Learning a Dynamic-based Representation for Multivariate Biomarker Time Series Classifications

Xi Hang Cao, Chao Han and Zoran Obradovic
Center for Data Analytics and Biomedical Informatics
Computer & Information Science Department, Temple University
Philadelphia, USA
{xi.hang.cao, chao.han, zoran.obradovic}@temple.edu

***Abstract*—Time series in healthcare practices and biomedical research are typically multivariate, i.e. multiple biomarkers are observed simultaneously at a time. However, they tend to be short, noisy, unaligned, irregularly sampled, partially observed, and with only limited samples. These imperfections pose a challenge for mining information from data. In this work, we propose to use dynamic-based representations to present such imperfect multivariate time series. Specifically, we propose an approach to learn a corresponding Linear Dynamical System (LDS) for a multivariate time series example and use the set of system parameters as a representation for that example. Such a representation is able to capture interactions of different variables and provide a unified view of multivariate time series with different lengths, different missingness mechanisms, and different starting points. Other techniques are then used to mine useful information and perform learning tasks based on the new representation. For example, we use support vector machine classification models with LDS kernels in time series classification tasks. To evaluate the effectiveness of the proposed approach, we conducted experiments on both synthetic data sets and real-life datasets. The results in synthetic datasets demonstrated that the proposed approach could correctly learn the similarities of underlying linear dynamical systems. Our real-life data sets included human influenza A (H3N2), Rhinovirus (HRV), and respiratory syncytial virus (RSV) gene expression time series. The accuracies in the leave-one-out symptomatic/asymptomatic diagnostic tasks showed that our approach outperformed three baseline algorithms. Moreover, in experiments where various levels of imperfections were imposed on the H3N2 dataset, the accuracies of other baseline methods degraded significantly, but the accuracy of our approach remained high.**

***Index Terms*—Time series analysis, Kernel method, Representation learning, Biomarker analysis, Unevenly spaced, Unaligned**

I. Background

A. Time series data in biomedical research

Data mining and machine learning techniques have great potential to improve healthcare quality by allowing effective knowledge extraction from observed data. Among different forms of data, time series data are very important for both medical research and clinical practice. A multivariate time series (MTS) is a sequence of observations on multiple variables in time. An efficient data mining model for time series analysis would capture the dynamics of the ongoing disease progression and interactions among biological system components and thus lead to an accurate forecasting of health state changes [1]. More advanced methodologies can reveal hidden phenomena in the biological/physiological mechanism of a subject and provide the insights for proper treatments [2].

B. Imperfections in biomedical time series data

To evaluate the health of a patient, we need to measure quantities related to the patient's physiological states in time. Some of the measurements can be made frequently; for example, heart rate, blood pressure, oxygen level, ECG, EEG, etc. Other measurements are made infrequently, for not only the cost but also the burdens to patients. Examples of these measurements include laboratory tests, CT images, MR Images, gene expressions etc. Many techniques (e.g. Fourier transform, Wavelet transform, etc. [3]) have been developed to analyze frequently made measurements where data points are sufficient, there are no missing values, and samples are regularly spaced. In most cases, these techniques are not applicable to time series measurements with imperfections; i.e. short, noisy, unaligned, irregularly sampled, partially observed, and with limited samples. A typical example of an imperfect time series is the blood cell count (e.g. white blood cell, cytokine, etc.) time series. Usually, blood cell counts are measured through time; however, they are measured *infrequently*, such that typically, a very *limited number* of samples are available. In addition, such measurements are often *noisy* due to uncontrollable factors. Sample contamination and other human errors may also cause missing measurements of some biomarkers, so the data are *partially observed* (only a subset of variables is observed at a time point). Clinical measurements are usually made by humans (nurses), so the data are *irregularly spaced*. The onset of the disease is often unknown, so the time series data are usually *unaligned*. In emergency situations, decisions should be made in a short time, so the time series may be very *short*. However, these clinical data are closely related to patients' health states; therefore, developing methodologies to efficiently analyze them could tremendously help healthcare practitioners to better serve patients.

C. Using a linear dynamical system to represent a multivariate time series

In this section, we introduce briefly the basics of a linear dynamical system (LDS) and how a multivariate time series (MTS) can be represented by an LDS.

In our work, we consider the LDS in the following form:

$$\dot{\boldsymbol{x}}(t) = \mathbf{A}\boldsymbol{x}(t), \tag{1}$$

where $t \in \mathbb{R}$ denotes time, $\boldsymbol{x}(t) \in \mathbb{R}^d$ denotes an d-dimensional state vector, $\mathbf{A} \in \mathbb{R}^{d\times d}$ is the dynamic matrix, and $\dot{\boldsymbol{x}}(t)$ is the derivative of the state vector with respect to time. If an MTS consists of samples taken from the state trajectories generated by an LDS, then we can create a connection between the MTS and LDS. Therefore, we can represent an MTS by the parameters of its corresponding LDS. We argue that we can use the initial condition, $\boldsymbol{x}_0$, and dynamics matrix, $\mathbf{A}$, to represent an MTS, for the trajectories of the states are intrinsically determined by these two parameters. In the case that the MTS's are not aligned, the initial conditions of the LDS are unknown, and we can then use the dynamics matrix, $\mathbf{A}$, to represent an MTS. For example, in the time series classification tasks in our conducted experiments, we firstly learned an LDS (represented by a parameter set $\{\boldsymbol{x}_0, \mathbf{A}\}$) from each MTS example, and then used the learned parameters as representations for model training and inference.

D. Contributions

In this work, we propose a method to learn informative representations from imperfect MTS's. The proposed method is able to overcome several important challenges of the MTS data in biomedical/healthcare researches and applications.

Contribution 1: the proposed approach learns an efficient representation from an MTS which is short and with a limited number of irregularly spaced samples. Many physiological variables, such as heart rate, blood pressure, and electrocardiogram, can be measured by specific devices. Measurements of these variables can be made noninvasively by machines at small costs; therefore, samples of the variables are usually acquired regularly and frequently. However, there are variables that are also very costly and need to be acquired by humans; as a result, only a *limited number* of samples are available. Moreover, these samples are *irregularly spaced* due to the schedule of the humans. Existing widely adapted time series analysis techniques usually require the samples of the variables to be acquired frequently and regularly. Inspired by [4], which showed that stable signals can be recovered from incomplete and inaccurate measurements, the proposed method employs an optimization-based approach to learn a continuous-time, time-invariant linear dynamical system to fit an MTS which has irregularly spaced samples; in addition, the LDS is regularized by both smoothness and ridge loss to ensure that the learned model overcomes the problem of overfitting.

Contribution 2: The proposed approach learns an efficient representation from an MTS with noisy, partially observed, and unaligned samples. Because of the limitations of the acquisition techniques and laboratory equipments, measurements tend to be *noisy*; in addition, contaminations and human errors may cause missing measurements, so the variables at a time point may be *partially observed.* The multivariate time series of different subjects/patients are usually *unaligned*; this is because the onsets of the conditions are usually unknown, hospitalization times are different, and observation lengths are different. The proposed approach uses Prediction Error Methods (PEM) to fit the state trajectories of an LDS to the MTS and simultaneously estimate the initial condition (first observation of the time series), which is corrupted by noise. There are many interpolation methods proposed to estimate the missing values in time series data; however, if the underlying missing mechanism is unknown, interpolations would introduce bias. In our method, when fitting the state trajectories to an MTS, we only consider the available values, thus, this approach does not suffer from the bias introduced by interpolations. We assume that the LDS's are *time-invariant*; therefore, even though the onsets of the conditions are unknown, the LDS's still represent the dynamics of the time series; also, because of the time-invariances of the LDS's, the observation lengths of the time series will have no impact on the learning. Because an LDS can be represented by a dynamics matrix, and the dimensions of the dynamics matrix are determined by the number of variables in the time series; therefore, an LDS is a unified representation of time series with various lengths.

