Comparison between Sparse Identification of Nonlinear Dynamics with Globally Linear Latent Dynamics Model in Dynamics Model Identification for Latent Space Control

1st Ruiyang Wang
Robotics Department
University of Michigan
Ann Arbor, United States
ruiyangw@umich.edu

***Abstract*—This project aims to compare Globally Linear Latent Dynamics (GLLD) models with Sparse Identification of Nonlinear Dynamics (SYNDy) models in dynamics model identification for latent space control. We evaluated models in the task of planar pushing in the Pybullet environment with a panda robot. We found that SYNDy model with poly degrees 2 and 3 performs better than GLLD model in validation loss, number of runs that arrived at the goal position, and average steps to reach the goal position. We also observed the problem of over-fitting in SYNDy models as SYNDy model with poly degree 2 performs better than with poly degree 3 in validation loss, number of runs that arrived at the goal position, and average steps to reach the goal position.**

***Index Terms*—Latent Space Control, Dynamics Model, Auto-encoder**

I. Introduction

Model identification is a critical aspect of robotics that involves the process of determining mathematical models that accurately describe the behavior of physical systems. In the context of robotics, this typically involves identifying the dynamics and kinematics of the robot, which are essential for various applications such as motion planning, control design, and system analysis.

An accurate model of a robot's behavior allows engineers to predict its performance under different scenarios, optimize its design and operation, and ultimately improve its overall functionality. Without an accurate model, it is difficult to design efficient controllers or perform accurate simulations, which can lead to poor performance, unstable behavior, and even dangerous situations. Therefore, model identification plays a vital role in the development of reliable and effective robotics systems.

However, identifying accurate models of a robot's dynamics and kinematics can be a challenging task, as it often involves dealing with complex nonlinear systems. Furthermore, robots are subject to uncertainty, such as varying friction and external disturbances, which can further complicate the model identification process.

Those difficulties made it nearly impossible in some scenarios to derive an accurate model from physical laws or other first principle methods. Luckily, many of these systems have rich time series of data so it is possible to learn the model from data. This has encouraged a new paradigm of data-driven model discovery in industry and academia.

In this project, we studied two data-driven model discovery frameworks: Sparse Identification of Nonlinear Dynamic (SINDy) [1] and Globally Linear Latent Space (GLLD) Model, which is a simplified version of E2C (Embed to Control) [2], and compared their performance in dynamics model identification and latent space control in simulation.

The rest of the paper is structured as follows. Section II will formulate the problem we want to solve, Section III will briefly introduce the two frameworks we compared and explain how we implemented the frameworks in Python, Section IV will present simulation results, and Section V concludes our findings.

II. Problem Formulation

Let $x \in R^n$ be the input data from a dynamical system, we can compress the data using an encoder, $\psi(x)$, and represent it in a reduced coordinate:

$$z = \psi(x) \in R^d \tag{1}$$

where $d \ll n$, and z is the representation of input data in latent space.

We can also convert data in latent space back into original data space using a decoder, $\phi(z)$:

$$x = \phi(z) \in R^n \tag{2}$$

For dynamic model identification in latent space control, we want to find the associated dynamical model:

In continuous space:

$$\frac{d}{dt}z(t) = g(z(t), a(t)) \tag{3}$$

This project is for ROB498-001: Robot Learning for Planning and Control at the University of Michigan

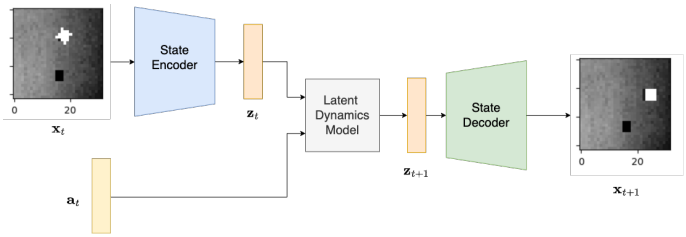

Fig. 1. The basic structure for Latent Space Control when input data is in discrete space

In discrete space:

$$z_{t+1} = g(z_t, a_t) \tag{4}$$

where $a \in R^m$ is the action applied in the latent space.

With this dynamical model in latent space, we can predict the change in input data space by first passing it through the encoder to get its latent space data representation, then apply action in the latent space and update latent space states through the latent space dynamical model, and finally passing it through the decoder to obtain the update states in the original data space. This process is illustrated in a flow chart in Fig. 1 when the input data is in discrete space, which is the default data format we deal with in this project.

