DATA AS DEMONSTRATOR with Applications to System Identification

Arun Venkatraman
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
aruvenk@cs.cmu.edu

Byron Boots
School of Interactive Computing
Georgia Institute of Technology
Atlanta, GA 30332
bboots@cc.gatech.edu

Martial Hebert
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
hebert@ri.cmu.edu

J. Andrew Bagnell
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
dbagnell@ri.cmu.edu

Abstract

Machine learning techniques for system identification and time series modeling often phrase the problem as the optimization of a loss function over a single time-step prediction. However, in many applications, the learned model is recursively applied in order to make a multiple-step prediction, resulting in compounding prediction errors. We present DATA AS DEMONSTRATOR [15], an approach that reuses training data to make a no-regret learner robust to errors made during multi-step prediction. We present results on the task of linear system identification applied to a simulated system and to a real world dataset.

1 Introduction

As robotic technology becomes more complex and is progressively integrated into natural and human environments, it becomes difficult to robustly characterize the robot dynamics *a priori* with simple analytic models. As a result, machine learning is an increasingly important tool: models of noisy, complicated dynamics can be learned directly from a robot's interaction with its environment.

Most existing work has focused on learning forward models, which predict the next state of a dynamical system given the current state and an action, and inverse models, which predict the action required to advance a dynamical system from the current state to a desired future state. Many popular learning-based approaches have been used to learn forward and inverse models. Examples include Support Vector Regression [10], Gaussian process regression [16, 7], Nadaraya-Watson kernel regression [2], Gaussian mixture models [6], and Kernel PCA [12]. However, virtually *all* of these approaches optimize a *single-step* criterion. This is despite the fact that in many robotics problems, such as forecasting and open-loop control, the ultimate goal is to recursively predict the next T-steps into the future. A straightforward idea is to apply single-step prediction models T times in sequence, however, this approach is particularly susceptible to error accumulation over time.

The prevalence of single-step modeling approaches seems to be a consequence of the difficulty in directly optimizing multiple-step prediction error. Consider a multi-step criterion for fitting a simple

linear dynamical system model over a time horizon T:

$$A^* = \arg\min_A \sum_{t=1}^{T} \left\| x_t - A^t x_0 \right\|_2^2 \tag{1}$$

Even this squared-loss objective is difficult to optimize in two ways: it is non-convex in A, and though differentiable, the matrix power derivatives are non-trivial. In comparison, the standard single-step squared loss used in supervised learning,

$$A^* = \arg\min_A \sum_{t=0}^{T-1} \left\| x_{t+1} - A x_t \right\|_2^2 \tag{2}$$

is considerably more appealing to solve as it has an easy, closed form solution. This scenario arises for loss functions other than the squared loss example above. To contend with this problem, Abbeel et al. propose a generalization of (Eq. 1) coined the "lagged error" criterion, which penalizes deviations during forward simulation. However, this objective is also non-linear and iterative optimization is used to find a local optimum to the multi-step error [1] .

If the predictive model is differentiable, one can apply "backpropagation-through-time" for learning dynamical systems and time series models [17, 9]. Unfortunately, such gradient methods are limited in the model classes they can consider and tend to a suffer from a "gradient collapse" and ill-conditioning [3], where the gradient decreases exponentially in the prediction horizon T. Finally, to our knowledge, no formal guarantees have been provided for any such optimization of multi-step predictive error. Perhaps these considerations make it preferable to use the latest advances in machine learning for solving the single-step prediction problem.

Motivated by these problems, we detail a new meta-algorithm DATA AS DEMONSTRATOR that can improve the multi-step prediction ability of single-step learned models. Through a reduction to imitation learning, we establish a strong performance guarantee on the relation between training error and the multi-step prediction error for our algorithm. We apply DATA AS DEMONSTRATOR to two important problems: learning recursive forward models and learning partially observable linear dynamical systems. In addition to providing strong theoretical guarantees we demonstrate the practical applicability of our algorithm on simulated and real data.

