Learning Linear Dynamical Systems without Sequence Information

Tzu-Kuo Huang TZUKUOH@CS.CMU.EDU
Machine Learning Department, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA

Jeff Schneider SCHNEIDE@CS.CMU.EDU
Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA

Abstract

Virtually all methods of learning dynamic systems from data start from the same basic assumption: that the learning algorithm will be provided with a sequence, or trajectory, of data generated from the dynamic system. In this paper 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.

While making this assumption may seem absurd at first glance, we observe that many scientific modeling tasks have exactly this property. In this paper we restrict our attention to learning linear, discrete time models. We propose several algorithms for learning these models based on optimizing approximate likelihood functions and test the methods on several synthetic data sets.

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, or trajectory, 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 usually come from many separate executions of the dynamic system.

Many scientific modeling tasks have exactly this property. Consider the task of learning dynamic models of galaxy or star evolution. The dynamics of 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. It is feasible to follow patients over the full course of their disease and many studies do exactly that. However, a scientist considering new hypotheses about the effects of previously unmeasured proteins or genes would prefer to collect data from their current pool of patients and assemble a dynamic model immediately rather than wait a decade for full trajectories of the disease to be collected.

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 even though a rough indication of the relative timing between samples may be known.

In all of these applications, scientists would like to construct a dynamic model using only unsequenced, individual data points collected from the system of interest. In this paper, we begin investigation of this topic by considering fully observable, linear, continuous-state, discrete-time systems. There is considerable opportunity for future work by relaxing or modifying any combination of these assumptions.

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009.

2. Learning Linear Dynamical Systems

We consider discrete-time linear dynamical systems, for which the system equation is defined as the following:

$$\mathbf{x}^{t+1} = A\mathbf{x}^t + \epsilon, \tag{1}$$

where $\mathbf{x}^t \in R^n$ is the state vector at time t, $A \in R^{n\times n}$ is the state transition matrix, and ϵ is the noise vector. We assume hereafter that $\epsilon \sim N(\mathbf{0}, \sigma^2 I)$, where I is the identity matrix. The system also has a start state, which we denote as $\mathbf{x}^0$. Thus, the linear dynamical systems considered in this paper are fully characterized by $\Theta = \{A, \sigma^2, \mathbf{x}^0\}$.

2.1. Learning from sequenced data

Before discussing our approach for non-sequenced data, we observe that learning from sequenced data can be done in a straightforward way. Suppose we have collected N data points, $\mathbf{x}_1, \ldots, \mathbf{x}_N$, from a single execution of the system where the ith data point occurred at time i. Then we can perform a standard regression to obtain estimates of the parameters. We form an input matrix, X, with $N-1$ rows containing the observations $\mathbf{x}_1, \ldots, \mathbf{x}_{N-1}$ and a corresponding output matrix, Y, with $N-1$ rows containing the observations $\mathbf{x}_2, \ldots, \mathbf{x}_N$. Then A is estimated as $(X^{'}X)^{-1}(X^{'}Y)$, where X' is the transpose of X, and σ^2 is estimated by $\|Y - XA\|_F^2/(N-1)/n$ as usual, where $\|\cdot\|_F$ is the matrix Frobenius norm. If more than one trajectory is observed, X and Y are assembled accordingly such that each row contains pairs of state observations at time t and $t+1$. In fact, this approach does not require long trajectories. It only needs enough *pairs of consecutive observations*. However, as mentioned in Section 1, with our target systems it is difficult to collect even two consecutive observations at desirable time intervals.

2.2. Learning from non-sequenced data

The problem without observed state sequences is much more difficult. We assume that N executions of the dynamic system (1) have taken place, and from each execution we have observed a single data point drawn uniformly at random from the sequence of states generated in that execution. The result is N data points, $\mathbf{x}_i$, each from a different trajectory and having occurred at an unknown point in time.

We focus on estimating A and σ^2, and treat the start state, $\mathbf{x}^0$, as a nuisance parameter. Let χ denote the state space, and $f_\theta(\mathbf{x})$ denote the state space density induced by the underlying linear system parameterized by $\theta \in \Theta$. To write down the likelihood, we first consider a single observation $\mathbf{x}_i$. Given any point $\mathbf{x}$ in the state space, the likelihood of $\mathbf{x}_i$ coming from it can be derived from (1) by having $\mathbf{x}^{t+1} = \mathbf{x}_i$ and $\mathbf{x}^t = \mathbf{x}$. But the true predecessor of $\mathbf{x}_i$ is not observed, so we take the expectation over the state space density at the previous time point. Assuming that the N trajectories are generated independently, we then have the following likelihood $l(\theta)$:

$$\prod_{\substack{i=1,\\ t(\mathbf{x}_i)>0}}^{N} \left(\int_{\mathbf{x}\in\chi} \frac{\exp(-\frac{\|\mathbf{x}_i - A\mathbf{x}\|^2}{2\sigma^2})}{(2\pi\sigma^2)^{\frac{n}{2}}} g_\theta(\mathbf{x}|t(\mathbf{x}_i)-1)d\mathbf{x} \right), \tag{2}$$

