Exploiting Non-sequence Data in Dynamic Model Learning

Tzu-Kuo Huang

October 2013
CMU-ML-13-112

Carnegie Mellon

Exploiting Non-sequence Data in Dynamic Model Learning

Tzu-Kuo Huang

October 2013
CMU-ML-13-112

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Jeff G. Schneider, Chair
Ziv Bar-Joseph
Geoffrey J. Gordon
Ali Shojaie, University of Washington

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was sponsored by the Air Force Research Laboratory under grant numbers FA865010C7059 and FA87501220324; the Department of Energy under grant number DESC0002607; the National Science Foundation under grant number IIS0911032; the Department of the Treasury under grant number TIRNO06D00020; the Association of Universities for Research in Astronomy, Inc. under award number C10625A, and a grant from ECCO.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Dynamic Model, Vector Auto-regression, Hidden Markov Model, Latent Variable Model, Expectation Maximization, Spectral Learning

For my family.

Abstract

Virtually all methods of learning dynamic models from data start from the same basic assumption: that the learning algorithm will be provided with a single or multiple sequences of data generated from the dynamic model. However, in quite a few modern time series modeling tasks, the collection of reliable time series data turns out to be a major challenge, due to either slow progression of the dynamic process of interest, or inaccessibility of repetitive measurements of the same dynamic process over time. In most of those situations, however, we observe that it is easier to collect a large amount of non-sequence samples, or random snapshots of the dynamic process of interest without time information.

This thesis aims to exploit such non-sequence data in learning a few widely used dynamic models, including fully observable, linear and nonlinear models as well as Hidden Markov Models (HMMs). For fully observable models, we point out several issues on model identifiability when learning from non-sequence data, and develop EM-type learning algorithms based on maximizing approximate likelihood. We also consider the setting where a small amount of sequence data are available in addition to non-sequence data, and propose a novel penalized least square approach that uses non-sequence data to regularize the model. For HMMs, we draw inspiration from recent advances in spectral learning of latent variable models and propose spectral algorithms that *provably* recover the model parameters, under reasonable assumptions on the generative process of non-sequence data and the true model. To the best of our knowledge, this is the first formal guarantee on learning dynamic models from non-sequence data. We also consider the case where little sequence data are available, and propose learning algorithms that, as in the fully observable case, use non-sequence data to provide regularization, but does so in combination with spectral methods. Experiments on synthetic data and several real data sets, including gene expression and cell image time series, demonstrate the effectiveness of our proposed methods.

In the last part of the thesis we return to the usual setting of learning from sequence data, and consider learning bi-clustered vector auto-regressive models, whose transition matrix is both sparse, revealing significant interactions among variables, and bi-clustered, identifying groups of variables that have similar interactions with other variables. Such structures may aid other learning tasks in the same domain that have abundant non-sequence data by providing better regularization in our proposed non-sequence methods.

Acknowledgments

First and foremost, I want to thank my advisor Jeff Schneider for his support and guidance throughout the years. I appreciate his ability of helping his students pursue their own research interests, while ensuring they make constant progress towards the completion of their degrees. Also admirable is his creativity and boldness in forming research agenda, as exemplified by the subject of this thesis research. If I had been able to push the frontier of our field any further, it was all because of Jeff taking the lead on exploring a new territory.

I also want to thank my thesis committee members, Ziv Bar-Joseph, Geoff Gordon, and Ali Shojaie for their encouragement, feedbacks and suggestions on this thesis work. When I am in doubt, their appreciation of my research reminds me that I am not the only one who finds it worth pursuing.

Heartfelt thanks go to current and former members of the Auton lab, including but not limited to: Artur Dubrawski, Barnabás Póczos, Roman Garnett, Yi Zhang, Robin Sabhnani, Liang Xiong, Madalina Fiterau, Dougal J. Sutherland, Yifei Ma, Xuezhi Wang, Junier Oliva, and Guillermo Cabrera. I very much enjoy our free-style, stimulating brainstorming sessions, which do help to break the box in my mind. The system administration team of Michael Baysek, Donghan Wang, and Predrag Punoševac does a great job in providing a powerful computing platform, without which none of this research could have been conducted. Karen Widmaier makes sure that all the logistic issues regarding conference travels and such are properly and efficiently handled.

I thank my fellow graduate students in the Machine Learning Department and the School of Computer Science at Carnegie Mellon University: Xi Chen, Hai-Son Le, Rob Hall, Julian Shun, Vincent Chu, Aaditya Ramdas, Martin Azizyan, Antonio Juarez, Min Xu, Wei Dai, and more, for their friendship and efforts in fostering a vibrant research environment. Thanks to Diane Stidle for her help since my first day in the Ph.D. program. Thanks to Yu-Ting Weng of University of Pittsburgh for bringing up many interesting discussions about research and life.

Thinking back to the very beginning of my research journey, I cannot forget to express my gratitude to Professor Chih-Jen Lin of National Taiwan University, my first research advisor who showed me the way of doing great research not only by words but mostly through his living. His insistence on research quality, pursuit of perfection, and desire for real understanding make him a long-lasting role model for me as well as many of his students.

Pittsburgh has become a home away from home thanks to my dear brothers and sisters in Christ: Faye and Alex Wang, Sharon and Zack Niu, Amy and Samuel Tu, Ling Shen and Jun Yu, Yan Liu and Shugong Wang, Irene and Joshua Lu, Jing Zhao and Yen-Chih Lin, Dylan Fang, Ker-Jiun Wang, Tim Savisky, Emily and Shawn

Fair, Pat and James Ruffin, Marilyn and Joe Pope, Esther and Vern Yoder, Jonathan Huang, Frank Lin, Jennifer and Abraham Lu, Ellen and Tom Piccone, Benjamin Cowley, George Montañez and many more. No matter where we go, we are one in the Lord.

Deep appreciation goes to my family for always being there for me. Since an early stage in my life, my parents Chao-Hsin Wang and Tao-Yu Huang have encouraged me to think independently and act responsibly, to develop my own view about the world rather than simply following the norm. They always believe in my potential and support me in whatever I pursue. I hope I have made them proud. Right after I left home for the US five years ago, my grand parents started counting the days to my return as the first Ph.D. in my family. While I wish they could have lived to see that happens, they have gone to be with the Lord. This thesis is to Pao-Pei Chen and Chuan-Liu Huang, my beloved grand parents. Special thanks to Lien-Chu Chen, my mother-in-law, for her help in a time when I needed her the most. Words cannot describe my gratefulness to Pei-Chun, my most wonderful wife, partner and friend, for her love, sacrifices and unconditional support during the past few years. She gave up her comfortable life at home to join me in a foreign country facing many challenges. Yet she got through them, for me. Finally, Isaac deserves to be on this list for being the source of surprise and joy in my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Learning dynamic models from data is the traditional topic of system identification [Ljung, 1999] in control theory and many algorithms have been proposed. In the machine learning literature, the learning of temporal graphical models, such as dynamic Bayesian networks [Ghahramani, 1998a; Murphy, 2002], and the learning of various types of Markov models [e.g., Abbeel and Ng, 2005; Beal et al., 2002; Ghahramani, 1998b; Hsu et al., 2009; Rabiner, 1989; Song et al., 2010], have been extensively studied.

Virtually all methods of learning dynamic models from data start from the same basic assumption: that the learning algorithm will be provided with a single or multiple sequences of data generated from the dynamic model. However, in quite a few modern dynamic modelling tasks, a major difficulty turns out to be the collection of reliable time series data. In some of these tasks, such as learning dynamic models of galaxy or star evolution, the dynamics of the processes of interest are far too slow for researchers to collect successive data points showing any meaningful changes. At more modest time scales, the same problem arises in the understanding of slow-evolving human diseases such as Alzheimer's or Parkinson's, which may progress over a decade or more. In other situations, the dynamic process of interest may not be able to undergo repetitive measurements, so researchers have to measure multiple instances of the same process while maintaining synchronization among these instances. One such example is gene expression time series. In their study, Tu et al. [2005] measured expression profiles of yeast genes along consecutive metabolic cycles. Due to the destructive nature of the measurement technique, they collected expression data from multiple yeast cells. In order to obtain reliable time series data, they spent a lot of effort developing a stable environment to synchronize the cells during the metabolic cycles. Yet, they point out in their discussion that such a synchronization scheme may not work for other species, e.g., certain bacteria and fungi, as effectively as for yeast. Another example is cell image time series. In a recent study [Buck et al., 2009] on cell cycle dependence of protein subcellular location inferred from images, the authors discussed some challenges in obtaining time series of cell images: "*... time-lapse images can be more difficult to obtain than single images of cells because many microscopes do not maintain a viable environment for the cells they image (e.g., cells die after some time, and even while alive they are not under constant conditions). Furthermore, repeated excitation of dyes for fluorescence imaging causes photobleaching, reducing signal and leading to toxic chemical changes (phototoxicity), further perturbing cells.*"

Table 1.1: Summary of thesis work

		Model Class	
		First-order Observable	**Hidden Markov Model**
Data Assumption	**seq.+ non-seq.**	• Non-sequence data as regularization • Significant improvement over standard sequence-only methods when sequence data is few [Chapters 4 and 6]	
	fully non-sequence	• EM-type algorithms maximizing approximate likelihood • Synthetic data, gene expressions and cell images [Chapter 3]	• Spectral algorithms with formal guarantee • First theoretical statement on learning from non-sequence data [Chapter 5]
Learning Bi-clustered Vector Auto-regressive Model [Chapter 7]			

While obtaining reliable time series can be difficult, it is often easier to collect non-sequence samples, or snapshots of the dynamic process of interest. For example, the Sloan Digital Sky Survey (SDSS)[1] has collected images of millions of celestial objects, each of which may be in a different phase of its life cycle. In medical sciences, a scientist studying Alzheimer's or Parkinson's can collect samples from his or her current pool of patients, each of whom may be in a different stage of the disease. Or in gene expression analysis, current technology already enables large-scale collection of static gene expression data. It is also the case in cell image analysis, as concluded by Buck et al. [2009]: *"A method using un-synchronized cells with single-image capture would have the advantages of avoiding repeated exposure to fluorescence excitation (permitting higher-energy exposure to obtain better signal) and fewer environment viability requirements."*

More broadly, in social and medical sciences it is usually the case that *longitudinal study*, the collection and analysis of data from the same subjects over long periods of time, is more powerful but also expensive than *cross-sectional study*, which uses observations collected from a large or representative portion of the population within a short time frame. With recent advances in sensing technology, there will likely be a large increase in cross-sectional data in various domains, and it would be great if they can be used not only in cross-sectional study but also to aid longitudinal study.

1.1 Thesis Summary

Motivated by challenges in time series data collection for a variety of modern dynamic modeling tasks, we propose and study several methods for learning various dynamic models using non-

[1] http://www.sdss.org/

sequence data that lack time information but are easy to obtain. Table 1.1 summarizes our thesis work and contributions. In brief, we consider learning two classes of dynamic models: first-order observable models and hidden Markov models (HMMs), under two conditions on the input data. When the input data consists of both sequence and non-sequence samples, our proposed methods use non-sequence data as regularization to existing sequence-only learning methods, and achieve significant improvement when sequence data is few. In the more challenging situation where all the input data are non-sequence, our methods for learning first-order observable models maximize approximate likelihood functions via EM-type procedures, and obtain encouraging results on synthetic data as well as several real data sets, including gene expression data and cell images. For HMMs, we take advantage of recent advances in spectral learning [Anandkumar et al., 2012a] and identify reasonable generative assumptions on non-sequence data that lead to spectral methods with consistent parameter learning guarantees. To the best of our knowledge, this is the first theoretical statement on learning from non-sequence data.

1.2 Thesis Overview

After surveying related work in Chapter 2, we first consider in Chapters 3 and 4 learning fully observable dynamic models. In Chapter 3, we assume the only data available are snapshots taken from multiple instantiations of a dynamic process at unknown times, and the dynamic process falls in the class of fully observable, discrete-time, first-order linear or non-linear dynamic models. Acknowledging several issues in model identifiability, we developed EM-type learning algorithms that maximize approximate likelihood functions, along with novel initialization methods based on the idea of temporal smoothing. In a number of experiments on synthetic and real data sets including gene expression data and cell images, the proposed algorithms are able to learn moderately to highly accurate dynamic models, but at times suffer severely from the model ambiguity inherent in this setting.

We thus in Chapter 4 consider slightly stronger assumptions: in addition to non-sequence data, a small amount of sequence data are also available. We restrict the class of dynamic models to first-order discrete-time stable vector auto-regressive (VAR) models, and assume the non-sequence data are independent samples drawn from the stationary distribution of the VAR model. The latter assumption is valid when, for example, snapshots are taken from multiple trajectories of a VAR process after they have reached stationarity. Based on these assumptions, we proposed learning algorithms that minimize a new penalized least square objective, which incorporates non-sequence data in a novel regularization term that quantifies violation of the Lyapunov equation relating the autoregressive model to the covariance of its stationary distribution. Experiments demonstrate that when the amount of sequence data is small, our proposed method of exploiting non-sequence data can significantly improve over standard learning algorithms, which use only the sequence data.

Although fully observable models like VAR are useful, in many applications only a subset of the variables in the underlying dynamical system can be observed. Thus in Chapters 5 and 6 we turn to learning dynamic models with hidden states. At first glance this seems formidable because even when sequence data are available, learning hidden-state models is in general difficult both statistically and computationally. However, an emerging line of research in machine

learning, known as spectral learning, has recently developed statistically consistent and computationally efficient algorithms for learning from sequence data perhaps the most widely-used class of hidden-state models, hidden Markov models (HMMs) [Anandkumar et al., 2012b; Hsu et al., 2009; Siddiqi et al., 2010; Song et al., 2010]. Unlike traditional EM-based learning methods, which are vulnerable to bad local optima, these new methods are based on spectral decomposition, such as Singular Value Decomposition (SVD), of empirical moments computed from data, and therefore result in *unique, local-minima free* estimates of model parameters, allowing formal statistical guarantees to be established. Building on these recent advances, we propose spectral algorithms for learning HMMs that exploit non-sequence data.

In Chapter 5 we consider the case where only non-sequence data are available. However, unlike in Chapter 3 where all the data points are assumed to have the same initial condition, here we need *multiple sets* of non-sequence data, each generated from a different initial hidden-state distribution. The main contribution of this chapter is to identify conditions on the initial hidden-state distributions, by drawing connections to spectral learning of Latent Dirichlet Allocation (LDA) models [Anandkumar et al., 2013], as well as distributional assumptions on the missing time information that allow us to develop spectral algorithms with formal guarantees on HMM parameter learning. To the best of our knowledge, these are the first theoretical guarantees in learning from non-sequence data. Compared with EM-based methods in simulation, our spectral algorithms perform significantly better in parameter estimation.

Then in Chapter 6 we look at the situation where, as in Chapter 4, some sequence data are available and the non-sequence data consist of independent samples from the stationary distribution of the underlying HMM. Extending state-of-the art spectral algorithms for learning *observable representation* of HMMs [Hsu et al., 2009; Siddiqi et al., 2010; Song et al., 2010], our proposed methods obtain improved estimates of lower-order moments by minimizing estimation error on the sequence data plus a regularization term on the non-sequence data, and then apply spectral decomposition to the improved moment estimates. Interestingly, although the high-level idea is similar to that of Chapter 4 and HMMs are more complex models than VARs, the optimization problems in this chapter turn out to be convex whereas the ones in Chapter 4 are non-convex. Experiments on simulated data and sensor recordings of human activities demonstrate improvement over existing sequence-only spectral algorithms.

In the final part of the thesis, Chapter 7, we return to the traditional setting of learning from sequence data and focus on learning structured vector auto-regressive models. Although this chapter is not directly related to the main theme of the thesis, the methodology developed here can aid learning in the non-sequence setting through its estimated structure of the VAR model, which may guide the design of the regularization terms in the proposed EM-type methods (Chapter 3) when applied to non-sequence data in the same domain. We are motivated by problems in biological time series analysis, where dependency graph and clustering of variables, such as expression levels of genes, are two of the most commonly sought structures. In spite of being closely related, these two structures are usually estimated in separate procedures. We thus propose a fully Bayesian approach to simultaneous learning of these two structures for vector auto-regressive models, using a novel bi-clustered and sparsity-promoting prior for the transition matrix and an efficient blocked Gibbs sampling procedure for posterior inference. Applied to a T-cell activation gene expression time series data set [Rangel et al., 2004], this new method finds a more biologically meaningful clustering of genes than state-of-the art gene expression time

series clustering methods.

This thesis contains our published work in several venues:

- Chapter 3 [Huang and Schneider, 2009; Huang et al., 2010]
- Chapter 4 [Huang and Schneider, 2011]
- Chapter 5 [Huang and Schneider, 2013b]
- Chapter 6 [Huang and Schneider, 2013a]
- Chapter 7 [Huang and Schneider, 2012]

Chapter 2

Related Work

In a good number of applications, a critical issue is to understand the dynamics or temporal dependency underlying observed data that lack temporal or sequential information. As a result, various methods were proposed independently in different areas, but to the best of our knowledge, no prior work studies the general problem of learning dynamic models from non-sequence data as comprehensively as this thesis. In this chapter we survey several such applications and briefly explain the methods developed therein.

As mentioned in Chapter 1, cell imaging has become a useful tool for studying certain types of cell dynamics, such as variation in protein subcellular localization during the cell cycle [Buck et al., 2009]. Instead of relying on time-series cell images as in most previous studies, Buck et al. [2009] propose to utilize static, asynchronous snapshots taken from multiple cells at various phases of the cell cycle because, as quoted in Chapter 1, such images are easier to obtain on a large scale than time-series images. Their approach is to first extract a one-dimensional surrogate of cell cycle time from static cell image features by manifold learning techniques[1], and then use this surrogate in place of cell cycle time for subsequent cell-cycle dependence tests. Through analysis of real data, they confirm that such a surrogate is well correlated with the cell cycle. However, they did not perform explicit dynamic modeling, i.e, building models to predict future observations.

A closely related problem studied in a number of disciplines is that of ordering a set of objects. Depending on the domain of interest, an ordering can be interpreted as progression of time, some coherent sequential structure or monotonic property. In natural language processing, the task of multi-document summarization requires ordering of sentences selected from different documents, and automatic title generation techniques construct a headline by selecting and ordering words from the input text [Barzilay and Elhadad, 2002; Deshpande et al., 2007]. In multimedia analysis and retrieval, automatic generation of video or slideshow from photos involves laying down a coherent and smoothly transitioning sequence of scenes [Chen et al., 2006; Hua et al., 2004]. Some of the techniques developed for these tasks are tailored to a specific problem domain, and most of them have access to some external knowledge about orderings

[1]Manifold learning techniques have been used in dynamic model learning to identify a subspace where the dynamics reside, leading to more accurate models. See, for example, [Boot and Gordon, 2011] and references therein. Similar techniques can be used in combination with our proposed methods as a pre-processing step to make the problem lower-dimensional and thus easier.

of objects, such as time stamps of photos or grammatical rules for sentence compositions. In contrast, we consider a more general problem setting which relies on no or little domain specific knowledge, though our proposed methods make more explicit model assumptions.

The computational biology community has also studied the problem of ordering objects, in the context of finding a temporal ordering of static, asynchronous microarray measurement data [Gupta and Bar-Joseph, 2008; Magwene et al., 2003]. The proposed methods therein are less domain dependent and fall in a large family of algorithms for solving the *curve reconstruction problem*, which has been studied in various fields such as computational geometry (e.g., Giesen [1999]), statistics [Hastie and Stuetzle, 1989], and machine learning [Smola et al., 2001]. More specifically, Magwene et al. [2003] proposed to reconstruct the temporal ordering of microarray samples through finding the minimum spanning tree on the graph formed by the sample points, while Gupta and Bar-Joseph [2008] proposed to solve an instance of the traveling salesman problem (TSP) and proved that under certain conditions on the dynamics generating the samples, the optimal TSP path accurately reconstructs the true ordering. A key assumption behind these two methods is that temporally close sample points should also be spatially close. Both of these methods are unable to choose an overall direction of time, a limitation due to the invariance to time direction in their objective functions. Our problem setting differs from the aforementioned in that we consider snapshots from *multiple trajectories* of some dynamic process rather than out-of-order samples from a *single sequence*. Moreover, we focus more on learning a model for the underlying dynamics than ordering the data points. Although the non-sequence data considered in our settings, as formalized in later chapters, can be ordered based on their unobserved time stamps, such an ordering may not be very useful to existing dynamic model learning methods because these methods require as input sequences tracking *the same instances* over time. Nevertheless, ordering objects is still a useful component in our proposed methods in Chapter 3, but the objects being ordered, instead of raw data points, are some representative points discovered by clustering algorithms.

Another problem involving learning dynamic models without temporal ordering is the network structure inference problem considered by Rabbat et al. [2008]. The authors point out that in many situations, ranging from telecommunication network tomography problems to construction of biological signal pathways or social networks, the goal is to reconstruct a directed graph representing the underlying network structure, but the only available data are sets of nodes *co-occurring* in random walks on the graph without the order in which they were visited. These problem can be cast as learning a first-order Markov chain from data lacking ordering information. To avoid the exponential-time complexity of enumerating all possible orderings, the authors propose a polynomial-time, importance sampling based EM algorithm with convergence guarantee to estimate the parameters of the Markov chain. Inspired by Rabbat et al. [2008], several researchers in computational linguistics study the problem of learning a bi-gram language model from the commonly-used, order-invariant bag-of-words representation of text corpus [Zhu et al., 2008], and develop a similar sampling-based EM algorithm. While empirically successful to some extent, these algorithms, like most EM procedures, do not have guarantees on the quality of their parameter estimates. Very recently, Gripon and Rabbat [2013] propose a combinatorial algorithm for graph reconstruction from co-occurrence data and provide some theoretical guarantees on the reconstruction accuracy. However, their results apply only to undirected graphs and require the input to the algorithm to be the exact set of triples of nodes that are connected but

cycle-free in the graph. In Chapter 5 we also study the problem of learning first-order Markov chains from data lacking temporal information. However, instead of data with hidden orderings, we consider data drawn from multiple, independent trajectories of the underlying Markov chain, so there was no ordering to begin with. At first glance, learning in this setting may seem more difficult than in the hidden-ordering setting, but as detailed in Chapter 5, the independence assumption in our setting actually makes learning easier.

In addition to the above general problem areas, there are two specific problems we find relevant to our work. One is collective inference on Markov models [Sheldon et al., 2008], which finds the most likely collection of paths on a trellis graph given observations on the collective behavior of a group of dynamic objects. Their motivation was to trace out trajectories of individual birds from aggregate statistics of an entire species of migrating birds. The other is connecting the dots between news articles [Shahaf and Guestrin, 2010], which aims to build a chronological *and* coherent story line of news that connects a given pair of starting and end articles, thereby providing readers a detailed description of the causal relationship between two events. A common feature in both problems is the need of identifying structures of sequentially matched objects from partially ordered data. A similar situation arises in one component of our methods, where the data points are put into ordered clusters for further processing (Section 3.3). But instead of finding hard matchings between data points in adjacent clusters, we take a soft-matching type of approach, updating the soft matching and the dynamic model alternatingly.

While our focus is on learning from data lacking time or ordering information, another common problem involving time in dynamic modeling is the misalignment of time measurements across multiple sequences of observed data, due to internal variation of the dynamic process of interest or measurement error. This problem arises in many time series modeling tasks, such as speech recognition [L. Rabiner, 1993; Vintsyuk, 1968], analysis of gene expression time series [Aach and Church, 2001], activity recognition [Junejo et al., 2011], and audio information retrieval [Chapter 4, Müller, 2007], bringing forth a large body of research, known in statistics as curve registration [Ramsay and Li, 1998; Silverman, 1995] and in computer sciences as dynamic time warping [Berndt and Clifford, 1994; Keogh and Ratanamahatana, 2005]. The general idea in these works is to first postulate a class of possible time transformations or warping operations, and then recover the most likely warping operation for each observation by optimizing some global matching score across all the data sequences. The final result is time-warped sequences of observations that are in better alignment with one another. While not directly related to our thesis focus, these methods can potentially aid our work in, for example, an iterative, EM-like manner, where time stamps and dynamic models are alternatingly re-estimated given the other until convergence.

Finally, we briefly mention where our work lies in the vast space of research on dynamical systems conducted in physics and mathematics. Most dynamical theories are concerned with the asymptotic behavior of some dynamical system, under various assumptions on the phase or state space of the system and the short-time evolution law [Katok and Hasselblatt, 1996]. But our work studies in some sense the reverse problem, that is, given observations that reflect the global status of a dynamical system, we try to develop methods that figure out the short-time or local evolution law.

Chapter 3

Learning Fully Observable Models From Non-sequence Data

In this chapter, we are interested in learning first-order, discrete-time, fully observable linear dynamic models described by the following transition function:

$$\mathbf{x}^{(t+1)} = A\mathbf{x}^{(t)} + \boldsymbol{\epsilon}^{(t+1)}, \tag{3.1}$$

where $\mathbf{x}^{(t)} \in \mathbb{R}^{p\times 1}$ is the state vector at time t, $A \in \mathbb{R}^{p\times p}$ is the state transition matrix, and $\boldsymbol{\epsilon}^{(t)}$ is the noise vector at time t. Such a model is also known as a first-order vector auto-regressive model (VAR) in the time series literature. For simplicity, we assume hereafter that $\forall t,\ \boldsymbol{\epsilon}^{(t)} \sim \mathcal{N}(\cdot \mid \mathbf{0}, \sigma^2 I)$, a Gaussian distribution with zero mean and covariance $\sigma^2 I$, where I is the identity matrix. However, the proposed methods in later sections all can be extended to handle general covariance matrices. The dynamical system also has a start state, which we denote as $\mathbf{x}^{(0)}$. Thus, the linear dynamic models we consider are fully characterized by $\Theta = \{A, \sigma^2, \mathbf{x}^{(0)}\}$.

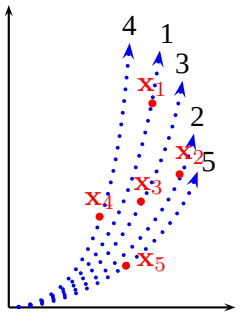
When sequenced observations are available, a basic learning method is least-square linear regression of the observations at time t on the observations at time $t-1$, whose properties have been studied extensively (see e.g., [Hamilton, 1994]). The problem without observed state sequences is much more difficult. We assume that n executions of the dynamic model (3.1) have taken place, and from each execution we have observed a single data point drawn at random from the sequence of states generated in that execution. The result is n data points, $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, each from a different trajectory and having occurred at an unknown point in time. To avoid confusion in indices, hereafter we use parenthesized super-script, e.g., $\mathbf{x}^{(t)}$, to denote the time index, but sub-script, e.g., $\mathbf{x}_i$, to denote the data index. A precise description of this generative process is given in Algorithm 3.1 along with a graphical illustration.

We focus on estimating A and σ^2, and treat the start state $\mathbf{x}^{(0)}$ as a nuisance parameter. For an observation $\mathbf{x}_i$, if its immediate predecessor $\tilde{\mathbf{x}}_i$ is known, then the likelihood is simply $\mathcal{N}(\mathbf{x}_i \mid A\tilde{\mathbf{x}}_i, \sigma^2 I)$. But $\tilde{\mathbf{x}}_i$ is unknown, so we integrate it out with respect to the distribution one time step earlier than $\mathbf{x}_i$ and obtain the following likelihood:

$$L(\mathbf{x}_i \mid \boldsymbol{\theta}, t_i) \;=\; \int \frac{\exp(-\frac{\|\mathbf{x}_i - A\mathbf{x}\|_2^2}{2\sigma^2})}{(2\pi\sigma^2)^{\frac{p}{2}}} \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}^{(t_i-1)}, \Sigma^{(t_i-1)}) d\mathbf{x}, \tag{3.2}$$

Algorithm 3.1 Sampling from multiple trajectories

Input: transition matrix 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**
    $\mathbf{x}^{(t)} \leftarrow A\mathbf{x}^{(t-1)} + \boldsymbol{\epsilon}^{(t)}$, $\boldsymbol{\epsilon}^{(t)} \sim \mathcal{N}(\cdot|\mathbf{0}, \sigma^2 I)$.
  end for
  Set $\mathbf{x}_i = \mathbf{x}^{(t_i)}$.
end for
Output: A sample $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$.



where t_i denotes the true but unknown time of $\mathbf{x}_i$, $\|\cdot\|_2$ is the vector two-norm, and the predecessor distribution, by the closure of Gaussian under linear transformation, is Gaussian with mean $\boldsymbol{\mu}^{(t_i-1)}$ and covariance $\Sigma^{(t_i-1)}$, where

$$\boldsymbol{\mu}^{(t)} := A^t\mathbf{x}^{(0)}, \qquad \Sigma^{(t)} := \begin{cases} \sigma^2 \sum_{i=0}^{t-1} A^i \left(A^i\right)^\top, & t \geq 1, \\ \mathbf{0}, & t = 0. \end{cases} \tag{3.3}$$

Since the n data points are drawn independently, we can factorize the likelihood of the sample points as

$$L(\mathbf{x}_1, \ldots, \mathbf{x}_n|\boldsymbol{\theta}, t_1, \ldots, t_n) = \prod_{i=1}^{n} L(\mathbf{x}_i|\boldsymbol{\theta}, t_i). \tag{3.4}$$

The maximization of (3.4) is a challenging task because, as suggested by (3.3), the transition matrix A appears in (3.4) as polynomials whose degrees depend on the missing time indices t_i's. In the following sections we proposed methods that avoid this difficulty by various approximations to (3.4), but before presenting our proposed methods, we first discuss several possibly non-identifiable properties of the model when the true temporal information is missing.

3.1 Identifiability Issues

Consider a simple linear dynamic model with the following transition matrix and initial point:

$$A = \begin{bmatrix} \cos\left(\frac{2\pi}{T}\right) & -\sin\left(\frac{2\pi}{T}\right) \\ \sin\left(\frac{2\pi}{T}\right) & \cos\left(\frac{2\pi}{T}\right) \end{bmatrix}, \qquad \mathbf{x}^{(0)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

The ideal trajectory rolled out by this simple dynamic model lies on the unit circle in the two-dimensional Euclidean space. Suppose we observe a set of points from the ideal trajectory, but do not know their time indices. It is easy to see that all of the following dynamic models:

$$A(t) = \begin{bmatrix} \cos\left(\frac{2\pi t}{T}\right) & -\sin\left(\frac{2\pi t}{T}\right) \\ \sin\left(\frac{2\pi t}{T}\right) & \cos\left(\frac{2\pi t}{T}\right) \end{bmatrix}, \qquad t \in \{\pm 1, \pm 2, \ldots, \pm(T-1)\},$$

as illustrated in Table 3.1, would explain the data equally well under any reasonable measure of goodness of fit. In the presence of process noise, some of these models may become less likely, but it would still be hard to uniquely determine the true dynamic model.

Table 3.1: An example demonstrating unidentifiability of time direction and speed

Data	Three equally possible dynamic models

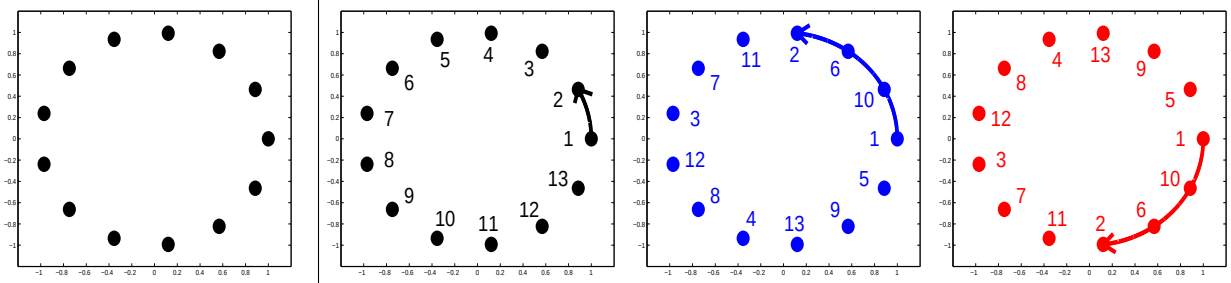
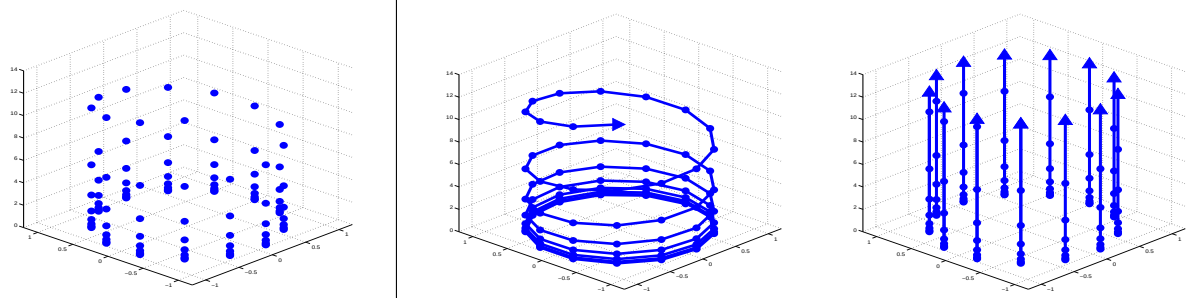


Table 3.2: An example demonstrating general unidentifiability.

Data	Model 1	Model 2



The above example suggests two possibly non-identifiable properties of the model: the overall direction in time and the speed of the underlying dynamics. In fact, Peters et al. [2009] showed that under some linear dynamic models the true direction in time is not identifiable. The methods proposed in subsequent sections thus do not resolve these ambiguities; the learnt model may follow either of the two directions in time, but usually corresponds to the slowest dynamics.

Another perhaps more intriguing example is depicted in Table 3.2, which presents a non-sequenced and noiseless data set in the left column and two possible dynamic models in the right column. On the one hand, according to our assumption of a single fixed start state as in Algorithm 3.1, Model 1 should be favored over Model 2 under any reasonable measure of goodness of fit that incorporates such an assumption. On the other hand, under a certain level of noise and/or some non-uniform sampling rate in the temporal domain, the data generated from Model 1 may be more similar to a cylinder than to a spiral, making Model 2 equally or even more likely to have generated the data. There are more examples of this type, such as a torus of points where rotations around the short and the long circumferences can hardly be distinguished from each other in the absence of any temporal information. Theoretical investigation into such issues as conditions under which these ambiguities can or cannot be resolved is thus an important, but challenging future direction.

3.2 Approximate Likelihood and Expectation Maximization

We present three methods for estimating A and σ^2 in Sections 3.2.1 to 3.2.3, based on maximizing various approximations to the likelihood (3.4). The optimization is carried out by Expectation Maximization (EM) types of algorithms. Then Section 3.2.4 demonstrates extensions of these three methods for learning nonlinear dynamic models, which make use of reproducing kernels.

3.2.1 Unordered Approximation

We first remove the problem of unknown time indices by marginalizing out the missing t_i's. According to the generative process in Algorithm 3.1, the distributions of t_i's are independent from A and σ^2, and also mutually independent. Let $P(t_i)$ denote the probability mass function of $t_i \in \{1, \ldots, T_{\max}\}$. We then write

$$\begin{aligned} L(\mathbf{x}_1, \ldots, \mathbf{x}_n|\boldsymbol{\theta}) &:= \sum_{t_1=1}^{T_{\max}} \cdots \sum_{t_n=1}^{T_{\max}} L(\mathbf{x}_1, \ldots, \mathbf{x}_n, t_1, \ldots, t_n|\boldsymbol{\theta}) \\ &= \sum_{t_1=1}^{T_{\max}} \cdots \sum_{t_n=1}^{T_{\max}} \left(\prod_{i=1}^{n} L(\mathbf{x}_i|\boldsymbol{\theta}, t_i) P(t_i) \right) \\ &= \prod_{i=1}^{n} \sum_{t_i=1}^{T_{\max}} L(\mathbf{x}_i|\boldsymbol{\theta}, t_i) P(t_i). \end{aligned}$$

Plugging in the conditional likelihood (3.2), we obtain

$$\begin{aligned} L(\mathbf{x}_1, \ldots, \mathbf{x}_n|\boldsymbol{\theta}) &= \prod_{i=1}^{n} \sum_{t_i=1}^{T_{\max}} \left(\int \frac{\exp(-\frac{\|\mathbf{x}_i - A\mathbf{x}\|^2}{2\sigma^2})}{(2\pi\sigma^2)^{\frac{p}{2}}} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}^{(t_i-1)}, \Sigma^{(t_i-1)}) d\mathbf{x} \right) P(t_i) \\ &= \prod_{i=1}^{n} \left(\int \frac{\exp(-\frac{\|\mathbf{x}_i - A\mathbf{x}\|^2}{2\sigma^2})}{(2\pi\sigma^2)^{\frac{p}{2}}} \left(\sum_{t_i=1}^{T_{\max}} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}^{(t_i-1)}, \Sigma^{(t_i-1)}) P(t_i) \right) d\mathbf{x} \right) \end{aligned} \quad (3.5)$$

In the case of $P(t_i) = 1/T_{\max}$, i.e. t_i's are uniformly distributed, and $T_{\max}$ is large, we have

$$\begin{aligned} \sum_{t_i=1}^{T_{\max}} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}^{(t_i-1)}, \Sigma^{(t_i-1)}) P(t_i) &= \sum_{t_i=1}^{T_{\max}} \frac{\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}^{(t_i-1)}, \Sigma^{(t_i-1)})}{T_{\max}} \\ &\approx \sum_{t_i=1}^{T_{\max}} \frac{\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}^{(t_i)}, \Sigma^{(t_i)})}{T_{\max}}, \end{aligned} \quad (3.6)$$

which is the density that the data points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ are generated from. This gives another view of the generative process: a random sample point is drawn (approximately) from (3.6) and an observation is created by applying (3.1) to it. However, (3.6) still depends on the unknown t_i's. To remove this dependency, we replace (3.6) with its empirical estimate given by the sample

points we have. This together with (3.5) leads to the following approximate likelihood:

$$\widehat{L}(\mathbf{x}_1, \ldots, \mathbf{x}_n|\boldsymbol{\theta}) \ := \ \prod_{i=1}^{n} \left(\sum_{j \neq i} \frac{\exp(-\frac{\|\mathbf{x}_i - A\mathbf{x}_j\|^2}{2\sigma^2})}{(n-1)(2\pi\sigma^2)^{\frac{p}{2}}} \right). \tag{3.7}$$

We exclude the case that $\mathbf{x}_i$ generates itself to avoid the degenerate estimate $A = I$. To avoid overfitting, we impose a zero-mean Gaussian prior on A with precision λI and an inverse Gamma prior on σ^2 with shape and scale parameters α and β, leading to the approximate log-posterior:

$$\log \widehat{P}_{\text{UM}}(\boldsymbol{\theta}|\mathbf{x}_1, \ldots, \mathbf{x}_n) \ := \ \sum_{i=1}^{n} \log \left(\sum_{j \neq i} \frac{\exp(-\frac{\|\mathbf{x}_i - A\mathbf{x}_j\|^2}{2\sigma^2})}{(n-1)(2\pi\sigma^2)^{\frac{p}{2}}} \right) - \frac{\lambda}{2}\|A\|_F^2 - (\alpha+1)\log\sigma^2 - \frac{\beta}{\sigma^2}. \tag{3.8}$$

This is the representative form of the objective our first method aims to maximize. Later in the experiments we may use variants of (3.8) such as allowing more general noise variances, but all the associated methods can be easily derived from the methods based on (3.8), which we present below. Since there is no notion of ordering involved in (3.8), we refer to it as the Unordered Approximation.

Before introducing our optimization algorithm, we point out that (3.8) considers the data points as if each $\mathbf{x}_i$ were generated from some other $\mathbf{x}_j$ in the sample by (3.1). However, according to Algorithm 3.1 no $\mathbf{x}_i$ was generated from any other $\mathbf{x}_j$ in the sample. Such a discrepancy is due to our replacing (3.6) with its empirical estimate, and an immediate consequence is that σ^2 in (3.8) now accounts for not only the noise $\boldsymbol{\epsilon}$ in the dynamic model (3.1), but also the approximation error introduced by replacing (3.6) with the empirical density.

To present our learning algorithm, we observe that the likelihood (3.7) is a product of summations of Gaussian densities. 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 (3.8) is not a GMM, its similar structure allows us to derive an EM procedure with analytical update rules. We first introduce a latent variable matrix $Z \in \{0,1\}^{n \times n}$ such that

$$\begin{aligned} Z_{ij} &= \begin{cases} 1, & \mathbf{x}_i \text{ was generated from } \mathbf{x}_j \\ 0, & \text{otherwise} \end{cases}, \ \ j \neq i, \\ Z_{ii} &= 0, \quad \sum_{j=1}^{n} Z_{ij} = 1. \end{aligned} \tag{3.9}$$

Again, here "$\mathbf{x}_i$ was generated from $\mathbf{x}_j$" is to be taken as an approximation due to our replacing (3.6) with the data. We then rewrite (3.8) using the standard variational equation (c.f. Section 9.4, [Bishop, 2006]):

$$\log \widehat{P}_{\text{UM}}(\boldsymbol{\theta}|\mathbf{x}_1, \ldots, \mathbf{x}_n) \ = \ \log \sum_{Z} \widehat{P}_{\text{UM}}(\boldsymbol{\theta}, Z|\mathbf{x}_1, \ldots, \mathbf{x}_n), \tag{3.10}$$

where

$$\widehat{P}_{\text{UM}}(\boldsymbol{\theta}, Z|\mathbf{x}_1, \ldots, \mathbf{x}_n) \ := \ \left(\prod_{i=1}^{n} \prod_{j=1}^{n} \left(\frac{\exp(-\frac{\|\mathbf{x}_i - A\mathbf{x}_j\|^2}{2\sigma^2})}{(n-1)(2\pi\sigma^2)^{\frac{p}{2}}} \right)^{Z_{ij}} \right) \exp\left(-\frac{\lambda}{2}\|A\|_F^2 - \sigma^{2(\alpha+1)} - \beta/\sigma^2 \right)$$

Algorithm 3.2 Expectation Maximization for (3.8)

Input: Data points $\mathbf{x}_1, \ldots, \mathbf{x}_n$
Initialize $A_{(0)}$ and $\sigma^2_{(0)}$, set $k = 0$
repeat
Update $\widetilde{Z}_{(k+1)}$ by (3.11) with $A_{(k)}$ and $\sigma^2_{(k)}$
Update $A_{(k+1)}$ by (3.13) with $\widetilde{Z}_{(k+1)}$ and $\sigma^2_{(k)}$
Update $\sigma^2_{(k+1)}$ by (3.14) with $A_{(k+1)}$ and $\widetilde{Z}_{(k+1)}$
$k \leftarrow k + 1$
until The approximate log posterior (3.8) does not increase

is referred to as the complete posterior. Following the standard EM derivation, in the E-step we compute the posterior probability of Z:

$$Q(Z|\boldsymbol{\theta}, \mathbf{x}_1, \ldots, \mathbf{x}_n) \;:=\; \frac{\widehat{P}_{\text{UM}}(\boldsymbol{\theta}, Z|\mathbf{x}_1, \ldots, \mathbf{x}_n)}{\widehat{P}_{\text{UM}}(\boldsymbol{\theta}|\mathbf{x}_1, \ldots, \mathbf{x}_n)},$$

which simplifies as

$$\widetilde{Z}_{ij} \;:=\; Q(Z_{ij} = 1|\boldsymbol{\theta}, \mathbf{x}_1, \ldots, \mathbf{x}_n) \;=\; \begin{cases} \dfrac{\exp\left(-\frac{\|\mathbf{x}_i - A\mathbf{x}_j\|^2}{2\sigma^2}\right)}{\sum_{s \neq i} \exp\left(-\frac{\|\mathbf{x}_i - A\mathbf{x}_s\|^2}{2\sigma^2}\right)}, & i \neq j, \\ 0, & i = j. \end{cases} \tag{3.11}$$

In the M-step we maximize the expectation of the log complete posterior with respect to the posterior probability $Q(Z|\boldsymbol{\theta}, \mathbf{x}_1, \ldots, \mathbf{x}_n)$:

$$\max_{\boldsymbol{\theta}'} \sum_Z Q(Z|\boldsymbol{\theta}, \mathbf{x}_1, \ldots, \mathbf{x}_n) \log \widehat{P}_{\text{UM}}(\boldsymbol{\theta}', Z|\mathbf{x}_1, \ldots, \mathbf{x}_n) \iff$$

$$\max_{A, \sigma^2} -\sum_{i=1}^n \sum_{j=1}^n \widetilde{Z}_{ij} \left(\frac{\|\mathbf{x}_i - A\mathbf{x}_j\|^2}{2\sigma^2} + \frac{p}{2} \log(2\pi\sigma^2) \right) - \frac{\lambda}{2}\|A\|_F^2 - (\alpha + 1)\log\sigma^2 - \frac{\beta}{\sigma^2}, \tag{3.12}$$

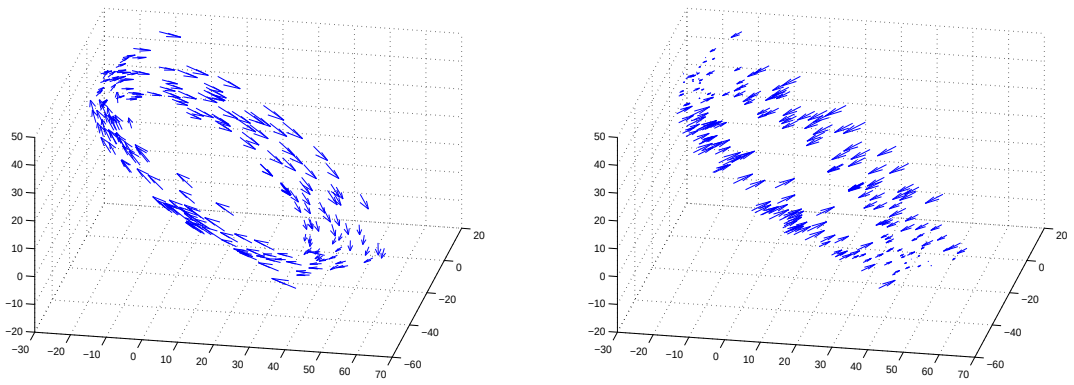
whose solution has a simple form:

$$A \;=\; \left(\sum_{i=1}^n \sum_{j=1}^n \widetilde{Z}_{ij} \mathbf{x}_i \mathbf{x}_j^\top \right) \left(\sum_{i=1}^n \sum_{j=1}^n \widetilde{Z}_{ij} \mathbf{x}_j \mathbf{x}_j^\top + \lambda\sigma^2 I \right)^{-1}, \tag{3.13}$$

$$\sigma^2 \;=\; \frac{\sum_{i=1}^n \sum_{j=1}^n \widetilde{Z}_{ij} \|\mathbf{x}_i - A\mathbf{x}_j\|^2 + 2\beta}{pn + 2(\alpha + 1)}, \tag{3.14}$$

A summary of the EM procedure is given in Algorithm 3.2. Note that (3.14) can be easily generalized to handle general covariance structures.

According to (3.11) and (3.12), we can view Algorithm 3.2 as a version of the iteratively re-weighted least square (IRLS) procedure. Although it is simple and often computationally efficient, one may worry that without enforcing any directional consistency in the Z_{ij}'s the EM



(a) True dynamics (b) Degenerate estimate

Figure 3.1: Degenerate estimate by the unordered approximation on 200 data points

algorithm may lead to degenerate dynamic models, especially when the sample size is limited. In fact, this happens in our experiments on simulated data. We thus propose a variant of (3.8) that incorporates additional constraints in the next section.

Although our focus is learning first-order models, it is worth noting that the proposed EM algorithm can be easily generalized to learn higher-order models. For example, consider the second-order model

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

Using approximations similar to those in (3.7), we may obtain the following unordered approximate likelihood:

$$\widehat{L}(\mathbf{x}_1, \ldots, \mathbf{x}_n|\boldsymbol{\theta}) \;:=\; \prod_{i=1}^{n} \left(\sum_{k \neq j \neq i} \frac{\exp(-\frac{\|\mathbf{x}_i - A\mathbf{x}_j - B\mathbf{x}_k\|^2}{2\sigma^2})}{(n-1)(2\pi\sigma^2)^{\frac{p}{2}}} \right).$$

The corresponding EM algorithm then involves a latent three-way tensor variable $Z \in \{0,1\}^{n \times n \times n}$: the E step computes posterior probabilities that $\mathbf{x}_i$ is generated from $\mathbf{x}_j$ and $\mathbf{x}_k$ for all triples $i \neq j \neq k$, and the M step solves weighted least square regression for A, B and σ^2.

3.2.2 Partially-ordered Approximation

As mentioned before, the replacement of the true state space density with the empirical density results in the approximate likelihood (3.7), where the data points are treated as if each one were actually generated from some other one. What might be more problematic is such an approximation ignores the fact that there is a latent temporal ordering induced by the unknown time indices of the data points, even though the data points are drawn independently. A possible consequence of ignoring the latent ordering is a degenerate estimate of the dynamic model, as illustrated in Figure 3.1, which shows the true one-step displacement vectors $A\mathbf{x}_i - \mathbf{x}_i$ and the UM estimates. On this data set, the UM likelihood of the true model is even worse than the degenerate estimate.

In our experiments on synthetic data (Section 3.4), we find that such degeneracy issues usually arise when the data is small, say a few hundred points. Thus in our second approach, we take into account the ordering relation explicitly. To do so, we introduce a set of weight parameters ω_{ij}'s, collectively denoted as an n-by-n matrix $\boldsymbol{\omega}$, and modify the approximate likelihood (3.7) as follows:

$$\widehat{L}_2(\mathbf{x}_1, \ldots, \mathbf{x}_n | \boldsymbol{\theta}, \boldsymbol{\omega}) \;:=\; \prod_{\substack{i=1,\\ i \notin S}}^{n} \sum_{j=1}^{n} \left(\frac{\exp(-\frac{\|\mathbf{x}_i - A\mathbf{x}_j\|^2}{2\sigma^2})}{(2\pi\sigma^2)^{\frac{p}{2}}} \omega_{ij} \right), \tag{3.15}$$

in which $S \;:=\; \{i : t_i \leq t_j \; \forall j\}$,

$$\begin{cases} \omega_{ij} \;\geq\; 0, & t_j < t_i, \\ \omega_{ij} \;=\; 0, & t_j \geq t_i, \\ \omega_{ii} \;=\; 0, & \forall\, i, \end{cases} \quad \text{and} \quad \sum_{j=1}^{n} \omega_{ij} = 1, \; \forall\, i \notin S. \tag{3.16}$$

The first set of constraints in (3.16) is to force the summation in (3.15) to be consistent with a global direction of time, while the normalization constraints are to maintain the notion of approximating the true state space density with an empirical density. The set S denotes the data points that are the earliest in time (hence cannot be generated from other data points), which can be viewed as rough estimates of the first state. If the underlying dynamic model exhibits a periodic behavior (such as rotation on a plane), the true first state may not be identifiable but A and σ^2 still may be. In that case, S is chosen arbitrarily and the relative time offsets between points may still be correct, thus leading to reasonable estimates of A and σ^2.

As mentioned before, the true time indices t_i's of the data points are missing, and even with the above approximation (3.15) and (3.16) it is still unclear how to jointly estimate them and the model parameters. We instead consider the ω_{ij}'s as unknown parameters to be estimated, which we interpret as decomposing the global ordering information into parameters of pairwise relations. Again, we make clear that as in Section 3.2.1, here we are also approximating the likelihood as if each point in the data were actually generated from some other point in the data. The set of constraints (3.16) can be restated only in terms of $\boldsymbol{\omega}$ as follows:

1. $\boldsymbol{\omega}$ is non-negative; each row of $\boldsymbol{\omega}$ sums to one or zero.
2. As a weighted adjacency matrix, $\boldsymbol{\omega}$ represents a *directed acyclic* graph.

Note that for both constraints to hold at the same time, one or more rows in $\boldsymbol{\omega}$ must sum to zero, and the corresponding data points form the set S. However, it is hard to maximize (3.15) with respect to ω under these constraints because they define a non-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 consider a subset of the previous two constraints:

1. $\boldsymbol{\omega}$ can only take values in $\{0, 1\}$.
2. As an adjacency matrix, $\boldsymbol{\omega}$ forms a *directed spanning tree*.

The new constraints turn the problem into a combinatorial one, which at first glance seems even more difficult. As we will show below, the fact that this discrete version is computationally tractable depends entirely on our restricting $\boldsymbol{\omega}$ to be a directed spanning tree. Under the new constraints, the set S has only one data point, which is the root of the directed spanning tree.