2 Problem Formulation

We consider the problem of modeling a discrete-time dynamical system characterized by a time-indexed state x_t generated from stationary dynamics,

$$x_{t+1} = f(x_t) + \varepsilon_t \tag{3}$$

Our goal is to learn a model $\widehat{M}$ given K sample trajectories $\xi \in \Xi$ of $\{x_0, x_1, \ldots, x_{T_k}\}$ generated by the system (Eq. 3). As motivated in the introduction, we learn a forward prediction model by minimizing a loss function over the sequential steps in each trajectory. To do so, we create a dataset D of input-target pairs $\{(x_t, x_{t+1})\}_i$ and optimize for a learned model:

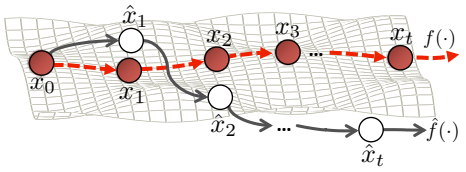
$$\widehat{M} = \arg\min_{M \in \Gamma} \sum_i \ell_M(\{(x_t, x_{t+1})\}_i) \tag{4}$$

for some regression loss function ℓ and class of models Γ. For multiple-step prediction and simulation with the learned model $\widehat{M}$, we follow the simple two-step procedure:

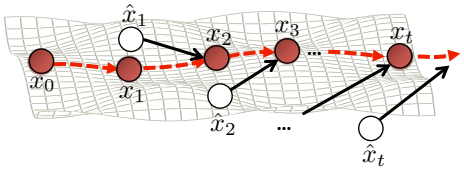
Step 1: $\hat{x}_{t+1} = \widehat{M}(\hat{x}_t)$

Step 2: Return to **Step 1** with $\hat{x}_t \leftarrow \hat{x}_{t+1}$

In the supervised learning setting, we would expect to achieve error linear in the number of predictions – the time horizon for multi-step prediction. However, this ignores the feedback effect of using the learner's output in order to make future predictions, breaking the train-test i.i.d. assumption common to supervised learning. In practice, this can result in cascading errors that quickly add up. For example, if a learned linear model is unstable (largest eigenvalue is greater than 1), a prediction error ϵ at the first time step can cause exponentially increasing error. We are able to upper bound the multi-step prediction error under a couple of "nice" assumptions about the learned model:



(a) Forward simulation of learned model (gray) introduces error at each prediction step compared to the true time-series (red)



(b) Data provides demonstration of corrections required to return back to proper prediction

Figure 1: In typical time-series systems, realized states of the true system are a small subset or even low-dimensional manifold of all possible states. **(1a)** Cascading prediction error from forward simulation with a learned model will often result in predicted infeasible states. **(1b)** Our algorithm, *Data as Demonstrator* (DAD), generates synthetic examples for the learner to ensure that prediction returns to typical states.

Theorem 1. *Let* $\widehat{M}$ *be learned model with bounded single-step prediction error* $\|\widehat{M}(x_t) - x_{t+1}\| \leq \epsilon$. *Also let* $\widehat{M}$ *be Lipshitz continuous with constant* $L > 1$ *under the metric* $\|\cdot\|$. *Then,* $\|\widehat{M}(\hat{x}_T) - x_{T+1}\| \in O\left(\exp(T\log(L))\epsilon\right)$

Proof. From the Lipshitz continuous property, bounded error assumption, and the triangle inequality, we can show that $\|\widehat{M}(\hat{x}_T) - \widehat{M}(x_T)\| \leq L\|\widehat{M}(\hat{x}_{T-1}) - \widehat{M}(x_{T-1})\| + L\epsilon$. Applying the same process, we eventually get $\|\widehat{M}(\hat{x}_T) - \widehat{M}(x_T)\| \leq \sum_{t=1}^{T} L^t\epsilon$. Using the bounded error assumption along with another application of triangle inequality, we arrive at $\|\widehat{M}(\hat{x}_T) - x_{T+1}\| \leq \sum_{t=0}^{T} L^t\epsilon \in O\left(\exp(T\log(L))\epsilon\right)$. □

That is, the error can be exponentially bad in the time horizon. In the following sections, we motivate and develop an algorithm which can achieve regret linear in the prediction horizon.

