Learning Multiple Nonlinear Dynamical Systems with Side Information

Naoya Takeishi and Yoshinobu Kawahara

***Abstract*— We address the problem of learning *multiple* dynamical systems, which is a kind of multi-task learning (MTL). The existing methods of MTL do not apply to learning dynamical systems in general. In this work, we develop a regularization method to perform MTL for dynamical systems appropriately. The proposed method is based on an operator-theoretic metric on dynamics that is agnostic of model parametrization and applicable even for nonlinear dynamics models. We calculate the proposed MTL-like regularization by estimating the metric from trajectories generated during training. Learning time-varying systems can be regarded as a special case of the usage of the proposed method. The proposed regularizer is versatile as we can straightforwardly incorporate it into off-the-shelf gradient-based optimization methods. We show the results of experiments on synthetic and real-world datasets, which exhibits the validity of the proposed regularizer.**

I. INTRODUCTION

Learning (or identifying) dynamical systems plays a key role in areas such as data mining, machine learning, and control. There are several types of dynamics models frequently appear in these areas. The most conventional yet popular one is linear time-invariant (LTI) systems, which commonly appear in control theory (see, e.g., [1]). There is also a line of research on the use of Gaussian processes for modeling dynamics [2]. Moreover, the use of deep neural networks has been intensively studied (see, e.g., [3], [4]).

We frequently encounter problems where we want to learn *multiple* dynamical systems, and such problems are often facilitated with *side information*. That is, we are to estimate dynamics not only on a single dataset but also on multiple datasets respectively, and some additional information on the relationship between datasets is available. For example, when learning multiple dynamical systems from measurements of multiple sets of spatially-distributed sensors, we can utilize the sensor locations as side information, from which we may anticipate the similarity between the sets of sensors (and thus dynamical systems to be learned).

Learning multiple dynamical systems can be regarded as a kind of multi-task learning (MTL), which has been studied for a few decades in the machine learning community. When the target of learning is dynamics, however, the existing methods for MTL (e.g., [5], [6], [7], [8]) are not technically suitable, or target types of dynamics models are quite limited, which raises the need for developing a versatile MTL-like method for learning dynamical systems.

N. Takeishi is with RIKEN Center for Advanced Intelligence Project, Tokyo, Japan naoya.takeishi@riken.jp

Y. Kawahara is with the Institute of Mathematics for Industry, Kyushu University, Fukuoka, Japan, and RIKEN Center for Advanced Intelligence Project, Tokyo, Japan kawahara@imi.kyushu-u.ac.jp

In this work, we propose a regularization method to learn multiple dynamical systems using similarity derived from side information. The proposed method is based on an operator-theoretic metric [9] on dynamics, which applies to nonlinear systems and is agnostic of model parametrization. The method is suitable for general dynamics learning problems as long as some mild assumptions are met. The proposed regularizer is versatile as we can incorporate it into off-the-shelf learning methods straightforwardly because the regularizer's gradients can be computed by a standard implementation of backpropagation. We show the results of experiments on synthetic and real-world datasets, which exhibits the validity of our proposal.

II. RELATED WORK

A popular formulation in MTL is to consider a common prior distribution on models for a set of related tasks (e.g., [7]). In another formulation, models for related tasks are forced to be similar in the sense of a norm in a function space [5], [6], [8]. For example, MTL with linear models is done by making the Euclidean distance of the parameters proportional to the dissimilarity derived from task relatedness [6]. However, for nonlinear dynamical systems, such soft parameter sharing cannot be applied directly because the discrepancy of model parameters does not necessarily represent the discrepancy of dynamics.

More generally, use of side information has been studied in several contexts. Vapnik and Vashist [10] considered privileged information that is available only in training phases for support vector machines. Huang *et al.* [11] developed a method to impose sparsity of model parameters based on graph structures of features. There is also a strain of researches [12], [13], [14] on incorporating knowledge of feature similarity to supervised learning. Regardless of these interests, there have been surprisingly few studies on learning nonlinear dynamical systems with side information.

III. PRELIMINARY

A. Learning Dynamical Systems

We denote a discrete-time dynamical system by

$$x_{t+1} = f(x_t), \tag{1}$$

where $t \in \mathbb{N}$ is a time index, $x \in \mathcal{M} \subset \mathbb{R}^p$ is a state vector in a state space $\mathcal{M}$, and $f \colon \mathcal{M} \to \mathcal{M}$ is a map on the state space. We consider to learn f given a set of measurements of x, i.e., $\{x_1, x_2, \dots, x_\tau\}$. This is a long-standing problem in various areas such as economics, machine learning, data mining, and control. Several types of methods have been considered in machine learning, such as EM algorithms [15] and Gaussian processes [2]. Specifically, methods based on deep neural networks have been attracting attention (e.g., [4], [3]). In control theory, there are many studies on system identification for LTI systems.