II. Related Work

A. Related work in learning LDS

Linear dynamical systems have been extensively used in various fields, including engineering, medicine, economics, etc. Learning an LDS from data is a long-lasting research topic. An Expectation-Maximization approach was proposed to learn the parameters of an LDS from data, and the relationships among LDS, factor analysis, and hidden Markov models were studied [5], [6]. Subspace methods were proposed to learn the LDS parameters by fitting the state observations [7], [8]. However, the above methods assume that the states are completely observed (no missing observations) and samples are regularly spaced; moreover, when the observation durations are short, the parameters learned using the above methods are susceptible to overfitting. Recently, regularization frameworks were proposed to address this problem. A framework was developed to regularize the largest eigenvalue of the dynamics matrix $\mathbf{A}$ and learn the parameters using a spectral algorithm [9]. More recently, an L1-regularization framework [10], a low-rank regularization framework [11], and a matrix factorization based framework with regularizations [12] were also proposed. While these frameworks overcome the overfitting problem and are able to learn an LDS efficiently from fully observed state trajectory samples, they do not explicitly handle the situations in which states are partially observed. Moreover, since they are based on the formulation of a discrete-time LDS, they require that the state trajectory samples are regularly spaced. In this work, we proposed a learning algorithm to learn the dynamics matrix, $\mathbf{A}$, and the initial condition, $\boldsymbol{x}_0$, from an imperfect MTS (i.e. state trajectory samples), based on the formulation of a continuous-time LDS. Therefore, our learning algorithm does not require that the observations are regularly spaced.

B. Related work in learning representation from MTS's

Learning representations from data is crucial for many machine learning tasks [13]. Time series representation learning does not only aim to reduce the storage of a large amount of time series data, but more importantly, it aims to extract informative features for classification, prediction, or clustering. The Discrete Fourier Transform (DFT) is one of the most well-known representations for time series. The first few coefficients of the DFT were proposed to represent a time series and showed promising results in searching and indexing time series in databases [14]. The Discrete Wavelet Transform (DWT) is proposed as a good alternative to the DFT [15], for the DWT is able to capture both the global and local shapes. However, these two representations require a rigorous sampling rate and completed observations. Piecewise Aggregate Approximation (PAA) [16] is an efficient representation for long-duration time series. Symbolic Aggregate Approximation (SAX) [17] is a representation based on PAA; it is obtained by discretizing the PAA coefficients of a time series into some predefined symbols. The aim of both PAA and SAX is to reduce the length of a long time series, so they are not applicable to short time series. The concept of shapelet and a time series representation called shapelet transform were proposed [18]. A generalized shapelet-based method on multivariate time series showed advantages in early classification tasks [19]. The shapelet-based representations are able to capture the class-specific local features of time series; however, they are not applicable to a time series that is partially observed or irregularly sampled. Discrete-time Linear Dynamical Systems (DTLDS's) were proposed as kernels for a Support Vector Machine classifier in the tasks of MTS classification, and demonstrated that DTLDS's were efficient as representations for MTS's [20]; however, the learned DTLDS's were susceptible to overfitting due to the lack of regularization, and are only applicable to MTS's with fixed lengths, due to the constraints in the learning algorithm.

III. Methods

In this section, we describe our approach to learn an LDS from an imperfect multivariate time series and use the LDS-based representation for the multivariate time series classification problem using a kernel support vector machine.

A. Notations

In this paper, scalars are denoted by lowercase alphabets (e.g., t). Vectors are represented by boldface alphabets (e.g., $\boldsymbol{x}$). Matrices are represented by boldface uppercase alphabets (e.g., $\mathbf{A}$). The $(i,j)^{th}$ element of a matrix, $\mathbf{A}$, is denoted by its lowercase alphabet with a subscript, namely a_{ij}. The identity matrix is denoted by $\mathbf{I}$ with suitable dimensions in equations. We list the main symbols in Table I.

Notation	Definition
m	The number of time points in a time series is $m+1$
t_i	$i = 0, 1, 2, \cdots, m$, the time stamp of the i-th time point
d	The Number of biomarkers measured at a time
h	A small time increment
n_i	The number of h's in between t_0 and t_i
$\boldsymbol{x}(t_i) \in \mathbb{R}^d$	The i-th biomarker measurement vector in a time series
$\hat{\boldsymbol{x}}(t_i) \in \mathbb{R}^d$	The approximation of $\boldsymbol{x}(t_i)$
$\boldsymbol{o}_i \in \{0,1\}^d$	The encoding of observed measurement at the i-th time point
$\boldsymbol{x}_0 \in \mathbb{R}^d$	Initial condition of a time series
$\mathbf{A} \in \mathbb{R}^{d\times d}$	Dynamics matrix
$\mathbf{O}_i \in \mathbb{R}^{d\times d}$	A diagonal matrix with $\boldsymbol{o}_i$ as the diagonal

TABLE I: Notations and definitions

B. Learning an LDS from an imperfect MTS

The LDS in the form of (1) is also known as the first order continuous-time autoregressive (AR) model. A general framework to estimate the higher-order continuous-time AR model was proposed in [21]; however, this framework lacked regularizations, so the learned model is highly susceptible to overfitting; more importantly, this general framework is not applicable to the imperfect MTS for which our algorithm is proposed.

Using the Euler forward method, (1) can be approximated by

$$\frac{\boldsymbol{x}(t+h) - \boldsymbol{x}(t)}{h} = \mathbf{A}\boldsymbol{x}(t), \tag{2}$$

where h is a small time increment. The state vector h units ahead of current time t is approximated by

$$\boldsymbol{x}(t+h) = (\mathbf{I} + h\mathbf{A})\boldsymbol{x}(t). \tag{3}$$

Let's say in an MTS, there are $m+1$ samples $\{\boldsymbol{x}(t_0), \boldsymbol{x}(t_1), ..., \boldsymbol{x}(t_m)\}$ which are taken from the state trajectories of an LDS at time point $\{t_0, t_1, t_2, ..., t_m\}$. $\boldsymbol{x}(t_i) \in \mathbb{R}^d$ is a state vector consists of d biomarker measurements at time t_i. Because h is small, we can approximate the future time point as the current time point with integer multiples of h's ahead. Namely,

$$t_i \simeq t_0 + n_i \cdot h, \quad i = 1, 2, ..., m$$

where $\{n_i | n_i = \lceil (t_i - t_0)/h \rceil, i = 1, 2, ..., m\}$ are positive integers. If the initial state vector is known, then the state vector at time t_i can be approximated by

$$\hat{\boldsymbol{x}}(t_i) = \boldsymbol{x}(t_0 + n_i \cdot h) = (\mathbf{I} + h\mathbf{A})^{n_i}\boldsymbol{x}(t_0) \tag{4}$$

and the squared error of the ith approximation and the sample can be computed as

$$\begin{aligned} \|\boldsymbol{e}_i\|^2 &= \|\hat{\boldsymbol{x}}(t_i) - \boldsymbol{x}(t_i)\|_2^2 \\ &= \|(\mathbf{I} + h\mathbf{A})^{n_i}\boldsymbol{x}(t_0) - \boldsymbol{x}(t_i)\|_2^2, \end{aligned} \tag{5}$$

where $\|\cdot\|_2$ is a vector L_2-norm.