3 Data as Demonstrator

Since the multi-step prediction error increases due to a mismatch between the training and test (prediction) distributions, it would be convenient if we could interactively collect new data and aggregate it into the training dataset. This is precisely the process suggested in Dataset Aggregation (DAgger) introduced in [13]. Unfortunately, collecting new data can be very difficult in practice. First, it can be expensive or dangerous to maneuver a physical system into the required states (e.g. an inverted helicopter). Second, there is no guarantee that predicted states are feasible for the true system. (e.g. predicted pendulum's Euclidean coordinates could be off of the constraint circle). This situation is illustrated in Figure 1a.

The main insight of this work is that instead of collecting new data from the true system, we synthetically generate correction examples for the learner to use. Since the given training trajectories are time-indexed, they provide a correction for each predicted timestep when simulating from the points along the training trajectories, as depicted in Figure 1b. This idea motivates our algorithm (Alg. 1), Data as Demonstrator (DAD).

3.1 Reduction to imitation learning

Data as Demonstrator forward simulates a learned model, collecting data on the encountered prediction ('test') time distribution by simulating from the starting point of training trajectories. The next model is trained from input-target pairs created by pointing the 'test' time prediction to the correct next time-indexed state along the trajectory. By iterating and retraining a new model on the aggregate dataset, DAD can be viewed as a *Follow-The-Leader* algorithm.

Since we are applying corrections from the dataset, we can also interpret DAD as a simplified scenario of interactive imitation learning. Let the expert be the training data which "demonstrates" expert actions by specifying at each point in time the correct state for the next time step. The learned

Algorithm 1 DATA AS DEMONSTRATOR (DAD)

Input:

▷ Number of iterations N, set $\{\xi_k\}$ of K trajectories of time lengths $\{T_k\}$.
▷ No-regret learning procedure LEARN
▷ Corresponding PREDICT procedure for multi-step prediction that takes an initial state, model, and number of time steps.

Output: Model $\widehat{M}$

Initialize aggregate data set $D \leftarrow \{(x_t, x_{t+1})\}$ of $(T_k - 1)$ input-target pairs from each trajectory ξ_k
Train initial model $M_0 \leftarrow \text{LEARNER}(D)$
for $n = 1, \ldots, N$ **do**
    for $k = 1, \ldots, K$ **do**
        $(\widehat{x}_1, \ldots, \widehat{x}_T) \leftarrow \text{PREDICT}(\xi_k(0), M_n, T_k)$
        $D' \leftarrow \{(\widehat{x}_1, \xi_k(2)), \ldots, (\widehat{x}_{T_k-1}, \xi_k(T_k))\}$
        $D \leftarrow D \cup D'$
    end for
    $M_n \leftarrow \text{LEARN}(D)$
end for
return M_n with lowest error on validation trajectories

time-series model M is equivalent to the state dependent action policy $\hat{\pi}$. The state dynamics simply pass on the predictions from the learned model as the input for the next state.