B. Operator-Theoretic View on Dynamical Systems

Learning dynamical systems in the state-space formalism (1) is inherently difficult when f is nonlinear. The operator-theoretic formalism of dynamical systems [16], [17] is attracting attention as it enables us to analyze nonlinear systems using the rich machinery of the linear operator theory. We also adopt this formalism in this work.

Let us consider an observable function $h\colon \mathcal{M} \to \mathbb{R}$ in some Hilbert space $\mathcal{H}$, and consider an operator $\mathcal{K}$ such that

$$\mathcal{K}h(x) = (h \circ f)(x) = h(f(x)), \tag{2}$$

where $\circ$ means the composition of functions. This operator, $\mathcal{K}$, is often termed the Koopman operator [16]. Because $\mathcal{H}$ is a Hilbert space, $\mathcal{K}$ is a linear operator. Using $\mathcal{K}$, we can analyze a nonlinear dynamical system f through a linear operator. For example, it is known that the spectral analysis of $\mathcal{K}$ is useful for understanding coherent patterns [17].

Let us restrict $\mathcal{H}$ to be a reproducing kernel Hilbert space (RKHS), and let $k : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ be the corresponding kernel function. Also let $\phi : \mathcal{M} \to \mathcal{H}$ be the corresponding feature map. Now define another operator K_f such that

$$K_f \phi(x) = \phi(f(x)). \tag{3}$$

In fact, this K_f is the adjoint of $\mathcal{K}$, namely the Perron–Frobenius operator in RKHS (see, e.g., [9]).

C. Metrics on Dynamical Systems

For measuring similarity between dynamical systems, several types of methods have been proposed. Martin [18] defined a metric on ARMA models, and De Cock and De Moor [19] extended it to linear state-space models using subspace angles between observability matrices. Vishwanathan *et al.* [20] proposed a family of kernels known as Binet–Cauchy kernels that can be used for dynamical systems. For nonlinear systems, researchers have proposed metrics based on the operator-theoretic formalism. For example, Fujii *et al.* [21] developed an extension of the Binet–Cauchy kernels using the spectral decomposition of the Koopman operator. Ishikawa *et al.* [9] proposed a metric on nonlinear dynamical systems with a rigorous theoretical background.

In this work, we adopt the metric proposed by Ishikawa *et al.* [9]. The advantages of this metric are as follows: 1) it can be applied to nonlinear systems, 2) it is agnostic of the parametric forms of dynamics models, and 3) it can be estimated with trajectories generated from target dynamics. Below we review its definition and computation.

Let f_1 and f_2 be the maps of two dynamical systems, and let X_1^{init} and X_2^{init}, respectively, be the sets of initial conditions for them. Using the notion of the Perron–Frobenius operator in RKHS, (i.e., K_f in (3)), Ishikawa *et al.* [9] define a positive definite kernel k_{PF} as

$$\begin{aligned} &k_{\text{PF}}^{m,T}\big((f_1, X_1^{\text{init}}), (f_2, X_2^{\text{init}})\big) \\ &\qquad := \operatorname{trace}\left(\bigwedge^{m} \sum_{r=0}^{T-1} \big(L_h K_{f_2}^r \mathcal{I}_2\big)^* L_h K_{f_1}^r \mathcal{I}_1\right), \end{aligned} \tag{4}$$

where $\bigwedge^m$ denotes the m-th exterior product, L_h is a linear operator defined according to an observable h, and $\mathcal{I}_1$ and $\mathcal{I}_2$ are operators defined according to the initial conditions, X_1^{init} and X_2^{init}, respectively. See [9] for the details. Using this kernel, a (pseudo-)metric d_{PF} is defined as

$$d_{\text{PF}}^{m,T}\big((f_1, X_1^{\text{init}}), (f_2, X_2^{\text{init}})\big) := \sqrt{1 - \frac{\big(k_{\text{PF},(1,2)}^{m,T}\big)^2}{k_{\text{PF},(1,1)}^{m,T}\, k_{\text{PF},(2,2)}^{m,T}}}, \tag{5}$$

where $k_{\text{PF},(1,2)}^{m,T}$ means $k_{\text{PF}}^{m,T}((f_1, X_1^{\text{init}}), (f_2, X_2^{\text{init}}))$ and so on.

As long as f's are semi-stable (non-diverging) [9], k_{PF} can be estimated from trajectories. For simplicity, consider the case of $m = 2$ and $X_j^{\text{init}} = \{x_0^{(j,1)}, x_0^{(j,2)}\}$. Let $(x_0^{(j,l)}, \dots, x_T^{(j,l)})$ be a trajectory from f_j initialized with $x_0 = x_0^{(j,l)}$, for $l = 1, 2$ and $j = 1, 2$. The value of k_{PF} can be estimated from such trajectories [9] as

$$\begin{aligned} &\hat{k}_{\text{PF}}^{2,T}\big((f_1, X_1^{\text{init}}), (f_2, X_2^{\text{init}})\big) \\ &\quad = \sum_{t_1=0}^{T} \sum_{t_2=0}^{T} \det \begin{bmatrix} k(x_{t_1}^{(1,1)}, x_{t_1}^{(2,1)}) & k(x_{t_1}^{(1,1)}, x_{t_2}^{(2,2)}) \\ k(x_{t_2}^{(1,2)}, x_{t_1}^{(2,1)}) & k(x_{t_2}^{(1,2)}, x_{t_2}^{(2,2)}) \end{bmatrix}. \end{aligned} \tag{6}$$

Here recall that $k(x, x')$ (that is different from k_{PF}) is some kernel function (e.g., Gaussian kernel) on $x, x' \in \mathcal{M}$.