C. Learning the dynamics matrix from complete and accurate measurements

To learn the dynamics matrix, $\mathbf{A} \in \mathbb{R}^{d\times d}$, we adopt the framework of regularized risk minimization [22]. Namely, we formulate an optimization problem in which the objective consists of an error term and regularization terms. The dynamics matrix is learned by minimizing such an objective; as shown in (6).

$$\underset{\mathbf{A}}{\text{minimize}} \quad J(\mathbf{A}) = J_e(\mathbf{A}) + J_r(\mathbf{A}) + J_s(\mathbf{A}); \tag{6}$$

where

$$J_e(\mathbf{A}) = \frac{1}{2m}\sum_{i=1}^{m} \|(\mathbf{I}+h\mathbf{A})^{n_i}\boldsymbol{x}_0 - \boldsymbol{x}(t_i)\|_2^2, \tag{7}$$

$$J_r(\mathbf{A}) = \frac{\lambda_1}{2d^2}\|\mathbf{A}\|_F^2, \tag{8}$$

$$J_s(\mathbf{A}) = \frac{\lambda_2 h}{2n_m}\sum_{j=1}^{n_m-1} \|(\mathbf{I}+h\mathbf{A})^{j+1}\boldsymbol{x}_0 - (\mathbf{I}+h\mathbf{A})^{j}\boldsymbol{x}_0\|_2^2. \tag{9}$$

The error term, denoted by J_e, is the sum of squared errors (5) normalized by the number of time points, m. There are two regularization terms in the objective: the term J_r denotes the ridge loss (Frobenius norm) of the dynamics matrix; minimizing such a loss aims to prevent the overfitting of the model [23]; the term J_s aims to ensure that the approximated time series are smooth [24]. Theoretically, we can apply any appropriate regularization terms (e.g. matrix 2-norm/spectral norm in [9] and nuclear norm in [11]) in this optimization formulation; however, in the considerations of computational efficiency and simplicity, we use the above regularizations which are differentiable, so that we can employ gradient descent optimization methods to solve the problem.

The limitations of this formulation are that: 1) it assumes all the measurements are accurate, especially the initial measurement; 2) it assumes all the measurements are complete, which means all the variables must be observed in a measurement. However, in many cases, these two assumptions are too strong; therefore, we have to further extend the model to accommodate the cases in which the MTS is *noisy* and *partially observed*.

D. Learning an LDS from noisy MTS measurements

If measurements are noisy, we have to formulate the objective function differently. In an LDS, the initial condition of the variables is extremely important because it is always one of the multipliers for approximating the time points, as shown in (4). Therefore, in order to learn the dynamics matrix, we need to also learn the initial state. To learn the initial state and the dynamics matrix simultaneously, we formulate the objective function of the optimization as follows:

$$\begin{aligned}\underset{\boldsymbol{x}_0,\mathbf{A}}{\text{minimize}} \quad J(\boldsymbol{x}_0,\mathbf{A}) = J_e(\boldsymbol{x}_0,\mathbf{A}) + J_r(\mathbf{A}) \\ + J_s(\boldsymbol{x}_0,\mathbf{A}) + J_i(\boldsymbol{x}_0).\end{aligned} \tag{10}$$

The first three terms, $J_e(\boldsymbol{x}_0,\mathbf{A})$, $J_r(\mathbf{A})$, and $J_s(\boldsymbol{x}_0,\mathbf{A})$, in this formulation are the same as defined in (7), (8), and (9). However, under the formulation in (6), we are only interested in solving the dynamics matrix, $\mathbf{A}$, so $J_e(\mathbf{A})$ and $J_s(\mathbf{A})$ were functions of variable $\mathbf{A}$ alone. In a more realistic formulation, (10), we are interested in both $\boldsymbol{x}_0$ and $\mathbf{A}$, and thus $J_e(\boldsymbol{x}_0,\mathbf{A})$ and $J_s(\boldsymbol{x}_0,\mathbf{A})$ are functions of both variables, $\boldsymbol{x}_0$ and $\mathbf{A}$. We use the regularization term,

$$J_i(\boldsymbol{x}_0) = \frac{\lambda_3}{2d}\|\boldsymbol{x}_0 - \boldsymbol{x}(t_0)\|_2^2 \tag{11}$$

to ensure that the optimized intitial observation is closed to the measurement.

E. Learning an LDS from partially observed MTS measurements

When only a subset of variables are observed, the error term in the objective function can be written as:

$$\begin{aligned}J_e(\boldsymbol{x}_0,\mathbf{A}) &= \frac{1}{2m}\sum_{i=1}^{m}\|\mathbf{O}_i\boldsymbol{e}_i\|_2^2 \\ &= \frac{1}{2m}\sum_{i=1}^{m}\|\mathbf{O}_i[(I+h\mathbf{A})^{n_i}\boldsymbol{x}_0 - \boldsymbol{x}(t_i)]\|_2^2,\end{aligned} \tag{12}$$

where $\mathbf{O}_i = diag(\boldsymbol{o}_i)$ is a diagonal matrix, and $\boldsymbol{o}_i = [o_{i1}, o_{i2}, ...]$ indicates whether the biomarkers are observed at time point, t_i; namely,

$$o_{ij} = \begin{cases} 1, & \text{if at time point } i, \text{the } j^{th} \text{ biomarker is observed} \\ 0, & \text{otherwise.}\end{cases}$$

F. Solving the optimization problem

It is challenging to simultaneously find $\boldsymbol{x}_0$ and $\mathbf{A}$ to minimize the objective function, especially when the objective function involves their products, as shown in (7). In order to solve this optimization problem, we employ an iterative strategy. Specifically, the algorithm minimizes the objective function, and finds the minimizers iteratively: in one iteration, the algorithm treats $\boldsymbol{x}_0$ as known, and solves for the optimal $\mathbf{A}$; in the following iteration, the algorithm treats $\mathbf{A}$ as known and solves for the optimal $\boldsymbol{x}_0$. In each iteration, we can solve the sub-problem efficiently using a gradient descent approach.

Using the identities described in [25] and [26], results in the first-order differentiation of each term in the objective function w.r.t. $\mathbf{A}$, while treating $\boldsymbol{x}_0$ was a constant:

$$\begin{aligned}\frac{\partial J_e}{\partial \mathbf{A}} &= \frac{1}{m}\sum_{i=1}^{m}\mathbf{O}_i[f_1(\mathbf{A},h,n_i) - f_1(\mathbf{A},h,0)] \\ &\quad \times f_2(\mathbf{A},h,n_i);\end{aligned} \tag{13}$$

$$\frac{\partial J_r}{\partial \mathbf{A}} = \frac{\lambda_1}{d^2}vec(\mathbf{A})^T; \tag{14}$$

$$\begin{aligned}\frac{\partial J_s}{\partial \mathbf{A}} &= \frac{\lambda_2 h}{n_m}\sum_{i=1}^{n_m-1}[f_1(\mathbf{A},h,i+1) - f_1(\mathbf{A},h,i)] \\ &\quad \times [f_2(\mathbf{A},h,i+1) - f_2(\mathbf{A},h,i)];\end{aligned} \tag{15}$$

where

$$f_1(\mathbf{A}, h, n) = ((\mathbf{I} + h\mathbf{A})^n \boldsymbol{x}_0)^T (\boldsymbol{x}_0^T \otimes \mathbf{I}); \tag{16}$$

$$f_2(\mathbf{A}, h, n) = \sum_{k=1}^{n} \binom{n}{k} h^k \sum_{j=1}^{k} (\mathbf{A}^T)^{k-j} \otimes \mathbf{A}^{j-1}. \tag{17}$$

Here, the symbol $\mathbf{I}$ represents the identity matrix with the same size of $\mathbf{A}$, $vec(\cdot)$ is the *vectorization operator* on a matrix, and the symbol $\otimes$ represents the *Kronecker product operator*.

