}^N \mathbb{E}_{(s,a)\sim\rho_i, s'\sim T_{sa}}[\ell(\hat{T}^i, s, a, s')]$. Now, given the sequence of policies $\pi_{1:N}$, let $\hat{\pi} = \operatorname{argmin}_{\pi\in\pi_{1:N}} J_\mu(\pi)$ be the best policy in the sequence and $\overline{\pi}$ the uniform mixture policy on the sequence.

Lemma 4.1. *The policies $\pi_{1:N}$ are s.t. for any policy π':*

$$J_\mu(\hat{\pi}) \leq J_\mu(\overline{\pi}) \leq J_\mu(\pi') + \overline{\epsilon}_{oc}^{\pi'} + c_\nu^{\pi'} H \overline{\epsilon}_{prd}^{L1}$$

This also holds as a function of $\overline{\epsilon}_{prd}^{KL}$ or $\overline{\epsilon}_{prd}^{cls}$ using Lem. 3.1.

We note that $\overline{\epsilon}_{\text{prd}}^{\text{KL}} = \frac{1}{N}\sum_{i=1}^N L_i^{KL}(\hat{T}^i) - L_i^{KL}(T)$ and $\overline{\epsilon}_{\text{prd}}^{\text{cls}} = \frac{1}{N}\sum_{i=1}^N L_i^{cls}(\hat{T}^i)$. Using a no-regret algorithm on the sequence of losses $L_{1:N}^{KL}$ implies $\frac{1}{N}\sum_{i=1}^N L_i^{KL}(\hat{T}^i) \leq \inf_{T'\in\mathcal{T}} \frac{1}{N}\sum_{i=1}^N L_i^{KL}(T') + \overline{\epsilon}_{\text{rgt}}^{\text{KL}}$, for $\overline{\epsilon}_{\text{rgt}}^{\text{KL}}$ the average regret of the algorithm after N iterations, s.t. $\overline{\epsilon}_{\text{rgt}}^{\text{KL}} \to 0$ as $N \to \infty$. This relates $\overline{\epsilon}_{\text{prd}}^{\text{KL}}$ to the modeling error of the class $\mathcal{T}$: $\overline{\epsilon}_{\text{mdl}}^{\text{KL}} = \inf_{T'\in\mathcal{T}} \mathbb{E}_{(s,a)\sim\overline{\rho}, s'\sim T_{sa}}[\log(T_{sa}(s)) - \log(T'_{sa}(s'))]$, *i.e.* $\overline{\epsilon}_{\text{prd}}^{\text{KL}} \leq \overline{\epsilon}_{\text{mdl}}^{\text{KL}} + \overline{\epsilon}_{\text{rgt}}^{\text{KL}}$, for $\overline{\epsilon}_{\text{rgt}}^{\text{KL}} \to 0$. Similarly define $\overline{\epsilon}_{\text{mdl}}^{\text{cls}} = \inf_{T'\in\mathcal{T}} \mathbb{E}_{(s,a)\sim\overline{\rho}, s'\sim T_{sa}}[\ell(T', s, a, s')]$ and by using a no-regret algorithm on $L_{1:N}^{cls}$, $\overline{\epsilon}_{\text{prd}}^{\text{cls}} \leq \overline{\epsilon}_{\text{mdl}}^{\text{cls}} + \overline{\epsilon}_{\text{rgt}}^{\text{cls}}$ for $\overline{\epsilon}_{\text{rgt}}^{\text{cls}} \to 0$. In some cases, even if the L_1 distance cannot be estimated from samples, statistical estimators can still be no-regret with high probability on the sequence of loss $L_i^{\text{L1}}(T') = \mathbb{E}_{(s,a)\sim\rho_i}[||T_{sa} - T'_{sa}||_1]$. This is the case in finite MDPs if we use the empirical estimator of T based on data seen so far (see supplementary material). If we define $\overline{\epsilon}_{\text{mdl}}^{\text{L1}} = \inf_{T'\in\mathcal{T}} \mathbb{E}_{(s,a)\sim\overline{\rho}}[||T_{sa} - T'_{sa}||_1]$, this implies that $\overline{\epsilon}_{\text{prd}}^{\text{L1}} \leq \overline{\epsilon}_{\text{mdl}}^{\text{L1}} + \overline{\epsilon}_{\text{rgt}}^{\text{L1}}$, for $\overline{\epsilon}_{\text{rgt}}^{\text{L1}} \to 0$. Our main result follows:

