Towards Scalable Koopman Operator Learning: Convergence Rates and Distributed Implementation

Zhiyuan Liu, Guohui Ding, Lijun Chen and Enoch Yeung

***Abstract*— In this paper, we propose an alternating optimization algorithm to the nonconvex Koopman operator learning problem, using shallow neural networks, for nonlinear dynamic systems. We show that the proposed algorithm will converge to a critical point with rate $O(1/T)$ or $O(\frac{1}{\log T})$ under some mild assumptions. To handle the high dimensional nonlinear dynamical systems, we present the first-ever distributed Koopman operator learning algorithm. We show that the distributed Koopman operator learning has the same convergence properties as a centralized Koopman operator learning problem, in the absence of optimal tracker, so long as the basis functions satisfy a set of state-based decomposition conditions. Experiments are provided to complement our theoretical results.**

I. Introduction

In recent years, there is an increasing interest in transferring operator theoretic techniques such as Koopman operator [1], [2] for the analysis of dynamical systems. Such operator based methods differ from classical approaches, in that they define the evolution of observable functions in a function space rather than using state vectors in a state space. The power of these operator theoretic methods is that it provides linear representations of nonlinear time-invariant systems, albeit in higher dimensional spaces that are sometimes countable or uncountable. Various numerical approaches, such as dynamic mode decomposition(DMD), Hankel-DMD, extended dynamic mode decomposition (E-DMD), structured dynamic mode decomposition (S-DMD) have been proposed for discovering the Koopman operator of a nonlinear system, using a series of dictionary functions with spanning or universal function approximation properties [2]–[6]. Recently, researchers have shown it is possible to integrate machine-driven learning representations with dynamic mode decomposition algorithms, using variational autoencoders to achieve phase-dependent representations of spectra [7] or delay embeddings [8], shallow neural networks [3], linearly recurrent neural networks for balancing expressiveness and overfitting [9], and deep RELU feedforward networks for predictive modeling in biological and transmission systems [10]. E-DMD [3] and Deep-DMD [10] have been utilized in various domains, including nonlinear system identification [11]–[13], image processing [4], [14] and robotic control [15], [16].

Z. Liu, L. Chen and G. Ding are with the College of Engineering and Applied Science, University of Colorado, Boulder, CO 80309, USA (emails: {zhiyuan.liu, lijun.chen, duohui.ding}@colorado.edu).

E. Yeung is with the Department of Mechanical Engineering, the Center for Control, Dynamical Systems, and the Center for Biological Engineering at the University of California Santa Barbara, CA 93106, USA (email: eyeung@ucsb.edu).

Generally speaking, the learning especially the training phase of Koopman operator is trying to minimize the empirical loss based on the training set, e.g., the data sampled from the real trajectory of dynamic system. Compared to the traditional machine learning problem which learns the unknown mapping from input to output, the Koopman learning has two tasks: 1) Learning the function space that lifts state space to a high even infinite dimensional space. 2) Learning a linear mapping within that function space. These two tasks are highly related to each other, e.g., inappropriate function space learned will lead to poor learning performance even the linear mapping is perfect. However, to the best of our knowledge, the method of Koopman training has not gotten enough attention up to now. Another challenge is that, when parameterized function approximation such as neural network is used, the learning problem is nonconvex. For instance, even for single layer neutral network, it is NP-complete to find the global optimal [17]. However, recent works [18]–[20] show that for over-parameterized (wide) shallow neural networks, local optima provide satisfactory performance. Specifically, they show that every local optimum is global optimum if the hidden layer is non-singular; every local minimum of the simplified objective is close to the global minimum. In this paper, we contribute a proof of convergence for Koopman learning algorithms utilizing shallow neural networks, and derive conditions for first-order optimality, the properties of the so-called dictionary functions used in deep and E-DMD that guarantee convergence. We propose alternating optimization algorithm with an optimal tracker for training the Koopman operator. By proving the objective function's smoothness property, we show that our algorithm admits $O(1/T)$ convergence rate for chosen constant learning rate and $O(1/\log T)$ for diminishing learning rate. We illustrate convergence of the alternating optimization algorithm for single-node training (non-distributed) on two nonlinear systems with oscillatory dynamics.

A second major contribution of this paper is the development of a distributed Koopman operator learning algorithm. Most Koopman operator learning algorithms operate under the assumption of full-state measurements. Frequently, in engineered and natural systems represented by data, full-state measurements are not available, or are too expensive to collect. For example, power distribution networks consisting of hundreds of thousands of nodes exhibit real-time dynamics on systems that are poorly modeled, calibrated, or dated. Biological networks operate on thousands of genes to generate transcriptomic reponse profiles as a function of time; full-state measurement via deep sequencing is prohibitively

Manuscript 1236 submitted to 2020 American Control Conference.
Received September 26, 2019.

expensive. In many instances, it is much more feasible to collect measurements from select locations, via strategic placement of observers [21], which gives rise to a different form of data — time-series data that is spatially distributed or fragmented across the whole network. We address the challenge of training distributed representations of Koopman operators and develop a *distributed* Koopman learning algorithm, proving asymptotic convergence, and illustrating predictive accuracy and convergence on several simulated examples.

The rest of the paper is organized as follows. Section II introduces the Koopman operator learning problem. Section III describes our alternating optimization algorithm for Koopman learning and proves the convergence. Section IV extends our algorithm to a distributed version and shows its convergence. Section V shows the performance of two algorithms validated by two nonlinear systems. Section VI concludes this paper.

II. Koopman operator Learning Problem

In this paper, we consider a discrete time open-loop nonlinear dynamic system of the following form

$$x_{n+1} = f(x_n), \tag{1}$$
$$y_n = h(x_n), \tag{2}$$

where $f : \mathbb{R}^d \to \mathbb{R}^d$ and $h : \mathbb{R}^d \to \mathbb{R}^p$ are continuously differentiable. The function f is the state-space model and the function h maps current state $x_n \in \mathbb{R}^d$ to a vector of observables or output $y_n \in \mathbb{R}^p$. The Koopman operator $\mathcal{K}$ of system (1), if it exists, is a linear operator that acts on observable functions $\psi(x_k)$ and forward propagates them in time. To be more specific, the Koopman operator for this system must satisfy the equation

$$\psi(x_{n+1}) = \mathcal{K}(\psi(x_n)), \tag{3}$$
$$y_n = \mathcal{H}(\psi(x_n)), \tag{4}$$

where $\psi(x_n) = [\psi_1(x_n), \cdots, \psi_m(x_n)]^\top : \mathbb{R}^d \to \mathbb{R}^m (m \leq \infty)$ is a basis function that defines the lifted space of observables and $\mathcal{K} \in \mathbb{R}^{m\times m}$ is a constant matrix. Based on the Koopman operator theory, ψ is the basis function of observables under which $\psi(x_n)$ is $\mathcal{K}$-invariant for all n. This implies that the Koopman operator comprehensively interprets the flow of the observable trajectory $(x_1, x_2, \cdots)$.

