# Improving Multi-step Prediction of Learned Time Series Models

**Arun Venkatraman, Martial Hebert,** and **J. Andrew Bagnell**
Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213
arunvenk@cs.cmu.edu, {hebert, dbagnell}@ri.cmu.edu

## Abstract

Most typical statistical and machine learning approaches to time series modeling optimize a single-step prediction error. In multiple-step simulation, the learned model is iteratively applied, feeding through the previous output as its new input. Any such predictor however, inevitably introduces errors, and these compounding errors change the input distribution for future prediction steps, breaking the train-test i.i.d assumption common in supervised learning. We present an approach that reuses training data to make a no-regret learner robust to errors made during multi-step prediction. Our insight is to formulate the problem as imitation learning; the training data serves as a "demonstrator" by providing corrections for the errors made during multi-step prediction. By this reduction of multi-step time series prediction to imitation learning, we establish theoretically a strong performance guarantee on the relation between training error and the multi-step prediction error. We present experimental results of our method, DAD, and show significant improvement over the traditional approach in two notably different domains, dynamic system modeling and video texture prediction.

Determining models for time series data is important in applications ranging from market prediction to the simulation of chemical processes and robotic systems. Many supervised learning approaches have been proposed for this task, such as neural networks (Narendra and Parthasarathy 1990), Expectation-Maximization (Ghahramani and Roweis 1999; Coates, Abbeel, and Ng 2008), Support Vector Regression (Müller, Smola, and Rätsch 1997), Gaussian process regression (Wang, Hertzmann, and Blei 2005; Ko et al. 2007), Nadaraya-Watson kernel regression (Basharat and Shah 2009), Gaussian mixture models (Khansari-Zadeh and Billard 2011), and Kernel PCA (Ralaivola and D'Alche-Buc 2004). Common to most of these methods is that the objective being optimized is the single-step prediction loss. However, this criterion does not guarantee accurate multiple-step simulation accuracy in which the output of a prediction step is used as input for the next inference.

The prevalence of single-step modeling approaches is a result of the difficulty in directly optimizing the multiple-step prediction error. As an example, consider fitting a simple linear dynamical system model for the multi-step error over the time horizon $T$ from an initial condition $x_0$,

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

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 single-step squared loss used in supervised learning,

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

is more appealing to solve as it has an easy, closed form solution. Abbeel et al. propose a generalization of (1) coined the "lagged error" criterion which penalizes deviations during forward simulation. However, as noted by the authors, this objective is also non-linear and non-convex as the second step prediction is conditioned on the output of the first. They propose either a simpler but still non-convex approximation (Abbeel, Ganapathi, and Ng 2005) that can be iteratively optimized or Expectation Maximization (Abbeel and Ng 2005b) to find a local optimum to the multi-step error.

If the predictive model is differentiable, one can apply "backpropagation-through-time" (Werbos 1990; Langford, Salakhutdinov, and Zhang 2009). Unfortunately, such gradient methods are limited in the model classes they can consider, effectively ruling out broad classes of some of the most effective regression algorithms including decision trees and random forests. Moreover, such methods – even on simple linear predictive models – tend to suffer from a "gradient collapse" and ill-conditioning (Bengio, Simard, and Frasconi 1994), 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 as a result, many approaches in the literature focus on using single-step prediction models to take advantage of techniques developed in the statistical and machine learning communities.

In this paper, we introduce a new meta-algorithm, DATA AS DEMONSTRATOR (DAD), for improving the multi-step prediction capability of a time-series learner. Our contributions are:

– We propose a simple, easy to implement meta-algorithm that can wrap an existing time-series learning procedure.
– Our method makes no assumption on differentiability. This allows our algorithm to be utilized on top of a larger array of supervised learning algorithms.
– Our method is data-efficient in improving a learned model. Without querying the actual system for more training data, our method is able to achieve better performance on the multi-step criterion by reusing training data to correct for prediction mistakes.
– Moreover, through a reduction to imitation learning, we demonstrate that when the learner exhibits the no-regret property, we can provide performance guarantees that relate the one-step predictive error to the multi-step error.

