Learning Nonlinear Dynamic Models from Non-sequenced Data

Tzu-Kuo Huang
Machine Learning Department
Carnegie Mellon University

Le Song
Machine Learning Department
Carnegie Mellon University

Jeff Schneider
Robotics Institute
Carnegie Mellon University

Abstract

Virtually all methods of learning dynamic systems from data start from the same basic assumption: the learning algorithm will be given a sequence of data generated from the dynamic system. We consider the case where the training data comes from the system's operation but with no temporal ordering. The data are simply drawn as individual disconnected points. While making this assumption may seem absurd at first glance, many scientific modeling tasks have exactly this property.

Previous work proposed methods for learning linear, discrete time models under these assumptions by optimizing approximate likelihood functions. We extend those methods to nonlinear models using kernel methods. We go on to propose a new approach that focuses on achieving temporal smoothness in the learned dynamics. The result is a convex criterion that can be easily optimized and often outperforms the earlier methods. We test these methods on several synthetic data sets including one generated from the Lorenz attractor.

1 Introduction

Learning dynamic systems from data is the traditional topic of system identification in control theory and many algorithms have been proposed. In the machine learning literature, the learning of graphical models, such as dynamic Bayesian networks, and the learning of various types of Markov models have been studied for the same problem, often with discrete state spaces.

Virtually all of these methods start from the same basic assumption: that the learning algorithm will be provided with a sequence of data generated from the dynamic system. Here we consider the case where the data is not sequenced. The learning algorithm is presented a set of data points from the system's operation but with no temporal ordering. The data are simply drawn as individual disconnected points, and may come from many separate executions of the dynamic system.

Many scientific modeling tasks have this property. Consider the task of learning dynamic models of galaxy or star evolution. These processes are far too slow for us to collect successive data points showing any meaningful changes. However, we do have billions of single data points showing these objects at various stages of their evolution. At more modest time scales, the same problem arises in the understanding of slow-moving human diseases such as Alzheimer's or Parkinson's, which may progress over a decade or more. At the other end of the spectrum, cellular or molecular biological processes may be too small or too fast to permit collection of trajectories from the system. Often, the measurement techniques are destructive and thus only one data point can be collected from each sample.

In these applications, scientists would like to construct a dynamic model using only unsequenced, individual data points collected from the system of interest. In previous work, two algorithms were proposed to solve this problem for linear, continuous-state, discrete-time systems. These methods were based on optimizing an approximate likelihood function. In this paper, we extend those methods to the non-linear case using kernel methods. We go on to propose a new method based on achieving temporal smoothness in the learned dynamics.

Appearing in Proceedings of the 13^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2010, Chia Laguna Resort, Sardinia, Italy. Volume 9 of JMLR: W&CP 9.

2 Learning linear dynamical systems from non-sequenced data

(Huang & Schneider, 2009) first proposed and studied the problem of learning linear dynamical systems from a set of unordered observations. They consider a simple discrete-time linear system:

$$\mathbf{x}^{t+1} = A\mathbf{x}^t + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 I), \tag{1}$$

and assume their data consists of N observations $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, each of which is drawn from an different execution of (1) at an unknown, random point in time. Since each data point comes from exactly one trajectory, (Huang & Schneider, 2009) write down the likelihood by integrating out the true predecessor for every observation:

$$\prod_{i=1}^{N} \left(\int_{\mathbf{x}\in\chi} \frac{\exp(-\frac{\|\mathbf{x}_i - A\mathbf{x}\|^2}{2\sigma^2})}{(2\pi\sigma^2)^{\frac{n}{2}}} f(\mathbf{x}) d\mathbf{x} \right), \tag{2}$$

where $f(\mathbf{x})$ denotes the distribution of $\mathbf{x}$. They then approximate the true distribution $f(\mathbf{x})$ by the data itself, and propose estimating the system matrix A by maximizing the following approximate log likelihood:

$$\max_{A,\sigma^2} \sum_{i=1}^{N} \log \left(\sum_{j\neq i} \frac{\exp(-\frac{\|\mathbf{x}_i - A\mathbf{x}_j\|^2}{2\sigma^2})}{(N-1)(2\pi\sigma^2)^{\frac{n}{2}}} \right), \tag{3}$$

for which a simple and efficient Expectation Maximization procedure is derived. They refer to this model as the Unordered Model (UM), as the estimation procedure does not take into account the underlying temporal relation of the data points.

To impose directionality consistency into the model, they require that each pair of points can only appear in one direction, and that the overall structure must be a tree:

$$\begin{aligned} \max_{\substack{A,\sigma^2,\boldsymbol{\omega},\\ r\in\{1,\ldots,N\}}} \quad & \sum_{\substack{i=1,\\ i\neq r}}^{N} \log \sum_{j=1}^{N} \left(\frac{\exp(-\frac{\|\mathbf{x}_i - A\mathbf{x}_j\|^2}{2\sigma^2})}{(2\pi\sigma^2)^{\frac{n}{2}}} \boldsymbol{\omega}_{ij} \right) \\ \text{s.t.} \quad & \boldsymbol{\omega}_{ij} = \{0,1\}, \\ & \sum_{j=1}^{N} \boldsymbol{\omega}_{ij} = 1,\ i \neq r,\ \sum_{j=1}^{N} \boldsymbol{\omega}_{rj} = 0, \\ & \boldsymbol{\omega} \text{ forms a tree with root } \mathbf{x}_r. \end{aligned} \tag{4}$$

where r denotes the data index of the starting point. They call the modified model as the Partial-ordered model, and derive an alternating maximization procedure to estimate the model parameters and the tree structure.

