SpReME: Sparse Regression for Multi-Environment Dynamic Systems

Anonymous submission

Abstract

Learning dynamical systems provides a new opportunity to tackle a central challenge in science and engineering. Model-based approaches show promising results with data samples captured from a single environment. On the other hand, pure data-driven approaches provide plausible results in extracting governing dynamics from multiple environments. The data-driven approach, however, raises a concern about the forecasting of dynamics as recent studies show the limitations of neural networks on extrapolation. In this work, we propose sparse regression for multi-environment (SpReME) that can leverage the best of both model-based and data-driven approaches to extract the governing dynamics from multiple environments. We cast the dynamic discovery as a sparse regression problem over multiple environments. The bases of the regression model can be curated with incomplete prior knowledge. We demonstrate our framework on four different dynamic systems ranging from simple linear to complex chaotic systems. The experimental results show that the extracted knowledge from multiple environments can be generalized to predict the dynamics of an unseen environment.

Introduction

Learning dynamic systems requires to understand underlying governing equations that are often difficult to identify. For example, uncovering the underlying form of the dynamics of flow passing a cylinder took nearly three decades for experts in fluid mechanics (Noack et al. 2003). Based on the recent progress of deep learning, data-driven approaches for learning dynamical systems have recently gained increasing attention across the physics and engineering communities. The deep learning approaches provide alternative ways to the cases where a traditional approach often fails to model. Especially, the previous studies show promising results in the cases where the physical model is unknown (Shi et al. 2015; Wang et al. 2018), underlying dynamics are incomplete (Yin et al. 2021b), and the presence of perturbation from unknown external sources (Li, Ratliff, and Açikmese 2021).

Many current approaches focus on learning a dynamical system from a single environment (Long et al. 2018). The approaches with partially known physical models especially show remarkable results in predicting future dynamics (Yin et al. 2021b). However, learning from a single environment raises the question of whether the learned dynamics can be generalized to the unseen environment. For example, suppose the training dynamics contain additional perturbation from external sources irrelevant to the governing dynamics. In that case, the learned model is unlikely to generalize to the new environment in which we want to predict future dynamics.

To tackle the limitation of the single environment, some recent works study the learning from multiple environments to capture the commonality across the different environments (Yin et al. 2021a; Kirchmeyer et al. 2022). Current developments with multiple environments are mostly data-driven without prior knowledge of the system. Although the studies with multiple environments show promising results, the pure data-driven approaches still raise a question on the extrapolation ability of neural networks, which remains unclear for the standard neural networks (Xu et al. 2021).

In this work, we aim to provide a general framework that can uncover governing dynamics from multiple environments with the help of incomplete prior knowledge. Based on previous work (Brunton, Proctor, and Kutz 2016), we formulate dynamics discovery as a sparse regression problem, assuming that only a few terms govern the dynamics. We expand the sparse regression problem to multiple environments by enforcing that different environments share the same sparse structure over the bases while having different coefficients for the same basis across different environments. To uncover the correct sparse structure, we provide a training algorithm alternating coefficients optimization and pruning. Four different dynamic systems are used to validate the performance of our framework. The experimental results show that the framework can accurately forecast the evolution of systems.

Related Work

Several model-based approaches leveraging prior knowledge for finding unknown dynamics have been proposed (Yin et al. 2021b; Brunton, Proctor, and Kutz 2016). APHYNITY (Yin et al. 2021b) assumes that a physical model is partially known and proposes using neural networks to learn unknown parts of dynamics. In sparse identification of non-linear dynamics (SINDy) (Brunton, Proctor, and Kutz 2016), possible elements of unknown dynamics are used as feature (basis) functions to form a sparse regression problem of dynamic systems. Under the presence of correct

basis functions, SINDy accurately discovers the underlying dynamics.

Data-driven approaches have been proposed to uncover the dynamics given the data collected from multiple environments (Yin et al. 2021a; Kirchmeyer et al. 2022; Norcliffe et al. 2021). LEADS (Yin et al. 2021a) consists of two neural network components: one for capturing common dynamics across all environments and the other for capturing environment-specific dynamics. CoDA (Kirchmeyer et al. 2022) has common parameters and adapts environment with a combination of environment-specific parameters. The number of environment-specific parameters is much less than that of common. Neural ODE Processes (NDP) (Norcliffe et al. 2021) combines Neural ODE (Chen et al. 2018) and Neural Processes (Garnelo et al. 2018). NDP estimates the uncertainty of neural ODE while dealing with multi-environment data. The proposed approaches for the multi-environment data are mostly data-driven. This raises a question of their ability to forecast the unseen environment given the limitation of neural networks on extrapolation (Ziyin, Hartwig, and Ueda 2020; Xu et al. 2021).