This reduction to the interactive imitation learning setting allows us to avail ourselves of the theoretical guarantees for DAgger [13]. Notationally, let a ground truth trajectory from the underlying system be $\xi = \{x_0, x_1, \ldots\}$, and let $\hat{\xi} = \{x_0, \hat{x}_1, \hat{x}_2, \ldots\}$ denote the trajectory induced by starting at x_0 from the true trajectory and iteratively applying the model M as described in the two-step forward prediction procedure. Let $P_M := P_M(\hat{\xi}, \xi)$ denote the distribution of the time-synchronized pairs $(\hat{x}_t, x_t)$ from the predicted states and the true system's trajectory. Let ϵ_N be the true loss of the best model in hindsight, defined as $\epsilon_N = \min_{M \in \Gamma} \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{x \sim P_{M_i}}[\ell_M(x)]$. Finally, let $\widehat{M} = \arg\min_{M \in M_{1:N}} \mathbb{E}_{x \sim P_M}[\ell_M(x)]$ be the model returned by DAD that performed best on its own induced distribution of states.

Theorem 2. *Given a bounded single-step prediction (regression) loss ℓ and associated no-regret learning procedure* LEARN*,* DAD *has found a model $\widehat{M} \in M_{1:N}$ as $N \to \infty$, such that $\mathbb{E}_{x \sim P_{\widehat{M}}}\left[\ell_{\widehat{M}}(x)\right] \leq \epsilon_N + o(1)$.*
Specifically, DAD *returns model $\widehat{M}$ for which this bound holds.*

Proof. We setup the imitation learning framework as described earlier: learned policy $\hat{\pi} = \widehat{M}$ and degenerate state dynamics that rely on the policy (learned model) to solely transition the state. By this reduction to imitation imitation, the result follows from [13]. □

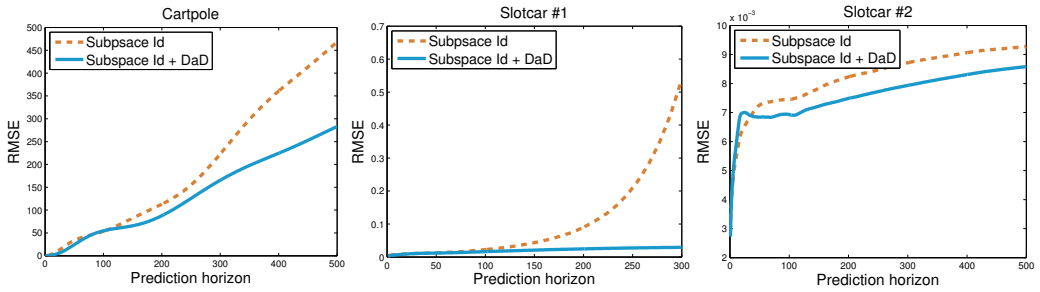
Intuitively, these results tell us that we either fail to find a model because the generation of synthetic training points creates conflicting inputs for the learner when our new data-points overlap or we guarantee good performance after a certain number of iterations.

4 Learning Linear Dynamical Systems for Accurate Multistep Prediction

For many dynamical system modeling problems, it is often sensible to assume that each observation is correlated with an underlying latent state that is evolving over time. In the case where the state, actions, and observations, are real-valued, the state update is linear, and the noise terms are assumed to be Gaussian, the resulting model is called a stochastic *linear dynamical system* (LDS). The evolution an LDS can be described by the following two equations:

$$x_{t+1} = Ax_t + Bu_t + w_t \quad w_t \sim \mathcal{N}(0, Q) \tag{5}$$

$$y_t = Cx_t + Du_t + v_t \quad v_t \sim \mathcal{N}(0, R) \tag{6}$$



(a) Cartpole Simulation Dataset (b) Slotcar Dataset # 1 (c) Slotcar Dataset # 2

Figure 2: RMSE vs Prediction horizon. We show up to a 500 step prediction horizon for the Cartpole and Slotcar #2. Additionally, we notice in Figure 2c, there are time horizons in which the baseline Subspace Id. performs better. However, in the long horizon, we achieve better performance with Subspace identification followed with DAD.