The above two models work quite well in the experiments in (Huang & Schneider, 2009). However, a major drawback of them is the non-convex estimation procedure. In this paper, we first generalize these two models for nonlinear dynamical systems through *kernel regression*, and then propose a new formulation of modeling nonlinear dynamics as smooth trajectories in time. One main advantage of the new approach is that the estimation procedure is a convex optimization problem. Moreover, it does not assume a parametric transition model. Instead of data generated from multiple trajectories, the new approach aims at the case where the data consists of points along a *single trajectory* of a nonlinear system, but the temporal order of the points is unknown.

3 A nonlinear auto-regressive model via kernel regression

To generalize the models proposed in (Huang & Schneider, 2009) for nonlinear dynamics, we use kernel regression. The system equation takes the following form:

$$\mathbf{x}^{t+1} = W\phi(\mathbf{x}^t) + \boldsymbol{\epsilon}, \tag{5}$$

where $\phi(\cdot)$ maps a point in $\mathbb{R}^n$ into a Reproducing Kernel Hilbert Space (RKHS) endowed with a kernel function $\mathcal{K}(\mathbf{x},\mathbf{y}) = \phi(\mathbf{x})^\top \phi(\mathbf{y})$, and W is a linear mapping from the RKHS to $\mathbb{R}^n$. We assume that the process noise follows a Gaussian distribution $\boldsymbol{\epsilon} \sim N(0, \sigma^2 I)$. To avoid over-fitting, we impose a zero-mean, unit-variance Gaussian prior on W and obtain the following approximate posterior:

$$\begin{aligned} &\hat{P}(X|W,\sigma^2) \\ &\propto \prod_{i=1}^{N} \left(\sum_{j\neq i} \frac{\exp(-\frac{\|\mathbf{x}_i - W\phi(\mathbf{x}_j)\|^2}{2\sigma^2})}{(N-1)(2\pi\sigma^2)^{\frac{n}{2}}} \right) \exp\left(-\frac{\lambda \|W\|_{Fro}^2}{2} \right) \end{aligned} \tag{6}$$

as a nonlinear version of the Unordered Model (3). To estimate W and σ^2, we derive an EM algorithm similar to Algorithm 1 in (Huang & Schneider, 2009), which can be viewed as an instance of the Iterative Re-weighted Least Squares method, alternating between estimating model parameters and re-weighting each pair of observations. Let $Z \in \{0,1\}^{N\times N}$ be a latent variable matrix indicating which observation is generated by which:

$$Z_{ij} = \begin{cases} 1 & \text{if } \mathbf{x}_i \text{ is generated from } \mathbf{x}_j \\ 0 & \text{otherwise} \end{cases}, \quad j \neq i,$$

$$Z_{ii} = 0,\ \sum_{j=1}^{N} Z_{ij} = 1.$$

Algorithm 1 Expectation Maximization for (6)

Input: Data points $\mathbf{x}_1, \ldots, \mathbf{x}_N$
Compute the kernel matrix K with $K_{ij} = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$.
Compute a low-rank factorization $K \approx \tilde{\phi}(X)^\top \tilde{\phi}(X)$.
Initialize W^1 and $(\sigma_2)^1$, set $t = 0$
repeat
  Update $\tilde{Z}^{t+1}$ by (8) with W^t and $(\sigma^2)^t$
  Update W^{t+1} by (9) with $\tilde{\phi}(X)$ and $\tilde{Z}^{t+1}$
  Update $(\sigma^2)^{t+1}$ by (11) with W^{t+1} and $\tilde{Z}^{t+1}$
  $t \leftarrow t + 1$
until $\|\tilde{Z}^t - \tilde{Z}^{t-1}\|_{Fro}^2 < \epsilon$

Then we can write the complete log posterior as

$$\begin{aligned}
&\log \hat{P}(X, Z|W, \sigma^2) \qquad (7)\\
&= \log \prod_{i=1}^{N}\prod_{j=1}^{N} \left(\frac{\exp(-\frac{\|\mathbf{x}_i - W\phi(\mathbf{x}_j)\|^2}{2\sigma^2})}{(N-1)(2\pi\sigma^2)^{\frac{n}{2}}} \right)^{Z_{ij}} - \frac{\lambda \|W\|_{Fro}^2}{2}\\
&\propto -\sum_{i,j} Z_{ij} \left(\frac{\|\mathbf{x}_i - W\phi(\mathbf{x}_j)\|^2}{2\sigma^2} + \frac{n\log(\sigma^2)}{2} \right) - \frac{\lambda\|W\|_{Fro}^2}{2}.
\end{aligned}$$

In the E-step, we compute the posterior mean $\tilde{Z}_{ij}$:

$$\begin{aligned}
\tilde{Z}_{ij} &= P(Z_{ij} = 1|X, W, \sigma^2)\\
&= \begin{cases} \frac{\exp(-\frac{\|\mathbf{x}_i - W\phi(\mathbf{x}_j)\|^2}{2\sigma^2})}{\sum_{s\neq i}\exp\left(-\frac{\|\mathbf{x}_i - W\phi(\mathbf{x}_s)\|^2}{2\sigma^2}\right)}, & i \neq j,\\ 0, & i = j. \end{cases} \qquad (8)
\end{aligned}$$