While treating $\mathbf{A}$ as a constant, the first differentiation of the objective function terms w.r.t. the initial condition, $\boldsymbol{x}_0$, are

$$\frac{\partial J_e}{\partial \boldsymbol{x}_0} = \frac{1}{m} \sum_{i=1}^{m} \mathbf{O}_i [(\mathbf{I} + h\mathbf{A})^{n_i} \boldsymbol{x}_0 - \boldsymbol{x}(t_i)]^T (\mathbf{I} + h\mathbf{A})^{n_i} \tag{18}$$

$$\begin{aligned}\frac{\partial J_s}{\partial \boldsymbol{x}_0} = \frac{\lambda_2 h}{n_m} \sum_{j=1}^{n_m - 1} & \boldsymbol{x}_0^T [(\mathbf{I} + h\mathbf{A})^{j+1} - (\mathbf{I} + h\mathbf{A})^j]^T \\ & \times [(\mathbf{I} + h\mathbf{A})^{j+1} - (\mathbf{I} + h\mathbf{A})^j] \boldsymbol{x}_0\end{aligned} \tag{19}$$

$$\frac{\partial J_i}{\partial \boldsymbol{x}_0} = \frac{\lambda_3}{d} (\boldsymbol{x}_0 - \boldsymbol{x}(t_0))^T \tag{20}$$

The iterative procedure to learn $\mathbf{A}$ and $\boldsymbol{x}_0$ from an imperfect MTS is summarized in Procedure 1. The regularization parameters are user-specified. In our experiments, we simply set $\{\lambda_1, \lambda_2, \lambda_3\}$ as $\{1, 1, 1\}$. The objective function is non-convex; therefore, the solution may not be at the global minima. For MTS classification tasks, where a parameter set is learned from an MTS example, in order to have fair comparisons between dynamic matrices learned from MTS examples, we do not recommend to initialize $\boldsymbol{x}_0$ and $\mathbf{A}$ randomly. In our experiments, we initialized the dynamics matrix as a zero-matrix and initialize the initial condition as the initial sample. In the optimization procedure, we need to set parameters $\{\eta, Tol, N_{iter}\}$, where η is the update *step size*, Tol is the *optimality tolerance*, and N_{iter} is the *maximum iteration number*. In our experiments, we empirically found that setting $\eta = 10^{-2}$ allowed a good balance of convergence and convergent rate, and setting $\{Tol, N_{iter}\} = \{10^{-4}, 20\}$ allowed a good balance of computational time and optimality.

Procedure 1 Iteratively learning $\mathbf{A}$ and $\boldsymbol{x}_0$ from an Imperfect MTS

```
Input:
1: An MTS with time stamp set $\{t_0, t_1, \cdots, t_m\}$ and state vectors $\{\boldsymbol{x}(t_0), \boldsymbol{x}(t_1), \cdots, \boldsymbol{x}(t_m)\}$
2: A small time increment $h$    $\triangleright$ A user-decide variable
3: $\eta$, $Tol$ and $N_{iter}$ $\triangleright$ Step size, tolerance and max iteration number
Output:
1: Estimated initial state vector, $\boldsymbol{x}_0$
2: Estimated dynamics matrix, $\mathbf{A}$
Procedure:
1: initialize $\mathbf{A}$, and initialize $\boldsymbol{x}_0$ as $\boldsymbol{x}(t_0)$
2: $J_{old} \leftarrow J(\boldsymbol{x}_0, \mathbf{A})$ as definted in (10)
3: compute $\{n_1, n_2, \cdots, n_i, \cdots, n_m\}$, where $n_i = (t_i - t_0)/h$
4: $Counter_1 \leftarrow 0$
5: repeat
6:     $counter_1 + +$
       // Use BCGD Aglorithm and treat $\boldsymbol{x}_0$ as a constant
7:     $counter_2 \leftarrow 0$
8:     repeat
9:         $counter_2 + +$
10:        compute $\frac{\partial J_e}{\partial \mathbf{A}}$ using (13)
11:        compute $\frac{\partial J_r}{\partial \mathbf{A}}$ using (14)
12:        compute $\frac{\partial J_s}{\partial \mathbf{A}}$ using (15)
13:        $\frac{\partial J}{\partial \mathbf{A}} \leftarrow \frac{\partial J_e}{\partial \mathbf{A}} + \frac{\partial J_r}{\partial \mathbf{A}} + \frac{\partial J_s}{\partial \mathbf{A}}$
14:        $\mathbf{A} \leftarrow \mathbf{A} - \eta \frac{\partial J}{\partial \mathbf{A}}$
15:    until $\|\frac{\partial J}{\partial \mathbf{A}}\| < Tol$ or $counter_2 > N_{iter}$
       // Use BCGD Aglorithm and treat $\mathbf{A}$ as a constant
16:    $counter_2 \leftarrow 0$
17:    repeat
18:        $counter_2 + +$
19:        compute $\frac{\partial J_e}{\partial \boldsymbol{x}_0}$ using (18)
20:        compute $\frac{\partial J_s}{\partial \boldsymbol{x}_0}$ using (19)
21:        compute $\frac{\partial J_i}{\partial \boldsymbol{x}_0}$ using (20)
22:        $\frac{\partial J}{\partial \boldsymbol{x}_0} \leftarrow \frac{\partial J_e}{\partial \boldsymbol{x}_0} + \frac{\partial J_r}{\partial \boldsymbol{x}_0} + \frac{\partial J_s}{\partial \boldsymbol{x}_0}$
23:        $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}_0 - \eta \frac{\partial J}{\partial \boldsymbol{x}_0}$
24:    until $\|\frac{\partial J}{\partial \boldsymbol{x}_0}\| < Tol$ or $counter_2 > N_{iter}$
25:    $J_{new} \leftarrow J(\boldsymbol{x}_0, \mathbf{A})$
26:    $\Delta J \leftarrow J_{old} - J_{now}$
27:    $J_{old} \leftarrow J_{new}$
28: until $\|\Delta J\| < Tol$ or $counter_1 > N_{iter}$
29: Return $\boldsymbol{x}_0$ and $\mathbf{A}$
```

G. Classifying LDS's via a Kernel method

After learning the dynamics matrices and the initial conditions of all imperfect MTS examples in the data set, we would like to use them as representations of the MTSs for classification tasks. In previous studies, various metrics were developed to quantify the similarities/dissimilarities of two dynamical systems. Examples include the Martin distance [27] and subspace angles [28]. In this study, we adapted the kernel-based framework proposed in [29] as the basis of our LDS classification model, as this framework is a generalization of the previous two examples and can be directly integrated into a support vector machine classifier.

A kernel of two LDS's, $(\boldsymbol{x}_0, \mathbf{A})$ and $(\boldsymbol{x}'_0, \mathbf{A}')$, can be defined as

$$\begin{aligned}& k((\boldsymbol{x}_0, \mathbf{A})), (\boldsymbol{x}'_0, \mathbf{A}')) \\ & := \boldsymbol{x}_0^T \left[\int_0^\infty exp(\mathbf{A}t)^T \mathbf{W} exp(\mathbf{A}'t) \mu(t) dt \right] \boldsymbol{x}'_0,\end{aligned} \tag{21}$$

where $exp(\mathbf{A}t)$ is a matrix exponential, matrix $\mathbf{W}$ is a positive semi-definate matrix which is used to weight the different variables in the MTS's, and the function $\mu(t)$ is a discount function. This kernel can be seen as a special inner product of the trajectories of the two LDS's.

In our study, since we do not have any prior knowledge of the weights of the states in an LDS, we assume the states are equally weighted; therefore, without loss of generality, we replace matrix $\mathbf{W}$ with an identity matrix $\mathbf{I}$. The definition of the discount function, $\mu(t)$, is problem specific. In general, there are two popular choices, one is the Dirac delta function, $\mu(t) = \delta(t - \tau)$, and the other one is the exponential decay function, $\mu(t) = e^{-\lambda t}$.