Based on the data-driven method [10] [22], a general model for approximating Koopman operator given the data trajectory $(x_i, x_{i+1}), i \in \{1, \cdots, N\}$ can be formulated as follows:

$$\min_{\psi,\mathcal{K}} \ \mathcal{D}(\psi, \mathcal{K}) := \frac{1}{2N}\sum_{i=1}^{N} \|\psi(x_{i+1}) - \mathcal{K}\psi(x_i)\|_2^2. \tag{5}$$

The above model aims to minimize the empirical loss from the learning perspective. One can slightly change the objective function by adding certain regularized term, e.g., $\|\mathcal{K}\|_1$ for sparse operator or $\|\mathcal{K}\|_2$ for avoiding large training lose, to make the tradeoff between the training and generalization error.

While there has been a surge of interest in using neural networks to perform Koopman learning, little is known regarding the convergence and numerical stability of the training process. This motivates us to investigate the property of Koopman learning during its training phase. There are two challenges in solving for the optimization problem (5) in practice. First, the basis function ψ is unknown. This makes it difficult to ascertain what functions and how many functions to include, let alone the minimal number of functions, to ensure $\mathcal{K}$-invariant. Recently, EDMD [22] uses an expansive set of orthonomal polynomial basis functions, but this approach does not scale well and suffers from overfitting with an increasing number of basis functions. Deep-DMD [10] adopts the neutral networks to approximate the basis function based on universal approximation theorem, but it lacks the theoretical guarantee, e.g., the stability and convergence. Second, the objective function is nonconvex. Therefore it is unrealistic to expect an algorithm to converge to global minima.

Here we focus on the basis function based on parametric method, specifically, $\psi(x_n) = \psi(\mathcal{W}, x_n)$. A typical example is a fully connected one-layer neutral network since for wide shallow neutral network, local optima provide satisfactory [18]–[20], where $\mathcal{W}$ is the layer parameter and ψ is the activation function. With the parametric basis method, problem (5) becomes

$$\min_{\mathcal{W},\mathcal{K}} \ \mathcal{F}(\mathcal{W}, \mathcal{K}) := \frac{1}{2N}\sum_{i=1}^{N} \|\psi(\mathcal{W}, x_{i+1}) - \mathcal{K}\psi(\mathcal{W}, x_i)\|_2^2. \tag{6}$$

Although this problem is nonconvex, there are some interesting structures. For example, if we fix the parameter $\mathcal{W}$ of the basis function, optimizing $\mathcal{K}$ is a quadratic problem that finds the linear mapping $\mathbb{R}^m$ to $\mathbb{R}^m$. On the other hand, with fixed $\mathcal{K}$, optimizing $\mathcal{W}$ is to adjust the parameter $\mathcal{W}$ to find the function space that satisfies the linear transformation mapping (this is still nonconvex but will reduce the complexity a lot). We thus consider the algorithm that alternatively optimize over $\mathcal{W}$ and $\mathcal{K}$.

III. Alternating Optimization Algorithm

In this section, we first states our alternating algorithm and then investigate its convergence properties. Let $\mathcal{F}_t = \mathcal{F}(\mathcal{W}^t, \mathcal{K}^t)$ and denote by $\|\cdot\|_F$ the Frobenius norm. The detail of the algorithm is shown in Algorithm 1. Here $\mathcal{E}$ measures how far the gradient is from that at the critical point and $\mathcal{K}^*, \mathcal{W}^*$ track the best parameters so far. We make the following assumptions.

Assumption 1: The basis function $\psi(\cdot)$ is bounded and has a bounded gradient and Hessian.

Assumption 2: The parameters $\mathcal{K}$ and $\mathcal{W}$ are bounded, e.g., there exist two constant $U_\mathcal{K}$ and $U_\mathcal{W}$ such that $\|\mathcal{K}\|_F \leq U_\mathcal{K}$ and $\|\mathcal{W}\|_F \leq U_\mathcal{W}$.

Assumption 1 looks strong. However, it holds for several popular activation functions such as logistic function $(\frac{1}{1+e^{-x}})$, hyperbolic tangent $(\tanh(x))$, and inverse hyperbolic tangent $\arctan(x)$. Based Assumptions 1 and 2, one

Manuscript 1236 submitted to 2020 American Control Conference.
Received September 26, 2019.

Algorithm 1: Alternating Operator Koopman Learning With Tracking

Initialization: randomly initialize $\mathcal{W}^0$ and $\mathcal{K}^0$, $\mathcal{E}^0 = \|\nabla_{\mathcal{K}}\mathcal{F}_0\|_F + \|\nabla_{\mathcal{W}}\mathcal{F}_0\|_F$,$\mathcal{W}^* = \mathcal{W}^0$,$\mathcal{K}^* = \mathcal{K}^0$.
while *Not Converge* **do**
  $\mathcal{K}^{t+1} = \mathcal{K}^t - \eta_{\mathcal{K}}\nabla_{\mathcal{K}}\mathcal{F}(\mathcal{W}^t, \mathcal{K}^t)$,
  $\mathcal{W}^{t+1} = \mathcal{W}^t - \eta_{\mathcal{W}}\nabla_{\mathcal{W}}\mathcal{F}(\mathcal{W}^t, \mathcal{K}^{k+1})$.
  $\mathcal{E}^{t+1} = \|\nabla_{\mathcal{K}}\mathcal{F}_{t+1}\|_F + \|\nabla_{\mathcal{W}}\mathcal{F}_{t+1}\|_F$.
  if $\mathcal{E}^{t+1} \leq \mathcal{E}^t$ **then**
    $\mathcal{K}^* = \mathcal{K}^{t+1}$; $\mathcal{W}^* = \mathcal{W}^{t+1}$
  end
end

can verify that the objective function $\mathcal{F}$ is bounded, i.e., there exists a constant R such that $\mathcal{F} \leq R$. We can show that $\mathcal{F}$ has Lipschitz-continuous gradient with respect to the parameter $\mathcal{W}$ of basis functions.

Lemma 1: Under Assumptions 1 and 2 and given the data trajectory $\{(x_i, x_{i+1})\}_{i=1}^N$, we have

$$\|\nabla_{\mathcal{W}}\mathcal{F}(\mathcal{W}^1, \mathcal{K}) - \nabla_{\mathcal{W}}\mathcal{F}(\mathcal{W}^2, \mathcal{K})\|_F \leq L_{\mathcal{W}}\|\mathcal{W}^1 - \mathcal{W}^2\|_F$$

with

$$L_{\mathcal{W}} = \sqrt{2d}U_{\mathcal{K}}L_\Psi\frac{\sum_{i=1}^N \|x_i\|_2\Delta_i}{N},$$

where $\Delta_i = \sqrt{(1+dU_{\mathcal{K}}^2)\|x_i\|_2^2 + \|x_{i+1}\|_2^2}$ and L_Ψ is the Lipschitz constant for the function $\Psi(x_1, x_2) := \psi(x_1)\psi'(x_2)$.