In the M-step, we replace Z_{ij} by $\tilde{Z}_{ij}$ and maximize (7) on W and σ^2. This is equivalent to fitting a *weighted* least squares regression with $\tilde{Z}_{ij}$'s as the weights, and the solution has a simple form:

$$\begin{aligned}
W &= X\tilde{Z}\phi(X)^\top \left(\phi(X)\Lambda_{\tilde{Z}}\phi(X)^\top + \lambda\sigma^2 I\right)^{-1} \qquad (9)\\
&= X\tilde{Z}\left(K\Lambda_{\tilde{Z}} + \lambda\sigma^2 I\right)^{-1}\phi(X)^\top, \qquad (10)\\
\sigma^2 &= \frac{\sum_{i=1}^N\sum_{j=1}^N \tilde{Z}_{ij}\|\mathbf{x}_i - W\phi(\mathbf{x}_j)\|^2}{n\sum_{i=1}^N\sum_{j=1}^N \tilde{Z}_{ij}}, \qquad (11)
\end{aligned}$$

where $\phi(X) \equiv [\phi(\mathbf{x}_1)\ \phi(\mathbf{x}_2)\ \cdots\ \phi(\mathbf{x}_N)]$ is the mapping of the entire set of observations in the RKHS, $\Lambda_{\tilde{Z}}$ is a diagonal matrix with $(\Lambda_{\tilde{Z}})_{ii} = \sum_{j=1}^N \tilde{Z}_{ji}$, and $K \equiv \phi(X)^\top\phi(X)$ is the kernel matrix. We obtain (10) from (9) by using the Matrix Inversion lemma.

One issue with the above procedure is that we cannot compute W when the mapping $\phi(\cdot)$ is of infinite dimension. However, we observe that the EM procedure only requires the computation of $W\phi(X)$, and according to (10)

$$\begin{aligned}
W\phi(X) &= X\tilde{Z}\left(K\Lambda_{\tilde{Z}} + \lambda\sigma^2\right)^{-1}\phi(X)^\top\phi(X)\\
&= X\tilde{Z}\left(K\Lambda_{\tilde{Z}} + \lambda\sigma^2\right)^{-1}K.
\end{aligned}$$

Therefore, instead of W we maintain and update an n-by-N matrix $B \equiv X\tilde{Z}\left(K\Lambda_{\tilde{Z}} + \lambda\sigma^2\right)^{-1}$ in the EM iterations. To predict the next state for a new observation $\mathbf{x}$, we compute $B\phi(X)^\top\phi(\mathbf{x})$, which also only requires kernel evaluations. Alternatively, we may compute a finite-dimensional approximation to $\phi(X)$ by doing a low-rank factorization of the kernel matrix $K \approx \tilde{\phi}(X)^\top\tilde{\phi}(X)$, and replace $\phi(X)$ in the EM procedure with $\tilde{\phi}(X) \in \mathbb{R}^{m\times N}, m < N$. Then we can maintain and update $W \in \mathbb{R}^{n\times m}$ explicitly. We outline such a procedure in Algorithm 1. To do prediction on a set of new data points, we project them onto the basis found by factorizing the training kernel matrix, thereby computing their finite-dimensional approximation $\tilde{\phi}$, and then apply the estimated W to the mapped points.

A similar nonlinear extension to the Partial-ordered Model (4) is straightforward. By using the same kind of kernel operations as above, we obtain an estimation procedure similar to Algorithm 1 with the E-step replaced by a maximum spanning tree search as in Algorithm 2 in (Huang & Schneider, 2009) and the same M-step.

4 Learning nonlinear dynamics by temporal smoothing

We begin by observing a drawback in the objective function (6). The dynamics learned by these methods are ensured of being spatially smooth. In other words, if you have two nearby states, the temporal gradients from those two states will be similar. However, temporal smoothness is not enforced. In other words, on one time step, the system may make a large jump to another area of state space from which the temporal gradient for the next time step would be drastically different. In fact, some runs of Algorithm 1 suffer from exactly this. The resulting dynamics jump back and forth between two distant regions of state space. The dynamics are spatially smooth, but temporally erratic. Therefore, we base our new proposed method on constraints that ensure temporal smoothness of the result.

The new method has two additional benefits. The EM algorithm proposed earlier suffers from another drawback. The objective being optimized is not convex and thus the optimization procedure is subject to getting trapped in local maxima. The new method will yield a convex penalty function to optimize. As a second benefit the new method is also able to tolerate a time-varying W.

Our new method is based on the following idea: the smoothness of a system trajectory can be quantified by the second order difference of adjacent points on

the trajectory. More specifically, this smoothness can be written as:

$$S = \sum_{t=2}^{T-1} ((\mathbf{x}^{t+1} - \mathbf{x}^t) - (\mathbf{x}^t - \mathbf{x}^{t-1}))^2, \qquad (12)$$

where T is the maximum time. Such a smoothness measure has been used as the regularization term in the Hodrick-Prescott filter (Hodrick & Prescott, 1997; Lesser, 1961), a common tool in macroeconomics for obtaining a smooth and nonlinear representation of a time series.