IV. PROPOSED METHOD

We first formalize our problem and then give the details of the proposed method. We add discussion via a special case that admits an analytical computation of the method.

A. Problem Setting

We consider a problem to learn multiple dynamical systems from multiple datasets, where some side information that encodes the relationship between the datasets is available. More formally, we learn n dynamical systems $f_1, \dots, f_n$ given n datasets $\mathcal{D}_1, \dots, \mathcal{D}_n$, where each dataset $\mathcal{D}_i$ is a pair of a sequence and side information:

$$\mathcal{D}_i := \big\{(x_1^{(i)}, \dots, x_{\tau_i}^{(i)}),\ Z_i\big\}. \tag{7}$$

Here, $x_t^{(i)} \in \mathbb{R}^p$ is the t-th snapshot of the i-th sequence, and $\tau_i \in \mathbb{N}$ is the length of the i-th sequence. $Z_i \in \mathcal{Z}$ denotes some side information, and it can take many forms, e.g., scalars, vectors, or matrices of numeric and/or categorical values, as long as a dissimilarity function on $\mathcal{Z}$ can be computed. We denote a dissimilarity function on $\mathcal{Z}$ by

$$d_{\mathcal{Z}}(Z, Z')\colon \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}_{\geq 0}, \tag{8}$$

and we assume that it fulfills the following properties: $d_{\mathcal{Z}}(Z, Z') = d_{\mathcal{Z}}(Z', Z)$ and $Z = Z' \Leftrightarrow d_{\mathcal{Z}}(Z, Z') = 0$.

B. Proposed Method

We propose the regularizer that will be defined in (10). It is computed via (11), (13), and (14), based on trajectories (12) generated from models being learned.

1) General Formulation: Our idea is as follows; to facilitate learning dynamical systems using side information, we impose regularization such that the dissimilarity between f_i and f_j follows the dissimilarity derived from the side information. Similar ideas have also been adopted in the well-known methodologies of MTL [5], [6], [8], but they are not for dynamical systems. Suppose that learning f_i on $\mathcal{D}_i$ is originally performed via solving an optimization problem:

$$\underset{f_i}{\text{minimize}} \quad L_i\big(f_i; (x_1^{(i)}, \dots, x_{\tau_i}^{(i)})\big) \tag{9}$$

with some loss function L_i. We regularize (9) as follows.

Definition 1. *Let L_i be the original loss function to learn dynamical system f_i on data $\mathcal{D}_i$. Let d_{DS} be some dissimilarity measure of dynamical systems. Suppose $d_{\mathcal{Z}}(Z_i, Z_j) > 0$ for every pair (i, j) of $i \neq j$. The regularized problem is*

$$\underset{f_1,\dots,f_n}{\text{minimize}} \quad \frac{1}{n}\sum_{i=1}^{n} L_i\big(f_i; (x_1^{(i)}, \dots, x_{\tau_i}^{(i)})\big) + \frac{\lambda}{n(n-1)}\sum_{i=1}^{n}\sum_{j>i}^{n} R_{i,j}, \tag{10}$$

where $\lambda \geq 0$ is the regularization hyperparameter, and

$$R_{i,j} := d_{DS}\big(f_i, f_j\big) \,/\, d_{\mathcal{Z}}(Z_i, Z_j). \tag{11}$$

Remark 1. This formulation is agnostic of the parametric forms of f, and models f_i and f_j $(i \neq j)$ can even have different parametric forms. When a pair s.t. $i \neq j$ and $d_{\mathcal{Z}}(Z_i, Z_j) = 0$ exists, we simply learn f_i on $\mathcal{D}_i$ and $\mathcal{D}_j$.

Definition 1 is an abstract formulation, so we have to define f, L, d_{DS}, and $d_{\mathcal{Z}}$ in practice. Whereas f and L are chosen in accordance with the base problem, we must design d_{DS} and $d_{\mathcal{Z}}$ appropriately. In the following, we introduce our proposal to use the operator-theoretic metric for d_{DS}. Afterward, we provide examples of $d_{\mathcal{Z}}$ in some use cases.

2) Choice of d_{DS}: To make the proposed regularized learning problem feasible, we pose two mild assumptions:

Assumption 1. *For $i = 1, \dots, n$, model f_i is parameterized with a set of parameters θ_i, and the gradients of $f_i(x)$ with regard to θ_i can be computed for every $x \in \mathcal{M}$.*

Assumption 2. *Models $f_1, \dots, f_n$ are semi-stable [9].*

Remark 2. Informally, the semi-stability of f means that trajectories generated from f do not diverge. It is not obviously satisfied in every case, but empirically, f remains semi-stable during training when data do not diverge, and the magnitude of f's parameters are in a reasonable range.