Proof: First, denote by $\mathcal{K}[:, i]$ the i-th column of matrix $\mathcal{K}$, $\mathcal{W}_j$ the j-th row of matrix $\mathcal{W}$, and $x_i[k]$ the k-th dimension of x_i. We can compute the element $[j, k]$ of $\nabla_{\mathcal{W}}\mathcal{F}(\mathcal{W}, \mathcal{K})$ as:

$$\begin{aligned}&\nabla_{\mathcal{W}}\mathcal{F}(\mathcal{W}, \mathcal{K})[j, k]\\ =&\frac{1}{N}\sum_{i=1}^N -(\psi(\mathcal{W}x_{i+1}) - \mathcal{K}\psi(\mathcal{W}x_i))^T\mathcal{K}[:, j]\psi'(\mathcal{W}x_i)x_i[k],\end{aligned}$$

and $\nabla_{\mathcal{W}}\mathcal{F}(\mathcal{W}, \mathcal{K})$ as:

$$\begin{aligned}&\nabla_{\mathcal{W}}\mathcal{F}(\mathcal{W}, \mathcal{K})\\ =&-\frac{1}{N}\sum_{i=1}^N \mathcal{K}^T\underbrace{(\psi(\mathcal{W}x_{i+1}) - \mathcal{K}\psi(\mathcal{W}x_i))\odot\psi'(\mathcal{W}x_i)}_{\alpha_i^{\mathcal{W}}}x_i^T\\ =&-\frac{1}{N}\sum_{i=1}^N \mathcal{K}^T\alpha_i^{\mathcal{W}}x_i^T,\end{aligned}$$

where $\odot$ denotes the element-wise production. We can then write the gradient difference with respect to $\mathcal{W}^1$ and $\mathcal{W}^2$ as

$$\begin{aligned}&\|\nabla_{\mathcal{W}}\mathcal{F}(\mathcal{W}^1, \mathcal{K}) - \nabla_{\mathcal{W}}\mathcal{F}(\mathcal{W}^2, \mathcal{K})\|_F\\ =&\frac{1}{N}\sum_{i=1}^N \|\mathcal{K}^T(\alpha_i^{\mathcal{W}^1} - \alpha_i^{\mathcal{W}^2})x_i^T\|_F\\ \leq&\frac{1}{N}\sum_{i=1}^N \|\mathcal{K}\|_F\|(\alpha_i^{\mathcal{W}^1} - \alpha_i^{\mathcal{W}^2})x_i^T\|_F\\ \leq&\frac{1}{N}\sum_{i=1}^N \|\mathcal{K}\|_F\|\alpha_i^{\mathcal{W}^1} - \alpha_i^{\mathcal{W}^2}\|_2\|x_i\|_2.\end{aligned}$$

So if we can show that $\alpha_i^{\mathcal{W}}$ is Lipschitz-continuous, the proof is done. We have

$$\begin{aligned}&\alpha_i^{\mathcal{W}^1}[j] - \alpha_i^{\mathcal{W}^2}[j]\\ =&\underbrace{(\psi(\mathcal{W}_j^1x_{i+1})\psi'(\mathcal{W}_j^1x_i) - \psi(\mathcal{W}_j^2x_{i+1})\psi'(\mathcal{W}_j^2x_i))}_{\beta_j^i}\\ &-\Big(\underbrace{\mathcal{K}_j\psi(\mathcal{W}^1x_i)\psi'(\mathcal{W}_j^1x_i) - \mathcal{K}_j\psi(\mathcal{W}^2x_i)\psi'(\mathcal{W}_j^2x_i)}_{\gamma_j^i}\Big).\end{aligned}$$

Consider function $\Psi(x_1, x_2) = \psi(x_1)\psi'(x_2)$ in $x_1, x_2 \in \mathbb{R}$ and its gradient $\nabla\Psi(x_1, x_2) = \begin{bmatrix}\psi'(x_1)\psi'(x_2)\\ \psi(x_1)\psi''(x_2)\end{bmatrix}$. By Assumption 1, $\|\nabla\Psi(\cdot)\|_2$ is bounded by some constant, denoted by L_Ψ. Let $\beta^i = [\beta_1^i, \cdots, \beta_n^i]^T$, we can bound β^i as follows:

$$\begin{aligned}\|\beta^i\|_2^2 =&\|\psi(\mathcal{W}^1x_{i+1})\odot\psi'(\mathcal{W}^1x_i) - \psi(\mathcal{W}^2y_i)\odot\psi'(\mathcal{W}^2x_i)\|_2^2\\ \leq& 2L_\Psi^2\Big(\|\mathcal{W}^1x_{i+1} - \mathcal{W}^2x_{i+1}\|_2^2 + \|\mathcal{W}^1x_i - \mathcal{W}^2x_i\|_2^2\Big)\\ \leq& 2L_\Psi^2(\|x_i\|_2^2 + \|x_{i+1}\|_2^2)\|\mathcal{W}^1 - \mathcal{W}^2\|_F^2,\end{aligned}$$

where the last inequality is due to Cauchy-Schwarz inequality. Similarly, we can bound γ^i as follows:

$$\begin{aligned}\|\gamma^i\|_2^2\leq&\sum_{j=1}^n \|\mathcal{K}_j\|_2^2L_\Psi^2\Big(\sum_{k=1}^d(\mathcal{W}_k^1x_i - \mathcal{W}_k^2x_i)^2 + (\mathcal{W}_j^1x_i - \mathcal{W}_j^2x_i)^2\Big)\\ \leq&\sum_{j=1}^d \|\mathcal{K}_j\|_2^2L_\Psi^2\Big(\|\mathcal{W}^1x_i - \mathcal{W}^2x_i\|_2^2 + d(\mathcal{W}_j^1x_i - \mathcal{W}_j^2x_i)^2\Big)\\ \leq&(dU_{\mathcal{K}}^2 + d^2\|\mathcal{K}_j^{\max}\|_2^2)L_\Psi^2\|\mathcal{W}^1x_i - \mathcal{W}^2x_i\|_2^2\\ \leq&(dU_{\mathcal{K}}^2 + d^2\|\mathcal{K}_j^{\max}\|_2^2)L_\Psi^2\|x_i\|_2^2\|\mathcal{W}^1 - \mathcal{W}^2\|_F^2\\ \leq&2dU_{\mathcal{K}}^2L_\Psi^2\|x_i\|_2^2\|\mathcal{W}^1 - \mathcal{W}^2\|_F^2,\end{aligned}$$

where the second inequality is by Assumption 2. Combining the above results, we have

$$\begin{aligned}&\|\nabla_{\mathcal{W}}\mathcal{F}(\mathcal{W}^1, \mathcal{K}) - \nabla_{\mathcal{W}}\mathcal{F}(\mathcal{W}^2, \mathcal{K})\|_F\\ \leq&\frac{1}{N}\sum_{i=1}^N \|\mathcal{K}\|_F\|\alpha_i^{\mathcal{W}^1} - \alpha_i^{\mathcal{W}^2}\|_2\|x_i\|_2\\ \leq&\sqrt{2d}\|\mathcal{K}\|_FL_\Psi\frac{\sum_{i=1}^N \|x_i\|_2\Delta_i}{N}\|\mathcal{W}^1 - \mathcal{W}^2\|_F,\end{aligned}$$

where $\Delta_i = \sqrt{(1+dU_{\mathcal{K}}^2)\|x_i\|_2^2 + \|x_{i+1}\|_2^2}$. ■

Similarly, $\mathcal{F}$ has Lipschitz-continuous gradient with respect to the parameter $\mathcal{K}$ of the linear mapping.