When $\mu(t) = \delta(t - \tau)$, the kernel is reduced to the inner product of the state vectors at time, τ. Namely, the kernel defined in (21), with $\mathbf{W}$ replaced by $\mathbf{I}$, is reduced to

$$\begin{aligned} &k((\boldsymbol{x}_0, \mathbf{A})), (\boldsymbol{x}'_0, \mathbf{A}')) \\ &:= \boldsymbol{x}_0^T exp(\mathbf{A}\tau)^T exp(\mathbf{A}'\tau)\boldsymbol{x}'_0. \end{aligned} \tag{22}$$

This kernel is useful if we know the MTS's (or LDS's) are the most distinguishable at time, τ, *a priori*. In our case, we used the exponential decay function, in which there is no discount at time zero, and the discount become large as the time series progress. Using the exponential decay function, the kernel defined in (21), with $\mathbf{W}$ replaced by $\mathbf{I}$, can be written as

$$\begin{aligned} &k((\boldsymbol{x}_0, \mathbf{A}), (\boldsymbol{x}'_0, \mathbf{A}')) \\ &:= \boldsymbol{x}_0^T \left[\int_0^\infty exp(\mathbf{A}t)^T exp(\mathbf{A}'t)e^{-\lambda t}dt\right] \boldsymbol{x}'_0, \end{aligned} \tag{23}$$

where λ is a hyper-parameter, which controls how fast the discount increases. However, there is a restriction on the value of λ; that is $\lambda > 2\Lambda$, where $\Lambda = max(\|\mathbf{A}\|_2, \|\mathbf{A}'\|_2)$. The symbol $\|\cdot\|_2$ represents the L-2 norm operation, and when the operand is a matrix, it is also called the spectral norm, which is the largest singular value of the matrix. Such a restriction on λ could ensure the convergence of the integral.

Even though the integral in (23) converges, it is hard to compute because of the infinite sum. To simplify the integral, we first let

$$\mathbf{M} = \int_0^\infty exp(\mathbf{A}t)^T exp(\mathbf{A}'t)e^{-\lambda t}dt, \tag{24}$$

and then, by assuming both $\mathbf{A}$ and $\mathbf{A}'$ are non-singular and using integration by parts, we arrived at

$$\begin{aligned} \mathbf{M} = &\int_0^\infty exp(\mathbf{A}t)^T exp(\mathbf{A}'t)e^{-\lambda t}dt \\ &=(\mathbf{A}^T)^{-1}e^{-\lambda t}(exp(\mathbf{A}t))^T exp(\mathbf{A}'t)\Big|_0^\infty \\ &\quad - \int_0^\infty (\mathbf{A}^T)^{-1}e^{-\lambda t}exp(\mathbf{A}t)^T exp(\mathbf{A}'t)(\mathbf{A}' - \lambda\mathbf{I})dt \\ &=(\mathbf{A}^T)^{-1} \\ &\quad - (\mathbf{A}^T)^{-1}\left[\int_0^\infty e^{-\lambda t}exp(\mathbf{A}t)^T exp(\mathbf{A}'t)dt\right](A' - \lambda\mathbf{I}) \\ &=(\mathbf{A}^T)^{-1} - (\mathbf{A}^T)^{-1}\mathbf{M}(\mathbf{A}' - \lambda\mathbf{I}). \end{aligned}$$

By multiplying both sides by $(\mathbf{A}^T)^{-1}$ and arranging the terms, we obtained an equation of $\mathbf{M}$:

$$\mathbf{A}^T\mathbf{M} + \mathbf{M}\mathbf{A}' - \lambda\mathbf{M} = -\mathbf{I}. \tag{25}$$

Solving for $\mathbf{M}$ is an easier task; by vectorizing both sides of (25) and organizing the terms, we obtain

$$[\mathbf{I} \otimes \mathbf{A}^T + (\mathbf{A}')^T \otimes \mathbf{I} - \lambda\mathbf{I}']vec(\mathbf{M}) = -vec(\mathbf{I}),$$

where $\mathbf{I}'$ is an identity matrix whose number of column is the same as the dimensionality of $vec(\mathbf{M})$. Then we can solve $vec(\mathbf{M})$ as

$$vec(\mathbf{M}) = [\lambda\mathbf{I}' - \mathbf{I} \otimes \mathbf{A}^T - (\mathbf{A}')^T \otimes \mathbf{I}]^{-1}vec(\mathbf{I}). \tag{26}$$

Combining (26), (24), and the vectorized (23), we get

$$\begin{aligned} &k((\boldsymbol{x}_0, \mathbf{A})), (\boldsymbol{x}'_0, \mathbf{A}')) = vec(\boldsymbol{x}_0^T\mathbf{M}\boldsymbol{x}'_0) \\ =&[\boldsymbol{x}'_0 \otimes \boldsymbol{x}_0]^T[\lambda\mathbf{I}' - \mathbf{I} \otimes \mathbf{A}^T - (\mathbf{A}')^T \otimes \mathbf{I}]^{-1}vec(\mathbf{I}) \end{aligned} \tag{27}$$

We would like to point out that the first equality in (27) holds true because $k((\boldsymbol{x}_0, \mathbf{A}), (\boldsymbol{x}'_0, \mathbf{A}'))$ is a scalar, and its vectorization is a scalar. Comparing to (23), (27) is computable, but we need to inverse the matrix, $\lambda\mathbf{I}' - \mathbf{I} \otimes \mathbf{A}^T - (\mathbf{A}')^T \otimes \mathbf{I}$. At the first sight, one may think the matrix inversion could cost high computation overhead; however, this matrix has a sparse and block diagonal structure, so we can exploit the structure and compute its inverse rather cheaply.

In the cases where the MTS's are not aligned, namely, the initial states of the LDS's were observed at different times among individual MTS's after the onset, the terms $\boldsymbol{x}_0$ and $\boldsymbol{x}'_0$ in (27) seem to become meaningless. Therefore, we can define the kernel between dynamics matrices by

$$k(\mathbf{A}, \mathbf{A}') = tr(\mathbf{M}) \tag{28}$$

With this computable kernel defined, we can employ the support vector machine model for classification.

IV. Experiments and Results

To evaluate our approach, we have conducted experiments on both synthetic and real-life data.

A. Experiments on synthetic MTS data

The purpose of the experiments on synthetic MTS data is to check whether the dynamics matrices learned by our approach can preserve the similarities of the underlying LDS's. Based on the state trajectory generation procedure, we had two experiments.

Sub-Experiment 1: noisy initial states

In this experiment, we generated the state trajectories based on the following procedure:

1) Randomly generate K LDS's; i.e. K tuples of $(\boldsymbol{x}_0, \mathbf{A})$, where $\boldsymbol{x}_0 \in \mathbb{R}^5$ and $\mathbf{A} \in \mathbb{R}^{5\times5}$.
2) Add random noise to a initial state and generate state trajectories using Euler forward method (eq. 4), and repeat L times for each $(\boldsymbol{x}_0, \mathbf{A})$ tuple.