Finally, we demonstrate experimentally that our method improves the multi-step prediction error.

## Problem Setup

We consider the problem of modeling a discrete-time time-series (system) characterized by a time-indexed state $x_t$ generated from some stationary dynamics,

$$x_{t+1} = \mathbb{E}[f(x_t)] \tag{3}$$

The problem we address is to learn a model given $K$ sample trajectories $\xi_k \in \Xi$ of $\{x_0, x_1, \ldots, x_{T_k}\}$ generated by the system (3). As motivated in the introduction, it is easiest to learn a forward prediction model by considering the prediction of single, consecutive 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 $\ell$ and class of models $\Gamma$. 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 $\hat{x}_t \leftarrow \hat{x}_{t+1}$

One may naively intuit that the error is linear in the number of predictions - the multiple-step prediction horizon - as would be expected in the supervised learning setting; however, this ignores the feedback effect from the learner's prediction error. At each execution of Step 1, prediction error results in a state that could be outside of the training data distribution in $\Xi$. In practice, this results in even more error, cascading through to later predictions.

**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 $\|\widehat{M}(\hat{x}_T) - x_{T+1}\| \leq \sum_{t=0}^{T} L^t\epsilon \in O\left(\exp(T\log(L))\epsilon\right)$. □

The bound in Theorem 1 tells us that the multi-step prediction error is bounded by an exponential in the time horizon. This bound is tight. Consider a fitted a linear model $\hat{A} > 1$ on a 1-D training trajectory $\xi$, with bounded error $\epsilon$ and Lipshitz constant $\hat{A}$. If we make a single-step error $\epsilon$ on the first step of forward simulation, $\hat{x}_1 = \hat{A}x_0 = x_1 + \epsilon$, we get: $\hat{x}_{T+1} = \hat{A}^T(x_1 + \epsilon)$ Then,

$$\|\hat{x}_{T+1} - x_{T+1}\| = \|\hat{A}^T(x_1 + \epsilon) - x_{T+1}\| \in \Omega(\hat{A}^T\epsilon) \tag{5}$$

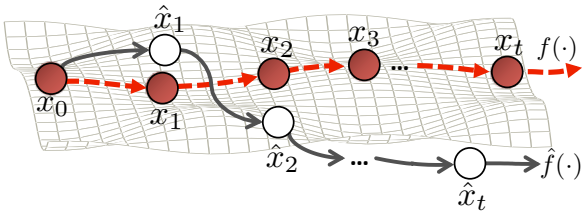
It is also worthwhile to note the Lipshitz constant conditions: For $L > 1$, we get the bound in Theorem 1. If $L = 1$, the bound reduces to $O(T\epsilon)$, which is equivalent to the result for the supervised learning setting. Finally, for $L < 1$, we get $O(L\epsilon)$. This situation specifies a stable fitted model that decays to zero change in state over the time horizon. For many time-series problems, this results in boring simulated behavior as the predictions converge to the mean, as exemplified with the "Fireplace" video texture in the experimental section. For many real-world problems, a model that is near the limit of stability is preferred. As a result, we may learn approximations that are unstable, yielding the exponential bound as shown above. In the following sections, we motivate and develop an algorithm which can achieve regret linear in the prediction horizon.

## DATA AS DEMONSTRATOR Algorithm

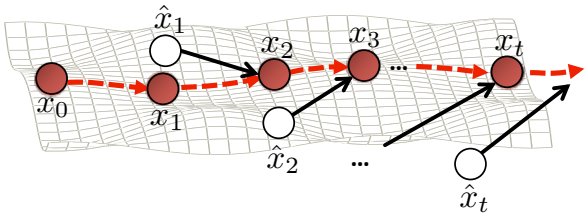
To resolve differences in the train and test (forward prediction) distributions, it would be convenient to augment our training set to meet the test distribution. This is reminiscent of recent work in interactive imitation learning in which the expert demonstrator is queried in order to learn a control policy. Though a natural approach, collecting new data on the induced 'test' distribution may be impossible.