where $\|\cdot\|$ is the vector two-norm, $t(\cdot)$ gives the time stamp of a state vector, and

$$g_\theta(\mathbf{x}|j) \equiv \frac{f_\theta(\mathbf{x})\mathbf{I}(t(\mathbf{x})=j)}{\int_{\mathbf{y}\in\chi} f_\theta(\mathbf{y})\mathbf{I}(t(\mathbf{y})=j)d\mathbf{y}} \tag{3}$$

is the state space density at time j, as shown by the indicator function $\mathbf{I}(\cdot)$. Since $g_\theta(\mathbf{x}|j)$ and the true time stamp $t(\mathbf{x}_i)$ are unknown, estimating A and σ^2 might require maximizing (2) jointly over the parameter space and the missing information, which is computationally formidable. We thus employ several approximations to (2) that lead to tractable estimation procedures.

3. Approximate Likelihood Approaches

This section presents two approximate likelihood approaches to estimate A and σ^2.

3.1. An Unordered Model

Maximizing (2) directly is difficult because we know neither the time stamps of the data points, $t(\mathbf{x}_i)$, nor the state space density, $g_\theta(\mathbf{x}|j)$. We first remove the problem of unknown time stamps by replacing $g_\theta(\mathbf{x}|j)$ with $f_\theta(\mathbf{x})$. This is equivalent to a generative model where a random sample is chosen from $f_\theta(\mathbf{x})$ and an observation is created by applying (1) to it. No explicit consideration of $t(\mathbf{x}_i)$ is necessary. This change means we need an estimate of $f_\theta(\mathbf{x})$. We simply use the empirical density given by the samples we have. This leads to the following approximate likelihood:

$$\hat{l}_1(X|\theta) = \prod_{i=1}^{N} \left(\sum_{j\neq i} \frac{\exp(-\frac{\|\mathbf{x}_i - A\mathbf{x}_j\|^2}{2\sigma^2})}{(N-1)(2\pi\sigma^2)^{\frac{n}{2}}} \right). \tag{4}$$

where X is the matrix of N state observations. We exclude the case that $\mathbf{x}_i$ generates itself to avoid the degenerate estimate $A = I$. Note that the corresponding Y matrix (as in section 2.1) is not only unknown,

but does not even exist because the data does not contain successive observations from the same trajectory.

Eq. (4) is a product of summations of Gaussian density functions. This structure is also shared by the likelihood of Gaussian Mixture Models (GMM), for which Expectation Maximization (EM) algorithms are the common choice for estimation. Although (4) is not a GMM, its similar structure allows us to derive an EM procedure for estimation with analytical update rules. We first introduce a latent variable matrix $Z \in \{0,1\}^{N\times N}$ that indicates which observation is generated by which:

$$Z_{ij} = \begin{cases} 1 & \text{if } \mathbf{x}_i \text{ is generated from } \mathbf{x}_j \\ 0 & \text{otherwise} \end{cases}, \quad j \neq i, \quad (5)$$

$$Z_{ii} = 0, \ \sum_{j=1}^{N} Z_{ij} = 1. \quad (6)$$

Note that here we have an additional approximation that is not shared by EM for GMMs. In GMMs, we assume each point really does come from a particular cluster, but we do not know which one it is. In our problem, the true predecessor point does not exist in the data at all and we make an approximation by assuming that it does. An immediate consequence is that σ^2 in (4) now accounts for two types of errors: the noise ϵ in the system (1) and the error introduced by approximating the true state space density with the empirical density. With Z, we can write the complete log likelihood as

$$\begin{aligned} &\log \hat{l}_1(X, Z|\theta) \\ =&\log \prod_{i=1}^{N}\prod_{j=1}^{N} \left(\frac{\exp(-\frac{\|\mathbf{x}_i - A\mathbf{x}_j\|^2}{2\sigma^2})}{(N-1)(2\pi\sigma^2)^{\frac{n}{2}}} \right)^{Z_{ij}} \\ \propto& -\sum_{i=1}^{N}\sum_{j=1}^{N} Z_{ij} \left(\frac{\|\mathbf{x}_i - A\mathbf{x}_j\|^2}{2\sigma^2} + \frac{n}{2}\log(2\pi\sigma^2) \right). \quad (7) \end{aligned}$$

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

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

In the M-step, we replace Z_{ij} by $\tilde{Z}_{ij}$ and maximize (7) on A and σ^2. The solution has a simple form:

$$A = \left(\sum_{i=1}^{N}\sum_{j=1}^{N} \tilde{Z}_{ij}\mathbf{x}_i\mathbf{x}_j' \right) \left(\sum_{i=1}^{N}\sum_{j=1}^{N} \tilde{Z}_{ij}\mathbf{x}_j\mathbf{x}_j' \right)^{-1}, \quad (9)$$

$$\sigma^2 = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} \tilde{Z}_{ij}\|\mathbf{x}_i - A\mathbf{x}_j\|^2}{n\sum_{i=1}^{N}\sum_{j=1}^{N}\tilde{Z}_{ij}}, \quad (10)$$