Manuscript 1236 submitted to 2020 American Control Conference.
Received September 26, 2019.

Lemma 2: Under Assumption 1 and assume that the basis function is bounded by h, we have

$$\|\nabla_{\mathcal{K}}\mathcal{F}(\mathcal{W},\mathcal{K}^1)-\nabla_{\mathcal{K}}\mathcal{F}(\mathcal{W},\mathcal{K}^2)\|_F \le L_{\mathcal{K}}\|\mathcal{K}^1-\mathcal{K}^2\|_F$$

with $L_{\mathcal{K}} = dh^2$.

Proof: The gradient

$$\nabla_{\mathcal{K}}\mathcal{F}(\mathcal{W},\mathcal{K}) = \frac{1}{N}\sum_{i=1}^{N}(\mathcal{K}\psi(\mathcal{W}x_i)-\psi(\mathcal{W}x_{i+1}))\psi(\mathcal{W}x_i)^T,$$

and

$$\begin{aligned}
&\|\nabla_{\mathcal{K}}\mathcal{F}(\mathcal{W},\mathcal{K}^1)-\nabla_{\mathcal{K}}\mathcal{F}(\mathcal{W},\mathcal{K}^2)\|_F\\
=&\frac{1}{N}\|\sum_{i=1}^{N}(\mathcal{K}^1-\mathcal{K}^2)\psi(\mathcal{W}x_i)\psi(\mathcal{W}x_i)^T\|_F\\
\le&\frac{1}{N}\|\mathcal{K}^1-\mathcal{K}^2\|_F\|\sum_{i=1}^{N}\psi(\mathcal{W}x_i)\psi(\mathcal{W}x_i)^T\|_F\\
\le&\frac{1}{N}\|\mathcal{K}^1-\mathcal{K}^2\|_F\sqrt{d^2(Nh^2)^2}\\
=&dh^2\|\mathcal{K}^1-\mathcal{K}^2\|_F.
\end{aligned}$$

■

With Lemmas 1 and 2, we now show that Algorithm 1 will converge to a critical point with convergence rate $O(\frac{1}{T})$ or $O(\frac{1}{\log T})$.

Theorem 1: Under Assumptions 1 and 2, Algorithm 1 for Koopman operator learning will converge to a critical point. With constant learning rate $\eta \le \min(\frac{2}{L_{\mathcal{W}}},\frac{2}{L_{\mathcal{W}}})$, its convergence rate is $O(\frac{1}{T})$; and with diminishing learning rate $\eta_t = \frac{1}{t+1}$, its convergence rate is $O(\frac{1}{\log T})$.

Proof: Since the objective function is Lipschitz gradient continuous with respect to $\mathcal{K}$, the descent lemma [23] can be applied and we have

$$\begin{aligned}
&\mathcal{F}(\mathcal{W}^t,\mathcal{K}^{t+1})\\
\le&\mathcal{F}(\mathcal{W}^t,\mathcal{K}^t)+(\nabla_{\mathcal{K}}\mathcal{F}(\mathcal{W}^t,\mathcal{K}^t)^T(\mathcal{K}^{t+1}-\mathcal{K}^t))\\
&+\frac{L_{\mathcal{K}}}{2}\|\mathcal{K}^{t+1}-\mathcal{K}^t\|_F^2\\
=&\mathcal{F}(\mathcal{W}^t,\mathcal{K}^t)-\eta_{\mathcal{K}}(\nabla_{\mathcal{K}}\mathcal{F}(\mathcal{W}^t,\mathcal{K}^t)^T\nabla_{\mathcal{K}}\mathcal{F}(\mathcal{W}^t,\mathcal{K}^t))\\
&+\frac{L_{\mathcal{K}}}{2}\|\mathcal{K}^{t+1}-\mathcal{K}^t\|_F^2\\
=&\mathcal{F}(\mathcal{W}^t,\mathcal{K}^t)+\left(\frac{\eta_{\mathcal{K}}^2L_{\mathcal{K}}}{2}-\eta_{\mathcal{K}}\right)\|\nabla_{\mathcal{K}}\mathcal{F}(\mathcal{W}^t,\mathcal{K}^t)\|_F^2, \quad (7)
\end{aligned}$$

where denotes the trace of the matrix. The first equality is due to the gradient update of $\mathcal{K}^t$ and the second equality is by the fact that $(A^TA) = \|A\|_F^2$.

As for the basis function's parameter $\mathcal{W}$, we can have the similar result since the objective function is Lipschitz gradient continuous with respect to $\mathcal{W}$.

$$\begin{aligned}
&\mathcal{F}(\mathcal{W}^{t+1},\mathcal{K}^{t+1})\\
\le&\mathcal{F}(\mathcal{W}^t,\mathcal{K}^{t+1})+\left(\frac{\eta_{\mathcal{W}}^2L_{\mathcal{W}}}{2}-\eta_{\mathcal{W}}\right)\|\nabla_{\mathcal{W}}\mathcal{F}(\mathcal{W}^t,\mathcal{K}^{t+1})\|_F^2. \quad (8)
\end{aligned}$$

So by the equation (7) and (8), we have the following for each complete update from $(\mathcal{W}^t,\mathcal{K}^t)\to(\mathcal{W}^{t+1},\mathcal{K}^{t+1})$.

$$\begin{aligned}
&\mathcal{F}(\mathcal{W}^{t+1},\mathcal{K}^{t+1})\\
\le&\mathcal{F}(\mathcal{W}^t,\mathcal{K}^t)+\left(\frac{\eta_{\mathcal{K}}^2L_{\mathcal{K}}}{2}-\eta_{\mathcal{K}}\right)\|\nabla_{\mathcal{K}}\mathcal{F}(\mathcal{W}^t,\mathcal{K}^t)\|_F^2\\
&+\left(\frac{\eta_{\mathcal{W}}^2L_{\mathcal{W}}}{2}-\eta_{\mathcal{W}}\right)\|\nabla_{\mathcal{W}}\mathcal{F}(\mathcal{W}^t,\mathcal{K}^{t+1})\|_F^2. \quad (9)
\end{aligned}$$