Theorem 4.1. *The policies $\pi_{1:N}$ are s.t. for any policy π':*

$$J_\mu(\hat{\pi}) \leq J_\mu(\overline{\pi}) \leq J_\mu(\pi') + \overline{\epsilon}_{oc}^{\pi'} + c_\nu^{\pi'} H[\overline{\epsilon}_{mdl}^{L1} + \overline{\epsilon}_{rgt}^{L1}]$$

This also holds as a function of $\bar{\epsilon}_{mdl}^{KL} + \bar{\epsilon}_{rgt}^{KL}$ *(or* $\bar{\epsilon}_{mdl}^{cls} + \bar{\epsilon}_{rgt}^{cls}$*) using Lem. 3.1. If the fitting procedure is no-regret w.r.t the sequence of losses* $L_{1:N}^{L1}$ *(or* $L_{1:N}^{KL}$*,* $L_{1:N}^{cls}$*), then* $\bar{\epsilon}_{rgt}^{L1} \to 0$ *(or* $\bar{\epsilon}_{rgt}^{KL} \to 0, \bar{\epsilon}_{rgt}^{cls} \to 0$*) as* $N \to \infty$*.*

Additionally, the performance of π_N can be related to $\overline{\pi}$ if the distributions D_{μ,π_i} converge to a small region:

Lemma 4.2. *If there exists a distribution* D^* *and some* $\epsilon_{cnv}^* \geq 0$ *s.t.* $\forall i$, $||D_{\mu,\pi_i} - D^*||_1 \leq \epsilon_{cnv}^* + \epsilon_{cnv}^i$ *for some sequence* $\{\epsilon_{cnv}^i\}_{i=1}^{\infty}$ *that is* $o(1)$*, then* π_N *is s.t.:*

$$J_\mu(\pi_N) \leq J_\mu(\overline{\pi}) + \frac{C_{rng}}{2(1-\gamma)}[2\epsilon_{cnv}^* + \epsilon_{cnv}^N + \frac{1}{N}\sum_{i=1}^N \epsilon_{cnv}^i]$$

Thus: $\limsup_{N\to\infty} J_\mu(\pi_N) - J_\mu(\overline{\pi}) \leq \frac{C_{rng}}{1-\gamma}\epsilon_{cnv}^*$

Thm. 4.1 illustrates how we can reduce the original MBRL problem to a *no-regret online learning* problem on a particular sequence of loss functions. In general, no-regret algorithms have average regret of $O(\frac{1}{\sqrt{N}})$ ($\tilde{O}(\frac{1}{N})$ in ideal cases) such that the regret term goes to 0 at a similar rate to the generalization error term for *Batch* in Cor. 3.1. Here, given enough iterations, the term $c_\nu^{\pi'} H \bar{\epsilon}_{\text{mdl}}^{\text{L1}}$ determines how performance degrades in the agnostic setting (or $c_\nu^{\pi'} H\sqrt{2\bar{\epsilon}_{\text{mdl}}^{\text{KL}}}$ or $2c_\nu^{\pi'} H\bar{\epsilon}_{\text{mdl}}^{\text{cls}}$ if we use a no-regret algorithm on the sequence of KL or classification loss respectively). Unlike for *Batch*, there is no dependence on $c_\nu^{\hat{\pi}}$, only on $c_\nu^{\pi'}$. Thus, if a low error model exists under training distribution $\overline{\rho}$, no-regret methods are guaranteed to learn policies that performs well compared to any policy π' for which $c_\nu^{\pi'}$ is small. Hence, ν is ideally $D_{\mu,\pi}$ of a near-optimal policy π (*i.e.* explore where good policies go).