III. Methodology and Implementation

In this section, we will mainly discuss the two different dynamical system identification frameworks we want to compare with. But before that, we want to first discuss the implementation of the encoder and decoder as they are shared for both frameworks.

A. Encoder and Decoder Implementations

To transform input into latent space representation, we pass the input image through 2 convolution layers, each is followed by a ReLU activation layer and a max pool layer. The data is then flattened and passed through two fully connected layers as final steps. We chose the latent dimension to be 16 in our implementation. The overall structure of the encoder is shown in Fig. 2.

The structure of the decoder is summarized in Fig. 3. We pass the latent space data first through two fully connected layers each followed by a ReLU activation layer. Then the data is passed through one final fully connected layer and reshaped to the input data shape. The overall structure of the encoder is shown in Fig. 3.

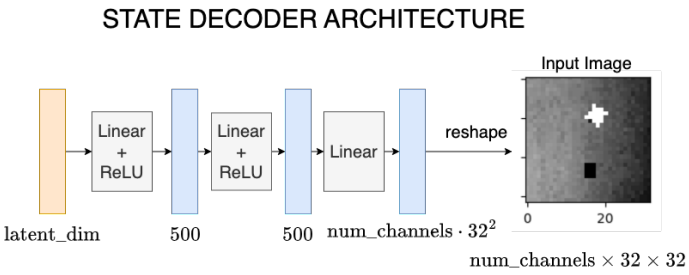

Fig. 3. The structure of the decoder

B. Globally Linear Latent Dynamics (GLLD) Model

In Globally Linear Latent Dynamics (GLLD) model identification, we assumed that the nonlinear dynamics can be linearized in the format of:

$$z_{t+1} = Az_t + Ba_t + b \tag{5}$$

where A and B are two constant matrices that describe the latent space dynamics and b is a constant offset in the dynamics.

We learn the exact values of A, B, and b through a three-layer fully connected neural network. We combined the training of the encoder, latent space dynamical model, and the decoder together in the implementation, and the final loss function is defined as:

$$\begin{aligned} L_{global} = \frac{1}{T+1}\sum_{t=0}^{T} ||x_t - \texttt{decode}(z_t)||^2 + \\ \frac{1}{T}\sum_{t=1}^{T} ||x_t - \hat{x}_t||^2 + \frac{\alpha}{T}\sum_{t=1}^{T} ||z_t - \hat{z}_t||^2 \\ = \text{Reconstruction Loss} + \\ \text{Prediction Loss} + \text{Latent Prediction Loss} \end{aligned} \tag{6}$$

where $\hat{z}_t$ and $\hat{x}_t$ are predicted states in latent space and input data space respectively from the dynamic model, and α is a

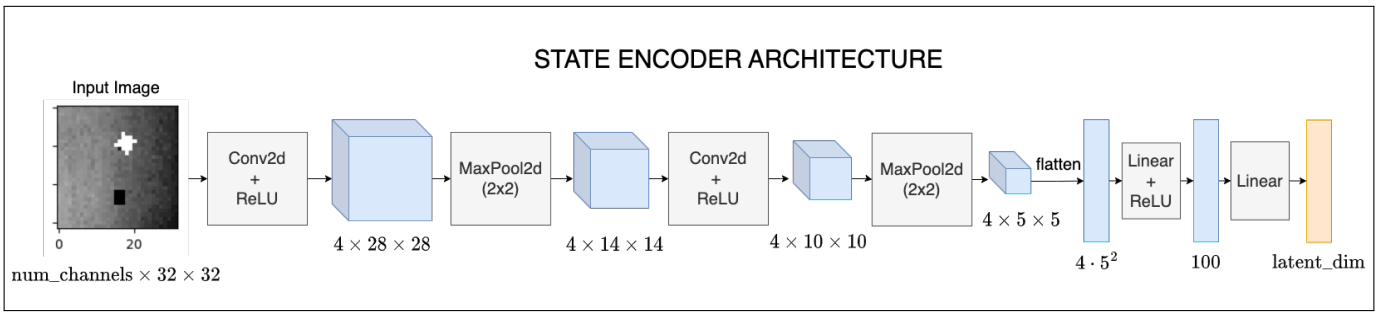

Fig. 2. The structure of the encoder

hyperparameter for the weight of latent prediction loss, we used 0.1 in our implementation.