We sum both sides of inequality (9) from $t=1,\cdots,T$ and obtain

$$\begin{aligned}
&\mathcal{F}(\mathcal{W}^{T+1},\mathcal{K}^{T+1})\\
\le&\mathcal{F}(\mathcal{W}^0,\mathcal{K}^0)+\sum_{t=0}^{T}\left(\frac{\eta_{\mathcal{K}}^2L_{\mathcal{K}}}{2}-\eta_{\mathcal{K}}\right)\|\nabla_{\mathcal{K}}\mathcal{F}(\mathcal{W}^t,\mathcal{K}^t)\|_F^2\\
&+\sum_{t=0}^{T}\left(\frac{\eta_{\mathcal{W}}^2L_{\mathcal{W}}}{2}-\eta_{\mathcal{W}}\right)\|\nabla_{\mathcal{W}}\mathcal{F}(\mathcal{W}^t,\mathcal{K}^{t+1})\|_F^2 \quad (10)
\end{aligned}$$

(1) Constant learning rate

If we choose the constant stepsize, e.g., $\eta_{\mathcal{W}} = \eta_{\mathcal{K}} = \eta$, $0<\eta<\min(\frac{2}{L_{\mathcal{W}}},\frac{2}{L_{\mathcal{K}}})$ and let $L = \max(\eta_{\mathcal{W}},\eta_{\mathcal{K}})$, $S = \eta-\frac{L\eta^2}{2}$, we can bound the gradients as follows.

$$\begin{aligned}
&\sum_{t=0}^{T}\|\nabla_{\mathcal{K}}\mathcal{F}(\mathcal{W}^t,\mathcal{K}^t)\|_F^2+\|\nabla_{\mathcal{W}}\mathcal{F}(\mathcal{W}^t,\mathcal{K}^{t+1})\|_F^2\\
\le&\frac{(\mathcal{F}(\mathcal{W}^0,\mathcal{K}^0)-\mathcal{F}(\mathcal{W}^{T+1},\mathcal{K}^{T+1}))}{S}. \quad (11)
\end{aligned}$$

One can see that, each term on the right is non-negative and their summation is bounded by some constant. Based on current analysis, we can conclude that the alternating optimization algorithm will converge asymptotically to one critical point even **without** optimal tracker $\mathcal{K}^*$ and $\mathcal{W}^*$ when $T\to\infty$.

Based on inequality (11), one can bound the minimum gradients up to T for Algorithm 1.

$$\begin{aligned}
&\min_{t=0,\cdots,T}\|\nabla_{\mathcal{K}}\mathcal{F}(\mathcal{W}^t,\mathcal{K}^t)\|_F^2+\|\nabla_{\mathcal{W}}\mathcal{F}(\mathcal{W}^t,\mathcal{K}^{t+1})\|_F^2\\
\le&\frac{(\mathcal{F}(\mathcal{W}^0,\mathcal{K}^0)-\mathcal{F}(\mathcal{W}^{T+1},\mathcal{K}^{T+1}))}{ST}\le\frac{2R}{ST}
\end{aligned}$$

(2) Diminishing learning rate

If we choose the diminishing learning rate, e.g, $\eta^t = \frac{1}{t+1}$, the result becomes

$$\begin{aligned}
&\min_{t=0,\cdots,T}\|\nabla_{\mathcal{K}}\mathcal{F}(\mathcal{W}^t,\mathcal{K}^t)\|_F^2+\|\nabla_{\mathcal{W}}\mathcal{F}(\mathcal{W}^t,\mathcal{K}^{t+1})\|_F^2\\
\le&\frac{(\mathcal{F}(\mathcal{W}^0,\mathcal{K}^0)-\mathcal{F}(\mathcal{W}^{T+1},\mathcal{K}^{T+1}))}{\sum_{t=0}^{T}\left(\eta^t-\frac{L(\eta^t)^2}{2}\right)}.
\end{aligned}$$

We know

$$\begin{aligned}
&\sum_{t=0}^{T}\left(\eta^t-\frac{L(\eta^t)^2}{2}\right)=\sum_{t=0}^{T}\left(\frac{1}{t+1}-\frac{L}{2(t+1)^2}\right)\\
\ge&\ln(T+2)-\frac{L}{2}-\sum_{t=1}^{T}\frac{L}{2t(t+1)}=\ln(T+2)-L+\frac{L}{2(T+1)}. \quad (12)
\end{aligned}$$

Manuscript 1236 submitted to 2020 American Control Conference.
Received September 26, 2019.

So for diminishing stepsize, we can obtain

$$\min_{t=0,\cdots,T} \|\nabla_{\mathcal{K}}\mathcal{F}(\mathcal{W}^t,\mathcal{K}^t)\|_F^2 + \|\nabla_{\mathcal{W}}\mathcal{F}(\mathcal{W}^t,\mathcal{K}^{t+1})\|_F^2 \leq \frac{2R}{O(\ln T)}.$$

So we can see for this problem, the constant stepsize has the better convergence rate than the diminishing stepsize with the help of optimal tracker. Both cases show that

$$\|\nabla_{\mathcal{K}}\mathcal{F}(\mathcal{W}^*,\mathcal{K}^*)\|_F^2 + \|\nabla_{\mathcal{W}}\mathcal{F}(\mathcal{W}^*,\mathcal{K}^*)\|_F^2 \to 0.$$

■

IV. Distributed Koopman Learning

We now develop an algorithm to treat the learning problem for Koopman operators of high dimensional nonlinear dynamical systems. Even if there are only a thousand states in the underlying nonlinear system, the dimension of the dictionary functions explodes exponentially with the number of states. Memory constraints thus make it infeasible to train a Koopman operator using a centralized or stand-alone computing node. This motivates the derivation of a scalable, distributed approximation algorithm to relieve this problem.

Assumption 3: The basis function $\psi(x)$ can be decomposed or approximated by $[\psi_1(x^1),\cdots,\psi_q(x_q)]^\top$, where $\psi_i : \mathbb{R}^{d_i} \to \mathbb{R}^{m_i}$ is the new basis function for x^i and x_i is a subset of x with $x = [x^1,\cdots,x^q]^\top$.

Based on Assumption 3, we can reformulate the centralized Koopman objective function as

$$\mathcal{F}(\mathcal{W},\mathcal{K}) = \frac{1}{2N}\sum_{i=1}^{N}\left\|\begin{bmatrix}\psi_1(x_{i+1}^1;\mathcal{W}_1)\\ \vdots \\ \psi_q(x_{i+1}^q;\mathcal{W}_q)\end{bmatrix} - \begin{bmatrix}\mathcal{K}_{11}\cdots\mathcal{K}_{1q}\\ \vdots\ \ddots\ \vdots \\ \mathcal{K}_{q1}\cdots\mathcal{K}_{qq}\end{bmatrix}\begin{bmatrix}\psi_1(x_i^1;\mathcal{W}_1)\\ \vdots \\ \psi_q(x_i^q;\mathcal{W}_q)\end{bmatrix}\right\|_F^2$$

Our distributed Koopman learning's structure is as follows. Denote by $\mathcal{Q} = [1,\cdots,q]$ the set of computation nodes which can communicate with each other. For each computation node $i \in \mathcal{Q}$, it only store part of the data set $\{(x_j^i, x_{j+1}^i)|j = 1,\cdots N\}$, its corresponding row and column of Koopman operator $\{\mathcal{K}_{ij},\mathcal{K}_{ji}|j \in \mathcal{Q}\}$ and its basis function ψ_i.