Practically, it could be expensive to query the true system at the incorrectly predicted state (e.g. an inverted helicopter). The more impeding concern, however, is that the predicted states may not be feasible for the generating system. Consider data collected from a pendulum mass on a string with radius 1. The mechanics of the system require the Euclidean coordinates to reside on the unit circle. During forward simulation with an approximate model, we may predict points off this constraint manifold. This situation is illustrated in Figure 1(a).

Instead, we synthetically generate correction examples for the learner to use. Since the given training trajectories are time-indexed, they can provide a correction for each time step when simulating from points along the training trajectories, as depicted in Figure 1(b). This idea motivates our algorithm, DATA AS DEMONSTRATOR (DAD).




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




(b) Data provides a 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 a low-dimensional manifold of all possible states. Cascading prediction errors from forward simulation with a learned model will often result in predicted infeasible states **(Fig. 1(a))**. Our algorithm, DATA AS DEMONSTRATOR (DAD), generates synthetic examples for the learner to ensure that prediction returns back to typical states **(Fig. 1(b))**.

## Reduction to imitation learning

DATA AS DEMONSTRATOR forward simulates a learned model, collecting data on the encountered prediction, 'test' time, distribution by simulating from points along 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 considered 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 time-series model $M$ acts as 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 the Dataset Aggregation algorithm (DAGGER) introduced in (Ross, Gordon, and Bagnell 2011). 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.

**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 $\xi_k$
Train initial model $M_0 \leftarrow$ LEARNER($D$)
**for** $n = 1, \ldots, N$ **do**
  **for** $k = 1, \ldots, K$ **do**
    $(\widehat{x}_1, \ldots, \widehat{x}_T) \leftarrow$ 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**
  $M_n \leftarrow$ LEARN($D$)
**end for**
**return** $M_n$ with lowest error on validation trajectories

Let $\epsilon_N$ 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. We are then able to achieve the following performance guarantee:

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

*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 learning, the result follows from Theorem 4.1 of (Ross, Gordon, and Bagnell 2011). □

We can also relate the number of iterations of DAD to get a linear performance guarantee with respect to the prediction horizon for the regularized squared loss, a commonly used regression loss. Letting $J(M) = \sum_{t=0}^{T} \ell[(\hat{\xi}(t), \xi(t))]$ define the multiple-step prediction error, we get:

**Theorem 3.** *If $\ell$ is the regularized squared loss over a linear predictor (or kernel linear regressor) and if $N$ is $\tilde{O}(T)$, then $\exists\, \widehat{M} \in M_{1:N}$ found by* DAD *such that $J(\widehat{M}) \leq O(T \cdot \epsilon_N)$*

*Proof.* The regularized squared loss on a linear prediction model (or on a kernel linear regressor) is strongly convex in the model parameters. The result then follows from our reduction and Theorem 3.2 of (Ross, Gordon, and Bagnell 2011). □

Intuitively, these results tell us that for a given time-series modeling problem, 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. Additionally, the performance guarantees given by Theorem 2 and Theorem 3 have some subtleties and depend on a few critical assumptions. Firstly, DAD gives no guarantee on any specfic learned model $M \in M_{1:N}$ from each iteration but only that there exists a generated model that performs well. In addition, since the efficacy of the learner shows up in the bound, it is necessary to have a learning procedure LEARN$(D)$ that is capable of achieving low training error on the single-step loss. Furthermore, the bounds we have shown are valid with an infinite number of completely new trajectories at each step $n$, a similar requirement for related methods such as SEARN (Daumé III, Langford, and Marcu 2009), Forward training (Ross and Bagnell 2010), and DAGGER (Ross, Gordon, and Bagnell 2011). Following the practice of the aforementioned methods, due to limited data and for data efficiency, the iterations share a single batch of training trajectories.