Finite Sample Analysis: A remaining issue is that the current guarantees apply if we can evaluate the expected loss (L_i^{L1}, L_i^{KL} or L_i^{cls}) exactly. This requires infinite samples at each iteration. If we run the no-regret algorithm on estimates of these loss functions, *i.e.* loss on m sampled transitions, we can still obtain good guarantees using martingale inequalities as in online-to-batch (Cesa-Bianchi et al., 2004) techniques. The extra generalization error term is typically $O(\sqrt{\frac{\log(1/\delta)}{Nm}})$ with high probability $1-\delta$. While our focus is not on providing such finite sample bounds, we illustrate how these can be derived for two scenarios in the supplementary material. For instance, in finite MDPs with $|S|$ states and $|A|$ actions, if $\hat{T}^i$ is the empirical estimator of T based on samples collected in the first $i-1$ iterations, then choosing $m=1$ and N in $\tilde{O}(\frac{C_{\text{rng}}^2|S|^2|A|\log(1/\delta)}{\epsilon^2(1-\gamma)^4})$ guarantees that w.p. $1-\delta$, for any policy π':

$$J_\mu(\hat{\pi}) \leq J_\mu(\overline{\pi}) \leq J_\mu(\pi') + \bar{\epsilon}_{\text{oc}}^{\pi'} + O(c_\nu^{\pi'}\epsilon)$$

Here, $\bar{\epsilon}_{\text{mdl}}$ does not appear as it is 0 (realizable case). Given a good state-action distribution ν, the sample complexity to get a near-optimal policy is $\tilde{O}(\frac{C_{\text{rng}}^2|S|^2|A|\log(1/\delta)}{\epsilon^2(1-\gamma)^4})$. This improves upon other state-of-the-art MBRL algorithms, such as $R_{\max}$, $\tilde{O}(\frac{C_{\text{rng}}^3|S|^2|A|\log(1/\delta)}{\epsilon^3(1-\gamma)^6})$ (Strehl et al., 2009) and a recent modification of $R_{\max}$, $\tilde{O}(\frac{C_{\text{rng}}^2|S||A|\log(1/\delta)}{\epsilon^2(1-\gamma)^6})$ (Szita & Szepesvári, 2010) (when $|S| < \frac{1}{(1-\gamma)^2}$). Here, the dependency on $|S|^2|A|$ is due to the complexity of the class ($|S|^2|A|$ parameters). With simpler classes, it can have no dependency on the size of the MDP. In the supplementary material, we analyze a scenario where $\mathcal{T}$ is a set of kernel SVM (deterministic models) with RKHS norm bounded by K. Choosing $m=1$ and N in $O(\frac{C_{\text{rng}}^2(K^2+\log(1/\delta))}{\epsilon^2(1-\gamma)^4})$ guarantees that w.p. $1-\delta$, for any policy π':

$$J_\mu(\hat{\pi}) \leq J_\mu(\overline{\pi}) \leq J_\mu(\pi') + \bar{\epsilon}_{\text{oc}}^{\pi'} + 2c_\nu^{\pi'} H\hat{\epsilon}_{\text{mdl}}^{\text{cls}} + O(c_\nu^{\pi'}\epsilon),$$

for $\hat{\epsilon}_{\text{mdl}}^{\text{cls}}$ the multi-class hinge loss on the training set after N iterations of the best SVM in hindsight. Thus, if we have a good exploration distribution and there exists a good model in $\mathcal{T}$ for predicting observed data, we obtain a near-optimal policy with sample complexity that depends only on the complexity of $\mathcal{T}$, not the size of the MDP.

5. Discussion

We emphasize that we provide reduction-style guarantees. DAgger may sometimes fail to find good policies, *e.g.*, when no model in the class achieves low error on the training data. However, DAgger guarantees that one of the following occur: either (1) we find good policies or (2) no models with low error on the aggregate dataset exist. If the latter occurs, we need a better model class. In contrast, *Batch* can find models with low training error, but *still* fail at obtaining a policy with good control performance, due to train/test mismatch. This occurs even in scenarios where DAgger finds good policies, as shown in the experiments.