Preliminary

In this section, we describe the details of the sparse identification of nonlinear dynamics (SINDy) (Brunton, Proctor, and Kutz 2016) algorithm served as a baseline model in our work.

We consider nonlinear dynamical systems formalized as

$$\frac{d\boldsymbol{x}(t)}{dt} = f(\boldsymbol{x}(t)), \tag{1}$$

where $\boldsymbol{x}(t) = [x_1(t), \cdots, x_n(t)] \in \mathbb{R}^{1\times n}$ represents n-dimensional state of the system at time t, and function $f(\cdot)$ describes a motion of the system. Under the assumption that the function f consists of only a few terms, SINDy casts the problem of uncovering dynamics into a problem of solving a sparse regression problem.

Let $\boldsymbol{X} = [\boldsymbol{x}(t_1); \boldsymbol{x}(t_2); \cdots; \boldsymbol{x}(t_m)] \in \mathbb{R}^{m\times n}$ and $\dot{\boldsymbol{X}} = [d\boldsymbol{x}(t_1)/dt; d\boldsymbol{x}(t_2)/dt; \cdots; d\boldsymbol{x}(t_m)/dt] \in \mathbb{R}^{m\times n}$ be a sequence of observations from m different time steps and their derivatives, respectively. Given a set of p non-linear feature functions $\phi_i : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times 1}$, SINDy constructs a data matrix $\Phi(\boldsymbol{X}) = [\phi_1(\boldsymbol{X}); \phi_2(\boldsymbol{X}); \cdots; \phi_p(\boldsymbol{X})] \in \mathbb{R}^{m\times p}$, where each column represents a candidate non-linear function. SINDy models the dynamics discovery as a sparse regression problem

$$\dot{\boldsymbol{X}} = \Phi(\boldsymbol{X})\boldsymbol{\Xi}, \tag{2}$$

where $\boldsymbol{\Xi} \in \mathbb{R}^{p\times n}$ represents a set of sparse coefficient.

To find a sparse structure of $\boldsymbol{\Xi}$, SINDy iterates over optimization and pruning steps alternatively. During the optimization step, mean-squared-error (MSE) loss with ℓ_1 regularization term is used to minimize the following objective:

$$\min_{\Xi} \left\| \dot{\boldsymbol{X}} - \Phi(\boldsymbol{X})\boldsymbol{\Xi} \right\|_2^2 + \lambda \left\| \boldsymbol{\Xi} \right\|_1, \tag{3}$$

where hyperparameter λ controls the strength of the regularization term. During the pruning step, the coefficients whose absolute values are smaller than a predetermined threshold are set to zero. Once the coefficients are pruned, they are not optimized in the further steps.

Note that the choice of the feature functions is important since the feature functions are the candidate terms to form the dynamic equation. Although, in the original work, constant, polynomial, and trigonometric feature functions are used as candidates, with prior knowledge of dynamic systems, one can choose a proper set of candidates that can potentially explain the dynamic systems well.

Methodology

In practice, observing similar dynamics from different environments is more common. For example, the dynamics of the flow passing a cylinder can be captured from varying sizes and lengths of cylinders. We propose SParse REgresssion for Multi Environment (SpReME) to uncover the underlying dynamics that govern multiple environments by utilizing the commonality between different environments. More specifically, the environments share the same form of the governing equation with a different coefficient for each term. Thus, our goal is to find the proper terms of the governing equation and a proper value of coefficients in each environment.

We consider the dynamics of the form

$$\frac{d\boldsymbol{x}_e(t)}{dt} = f_e(\boldsymbol{x}_e(t)), \tag{4}$$

$\boldsymbol{x}_e(t) \in \mathbb{R}^{1\times n}$ is state observed from environment e. Assume that we have a dataset collected from E different environments. Our goal is to find a general form of dynamics when data collected over m time stamps, i.e., $\boldsymbol{X}_e = [\boldsymbol{x}_e(t_1); \boldsymbol{x}_e(t_2); \cdots; \boldsymbol{x}_e(t_m)] \in \mathbb{R}^{m\times n}$ and $\dot{\boldsymbol{X}}_e = [d\boldsymbol{x}_e(t_1)/dt; d\boldsymbol{x}_e(t_2)/dt; \cdots; d\boldsymbol{x}_e(t_m)/dt] \in \mathbb{R}^{m\times n}$ are given for each environment e.