Algorithm 3.3 Alternating Maximization for (3.17)

Input: Data points $\mathbf{x}_1, \ldots, \mathbf{x}_n$.
Initialize $A_{(0)}$ and $\sigma^2_{(0)}$, set $k = 0$
repeat
Construct the weight matrix $W_{(k)}$ by (3.24) with $A_{(k)}$ and $\sigma^2_{(k)}$
$\boldsymbol{\omega}_{(k+1)} \leftarrow$OptimumBranch$(W_{(k)})$
Update $A_{(k+1)}$ by (3.22) with $\boldsymbol{\omega}_{(k+1)}$ and $\sigma^2_{(k)}$
Update $\sigma^2_{(k+1)}$ by (3.23) with $A_{(k+1)}$ and $\boldsymbol{\omega}_{(k+1)}$
$k \leftarrow k + 1$
until The approximate log posterior (3.17) does not increase

Combining these tree-based constraints with the approximate likelihood (3.15) and imposing the same priors on A and σ^2 as before, we propose maximizing the following approximate log posterior for estimation:

$$\max_{\substack{A,\sigma^2,\boldsymbol{\omega},\\ r\in\{1,\ldots,n\}}} \sum_{\substack{i=1,\\ i\neq r}}^{n} \log \sum_{j=1}^{n} \left(\frac{\exp(-\frac{\|\mathbf{x}_i - A\mathbf{x}_j\|^2}{2\sigma^2})}{(2\pi\sigma^2)^{\frac{p}{2}}} \omega_{ij} \right) - \frac{\lambda}{2}\|A\|_F^2 - (\alpha+1)\log\sigma^2 - \frac{\beta}{\sigma^2} \tag{3.17}$$

$$\text{s.t. } \omega_{ij} = \{0, 1\}, \tag{3.18}$$

$$\sum_{j=1}^{n} \omega_{ij} = 1,\ i \neq r,\ \sum_{j=1}^{n} \omega_{rj} = 0, \tag{3.19}$$

$$\boldsymbol{\omega} \text{ forms a tree with root } \mathbf{x}_r. \tag{3.20}$$

We refer to (3.17) as the Partially-ordered Approximation, and with the constraints (3.18) and (3.19) it can be simplified as follows:

$$\begin{aligned}
&\sum_{\substack{i=1,\\ i\neq r}}^{n} \log \sum_{j=1}^{n} \left(\frac{\exp(-\frac{\|\mathbf{x}_i - A\mathbf{x}_j\|^2}{2\sigma^2})}{(2\pi\sigma^2)^{\frac{p}{2}}} \omega_{ij} \right) - \frac{\lambda}{2}\|A\|_F^2 - (\alpha+1)\log\sigma^2 - \frac{\beta}{\sigma^2} \\
=&\sum_{i=1}^{n} \log \prod_{j=1}^{n} \left(\frac{\exp(-\frac{\|\mathbf{x}_i - A\mathbf{x}_j\|^2}{2\sigma^2})}{(2\pi\sigma^2)^{\frac{p}{2}}} \right)^{\omega_{ij}} - \frac{\lambda}{2}\|A\|_F^2 - (\alpha+1)\log\sigma^2 - \frac{\beta}{\sigma^2} \\
=&-\sum_{i=1}^{n}\sum_{j=1}^{n} \omega_{ij} \left(\frac{\|\mathbf{x}_i - A\mathbf{x}_j\|^2}{2\sigma^2} + \frac{p}{2}\log(2\pi\sigma^2) \right) - \frac{\lambda}{2}\|A\|_F^2 - (\alpha+1)\log\sigma^2 - \frac{\beta}{\sigma^2}.
\end{aligned} \tag{3.21}$$

Interestingly, this objective function is in the same form as the expected log complete posterior (3.12) in Section 3.2.1 with $\widetilde{Z}_{ij}$ being replaced by ω_{ij}. One may thus view $\boldsymbol{\omega}$ as the latent variable Z in Section 3.2.1 with additional constraints. However, there is a subtle difference: Z serves only as a means to derive the EM algorithm and does not appear in the maximization objective (3.7) of the Unordered Approximation, whereas $\boldsymbol{\omega}$ explicitly appears in the optimization problem (3.17) as an unknown parameter to be estimated.

Next we discuss how to maximize (3.17). Since (3.21) has the same form as (3.12), the optimal A and σ^2 under a fixed $\boldsymbol{\omega}$ have the same form as (3.13) and (3.14):

$$A = \left(\sum_{i=1}^{n}\sum_{j=1}^{n}\boldsymbol{\omega}_{ij}\mathbf{x}_i\mathbf{x}_j^\top\right)\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\boldsymbol{\omega}_{ij}\mathbf{x}_j\mathbf{x}_j^\top + \lambda\sigma^2 I\right)^{-1}, \tag{3.22}$$

$$\sigma^2 = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}\boldsymbol{\omega}_{ij}\|\mathbf{x}_i - A\mathbf{x}_j\|^2 + 2\beta}{p(n-1) + 2(\alpha+1)}. \tag{3.23}$$

When A and σ^2 are fixed, maximizing (3.21) with respect to $\boldsymbol{\omega}$ under (3.18), (3.19) and (3.20) is equivalent to finding the *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

$$W_{ij} := -\left(\frac{\|\mathbf{x}_i - A\mathbf{x}_j\|^2}{2\sigma^2} + \frac{p}{2}\log(2\pi\sigma^2)\right). \tag{3.24}$$

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 by Tarjan [1977] and provided a remedy retaining the same time complexity.

With these results, we present an alternate maximization procedure, Algorithm 3.3 for maximizing (3.17), where OptimumBranch$(\cdot)$ taking an edge-weight matrix as the input argument uses an implementation[1] of Tarjan [1977] and Camerini et al. [1979]. Since Algorithm 3.3 always increases the objective (3.21), it converges to a local maximum.

3.2.3 Expectation Maximization over Directed Spanning Trees

Recently in the Natural Language Processing community, researchers [Globerson et al., 2007; Smith and Smith, 2007] have developed sum-product inference algorithms for directed spanning trees, which make use of the matrix tree theorem [Tutte, 1984]. Based on their techniques, we develop an EM procedure whose E-step computes the the expectation of the latent variable Z over all directed spanning trees. Such a tree-based EM procedure can be viewed as being midway between the previous two methods, the first of which averages out entirely the latent sequential nature of the data, while the second aggressively selects the single most likely partial order.

Consider the set of spanning trees, $T(X)$, on the complete directed graph whose nodes are the sample points. We represent a spanning tree by its adjacency matrix, whose rows correspond to heads and columns correspond to tails. We also slightly abuse the notation Z to mean both a predecessor matrix and the corresponding set of edges, so $(i, j) \in Z$ means $Z_{ij} = 1$. We then

[1] Available at `http://edmonds-alg.sourceforge.net/`

maximize the following approximate log posterior:

$$\begin{aligned}
&\log \widehat{P}_{tree}(A, \sigma^2 | X) \qquad (3.25)\\
\propto\ &\log \left(\sum_{Z \in T(X)} \prod_{(i,j) \in Z} \exp\left(-\frac{\|\mathbf{x}_i - A\mathbf{x}_j\|^2}{2\sigma^2} \right) \right) - \frac{\lambda}{2}\|A\|_F^2 - \left(\frac{p(n-1)}{2} + (\alpha+1) \right) \log \sigma^2 - \frac{\beta}{\sigma^2},
\end{aligned}$$

where as before we place a zero-mean Gaussian prior on A with hyper-parameter λ and an inverse Gamma prior on σ^2 with hyper-parameters α and β. A major difference between (3.25) and the unordered approximate log posterior (3.8) is that the former sums over "global" latent structures, i.e., spanning trees, whereas the latter sums over "local" latent predecessor variables as shown in (3.10). We thus expect (3.25) to be more robust against undesirable local maxima than (3.8).

To derive an estimation procedure based on maximizing (3.25), we first denote the posterior distribution over $Z \in T(X)$ by

$$Q(Z|A, \sigma^2, X) \ := \ \frac{\mathbf{1}\{Z \in T(X)\} \prod_{(i,j) \in Z} \exp\left(-\frac{\|\mathbf{x}_i - A\mathbf{x}_j\|^2}{2\sigma^2} \right)}{\sum_{Z' \in T(X)} \prod_{(i,j) \in Z'} \exp\left(-\frac{\|\mathbf{x}_i - A\mathbf{x}_j\|^2}{2\sigma^2} \right)}. \qquad (3.26)$$

Then, by applying the standard variational equation we obtain the following lower bound:

$$\begin{aligned}
&\log \widehat{P}_{tree}(A, \sigma^2 | X)\\
\geq\ &\left(\sum_{Z \in T(X)} Q(Z|A, \sigma^2, X) \left(\log \prod_{(i,j) \in Z} \exp\left(-\frac{\|\mathbf{x}_i - A\mathbf{x}_j\|^2}{2\sigma^2} \right) \right) \right) - \frac{\lambda}{2}\|A\|_F^2 - \frac{\beta}{\sigma^2}\\
&- \left(\frac{p(n-1)}{2} + (\alpha+1) \right) \log \sigma^2\\
=\ &- \left(\sum_{i,j} \widetilde{Z}_{ij} \frac{\|\mathbf{x}_i - A\mathbf{x}_j\|^2}{2\sigma^2} \right) - \frac{\lambda}{2}\|A\|_F^2 - \frac{\beta}{\sigma^2} - \left(\frac{p(n-1)}{2} + (\alpha+1) \right) \log \sigma^2, \qquad (3.27)
\end{aligned}$$

where

$$\widetilde{Z}_{ij} \ := \ E_Q[Z] \ = \ \sum_{Z \in T(X)} \mathbf{1}\{(i,j) \in Z\} Q(Z|A', (\sigma')^2, X). \qquad (3.28)$$

The lower bound (3.27) holds for all choices of A' and $(\sigma')^2$ in the posterior mean (3.28), suggesting an EM procedure that alternates between computing $\widetilde{Z}_{ij}$ and maximizing (3.27) with respect to A and σ^2.

For the M-step, the lower bound (3.27), as a function of A and σ^2, is in the same form as the complete log posterior (3.12), leading to update rules similar to (3.13) and (3.14):

$$A \ = \ \left(\sum_{i=1}^n \sum_{j=1}^n \widetilde{Z}_{ij} \mathbf{x}_i \mathbf{x}_j^\top \right) \left(\sum_{i=1}^n \sum_{j=1}^n \widetilde{Z}_{ij} \mathbf{x}_j \mathbf{x}_j^\top + \lambda\sigma^2 I \right)^{-1}, \qquad (3.29)$$

$$\sigma^2 \ = \ \frac{\sum_{i=1}^n \sum_{j=1}^n \widetilde{Z}_{ij} \|\mathbf{x}_i - A\mathbf{x}_j\|^2 + 2\beta}{p(n-1) + 2(\alpha+1)}. \qquad (3.30)$$

Algorithm 3.4 Expectation Maximization for (3.25)

Input: Data points $\mathbf{x}_1, \ldots, \mathbf{x}_n$
Initialize $A_{(0)}$ and $\sigma^2_{(0)}$, set $k = 0$
repeat
  Update $\widetilde{Z}_{(k+1)}$ by (3.33) with $A_{(k)}$ and $\sigma^2_{(k)}$
  Update $A_{(k+1)}$ by (3.29) with $\widetilde{Z}_{(k+1)}$ and $\sigma^2_{(k)}$
  Update $\sigma^2_{(k+1)}$ by (3.30) with $A_{(k+1)}$ and $\widetilde{Z}_{(k+1)}$
  $k \leftarrow k + 1$
until The approximate log posterior (3.25) does not increase

For the E-step, we resort to the techniques in Sections 3.1 and 3.2 of [Globerson et al., 2007]. Let

$$W_{ij} \;:=\; \begin{cases} \exp\left(-\frac{\|\mathbf{x}_i - A\mathbf{x}_j\|^2}{2\sigma^2}\right), & i \neq j, \\ 0, & i = j. \end{cases} \tag{3.31}$$

denote the weight on the edge $\mathbf{x}_j$ to $\mathbf{x}_i$. Based on the Laplacian of the corresponding weighted directed graph, we define the following matrix:

$$\widetilde{L}_{ij} \;:=\; \begin{cases} r_i, & j = 1, \\ \sum_{j'=1}^{n} W_{ij'}, & i = j,\ j > 1, \\ -W_{ij}, & i \neq j,\ j > 1, \end{cases} \tag{3.32}$$

which replaces the first column of the Laplacian with a non-negative root selection score vector $\mathbf{r} \in \mathcal{R}^n$. The values in $\mathbf{r}$ reflect how likely each sample point $\mathbf{x}_i$ would be the root of a spanning tree. When prior knowledge is unavailable, we simply set $r_i = 1,\ i = 1, \ldots, n$. Then, we compute $\widetilde{Z}_{ij}$ by

$$\widetilde{Z}_{ij} \;=\; (1 - \mathbf{1}\{1 = i\})W_{ij}(\widetilde{L}^{-1})_{ii} - (1 - \mathbf{1}\{j = 1\})W_{ij}(\widetilde{L}^{-1})_{ji}. \tag{3.33}$$

We determine convergence of the algorithm by checking the value of the log posterior (3.25), which is computed by

$$\log \widehat{P}_{tree}(A, \sigma^2 | X) \;\propto\; \log|\widetilde{L}| - \frac{\lambda}{2}\|A\|_F^2 - \left(\frac{p(n-1)}{2} + (\alpha + 1)\right)\log\sigma^2 - \frac{\beta}{\sigma^2}.$$

A summary of the EM algorithm is in Algorithm 3.4.

3.2.4 Nonlinear Extension via Kernel Regression

To learn nonlinear dynamic models, we extend the aforementioned three methods through the use of kernel regression. We consider nonlinear dynamic models of the following form:

$$\mathbf{x}^{(t+1)} = B\phi(\mathbf{x}^{(t)}) + \boldsymbol{\epsilon}^{(t)},\ \boldsymbol{\epsilon}^{(t)} \sim \mathcal{N}(\cdot|\mathbf{0}, \sigma^2 I). \tag{3.34}$$

where $\phi(\cdot)$ maps a point in $\mathbb{R}^p$ 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 B is a linear mapping from the RKHS to $\mathbb{R}^p$. Replacing $A\mathbf{x}_j$ in (3.8), (3.17) and (3.25) by $B\phi(\mathbf{x}_j)$ then leads to nonlinear extensions of the three approximate log posteriors.

Next we extend Algorithms 3.2, 3.3 and 3.4 for learning B and σ^2. For the E-steps, we only need to replace $A\mathbf{x}_j$ in (3.11), (3.24) and (3.31) by $B\phi(\mathbf{x}_j)$. For the M-steps, we solve the weighted least squares problems (3.12), (3.21) and (3.27) with $A\mathbf{x}_j$ replaced by $B\phi(\mathbf{x}_j)$. The resulting three update rules for B and σ^2 are very similar, so for brevity here we only give the one for maximizing the unordered approximate posterior:

$$\begin{aligned} B &= \left(\sum_{i=1}^n \sum_{j=1}^n \widetilde{Z}_{ij} \mathbf{x}_i \phi(\mathbf{x}_j)^\top\right) \left(\sum_{i=1}^n \sum_{j=1}^n \widetilde{Z}_{ij} \phi(\mathbf{x}_j)\phi(\mathbf{x}_j)^\top + \lambda\sigma^2 I\right)^{-1} \\ &= X\widetilde{Z}\phi(X)^\top \left(\phi(X)\Lambda_{\widetilde{Z}}\phi(X)^\top + \lambda\sigma^2 I\right)^{-1} \qquad (3.35) \\ &= X\widetilde{Z}\left(K\Lambda_{\widetilde{Z}} + \lambda\sigma^2 I\right)^{-1}\phi(X)^\top, \qquad (3.36) \\ \sigma^2 &= \frac{\sum_{i=1}^n \sum_{j=1}^n \widetilde{Z}_{ij}\|\mathbf{x}_i - B\phi(\mathbf{x}_j)\|^2 + 2\beta}{pn + 2(\alpha+1)}, \qquad (3.37) \end{aligned}$$

where $X := [\mathbf{x}_1 \ \cdots \ \mathbf{x}_n]$ collects the data into a p-by-n matrix, $\phi(X) := [\phi(\mathbf{x}_1) \ \cdots \ \phi(\mathbf{x}_n)]$ is the RKHS mapping of the entire data set, $\Lambda_{\widetilde{Z}}$ is a diagonal matrix with $(\Lambda_{\widetilde{Z}})_{ii} := \sum_{j=1}^n \widetilde{Z}_{ji}$, and $K := \phi(X)^\top\phi(X)$ is the kernel matrix. We obtain (3.36) from (3.35) by using the Matrix Inversion lemma.

One issue with the above extensions is that we cannot compute B when the mapping $\phi(\cdot)$ is of infinite dimension. However, we observe that the EM procedures only make use of $B\phi(X)$, and according to (3.36)

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

Therefore, instead of B we maintain and update a p-by-n matrix $M := X\widetilde{Z}\left(K\Lambda_{\widetilde{Z}} + \lambda\sigma^2\right)^{-1}$ in the EM iterations. To predict the next state for a new observation $\mathbf{x}$, we compute $M\phi(X)^\top\phi(\mathbf{x})$, 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 $K \approx \widetilde{\phi}(X)^\top\widetilde{\phi}(X)$, and replace $\phi(X)$ in the EM procedure with $\widetilde{\phi}(X) \in \mathbb{R}^{m\times n}, m < n$. This can be viewed as dimensionality reduction in the RKHS. Then we can maintain and update $B \in \mathbb{R}^{p\times m}$ explicitly. 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 $\widetilde{\phi}$, and then apply the estimated B to the mapped points.

3.3 Initialization of EM by Temporal Smoothing

All of the proposed methods are solving non-convex optimization problems, and avoiding local optima is a critical issue. A common practice in applying EM methods is to run the algorithm

multiple times, each with a randomly initialized model, and then choose the best local optimum as the final estimate. We follow this practice in our experiments on simulated data in Section 3.4, but observe that the number of random restarts needed to obtain a good model is usually large, meaning that a lot of random initializations lead to undesirable local optima. Moreover, our simulated data are low dimensional, but the problem caused by local optima will only become worse in a higher dimension, which is common with real data. We thus investigate an alternative way of initialization.

We begin by observing that in the case of a linear dynamic model, the samples generated by Algorithm 3.1 can be viewed as i.i.d. samples drawn from the following mixture of Gaussians:

$$\mathbf{x} \sim \sum_{t=1}^{T_{\max}} \pi^{(t)} \mathcal{N}(\cdot|\boldsymbol{\mu}^{(t)}, \Sigma^{(t)}), \tag{3.38}$$

where $\boldsymbol{\mu}^{(t)}$ and $\Sigma^{(t)}$ are defined in (3.3) and $\pi^{(t)} \geq 0$ is the probability that $\mathbf{x}$ is drawn at time t. Based on this view, we devise a heuristic to initialize the model:

1. Estimate $\boldsymbol{\mu}^{(t)}$'s by fitting a GMM to or clustering the data
2. Estimate the true temporal order of $\boldsymbol{\mu}^{(t)}$'s based on their estimates from Step 1
3. Learn a dynamic model from the estimated sequence of $\boldsymbol{\mu}^{(t)}$'s by existing dynamic model learning methods

For Step 1 we can use the standard EM algorithm for learning GMMs, or simply the k-means algorithm since subsequent steps only need estimates of the means. Step 2 in its own right is a challenging problem. If we believe temporally close $\boldsymbol{\mu}^{(t)}$'s should be similar, we can compute pairwise distances between estimates of $\boldsymbol{\mu}^{(t)}$'s and solve a traveling salesman problem (TSP). Then we need to decide the direction of time on the TSP path, which is often impossible without prior or expert knowledge. In our experiments in Sections 3.5 and 3.5.2 we simply try both directions and report the one that performs better. In high dimensions, Euclidean distances suffer from the curse of dimensionality and are vulnerable to noise. We thus propose an alternative way to recover the true temporal order, which is based on the idea of temporal smoothing.

Unlike methods proposed in previous sections, the method we present in the following does not make any assumptions about the functional form of the underlying dynamic model. It only assumes the underlying dynamics to be *smooth*, i.e., the curvature of the trajectory rolled out by the dynamic model is small. More precisely, we quantify smoothness by the second order differences of temporally adjacent points generated by the dynamic model:

$$S = \sum_{t=2}^{T_{\max}-1} \|(\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}) - (\mathbf{x}^{(t)} - \mathbf{x}^{(t-1)})\|^2, \tag{3.39}$$

where $T_{\max}$ is the maximum time. Small values of S correspond to smooth trajectories. An example is in Figure 3.2. Such a smoothness measure has been used as the regularization term in the Hodrick-Prescott filter [Hodrick and Prescott, 1997; Lesser, 1961], a common tool in macroeconomics for obtaining a smooth and nonlinear representation of a time series.

The quantity (3.39) cannot be computed on our data since the true time indices of the data points are missing. Nevertheless, it can be succinctly expressed using the Laplacian L of the

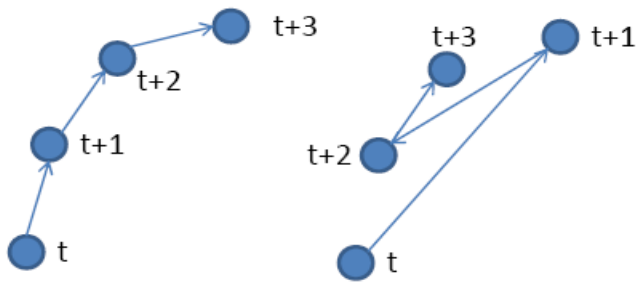


Figure 3.2: An example of smooth (left) v.s. non-smooth (right) trajectory

temporal adjacency graph obtained by connecting temporally adjacent pairs of data points. More specifically, we let $X := [\mathbf{x}_1 \ \cdots \ \mathbf{x}_n]$ be the p-by-n data matrix as before, and Z be a directed *temporal* adjacency matrix such that $Z_{ij} = 1$ if $\mathbf{x}_j$ precedes $\mathbf{x}_i$ immediately in time, and 0 otherwise. Then, we define $\overline{Z} := Z + Z^\top$ to represent the undirected, symmetric temporal adjacency of the data points. If the data points were sorted according to their true temporal order, the matrix $\overline{Z}$ would consist of ones in the upper-first and lower-first off-diagonals and zeros elsewhere. The graph Laplacian based on the adjacency matrix $\overline{Z}$ is then $L = \mathsf{diag}(\overline{Z}\mathbf{1}) - \overline{Z}$, where $\mathbf{1}$ is a vector of ones and $\mathsf{diag}(\overline{Z}\mathbf{1})$ denotes the diagonal matrix with the vector $\overline{Z}\mathbf{1}$ in the main diagonal. Simple algebraic manipulation shows that the smoothness S can be expressed in terms of L (hence Z) as follows:

$$S(Z) \ = \ \|XL\|_F^2 \ = \ \mathrm{Tr}((\mathsf{diag}(\overline{Z}\mathbf{1}) - \overline{Z})^\top X^\top X(\mathsf{diag}(\overline{Z}\mathbf{1}) - \overline{Z})), \tag{3.40}$$

which is a quadratic and convex function of $\overline{Z}$ and hence Z. Since we assume the true dynamics to be smooth, a natural way to reconstruct a temporal ordering would be to solve the following problem:

$$\begin{aligned} Z^* = \arg\min_{Z} \ & S(Z) \\ \text{s.t. } & Z \text{ represents a directed Hamiltonian path through the data points.} \end{aligned} \tag{3.41}$$

However, this problem is essentially a quadratic version of TSP, and to the best of our knowledge, no efficient solver exists for such problems. We thus consider the following two-step heuristics. In the first step, we minimize $S(Z)$ under a modified set of constraints:

$$\begin{aligned} \widehat{Z} \ &= \ \arg\min_{Z} \ \ S(Z) \\ \text{s.t. } \ & Z\mathbf{1} = \mathbf{1}, \quad Z^\top\mathbf{1} = \mathbf{1}, \quad Z_{ij} \geq 0, \quad Z_{ii} = 0. \end{aligned} \tag{3.42}$$

The constraints in (3.42) are not a proper relaxation of (3.41) because Z must have one zero row and one zero column to represent a Hamiltonian path. Nevertheless, we can interpret solving (3.42) as learning a pairwise similarity $\widehat{Z}$ whose (i,j)-th entry reflects how likely $\mathbf{x}_j$ is to precede $\mathbf{x}_i$ temporally. Then in the second step, we solve an instance of TSP with $1 - (\widehat{Z} + \widehat{Z}^\top)/2$ as the distance, and obtain an ordering from the optimal TSP path. In our experiments we use the state-of-the art *Concorde TSP solver* [Applegate et al.], which implements an exact algorithm

Algorithm 3.5 Projected Gradient Method for (3.42)

Input: Data matrix $X = [\mathbf{x}_1 \ \cdots \ \mathbf{x}_n]$

Output: $\widehat{Z}$

Set $\alpha = 0.1$, $\epsilon = 10^{-6}$, $\sigma = 10^{-2}$
Initialize $Z_{(1)}$, set $k = 1$
repeat
  Compute the gradient $\nabla_{(k)} := \nabla S(Z_{(k)})$, $\eta \leftarrow 1.0$
  repeat
    $Z = Z_{(k)} - \eta \nabla_{(k)}$, $D^{row} = D^{col} = [0]_{n \times n}$
    repeat
      $\widetilde{Z} \leftarrow Z$
      $Z'_{i\cdot} \leftarrow \ell_1$-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 $\|Z - \widetilde{Z}\|_F \leq \epsilon$
    $\eta \leftarrow \alpha \eta$
  until $S(Z) - S(Z_{(k)}) \leq \sigma \mathrm{Tr}\Big(\nabla_{(k)}(Z - Z_{(k)})\Big)$
  $t \leftarrow t + 1, \ Z_{(k)} \leftarrow Z$
until $\|Z_{(k)} - Z_{(k-1)}\|_F \leq \epsilon$
$\widehat{Z} \leftarrow Z_{(k)}$

that has exponential time-complexity in the worst case but is very efficient in practice due to its carefully designed pruning techniques.

The optimization problem (3.42) is essentially convex quadratic programming (QP) 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:

$$Z^{t+1} \leftarrow \Pi_\Omega(Z^t - \eta \nabla S(Z^t)), \tag{3.43}$$

$$\Omega = \{Z_{i\cdot}\mathbf{1} = 1, Z_{\cdot,j}^\top \mathbf{1} = 1, Z_{ij} \geq 0, Z_{ii} = 0\}, \tag{3.44}$$

where $Z_{i\cdot}$ and $Z_{\cdot j}$ denote a row and a column of Z respectively, and $\Pi_\Omega(\mathbf{a}) := \arg\min_{\mathbf{b}}\{\|\mathbf{a} - \mathbf{b}\| \mid \mathbf{b} \in \Omega\}$ is the Euclidean projection of a vector a onto a region Ω. The gradient of $S(Z)$ is given by

$$\nabla S(Z) \ = \ 2(\widetilde{Q} + \widetilde{Q}^\top),$$

Algorithm 3.6 ℓ_1-Projection [Duchi et al., 2008]

Input: $\mathbf{v} \in \mathbb{R}^n$

Output: $\mathbf{w} := \arg\min \|\mathbf{x} - \mathbf{v}\|_2$ s.t. $\mathbf{x}^\top \mathbf{1} = 1,\ x_i \geq 0\ \forall i$.

1: Sort $\mathbf{v}$ into $\boldsymbol{\mu} : \mu_1 \geq \mu_2 \geq \ldots \geq \mu_n$.
2: Find $\rho = \max\{j \in \{1, 2, \ldots, n\} : \mu_j - \frac{1}{j}(\sum_{r=1}^{j} \mu_r - 1) > 0\}$
3: Define $\theta = \frac{1}{\rho}(\sum_{i=1}^{\rho} \mu_i - 1)$
4: Output $\mathbf{w}$ s.t. $w_i = \max\{v_i - \theta, 0\}$

where

$$\begin{aligned} \widetilde{Q}_{ij} &:= Q_{jj} - Q_{ij}, \\ Q &:= X^\top X(\mathsf{diag}((Z + Z^\top)\mathbf{1}) - (Z + Z^\top)). \end{aligned}$$

Moreover, the feasible region (3.44) is the intersection of two closed convex sets Ω_1 and Ω_2:

$$\begin{aligned} \Omega_1 &= \{Z_{i\cdot}\mathbf{1} = 1,\ Z_{ij} \geq 0, Z_{ii} = 0, 1 \leq i, j \leq n\}, \\ \Omega_2 &= \{Z_{\cdot j}^\top \mathbf{1} = 1,\ Z_{ij} \geq 0, Z_{ii} = 0, 1 \leq i, j \leq n\}, \end{aligned}$$

which correspond to the normalization constraints for rows and columns, respectively. Using Dykstra's cyclic projection algorithm [Boyle and Dykstra, 1986], we perform the projection operation (3.43) 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 Duchi et al. [2008] proposed, which we outline in Algorithm 3.6. The required operations are simply sorting and thresholding[2]. Algorithm 3.5 gives a summary of the projected gradient method for the optimization problem (3.42). As in all gradient-based methods, we conduct back-tracking line search for the step size η to ensure convergence.

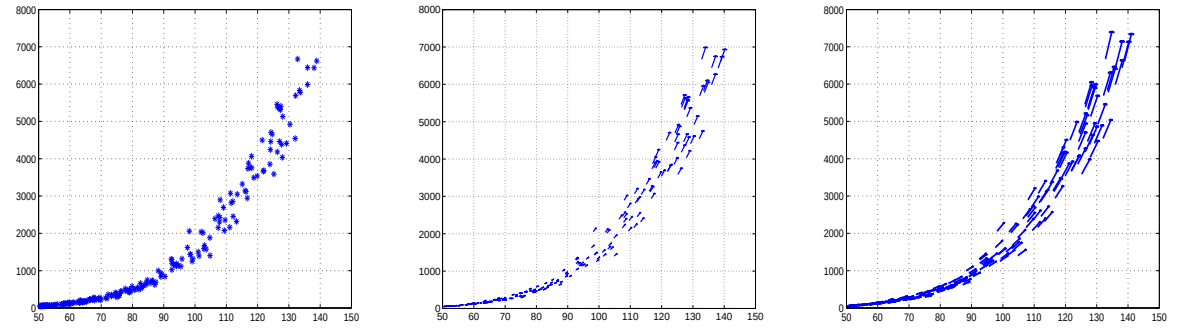
3.4 Experiments on Synthetic Data

We consider five dynamical systems. The first three are linear systems, while the last two are nonlinear systems. Our experiments here focus on the unordered approximation (Section 3.2.1), the partially-ordered approximation (Section 3.2.2), and their kernelized versions (Section 3.2.4), referred to as UM, PM, KUM and KPM, respectively. For our experiments here and in Section 3.5, we implement the proposed algorithms in MATLAB and use the maximum directed spanning tree solver available at `http://edmonds-alg.sourceforge.net/`, version 1.1.0.

3.4.1 Linear Systems

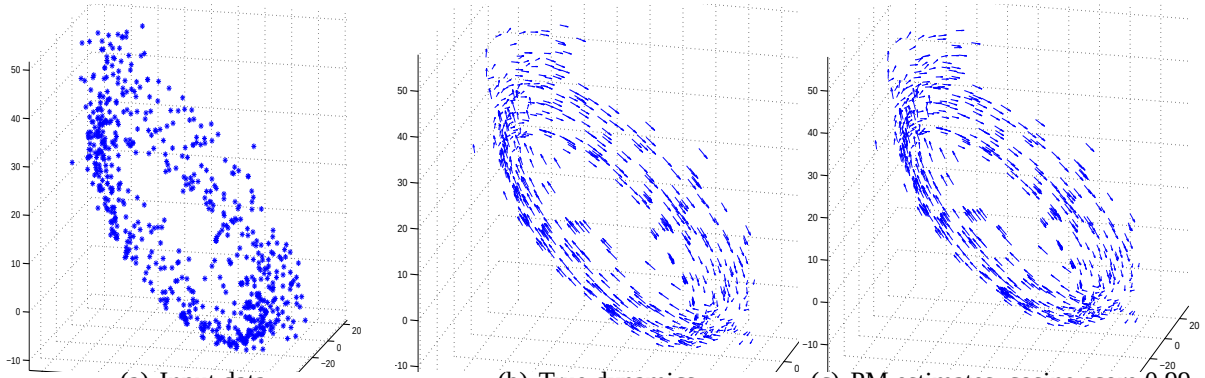
We consider the following three linear systems.

[2]For the ease of presentation, in Algorithm 3.6 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.



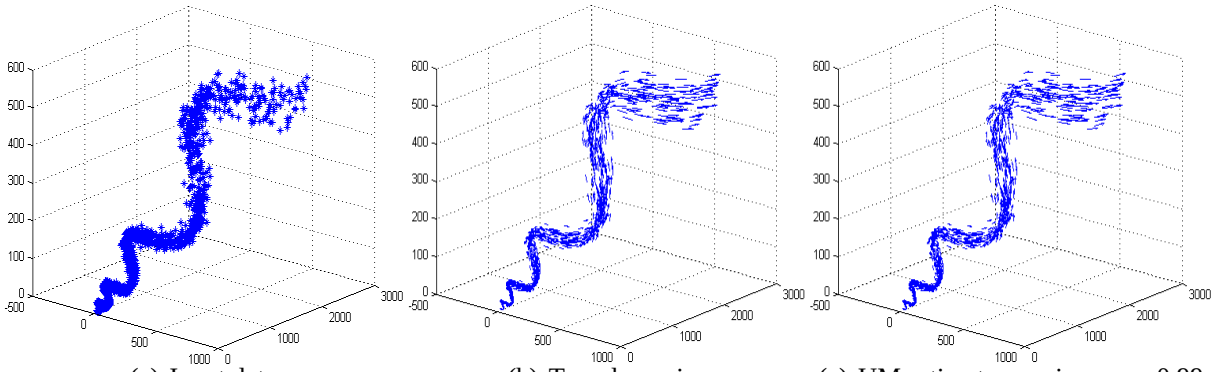
(a) Input data (b) True dynamics (c) UM estimates

Figure 3.3: 2D sample points



(a) Input data (b) True dynamics (c) PM estimates, cosine score 0.99

Figure 3.4: 3D-1 sample points



(a) Input data (b) True dynamics (c) UM estimates, cosine score 0.99

Figure 3.5: 3D-2 sample points

- **2D**, a two-dimensional diverging system:

$$A = \begin{bmatrix} 1.01 & 0 \\ 0 & 1.05 \end{bmatrix}, \ \mathbf{x}^0 = \begin{bmatrix} 50 \\ 50 \end{bmatrix}.$$

  A sample of 200 points generated by Algorithm 3.1 is shown in Figure 3.3(a).

- **3D-1**, a three-dimensional diverging system:

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

  The Eigenvalues of the transition matrix are 1.0143 and 0.9739 $\pm$ 0.24 i, so the system dynamics behaves like a diverging spiral in the 3-d space. A sample generated by Algorithm 3.1 is shown in Figure 3.4(a), suggesting that temporally close points (those along the spiral) can be spatially further away from each other than temporally remote points (those cutting across the spiral).

- **3D-2**, another three-dimensional diverging system:

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

  The Eigenvalues of the transition matrix are 1.0439 and 0.9982 $\pm$ 0.267 i. A sample is shown in Figure 3.5(a). Unlike in 3D-1, here temporal and spatial proximity are more consistent with each other.

While the results presented here are all on diverging systems, we also experimented with converging systems and got similar results. Using Algorithm 3.1, we generated data 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. In all of the data sets we set $T_{\max} = 100$.

We applied UM and PM to these data sets, maximizing approximate likelihood functions without any prior or regularization on the parameters of interest, i.e., setting the hyper-parameters α, β, and λ in (3.8) and (3.21) to zero. For every data set we ran Algorithms 3.2 and 3.3 each with M random initializations, and chose the model with the largest approximate likelihood 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.

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 usual least-square estimation technique. The fitted system itself is already an estimate, and can be used to initialize Algorithms 3.2 and 3.3. 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. 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.

We consider two performance metrics. The first compares the estimated system matrix $\hat{A}$ and the true A. To account for the ambiguities described in Section 3.1, we use the following rate-adjusted matrix error:

$$\text{ME}(A, \hat{A}) \equiv \min_t \|A - \hat{A}^t\|_F, \tag{3.45}$$

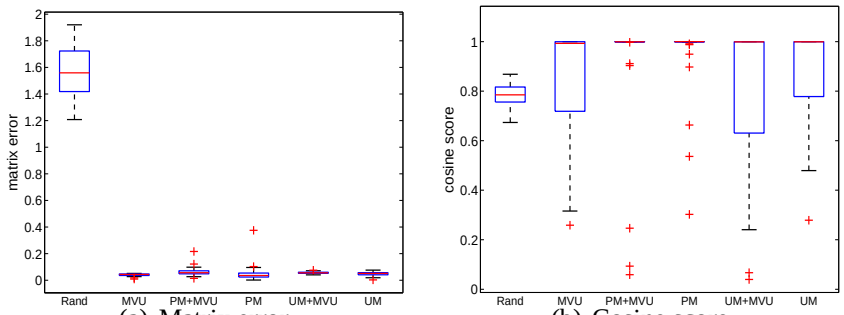
where $\hat{A}^t$ is $\hat{A}$ raised to the power t. The minimum in (3.45) is hard to solve, so we search for t in $\{\pm 1, \pm 2, \ldots, \pm 10, \pm 1/2, \pm 1/3, \ldots, \pm 1/10\}$ and choose the one that minimizes (3.45). Such a metric may overstate the quality of an estimate. We thus consider another criterion that compares system matrices based on one-step displacement vectors

$$\text{CS}(A, \hat{A}) \equiv \frac{1}{n} \left| \sum_{i=1}^{n} \frac{(A\mathbf{x}_i - \mathbf{x}_i)^\top (\hat{A}\mathbf{x}_i - \mathbf{x}_i)}{\|A\mathbf{x}_i - \mathbf{x}_i\| \|\hat{A}\mathbf{x}_i - \mathbf{x}_i\|} \right|, \tag{3.46}$$

which we refer to as the cosine score. This criterion measures the similarity between the one-step displacement vector $A\mathbf{x}_i - \mathbf{x}_i$ of the true system and that of the estimated system, averaging over all the sample points; a higher score (3.46) 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 (3.46) takes the absolute value after averaging, going forward and backward in time are considered equally good as long as they do so consistently.

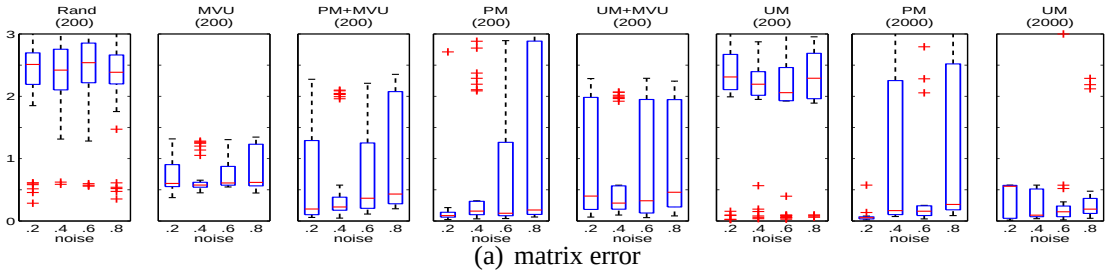
We tested the following methods: MVU, PM+MVU (PM initialized by MVU), PM, UM+MVU (UM initialized by MVU), UM, and Rand. Results on 2D are in Figure 3.6. For this baseline system, every approach performs quite well. Figure 3.3(c) shows displacement vectors $\hat{A}\mathbf{x}_i - \mathbf{x}_i$ estimated by UM in one of the 2D samples, which are quite consistent with the true dynamics.

Performance metrics on the more complex systems 3D-1 and 3D-2 are in Figures 3.7 and 3.8, respectively. To qualitatively demonstrate the performance, we also plot the one-step displacement vectors by the true and the learnt dynamic models in Figures 3.4 and 3.5 for 3D-1 and 3D-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.

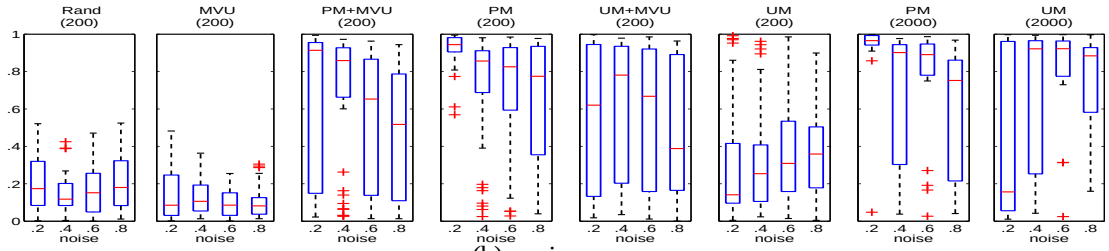


(a) Matrix error (b) Cosine score

Figure 3.6: Results on 2D, $\sigma = 0.2$

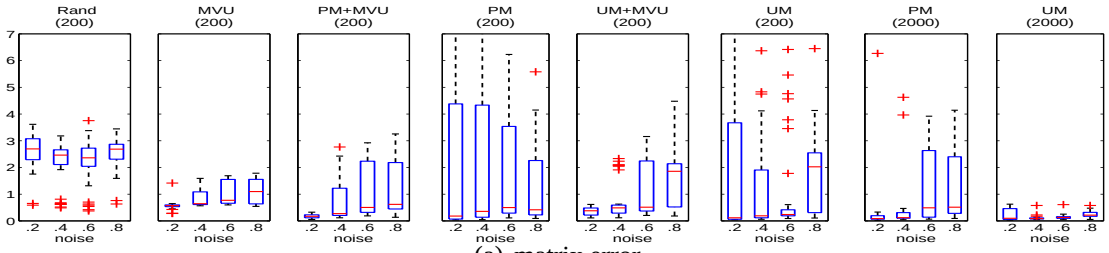


(a) matrix error

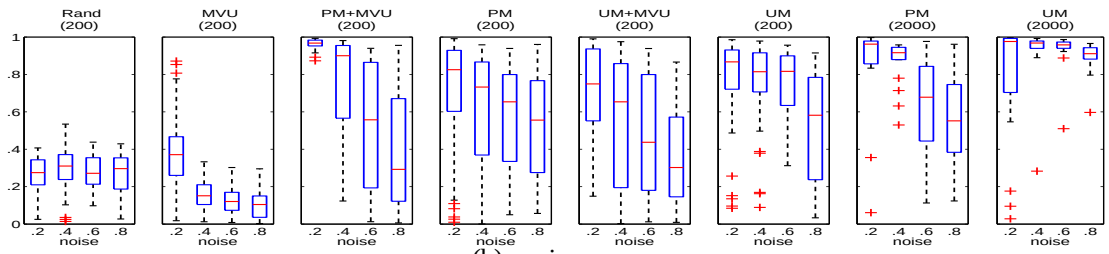


(b) cosine score

Figure 3.7: Results on 3D-1



(a) matrix error



(b) cosine score

Figure 3.8: Results on 3D-2

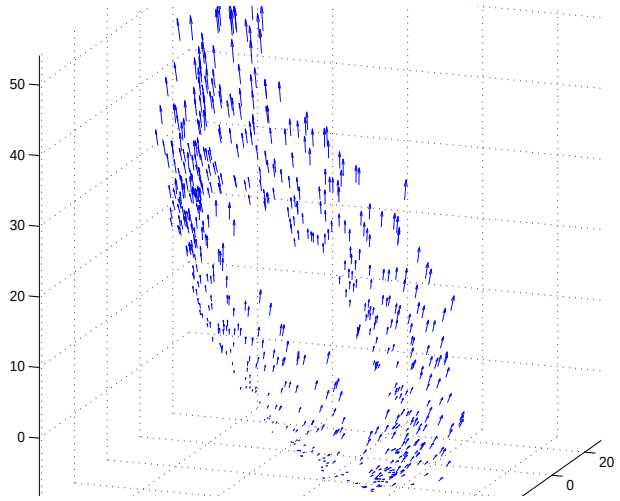


Figure 3.9: A 2000-points sample with small noise $\sigma = 0.2$ on which UM failed (cosine score 0.0108). For better visualization, only 1/3 of the points are plotted.

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 3.9. This is a concrete example of the identifiability issue pointed out in Section 3.1.

3.4.2 Nonlinear Systems

We consider the following two systems.

- **3D-conv**, a converging three-dimensional nonlinear system considered by Girard and Pappas [2005], governed by the following differential equations:

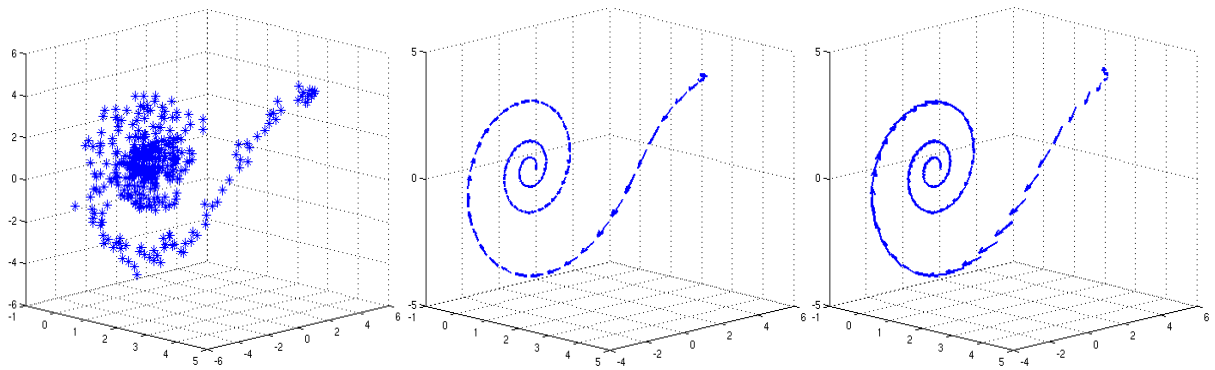
$$\begin{aligned} dx(t)/dt &= -(1+0.1y(t)^2)x(t), \\ dy(t)/dt &= -(1-0.1x(t)^2)y(t)/2 + 2z(t), \\ dz(t)/dt &= -(1-0.1x(t))2y(t) - z(t)/2, \end{aligned} \tag{3.47}$$

  where $x(t), y(t)$, and $z(t)$ are the three states at time t. The initial point is set to $[5\ 1\ 5]^\top$.

- **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 $[0\ 1\ 1.05]$. Figure 3.11(a) shows a trajectory of 800 points evenly sampled in the time interval $[0, 20]$.



(a) Input data (b) True dynamics (c) KUM estimates, cosine score 0.98

Figure 3.10: 3D-conv sample points

We generate data from these two systems as follows. For 3D-conv we use Algorithm 3.1 with line 5 replaced by a discrete-time approximation of the system equations (3.47), where the derivatives remain constant during a time step of $\Delta t = 0.1$:

$$\mathbf{x}^{(t+1)} := \mathbf{x}^{(t)} + 0.1\frac{d\mathbf{x}^{(t)}}{dt} + \boldsymbol{\epsilon}^{(t)}. \tag{3.48}$$

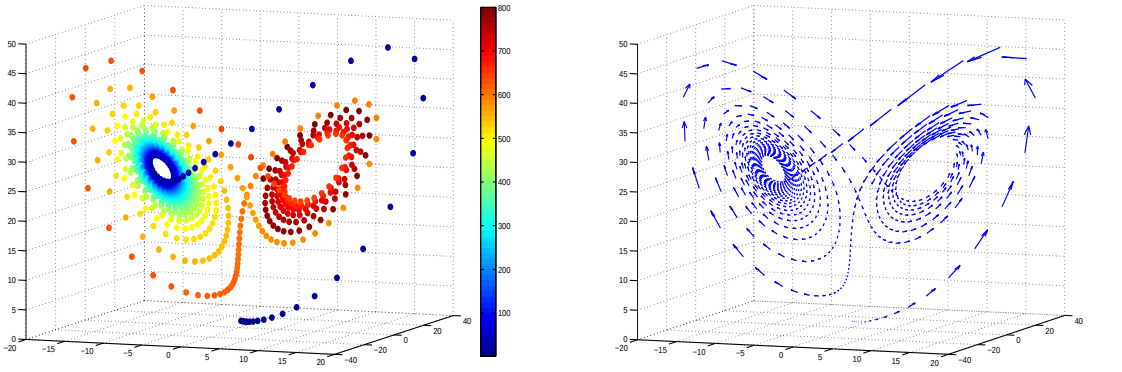
The process noise $\boldsymbol{\epsilon}^{(t)}$ follows a zero-mean Gaussian with standard deviations $\{0.1\Delta t, 0.5\Delta t\}$. We generate 20 training data sets of 400 points with $T_{\max} = 100$. Figure 3.10(a) shows one of the data sets. For Lorenz Attractor we did not use Algorithm 3.1 due to the chaotic nature of the system. Instead, we add independent Gaussian noise to a system trajectory of 400 points evenly sampled in the time interval $[10, 20]$[3] with noise standard deviation $\sigma_{noise} \in \{0.01\delta, 0.05\delta\}$, where δ is the median of all the pairwise distances of the 800 points shown in Figure 3.11(a). For each noise level we generate 20 training data sets without the true temporal order.

Our evaluation scheme here is slightly different from Section 3.4.1. Because our nonlinear methods give nonparametric estimates and the true models are described by differential equations, checking the model estimation error is difficult and not considered. We focus on evaluating the prediction performance in terms of the cosine score. For 3D-conv we evenly sampled 200 points in the time interval $[0, 10]$ as the testing sequence, shown in Figure 3.10(b) along with the true dynamics represented by vectors of displacement between consecutive points. For Lorenz Attractor we use the noise-free trajectory of 400 points as the testing sequence. Given a dynamic model learnt from the training data, we predict for each data point $\mathbf{x}^{(t)}$ in the testing sequence its next observation $\widehat{\mathbf{x}}^{(t+1)}$ and compute the testing cosine score:

$$\frac{1}{T-1}\left|\sum_{t=1}^{T-1}\frac{(\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)})^\top(\widehat{\mathbf{x}}^{(t+1)} - \mathbf{x}^{(t)})}{\|\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\|\|\widehat{\mathbf{x}}^{(t+1)} - \mathbf{x}^{(t)}\|}\right|, \tag{3.49}$$

where T is the length of the testing sequence.

[3]This trajectory is the second half of the trajectory in Figure 3.11(a). It preserves the butterfly shape but leaves out the highly dense spirals in the core of the left wing.



(a) System trajectory colored by time (b) Estimated dynamics by KUM, cosine score 0.98

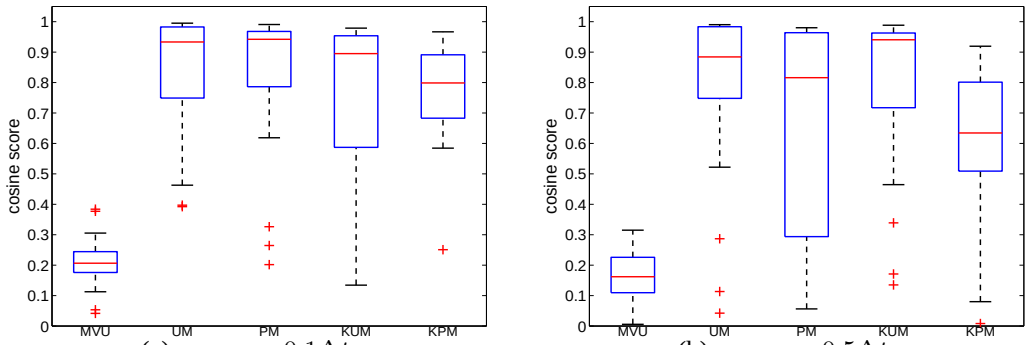
Figure 3.11: Lorenz Attractor sample trajectory

Table 3.3: Cosine scores on noise-free data

	MVU	UM	PM	KUM	KPM
3D-conv	**0.9954**	0.9903	0.5570	0.9909	0.9225
Lorenz	0.1383	0.5644	0.2155	**0.9884**	0.334

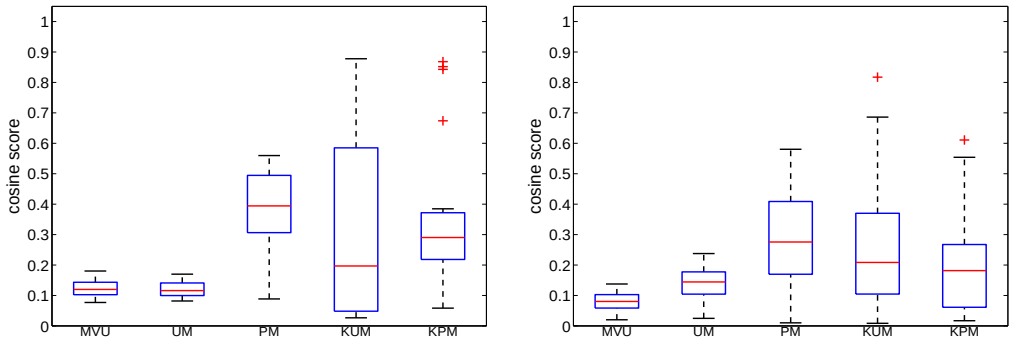
We compare the two nonlinear versions of approximate likelihood methods, KUM and KPM, against the Maximum Variance Unfolding [Weinberger et al., 2004] based approach described in Section 3.4.1 combined with kernel regression. We also include results by the linear methods UM and PM. Hyper-parameter settings are as follows. We use kernel regression with the Gaussian kernel $\exp(-(\|\mathbf{x}-\mathbf{y}\|^2)/(2h))$. For KUM and KPM, we set the kernel bandwidth h to $50\tilde{\delta}$, where $\tilde{\delta}$ is the median of all pairwise distances in a training data set. For MVU we set $h = \tilde{\delta}$. Because plain kernel regression tends to over-fit the training data, we regularize the model in two ways. First we make use of the standard penalty in ridge regression, setting the regularization parameter λ in (3.36) to 10^{-4} and 10^{-3} for the two noise levels for each nonlinear system. Secondly, when applying KUM and KPM, we use the low-rank approximation to the kernel matrix described in Section 3.2.4 with rank $m = 5$, reducing the model complexity. For each data set we run KUM and KPM with 50 random initializations of the regression coefficient matrix B and σ^2. Entries of B are drawn independently from a zero-mean Gaussian with standard deviation 100, and σ^2 is drawn uniformly random between 0 and 100 times of the median of pairwise distances.

Results are in Table 3.3, Figures 3.12 and 3.13. Table 3.3 reports cosine scores obtained by training (without temporal order) and predicting on the noise-free trajectories for both systems (Figures 3.10(a) and 3.11(a)), bold-facing the best method for each system. For 3D-conv all methods perform quite well except PM. Interestingly, UM performs very well even if it learns a linear model, suggesting 3D-conv may be well approximated by a linear system in terms of one-step predictions. For Lorenz Attractor, only KUM does well and other methods are significantly worse. Figure 3.11(b) shows the estimated dynamics by KUM, which are very close to the true dynamics. However, it takes hundreds of random initializations of KUM to obtain such a high cosine score, and many of the initializations led to degenerate or undesirable models.



(a) $\sigma_{noise} = 0.1\Delta t$ (b) $\sigma_{noise} = 0.5\Delta t$

Figure 3.12: Results on 3D-conv



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

Figure 3.13: Results on Lorenz Attractor

Figure 3.12 gives boxplots of cosine scores for 3D-conv obtained by training on the 20 noisy data sets and predicting on the noise-free testing sequence. The proposed methods outperform the MVU based approach significantly, but performances have a large variance across different training data sets. Linear and nonlinear methods achieve comparable scores, showing again that 3D-conv may be well approximated by a linear model in a short time period.

Figure 3.13 shows cosine scores on Lorenz Attractor. MVU still performs poorly, but the proposed methods all perform worse than in 3D-conv, especially UM. This is not surprising because Lorenz Attractor has more complex dynamics than 3D-conv. We also see that although the median score of PM across all the 20 training data sets is better than those of KUM and KPM, the nonlinear methods are able to reach a much higher score than PM. This indicates that the nonlinear methods are on the one hand more powerful than the linear methods, but on the other hand more vulnerable to overfitting and require careful initialization/regularization based on domain or prior knowledge.