DAgger needs to solve many OC problems. This can be computationally expensive, *e.g.*, with non-linear or high-dimensional models. Many approximate methods can be used, *e.g.*, policy gradient (Williams, 1992), fitted value iteration (Szepesvári, 2005) or iLQR (Li & Todorov, 2004). As the models often change only slightly from one iteration to the next, we can often run only a few iterations of dynamic programming/policy gradient from the last value function/policy to obtain a good policy for the current model. As long as we get good solutions on average, $\bar{\epsilon}_{\text{oc}}^{\pi'}$ remains small and does not hinder performance.

DAgger generalizes the approach of Atkeson & Schaal (1997) and Abbeel & Ng (2005) so that we can use any no-regret algorithm to update the model, as well as any exploration distribution. A key difference is that DAgger keeps an even balance between exploration data and data from running the learned policies. This is crucial to avoid settling on suboptimal performance in agnostic settings as the exploration data could be ignored if it occupies only a small fraction of the dataset, in favor of models with lower error on the data from the learned policies. With this modification, our main contribution is showing that such methods have good guarantees even in agnostic settings.

6. Experiments on Helicopter Domain

We demonstrate the efficacy of DAgger on a challenging problem: learning to perform aerobatic maneuvers with a simulated helicopter, using the simulator of Abbeel & Ng (2005), which has a continuous 21-dimensional state and 4-dimensional control space. We consider learning to 1) hover and 2) perform a "nose-in funnel" maneuver. We compare DAgger to *Batch* with several choices for ν: 1) ν_t: adding small white Gaussian noise[5] to each state and action along the desired trajectory, 2) ν_e: run an expert controller, and 3) ν_{en}: run the expert controller with additional white Gaussian noise[6] in the controls of the expert. The expert controller is obtained by linearizing the true model about the desired trajectory and solving the LQR (iLQR for the nose-in funnel). We also compare against Abbeel's algorithm, where the expert is only used at the first iteration.

Hover: All approaches begin with an initial model $\Delta x_{t+1} = A\Delta x_t + B\Delta u_t$, for Δx_t the difference between the current and hover state at time t, Δu_t the delta controls at time t, A is identity and B adds the delta controls to the actual controls in Δx_t. We seek to learn offset matrices A', B' that minimizes $||\Delta x_{t+1} - [(A + A')\Delta x_t + (B + B')\Delta u_t]||_2$ on observed data[7]. We attempt to learn to hover in the presence of noise[8] and delay of 0 and 1. A delay of 1 introduces high-order dynamics that cannot be modeled with the current state. All methods sample 100 transitions per iteration and run for: 50 iterations when delay is 0; 100 iterations when delay is 1. Figure 2 shows the test performance of each method after each iteration. In both cases, for any choice of ν, DAgger outperforms *Batch* significantly and converges to a good policy faster. DAgger is more robust to the choice of ν, as it always obtains good performance given enough iterations, whereas *Batch* obtains good performance with only one choice of ν in each case. Also, DAgger eventually learns a policy that outperforms the expert policy (L). As the expert policy is inevitably visiting states far from the hover state due to the large noise and delay (unknown to the expert), the linearized model is not as good at those states, leading to slightly suboptimal performance. Thus DAgger is learning a better linear model for the states visited by the learned policy which leads to better performance. Abbeel's algorithm improves the initial policy but reaches a plateau. This is due to lack of exploration (expert demonstrations) after the first iteration. While our objective is to show that DAgger outperforms other model-based approaches, we also compared against a model-free policy gradient method similar to CPI[9]. However, 100 samples per iteration were insufficient to get good gradient estimates and lead to only small improvement. Even with 500 samples per iteration, it could only reach an avg. total cost $\sim$15000 after 100 iterations.

Nose-In Funnel: This maneuver consists in rotating at a fixed speed and distance around an axis normal to the ground with the helicopter's nose pointing towards the axis of rotation (desired trajectory in Fig. 1). We attempt to learn to perform 4 complete rotations of radius 5 in the presence of noise[10] but no delay. We start each approach with a linearized model about the hover state and learn a time-varying linear model[11]. All methods collect 500 samples per iteration over 100 iterations. Figure 2 (bottom) shows the test performance after each iteration. With the initial model (0 data), the controller fails to produce the maneuver and performance is quite poor. Again, with any choice of ν, DAgger outperforms *Batch*, and unlike *Batch*, it performs well with all choices of ν. A video comparing qualitatively the learned maneuver with DAgger and *Batch* is available on YouTube (Ross, 2012). Abbeel's method improves performance slightly but again suffers from lack of expert demonstrations after the first iteration.