Algorithm 1 Expectation Maximization for (4)

Input: Data points $\mathbf{x}_1, \ldots, \mathbf{x}_N$
Initialize A_0 and σ_0^2, set $T = 0$
repeat
  Update $\tilde{Z}_{T+1}$ by (8) with A_T and σ_T^2
  Update A_{T+1} by (9) with $\tilde{Z}_{T+1}$
  Update σ_{T+1}^2 by (10) with A_{T+1} and $\tilde{Z}_{T+1}$
  $T \leftarrow T + 1$
until The likelihood (4) does not increase

where $\mathbf{x}_j'$ is the transpose of $\mathbf{x}_j$. (10) is the maximum likelihood estimator (MLE) for the variance of Gaussians[1]. Combining (8), (9), and (10), we present an EM procedure, Algorithm 1, to maximize (4). In the empirical results section we will use a random initialization at the first step.

Although (4) leads to simple and computationally efficient estimation, one may worry that ignoring the need for directional consistency in the Z_{ij}s makes the approximation too loose and results in poor estimates, especially when the sample size is limited. We thus propose a variant of (4) that incorporates additional constraints in the next section.

3.2. A Partial-order Model

The Unordered Model approximates the state space density without consideration of the ordering/directionality implicitly embedded in Z, and hopes that as the sample size grows, the estimates of A and σ^2 converge to the truth and automatically recover the ordering. In our second approach, we take into account the ordering relation explicitly. Our second approximate likelihood is as follows:

$$\hat{l}_2(\theta) = \prod_{\substack{i=1,\\ i\notin S}}^{N}\sum_{j=1}^{N} \left(\frac{\exp(-\frac{\|\mathbf{x}_i - A\mathbf{x}_j\|^2}{2\sigma^2})}{(2\pi\sigma^2)^{\frac{n}{2}}}\omega_{ij} \right), \quad (11)$$

in which $S \equiv \{i : t(\mathbf{x}_i) \leq t(\mathbf{x}_j)\ \forall j\}$ and

$$\omega_{ij} \equiv \begin{cases} 0 & i \in S, \\ \frac{\mathbf{I}(t(\mathbf{x}_j)=t(\mathbf{x}_i)-1)}{\sum_{j'=1}^{N}\mathbf{I}(t(\mathbf{x}_j')=t(\mathbf{x}_i)-1)} & i \notin S. \end{cases} \quad (12)$$

is the empirical frequency that $\mathbf{x}_i$ is generated from $\mathbf{x}_j$. The set S denotes the observations that are the earliest in time (hence cannot be generated from other observations), which can be viewed as rough estimates of the start state. If the system does cycle through the same states (such as rotation on a plane), the true

[1]For the case of a full covariance matrix, it is easy to check that (9) is still valid and (10) should be replaced by the MLE of a Gaussian covariance.

start state may not be identifiable but A and σ^2 still may be. In that scenario, S is chosen arbitrarily and the relative time offsets between points may still be correct, thus leading to good estimates of A and σ^2.

As mentioned before, the true time stamps $t(\mathbf{x}_i)$ of observations are missing, and one may think of treating them as latent variables. However, this turns the estimation into optimization over permutations, which is computationally infeasible. We instead consider the ω_{ij} as unknown parameters to be estimated, which we interpret as decomposing the global sequence information into parameters that indicate the relative order in each pair of observations. As with the Unordered Model, we again make an approximation by assuming that the predecessor of each point exists in the data.

Similarly to (4), (11) is also a product of summations of Gaussians. In the terminology of GMMs, (11) considers the "mixture probabilities" ω_{ij} as unknown parameters, whereas (4) adopts fixed, uniform mixture probabilities $1/(N-1)$. One may thus suggest adding an M-step in Algorithm 1 to estimate ω_{ij}. However, due to the set S of estimated start states in (11), the M-step for ω_{ij} may be difficult to derive analytically. Moreover, typical EM procedures for mixture models do not consider the directionality constraints embedded in ω_{ij}. For instance, if $\mathbf{x}_i$ is likely to be generated from $\mathbf{x}_j$ ($\omega_{ij} > 0$) and $\mathbf{x}_j$ is likely to be generated from $\mathbf{x}_k$ ($\omega_{jk} > 0$), then $\mathbf{x}_k$ is unlikely to be generated from $\mathbf{x}_i$ ($\omega_{ki} \approx 0$). Formally speaking, we want ω, the matrix with ω_{ij} as the (i,j)th entry, to have the following two properties:

1. Each row of ω sums to one or zero.
2. As a weighted adjacency matrix, ω represents a *directed acyclic* graph.

The first constraint simply restates (12), while the second enforces a partial order among the observations. Note that for both constraints to be satisfied, one or more rows in ω must sum to zero, and the corresponding data points form the set S.