The quantity (12) cannot be computed on our data since there is no time information. Nevertheless, it can be succinctly expressed using the Laplacian L of the graph formed by connecting neighboring points in the manifold. More specifically, let $X := [\mathbf{x}_1\ \mathbf{x}_2 \cdots \mathbf{x}_N]$ be the data matrix where each column is an observation vector, and Z be the directed adjacency matrix where $Z_{ij} = 1$ if node j sends an edge to node i, and 0 otherwise (Our use of Z as the directed adjacency matrix is on purpose here; as we explain later, the permutation matrix Z can be interpreted as the directed adjacency matrix). Then the undirected, symmetric adjacency matrix $\overline{Z}$ of the graph can be computed as $\overline{Z} = Z + Z^\top$, and hence the graph Laplacian is $L = \mathsf{diag}(\overline{Z}\mathbf{e}) - \overline{Z}$, where $\mathbf{e}$ is a vector of ones and $\mathsf{diag}(\overline{Z}\mathbf{e})$ is the diagonal matrix with the vector $\overline{Z}\mathbf{e}$ as the main diagonal. The smoothness S of the manifold can then be written as:

$$\begin{aligned} S &= \|XL\|_{Fro}^2 \\ &= \mathrm{tr}((\mathsf{diag}(\overline{Z}\mathbf{e}) - \overline{Z})^\top X^\top X(\mathsf{diag}(\overline{Z}\mathbf{e}) - \overline{Z})), \end{aligned} \qquad (13)$$

which is quadratic and convex in $\overline{Z}$ and hence Z.

The ordering of the points from the dynamical system can be specified by a matching matrix Z with zero diagonal, where entry Z_{ij} indicates whether the predecessor of node i is j. This coincides with the definition of the directed adjacency matrix of a graph. Therefore we can minimize the smoothness quantity in (13) with respect to the matching matrix Z:

$$\begin{aligned} &Z^* = \mathrm{argmin}_Z \ \ S(Z) \\ &s.t. \ \ Z\mathbf{e} = \mathbf{e}, \quad Z^\top\mathbf{e} = \mathbf{e}, \quad Z_{ij} \in \{0,1\}, \quad Z_{ii} = 0, \\ &\qquad \overline{Z} = Z + Z^\top. \end{aligned} \qquad (14)$$

Note that a legitimate matching matrix is supposed to have exactly one zero row and one zero column corresponding to the starting and the end points in the sequence. However, searching over all possible choices of end points may be computationally infeasible, so we simply require all rows and all columns to sum to one, making Z a *permutation matrix*. The set of permutation matrices is still hard to search, and we further relax the integer constraints on the entries of Z to have the following optimization problem over doubly stochastic matrices:

$$\begin{aligned} &Z^* = \mathrm{argmin}_Z \ \ S(Z) \\ &s.t. \ \ Z\mathbf{e} = \mathbf{e}, \quad Z^\top\mathbf{e} = \mathbf{e}, \quad Z_{ij} \in \mathbb{R}^+, \quad Z_{ii} = 0, \\ &\qquad \overline{Z} = Z + Z^\top. \end{aligned} \qquad (15)$$

where $\mathbb{R}^+$ denotes the set of non-negative real numbers. Such a relaxation is justified by the *Birkhoff-von Neumann theorem* (van Lint & Wilson, 2001): the space of doubly stochastic matrices is a convex polytope in $\mathbb{R}^{N^2}$ whose extreme points are permutation matrices.

The optimization problem (15) is essentially convex quadratic programming under linear and bound constraints. However, the number of variables is *quadratic* in the number of data points, and as the data size increases, directly applying a general-purpose QP or nonlinear programming solver may become inefficient or even infeasible. We thus devise a simple and efficient *projected gradient method* that iteratively updates the rows and the columns of Z.

The key idea of a projected gradient method is to move the parameter vector along the negative gradient direction, and project the updated vector back into the feasible region Ω whenever it goes out. The cost of a projected gradient procedure is mainly determined by the projection operation, so we need to compute efficiently the projection step:

$$Z^{t+1} \leftarrow \Pi_\Omega(Z^t - \eta\nabla^t), \qquad (16)$$

$$\Omega = \{Z_{i\cdot}\mathbf{e} = 1, Z_{\cdot j}^\top\mathbf{e} = 1, Z_{ij} \in \mathbb{R}^+, Z_{ii} = 0\}, \qquad (17)$$

where $Z_{i\cdot}$ and $Z_{\cdot j}$ denote a row and a column of Z respectively, and $\Pi_\Omega(a) := \mathrm{argmin}_b\{\|a - b\| \mid b \in \Omega\}$ is the Euclidean projection of a vector a onto a region Ω. The formula for computing the gradient is given in Appendix A. We observe that the feasible region (17) is the intersection of two closed convex sets Ω_1 and Ω_2:

$$\begin{aligned} \Omega_1 &= \{Z_{i\cdot}\mathbf{e} = 1,\ Z_{ij} \in \mathbb{R}^+, Z_{ii} = 0, 1 \le i, j \le N\}, \\ \Omega_2 &= \{Z_{\cdot j}^\top\mathbf{e} = 1,\ Z_{ij} \in \mathbb{R}^+, Z_{ii} = 0, 1 \le i, j \le N\}, \end{aligned}$$

which correspond to the normalization constraints for rows and columns, respectively. Using Dykstra's cyclic projection algorithm (Boyle & Dykstra, 1986), we perform the projection operation (16) by alternately projecting onto Ω_1 and Ω_2. A very nice property of this procedure is that projecting onto Ω_1 or Ω_2 alone can be further decomposed as doing row-wise (or column-wise) projections, and a single-row or single-column projection can be computed very efficiently by the ℓ_1 projection technique proposed in (Duchi et al., 2008),