7. Conclusion

We presented a no-regret online learning approach to MBRL that has strong performance, both in theory and practice, even in agnostic settings. It is simple to implement, formalizes and makes algorithmic the engineering practice of iterating between controller synthesis and system identification, and can be applied to any control problem where approximately solving the OC problem is feasible. Additionally, its sample complexity scales with model

[5]Covariance of $0.0025I$ for states and $0.0001I$ for actions.

[6]Covariance of $0.0001I$.

[7]We also use a Frobenius norm regularizer on A' and B': $\min_{A',B'} \frac{1}{n}\sum_{i=1}^{n} ||\Delta x'_i - [(A+A')\Delta x_i + (B+B')\Delta u_i]||_2 + \frac{\lambda}{\sqrt{n}}(||A'||_F^2 + ||B'||_F^2)$, for $\lambda = 10^{-3}$, n the number of samples and $(\Delta x_i, \Delta u_i, \Delta x'_i)$ the i^{th} transition in the dataset. During training we stop a trajectory if it becomes too far from the hover state, *i.e.* if $||[\Delta x; \Delta u]||_2 > 5$ as this represents an event that would have to be recovered from. During testing, we run the trajectory until completion (400 timesteps of 0.05s, 20s total).

[8]White Gaussian noise with covariance I on the forces and torques applied to the helicopter at each step.

[9]Same as CPI, except gradient descent is done directly on deterministic linear controller. We solve a linear system to estimate the gradient from sample cost with perturbed parameters.

[10]Zero-mean spherical Gaussian with standard deviation 0.1 on the forces and torques applied to the helicopter at each step.

[11]For each time step t, we learn offset matrices A'_t, B'_t such that $\Delta x_{t+1} = (A + A'_t)\Delta x_t + (B + B'_t)\Delta u_t + x^*_{t+1} - x^*_t$, for x^*_t the desired state at time t and A, B the given hover model.