3.5 Experiments on Real Data

We consider three real data sets. For the purpose of evaluation, we choose data whose temporal orderings are known: a video stream of a swinging pendulum (Section 3.5), gene expression time

series of the yeast metabolic cycle (Section 3.5.2), and time series of HeLa cell images (Section 3.5.3). While the first one is for evaluation purposes only, the other two data sets, as briefly described in Chapter 1 and explained in more detail later, represent application areas that would truly benefit from the proposed methods of learning dynamic models from non-sequence data.

In real data we do not have true dynamic models to compare with, but knowing the temporal order allows us to obtain reference models by applying existing dynamic model learning methods with the available temporal order. Our main evaluation scheme is then to compare the prediction performances of the proposed methods against standard methods of learning from sequence data. We consider two performance metrics. One is the cosine score (3.49). However, since the real data are higher-dimensional, when interpreting cosine scores we need to account for the effect of high dimensions. To see that effect, we consider the probability that random prediction achieves a cosine score of s or greater. Let the random variable S denote the cosine between a vector drawn uniformly at random from the unit p-sphere and an arbitrary fixed p-dimensional unit vector. Basic geometry shows that the probability $|S| \geq s$ for some $s > 0$ is equivalent to two times the ratio of the surface area of a cap with height $1-s$ on a unit p-sphere to the unit p-sphere surface area, which has the following closed-form [Li, 2011]:

$$\text{Prob}(|S| \geq s) \;=\; \mathcal{I}_{1-s^2}\left(\frac{p-1}{2}, \frac{1}{2}\right), \tag{3.50}$$

where $\mathcal{I}_x(a,b)$ is the regularized incomplete beta function. Now consider the cosine score of n independent random predictions $|\bar{S}| := |\frac{\sum_{i=1}^n S_i}{n}|$, where S_i's are independent copies of S. By Bennett's inequality [1962] and symmetry of S, we have

$$\text{Prob}(|\bar{S}| \geq s) \;=\; 2\text{Prob}(\bar{S} \geq s) \leq 2\exp\left(-n\text{Var}(S)h(s/\text{Var}(S))\right), \tag{3.51}$$

where $h(x) := (1+x)\log(1+x) - x$. To derive $\text{Var}(S)$, we first obtain the p.d.f of S:

$$f_S(s) \;=\; \frac{d\text{Prob}(S \leq s)}{ds} \;=\; \frac{d\left(1 - \frac{1}{2}\mathcal{I}_{1-s^2}\left(\frac{p-1}{2}, \frac{1}{2}\right)\right)}{ds} \;=\; \frac{(1-s^2)^{\frac{p-3}{2}}}{B(\frac{p-1}{2}, \frac{1}{2})}, \tag{3.52}$$

where $B(x,y)$ is the beta function. Then we have

$$\text{Var}(S) \;=\; \int_{-1}^{1} s^2 f_S(s) ds \;=\; \frac{B(\frac{p-1}{2}, \frac{3}{2})}{B(\frac{p-1}{2}, \frac{1}{2})} \;=\; p^{-1}, \tag{3.53}$$

leading to the following upper bound:

$$\text{Prob}(|\bar{S}| \geq s) \;\leq\; 2\exp(-nh(sp)/p) = 2\left(\frac{\exp(s)}{(1+sp)^{1/p+s}}\right)^n, \tag{3.54}$$

which decreases in the order of p^{-sn} for fixed s. Figure 3.14 shows this upper bound for four values of s and $n = 10$ as a function of p. We can see that when the dimension p is large, even if the number n of predictions is small (as in Section 3.5.2) it is still difficult for random prediction

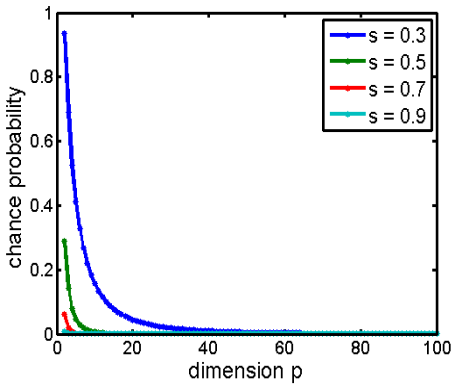


Figure 3.14: Chance probability (3.54) for various cosine scores against the dimension p

to achieve some modest cosine score, say 0.3. In later sections we will use the upper bound (3.54) to provide some sense of significance of the cosine scores obtained in our experiments.

Our second performance metric is **normalized error**:

$$\frac{1}{T-1}\sum_{t=1}^{T-1}\frac{\|\mathbf{x}^{(t+1)}-\widehat{\mathbf{x}}^{(t+1)}\|}{\|\mathbf{x}^{(t+1)}-\mathbf{x}^{(t)}\|}, \tag{3.55}$$

which measures how close the predictions are to the true next state vectors. A smaller normalized error means a better prediction, and predicting with the current observation ($\widehat{\mathbf{x}}^{(t+1)}=\mathbf{x}^{(t)}$) gives a normalized error of one.

In all three data sets we apply UM and PM, and in Sections 3.5 and 3.5.2 also use the tree-based EM (TEM) method. For the experiments in Section 3.5.3 we use random initialization, while in Sections 3.5 and 3.5.2 we apply the temporal clustering heuristics in Section 3.3 for initialization with the following detailed settings.

We use the K-means method to cluster the data points and compare the following four methods for ordering the cluster centers:

1. MVU: Project the cluster centers to the one-dimensional space found by Maximum Variance Unfolding, and then sort the cluster centers according to the projections.
2. l1+TSP: Solve a TSP with the 1-norm pairwise distances between the cluster centers.
3. l2+TSP: Solve a TSP with the 2-norm pairwise distances between the cluster centers.
4. TSM+TSP: The two-step heuristics outlined in Section 3.3.

Then, we learn a linear model (3.1) from the ordered cluster centers, and initialize the proposed methods with the learnt model. As mentioned in Section 3.3, methods based on TSP do not decide the overall direction of time. Here we learn dynamic models using both directions, and report the one that leads to a better prediction performance. To solve a TSP, we use the state-of-the-art *Concorde TSP solver* [Applegate et al.].

For UM and TEM, we not only initialize the estimation procedures with clustering, but also consider restricting the approximate likelihood functions by the ordering of the cluster centers: when summing over the latent variables in (3.8) and (3.25), we only include those consistent with

Figure 3.15: A frame of the swinging pendulum video stream.

the ordering of the clusters. We refer to the restricted versions of UM and TEM respectively as "UM rest" and "TEM rest."

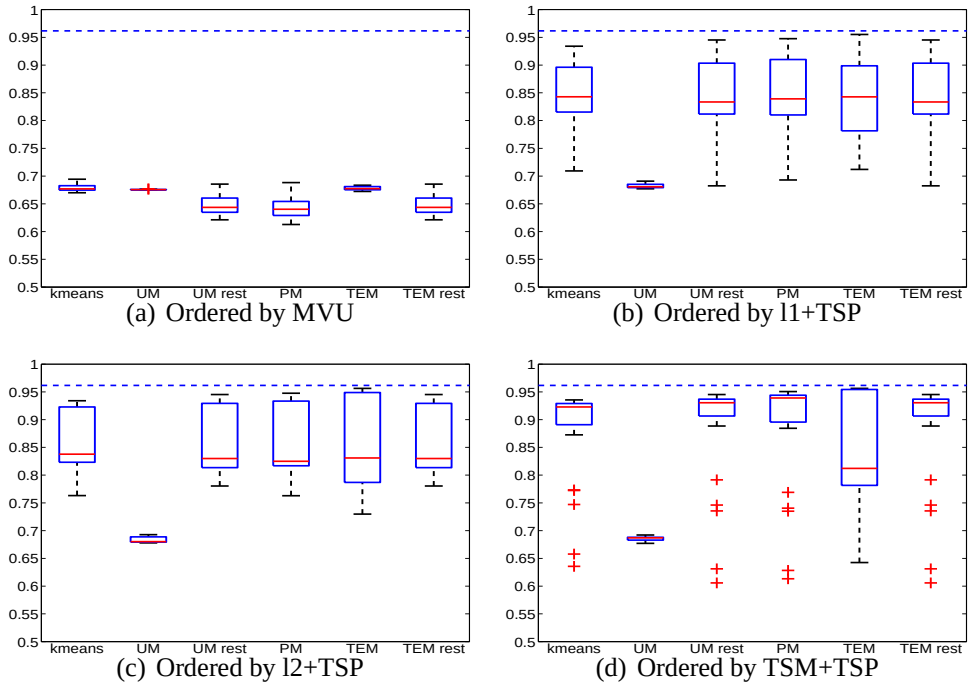
We extend the proposed methods to allow each state variable to have a different noise variance. The update rules can be easily derived from those in Section 3.2 and have a similar form. We choose the regularization parameter λ by leave-one-out cross validation on the ordered cluster centers, but set α and β by manual selection. Our choice of α and β is mainly to avoid numerical issues caused by small values of the estimated noise variances during the EM iterations. We find the follow choice to be effective: $\alpha = 1$ and $\beta \approx n$, which correspond to a prior of noise variance whose mean is around n, and in our experiments leads to a posterior mean around 2.

3.5.1 Video of Swinging Pendulum

Our first real data is a video analyzed by Siddiqi et al. [2010]. The video consists of 500 frames of 240-by-240 colored images of a swinging pendulum. An image is shown in Figure 3.15. The underlying dynamics is highly periodic and stable as the pendulum completes about 22 full swings[4]. We center the pixel values to be zero across the 500 frames, and then apply Singular Value Decomposition (SVD) to reduce the dimension from $240 \times 240 \times 3 = 172800$ to 20 by projecting the data onto the subspace corresponding to the 20 largest singular values. Such a subspace preserves about 72 percent of the total energy. We further normalize each of the 20 temporal sequences to be zero-mean and unit-variance. Then we use the first 400 points as training data and the last 100 points as testing data.

In the initialization step we combine the K-means method with the AIC criterion to determine the number of clusters. For each possible number of clusters in our search range, we run the K-means method with 30 random restarts and choose the best clustering to initialize the dynamic model. We repeatedly train 30 linear dynamic models, all of which are initialized by K-means combined with AIC. In most of the 30 runs the number of clusters determined by K-means and AIC is 31, which is about the number of time steps one full swing takes. We then evaluate these learnt models by their prediction performances on the test data.

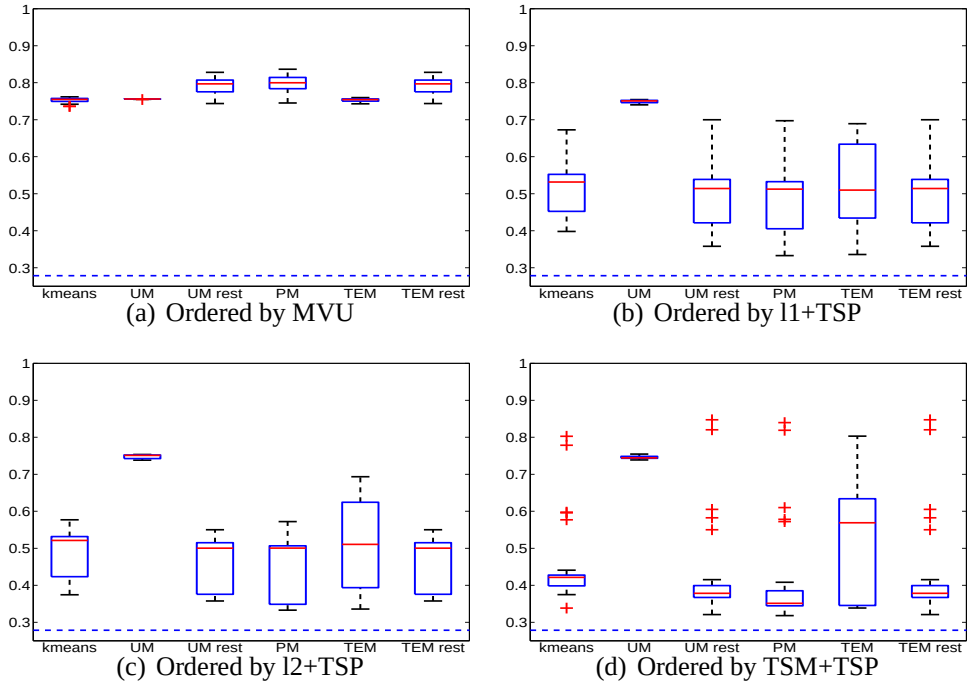
[4] A full swing means the pendulum ended where it started.



(a) Ordered by MVU (b) Ordered by l1+TSP

(c) Ordered by l2+TSP (d) Ordered by TSM+TSP

Figure 3.16: Cosine scores on the pendulum data by the linear model. Larger is better. The blue dashed line is by a dynamic model learnt with the known temporal order.



(a) Ordered by MVU (b) Ordered by l1+TSP

(c) Ordered by l2+TSP (d) Ordered by TSM+TSP

Figure 3.17: Normalized errors on the pendulum data by the linear model. Smaller is better. The blue dashed line is by a model learnt with the known order.

We present the box plots of the testing cosine scores and normalized errors in Figures 3.16 and 3.17. The left most column in each plot, the "kmeans" column, gives the performance of the initial model found by K-means and some ordering method. In each box-plot we also indicate the performance of the reference model learnt with the known temporal ordering. There are two main observations:

- Comparing the four ordering methods, we find that MVU is worse than l1+TSP and l2+TSP, which are in turn worse than TSM+TSP. Moreover, TSM+TSP does almost as well as the model learnt with the known temporal ordering. This suggests that orderings solely based on pairwise distances, such as those by MVU, l1+TSP, and l2+TSP, may be more sensitive to distances between cluster centers, which are not always equally separated in space. On the contrary, TSM+TSP is more robust against irregular distances, suggesting that the pairwise similarity learnt through solving the convex program (3.42) better captures the dynamic nature of the data.
- The initial models learnt from ordered cluster centers already perform quite well, and the proposed methods result in only marginal improvements. Moreover, without the restriction imposed by cluster orderings UM even performs worse than the initial model. This suggests that our approximation to the likelihood function may introduce too many undesirable local maxima.

3.5.2 Gene Expression Time Series of Yeast Metabolic Cycle

To study gene expression dynamics of yeasts during the metabolic cycle, Tu et al. [2005] collected expression profiles of about 6,000 yeast genes along three consecutive metabolic cycles, each containing 12 samples. Due to the destructive nature of the measurement technique, gene expression profiles were measured on different yeast cells, and therefore synchronization of yeast cells in the metabolic cycle is necessary for obtaining reliable gene expression time series data. To address this issue, Tu et al. [2005] developed a continuous culture system that provides a stable environment for yeast cells to grow, and chose a particular strain of yeasts that exhibit "unusually robust periodic behavior," i.e., cells of that strain of yeasts are in a sense self-synchronizing. However, Tu et al. [2005] noted that the periodic gene expression observed in their experiment is more robust than those in certain other species (c.f. **Discussion** in [Tu et al., 2005]), suggesting that in general it may be quite difficult to obtain reliable time series gene expression measurements. In those cases, our proposed methods of learning dynamic models from non-sequenced data may be very useful.

We focus on a subset of 3,552 genes found by Tu et al. [2005] to exhibit strong periodical behaviors during the metabolic cycle. We normalize each gene expression time series to be zero-mean and unit-variance, and use the first two cycles (24 points) as training data and the last cycle (12 points) as testing data. Here the number of state variables (genes) is much higher than the sample size, and thus learning dynamic models is much more difficult than in the previous experiment.

Since the number of sample points is disproportionally smaller than the dimension, in the initialization step we specify the number of clusters in the K-means method to be 12, the number of time steps in one cycle. This means each cluster will contain only few data points. We

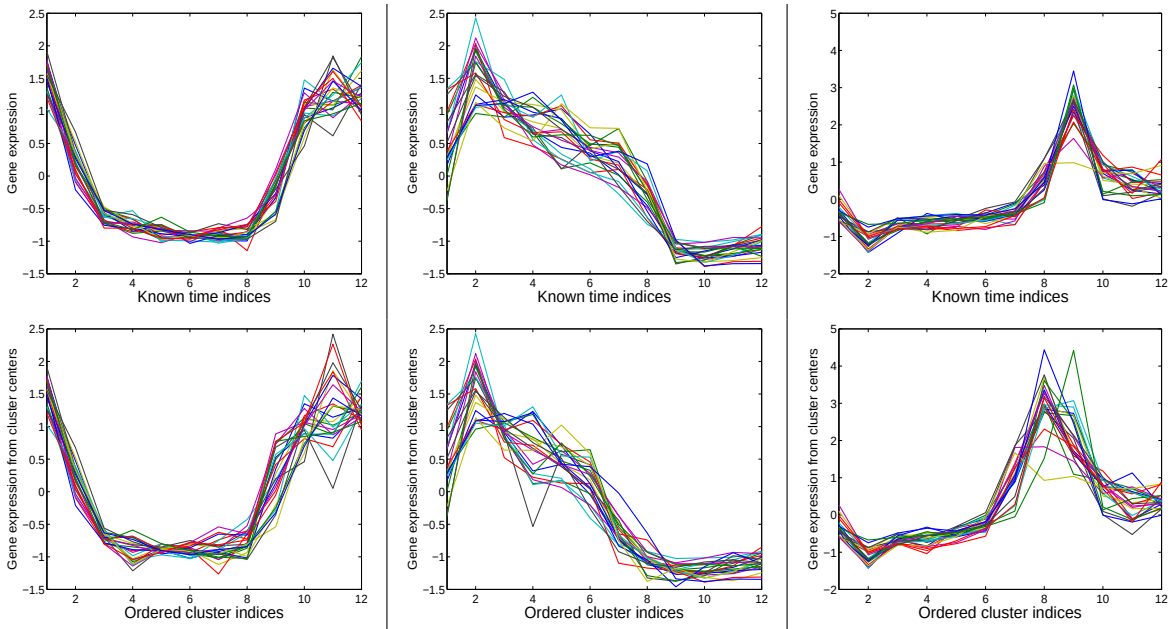
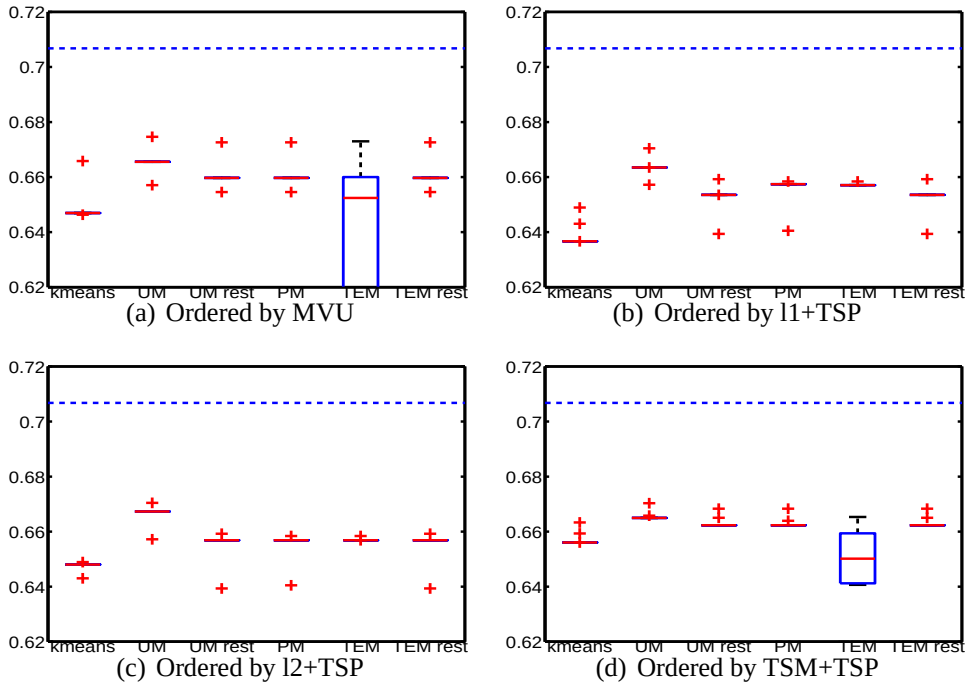


Figure 3.18: Gene expression profiles in three major gene groups: MRPL10, POX1, and RPL17B. Top row: original gene expression. Bottom row: gene expression from estimated cluster centers ordered by TSM+TSP.

repeatedly train 20 linear models, and in each of the 20 runs we randomly restart the K-means method 30 times and choose the best clustering to initialize the model.

To evaluate the proposed methods, we first qualitatively examine our initial step of temporal clustering and ordering. Among the 3,552 genes, MRPL10, POX1 and RPL17B were found to be strongly periodical and yet exhibit different dynamics. Treating these three genes as fixed seeds in clustering analysis, Tu et al. [2005] identified three major clusters of genes. From each cluster we pick the 24 most representative genes and plot their average expression profiles over the first two cycles in the top row of Figure 3.18. In the bottom row of the same figure we plot the expression profiles of the same genes from the estimated cluster centers in the order found by TSM+TSP. Comparing the two rows shows our initial step of temporal clustering and ordering effectively recovers the major trends of gene dynamics.

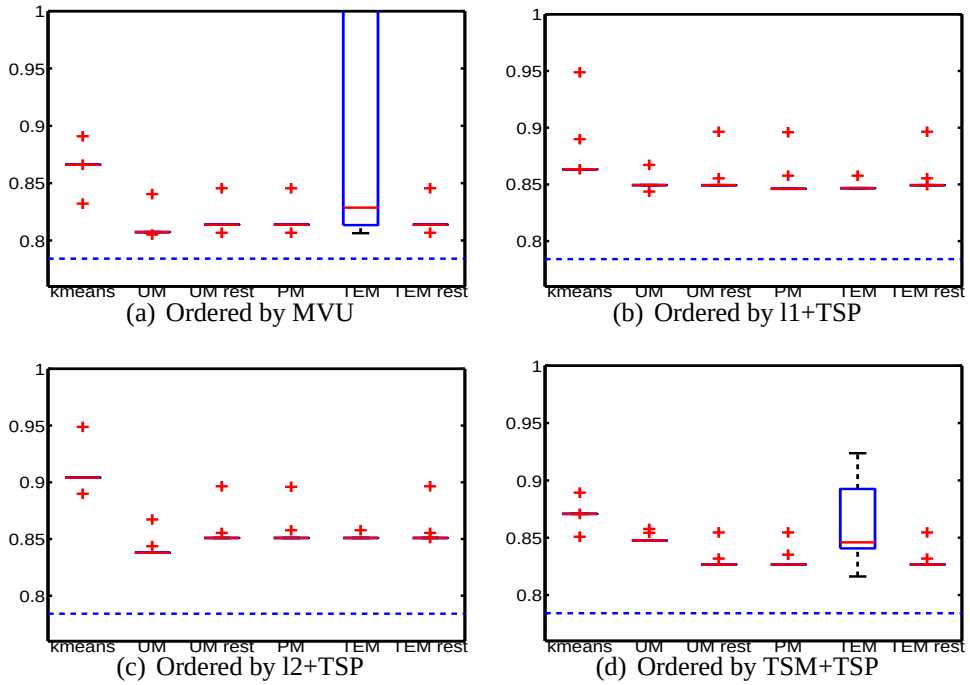
We then evaluate the proposed methods quantitatively. Figures 3.19 and 3.20 present box-plots of cosine scores and normalized errors. The cosine scores are between 0.6 and 0.7, which by themselves do not seem impressive, but because of the high dimension, the probability for random predictions to achieve such scores, according to (3.54), is less than 10^{-19} even though the testing sequence is short. Moreover, the improvements due to the proposed EM-based methods over the initial model are more significant here than in Section 3.5, though the gap between the proposed methods and the sequential learning method is bigger. Most of the performance measures here are rather stable across different runs, possibly because on such a small sample most initializations turn out to be similar. The only exception is TEM, which occasionally results in extremely poor performance. This is due to numerical difficulties encountered in its E-step;



(a) Ordered by MVU (b) Ordered by l1+TSP

(c) Ordered by l2+TSP (d) Ordered by TSM+TSP

Figure 3.19: Cosine scores on the yeast time series. Larger is better. The blue dashed line is by a dynamic model learnt using the known temporal order.



(a) Ordered by MVU (b) Ordered by l1+TSP

(c) Ordered by l2+TSP (d) Ordered by TSM+TSP

Figure 3.20: Normalized errors on the yeast time series. Smaller is better. The blue dashed line is by a dynamic model learnt using the known temporal order.

the main computation there is inverting a matrix of exponentiated negative distances, which are numerically unstable for high dimensional data points. Regarding the different ordering methods, unlike in Section 3.5 TSM+TSP does not outperform the other ordering methods; all four methods perform equally well. Again, this may be attributed to the training points being too few for different ordering methods to behave differently.

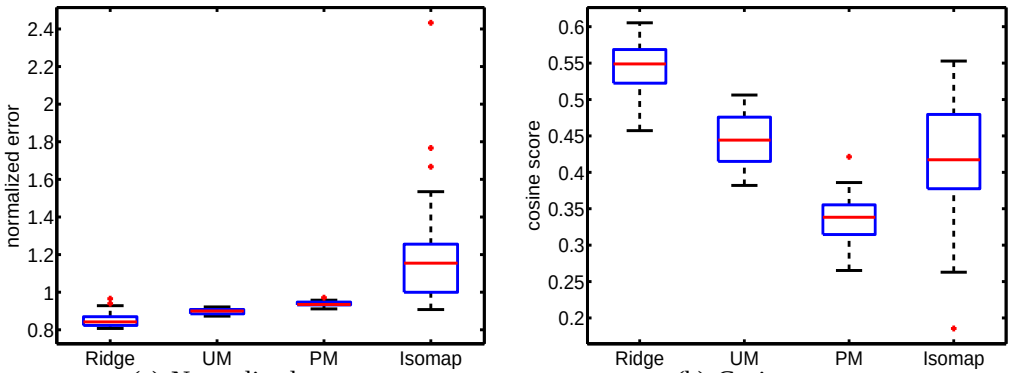
3.5.3 Cell Image Time Series

We apply the proposed method to a time series data set of HeLa cell images originally collected by Zhou et al. [2009], and subsequently analyzed by Buck et al. [2009], who were interested in the dependence of protein subcellular localization on the cell cycle. Instead of relying on time-series cell images as in most existing studies, they aim to utilize static, asynchronous snapshots taken from multiple cells at various phases of the cell cycle because such images are easier to obtain on a large scale than time-series images. To do so, they proposed to find a one-dimensional surrogate of cell cycle time from static cell image features by manifold learning techniques, and verified on real data that such a surrogate is well correlated with the cell cycle. However, it is not clear how to use or augment their approach for predictive analysis, which can be important in understanding cell dynamics. In contrast, our work bypasses the issue of estimating the cell cycle time and focuses directly on learning dynamic models.

The data set consists of 100 time frames, and each frame contains from tens to a hundred or so cell regions. Details regarding cell segmentation and tracking can be found in [Zhou et al., 2009]. Each segmented cell region is represented by a 49 dimensional feature vector as in [Buck et al., 2009]. During the 100 time frames, some cells went through more than one division while others never divided. Buck et al. [2009] identified a total of 34 sequences of cells that completed at least one full division-to-division cell cycle spanning at least 30 time frames, and conducted their analysis on these sequences. We instead treat these 34 sequences, which contain a total of 1,740 data points, as testing data, and run UM and PM on the other short or incomplete sequences as if they were non-sequence samples. Out of the 7,692 feature points in these partial cell cycle sequences, 1,267 appear in only one time frame, fitting exactly our non-sequence scenario. We normalize the entire data set so that each feature has mean zero and standard deviation 1. To obtain a performance reference from models learnt with sequence information, we apply least square ridge regression to partial sequences of length at least 6, a total of 6099 feature points. The regularization parameter for ridge regression was chosen by training on the first half of each training sequence and validating on the second half.

In applying UM and PM we made several changes. We allow each feature to have a different noise variance, but instead of optimizing over its value, we simply set the noise variance of each feature to be the median of pairwise distances between training data points along that feature dimension. Moreover, we add an extra regularization term $2^{-3}\|A - I\|_F^2$ to our approximate likelihood functions and set λ for the ℓ_2 penalty perm on A to be 1. We initialize UM and PM with 20 different models, one being the identity matrix and the other 19 having entries drawn independently from a standard normal distribution. We choose the final estimate based on the regularized approximate likelihood functions for UM and PM, respectively.

We compare with a baseline that uses manifold learning. Following Buck et al. [2009], we use Isomap [Tenenbaum et al., 2000] to map all the 7,692 training data points to a two-



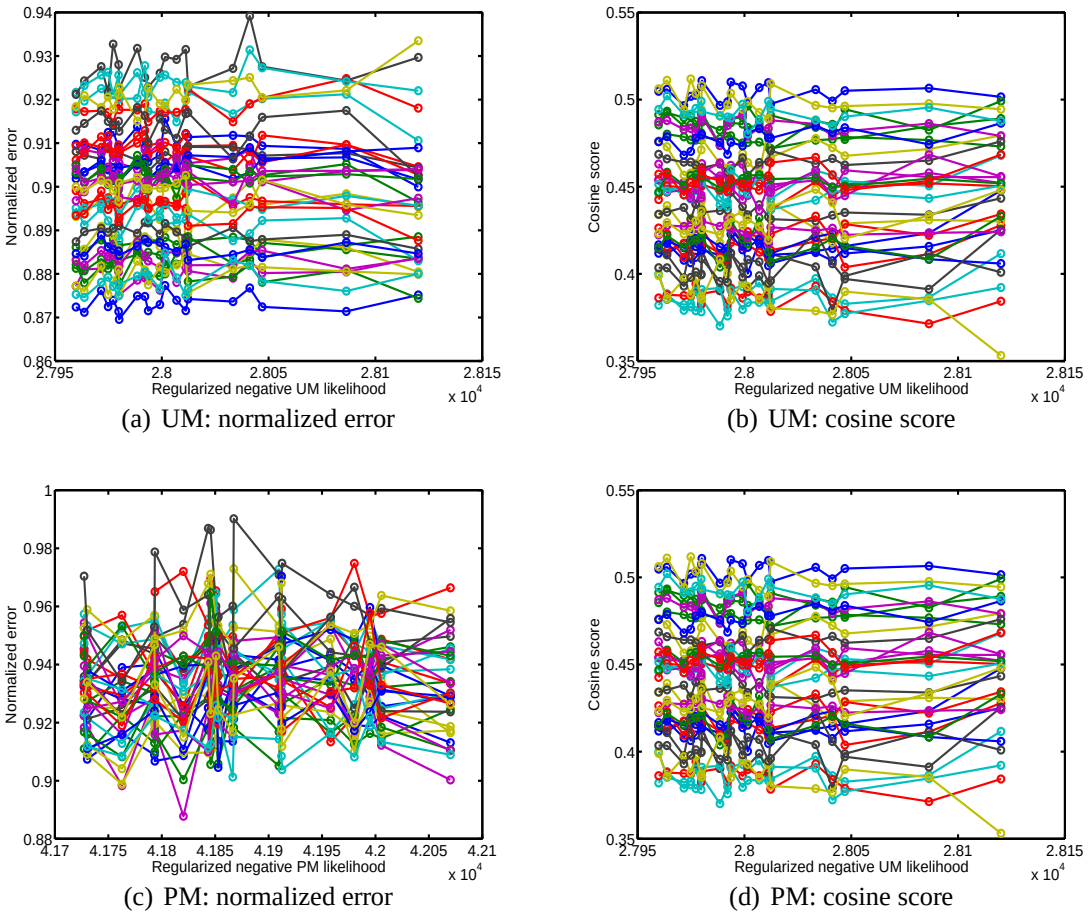
(a) Normalized error (b) Cosine score

Figure 3.21: Testing performances on cell image time series

dimensional space, sort the data points according to their mappings along the first coordinate, and then apply ridge regression with both the estimated ordering and the reverse ordering. At test time, we apply both learnt models and report the better performance.

Figures 3.21(a) and 3.21(b) are boxplots of the two performance measures over the 34 testing sequences. Again, although the cosine scores do not seem impressive, the chance probability for achieving such scores, given by (3.50), is lower than 10^{-6}. As expected, Ridge performs the best, but the proposed UM and PM are quite close and even have a smaller variance in normalized error. The baseline that uses Isomap is competitive with UM in terms of cosine score, but shows larger variability across different testing sequences and has much larger normalized errors. Unlike in the last two experiments, PM performs noticeably worse than UM here. We suspect that PM's strong approximation bias of enforcing the spanning-tree constraint hinders effective use of this quite large data set, but do not have solid empirical evidences. More generally, it requires further research to understand when UM or PM will be a better choice in terms of quality of the learnt model, but UM certainly scales better with the data size: UM's E step of normalization is embarrassingly parallelizable, whereas PM's maximum directed spanning tree procedure is harder to parallelize. In this experiment, our MATLAB implementation of UM, which performs efficient matrix normalization via parallelization, is more than 10 times faster than PM, which spends most of the running time in the maximum directed spanning tree solver (`http://edmonds-alg.sourceforge.net/`, version 1.1.0).

Another way of evaluating the proposed approximate likelihood functions is to check whether a better training likelihood leads to a better testing performance. Figure 3.22 gives scatter plots of the two testing performance measures against regularized UM and PM training negative likelihood functions over the 20 initializations. Each curve represents results on a testing sequence and is sorted by the training likelihood. We can see that for both UM and PM, the training approximate likelihood has a rather small numerical range, and there is no significant correlation between the testing performance and the training likelihood. Moreover, on most testing sequences the UM performances are similar across the 20 initial models, while the PM normalized error has a larger variation. These results illustrate limitations of our proposed methods when applied to real data, and suggest that a different strategy of choosing the final estimate is needed in order to achieve better testing performance, which we leave as an open issue for future work.



(a) UM: normalized error (b) UM: cosine score

(c) PM: normalized error (d) PM: cosine score

Figure 3.22: Scatter plot of testing performance against training approximate likelihood

3.6 Conclusion

In this chapter we consider learning fully observable dynamic models from data drawn from independent trajectories at unknown times. Acknowledging several identifiability issues, we propose learning methods based on maximizing various approximate likelihood functions via EM-type algorithms, together with novel initialization methods. Experiments on synthetic and real data demonstrate that the proposed methods can learn reasonably good models from non-sequence data, though their success requires some hyper-parameter tuning, and more critically, good initialization. We thus in later chapters consider settings requiring extra information or assumptions, but leading to simpler learning problems.

Chapter 4

Learning Vector Autoregressive Models from Sequence and Non-sequence Data

As concluded by the previous chapter, the assumption of multi-trajectory, independent samples leads to several identifiability issues that compromise effective learning. We thus consider making stronger assumptions in several ways, starting in this chapter with the availability of a small amount of sequence data in addition to the supposedly larger non-sequence data. Our goal is to combine these two types of data to improve dynamic model learning. As in the previous chapter, we consider p-dimensional vector auto-regressive models, but treat the state variables as a row vector instead of a column vector:

$$\mathbf{x}^{(t+1)} \;=\; \mathbf{x}^{(t)} A + \boldsymbol{\epsilon}^{(t+1)}, \tag{4.1}$$

where $\boldsymbol{\epsilon}^{(t)}$ is again an independent Gaussian noise process with a time-invariant variance $\sigma^2 I$. In addition, we assume that the process (4.1) is stable, i.e., the eigenvalues of A have modulus less than one. As a result, the process (4.1) has a stationary distribution, whose covariance Q is determined by the following discrete-time Lyapunov equation:

$$A^\top Q A + \sigma^2 I = Q. \tag{4.2}$$

Linear quadratic Lyapunov theory (see e.g., [Antsaklis and Michel, 2005]) gives that Q is *uniquely* determined if and only if $\lambda_i(A)\lambda_j(A) \neq 1$ for $1 \leq i, j \leq p$, where $\lambda_i(A)$ is the i-th eigenvalue of A. If the noise process $\boldsymbol{\epsilon}^t$ follows a normal distribution, the stationary distribution also follows a normal distribution, with covariance Q determined as above. Since our goal is to estimate A, a more relevant perspective is viewing (4.2) as a system of constraints on A. What motivates the propose approach in this chapter is that the estimation of Q requires only samples drawn from the stationary distribution rather than sequence data. However, even if we have the true Q and σ^2, we still cannot uniquely determine A because (4.2) is an under-determined system[1] of A. We thus rely on the few sequence samples to resolve the ambiguity.

Let $\{\mathbf{x}^{(i)}\}_{i=1}^{T}$ be a sequence of observations generated by the process (4.1). The standard least-square estimator for the transition matrix A is the solution to the following minimization

[1]If we further require A to be symmetric, (4.2) would be a simplified *Continuous-time Algebraic Riccati Equation*, which has a unique solution under some conditions (c.f. [Antsaklis and Michel, 2005]).

problem:

$$\min_{A} \quad \|Y - XA\|_F^2, \tag{4.3}$$

where $Y^\top := [(\mathbf{x}^{(2)})^\top \ (\mathbf{x}^{(3)})^\top \cdots (\mathbf{x}^{(T)})^\top]$, $X^\top := [(\mathbf{x}^{(1)})^\top \ (\mathbf{x}^{(2)})^\top \cdots (\mathbf{x}^{(T-1)})^\top]$, and $\|\cdot\|_F$ denotes the matrix Frobenius norm. When $p > T$, which is often the case in modern time series modeling tasks, the least square problem (4.3) has multiple solutions all achieving zero squared error, and the resulting estimator overfitts the data. A common remedy is adding a penalty term on A to (4.3) and minimizing the resulting regularized sum of squared errors. Usual penalty terms include the ridge penalty $\|A\|_F^2$ and the sparse penalty $\|A\|_1 := \sum_{i,j} |A_{ij}|$.

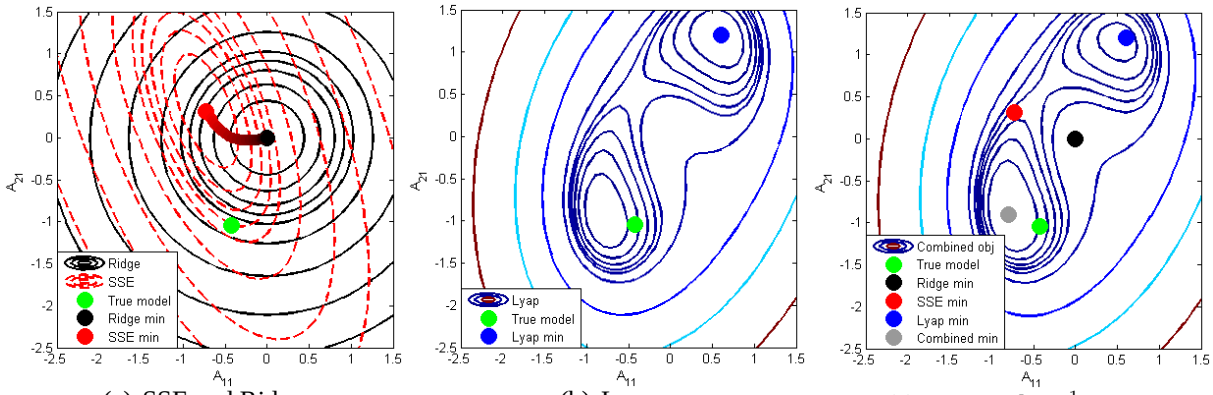
Now suppose we also have a set of non-sequence observations $\{\mathbf{z}_i\}_{i=1}^n$ drawn independently from the stationary distribution of (4.1). Note that we use superscripts for time indices and subscripts for data indices. As described in Chapter 1, the size n of the non-sequence sample can usually be much larger than the size T of the sequence data. To incorporate the non-sequence observations into the estimation procedure, we first obtain a covariance estimate $\widehat{Q}$ of the stationary distribution from the non-sequence sample, and then turn the Lyapunov equation (4.2) into a regularization term on A. More precisely, in addition to the usual ridge or sparse penalty terms, we also consider the following regularization:

$$\|A^\top \widehat{Q} A + \sigma^2 I - \widehat{Q}\|_F^2, \tag{4.4}$$

which we refer to as the *Lyapunov penalty*. To compare (4.4) with the ridge penalty and the sparse penalty, we consider (4.3) as a multiple-response regression problem and view the i-th column of A as the regression coefficient vector for the i-th output dimension. From this viewpoint, we immediately see that both the ridge and the sparse penalizations treat the p regression problems as unrelated. On the contrary, the Lyapunov penalty incorporates relations between pairs of columns of A by using a covariance estimate $\widehat{Q}$. In other words, although the non-sequence sample does not provide direct information about the individual regression problems, it does reveal how the regression problems are related to one another. To illustrate how the Lyapunov penalty may help to improve learning, we give an example in Figure 4.1. The true transition matrix is

$$A \ = \ \begin{bmatrix} -0.4280 & 0.5723 \\ -1.0428 & -0.7144 \end{bmatrix} \tag{4.5}$$

and $\boldsymbol{\epsilon}^t \sim \mathcal{N}(\mathbf{0}, I)$. We generate a sequence of 4 points, draw a non-sequence sample of 20 points independently from the stationary distribution and obtain the sample covariance $\widehat{Q}$. We fix the second column of A but vary the first, and plot in Figure 4.1(a) the resulting level sets of the sum of squared errors on the sequence (SSE) and the ridge penalty (Ridge), and in Figure 4.1(b) the level sets of the Lyapunov penalty (Lyap). We also give coordinates of the true $[A_{11} \ A_{21}]^\top$, the minima of SSE, Ridge, and Lyap, respectively. To see the behavior of the ridge regression, we trace out a path of the ridge regression solution by varying the penalization parameter, as indicated by the red-to-black curve in Figure 4.1(a). This path is pretty far from the true model, due to insufficient sequence data. For the Lyapunov penalty, we observe that it has two local minima, one of which is very close to the true model, while the other, also the global minimum, is very far. Thus, neither ridge regression nor the Lyapunov penalty can be used on its own to estimate the true model well. But as shown in Figure 4.1(c), the combined objective, SSE+Ridge+$\frac{1}{2}$Lyap,



(a) SSE and Ridge (b) Lyap (c) SSE+Ridge+$\frac{1}{2}$Lyap

Figure 4.1: Level sets of different functions in a bivariate AR example

has its global minimum very close to the true model. This demonstrates how the ridge regression and the Lyapunov penalty may complement each other: the former by itself gives an inaccurate estimation of the true model, but is just enough to identify a good model from the many candidate local minima provided by the latter.

In the following we describe our proposed methods for incorporating the Lyapunov penalty (4.4) into ridge and sparse least-square estimation. We also discuss robust estimation for the covariance Q.

4.1 Ridge and Lyapunov penalty

Here we estimate A by solving the following problem:

$$\min_{A} \quad \frac{1}{2}\|Y - XA\|_F^2 + \frac{\lambda_1}{2}\|A\|_F^2 + \frac{\lambda_2}{4}\|A^\top \widehat{Q} A + \sigma^2 I - \widehat{Q}\|_F^2, \tag{4.6}$$

where $\widehat{Q}$ is a covariance estimate obtained from the non-sequence sample. We treat λ_1, λ_2 and σ^2 as hyperparameters and determine their values on a validation set. Given these hyperparameters, we solve (4.6) by gradient descent with back-tracking line search for the step size. The gradient of the objective function is given by

$$-X^\top Y + X^\top X A + \lambda_1 A + \lambda_2 \widehat{Q} A (A^\top \widehat{Q} A + \sigma^2 I - \widehat{Q}). \tag{4.7}$$

As mentioned before, (4.6) is a non-convex problem and thus requires good initialization. We use the following two initial estimates of A:

$$\widehat{A}^{lsq} := (X^\top X)^\dagger X^\top Y \quad \text{and} \quad \widehat{A}^{ridge} := (X^\top X + \lambda_1 I)^{-1} X^\top Y, \tag{4.8}$$

where $(\cdot)^\dagger$ denotes the Moore-Penrose pseudo inverse of a matrix, making $\widehat{A}^{lsq}$ the minimum-norm solution to the least square problem (4.3). We run the gradient descent algorithm with these two initial estimates, and choose the estimated A that gives a smaller objective.

4.2 Sparse and Lyapunov penalty

Sparse learning for vector auto-regressive models has become a useful tool in many modern time series modeling tasks, where the number p of states in the system is usually larger than the length T of the time series. For example, an important problem in computational biology is to understand the progression of certain biological processes from some measurements, such as temporal gene expression data.

Using an idea similar to (4.6), we estimate A by

$$\begin{array}{ll} \min\limits_{A} & \frac{1}{2}\|Y - XA\|_F^2 + \frac{\lambda_2}{4}\|A^\top \widehat{Q} A + \sigma^2 I - \widehat{Q}\|_F^2, \\ \text{s.t.} & \|A\|_1 \leq \lambda_1. \end{array} \tag{4.9}$$

Instead of adding a sparse penalty on A to the objective function, we impose a constraint on the ℓ_1 norm of A. Both the penalty and the constraint formulations have been considered in the sparse learning literature, and shown to be equivalent in the case of a convex objective. Here we choose the constraint formulation because it can be solved by a simple projected gradient descent method. On the contrary, the penalty formulation leads to a non-smooth and non-convex optimization problem, which is difficult to solve with standard methods for sparse learning. In particular, the soft-thresholding-based coordinate descent method for LASSO does not apply due to the Lyapunov regularization term. Moreover, most of the common methods for non-smooth optimization, such as bundle methods, solve convex problems and need non-trivial modification in order to handle non-convex problems [Noll et al., 2008].

Let $J(A)$ denote the objective function in (4.9) and $A^{(k)}$ denote the intermediate solution at the k-th iteration. Our projected gradient method updates $A^{(k)}$ to $A^{(k+1)}$ by the following rule:

$$A^{(k+1)} \leftarrow \Pi(A^{(k)} - \eta^{(k)} \nabla J(A^{(k)})), \tag{4.10}$$

where $\eta^{(k)} > 0$ denotes a proper step size, $\nabla J(A^{(k)})$ denotes the gradient of $J(\cdot)$ at $A^{(k)}$, and $\Pi(\cdot)$ denotes the projection onto the feasible region $\|A\|_1 \leq \lambda_1$. More precisely, for any p-by-p real matrix V we define

$$\Pi(V) := \arg\min_{\|A\|_1 \leq \lambda_1} \|A - V\|_F^2. \tag{4.11}$$

To compute the projection, we use the efficient ℓ_1 projection technique outlined in Algorithm 3.6 of Chapter 3.

For choosing a proper step size $\eta^{(k)}$, we consider the simple and effective *Armijo rule along the projection arc* described by Bertsekas [1999]. This procedure is given in Algorithm 4.2, and the main idea is to ensure a sufficient decrease in the objective value per iteration (4.13). Bertsekas [1999] proved that there always exists $\eta^{(k)} = \beta^{r_k} > 0$ satisfying (4.13), and every limit point of $\{A^{(k)}\}_{k=0}^{\infty}$ is a stationary point of (4.9). In our experiments we set $c = 0.01$ and $\beta = 0.1$, both of which are typical values used in gradient descent. As in the previous section, we need good initializations for the projected gradient descent method. Here we use these two initial estimates:

$$\widehat{A}^{lsq'} := \arg\min_{\|A\| \leq \lambda_1} \|A - \widehat{A}^{lsq}\|_F^2 \quad \text{and} \quad \widehat{A}^{sp} := \arg\min_{\|A\| \leq \lambda_1} \frac{1}{2}\|Y - XA\|_F^2, \tag{4.12}$$

where $\widehat{A}^{lsq}$ is defined in (4.8), and then choose the one leading to a smaller objective value.

Algorithm 4.1 Armijo's rule along the projection arc

Input: $A^{(k)}, \nabla J(A^{(k)}), 0 < \beta < 1, 0 < c < 1$

Output: $A^{(k+1)}$

1: Find $\eta^{(k)} = \max\{\beta^{r_k} | r_k \in \{0, 1, \ldots\}\}$ such that $A^{(k+1)} := \Pi(A^{(k)} - \eta^{(k)} \nabla J(A^{(k)}))$ satisfies

$$J(A^{(k+1)}) - J(A^{(k)}) \leq c \text{ Tr} \left(\nabla J(A^{(k)})^\top (A^{(k+1)} - A^{(k)}) \right) \tag{4.13}$$

4.3 Robust estimation of covariance matrices

To obtain a good estimator for A using the proposed methods, we need a good estimator for the covariance of the stationary distribution of (4.1). Given an independent sample $\{\mathbf{z}_i\}_{i=1}^n$ drawn from the stationary distribution, the sample covariance is defined as

$$S := \frac{1}{n-1} \sum_{i=1}^{n} (\mathbf{z}_i - \bar{\mathbf{z}})^\top (\mathbf{z}_i - \bar{\mathbf{z}}), \quad \text{where } \bar{\mathbf{z}} := \frac{\sum_{i=1}^n \mathbf{z}_i}{n}. \tag{4.14}$$

Although unbiased, the sample covariance is known to be vulnerable to outliers, and ill-conditioned when the number of sample points n is smaller than the dimension p. Both issues arise in many real world problems, and the latter is particularly common in gene expression analysis. Therefore, researchers in many fields, such as statistics [Ledoit and Wolf, 2004; Stein, 1975; Yang and Berger, 1994], finance [Ledoit and Wolf, 2003], signal processing [Chen et al., 2010a,b], and recently computational biology [Schäfer and Strimmer, 2005], have investigated robust estimators of covariances. Most of these results originate from the idea of *shrinkage estimators*, which shrink the covariance matrix towards some target covariance with a simple structure, such as a diagonal matrix. It has been shown by, e.g., Ledoit and Wolf [2003]; Stein [1975] that shrinking the sample covariance can achieve a smaller mean-squared error (MSE). More specifically, Ledoit and Wolf [2003] consider the following linear shrinkage:

$$\widehat{Q} = (1 - \alpha) S + \alpha F \tag{4.15}$$

for $0 < \alpha < 1$ and some target covariance F, and derive a formula for the optimal α that minimizes the mean-squared error:

$$\alpha^* := \arg \min_{0 \leq \alpha \leq 1} \mathbb{E}(\|\widehat{Q} - Q\|_F^2), \tag{4.16}$$

which involves unknown quantities such as true covariances of S. Schäfer and Strimmer [2005] propose to estimate α^* by replacing all the population quantities appearing in α^* by their unbiased empirical estimates, and derived the resulting estimator $\widehat{\alpha}^*$ for several types of target F. For the experiments later in this chapter we use the estimator proposed by Schäfer and Strimmer [2005] with the following F:

$$F_{ij} = \begin{cases} S_{ij}, & \text{if } i = j, \\ 0 & \text{otherwise}, \end{cases} \quad 1 \leq i, j \leq p. \tag{4.17}$$

Denoting the sample correlation matrix as R, we give in below the final estimator $\widehat{Q}$ [Table 1, Schäfer and Strimmer, 2005]:

$$\widehat{Q}_{ij} := \begin{cases} S_{ij}, & \text{if } i = j, \\ \widehat{R}_{ij}\sqrt{S_{ii}S_{jj}} & \text{otherwise,} \end{cases} \quad \widehat{R}_{ij} := \begin{cases} 1, & \text{if } i = j, \\ R_{ij}\min(1, \max(0, 1 - \widehat{\alpha}^*)) & \text{otherwise,} \end{cases} \tag{4.18}$$

$$\widehat{\alpha}^* := \frac{\sum_{i \neq j} \widehat{\text{Var}}(R_{ij})}{\sum_{i \neq j} R_{ij}^2} = \frac{\sum_{i \neq j} \frac{n}{(n-1)^3} \sum_{k=1}^n (w_{kij} - \bar{w}_{ij})^2}{\sum_{i \neq j} R_{ij}^2}, \tag{4.19}$$

where

$$w_{kij} := (\tilde{\mathbf{z}}_k)_i (\tilde{\mathbf{z}}_k)_j, \qquad \bar{w}_{ij} := \frac{\sum_{k=1}^n w_{kij}}{n}, \tag{4.20}$$

and $\{\tilde{\mathbf{z}}_i\}_{i=1}^n$ are *standardized* non-sequence samples.

4.4 Experiments

To evaluate the proposed methods, we conduct experiments on synthetic and video data. We use the same performance metrics as in Chapter 3 for evaluating a learnt model $\widehat{A}$:

$$\text{Normalized error:} \quad \frac{1}{T-1} \sum_{t=1}^{T-1} \frac{\|\mathbf{x}^{t+1} - \mathbf{x}^t \widehat{A}\|}{\|\mathbf{x}^{t+1} - \mathbf{x}^t\|}.$$

$$\text{Cosine score:} \quad \frac{1}{T-1} \left| \sum_{t=1}^{T-1} \frac{(\mathbf{x}^{t+1} - \mathbf{x}^t)^\top (\mathbf{x}^t \widehat{A} - \mathbf{x}^t)}{\|\mathbf{x}^{t+1} - \mathbf{x}^t\| \|\mathbf{x}^t \widehat{A} - \mathbf{x}^t\|} \right|.$$

In experiments on synthetic data we have the true transition matrix A, so we consider a third criterion, the matrix error: $\|\widehat{A} - A\|_F / \|A\|_F$.

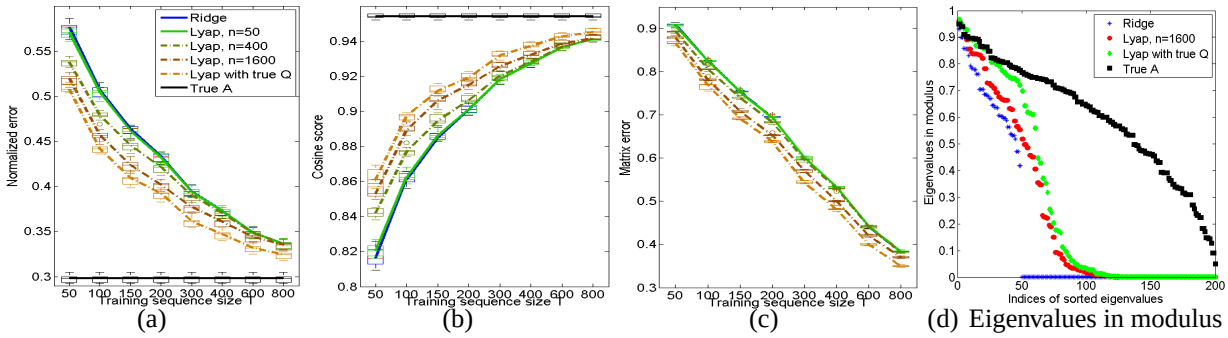
In all our experiments, we have a training sequence, a testing sequence, and a non-sequence sample. To choose the hyper-parameters λ_1, λ_2 and σ^2, we split the training sequence into two halves and use the second half as the validation sequence. Once we find the best hyper-parameters according to the validation performance, we train a model on the full training sequence and predict on the testing sequence. For λ_1 and λ_2, we adopt the usual grid-search scheme with a suitable range of values. For σ^2, we observe that (4.2) implies $\widehat{Q} - \sigma^2 I$ should be positive semidefinite, and thus search the set $\{0.9^j \min_i \lambda_i(\widehat{Q}) \mid 1 \leq j \leq 3\}$. In most of our experiments, we find that the proposed methods are much less sensitive to σ^2 than to λ_1 and λ_2.