For ease of visualization, we set $K = 3$ and $L = 10$ in our experiments; therefore, there are 3 different LDS's, and each LDS is used to generate 10 trajectories with noisy initial states. MTS's were obtained by sampling the trajectories at 20 equally spaced time points. At the end, we have 30 MTS's with 20 time points and equal lengths. We applied our algorithm on the MTS's to learn the dynamics matrices. The pairwise similarity of dynamics matrices, $sim_{\mathbf{A}}$, is computed by (29).

$$sim_{\mathbf{A}}(i,j) = \|\mathbf{A}_i - \mathbf{A}_j\|_F \tag{29}$$

Based on the pairwise similarities, we projected the learned dynamics matrices on a 2D space using multidimensional scaling [30]. Each object in the 2D space represents a learned dynamics matrix, thus, ideally, dynamics matrices learned from the MTS's sampled from the same LDS (with noisy initial states) should form a cluster. For comparison, we also computed the pairwise similarity of the generated trajectories, $sim_{\boldsymbol{x}}$, by using (30)

$$sim_{\boldsymbol{x}}(i,j) = \sum_k \|\boldsymbol{x}_i(t_k) - \boldsymbol{x}_j(t_k)\|_2 \tag{30}$$

Although the dynamics matrices were the same, trajectories generated from two noisy initial states varied significantly, as shown in Figure 1a and 1b, in which Example 1 and Example 2 are trajectories generated from the same LDS, with the same dynamics matrix but noisy initial states. These variations of trajectories are also shown in the trajectory similarity plot, as shown in Figure 1c), in which the symbol of Example 1 and the symbol of Example 2 are far apart. Despite the variations in the trajectories, the learned dynamics matrices preserved the similarities of the LDS's, as shown in Figure 1d), in which the symbol of Example 1 and the symbol of Example 2 are close. Therefore, our learning approach is able to learn the underlying LDS despite the initial states being noisy.

Sub-Experiment 2: noisy dynamics matrices

This experiment was similar to the above experiment; however, in the trajectory generation procedure, instead of making the initial states noisy, we made the dynamics matrices noisy. Using the same parameter settings (i.e. $K = 3$ and $L = 10$), we generated 30 trajectories and thus 30 MTS's. Dynamics matrices were learned from the generated MTS's.

Trajectories generated from the same LDS with noisy dynamics matrices also varied, as shown in Figure 2a and 2b, in which, Example 1 and Example 2 are trajectories generated from the same LDS, with the same initial states but noisy dynamics matrices. The trajectories generated with noisy dynamics matrix may vary significantly, as shown in Figure 2c, in which Example 1 and Example 2 are far apart. But the dynamics matrices learned by our algorithm preserved the similarities much better, as shown in Figure 2d, in which the clusters of the 3 LDS's are very distinct.

B. Experiments on real-life datasets

1) Datasets and baselines: To evaluate the efficiency of using LDS's as representations of imperfect MTS's, we conducted experiments on three human blood gene expression time series datasets, influenza A (H3N2), Rhinovirus (HRV), and respiratory syncytial virus (RSV) [31]. The number of MTS's in the H3N2, HRV, and RSV datasets were 17, 20, and 20, respectively. At each time point of the MTS, expression levels of multiple genes were measured. The number of genes in each dataset was originally over 10,000. In our experiments, in each dataset, we only included the genes suggested by a previous study [31]; therefore, the number of genes included in the H3N2, HRV, and RSV datasets were 23, 26, and 24, respectively. Each MTS in the datasets has a different number of time points (i.e., the number of temporal gene expression measurements was different from subject to subject). In the H3N2 dataset, most of the subjects had 16 temporal gene expression measurements, and thus there were 16 time points available in the MTS's of those subjects; a small number of subjects in the H3N2 dataset had 15 time points available in their MTS's. The numbers of temporal gene expressions in the subjects in the HRV dataset varied significantly, ranging from 7 to 14. The numbers of temporal gene expressions in the subjects in the RSV dataset varied even more significantly, ranging from 6 to 21. In the MTS's of the H3N2 dataset, every two consecutive gene expressions were spaced by 1 time unit. However, the gene expressions in MTS's of the HRV and RSV datasets were not equally spaced. The interval of two consecutive gene expressions may be separated as much as 7 units and as little as 1 unit in the HRV dataset. The interval of two consecutive gene expressions may be separated as much as 10 units and as little as 1 unit in the RSV dataset. A summary of the datasets is given in Table II. MTS's in the H3N2 dataset are close to "perfect"; namely, most of them have the same lengths, and samples are regularly spaced. In contrast, the RSV dataset is a good demonstration of an imperfect MTS dataset, in which both lengths and sampling intervals of the MTS's vary significantly. Within each dataset, we formulated a binary classification task to determine whether an MTS was from an individual with symptomatic acute respiratory infection or an individual with no infection.

TABLE II: Summary of 3 datasets

	H3N2	HRV	RSV
Number of genes	23	26	24
Number of MTS's (pos/neg)	9/8	10/10	9/11
Number of time points (range)	15-16	7-14	6-21
Sampling interval (range)	1	1-7	1-10

For comparison, in our experiments, we also included three baseline representations/methods:

- ***stat + SVM*** Using statistical summaries of temporal samples as time series features has been proven effective in biomedical applications [32]. In our experiments, for each variable in an MTS instance, we computed four statistical measures (*mean*, *standard deviation*, *maximum*, *minimum*) to summarize the time series. That is, if there are n variables, in this approach, an MTS is represented by a $4n$-dimensional feature vector. After the features were generated, we used a support vector machine (SVM) model as the classifier. The kernels used in the SVM model were linear, radial basis function, and polynomial

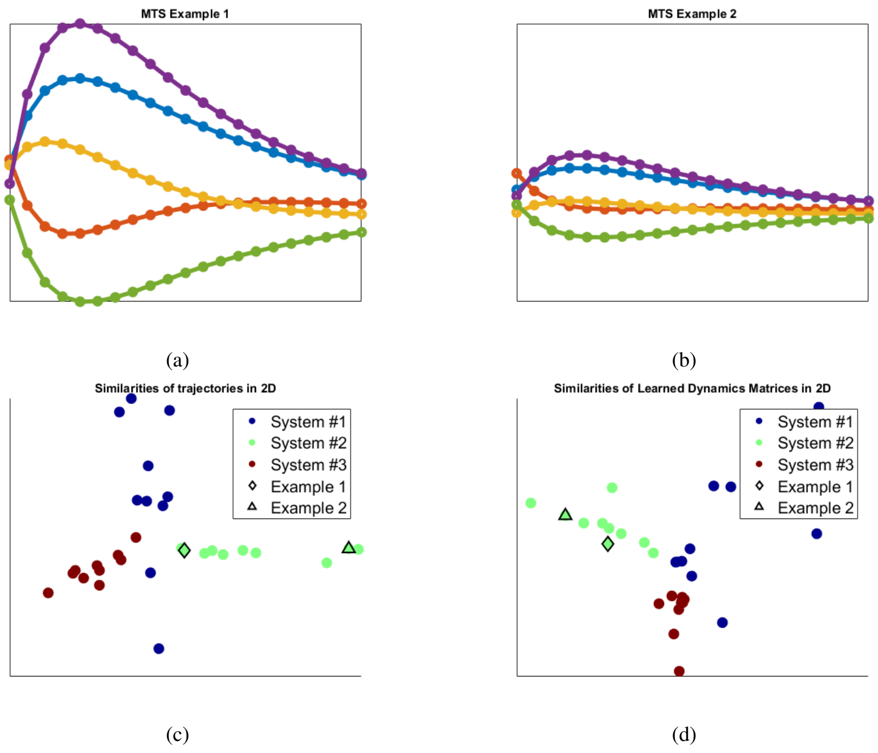

Fig. 1: Examples of trajectories generated by the same LDS with noisy initial conditions and similarity plots of generated trajectories and learned dynamics matrices. a) and b) two example trajectories generated by the same LDS with noisy initial states; the same color indicates the same state. c) and d) similarity plots of generated trajectories and learned dynamics matrices; the same color indicates the trajectories generated from the same LDS.

of order 3. The best result among the different kernels will be reported. The hyperparameter in the model was determined by a nested leave-one-out cross-validation in the training set.