For node i, its gradient will compose two parts. The first part can be calculated based on its own knowledge. Another part needs the information from other nodes. We first define by $e^i \in \mathbb{R}^{m_i}$ the error term for node i with data point (x_j^i, x_{j+1}^i), where

$$e_j^i = \psi_i(x_{j+1}^i;\mathcal{W}_i) - \begin{bmatrix}\mathcal{K}_{i1},\cdots,\mathcal{K}_{iq}\end{bmatrix}\begin{bmatrix}\psi_1(x_j^1;\mathcal{W}_1)\\ \vdots \\ \psi_q(x_j^q;W_q)\end{bmatrix},$$

and define by $J(\psi_i(;\mathcal{W}_i)) \in \mathbb{R}^{m_i\times d_i}$ the Jacobi matrix of function ψ_i, we then have the following distributed Koopman learning algorithm shown in Algorithm 2.

For our distributed Koopman operator learning Algorithm 2, line 6-8 is the communication stage, each computation node i calculates its result in the lifted dimensional space and broadcast within our communication network. After the communication, the information is enough to compute local error term e_j^i, and node i send $S_{iv}^{'}$ to node v (line 8). Here the communication stage ends and computation stage (line 9-11) begins. A_i, B_i, C_i will sum up all the information for each data point. The last is update stage with gradient descent method (line 14-15). Based on this distributed algorithm and Assumption 3, we can prove this is equivalent to the centralized gradient descent algorithm.

Algorithm 2: Distributed Koopman Operator Learning

1 **Initialization:**
2 node i randomly initilizes its W_i, θ_{ik} $\forall i,k \in \mathcal{Q}$.
3 **while** *Not Converge* **do**
4 $A_i = 0, B_i = 0, C_i = 0, \forall i \in \mathcal{Q}$.
5 **for** $j = 1; j \leq N; j = j+1$ **do**
6 node i calculates $\psi_i(x_j^i;\mathcal{W}_i)$,$\psi_i(x_{j+1}^i;\mathcal{W}_i)$,$\forall i \in \mathcal{Q}$.
7 node i broadcasts $S_j^i = \psi_i(x_j^i;\mathcal{W}_i)$,$\forall i \in \mathcal{Q}$.
8 node i calculates e_j^i, sends $S_{iv}^{'} = \mathcal{K}_{iv}^T e_j^i$ to node v, $\forall i,v \in \mathcal{Q}$.
9 $A_i = A_i + J(\psi_i(x_{j+1}^i;\mathcal{W}_i))^\top e_j^i, \forall i \in \mathcal{Q}$
10 $B_i = B_i - J(\psi_i(x_j^i;\mathcal{W}_i))^\top \sum_{k\in\mathcal{Q}} S_{ki}^{'}, \forall i \in \mathcal{Q}$.
11 $C_i = C_i + e_j^i[S_j^1,\cdots,S_j^q]$
12 **end**
13 **for** $i \in \mathcal{Q}$ **do**
14 $\mathcal{W}_i \leftarrow \mathcal{W}_i - \eta_{\mathcal{W}_i}\frac{1}{N}(A_i + B_i)$.
15 $[\mathcal{K}_{i1},\cdots,\mathcal{K}_{iq}] \leftarrow [\mathcal{K}_{i1},\cdots,\mathcal{K}_{iq}] - \frac{\eta_{\mathcal{K}_i}}{N}C_i$
16 **end**
17 **end**

Lemma 3: Under Assumption 1, the distributed Koopman learning in Algorithm 2 is equivalent to the following update:

$$\begin{aligned}\mathcal{K}^{t+1} &= \mathcal{K}^t - \eta_{\mathcal{K}}\nabla_{\mathcal{K}}\mathcal{F}(\mathcal{W}^t,\mathcal{K}^t),\\ \mathcal{W}^{t+1} &= \mathcal{W}^t - \eta_{\mathcal{W}}\nabla_{\mathcal{W}}\mathcal{F}(\mathcal{W}^t,\mathcal{K}^k).\end{aligned} \tag{13}$$

Proof: Based on line 9-11 in Algorithm 2, one can verify the following after updating all the data points

$$\begin{aligned}A_i &= \sum_{j=1}^{N} J(\psi_i(x_{j+1}^i;\mathcal{W}_i))^\top e_j^i,\\ B_i &= -\sum_{j}^{N} J(\psi_i(x_i;\mathcal{W}_i))^\top \sum_{k\in\mathcal{Q}} \mathcal{K}_{iv}^T e_j^i,\\ C_i &= \sum_{j}^{N} e_j^i[S_j^1,\cdots,S_j^q].\end{aligned}$$

Compared to gradients of $\mathcal{F}(\mathcal{W},\mathcal{K})$, we can find that

$$\begin{aligned}\frac{1}{N}(A_i + B_i) &= \nabla_{\mathcal{W}}\mathcal{F}(\mathcal{W}^t,\mathcal{K}^k),\\ \frac{1}{N}C_i &= \nabla_{\mathcal{K}}\mathcal{F}(\mathcal{W}^t,\mathcal{K}^t).\end{aligned}$$

Manuscript 1236 submitted to 2020 American Control Conference.
Received September 26, 2019.

So the update stage (line 14-15) is the same with equation (13) which finishes the proof. ■

Remark 1: Our alternating Koopman operator learning (Algorithm 1) can be regarded as nonlinear Gauss-Seidel iterations [24], while our distributed Koopman operator learning lies in the model of nonlinear Jacobi iteration [24]. Here we choose nonlinear Jacobi iteration for distributed Koopman operator learning due to that nonlinear Jacobi iteration is **(1)** suitable for parallel computation and **(2)** with less communication overhead.

Based on lemma 3, we can get the convergence result for our distributed Koopman operator learning.

Theorem 2: Under Assumption 1,2 and 3, the distributed Koopman operator learning based on Algorithm 2 will converge to one critical point asymptotically.

Proof: The equation (13) is the case without optimal tracker of Algorithm 1. The proof of theorem 1 can be applied directly here (equation (11)). ■

The advantages of distributed Koopman learning over centralized one are not only the scalability, e.g., the ability to handle the high dimensional nonlinear dynamics, but also the feasibility to adjust to different complexity of the partial observations. For example, if one partial observation (x_j^i, x_{j+1}^i) is with complexity dynamic, we can increase the number of basis function.

On the other hand, algorithm 2 for distributed Koopman learning is under the ideal synchronous model. Although the computation of each node i is parallel, the computation will not start until the broadcast process finishes. This can harm heavily on the efficiency of the distributed Koopman learning due to some reason, e.g., if one node has a very low link speed, all the other need wait for this node. Also, one packet loss will lead to all nodes waiting for the resending. However, it is easily to extend Algorithm 2 to handle asynchronous model as displayed in Algorithm 3. Each node will store the received information $(S_j^i, S_{iv}^{'})$ in its memory keeping updating when new information comes. once the computation node comes to computation stage, it will directly use the information stored in the memory instead of waiting for the newest information.