4.4.1 Synthetic Data

We consider the following two VAR models with Gaussian noise $\boldsymbol{\epsilon}^t \sim \mathcal{N}(\mathbf{0}, I)$.

$$\text{Dense Model:} \quad A = \frac{0.95M}{\max(|\lambda_i(M)|)}, M_{ij} \sim \mathcal{N}(0,1), 1 \leq i, j \leq 200.$$

$$\text{Sparse Model:} \quad A = \frac{0.95(M \odot B)}{\max(|\lambda_i(M \odot B)|)}, M_{ij} \sim \mathcal{N}(0,1), B_{ij} \sim \text{ Bern } (1/8), 1 \leq i, j \leq 200,$$



(a) (b) (c) (d) Eigenvalues in modulus

Figure 4.2: Testing performances and eigenvalues in modulus for the dense model

where Bern(h) is the Bernoulli distribution with success probability h, and $\odot$ denotes the entry-wise product of two matrices. By setting $h = 1/8$, we make the sparse transition matrix A have roughly $40000/8 = 5000$ non-zero entries. Both models are stable, and the stationary distribution for each model is a zero-mean Gaussian. We obtain the covariance Q of each stationary distribution by solving the Lyapunov equation (4.2). For a single experiment, we generate a training sequence and a testing sequence, both initialized from the stationary distribution, and draw a non-sequence sample independently from the stationary distribution. We set the length of the testing sequence to be 800, and vary the training sequence length T and the non-sequence sample size n: for the dense model, $T \in \{50, 100, 150, 200, 300, 400, 600, 800\}$ and $n \in \{50, 400, 1600\}$; for the sparse model, $T \in \{25, 75, 150, 400\}$ and $n \in \{50, 400, 1600\}$. Under each combination of T and n, we compare the proposed Lyapunov penalization method with the baseline approach of penalized least square, which uses only the sequence data. To investigate the limit of the proposed methods, we also use the true Q for the Lyapunov penalization. We run 10 such experiments for the dense model and 5 for the sparse model, and report the overall performances of both the proposed and the baseline methods.

Experimental results for the dense model

We give boxplots of the three performance measures in the 10 experiments in Figures 4.2(a) to 4.2(c). The ridge regression approach and the proposed Lyapunov penalization method (4.6) are abbreviated as Ridge and Lyap, respectively. For normalized error and cosine score, we also report the performance of the true A on testing sequences.

We observe that Lyap improves over Ridge more significantly when the training sequence length T is small (≤ 200) and the non-sequence sample size n is large (≥ 400). When T is large, Ridge already performs quite well and Lyap does not improve the performance much. But with the true stationary covariance Q, Lyap outperforms Ridge significantly for all T. When n is small, the covariance estimate $\widehat{Q}$ is far from the true Q and the Lyapunov penalty does not provide useful information about A. In this case, the value of λ_2 determined by the validation performance is usually quite small (0.5 or 1) compared to λ_1 (256), so the two methods perform similarly on testing sequences. We note that if instead of the robust covariance estimate in (4.18) and (4.19) we use the sample covariance, the performance of Lyap can be marginally worse than

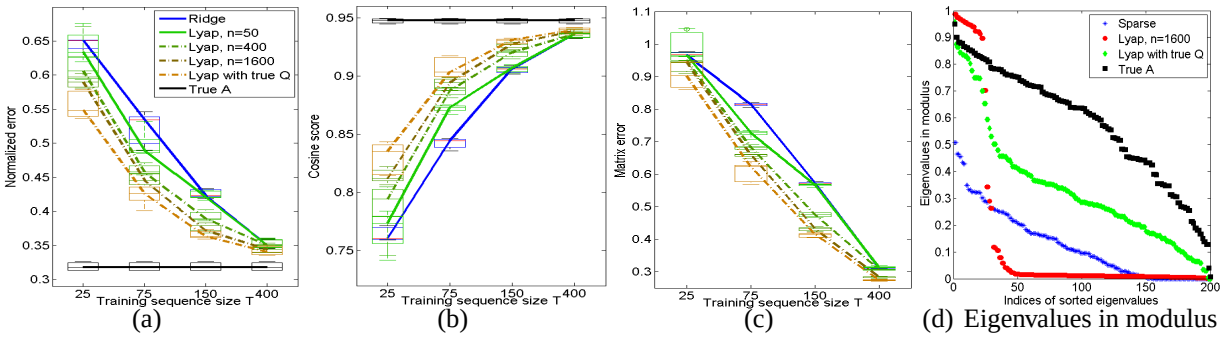


Figure 4.3: Testing performances and eigenvalues in modulus for the sparse model

Ridge when n is small. A precise statement on how the estimation error in Q affects $\widehat{A}$ is worth studying in the future. As a qualitative assessment of the estimated transition matrices, in Figure 4.2(d) we plot the eigenvalues in modulus of the true A and the $\widehat{A}$'s obtained by different methods when $T = 50$ and $n = 1600$. The eigenvalues are sorted according to their modulus. Both Ridge and Lyap severely under-estimate the eigenvalues in modulus, but Lyap preserves the spectrum much better than Ridge.

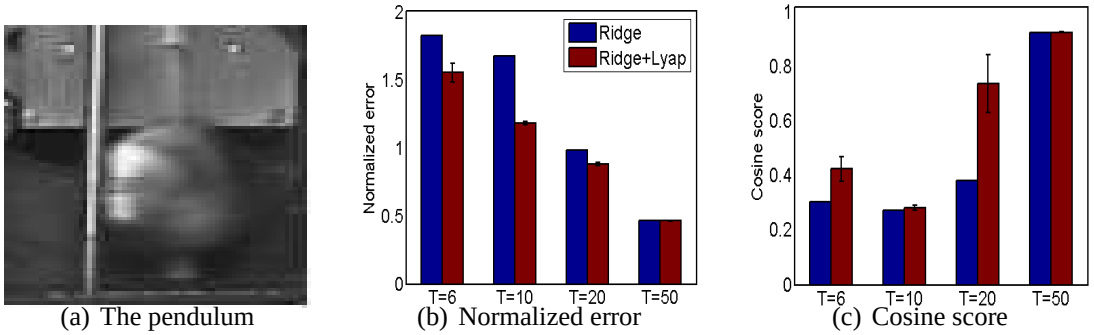
Experimental results for the sparse model

We give boxplots of the performance measures in the 5 experiments in Figures 4.3(a) to 4.3(c), and the eigenvalues in modulus of the true A and some $\widehat{A}$'s in Figure 4.3(d). The sparse least-square method and the proposed method (4.9) are abbreviated as Sparse and Lyap, respectively.

We observe the same type of improvement as in the dense model: Lyap improves over Sparse more significantly when T is small and n is large. But the largest improvement occurs when $T = 75$, not the shortest training sequence length $T = 25$. A major difference lies in the impact of the Lyapunov penalization on the spectrum of $\widehat{A}$, as revealed in Figure 4.3(d). When T is as small as 25, the sparse least-square method shrinks all the eigenvalues but still keep most of them non-zero, while Lyap with a non-sequence sample of size 1600 over-estimates the first few largest eigenvalues in modulus but shrink the rest to have very small modulus. In contrast, Lyap with the true Q preserves the spectrum much better. We may thus need an even better covariance estimate for the sparse model.

4.4.2 Video Data

We test our methods using a video sequence of a periodically swinging pendulum, which is cut from the video used in Chapter 3 and consists of 500 frames of 75-by-80 grayscale images. One such frame is given in Figure 4.4(a) The period is about 23 frames. To further reduce the dimension we take the second-level Gaussian pyramids, resulting in images of size 9-by-11. We then treat each reduced image as a 99-dimensional vector, and normalize each dimension to be zero-mean and standard deviation 1. We analyze this sequence with a 99-dimensional first-order VAR model. To check whether a VAR model is a suitable choice, we estimate a transition matrix



(a) The pendulum (b) Normalized error (c) Cosine score

Figure 4.4: Results on the pendulum video data

from the first 400 frames by ridge regression while choosing the penalization parameter on the next 50 frames, and predict on the last 50 frames. The best penalization parameter is 0.0156, and the testing normalized error and cosine score are 0.33 and 0.97, respectively, suggesting that the dynamics of the video sequence is well-captured by a VAR model.

We compare the proposed method (4.6) with the ridge regression for two lengths of the training sequence: $T \in \{6, 10, 20, 50\}$, and treat the last 50 frames as the testing sequence. For both methods, we split the training sequence into two halves and use the second half as a validation sequence. For the proposed method, we simulate a non-sequence sample by randomly choosing 300 frames from between the $(T+1)$-st frame and the 450-th frame without replacement. We repeat this 10 times.

The testing normalized errors and cosine scores of both methods are given in Figures 4.4(b) and 4.4(c). For the proposed method, we report the mean performance measures over the 10 simulated non-sequence samples with standard deviation. When $T \leq 20$, which is close to the period, the proposed method outperforms ridge regression very significantly except when $T = 10$ the cosine score of Lyap is barely better than Ridge. However, when we increase T to 50, the difference between the two methods vanishes, even though there is still much room for improvement as indicated by the result of our model sanity check before. This may be due to our use of dependent data as the non-sequence sample, or simply insufficient non-sequence data. As for λ_1 and λ_2, their values decrease respectively from 512 and 2,048 to less than 32 as T increases, but since we fix the amount of non-sequence data, the interaction between their value changes is less clear than on the synthetic data.

4.5 Conclusion

In this chapter we propose to improve penalized least-square estimation of VAR models by incorporating non-sequence data independently drawn from the stationary distribution of the underlying VAR model. We construct a novel penalization term based on the discrete-time Lyapunov equation incorporating the covariance (estimate) of the stationary distribution. Although the resulting optimization problems are non-convex, the standard least-square solution obtained by using only sequence data often serves as a good initial point, reducing the need for multiple random initializations. Experimental results demonstrate that our methods can improve signifi-

cantly over standard penalized least-square methods when there are only few sequence data but abundant non-sequence data and when the model assumption is valid. Future directions include investigating the impact of $\widehat{Q}$ on $\widehat{A}$ in a precise manner, generalizing the proposed Lyapunov penalization scheme to handle general noise covariances, and applying the proposed methods to other real-world data.

Chapter 5

Learning Hidden Markov Models from Non-sequence Data

In this and the next chapters we turn to learning hidden Markov models (HMMs) from non-sequence data. At first glance this seems to be an unthinkable attempt because, as shown in Chapter 3, it is not clear how to deal with the various identifiability issues that can seriously compromise learning of the fully observable VAR model, let alone the more complicated HMM, whose estimation is challenging even in the usual sequential learning setting. One major hurdle lies in the use of the EM learning paradigm, which often casts learning as a highly non-convex optimization problem due to hidden variables and consequently suffers from multiple local optima with no guarantee. Moreover, the EM approach usually does not shed much light on ways to reduce the ambiguity of the learning problem without making strong assumptions, because as long as the resulting optimization problem remains non-convex, formal analysis of learning guarantees is still formidable.

We thus propose to take a different approach, based on another long-standing estimation principle: *the method of moments* (MoM). The basic idea of MoM is to find model parameters such that the resulting moments match or resemble the empirical moments. For some estimation problems, this approach is able to give unique and consistent estimates while the maximum-likelihood method gets entangled in multiple and potentially undesirable local maxima. Taking advantage of this property, an emerging area of research in machine learning has recently developed MoM-based learning algorithms *with formal guarantees* for some widely used latent variable models, such as Gaussian Mixture Models [Hsu and Kakade, 2013], Hidden Markov Models [Anandkumar et al., 2012b], Latent Dirichlet Allocation [Anandkumar et al., 2013; Arora et al., 2012], etc. Although many learning algorithms for these models exist, some having been very successful in practice, barely any formal learning guarantee was given until the MoM-based methods were proposed. Such breakthroughs seem surprising, but it turns out that they are mostly based on one crucial property: for quite a few latent variable models, the model parameters can be uniquely determined from *spectral decompositions* of certain low-order moments of observable quantities.

In this chapter we demonstrate that under the MoM and spectral learning framework, there are reasonable assumptions on the generative process of non-sequence data, under which *the tensor decomposition method* [Anandkumar et al., 2012a], a recent advancement in spectral learning,

can provably recover the parameters of certain types of *first-order Markov models* and *hidden Markov models*. To the best of our knowledge, ours is the first work that provides formal guarantees for learning from non-sequence data in terms of parameter estimation accuracy. Interestingly, these assumptions bear much similarity to the usual idea behind *topic modeling*: with the bag-of-words representation which is *invariant to word orderings*, the task of inferring topics is almost impossible given *one single document* (no matter how long it is!), but becomes easier as more documents touching upon various topics become available. For learning dynamic models, what we need in the non-sequence data are *multiple sets* of observations, where each set contains independent samples generated from *its own initial distribution*, and the many different initial distributions together cover the entire (hidden) state space. In some of the scientific applications described in Chapter 1, such as biological studies, this type of assumptions might be realized by running multiple experiments with different initial configurations or amounts of stimuli.

This chapter consists of four sections. Section 5.1 reviews the essentials of the tensor decomposition framework [Anandkumar et al., 2012a]; Section 5.2 details our assumptions on non-sequence data, tensor-decomposition based learning algorithms, and theoretical guarantees; Section 5.3 reports some simulation results confirming our theoretical findings, followed by conclusions in Section 5.4. Proofs of theoretical results are given in Appendix B.

5.1 Tensor Decomposition

We briefly introduce the tensor decomposition framework [Anandkumar et al., 2012a], mainly following their exposition and describing only the components necessary for our work. We first give some preliminaries and notations. A real p-th order tensor A is a member of the tensor product space $\bigotimes_{i=1}^{p} \mathbb{R}^{m_i}$ of p Euclidean spaces. For a vector $\mathbf{x} \in \mathbb{R}^m$, we denote by $\mathbf{x}^{\otimes p} := \mathbf{x} \otimes \mathbf{x} \otimes \cdots \otimes \mathbf{x} \in \bigotimes_{i=1}^{p} \mathbb{R}^m$ its p-th tensor power. A convenient way to represent $A \in \bigotimes_{i=1}^{p} \mathbb{R}^m$ is through a p-way array of real numbers $[A_{i_1 i_2 \cdots i_p}]_{1 \le i_1, i_2, \ldots, i_p \le m}$, where $A_{i_1 i_2 \cdots i_p}$ denotes the $(i_1, i_2, \ldots, i_p)$-th coordinate of A with respect to a canonical basis. With this representation, we can view A as a multi-linear map that, given a set of p matrices $\{X_i \in \mathbb{R}^{m \times m_i}\}_{i=1}^{p}$, produces another p-th order tensor $A(X_1, X_2, \cdots, X_p) \in \bigotimes_{i=1}^{p} \mathbb{R}^{m_i}$ with the following p-way array representation:

$$A(X_1, X_2, \cdots, X_p)_{i_1 i_2 \cdots i_p} := \sum_{1 \le j_1, j_2, \ldots, j_p \le m} A_{j_1 j_2 \cdots j_p} (X_1)_{j_1 i_1} (X_2)_{j_2 i_2} \cdots (X_p)_{j_p i_p}. \tag{5.1}$$

In this work we consider tensors that are up to the third-order ($p \le 3$) and, for most of the time, also *symmetric*, meaning that their p-way array representations are invariant under permutations of array indices. More specifically, we focus on second and third-order symmetric tensors in or slightly perturbed from the following form:

$$M_2 := \sum_{i=1}^{k} \omega_i \boldsymbol{\mu}_i \otimes \boldsymbol{\mu}_i, \quad M_3 := \sum_{i=1}^{k} \omega_i \boldsymbol{\mu}_i \otimes \boldsymbol{\mu}_i \otimes \boldsymbol{\mu}_i, \tag{5.2}$$

satisfying the following non-degeneracy conditions:

Condition 1. *$\omega_i \ge 0 \; \forall \; 1 \le i \le k$, $\{\boldsymbol{\mu}_i \in \mathbb{R}^m\}_{i=1}^{k}$ are linearly independent, and $k \le m$.*

Algorithm 5.1 Whitening transformation

input A symmetric matrix $M_2 \in \mathbb{R}^{m \times m}$, a symmetric third-order tensor $M_3 \in \mathbb{R}^{m \times m \times m}$, and the target dimension k.

output A reduced third-order tensor $\mathcal{T} \in \mathbb{R}^{k \times k \times k}$ and a whitening transformation $W \in \mathbb{R}^{m \times k}$.

1: Compute $W := QD^{-1/2}$, where $Q \in \mathbb{R}^{m \times k}$ denotes the top-k orthonormal eigenvectors of M_2, and $D \in \mathbb{R}^{k \times k}$ is a diagonal matrix of the corresponding k positive eigenvalues.

2: Compute $\mathcal{T} := M_3(W, W, W)$.

As described in later sections, the core of our learning task is the problem of estimating $\{\omega_i\}_{i=1}^k$ and $\{\boldsymbol{\mu}_i\}_{i=1}^k$ from perturbed or noisy versions of M_2 and M_3, which we solve with the tensor decomposition method recently proposed by Anandkumar et al. [2012a], summarized below. Suppose the noiseless M_2 and M_3 are available, we first perform a *whitening step* on them, as outlined in Algorithm 5.1, to obtain a whitened, lower-dimensional tensor $\mathcal{T} \in \mathbb{R}^{k \times k \times k}$ and a whitening transformation $W \in \mathbb{R}^{m \times k}$ such that

$$\mathcal{T} \;:=\; M_3(W, W, W) \;=\; \sum_{i=1}^{k} \omega_i (W^\top \boldsymbol{\mu}_i)^{\otimes 3} = \sum_{i=1}^{k} \frac{1}{\sqrt{\omega_i}} \widetilde{\boldsymbol{\mu}}_i^{\otimes 3},$$

where the vectors $\widetilde{\boldsymbol{\mu}}_i := \sqrt{\omega_i} W^\top \boldsymbol{\mu}_i$ form an orthonormal basis of $\mathbb{R}^k$ because $I = W^\top M_2 W = \sum_{i=1}^k W^\top (\sqrt{\omega_i}\boldsymbol{\mu}_i)(\sqrt{\omega_i}\boldsymbol{\mu}_i)^\top W = \sum_{i=1}^k \widetilde{\boldsymbol{\mu}}_i \widetilde{\boldsymbol{\mu}}_i^\top$. Hence, the symmetric tensor $\mathcal{T}$ has a so-called *orthogonal decomposition*, which may not exist for general symmetric tensors. Then by Theorem 4.3 of [Anandkumar et al., 2012a], which establishes the following results under Condition 1:

1. the set of *robust eigenvectors* (c.f. Section 4.2 of [Anandkumar et al., 2012a]) of $\mathcal{T}$ correspond exactly to $\{\widetilde{\boldsymbol{\mu}}_i\}_{i=1}^k$;
2. the eigenvalue associated with $\widetilde{\boldsymbol{\mu}}_i$ is equal to $1/\sqrt{\omega_i},\ \forall\ 1 \le i \le k$;
3. if $(\mathbf{v}, \lambda)$ is a pair of robust eigenvector/eigenvalue of $\mathcal{T}$, then $\boldsymbol{\mu}_i = \lambda (W^\top)^\dagger \mathbf{v}$ for some $1 \le i \le k$, where $\dagger$ denotes the Moore-Penrose pseudo inverse;

we can reduce the original problem of recovering the structure in (5.2) into a robust tensor eigen-decomposition problem. Motivated by power iteration for matrix eigen computation, Anandkumar et al. [2012a] verify that starting from almost every vector $\boldsymbol{\theta}_0 \in \mathbb{R}^k$, the tensor power iteration

$$\boldsymbol{\theta}_t \;:=\; \frac{\mathcal{T}(I, \boldsymbol{\theta}_{t-1}, \boldsymbol{\theta}_{t-1})}{\|\mathcal{T}(I, \boldsymbol{\theta}_{t-1}, \boldsymbol{\theta}_{t-1})\|},$$

where $\|\cdot\|$ denotes the vector 2-norm, converges to some robust eigenvector of $\mathcal{T}$ at a quadratic rate, and therefore k successive applications of tensor power iteration with deflation result in all pairs of robust eigenvectors/eigenvalues.

In practice we almost never have the exact M_2 and M_3, but only noisy or perturbed versions $\widehat{M_2}$ and $\widehat{M_3}$, which are usually estimates from the data. Perturbation may destroy the tensor structure (5.2), so the reduced tensor $\widehat{\mathcal{T}}$ resulting from applying Algorithm 5.1 to $\widehat{M_2}$ and $\widehat{M_3}$ may no longer be orthogonally decomposable, hindering the subsequent robust tensor eigendecomposition. Nevertheless, Anandkumar et al. [2012a] demonstrate that if the perturbation $E := \widehat{\mathcal{T}} - \mathcal{T}$ is a symmetric tensor with a small operator norm defined as $\|E\| \;:=\; \sup_{\|\boldsymbol{\theta}\|=1} |E(\boldsymbol{\theta}, \boldsymbol{\theta}, \boldsymbol{\theta})|$,

Algorithm 5.2 Robust tensor power method

input A symmetric tensor $\mathcal{T} \in \mathbb{R}^{k\times k\times k}$, number of iterations $\mathtt{L}, \mathtt{N}$.

output the estimated eigenvector/eigenvalue pair; the deflated tensor.

1: **for** $\tau = 1$ **to** $\mathtt{L}$ **do**
2: Draw $\boldsymbol{\theta}_0^{(\tau)}$ uniformly at random from the unit sphere in $\mathbb{R}^k$.
3: **for** $t = 1$ **to** $\mathtt{N}$ **do**
4: $$\boldsymbol{\theta}_t^{\tau} := \frac{\mathcal{T}(I, \boldsymbol{\theta}_{t-1}^{(\tau)}, \boldsymbol{\theta}_{t-1}^{(\tau)})}{\|\mathcal{T}(I, \boldsymbol{\theta}_{t-1}^{(\tau)}, \boldsymbol{\theta}_{t-1}^{(\tau)})\|}.$$
5: **end for**
6: **end for**
7: Let $\tau^* := \arg\max_{1\le\tau\le\mathtt{L}} \mathcal{T}(\boldsymbol{\theta}_{\mathtt{N}}^{(\tau)}, \boldsymbol{\theta}_{\mathtt{N}}^{(\tau)}, \boldsymbol{\theta}_{\mathtt{N}}^{(\tau)})$.
8: Do $\mathtt{N}$ power iteration updates (Line 4) starting from $\boldsymbol{\theta}_{\mathtt{N}}^{(\tau^*)}$ to obtain $\widehat{\boldsymbol{\theta}}$, and set $\hat{\lambda} := \mathcal{T}(\widehat{\boldsymbol{\theta}}, \widehat{\boldsymbol{\theta}}, \widehat{\boldsymbol{\theta}})$
9: **return** the estimated eigenvector/eigenvalue pair $(\widehat{\boldsymbol{\theta}}, \hat{\lambda})$; the deflated tensor $\mathcal{T} - \hat{\lambda}\widehat{\boldsymbol{\theta}}^{\otimes 3}$.

then k successive applications of some *randomized* tensor power iteration coupled with deflation yield accurate estimates of all robust eigenvector/eigenvalue pairs with high probability. More precisely, they propose *the Robust tensor power method* outlined in Algorithm 5.2, which employs multiple random restarts, and provide a theoretical guarantee on its robustness against the input perturbation:

Theorem 1. *(Theorem 5.1 of [Anandkumar et al., 2012a]) Let $\widehat{\mathcal{T}} = \mathcal{T} + E \in \mathbb{R}^{k\times k\times k}$, where $\mathcal{T}$ is a symmetric tensor with orthogonal decomposition $\mathcal{T} = \sum_{i=1}^k \lambda_i \mathbf{v}_i^{\otimes 3}$ where each $\lambda_i > 0$, $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k\}$ is an orthonormal basis, and E has operator norm $\epsilon := \|E\|$. Define $\lambda_{\min} := \min(\{\lambda_i\}_{i=1}^k)$ and $\lambda_{\max} := \max(\{\lambda_i\}_{i=1}^k)$. There exists universal constants $c_1, c_2, c_3 > 0$ such that the following holds. Pick any $\eta \in (0, 1)$, and suppose*

$$\epsilon \le c_1 \cdot \frac{\lambda_{\min}}{k}, \qquad \mathsf{N} \ge c_2 \cdot \big(\log(k) + \log\log(\lambda_{\max}/\epsilon)\big), \qquad \text{and}$$

$$\sqrt{\frac{\ln(\mathsf{L}/\log_2(\frac{k}{\eta}))}{\ln(k)}} \cdot \left(1 - \frac{\ln(\ln(\mathsf{L}/\log_2(\frac{k}{\eta}))) + c_3}{4\ln(\mathsf{L}/\log_2(\frac{k}{\eta}))} - \sqrt{\frac{\ln(8)}{\ln(\mathsf{L}/\log_2(\frac{k}{\eta}))}}\right) \ge 1.02\left(1 + \sqrt{\frac{\ln(4)}{\ln(k)}}\right).$$

(Note that the condition on L holds with $\mathsf{L} = \text{poly}(k)\log(1/\eta)$.) Suppose that Algorithm 5.2 is iteratively called k times with numbers of iterations L and N, where the input tensor is $\widehat{\mathcal{T}}$ in the first call, and in each subsequent call, the input tensor is the deflated tensor returned by the previous call. Let $(\widehat{\mathbf{v}}_1, \hat{\lambda}_1), (\widehat{\mathbf{v}}_2, \hat{\lambda}_2), \ldots, (\widehat{\mathbf{v}}_k, \hat{\lambda}_k)$ be the sequence of estimated eigenvector/eigenvalue pairs returned in these k calls. With probability at least $1 - \eta$, there exists a permutation ρ on $\{1, \ldots, k\}$ such that

$$\|\mathbf{v}_{\rho(j)} - \widehat{\mathbf{v}}_j\| \le 8\epsilon/\lambda_{\rho(j)}, \quad |\lambda_{\rho(j)} - \hat{\lambda}_j| \le 5\epsilon, \ \forall 1 \le j \le k, \quad \text{and} \quad \|\mathcal{T} - \sum_{j=1}^k \hat{\lambda}_j \widehat{\mathbf{v}}_j^{\otimes 3}\| \le 55\epsilon.$$

This result, together with existing perturbation theory regarding the whitening procedure (e.g., Appendix C.1 of [Anandkumar et al., 2013]), allow us to translate the perturbations in $\widehat{M}_2$ and $\widehat{M}_3$ into the estimation errors in ω_i's and $\boldsymbol{\mu}_i$'s, guaranteeing accurate estimation under small input perturbation.

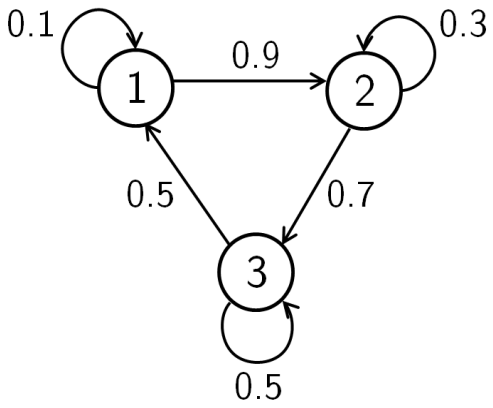


Figure 5.1: Running example of Markov chain with three states

5.2 Learning from Non-sequence Data

We first describe a generative process of non-sequence data for first-order Markov models and demonstrate how to apply tensor decomposition methods to perform consistent learning. Then we extend these ideas to hidden Markov models and provide theoretical guarantees on the sample complexity of the proposed learning algorithm. For notational conveniences we define the following vector-matrix cross product $\otimes_{d\in\{1,2,3\}}$: $(\mathbf{v} \otimes_1 M)_{ijk} := v_i(M)_{jk}, (\mathbf{v} \otimes_2 M)_{ijk} = v_j(M)_{ik}, (\mathbf{v} \otimes_3 M)_{ijk} = v_k(M)_{ij}$. For a matrix M we denote by M_i its i-th column.

5.2.1 First-order Markov Models

Let $P \in [0,1]^{m\times m}$ be the transition probability matrix of a discrete, first-order, ergodic Markov chain with m states and a unique stationary distribution $\boldsymbol{\pi}$. Let P be of full rank and $\mathbf{1}^\top P = \mathbf{1}^\top$. To give a high-level idea of what makes it possible to learn P from non-sequence data, we use the simple Markov chain with three states shown in Figure 5.1 as our running example, demonstrating step by step how to extend from a very restrictive generative setting of the data to a reasonably general setting, along with the assumptions made to allow consistent parameter estimation. In the usual setting where we have sequences of observations, say $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots\}$ with parenthesized superscripts denoting time, it is straightforward to consistently estimate P. We simply calculate the empirical frequency of consecutive pairs of states:

$$\widehat{P_{ij}} := \frac{\sum_t \mathbb{1}(\mathbf{x}^{(t+1)} = i, \mathbf{x}^{(t)} = j)}{\sum_t \mathbb{1}(\mathbf{x}^{(t)} = j)}.$$

Alternatively, suppose for each state j, we have an *i.i.d. sample* of its immediate next state $D_j := \{\mathbf{x}_1^{(1)}, \mathbf{x}_2^{(1)}, \ldots \mid \mathbf{x}^{(0)} = j\}$, where subscripts are data indices. Consistent estimation in this case is also easy: the empirical distribution of D_j consistently estimates P_j, the j-th column of P. For example, the Markov chain in Figure 5.1 may produce the following three samples,

whose empirical distributions estimate the three columns of P respectively:

$$\begin{aligned} D_1 &= \{2,1,2,2,2,2,2,2,2,2\} \Rightarrow \widehat{P_1} = [0.1\ \ 0.9\ \ 0.0]^\top, \\ D_2 &= \{3,3,2,3,2,3,3,2,3,3\} \Rightarrow \widehat{P_2} = [0.0\ \ 0.3\ \ 0.7]^\top, \\ D_3 &= \{1,1,3,1,3,3,1,3,3,1\} \Rightarrow \widehat{P_3} = [0.5\ \ 0.0\ \ 0.5]^\top. \end{aligned}$$

A nice property of these estimates is that, unlike in the sequential setting, they do not depend on any particular ordering of the observations in each set. Nevertheless, such data are not quite non-sequenced because all observations are made at exactly the next time step. We thus consider the following generalization: for each state j, we have $D_j := \{\mathbf{x}_1^{(t_1)}, \mathbf{x}_2^{(t_2)}, \ldots \mid \mathbf{x}^{(0)} = j\}$, i.e., independent samples of states drawn at *unknown* future times $\{t_1, t_2, \ldots\}$. For example, our data in this setting might be

$$\begin{aligned} D_1 &= \{2,1,2,3,2,3,3,2,2,3\}, \\ D_2 &= \{3,3,2,3,2,1,3,2,3,1\}, \\ D_3 &= \{1,1,3,1,2,3,2,3,3,2\}. \end{aligned} \tag{5.3}$$

Obviously it is hard to extract information about P from such data. However, if we assume that the unknown times $\{t_i\}$ are i.i.d. random variables following some distribution independent of the initial state j, it can then be easily shown that D_j's empirical distribution consistently estimates T_j, the j-th column of the the *expected transition probability matrix* $T := \mathbb{E}_t[P^t]$:

$$\begin{aligned} D_1 &= \{2,1,2,3,2,3,3,2,2,3\} \Rightarrow \widehat{T_1} = [0.1\ \ 0.5\ \ 0.4]^\top, \\ D_2 &= \{3,3,2,3,2,1,3,2,3,1\} \Rightarrow \widehat{T_2} = [0.2\ \ 0.3\ \ 0.5]^\top, \\ D_3 &= \{1,1,3,1,2,3,2,3,3,2\} \Rightarrow \widehat{T_3} = [0.3\ \ 0.3\ \ 0.4]^\top. \end{aligned}$$

In general there exist many P's that result in the same T. Therefore, as detailed later, we make a specific distributional assumption on $\{t_i\}$ to enable unique recovery of the transition matrix P from T (Assumption A.1). Next we consider a further generalization, where the unknowns are not only the time stamps of the observations, but also the initial state j. In other words, we only know each set was generated from the same initial state, but do not know the actual initial state. In this case, the empirical distributions of the sets consistently estimate the columns of T in some *unknown permutation* Π:

$$\begin{aligned} D_{\Pi(3)} &= \{1,1,3,1,2,3,2,3,3,2\} \Rightarrow \widehat{T_{\Pi(3)}} = [0.3\ \ 0.3\ \ 0.4]^\top. \\ D_{\Pi(2)} &= \{3,3,2,3,2,1,3,2,3,1\} \Rightarrow \widehat{T_{\Pi(2)}} = [0.2\ \ 0.3\ \ 0.5]^\top, \\ D_{\Pi(1)} &= \{2,1,2,3,2,3,3,2,2,3\} \Rightarrow \widehat{T_{\Pi(1)}} = [0.1\ \ 0.5\ \ 0.4]^\top. \end{aligned}$$

In order to be able to identify Π, we will again resort to randomness and assume the unknown initial states are random variables following a certain distribution (Assumption A.2) so that the data carry information about Π. Finally, we generalize from a single unknown initial state to an unknown *initial state distribution*, where each set of observations $D := \{\mathbf{x}_1^{(t_1)}, \mathbf{x}_2^{(t_2)}, \ldots \mid \boldsymbol{\pi}^{(0)}\}$

consists of independent samples of states drawn at random times from some unknown initial state distribution $\boldsymbol{\pi}^{(0)}$. For example, the data may look like:

$$\begin{aligned} D_{\boldsymbol{\pi}_1^{(0)}} &= \{1,3,3,1,2,3,2,3,3,2\}, \\ D_{\boldsymbol{\pi}_2^{(0)}} &= \{3,1,2,3,2,1,3,2,3,1\}, \\ D_{\boldsymbol{\pi}_3^{(0)}} &= \{2,1,2,3,3,3,3,1,2,3\}, \\ &\vdots \end{aligned}$$

With this final generalization, most would agree that the generated data are non-sequenced and that the generative process is flexible enough to model some of the real-world situations described in Chapter 1. However, simple estimation with empirical distributions no longer works because each set may now contain observations from multiple initial states. This is where we take advantage of the tensor decomposition framework outlined in Section 5.1, which requires proper assumptions on the initial state distribution $\boldsymbol{\pi}^{(0)}$ (Assumption A.3).

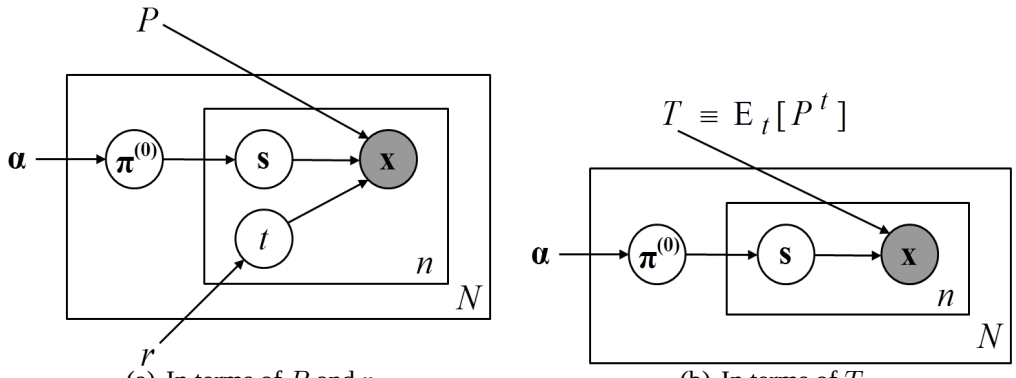
More formally, the aforementioned ideas motivate the following three assumptions:

- **Assumption A.1.** The missing times $\{t_i\}$ are i.i.d. according to a Geometric distribution. This makes it possible to uniquely recover the transition matrix P from the expected transition matrix T.
- **Assumption A.2.** The stationary distribution $\boldsymbol{\pi}$ of the Markov chain is such that $\pi_i \neq \pi_j, i \neq j$. This allows recovering the correct column permutation of T.
- **Assumption A.3.** The initial state distribution $\boldsymbol{\pi}^{(0)}$ is a random quantity following a Dirichlet distribution, and $\mathbb{E}[\boldsymbol{\pi}^{(0)}] = \boldsymbol{\pi}$, the stationary distribution. This allows the use of tensor decomposition methods.

Now we are ready to give the definition of our entire generative process. Assume we have N sets of non-sequence data each containing n observations, and each set of observations $\{\mathbf{x}_i\}_{i=1}^n$ were independently generated by the following:

- Draw an initial distribution
  $\boldsymbol{\pi}^{(0)} \sim \mathsf{Dirichlet}(\boldsymbol{\alpha})$, (Assumption A.3)
  $\mathbb{E}[\boldsymbol{\pi}^{(0)}] = \boldsymbol{\alpha}/(\sum_{i=1}^m \alpha_i) = \boldsymbol{\pi}, \quad \pi_i \neq \pi_j \; \forall \; i \neq j.$ (Assumption A.2)
- For $i = 1, \ldots, n$,
  - Draw a discrete time
    $t_i \sim \mathsf{Geometric}(r), \; t_i \in \{1,2,3,\ldots\}.$ (Assumption A.1)
  - Draw an initial state
    $\mathbf{s}_i \sim \mathsf{Multinomial}(\boldsymbol{\pi}^{(0)}), \; \mathbf{s}_i \in \{0,1\}^m.$
  - Draw an observation
    $\mathbf{x}_i \sim \mathsf{Multinomial}(P^{t_i}\mathbf{s}_i), \; \mathbf{x}_i \in \{0,1\}^m.$

As mentioned earlier, our generative process captures some characteristics of real-world situations. First, all the data points in the same set share the same initial state distribution but can have different initial states; the initial state distribution varies across different sets and yet centers around the stationary distribution of the Markov chain. As mentioned in the beginning of this



(a) In terms of P and r (b) In terms of T

Figure 5.2: Graphical models of the data generative process for first-order Markov models

chapter, this may be achieved in biological studies by running multiple experiments with different input stimuli, so the data collected in the same experiment can be assumed to have the same initial state distribution. Second, each data point is drawn from an independent trajectory of the Markov chain, a similar situation in the modeling of galaxies or Alzheimer's, and random time steps could be used to compensate for individual variations in speed: a small/large t_i corresponds to a slowly/fast evolving individual object. Finally, the geometric distribution can be interpreted as an overall measure of the magnitude of speed variation: a large success probability r would result in many small t_i's, meaning that most objects evolve at similar speeds, while a small r would lead to t_i's taking a wide range of values, indicating a large speed variation.

Figure 5.2(a) shows the graphical model of our generative process. Interestingly, this graphical model is very similar to the widely-used topic model Latent Dirichlet Allocation (LDA) [Blei et al., 2003]. In fact, by summing out the random time t, we obtain a graphical model that depends only on the expected transition probability matrix T and, as shown in Figure 5.2(b), has exactly the same structure as LDA. More specifically, we can view a set of non-sequence data points as a document generated by an LDA model, where each $\mathbf{x}_i$ corresponds to a word, $\mathbf{s}_i$ to a topic, the expected transition matrix T to the word-topic matrix, the initial distribution $\boldsymbol{\pi}^{(0)}$ to the topic distribution of the document, and the stationary distribution $\boldsymbol{\pi}$ to the overall topic proportions. Such a structural equivalence to LDA allows us to take advantage of recent advances in spectral learning [Anandkumar et al., 2012a] with rigorous guarantees on parameter estimation. However, our generative process has a critically distinct property: *the words are the topics, both of which correspond to states*, and as a result, unlike most topic models, is NOT invariant to column permutations of the word-topic matrix. We thus need Assumption A.2 to be able to recover the correct permutation.

Now we describe our spectral learning algorithm, which consists of three main steps:

1. Compute certain low-order moments of the data;
2. Perform tensor decomposition of the empirical moments;
3. Recover model parameters from the factors given by tensor decomposition.

The high-level idea is that according to our generative process, certain low-order moments of the data have the tensor structure (5.2) with the factors being the Markov model parameters, so we

can use the tensor decomposition method in Section 5.1 to extract the model parameters from the moments. The following theorem gives the desired low-order moments and their structure:

Theorem 2. *Let $\alpha_0 := \sum_i \alpha_i$. Define the expected transition probability matrix $T := \mathbb{E}_t[P^t] = rP(I-(1-r)P)^{-1}$ and*

$$\begin{aligned} C_2 &:= \mathbb{E}[\mathbf{x}_1 \otimes \mathbf{x}_2], \\ C_3 &:= \mathbb{E}[\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \mathbf{x}_3], \\ M_2 &:= (\alpha_0+1)C_2 - \alpha_0 \mathbb{E}[\mathbf{x}_1] \otimes \mathbb{E}[\mathbf{x}_1], \\ M_3 &:= \tfrac{(\alpha_0+2)(\alpha_0+1)}{2} C_3 - \tfrac{(\alpha_0+1)\alpha_0}{2} \sum_{d=1}^{3} \mathbb{E}[\mathbf{x}_1] \otimes_d C_2 + \alpha_0^2 \mathbb{E}[\mathbf{x}_1]^{\otimes 3}. \end{aligned}$$

Then the following holds:

$$\begin{aligned} \mathbb{E}[\mathbf{x}_1] &= \boldsymbol{\pi}, \\ C_2 &= \tfrac{1}{\alpha_0+1} T \mathsf{diag}(\boldsymbol{\pi}) T^\top + \tfrac{\alpha_0}{\alpha_0+1} \boldsymbol{\pi} \otimes \boldsymbol{\pi}, \\ C_3 &= \tfrac{2}{(\alpha_0+2)(\alpha_0+1)} \sum_i \pi_i T_i^{\otimes 3} + \tfrac{\alpha_0}{\alpha_0+2} \sum_{d=1}^{3} \boldsymbol{\pi} \otimes_d C_2 - \tfrac{2\alpha_0^2}{(\alpha_0+2)(\alpha_0+1)} \boldsymbol{\pi}^{\otimes 3}, \\ M_2 &= T \mathsf{diag}(\boldsymbol{\pi}) T^\top, \\ M_3 &= \sum_i \pi_i T_i^{\otimes 3}. \end{aligned}$$

We call M_2 and M_3 the adjusted moments because they are computed from the raw moments $\mathbb{E}[\mathbf{x}_1]$, C_2 and C_3. Because of the connection of our generative process to LDA, the proof of this theorem, given in Appendix B.1.1, mainly uses existing results in spectral learning of LDA [Anandkumar et al., 2013], which rely on the special structure in the moments of the Dirichlet distribution (Assumption A.3). According to Theorem 2, it is clear that the adjusted moments M_2 and M_3 have the desired tensor structure (5.2). Assuming α_0 is known, we can form estimates $\widehat{M_2}$ and $\widehat{M_3}$ by computing empirical moments from the data. Note that the $\mathbf{x}_i$'s are exchangeable, so we can use all pairs and triples of data points to compute the estimates. Since the tensor decomposition method may return $\widehat{T}$ under any column permutation, we need to recover the correct matching between its rows and columns. To do so, we note that the $\widehat{\boldsymbol{\pi}}$ returned by the tensor decomposition method undergoes the same permutation as the columns of $\widehat{T}$, and because all π_i's have different values by Assumption A.2, we may recover the correct matching by sorting both the returned $\widehat{\boldsymbol{\pi}}$ and the mean $\bar{\boldsymbol{\pi}}$ of all data.

The last issue is recovering P from $\widehat{T}$, for which we make the distributional assumption A.1 on the random times $\{t_i\}$. With such an assumption, we have reduced the search space from all possible mappings between T and P to one single parameter, the success probability r. Nevertheless, recovering P and r is in general still difficult even when the exact T is available, because multiple choices of P and r may result in the same T. In practical situations, however, we can often assume the underlying transition probability matrix P has some zero entries, e.g., when the true Markov chain is based on a graph, or when the state transition is under some external or physical constraint. With this extra assumption, we prove that unique recovery is possible in the population case:

Theorem 3. *Let P^*, r^*, T^* and $\boldsymbol{\pi}^*$ denote the true values of the transition probability matrix, the success probability, the expected transition matrix, and the stationary distribution, respectively. Assume that P^* is ergodic and of full rank, and $P^*_{ij} = 0$ for some arbitrary i and j. Let $\mathcal{S} := \{\lambda/(\lambda-1) \mid \lambda$ is a real negative eigenvalue of $T^*\} \cup \{0\}$. Then the following holds:*

- $0 \leq \max(\mathcal{S}) < r^* \leq 1$.
- *For all $r \in (0,1] \setminus \mathcal{S}$, $P(r) := (rI + (1-r)T^*)^{-1}T^*$ is well-defined and satisfies*
  - $\mathbf{1}^\top P(r) = \mathbf{1}^\top$, $P(r)\boldsymbol{\pi}^* = \boldsymbol{\pi}^*$, $P^* = P(r^*)$.
  - $P(r)_{ij} \geq 0 \ \forall\, i,j \quad \Longleftrightarrow \quad r \geq r^*$.

That is, $P(r)$ is a stochastic matrix if and only if $r \geq r^*$.

The proof is in Appendix B.2, and the key step is to show that the zero entries in P^* become negative when $r < r^*$. According to this theorem, binary search on $(0,1]$ suffices to recover r^* and P^* from T^*. However, it may fail when we replace T^* by an estimate $\widehat{T}$ because even $\widehat{P}(r^*)$ might contain negative values. A more practical estimation procedure is the following: for each value of r in a decreasing sequence starting from 1, we project $\widehat{P}(r) := (rI + (1-r)\widehat{T})^{-1}\widehat{T}$ onto the space of stochastic matrices and record the projection distance. Then starting from 1, we search in the sequence of projection distances for the first sudden increase, and take the corresponding value of r and (projected) $\widehat{P}(r)$ as the final estimates. The idea is that $\widehat{P}(r)$ should be close to the space of stochastic matrices when $r \geq r^*$, but starts to move away by having negative entries as r gets smaller than r^*. It is easy to see the estimates produced by such a procedure converge to r^* and P^* as $\widehat{T}$ gets closer to T^* and the discrete search space for r becomes denser. However, a formal convergence rate is yet to be identified. Also, while lacking a formal proof, we suspect that the more zero entries P^* has, the easier it is to estimate r^* because $\widehat{P}(r)$ for $r < r^*$ would be further away from the space of stochastic matrices by having more negative entries. Finally, although sparsity is sufficient for unique recovery of P^* and r^*, more investigation is needed to clarify whether it is also necessary.

We summarize the entire learning procedure in Algorithm 5.3, which assumes the true r and α_0 are known. Because the empirical moments are consistent estimators for the true moments and the tensor decomposition method returns accurate estimates under small input perturbation, we can conclude that the estimates output by Algorithm 5.3 will converge (with high probability) to the true quantities as the sample size N increases. Sample complexity bounds can be obtained with techniques similar to those for Theorem 5 in the next section.

Remarks on Identifiability. Unlike in Chapter 3, where some properties of the true dynamic model are not identifiable from non-sequence data, our proposed method here guarantees consistent parameter estimation. The main reason, as one can imagine, is that non-identifiability is assumed away in our data generative model. For example, consider the following transition probability matrix and its transpose:

$$P = \begin{bmatrix} q & 0 & 1-q \\ 1-q & q & 0 \\ 0 & 1-q & q \end{bmatrix}, \quad P^\top = \begin{bmatrix} q & 1-q & 0 \\ 0 & q & 1-q \\ 1-q & 0 & q \end{bmatrix},$$

which are ergodic for $0 < q < 1$. The sequences of observations generated by these two Markov chains will be in approximately opposite directions of time, and therefore causes non-identifiability when time information is missing. However, such Markov chains are excluded by

Algorithm 5.3 Tensor decomposition method for learning Markov chains from non-sequence data

input N sets of non-sequence data points, the success probability r, the Dirichlet parameter α_0, and numbers of iterations L and N.

output Estimates $\widehat{\boldsymbol{\pi}}$ and $\widehat{P}$.

1: Compute empirical averages $\widehat{\boldsymbol{\pi}}, \widehat{C_2}$ and $\widehat{C_3}$.

2: Compute $\widehat{M_2}$ and $\widehat{M_3}$.

3: Run Algorithm 5.1 on $\widehat{M_2}$ and $\widehat{M_3}$ with target dimension m to obtain a symmetric tensor $\widehat{\mathcal{T}} \in \mathbb{R}^{m \times m \times m}$ and a whitening transformation $\widehat{W} \in \mathbb{R}^{m \times m}$.

4: Run Algorithm 5.2 m times each with numbers of iterations L and N, the input tensor in the first run set to $\widehat{\mathcal{T}}$ and in each subsequent run set to the deflated tensor returned by the previous run, resulting in m pairs of eigenvalue/eigenvector $\{(\hat{\lambda}_i, \widehat{\mathbf{v}}_i)\}_{i=1}^m$.

5: Match $\{(\hat{\lambda}_i, \widehat{\mathbf{v}}_i)\}_{i=1}^m$ with observation symbols by sorting $\{\hat{\lambda}_i\}_{i=1}^m$ and $\{\hat{\pi}_i^{-1/2}\}_{i=1}^m$.

6: Obtain estimate of the transition probability matrix:

$$\widehat{P} \;:=\; (rI + (1-r)\widehat{T})^{-1}\widehat{T},$$

where $\widehat{T} := (\widehat{W}^\top)^\dagger \widehat{V}\widehat{\Lambda}, \widehat{V} := [\widehat{\mathbf{v}}_1 \cdots \widehat{\mathbf{v}_m}]$, and $\widehat{\Lambda} := \mathsf{diag}([\hat{\lambda}_1 \cdots \hat{\lambda}_m]^\top)$.

7: (Optional) Project $\widehat{P}$ onto the space of stochastic matrices.

our assumption that the stationary distribution $\boldsymbol{\pi}$ satisfies $\pi_i \neq \pi_j \; \forall \; i \neq j$ because $P\mathbf{1} = \mathbf{1}$. More generally, all Markov chains with a doubly-stochastic transition probability matrix are excluded by that assumption because their stationary distributions are the uniform distribution. Another potentially non-identifiable class of models are the *time-reversible Markov chains* [Chapter 6.5 Grimmett and Stirzaker, 2001], where the transition probability P and the stationary distribution $\boldsymbol{\pi}$ satisfy

$$\pi_j P_{ij} = \pi_i P_{ji} \quad \forall \; i, j.$$

A well-known result is that the time direction of such a Markov chain cannot be distinguished from the reverse direction after it fully mixes. According to our generative assumption, each of the non-sequence data sets contains, with a positive probability, observations made *before* the Markov chain fully mixes. Those observations make it possible to eliminate the non-identifiability of time direction even in the case of time reversible models.

5.2.2 Hidden Markov Models

Equipped with the intuition and strategies for learning first-order Markov models, we are now ready to handle the more complicated hidden Markov models. As detailed later, it turns out that both the generative process and the learning procedure for HMMs are quite similar to those for Markov chains, with the main distinction being *two* applications of tensor decomposition, where the extra one is due to the mapping from the hidden state space to the observation space.

Let P and $\boldsymbol{\pi}$ now be defined over the hidden discrete state space of cardinality k and have the same properties as the first-order Markov model in Section 5.2.1. Again, we assume the non-sequence data consists of N sets of data points, where each set now contains continuous

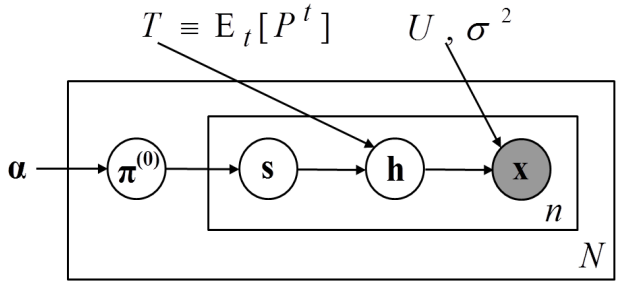


Figure 5.3: Graphical model of the data generative process for HMMs

observations in $\mathbb{R}^m$. The generative process for each set is almost identical to and therefore shares the same interpretation with the one for Markov chains, except for an extra mapping from the discrete hidden state space to a continuous observation space:

- Draw an initial hidden state distribution
  $\boldsymbol{\pi}^{(0)} \sim \mathsf{Dirichlet}(\boldsymbol{\alpha})$,
  $\mathbb{E}[\boldsymbol{\pi}^{(0)}] = \boldsymbol{\alpha}/(\sum_{i=1}^{k} \alpha_i) = \boldsymbol{\pi}, \quad \pi_i \neq \pi_j \; \forall \; i \neq j.$
- For $i = 1, \ldots, n$,
  - Draw a discrete time
    $t_i \sim \mathsf{Geometric}(r), \; t_i \in \{1, 2, 3, \ldots\}.$
  - Draw an initial hidden state
    $\mathbf{s}_i \sim \mathsf{Multinomial}(\boldsymbol{\pi}^{(0)}), \; \mathbf{s}_i \in \{0, 1\}^k.$
  - Draw a hidden state at time t_i
    $\mathbf{h}_i \sim \mathsf{Multinomial}(P^{t_i}\mathbf{s}_i), \; \mathbf{h}_i \in \{0, 1\}^k.$
  - Draw an observation:
    $$\mathbf{x}_i \; = \; U\mathbf{h}_i + \boldsymbol{\epsilon}_i,$$
    where $U \in \mathbb{R}^{m \times k}$ denotes a rank-k matrix of mean observation vectors for the k hidden states, and the random noise vectors $\boldsymbol{\epsilon}_i$'s are i.i.d satisfying $\mathbb{E}[\boldsymbol{\epsilon}_i] = \mathbf{0}$, $\mathrm{Var}[\boldsymbol{\epsilon}_i] = \sigma^2 I$, and $\mathbb{E}[(\boldsymbol{\epsilon}_i)_d^3] = 0, 1 \leq d \leq m$.