There are a large class of learning algorithms that meet the no-regret requirement for the internal learning procedure in DAD. A no-regret learning algorithm is one that produces a sequence of models $M_n$, such that the average regret

$$R_{\text{LEARNER}} = \frac{1}{N}\sum_n \ell_{M_n}(x_n) - \min_{M\in\Gamma}\frac{1}{N}\sum_n \ell_M(x_n) \quad (6)$$

tends to 0 as $N \to \infty$. Since DAD is a *Follow-The-Leader* algorithm, any strongly convex loss (e.g. regularized squared loss) on the aggregated dataset will result in a no-regret learner. In fact, stability and asymptotic consistency are enough to ensure no-regret (Ross and Bagnell 2011). Additionally, we can even utilize other online learning methods to achieve the same performance bounds since many are also no-regret (Cesa-Bianchi, Conconi, and Gentile 2004). The advantage of online methods is that we no longer have to store the ever-growing dataset but simply update the learned model for every new data point.

Finally, we would like to note a few additional details on using the DATA AS DEMONSTRATOR algorithm. Even though Algo. 1 starts the predictions at $\xi_k(0)$, better utilization of the dataset can be achieved by starting at other points in the training trajectories. Also, the aggregation phase may require a thresholding or filtering step. For longer horizon problems and with weaker learners, it may be difficult to achieve improvement by giving equal importance to the larger corrections at the end of the prediction horizon and to the small errors made at the beginning. Intuitively, we hope that by the end, the learned model should not wander far away from the ground truth trajectory, and thus we ignore those points during the training iterations.

## Experimental Evaluation

To demonstrate the efficacy of the DATA AS DEMONSTRATOR algorithm, we consider two markedly different, challenging time series prediction problems. Since our approach is a meta-algorithm, it requires an underlying no-regret learning procedure to optimize the single-step prediction error. Below, we use a simple but expressive learning procedure, but we wish to emphasize that it could be replaced with other domain specific learning methods.

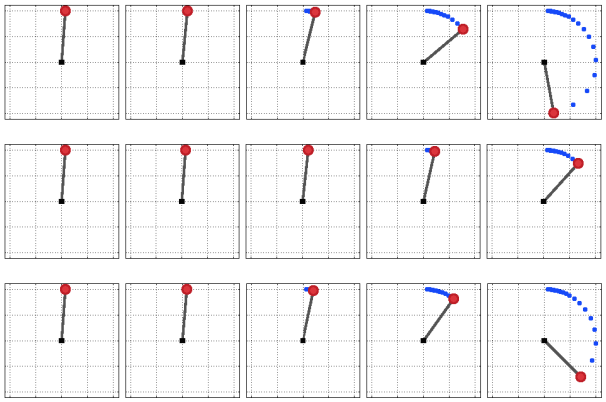


Figure 2: A pendulum trajectory's ground truth (top) compared with multi-step predictions displayed for time steps 1, 8, 16, 23, and 30 from the traditional minimization of single-step error (middle) and from our approach (bottom). The final output of DAD is closer to the true final state.

### Single-step learning procedure: Random Fourier Feature Regression

Motivated by prior work in kernelized modeling of dynamical systems (Ralaivola and D'Alche-Buc 2004; Chaudhry et al. 2009; Wingate and Singh 2006; Kawahara, Yairi, and Machida 2006), we use kernel regression, a nonparametric learning procedure that allows us to embed our data points into a high, possibly infinite, dimensional space. Let $\phi(x)$ define the feature map that induces the kernel $k(x,y) = \langle\phi(\cdot),\phi(\cdot)\rangle$. We minimize the $\lambda$-regularized squared loss:

$$\min_C \frac{1}{N}\cdot\frac{1}{2}\left\|C\Phi_t - X_{t+1}\right\|_F^2 + \lambda\frac{1}{2}\left\|C\right\|_F^2 \quad (7)$$

for the forward prediction model:

$$\hat{x}_{t+1} = C\phi(\hat{x}_t) \quad (8)$$

However, for some choice of kernels, such as the Gaussian kernel, the solution to (7) cannot be evaluated as $\phi$ embeds the original data points into an infinite dimensional space. A standard approach would be to solve the objective in the dual. Since DAD is a dataset aggregation algorithm that augments the dataset with $O(N)$ new input-target pairs every iteration, this procedure quickly becomes intractable.