With these two constraints, it becomes very difficult to develop an EM procedure with compact update rules. In particular, maximizing (11) on ω is hard, since the aforementioned two constraints on ω do not form a convex set. Moreover, Nicholson (1975) proved that a weighted adjacency matrix M contains no cycle if and only if permanent$(M+I) = 1$, and Valiant (1979) showed that computing the matrix permanent is #P-complete. We therefore employ further approximations here. Instead of using the common technique of relaxation to deal with complicated constraints, we *tighten* the constraints to have the feasible region more computationally amenable. The tighter version is the following:

1. ω can only take values in $\{0,1\}$, and each row contains at most one positive entry.
2. As an adjacency matrix, ω forms a *directed tree*.

The new constraints turn the problem into a combinatorial one, which, at first glance, seems even more difficult. As we will show later, the fact that the binary version is more computable depends entirely on our restricting the adjacency matrix to be a directed tree, i.e, a directed acyclic graph where each node, other than the root, has exactly one parent. Under the new constraints, the set S has only one data point, which is the root of the directed tree. Combining the aforementioned directionality constraints with (11), we propose the following constrained maximization problem for estimation:

$$\max_{\substack{A,\sigma^2,\omega,\\ r\in\{1,\dots,N\}}} \quad \sum_{\substack{i=1,\\ i\neq r}}^{N} \log \sum_{j=1}^{N} \left(\frac{\exp(-\frac{\|\mathbf{x}_i - A\mathbf{x}_j\|^2}{2\sigma^2})}{(2\pi\sigma^2)^{\frac{n}{2}}} \omega_{ij} \right) \tag{13}$$

$$\text{s.t.} \quad \omega_{ij} = \{0,1\}, \tag{14}$$

$$\sum_{j=1}^{N} \omega_{ij} = 1,\ i \neq r,\ \sum_{j=1}^{N} \omega_{rj} = 0, \tag{15}$$

$$\omega \text{ forms a tree with root } \mathbf{x}_r. \tag{16}$$

where r is the index of the point in S. With the constraints (14) and (15), the log-likelihood in (13) can be simplified as the following:

$$\begin{aligned}
&\sum_{\substack{i=1,\\ i\neq r}}^{N} \log \sum_{j=1}^{N} \left(\frac{\exp(-\frac{\|\mathbf{x}_i - A\mathbf{x}_j\|^2}{2\sigma^2})}{(2\pi\sigma^2)^{\frac{n}{2}}} \omega_{ij} \right) \\
=&\sum_{i=1}^{N} \log \prod_{j=1}^{N} \left(\frac{\exp(-\frac{\|\mathbf{x}_i - A\mathbf{x}_j\|^2}{2\sigma^2})}{(2\pi\sigma^2)^{\frac{n}{2}}} \right)^{\omega_{ij}} \\
=&-\sum_{i=1}^{N}\sum_{j=1}^{N} \omega_{ij} \left(\frac{\|\mathbf{x}_i - A\mathbf{x}_j\|^2}{2\sigma^2} + \frac{n}{2}\log(2\pi\sigma^2) \right).
\end{aligned} \tag{17}$$

Interestingly, this objective function is in the same form as the complete log likelihood (7) of the Unordered Model. One may also notice that ω in (13) can be viewed as the latent variable Z in Section 3.1 plus some directionality constraints. However, there is a subtle difference between the roles they play for each model. For the Unordered Model, Z serves only as a means to derive the two steps in the EM algorithm, and it is not in the objective function (4) being maximized nor is it in Algorithm 1, where it has been replaced by the posterior mean $\tilde{Z}$. In contrast, ω explicitly appears in the optimization problem (13) as

Algorithm 2 Alternate Maximization for (13)

Input: Data points $\mathbf{x}_1, \ldots, \mathbf{x}_N$.
Initialize A_0 and σ_0^2, set $T = 0$
repeat
  Construct W_T by (20) with A_T and σ_T^2
  $\omega_{T+1} \leftarrow$ OptimumBranch(W_T)
  Update A_{T+1} by (18) with ω_{T+1}
  Update σ_{T+1}^2 by (19) with A_{T+1} and ω_{T+1}
  $T \leftarrow T + 1$
until The likelihood (17) does not increase

an unknown parameter to be estimated. To see how introducing latent variables helps to maximize the likelihood, we refer the readers to Section 9.4 of (Bishop, 2006).

