output-exponent-marker=e

Extended Dynamic Mode Decomposition with Jacobian Residual Penalization for Learning Bilinear, Control-affine Koopman Models

Brian E. Jackson
Robotics Institute
Carnegie Mellon University
brianjackson@cmu.edu

Jeong Hun Lee
Robotics Institute
Carnegie Mellon University
jeonghunlee@cmu.edu

Kevin Tracy
Robotics Institute
Carnegie Mellon University
ktracy@cmu.edu

Zachary Manchester
Robotics Institute
Carnegie Mellon University
zacm@cmu.edu

Abstract: Data-driven, Koopman-based learning methods offer a practical and more efficient technique to control nonlinear dynamical systems by lifting the dynamics into a linear space of observable functions. However, many Koopman learning frameworks specifically learn *lifted linear* models, which assume linearity with respect to the controls, without also incorporating any *derivative* information of the system. This paper presents an extension of the commonly-used Extended Dynamic Mode Decomposition (EDMD) method for learning a *lifted bilinear*, control-affine dynamical system while also penalizing residuals of the *jacobians* in addition to the states. We show that this regularization technique can be used to incorporate prior model knowledge in the EDMD process to reduce overfitting, resulting in more accurate learned, dynamics models for control, where the learned jacobians can be exploited in the feedback policy. This benefit of penalizing the jacobian residual is highlighted in both a simulated and empirical quadrotor example, for which LQR and MPC can successfully stabilize about a fixed point and track a trajectory respectively.

Keywords: Koopman, Learning, Dynamics

1 Introduction

Controlling complex, underactuated, and highly nonlinear autonomous systems remains an active area of research in robotics, despite decades of previous work exploring effective algorithms and the development of substantial theoretical analysis. Classical approaches typically rely on local linear approximations of the nonlinear system, which are then used in any of a multitude of linear control techniques, such as PID, pole placement, Bode analysis, H-infinity, LQR, or linear MPC. These approaches only work well if the states of the system always remain close to the linearization point or reference trajectory. The region for which these linearizations remain valid can be extremely small for highly nonlinear systems. Alternatively, model-based methods for nonlinear control based on optimal control have shown great success, as long as the model is well known and an accurate estimate of the global state can be provided. These model-based techniques leverage centuries of insight into dynamical systems and have demonstrated incredible performance on extremely complicated autonomous systems TODO: add citations for some recent work. On the other hand, data-driven techniques such as reinforcement learning have received tremendous attention over the last decade and have begun to demonstrate impressive performance and robustness for complicated robotic systems in unstructured environments TODO: cite a few RL results, including Marco Hutter's recent stuff. While these approaches are attractive since they require little to no previous knowledge about the system, they often require large amounts of data and fail to generalize outside of the domain or task on which they were "trained."

In this work we propose a novel method that combines the benefits of model-based and data-driven methods, based on recent work applying Koopman Operator Theory to controlled dynamical systems TODO: cite a few Koopman papers, including the CalTech ones. By leveraging data collected from an unknown dynamical system along with derivative information from an approximate analytical model, we can efficiently learn a bilinear representation of the system dynamics that performs well when used in traditional model-based control techniques such as linear MPC. The result is nonlinear dynamical system that is expressive enough to capture the full nonlinear dynamics across a broad range of the state space, but has a well-defined structure that is very amenable to specialized optimization-based controllers. By leveraging information from an analytical model, we can dramatically reduce the number of samples required to learn a good approximation of the true nonlinear dynamics.

TODO: bullet out contributions.

TODO: add summary of paper layout.

2 Background and Related Work

2.1 The Koopman Operator

The theoretical underpinnings of the Koopman operator and its application to dynamical systems has been extensively studied, especially within the last decade TODO: cite some important theoretical works on Koopman theory here. Rather than describe the theory in detail, we highlight the key concepts employed by the current work, and defer the motivated reader to the existing literature on Koopman theory.

We start by assuming we have some discrete approximation a of controlled nonlinear, time-dynamical system whose underlying continuous dynamics are Lipschitz continuous:

$$x_{k+1} = f(x_k, u_k) \tag{1}$$

where $x \in \mathcal{X} \subseteq \mathbb{R}^{N_x}$ is the state vector and $u_k \in \mathbb{R}^{N_u}$ is the control vector. This discrete approximation can be obtained for any continuous dynamical system in many ways, including implicit and explicit Runge-Kutta methods, or by solving the Discrete Euler-Lagrange equations TODO: add citations.