A graphical model representation is in Figure 5.3. Compared with the graphical model in Figure 5.2(b), the observation model here is a mixture distribution rather than a discrete state, which makes learning more complicated, but still manageable. For simplicity we require a common spherical noise covariance, but our method can be easily modified to allow different spherical covariances $\sigma_j^2 I$ for different hidden states (c.f. Section 3.2 of [Anandkumar et al., 2012a]). In Section 5.4 we will discuss possible ways to handle more general noise covariances. Another important requirement on the observation noise is zero skewness, i.e., zero third-order moment. As discussed later, we need this condition to ensure that certain moments have the desired tensor structure. While zero skewness rules out some potentially useful observation models, we discuss in Section 5.4 how to handle one interesting class of skewed observation noise: discrete observations.

As in Section 5.2.1, we develop our spectral learning algorithm around the tensor structure (5.2) in low-order moments of the data:

Theorem 4. *Let $\alpha_0 := \sum_i \alpha_i$. Define the expected hidden state transition matrix $T := \mathbb{E}_t[P^t] = rP(I-(1-r)P)^{-1}$ and*

$$\begin{aligned}
V_1 &:= \mathbb{E}[\mathbf{x}_1], \\
V_2 &:= \mathbb{E}[\mathbf{x}_1 \otimes \mathbf{x}_1], \\
V_3 &:= \mathbb{E}[\mathbf{x}_1 \otimes^3], \\
M_2' &:= V_2 - \sigma^2 I, \\
M_3' &:= V_3 - \sum_{d=1}^{3} V_1 \otimes_d (\sigma^2 I), \\
C_2 &:= \mathbb{E}[\mathbf{x}_1 \otimes \mathbf{x}_2], \\
C_3 &:= \mathbb{E}[\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \mathbf{x}_3], \\
M_2 &:= (\alpha_0+1)C_2 - \alpha_0 V_1 \otimes V_1, \\
M_3 &:= \tfrac{(\alpha_0+2)(\alpha_0+1)}{2} C_3 - \tfrac{(\alpha_0+1)\alpha_0}{2} \sum_{d=1}^{3} V_1 \otimes_d C_2 + \alpha_0^2 V_1^{\otimes 3}.
\end{aligned}$$

Then the following holds:

$$\begin{aligned}
V_1 &= U\boldsymbol{\pi}, \\
V_2 &= U\mathsf{diag}(\boldsymbol{\pi})U^\top + \sigma^2 I, \\
V_3 &= \sum_i \pi_i U_i^{\otimes 3} + \sum_{d=1}^{3} V_1 \otimes_d (\sigma^2 I), \\
M_2' &= U\mathsf{diag}(\boldsymbol{\pi})U^\top, \\
M_3' &= \sum_i \pi_i U_i^{\otimes 3}, \\
C_2 &= \tfrac{1}{\alpha_0+1} UT\mathsf{diag}(\boldsymbol{\pi})(UT)^\top + \tfrac{\alpha_0}{\alpha_0+1} V_1 \otimes V_1, \\
C_3 &= \tfrac{2}{(\alpha_0+2)(\alpha_0+1)} \sum_i \pi_i (UT)_i^{\otimes 3} + \tfrac{\alpha_0}{\alpha_0+2} \sum_{d=1}^{3} V_1 \otimes_d C_2 - \tfrac{2\alpha_0^2}{(\alpha_0+2)(\alpha_0+1)} V_1^{\otimes 3} \\
M_2 &= UT\mathsf{diag}(\boldsymbol{\pi})(UT)^\top, \\
M_3 &= \sum_i \pi_i (UT)_i^{\otimes 3}.
\end{aligned}$$

The proof is in Appendix B.1.2. This theorem suggests that HMMs require *two* applications of the tensor decomposition method: one on the adjusted cross moments M_2 and M_3, as in learning Markov chains, for extracting the matrix product UT, and the other on the adjusted covariance M_2' and tri-variance M_3' for extracting the mean observation vectors U. Just as the low-order moments of first-order Markov models in Theorem 2 are similar to those of LDA, the tensor structures here also have connections to other latent variable models. First, M_2' and M_3' are reminiscent of mixtures of spherical Gaussians. Indeed, each set of observations can be viewed as independent samples drawn from a mixture model where each mixture component is a distribution with a spherical covariance and the mixture weights are $T\boldsymbol{\pi}^{(0)}$, $\boldsymbol{\pi}^{(0)}$ denoting

Algorithm 5.4 Tensor decomposition method for learning HMM from non-sequence data

input N sets of non-sequence data points, the success probability r, the Dirichlet parameter α_0, the number of hidden states k, and numbers of iterations L and N.

output Estimates $\widehat{\boldsymbol{\pi}}$, $\widehat{P}$ and $\widehat{U}$ possibly under permutation of state labels.

1: Compute empirical averages $\widehat{V}_1, \widehat{V}_2, \widehat{V}_3, \widehat{C_2}, \widehat{C_3}$, and $\widehat{\sigma^2} := \lambda_{\min}(\widehat{V}_2 - \widehat{V}_1\widehat{V}_1^{\top})$.

2: Compute $\widehat{M_2}, \widehat{M_3}, \widehat{M_2'}, \widehat{M_3'}$

3: Run Algorithm 5.1 on $\widehat{M_2}$ and $\widehat{M_3}$ with the number of hidden states k to obtain a symmetric tensor $\widehat{\mathcal{T}} \in \mathbb{R}^{k\times k\times k}$ and a whitening transformation $\widehat{W} \in \mathbb{R}^{m\times k}$.

4: Run Algorithm 5.2 k times each with numbers of iterations L and N, the input tensor in the first run set to $\widehat{\mathcal{T}}$ and in each subsequent run set to the deflated tensor returned by the previous run, resulting in k pairs of eigenvalue/eigenvector $\{(\hat{\lambda}_i, \widehat{\mathbf{v}}_i)\}_{i=1}^k$.

5: Repeat Steps 4 and 5 on $\widehat{M_2'}$ and $\widehat{M_3'}$ to obtain $\widehat{\mathcal{T}'}, \widehat{W'}$ and $\{(\hat{\lambda'}_i, \widehat{\mathbf{v}'_i})\}_{i=1}^k$.

6: Match $\{(\hat{\lambda}_i, \widehat{\mathbf{v}}_i)\}_{i=1}^k$ with $\{(\hat{\lambda'}_i, \widehat{\mathbf{v}'_i})\}_{i=1}^k$ by sorting $\{\hat{\lambda}_i\}_{i=1}^k$ and $\{\hat{\lambda'}_i\}_{i=1}^k$.

7: Obtain estimates of HMM parameters:

$$\widehat{UT} := (\widehat{W})^{\dagger}\widehat{V}\widehat{\Lambda}, \quad \widehat{U} := (\widehat{W'}^{\top})^{\dagger}\widehat{V'}\widehat{\Lambda'},$$
$$\widehat{P} := (r\widehat{U} + (1-r)\widehat{UT})^{\dagger}\widehat{UT}, \quad \widehat{\boldsymbol{\pi}} := [\hat{\lambda}_1^{-2} \cdots \hat{\lambda}_k^{-2}]^{\top},$$

where $\widehat{V} := [\widehat{\mathbf{v}}_1 \cdots \widehat{\mathbf{v}}_k], \widehat{\Lambda} := \mathsf{diag}([\hat{\lambda}_1 \cdots \hat{\lambda}_k]^{\top})$; $\widehat{V'}$ and $\widehat{\Lambda'}$ are defined in the same way.

8: (Optional) Project $\widehat{\boldsymbol{\pi}}$ onto the simplex and $\widehat{P}$ onto the space of stochastic matrices.

the initial hidden state distribution of that set. Therefore, when forming estimates for M_2' and M_3', which require an estimate for the noise variance σ^2, we may use the existing result for spherical Gaussians (Theorem 3.2 in [Anandkumar et al., 2012a]) to obtain an estimate $\widehat{\sigma}^2 = \lambda_{\min}(\widehat{V}_2 - \widehat{V}_1\widehat{V}_1^{\top})$. Also worth noting is that the zero skewness condition on the observation noise, as detailed in Appendix B.1.2, is needed so that M_3' has the desired tensor structure. Second, M_2 and M_3 can be viewed as cross moments of a topic model with continuous observations, i.e., a "word" is a real vector drawn from a topic-specific continuous distribution, whose mean is a column of UT. Interestingly, the proof of Theorem 4 in Appendix B.1.2 indicates that M_2 and M_3 always have the same form regardless of the observation noise model. This property, as discussed later in Section 5.4, allows the possibility of handling more general noise covariances.

As in learning Markov chains, we need to resolve issues related to permutation invariance inherent in tensor decomposition. The situation is a bit more complicated here. First note that $P = (rU + (1-r)UT)^{\dagger}UT$, which implies that permuting the columns of U and the columns of UT in the same manner has the effect of permuting both the rows and the columns of P, essentially re-labeling the hidden states. Hence we can only expect to recover P up to some simultaneous row and column permutation. By the assumption that π_i's are all different, we can sort the two estimates $\widehat{\boldsymbol{\pi}}'$ and $\widehat{\boldsymbol{\pi}}$ to match the columns of $\widehat{U}$ and $\widehat{UT}$, and obtain $\widehat{P}$ if r is known. When r is unknown, a similar heuristic to the one for first-order Markov models can be used to estimate r, based on the fact that $P = (rU + (1-r)UT)^{\dagger}UT = (rI + (1-r)T)^{-1}T$, meaning Theorem 3 still holds when expressing P by U and UT.

Algorithm 5.4 gives the complete procedure for learning HMM from non-sequence data. Combining the perturbation bound of the tensor decomposition method in Theorem 1, perturbation theory on the whitening procedure (Appendix B.3.1) and the matrix pseudo inverse [Stewart, 1977], and concentration bounds on empirical moments (Appendix 7), we provide a sample complexity analysis of the proposed algorithm:

Theorem 5. *Suppose the numbers of iterations* N *and* L *for the tensor decomposition methods satisfy the conditions in Theorem 1, and the number of hidden states* k*, the success probability* r*, and the Dirichlet parameter* α_0 *are known. For any* $\eta \in (0,1)$ *and* $\epsilon > 0$*, if the number of sets*

$$N \geq \frac{12\max(k^2,m)m^3\nu^3(\alpha_0+2)^2(\alpha_0+1)^2}{\eta} \cdot \max\left(\frac{225000}{\delta_{\min}^2}, \frac{4600}{\min(\sigma_k(M_2'),\sigma_k(M_2))^2}, \frac{42000c^2\sigma_1(UT)^2\max(\sigma_1(UT),\sigma_1(U),1)^2}{\epsilon^2\sigma_k(rU+(1-r)UT)^4\min(\sigma_k(UT),\sigma_k(U),1)^4}\right),$$

where c *is some constant,* $\nu := \max(\sigma^2 + \max_{i,k}(|U_{ik}|^2), 1)$, $\delta_{\min} := \min_{i,j}|1/\sqrt{\pi_i} - 1/\sqrt{\pi_j}|$*, and* $\sigma_i(\cdot)$ *denotes the* i*-th largest singular value, then there exists a permutation matrix* Π *such that the* $\widehat{P}$ *and* $\widehat{U}$ *returned by Algorithm 5.4 satisfy*

$$\textit{Prob}(\|P - \Pi^\top \widehat{P}\Pi\| \leq \epsilon) \geq 1-\eta \textit{ and Prob}\left(\|U - \widehat{U}\Pi\| \leq \frac{\epsilon\sigma_k(rU+(1-r)UT)^2}{6\sigma_1(UT)}\right) \geq 1-\eta,$$

where $\|\cdot\|$ *denotes the matrix spectral norm.*

The proof is in Appendix B.4. In this result, the sample size N exhibits a fairly high-order polynomial dependency on m, k, ϵ^{-1} and scales with the inverse of the failure probability $1/\eta$ linearly instead of logarithmically, as is common in sample complexity results on spectral learning [Anandkumar et al., 2012a,b]. This is mainly because we do not impose boundedness or sub-Gaussianity constraints on the observation model, and only use the weaker Markov inequality for bounding the deviation in the empirical moments. Note that simply assuming the state-conditioned observation noise to be sub-Gaussian does not enable the use of stronger bounds such as Hoeffding bounds, because a mixture of sub-Gaussian distributions may not be sub-Gaussian. One possible strategy to strengthen our result is applying the analysis techniques of Hsu and Kakade [2013], who demonstrate that the sample complexity of spectral learning of certain mixture models has a logarithmic dependency on $1/\eta$. However, efforts beyond a direct use of their results are likely needed due to our LDA-like moments, M_2 and M_3. Also worth noting is that $\delta_{\min}^{-2}$ acts as a threshold. As shown in our proof, as long as the operator norm of the tensor perturbation is sufficiently smaller than $\delta_{\min}$, which measures the gaps between different π_i's, we can correctly match the two sets of estimated tensor eigenvalues. Lastly, the lower bound of N, as one would expect, depends on conditions of the matrices being estimated as reflected in the various ratios of singular values.

An interesting quantity missing from the sample analysis is the size of each set n. To simplify the analysis we essentially assume[1] $n = 3$, but understanding how n might affect the sample complexity may have a critical impact in practice: given a fixed budget on the total number of

[1]To be rigorous, we assume n to be the smallest number so that we can compute from a single set all the various empirical moments with non-overlapping data points.

observations that can be made, should we collect more sets or larger sets? What quantities may this choice depend on? We do not have a formal result yet, but intuitively, more sets seem to be always as good as, if not better than larger sets. According to our generative process, a larger set provides more information only about a subset of T's columns, those corresponding to the hidden states on which the set-specific initial state distribution $\boldsymbol{\pi}^{(0)}$ has large probability mass, whereas more sets provide more information about the entire model. A rigorous analysis is an interesting direction for future work.

5.3 Simulation

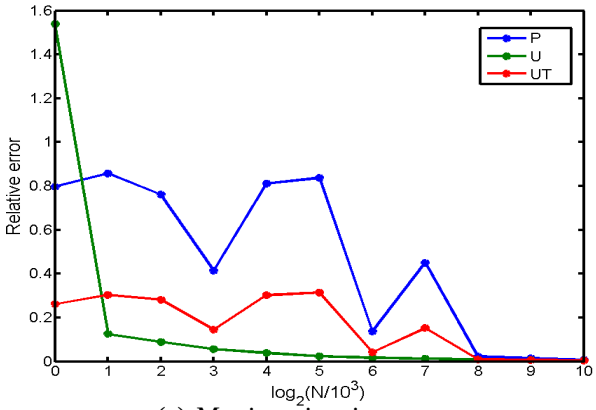
We consider learning HMMs from non-sequence data produced by the assumed generative process. The true HMM has $m = 40, k = 5$ and spherical Gaussian noise with $\sigma^2 = 2$. The mean vectors U were sampled from an independent univariate standard normal and then normalized to lie on the unit sphere. The transition matrix P and the stationary hidden state distribution $\boldsymbol{\pi}$ are

$$P = \begin{bmatrix} 0.1088 & 0.3512 & 0.4114 & 0 & 0.0642 \\ 0.1271 & 0.0411 & 0.0844 & 0.2125 & 0.4950 \\ 0.1310 & 0.0424 & 0.0251 & 0.5287 & 0.3770 \\ 0.5306 & 0.2957 & 0.4564 & 0.0489 & 0.0251 \\ 0.1026 & 0.2697 & 0.0228 & 0.2100 & 0.0386 \end{bmatrix}, \quad \boldsymbol{\pi} = \begin{bmatrix} 0.1858 \\ 0.1747 \\ 0.2324 \\ 0.2730 \\ 0.1340 \end{bmatrix}. \tag{5.4}$$

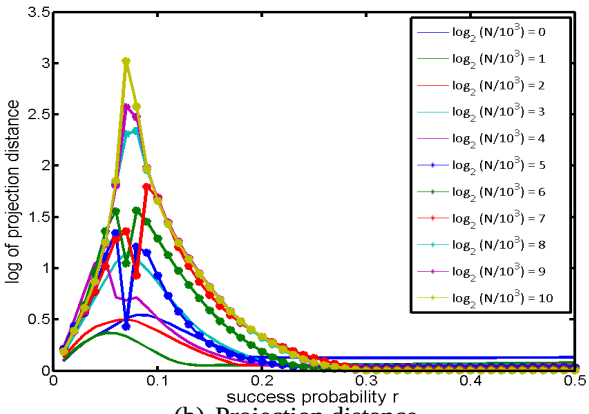
The transition probability matrix has exactly one zero entry. We conduct two experiments.

The first experiment is a sanity check on the consistency of the proposed algorithm. We set $\alpha_0 = 1$ and $r = 0.3$ in the generative process, and consider different numbers of sets $N \in 1000\{2^0, 2^1, \ldots, 2^{10}\}$, while fixing the size of each set $n = 1000$. The numbers of iterations for the tensor decomposition method were $\mathsf{N} = 200$ and $\mathsf{L} = 1000$. Figure 5.4(a) plots the relative matrix estimation error (in spectral norm) against the sample size N for P, U, and UT, showing that U is the easiest to learn, followed by UT, and P is the most difficult, and that all three errors converge to a very small value for sufficiently large N. Note that in Theorem 5 the bounds for P and U are different. With the model used here, the extra multiplicative factor in the bound for U is less than 0.007, suggesting that U is indeed easier to estimate than P. Figure 5.4(b) demonstrates the heuristics for determining r, showing projection distances (in logarithm) versus r. As N increases, the take-off point gets closer to the true $r = 0.3$. The large peak indicates a pole (the set S in Theorem 3).

The second experiment compares the proposed method with the popular EM-based learning paradigm. In Appendix A we derive a variational EM algorithm for learning HMM parameters assuming the generative process in Section 5.2.2. In this experiment, the generative process has the same settings as in the first experiment except the number of sets N, which takes smaller values $\{125, 250, 500, 1000, 2000, 4000\}$. We repeat the experiment 20 times with different random draws from the generative process. Figure 5.5 gives the relative estimation errors for U (in spectral norm) and P (in entrywise 1-norm) for three methods: Algorithm 5.4 (tensor), variational EM initialized with the output of Algorithm 5.4 (tensor+vbEM), and variational EM initialized with 100 random parameter values (rand+vbEM). Clearly, Algorithm 5.4 outperforms the randomly initialized variation EM, and there is barely any improvement resulting from combining

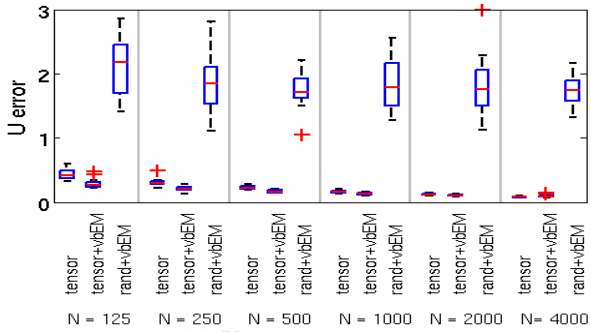


(a) Matrix estimation error

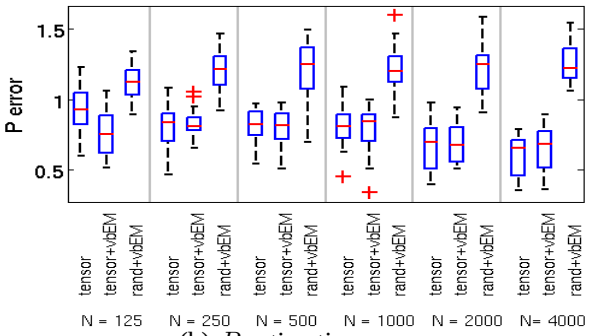


(b) Projection distance

Figure 5.4: Simulation confirming consistency of the proposed algorithm



(a) U estimation error



(b) P estimation error

Figure 5.5: Comparison between Algorithm 5.4 and EM

the two methods, except when N is very small. In terms of computational efficiency, we observe that Algorithm 5.4 is orders of magnitude faster than the variational EM algorithm. On our platform with 48 cores (2.3 GHz each) and 512GB of memory, Algorithm 5.4 takes a couple of hours to finish all 20 experiments, but the variational EM method takes days.

5.4 Discussion

We have demonstrated that under reasonable assumptions, tensor decomposition methods can provably learn first-order Markov models and hidden Markov models from non-sequence data. We believe this is the first formal guarantee on learning dynamic models in a non-sequential setting. There are several possibilities in improving or generalizing our results.

Procedure for estimating r with formal guarantees

Our current heuristics for estimating r requires a good measure of sudden increase or take-off spot in the curve of projection distance v.s. r, which is hard to define because, depending on the true transition matrix P, the curve may be rather smooth near the true take-off point, as shown in Figure 5.4(b). We suspect that as P becomes sparser, the curve shows a sharper increase at the

true take-off point, but do not have a concrete result yet. If such results can be established, it is then possible to develop a change-point detection based procedure for estimating r with formal guarantees.

Other distributions for the missing times

No matter what distribution generates the random time steps, tensor decomposition methods can always learn the expected transition probability matrix T. Depending on the specific modeling task, one may replace the geometric distribution with some other distribution, such as Poisson.

HMMs with discrete observations

With extra assumptions on the state-observation probability matrix, we can modify our proposed algorithm to guarantee consistent parameter learning in the case of discrete observations. More precisely, let $O \in [0,1]^{m \times k}$ denote the state-observation probability matrix, where each column is the observation probability vector for a hidden state. We first apply our tensor decomposition based method to recover the matrix product OT, and then obtain estimates of O and T with the non-negative matrix factorization (NMF) algorithm proposed by Arora et al. [2012], which guarantees consistency under the "anchor word" assumption, requiring that each column of O has a corresponding row whose only positive entry coincides with itself.

Weaker assumption on Dirichlet parameters $\boldsymbol{\alpha}$

So far we have assumed that the normalized Dirichlet parameter vector $\boldsymbol{\alpha}/\sum_i \alpha$ is equal to the stationary hidden state distribution, allowing the columns and rows of the expected transition probability matrix T to be correctly matched. However, a careful look into the tensor structures in Theorems 2 and 4 reveals that the weaker conditions $\alpha_i > \alpha_j \iff (T\boldsymbol{\alpha})_i > (T\boldsymbol{\alpha})_j \; \forall i \neq j$ and $\alpha_i \neq \alpha_j \; \forall i \neq j$ are sufficient for correct matching. To interpret such conditions, we note that $\boldsymbol{\alpha}$ is proportional to the average initial hidden state distribution, while $T\boldsymbol{\alpha}$ to the average hidden state distribution that generates the observations. As long as the hidden states, when sorted by probability mass, are in the same unique order under these two distributions, we can correctly match the rows and columns of T.

General observation noise covariance

As pointed out in Section 5.2.2, it is the tensor decomposition-based estimation of the mean observation vectors U that requires the assumption of a spherical noise covariance, while the estimation of UT can always be carried out via tensor decomposition regardless of the noise distribution. This property, together with the fact that each set of non-sequence data can be viewed as independent samples drawn from a mixture model, suggests modifications for handling more general noise covariances by using alternative methods to estimate U. For example, under reasonable assumptions on the separation between the mean observation vectors U, the spectral projection based methods proposed by Achlioptas and McSherry [2005]; Kannan et al. [2005] are guaranteed to return accurate parameter estimates of mixtures of log-concave distributions with general noise covariances. In the case of Gaussian mixtures, the approach by Moitra and Valiant [2010] provably learns the parameters with minimal assumptions that require no separation between the mean vectors and allow general covariances, though their algorithm, despite its polynomial time and sample complexity, is far from being practical. Or in the case of the multi-view Gaussian mixture models considered by Anandkumar et al. [2012b], where the covariances of different mixture components share the same block diagonal structure, tensor decomposition based methods can also provably learn the parameters.

Chapter 6

Learning Hidden Markov Models from Sequence and Non-sequence Data

In this chapter we consider learning HMMs when, in addition to non-sequence data, there are also some sequence data. The non-sequence data here are assumed to be independent samples drawn from the stationary distribution of the underlying HMM. Unlike the methods proposed in the last chapter, which give direct estimates of HMM parameters, our proposed methods here learn an *observable representation* of the underlying HMM. In the usual sequence-data only setting, spectral learning of observable representation of HMMs [Hsu et al., 2009; Siddiqi et al., 2010; Song et al., 2010] is becoming an appealing alternative to the popular EM method because of its formal theoretical guarantee and more importantly, empirical success in several applications ranging from robot vision to music analysis [Song et al., 2010]. Building on these recent advances, we propose spectral methods that combine sequence and non-sequence data for learning HMMs. Unlike most spectral algorithms which apply Singular Value Decomposition (SVD) to moments estimated by empirical averages of data, our methods first solve a penalized least square problem to get better estimates of moments, and then apply SVD [1]. As one may imagine, the penalized least square problem here has a similar structure to the one in Chapter 4, where the objective consists of a squared *error* function on the sequence and a *regularization* term based on non-sequence data. But somewhat surprisingly, as we will show in details later, the optimization problems here turn out to be convex, even though they are dealing with a more complex model than the VAR model in Chapter 4. Through experiments on synthetic data and real Inertia-Measurement Unit recordings of human activities, we demonstrate that, as with VARs, incorporating non-sequence data also improves estimation of HMMs.

This chapter is organized as follows. Section 6.1 briefly reviews spectral learning algorithms, and Section 6.2 details the proposed algorithms, followed by experiments and results in Section 6.3 and conclusions in Section 6.4.

[1] The general idea of invoking convex optimization in spectral learning has been proposed recently in the sequential learning setting. Among others, Balle et al. [2012] solve a convex program in place of SVD, while Balle and Mohri [2012] use convex optimization to obtain input matrices to spectral algorithms.

6.1 Spectral Learning of HMMs

We begin with discrete observations, and mainly follow the exposition by Siddiqi et al. [2010]. Instead of learning the original model parameters, i.e., initial state probabilities, state transition probabilities, and state-conditioned observation probabilities, the spectral algorithm learns an *observable representation* of the HMM, which consists of the following parameters:

$$\begin{aligned} \mathbf{b}_1 &:= U^\top \mathbf{p}, && (6.1)\\ \mathbf{b}_\infty &:= (P_{2,1}^\top U)^\dagger \mathbf{p}, && (6.2)\\ B_x &:= (U^\top P_{3,x,1})(U^\top P_{2,1})^\dagger,\ 1 \le x \le N, && (6.3) \end{aligned}$$

where † denotes the pseudo inverse, N is the number of observation symbols, $\mathbf{p}$ is the stationary distribution of observations, and $P_{2,1}$ and $P_{3,x,1}$ are joint observation probability matrices such that for $1 \le i, x, j \le N$,

$$\begin{aligned} (P_{2,1})_{ij} &:= \text{Prob}(x_{t+1} = i, x_t = j),\\ (P_{3,x,1})_{ij} &:= \text{Prob}(x_{t+1} = i, x_t = x, x_{t-1} = j), \end{aligned} \tag{6.4}$$

x_t being the observation symbol at time t, and $U \in \mathbb{R}^{N \times k}$ is column concatenation of the top k left singular vectors of $P_{2,1}$. As the name suggests, the observable representation parameters (6.1) to (6.3) only depend on observable quantities, leading naturally to the estimates $\widehat{\mathbf{b}}_1, \widehat{\mathbf{b}}_\infty$, and $\widehat{B}_x$ based on empirical averages $\widehat{\mathbf{p}}$, $\widehat{P}_{2,1}$, $\widehat{P}_{3,x,1}$, and $\widehat{U}$, the top-k left singular vectors of $\widehat{P}_{2,1}$. These estimates allow us to perform inferences on a new sequence of observations $y_1, \ldots, y_t$:

- Predict whole sequence probability:

$$\widehat{\text{Prob}}(y_1, \ldots, y_t) = \widehat{\mathbf{b}}_\infty^\top \widehat{B}_{y_t} \cdots \widehat{B}_{y_1} \widehat{\mathbf{b}}_1. \tag{6.5}$$

- Internal state update: $\widehat{\mathbf{b}}_{t+1} := \widehat{B}_{y_t} \widehat{\mathbf{b}}_t / (\widehat{\mathbf{b}}_\infty^\top \widehat{B}_{y_t} \widehat{\mathbf{b}}_t)$.
- Conditional probability of y_t given $y_1, \ldots, y_{t-1}$:

$$\widehat{\text{Prob}}(y_t | y_1, \ldots, y_{t-1}) := \frac{\widehat{\mathbf{b}}_\infty^\top \widehat{B}_{y_t} \widehat{\mathbf{b}}_t}{\sum_x \widehat{\mathbf{b}}_\infty^\top \widehat{B}_x \widehat{\mathbf{b}}_t}. \tag{6.6}$$

Under some mild conditions, of which the most critical being that both the state transition and state-conditioned observation probability matrices are of rank k, Siddiqi et al. [2010] showed that the whole sequence probability estimate (6.5) is consistent (with high probability) and gives a finite-sample bound on the estimation error.

Based on the same idea, Song et al. [2010] developed a spectral algorithm for learning HMMs with continuous observations. Instead of operating on probability distributions directly, their algorithm operates on *Hilbert space embeddings* of distributions of observable quantities (assuming stationarity of the HMM):

$$\begin{aligned} \mu_1 &:= \mathbb{E}_{\mathbf{x}_t}[\phi(\mathbf{x}_t)], && (6.7)\\ \mathcal{C}_{2,1} &:= \mathbb{E}_{\mathbf{x}_{t+1}\mathbf{x}_t}[\phi(\mathbf{x}_{t+1}) \otimes \phi(\mathbf{x}_t)], && (6.8)\\ \mathcal{C}_{3,\mathbf{x},1} &:= \mathbb{E}_{\mathbf{x}_{t+2}(\mathbf{x}_{t+1}=\mathbf{x})\mathbf{x}_t}[\phi(\mathbf{x}_{t+2}) \otimes \phi(\mathbf{x}_t)]\\ &= \mathbb{P}(\mathbf{x}_t = \mathbf{x}) \mathcal{C}_{3,1|2} \phi(\mathbf{x}), && (6.9) \end{aligned}$$

where $\mathbf{x}_t$ denotes the continuous observation vector at time t, $\phi(\cdot)$ maps the real observation space to a Reproducing Kernel Hilbert Space (RKHS), $\otimes$ denotes the tensor product, and $\mathcal{C}_{3,1|2} := \mathcal{C}_{\mathbf{x}_{t+2}\mathbf{x}_t|\mathbf{x}_{t+1}}$ is a *conditional embedding operator* [Song et al., 2009]. Using these embeddings, they derived an observable representation of the embedded HMM, which consists of the following parameters:

$$\boldsymbol{\beta}_1 := \mathcal{U}^\top \mu_1, \tag{6.10}$$

$$\boldsymbol{\beta}_\infty := \mathcal{C}_{2,1}(\mathcal{U}^\top \mathcal{C}_{2,1})^\dagger, \tag{6.11}$$

$$\mathcal{B}_\mathbf{x} := (\mathcal{U}^\top \mathcal{C}_{3,\mathbf{x},1})(\mathcal{U}^\top \mathcal{C}_{2,1})^\dagger, \tag{6.12}$$

where $\mathcal{U}$ is the top-k left singular vectors of $\mathcal{C}_{2,1}$. They then showed that the embedding of the predictive distribution $\mathbb{P}(\mathbf{x}_t|\mathbf{x}_1, \ldots, \mathbf{x}_{t-1})$ takes the form $\mu_{\mathbf{x}_t|\mathbf{x}_1,\ldots,\mathbf{x}_{t-1}} = \boldsymbol{\beta}_\infty \mathcal{B}_{\mathbf{x}_1} \cdots \mathcal{B}_{\mathbf{x}_{t-1}} \boldsymbol{\beta}_1$ and, as in the case of discrete observations, proposed estimates based on empirical averages $\widehat{\mu}_1, \widehat{\mathcal{C}}_{2,1}, \widehat{\mathcal{C}}_{3,\mathbf{x},1}$, and $\widehat{\mathcal{U}}$, which is the top-k left singular vectors of $\widehat{\mathcal{C}}_{2,1}$. Using the kernel trick and techniques from Kernel Principle Component Analysis [Schölkopf et al., 1998], they gave an estimation procedure that operates solely on finite-dimensional quantities. Moreover, to avoid the difficulty of partitioning the observation space required by estimation of $\mathcal{B}_\mathbf{x}$, they proposed to estimate instead

$$\bar{\mathcal{B}}_\mathbf{x} := (\mathcal{U}^\top \mathcal{C}_{3,1|2} \phi(\mathbf{x}))(\mathcal{U}^\top \mathcal{C}_{2,1})^\dagger, \tag{6.13}$$

which is only a fixed multiplicative factor $\mathbb{P}(\mathbf{x})$ away from $\mathcal{B}_\mathbf{x}$, and have $\mu_{\mathbf{x}_t|\mathbf{x}_1,\ldots,\mathbf{x}_{t-1}}$ proportional to $\boldsymbol{\beta}_\infty \bar{\mathcal{B}}_{\mathbf{x}_1} \cdots \bar{\mathcal{B}}_{\mathbf{x}_{t-1}} \boldsymbol{\beta}_1$. Under some mild conditions, they established the consistency (with high probability) of their estimator for $\mu_{\mathbf{x}_t|\mathbf{x}_1,\ldots,\mathbf{x}_{t-1}}$ and gave a finite-sample bound on the estimation error.

In addition to estimation, Song et al. [2010] also discussed possible ways to carry out prediction. In particular, they showed that in the case of Gaussian RBF kernel, $\widehat{\mu}_{\mathbf{x}_t|\mathbf{x}_1,\ldots,\mathbf{x}_{t-1}}$ takes the form of a nonparametric density estimator after proper normalization, and one may choose, from training data or a pool of samples, the observation with the highest predictive density as the prediction.

6.2 Spectral Methods for Learning HMMs from Sequence and Non-sequence Data

Suppose in addition to sequence data, which can be time series of observations or triples of consecutive observations, we also have a set of *non-sequence data points*, which are drawn *independently* from the stationary distribution of the underlying HMM. We propose to improve the estimation of the observable representation of HMMs by solving regularized least square problems, which minimize a squared error term on the sequence data *and* a regularization term based on the non-sequence data. As in existing work on spectral learning of HMMs, we assume that the sequence data are observed after the HMM has fully mixed.

6.2.1 Discrete Observations

Our method has two main steps. We first estimate $P_{2,1}$, and then $\mathbf{b}_1, \mathbf{b}_\infty$, and B_x's. Let N denote the number of unique observation symbols. To make use of non-sequence data in estimating $P_{2,1}$, we note that the marginal of $P_{2,1}$ is the stationary distribution of the discrete HMM. Moreover, from spectral learning methods we have the assumption of $P_{2,1}$ being low-rank. We thus propose the following estimator $\widetilde{P}_{2,1}$ defined as

$$\begin{aligned} \arg\min_{P} \quad & \frac{1}{2}\|W \odot (P - \widehat{P}_{2,1})\|_F^2 + \tau\|P\|_* + \\ & \frac{u}{2}\Big(\|\widetilde{\mathbf{p}} - P\mathbf{1}\|_2^2 + \|\widetilde{\mathbf{p}} - P^\top \mathbf{1}\|_2^2\Big), \\ \text{s.t.} \quad & \mathbf{1}^\top P \mathbf{1} = 1,\ P_{ij} \geq 0, \end{aligned} \tag{6.14}$$

where $\widetilde{\mathbf{p}}$ is the empirical observation distribution of *both the sequence and the non-sequence data*, W is an indicator matrix such that $W_{ij} = 1 \iff (\widehat{P}_{2,1})_{ij} > 0$, $\odot$ denotes the Hadamard product, $\|\cdot\|_*$ denotes the matrix nuclear norm, a standard convex relaxation of matrix rank, $\mathbf{1}$ is a vector of ones, and $u, \tau > 0$ are regularization parameters. The objective in (6.14) minimizes the squared error from the sequence-only estimate $\widehat{P}_{2,1}$ while penalizing the rank and the deviation from the marginal $\widetilde{\mathbf{p}}$. It is easy to see that (6.14) is a convex but non-smooth problem due to the matrix nuclear norm. Projected sub-gradient descent methods are a common way to solve such problems, but are known to suffer from slow convergence [Bertsekas, 1999]. We solve (6.14) by a variant of the smoothing proximal gradient (SPG) method proposed by Chen et al. [2012], which achieves a provably faster convergence rate than projected sub-gradient methods but has a similar per-iteration time complexity. In Section 6.2.2 we use SPG to solve the continuous version of the estimation problem, which has a more general form, and hence describe more details there.

To set τ in the right scale, we use the following fact about matrix norms:

$$\|P_{2,1}\|_*/N \leq (r/N)\sqrt{\|P_{2,1}\|_\infty \|P_{2,1}\|_1}, \tag{6.15}$$

where r is the rank of $P_{2,1}$, and $\|\cdot\|_\infty$ and $\|\cdot\|_1$ denote matrix ∞-norm and 1-norm, respectively. Assuming stationarity, we have $\|P_{2,1}\|_\infty = \|P_{2,1}\|_1 = \max_i \mathbf{p}_i$, where $\mathbf{p}$ is the stationary distribution of observations. Therefore, $P_{2,1}$'s average singular value is $O((\max_i \mathbf{p}_i)/N)$. As shown by Cai et al. [2010], τ has an effect of soft-thresholding singular values of $P_{2,1}$, so we let $\tau = \lambda \max_i \widetilde{\mathbf{p}}_i/N$ and tune λ instead.

We then compute the SVD of $\widetilde{P}_{2,1}$, denoting its top-k left singular vectors as an N-by-k matrix $\widetilde{U}$, and obtain estimates of $\mathbf{b}_1$ and $\mathbf{b}_\infty$ in the same ways as (6.1) and (6.2) using $\widetilde{P}_{2,1}, \widetilde{U}$, and $\widetilde{\mathbf{p}}$. To derive our estimator of B_x, we first note that the original estimator based on (6.3) is the solution to the following problem:

$$\widehat{B}_x := \arg\min_B \|\widehat{P}_{3,x,1} - \widehat{U}B\widehat{U}^\top \widehat{P}_{2,1}\|_F^2, \tag{6.16}$$

showing that $\widehat{B}_x$ is a low-dimensional representation of $\widehat{P}_{3,x,1}$. As in (6.14), we aim to regularize the least-square problem (6.16) with non-sequence data. Instead of constructing a regularization

term directly from non-sequence data, we use our new estimator $\widetilde{P}_{2,1}$ based on the fact that $(\mathbf{1}^\top P_{3,x,1})_j = (P_{2,1})_{xj}$ and $(P_{3,x,1}\mathbf{1})_i = (P_{2,1})_{ix}$, i.e., the marginals of $\{P_{3,x,1}\}$ are equal to $P_{2,1}$. We thus propose the following estimator $\{\widetilde{B}_x\}$ defined as

$$\begin{aligned}\arg\min_{\{B_x\}} \sum_x \frac{1}{2}\|W_x \odot (\widetilde{U}B_x\widetilde{V}^\top - \widehat{P}_{3,x,1})\|_F^2 + \\ \frac{u}{2}\sum_{x,i}\left((\widetilde{P}_{2,1})_{ix} - (\widetilde{U}B_x\widetilde{V}^\top\mathbf{1})_i\right)^2 + \\ \frac{u}{2}\sum_{x,i}\left((\widetilde{P}_{2,1})_{xi} - (\mathbf{1}^\top\widetilde{U}B_x\widetilde{V}^\top)_i\right)^2, \\ \text{s.t. } (\widetilde{U}B_x\widetilde{V}^\top)_{ij} \geq 0, \sum_x \mathbf{1}^\top\widetilde{U}B_x\widetilde{V}^\top\mathbf{1} = 1,\end{aligned} \tag{6.17}$$

where W_x is an indicator matrix such that $(W_x)_{ij} > 0 \iff (\widehat{P}_{3,x,1})_{ij} > 0$ and $\widetilde{V} := \widetilde{U}^\top\widetilde{P}_{2,1}$. Note that we not only add regularization terms but also constrain the fitted matrices $\{\widetilde{U}B_x\widetilde{V}^\top\}$ to lie on a simplex, mainly to reduce negative values in the predictive distribution (6.6) during inference. The simplex constraints may incur more bias than desired and may not always be feasible[2], but in our experiments we do not observe any negative effect. Later in Section 6.4 we discuss the possibility of combining the two optimization problems (6.14) and (6.17) into one, which may fix some of these constraint-related issues but pay the price of a bigger problem size.

Eq. (6.17) is a quadratic program of k^2N variables under one linear equality constraint and N^3 linear inequality constraints. When N is on the order of a few hundreds and k is a few tens, a reformulation that takes advantage of the block-diagonal structure in the Hessian of (6.17) can be solved quite efficiently with state-of-the art optimization software, such as MOSEK (`www.mosek.com`). For larger problems, one possible solution is the Alternating Direction Method of Multipliers [Boyd et al., 2011], which handles constraints by minimizing the original objective augmented with a iteratively-refined constraint violation term. Our experiments in Section 6.3.1 have $N = 100$, so we solve (6.17) with MOSEK.

6.2.2 Continuous Observations

Our method for continuous observations builds on the Hilbert space embedding approach by Song et al. [2010], and consists of two main steps: estimating the feature covariance $\mathcal{C}_{2,1}$ and then the observable representation $\boldsymbol{\beta}_1$, $\boldsymbol{\beta}_\infty$, and $\mathcal{B}_\mathbf{x}$. Let the feature mappings of the sequence data be organized into three matrices Φ_1, Φ_2, and Φ_3 such that their i-th columns Φ_1^i, Φ_2^i, and Φ_3^i are consecutive and going forward in time. By the definition of the feature covariance (6.8), we have $\mathcal{C}_{2,1} := \int \phi(\mathbf{x}) \otimes \phi(\mathbf{y}) p_{X_{t+1}X_t}(\mathbf{x}, \mathbf{y}) d\mathbf{x} d\mathbf{y}$. If we have a set of feature points grouped column-wise as a feature matrix Φ, and know exactly which pairs of points are consecutive in time via a (normalized) temporal adjacency matrix $T_{2,1}$, we may then compute the quantity $\Phi T_{2,1}\Phi^\top$ as an unbiased estimator of $\mathcal{C}_{2,1}$ It is easy to see that $\widehat{\mathcal{C}}_{2,1} := \frac{1}{n}\Phi_2\Phi_1^\top$ is one special

[2]When this happens one may choose the smallest k that makes the constraints feasible, and then solve (6.17).

case of such an estimator. To incorporate non-sequence data into our estimation procedure, we denote its feature matrix by $\mathcal{Z}$ and consider another special case:

$$\widetilde{\mathcal{C}}_{2,1} \;:=\; \mathcal{Z}_2 P \mathcal{Z}_1^\top, \tag{6.18}$$

where $\mathcal{Z}_1 := [\Phi_1 \; \mathcal{Z}]$ and $\mathcal{Z}_2 := [\Phi_2 \; \mathcal{Z}]$. It then suffices to estimate P subject to $\mathbf{1}^\top P \mathbf{1} = 1$ and $P_{ij} \geq 0$.

Similar to Section 6.2.1, our estimation objective consists of three terms: the squared error between $\widetilde{\mathcal{C}}_{2,1}$ and $\widehat{\mathcal{C}}_{2,1}$, penalization on $\widetilde{C}_{2,1}$'s rank, and deviation of $\widetilde{C}_{2,1}$'s marginal from the mean of the stationary distribution. The last term is based on the fact that, under the assumption of stationarity, $\mathcal{C}_{2,1}\mathbf{f} \;=\; \mathbb{E}[\phi(X)]$ holds for some constant function $\mathbf{f}$ in $\mathcal{G}$ such that $\mathbf{f}(\mathbf{x}) = \mathbf{f}^\top \phi(\mathbf{x}) = 1 \; \forall \mathbf{x}$. Formally, our estimator $\widetilde{P}$ is the solution to the following convex program:

$$\begin{aligned} \min_P \quad & \frac{1}{2}\|\mathcal{Z}_2 P \mathcal{Z}_1^\top - \widehat{\mathcal{C}}_{2,1}\|^2_{\mathcal{G}\otimes\mathcal{G}} + \tau \|\mathcal{Z}_2 P \mathcal{Z}_1^\top\|_* + \\ & \frac{u}{2}\left(\left\| \mathcal{Z}_2 P \mathbf{1} - \frac{\mathcal{S}\mathbf{1}}{m_S} \right\|^2_{\mathcal{G}} + \left\| \mathcal{Z}_1 P^\top \mathbf{1} - \frac{\mathcal{S}\mathbf{1}}{m_S} \right\|^2_{\mathcal{G}} \right) \\ \text{s.t.} \quad & \mathbf{1}^\top P \mathbf{1} = 1, P_{ij} \geq 0, \end{aligned} \tag{6.19}$$

where we introduce $\mathcal{S}$ and m_S to denote the feature matrix and the size of the entire set of non-sequence data and let $\mathcal{Z}$ denote a sub-sample of it, mainly to limit the number of variables when the non-sequence dataset is very large. As shown in Appendix C.1, using the kernel trick and properties of the matrix trace and nuclear norm, we re-write the objective function in (6.19) as follows (dropping constants):

$$\begin{aligned} & \frac{1}{2}\text{Tr}(P^\top M_2 P M_1) - \text{Tr}(P^\top F) + \tau \|L_2^\top P L_1\|_* + \\ & \frac{u}{2}\mathbf{1}^\top (P^\top M_2 P + P M_1 P^\top)\mathbf{1} - u\mathbf{1}^\top (P^\top \boldsymbol{\mu}_2 + P\boldsymbol{\mu}_1), \end{aligned} \tag{6.20}$$

where $\text{Tr}(\cdot)$ is the matrix trace, $M_i := \mathcal{Z}_i^\top \mathcal{Z}_i, \boldsymbol{\mu}_i := \frac{\mathcal{Z}_i^\top \mathcal{S}\mathbf{1}}{m_S}, F := \mathcal{Z}_2^\top \widehat{C}_{2,1} \mathcal{Z}_1$, and L_i is a finite matrix such that $M_i = L_i L_i^\top$. To set τ in a proper scale, we use an inequality similar to (6.15) to upper-bound the average singular value of $L_2^\top P L_1$, and then replace the unknown P by the uniform distribution to have $\tau := (\lambda/m^3)(\|L_2^\top \mathbf{1}\mathbf{1}^\top L_1\|_\infty \|L_2^\top \mathbf{1}\mathbf{1}^\top L_1\|_1)^{1/2}$, where m is the size of P and $\lambda > 0$ takes values in some reasonable range.

As mentioned in Section 6.2.1, we solve (6.19) with a variant of the smoothing proximal gradient (SPG) method outlined in Algorithm 6.1, which minimizes $f_\mu(P)$, a smooth approximation of (6.20) that approximates the non-smooth regularization $\tau\|L_2^\top P L_1\|_*$ by

$$g_\mu(P) \;:=\; \max_{\|Y\|_2 \leq 1} \tau \text{Tr}(Y^\top L_2^\top P L_1) - \frac{\mu}{2}\|Y\|_F^2, \tag{6.21}$$

where $\mu \geq 0$ is a smoothing parameter, $\|\cdot\|_2$ and $\|\cdot\|_F$ denote the matrix spectral and Frobenius norms, respectively. Nesterov [2005] shows that (6.21) is continuously differentiable in P and $\nabla g_\mu(P) = \tau L_2 Y^* L_1^\top$, where Y^* is the optimal solution to (6.21) obtained by projecting $(\tau/\mu)L_2^\top P L_1$ to the unit spectral-norm ball, i.e., truncating its singular values at 1. The update

Algorithm 6.1 Smoothing Proximal Gradient for (6.19)

Initialize $Y^{(0)} = P^{(0)}$ to some feasible point.
Set $t := 0, \theta^{(0)} := 1, \eta := 10$, and $\gamma^{(0)} := 1$.
repeat
Find the smallest $\kappa \in \{0, 1, \cdots\}$ that satisfies

$$f_\mu(P^{(t+1)}) - f_\mu(Y^{(t)}) \leq \frac{\gamma^{(t+1)}}{2}\|P^{(t+1)} - Y^{(t)}\|_F^2 + \mathrm{Tr}\big((P^{(t+1)} - Y^{(t)})^\top \nabla f_\mu(Y^{(t)})\big)$$

where $\gamma^{(t+1)} := \eta^\kappa \gamma^{(t)}$ and

$$P^{(t+1)} := \arg\min_P \|Y^{(t)} - \nabla f_\mu(Y^{(t)})/\gamma^{(t+1)} - P\|_F^2 \quad \text{s.t. } P_{ij} \geq 0, \mathbf{1}^\top P \mathbf{1} = 1. \tag{6.22}$$

$\theta^{(t+1)} := (1 + \sqrt{1 + 4(\theta^{(t)})^2})/2$.
$Y^{(t+1)} := P^{(t+1)} + \frac{\theta^{(t)}-1}{\theta^{(t+1)}}(P^{(t+1)} - P^{(t)})$.
$t := t + 1$.
until convergence or $t = T_{\max}$, an iteration limit.

(6.22) for $P^{(t+1)}$ requires projection onto a simplex, for which several efficient algorithms exist, such as the sorting-based method proposed by Duchi et al. [2008]. The convergence theory of Chen et al. [2012] suggests setting[3] $\mu = \epsilon/m$, m being the column dimension of $\mathcal{Z}_2$, so that the objective values (6.20) converge in $O(1/\epsilon^2)$ iterations to at most ϵ plus the minimum.

We then compute the top k left singular vectors of $\widetilde{\mathcal{C}}_{2,1}$ in a similar way to Kernel Principle Component Analysis [Schölkopf et al., 1998], starting with the fact that any left singular vector of $\widetilde{\mathcal{C}}_{2,1} = \mathcal{Z}_2 \widetilde{P} \mathcal{Z}_1^\top$ can be expressed as $\mathcal{Z}_2 \boldsymbol{\alpha}$ for some $\boldsymbol{\alpha} \in \mathbb{R}^m$, and any left singular vector of $\widetilde{C}_{2,1}$ is an Eigenvector of $\widetilde{C}_{2,1}\widetilde{C}_{2,1}^\top$ and vice versa. Thus we have

$$\begin{aligned} &\mathcal{Z}_2 \widetilde{P} M_1 \widetilde{P}^\top M_2 \boldsymbol{\alpha} = \mathcal{Z}_2 \widetilde{P} \mathcal{Z}_1^\top \mathcal{Z}_1 \widetilde{P}^\top \mathcal{Z}_2^\top (\mathcal{Z}_2 \boldsymbol{\alpha}) = \omega \mathcal{Z}_2 \boldsymbol{\alpha} \\ &\iff M_2 \widetilde{P} M_1 \widetilde{P}^\top M_2 \boldsymbol{\alpha} = \omega M_2 \boldsymbol{\alpha}, \end{aligned} \tag{6.23}$$

which is a generalized Eigensystem. Let Ω denote the diagonal matrix formed by the top k generalized Eigenvalues of (6.23), and A denote the column concatenation of the corresponding generalized Eigenvectors. It is then clear that $D := (A^\top M_2 A)^{-1/2}$ is diagonal, and we obtain a concise form of $\widetilde{\mathcal{C}}_{2,1}$'s top k left singular vectors $\widetilde{\mathcal{U}} = \mathcal{Z}_2 A D$. We also have the following useful identity:

$$M_2 \widetilde{P} M_1 \widetilde{P}^\top M_2 A = M_2 A \Omega. \tag{6.24}$$

[3]For solving (6.14) we set $\mu = \epsilon/N$.

Next we describe our estimators for the observable representation. First we have

$$\widetilde{\boldsymbol{\beta}}_1 := \widetilde{\mathcal{U}}^\top \mathcal{S}\mathbf{1}/m_S = DA^\top \boldsymbol{\mu}_2, \tag{6.25}$$

$$\widetilde{\boldsymbol{\beta}}_\infty := \widetilde{\mathcal{C}}_{2,1}(\widetilde{\mathcal{U}}^\top \widetilde{\mathcal{C}}_{2,1})^\dagger = \mathcal{Z}_2 \widetilde{P} M_1 \widetilde{P}^\top M_2 AD\Omega^{-1} \tag{6.26}$$

by using the identity $(\widetilde{\mathcal{U}}^\top \widetilde{\mathcal{C}}_{2,1})^\dagger = \mathcal{Z}_1 \widetilde{P}^\top M_2 AD\Omega^{-1}$ established from properties of pseudo inverse, (6.24), and the definition of D. To derive our estimator for $\bar{\mathcal{B}}_{\mathbf{x}}$ defined in (6.13), we start from the conditional covariance operator defined by Song et al. [2009]

$$\begin{aligned}
\mathcal{C}_{3,1|2} &:= \mathcal{C}_{3,1,2}\mathcal{C}_{2,2}^{-1}\phi(\mathbf{x}), \quad \text{where} \\
\mathcal{C}_{3,1,2} &:= \mathbb{E}_{X_{t+2}X_tX_{t+1}}[\phi(X_{t+2}) \otimes \phi(X_t) \otimes \phi(X_{t+1})], \\
\mathcal{C}_{2,2} &:= \mathbb{E}_{X_{t+1}}[\phi(X_{t+1}) \otimes \phi(X_{t+1})].
\end{aligned}$$

Using a similar idea to (6.18), we encode the empirical distribution of triples of consecutive observations by a third-order tensor Q and have the following estimator

$$\widetilde{\mathcal{C}}_{3,1|2} := \left(\sum_{i,j,l} Q_{ijl} \mathcal{Z}_3^i \otimes \mathcal{Z}_1^j \otimes \mathcal{Z}_2^l\right) \left(\frac{1}{m}\mathcal{Z}_2\mathcal{Z}_2^\top + \nu I\right)^{-1},$$

where $\mathcal{Z}_3 := [\Phi_3 \ \mathcal{Z}]$, $\nu > 0$ is a regularization parameter, and superscripts denote column indices. We then define our estimator for $\bar{\mathcal{B}}_{\mathbf{x}}$ as

$$\widetilde{\mathcal{B}}_{\mathbf{x}} := (\widetilde{\mathcal{U}}^\top(\widetilde{\mathcal{C}}_{3,1|2}\phi(\mathbf{x})))(\widetilde{\mathcal{U}}^\top \widetilde{\mathcal{C}}_{2,1})^\dagger \tag{6.27}$$

$$= m \sum_l B_l \Big(\big(M_2 + \nu m I\big)^{-1} \mathcal{Z}_2^\top \phi(\mathbf{x}) \Big)_l, \tag{6.28}$$

where $B_l \in \mathbb{R}^{k \times k}$ is a linear transformation of $Q_{\cdot\cdot l} \in \mathbb{R}^{m \times m}$, the lth slice of Q along the third dimension:

$$B_l := \widetilde{\mathcal{U}}^\top \mathcal{Z}_3 Q_{\cdot\cdot l} \mathcal{Z}_1^\top (\widetilde{\mathcal{U}}^\top \widetilde{\mathcal{C}}_{2,1})^\dagger. \tag{6.29}$$

Note that in the usual setting of learning from dynamic data, the third-order tensor Q is diagonal and B_l becomes a rank-one matrix, so (6.28) reduces to the estimator proposed by Song et al. [2010].