Under these assumptions, we propose to exploit the operator-theoretic metric [9], namely d_{PF} in (5), for dissimilarity measure d_{DS} in (11). This metric is advantageous mainly because of two reasons. First, it can be computed from trajectories of dynamical systems, and thus in practice, we can adopt gradient-based optimization as long as k in (6) is differentiable (at least almost everywhere). Second, the metric applies to *any* parametric forms of semi-stable dynamical systems, so we can regularize f's even if we do not know the semantics of their parametrization, which is usually the case with nonlinear models such as neural nets.

Here we present the actual procedures to use d_{PF} for d_{DS}. For ease of discussion, we fix $m = 2$ and let $X_i^{\text{init}} = \{x_0^{(i,1)}, x_0^{(i,2)}\}$, which was empirically satisfactory. When computing the gradient of the objective of (10), we generate trajectories of length σ using f_i and f_j with initial conditions in X_i^{init} and X_j^{init}, respectively. That is, we generate *four* trajectories in an evaluation of gradient, and one of them is

$$Y_{i,1} := \Big(y_s^{(i,1)} = f_i^s\big(x_0^{(i,1)}\big) \;\Big|\; s = 1, \dots, \sigma\Big), \tag{12}$$

where f^s means the action of f for s times. Other three, $Y_{i,2}$, $Y_{j,1}$, and $Y_{j,2}$ are defined analogously. Given such trajectories, we can compute an empirical value of k_{PF} as

$$\hat{k}_{\text{PF},(i,j)} = \sum_{s_1=1}^{\sigma}\sum_{s_2=1}^{\sigma} \Big(k(y_{s_1}^{(i,1)}, y_{s_1}^{(j,1)})k(y_{s_2}^{(i,2)}, y_{s_2}^{(j,2)}) - k(y_{s_1}^{(i,1)}, y_{s_2}^{(j,2)})k(y_{s_2}^{(i,2)}, y_{s_1}^{(j,1)})\Big), \tag{13}$$

where k is a differentiable kernel function on $\mathcal{M}$. We then compute an empirical estimation of the metric as

$$\hat{d}_{\text{PF},(i,j)} = \sqrt{1 - \hat{k}^2_{\text{PF},(i,j)} / \hat{k}_{\text{PF},(i,i)} / \hat{k}_{\text{PF},(j,j)}}. \tag{14}$$

Each summand of the regularizer (i.e., $R_{i,j}$ in (11)) can be computed by substituting $\hat{d}_{\text{PF}}$ for d_{DS}. As long as k is differentiable, the gradient of $R_{i,j}$ is also computed easily by backpropagation because $R_{i,j}$ comprises only arithmetic operations on trajectories Y. Hence, when the original problem is solved with a gradient-based optimizer, the regularized one can also be solved using the same optimizer.

The computation of the regularization term needs $O(pn^2\sigma^2)$ operations. To reduce the computational cost, it is effective to reduce n by ignoring $R_{i,j}$ for pairs whose $d_{\mathcal{Z}}(Z_i, Z_j)$ is large. We empirically found that σ does not have to be too large, and a large σ might even be harmful for performance. It is probably because a model cannot generate meaningful long trajectories when the training is in progress.

Metric d_{PF} depends on the initial conditions [9]. In our application, however, such dependence should be eliminated because we want to compare dynamical systems regardless of their initial conditions. To this end, in computing $\hat{d}_{\text{PF},(i,j)}$ in (14), we use two common initial conditions, $x_0^{(i,j,1)}$ and $x_0^{(i,j,2)}$. That is, we set $X_i^{\text{init}} = X_j^{\text{init}} = \{x_0^{(i,j,1)}, x_0^{(i,j,2)}\}$. Consequently, $\hat{d}_{\text{PF},(i,j)}$ only depends on the choice of $x_0^{(i,j,1)}$ and $x_0^{(i,j,2)}$. This dependence can further be eliminated (only roughly, though) by randomly choosing them multiple times during training. However, in the numerical examples in Section V, we simply fixed $x_0^{(i,j,1)}$ and $x_0^{(i,j,2)}$ to be $x_1^{(i)}$ and $x_1^{(j)}$ (the first snapshots of each dataset), respectively, which we found was not problematic empirically.