C. Sparse Identification of Nonlinear Dynamics (SINDy) Model

The Sparse Identification of Nonlinear Dynamics (SINDy) model first constructed a library $\Theta(z) = [\theta_0(z), \theta_1(z), ..., \theta_p(z)]$ of candidate basis functions of z, and we want to learn a sparse set of coefficients $\Xi = [\xi_0, \xi_1, ..., \xi_p]$ such that:

$$z_{t+1} = \Theta(z_t)\Xi \tag{7}$$

In our implementation, we used different degrees of polynomials between each state in the latent space, along with a single element in control input to form our candidate basis functions:

$$\Theta = [z[0], z[1], ..., z[0]^2, z[0]z[1], z[0]z[2], ..., a[0], a[1], .., a[m]] \tag{8}$$

We used a one-layer fully connect neural network to learn the values of Ξ, and we also combined the training of the encoder, SINDy model, and the decoder together in the implementation. The final loss function is defined as:

$$\begin{aligned} L_{SINDy} = \frac{1}{T+1}\sum_{t=0}^{T} ||x_t - \texttt{decode}(z_t)||^2 + \\ \lambda_1 \frac{1}{T}\sum_{t=1}^{T} ||x_t - \hat{x}_t||^2 + \lambda_2 \frac{1}{T}\sum_{t=1}^{T} ||z_t - \hat{z}_t||^2 + \\ +\lambda_3 ||\Xi||^1 = \text{Reconstruction Loss} + \\ \text{Prediction Loss} + \text{Latent Prediction Loss} + \\ \text{Regularization Loss} \end{aligned} \tag{9}$$

where $\hat{z}_t$ and $\hat{x}_t$ are predicted states in latent space and input data space from the dynamic model, and λ_1, λ_2, λ_3 are hyperparameters for the weight of prediction loss, latent perdition loss, and regularization loss. We used 0.1 for all in our implementation.

IV. Simulation Results

We evaluated the latent dynamics models from two frameworks using PyBullet to simulate a panda robot that pushes a block on top of a table. An illustration of the simulation environment is shown in Fig. 4.

In the training process, each data sample contains an initial state image and 4 actions applied consecutively to the system along with their resulting images.

We trained each model with 560 data samples, with a batch size of 512 for 2000 epochs on the CPU provided by Google CoLab for each model, and we saved 140 data samples for validation. The results are summarized in Table. I

The simulation suggested that SINDy models generally take longer to train compared to GLLD, we think this is mainly because of our inefficient implementation of constructing the Θ library using nested for loops. The training time of the SINDy model can be much shorter if we can parallelize the construction of the Θ library. On the other hand, the SINDy models generally have a lower validation loss compared to the GLLD model. This can be better visualized in Fig. 5. We used the different dynamics models in latent space to predict 10 steps forward after applying 10 actions consecutively to the system from the initial state.

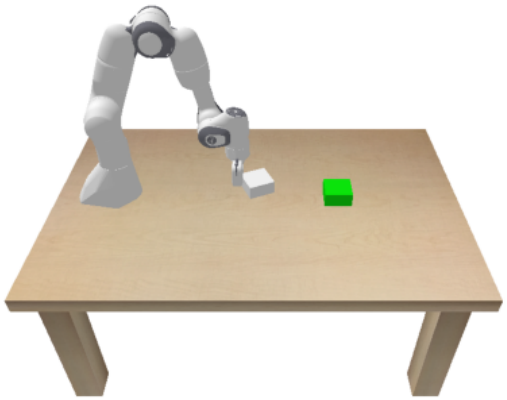

Fig. 4. Simulation Environment using PyBullet, the green box shows the target position of the white block.

Model	Poly Degree	Training Time(s)	Validation Loss	Total # of learning parameters	Proportion of learning parameters equal to 0
GLLD		1503	0.1885	13716	0.0001
SINDy	1	1311	0.1338	320	0.7094
	2	1842	0.0779	2496	0.8197
	3	8072	0.0959	15552	0.815

TABLE I
SIMULATION RESULT COMPARISON BETWEEN GLLD MODEL AND SINDY MODEL WITH POLY DEGREES OF 1, 2, AND 3.

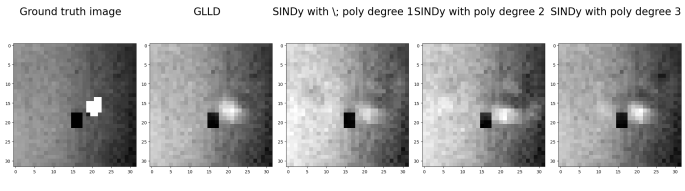

Fig. 5. Prediction in input space (image space) after 10 actions being applied to the system from the initial state.