Recent work in kernel machines enables us to instead approximate $\phi$ through generating random Fourier features (Rahimi and Recht 2007; 2008; Le, Sarlós, and Smola 2013; Lázaro-Gredilla 2010). For a shift-invariant kernel function $k$, the feature map $\phi$ can be approximated by constructing a mapping

$$\hat{\phi}_i(x) = \begin{bmatrix}\cos(\omega_i^T x)\\ \sin(\omega_i^T x)\end{bmatrix},\ \omega_i \in \mathbb{R}^d \sim \mathcal{F}(k(\cdot,\cdot)) \quad (9)$$

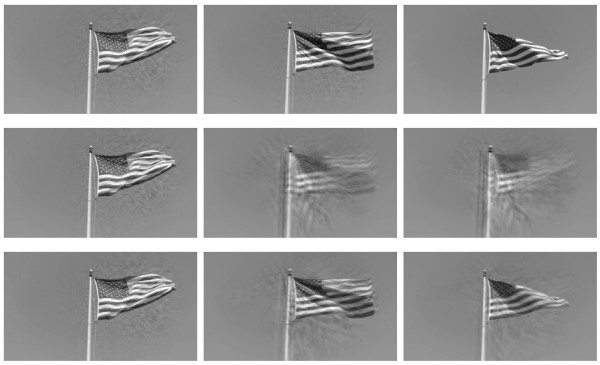


Figure 3: Ground truth (top) compared with predictions using the single-step optimized learner (middle) and with our approach (bottom) for frames 1, 9, and 16 on a trajectory of the Flag video texture. DAD produces results closer to the ground truth.

| System | SS | DAD | Relative Improvement |
|---|---|---|---|
| Pendulum | 6.00% | 3.32% | 44.7% |
| Cartpole | 19.7% | 8.46% | 57.1% |
| Helicopter | 44.2% | 25.3% | 42.8% |

Table 1: Using DATA AS DEMONSTRATOR (DAD) to wrap around the traditional Single-Step learner greatly improves performance on dynamical system modeling.

where $\mathcal{F}(k(\cdot,\cdot))$ is the Fourier transform of the kernel function. This approximates the kernal function in expectation, i.e. $\mathbb{E}[\langle\phi_i,\phi_i\rangle] = k(\cdot,\cdot)$. By sampling $m$ such $\omega_i$ and concatenating to create feature vector $\hat{\phi}(x)$, we can get a lower variance approximation of the kernel function. Using $\hat{\phi}$ in (7), allows us to retrieve the model parameter $C$. In the experiments described below, we choose $m = 500$ and use the Gaussian kernel. The hyperparameters $\lambda$ and kernel bandwidth $\sigma$ were chosen using cross-validated grid search.

## Performance Evaluation

For each test bench, we evaluate the performance of the baseline single-step optimized predictor as well as DAD on the task of multiple-step prediction. The multiple-step prediction error is measured as the RMS error $e_k$ between the prediction $\hat{\xi}_k$ and the ground truth trajectory $\xi_k$, computed as $e_k = \sqrt{\frac{1}{T_k}\sum_{t=1}^{T_k}\left\|\hat{\xi}_k(t) - \xi_k(t)\right\|_2^2}$. For each experiment, a random 10% of the trajectories were used as hold out data for performance reporting. In Tables 1 and 2, we report the mean normalized RMSE by normalizing against the signal power for each trajectory, $p_k = \sqrt{\frac{1}{T_k}\sum_{t=1}^{T_k}\|\xi_k(t)\|_2^2}$. This normalization allows us to better ascertain the magnitude of error compared to the original trajectory.