In the ideal situation, if we solve the sparse regression problem for each environment individually, then the coefficient matrix Ξ_e obtained from different environments has the same sparse structure with different values for each non-zero entry. In reality, however, it is often difficult to obtain the same sparse structure from the results of the optimization process. To enforce the multiple environments share the same government dynamics, we introduce a binary meta-mask matrix $\boldsymbol{M}^* \in \{0,1\}^{p\times n}$. With the meta-mask, we model the sparse regression problem given environment e as

$$\dot{\boldsymbol{X}}_e = \Phi(\boldsymbol{X}_e)(\boldsymbol{M}^* \circ \boldsymbol{\Xi}_e), \tag{5}$$

where $\circ$ indicates element-wise multiplication. By explicitly introducing a binary mask matrix, we force the multiple environments to share the same candidate feature functions while having a different coefficient for each candidate across different environments.

Meta-mask training

To uncover the dynamics, we need to train the model to find the meta-mask applied to all environments and coefficient matrix $\boldsymbol{\Xi}_e$ for each environment. The standard gradient-based methods cannot be applied due to the presence of the meta-mask. To overcome the such challenge, we propose a

Algorithm 1: Training algorithm for meta-mask

Input: dynamics state $\{\boldsymbol{X}_e\}_{e=1}^E$, state derivative $\{\dot{\boldsymbol{X}}_e\}_{e=1}^E$
Output: binary meta-mask $\boldsymbol{M}^*$

$\boldsymbol{\Xi}_e \leftarrow (\Phi(\boldsymbol{X}_e)^\top \Phi(\boldsymbol{X}_e))^{-1}\Phi(\boldsymbol{X}_e)^\top \dot{\boldsymbol{X}}_e, \quad \forall e$
Randomly initialize $\boldsymbol{M}$
for $\tau = 1, \cdots \mathcal{T}$ **do**
  Update $\boldsymbol{M}^*$ via Equation 9
  $\boldsymbol{\Xi}_e \leftarrow \boldsymbol{\Xi}_e - \alpha_\Xi \nabla \mathcal{L}_\Xi, \quad \forall e$ ▷ Optimize coefficient
  Prune coefficients smaller than threshold η_Ξ
  $\boldsymbol{M} \leftarrow \boldsymbol{M} - \alpha_M \nabla \mathcal{L}_{\boldsymbol{M}}$ ▷ Optimize mask
  Prune $\boldsymbol{M}$ via Equation 8
end for
$\boldsymbol{M}^* \leftarrow Q_{\eta_M}(\sigma(\boldsymbol{M}))$

training algorithm that iteratively updates the coefficient matrix and meta-mask in turn.

Given meta-mask $\boldsymbol{M}^*$, we formulate the sparse regression problem as a minimization of the following loss for coefficient matrices $\{\boldsymbol{\Xi}_e\}_{e=1}^E$ at τ-th run:

$$\mathcal{L}_\Xi(\boldsymbol{M}^*, \lambda, s, \tau) = \sum_e \left\| \dot{\boldsymbol{X}}_e - \Phi(\boldsymbol{X}_e)(\boldsymbol{M}^* \circ \boldsymbol{\Xi}_e) \right\|_2^2 + \lambda(1+s)^{-\tau} \sum_e \|\boldsymbol{M}^* \circ \boldsymbol{\Xi}_e\|_1, \quad (6)$$

where $\lambda \geq 0$ and $s \geq 0$ are hyperparameters. We introduce a weight scheduler $(1+s)^{-\tau}$ for the regularizer term. The weight decreases as the training step increases. A gradient descent is used to optimize the coefficients. In practice, we find that further pruning the coefficients based on predefined threshold η_Ξ improves the sparsity after the coefficient optimization step.

Given coefficients $\{\boldsymbol{\Xi}_e\}_{e=1}^E$, optimization over the discrete binary mask requires solving the combinatorial problem, which is intractable in general. To make the optimization tractable, we first relax the binary meta-mask to a continuous meta-mask $\boldsymbol{M} \in \mathbb{R}^{p \times n}$. With the continuous relaxation, we minimize the following objective to update $\boldsymbol{M}$:

$$\mathcal{L}_M(\boldsymbol{M}, \lambda, s, \tau) = \sum_e \left\| \dot{\boldsymbol{X}}_e - \Phi(\boldsymbol{X}_e)(\sigma(\boldsymbol{M}) \circ \boldsymbol{\Xi}_e) \right\|_2^2 + \lambda(1+s)^{\tau} \sum_e \|\sigma(\boldsymbol{M}) \circ \boldsymbol{\Xi}_e\|_1, \quad (7)$$

where σ is a sigmoid function making the relaxed mask ranges between zero and one. The relaxed meta-mask is updated through gradient descent. Note that unlike the coefficient loss in Equation 6, the weight of the regularizer increases as the training step increases. We find that increasing weight helps to explore the sparse structure more in practice.