Algorithm 3: Asynchronous Distributed Koopman Learning

1 lines 1-7 of Algorithm 2.
2 node i calculates e_j^i based on the current S_l in the memory, sends $S_{iv}^{'} = \mathcal{K}_{iv}^T e_j^i$ to node v, $\forall i, v, l \in \mathcal{Q}$.
3 $A_i \leftarrow A_i + J(\psi_i(y_j^i; \mathcal{W}_i)^\top e_i, \forall i \in \mathcal{Q}$
4 $B_i \leftarrow B_i - J(h_i(x_i); W_i)^\top \sum_{k \in \mathcal{Q}} S_{ki}^{'}$ based on the current S_{ki} in the memory, $\forall i \in \mathcal{Q}$.
5 $C_i \leftarrow C_i + e_j^i[S_1, \cdots, S_q]$ based on the current S_l in the memory, $\forall l \in \mathcal{Q}$.
6 lines 13-16 of Algorithm 2.

For the asynchronous version of gradient descent algorithm, [25] (Theorem 2), [26] (Proposition 2.1), [27] (Theorem 7) e.t.c. show that synchronous and asynchronous algorithm will converge to the same point once the communication delay is bounded. Their proof can be applied to our case with slight change.

Lemma 4: ([25], [26], [27]) If the communication delay is bounded by some constant, with small enough stepsize, asynchronous algorithm 3 will asymptotically converge to the same point with synchronous one.

V. Experiments

In our experiment, we evaluate the performance of our alternating optimization and distributed algorithms respectively. At each experiment, we sample some points from the real trajectory to prepare the training set and prediction set. Note that our prediction phase is multi-step prediction, e.g, given one initial state, our algorithm will predict the following trajectory using the $\mathcal{K}$-invariant property: $\psi(x_n) = \mathcal{K}^n \psi(x_0)$.

To evaluate the performance of alternating optimization, we consider Van der Pol oscillator shown in Example 1.

Example 1: Van der Pol oscillator

$$\dot{x_1} = \mu \left(x_1 - \frac{1}{3} x_1^3 - x_2 \right) \tag{14}$$

$$\dot{x_2} = \frac{1}{\mu} x_1 \tag{15}$$

In this example, we choose $\mu = 0.5$. The number of date points we sampled is 600 with 400 points for training and 200 for prediction. We construct a very simple network with one layer and 3 dimensions to learn the pattern of Van der Pol oscillator. The total training time is around 1.08s (i7-8700 CPU @ 3.20 GHz, 8 GB RAM) with 500 iterations and constant stepsize is 0.23. Fig. 1 shows the multi-step prediction result with alternating optimization method. One step prediction error is around 0.16% and 200 step prediction error is around 1.89%.

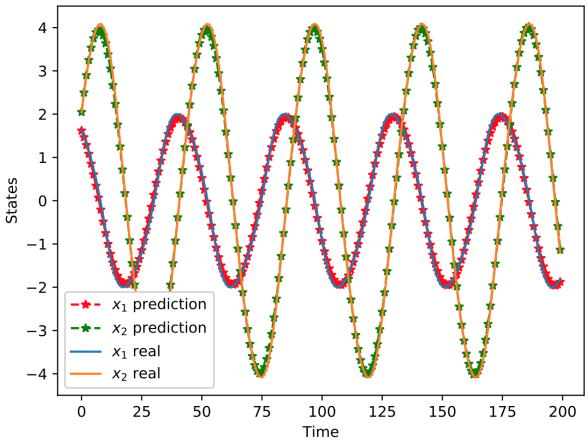

Fig. 1: Alternating optimization for centralized Koopman operator learning with Van der Pol oscillator. In this experiments, only the points at time 0 are given. All the data points [1-200] are our predictions with Koopman learning.

Example 2: Glycolytic pathway

$$\dot{x_1} = J - \frac{k_1 x_1 x_6}{1 + \left(\frac{x_6}{k_1}\right)^q} \tag{16}$$

$$\dot{x_2} = \frac{2k_1 x_1 x_6}{1 + \left(\frac{x_6}{k_1}\right)^q} - k_2 x_2 (n - x_5) - k_6 x_2 x_5 \tag{17}$$

$$\dot{x_3} = k_2 x_2 (n - x_5) - k_3 x_3 (a - x_6) \tag{18}$$

$$\dot{x_4} = k_3 x_3 (a - x_6) - k_4 x_4 x_5 - \kappa (x_4 - x_7) \tag{19}$$

$$\dot{x_5} = k_2 x_2 (n - x_5) - k_4 x_4 x_5 - k_6 x_2 x_5 \tag{20}$$

$$\dot{x_6} = -\frac{2k_1 x_1 x_6}{1 + \left(\frac{x_6}{k_1}\right)^q} + 2k_3 x_3 (a - x_6) - k_5 x_6 \tag{21}$$

$$\dot{x_7} = \phi\kappa (x_4 - x_7) - k x_7 \tag{22}$$

Our distributed Koopman operator learning is implemented on a larger nonlinear dynamical system displayed in Example 2, namely the glycolysis network from cellular biology [28]. We adopt the parameter setting: $J = 2.5, a = 4, n = 1, k_1 = 0.52, \kappa = 13, \phi = 0.1, q = 4, k = 1.8, k_1 = 100, k_2 = 6, k_3 = 16, k_4 = 100, k_5 = 1.28, k_6 = 12$ from [28]. 1000 data points are sampled from the real trajectory with 900 points for training and 100 points for prediction. We create 7 threads to simulate the distributed learning and each thread only learn the dynamic pattern of one state by a simple 3-layer neutral network with 15 dimensions. The total training time is 78.4s with 6000 iterations. Results for each state is shown in Fig. 2. One step error is around 0.34% and 100 step prediction error is around 7.6%.

As we can see from the experiments, our alternating optimization and distributed algorithms both achieve good performance with multi-step prediction. Even though partial state measurements are provided for training, the trained distributed Koopman operator is able to predict the behavior of the glycolysis network over 100 steps. Further, these results provide a glimmer of hope for whole cell network modeling, using strategically placed reporter libraries that provide partial measurements of an entire transcriptome [21], [29].

VI. Conclusion

In this paper, we have proposed an alternating optimization algorithm to the nonconvex Koopman operator learning problem for nonlinear dynamic systems. We prove that the proposed algorithm will converge to a critical point with rate $O(1/T)$ or $O(\frac{1}{\log T})$ under some mild assumptions. To handle the high dimensional nonlinear dynamical systems, we have further proposed a distributed Koopman operator learning algorithm with appropriate communication mechanism. We show that the distributed Koopman operator learning is of the same convergence property with centralized one if the basis functions are decomposable. Experiments are provided to complement our theoretical results.