## Dynamical Systems

We examine the performance of the proposed method on simulation test benches of physical dynamical systems of varying modeling difficulty. The pendulum and cart pole datasets are constructed from their respective uncontrolled dynamics and consist of 1000 trajectories of 30 time steps each. Both of these systems exhibit interesting limit-cycle behaviors. The final dynamical system we test on is the simulated helicopter from (Abbeel and Ng 2005a). This dynamical system operates in a larger, 21-dimensional state space and is governed by more complicated dynamics equations. In addition, the helicopter simulation is controlled by a closed-loop LQR controller trying to hover the helicopter around the origin from randomly chosen starting points in the basin of attraction. The LQR controller chooses actions based on state and thus it poses a challenge for the learner to extract this implicit relationship from data of the system.

Qualitatively, we can see the effect of improving a single-step prediction model in Figure 2 for the pendulum system. Numerical results are shown in Table 1. For all three of the examples, including the difficult controlled helicopter simulation, DAD is able to achieve over 40% relative improvement over the traditional baseline approach.

| Video | SS | DAD | Relative Improvement |
|---|---|---|---|
| Flag | 80.7% | 67.2% | 16.7% |
| Fireplace | 92.8% | 81.4% | 12.3% |
| Beach | 48.0% | 38.4% | 20.0% |
| Pumpjack | 18.6% | 15.4% | 17.0% |
| Windfarm | 39.8% | 38.8% | 2.57% |

Table 2: DATA AS DEMONSTRATOR (DAD) significantly improves the performance of the traditional Single-Step method on the more complex video textures and marginal improvement on the simpler, periodic 'Windfarm' texture.

## Video Textures

Dynamic video textures are image sequences generated from some underlying structure such as a flag waving, waves lapping, or a fire burning. Such systems have been studied in the literature as time-series modeling or system identification problems (Siddiqi, Boots, and Gordon 2007; Chan and Vasconcelos 2007; Basharat and Shah 2009). Video textures are inherently complicated to simulate due to being visual observations of complex underlying dynamics. In order to test DAD, we require many training trajectories, which is not present in standard video texture datasets. We instead use online videos that range from minutes to a half-hour at 24-30 fps. To make the learning faster and tractable, we use PCA to lower the dimensionality, sample every fifth frame, and construct trajectories with 16-26 time steps.

For many of the video texture test benches, the baseline of single-shot, single-step regression loss optimization was significantly improved upon by DAD. This is especially true for the harder and less repetitive video textures such as 'Flag' (Figure 3) and 'Beach'. These video textures are influenced by complex real-world dynamics. The simpler ones, such as 'Pumpjack' and 'Windfarm' are more cyclic, making the test and train trajectories similar. We see minor absolute improvement in both, but meaningful relative improvement for the pumpjack. This small change in absolute improvement implies that the single-step learner was able to capture the

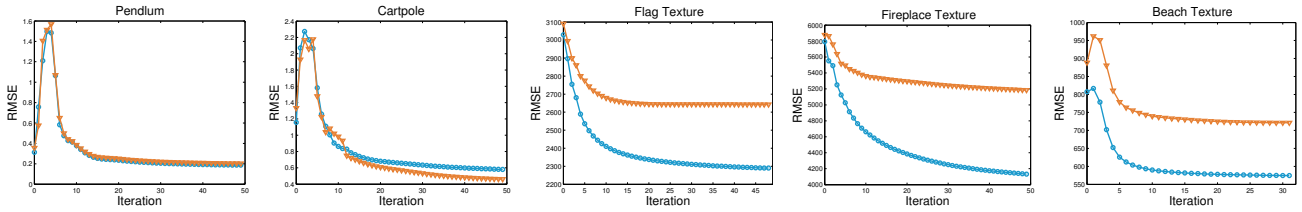



Figure 4: RMSE of multiple-step predictions for train (blue, circles) and test (orange, triangles) trajectories. Note that the shown test RMSE was not used by DAD during training and was computed for visualization only.