Once $\boldsymbol{M}$ is optimized, we further prune the mask based on the predefined threshold η_M by setting the corresponding entries as negative infinity. We also prune the entries if the corresponding coefficients are zero across all environments:

$$M_{ij} \leftarrow \begin{cases} -\infty, & \text{if } M_{ij} < \eta_M \text{ or } \Xi_{eij} = 0 \ \forall e \\ M_{ij}, & \text{otherwise.} \end{cases} \quad (8)$$

Note that the pruned entries are not optimized in further steps. After pruning, binary mask $\boldsymbol{M}^*$ can be obtained by quantization:

$$M_{ij}^* = \begin{cases} 0, & \text{if} \quad \sigma(\boldsymbol{M}_{ij}) = 0, \\ 1, & \text{otherwise,} \end{cases} \quad (9)$$

which is again used to update the coefficient.

We initialize coefficient Ξ_e with the analytical solution of the least square loss, i.e., $(\Phi(\boldsymbol{X}_e)^\top \Phi(\boldsymbol{X}_e))^{-1}\Phi(\boldsymbol{X}_e)^\top \dot{\boldsymbol{X}}_e$, making train more efficient. To initialize the continuous meta mask, we use Xavier initialization (Glorot and Bengio 2010).

The overall optimization algorithm is described in Algorithm 1. Note that the optimization steps for the masking matrix are similar to the straight through estimator (Bengio, Léonard, and Courville 2013). Our algorithm alternates the optimization over the continuous mask and the optimization over the coefficients through the quantization.

Validation process

At train time, the ground truth of the meta-mask is not available. Given the assumption that the learned model extrapolates well on future dynamics, we measure the extrapolation performance as a validation criterion. We split the given data $\{\boldsymbol{X}_e, \dot{\boldsymbol{X}}_e\}_{e=1}^E$ into data for timestamp 1 to v and $v+1$ to m, i.e., $\{(\boldsymbol{X}_e)_{:v,*}, (\dot{\boldsymbol{X}}_e)_{:v,*}\}_{e=1}^E$ and $\{(\boldsymbol{X}_e)_{v+1:,*}, (\dot{\boldsymbol{X}}_e)_{v+1:,*}\}_{e=1}^E$. We use the data point from 1 to v for training and $v+1$ to m for validation. The mean-squared error without regularization term, i.e., Equation 6 with $\lambda = 0$, is used as a validation loss.

Adaptation at test time

During the test time, the model encounters an unobserved environment e^* with partial observation $\{\boldsymbol{X}_{e^*}, \dot{\boldsymbol{X}}_{e^*}\}$. To adapt the new environment, we optimize coefficient $\boldsymbol{\Xi}_{e^*}$ conditioned on the optimized meta-mask $\boldsymbol{M}^*$:

$$\boldsymbol{\Xi}_{e^*} = \arg\min_{\boldsymbol{\Xi}} \left\| \dot{\boldsymbol{X}}_{e^*} - \Phi(\boldsymbol{X}_{e^*})(\boldsymbol{M}^* \circ \boldsymbol{\Xi}) \right\|_2^2. \quad (10)$$

We optimize randomly initialized $\boldsymbol{\Xi}$ using gradient descent until convergence.

Experiment

We conducted experiments on four systems of ordinary differential equations (ODE): one linear ordinary differential equation (3D linear model) and three complex non-linear differential equations (Lotka-Volterra, damped pendulum, and Lorenz models).

Experimental settings

To test our model, we generate training trajectories for each ODE system. Given a system of ODE, LSODA ODE solver (Hindmarsh and Petzold Sep 2005) is used to generate a trajectory from an initial state with a fixed time interval over a limited time horizon. Using state trajectories and ODE, derivative of states are computed except the 3D linear model. The numerical method is used for 3D linear. We use

	meta-mask train			adaptation					
				train			test		
	time horizon	dt	# trajectories	time horizon	dt	# trajectories	time horizon	dt	# trajectories
3D linear	4	0.02	16	4	0.02	16	10	0.01	16
Lorenz	4	0.02	16	4	0.02	16	10	0.01	16
LV	10	0.50	4	10	0.50	1	25	0.25	32
DP	4	0.20	8	4	0.20	1	10	0.10	32