3) Use Cases: The type of side information, $\mathcal{Z}$, and the dissimilarity function, $d_{\mathcal{Z}}$, are defined in accordance with a given application. Below we show two examples.

a) Multi-dynamics learning: The principal focus of the proposed regularizer is to learn multiple dynamical systems. A typical example is to learn dynamics on data from spatially distributed sensors; in this case, Z_i refers to the location of the i-th set of sensors, and $d_{\mathcal{Z}}$ denotes the distance between the sensor locations. Another example is to learn dynamics of human motion for multiple subjects, in which some physical properties (e.g., heights and weights) of the subjects are available as side information to anticipate dissimilarity between the subjects. Furthermore, any other types of descriptions (e.g., texts), can also be used for evaluating relatedness between dynamics. The proposed method is applicable even if the parametrizations of dynamics models are different from each other. This is useful, for example, when model architecture searches are run for each task independently.

b) Time-varying dynamical systems: Learning a time-varying dynamical system can be regarded as a special case of multi-dynamics learning, though it is not literally "multiple" dynamical systems. Let $\{\bar{x}_1, \ldots, \bar{x}_{T+1}\}$ be a time-series dataset of length $T+1$ and suppose to learn

$$\bar{x}_{t+1} = f_{i(t)}(\bar{x}_t), \quad \text{for} \quad t = 1, \ldots, T, \tag{15}$$

where $i\colon \{1, \ldots, T\} \to \{1, \ldots, n\}$ ($n \le T$) is a function that represents the transition timing of the time-varying system. As learning $f_1, \ldots, f_n$ can become severely ill-posed when $n \approx T$, it is indispensable to impose a regularization, for example, to make the dynamics change gradually along time. We can cast this problem as a special case of the proposed formulation. Consider the most extreme case, i.e., $n = T$ and $i(t) = t$. For this case, we redefine datasets as

$$\mathcal{D}_i = \left\{(\bar{x}_i, \bar{x}_{i+1}),\ Z_i = i\right\}, \quad \text{for} \quad i = 1, \ldots, T,$$

and set the dissimilarity on $\mathcal{Z} = \{1, \ldots, T\}$ as

$$d_{\mathcal{Z}}(Z_i = i, Z_j = j) = \begin{cases} 1, & \text{for } j = i+1, \\ \infty, & \text{otherwise.} \end{cases} \tag{16}$$

C. Special Case: Learning Stable LTI Systems

Consider learning LTI systems

$$x_{t+1}^{(i)} = A_i x_t^{(i)}, \quad A_i \in \mathbb{R}^{p \times p},$$

for $i = 1, \ldots, n$, where A_i is stable, that is, $\rho(A_i) < 1$. For stable linear systems, it is known [9] that the operator-theoretic metric, d_{PF}, coincides with the metric based on the subspace angles between the extended observability matrices of systems [19]. Let O_i be the extended observability matrix corresponding to A_i, that is, $O_i := [I\ A_i\ A_i^2\ \cdots]^\top$, and let $\mathcal{C}_i$ be the column space of O_i. A metric based on O can be defined via the projection kernel (see, e.g., [22]):

$$k_{\text{proj}}(A_i, A_j) := \operatorname{trace}(\mathbb{P}_{\mathcal{C}_i} \mathbb{P}_{\mathcal{C}_j}),$$

where $\mathbb{P}_{\mathcal{C}} := O(O^\top O)^{-1} O^\top$ is the projection operator onto $\mathcal{C}$. The corresponding metric, namely d_{proj}, is

$$d_{\text{proj}}^2(A_i, A_j) := 1 - p^{-2} k_{\text{proj}}^2(A_i, A_j),$$

where we used the fact $k_{\text{proj}}(A, A) = p$. We can use d_{proj}, which is equivalent to d_{PF} in this case, as d_{DS} in (11).

It is known that a matrix $G_{i,j} := O_i^\top O_j$ can be computed by solving a Sylvester equation:

$$A_i^\top G_{i,j} A_j - G_{i,j} + I = 0.$$

Consequently, we can compute the values of k_{proj} analytically. The gradients of d_{proj} can be computed similarly via solving Sylvester equations, but we omit the details here (see, e.g., [22]). Note that an exact solution of a Sylvester equation requires $O(p^3)$ operations, where p denotes the dimensionality of each snapshot. For large p, we should resort to some iterative methods for the Sylvester equations or dimensionality reduction of data as preprocessing.

For discussion, let us also consider the extreme case of $p = 1$, where we denote A_i and A_j by a_i and $a_j \in \mathbb{R}$, respectively. In this case, d_{proj} becomes

$$d_{\text{proj}}^2(a_i, a_j) = (a_i - a_j)^2 / (1 - a_i a_j)^2.$$

We see that in $p = 1$, d_{proj} emphasizes differences of dynamics in the small decay rate region ($a \approx 1$). The proposed method naturally implicates such a behavior as in this special case, whereas the existing methods do not.