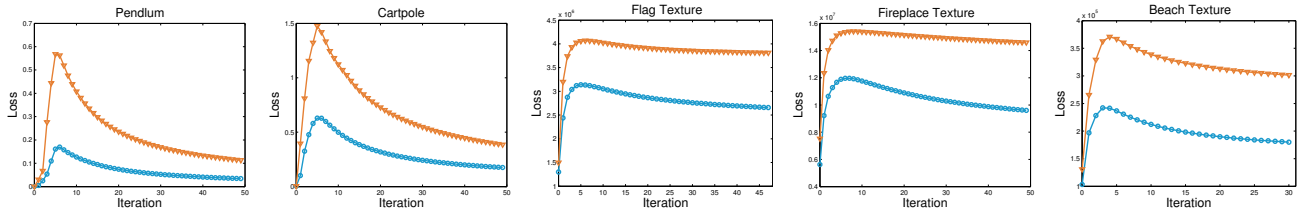


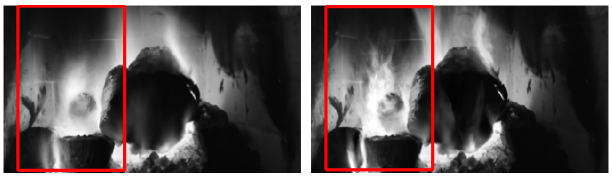

Figure 5: Loss on single-step predictions on the aggregated training (blue, circles) and test (orange, triangles) data set. Notice that the error increases and then stabilizes over iterations as distribution of states induced by learner's multiple-step prediction stabilizes. Note that the aggregated test set is never used during training and is shown for comparative purposes only.

primary evolution, but even with DAD, it is unable to capture the remaining detail in the system (e.g. clouds in the background of the 'Windfarm').

A common failing of dynamical system modeling methods is the degeneration of the forward simulation to the mean value in the dataset. In the top frames of Figure 6(a), we see the system converge by halfway through the prediction horizon to the mean. DAD provides corrections that can mitigate this tendency (bottom frames in Figure 6(a)). This results in a qualitatively crisper and more believable video, as shown in the close up (Figure 6(b)).

Finally, it is interesting to observe the multi-step error and the single-step prediction loss over iterations of DAD. As mentioned previously, the multi-step error does not necessarily decrease with each iteration (e.g. second from left in Figure 4). Additionally in Figure 5, we notice that the single-step regression loss on the aggregate data set initially increases as the learning problem is made harder through the addition of more data. As the multi-step prediction distribution converges, the single-step loss also stabilizes.

(a) Prediction of frames 14, 19, 26 for the baseline method (top) and DAD (bottom). Note the averaging effect with the single-step predictor with predictions converging on the mean from the dataset. DAD yields crisper predicted images.



(b) Comparison of final predicted frames

Figure 6: In the Fireplace video texture, our method produces a more believable evolution.

## Conclusion

We presented DAD, a data-efficient, simple to implement meta-algorithm for improving the multiple-step prediction capability of a learner for modeling time-series data. Through a reduction to imitation learning, we establish strong theoretical performance guarantees. On a challenging set of experiments, we show significant performance gains over the traditional, baseline approach of minimizing the single-step prediction error. It may be interesting to study how DATA AS DEMONSTRATOR could be adapted for other system identification techniques, and we have already seen promising results in preliminary investigations with linear regression for subspace identification on a slotcar dataset.

## Acknowledgements

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1252522 and in part by the DARPA Autonomous Robotic Manipulation Software Track program. The authors also thank Jeff Schneider and Byron Boots for valuable and insightful discussions.

## References


Abbeel, P., and Ng, A. Y. 2005a. Exploration and apprenticeship learning in reinforcement learning. In *ICML*, 1–8. ACM.

Abbeel, P., and Ng, A. Y. 2005b. Learning first-order markov models for control. In *NIPS*, 1–8.