The key idea behind the Koopman operator is that the nonlinear finite-dimensional discrete dynamics (**??**) can be represented by an infinite-dimensional *bilinear* system:

$$y_{k+1} = Ay_k + Bu_k + \sum_{i=1}^{m} u_{k,i} C_i y_k = g(y, u) \tag{2}$$

where $y = \phi(x)$ is a mapping from the finite-dimensional state space $\mathcal{X}$ to the (possibly) infinite-dimensional Hilbert space of *observables* $y \in \mathcal{Y}$. We also assume the inverse map is approximately linear: $x = Gy$. In practice, we approximate (**??**) by choosing ϕ to be some arbitrary finite set of nonlinear functions of the state variables, which in general include the states themselves such that the linear mapping $G \in \mathbb{R}^{N_x \times N_y}$ is exact. Intuitively, ϕ "lifts" our states into a higher dimensional space where the dynamics are approximately (bi)linear, effectively trading dimensionality for (bi)linearity. This idea should be both unsurprising and familiar to most roboticsts, since similar techniques have already been employed in other forms, such as maximal-coordinate representations of rigid body dynamics TODO: add citations, the "kernel trick" for state-vector machines TODO: add citation, or the observation that solving large, sparse nonlinear optimization problems is often more effective than solving small, tightly-coupled dense problems TODO: add citations from the trajectory optimization literature.

The lifted bilinear system (**??**) can be easily learned from samples of the system dynamics (x_i^+, x_i, u_i) (where x^+ is the state at the next time step) using extended Dynamic Mode Decomposition (eDMD), which is just the application of linear least squares (LLS) to the lifted states. Details of this method will be covered later when we introduce our adaptation of eDMD and present an effective numerical technique for solving the resulting LLS problems.

2.2 Model-Predictive Control

TODO: add background on MPC?

3 EDMD with Jacobian Residual-Penalization

Existing Koopman-based approaches to learning dynamical systems only rely on samples of the unknown dynamics. Here we present a novel method for incorporating prior knowledge about the dynamics by adding derivative information of an approximate model into the data set to be learned via eDMD.

Given P samples of the dynamics (x_i^+, x_i, u_i), and an approximate discrete dynamics model

$$x^+ = \tilde{f}(x, u) \tag{3}$$

we can evaluate the Jacobians of our approximate model $\hat{f}$ at each of the sample points: $\tilde{A}_i = \frac{\partial \tilde{f}}{\partial x}, \tilde{B}_i = \frac{\partial \tilde{f}}{\partial x}$. After choosing nonlinear mapping $\phi : \mathbb{R}^{N_x} \mapsto \mathbb{R}^{N_y}$ We then want to find a bilinear dynamics model (**??**) that matches the Jacobians of our approximate model, while also matching our dynamics samples. If we define $\hat{A}_j \in \mathbb{R}^{N_x \times N_x}$ and $\hat{B}_j \in \mathbb{R}^{N_x \times N_u}$ to be the Jacobians of our bilinear dynamics model, projected back into the original state space, our objective is to find the matrices $A \in \mathbb{R}^{N_y \times N_y}, B \in \mathbb{R}^{N_y \times N_u}$, and $C_{1:m} \in \mathbb{R}^{N_u} \times \mathbb{R}^{N_y \times N_y}$ that minimize the following objective:

$$(1-\alpha) \sum_{j=1}^{P} \left\| \hat{y}_j - y_j^+ \right\|_2^2 + \alpha \sum_{j=1}^{P} \left\| \hat{A}_j - \tilde{A}_j \right\|_2^2 + \left\| \hat{B}_j - \tilde{B}_j \right\|_2^2 \tag{4}$$

where $\hat{y}_j^+ = g\left(\phi(x_j), u_j\right)$ is the output of our bilinear dynamics model, and $y_j^+ = \phi(y_j^+)$ is the actual lifted state (i.e. observables) at the next time step.

While not immediately apparent, we can minimize (**??**) using linear least-squares, using techniques similar to those used previously in the literature TODO: cite CalTech papers.

To start, we combine all the data we're trying to learn in a single matrix:

$$E = \begin{bmatrix} A & B & C_1 & \dots & C_m \end{bmatrix} \in \mathbb{R}^{N_y \times N_z} \tag{5}$$

where $N_z = N_y + N_u + N_y \cdot N_u$. We now rewrite the terms in (**??**) in terms of E. By defining the vector

$$z = \begin{bmatrix} y^T & u^T & u_1 y^T & \dots & u_m y^T \end{bmatrix} \in \mathbb{R}^{N_z} \tag{6}$$

we can write down the output of our bilinear dynamics (**??**) as

$$\hat{y}^+ = Ez. \tag{7}$$

The projected Jacobians of our bilinear model, $\hat{A}$ and $\hat{B}$, are simply the Jacobians of the bilinear dynamics in terms of the original state. We obtain these dynamics by "lifting" the state via ϕ and then projecting back onto the original states using G:

$$x^+ = G\left(A\phi(x) + Bu + \sum_{i=1}^{m} u_i C_i \phi(x)\right) = \hat{f}(x, u) \tag{8}$$