- ***PAA + 1NN*** Piecewise Aggregate Approximation (PPA) [16] is an efficient representation for long time series. In our experiment, for each (MTS) instance, we applied PAA on each variable, and then concatenated all the variables as a feature vector. The feature vectors of all the instances have the same dimensionalities. After the transformation, we used 1-*nearest-neighbor* as the classification model.
- ***DTLDS-kernel SVM*** In our experiments, we implemented this baseline based on [20], in which a singular value decomposition based approach is used to learn a discrete time linear dynamical system (DTLDS) from an MTS with regularly spaced samples. Then a classification was performed on an SVM classifier by constructing a kernel using learned DTLDS's. This method is similar to our proposed method; however, it is not applicable to time series with imperfectness; therefore, in order to use this method on the MTS's used in the experiments, interpolation and(or) truncation of the time series were necessary. More details are given in the following section about data preprocessing.

C. Data preprocessing

Different genes are expressed in different scales; therefore, we needed to normalize the MTS before learning and modeling. In our experiments, we scaled each gene in all MTS into the (0,1) interval [33]. In some of the baseline models (i.e. *PAA + 1NN* and *DTLDS-kernel SVM*), gene expressions are required to be measured regularly; therefore, for MTS's with irregularly spaced measurements, we performed linear interpolations. In addition, in *PAA + 1NN* and *LDS-kernel SVM* models, MTS's are required to be of equal length. In our experiments, we truncated the MTS's in a dataset to the length of the shortest MTS in that dataset.

D. Experiments on using all available data in the datasets

In this experiment, we ran all the baseline methods and the proposed method on all three datasets and used all available time points in each dataset. Due to the limited number of MTS in each dataset (Table II), in our experiments, the results were obtained by using leave-one-out cross-validations. The prediction accuracy of each representation and model is shown in Table III.

All the representations/methods performed equally extremely well on H3N2. That is not surprising because instances in that dataset are of very high quality (see Table II): the lengths of the MTS's are almost equal, samples are acquired in regular intervals, and all MTS's have a decent number of

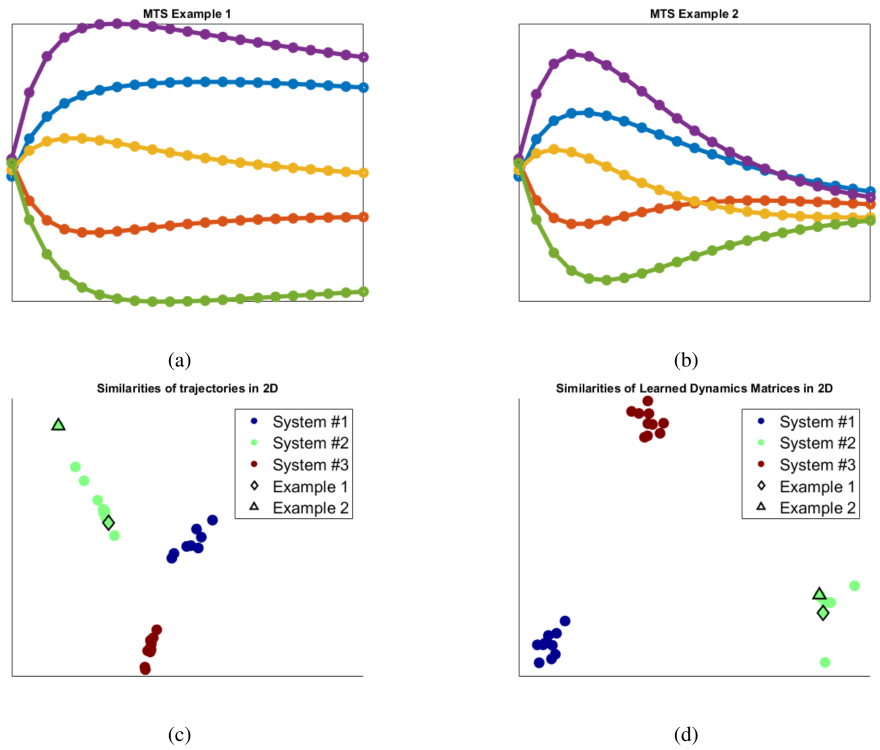

Fig. 2: Examples of trajectories generated by the same LDS with noisy dynamics matrices and similarity plots of generated trajectories and learned dynamics matrices. a) and b) two example trajectories generated by the same LDS with noisy initial states; the same color indicates the same state. c) and d) similarity plots of generated trajectories and learned dynamics matrices; the same color indicates the trajectories generated from the same LDS.

TABLE III: Leave-one-out cross validation accuracies on 3 binary classification tasks. Best performances are in bold

	H3N2	HRV	RSV
PAA + 1NN	**1.000**	0.800	0.750
stat + SVM	**1.000**	0.850	0.750
DTLDS-kernel SVM	**1.000**	0.750	0.650
proposed	**1.000**	**1.000**	**0.800**

time points. In contrast, the accuracies on HRV and RSV were negatively affected by the imperfectness of MTS's, such that all the baseline methods could not achieve as high accuracy as on H3N2. Our proposed method has the best accuracies across all datasets. It is worth pointing out that even in an imperfect MTS HRV dataset, our proposed method still achieved 100% accuracy, while the second best accuracy was only 85%.

E. Experiments on the H3N2 dataset with imposed imperfectness

When using all available time points in the H3N2 dataset, all the models achieved perfect classification accuracies. In the following experiments, we manually imposed imperfectness into the H3N2 dataset to characterize the performance of four methods when affected by various levels of defects seen in real-life applications.

MTS's with missing time points In this experiment, we removed completely at random approximately 20%, 40%, 60%, and 80% of the time points from the original MTS's. Because each MTS in the dataset has 15 to 16 time points, after the removal, there were 13, 10, 7, and 4 time points left. We repeated the removal process for 5 times, and each time, different time points were removed. By doing this, we can emulate imperfectness such as irregular sampling intervals and potentially unaligned MTS's. Then we applied the classification methods on the imperfect MTS's. The mean accuracies of each method and their standard deviations as the percentage missing increases are shown at Figure 3.

From Figure 3, we notice that the *DTLDS-kernel SVM* was affected the most by irregular sampling intervals. The performances of *PAA+1NN* and *stat+SVM* were affected minimally, as they both kept 100% accuracy when up to 40% of time points were missing, and their accuracies only degraded around 5% when 80% of time points were missing. Our proposed method achieved 100% accuracy even when 80% of data were missing.

MTS's with truncations In this experiment, we imposed imperfectness by randomly truncating the leading and ending time points from individual MTS's; specifically, we randomly truncated 25%, 50%, and 75% time points at the beginning, at the end, or both (at the beginning and the end) from the original MTS's. By doing this, we obtained MTS's which

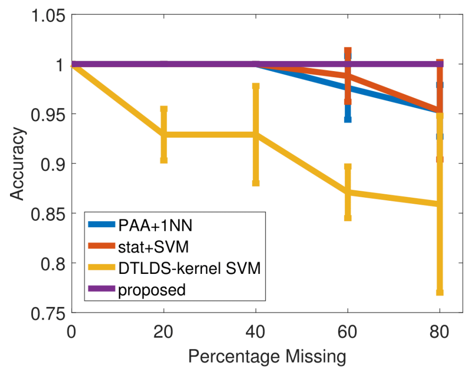

Fig. 3: Means and standard deviations of each methods as the percentage of missing time points increases from 20% to 80%. The vertical line indicates one standard deviation.

are unaligned, short, and with a limited number of samples. After the truncation, each MTS had 12, 8, and 4 regularly spaced time points. We repeated the truncation process 5 times, and at each time, different portions of the leading and ending time points would be truncated. The mean accuracies of each method and their standard deviations as the percentage truncation increases are shown at Figure 4.