Time is indexed by the discrete variable t, x_t denotes *latent* states in $\mathbb{R}^n$, y_t the observations in $\mathbb{R}^m$, u_t the exogenous input in $\mathbb{R}^l$, and the parameters of the system are the dynamics model $A \in \mathbb{R}^{n \times n}$, the input model $B \in \mathbb{R}^{n \times l}$, the output model $C \in \mathbb{R}^{m \times n}$, and the direct-feedthrough model $D \in \mathbb{R}^{m \times l}$. The variables w_t and v_t describe zero-mean normally distributed process and observation noise respectively, with covariance matrices $Q \in \mathbb{R}^{n \times n}$ and $R \in \mathbb{R}^{m \times m}$.

Learning a linear dynamical system from data (linear system identification) involves finding the parameters $\theta = \{A, B, C, D, Q, R\}$ that explain the observed data. Linear system identification is a well-studied subject, and there are several different approaches to finding the parameters θ. A common approach is to search for the parameters that maximize the likelihood of the observed data through iterative techniques such as expectation maximization (EM). An alternative approach, popular in the controls community, is to use *subspace identification* methods to compute a statistically consistent solution in closed form [11].

The standard algorithms for linear system identification, like EM and subspace identification, only consider single-step prediction error when learning the dynamics models A. As a result, the solutions found by these methods may make very poor multistep predictions or even be *unstable* due to sampling constraints, modeling errors, and measurement noise [5]. This can cause serious problems when predicting and simulating from a learned LDS [14, 4]. To combat these problems, we apply DATA AS DEMONSTRATOR (Alg. 1) during linear system identification to improve the dynamics model A initially learned by the traditional approach of minimizing single-step squared loss on adjacent estimated states.

5 Results

We present preliminary results of DATA AS DEMONSTRATOR for the task of subspace identification on a simulated cartpole dataset and on a real dataset consisting of a tracked slot car racing around a racetrack [8]. The cartpole example was constructed using Simulink with white noise added to the control input. The cartpole's position, angle, and respective time derivatives were used as the observations. For the slotcar, we take as observations y the filtered track position parameter $\in [0, 1]$ provided by an overhead camera.

We measure performance of system identification on a filtering task interleaved with forward simulation. After initial purely filtering steps to initialize the filter to a reasonable state, we introduce a forward simulation after each subsequent predict-and-update step. The multiple-step prediction error is measured as the root mean squared error (RMSE) e_k between the prediction $\hat{\xi}$ and the ground truth trajectory ξ, computed as $e_k = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\left\|\hat{\xi}_k(t) - \xi(t)\right\|_2^2}$ for all time horizons $T \leq T_{max}$.

We show how RMSE increases versus the prediction horizon for Subspace Identification [4] compared to Subspace Identification augmented with DATA AS DEMONSTRATOR in Figure 2. We show results from two slotcar runs. For these experiments we choose to only learn improvements on the

	$T = 50$		$T = 150$		$T = 300$	
System	**SS Id.**	**+ DAD**	**SS Id.**	**+ DAD**	**SS Id.**	**+ DAD**
`Cartpole`	33.5e0	24.3e0	82.3e0	65.6e0	222.9e0	165.4e0
`Slotcar #1`	12.5e-3	12.0e-3	43.5e-5	20.8e-3	540.6e-3	29.3e-3
`Slotcar #2`	7.3e-3	6.8e-3	7.8e-3	7.2e-3	8.7e-3	7.9e-3

Table 1: RMSE at various prediction horizons for Subspace Identification method and for Subspace Identification with DATA AS DEMONSTRATOR on the Slotcar Datasets. Using DAD on these datsets improves upon the learned model from Subspace Identification for the listed prediction horizons.

learned dynamics matrix, A, keeping the input (controls) matrix B fixed from the initial learning since the control inputs u are fixed and do not have a distribution change as a result of recursive prediction during forward simulation.