VII. Acknowledgments

We thank Igor Mezic, Nathan Kutz, Robert Egbert, Bassam Bamieh, Sai Nandanoori Pushpak, Sean Warnick, Jongmin Kim, Umesh Vaidya, and Erik Bollt for stimulating conversations. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA), the Department of Defense, or the United States Government. This work was supported partially by a Defense Advanced Research Projects Agency (DARPA) Grant No. DEAC0576RL01830 and an Institute of Collaborative Biotechnologies Grant.

References

[1] Bernard O Koopman. Hamiltonian systems and transformation in hilbert space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931.

[2] Igor Mezić. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41(1-3):309–325, 2005.

[3] Qianxiao Li, Felix Dietrich, Erik M Bollt, and Ioannis G Kevrekidis. Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the koopman operator. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(10):103111, 2017.

[4] J Nathan Kutz, Xing Fu, and Steven L Brunton. Multiresolution dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 15(2):713–735, 2016.

[5] Clarence W Rowley, Igor Mezić, Shervin Bagheri, Philipp Schlatter, and Dan S Henningson. Spectral analysis of nonlinear flows. *Journal of fluid mechanics*, 641:115–127, 2009.

[6] Subhrajit Sinha, Umesh Vaidya, and Enoch Yeung. On computation of koopman operator from sparse data. In *2019 American Control Conference (ACC)*, pages 5519–5524. IEEE, 2019.

[7] Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1):4950, 2018.

[8] Naoya Takeishi, Yoshinobu Kawahara, and Takehisa Yairi. Learning koopman invariant subspaces for dynamic mode decomposition. In *Advances in Neural Information Processing Systems*, pages 1130–1140, 2017.

[9] Samuel E Otto and Clarence W Rowley. Linearly recurrent autoencoder networks for learning dynamics. *SIAM Journal on Applied Dynamical Systems*, 18(1):558–593, 2019.

[10] Enoch Yeung, Soumya Kundu, and Nathan Hodas. Learning deep neural network representations for koopman operators of nonlinear dynamical systems. In *2019 American Control Conference (ACC)*, pages 4832–4839. IEEE, 2019.

[11] Charles A Johnson and Enoch Yeung. A class of logistic functions for approximating state-inclusive koopman operators. In *2018 Annual American Control Conference (ACC)*, pages 4803–4810. IEEE, 2018.

[12] Zhiyuan Liu, Soumya Kundu, Lijun Chen, and Enoch Yeung. Decomposition of nonlinear dynamical systems using koopman gramians. In *2018 Annual American Control Conference (ACC)*, pages 4811–4818. IEEE, 2018.

[13] Prashant G Mehta and Umesh Vaidya. On stochastic analysis approaches for comparing complex systems. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 8082–8087. IEEE, 2005.

[14] J Nathan Kutz, Xing Fu, Steve L Brunton, and N Benjamin Erichson. Multi-resolution dynamic mode decomposition for foreground/background separation and object tracking. In *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, pages 921–929. IEEE, 2015.

[15] Ian Abraham, Gerardo De La Torre, and Todd D Murphey. Model-based control using koopman operators. *arXiv preprint arXiv:1709.01568*, 2017.

[16] Erik Berger, Mark Sastuba, David Vogt, Bernhard Jung, and Heni Ben Amor. Estimation of perturbations in robotic behavior using dynamic mode decomposition. *Advanced Robotics*, 29(5):331–343, 2015.

[17] Avrim Blum and Ronald L Rivest. Training a 3-node neural network is np-complete. In *Advances in neural information processing systems*, pages 494–501, 1989.

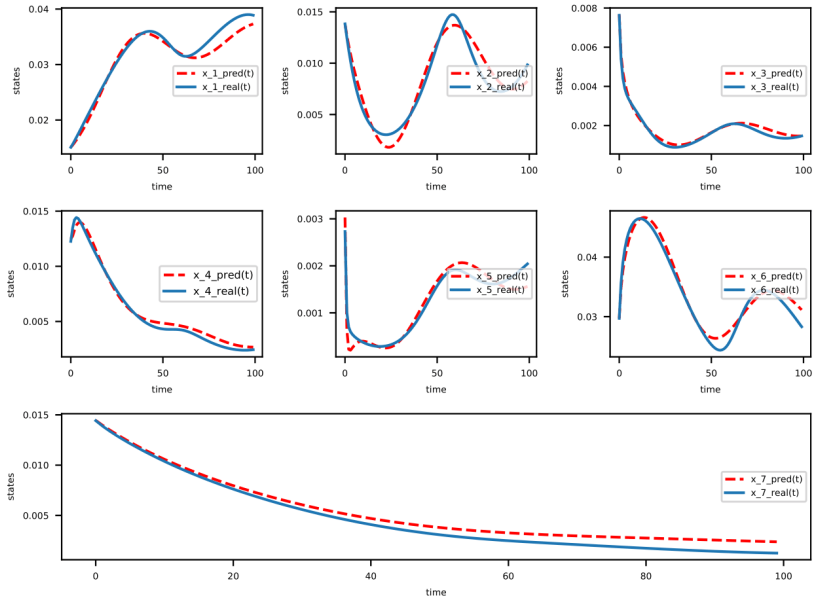

Fig. 2: Distributed Koopman learning for Glycolytic pathway.

[18] Mahdi Soltanolkotabi, Adel Javanmard, and Jason D Lee. Theoretical insights into the optimization landscape of over-parameterized shallow neural networks. *IEEE Transactions on Information Theory*, 65(2):742–769, 2018.

[19] Digvijay Boob and Guanghui Lan. Theoretical properties of the global optimizer of two layer neural network. *arXiv preprint arXiv:1710.11241*, 2017.

[20] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pages 192–204, 2015.

[21] Aqib Hasnain, Nibodh Boddupalli, and Enoch Yeung. Optimal reporter placement in sparsely measured genetic networks using the koopman operator. *arXiv preprint arXiv:1906.00944*, 2019.

[22] Matthew O Williams, Ioannis G Kevrekidis, and Clarence W Rowley. A data–driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015.

[23] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[24] MN Vrahatis, GD Magoulas, and VP Plagianakos. From linear to nonlinear iterative methods. *Applied Numerical Mathematics*, 45(1):59–77, 2003.

[25] Steven H Low and David E Lapsley. Optimization flow control—i: basic algorithm and convergence. *IEEE/ACM Transactions on Networking (TON)*, 7(6):861–874, 1999.

[26] Dimitri P Bertsekas and John N Tsitsiklis. *Parallel and distributed computation: numerical methods*, volume 23. Prentice hall Englewood Cliffs, NJ, 1989.

[27] Zhiyuan Liu and Lijun Chen. Proportional control applied to dynamic network resource allocation. In *American Control Conference (ACC), 2017*, pages 1948–1953. IEEE, 2017.

[28] Bryan C Daniels and Ilya Nemenman. Efficient inference of parsimonious phenomenological models of cellular dynamics using s-systems and alternating regression. *PloS one*, 10(3):e0119821, 2015.

[29] C Ward, E Yeung, T Brown, B Durtschi, S Weyerman, R Howes, and Jorge Goncalves. A comparison of network reconstruction methods for chemical reaction networks.