The definitions above naturally lead to an estimation procedure that first estimates Q and then applies (6.29) to estimate B_l. However, such a procedure involves m^3 variables when the quantities of interest consist of only km^2 variables. We thus propose to estimate B_l's directly. Viewing (6.29) as the solution to

$$\begin{aligned}
&\arg\min_{B_l} \|Q_{\cdot\cdot l} - \widetilde{U} B_l \widetilde{V}^\top\|_F^2, \quad \text{where} \\
\widetilde{U} &:= (\widetilde{\mathcal{U}}^\top \mathcal{Z}_3)^\dagger = (DA^\top \mathcal{Z}_2^\top \mathcal{Z}_3)^\dagger = (DA^\top M_{23})^\dagger, \\
\widetilde{V}^\top &:= (\mathcal{Z}_1^\top (\widetilde{\mathcal{U}}^\top \widetilde{\mathcal{C}}_{2,1})^\dagger)^\dagger = (M_1 \widetilde{P}^\top M_2 AD\Omega^{-1})^\dagger,
\end{aligned}$$

we propose to estimate B_l's by the following:

$$\arg\min_{\{B_l\}} \quad \frac{1}{2}\|\widetilde{\mathcal{C}}_{3,1,2}(\{B_l\}) - \widehat{\mathcal{C}}_{3,1,2}\|^2_{\mathcal{G}\otimes\mathcal{G}\otimes\mathcal{G}} + \tag{6.30}$$
$$\frac{u}{2}\Big(\|\widetilde{\mathcal{C}}_{3,\cdot,2}(\{B_l\}) - \widetilde{\mathcal{C}}_{2,1}\|^2_{\mathcal{G}\otimes\mathcal{G}} + \|\widetilde{\mathcal{C}}_{\cdot,1,2}(\{B_l\})^\top - \widetilde{\mathcal{C}}_{2,1}\|^2_{\mathcal{G}\otimes\mathcal{G}}\Big)$$

in which

$$\widetilde{\mathcal{C}}_{3,1,2}(\{B_l\}) := \sum_{i,j,l}(\widetilde{U}B_l\widetilde{V}^\top)_{ij}\mathcal{Z}_3^i \otimes \mathcal{Z}_1^j \otimes \mathcal{Z}_2^l, \tag{6.31}$$

$$\widetilde{\mathcal{C}}_{3,\cdot,2}(\{B_l\}) := \sum_{i,j,l}(\widetilde{U}B_l\widetilde{V}^\top)_{ij}\mathcal{Z}_3^i \otimes \mathbf{f}^\top\mathcal{Z}_1^j \otimes \mathcal{Z}_2^l, \tag{6.32}$$

$$\widetilde{\mathcal{C}}_{\cdot,1,2}(\{B_l\}) := \sum_{i,j,l}(\widetilde{U}B_l\widetilde{V}^\top)_{ij}\mathbf{f}^\top\mathcal{Z}_3^i \otimes \mathcal{Z}_1^j \otimes \mathcal{Z}_2^l. \tag{6.33}$$

Again, our estimation objective consists of a squared error term on the observed tri-variance and two regularization terms on the deviation of the marginals $\widetilde{\mathcal{C}}_{3,\cdot,2}$ and $\widetilde{\mathcal{C}}^\top_{\cdot,1,2}$ from our estimated covariance $\widetilde{\mathcal{C}}_{2,1}$. As shown in Appendix C.2, we use kernel tricks to re-write the objective function (6.30) in terms of finite-dimensional quantities. Moreover, by re-defining the notation B to be a k^2-by-m matrix whose l-th column denotes the column concatenation of the k-by-k matrix B_l, we obtain the following succinct form of (6.30) (dropping constants):

$$\min_B \quad \frac{1}{2}\mathrm{Tr}(B^\top CBM_2) - \mathrm{Tr}(J^\top B) \tag{6.34}$$

with an analytical solution $C^{-1}JM_2^{-1}$, where C and J are defined[4] in Appendix C.2.

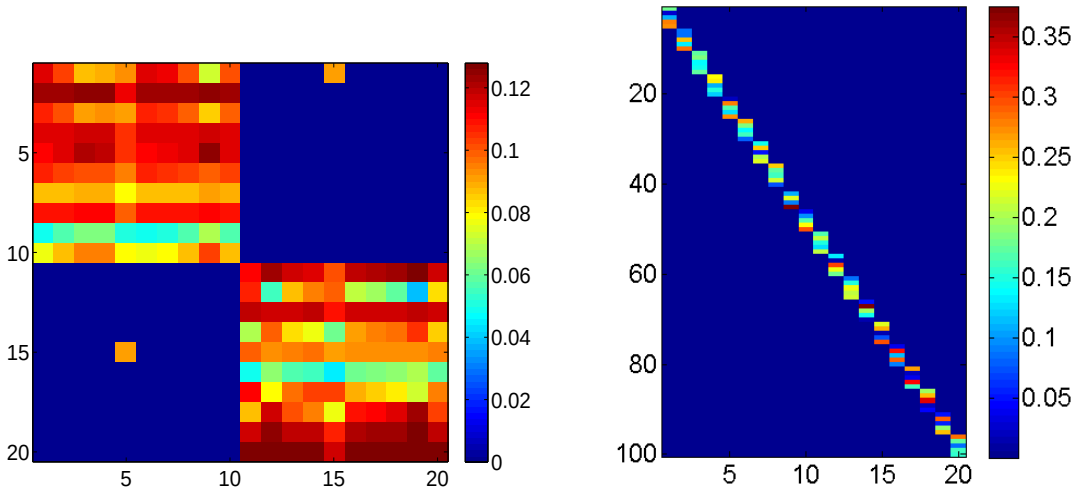
6.3 Experiments

We compare our proposed methods with the original spectral algorithms (Section 6.1) that only use sequence data. In the case of discrete observations we conduct a simulation study, and we apply the algorithms for continuous observations to an activity monitoring dataset.

6.3.1 Simulation

We create a discrete HMM with 20 states and 100 observation symbols. The state transition probability matrix is of rank nearly 7. The heatmaps of the state transition probability and the state-conditioned observation probability matrices are in Figures 6.1(a) and 6.1(b). From this HMM we generate 50 datasets, each containing a training sequence of length 1000 initialized from the stationary distribution as the sequence data, a set of 10000 observations independently drawn from the stationary distribution as the non-sequence data, and a testing sequence of length 1000, also initialized from the stationary distribution. We set the dimension $k = 7$, and for the proposed estimate set $u = 100$ and $\lambda = 15$. We then perform filtering and prediction along the testing sequence. To give bounds on the prediction performance, we also give prediction results by the true observable representation and the stationary distribution.

[4]When the kernel is positive definite, it is easy to verify that both C and M_2 are invertible.



(a) State transition matrix (b) Observation matrix

Figure 6.1: Discrete HMM model parameters

Table 6.1: Paired t test results. Each cell shows the number of testing time points at which the row method outperforms the column method statistically significantly. The total number of testing time points is 999.

	true	proposed	sequence-only	stationary
true		827	999	975
proposed	0		999	470
sequence-only	0	0		0
stationary	0	0	999	

Figure 6.2 shows the median testing log-likelihood over 50 experiments at each testing time point. The proposed estimator outperforms the sequence-only estimator at most time points. For each pair among the four predictions, we performed paired t-tests of their testing likelihoods at all time points, and counted the number of time points at which one prediction outperforms the other statistically significantly. The results are in Table 6.1. The proposed estimator predicts better than the sequence-only estimator at all time points and the stationary distribution at many time points, but these two other methods never predict significantly better than the proposed method. It is surprising that the sequence-only estimator performs even worse than the stationary distribution. As pointed out by Siddiqi et al. [2010], the filtering and prediction steps (6.6) do not guarantee non-negativity of the probability estimates, especially when, as in the current experiment, there is few dynamic data. Indeed, we observe quite a few negative values in the sequence-only estimates and replace them with 10^{-12}. This is an indication of unreliable estimates leading to poor prediction. On the contrary, the proposed estimates almost always take non-negative values.

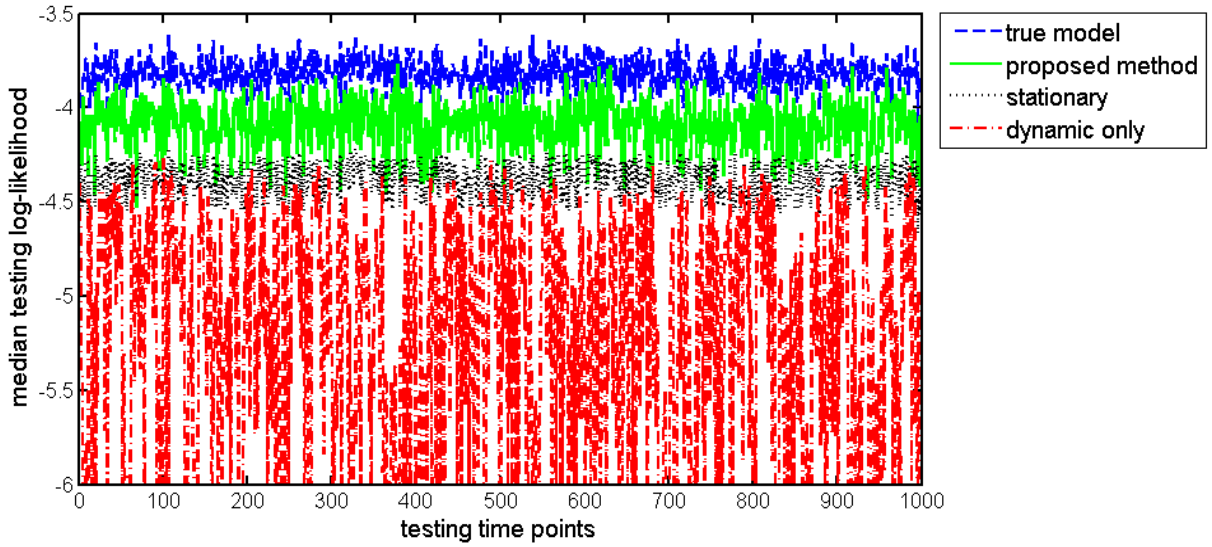


Figure 6.2: Median testing log-likelihood. The y-axis lower limit is set to -6 for better visualization; the red dashed line actually takes values as small as -17.

6.3.2 IMU Measurements of Human Activities

The PAMAP2 physical activity monitoring dataset [Reiss and Stricker, 2012] contains recordings of 18 different physical activities performed by 9 subjects wearing 3 inertial measurement units (IMUs) and a heart-rate monitor. Each subject follows a protocol to perform a sequence of activities with breaks in between. For our experiment we use data collected from subject 101 while walking and running. We focus our experiment on recordings from the three IMUs, and for each IMU only use the 3D-acceleration data (ms^{-2}) with scale $\pm$16g, as recommended by the authors, and the 3D-gyroscope data (rad/s), resulting in an observation space of $6 \times 3 = 18$ dimensions. Subject 101 performs walking and running for approximately 3.5 minutes each, and we discard the first and the last 10 seconds of data to remove transitioning between activities. To make the experiment more interesting, we break the IMU recordings into short segments of 10 seconds each and interleave the walking segments with the running ones to generate a sequence of alternating activities. The IMUs operate at a sampling frequency of 100Hz, so each segment has 1000 data points and the entire sequence has 39265 data points. We normalize each of the 18 dimensions to be zero-mean and standard deviation 1. Figure 6.3 shows one of the dimensions from the first 2000 data points, revealing significant differences between walking and running.

We take the last 4256 data points as the testing sequence, and generate 10 training datasets as follows. We randomly sample n triples of consecutive observations from the first 35000 data points as the sequence data, and another non-overlapping set of $m + m_S$ single observations as the non-sequence data, in which m points are used to form $\mathcal{Z}$ and the rest m_S points constitute $\mathcal{S}$ in the proposed algorithm. The values of $n, m,$ and m_S are: $n \in \{25, 50, 100, 200\}$, $m \in \{500, 1000\}$, and $m_S = 4000$. We use the Gaussian RBF kernel $\kappa(\mathbf{x}, \mathbf{x}') := \exp(\|\mathbf{x} - \mathbf{x}'\|^2/\sigma^2)$, and set σ^2 to be half of the median squared pairwise distances of the sequence data. The dimen-

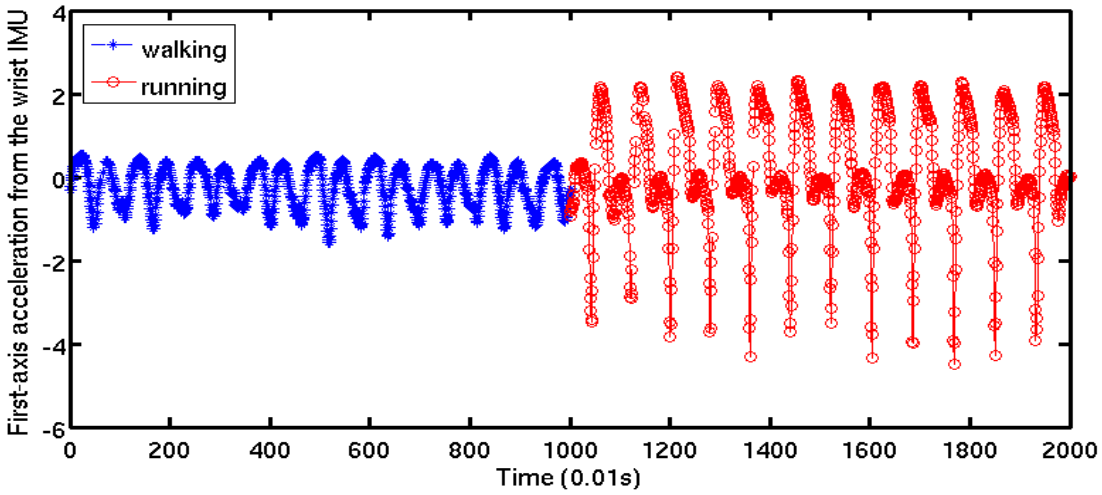
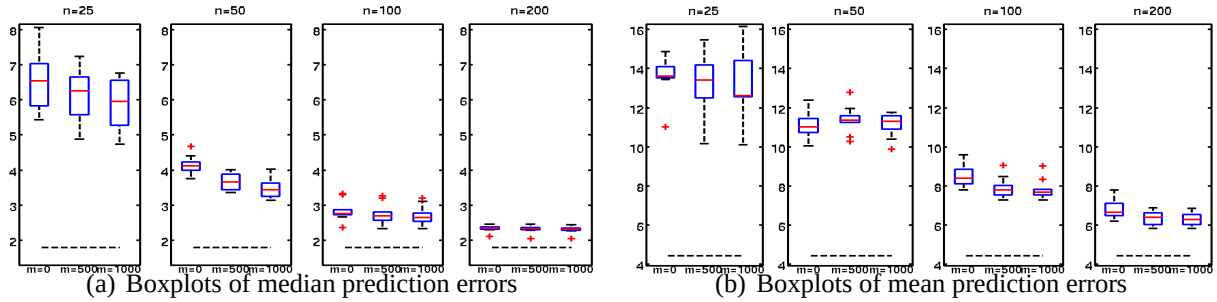


Figure 6.3: First-axis acceleration from the wrist IMU



(a) Boxplots of median prediction errors (b) Boxplots of mean prediction errors

Figure 6.4: Prediction performance on the IMU data. The black-dashed line is obtained by using $n = 5000$ dynamic data points, serving as a performance limit.

sion k, i.e., the number of top left singular vectors, is set to 20 for $n = 25$ and 50 for the rest. The proposed algorithm has three regularization parameters: u_P and λ in (6.19) and u_B in (6.34). We determine these parameters by minimizing 5-fold[5] cross validation error on the sequence data over a cube of values $(\log_2 u_P, \log_2 \lambda, \log_2 u_B) \in \{-8, -6, \ldots, 6\} \times \{-9, -7, \ldots, 1\} \times \{-5, -3, \ldots, 9\}$.

After learning the model parameters, we perform filtering and prediction along the testing sequence. As mentioned in Section 6.1, the Hilbert space embedding of the predictive distribution takes the form of a non-parametric density estimator thanks to the Gaussian RBF kernel, and we predict the next observation by selecting from $\mathcal{S}$, the m_S static data points, the one with the highest predictive density. For each predicted observation we compute the squared error against the true observation, and for each predicted sequence we take the median and the mean of the squared prediction errors as sequence-wise performance indicators. Figure 6.4(a) gives the boxplot of the 10 median prediction errors, showing that the proposed method of incorporating static data improves on the prediction performance more significantly when the sequence data size n is small. Figure 6.4(b) gives the boxplot of the 10 means, demonstrating a similar trend of

[5]We only split the sequence data but not the static data.

improvement except when $n = 50$. Looking more into that result, we find that it is the running part of the testing sequence the proposed method fails to predict better, possibly due to the more extreme values and changes in its IMU readings, as shown in Figure 6.3.

6.4 Discussions and Conclusions

We propose spectral learning algorithms for HMMs that incorporate static data as regularization. Experiments on synthetic and real human activities data demonstrate a clear advantage of using static data when sequence data is limited. There are several interesting directions for future work, including deriving theoretical guarantees for the proposed methods and solving real problems where sequence data is much more difficult to obtain than non-sequence data. In terms of methodology, a possible improvement is to combine the two stages in the proposed methods into one optimization problem, where the optimization variable is a three-way tensor representing the joint probability of observation triples, and the objective takes a similar form of an error term on sequence data plus regularization terms based on non-sequenced data. Given an estimate for the three-way probability tensor, lower-order probabilities can be easily obtained by marginalization, and then spectral learning algorithms in Section 6.1 can be applied. One advantage of such a procedure is that the estimates of the probability matrix and tensor are inherently consistent, and therefore the sub-spaces computed by spectral decomposition are optimal with respect to both, whereas in the proposed two-stage methods, the sub-spaces are optimal with respect to only the estimated joint probability matrix. The downside is obviously the optimization in the space of three-way tensors, which is computationally intensive in terms of both time and storage.

Although not explicitly described in this chapter, it is possible to extend the regular sequence-based EM learning algorithm for HMMs to make use of non-sequence data drawn from the stationary distribution. More specifically, such non-sequence data can be easily incorporated into the EM estimation procedure for parameters in the observation model, e.g., the state-specific mean observation vectors and noise covariances in a Gaussian observation model, because these parameters are time-independent. However, as with the regular EM approach, finding a good local optimum is always an issue and may require a lot of tuning.

Chapter 7

Learning Bi-clustered Vector Auto-regressive Model

In this chapter we return to the usual setting of learning from sequence data, and consider learning structured Vector Auto-regressive (VAR) models. Although not directly related to the main theme of the thesis, the methods developed here, as we explain later, can benefit learning from non-sequence data. Our motivation is from the use of VARs for analyzing the temporal dependencies in multivariate time series data, known as *Granger causality*[1] [Granger, 1969]. For example, recently researchers in computational biology, using ideas from sparse linear regression, developed sparse estimation techniques for VAR models [Fujita et al., 2007; Lozano et al., 2009; Shojaie et al., 2011] to learn from high-dimensional genomic time series a small set of pairwise, directed interactions, referred to as gene regulatory networks, some of which lead to novel biological hypotheses.

While individual edges convey important information about interactions, it is often desirable to obtain an aggregate and more interpretable description of the network of interest. One useful set of tools for this purpose are graph clustering methods [Schaeffer, 2007], which identify groups of nodes or vertices that have similar types of connections, such as a common set of neighboring nodes in undirected graphs, and shared parent or child nodes in directed graphs. These methods have been applied in the analysis of various types of networks, such as [Girvan and Newman, 2002], and play a key role in graph visualization tools [Herman et al., 2000].

Motivated by the wide applicability of the above two threads of work and the observation that their goals are tightly coupled, we develop a methodology that integrates both types of analyses, estimating the underlying Granger causal network and its clustering structure *simultaneously*. One can image that such a structure, once estimated, could be used as prior knowledge for other learning tasks in the same domain, and as suggested in Chapter 3, such prior knowledge may aid learning VARs from non-sequence data by providing better regularization of the model.

In this chapter we use the following notation for a first-order p-dimensional VAR model:

$$\mathbf{x}_{(t)} \;=\; \mathbf{x}_{(t-1)}A + \boldsymbol{\epsilon}_{(t)}, \quad \boldsymbol{\epsilon}_{(t)} \;\sim\; \mathcal{N}(\mathbf{0}, \sigma^2 I), \tag{7.1}$$

[1] More precisely, *graphical* Granger causality for more than two time series.

where $\mathbf{x}_{(t)} \in \mathbb{R}^{1\times p}$ denotes the vector of variables observed at time t, $A \in \mathbb{R}^{p\times p}$ is known as the transition matrix, whose non-zero entries encode Granger-causal relations among the variables, and $\boldsymbol{\epsilon}_{(t)}$'s denote independent noise vectors drawn from a zero-mean Gaussian with a spherical covariance $\sigma^2 I$. Our goal is to obtain a transition matrix estimate $\widehat{A}$ that is both *sparse*, leading directly to a Granger-causal network, and *clustered* so that variables sharing a similar set of connections are grouped together. Since the rows and the columns of A indicate different roles of the variables, the former revealing how variables affect themselves and the latter showing how variables get affected, we consider the more general *bi-clustering* setting, which allows two different sets of clusters for rows and columns, respectively. We take a nonparametric Bayesian approach, placing over A a nonparametric bi-clustered prior and carrying out full posterior inferences via a blocked Gibbs sampling scheme. Our simulation study demonstrates that when the underlying VAR model exhibits a clear bi-clustering structure, our proposed method improves over some natural alternatives, such as adaptive sparse learning methods [Zou, 2006] followed by bi-clustering, in terms of model estimation accuracy, clustering quality, and forecasting capability. More encouragingly, on a real-world T-cell activation gene expression time series data set [Rangel et al., 2004] our proposed method finds an interesting bi-clustering structure, which leads to a biologically more meaningful interpretation than those by some state-of-the art time series clustering methods.

Before introducing our method, we briefly discuss related work in Section 7.1. Then we define our bi-clustered prior in Section 7.2, followed by our sampling scheme for posterior inferences in Section 7.3. Lastly, we report our experimental results in Section 7.4 and conclude with Section 7.5.

7.1 Related work

There has been a lot of work on sparse estimation of Granger-causal networks under VAR models, and perhaps even more on graph clustering. However, to the best of our knowledge, none of them has considered the simultaneous learning scheme we propose here. Some of the more recent sparse VAR estimation work [Lozano et al., 2009; Shojaie et al., 2011] takes into account dependency further back in time and can even select the right length of history, known as the order of the VAR model. While focusing on first-order VAR models, we observe that it is possible to extend our method to learn higher-order bi-clustered VAR models, where the same bi-clustering structure is shared by all the time-lagged transition matrices, an extension to the grouped graphical Granger modeling approach of Lozano et al. [2009].

Another large body of related work [e.g., Busygin et al., 2008; Meeds and Roweis, 2007; Porteous et al., 2008] concerns bi-clustering (or co-clustering) a data matrix, which usually consists of relations between two sets of objects, such as user ratings on items, or word occurrences in documents. Most of this work models data matrix entries by mixtures of distributions with different *means*, representing, for example, different mean ratings by different user groups on item groups. In contrast, common regularization schemes or prior beliefs for VAR estimation usually assume zero-mean entries for the transition matrix, biasing the final estimate towards being stable. Following such a practice, our method models transition matrix entries as *scale mixtures* of zero-mean distributions.

Finally, clustering time series data has been an active research topic in a number of areas, in particular computational biology. However, unlike our Granger causality based bi-clustering method, most of the existing work, such as [Cooke et al., 2011; Ramoni et al., 2002] and the references therein, focus on grouping together *similar* time series, with a wide range of similarity measures from simple linear correlation to complicated Gaussian process based likelihood scores. Differences between our method and existing similarity-based approaches are demonstrated in Section 7.4 through both simulations and experiments on real data.

7.2 Bi-clustered prior

We treat the transition matrix $A \in \mathcal{R}^{p \times p}$ as a random variable and place over it a "bi-clustered" prior, as defined by the following generative process:

$$\begin{aligned} \boldsymbol{\pi}_u &\sim \text{Stick-Break}(\alpha_u), & \boldsymbol{\pi}_v &\sim \text{Stick-Break}(\alpha_v), \\ \{u_i\}_{1 \le i \le p} &\overset{i.i.d}{\sim} \text{Multinomial}(\boldsymbol{\pi}_u), & \{v_j\}_{1 \le j \le p} &\overset{i.i.d}{\sim} \text{Multinomial}(\boldsymbol{\pi}_v), \\ \{\lambda_{kl}\}_{1 \le k,l \le \infty} &\overset{i.i.d.}{\sim} \text{Gamma}(h, c), & & \qquad (7.2) \\ A_{ij} &\sim \text{Laplace}(0, 1/\lambda_{u_i v_j}), \quad 1 \le i, j \le p. & & \qquad (7.3) \end{aligned}$$

The process starts by drawing row and column mixture proportions $\boldsymbol{\pi}_u$ and $\boldsymbol{\pi}_v$ from the "stick-breaking" distribution [Sethuraman, 1994], denoted by $\text{Stick-Break}(\alpha)$ and defined on an infinite-dimensional simplex as follows:

$$\begin{aligned} \beta_k &\sim \text{Beta}(1, \alpha), \\ \pi_k &:= \beta_k \prod_{m<k} (1 - \beta_m), \quad 1 \le k \le \infty, \end{aligned} \qquad (7.4)$$

where $\alpha > 0$ controls the average length of pieces broken from the stick, and may take different values α_u and α_v for rows and columns, respectively. Such a prior allows for an infinite number of mixture components or clusters, and lets the data decide the number of *effective* components having positive probability masses, thereby increasing modeling flexibility. The process then samples row-cluster and column-cluster indicator variables u_i's and v_j's from mixture proportions $\boldsymbol{\pi}_u$ and $\boldsymbol{\pi}_v$, and for the k-th row-cluster and the l-th column-cluster draws an inverse-scale, or rate parameter λ_{kl} from a Gamma distribution with shape parameter h and scale parameter c. Finally, the generative process draws each matrix entry A_{ij} from a zero-mean Laplace distribution with inverse scale $\lambda_{u_i v_j}$, such that entries belonging to the same bi-cluster share the same inverse scale, and hence represent interactions of similar *magnitudes*, whether positive or negative.

The above bi-clustered prior subsumes a few interesting special cases. In some applications researchers may believe the clusters should be symmetric about rows and columns, which corresponds to enforcing $\mathbf{u} = \mathbf{v}$. If they further believe that within-cluster interactions should be stronger than between-cluster ones, they may adjust accordingly the hyper-parameters in the Gamma prior (7.2), or as in the group sparse prior proposed by Marlin et al. [2009] for Gaussian precision estimation, simply require all within-cluster matrix entries to have the same inverse

Algorithm 7.1 Blocked Gibbs Sampler

Input: Data X and Y, hyper-parameters h, c, α_u, α_v, and initial values $A^{(0)}, L^{(0)}$, $\mathbf{u}^{(0)}$, $\mathbf{v}^{(0)}$, $(\sigma^{(0)})^2$

Output: Samples from the full joint posterior $p(A, L, \mathbf{u}, \mathbf{v}, \sigma^2 \mid X, Y)$

Set iteration $t = 1$

repeat

  for $i = 1$ **to** p **do**

    $A_i^{(t)} \sim p(A_i \mid A_{1:(i-1)}^{(t)}, A_{(i+1):p}^{(t-1)}, \mathbf{u}^{(t-1)}, \mathbf{v}^{(t-1)}, (\sigma^{(t-1)})^2), L^{(t-1)}, X, Y)$

  end for

  for $i = 1$ **to** p **do**

    $u_i^{(t)} \sim p(u_i \mid A^{(t)}, \mathbf{u}_{1:(i-1)}^{(t)}, \mathbf{u}_{(i+1):p}^{(t-1)}, \mathbf{v}^{(t-1)}, (\sigma^{(t-1)})^2, L^{(t-1)}, X, Y)$

  end for

  for $j = 1$ **to** p **do**

    $v_j^{(t)} \sim p(v_j \mid A^{(t)}, \mathbf{u}^{(t)}, \mathbf{v}_{1:(j-1)}^{(t)}, \mathbf{v}_{(j+1):p}^{(t-1)}, (\sigma^{(t-1)})^2, L^{(t-1)}, X, Y)$

  end for

  $(\sigma^{(t)})^2 \sim p(\sigma^2 \mid A^{(t)}, \mathbf{u}^{(t)}, \mathbf{v}^{(t)}, L^{(t-1)}, X, Y)$

  $L^{(t)} \sim p(L \mid A^{(t)}, \mathbf{u}^{(t)}, \mathbf{v}^{(t)}, (\sigma^{(t)})^2, X, Y)$

  Increase iteration t

until convergence

Notations: superscript (t) denotes iteration, A_i denotes the i-th row of A, $A_{i:j}$ denotes the sub-matrix in A from the i-th until the j-th row, and $\mathbf{u}_{i:j}$ denotes $\{u_n\}_{i \le n \le j}$.

scale constrained to be smaller than the one shared by all between-cluster entries. Our inference scheme detailed in the next section can be easily adapted to all these special cases.

There can be interesting generalizations as well. For example, depending on the application of interest, it may be desirable to distinguish positive interactions from negative ones, so that a bi-cluster of transition matrix entries possess not only similar strengths, but also *consistent signs*. However, such a generalization requires a more delicate per-entry prior and therefore a more complex sampling scheme, which we leave as an interesting direction for future work.

7.3 Posterior inference

Let L denote the collection of λ_{kl}'s, $\mathbf{u}$ and $\mathbf{v}$ denote $\{u_i\}_{1 \le i \le p}$ and $\{v_j\}_{1 \le j \le p}$, respectively. Given one or more time series, collectively denoted as matrices X and Y whose rows represent successive pairs of observations, i.e.,

$$Y_i \;=\; X_i A + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \;\sim\; \mathcal{N}(\mathbf{0}, \sigma^2 I),$$

we aim to carry out posterior inferences about the transition matrix A, and row and column cluster indicators $\mathbf{u}$ and $\mathbf{v}$. To do so, we consider sampling from the full joint posterior $p(A, L, \mathbf{u}, \mathbf{v}, \sigma^2 \mid X, Y)$, and develop an efficient blocked Gibbs sampler outlined in Algorithm 7.1. Starting with some reasonable initial configuration, the algorithm iteratively samples rows of A, row

and column-cluster indicator variables $\mathbf{u}$ and $\mathbf{v}$, the noise variance[2] σ^2, and the inverse scale parameters L from their respective conditional distributions. Next we describe in more details sampling from those conditional distributions.

7.3.1 Sampling the transition matrix A

Let A_{-i} denote the sub-matrix of A excluding the i-th row, X'_i and X'_{-i} denote the i-th column of X and the sub-matrix of X excluding the i-th column. Algorithm 7.1 requires sampling from the following conditional distribution:

$$p(A_i \mid A_{-i}, \mathbf{u}, \mathbf{v}, \sigma^2, L, X, Y) \;\propto\; \prod_{1 \le j \le p} \mathcal{N}(A_{ij} \mid \mu_{ij}, \sigma_i^2) \mathsf{Laplace}(A_{ij} \mid 0, 1/\lambda_{u_i v_j}),$$

where

$$\mu_{ij} \;:=\; (X'_i / \|X'_i\|_2^2)^\top (Y - X'_{-i} A_{-i})'_j, \quad \sigma_i^2 \;:=\; \sigma^2 / \|X'_i\|^2.$$

Therefore, all we need is sampling from univariate densities of the form:

$$f(x) \;\propto\; \mathcal{N}(x \mid \mu, \sigma^2) \mathsf{Laplace}(x \mid 0, 1/\lambda), \tag{7.5}$$

whose c.d.f. $F(x)$ can be expressed in terms of the standard normal c.d.f. $\Phi(\cdot)$:

$$F(x) \;=\; \frac{C_1}{C} \Phi\Big(\frac{x^- - (\mu + \sigma^2\lambda)}{\sigma}\Big) + \frac{C_2}{C}\Big(\Phi\Big(\frac{x^+ - (\mu - \sigma^2\lambda)}{\sigma}\Big) - \Phi\Big(-\frac{\mu - \sigma^2\lambda}{\sigma}\Big)\Big),$$

where $x^- := \min(x, 0)$, $x^+ := \max(x, 0)$, and

$$\begin{aligned} C \;&:=\; C_1 \Phi\Big(-\frac{\mu + \sigma^2\lambda}{\sigma}\Big) + C_2\Big(1 - \Phi\Big(-\frac{\mu - \sigma^2\lambda}{\sigma}\Big)\Big), \\ C_1 \;&:=\; \frac{\lambda}{2} \exp\Big(\frac{\lambda(2\mu + \sigma^2\lambda)}{2}\Big), \quad C_2 := \frac{\lambda}{2} \exp\Big(\frac{\lambda(\sigma^2\lambda - 2\mu)}{2}\Big). \end{aligned}$$

We then sample from $f(x)$ with the inverse c.d.f. method. To reduce the potential sampling bias introduced by a fixed sampling schedule, we follow a random ordering of the rows of A in each iteration.

Algorithm 7.1 generates samples from the full joint posterior, but sometimes it is desirable to obtain a point estimate of A. One simple estimate is the (empirical) posterior mean; however, it is rarely sparse. To get a sparse estimate, we carry out the following "sample EM" step after Algorithm 7.1 converges:

$$\widehat{A}^{\text{Biclus-EM}} \;:=\; \arg\max_A \sum_t \log p(A \mid \mathbf{u}^{(t)}, \mathbf{v}^{(t)}, (\sigma^{(t)})^2, L^{(t)}, X, Y), \tag{7.6}$$

where t starts at a large number and skips some fixed number of iterations to give better-mixed and more independent samples. The optimization problem (7.6) is in the form of sparse least square regression, which we solve with a simple coordinate descent algorithm.

[2]Our sampling scheme can be easily modified to handle diagonal covariances.

7.3.2 Sampling row and cluster indicators

Since our sampling procedures for $\mathbf{u}$ and $\mathbf{v}$ are symmetric, we only describe the one for $\mathbf{u}$. It can be viewed as an instantiation of the general Gibbs sampling scheme studied by Meeds and Roweis [2007]. According to our model assumption, $\mathbf{u}$ is independent of the data X, Y and the noise variance σ^2 conditioned on all other random variables. Moreover, under the stick-breaking prior (7.4) over the row mixture proportions $\boldsymbol{\pi}_u$ and some fixed $\mathbf{v}$, we can view $\mathbf{u}$ and the rows of A as cluster indicators and samples drawn from a Dirichlet process mixture model with $\mathsf{Gamma}(h, c)$ as the base distribution over cluster parameters. Finally, the Laplace distribution and the Gamma distribution are conjugate pairs, allowing us to integrate out the inverse scale parameters L and derive the following "collapsed" sampling scheme:

$$
\begin{aligned}
&p(u_i = k' \in \text{existing row-clusters} \mid A, \mathbf{u}_{-i}, \mathbf{v}) \\
\propto\ &\left(\prod_{k,l} \frac{\Gamma((N_{-i}[k] + \delta_{kk'})M[l] + h)/(\Gamma(h)c^h)}{\Big(\|A_{-i}[k,l]\|_1 + \delta_{kk'}\|A_i[l]\|_1 + 1/c\Big)^{(N_{-i}[k]+\delta_{kk'})M[l]+h}} \right) \frac{N_{-i}[k']}{p - 1 + \alpha_u}, \\
\\
&p(u_i = \text{a new row-cluster} \mid A, \mathbf{u}_{-i}, \mathbf{v}) \\
\propto\ &\left(\prod_{k,l} \frac{\Gamma(N_{-i}[k]M[l] + h)/(\Gamma(h)c^h)}{\Big(\|A_{-i}[k,l]\|_1 + 1/c\Big)^{N_{-i}[k]M[l]+h}} \cdot \frac{\Gamma(M[l] + h)/(\Gamma(h)c^h)}{\Big(\|A_i[l]\|_1 + 1/c\Big)^{M[l]+h}} \right) \frac{\alpha_u}{p - 1 + \alpha_u},
\end{aligned}
$$

where $\Gamma(\cdot)$ is the Gamma function, δ_{ab} denotes the Kronecker delta function, $N_{-i}[k]$ is the size of the k-th row-cluster excluding A_i, $M[l]$ is the size of the l-th column-cluster, and

$$
\|A_{-i}[k,l]\|_1 \ := \sum_{s \neq i, u_s = k, v_j = l} |A_{sj}|, \qquad \|A_i[l]\|_1 \ := \sum_{v_j = l} |A_{ij}|.
$$

As in the previous section, we randomly permute u_i's and v_j's in each iteration to reduce sampling bias, and also randomly choose to sample $\mathbf{u}$ or $\mathbf{v}$ first.

Just as with the transition matrix A, we may want to obtain point estimates of the cluster indicators. The usual empirical mean estimator does not work here because the cluster labels may change over iterations. We thus employ the following procedure:

1. Construct a similarity matrix S such that

$$
S_{ij} \ := \ \frac{1}{T} \sum_t \delta_{u_i^{(t)} u_j^{(t)}}, \qquad 1 \leq i, j \leq p,
$$

   where t selects iterations to approach mixing and independence as in (7.6), and T is the total number of iterations selected.

2. Run normalized spectral clustering [Ng et al., 2001] on S, with the number of clusters set according to the spectral gap of S.

7.3.3 Sampling noise variance and inverse scale parameters

On the noise variance σ^2 we place an inverse-Gamma prior with shape $a > 0$ and scale $\beta > 0$, leading to the following posterior:

$$\sigma^2 \mid A, X, Y \;\sim\; \text{I-Gamma}(a + pT/2, 2\|Y - XA\|_F^{-2} + \beta), \tag{7.7}$$

where T is the number of rows in X and $\|\cdot\|_F$ denotes the matrix Frobenius norm. Due to the conjugacy mentioned in the last section, the inverse scale parameters λ_{kl}'s have the following posterior:

$$\lambda_{kl} \mid A, \mathbf{u}, \mathbf{v} \;\sim\; \text{Gamma}(N[k]M[l] + h, (\|A[k,l]\|_1 + 1/c)^{-1}).$$

7.4 Experiments

We conduct both simulations and experiments on a real gene expression time series dataset, and compare the proposed method with two types of approaches:

Learning VAR by sparse linear regression, followed by bi-clustering

Unlike the proposed method, which makes inferences about the transition matrix A and cluster indicators jointly, this natural baseline method first estimates the transition matrix by adaptive sparse or L_1 linear regression [Zou, 2006]:

$$\widehat{A}^{L_1} \;:=\; \arg\min_A \frac{1}{2}\|Y - XA\|_F^2 + \lambda \sum_{i,j} \frac{|A_{ij}|}{|\widehat{A}^{\text{ols}}_{ij}|^\gamma}, \tag{7.8}$$

where $\widehat{A}^{\text{ols}}$ denotes the ordinary least-square estimator, and then bi-clusters $\widehat{A}^{L_1}$ by either the cluster indicator sampling procedure in Section 7.3.2 or standard clustering methods applied to rows and columns separately. We compare the proposed method and this baseline in terms of predictive capability, clustering performance, and in the case of simulation study, model estimation error.

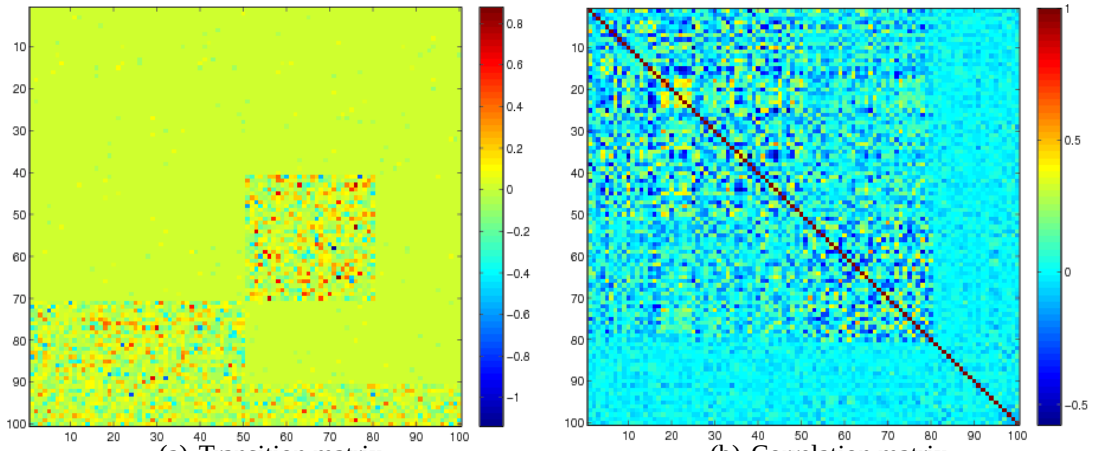
Clustering based on time series similarity

As described in Section 7.1, existing time series clustering methods are designed to group together time series that exhibit a similar behavior or dependency over time, whereas our proposed method clusters time series based on their (Granger) causal relations. We compare the proposed method with the time series clustering method proposed by Cooke et al. [2011], which models time series data by Gaussian processes and performs Bayesian Hierarchical Clustering [Heller and Ghahramani, 2005], achieving state-of-the art clustering performances on the real genes time series data used in Section 7.4.

7.4.1 Simulation

We generate a transition matrix A of size 100 by first sampling entries in bi-clusters:

$$A_{ij} \;\sim\; \begin{cases} \text{Laplace}(0, \sqrt{60}^{-1} i), & 41 \le i \le 70, 51 \le j \le 80, \\ \text{Laplace}(0, \sqrt{70}^{-1}), & 71 \le i \le 90, 1 \le j \le 50, \\ \text{Laplace}(0, \sqrt{110}^{-1}), & 91 \le i \le 100, 1 \le j \le 100, \end{cases} \tag{7.9}$$



(a) Transition matrix (b) Correlation matrix

Figure 7.1: Heat maps of the synthetic bi-clustered VAR

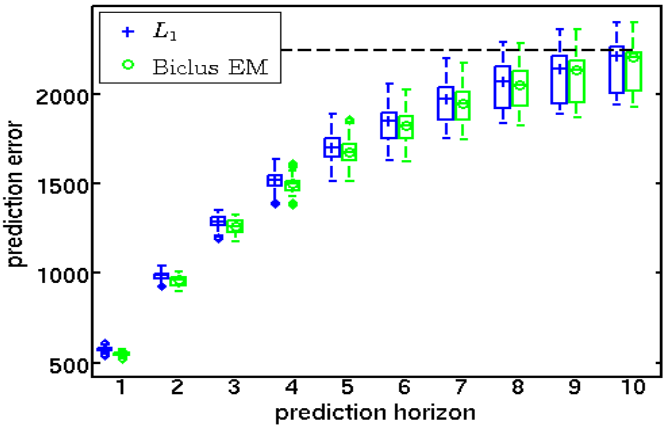


Figure 7.2: Prediction errors up to 10 time steps. Errors for longer horizons are close to those by the mean (zero) prediction, shown in black dashed line, and are not reported.

and then all the remaining entries from a sparse back-ground matrix:

$$A_{ij} = \begin{cases} B_{ij} & \text{if } |B_{ij}| \geq q_{98}\big(\{|B_{i'j'}|\}_{1\leq i',j'\leq 100}\big), \\ 0 & \text{otherwise}, \end{cases} \qquad i,j \text{ not covered in (7.9)},$$

where

$$\{B_{ij}\}_{1\leq i,j,\leq 100} \overset{i.i.d.}{\sim} \mathsf{Laplace}(0, (5\sqrt{200})^{-1})$$

and $q_{98}(\cdot)$ denotes the 98-th percentile. Figure 7.1(a) shows the heat map of the actual A we obtain by the above sampling scheme, showing clearly four row-clusters and three column-clusters. This transition matrix has the largest eigenvalue modulus of 0.9280, constituting a stable VAR model.

We then sample 10 independent time series of 50 time steps from the VAR model (7.1), with noise variance $\sigma^2 = 5$. We initialize each time series with an independent sample drawn from the

Table 7.1: Model estimation error on simulated data

	Normalized matrix error	Signed-support error
L_1	0.3133±0.0003	0.3012±0.0008
Biclus EM	**0.2419±0.0003**	**0.0662±0.0012**

stationary distribution of (7.1), whose correlation matrix is shown in Figure 7.1(b), suggesting that clustering based on correlations among time series may not recover the bi-cluster structure in Figure 7.1(a).

To compare the proposed method with the two baselines described in the beginning of Section 7.4, we repeat the following experiment 20 times: a random subset of two time series are treated as testing data, while the other eight time series are used as training data. For L_1 linear regression (7.8) we randomly hold out two time series from the training data as a validation set for choosing the best regularization parameter λ from $\{2^{-2}, 2^{-1}, \ldots, 2^{10}\}$ and weight-adaption parameter γ from $\{0, 2^{-2}, 2^{-1}, \ldots, 2^{2}\}$, with which the final $\widehat{A}^{L_1}$ is estimated from all the training data. To bi-cluster $\widehat{A}^{L_1}$, we consider the following:

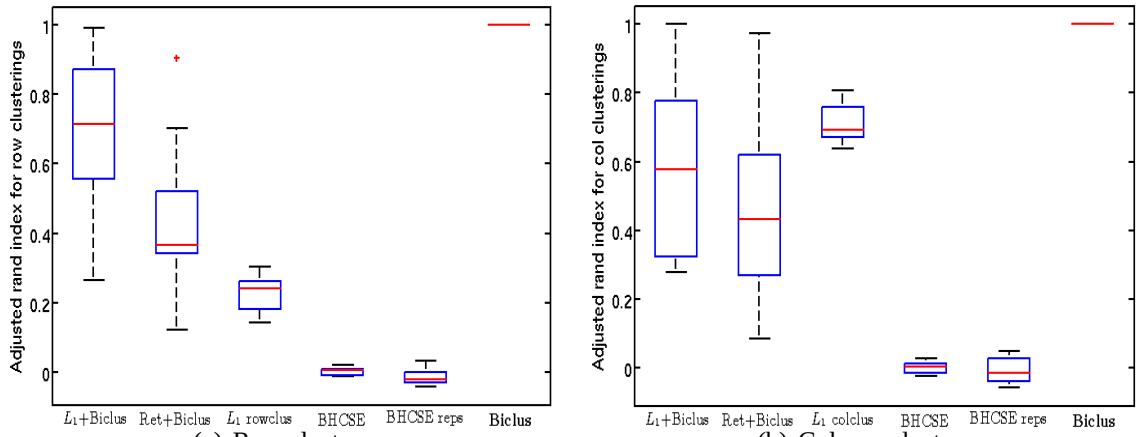
- **L_1+Biclus**: run the sampling procedure in Section 7.3.2 on $\widehat{A}^{L_1}$.
- **Refit+Biclus**: refit the non-zero entries of $\widehat{A}^{L_1}$ using least-square, and run the sampling procedure in Section 7.3.2.
- **L_1 row-clus** (**col-clus**): construct similarity matrices

$$S^u_{ij} := \sum_{1 \le s \le p} |\widehat{A}^{L_1}_{is}||\widehat{A}^{L_1}_{js}|, \quad S^v_{ij} := \sum_{1 \le s \le p} |\widehat{A}^{L_1}_{si}||\widehat{A}^{L_1}_{sj}|, \quad 1 \le i, j \le p.$$

  Then run normalized spectral clustering [Ng et al., 2001] on S^u and S^v, with the number of clusters set to 4 for rows and 3 for columns, respectively.

For the second baseline, Bayesian Hierarchical Clustering and Gaussian processes (GPs), we use the R package BHC (version 1.8.0) with the squared-exponential covariance for Gaussian processes, as suggested by the author of the package. Following Cooke et al. [2011] we normalize each time series to have mean 0 and standard deviation 1. The package can be configured to use replicate information (multiple series) or not, and we experiment with both settings, abbreviated as BHC-SE reps and BHC-SE, respectively. In both settings we give the BHC package the mean of the eight training series as input, but additionally supply BHC-SE reps a noise variance estimated from multiple training series to aid GP modeling.

In our proposed method, several hyper-parameters need to be specified. For the stick-breaking parameters α_u and α_v, we find that values in a reasonable range often lead to similar posterior inferences, and simply set both to be 1.5. We set the noise variance prior parameters in (7.7) to be $a = 9$ and $\beta = 10$. For the two parameters in the Gamma prior (7.2), we set $h = 2$ and $c = \sqrt{2p} = \sqrt{200}$ to bias the transition matrices sampled from the Laplace prior (7.3) towards being stable. Another set of inputs to Algorithm 7.1 are the initial values, which we set as follows: $A^{(0)} = \mathbf{0}$, $\mathbf{u}^{(0)} = \mathbf{v}^{(0)} = \mathbf{1}$, $(\sigma^{(0)})^2 = 1$, and $L^{(0)} = (h-1)c = \sqrt{200}$. We run Algorithm 7.1 and the sampling procedures for L_1+Biclus and Refit+Biclus for 2,500 iterations, and take samples in every 10 iterations starting from the 1,501-st iteration, at which the sampling algorithms have mixed quite well, to compute point estimates for A, $\mathbf{u}$ and $\mathbf{v}$ as described in Sections



(a) Row clusters (b) Column clusters

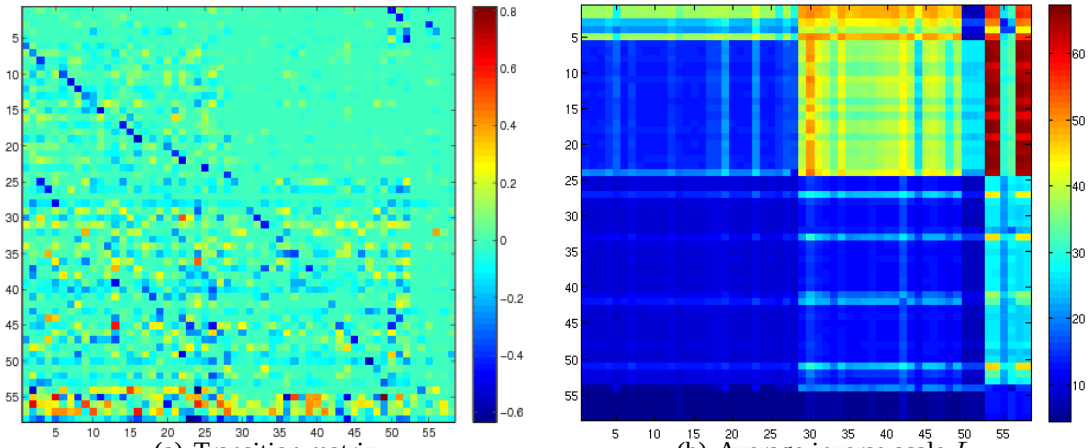
Figure 7.3: Adjusted Rand index on simulated data

7.3.1 and 7.3.2.

Figure 2 shows the squared prediction errors of L_1 linear regression (L_1) and the proposed method with a final sample EM step (Biclus EM) for various prediction horizons up to 10. Predictions errors for longer horizons are close to those by predicting the mean of the series, which is zero under our stable VAR model, and are not reported here. Biclus EM slightly outperforms L_1, and paired t tests show that the improvements for all 10 horizons are significant at a p-value ≤ 0.01. This suggests that when the underlying VAR model does have a bi-clustering structure, our proposed method can improve the prediction performance over adaptive L_1 regression, though by a small margin.

Another way to compare L_1 and Biclus EM is through model estimation error, and we report in Table 7.1 these two types of error:
Normalized matrix error: $\|\widehat{A} - A\|_F / \|A\|_F$,
Signed-support error: $\frac{1}{p^2} \sum_{1 \leq i,j \leq p} \mathbb{I}(\mathsf{sign}(\widehat{A}_{ij}) \neq \mathsf{sign}(A_{ij}))$.
Clearly, Biclus EM performs much better than L_1 in recovering the underlying model, and in particular achieves a huge gain in signed support error, thanks to its use of bi-clustered inverse scale parameters L.

Perhaps the most interesting is the clustering quality, which we evaluate by the *Adjusted Rand Index* [Hubert and Arabie, 1985], a common measure of similarity between two clusterings based on co-occurrences of object pairs across clusterings, with correction for chance effects. An adjusted Rand index takes the maximum value of 1 only when the two clusterings are identical (modulo label permutation), and is close to 0 when the agreement between the two clusterings could have resulted from two random clusterings. Figure 7.3 shows the clustering performances of different methods. The proposed method, labeled as Biclus, outperforms all alternatives greatly and always recovers the correct row and column clusterings. The two-stage baseline methods L_1+Biclus, Refit+Biclus, and L_1 row-clus (col-clus) make a significant amount of errors, but still recover moderately accurate clusterings. In contrast, the clusterings by the time-series similarity based methods, BHC-SE and BHC-SE reps, are barely better than random clusterings. To explain this, we first point out that BHC-SE and BHC-SE reps are



(a) Transition matrix (b) Average inverse scale L

Figure 7.4: Heat maps of the Biclus-EM estimate of A and the inverse scale parameters L averaged over posterior samples; rows and columns permuted according to clusters.

designed to model time series as noisy observations of deterministic, time-dependent "trends" or "curves" and to group similar curves together, but the time series generated from our stable VAR model all have zero expectation *at all time points* (not just *across time*). As a result, clustering based on similar trends may just be fitting noise in our simulated series. These results on clustering quality suggest that when the underlying cluster structure stems from (Granger) causal relations, clustering methods based on series similarity may give irrelevant results, and we really need methods that explicitly take into account dynamic interaction patterns, such as the one we propose here.