Differentiating these dynamics give us our projected Jacobians:

$$\hat{A}_j = G\frac{\partial \hat{f}}{\partial x}(x_j, u_j) = G\left(A + \sum_{i=1}^{m} u_{j,i} C_i\right)\Phi(x_j) = GE\bar{A}(x_j, u_j) = GE\bar{A}_j \tag{9a}$$

$$\hat{B}_i = G\frac{\partial \hat{f}}{\partial u}(x_j, u_j) = G\Big(B + [C_1 x_j \quad \dots \quad C_m x_j]\Big) = GE\bar{B}(x_j, u_j) = GE\bar{B}_j \tag{9b}$$

where $\Phi(x) = \partial\phi/\partial x$ is the Jacobian of the nonlinear map ϕ, and

$$\bar{A}(x, u) = \begin{bmatrix} I \\ 0 \\ u_1 I \\ u_2 I \\ \vdots \\ u_m I \end{bmatrix} \in \mathbb{R}^{N_z \times N_x}, \quad \bar{B}(x, u) = \begin{bmatrix} 0 \\ I \\ [x\ 0\ ...\ 0] \\ [0\ x\ ...\ 0] \\ \vdots \\ [0\ 0\ ...\ x] \end{bmatrix} \in \mathbb{R}^{N_z \times N_u}. \tag{10}$$

Substituting (**??**) and (**??**) into (**??**), we can rewrite our LLS problem as:

$$\underset{E}{\text{minimize}} \quad \sum_{j=0}^{P} (1-\alpha)\big\|Ez_j - y_j^+\big\|_2^2 + \alpha\Big\|GE\bar{A}_j - \tilde{A}_j\Big\|_2^2 + \alpha\Big\|GE\bar{B}_j - \tilde{B}_j\Big\|_2^2 \tag{11}$$

which is equivalent to

$$\underset{E}{\text{minimize}} \quad (1-\alpha)\big\|E\mathbf{Z_{1:P}} - \mathbf{Y^+_{1:P}}\big\|_2^2 + \alpha\Big\|GE\bar{\mathbf{A}}_\mathbf{1:P} - \tilde{\mathbf{A}}_\mathbf{1:P}\Big\|_2^2 + \alpha\Big\|GE\bar{\mathbf{B}}_\mathbf{1:P} - \tilde{\mathbf{B}}_\mathbf{1:P}\Big\|_2^2 \tag{12}$$

where $\mathbf{Z_{1:P}} \in \mathbb{R}^{N_z \times P} = [z_1\ z_2\ ...\ z_P]$ horizontally concatenates all of the samples (equivalent definition for $\mathbf{Y^+_{1:P}} \in \mathbb{R}^{N_y \times P}$, $\bar{\mathbf{A}}_\mathbf{1:P} \in \mathbb{R}^{N_z \times N_x \cdot P}$, $\tilde{\mathbf{A}}_\mathbf{1:P} \in \mathbb{R}^{N_z \times N_x \cdot P}$, $\bar{\mathbf{B}}_\mathbf{1:P} \in \mathbb{R}^{N_z \times N_u \cdot P}$, and $\tilde{\mathbf{B}}_\mathbf{1:P} \in \mathbb{R}^{N_z \times N_u \cdot P}$).

We can rewrite (**??**) in standard form using the "vec trick"

$$\text{vec}(AXB) = (B^T \otimes A)\text{vec}(X) \tag{13}$$

where vec(A) stacks the columns of A into a single vector.

Setting E in (**??**) equal to X in (**??**), we get

$$\underset{E}{\text{minimize}} \quad \left\| \begin{bmatrix} (\mathbf{Z_{1:P}})^T \otimes I_{N_y} \\ (\bar{\mathbf{A}}_\mathbf{1:P})^T \otimes G \\ (\bar{\mathbf{G}}_\mathbf{1:P})^T \otimes G \end{bmatrix} \text{vec}(E) + \begin{bmatrix} \text{vec}(\mathbf{Y^+_{1:P}}) \\ \text{vec}(\tilde{\mathbf{A}}_\mathbf{1:P}) \\ \text{vec}(\tilde{\mathbf{G}}_\mathbf{1:P}) \end{bmatrix} \right\|_2^2 \tag{14}$$

such that the matrix of cofficients has $(N_y + N_x^2 + N_x \cdot N_u) \cdot P$ rows and $N_y \cdot N_z$ columns.

3.1 Efficient Recursive Least Squares