Algorithm 2 $w = \ell_1$-Projection(v), $v \in \mathbb{R}^N$

Sort v into $\mu : \mu_1 \geq \mu_2 \geq \ldots \geq \mu_N$.
Find $\rho = \max\{j \in [N] : \mu_j - \frac{1}{j}(\sum_{r=1}^{j} \mu_r - 1) > 0\}$
Define $\theta = \frac{1}{\rho}(\sum_{i=1}^{\rho} \mu_i - 1)$
Output w s.t. $w_i = \max\{v_i - \theta, 0\}$

Algorithm 3 Projected Gradient Method for (15)

Input: Data matrix $X = [\mathbf{x}_1 \ \cdots \ \mathbf{x}_N]$
Output: Z^*

Set $\alpha = 0.1$, $\epsilon = 10^{-6}$, $\sigma = 10^{-2}$
Initialize Z^1, set $t = 1$
repeat
  Compute the gradient $\nabla^t := \nabla S(Z^t)$, $\eta \leftarrow 1.0$
  repeat
    $Z = Z^t - \eta\nabla^t$, $D^{row} = D^{col} = [0]_{N\times N}$
    repeat
      $\tilde{Z} \leftarrow Z$
      $Z'_{i\cdot} \leftarrow \ell_1$-Projection$((Z - D^{row})_{i\cdot})$, $\forall i$
      $D^{row} \leftarrow Z' - (Z - D^{row})$
      $Z''_{\cdot j} \leftarrow \ell_1$-Projection$((Z' - D^{col})_{\cdot j})$, $\forall j$
      $D^{col} \leftarrow Z'' - (Z' - D^{col})$
      $Z \leftarrow Z''$
    until $\|Z - \tilde{Z}\|_{Fro} \leq \epsilon$
    $\eta \leftarrow \alpha\eta$
  until $S(Z) - S(Z^t) \leq \sigma\nabla^{(t)}(Z - Z^t)$
  $t \leftarrow t + 1$, $Z^t \leftarrow Z$
until $\|Z^t - Z^{t-1}\|_{Fro} \leq \epsilon$
$Z^* \leftarrow Z^t$

which we outline in Algorithm 2. The required operations are simply sorting and thresholding[1].

Algorithm 3 gives a summary of the projected gradient method for the optimization problem (15). As in all gradient-based methods, we conduct back-tracking line search for the step size η to ensure convergence.

5 Experiments

The proposed methods are evaluated on several synthetic data sets. We describe the data generation process in Section 5.1, give parameter settings of methods compared and evaluation criterion in Section 5.2, and report results and findings in Section 5.3.

5.1 Data

We generate data from three dynamical systems:
The 3D-1 linear system used in (Huang & Schneider, 2009). The transition matrix and the initial point are:

$$A = \begin{bmatrix} 1.1882 & 0.3732 & 0.1660 \\ -0.1971 & 0.8113 & -0.0107 \\ -0.1295 & -0.1886 & 0.9628 \end{bmatrix}, \ \mathbf{x}^0 = \begin{bmatrix} 10 \\ 10 \\ 10 \end{bmatrix}.$$

The maximum time $T_{\max}$ is set to 100.
3D-conv: a convergent three-dimensional nonlinear system (Girard & Pappas, 2005) governed by the following differential equations:

$$\begin{aligned} dx(t)/dt &= -(1 + 0.1y(t)^2)x(t), \\ dy(t)/dt &= -(1 - 0.1x(t)^2)y(t)/2 + 2z(t), \\ dz(t)/dt &= -(1 - 0.1x(t))2y(t) - z(t)/2, \end{aligned}$$

where $x(t), y(t)$, and $z(t)$ are the three states at time t. The initial point is set to $[5\ 1\ 5]^\top$ and 200 points are evenly sampled in the time interval $[0, 10]$.
The Lorenz attractor (Lorenz, 1963):

$$\begin{aligned} dx(t)/dt &= 10(y(t) - x(t)), \\ dy(t)/dt &= x(t)(28 - z(t)) - y(t), \\ dz(t)/dt &= x(t)y(t) - 8z(t)/3. \end{aligned}$$

The initial point is set to $[0\ 1\ 1.05]$ and 800 points are evenly sampled in the time interval $[0, 20]$.

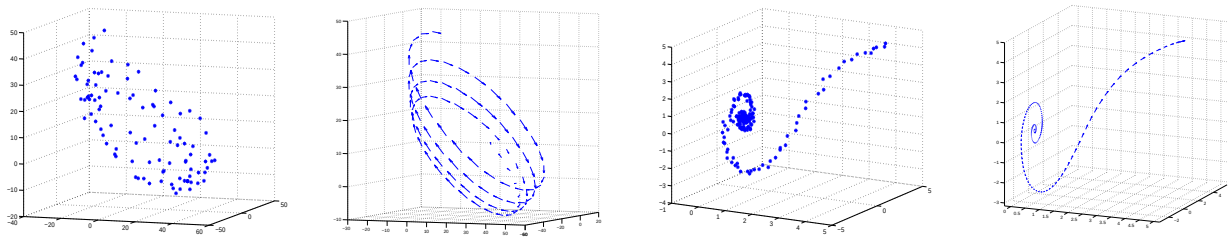
We consider two different settings:
Single trajectory with observational noise: Independent zero-mean Gaussian noise is added to the ideal trajectories, with three levels of noise standard deviation: $\sigma_{noise} = \{0, 0.01\delta, 0.05\delta\}$, where δ is the median of all the pairwise distances of points on an ideal trajectory. With positive noise levels, we generate 20 data sets for 3D-1, 3D-conv, and the last 400 points of the Lorenz trajectory[2] generated as above. Examples of the data for the three systems are in Figures 1(a), 1(c), and 2.
Multiple trajectories with process noise: Algorithm 3 of (Huang & Schneider, 2009) is applied to 3D-1 and 3D-conv to generate samples from multiple independent executions of a system. For 3D-conv we use its discrete-time counterpart, treating the derivatives as constant in a small duration $\Delta t = 0.1$. The process noise follows a zero-mean Gaussian, whose standard deviations are $\{0.01\delta, 0.05\delta\}$ for 3D-1 and $\{0.1\Delta t, 0.5\Delta t\}$ for 3D-conv. We generate 20 data sets of 400 points for each system and each noise level. We did not include the Lorenz attractor here because it is a chaotic system.