Table 1: Experiment settings for data generation. LV and DP refer to Lotka Volterra and Damped Pendulum, respectively. dt means time interval.

the first 80% for the train and the remaining 20% for validation for each trajectory. Each dimension of the initial state for each trajectory is sampled from the standard normal distribution, i.e., $\mathcal{N}(0,1)$, except in Lotka-Volterra. In Lotka-Volterra, each dimension of the initial state is sampled from the normal distribution $\mathcal{N}(1,1)$. Using state trajectories and ground truth ODE, $\{\dot{\boldsymbol{X}}_e\}_{e=1}^E$ are computed except the 3D linear model. In 3D linear, we use the numerical method for computing derivatives. We use the first 80% for the train and the remaining 20% for validation for each trajectory.

At test time, we use a single unobserved environment. We use the same time horizon and interval for adaptation. To interpolate and extrapolate trajectories, we use half of the time interval over $2.5\times$ longer time horizon to predict unobserved states. Table 1 describes the settings used to generate the training and test data sets.

To construct a data matrix $\boldsymbol{\Phi}(\cdot)$, we choose the feature functions for the candidate as constant and polynomial with degree five, except for a system of the damped pendulum. We additionally choose trigonometric feature function $\sin(x_1(t))$, $\sin(x_2(t))$ and $\sin(x_1(t)+x_2(t))$ as candidates for a system of damped pendulum.

3D linear model. 3D linear model is a toy example of ODE. Given state $\boldsymbol{x} \in \mathbb{R}^3$, ODE has the form:

$$\begin{aligned} dx_0/dt &= \alpha x_0 + \beta x_1, \\ dx_1/dt &= \gamma x_0 + \delta x_1, \\ dx_2/dt &= \omega x_2, \end{aligned}$$

where $\alpha, \beta, \gamma, \delta, \omega$ are parameters of the model. We use three environments in meta-mask training. For each environment, parameter $\alpha, \beta, \gamma, \delta, \omega$ are randomly sampled from an independent normal distribution with a mean of $-0.1, 2, -2, -0.1, -0.3$, respectively, and standard deviation of 0.01. At test time, we use the mean values as the parameters of ODE to generate the test trajectory.

Lorenz. Lorenz is a non-linear model representing atmospheric convection in $\mathbb{R}^3$. This is a chaotic system where a small difference in the initial point rapidly becomes a large difference (Sparrow 1982). The dynamics of the Lorenz model follows:

$$\begin{aligned} dx/dt &= \sigma(y-x), \\ dy/dt &= x(\rho - z) - y, \\ dz/dt &= xy - \beta z, \end{aligned} \tag{11}$$

where σ, ρ, β are parameters assumed to be positive values. For meta-mask training, we use three environments. For each environment, parameter σ, ρ, β are independently sampled from a normal distribution with a mean of $10, 8/3, 28$, respectively, and a standard deviation of 0.01. At test time, we use the mean values as the parameters of ODE to generate an unobserved environment.

Lotka-Volterra. This model describes the dynamics of biological interaction between predator and prey. The dynamics have the form of ODE (Bacaër 2011):

$$\begin{aligned} du/dt &= \alpha u - \beta uv, \\ dv/dt &= \delta uv - \gamma v, \end{aligned}$$

where $\alpha, \beta, \gamma, \delta$ are parameters of LV model. We use nine environments for meta-mask training. For training and testing, we use the same value of the parameters following the prior work (Kirchmeyer et al. 2022).

Damped Pendulum. This model describes the dynamics of a damped pendulum (Quiroga and Ospina-Henao 2017):

$$d\theta^2/dt^2 + (b/m)d\theta/dt + g/L\sin(\theta) = 0,$$

where θ, b, m, g, L are angular, damping factor, the pendulum's mass, gravity, and length, respectively. $\boldsymbol{x}$ is defined as $\{\theta, d\theta/dt\}$. Nine environments are used for meta-mask training. For training and testing, we use the same value of the parameters following the prior work (Kirchmeyer et al. 2022).

Baselines

We evaluate adaptation in a new environment with the mean squared error (MSE) between prediction and ground truth. We use two baseline models for comparison: LEADS (Yin et al. 2021a) and CoDA (Kirchmeyer et al. 2022). Both approaches are proposed to deal with multi-environment data.