V. NUMERICAL EXAMPLES

A. Datasets and Side Information

We used the following four datasets with side information.

a) Van der Pol oscillator: We synthesized data (termed VDP hereafter) by numerically solving $\ddot{x} - \mu(1-x^2)\dot{x} + x = 0$, which is known as the Van der Pol oscillator. The data comprise trajectories of $x = [x\ \dot{x}]^\top$. As VDP, we created five datasets with $\mu = 1.0, 1.5, 2.0, 2.5, 3.0$, each of which was processed as follows. First, using MATLAB's `ode45` with $x_0 = \dot{x}_0 = 1$, $\Delta t = 0.05$, we generated a trajectory of length 800. Within the trajectory of length 800, we used $(x_1, \ldots, x_{\tau_{\text{VdP}}})$ as a training set, $(x_{401}, \ldots, x_{600})$ as a validation set, and $(x_{601}, \ldots, x_{800})$ as a test set. We set $\tau_{\text{VdP}} = 40$. We added noise generated independently from $\mathcal{N}(0, 10^{-4})$ only to the training sets. As side information, we set $d_{\mathcal{Z}} = 1$ for adjacent μ values (e.g., $\mathcal{D}_{\mu=1.0}$ and $\mathcal{D}_{\mu=1.5}$) and $d_{\mathcal{Z}} = \infty$ otherwise.

b) Rössler system: We created data (termed RÖSSLER) by numerically solving the Rössler system:

$$\dot{x} = \begin{bmatrix} -[x]_2 - [x]_3 \\ [x]_1 + a[x]_2 \\ b + [x]_3([x]_1 - c) \end{bmatrix}.$$

As RÖSSLER, we created five datasets with parameters $c = 4$, 5, 6, 7, 8, whereas $a = b = 0.1$ were fixed. Each dataset was processed similarly to the VDP dataset. The length of the training set, $\tau_{\text{Rössler}}$, was varied within $\{24, 32, 40, 48\}$. Note that with the above values of a, b, c, the Rössler system is *not* chaotic. As side information, we set $d_{\mathcal{Z}} = 1$ for adjacent datasets (e.g., $\mathcal{D}_{c=4}$ and $\mathcal{D}_{c=5}$) and $d_{\mathcal{Z}} = \infty$ otherwise.

TABLE I

TEST SET LOSS. AVERAGES (AND STANDARD DEVIATIONS) FOR 20 RANDOM TRIALS ARE DISPLAYED.

	(A) only weight decay	(B) weight decay + Euclid. reg.	(C) weight decay + fused lasso	(D) weight decay + proposed reg.
VDP	$2.936\,(.56) \times 10^{-1}$	$2.402\,(.82) \times 10^{-1}$	$2.659\,(.51) \times 10^{-1}$	$\mathbf{2.272}\,(.71) \times 10^{-1}$
RÖSSLER	$1.358\,(.05) \times 10^{-2}$	$1.359\,(.05) \times 10^{-2}$	$1.358\,(.05) \times 10^{-2}$	$\underline{\mathbf{1.319}}\,(.05) \times 10^{-2}$
SOLAR	$9.231\,(.01) \times 10^{-4}$	$\mathbf{9.229}\,(.01) \times 10^{-4}$	$\mathbf{9.226}\,(.01) \times 10^{-4}$	$\underline{\mathbf{9.101}}\,(.03) \times 10^{-4}$
DEMAND	$1.486\,(.03) \times 10^{-3}$	$1.487\,(.03) \times 10^{-3}$	$1.488\,(.03) \times 10^{-3}$	$\underline{\mathbf{1.439}}\,(.03) \times 10^{-3}$

Significant difference, **bold**: from (A) / underline: from (B), by paired t-test at $p < 10^{-3}$.

c) Solar power production data: We used the solar power production data[1] (termed SOLAR) of plants in Alabama. As SOLAR, we prepared 28 datasets by selecting 28 power plants of type DPV, whose latitude was from $33.15°$ to $33.75°$ and longitude was from $-87.05°$ to $-86.55°$. We used the data from 1 June 2016 to 20 June as training sets, the data from 21 June to 25 June validation sets, and the data from 25 June to 30 June as test sets. We subsampled the original sequence (measurements of every five minutes) by $1/3$, computed the moving average of width 4 ($=$ an hour), and normalized so that the minimum and the maximum values become 0 and 1, respectively. Finally, we regarded every 24 measurements ($=$ six hours) as a 24-dimensional snapshot x. The sizes of the training, validation, and test sets were 80, 20, and 20, respectively. As side information, we used the geometric distances between the power plants.

d) Electricity demand data: We used the electricity demand data[2] (termed DEMAND) in Tokyo from 2016 to 2018. We normalized the data so that the maximum value becomes 1. The original sequences were hourly records, and we subsampled them by $1/3$ and treated measurements in every 12 hours as a four-dimensional snapshot x. We used the three years' sequences spanning 32 weeks from April to December (because the data from January to March 2016 were missed). We partitioned each sequence into consequent four datasets, so it is like we learn a time-varying dynamical system that changes every eight weeks. We used data in 2016, 2017, and 2018 as training, validation, and test sets, respectively. As side information, we set $d_{\mathcal{Z}}$ as in (16); that is, this dataset corresponds to the special case of learning time-varying dynamics, presented in Section IV-B.3.