5.2 Evaluation criterion and parameter settings

The evaluation criterion we used is essentially the cosine score in (Huang & Schneider, 2009). Let

[1] For the ease of presentation, in Algorithm 2 we ignore the constraint $Z_{ii} = 0$, which can be easily enforced by setting $Z_{ii} = 0$ and updating only the other $N - 1$ entries.

[2] This partial trajectory preserves the butterfly shape.



(a) 3D-1: $\sigma_{noise} = 0.05\delta$ (b) 3D-1: predictions (c) 3D-conv: $\sigma_{noise} = 0.05\delta$ (d) 3D-conv: predictions

Figure 1: Examples of training data (single trajectory) and predicted dynamics on the ideal trajectory by TSM.

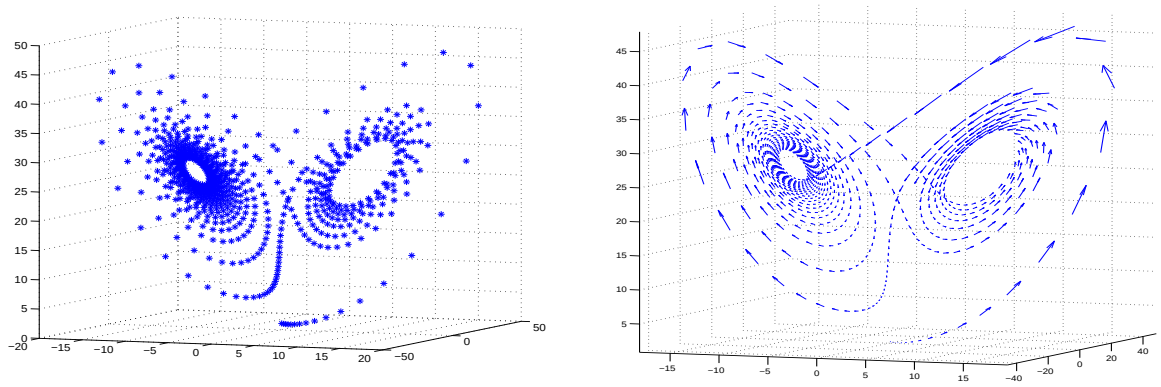


Figure 2: Left: Lorenz attractor ideal trajectory. Right: estimated dynamics by KUM.

$\{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^{T_{\max}}\}$ be the points sampled from an ideal system trajectory. After we learn a dynamic model from a training data set, we predict for each $\mathbf{x}^t, t = 1, \ldots, T_{\max} - 1$ the next state $\hat{\mathbf{x}}^{t+1}$, and compute the similarity score between the predicted difference vectors and the true ones:

$$\frac{1}{T_{\max} - 1} \left| \sum_{t=1}^{T_{\max}-1} \frac{(\mathbf{x}^{t+1} - \mathbf{x}^t)^\top (\hat{\mathbf{x}}^{t+1} - \mathbf{x}^t)}{\|\mathbf{x}^{t+1} - \mathbf{x}^t\| \|\hat{\mathbf{x}}^{t+1} - \mathbf{x}^t\|} \right|. \quad (18)$$

A higher score means a better prediction. By using such a normalized score of similarity, we focus on measuring the consistency of the predicted dynamics with the truth, but do not take into account the lengths of the difference vectors and the overall direction in time.

We compare the three proposed methods: the nonlinear Unordered model via kernel regression (KUM), the nonlinear Partial-ordered model via kernel regression (KPM), and the temporal smoothing method (TSM) against a baseline approach that exploits manifold learning techniques. The baseline approach maps the data points to the real line through some manifold learning method, sorts the points according to their one-dimensional projections, and then learns a dynamic model of the form (5) from the ordered data. In our experiments, we find Maximum Variance Unfolding (MVU) by (Weinberger et al., 2004) to be the best manifold learning choice. For KUM and KPM we also include results by their linear versions UM and PM.

As mentioned in Section 4, we relax the constraints on having end points and hence the matching matrix Z obtained by the temporal smoothing method is usually a mixture of dynamics in both directions. In Figure 3 we show the heat map of a Z matrix for one of the single-trajectory data set generated from 3D-conv, where the rows and the columns are sorted according to the true temporal order. The energy is concentrated around the two leading off-diagonals, showing that the temporal smoothing method nicely recovers the sequential structure in time, but does not choose one direction over the other[3]. To resolve this ambiguity, we apply the following heuristics to obtain an adjusted matching matrix. For single-trajectory data, we take the lower-triangular part of $Z + Z^\top$ and normalize it to be doubly-stochastic, thereby specifying the correct starting and the end points in time. For multiple-trajectory data, we treat Z as a weighted adjacency matrix, find the maximum spanning tree rooted at each data point, and choose the tree that exhibits the

[3]In fact, for some dynamic systems the true direction in time is not identifiable (Peters et al., 2009).