LEADS. LEADS generalizes dynamics from multi-environment data with a combination of the neural network capturing common dynamics across all environments and environment-specific neural networks:

$$\frac{d\boldsymbol{x}_e(t)}{dt} = (f + g_e)(\boldsymbol{x}_e(t)), \tag{12}$$

where f is a shared neural network across all environments, and g_e is an environment-specific neural network. LEADS generalizes dynamics at the function level.

	3D Linear	Lorenz	LV	DP
Precision	1	0.7	1	1
Recall	1	1	1	1
# candidates	56	56	21	24

Table 2: The precision and recall of the trained mask obtained from the lowest validation loss. LV and DP refer to Lotka Volterra and Damped Pendulum, respectively.

		3D Linear	Lorenz	LV	DP
Avg	LEADS	0.0099	1.019	0.2682	0.0275
	CoDA	0.0217	3.591e+10	0.0610	0.0162
	SpReME	**1.648e-05**	**0.3841**	**0.0022**	**1.576e-6**
Best	LEADS	0.0015	0.1220	0.1733	0.0105
	CoDA	0.0021	293.5	0.0158	0.0006
	SpReME	**1.059e-6**	**0.0468**	**0.0006**	**2.093e-8**
Worst	LEADS	0.0223	4.4185	0.3917	0.2557
	CoDA	0.0554	2.288e+11	0.2089	0.1081
	SpReME	**4.254e-5**	**2.3557**	**0.0052**	**3.150e-5**

Table 3: Test MSE loss in adaptation. Avg represents the average MSE loss across all trajectories. Best and worst represent the best and worst MSE loss among all trajectories. LV and DP refer to Lotka Volterra and Damped Pendulum, respectively. **Bold** denotes the best result for each dataset.

CoDA. CoDA uses meta-learning for the generalization of dynamics. It has common parameters across all environments. It adapts a few environment-specific parameters and combines them with common parameters:

$$\frac{d\boldsymbol{x}_e(t)}{dt} = f_{\theta + W\phi_e}(\boldsymbol{x}_e(t)), \tag{13}$$

where f is neural networks, $\theta \in \mathbb{R}^p$ are common parameters, $\phi_e \in \mathbb{R}^a$ are environment-specific parameters, and $W \in \mathbb{R}^{p \times a}$ is a matrix for parameter combination. CoDA generalizes dynamics at the parameter level.

We search hyper-parameters of both models with the validation steps described in the previous section.

Results

Table 2 shows the performance on the accuracy of trained mask results obtained from the best validation loss. We evaluate trained masks with precision and recall. The recall of trained masks is one in all cases, meaning each mask has all elements of governing equation. The precision of the mask is one without in the Lorenz dataset, which means the mask has no dummy elements. In adaptation, we use the result of Table 2.

Table 3 shows test MSE loss of SpReME, LEADS, and CoDA in adaptation. We report the average performance over all trajectories as well as the cases with the best and worst performance. SpReME outperforms LEADS and CoDA in all cases. Note that the performance on the Lorenz model is relatively worse than the other cases. CoDA fails to uncover the original dynamics in almost all cases, whereas our approach shows more stable results across all cases than the others. In Figure 1 and 2, we plot the predicted trajectories of SpReME, LEADS, and CoDA with different models with the same training sets. In Lorenz and Lotka-Volterra datasets, LEADs and SpReME achieve similar performance in terms of interpolation. The extrapolation results of SpReME are better than those of Lorenz, showing the importance of using prior knowledge in modeling dynamics even though the knowledge is incomplete.

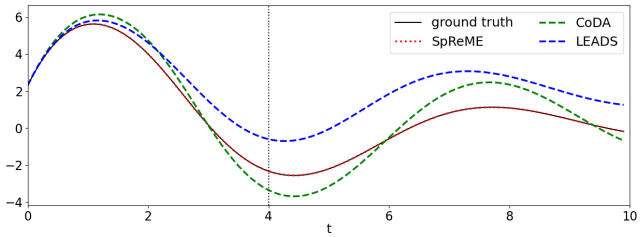

Figure 1: Interpolation and extrapolation results of pendulum dynamics with various approaches. The y-axis represents $L\sin(\theta(t))$. The solid black line is the ground truth, the dotted orange line is our prediction, the dashed blue line is prediction of LEADS, and the dashed green line is prediction of CoDA. Note that the first four seconds are used to train the models.