Next we discuss how to solve (13). Since (17) has the same form as (7), the optimal A and σ^2 under a fixed ω have the same expression as (9) and (10):

$$A = \left(\sum_{i=1}^{N}\sum_{j=1}^{N}\omega_{ij}\mathbf{x}_i\mathbf{x}_j'\right)\left(\sum_{i=1}^{N}\sum_{j=1}^{N}\omega_{ij}\mathbf{x}_j\mathbf{x}_j'\right)^{-1}, \quad (18)$$

$$\sigma^2 = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N}\omega_{ij}\|\mathbf{x}_i - \hat{A}\mathbf{x}_j\|^2}{n\sum_{i=1}^{N}\sum_{j=1}^{N}\omega_{ij}}. \quad (19)$$

When A and σ^2 are fixed, maximizing (17) on ω subject to (14), (15), and (16) is equivalent to finding a *maximum spanning tree on a directed weighted graph*, in which each data point $\mathbf{x}_i$ is a node, each pair of nodes is connected in both directions, and the weight on the edge (i, j) is

$$W(i,j) \equiv -\left(\frac{\|\mathbf{x}_i - A\mathbf{x}_j\|^2}{2\sigma^2} + \frac{n}{2}\log(2\pi\sigma^2)\right). \quad (20)$$

The problem of finding maximum spanning trees on directed graphs is a special case of the *optimum branchings* problem, which seeks a maximum or minimum forest of rooted trees (branching) on a directed graph. Chu and Liu (1965), Edmonds (1967), and Bock (1971) independently developed efficient algorithms for the optimum branchings problem. The ones by the former two are virtually identical, and are usually referred to as the Chu-Liu-Edmonds algorithm, for which Tarjan (1977) gave an efficient implementation that runs in $O(N^2)$ time, where N is the number of nodes, for densely connected graphs. Camerini et al. (1979) pointed out an error of Tarjan (1977) and provided a remedy retaining the same time complexity.

With these results, we present an alternate maximization procedure, Algorithm 2 for solving (13), where OptimumBranch$(\cdot)$ taking an edge-weight matrix as the input argument uses the implementation of Tarjan (1977) and Camerini et al. (1979). Since Algorithm 2 always increases the likelihood (17), it converges to at least a local maximum.

Recently in the Natural Language Processing community, researchers (Smith & Smith, 2007; Globerson et al., 2007) have developed sum-product inference algorithms for directed spanning trees. These methods can be used to compute the posterior mean of Z in Section 3.1 over all directed spanning trees, and thus lead to an EM algorithm for tree structures. However, the resulting E-step requires inverting a matrix of dimension $N \times N$, which becomes difficult numerically and computationally as N increases.

4. Experiments

The proposed methods are evaluated on several synthetic data sets. We describe our experiment setting in Section 4.1, discuss two evaluation criteria in Section 4.2, and report results and findings in Section 4.3.

4.1. Experiment Setting

Given a system matrix $A \in R^{n\times n}$, a Gaussian noise variance σ^2, a starting state vector $\mathbf{x}^0$, the maximum time $T_{\max}$, and the desired number N of sample points, we use Algorithm 3 to sample from multiple trajectories. In all of the experiments, we set $T_{\max} = 100$. We consider the following three linear dynamical systems.

A two-dimensional divergent system:

$$A_{\text{2D}} = \begin{bmatrix} 1.01 & 0 \\ 0 & 1.05 \end{bmatrix}, \ \mathbf{x}^0 = \begin{bmatrix} 50 \\ 50 \end{bmatrix}.$$

A 200-point sample is shown in Figure 1.

Two three-dimensional divergent systems:

$$A_{\text{3D-1}} = \begin{bmatrix} 1.1882 & 0.3732 & 0.1660 \\ -0.1971 & 0.8113 & -0.0107 \\ -0.1295 & -0.1886 & 0.9628 \end{bmatrix}, \ \mathbf{x}^0 = \begin{bmatrix} 10 \\ 10 \\ 10 \end{bmatrix},$$

$$A_{\text{3D-2}} = \begin{bmatrix} 1.0686 & -0.0893 & 0.3098 \\ 0.4385 & 1.0091 & -0.2884 \\ -0.0730 & 0.0405 & 0.9625 \end{bmatrix}, \ \mathbf{x}^0 = \begin{bmatrix} 10 \\ 10 \\ 10 \end{bmatrix}.$$

We refer to them as 3D-1 and 3D-2, respectively. Although not differing much in entry values, these two systems have quite different dynamics, as indicated by the gradients in Figures 3 and 4. The complex eigenvalues of 3D-1 have a larger absolute value than the real one, but it is the opposite for 3D-2.

While the results presented here are all on divergent systems, we also experimented with convergent systems and got similar results. We tested the proposed

Algorithm 3 Sampling from multiple trajectories

Input: A, σ^2, $\mathbf{x}^0$, $T_{\max}$, and N
for $i = 1$ **to** N **do**
  Pick a random time stamp T_i from $\{1, \ldots, T_{\max}\}$.
  for $t = 1$ **to** T_i **do**
    $\mathbf{x}^t \leftarrow A\mathbf{x}^{t-1} + \epsilon$, $\epsilon \sim N(0, \sigma^2 I)$.
  end for
  Set $\mathbf{x}_i = \mathbf{x}^{T_i}$.
end for
Output: A sample $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$.