We see improvement by using subspace identification in conjunction with DAD for the cart pole and both slot car trials. We also notice for Slotcar #1 (Fig. 2b), that DATA AS DEMONSTRATOR was able to find a stable model that better reflects the system that generated the data, whereas subspace identification alone found an unstable model that makes very poor long-range predictions. In this case, the baseline subspace identification's learned A has a top eigenvalue ($\max|\lambda|$) of 1.0122. Utilizing DAD, the maximum eigenvalue dropped to 0.9986. Overall, we noticed that our meta-algorithm pushed the maximum eigenvalue closer to the stability threshold of 1, which results in better accuracy during the simulations. Results using a different learning algorithm, Random Fourier Feature regression, are presented in the original DATA AS DEMONSTRATOR paper on a variety of other benchmarks, including video textures [15]. We do not reproduce them here for brevity.

6 Conclusion

DATA AS DEMONSTRATOR is a meta-algorithm for improving the multiple step prediction capability of a learner in a data-efficient fashion. Using only the training data, DAD synthesizes examples to alleviate the train-test distribution difference that hinders simulation using models trained to minimize the single-step error. In this work, we applied this algorithm to the context of subspace identification and showed promising results on a simulated cartpole and on a real world slotcar dataset. We hope to continue investigating how this algorithm can be used in other system identification scenarios.

Acknowledgments

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1252522, the DARPA Autonomous Robotic Manipulation Software Track program, and the National Science Foundation NRI *Purposeful Prediction* Project.

References

[1] Pieter Abbeel and Andrew Y Ng. Learning first-order markov models for control. In *NIPS*, pages 1–8, 2005.

[2] Arslan Basharat and M Shah. Time series prediction by chaotic modeling of nonlinear dynamical systems. *IEEE International Conference on Computer Vision*, pages 1941–1948, 2009.

[3] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166, 1994.

[4] B. Boots. Learning Stable Linear Dynamical Systems. Data Analysis Project, Carnegie Mellon University, 2009.

[5] Nelson Loong Chik Chui and Jan M Maciejowski. Realization of stable models with subspace methods. *Automatica*, 32(11):1587–1595, 1996.

[6] Seyed Mohammad Khansari-Zadeh and Aude Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *Robotics, IEEE Transactions on*, 27(5):943–957, 2011.

[7] J Ko, D J Klein, D Fox, and D Haehnel. GP-UKF: Unscented kalman filters with Gaussian process prediction and observation models. pages 1901–1907, 2007.

[8] Jonathan Ko and Dieter Fox. Learning GP-Bayes Filters via Gaussian process latent variable models. *Autonomous Robots*, 30(1):3–23, 2011.

[9] John Langford, Ruslan Salakhutdinov, and Tong Zhang. Learning nonlinear dynamic models. In *ICML*, pages 593–600. ACM, 2009.

[10] KR Müller, AJ Smola, and G Rätsch. Predicting time series with support vector machines. *Artificial Neural Networks ICANN'9*, 1327:999–1004, 1997.

[11] P. Van Overschee and B. De Moor. *Subspace Identification for Linear Systems: Theory, Implementation, Applications*. Kluwer Academic, 1996.

[12] Liva Ralaivola and Florence D'Alche-Buc. Dynamical modeling with kernels for nonlinear time series prediction. *NIPS*, 2004.

[13] Stéphane Ross, Geoffrey J Gordon, and J. Andrew Bagnell. No-regret reductions for imitation learning and structured prediction. In *AISTATS*, 2011.

[14] Sajid Siddiqi, Byron Boots, and Geoffrey J. Gordon. A constraint generation approach to learning stable linear dynamical systems. In *NIPS 20 (NIPS-07)*, 2007.

[15] Arun Venkatraman, Martial Hebert, and J. Andrew Bagnell. Improving multi-step prediction of learned time series models. In *AAAI (Awaiting Publication)*, 2015.

[16] Jack Wang, Aaron Hertzmann, and David M Blei. Gaussian process dynamical models. In *NIPS*, pages 1441–1448, 2005.

[17] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.