Case study As shown in Table 2, the Lorenz model has identified incorrect terms in trained mask. The model identifies additional three terms incorrectly. The dynamics uncovered by the trained mask is as follows:

$$\begin{aligned} dx/dt &= \sigma(y - x) + \underline{\epsilon_1 x z^2}, \\ dy/dt &= x(\rho - z) - y + \underline{\epsilon_2 x^2 y z}, \\ dz/dt &= xy - \beta z + \underline{\epsilon_3 y^2}, \end{aligned}$$

where incorrectly identified terms are the underlined with their coefficient ϵ_i (c.f., Equation 11). We find that, at test time, the coefficients of incorrect terms are small (< 0.00605), leading to a better performance than the other baseline models.

Conclusion and Future Work

We propose SpReME for discovering dynamics from multiple environments data with incomplete prior knowledge. Unlike prior work, SpReME use incomplete prior knowledge while dealing with multiple environments. Through the meta-masking scheme, we extract commonalities between different environments. Our model adapts to new environments based on the meta-mask. The experimental results show that SpReME outperforms LEADS and CoDA in most cases.

Throughout the experiments, we assume that the multiple environments share the same underlying dynamics with changes in parameters. In reality, the signals captured from the different environments may not share the same governing equations. In such a scenario, having an additional component explaining the difference would help to capture the commonality while distinguishing subtle differences, as done in LEADS. We leave this for future work.