methods under a variety of settings. For 2D, we generated 40 data sets, each containing 200 observations, with $\sigma = 0.2$. For 3D-1 and 3D-2, we varied both the sample sizes and σ^2. For the small-sized experiments, we generated 40 data sets, each containing 200 points, with $\sigma = 0.2, 0.4, 0.6$ and 0.8. For the large-sized experiments, we generated 20 data sets, each containing 2,000 points, with σ in the same range. We found that larger values of σ overwhelmed the dynamics to such an extent that no algorithm performed well. We report the performances of the proposed methods in Section 4.3

For both the Unordered Model and the Partial-order Model, the estimates depend on the initializations of the estimation procedures. Thus, for every data set we ran Algorithms 1 and 2 each with M random initializations, and chose the one that gave the largest likelihood ((4) and (13), respectively) as the final estimate. The entries of these random matrices were sampled independently and uniformly from $[0, 1]$. We set M to be 20 and 10 for the small-sized and the large-sized experiments, respectively. Hereafter we refer to the two models with random initializations as UM (the Unordered model) and PM (the Partial-order Model).

In addition to random initializations, we also explored the use of manifold learning techniques for finding initial estimates. The rationale is that, for sample points generated by a linear system, there should be a one-dimensional projection that indicates the correct order in time. More specifically, we applied a manifold learning technique to our data, and mapped the sample points to the most significant coordinate it found. Then, we sorted the data points according to their one-dimensional projections and fitted a linear dynamical system by the technique in Section 2.1. The fitted system itself is already an estimate, and can be used to initialize Algorithms 1 and 2. In our experiments, we found Maximum Variance Unfolding (MVU) by Weinberger et al. (2004) to be the best manifold learning choice. Finally, to indicate the baseline performance, we report results from randomly generated matrices.

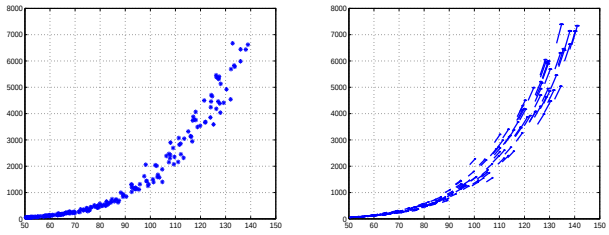


Figure 1. Left: Sample points of 2D with $\sigma = 0.2$. Right: Gradients estimated by UM.

Table 1. Results on 2D with standard deviations, $\sigma = 0.2$

	Rand	PM+MVU	PM	UM+MVU	UM
ME	1.57±0.19	0.06±0.03	0.05±0.06	0.06±0.01	0.05±0.02
CS	0.78±0.05	0.93±0.23	0.96±0.14	0.81±0.29	0.89±0.19

We used the same method and generate the same number, M, of random matrices as we did to initialize PM and UM, and selected the one with the highest score. We refer to this baseline as Rand.

4.2. Evaluation Criteria

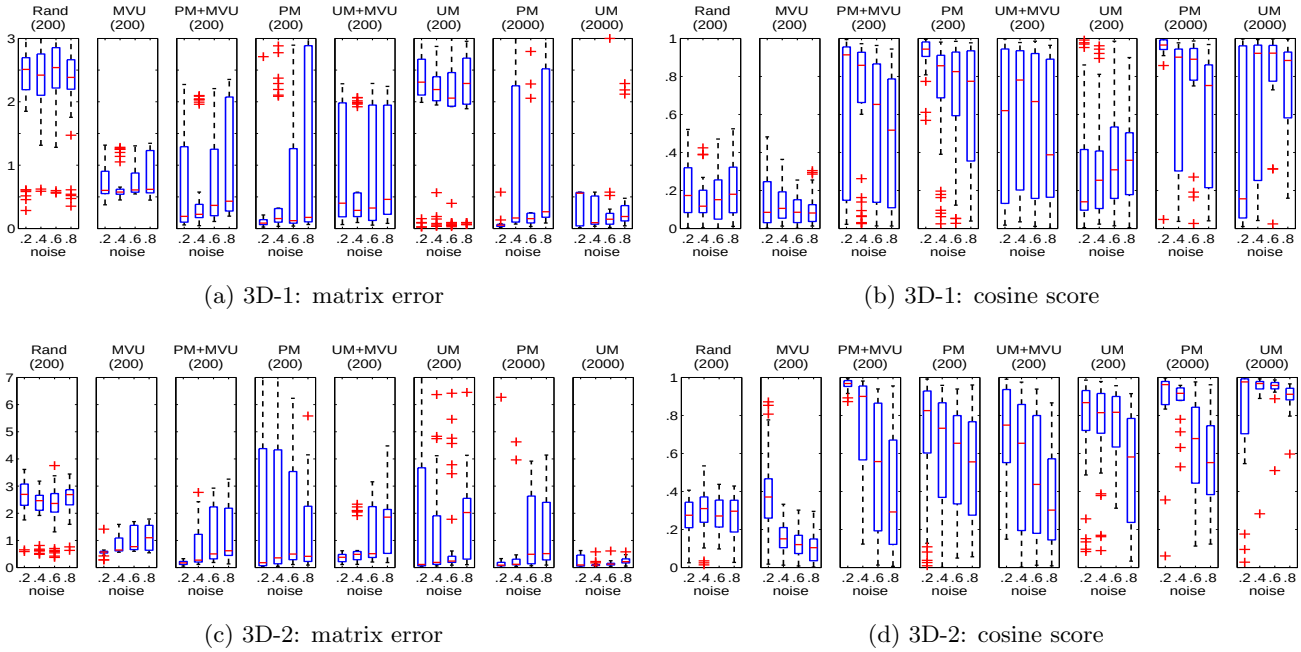
One evaluation measure is the error $\|\hat{A} - A\|_F$ between the estimated system matrix $\hat{A}$ and the true one A. However, due to the lack of time stamps, the step size of the system may not be identifiable from the data, and the best possible estimate may have a step size much smaller or larger than the true one. Another identifiability issue is the direction: It may happen that the reverse dynamics of a system, meaning going backward in time, explains the data equally well. In those cases, even an estimate that successfully captures the dynamics may produce a large error. We thus propose the following rate-adjusted matrix error:

$$\mathrm{ME}(A, \hat{A}) \equiv \min_t \|A - \hat{A}^t\|_F, \tag{21}$$

where $\hat{A}^t$ is $\hat{A}$ raised to the power t. The minimum in (21) is hard to solve, so we search for t in $\{\pm 1, \pm 2, \ldots, \pm 10, \pm 1/2, \pm 1/3, \ldots, \pm 1/10\}$ and choose the one that minimizes (21). While (21) takes into account the changing rate, it has the potential to overstate the quality of an estimate since it optimizes for the evaluation criterion. To have a more even evaluation, we consider another criterion that compares system matrices based on gradients at the data points:

$$\mathrm{CS}(A, \hat{A}) \equiv \frac{1}{N} \left| \sum_{i=1}^{N} \frac{(A\mathbf{x}_i - \mathbf{x}_i)'(\hat{A}\mathbf{x}_i - \mathbf{x}_i)}{\|A\mathbf{x}_i - \mathbf{x}_i\| \|\hat{A}\mathbf{x}_i - \mathbf{x}_i\|} \right|, \tag{22}$$

which we refer to as the cosine score. This criterion measures the similarity between the gradient $A\mathbf{x}_i - \mathbf{x}_i$ of the true system and that of the estimated system, averaging over all the sample points; a higher score



(a) 3D-1: matrix error

(b) 3D-1: cosine score

(c) 3D-2: matrix error

(d) 3D-2: cosine score

Figure 2. Results on 3D-1 and 3D-2. The title of each diagram indicates the method and the sample size. For each noise level, performance measures on different samples are presented in standard box plots with outliers marked as red crosses.

(22) thus means a better estimate. Note that cosine is a normalized measure of similarity, and therefore alleviates the issue of different system step sizes. Also, since (22) takes the absolute value after averaging, going forward and backward in time are considered equally good as long as they do so consistently.

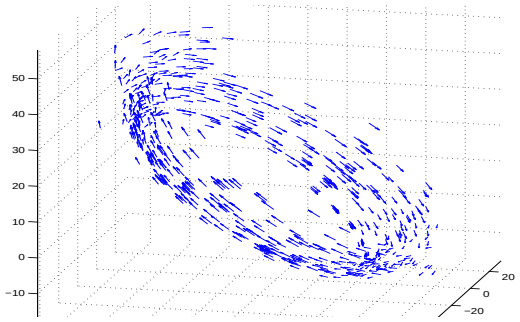
Although our algorithms estimate σ^2, the approximations we employ make the estimates biased. This is because the estimates effectively encapsulate error due to the noise in the dynamic system *plus* error due to the approximation of the continuous state distributions by the data. Thus, the estimated σ^2 values are much higher than those from the system and the comparison of them is omitted. Development of an appropriate unbiased estimate is a topic for future work.

4.3. Results and Findings

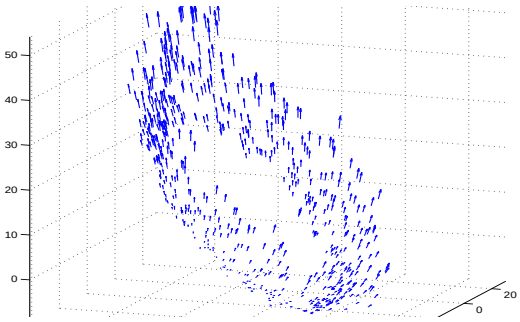
We tested the following methods: MVU, PM+MVU (PM initialized by MVU), PM, UM+MVU (UM initialized by MVU), UM, and Rand. Results on 2D are in Table 1; MVU is omitted due to space limitation. For this baseline system, every approach performs quite well. Figure 1 shows gradients $(\hat{A}\mathbf{x}_i - \mathbf{x}_i)$ given by UM in one of the 2D samples, which are quite consistent with the true dynamics.