Abbeel, P.; Ganapathi, V.; and Ng, A. Y. 2005. Learning vehicular dynamics, with application to modeling helicopters. In *NIPS*, 1–8.

Basharat, A., and Shah, M. 2009. Time series prediction by chaotic modeling of nonlinear dynamical systems. 1941–1948.

Bengio, Y.; Simard, P.; and Frasconi, P. 1994. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on* 5(2):157–166.

Cesa-Bianchi, N.; Conconi, A.; and Gentile, C. 2004. On the generalization ability of on-line learning algorithms. *Information Theory, IEEE Transactions on* 50(9):2050–2057.

Chan, A. B., and Vasconcelos, N. 2007. Classifying Video with Kernel Dynamic Textures. *IEEE Conference on Computer Vision and Pattern Recognition* 1–6.

Chaudhry, R.; Ravichandran, A.; Hager, G.; and Vidal, R. 2009. Histograms of oriented optical flow and Binet-Cauchy kernels on nonlinear dynamical systems for the recognition of human actions. *IEEE Conference on Computer Vision and Pattern Recognition* 1932–1939.

Coates, A.; Abbeel, P.; and Ng, A. Y. 2008. Learning for control from multiple demonstrations. In *ICML*, 144–151. New York, NY, USA: ACM.

Daumé III, H.; Langford, J.; and Marcu, D. 2009. Search-based structured prediction. *Machine learning* 75(3):297–325.

Ghahramani, Z., and Roweis, S. T. 1999. Learning nonlinear dynamical systems using an EM algorithm. 431—-437.

Kawahara, Y.; Yairi, T.; and Machida, K. 2006. A Kernel Subspace Method by Stochastic Realization for Learning Nonlinear Dynamical Systems. *NIPS* 665–672.

Khansari-Zadeh, S. M., and Billard, A. 2011. Learning stable nonlinear dynamical systems with gaussian mixture models. *Robotics, IEEE Transactions on* 27(5):943–957.

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

Langford, J.; Salakhutdinov, R.; and Zhang, T. 2009. Learning nonlinear dynamic models. In *ICML*, 593–600. ACM.

Lázaro-Gredilla, M. 2010. Sparse spectrum Gaussian process regression. *The Journal of Machine Learning Research* 11:1865–1881.

Le, Q.; Sarlós, T.; and Smola, A. 2013. FastfoodApproximating Kernel Expansions in Loglinear Time. *ICML* 244–252.

Müller, K.; Smola, A.; and Rätsch, G. 1997. Predicting time series with support vector machines. *Artificial Neural Networks ICANN'9* 1327:999–1004.

Narendra, K. S., and Parthasarathy, K. 1990. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks* 4–27.

Rahimi, A., and Recht, B. 2007. Random features for large-scale kernel machines. *NIPS* 1–8.

Rahimi, A., and Recht, B. 2008. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. *NIPS* 1–8.

Ralaivola, L., and D'Alche-Buc, F. 2004. Dynamical modeling with kernels for nonlinear time series prediction. *NIPS*.

Ross, S., and Bagnell, J. A. 2010. Efficient reductions for imitation learning. In *International Conference on Artificial Intelligence and Statistics*, 661–668.

Ross, S., and Bagnell, J. A. 2011. Stability conditions for online learnability. *arXiv preprint arXiv:1108.3154*.

Ross, S.; Gordon, G. J.; and Bagnell, J. A. 2011. No-regret reductions for imitation learning and structured prediction. In *International Conference on Artificial Intelligence and Statistics*. Citeseer.

Siddiqi, S.; Boots, B.; and Gordon, G. J. 2007. A constraint generation approach to learning stable linear dynamical systems. In *NIPS 20 (NIPS-07)*.

Wang, J.; Hertzmann, A.; and Blei, D. M. 2005. Gaussian process dynamical models. In *NIPS*, 1441–1448.

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

Wingate, D., and Singh, S. 2006. Kernel predictive linear Gaussian models for nonlinear stochastic dynamical systems. *ICML* 1017–1024.