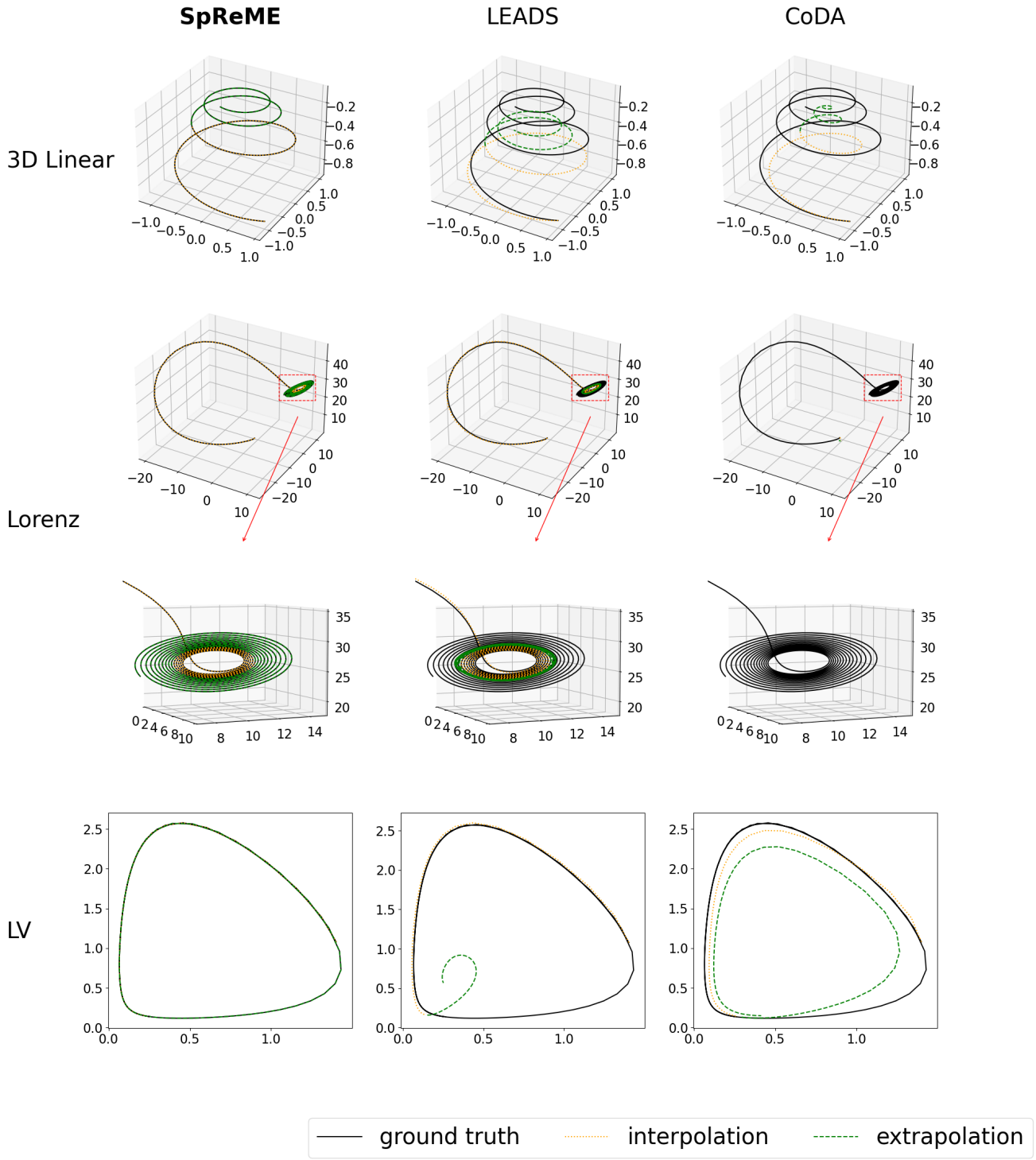

Figure 2: Comparison between the prediction of SpReME, LEADS and CoDA for 3D linear, Lorenz and Lotka-Volterra models. The solid black lines are ground truth, the dotted orange lines are interpolation results, and the dashed green lines are extrapolation results. With the Lorenz model, we provide the enlarged views of the extrapolated region (red boxed), changing dramatically from the training trajectory.

References

Bacaër, N. 2011. *Lotka, Volterra and the predator–prey system (1920–1926)*, 71–76. London: Springer London. ISBN 978-0-85729-115-8.

Bengio, Y.; Léonard, N.; and Courville, A. C. 2013. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *CoRR*, abs/1308.3432.

Brunton, S. L.; Proctor, J. L.; and Kutz, J. N. 2016. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *PNAS*, 113(15): 3932–3937.

Chen, R. T. Q.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. K. 2018. Neural Ordinary Differential Equations. In *NeurIPS*.

Garnelo, M.; Schwarz, J.; Rosenbaum, D.; Viola, F.; Rezende, D. J.; Eslami, S. M. A.; and Teh, Y. W. 2018. Neural Processes. *CoRR*, abs/1807.01622.

Glorot, X.; and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 249–256.

Hindmarsh, A. C.; and Petzold, L. R. Sep 2005. LSODA, Ordinary Differential Equation Solver for Stiff or Non-Stiff System.

Kirchmeyer, M.; Yin, Y.; Dona, J.; Baskiotis, N.; Rakotomamonjy, A.; and Gallinari, P. 2022. Generalizing to New Physical Systems via Context-Informed Dynamics Model. In *ICML*.

Li, S. H. Q.; Ratliff, L. J.; and Açikmese, B. 2021. Disturbance Decoupling for Gradient-Based Multi-Agent Learning With Quadratic Costs. *L-CSS*, 5: 223–228.

Long, Z.; Lu, Y.; Ma, X.; and Dong, B. 2018. PDE-Net: Learning PDEs from Data. In *ICML*.

Noack, B. R.; Afanasiev, K.; MORZYŃSKI, M.; Tadmor, G.; and Thiele, F. 2003. A hierarchy of low-dimensional models for the transient and post-transient cylinder wake. *JFM*, 497: 335–363.

Norcliffe, A.; Bodnar, C.; Day, B.; Moss, J.; and Liò, P. 2021. Neural ODE Processes. In *ICLR*.

Quiroga, G. D.; and Ospina-Henao, P. A. 2017. Dynamics of damped oscillations: physical pendulum. *Eur. J. Phys.*, 38(6): 065005.

Shi, X.; Chen, Z.; Wang, H.; Yeung, D.-Y.; Wong, W.-k.; and Woo, W.-c. 2015. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *NeurIPS*.

Sparrow, C. 1982. *Introduction and Simple Properties*, 1–12. New York, NY: Springer New York. ISBN 978-1-4612-5767-7.

Wang, Y.; Gao, Z.; Long, M.; Wang, J.; and Yu, P. S. 2018. PredRNN++: Towards A Resolution of the Deep-in-Time Dilemma in Spatiotemporal Predictive Learning. In *ICML*.

Xu, K.; Zhang, M.; Li, J.; Du, S. S.; Kawarabayashi, K.-i.; and Jegelka, S. 2021. How neural networks extrapolate: from feedforward to graph neural networks. *ICLR*.

Yin, Y.; Ayed, I.; de Bézenac, E.; Baskiotis, N.; and Gallinari, P. 2021a. LEADS: Learning Dynamical Systems that Generalize Across Environments. *CoRR*, abs/2106.04546.

Yin, Y.; GUEN, V. L.; DONA, J.; de Bezenac, E.; Ayed, I.; THOME, N.; and patrick gallinari. 2021b. Augmenting Physical Models with Deep Networks for Complex Dynamics Forecasting. In *ICLR*.

Ziyin, L.; Hartwig, T.; and Ueda, M. 2020. Neural Networks Fail to Learn Periodic Functions and How to Fix It. In *NeurIPS*.