7.4.2 Modeling T-cell activation gene expression time series

We analyze a gene expression time series dataset[3] collected by Rangel et al. [2004] from a T-cell activation experiment. To facilitate the analysis, they pre-processed the raw data to obtain 44 replicates of 58 gene time series across 10 unevenly-spaced time points. Recently Cooke et al. [2011] carried out clustering analysis of these time series data, with their proposed Gaussian process (GP) based Bayesian Hierarchical Clustering (BHC) and quite a few other state-of-the art time series clustering methods. BHC, aided by GP with a cubic spline covariance function, gave the best clustering result as measured by the Biological Homogeneity Index (BHI) [Datta and Datta, 2006], which scores a gene cluster based on its number of gene pairs that share certain biological annotations (Gene Ontology terms).

To apply our proposed method, we first normalize each time series to have mean 0 and standard deviation 1 across both time points and replicates, and then "de-trend" the series by taking the first order difference, resulting in 44 replicates of 58 time series of gene expression differences across 9 time points. We run Algorithm 7.1 on this de-trended dataset, with all the hyper-parameters and initial values set in the same way as in our simulation study. In 3,000 iterations the algorithm mixes reasonably well; we let it run for another 2,000 iterations and take

[3] Available in the R package longitudinal.

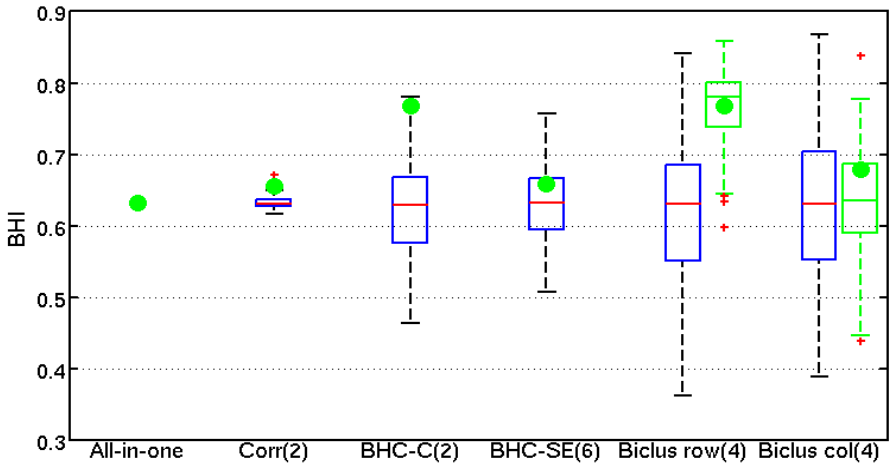


Figure 7.5: BHI. Green dots show BHIs of different methods; blue boxes are BHIs obtained by 200 random permutations of cluster labels by those methods; green boxes are BHIs computed on posterior cluster indicator samples from the proposed method. In parentheses are numbers of clusters given by different methods.

samples from every 10 iterations, resulting in 200 posterior samples, to compute point estimates for A, cluster indicators $\mathbf{u}$ and $\mathbf{v}$ as described in Sections 7.3.1 and 7.3.2. Figures 7.4(a) and 7.4(b) show the heat maps of the transition matrix point estimate and the inverse scale parameters λ_{ij}'s averaged over the posterior samples, with rows and columns permuted according to clusters, revealing a quite clear bi-clustering structure.

For competing methods, we use the GP based Bayesian Hierarchical Clustering (BHC) by Cooke et al. [2011], with two GP covariance functions: cubic spline (BHC-C) and squared-exponential (BHC-SE)[4]. We also apply the two-stage method L_1+Biclus described in our simulation study, but its posterior samples give an average of 15 clusters, which is much more than the number of clusters, around 4, from the spectral analysis described in Section 7.3.2, suggesting a high level of uncertainty in their posterior inferences about cluster indicators. We thus do not report their results here. The other two simple baselines are: Corr, standing for normalized spectral clustering on the correlation matrix of the 58 time series averaged over all 44 replicates, the number of clusters 2 determined by the spectral gap, and All-in-one, which simply puts all genes in one cluster.

Figure 7.5 shows the BHI scores[5] given by different methods, and higher-values indicate bettering clusterings. Biclus row and Biclus col respectively denote the row and column clusterings given by our method. To measure the significance of the clusterings, we report BHI scores computed on 200 random permutations of the cluster labels given by each method. For Biclus row and Biclus col, we also report the scores computed on the 200 posterior samples. All-in-one has a BHI score around 0.63, suggesting that nearly two-thirds of all gene pairs share some biolog-

[4]We did not report results obtained using replicate information because they are not better. Cluster labels are from `http://www.biomedcentral.com/1471-2105/12/399/additional`.

[5]We compute BHIs by the BHI function in the R package clValid (version 0.6-4) [Brock et al., 2008] and the database hgu133plus2.db (version 2.6.3), following Cooke et al. [2011].

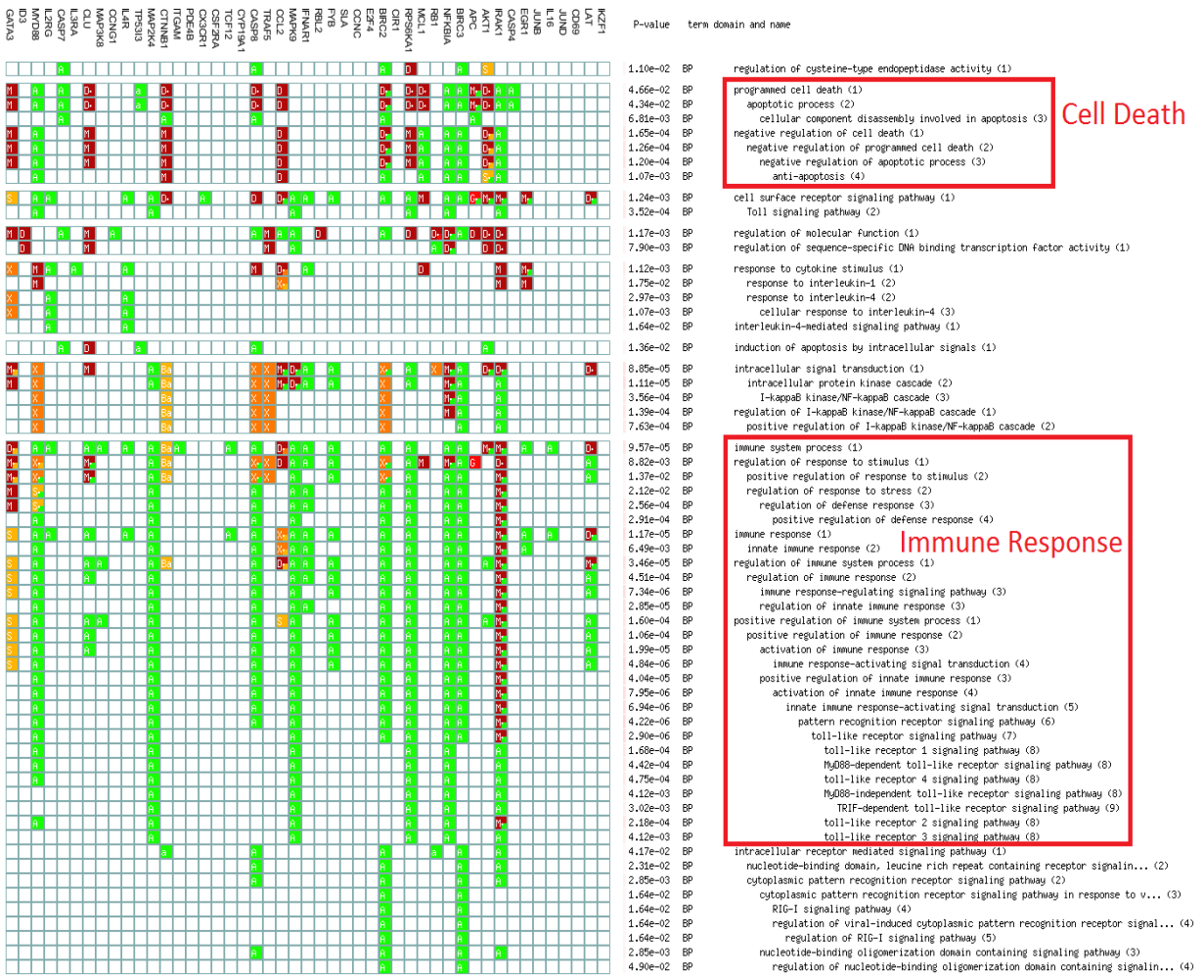
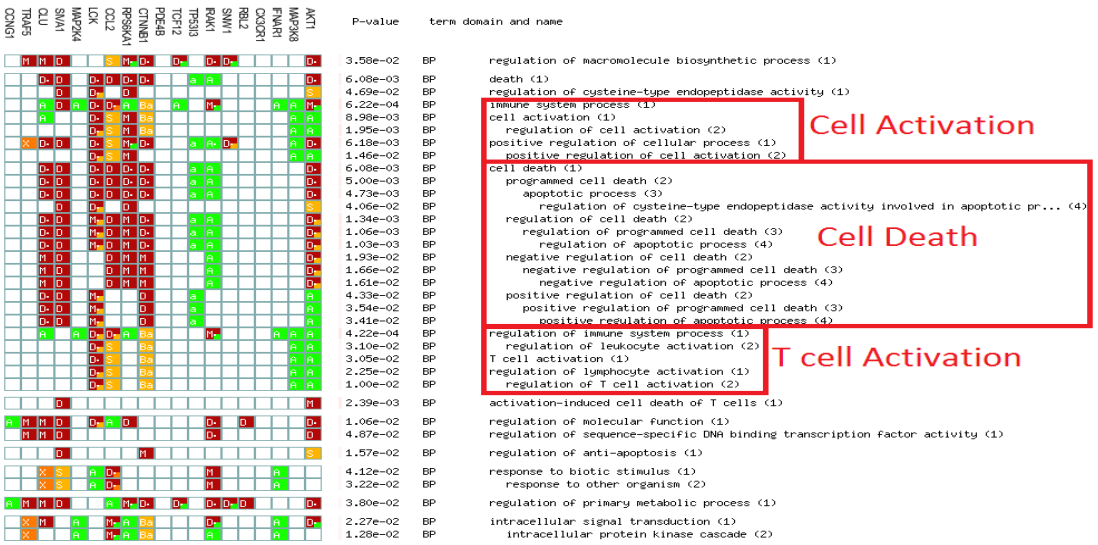
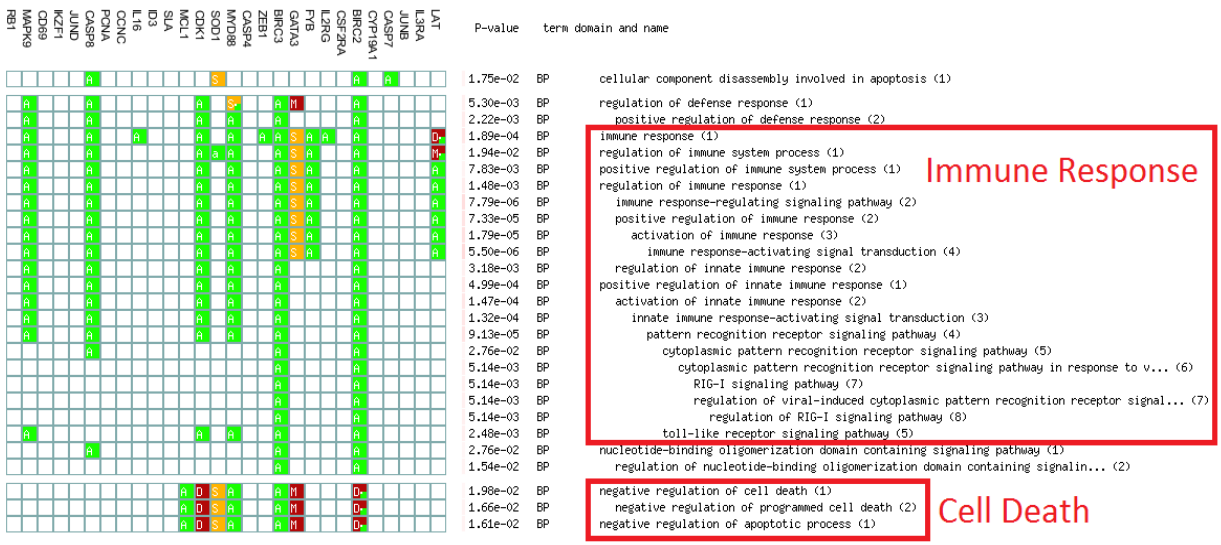


Figure 7.6: Gene functional profiling of the large BHC-C cluster

ical annotations. Corr puts genes into two nearly equal-sized clusters (28 and 30), but does not increase the BHI score much. In contrast, BHC-C and Biclus row achieve substantially higher scores, and both are significantly better than those by random permutations, showing that the improvements are much more likely due to the methods rather than varying numbers or sizes of clusters. We also note that even though Corr and BHC-C both give two clusters, the two BHC-C clusters have very different sizes (48 and 10), which cause a larger variance in their BHI distribution under random label permutations. Lastly, BHC-SE and Biclus col give lower scores that are not significantly better than random permutations. One possible explanation for the difference in scores by Biclus row and Biclus col is that the former bases itself on how genes *affect* one another while the latter on how genes *are affected* by others, and Gene Ontology terms, the biological annotations underlying the BHI function, describe more about genes' active roles or molecular functions in various biological processes than what influence genes.



(a) Second row cluster



(b) Third row cluster

Figure 7.7: Gene functional profiling of two large row clusters by the proposed method

Finally, to gain more understanding on the clusters by BHC-C and Biclus row, we conduct gene function profiling with the web-based tool **g:Profiler** [Reimand et al., 2011], which performs "statistical enrichment analysis to provide interpretation to user-defined gene lists." We select the following three options: *Significant only*, *Hierarchical sorting*, and *No electronic GO annotations*. For BHC-C, 4 out of 10 genes in the small cluster are found to be associated with the KEGG cell-cycle pathway (04110), but the other 6 genes are not mapped to collectively meaningful annotations. The profiling results of the large BHC-C cluster with 48 genes are in Figure 7.6; for better visibility we show only the Gene Ontology (GO) terms and high-light similar terms with red rectangles and tags. About a half of the terms are related to cell death and immune response, and the other half are lower-level descriptions involving, for example, signal-

Table 7.2: Contingency table of row and column clusterings

row \ col	1	2	3	4
1	0	0	3	2
2	17	2	0	0
3	10	17	0	2
4	1	2	0	2

ing pathways. For Biclus row, we report the profiling results of only the two larger clusters (the second and the third) in Figure 7.7, because the two smaller clusters, each containing 5 genes, are not mapped to collectively meaningful GO terms. Interestingly, the two large Biclus row clusters are associated with T-cell activation and immune response respectively, and together they cover 41 of the 48 genes in the large BHC-C cluster. This suggests that our method roughly splits the large BHC-C cluster into two smaller ones, each being mapped to a more focused set of biological annotations. Moreover, these Biclus profiling results, the heat map (Figure 7.4(a)), and the contingency table between the row and column clusters (Table 7.2) altogether constitute a nice resonance with the fact that T-cell activation results from, rather than leading to, the emergence of immune responses.

7.5 Conclusion

We develop a nonparametric Bayesian method to simultaneously infer sparse VAR models and bi-clusterings from multivariate time series data, and demonstrate its effectiveness via simulations and experiments on real T-cell activation gene expression time series, on which the proposed method finds a more biologically interpretable clustering than those by some state-of-the art methods. Future directions include modeling signs of transition matrix entries, generalizations to higher-order VAR models, and applications to other real time series.

Chapter 8

Conclusions and Future Directions

Motivated by the difficulties in collecting reliable time series data in a variety of modern dynamic modeling tasks, we study in this thesis the problem of learning dynamic models from data that lack time information but are easier to obtain. We observe that such non-sequence data can often be modeled as independent samples drawn from multiple, independent executions of the underlying dynamic process. Based on this assumption, we propose and study learning algorithms for several widely-used dynamic models, including fully observable linear and non-linear models, and Hidden Markov Models.

For fully observable models, we first point out some model identifiability issues in learning from non-sequence data. Then we develop several EM-type learning algorithms based on maximizing approximate likelihood, and for the case where a small amount of sequence data are available, we propose a novel penalized least square approach that uses both sequence and non-sequence data. Empirical evaluation on synthetic data and several real data sets, including gene expression and cell image time series, demonstrates that our proposed methods can learn reasonably accurate dynamic models with little or even no time information. However, we also observe several failure modes that are hard to overcome without extra information or assumption. This suggests that for the proposed methods to make impact in real applications, they should incorporate as much expert domain knowledge as possible. For example, knowing how the variables in the dynamic model might interact with one another can help the design of a better regularization scheme. This motivates us to develop methods for learning bi-clustered vector autoregressive models. Or, in some applications there might be partial ordering information about the data, which can provide constraints in our EM-type algorithms.

For Hidden Markov Models, we build on recent advances in spectral learning of latent variable models and propose tensor factorization based methods that guarantee consistent parameter estimation, under reasonable assumptions on the underlying HMM and the generative process of non-sequence data. These assumptions are inspired by spectral learning of topic models, but have a few key differences, such as conditions on the Dirichlet prior for the initial state distribution and modeling missing times as geometric random variables, that are specific to the HMM setting. Although these generative assumptions may not hold in observational data, they may be fairly easy to implement in some scientific experiments. We also consider the situation when little sequence data are available, and propose a spectral algorithm using both types of data, which outperforms sequence-only learning algorithms.

Going forward, one interesting direction is to investigate whether spectral methods can be used to consistently learn first-order observable models from non-sequence data, and under what conditions. As demonstrated in Chapter 5, it is primarily the discreteness, or more generally, non-Gaussianity of the hidden state space dynamics that leads to nice tensor structures in observable moments and easy characterization of assumptions ensuring unique parameter estimation. Therefore, in the case of first-order models with continuous observations, we expect that non-Gaussian initial distribution is needed for consistent spectral learning from non-sequence data. Moreover, it is likely that extra assumptions on the initial distribution, such as distinct variances or means in different dimensions, are required to eliminate the invariance to parameter permutation inherent in spectral learning.

Another important future direction is to make impact in real applications with our proposed methods. In order for that to happen, we expect to see various interesting extensions or modifications to our approaches that are tailored to the application of interest. In particular, our proposed modeling assumption of non-sequence data has several components that can be replaced to better suit different applications, such as the distributional assumption on the missing times and the observational noise model. More broadly, our work has demonstrated the possibility of using cross sectional data to aid longitudinal study. As mentioned in the very beginning of the thesis, it is common in medical and social sciences that cross sectional data are much easier to collect than longitudinal data, and yet a lot of cross sectional data were actually collected under some longitudinal effect. With advances in large-scale sensing technology, this situation will likely become more prevalent. We think there are several possibilities for our work to make concrete contributions. For example, at the initial stage of longitudinal studies, researchers often have to pose reasonable hypotheses to guide the design of experiments or data collection protocols. However, even forming good hypotheses may be difficult when the subject matter involves a complicated system. In this situation, our methods may serve as a good hypothesis generator, using cross sectional data that are available to produce possible models. Or, sometimes researchers may want some immediate, preliminary assessment even though the longitudinal study is still ongoing and only produced limited data. If there are abundant cross sectional data in the same domain, our methods of combining sequence and non-sequence data may be used to provide a reasonable estimate of the dynamic model under study.

In conclusion, our work demonstrates the possibility of learning dynamic models from data that lack time information, and we hope it stimulates more research in making better use of the large amount of cross sectional data brought by modern sensing technology.

Appendix A

A Variational EM algorithm for Learning HMMs from Non-sequence Data

Based on the generative process in Section 5.2.2, we derive a variational EM algorithm for parameter learning assuming the observation noise follows a spherical Gaussian with variance σ^2. The full joint probability of data and latent variables takes the following form:

$$
\begin{aligned}
&f(\{\mathbf{x}_i^j\}, \{\mathbf{h}_i^j\}, \{t_i^j\}, \{\mathbf{s}_i^j\}, \{\boldsymbol{\pi}_0^j\} \mid U, \sigma^2, P, r, \boldsymbol{\alpha}) \\
=&\prod_{j=1}^{N} \Bigg(\prod_{i=1}^{n} \Big(\prod_{l=1}^{k} \mathcal{N}(\mathbf{x}_i^j \mid U_l, \sigma^2 I)^{\mathbf{h}_{il}^j} \Big) \Big(\prod_{l',l} ((P^{t_i^j})_{l'l})^{\mathbf{h}_{il'}^j \mathbf{s}_{il}^j} \Big) \mathsf{Geometric}(t_i^j \mid r) \Big(\prod_{l} ((\boldsymbol{\pi}_0^j)_l)^{\mathbf{s}_{il}^j} \Big) \Bigg) \cdot \\
&\quad \mathsf{Dirichlet}(\boldsymbol{\pi}_0^j \mid \boldsymbol{\alpha}),
\end{aligned}
$$

in which we use super-script as set indices and sub-scripts as data indices within a set wherever appropriate. The goal is to maximize the marginal likelihood of the data w.r.t to the parameters. We begin by marginalizing over the latent times $\{t_i^j\}$:

$$
\begin{aligned}
&f(\{\mathbf{x}_i^j\}, \{\mathbf{h}_i^j\}, \{\mathbf{s}_i^j\}, \{\boldsymbol{\pi}_0^j\} \mid U, \sigma^2, T, \boldsymbol{\alpha}) \\
=&\prod_{j=1}^{N} \Bigg(\prod_{i=1}^{n} \Big(\prod_{l=1}^{k} \mathcal{N}(\mathbf{x}_i^j \mid U_l, \sigma^2 I)^{\mathbf{h}_{il}^j} \Big) \Big(\prod_{l',l} T_{l'l}^{\mathbf{h}_{il'}^j \mathbf{s}_{il}^j} \Big) \Big(\prod_{l} ((\boldsymbol{\pi}_0^j)_l)^{\mathbf{s}_{il}^j} \Big) \Bigg) \mathsf{Dirichlet}(\boldsymbol{\pi}_0^j \mid \boldsymbol{\alpha}),
\end{aligned}
$$

where T denotes the expected transition probability matrix. As in the tensor factorization approach, we recover P and r from the estimated T using the proposed search heuristics. Because the posterior distribution of the remaining latent variables still leads to an intractable E step, we employ the following factorized approximation:

$$
f(\{\mathbf{h}_i^j\}, \{\mathbf{s}_i^j\}, \{\boldsymbol{\pi}_0^j\} \mid \{\mathbf{x}_i^j\}, U, \sigma^2, T, \boldsymbol{\alpha}) \approx q(\{\mathbf{h}_i^j\}, \{\mathbf{s}_i^j\} \mid \{\Phi_j^i\}) q(\{\boldsymbol{\pi}_0^j\} \mid \{\boldsymbol{\beta}^j\}),
$$

where

$$
\begin{aligned}
q(\{\mathbf{h}_i^j\}, \{\mathbf{s}_i^j\} \mid \{\Phi_i^j\}) &:= \prod_{i,j,l',l} ((\Phi_i^j)_{l'l})^{\mathbf{h}_{il'}^j \mathbf{s}_{il}^j}, \quad \Phi_i^j \in [0,1]^{k \times k}, \\
q(\{\boldsymbol{\pi}_0^j\} \mid \{\boldsymbol{\beta}^j\}) &:= \prod_{j} \mathsf{Dirichlet}(\boldsymbol{\pi}_0^j \mid \boldsymbol{\beta}^j),
\end{aligned}
$$

and obtain the following lower bound on the log marginal likelihood:

$$
\begin{aligned}
&g(\{\Phi_i^j\}, \{\boldsymbol{\beta}^j\}, U, \sigma^2, T, \boldsymbol{\alpha}) \\
:=&\mathbb{E}_{\{\mathbf{h}_i^j\}, \{\mathbf{s}_i^j\} \mid \{\Phi_i^j\}, \{\boldsymbol{\pi}_0^j\} \mid \{\boldsymbol{\beta}^j\}} \left[\log \left(\frac{f(\{\mathbf{x}_i^j\}, \{\mathbf{h}_i^j\}, \{\mathbf{s}_i^j\}, \{\boldsymbol{\pi}_0^j\} \mid U, \sigma^2, T, \boldsymbol{\alpha})}{q(\{\mathbf{h}_i^j\}, \{\mathbf{s}_i^j\} \mid \{\Phi_i^j\}) q(\{\boldsymbol{\pi}_0^j\} \mid \{\boldsymbol{\beta}^j\})} \right) \right] \\
=&\mathbb{E}_{\{\mathbf{h}_i^j\}, \{\mathbf{s}_i^j\} \mid \{\Phi_i^j\}, \{\boldsymbol{\pi}_0^j\} \mid \{\boldsymbol{\beta}^j\}} \big[\log f(\{\mathbf{x}_i^j\}, \{\mathbf{h}_i^j\}, \{\mathbf{s}_i^j\}, \{\boldsymbol{\pi}_0^j\} \mid U, \sigma^2, T, \boldsymbol{\alpha}) \big] - \\
&\mathbb{E}_{\{\mathbf{h}_i^j\}, \{\mathbf{s}_i^j\} \mid \{\Phi_i^j\}} \big[\log q(\{\mathbf{h}_i^j\}, \{\mathbf{s}_i^j\} \mid \{\Phi_i^j\}) \big] - \mathbb{E}_{\{\boldsymbol{\pi}_0^j\} \mid \{\boldsymbol{\beta}_0^j\}} \big[\log q(\{\boldsymbol{\pi}_0^j\} \mid \{\boldsymbol{\beta}_0^j\}) \big] \\
=&\sum_{j,i,l,l'} (\Phi_i^j)_{ll'} (\log \mathcal{N}(\mathbf{x}_i^j \mid U_l, \sigma^2 I) + \log T_{ll'}) + \sum_{j,l} \Big(\sum_{i,l'} (\Phi_i^j)_{l'l} + \alpha_l - 1 \Big) \big(\psi(\boldsymbol{\beta}_l^j) - \psi(\beta_0^j) \big) \\
&- N \Big(\sum_{l} \log \Gamma(\alpha_l) - \log \Gamma(\alpha_0) \Big) - \sum_{j,i,l,l'} (\Phi_i^j)_{ll'} \log (\Phi_i^j)_{ll'} \\
&- \sum_{j,l} (\boldsymbol{\beta}_l^j - 1)(\psi(\boldsymbol{\beta}_l^j) - \psi(\beta_0^j)) + \sum_{j} \Big(\sum_{l} \log \Gamma(\boldsymbol{\beta}_l^j) - \log \Gamma(\beta_0^j) \Big),
\end{aligned}
$$

where $\psi(\cdot)$ is the digamma function. The variational EM algorithm then amounts to maximizing g iteratively, alternating between the following two steps until convergence:

Variational E-step

Holding the model parameters fixed, repeat the updates

$$\begin{aligned}(\Phi_i^j)_{ll'} &\propto \mathcal{N}(\mathbf{x}_i^j \mid U_l, \sigma^2 I) T_{ll'} \exp(\psi(\boldsymbol{\beta}_{l'}^j) - \psi(\beta_0^j)),\\ (\boldsymbol{\beta}^j)_l &= \sum_{i,l'} (\Phi_i^j)_{l'l} + \alpha_l,\end{aligned}$$

until convergence.

M-step

Holding the variational parameteres $\{\Phi_i^j\}$ and $\{\boldsymbol{\beta}_j\}$ fixed, update model parameters:

$$\begin{aligned}U_l &:= \frac{\sum_{j=1}^N \sum_{i=1}^n \sum_{l'=1}^k (\Phi_i^j)_{ll'} \mathbf{x}_i^j}{\sum_{j=1}^N \sum_{i=1}^n \sum_{l'=1}^k (\Phi_i^j)_{ll'}},\\ \sigma^2 &:= \frac{\sum_{j=1}^N \sum_{i=1}^n \sum_{l,l'} (\Phi_i^j)_{ll'} \|\mathbf{x}_i^j - U_l\|^2}{Nnm},\\ T_{ll'} &:= \frac{\sum_{j=1}^N \sum_{i=1}^n (\Phi_i^j)_{ll'}}{\sum_{j=1}^N \sum_{i=1}^n \sum_{l=1}^k (\Phi_i^j)_{ll'}},\\ \boldsymbol{\alpha} &:= \arg\max_{\{\alpha_l \geq 0\}} \sum_{j=1}^N \sum_{l=1}^k (\alpha_l - 1)(\psi(\boldsymbol{\beta}_l^j) - \psi(\beta_0^j)) - N\Big(\sum_{l=1}^k \log\Gamma(\alpha_l) - \log\Gamma(\alpha_0)\Big).\end{aligned}$$

The update for $\boldsymbol{\alpha}$ is a convex optimization problem whose inverse Hessian can be computed in linear time Blei et al. [2003].

Finally, we have to match the columns of U with the columns of T. Note that the updates imply that the columns of U are aligned with the rows of T, so it suffices to match T's rows with its columns. Using the assumptions that $\boldsymbol{\alpha}/\alpha_0 = \boldsymbol{\pi}$ and $\pi_i \neq \pi_j \; \forall \; i \neq j$, we recover the matching by sorting $\boldsymbol{\alpha}/\alpha_0$ and $T\boldsymbol{\alpha}/\alpha_0$.

Appendix B

Proofs of Theorems in Chapter 5

B.1 Tensor structure in low-order moments

Here we give proofs of theorems on tensor structures in low-order moments of observable data. The proofs make repeated use the following facts:

- $T\boldsymbol{\pi} = \boldsymbol{\pi}$, i.e., the stationary state distribution is invariant under the expected transition probability matrix T.
- The missing time steps t_i's are independent of everything else.
- Conditioned on the initial state distribution $\boldsymbol{\pi}_0$, i.e., within the same set of data, the observations $\{\mathbf{x}_i\}$ are mutually independent, so are the hidden states $\{\mathbf{h}_i\}$ and the initial states $\{\mathbf{s}_i\}$.
- The low-order moments of the Dirichlet distribution have a special form (c.f. Appendix B.1 of Anandkumar et al. [2013]), which leads to the desired symmetric tensor structure.

B.1.1 Proof of Theorem 2

$$
\begin{aligned}
\mathbb{E}[\mathbf{x}_1] &= \mathbb{E}_{\boldsymbol{\pi}_0}\mathbb{E}[\mathbf{x}_1 \mid \boldsymbol{\pi}_0] \\
&= \mathbb{E}_{\boldsymbol{\pi}_0}\mathbb{E}[P^{t_1}\mathbf{s}_1 \mid \boldsymbol{\pi}_0] \\
&= \mathbb{E}_{\boldsymbol{\pi}_0}[\mathbb{E}[P^{t_1}]\boldsymbol{\pi}_0] \\
&= T\boldsymbol{\pi} \\
&= \boldsymbol{\pi}. \\
C_2 &= \mathbb{E}[\mathbf{x}_1\mathbf{x}_2^\top] \\
&= \mathbb{E}_{\boldsymbol{\pi}_0}\mathbb{E}[P^{t_1}\mathbf{s}_1\mathbf{s}_2^\top(P^{t_2})^\top \mid \boldsymbol{\pi}_0] \\
&= \mathbb{E}_{\boldsymbol{\pi}_0}[\mathbb{E}[P^{t_1}]\mathbb{E}[\mathbf{s}_1\mathbf{s}_2^\top \mid \boldsymbol{\pi}_0]\mathbb{E}[(P^{t_2})^\top]] \\
&= T\mathbb{E}_{\boldsymbol{\pi}_0}[\boldsymbol{\pi}_0\boldsymbol{\pi}_0^\top]T^\top \\
&= T\left(\frac{\mathsf{diag}(\boldsymbol{\pi})}{\alpha_0+1} + \frac{\alpha_0\boldsymbol{\pi}\boldsymbol{\pi}^\top}{\alpha_0+1}\right)T^\top \qquad \text{(B.1)} \\
&= \frac{T\mathsf{diag}(\boldsymbol{\pi})T^\top}{\alpha_0+1} + \frac{\alpha_0\boldsymbol{\pi}\boldsymbol{\pi}^\top}{\alpha_0+1}. \qquad \text{(B.2)} \\
C_3 &= \mathbb{E}[\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \mathbf{x}_3] \\
&= \mathbb{E}_{\boldsymbol{\pi}_0}\mathbb{E}[(P^{t_1}\mathbf{s}_1) \otimes (P^{t_2}\mathbf{s}_2) \otimes (P^{t_3}\mathbf{s}_3) \mid \boldsymbol{\pi}_0] \\
&= \mathbb{E}_{\boldsymbol{\pi}_0}\left[(T\boldsymbol{\pi}_0) \otimes (T\boldsymbol{\pi}_0) \otimes (T\boldsymbol{\pi}_0)\right] \\
&= \frac{\sum_i 2\pi_i T_i \otimes T_i \otimes T_i}{(\alpha_0+2)(\alpha_0+1)} + \frac{\alpha_0^2\boldsymbol{\pi} \otimes \boldsymbol{\pi} \otimes \boldsymbol{\pi}}{(\alpha_0+2)(\alpha_0+1)} \qquad \text{(B.3)} \\
&\quad + \frac{\alpha_0\left(\sum_{ij}\left(T_i \otimes T_i \otimes T_j + T_i \otimes T_j \otimes T_i + T_j \otimes T_i \otimes T_i\right)\pi_i\pi_j\right)}{(\alpha_0+2)(\alpha_0+1)} \\
&= \frac{\sum_i 2\pi_i T_i^{\otimes 3} - 2\alpha_0^2\boldsymbol{\pi}^{\otimes 3}}{(\alpha_0+2)(\alpha_0+1)} + \frac{\alpha_0(\boldsymbol{\pi} \otimes_3 C_2 + \boldsymbol{\pi} \otimes_2 C_2 + \boldsymbol{\pi} \otimes_1 C_2)}{\alpha_0+2}. \qquad \text{(B.4)}
\end{aligned}
$$

We obtain (B.1) and (B.3) by using the expressions of Dirichlet moments derived in Appendix B.1 of Anandkumar et al. [2013]. Re-arranging (B.2) and (B.4) leads to the adjusted moments M_2 and M_3.

B.1.2 Proof of Theorem 4

$$
\begin{aligned}
V_1 &:= \mathbb{E}[\mathbf{x}_1] \\
&= \mathbb{E}[U\mathbf{h}_1 + \epsilon_1] \\
&= U\mathbb{E}[P^{t_1}\mathbf{s}_1] \\
&= UT\mathbb{E}[\boldsymbol{\pi}_0] \\
&= U\boldsymbol{\pi}. \\
V_2 &:= \mathbb{E}[\mathbf{x}_1\mathbf{x}_1^\top] \\
&= \mathbb{E}[(U\mathbf{h}_1 + \epsilon_1)(U\mathbf{h}_1 + \epsilon_1)^\top] \\
&= \mathbb{E}[U\mathbf{h}_1\mathbf{h}_1^\top U^\top] + \sigma^2 I \\
&= U\mathbb{E}[\mathsf{diag}(\mathbf{h}_1)]U^\top + \sigma^2 I \\
&= U\mathbb{E}[\mathsf{diag}(P^{t_1}\mathbf{s}_1)]U^\top + \sigma^2 I \\
&= U\mathbb{E}[\mathsf{diag}(T\boldsymbol{\pi}_0)]U^\top + \sigma^2 I \\
&= U\mathsf{diag}(\boldsymbol{\pi})U^\top + \sigma^2 I. \\
V_3 &:= \mathbb{E}[\mathbf{x}_1 \otimes \mathbf{x}_1 \otimes \mathbf{x}_1] \\
&= \mathbb{E}[(U\mathbf{h}_1 + \epsilon_1) \otimes (U\mathbf{h}_1 + \epsilon_1) \otimes (U\mathbf{h}_1 + \epsilon_1)] \\
&= \mathbb{E}[(U\mathbf{h}_1)^{\otimes 3}] + \mathbb{E}[(U\mathbf{h}_1) \otimes \epsilon_1 \otimes \epsilon_1] + \mathbb{E}[\epsilon_1 \otimes (U\mathbf{h}_1) \otimes \epsilon_1] + \mathbb{E}[\epsilon_1 \otimes \epsilon_1 \otimes (U\mathbf{h}_1)] \\
&= \sum_i \pi_i U_i \otimes U_i \otimes U_i + V_1 \otimes_1 (\sigma^2 I) + V_1 \otimes_2 (\sigma^2 I) + V_1 \otimes_3 (\sigma^2 I),
\end{aligned}
$$

which relies on the assumption of zero skewness $\mathbb{E}[(\boldsymbol{\epsilon}_1)_d^3] = 0, 1 \leq d \leq m$.

$$
\begin{aligned}
C_2 &:= \mathbb{E}[\mathbf{x}_1\mathbf{x}_2^\top] \\
&= \mathbb{E}[(U\mathbf{h}_1 + \epsilon_1)(U\mathbf{h}_2 + \epsilon_2)^\top] \\
&= \mathbb{E}[U\mathbf{h}_1\mathbf{h}_2^\top U^\top] \qquad \text{(B.5)} \\
&= U\mathbb{E}[P^{t_1}\mathbf{s}_1\mathbf{s}_2^\top (P^{t_2})^T]U^\top \\
&= UT\mathbb{E}[\boldsymbol{\pi}_0\boldsymbol{\pi}_0^\top]T^\top U^\top \\
&= \frac{UT\mathsf{diag}(\boldsymbol{\pi})(UT)^\top}{\alpha_0 + 1} + \frac{\alpha_0 V_1 V_1^\top}{\alpha_0 + 1}. \qquad \text{(B.6)} \\
C_3 &:= \mathbb{E}[\mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \mathbf{x}_3] \\
&= \mathbb{E}[(U\mathbf{h}_1 + \epsilon_1) \otimes (U\mathbf{h}_2 + \epsilon_2) \otimes (U\mathbf{h}_3 + \epsilon_3)] \\
&= \mathbb{E}[(U\mathbf{h}_1) \otimes (U\mathbf{h}_2) \otimes (U\mathbf{h}_3)] \qquad \text{(B.7)} \\
&= \mathbb{E}[(UP^{t_1}\mathbf{s}_1) \otimes (UP^{t_2}\mathbf{s}_2) \otimes (UP^{t_3}\mathbf{s}_3)] \\
&= \mathbb{E}[(UT\boldsymbol{\pi}_0) \otimes (UT\boldsymbol{\pi}_0) \otimes (UT\boldsymbol{\pi}_0)]
\end{aligned}
$$

$$= \frac{\sum_i 2\pi_i (UT)_i^{\otimes 3} - 2\alpha_0^2 V_1^{\otimes 3}}{(\alpha_0+2)(\alpha_0+1)} + \frac{\alpha_0(V_1 \otimes_3 C_2 + V_1 \otimes_2 C_2 + V_1 \otimes_1 C_2)}{\alpha_0+2}. \quad \text{(B.8)}$$

Note that due to the independence assumption, there are no noise-related terms in (B.5) and (B.7), indicating that C_2 and C_3 are unaffected by the noise distribution. And again, (B.6) and (B.8) are established with the expressions of Dirichlet moments in Appendix B.1 of Anandkumar et al. [2013]. As in Appendix B.1.1, M_2, M_3, M_2' and M_3' result from adjusting the raw moments.

B.2 Proof of Theorem 3

We first prove the following lemma:

Lemma 1. *If $P(r) := (rI + (1-r)T^*)^{-1}T^*$ exists and is a stochastic matrix for some $r \in (0,1]$, then $P(r')$ exists and is a stochastic matrix for all $r' \in [r,1]$.*

Proof. Since $P(r)$ exists we can write $T^* = rP(r)(I-(1-r)P(r))^{-1}$. By assumption P^* is invertible, so T^* is invertible. We then have

$$r'(T^*)^{-1} + (1-r')I = \frac{r'}{r}(P(r)^{-1} - (1-r)I) + (1-r')I = \frac{r'}{r}P(r)^{-1}(I-(1-r/r')P(r)),$$

which is invertible for all $r' \in [r,1]$. Therefore, we can write

$$P(r') = (r'(T^*)^{-1} + (1-r')I)^{-1} = \frac{r}{r'}P(r)(I-(1-r/r')P(r))^{-1} = \mathbb{E}_t[P(r)],$$

where $t \sim \mathsf{Geometric}(r/r')$, showing that $P(r')$ is a stochastic matrix. □

To prove Theorem 3 we begin by noting that $\mathcal{S}$ contains all values of r for which $rI+(1-r)T^*$ is singular. Therefore, $P(r)$ is well-defined and invertible for $r \in (0,1] \setminus \mathcal{S}$. From the identity $T^*\boldsymbol{\pi}^* = \boldsymbol{\pi}^* = (rI+(1-r)T^*)\boldsymbol{\pi}^*$ we have $P(r)\boldsymbol{\pi}^* = \boldsymbol{\pi}^*,\ r \notin \mathcal{S}$. Similarly, the identity $\mathbf{1}^\top T^* = \mathbf{1}^\top = \mathbf{1}^\top(rI+(1-r)T^*)$ and the fact that $(rI+(1-r)T^*)^{-1}T^* = T^*(rI+(1-r)T^*)^{-1}$ imply that $\mathbf{1}^\top P(r) = \mathbf{1}^\top,\ r \notin \mathcal{S}$. It is easy to verify $P(r^*) = P^*$ by plugging in the definition of T^*. Lemma 1 then implies that $\max(\mathcal{S}) < r^*$ and that $P(r')$ is a stochastic matrix for $r' \geq r^*$. To prove the last statement of the theorem we rewrite $P(r)$ by plugging in the definition of T^*:

$$P(r) \;=\; \frac{r^*}{r}\left(I-(1-r^*/r)P^*\right)^{-1}P^*$$

and consider its first-order derivative w.r.t. r:

$$\frac{\partial P(r)}{\partial r} \;=\; -\Big(\frac{r}{r^*}I + \Big(1-\frac{r}{r^*}\Big)P^*\Big)^{-2}\frac{(I-P^*)P^*}{r^*},$$

which exists for $r \in (0,1] \setminus \mathcal{S}$. By assumption we have $P^*_{ij} = 0$, and by ergodicity of P^* we can assume $(P^*)^2_{ij} > 0$ (otherwise there exists $k \neq j$ such that $P^*_{ik} = 0$ and $(P^*)^2_{ik} > 0$). Then we have

$$\frac{\partial P(r)_{ij}}{\partial r}\Big|_{r=r^*} = \frac{(P^*)^2_{ij}}{r^*} > 0,$$

implying that there exists $c > 0$ such that for $r \in [r^*-c, r^*)$, $P(r)_{ij} < P^*_{ij} = 0$. This and Lemma 1 then imply the last statement of the theorem.

B.3 Sample Complexity Analysis

The analyses here mostly follow those in Anandkumar et al. [2013]. Let O denote the observation matrix, which can be the T matrix in First-order Markov models, the U matrix or the product UT in Hidden Markov Models. Define

$$\widetilde{O} \;:=\; O\mathsf{diag}([\sqrt{\pi_1}\ \sqrt{\pi_2}\ \cdots\ \sqrt{\pi_k}]),$$

$$M_2 \;:=\; O\mathsf{diag}(\boldsymbol{\pi})O^\top \;=\; \widetilde{O}\widetilde{O}^\top \quad \text{and} \quad M_3 := \sum_{i=1}^{k} \pi_i O_i \otimes O_i \otimes O_i.$$

Let $\pi_{\min} := \min_i \pi_i$. We have

$$\sigma_k(O)\sqrt{\pi_{\min}} \le \sigma_k(\widetilde{O}),$$
$$\sigma_1(\widetilde{O}) \le \sigma_1(O),$$

where $\sigma_j(\cdot)$ denotes the jth largest singular value.

Denote by $\|\cdot\|$ the spectral norm of a matrix or the operator norm of a symmetric third-order tensor induced by the vector 2-norm:

$$\|M\| \;:=\; \sup_{\|\boldsymbol{\theta}\|_2=1} |M(\boldsymbol{\theta},\boldsymbol{\theta},\boldsymbol{\theta})|.$$

Suppose

$$\begin{aligned} \|\widehat{M}_2 - M_2\| &= E_2, \\ \|\widehat{M}_3 - M_3\| &\le E_3, \end{aligned}$$

for some E_2 and E_3 to be determined.

B.3.1 Perturbation Lemmas

Let $\widehat{M}_{2,k}$ be the best rank k approximation to $\widehat{M}_2$ in terms of the matrix 2-norm. According to Algorithm 5.1, we have

$$\widehat{W}^\top \widehat{M}_{2,k} \widehat{W} = I.$$

Let

$$\widehat{W}^\top M_2 \widehat{W} = ADA^\top$$

be an SVD of $\widehat{W}^\top M_2 \widehat{W}$, where $A \in \mathcal{R}^{k\times k}$. Define

$$W \;:=\; \widehat{W} A D^{-1/2} A^\top$$

and notice that

$$W^\top M_2 W = AD^{-1/2}A^\top \widehat{W}^\top M_2 \widehat{W} A D^{-1/2} A^\top = I.$$

Let $Q := W^\top \widetilde{O}$ and $\widehat{Q} := \widehat{W}^\top \widetilde{O}$.

Lemma 2. *(Lemma C.1 of Anandkumar et al. [2013]) Let Π_W be the orthogonal projection onto the range of W and Π be the orthogonal projection onto the range of O. Suppose $E_2 \leq \sigma_k(M_2)/2$. We have the following:*

$$\begin{aligned}
\|Q\| &= 1, \\
\|\widehat{Q}\| &\leq 2, \\
\|\widehat{W}\| &\leq \frac{2}{\sigma_k(\widetilde{O})}, \\
\|\widehat{W}^\dagger\| &\leq 2\sigma_1(\widetilde{O}), \\
\|W^\dagger\| &\leq 3\sigma_1(\widetilde{O}), \\
\|Q - \widehat{Q}\| &\leq \frac{4E_2}{\sigma_k(\widetilde{O})^2}, \\
\|\widehat{W}^\dagger - W^\dagger\| &\leq \frac{6\sigma_1(\widetilde{O})E_2}{\sigma_k(\widetilde{O})^2}, \\
\|\Pi - \Pi_W\| &\leq \frac{4E_2}{\sigma_k(\widetilde{O})^2}.
\end{aligned}$$

Lemma 3. *Weyl's Theorem. (Theorem 4.11, p.204 in Stewart and Sun [1990]). Let $A, E \in \mathcal{R}^{m\times n}$ with $m \geq n$ be given. Then*

$$\max_{1\leq i\leq n} |\sigma_i(A+E) - \sigma_i(A)| \leq \|E\|.$$

B.3.2 Reconstruction Accuracy

Throughout this section we assume that the number of iterations N and L for Algorithm 5.2 satisfy the conditions in Theorem 1.

Lemma 4. *Suppose $\max(E_2, E_3) \leq \sigma_k(M_2)/2$. For any $\eta \in (0,1)$, with probability at least $1-\eta$ the following holds:*

$$\|O - (\widehat{W}^\top)^\dagger \widehat{V}\widehat{\Lambda}\| \leq c\frac{\max(\sigma_1(O), 1)}{\pi_{\min}^{3/2}\min(\sigma_k(O)^2, 1)}\max(E_2, E_3)$$

for some constant $c > 0$.