B. Settings

a) Model and loss: For f, we used multilayer perceptrons with one hidden layer and $\tanh$ activation function. The size of the hidden layer was 32 for VDP, RÖSSLER, and DEMAND datasets, and 64 for SOLAR dataset. We optimized the model with the mean squared loss using gradient descent with an adaptive learning rate (Adam) [23].

b) Baselines: While there are no existing methods that have the same objective with ours, we compared the proposed method to the following three baseline settings, (A)–(C). The first baseline (A) is learning only with weight decay (i.e., L2 regularization on parameters). The second baseline (B) is the regularization based on Euclidean distance of parameters, that is, we use

$$d_{\text{Euclid}}(f_i, f_j) = \frac{1}{m} \sum_{\ell=1}^{m} |\theta_{i,\ell} - \theta_{j,\ell}|^2$$

as d_{DS} in (11). Here $\theta_{i,\ell} \in \mathbb{R}$ denotes the ℓ-th parameter of f_i, and m is the number of parameters in f_i. We refer to this baseline as "Euclidean regularization." The third baseline (C) is similar to fused lasso [24], that is, we use

$$d_{\text{fused-lasso}}(f_i, f_j) = \frac{1}{m} \sum_{\ell=1}^{m} |\theta_{i,\ell} - \theta_{j,\ell}|$$

as d_{DS} in (11), instead of the L2 norm in (B).

c) Proposed method: For the kernel k on state space, which appeared on the right-hand side of (13), we used the Gaussian kernel with the bandwidth chosen by the median heuristics. We set the length of generated trajectories to be $\sigma = 4$ unless otherwise stated. In every setting, the regularization parameter λ (both for the baselines and the proposed regularizer) was chosen according to the validation loss. We ran experiments with each configuration for 20 random trials, where the randomness involves the initialization and the artificial noise added to training data.

C. Results

a) Evaluation by test loss: In Table I, we show the loss values on test sets (the smaller, the better). Note that the results on the RÖSSLER dataset are shown only for the case of training set size $\tau_{\text{Rössler}} = 40$. We can observe that, while (B) the Euclidean regularization and (C) the fused lasso regularization do not necessarily improve the performance, (D) the proposed regularization method consistently achieves smaller test loss values.

b) Changing training set size: We also calculated the rates of improvement in the test loss compared to the case without a structured regularizer, i.e., (improvement rate) $=$ $(L_{\text{A}} - L_{\text{D}})/L_{\text{A}}$, where L_{A} and L_{D} denote the test loss in the two cases (A) and (D) listed in the top of Table I, respectively. In Figure 1, the improvement rates on the RÖSSLER dataset are shown for different sizes of training set. This result implies that the impact of the proposed regularizer is particularly notable when the training set is

[1] `www.nrel.gov/grid/solar-power-data.html`
[2] `www.tepco.co.jp/en/forecast/html/download-e.html`

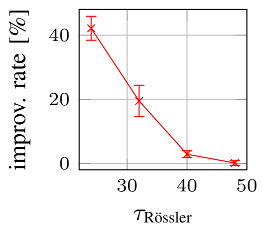

Fig. 1. Improvement rates (avg. and SDs) of test loss with different sizes of training set on RÖSSLER dataset.

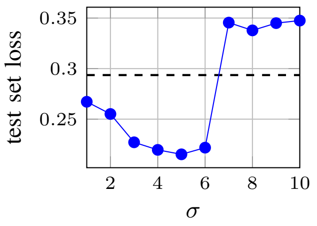

Fig. 2. Test loss on VDP dataset with different σ. The dashed line shows the average performance of the case (A) weight decay only.

small. It is advantageous because in many practices of learning dynamical systems, gathering much data is the most expensive (and often infeasible) part of applications.

c) Changing trajectory length: We examined the effects of varying the length of trajectories generated and used by the proposed regularizer (i.e., σ in (12)). In Figure 2, we show the test loss values on the VDP dataset for a random trial under the same setting except σ; we changed σ from 1 to 10. We can see that a larger σ achieves better performance in $\sigma \leq 6$, but the performance even deteriorates in $\sigma > 6$ in this trial. There were some successful trials with $\sigma > 6$ in other trials, but the improvement was marginal with a large σ. This is probably because a model whose learning is still in progress cannot generate meaningful long trajectories. As the generation capability of a model improves as learning proceeds, changing σ adaptively may improve performance, which should be elaborated in the future.

d) Runtime: We examined the runtime using an Intel Xeon Gold 6148 processor, DDR4 2400MHz 256GB RAM, and the implementation based on PyTorch 1.1.0. For a trial to learn the model on DEMAND dataset (i.e., $n = 28$), it took 3.9 seconds for $1,000$ iteration without the proposed regularizer, whereas it took 46 seconds for the same number of iterations with the proposed regularizer.

VI. CONCLUSION

In this work, we developed a regularization method for learning multiple dynamical systems with side information. We used the metric on dynamical systems [9], which is applicable to nonlinear dynamics and is agnostic of model parametrization. We proposed to estimate the metric using trajectories generated from models being learned and to use it for the regularization. Because the gradients of the proposed regularizer can be computed using an off-the-shelf implementation of backpropagation, it can be incorporated into existing methods of learning dynamical systems, such as ones based on gradient-based optimization.