Results on the more complex systems, 3D-1 and 3D-2, are in Figure 2. Since Rand is independent of data, we only report its results on the small samples. We did not apply MVU to the large-sized samples with 2,000 data points, since its underlying semidefinite program requires a huge amount of time and memory. Moreover, MVU alone usually gave cosine scores as low as Rand, and as an initialization, it provided little or no improvement over random initialization in most cases except UM in small-sized experiments for 3D-1. UM was competitive with or better than PM in quite a few cases. However, on the small samples of 3D-1, PM performed much better than UM. We also see that as the sample size grew, UM improved more significantly than PM did. This suggests that imposing directionality constraints may improve the estimation when samples are small, but it does so at the expense of introducing some bias to the estimate.

Regarding the effects of different noise levels, most methods became worse as the noise level increased. While UM was the most robust against noise in several cases, it performed very badly on the large samples of 3D-1 when $\sigma = 0.2$, but dramatically improved as noise increased. We found that for the 20 large samples of 3D-1 generated with $\sigma = 0.2$, UM recovered the true system matrix on nearly half of them, but totally failed on the rest. When it failed, the estimated A was always nearly diagonal and exhibited dynamics as depicted in Figure 3(b). This suggests a potential limitation of UM: Some combinations of dynamics and noise generate samples on which UM finds more than one type of estimates that are equally good. Estab-



(a) PM (cosine score: 0.9921)



(b) UM (cosine score: 0.0108)

Figure 3. Gradients on 3D-1, 2000-points sample, $\sigma = 0.2$. For better visualization, only 1/3 of the points are plotted.

lishing concrete statements on the limitations of our methods is thus an important future direction.

Finally, we present several gradient plots in Figures 3 and 4, which show arrows $A\mathbf{x}_i - \mathbf{x}_i$ at sample points. For 3D-1, we show a sample where PM succeeded but UM failed. For 3D-2, PM and UM performed similarly on most samples, so we present a typical result by PM.

5. Conclusions and Future Directions

We propose and study the problem of learning linear dynamical systems from non-sequenced data. This problem is found in several disciplines, ranging from Astronomy to Biology and Medicine. We propose two models that approximate the true likelihood; one ignores the hidden order relations between observations, while the other employs partial order constraints. We develop simple estimation algorithms for both, and evaluate them on several synthetic data sets. There are many interesting opportunities for future work in developing corresponding methods for other dynamic models (e.g. discrete state, partial observability, nonlinear dynamics, etc.). We plan to apply our methods to real astronomical and medical data to assess their ability of making meaningful scientific discoveries.

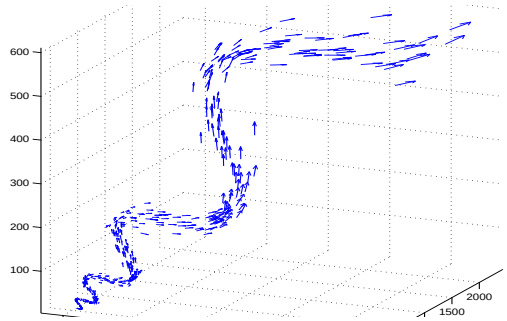


Figure 4. Gradients by PM on 3D-2, 2000-points sample, $\sigma = 0.2$. Cosine score: 0.9574. For better visualization, only 1/3 of the points are plotted.

References

Bishop, C. M. (2006). *Pattern recognition and machine learning*. New York: Springer.

Bock, F. (1971). An algorithm to construct a minimum directed spanning tree in a directed network. In *Developments in operations research*, 29–44.

Camerini, P. M., Fratta, L., & Maffioli, F. (1979). A note on finding optimum branchings. *Networks*, *9*, 309–312.

Chu, Y. J., & Liu, T. H. (1965). On the shortest arborescence of a directed graph. *Science Sinica*, *14*, 1396–1400.

Edmonds, J. (1967). Optimum branchings. *J. Res. Natl. Bur. Stand.*, *71B*, 233–240.

Globerson, A., Koo, T., Carreras, X., & Collins, M. (2007). Structured prediction models via the matrix-tree theorem. *Conf. on Emp. Methods in Nat. Lang. Process. – Comp. Nat. Lang. Learn.* (pp. 141–150).

Nicholson, V. A. (1975). Matrices with permanent equal to one. *Linear Algebra and its Applications*, *12*, 185–188.

Smith, D. A., & Smith, N. A. (2007). Probabilistic models of nonprojective dependency trees. *Conf. on Emp. Methods in Nat. Lang. Process. – Comp. Nat. Lang. Learn.* (pp. 132–140).

Tarjan, R. E. (1977). Finding optimum branchings. *Networks*, *7*, 25–35.

Valiant, L. G. (1979). The complexity of computing the permanet. *Theoretical Computer Science*, *8*, 189–201.

Weinberger, K. Q., Sha, F., & Saul, L. K. (2004). Learning a kernel matrix for nonlinear dimensionality reduction. *Int. Conf. on Mach. Learn.* (pp. 839–846).