The reconstructed images suggested that as the poly degree in SINDy models increases, we are more confident about the position of the block, and we have a better estimation of the block's position from SYNDy models compared to GLLD model. This is also the trend we have seen previously in terms of validation losses.

In Table I, we can also observe a trend of increasing sparsity

Model	Poly Degree	# of runs arrived the goal	Average Steps to reach the goal
GLLD		6	10.1
SINDy	1	0	20.0
	2	10	8.2
	3	9	9.1

TABLE II
SIMULATION RESULT COMPARISON BETWEEN GLLD MODEL AND SINDY MODEL WITH POLY DEGREE OF 1, 2 AND 3 IN PYBULLET SIMULATION WITH PANDA ROBOT.

in SINDy models as the polynomial degrees in the model increase. The sparsity is calculated using the proportion of learning parameters equal to zero in all learning parameters. This indicates that a large portion of the candidate function in Θ is irrelevant to the latent dynamics of the system, and we generally have a small number of active learning parameters in SYNDy models.

The next simulation result is to show how different latent dynamics models can affect the efficiency of the panda robot pushing the block to its desired position in PyBullet simulation environment.

We used an MPPI controller to generate control inputs for the panda robot, and we used the MSE loss in latent space as our loss function for the MPPI controller. The simulation has been run for each model 10 times, with a maximum steps of 20. If the block has not been pushed to the target position after the maximum number of steps, we record the model using 20 steps to reach the goal. If the block reached the goal earlier than 20 steps, we record the used steps to reach the goal. The result is summarized in Table II

As we can see from the table, SINDy models with poly degree 2 and 3 have better performance compared to GLLD model in terms of both the number of runs that arrived the goal and the average number of steps to reach the goal. On the other hand, SINDy model with poly degree of 1 cannot reach the goal position under 20 steps in any of its runs. We think the reason is that the SINDy model with poly degree 1 has a limited ability to describe the latent dynamical system effectively due to its limited complexity and number of learning parameters in the model.

We also observed a phenomenon of over-fitting when we compare SINDy models with poly degree 2 and 3. We observed a lower validation loss in Table. I, and more successful runs to reach the goal with fewer steps in Table II when using SINDy model with poly degree 2 compared with poly degree 3. We think this phenomenon is because SINDy model with poly degree 3 has far more learning parameters, as shown in Table. I, compared to the SINDy model with poly degree 2, therefore, it is much easier to over-fit the model with poly degree 3. But if we have more training data, we believe SINDy model with poly degree 3 can perform even better because of its capability of catching more dynamic details.

V. CONCLUSION

In conclusion, we compared the Globally Linear Latent Dynamics (GLLD) Model with Sparse Identification of Nonlinear Dynamics (SYNDy) model in dynamics model identification for latent space control. We evaluated models in the PyBullet environment with the task of using a panda robot to push a block to the desired position.

We found that the training time of GLLD model is close to the training time of SYNDy model with poly degree 1 and 2. Meanwhile, the validation loss in multi-step dynamics prediction of SYNDy models are generally less compared to the SYNDy models. We found an increasing sparsity in SYNDy models when a larger portion of the learning parameters becomes zero when increasing the poly degree of the model.

In the task of using the panda robot to push a block to its desired position in a maximum of 20 steps. We found that SINDy models with poly degrees 2 and 3 perform better than GLLD models in terms of both number of runs arrived at the goal position as well as the average steps to reach the goal position. SYNDy model with poly degree 1 cannot guide the panda robot to push the block successfully to desired position, we think this is because of the limited capability of SYNDy model with poly degree 1 to describe the latent space dynamics.

We also found the problem of over-fitting with SINDy models since we observed that SINDy model with poly degree 2 performs better than with poly degree 3 in terms of validation loss, number of successful runs to reach the goal position, and average steps to arrive at the goal position.

For future works, we want to make the implementation of SYNDy models training more efficient by finding a way of constructing the candidate function library Θ in a parallel fashion instead of using nested for loops.

REFERENCES

[1] K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton, "Data-driven discovery of coordinates and governing equations," *Proceedings of the National Academy of Sciences*, vol. 116, no. 45, pp. 22 445–22 451, 2019.

[2] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," *Advances in neural information processing systems*, vol. 28, 2015.