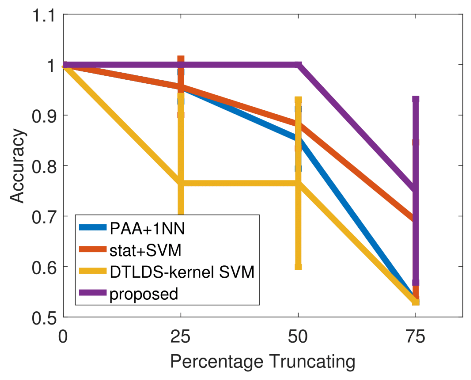

Fig. 4: Means and standard deviations of each methods as the percentage truncating increases from 20% to 80%. The vertical line indicates one standard deviation.

In contrast to the previous experiment, the accuracies of all the methods were affected more in this experiment, although the MTS's had roughly the same number of time points. When time points were removed uniformly and randomly, the length of an MTS might not be reduced; however, in this experiment, we did not only reduce the number of time points in an MTS but also reduced its length. Therefore, methods such as *PAA+1NN* and $stat + SVM$, which are highly dependent on the trend and the values (usually the minimal/maximum values) at the beginning/end of a process, were hurt more by this kind of data deficiency. Our proposed method kept its perfect classification accuracy until the percentage of truncating exceed 50%. The proposed method was able to continue performing well because the learned LDS's were able to capture the unique characteristics of the dynamics in the MTS's from different classes.

V. Conclusion

In this paper, we proposed a method to learn continuous-time LDS's from MTS's with various forms of imperfectness; i.e. limited time points, irregular sampling intervals, unaligned, noisy, partially observed, and short spanned. By adopting a powerful LDS kernel formulation, we employed a support vector machine model for classification tasks. Empirical results on three diagnostic tasks with different levels of imperfectness provided evidence that our proposed method is effective and able to outperform alternative methods.

VI. Acknowledgment

This study is partially supported by the Defense Advanced Research Projects Agency (DARPA) and the Army Research Office (ARO) under Contract no. DARPA-THOR-W911NF-16-C-0050.

References

[1] N. Omranian, B. Mueller-Roeber, and Z. Nikoloski, "Segmentation of biological multivariate time-series data," *Scientific reports*, vol. 5, 2015.
[2] A. D. Henn, S. Wu, X. Qiu, M. Ruda, M. Stover, H. Yang, Z. Liu, S. L. Welle, J. Holden-Wiltse, H. Wu *et al.*, "High-resolution temporal response patterns to influenza vaccine reveal a distinct human plasma cell gene signature," *Scientific reports*, vol. 3, 2013.
[3] R. M. Rangayyan, *Biomedical signal analysis*. New Jersey: John Wiley & Sons, 2015, vol. 33.
[4] E. J. Candes, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on pure and applied mathematics*, vol. 59, no. 8, pp. 1207–1223, 2006.
[5] Z. Ghahramani and G. E. Hinton, "Parameter estimation for linear dynamical systems," *University of Toronto technical report CRGTR962*, vol. 6, no. CRG-TR-96-2, pp. 1–6, 1996.
[6] R. H. Shumway and D. S. Stoffer, "An approach to time series smoothing and forecasting using the em algorithm," *Journal of time series analysis*, vol. 3, no. 4, pp. 253–264, 1982.
[7] T. Katayama, *Subspace methods for system identification*. New York: Springer Science & Business Media, 2006.
[8] P. Van Overschee and B. De Moor, *Subspace identification for linear systems: Theory—Implementation—Applications*. New York: Springer Science & Business Media, 2012.
[9] B. Boots, G. J. Gordon, and S. M. Siddiqi, "A constraint generation approach to learning stable linear dynamical systems," in *Advances in Neural Information Processing Systems*, 2007, pp. 1329–1336.
[10] N. Städler, S. Mukherjee *et al.*, "Penalized estimation in high-dimensional hidden markov models with state-specific graphical models," *The Annals of Applied Statistics*, vol. 7, no. 4, pp. 2157–2179, 2013.
[11] Z. Liu and M. Hauskrecht, "A regularized linear dynamical system framework for multivariate time series analysis," in *Proceedings of the... AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence*, vol. 2015. NIH Public Access, 2015, p. 1798.
[12] ——, "Learning linear dynamical systems from multivariate time series: A matrix factorization based framework."
[13] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 8, pp. 1798–1828, 2013.
[14] R. Agrawal, C. Faloutsos, and A. Swami, *Efficient similarity search in sequence databases*. New York: Springer, 1993.
[15] K.-P. Chan and A. W.-C. Fu, "Efficient time series matching by wavelets," in *Data Engineering, 1999. Proceedings., 15th International Conference on*. IEEE, 1999, pp. 126–133.
[16] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," *Knowledge and information Systems*, vol. 3, no. 3, pp. 263–286, 2001.

[17] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*. ACM, 2003, pp. 2–11.

[18] L. Ye and E. Keogh, "Time series shapelets: a new primitive for data mining," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 947–956.

[19] M. F. Ghalwash and Z. Obradovic, "Early classification of multivariate temporal observations by extraction of interpretable shapelets," *BMC bioinformatics*, vol. 13, no. 1, p. 1, 2012.

[20] K. M. Borgwardt, S. Vishwanathan, and H.-P. Kriegel, "Class prediction from time series gene expression profiles using dynamical systems kernels." in *Pacific symposium on biocomputing*, vol. 11, 2006, pp. 547–558.

[21] A. Harvey and J. H. Stock, "The estimation of higher-order continuous time autoregressive models," *Econometric Theory*, vol. 1, no. 01, pp. 97–117, 1985.

[22] K. P. Murphy, *Machine learning: a probabilistic perspective*. Cambridge, Boston: MIT press, 2012.

[23] V. N. Vapnik and V. Vapnik, *Statistical learning theory*. New York: Wiley New York, 1998, vol. 1.

[24] J. D. Hamilton, *Time series analysis*. Princeton: Princeton university press, 1994, vol. 2.

[25] J. Gallier, *Geometric methods and applications: for computer science and engineering*. New York: Springer Science & Business Media, 2011, vol. 38.

[26] B. Chen and P. A. Zadrozny, "Analytic derivatives of the matrix exponential for estimation of linear continuous-time models," *Journal of Economic Dynamics and Control*, vol. 25, no. 12, pp. 1867–1879, 2001.

[27] R. J. Martin, "A metric for arma processes," *IEEE Transactions on Signal Processing*, vol. 48, no. 4, pp. 1164–1170, 2000.

[28] K. De Cock and B. De Moor, "Subspace angles between arma models," *Systems & Control Letters*, vol. 46, no. 4, pp. 265–270, 2002.

[29] S. Vishwanathan, A. J. Smola, and R. Vidal, "Binet-cauchy kernels on dynamical systems and its application to the analysis of dynamic scenes," *International Journal of Computer Vision*, vol. 73, no. 1, pp. 95–119, 2007.

[30] I. Borg and P. J. Groenen, *Modern multidimensional scaling: Theory and applications*. New York: Springer Science & Business Media, 2005.

[31] A. K. Zaas, M. Chen, J. Varkey, T. Veldman, A. O. Hero, J. Lucas, Y. Huang, R. Turner, A. Gilbert, R. Lambkin-Williams *et al.*, "Gene expression signatures diagnose influenza and other symptomatic respiratory viral infections in humans," *Cell host & microbe*, vol. 6, no. 3, pp. 207–217, 2009.

[32] K. E. Henry, D. N. Hager, P. J. Pronovost, and S. Saria, "A targeted real-time early warning score (trewscore) for septic shock," *Science Translational Medicine*, vol. 7, no. 299, pp. 299ra122–299ra122, 2015.

[33] X. H. Cao and Z. Obradovic, "A robust data scaling algorithm for gene expression classification," in *Bioinformatics and Bioengineering (BIBE), 2015 IEEE 15th International Conference on*. IEEE, 2015, pp. 1–4.