Proof. By Theorem 1, the following hold with probability at least $1-\eta$:

$$\begin{aligned}
&\|V - \widehat{V}\|_F = \sqrt{\sum_i \|V_i - \widehat{V}_i\|^2} \leq \sqrt{\sum_i (64E_3^2)/(1/\sqrt{\pi_{\min}})^2} = 8E_3, \\
&\|\widehat{\Lambda}\| = \max_i \widehat{1/\sqrt{\pi_i}} \leq \max_i (1/\sqrt{\pi_i} + 5E_3) \leq \pi_{\min}^{-1/2} + 5E_3.
\end{aligned}$$

With the above two bounds and Lemma 2 we have

$$
\begin{aligned}
&\|O - (\widehat{W}^\top)^\dagger \widehat{V}\widehat{\Lambda}\| \le \|O - \Pi_W O\| + \|\Pi_W O - (\widehat{W}^\top)^\dagger \widehat{V}\widehat{\Lambda}\| \\
=&\|\Pi O - \Pi_W O\| + \|(W^\dagger)^\top V\Lambda - (\widehat{W}^\dagger)^\top \widehat{V}\widehat{\Lambda}\| \\
\le&\|\Pi - \Pi_W\|\|O\| + \|(W^\dagger)^\top V\Lambda - (W^\dagger)^\top V\widehat{\Lambda}\| + \|(W^\dagger)^\top V\widehat{\Lambda} - (\widehat{W}^\dagger)^\top \widehat{V}\widehat{\Lambda}\| \\
\le&\|\Pi - \Pi_W\| + \|W^\dagger\|\|V\|\|\Lambda - \widehat{\Lambda}\| + \|(W^\dagger)^\top V\widehat{\Lambda} - (W^\dagger)^\top \widehat{V}\widehat{\Lambda}\| + \|(W^\dagger)^\top \widehat{V}\widehat{\Lambda} - (\widehat{W}^\dagger)^\top \widehat{V}\widehat{\Lambda}\| \\
\le&\|\Pi - \Pi_W\| + \|W^\dagger\|E_3 + \|W^\dagger\|\|V - \widehat{V}\|\|\widehat{\Lambda}\| + \|W^\dagger - \widehat{W}^\dagger\|\|\widehat{V}\|\|\widehat{\Lambda}\| \\
\le&\frac{4E_2}{\sigma_k(\widetilde{O})^2} + 3\sigma_1(\widetilde{O})E_3 + 3\sigma_1(\widetilde{O})\|V - \widehat{V}\|_F\|\widehat{\Lambda}\| + \frac{6\sigma_1(\widetilde{O})E_2}{\sigma_k(\widetilde{O})^2}(\|\widehat{V} - V\|_F + 1)\|\widehat{\Lambda}\| \\
\le&c\left(\Big(\frac{24}{\sqrt{\pi_{\min}}} + 3\Big)\sigma_1(O)E_3 + \Big(4 + \frac{6\sigma_1(O)}{\sqrt{\pi_{\min}}}\Big)\frac{E_2}{\sigma_k(O)^2\pi_{\min}}\right) \\
\le&c\left(\frac{27\sigma_1(O)}{\sqrt{\pi_{\min}}} + \frac{10\max(\sigma_1(O), 1)}{\pi_{\min}^{3/2}\sigma_k(O)^2}\right)\max(E_2, E_3) \\
\le&c\frac{37\max(\sigma_1(O), 1)}{\pi_{\min}^{3/2}\min(\sigma_k(O)^2, 1)}\max(E_2, E_3)
\end{aligned}
$$

where $c > 0$ is a constant large enough to dominate low-order terms like E_2E_3. □

Lemma 5. *With a slight abuse of notation, let U denote a column permutation of the true U, UT denote a column permutation of the true UT, and P denote a column-and-row permutation of the true P, where the permutations involved are the same. Suppose*

$$\max(\|U - \widehat{U}\|, \|\widehat{UT} - UT\|) \le \sigma_k(rU + (1-r)UT)/2.$$

We then have

$$\|P - (r\widehat{U} + (1-r)\widehat{UT})^\dagger\widehat{UT}\| \le \frac{6\sigma_1(UT)}{\sigma_k(rU + (1-r)UT)^2}\max(\|U - \widehat{U}\|, \|UT - \widehat{UT}\|).$$

Proof. First notice that

$$
\begin{aligned}
&(rU + (1-r)UT)^\dagger(UT) \\
=&\big((rI + (1-r)T)^\top U^\top U(rI + (1-r)T)\big)^{-1}(rI + (1-r)T)^\top U^\top UT \\
=&(rI + (1-r)T)^{-1}T \;=\; P.
\end{aligned}
$$

Then we have

$$
\begin{aligned}
&\|P - (r\widehat{U} + (1-r)\widehat{UT})^\dagger\widehat{UT}\| \;=\; \|(rU + (1-r)(UT))^\dagger UT - (r\widehat{U} + (1-r)\widehat{UT})^\dagger\widehat{UT}\| \\
\le&\|(rU + (1-r)UT)^\dagger(UT) - (r\widehat{U} + (1-r)\widehat{UT})^\dagger(UT)\| + \\
&\|(r\widehat{U} + (1-r)\widehat{UT})^\dagger(UT) - (r\widehat{U} + (1-r)\widehat{UT})^\dagger\widehat{UT}\| \\
\le&\|(rU + (1-r)UT)^\dagger - (r\widehat{U} + (1-r)\widehat{UT})^\dagger\|\|UT\| + \|(r\widehat{U} + (1-r)\widehat{UT})^\dagger\|\|UT - \widehat{UT}\|.
\end{aligned}
\tag{B.9}
$$

By Lemma 3 and the assumption of the lemma, we have

$$\sigma_k(rU + (1-r)UT)/2 \ \leq \sigma_k(r\widehat{U} + (1-r)\widehat{UT}) \ \leq \ 3\sigma_k(rU + (1-r)UT)/2,$$

showing that rank$(r\widehat{U} + (1-r)\widehat{UT}) = k$ and

$$\|(r\widehat{U} + (1-r)\widehat{UT})^\dagger\| = 1/\sigma_k(r\widehat{U} + (1-r)\widehat{UT}) \leq 2/\sigma_k(rU + (1-r)UT).$$

Because rank$(r\widehat{U} + (1-r)\widehat{UT}) =$ rank$(rU + (1-r)UT) = k$, Theorem 3.4 in Stewart [1977] indicates that

$$\begin{aligned}
&\|(rU + (1-r)UT)^\dagger - (r\widehat{U} + (1-r)\widehat{UT})^\dagger\| \\
\leq&\sqrt{2}\|(rU + (1-r)UT)^\dagger\|\|(r\widehat{U} + (1-r)\widehat{UT})^\dagger\|\|r(U-\widehat{U}) + (1-r)(UT - \widehat{UT})\| \\
\leq&\frac{\sqrt{2}(r\|U-\widehat{U}\| + (1-r)\|\widehat{UT} - UT\|)}{\sigma_k(rU + (1-r)UT)\sigma_k(r\widehat{U} + (1-r)\widehat{UT})} \leq \frac{2\sqrt{2}(r\|U-\widehat{U}\| + (1-r)\|UT - \widehat{UT}\|)}{\sigma_k(rU + (1-r)UT)^2}.
\end{aligned}$$

Applying these bounds to (B.9) then leads to

$$\begin{aligned}
&\|P - (r\widehat{U} + (1-r)\widehat{UT})^\dagger\widehat{UT}\| \\
\leq&\frac{2\sqrt{2}\sigma_1(UT)\big(r\|U-\widehat{U}\| + (1-r)\|UT - \widehat{UT}\|\big)}{\sigma_k(rU + (1-r)UT)^2} + \frac{2\|UT - \widehat{UT}\|}{\sigma_k(rU + (1-r)UT)} \\
=&\frac{r2\sqrt{2}\sigma_1(UT)\|U-\widehat{U}\|}{\sigma_k(rU + (1-r)UT)^2} + \frac{\big((1-r)2\sqrt{2}\sigma_1(UT) + 2\sigma_k(rU + (1-r)UT)\big)\|UT - \widehat{UT}\|}{\sigma_k(rU + (1-r)UT)^2} \\
\leq&\frac{\max(r2\sqrt{2}, (1-r)2\sqrt{2} + 2)\sigma_1(UT)}{\sigma_k(rU + (1-r)UT)^2}\max(\|U-\widehat{U}\|, \|UT - \widehat{UT}\|) \\
\leq&\frac{6\sigma_1(UT)}{\sigma_k(rU + (1-r)UT)^2}\max(\|U-\widehat{U}\|, \|UT - \widehat{UT}\|),
\end{aligned}$$

in which we use the fact $\sigma_1(UT) \geq \sigma_1(rU + (1-r)UT) \geq \sigma_k(rU + (1-r)UT)$. □

B.3.3 Concentration of empirical averages

Lemma 6. *Let $\{\mathbf{y}_i\}_{i=1}^N$ be N i.i.d. random vectors in $\mathcal{R}^m$. Let $\boldsymbol{\mu} := \mathbb{E}[\mathbf{y}_i], \Sigma :=$ Var$(\mathbf{y}_i)$ and $\sigma^2_{\max} := \max_d \Sigma_{dd}$. Let $\bar{\boldsymbol{\mu}} := (\sum_i \mathbf{y}_i)/N$. Then*

$$\textit{Prob}(\|\bar{\boldsymbol{\mu}} - \boldsymbol{\mu}\|_2 \geq \epsilon) \leq \frac{m\sigma^2_{\max}}{N\epsilon^2}.$$

Proof. This lemma is a straightforward consequence of the Markov inequality:

$$\begin{aligned}
\text{Prob}(\|\bar{\boldsymbol{\mu}} - \boldsymbol{\mu}\|_2 \geq \epsilon) \ &= \ \text{Prob}(\|\bar{\boldsymbol{\mu}} - \boldsymbol{\mu}\|_2^2 \geq \epsilon^2) \\
&\leq \ \frac{\mathbb{E}[\|\bar{\boldsymbol{\mu}} - \boldsymbol{\mu}\|_2^2]}{\epsilon^2} \\
&= \ \frac{\sum_d \mathbb{E}(\bar{\boldsymbol{\mu}}_d - \boldsymbol{\mu}_d)^2}{\epsilon^2} \ = \ \frac{\text{Tr}(\Sigma)}{N\epsilon^2} \ \leq \ \frac{m\sigma^2_{\max}}{N\epsilon^2}.
\end{aligned}$$

□

Lemma 7. *Let $\widehat{V}_1, \widehat{V}_2, \widehat{V}_3, \widehat{C}_2, \widehat{C}_3$ denote averages of N independent draws of $\mathbf{x}_1, \mathbf{x}_1 \otimes \mathbf{x}_1, \mathbf{x}_1 \otimes \mathbf{x}_1 \otimes \mathbf{x}_1, \mathbf{x}_1 \otimes \mathbf{x}_2, \mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \mathbf{x}_3$ from the generative process in Section 5.2.2. Let $u_{\max} := \max_{i,j} |U_{ij}|$. Then*

$$\begin{aligned}
\textit{Prob}(\|\widehat{V}_1 - V_1\|_2 \geq \epsilon) &\leq \frac{m(u_{\max}^2 + \sigma^2)}{N\epsilon^2}, \\
\textit{Prob}(\|\widehat{V}_2 - V_2\|_F \geq \epsilon) &\leq \frac{m^2(u_{\max}^2 + \sigma^2)^2}{N\epsilon^2}, \\
\textit{Prob}(\|\widehat{V}_3 - V_3\|_F \geq \epsilon) &\leq \frac{m^3(u_{\max}^2 + \sigma^2)^3}{N\epsilon^2}, \\
\textit{Prob}(\|\widehat{C}_2 - C_2\|_F \geq \epsilon) &\leq \frac{m^2(u_{\max}^2 + \sigma^2)^2}{N\epsilon^2}, \\
\textit{Prob}(\|\widehat{C}_3 - C_3\|_F \geq \epsilon) &\leq \frac{m^3(u_{\max}^2 + \sigma^2)^3}{N\epsilon^2}.
\end{aligned}$$

Proof. Based on Lemma 6, it suffices to bound $\sigma_{\max}^2$ in these five cases:

$$\begin{aligned}
\max_i \textsf{Var}((\mathbf{x}_1)_i) &\leq \max_i \mathbb{E}[(\mathbf{x}_1)_i^2] = \max_i \mathbb{E}_{\mathbf{h}_1}[\sigma^2 + (U\mathbf{h}_1)_i^2] \leq \sigma^2 + \max_{i,k} U_{ik}^2, \\
\max_{i,j} \textsf{Var}((\mathbf{x}_1)_i(\mathbf{x}_1)_j) &\leq \max_{i,j} \mathbb{E}[(\mathbf{x}_1)_i^2(\mathbf{x}_1)_j^2] = \max_{i,j} \mathbb{E}_{\mathbf{h}_1}[(\sigma^2 + (U\mathbf{h}_1)_i^2)(\sigma^2 + (U\mathbf{h}_1)_j^2)] \\
&\leq \max_{i,j,l} (\sigma^2 + U_{il}^2)(\sigma^2 + U_{jl}^2) \leq (\sigma^2 + \max_{i,j} U_{ij}^2)^2, \\
\max_{i,j} \textsf{Var}((\mathbf{x}_1)_i(\mathbf{x}_2)_j) &\leq \max_{i,j} \mathbb{E}[(\mathbf{x}_1)_i^2(\mathbf{x}_2)_j^2] = \max_{i,j} \mathbb{E}_{\boldsymbol{\pi}_0}\big[\mathbb{E}[(\mathbf{x}_1)_i^2|\boldsymbol{\pi}_0]\mathbb{E}[(\mathbf{x}_2)_i^2|\boldsymbol{\pi}_0]\big] \\
&\leq \max_{i,j} \sup_{\boldsymbol{\pi}_0} \mathbb{E}[(\mathbf{x}_1)_i^2|\boldsymbol{\pi}_0]\mathbb{E}[(\mathbf{x}_2)_i^2|\boldsymbol{\pi}_0] \leq \big(\max_i \sup_{\boldsymbol{\pi}_0} \mathbb{E}[(\mathbf{x}_1)_i^2|\boldsymbol{\pi}_0]\big)^2 \\
&= \big(\max_i \sup_{\boldsymbol{\pi}_0} \sum_k U_{ij}^2 (T\boldsymbol{\pi}_0)_k + \sigma^2\big)^2 \\
&= \big(\max_i \max_{j'} \sum_k U_{ij}^2 T_{jj'} + \sigma^2\big)^2 \leq (\max_{i,j} U_{ij}^2 + \sigma^2)^2.
\end{aligned}$$

With similar arguments, we have that

$$\begin{aligned}
\max_{i,j,l} \textsf{Var}((\mathbf{x}_1)_i(\mathbf{x}_1)_j(\mathbf{x}_1)_l) &\leq (\max_{i,j} U_{ij}^2 + \sigma^2)^3, \\
\max_{i,j,l} \textsf{Var}((\mathbf{x}_1)_i(\mathbf{x}_2)_j(\mathbf{x}_3)_l) &\leq (\max_{i,j} U_{ij}^2 + \sigma^2)^3.
\end{aligned}$$

□

Lemma 8. *Let $\widehat{M}_2, \widehat{M}_3, \widehat{M}_2', \widehat{M}_3'$ denote estimates of the population quantities defined in Theorem 4 obtained by plugging in empirical averages of independent samples as in Lemma 7, and $\widehat{\sigma_2} := \lambda_{\min}(\widehat{V}_2 - \widehat{V}_1\widehat{V}_1^\top)$, where $\lambda_{\min}(\cdot)$ denotes the smallest eigenvalue in modulus. Define*

$\nu := \max(\sigma^2 + u_{\max}^2, 1)$. *We then have the following:*

$$\begin{aligned}
\textit{Prob}(\|M_2' - \widehat{M_2'}\| \geq \epsilon) &\leq \frac{75m^2\nu^2}{N\epsilon^2}, \\
\textit{Prob}(\|M_3' - \widehat{M_3'}\| \geq \epsilon) &\leq \frac{1000m^4\nu^3}{N\epsilon^2}, \\
\textit{Prob}(\|M_2 - \widehat{M_2}\| \geq \epsilon) &\leq \frac{50(\alpha_0+1)^2m^2\nu^2}{N\epsilon^2}, \\
\textit{Prob}(\|M_3 - \widehat{M_3}\| \geq \epsilon) &\leq \frac{1100k^2m^3(\alpha_0+2)^2(\alpha_0+1)^2\nu^3}{N\epsilon^2}.
\end{aligned}$$

Proof. We first note that it is easy verify $\mathbf{z}^\top(\widehat{V_2} - \widehat{V_1}\widehat{V_1}^\top)\mathbf{z} \geq 0$ for any real vector $\mathbf{z}$, so $\widehat{\sigma^2}$ is always non-negative. By Lemma 3, we have

$$\begin{aligned}
|\sigma^2 - \widehat{\sigma^2}| &\leq \|V_2 - V_1V_1^\top - (\widehat{V_2} - \widehat{V_1}\widehat{V_1}^\top)\| \leq \|V_2 - \widehat{V_2}\| + \|V_1V_1^\top - \widehat{V_1}\widehat{V_1}^\top\| \\
&\leq \|V_2 - \widehat{V_2}\| + \|\widehat{V_1} - V_1\|(\|\widehat{V_1}\| + \|V_1\|) \\
&\leq \|V_2 - \widehat{V_2}\| + 2\|V_1\|\|\widehat{V_1} - V_1\| + \|V_1 - \widehat{V_1}\|^2.
\end{aligned}$$

We also need the following

$$\|V_1\|^2 = \|U\boldsymbol{\pi}\|^2 = \sum_i \left(\sum_j U_{ij}\pi_j\right)^2 \leq \sum_{i,j}\pi_j U_{ij}^2 \leq \sum_i \max_j U_{ij}^2 \leq mu_{\max}^2.$$

Then we have

$$\begin{aligned}
\|\widehat{M_2'} - M_2'\| &\leq \|\widehat{V_2} - V_2\| + |\widehat{\sigma^2} - \sigma^2| \\
&\leq 2\|\widehat{V_2} - V_2\| + 2\|V_1\|\|\widehat{V_1} - V_1\| + \|\widehat{V_1} - V_1\|^2 \\
&\leq 2\|\widehat{V_2} - V_2\|_F + 2\|V_1\|\|\widehat{V_1} - V_1\| + \|\widehat{V_1} - V_1\|^2,
\end{aligned}$$

which implies

$$\begin{aligned}
& \text{Prob}(\|\widehat{M_2'} - M_2'\| \geq \epsilon) \\
\leq\ & \text{Prob}(2\|\widehat{V_2} - V_2\|_F + 2\|V_1\|\|\widehat{V_1} - V_1\| + \|\widehat{V_1} - V_1\|^2 \geq \epsilon) \\
\leq\ & \text{Prob}(2\|\widehat{V_2} - V_2\|_F \geq \epsilon/3) + \text{Prob}(2\|V_1\|\|\widehat{V_1} - V_1\| \geq \epsilon/3) + \text{Prob}(\|\widehat{V_1} - V_1\|^2 \geq \epsilon/3) \\
\leq\ & \frac{36m^2(u_{\max}^2+\sigma^2)^2}{N\epsilon^2} + \frac{36\|V_1\|^2m(u_{\max}^2+\sigma^2)}{N\epsilon^2} + \frac{3m(u_{\max}^2+\sigma^2)}{N\epsilon} \\
\leq\ & \frac{36m^2(u_{\max}^2+\sigma^2)^2}{N\epsilon^2} + \frac{36m^2u_{\max}^2(u_{\max}^2+\sigma^2)}{N\epsilon^2} + \frac{3m(u_{\max}^2+\sigma^2)}{N\epsilon} \\
\leq\ & \frac{75m^2(u_{\max}^2+\sigma^2)^2}{N\epsilon^2}.
\end{aligned}$$

Similarly, we have

$$\begin{aligned}
\|M_3' - \widehat{M_3'}\| &\leq \|V_3 - \widehat{V_3}\|_F + 3\|V_1 \otimes_1 (\sigma^2 I) - \widehat{V_1} \otimes_1 (\widehat{\sigma^2} I)\|_F \\
&= \|V_3 - \widehat{V_3}\|_F + 3\sqrt{m}\|\sigma^2 V_1 - \widehat{\sigma^2}\widehat{V_1}\| \\
&\leq \|V_3 - \widehat{V_3}\|_F + 3\sqrt{m}(\sigma^2\|V_1 - \widehat{V_1}\| + |\sigma^2 - \widehat{\sigma^2}|(\|V_1\| + \|\widehat{V_1} - V_1\|)) \\
&\leq \|V_3 - \widehat{V_3}\|_F + \|V_1 - \widehat{V_1}\|3\sqrt{m}(\sigma^2 + 2mu_{\max}^2) + \|V_2 - \widehat{V_2}\|3u_{\max}m \\
&\quad + \|V_1 - \widehat{V_1}\|^2 9u_{\max}m + 3\sqrt{m}(\|V_1 - \widehat{V_1}\|\|\widehat{V_2} - V_2\| + \|V_1 - \widehat{V_1}\|^3),
\end{aligned}$$

implying

$$\begin{aligned}
& \text{Prob}(\|M_3' - \widehat{M_3'}\| \geq \epsilon) \\
\leq\ & \text{Prob}(\|V_3 - \widehat{V_3}\|_F \geq \epsilon/6) + \text{Prob}(\|V_1 - \widehat{V_1}\| \geq \epsilon/(18\sqrt{m}(\sigma^2 + 2mu_{\max}^2))) \\
& + \text{Prob}(\|V_2 - \widehat{V_2}\| \geq \epsilon/(18u_{\max}m)) + \text{Prob}(\|V_1 - \widehat{V_1}\|^2 \geq \epsilon/(54u_{\max}m)) \\
& + \text{Prob}\left(\|V_1 - \widehat{V_1}\| \geq \sqrt{\epsilon/(18\sqrt{m})}\right) + \text{Prob}\left(\|\|V_2 - \widehat{V_2}\| \geq \sqrt{\epsilon/(18\sqrt{m})}\right) \\
& + \text{Prob}(\|V_1 - \widehat{V_1}\|^3 \geq \epsilon/(18\sqrt{m})) \\
\leq\ & \frac{36m^3(u_{\max}^2 + \sigma^2)^3}{N\epsilon^2} + \frac{324m^2(\sigma^2 + 2mu_{\max}^2)^2(\sigma^2 + u_{\max}^2)}{N\epsilon^2} + \frac{324u_{\max}^2 m^4(\sigma^2 + u_{\max}^2)^2}{N\epsilon^2} \\
& + \frac{54u_{\max}m^2(\sigma^2 + u_{\max}^2)}{N\epsilon} + \frac{18m^{3/2}(\sigma^2 + u_{\max}^2)}{N\epsilon} + \frac{18m^{5/2}(\sigma^2 + u_{\max}^2)^2}{N\epsilon} \\
& + \frac{36^{1/3}m^{4/3}(\sigma^2 + u_{\max}^2)}{N\epsilon^{2/3}} \\
\leq\ & \frac{1000m^4(\max(\sigma^2 + u_{\max}^2, 1))^3}{N\epsilon^2}.
\end{aligned}$$

Using similar arguments, we have

$$\begin{aligned}
\|M_2 - \widehat{M_2}\| &\leq (\alpha_0 + 1)\|C_2 - \widehat{C_2}\|_F + \alpha_0\|V_1 V_1^\top - \widehat{V_1}\widehat{V_1}^\top\|_F \\
&\leq (\alpha_0 + 1)\|C_2 - \widehat{C_2}\|_F + 2\alpha_0\|V_1\|\|\widehat{V_1} - V_1\| + \alpha_0\|\widehat{V_1} - V_1\|^2,
\end{aligned}$$

and therefore

$$\begin{aligned}
& \text{Prob}(\|M_2 - \widehat{M_2}\| \geq \epsilon) \\
\leq\ & \text{Prob}(\|C_2 - \widehat{C_2}\|_F \geq \frac{\epsilon}{3(\alpha_0 + 1)}) + \text{Prob}(\|\widehat{V_1} - V_1\| \geq \frac{\epsilon}{6\alpha_0\|V_1\|}) \\
& + \text{Prob}(\|\widehat{V_1} - V_1\|^2 \geq \frac{\epsilon}{3\alpha_0}) \\
\leq\ & \frac{9(\alpha_0 + 1)^2 m^2(\sigma^2 + u_{\max}^2)^2}{N\epsilon^2} + \frac{36\alpha_0^2 m^2 u_{\max}^2(\sigma^2 + u_{\max}^2)}{N\epsilon^2} + \frac{3\alpha_0 m(\sigma^2 + u_{\max}^2)}{N\epsilon} \\
\leq\ & \frac{50(\alpha_0 + 1)^2 m^2(\sigma^2 + u_{\max}^2)^2}{N\epsilon^2}.
\end{aligned}$$

Finally, we have

$$
\begin{aligned}
& \|M_3 - \widehat{M_3}\| \\
\leq\ & \frac{(\alpha_0+2)(\alpha_0+1)}{2}\|C_3 - \widehat{C_3}\|_F + \frac{3(\alpha_0+1)\alpha_0}{2}\|V_1 \otimes_1 C_2 - \widehat{V_1} \otimes \widehat{C_2}\|_F \\
& + \alpha_0^2\|V_1 \otimes V_1 \otimes V_1 - \widehat{V_1} \otimes \widehat{V_1} \otimes \widehat{V_1}\|_F \\
\leq\ & \frac{(\alpha_0+2)(\alpha_0+1)}{2}\|C_3 - \widehat{C_3}\|_F + \frac{3(\alpha_0+1)\alpha_0}{2}\|V_1 - \widehat{V_1}\|\|C_2\|_F + \frac{3(\alpha_0+1)\alpha_0}{2}\|\widehat{V_1}\|\|C_2 - \widehat{C_2}\|_F \\
& + 3\alpha_0^2\|V_1\|^2\|V_1 - \widehat{V_1}\| + 3\alpha_0^2\|V_1\|\|V_1 - \widehat{V_1}\|^2 + \alpha_0^2\|V_1 - \widehat{V_1}\|^3 \\
\leq\ & \frac{(\alpha_0+2)(\alpha_0+1)}{2}\|C_3 - \widehat{C_3}\|_F + \frac{3(\alpha_0+1)\alpha_0}{2}\|V_1 - \widehat{V_1}\|\|C_2\|_F + \frac{3(\alpha_0+1)\alpha_0}{2}\|V_1\|\|C_2 - \widehat{C_2}\|_F \\
& + \frac{3(\alpha_0+1)\alpha_0}{2}\|V_1 - \widehat{V_1}\|\|C_2 - \widehat{C_2}\|_F + 3\alpha_0^2\|V_1\|^2\|V_1 - \widehat{V_1}\| + 3\alpha_0^2\|V_1\|\|V_1 - \widehat{V_1}\|^2 + \alpha_0^2\|V_1 - \widehat{V_1}\|^3 \\
\leq\ & \frac{(\alpha_0+2)(\alpha_0+1)}{2}\|C_3 - \widehat{C_3}\|_F + 5(\alpha_0+1)\alpha_0 k m u_{\max}^2\|V_1 - \widehat{V_1}\| + \frac{3(\alpha_0+1)\alpha_0}{2}\|V_1\|\|C_2 - \widehat{C_2}\|_F \\
& + \frac{3(\alpha_0+1)\alpha_0}{2}\|V_1 - \widehat{V_1}\|\|C_2 - \widehat{C_2}\|_F + 3\alpha_0^2\|V_1\|\|V_1 - \widehat{V_1}\|^2 + \alpha_0^2\|V_1 - \widehat{V_1}\|^3
\end{aligned}
$$

using the fact that

$$
\|C_2\|_F \ = \ \left\| UT\left(\frac{\mathsf{diag}\boldsymbol{\pi} + \alpha_0\boldsymbol{\pi}\boldsymbol{\pi}^\top}{\alpha_0+1}\right)T^\top U^\top \right\|_F \ \leq \ \|UT\|_F^2 \ \leq \ k m u_{\max}^2,
$$

and thus

$$
\begin{aligned}
& \text{Prob}(\|M_3 - \widehat{M_3}\| \geq \epsilon) \ \leq \ \text{Prob}\left(\|C_3 - \widehat{C_3}\|_F \geq \frac{\epsilon}{3(\alpha_0+2)(\alpha_0+1)}\right) \\
& + \text{Prob}\left(\|V_1 - \widehat{V_1}\|_F \geq \frac{\epsilon}{30(\alpha_0+1)\alpha_0 k m u_{\max}^2}\right) + \text{Prob}\left(\|C_2 - \widehat{C_2}\|_F \geq \frac{\epsilon}{9(\alpha_0+1)\alpha_0}\right) \\
& + \text{Prob}\left(\|V_1 - \widehat{V_1}\|^2 \geq \frac{\epsilon}{18\alpha_0^2\|V_1\|}\right) + \text{Prob}\left(\|V_1 - \widehat{V_1}\|^3 \geq \frac{\epsilon}{6\alpha_0^2}\right) \\
\leq\ & \frac{9m^2(\alpha_0+2)^2(\alpha_0+1)^2(\sigma^2+u_{\max}^2)^3}{N\epsilon^2} + \frac{900k^2m^3(\alpha_0+1)^2\alpha_0^2u_{\max}^4(\sigma^2+u_{\max}^2)}{N\epsilon^2} \\
& + \frac{81(\alpha_0+1)^2\alpha_0^2m^2(\sigma^2+u_{\max}^2)^2}{N\epsilon^2} + \frac{18\alpha_0^2m^{3/2}u_{\max}(\sigma^2+u_{\max}^2)}{N\epsilon} + \frac{6m\alpha_0^{4/3}(\sigma^2+u_{\max}^2)}{N\epsilon^{2/3}} \\
\leq\ & \frac{1100k^2m^3(\alpha_0+2)^2(\alpha_0+1)^2(\sigma^2+u_{\max}^2)^3}{N\epsilon^2}.
\end{aligned}
$$

□

B.4 Proof of Theorem 5

Let $\widehat{U}$ and $\widehat{UT}$ be column-permuted as described in Algorithm 5.4. Let

$$
\delta_{\min} := \min_{i,j} |1/\sqrt{\pi_i} - 1/\sqrt{\pi_j}|.
$$

If $\max(E_3, E_3') \leq \delta_{\min}/15$, Theorem 5.1 of Anandkumar et al. [2012a] implies that for any $\eta \in (0,1)$, with probability at least $1-\eta$, the columns of $\widehat{U}$ and $\widehat{UT}$ are matched to the same permutation of the columns of the true U and UT, respectively. As in Lemma 5, let U, UT, and P denote proper permutations of the true matrices. We then have

$$\begin{aligned}
&\text{Prob}\left(\max(\|U-\widehat{U}\|, \|UT-\widehat{UT}\|) \geq \frac{\epsilon\sigma_k(rU+(1-r)UT)^2}{6\sigma_1(UT)}\right) \\
\leq &\text{Prob}\left(\|U-\widehat{U}\| \geq \frac{\epsilon\sigma_k(rU+(1-r)UT)^2}{6\sigma_1(UT)}\right) + \text{Prob}\left(\|UT-\widehat{UT}\| \geq \frac{\epsilon\sigma_k(rU+(1-r)UT)^2}{6\sigma_1(UT)}\right).
\end{aligned}$$

Let the failure probability for the tensor decomposition method be set to $\frac{\eta}{4}$. Then by Lemma 4 we can bound the first term as follows:

$$\begin{aligned}
&\text{Prob}\left(\|U-\widehat{U}\| \geq \frac{\epsilon\sigma_k(rU+(1-r)UT)^2}{6\sigma_1(UT)}\right) \\
\leq &\text{Prob}\left(\max(E_2', E_3') \geq \frac{\epsilon\sigma_k(rU+(1-r)UT)^2\pi_{\min}^{3/2}\min(\sigma_k(U)^2,1)}{6\sigma_1(UT)c\max(\sigma_1(U),1)}\right) \\
&+ \text{Prob}(\max(E_2', E_3') \geq \sigma_k(M_2')/2) + \frac{\eta}{4} + \text{Prob}(E_3' \geq \delta_{\min}/15),
\end{aligned}$$

where the first term in the r.h.s is based on Lemma 4 conditioned on the event that $\max(E_2', E_3') \geq \sigma_k(M_2')/2$ and the tensor decomposition method succeeds, the second and the third terms bound the probability that the event does not occur, and the last term bounds the probability of incorrectly matching the columns of $\widehat{U}$ and U. To continue bounding these terms we use Lemma 8 to have

$$\begin{aligned}
&\text{Prob}\left(\max(E_2', E_3') \geq \frac{\epsilon\sigma_k(rU+(1-r)UT)^2\pi_{\min}^{3/2}\min(\sigma_k(U)^2,1)}{6\sigma_1(UT)c\max(\sigma_1(U),1)}\right) \\
\leq &\frac{(2700m^2\nu^2+36000m^4\nu^3)\sigma_1(UT)^2c^2\max(\sigma_1(U)^2,1)}{N\epsilon^2\sigma_k(rU+(1-r)UT)^4\pi_{\min}^3\min(\sigma_k(U)^4,1)} \\
\leq &\frac{39000m^4\nu^3\sigma_1(UT)^2c^2\max(\sigma_1(U)^2,1)}{N\epsilon^2\sigma_k(rU+(1-r)UT)^4\pi_{\min}^3\min(\sigma_k(U)^4,1)}, \\
&\text{Prob}(\max(E_2', E_3') \geq \sigma_k(M_2')/2) \leq \frac{300m^2\nu^2+4000m^4\nu^3}{N\sigma_k(M_2)^2} \leq \frac{4300m^4\nu^3}{N\sigma_k(M_2)^2}, \\
&\text{Prob}(E_3' \geq \delta_{\min}/15) \leq \frac{225000m^4\nu^3}{N\delta_{\min}^2}.
\end{aligned}$$

Thus, by setting the sample size N so that

$$N \geq \frac{12m^4\nu^3}{\eta}\max\left(\frac{225000}{\delta_{\min}^2}, \frac{4300}{\sigma_k(M_2)^2}, \frac{39000\sigma_1(UT)^2c^2\max(\sigma_1(U)^2,1)}{\epsilon^2\sigma_k(rU+(1-r)UT)^4\pi_{\min}^3\min(\sigma_k(U)^4,1)}\right),$$

we have

$$\text{Prob}\left(\|U-\widehat{U}\| \geq \frac{\epsilon\sigma_k(rU+(1-r)UT)^2}{6\sigma_1(UT)}\right) \leq \frac{\eta}{2}, \tag{B.10}$$

where the randomness is from both the data and the algorithm. Using similar arguments, we have that for sample size N such that

$$N \geq \frac{12k^2m^3(\alpha_0+2)^2(\alpha_0+1)^2\nu^3}{\eta} \cdot \max\left(\frac{225000}{\delta_{\min}^2}, \frac{4600}{\sigma_k(M'_2)^2}, \frac{42000\sigma_1(UT)^2(c')^2\max(\sigma_1(UT)^2,1)}{\epsilon^2\sigma_k(rU+(1-r)UT)^4\pi_{\min}^3\min(\sigma_k(UT)^4,1)}\right),$$

the following holds:

$$\text{Prob}\left(\|UT-\widehat{UT}\| \geq \frac{\epsilon\sigma_k(rU+(1-r)UT)^2}{6\sigma_1(UT)}\right) \leq \frac{\eta}{2}. \tag{B.11}$$

Combining the two bounds (B.10) and (B.11), we have for

$$N \geq \frac{12\max(k^2,m)m^3\nu^3(\alpha_0+2)^2(\alpha_0+1)^2}{\eta} \cdot \max\left(\frac{225000}{\delta_{\min}^2}, \frac{4600}{\min(\sigma_k(M'_2),\sigma_k(M_2))^2}, \frac{42000c^2\sigma_1(UT)^2\max(\sigma_1(UT),\sigma_1(U),1)^2}{\epsilon^2\sigma_k(rU+(1-r)UT)^4\min(\sigma_k(UT),\sigma_k(U),1)^4}\right),$$

the following bound holds for any $\epsilon > 0$ and $\eta \in (0,1)$:

$$\text{Prob}\left(\max(\|U-\widehat{U}\|, \|UT-\widehat{UT}\|) \leq \frac{\epsilon\sigma_k(rU+(1-r)UT)^2}{6\sigma_1(UT)}\right) \geq 1-\eta,$$

which by Lemma 5 implies that

$$\text{Prob}(\|P-(r\widehat{U}+(1-r)\widehat{UT})^\dagger\widehat{UT}\| \leq \epsilon) \geq 1-\eta.$$

Appendix C

Derivations in Chapter 6

C.1 Derivation of (6.20)

Using properties of the matrix trace and the kernel trick, we immediately have

$$\begin{aligned}
&\frac{1}{2}\|\mathcal{Z}_2 P \mathcal{Z}_1^\top - \widehat{\mathcal{C}}_{2,1}\|^2_{\mathcal{G}\otimes\mathcal{G}} \propto \frac{1}{2}\text{Tr}(P^\top M_2 P M_1) - \text{Tr}(P^\top F),\\
&\frac{u}{2}\Big(\|\mathcal{Z}_2 P\mathbf{1} - \frac{\mathcal{S}\mathbf{1}}{m_S}\|^2_{\mathcal{G}} + \|\mathcal{Z}_1 P^\top \mathbf{1} - \frac{\mathcal{S}\mathbf{1}}{m_S}\|^2_{\mathcal{G}}\Big)\\
&\propto \frac{u}{2}\mathbf{1}^\top(P^\top M_2 P + P M_1 P^\top)\mathbf{1} - u\mathbf{1}^\top(P^\top \boldsymbol{\mu}_2 + P\boldsymbol{\mu}_1).
\end{aligned}$$

Let $\lambda_i(\cdot)$ denotes the i-th Eigenvalue of a matrix. We then rewrite the nuclear norm term:

$$\begin{aligned}
\tau\|\mathcal{Z}_2 P \mathcal{Z}_1^\top\|_* \;&=\; \tau\sum_i \sqrt{\lambda_i\big(\mathcal{Z}_2 P L_1^\top L_1 P^\top \mathcal{Z}_2^\top\big)}\\
&= \tau\sum_i \sqrt{\lambda_i\big(L_1^\top P^\top L_2 L_2^\top P L_1\big)} = \tau\|L_2^\top P L_1\|_*,
\end{aligned}$$

C.2 Derivation of (6.34)

We begin by defining some notations:

$$\begin{aligned}
&H := \widetilde{U}^\top M_3 \widetilde{U},\; R := \widetilde{V}^\top M_1 \widetilde{V},\; \mathbf{u} := \widetilde{U}^\top \mathbf{1},\; \mathbf{v} := \widetilde{V}^\top \mathbf{1},\\
&F_1 := \Phi_1^\top \mathcal{Z}_1 \widetilde{V},\; F_2 := \frac{\Phi_2^\top \mathcal{Z}_2}{n},\; F_3 := \Phi_3^\top \mathcal{Z}_3 \widetilde{U}.
\end{aligned}$$

Let $\mathsf{vec}(X)$ be the vector resulting from column concatenation of a matrix X, $\mathsf{diag}(\mathbf{x})$ be the diagonal matrix with the vector $\mathbf{x}$ being its main diagonal. Superscripts denote column indices. Using properties of the matrix trace and the kernel trick, we re-write the three terms in (6.34) as follows. For the first term we have

$$\begin{aligned}
&\|\widetilde{\mathcal{C}}_{3,1,2}(\{B_l\}) - \widehat{\mathcal{C}}_{3,1,2}\|^2_{\mathcal{G}\otimes\mathcal{G}\otimes\mathcal{G}}\\
\propto&\sum_d \text{Tr}\Big(\sum_{l,l'}(\mathcal{Z}_2^l)_d(\mathcal{Z}_2^{l'})_d \widetilde{V} B_l^\top \widetilde{U}^\top M_3 \widetilde{U} B_{l'} \widetilde{V}^\top M_1\Big) -\\
&2\sum_d \text{Tr}\Big(\sum_l \widetilde{V} B_l^\top \widetilde{U}^\top (\mathcal{Z}_2^l)_d \mathcal{Z}_3^\top \Phi_3 \frac{\mathsf{diag}((\Phi_2)_{d\cdot})}{n} \Phi_1^\top \mathcal{Z}_1\Big)\\
=&\text{Tr}\Big(\sum_{ll'}(M_2)_{ll'} B_l^\top H B_{l'} R - 2\sum_l B_l^\top F_3^\top \mathsf{diag}(F_2^l) F_1\Big),
\end{aligned}$$

and then for the second term

$$\begin{aligned}
&\|\widetilde{\mathcal{C}}_{3,\cdot,2}(\{B_l\}) - \widetilde{\mathcal{C}}_{2,1}\|^2_{\mathcal{G}\otimes\mathcal{G}} \propto\\
&\text{Tr}\big(\big[B_1\mathbf{v}\;\cdots\;B_m\mathbf{v}\big]^\top H\big[B_1\mathbf{v}\;\cdots\;B_m\mathbf{v}\big]M_2\big) -\\
&2\text{Tr}\big(\big[B_1\mathbf{v}\;\cdots\;B_m\mathbf{v}\big]^\top \widetilde{U}^\top M_{32}\widetilde{P}M_{12}\big) =\\
&\text{Tr}\big(\sum_{il}(M_2)_{il} B_i^\top H B_l \mathbf{v}\mathbf{v}^\top - 2\sum_i B_i^\top \widetilde{U}^\top M_{32}\widetilde{P} M_{12}^i \mathbf{v}^\top\big),
\end{aligned}$$

and finally for the third term

$$
\begin{aligned}
&\|\widetilde{\mathcal{C}}_{\cdot,1,2}(\{B_l\})^\top - \widetilde{\mathcal{C}}_{2,1}\|^2_{\mathcal{G}\otimes\mathcal{G}} \propto \\
&\mathrm{Tr}\big(\big[B_1^\top \mathbf{u} \ \cdots \ B_m^\top \mathbf{u}\big] M_2 \big[B_1^\top \mathbf{u} \ \cdots \ B_m^\top \mathbf{u}\big]^\top R\big) - \\
&2\mathrm{Tr}\big(\big[B_1^\top \mathbf{u} \ \cdots \ B_m^\top \mathbf{u}\big] M_2 \widetilde{P} M_1 \widetilde{V}\big) = \\
&\mathrm{Tr}\big(\sum_{ij} (M_2)_{ij} B_i^\top \mathbf{u}\mathbf{u}^\top B_j R - 2\sum_i B_i^\top \mathbf{u} (M_2^i)^\top \widetilde{P} M_1 \widetilde{V}\big).
\end{aligned}
$$

To further simplify these expressions, we re-define the notation B to be a k^2-by-m matrix whose l-th column B^l denotes column concatenation of the k-by-k matrix B_l in the above expressions. With the new notation and the identity:

$$
\mathsf{vec}(XYZ) = (Z^\top \circ X)\mathsf{vec}(Y) \tag{C.1}
$$

where $\circ$ denotes the Kronecker product, we obtain the succinct form (6.34) in which

$$
\begin{aligned}
C &:= R \circ H + u((\mathbf{v}\mathbf{v}^\top) \circ H + R \circ (\mathbf{u}\mathbf{u}^\top)), \\
J &:= (F_1 \circ F_3)^\top \big[\mathsf{vec}(\mathsf{diag}(F_2^1)) \ \cdots \ \mathsf{vec}(\mathsf{diag}(F_2^m))\big] \\
&\quad + u\Big(\big(\mathbf{v} \circ (\widetilde{U}^\top M_{32} \widetilde{P})\big) M_{12} + \big((\widetilde{V}^\top M_1 \widetilde{P}^\top) \circ \mathbf{u}\big) M_2\Big).
\end{aligned}
$$

Bibliography

J. Aach and G. M. Church. Aligning gene expression time series with time warping algorithms. *Bioinformatics*, 17(6):495–508, 2001. 2

P. Abbeel and A. Y. Ng. Learning first order Markov models for control. In *Advances in Neural Information Processing Systems 17*, 2005. 1

D. Achlioptas and F. McSherry. On spectral learning of mixtures of distributions. In *Learning Theory*, pages 458–469. Springer, 2005. 5.4

A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky. Tensor decompositions for learning latent variable models. *arXiv preprint arXiv:1210.7559v2*, 2012a. 1.1, 5, 5.1, 5.1, 1, 5.1, 5.1, 1, 5.2.1, 5.2.2, 5.2.2, 5.2.2, B.4

A. Anandkumar, D. P. Foster, D. Hsu, S. M. Kakade, and Y.-K. Liu. A spectral algorithm for latent dirichlet allocation. *arXiv preprint arXiv:1204.6703v4*, 2013. 1.2, 5, 5.1, 5.2.1, B.1, B.1.1, B.1.2, B.3, 2

A. Anandkumar, D. Hsu, and S. M. Kakade. A method of moments for mixture models and hidden Markov models. *arXiv preprint arXiv:1203.0683*, 2012b. 1.2, 5, 5.2.2, 5.4

P. Antsaklis and A. Michel. *Linear systems*. Birkhauser, 2005. 4, 1

D. Applegate, R. Bixby, V. Chvátal, and W. Cook. Concorde TSP solver. URL `http://www.tsp.gatech.edu/concorde/index.html`. 3.3, 3.5

S. Arora, R. Ge, Y. Halpern, D. Mimno, A. Moitra, D. Sontag, Y. Wu, and M. Zhu. A practical algorithm for topic modeling with provable guarantees. *arXiv preprint arXiv:1212.4777*, 2012. 5, 5.4

B. Balle and M. Mohri. Spectral learning of general weighted automata via constrained matrix completion. In *Advances in Neural Information Processing Systems 25*, pages 2168–2176, 2012. 1

B. Balle, A. Quattoni, and X. Carreras. Local loss optimization in operator models: A new insight into spectral learning. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1879–1886, 2012. 1

R. Barzilay and N. Elhadad. Inferring strategies for sentence ordering in multidocument news summarization. *Journal of Artificial Intelligence Research*, 17:35–55, 2002. 2

M. J. Beal, Z. Ghahramani, and C. E. Rasmussen. The infinite hidden Markov model. In *Advances in Neural Information Processing Systems 14*, pages 577–584, 2002. 1

G. Bennett. Probability inequalities for the sum of independent random variables. *Journal of the*

American Statistical Association, 57(297):33–45, 1962. 3.5

D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994. 2

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA 02178-9998, second edition, 1999. 4.2, 6.2.1

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006. 3.2.1

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003. 5.2.1, A

F. Bock. An algorithm to construct a minimum directed spanning tree in a directed network. In *Developments in Operations Research*, pages 29–44. 1971. 3.2.2

B. Boot and G. J. Gordon. Two-manifold problems. *http://arxiv.org/abs/1112.6399*, 2011. 1

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011. 6.2.1

J. P. Boyle and R. L. Dykstra. A method for finding projections onto the intersection of convex sets in Hilbert spaces. *Lecture Notes in Statistics*, 37:28–47, 1986. 3.3

G. Brock, V. Pihur, S. Datta, and S. Datta. clvalid: An R package for cluster validation. *Journal of Statistical Software*, 25(4):1–22, 2008. 5

T. Buck, A. Rao, L. Coelho, M. Fuhrman, J. Jarvik, P. Berget, and R. Murphy. Cell cycle dependence of protein subcellular location inferred from static, asynchronous images. In *Proceedings of the 31st Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 1016–1019. IEEE, 2009. 1, 2, 3.5.3

S. Busygin, O. Prokopyev, and P. Pardalos. Biclustering in data mining. *Computers & Operations Research*, 35(9):2964–2987, 2008. 7.1

J. Cai, E. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010. 6.2.1

P. M. Camerini, L. Fratta, and F. Maffioli. A note on finding optimum branchings. *Networks*, 9: 309–312, 1979. 3.2.2

J. Chen, W. Chu, J. Kuo, C. Weng, and J. Wu. Tiling slideshow. In *Proceedings of the 14th annual ACM international conference on Multimedia*, pages 25–34. ACM, 2006. 2

X. Chen, Q. Lin, S. Kim, J. Carbonell, and E. Xing. Smoothing proximal gradient method for general structured sparse regression. *The Annals of Applied Statistics*, 6(2):719–752, 2012. 6.2.1, 6.2.2

Y. Chen, A. Wiesel, Y. C. Eldar, and A. O. Hero. Shrinkage algorithms for mmse covariance estimation. *IEEE Transactions on Signal Processing*, 58:5016–5029, 2010a. 4.3

Y. Chen, A. Wiesel, and A. O. Hero. Robust shrinkage estimation of high-dimensional covariance matrices. Technical report, arXiv:1009.5331v1 [stat.ME], September 2010b. 4.3

Y. J. Chu and T. H. Liu. On the shortest arborescence of a directed graph. *Science Sinica*, 14: 1396–1400, 1965. 3.2.2

E. Cooke, R. Savage, P. Kirk, R. Darkins, and D. Wild. Bayesian hierarchical clustering for microarray time series data with replicates and outlier measurements. *BMC bioinformatics*, 12(1):399, 2011. 7.1, 7.4, 7.4.1, 7.4.2, 7.4.2, 5

S. Datta and S. Datta. Methods for evaluating clustering algorithms for gene expression data using a reference set of functional classes. *BMC bioinformatics*, 7(1):397, 2006. 7.4.2

P. Deshpande, R. Barzilay, and D. Karger. Randomized decoding for selection-and-ordering problems. In *Proceedings of NAACL HLT*, pages 444–451, 2007. 2

J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the ℓ_1-ball for learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning*, pages 272–279, 2008. 3.6, 3.3, 6.2.2

J. Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240, 1967. 3.2.2

A. Fujita, J. Sato, H. Garay-Malpartida, R. Yamaguchi, S. Miyano, M. Sogayar, and C. Ferreira. Modeling gene expression regulatory networks with the sparse vector autoregressive model. *BMC Systems Biology*, 1(1):39, 2007. 7

Z. Ghahramani. Learning dynamic bayesian networks. *Lecture Notes in Computer Science*, 1387:168–197, 1998a. 1

Z. Ghahramani. Variational learning for switching state-space models. *Neural Computation*, 12: 963–996, 1998b. 1

J. Giesen. Curve reconstruction in arbitrary dimension and the traveling salesman problem. *Proceedings of the 8th International Conference on Discrete Geometry for Computational Imagery, Lecture Notes in Computer Science*, 1568:164–176, 1999. 2

A. Girard and G. J. Pappas. Approximate bisimulations for nonlinear dynamical systems. In *44th IEEE Conference on Decision and Control and European Control Conference*, pages 684–689, 2005. 3.4.2

M. Girvan and M. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821, 2002. 7

A. Globerson, T. Koo, X. Carreras, and M. Collins. Structured prediction models via the matrix-tree theorem. In *Poceedings of the Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 141–150, 2007. 3.2.3, 3.2.3

C. Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, pages 424–438, 1969. 7

G. Grimmett and D. Stirzaker. *Probability and random processes*. Oxford university press, 2001. 5.2.1

V. Gripon and M. Rabbat. Reconstructing a graph from path traces. In *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, 2013. 2

A. Gupta and Z. Bar-Joseph. Extracting dynamics from static cancer expression data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 5:172–182, 2008. 2

J. Hamilton. *Time series analysis*. Princeton Univ Pr, 1994. 3

T. Hastie and W. Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84: 502–516, 1989. 2

K. Heller and Z. Ghahramani. Bayesian hierarchical clustering. In *The 22nd international conference on Machine learning*, pages 297–304. ACM, 2005. 7.4

I. Herman, G. Melançon, and M. Marshall. Graph visualization and navigation in information visualization: A survey. *Visualization and Computer Graphics, IEEE Transactions on*, 6(1): 24–43, 2000. 7

R. Hodrick and E. C. Prescott. Postwar U.S. business cycles: An empircal investigation. *Journal of Money, Credit, and Banking*, 29:1–16, 1997. 3.3

D. Hsu and S. M. Kakade. Learning mixtures of spherical gaussians: moment methods and spectral decompositions. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 11–20. ACM, 2013. 5, 5.2.2

D. Hsu, S. M. Kakade, and T. Zhang. A spectral algorithm for learning hidden Markov models. In *Proceedings of the Twenty-Second Annual Conference on Learning Theory*, 2009. 1, 1.2, 6

X. Hua, L. Lu, and H. Zhang. Automatically converting photographic series into video. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 708–715. ACM, 2004. 2

T.-K. Huang and J. Schneider. Learning linear dynamical systems without sequence information. In *Proceedings of the 26th International Conference on Machine Learning*, pages 425–432, 2009. 1.2

T.-K. Huang and J. Schneider. Learning auto-regressive models from sequence and non-sequence data. In *Advances in Neural Information Processing Systems 24*, pages 1548–1556. 2011. 1.2

T.-K. Huang and J. Schneider. Learning bi-clustered vector autoregressive models. In *Machine Learning and Knowledge Discovery in Databases*, pages 741–756. Springer, 2012. 1.2

T.-K. Huang and J. Schneider. Spectral learning of hidden Markov models from dynamic and static data. In *Proceedings of the 30th International Conference on Machine Learning*, 2013a. 1.2

T.-K. Huang and J. Schneider. Learning hidden Markov models from non-sequence data via tensor decomposition. In *Advances in Neural Information Processing Systems 26*. 2013b. To appear. 1.2

T.-K. Huang, L. Song, and J. Schneider. Learning nonlinear dynamic models from non-sequenced data. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 2010. 1.2

L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985. 7.4.1

I. N. Junejo, E. Dexter, I. Laptev, and P. Pérez. View-independent action recognition from temporal self-similarities. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33 (1):172–185, 2011. 2

R. Kannan, H. Salmasian, and S. Vempala. The spectral method for general mixture models. In *Learning Theory*, pages 444–457. Springer, 2005. 5.4

A. Katok and B. Hasselblatt. *Introduction to the modern theory of dynamical systems*, volume 54. Cambridge Univ Pr, 1996. 2

E. Keogh and C. A. Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and information systems*, 7(3):358–386, 2005. 2

B. J. L. Rabiner. *Fundamentals of Speech Recognition*. Prentice Hall, 1993. 2

O. Ledoit and M. Wolf. Improved estimation of the covariance matrix of stock returns with an application to portfolio selection. *Journal of Empirical Finance*, 10:603–621, 2003. 4.3

O. Ledoit and M. Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, 88:365–411, 2004. 4.3

C. E. V. Lesser. A simple method of trend construction. *Journal of the Royal Statistical Society, Series B*, 23:91–107, 1961. 3.3

S. Li. Concise formulas for the area and volume of a hyperspherical cap. *Asian Journal of Mathematics and Statistics*, 4(1):66–70, 2011. 3.5

L. Ljung. *System Identification: Theory for the User*. Prentice Hall, New Jersey, second edition, 1999. 1

E. N. Lorenz. Deterministic nonperiodic flow. *Journal of the Atmospheric Sciences*, 20(2): 130–141, 1963. 3.4.2

A. Lozano, N. Abe, Y. Liu, and S. Rosset. Grouped graphical granger modeling for gene expression regulatory networks discovery. *Bioinformatics*, 25(12):i110, 2009. 7, 7.1

P. M. Magwene, P. Lizardi, and J. Kim. Reconstructing the temporal ordering of biological samples using microarray data. *Bioinformatics*, 19:842–850, 2003. 2

B. M. Marlin, M. Schmidt, and K. P. Murphy. Group sparse priors for covariance estimation. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI 2009)*, Montreal, Canada, 2009. 7.2

E. Meeds and S. Roweis. Nonparametric Bayesian biclustering. Technical report, Department of Computer Science, University of Toronto, 2007. URL `http://www.cs.toronto.edu/~ewm/index_files/biclustering.pdf`. 7.1, 7.3.2

A. Moitra and G. Valiant. Settling the polynomial learnability of mixtures of Gaussians. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 93–102. IEEE, 2010. 5.4

M. Müller. *Information retrieval for music and motion*. Springer, 2007. 2

K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California at Berkeley, 2002. 1

Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103 (1):127–152, 2005. 6.2.2

A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In

Advances in Neural Information Processing Systems, 2001. 2, 7.4.1

V. A. Nicholson. Matrices with permanent equal to one. *Linear Algebra and its Applications*, 12 (2):185–188, 1975. 3.2.2

D. Noll, O. Prot, and A. Rondepierre. A proximity control algorithm to minimize nonsmooth and nonconvex functions. *Pacific Journal of Optimization*, 4(3):569–602, 2008. 4.2

J. Peters, D. Janzing, A. Gretton, and B. Schölkopf. Detecting the direction of causal time series. In *Proceedings of the 26th International Conference on Machine Learning*, pages 801–808, 2009. 3.1

I. Porteous, E. Bart, and M. Welling. Multi-hdp: A non-parametric bayesian model for tensor factorization. In *Proc. of the 23rd National Conf. on Artificial Intelligence*, pages 1487–1490, 2008. 7.1

M. G. Rabbat, M. A. Figueiredo, and R. D. Nowak. Network inference from co-occurrences. *Information Theory, IEEE Transactions on*, 54(9):4053–4068, 2008. 2

L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989. 1

M. Ramoni, P. Sebastiani, and I. Kohane. Cluster analysis of gene expression dynamics. *Proceedings of the National Academy of Sciences*, 99(14):9121, 2002. 7.1

J. Ramsay and X. Li. Curve registration. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(2):351–363, 1998. 2

C. Rangel, J. Angus, Z. Ghahramani, M. Lioumi, E. Sotheran, A. Gaiba, D. Wild, and F. Falciani. Modeling T-cell activation using gene expression profiling and state-space models. *Bioinformatics*, 20(9):1361–1372, 2004. 1.2, 7, 7.4.2

J. Reimand, T. Arak, and J. Vilo. g: Profiler – a web server for functional interpretation of gene lists (2011 update). *Nucleic acids research*, 39(suppl 2):W307–W315, 2011. 7.4.2

A. Reiss and D. Stricker. Introducing a new benchmarked dataset for activity monitoring. In *Wearable Computers (ISWC), 2012 16th International Symposium on*, pages 108–109. IEEE, 2012. 6.3.2

S. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007. 7

J. Schäfer and K. Strimmer. A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. *Statistical Applications in Genetics and Molecular Biology*, 4, 2005. 4.3, 4.3, 4.3

B. Schölkopf, A. Smola, and K. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998. 6.1, 6.2.2

J. Sethuraman. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4:639–650, 1994. 7.2

D. Shahaf and C. Guestrin. Connecting the dots between news articles. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 623–632. ACM, 2010. 2

D. Sheldon, M. S. Elmohamed, and D. Kozen. Collective inference on Markov models for

modeling bird migration. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1321–1328. MIT Press, Cambridge, MA, 2008. 2

A. Shojaie, S. Basu, and G. Michailidis. Adaptive thresholding for reconstructing regulatory networks from time-course gene expression data. *Statistics in Biosciences*, pages 1–18, 2011. 7, 7.1

S. M. Siddiqi, B. Boots, and G. J. Gordon. Reduced-rank hidden Markov models. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 2010. 1.2, 3.5.1, 6, 6.1, 6.1, 6.3.1

B. W. Silverman. Incorporating parametric effects into functional principal components analysis. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 673–689, 1995. 2

D. A. Smith and N. A. Smith. Probabilistic models of nonprojective dependency trees. In *Poceedings of the Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 132–140, 2007. 3.2.3

A. J. Smola, S. Mika, B. Schölkopf, and R. C. Williamson. Regularized principal manifolds. *Journal of Machine Learning Research*, 1:179–209, 2001. 2

L. Song, J. Huang, A. Smola, and K. Fukumizu. Hilbert space embeddings of conditional distributions with applications to dynamical systems. In *Proceedings of the 26th International Conference on Machine Learning*, 2009. 6.1, 6.2.2

L. Song, B. Boots, S. Siddiqi, G. Gordon, and A. Smola. Hilbert space embeddings of hidden Markov models. In *Proceedings of the 27th International Conference on Machine Learning*, 2010. 1, 1.2, 6, 6.1, 6.1, 6.2.2, 6.2.2

C. Stein. Estimation of a covariance matrix. In *Rietz Lecture, 39th Annual Meeting, Atlanta, GA*, 1975. 4.3

G. W. Stewart and J.-g. Sun. Matrix perturbation theory. 1990. 3

G. Stewart. On the perturbation of pseudo-inverses, projections and linear least squares problems. *SIAM review*, 19(4):634–662, 1977. 5.2.2, B.3.2

R. E. Tarjan. Finding optimum branchings. *Networks*, 7:25–35, 1977. 3.2.2

J. Tenenbaum, V. Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319, 2000. 3.5.3

B. P. Tu, A. Kudlicki, M. Rowicka, and S. L. McKnight. Logic of the yeast metabolic cycle: Temporal compartmentalization of cellular processes. *Science*, 310(5751):1152–1158, 2005. 1, 3.5.2, 3.5.2, 3.5.2

W. T. Tutte. *Graph Theory*. Addison-Wesley, 1984. 3.2.3

L. G. Valiant. The complexity of computing the permanet. *Theoretical Computer Science*, 8(2): 189–201, 1979. 3.2.2

T. Vintsyuk. Speech discrimination by dynamic programming. *Cybernetics and Systems Analysis*, 4(1):52–57, 1968. 2

K. Q. Weinberger, F. Sha, and L. K. Saul. Learning a kernel matrix for nonlinear dimensionality

reduction. In *Proceedings of the 21st International Conference on Machine Learning*, pages 839–846, 2004. 3.4.1, 3.4.2

R. Yang and J. O. Berger. Estimation of a covariance matrix using the reference prior. *Annals of Statistics*, 22:1195–1211, 1994. 4.3

X. Zhou, F. Li, J. Yan, and S. Wong. A novel cell segmentation method and cell phase identification using Markov model. *Information Technology in Biomedicine, IEEE Transactions on*, 13(2):152–157, 2009. 3.5.3

X. Zhu, A. B. Goldberg, M. Rabbat, and R. Nowak. Learning bigrams from unigrams. In *the Proceedings of 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technology, Columbus, OH*, 2008. 2

H. Zou. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101(476):1418–1429, 2006. 7, 7.4

MACHINE LEARNING
DEPARTMENT

Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA15213

Carnegie Mellon