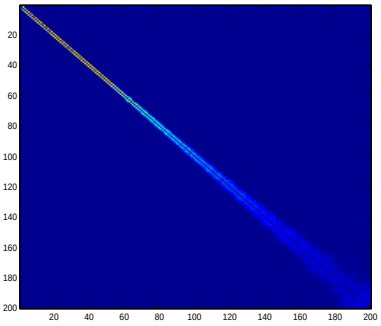


Figure 3: Heat map of the matching matrix Z by TSM

"smoothest" dynamics, i.e., the maximum total cosine score between the difference vectors corresponding to adjacent tree edges. After this post-processing, we fit a re-weighted regression as in (10) using the adjusted matching matrix as the weights.

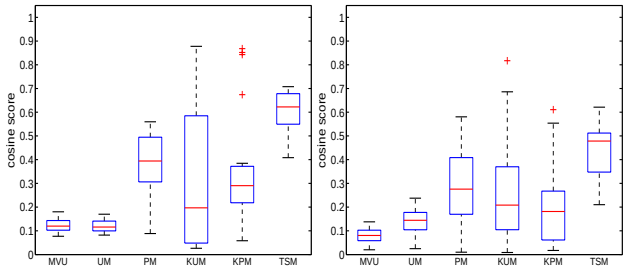
For 3D-1, we use a linear model for MVU and TSM in their final regression. For all the nonlinear systems we use kernel regression with the Gaussian kernel $\exp(-(\|\mathbf{x}-\mathbf{y}\|^2)/(2h))$. When applying KUM and KPM, we set the kernel bandwidth h to $10\tilde{\delta}$ and $50\tilde{\delta}$ respectively for 3D-1 and the two nonlinear systems, where $\tilde{\delta}$ is the median of all pairwise distances in a training data set. For TSM and MVU we set $h = \tilde{\delta}$. Regarding the regularization parameter λ in (6), on noiseless data we set it to 10^{-3}. On 3D-1 single(multiple)-trajectory data we set it to $10^{-7}(10^{-6})$ and $10^{-6}(10^{-5})$ for small and large noise levels, and on all the other data we set it to 10^{-4} and 10^{-3} for the two noise levels. When applying KUM and KPM, we use a low-rank approximation to the kernel matrix as described in Section 3, with the reduced rank $m = 5$. For each data set we run KUM and KPM with 50 random initializations of W and σ^2 to avoid local minima. Each entry of W is drawn independently from a zero-mean Gaussian with the standard deviation set to 100, and σ^2 is drawn uniformly random between 0 and 100 times of the median of pairwise distances.

5.3 Results and findings

Our experiment results are in Table 1 and Figures 4 to 6. Table 1 reports cosine scores on noiseless trajectories and bold-faces the best method for each dynamical system. It is interesting that KUM performs better than its linear version UM on the linear system 3D-1, and that UM performs very well on the nonlinear system 3D-conv; the latter suggests 3D-conv may be well approximated by a linear system in terms of one-step predictions. For Lorenz attractor, however,

	MVU	UM	PM	KUM	KPM	TSM
3D-1	0.0152	0.9120	0.9898	**0.9972**	0.3239	0.8757
3D-conv	**0.9954**	0.9903	0.5570	0.9909	0.9225	0.9545
Lorenz	0.1383	0.5644	0.2155	**0.9884**	0.334	0.324

Table 1: Cosine scores on noiseless data



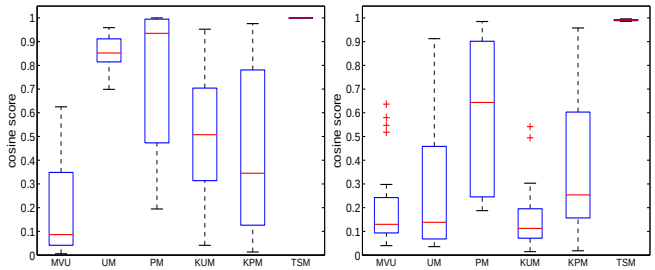
(a) $\sigma_{noise} = 0.01\delta$ (b) $\sigma_{noise} = 0.05\delta$

Figure 4: Box plots of cosine scores for Lorenz attractor (partial trajectory), outliers marked as red crosses.

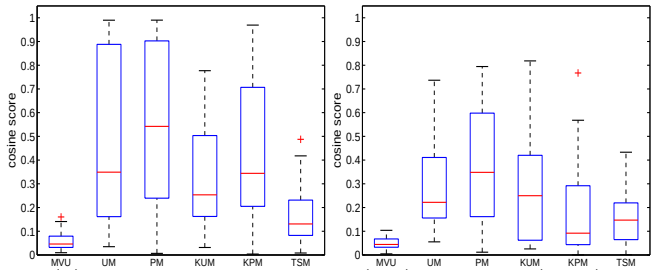
only KUM performs well and other methods are significantly worse. Figure 2 shows the estimated dynamics by KUM, which is very close to the true dynamics.