A possible limitation of the proposed method (and many other MTL-like methods) lies in the computational burden in large n, that is, the number of tasks involved. In order to scale to a large n, we have to develop some techniques, e.g., a way to efficiently prune side information. Moreover, the proposed method should be generalized to controlled systems, stochastic systems, and continuous-time systems.

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grant No. JP18H03287 and JST CREST Grant No. JPMJCR1913.

REFERENCES

[1] T. Katayama, *Subspace Methods for System Identification*. Springer-Verlag London, 2005.

[2] J. M. Wang, D. J. Fleet, and A. Hertzmann, “Gaussian process dynamical models for human motion,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 283–298, 2008.

[3] M. Karl, M. Soelch, J. Bayer, and P. van der Smagt, “Deep variational Bayes filters: Unsupervised learning of state space models from raw data,” in *Proc. of the 5th Int. Conf. on Learning Representations*, 2017.

[4] R. Krishnan, U. Shalit, and D. Sontag, “Structured inference networks for nonlinear state space models,” in *Proc. of the 31st AAAI Conf. on Artificial Intelligence*, 2017, pp. 2101–2109.

[5] C. A. Micchelli and M. Pontil, “Kernels for multi–task learning,” in *Advances in Neural Information Processing Systems 17*, 2005, pp. 921–928.

[6] T. Evgeniou, C. A. Micchelli, and M. Pontil, “Learning multiple tasks with kernel methods,” *J. Mach. Learn. Res*, vol. 6, pp. 615–637, 2005.

[7] E. V. Bonilla, K. M. Chai, and C. Williams, “Multi-task Gaussian process prediction,” in *Advances in Neural Information Processing Systems 20*, 2008, pp. 153–160.

[8] C. Ciliberto, Y. Mroueh, T. Poggio, and L. Rosasco, “Convex learning of multiple tasks and their structure,” in *Proc. of the 32nd Int. Conf. on Machine Learning*, 2015, pp. 1548–1557.

[9] I. Ishikawa, K. Fujii, M. Ikeda, Y. Hashimoto, and Y. Kawahara, “Metric on nonlinear dynamical systems with Perron–Frobenius operators,” in *Advances in Neural Information Processing Systems 31*, 2018, pp. 2856–2866.

[10] V. Vapnik and A. Vashist, “A new learning paradigm: Learning using privileged information,” *Neural Networks*, vol. 22, no. 5–6, pp. 544–557, 2009.

[11] J. Huang, T. Zhang, and D. Mataxas, “Learning with structured sparsity,” *Journal of Machine Learning Research*, vol. 12, pp. 3371–3412, 2011.

[12] E. Krupka and N. Tishby, “Incorporating prior knowledge on features into learning,” in *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics*, 2007, pp. 227–234.

[13] T. Sandler, J. Blitzer, P. P. Talukdar, and L. H. Ungar, “Regularized learning with networks of features,” in *Advances in Neural Information Processing Systems 21*, 2009, pp. 1401–1408.

[14] A. Mollaysa, P. Strasser, and A. Kalousis, “Regularising non-linear models using feature side-information,” in *Proceedings of the 34th International Conference on Machine Learning*, 2017, pp. 2508–2517.

[15] Z. Ghahramani and S. T. Roweis, “Learning nonlinear dynamical systems using an EM algorithm,” in *Advances in Neural Information Processing Systems 11*, 1999, pp. 431–437.

[16] I. Mezić, “Spectral properties of dynamical systems, model reduction and decompositions,” *Nonlinear Dynamics*, vol. 41, pp. 309–325, 2005.

[17] M. Budišić, R. M. Mohr, and I. Mezić, “Applied Koopmanism,” *Chaos*, vol. 22, no. 4, p. 047510, 2012.

[18] R. J. Martin, “A metric for ARMA processes,” *IEEE Trans. Signal Process.*, vol. 48, no. 4, pp. 1164–1170, 2000.

[19] K. De Cock and B. De Moor, “Subspace angles between ARMA models,” *Systems Control Lett.*, vol. 46, pp. 265–270, 2002.

[20] S. Vishwanathan, A. J. Smola, and R. Vidal, “Binet–Cauchy kernels on dynamical systems and its application to the analysis of dynamic scenes,” *Int. J. Comput. Vis.*, vol. 73, no. 1, p. 95–119, 2007.

[21] K. Fujii, Y. Inaba, and Y. Kawahara, “Koopman spectral kernels for comparing complex dynamics: Application to multiagent sport plays,” in *Lecture Notes in Computer Science*, vol. 10536, 2017, pp. 127–139.

[22] W. Huang, M. Harandi, T. Zhang, L. Fan, F. Sun, and J. Huang, “Efficient optimization for linear dynamical systems with applications to clustering and sparse coding,” in *Advances in Neural Information Processing Systems 30*, 2017, pp. 3444–3454.

[23] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” in *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

[24] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, and K. Knight, “Sparsity and smoothness via the fused lasso,” *J. R. Stat. Soc. Ser. B. Stat. Methodol.*, vol. 67, no. 1, pp. 91–108, 2005.