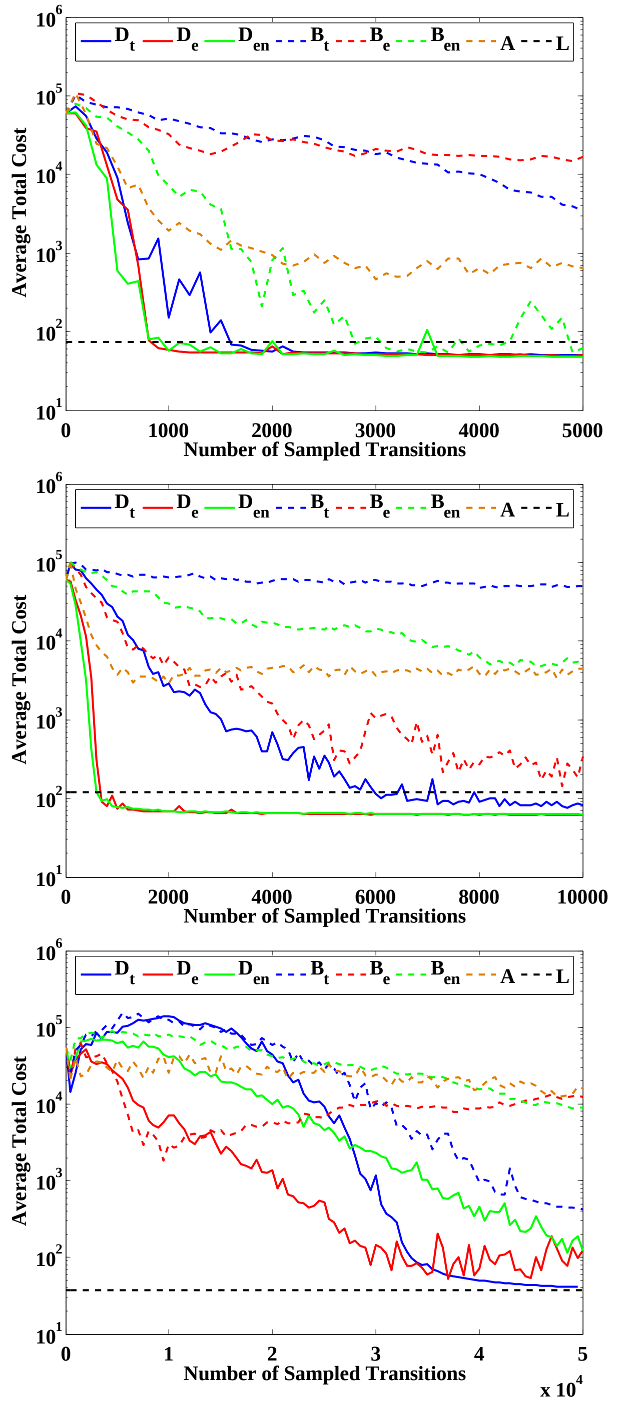

Figure 2. Average total cost on test trajectories as a function of data collected so far, averaged over 20 repetitions of the experiments, each starting with a different random seed (all approaches use the same 20 seeds) From top to bottom: hover with no delay, hover with delay of 1, nose-in funnel. D_t, D_e and D_{en} denotes DAgger using exploration distribution ν_t, ν_e and ν_{en} respectively, similarly B_t, B_e and B_{en} for the Batch algorithm, A for Abbeel's algorithm, and L for the linearized model's optimal controller.

class complexity, not the size of the MDP. To our knowledge, this is the first practical MBRL algorithm with agnostic guarantees. The only other agnostic MBRL approach we are aware of is a recent agnostic extension of $R_{\max}$ (Szita & Szepesvári, 2011) that is largely theoretical: it requires unknown quantities to run the algorithm (*e.g.*, distance between the real system and the model class) and its sample complexity is exponential in the class complexity.

Acknowledgements

This work is supported by the ONR MURI grant N00014-09-1-1052, Reasoning in Reduced Information Spaces.

References

Abbeel, P. and Ng, A. Y. Exploration and apprenticeship learning in reinforcement learning. In *ICML*, 2005.

Atkeson, C. G. and Schaal, S. Robot learning from demonstration. In *ICML*, 1997.

Bagnell, J. A., Ng, A. Y., Kakade, S., and Schneider, J. Policy search by dynamic programming. In *NIPS*, 2003.

Beygelzimer, A., Dani, V., Hayes, T., Langford, J., and Zadrozny, B. Error limiting reductions between classification tasks. In *ICML*, 2005.

Cesa-Bianchi, N., Conconi, A., and Gentile, C. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 2004.

Hazan, E., Kalai, A., Kale, S., and Agarwal, A. Logarithmic regret algorithms for online convex optimization. In *COLT*, 2006.

Kakade, S. and Langford, J. Approximately optimal approximate reinforcement learning. In *ICML*, 2002.

Kakade, S. and Shalev-Shwartz, S. Mind the duality gap: Logarithmic regret algorithms for online optimization. In *NIPS*, 2008.

Li, W. and Todorov, E. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *ICINCO*, 2004.

Ljung, L. *System Identification: Theory for the User*. Prentice Hall, 1999.

Puterman, M. L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 1994.

Ross, S. Helicopter learning nose-in funnel., 2012. URL `http://www.youtube.com/user/icml12rl`.

Ross, S., Gordon, G., and Bagnell, J. A. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011.

Strehl, A. L., Li, L., and Littman, M. L. Reinforcement learning in finite MDPs: PAC analysis. *JMLR*, 2009.

Szepesvári, C. Finite time bounds for sampling based fitted value iteration. In *ICML*, 2005.

Szita, I. and Szepesvári, C. Model-based reinforcement learning with nearly tight exploration complexity bounds. In *ICML*, 2010.

Szita, I. and Szepesvári, C. Agnostic kwik learning and efficient approximate reinforcement learning. In *COLT*, 2011.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 1992.