Figures 4, 5(a), and 6(a) report results on single-trajectory noisy data. TSM is the best for all three systems while MVU is the worst except for 3D-conv with small noise. Figure 1 shows examples of noisy training data and predicted dynamics by TSM. On Lorenz attractor, a highly nonlinear system, KUM and KPM are better than UM and PM. It is interesting that TSM performs better here than in the noiseless case; the reason may be that the partial trajectory used here does not contain the highly-dense spirals in the core of the left wing shown in Figure 2, which cause difficulties for TSM as the smoothness measure (13) may be sensitive to different spacings of the data. For 3D-1, it is as expected that UM and PM are better than KUM and KPM, but for 3D-conv UM sometimes outperforms them by a small margin. This is aligned with our finding from noiseless data and the known fact that linear models may still be useful for nonlinear systems. Figures 5(b) and 6(b) show results on multiple-trajectories data, which are roughly consistent with single-trajectory results except that TSM performs a lot worse. This is not surprising since TSM's assumption of a single ordering of data points is invalid here.

As shown in the box plots, KUM and KPM may suffer seriously from local optima. While TSM uses convex optimization, it is sensitive to irregular data distributions and requires an effective post-processing step that chooses an overall direction. These observations suggest that some proper combination of the two approaches would lead to a more effective method.

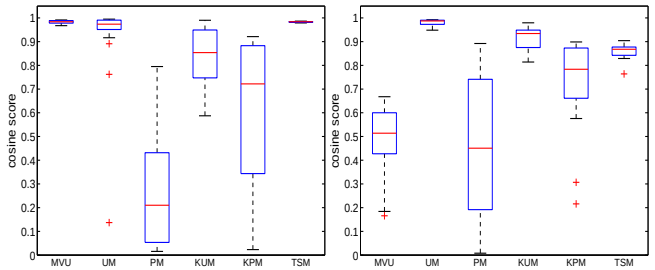


(a) Single: $\sigma_{noise} = 0.01\delta$ (left) and 0.05δ (right)

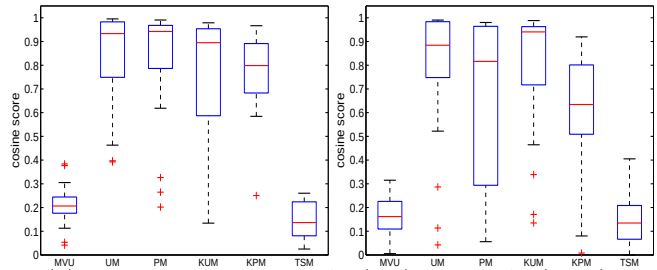


(b) Multiple: $\sigma_{noise} = 0.01\delta$ (left) and 0.05δ (right)

Figure 5: Box plots of cosine scores for 3D-1.



(a) Single: $\sigma_{noise} = 0.01\delta$ (left) and 0.05δ (right)



(b) Multiple: $\sigma_{noise} = 0.1\Delta t$ (left) and $0.5\Delta t$ (right)

Figure 6: Box plots of cosine scores for 3D-conv.

6 Conclusions

We study the problem of learning nonlinear dynamic models from non-sequenced data. The proposed methods include generalizations of previous work for learning linear dynamic systems, and a new formulation that recovers the order of data points by temporal smoothing. We evaluate the proposed methods on several synthetic dynamic systems, including a famous chaotic system, the Lorenz attractor, and obtain reasonably good results. We believe such a development represents a further step towards the goal of making new discoveries from real world data.

A The gradient of (13)

The gradient of the smoothness measure (13) at some Z can be computed by the following steps:

$\overline{Z} \leftarrow Z + Z^\top$.

$Q \leftarrow X^\top X(\mathsf{diag}(\overline{Z}\mathbf{e}) - \overline{Z})$.

Compute $\tilde{Q} \in \mathbb{R}^{N\times N}$ such that $\tilde{Q}_{ij} = Q_{jj} - Q_{ij}$.

Return $2(\tilde{Q} + \tilde{Q}^\top)$ as the gradient.

Acknowledgments

This work was funded in part by the Department of Energy under grant DESC0002607.

References

Boyle, J. P., & Dykstra, R. L. (1986). A method for finding projections onto the intersection of convex sets in Hilbert spaces. *Lecture Notes in Statistics*, *37*, 28–47.

Duchi, J., Shalev-Shwartz, S., Singer, Y., & Chandra, T. (2008). Efficient projections onto the ℓ_1-ball for learning in high dimensions. *Proceedings of the 25th International Conference on Machine Learning* (pp. 272–279).

Girard, A., & Pappas, G. J. (2005). Approximate bisimulations for nonlinear dynamical systems. *44th IEEE Conference on Decision and Control and European Control Conference* (pp. 684–689).

Hodrick, R., & Prescott, E. C. (1997). Postwar U.S. business cycles: An empircal investigation. *Journal of Money, Credit, and Banking*, *29*, 1–16.

Huang, T.-K., & Schneider, J. (2009). Learning linear dynamical systems without sequence information. *Proceedings of the 26th International Conference on Machine Learning* (pp. 425–432).

Lesser, C. E. V. (1961). A simple method of trend construction. *Journal of the Royal Statistical Society, Series B*, *23*, 91–107.

Lorenz, E. N. (1963). Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, *20*, 130–141.

Peters, J., Janzing, D., Gretton, A., & Schölkopf, B. (2009). Detecting the direction of causal time series. *Proceedings of the 26th International Conference on Machine Learning* (pp. 801–808).

van Lint, J., & Wilson, R. M. (2001). *A course in combinatorics*. Cambridge University Press.

Weinberger, K. Q., Sha, F., & Saul, L. K. (2004). Learning a kernel matrix for nonlinear dimensionality reduction. *Proceedings of the 21st International Conference on Machine Learning* (pp. 839–846).